SIEMENS

Create MyHMI /Automotive PLC blocks

Function Manual

Introduction 1

Security compliance 2

Configuring Automotive PLC program plus blocks 3

Business logic blocks 4

Abbreviations 5

Appendix A A

V20 SP1

10/2025
A5E53211851B AE

Legal information

Warning notice system

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

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

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

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

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

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

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified persons are those who, because of their training and experience, are familiar with the installation, assembly, commissioning, operation, decommissioning and disassembly of the product and can recognize risks and avoid possible hazards.

Proper use of Siemens products

Note the following:

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

Trademarks

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

Disclaimer of Liability

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

Siemens Aktiengesellschaft
Digital Industries
Postfach 48 48
90026 NÜRNBERG
GERMANY

A5E53211851B AE
Ⓟ 11/2025 Subject to change

Copyright © Siemens 2025.
All rights reserved

Table of contents

1 Introduction 9

1.1 About SINUMERIK 9

1.2 About this documentation 9

1.3 Documentation on the internet 10
1.3.1 Documentation overview SINUMERIK ONE 10
1.3.2 Documentation overview SINUMERIK 840D sl 11
1.3.3 Documentation overview SINUMERIK 828D 12
1.3.4 Documentation overview SINUMERIK MC 13
1.3.5 Documentation overview SINUMERIK 808D 14
1.3.6 Documentation overview SINUMERIK operator components 14

1.4 Feedback on the technical documentation 15

1.5 mySupport documentation 15

1.6 Service and Support 15

1.7 OpenSSL 17

1.8 Compliance with the General Data Protection Regulation 18

2 Security compliance 19

2.1 Sourcing software updates 19

2.2 Secure product disposal 20

2.3 Ports and protocols 20

2.4 Cybersecurity 20

3 Configuring Automotive PLC program plus blocks 21

3.1 Overview 21

3.2 Programming PLC Instructions 23

3.3 Upgrading V19 to V20 TIA portal project 24

3.4 LAPP_Init 32
3.4.1 Block interface - LAPP_Init 32
3.4.2 Status codes 34

3.5 Configuring station data 35
3.5.1 Configuring header 35
3.5.2 Data type 39

3.6 Screen switching 40
3.6.1 Overview 40
3.6.2 Block interface - LAPP_ScreenSwitch 41
3.6.3 Status codes 42
3.6.4 Example for manual operation screens 43

3.7 Shift planning 43

3.7.1 Overview ... 43
3.7.2 Block interface - LAPP_ShiftPlanning ... 45
3.7.3 Additional information ... 46
3.7.4 Status codes ... 47

3.8 Workpiece manager ... 47
3.8.1 Workpiece manager for PLC ... 47
3.8.1.1 Overview ... 47
3.8.1.2 Block interface - LAPP_WorkpieceManager_PLC ... 49
3.8.1.3 Status codes ... 51
3.8.2 Workpiece manager for CNC ... 51
3.8.2.1 Overview ... 51
3.8.2.2 Block interface - LAPP_WorkpieceManager_CNC ... 53
3.8.2.3 Configuring chanData ... 55
3.8.2.4 Status codes ... 56
3.8.3 Additional widget for workpiece data ... 56

3.9 Manual operation ... 60
3.9.1 Manual Operation as a screen ... 60
3.9.1.1 Overview ... 60
3.9.1.2 Manual operation as a screen ... 63
3.9.1.3 Block interface - LAPP_FuncGrpConfig ... 66
3.9.1.4 Block interface - LAPP_ManualOperation ... 67
3.9.1.5 Data type ... 73
3.9.1.6 Configuring addressing modes ... 74
3.9.1.7 Examples ... 75
3.9.1.8 Status codes ... 80
3.9.2 Manual Operation as faceplate ... 82
3.9.2.1 Manual Operation as faceplate ... 82
3.9.2.2 Block interface - LAPP_ManualOperationFaceplate ... 84
3.9.2.3 Status codes ... 86

3.10 Tool life monitor ... 87
3.10.1 Overview ... 87
3.10.2 Configuring number of tools ... 89
3.10.3 Block interface - LAPP_ToolLifeMain ... 89
3.10.4 Block interface - LAPP_ToolLifeMonitor ... 91
3.10.5 Data type ... 93
3.10.6 Notifier data ... 94
3.10.7 Status codes ... 95
3.10.8 Modifying the actual value ... 95

3.11 Process Dashboard ... 96
3.11.1 Overview ... 96
3.11.2 Creating data block ... 97
3.11.3 Block interface - LAPP_WidgetToolLife ... 100
3.11.4 Data type ... 101
3.11.5 Status codes ... 103

3.12 Hardware diagnostics ... 103
3.12.1 Camera ... 103
3.12.1.1 Overview ... 103
3.12.1.2 Block interface - LAPP_Camera ... 104
3.12.1.3 Data type ... 105
3.12.1.4 Status codes ... 106

3.12.2 Energy efficiency ... 106
3.12.2.1 Overview ... 106
3.12.2.2 Block interface - LAPP_EETransfer ... 109
3.12.2.3 Status codes ... 112
3.12.3 Interface ... 113
3.12.3.1 Overview ... 113
3.12.3.2 Block interface - LAPP_Interface ... 114
3.12.3.3 Data type ... 115
3.12.3.4 Example ... 116
3.12.3.5 Status codes ... 116
3.12.4 Enhanced interface ... 117
3.12.4.1 Overview ... 117
3.12.4.2 Creation of UDT ... 118
3.12.4.3 Creation of tags ... 122
3.12.4.4 Creation of parameter set types ... 126
3.12.5 Motor starter ... 129
3.12.5.1 Overview ... 129
3.12.5.2 Block interface - LAPP_MotorStarter ... 130
3.12.5.3 Data type ... 131
3.12.5.4 Status codes ... 131
3.12.6 PLC system data ... 132
3.12.6.1 Overview ... 132
3.12.6.2 Block interface - LAPP_PLCSystemData ... 133
3.12.6.3 Enabling first scan ... 134
3.12.6.4 Status codes ... 135
3.12.7 RFID ... 135
3.12.7.1 Overview ... 135
3.12.7.2 Block interface - LAPP_RFID ... 137
3.12.7.3 Status codes ... 138
3.12.8 Safety ... 138
3.12.8.1 Overview ... 138
3.12.8.2 Block interface - LAPP_Safety ... 139
3.12.8.3 Status codes ... 141
3.12.9 SINAMICS ... 142
3.12.9.1 Overview ... 142
3.12.9.2 SINAMICS diagnostic blocks using EPOS ... 143
3.12.9.3 SINAMICS diagnostic block using technology objects ... 145
3.12.10 EKS ... 148
3.12.10.1 Overview ... 148
3.12.10.2 EKS for CNC ... 150
3.12.10.3 EKS for PLC ... 155
3.12.10.4 Authorization levels concept ... 157
3.12.10.5 Format of the EKS key ... 158
3.12.10.6 Examples ... 159

3.13 Machine control panel ... 160
3.13.1 LAPP_Powerride ... 160
3.13.1.1 Overview ... 160
3.13.1.2 Block interface - LAPP_Powerride ... 161
3.13.1.3 Create data block MyPowerrideConfig ... 164
3.13.1.4 LED color palette ... 168
3.13.1.5 Example: Preparing to integrate a Powerride via an MCP ... 168
3.13.1.6 Example: calling LAPP_Powerride ... 170

3.13.2 LAPP_UpdateEthernetMcp ... 173
3.13.2.1 Overview ... 173
3.13.2.2 Block interface - LAPP_UpdateEthernetMcp ... 174
3.13.2.3 Status codes ... 180
3.13.3 LAPP_WriteRecordOverEthernet ... 182
3.13.3.1 Overview ... 182
3.13.3.2 Block interface - WriteRecordOverEthernet ... 182
3.13.3.3 Status codes ... 189

4 Business logic blocks ... 191

4.1 GRAPH for manual operation ... 191
4.1.1 Overview ... 191
4.1.2 PLC data types for manual operation using GRAPH blocks ... 191
4.1.3 Program blocks for manual operation using GRAPH blocks ... 192
4.1.3.1 Program blocks and storage in the project navigation ... 192
4.1.3.2 LAPP_blGraphManConfigData ... 192
4.1.3.3 LAPP_blGraphManExtn ... 194
4.1.3.4 LAPP_blGraphManInParam ... 196
4.1.3.5 LAPP_blGraphManOutParam ... 198

4.2 Process data ... 199
4.2.1 LAPP_blPartCnt ... 199
4.2.1.1 Overview ... 199
4.2.1.2 Data type ... 199
4.2.1.3 Block interface ... 201
4.2.1.4 Error codes ... 202
4.2.1.5 Example ... 202
4.2.2 LAPP_blCycleTime ... 202
4.2.2.1 Overview ... 202
4.2.2.2 Data type ... 203
4.2.2.3 Block interface ... 203
4.2.2.4 Error codes ... 204
4.2.2.5 Cycle time chart ... 204

4.3 Modes of operation ... 205
4.3.1 Overview ... 205
4.3.2 LAPP_blModularMCP ... 206
4.3.2.1 Overview ... 206
4.3.2.2 Data type ... 206
4.3.2.3 Block interface ... 208
4.3.2.4 Error codes ... 210
4.3.3 LAPP_blFooter ... 211
4.3.3.1 Overview ... 211
4.3.3.2 Data type ... 212
4.3.3.3 Block interface ... 212
4.3.3.4 Error codes ... 213
4.3.4 LAPP_blOperatingMode ... 213
4.3.4.1 Overview ... 213
4.3.4.2 Data types ... 214
4.3.4.3 Block interface ... 214
4.3.4.4 Error codes ... 215
4.3.5 LAPP_blFctMode ... 216
4.3.5.1 LAPP_blFctModeTl ... 216

4.3.5.2 LAPP_blFctModeNC ... 217
4.3.6 Protection against manipulation ... 221
4.3.6.1 Overview ... 221
4.3.6.2 Block interface - LAPP_blMoCtrl ... 221
5 Abbreviations ... 223
A Appendix A ... 229
A.1 Rulesets ... 229
Index ... 231

1 Introduction

1.1 About SINUMERIK

From simple, standardized CNC machines to premium modular machine designs – the SINUMERIK CNCs offer the right solution for all machine concepts. Whether for individual parts or mass production, simple or complex workpieces – SINUMERIK is the highly dynamic automation solution, integrated for all areas of production. From prototype construction and tool design to mold making, all the way to large-scale series production.

Visit our website for more information SINUMERIK (https://www.siemens.com/sinumerik).

1.2 About this documentation

Target group

The operating manual is aimed at all machine tool users. This publication provides information required by the user to understand the software.

Standard scope

This documentation only describes the functionality of the standard version. This may differ from the scope of the functionality of the system that is actually supplied. Please refer to the ordering documentation only for the functionality of the supplied drive system.

It may be possible to execute other functions in the system which are not described in this documentation. This does not, however, represent an obligation to supply such functions with a new control or when servicing.

For reasons of clarity, this documentation cannot include all of the detailed information on all product types. Further, this documentation cannot take into consideration every conceivable type of installation, operation and service/maintenance.

The machine manufacturer must document any additions or modifications they make to the product themselves.

Websites of third-party companies

This document may contain hyperlinks to third-party websites. Siemens is not responsible for and shall not be liable for these websites and their content. Siemens has no control over the information which appears on these websites and is not responsible for the content and information provided there. The user bears the risk for their use.

1.3 Documentation on the internet

1.3.1 Documentation overview SINUMERIK ONE

Comprehensive documentation about the functions provided in SINUMERIK ONE Version 6.13 and higher is provided in the Documentation overview SINUMERIK ONE (https://support.industry.siemens.com/cs/ww/en/view/109768483).



You can display documents or download them in PDF and HTML5 format.

The documentation is divided into the following categories:

- User: Operating
- User: Programming
- Manufacturer/Service: Functions
- Manufacturer/Service: Hardware
- Manufacturer/Service: Configuration/Setup
- Manufacturer/Service: Safety Integrated
- Information and training
- Manufacturer/Service: SINAMICS

1.3.2 Documentation overview SINUMERIK 840D sl

Comprehensive documentation about the functions provided in SINUMERIK 840D sl Version 4.8 SP4 and higher is provided in the Documentation overview SINUMERIK 840D sl (https://support.industry.siemens.com/cs/ww/en/view/109766213).



You can display the documents or download them in PDF and HTML5 format.

The documentation is divided into the following categories:

- User: Operating
- User: Programming
- Manufacturer/Service: Functions
- Manufacturer/Service: Hardware
- Manufacturer/Service: Configuration/Setup
- Manufacturer/Service: Safety Integrated
- Manufacturer/Service: SINUMERIK Integrate/MindApp
- Information and training
- Manufacturer/Service: SINAMICS

1.3.3 Documentation overview SINUMERIK 828D

Comprehensive documentation about the functions provided in SINUMERIK 828D Version 4.8 SP4 and higher is provided in the 828D documentation overview (https://support.industry.siemens.com/cs/ww/en/view/109766724).



You can display documents or download them in PDF and HTML5 format.

The documentation is divided into the following categories:

- User: Operating
- User: Programming
- Manufacturer/Service: Configuring
- Manufacturer/Service: Commissioning
- Manufacturer/Service: Functions
- Manufacturer/Service: Safety Integrated
- SINUMERIK Integrate/MindApp
- Info & Training

1.3.4 Documentation overview SINUMERIK MC

Comprehensive documentation about the functions provided in SINUMERIK MC Version 1.12 and higher is provided in the SINUMERIK MC documentation overview (https://support.industry.siemens.com/cs/ww/en/view/109771043).



You can display documents or download them in PDF and HTML5 format.

The documentation is divided into the following categories:

- User: Operating
- User: Programming
- Manufacturer/Service: Commissioning
- Manufacturer/Service: Functions
- Manufacturer/Service: Hardware
- Manufacturer/Service: Safety Integrated
- Information and training

1.3.5 Documentation overview SINUMERIK 808D

Comprehensive documentation about the functions provided in SINUMERIK 808D Version 4.91 and higher is provided in the Documentation overview SINUMERIK 808D (https://support.industry.siemens.com/cs/ww/en/view/109780845).



You can display documents or download them in PDF format.

The documentation is divided into the following categories:

- User: Operating
- Manufacturer/Service: Configuring/Commissioning

1.3.6 Documentation overview SINUMERIK operator components

Comprehensive documentation about the SINUMERIK operator components is provided in the Documentation overview SINUMERIK operator components (https://support.industry.siemens.com/cs/document/109783841/technische-dokumentation-zu-sinumerik-bedienkomponenten?dti=0&lc=en-WW).

You can display documents or download them in PDF and HTML5 format.

The documentation is divided into the following categories:

- Operator Panels
- Machine control panels
- Machine Pushbutton Panel
- Handheld Unit / Mini handheld devices
- Further operator components

An overview of the most important documents, entries and links to SINUMERIK is provided at SINUMERIK Overview - Topic Page (https://support.industry.siemens.com/cs/document/109766201/sinumerik-an-overview-of-the-most-important-documents-and-links?dti=0&lc=en-WW).

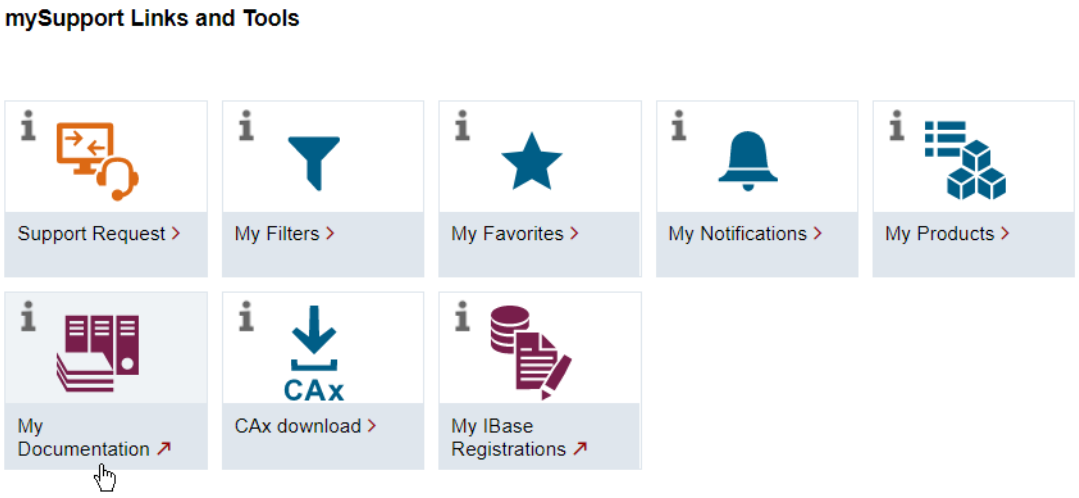
1.4 Feedback on the technical documentation

If you have any questions, suggestions or corrections regarding the technical documentation which is published in the Siemens Industry Online Support, use the link "Provide feedback" which appears at the end of the entry.

1.5 mySupport documentation

With the "mySupport documentation" web-based system you can compile your own individual documentation based on Siemens content, and adapt it for your own machine documentation.

To start the application, click on the "My Documentation" tile on the mySupport homepage (https://support.industry.siemens.com/cs/ww/en/my):



The configured manual can be exported in RTF, PDF or XML format.

Note

Siemens content that supports the mySupport documentation application can be identified by the presence of the "Configure" link.

1.6 Service and Support

Product support

You can find more information about products on the internet:

Product support (https://support.industry.siemens.com/cs/ww/en/)

The following is provided at this address:

- Up-to-date product information (product announcements)
- FAQs (frequently asked questions)
- Manuals
- Downloads
- Newsletters with the latest information about your products
- Global forum for information and best practice sharing between users and specialists
- Local contact persons via our Contacts at Siemens database (→ "Contact")
- Information about field services, repairs, spare parts, and much more (→ "Field Service")

Technical support

Country-specific telephone numbers for technical support are provided on the internet at address (https://support.industry.siemens.com/cs/ww/en/sc/4868) in the "Contact" area.

If you have any technical questions, please use the online form in the "Support Request" area.

Training

You can find information on SITRAIN at the following address (https://www.siemens.com/sitrain).
SITRAIN offers training courses for automation and drives products, systems and solutions from Siemens.

Siemens support on the go



With the award-winning "Industry Online Support" app, you can access more than 300,000 documents for Siemens Industry products – any time and from anywhere. The app can support you in areas including:

- Resolving problems when implementing a project
- Troubleshooting when faults develop
- Expanding a system or planning a new system

Furthermore, you have access to the Technical Forum and other articles from our experts:

- FAQs
- Application examples
- Manuals
- Certificates
- Product announcements and much more

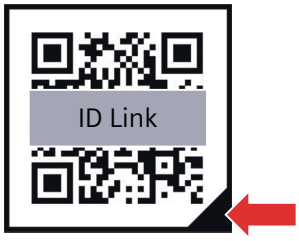
The "Industry Online Support" app is available for Apple iOS and Android.

Digital product information as QR code

The ID link according to IEC 61406 provides you with digital information about your product.

The ID link is a globally unique identifier and is located as a QR code on the product.

You can recognize the ID link by the frame with a black corner at the bottom right.



In addition to the digital nameplate you will find the following product information:

- Technical specifications
- FAQs
- Manuals
- Certificates
- Product announcements
- Application examples

1.7 OpenSSL

Using OpenSSL

This product can contain the following software:

- Software developed by the OpenSSL project for use in the OpenSSL toolkit
- Cryptographic software created by Eric Young.
- Software developed by Eric Young

You can find more information on the internet:

- OpenSSL (https://www.openssl.org)
- Cryptsoft (https://www.cryptsoft.com)

1.8 Compliance with the General Data Protection Regulation

Siemens observes standard data protection principles, in particular the data minimization rules (privacy by design).

For this product, this means:

The product does not process or store any personal data, only technical function data (e.g. time stamps). If the user links this data with other data (e.g. shift plans) or if he/she stores person-related data on the same data medium (e.g. hard disk), thus personalizing this data, he/she must ensure compliance with the applicable data protection stipulations.

Security compliance

2

2.1 Sourcing software updates

Information about software updates

Siemens support center

Siemens software support provides a range of different support resources and ensures that you are working with the latest product versions. Support Center displays Siemens products for which you have valid product access and license entitlement.

On purchase of Create MyHMI /Automotive, a support contract can be signed up which notifies you when there is a new package that can be installed. Alternatively, by using sold-to-number and Webkey Access Code you can check for updates in the Support center. (https://support.sw.siemens.com/en-US)

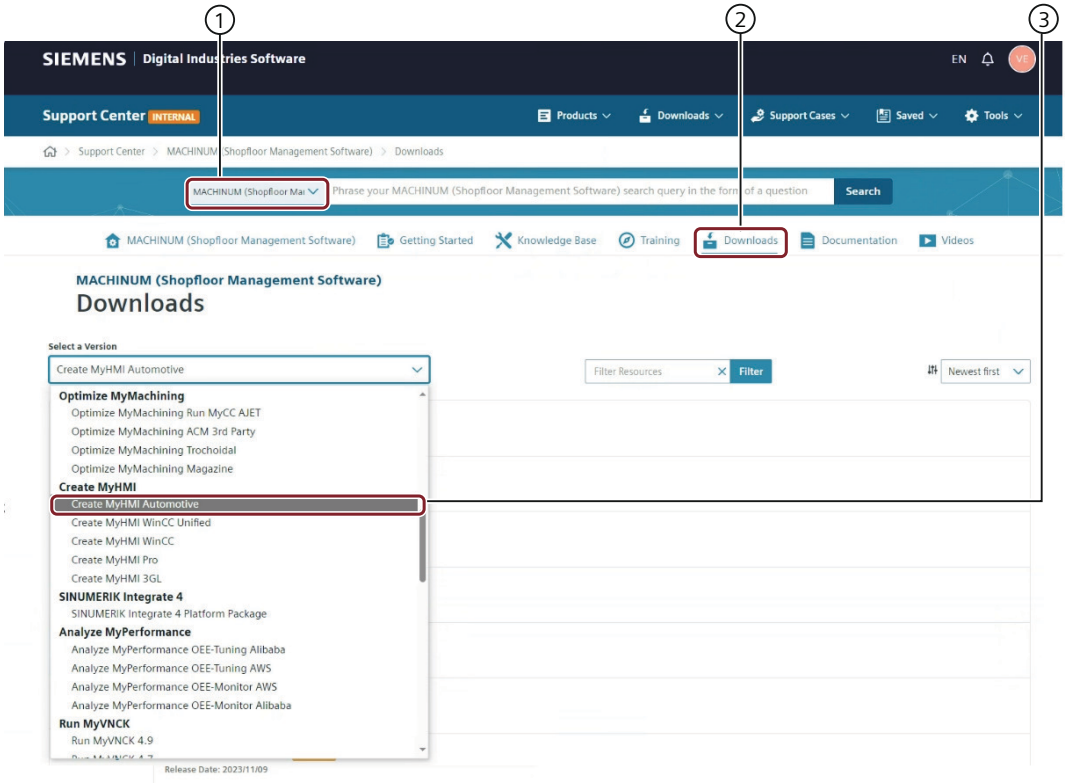


Figure 2-1 Support center

To check for updates in the software center manually, use the following instructions:

1. Enter MACHINUM(Shopfloor Management Software) in the search field.
2. Click the "Downloads" tab.

3. Search “Create MyHMI Automotive” from the list and select it.
4. Download package of the latest version of Create MyHMI /Automotive is displayed.

Siemens ProductCERT

Siemens ProductCERT (https://siemens.com/cert) (Cyber Emergency Readiness Team) is the central department for security-related incidents in the Siemens product and solution environment. Siemens ProductCERT supports development departments in the form of consulting and other services. ProductCERT provides information about current threats and vulnerabilities as well as the appropriate countermeasures.

Sourcing software updates

You can directly source a new software update through your SIEMENS sales person.

You can scan our Contact person database (https://support.industry.siemens.com/cs/ww/en/) if you are not sure who your contact person is.

2.2 Secure product disposal

The data from Create MyHMI /Automotive and its related files can be removed securely from the PC, by uninstalling the application via the setup file "start.exe" of the package.

2.3 Ports and protocols

Ports

- Ensure to keep Port 20008 open in NCU for the SINUMERIK device to support direct download.
- For SINUMERIK OPC UA protocol, Port 48400 is used.

Protocols

- The communication to the PLC with HMI is done via S7 protocol.
- The communication to the SINUMERIK ONE PLC and HMI is done via S7 protocol and SINUMERIK OPC UA.

2.4 Cybersecurity

Create MyHMI /Automotive is addon package to TIA portal. Refer respective documents of WinCC Unified, WinCC Unified Runtime, and Step 7 for Cybersecurity information.

For more information on SINUMERIK ONE Industrial Cybersecurity, refer SIOS portal (https://support.industry.siemens.com/cs/ww/en/view/109925842).

Configuring Automotive PLC program plus blocks 3

3.1 Overview

Overview

NCU device interacts with automotive screens using the PLC blocks. The PLC blocks are responsible for coordination of the interface DBs and HMI. Each block has specific PLC data types (UDT). Use the specific UDT of the block to configure the block. For more information on using PLC data types, refer to "Information system > Programming a PLC > Programming recommendations > Using PLC data types (UDT)" in TIA Portal online help.

From Create MyHMI /Automotive V20, the PLC blocks are converted to Instructions.

Note

To use Create MyHMI /Automotive V19.00.01.01 global library in TIA portal V20 SP1, you can find the below libraries in the OSD setup:

- SINUMERIK ONE PLC Automotive Program V19.00.01.01
- S71500 PLC Automotive Program V19.00.01.01
- S71500 SoftPLC Automotive Program V19.00.01.01
- SINUMERIK ONE PLC Automotive Program V19.00.01.02
- S71500 PLC Automotive Program V19.00.01.02
- S71500 SoftPLC Automotive Program V19.00.01.02

For more information, refer to Create MyHMI /Automotive installation guide in "Support" folder of the OSD.

Note

- Upgrade of global library to V20 is not possible.
- To use the new "CMHA Data Blocks" and Instructions, you need to create a new project and program as per the need using Create MyHMI /Automotive Instructions.

The following table lists the instructions of Create MyHMI /Automotive:

Name	Description
LAPP_Registration	Mandatory block Registers Create MyHMI /Automotive basic program. Recommended to call at the beginning of the startup OB before all other instructions.
LAPP_Init	Mandatory block Initializes the basic program when starting. Recommended to call at the beginning of the startup OB before all other instructions.
LAPP_ScreenSwitch	Allows to dynamically switch between the screens in the HMI.
LAPP_ShiftPlanning	Allows to get a detailed report on the number of workpieces produced during different shifts in a day.

Name	Description
LAPP_WidgetToolLife	Allows to configure the tool life widget in the "LAHP_ProcessDashboard" runtime screen.
LAPP_Camera*	Allows to view live image from camera.
LAPP_EETransfer	Allows to measure the energy consumption.
LAPP_Interface	Allows to view the status of the inputs and outputs for the selected interface.
LAPP_MotorStarter	Writes data to WinCC screens for the motor starter diagnostics.
LAPP_PLCSystemData	Reads the PLC system data on the first scan of PLC startup and writes the data to WinCC Screen.
LAPP_RFID	Writes data to WinCC screens for the RFID diagnostics.
LAPP_Safety	Used to monitor failsafe functions of the PLC.
LAPP_SINAMICSEpos	Reads the drive parameters and actual values directly from drive.
LAPP_SINAMICSTO	Reads data of the technology objects.
LAPP_EKS_PLC*	Reads the key data of Electronic Key System.
LAPP_EKS_CNC**	Reads the key data of Electronic Key System.
LAPP_SetAccessLevel**	Used for SINUMERIK Operate login and logout.
LAPP_FuncGrpConfig	Used to configure the number of lines, areas, and the HMI index required for the function group.
LAPP_ManualOperationMain	Mandatory block for Manual operation Configure HMI index.
LAPP_ManualOperation	Used to configure the mode of operation and the manualoperation line parameters.
LAPP_ManualOperationFaceplate	Used to configure manual operation lines in any of the required screen as a faceplate.
LAPP_ToolLifeMain	Mandatory block for tool life monitoring Used to define number of tools once in the cyclic block or main block.
LAPP_ToolLifeMonitor	Used to indicate the life expectancy of the tools used in machining.
LAPP_WorkpieceManager_PLC*	Used to tune the part program created and record the actual time taken to complete manufacturing of a part.
LAPP_WorkpieceManager_CNC**	Used to tune the part program created and record the actual time taken to complete manufacturing of a part.
LAPP_Powerride	Integrates a rotary/push control dial (SINUMERIK Powerride) into the SINUMERIK environment.
LAPP_UpdateEthernetMcp	Used for communication with operator components via Industrial Ethernet.
LAPP_WriteRecordOverEthernet	Used to transfers a data set (record) to an MCP which is connected to the SINUMERIK ONE via Ethernet.

* - Used only for PLC

** - Used only for CNC

3.2 Programming PLC Instructions

Procedure

Follow the below procedure to add block in function block:

1. In startup OB, e.g. Startup [OB100], call "LAPP_Registration" block and interface the input parameters. For more information, refer to AUTOHOTSPOT .

   Note

   This is a mandatory block to be called.

2. In the cyclic block (Main [OB1]), call "LAPP_Init" block and interface the input parameters. For more information, refer to "LAPP_Init (Page 32)".

   Note

   This is a mandatory block to be called.

3. Drag the LAPP_<block name> block from "Instructions > Create MyHMI /Automotive" and add it to the cyclic block or main block.

   Note

   Ensure CMHA data blocks from "Global libraries > CMH Automotive Program Vxx.xx.xx.xx > Master copies > CMHA PLC Programs" are added to "CNC [NCU xxxx] > PLC > Program blocks".

4. Create a data block and create the required PLC tags using UDT of the block. For more information on using UDT, refer to "Information system > Programming a PLC > Programming recommendations > Using PLC data types (UDT)" in TIA Portal online help.

   Note

   You can find the required data type only after adding the instruction to OB.

5. Assign the necessary inputs and outputs of the block.

Note

- Runtime screens are inter-dependent and will not work as expected if you call wrong PLC instruction.
- Refer to the subsequent topics to know more about the data types, input, and output parameters of the library blocks.

Example

Using UDT for "LAPP_Camera"

To use PLC data type within a data block as data type of a tag. Proceed as follows:

1. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
2. Click the "Data block (DB)" button.

3. Enter the name Demo and click "OK".
4. In Demo DB, create tag camera and select "LAPP_typeCameraIPConfig" as data type.
   or
   In Demo DB, create tag camera and select Array[0...2] of "LAPP_typeCameraIPConfig" as data type. (You can configure up to 3 cameras).
5. Create a "hmiIndex" tag with USInt as data type.
6. Call the "LAPP_Camera" block and interface the parameters as shown below.

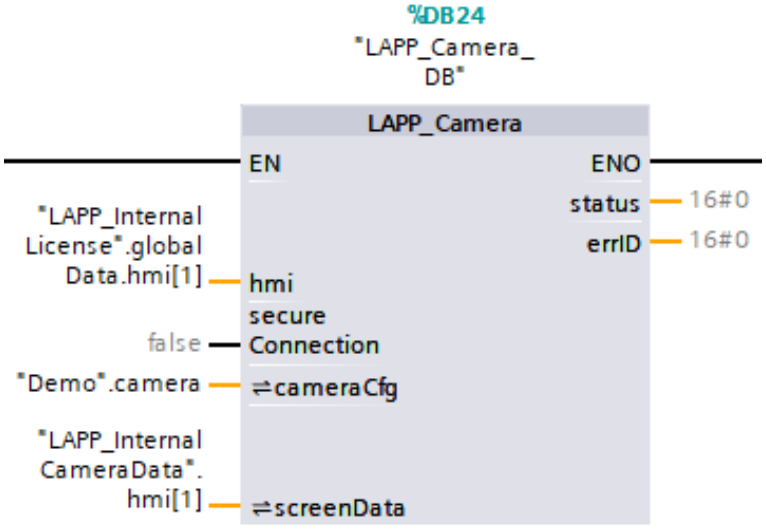


Figure 3-1 "LAPP_Camera"

3.3 Upgrading V19 to V20 TIA portal project

You can upgrade CMHA TIA Portal project from V19 to V20 Update 2 of the TIA portal. You need to perform the following steps to upgrade the project from an earlier version:

1. Open your TIA portal project.
2. Upgrade the device versions.
3. Upgrade PLC data blocks and business logic blocks
4. Convert PLC blocks to instructions
5. Upgrade CMHA library to HMI

Precondition

TIA Portal V20 Update 2 is installed.

Procedure

Follow the steps below for upgrading V19 project:

1. Open TIA Portal project and confirm the upgrade in the query dialog.

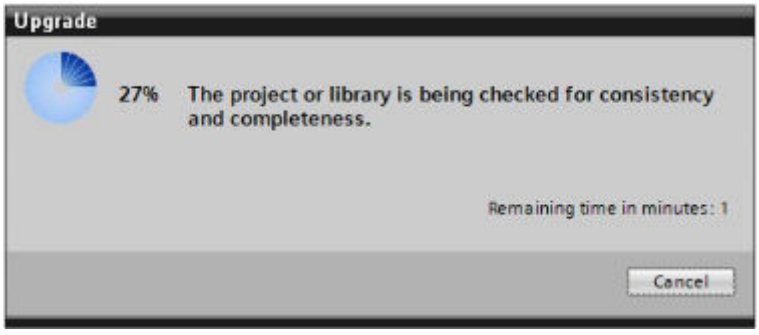


Figure 3-2 Upgrade

2. In "Project tree", navigate to "CNC_1 > Device configuration > Device view" and select the module.
3. In the Inspector window, navigate to "Properties > General > Catalog information". Click "Change firmware version" button.
   The "Change device" window opens.

4. In "New device:" section, select the "Version" to V6.25 as shown and then click "OK".

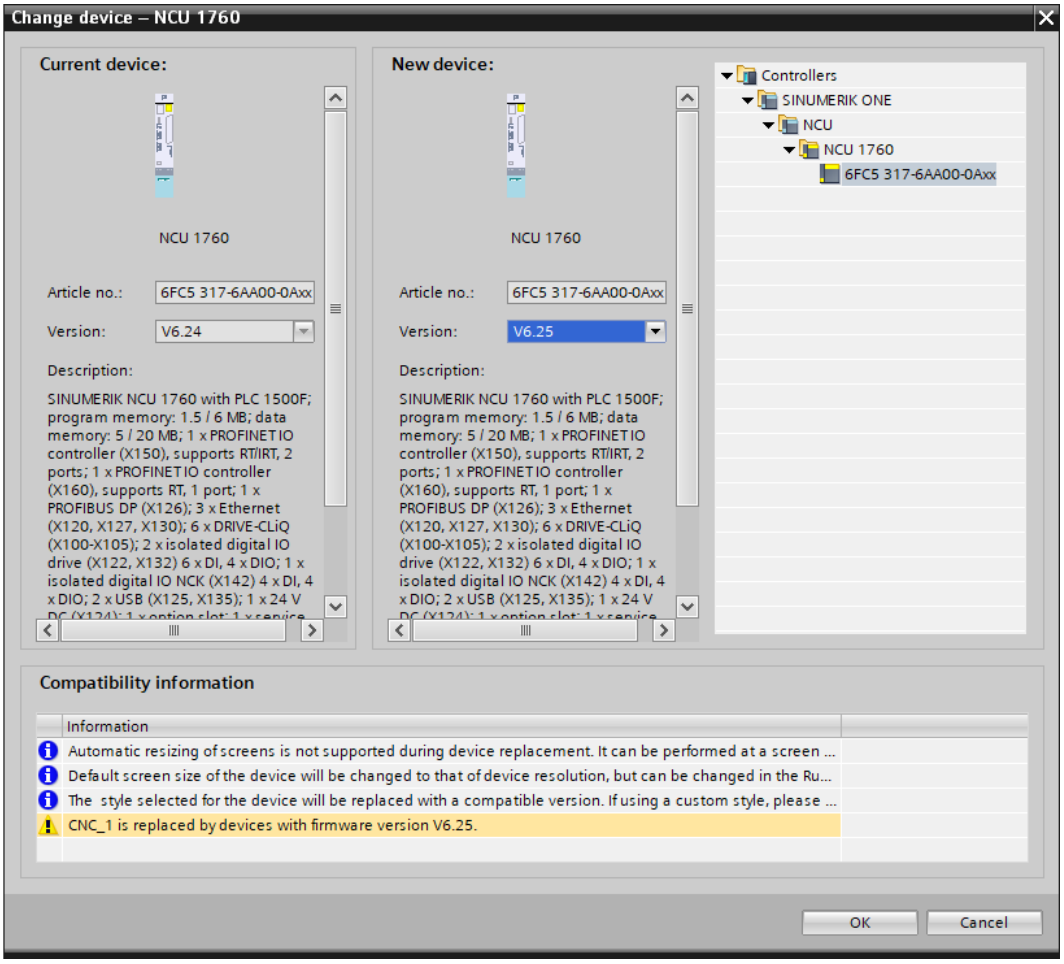


Figure 3-3 Change device

5. In "Project tree", open the "Program blocks" folder and then delete "BusinessLogic Blocks", "CMHA Blocks", and "UserTemplate blocks" folders.

Note

Ensure to delete only the blocks used from V19 CMHA library.

6. Open the "PLC data types" folder and then delete "BusinessLogic Types" and "CMHA Types" folders.

Note

Ensure to delete only the types used from V19 CMHA library.

7. From "Program blocks", generate and save the source files of all the DBs created. For example, in the below image for all the selected DBs of manual operation, source files are generated.

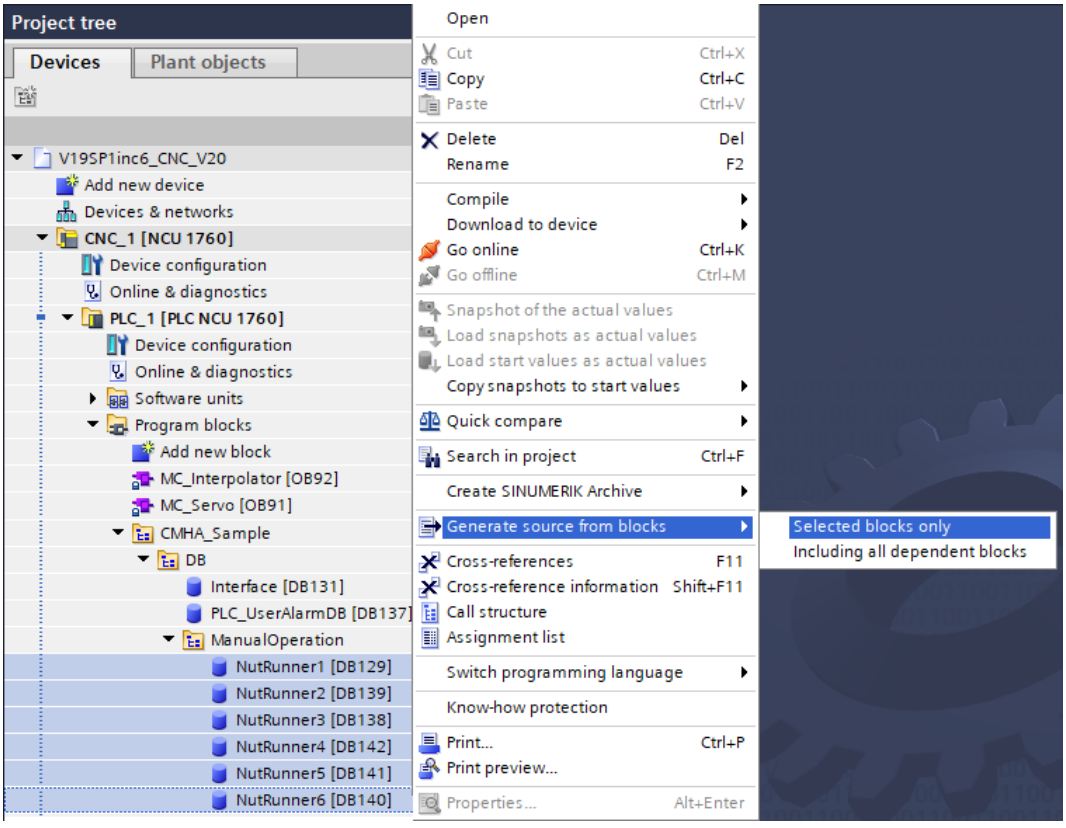


Figure 3-4 Source generation

8. Delete the CMHA DBs created in the project once the source is generated and saved.
9. In "Libraries" task card, open "CMH Automotive Program V20.0.0.0" library.
10. From "Master copies > CMHA PLC Program", drag and drop "BuisnessLogic Blocks > SINUMERIK ONE" and "UserTemplate Blocks" folders to "Program Blocks".
11. From "Master copies > CMHA PLC Program", drag and drop "CMHA Data Blocks" to "External source files". For more information, refer to TIA Portal online help.
12. In "Project tree > PLC > External source files > CMHA Data Blocks", right-click on the DB and select "Generate blocks from source".

Note

You can move the DBs in the required folder structure.

13. From "Master copies > CMHA PLC Program", drag and drop "BuisnessLogic Types > SINUMERIK ONE" to "PLC data types".
14. Right-click on the SINUMERIK ONE device and select "Compile > Hardware (rebuild all)".
15. Right-click on the SINUMERIK ONE device and select "Compile > Software (rebuild all)". You will encounter errors with respect to the PLC blocks or any tag which is not defined.
16. Double-click on the error to navigate to the respective block or tag which needs to be fixed. For example, to fix the issue with "NutRunner2".Line in "Network 1".

17. In "Instructions" task card, navigate to "Optional packages > Create MyHMI /Automotive > Manual operation" and then drag and drop "LAPP_FuncGrpConfig" to "Network 1" as shown.

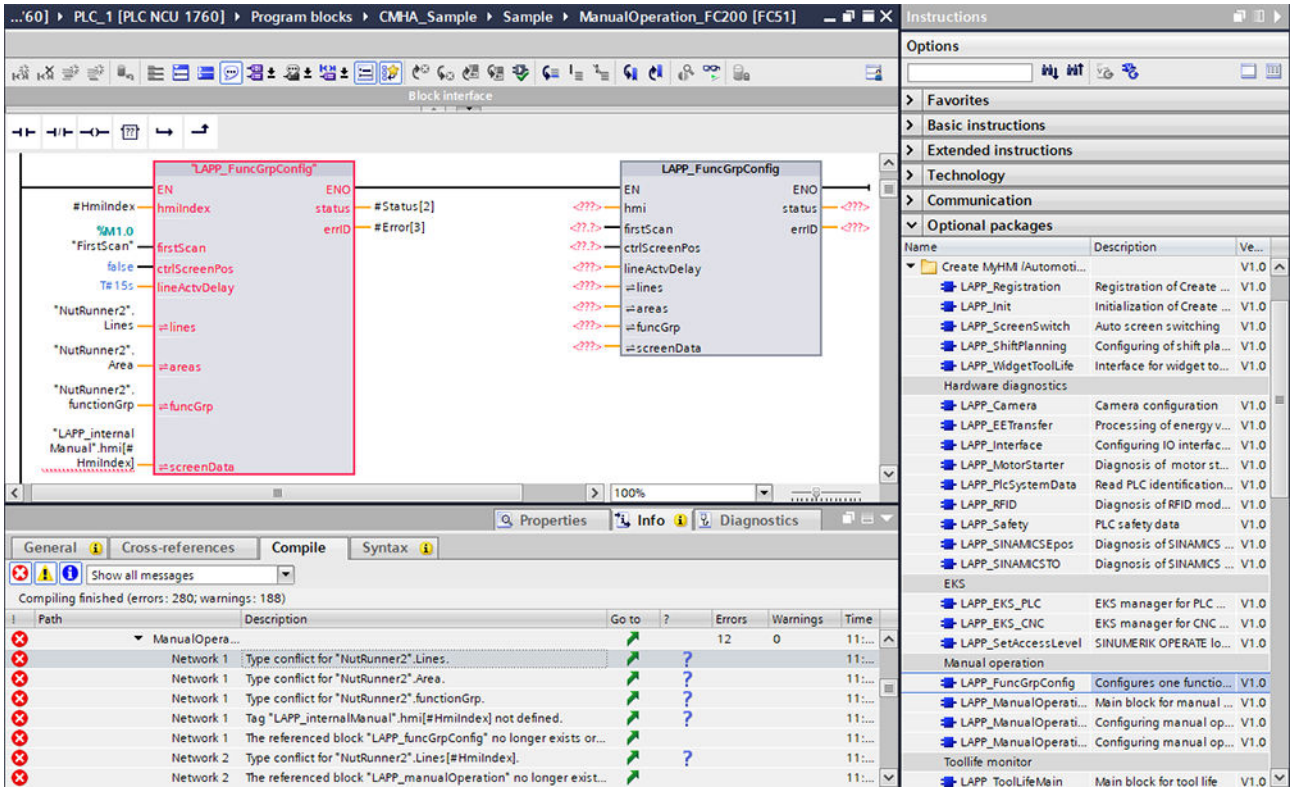


Figure 3-5 Block error

18. Interface all the input and output parameters for the instruction as shown.

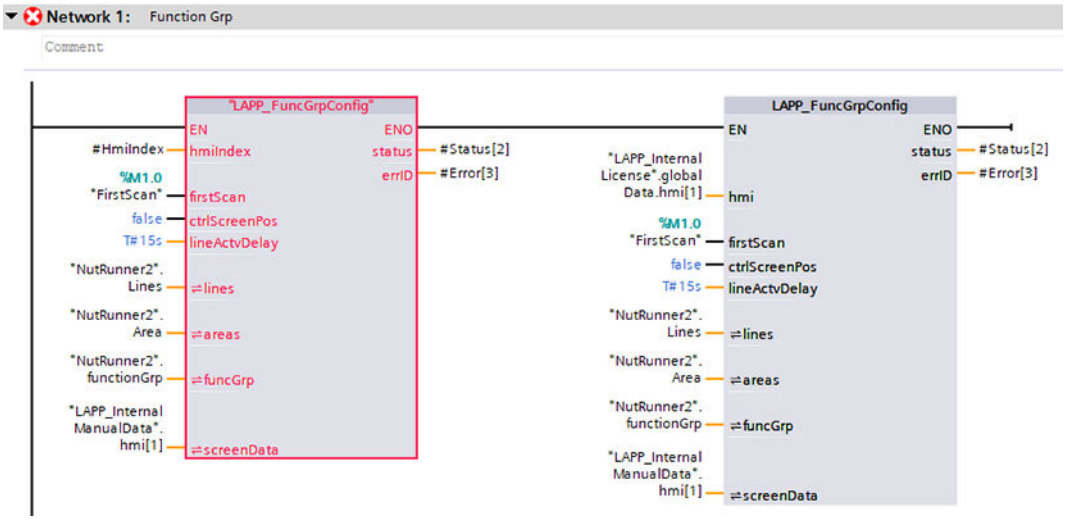


Figure 3-6 Network error

Note

- For any new input parameter added in the instruction, refer to TIA Portal online help for more information.
- Ensure to interface the right input and output parameters for the instruction. For more information, refer to the TIA Portal online help.
- For few instructions, you are asked to create DB. It is recommended to create "Multi instance DB".

19. Delete the PLC block.
    Similarly, fix all the errors by replacing PLC block with PLC Instructions.
20. Right-click on the SINUMERIK ONE device and select "Compile > Hardware (rebuild all)".
21. Right-click on the SINUMERIK ONE device and select "Compile > Software (rebuild all)".

22. In "Project tree", navigate to "HMI device > Device configuration > Device view" and right-click on the module and then select "Change device / version".

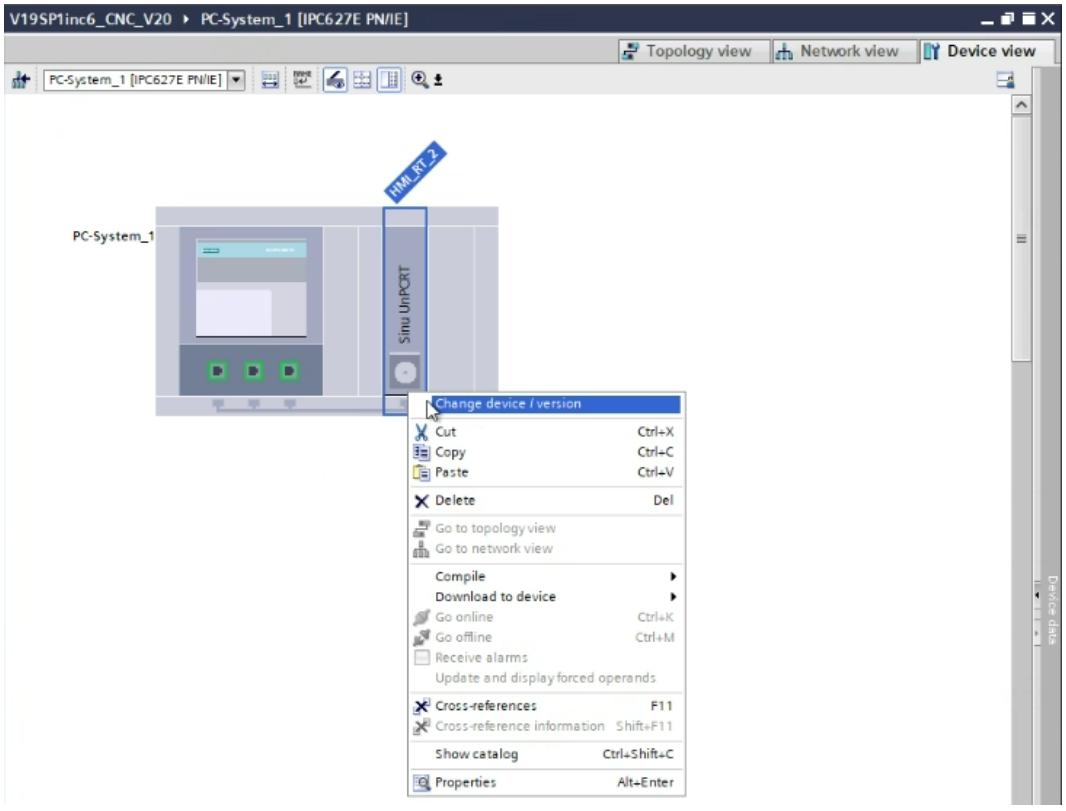


Figure 3-7 Change device

23. In "New device:" section, select the "Version" to V20.0.0.0 as shown and then click "OK".

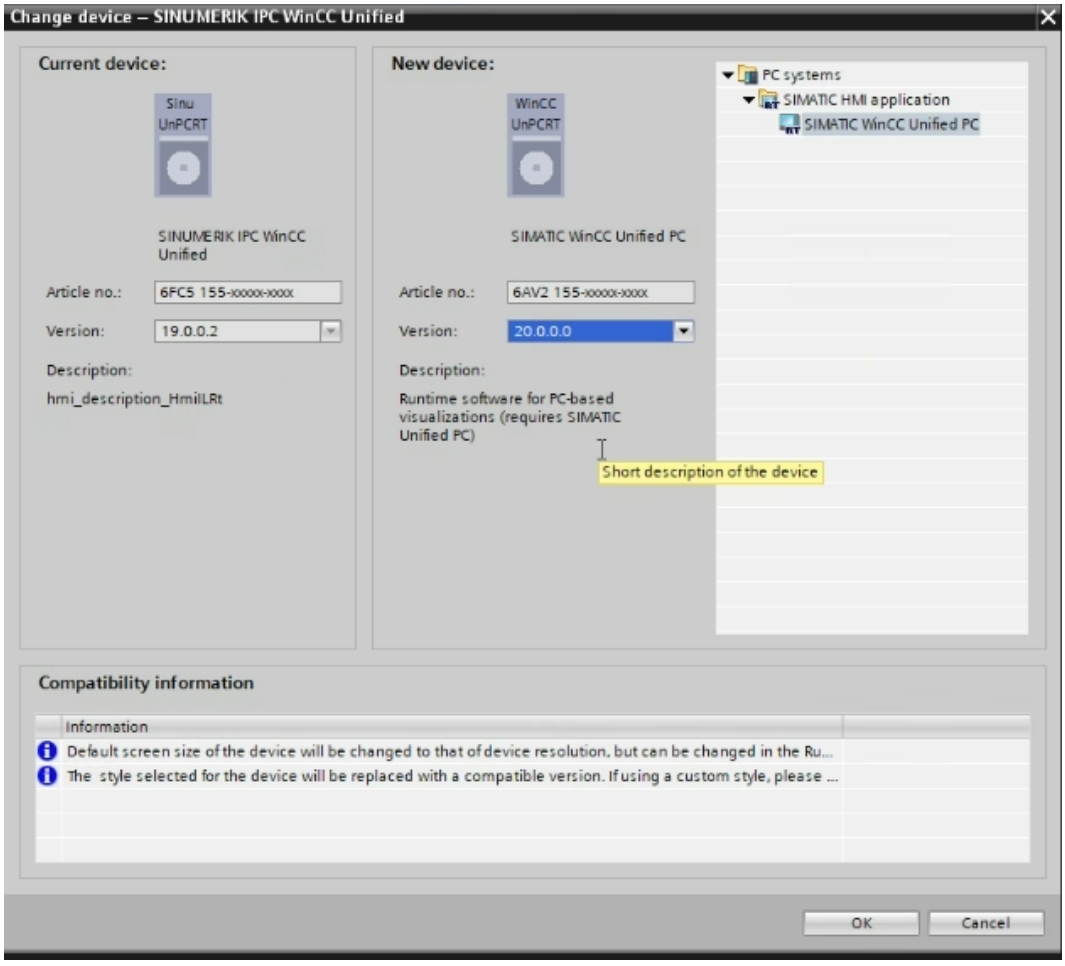


Figure 3-8 Change device

Note

This option is only available from V20 Update 2 of TIA portal.

24. Save all the OEM screens and customer changes for CMHA screens.
25. Rename the CMHA screens which you wish to save.
26. Delete all the CMHA V19 library screens, tags, and CMHA alarms which are shown as error.
27. In "Project library > Types", delete ControlerTypes_CNC, Faceplates_CNC, HMI types_CNC, Scripts_CNC.

Note

This is a mandatory step to remove the folders. For PLC, remove folders with "*_PLC".

28. In "Libraries" task card, open "CMH Automotive Program V20.0.0.0 > CMHA HMI Controld" library.

29. Drag and drop CMHA HMI Tags, Cycles, ProjectGraphics, ScheduledTasks, Scripts, TextLists, and Screens (Required resolution) to HMI device in the respective folders.

Note

Ensure the project status is shown as green in "Types".

30. From "CMH Automotive Program V20.0.0.0 > Types > Faceplates", drag and drop "Common" folder to "Project library".
31. Right-click on the HMI device and select "Compile > Hardware (rebuild all)".
32. Right-click on the SINUMERIK ONE device and select "Compile > Software (rebuild all)". Project is upgraded to V20 of TIA portal.

Note

In case of further errors, ensure to do HMI tag synchronize with PLC.

3.4 LAPP_Init

3.4.1 Block interface - LAPP_Init

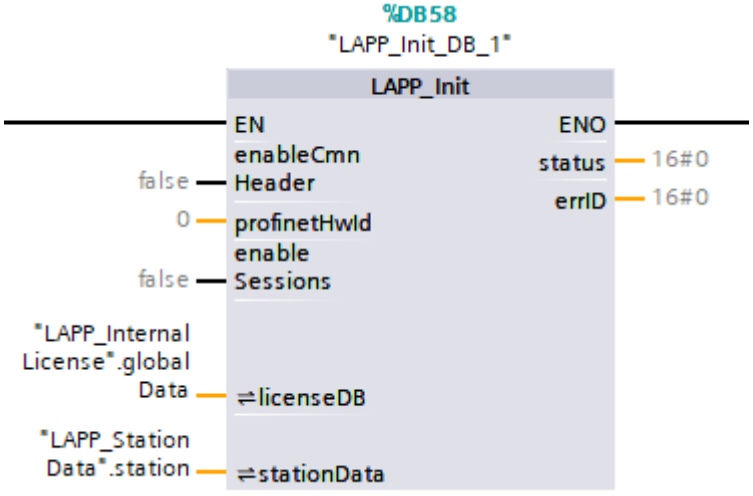


Figure 3-9 "LAPP_Init"

Parameters

Table 3-1 Parameters of "LAPP_Init"

Name	Declaration	Type	Description
enableCmnHeader	Input	Bool	Enables the common header across all the HMIs configured. Displays the selected station details across all the HMIs. • 0 - Disable • 1 - Enable For example, Consider that you have 3 HMIs with common header enabled and station 4 is selected. • If you select station 10 on HMI 2, then the information of machine info, transline mode, basic position, plant status, and part ready of station 10 is displayed on all the 3 HMI's header. • If you select station 1 on HMI 2, then the information of machine info, transline mode, basic position, plant status, and part ready of station 10 is displayed on all the 3 HMI's header. **Note**: If common header is not enabled, then each HMI header will show the details of the respective station selected.
profinetHwId	Input	HW_IO	Interface the system constant value of the PROFINET port used for license generation. **Note:** This parameter is not applicable for CNC device.
enableSessions	Input	Bool	Enables session management. • 0 - Disable • 1 - Enable
licenseDB	InOut	LAPP_typeIntlGlobalData	Interface the "LAPP_InternalLicense" DB. Ensure to interface the same DB in "LAPP_Registration" block. **Note:** If you cannot find "LAHP_InternalLicense" DB, then drag "CMHA Data Blocks" folder from "Global libraries > CMH Automotive Program Vxx.xx.xx.xx > Master copies > CMHA PLC Program".
stationData	InOut	Array of LAPP_typeStation	Interface the number of stations available. For more information, refer to Configuring station data (Page 35).
status	Output	Word	Displays the status codes of the block. For more information, refer to Status codes (Page 34).
errID	Output	Word	Displays the error codes of the block. For more information, refer to "Information System > Programming a PLC > Instructions > Instructions (S7-1200, S7-1500) > Basic instructions (S7-1200, S7-1500) > Program control operations (S7-1500) > Runtime control (S7-1500) > GET_ERR_ID" instruction in TIA portal online help.

Hardware ID of the PROFINET interface (PLC)

In "Device view", select the PROFINET interface and navigate to "Properties > System constants" tab in the inspector window.

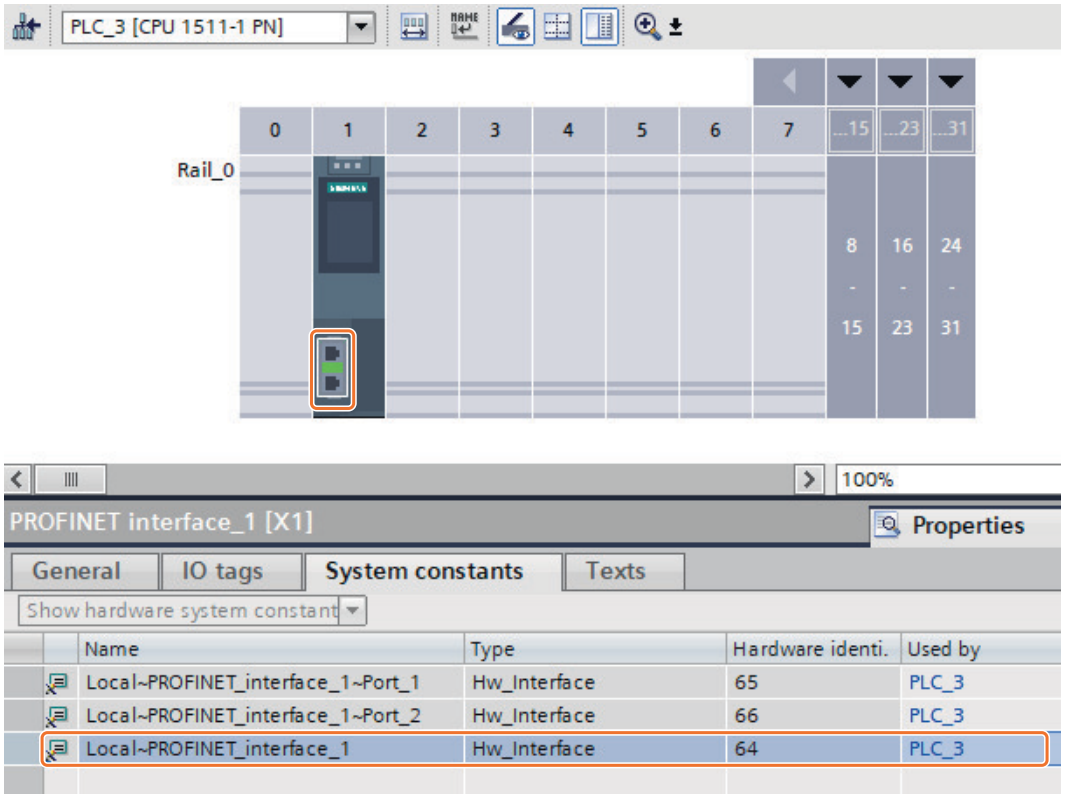


Figure 3-10 PROFINET interface hardware ID

3.4.2 Status codes

List

Following are the status codes for "LAPP_Init".

Status code	Description
16#7000	Block is in initialized state.
16#7002	Block is in execution state and has no errors.
16#8201	The input parameter "numOfHmis" in "LAPP_Registration" block is not in the valid range. Minimum value is 0 and maximum is 40.
16#8202	Invalid array value configuration for stations.

3.5 Configuring station data

3.5.1 Configuring header

Using "LAPP_Init" mandatory configuration block, you can configure header for different stations. You can configure the station data on header in the following ways:

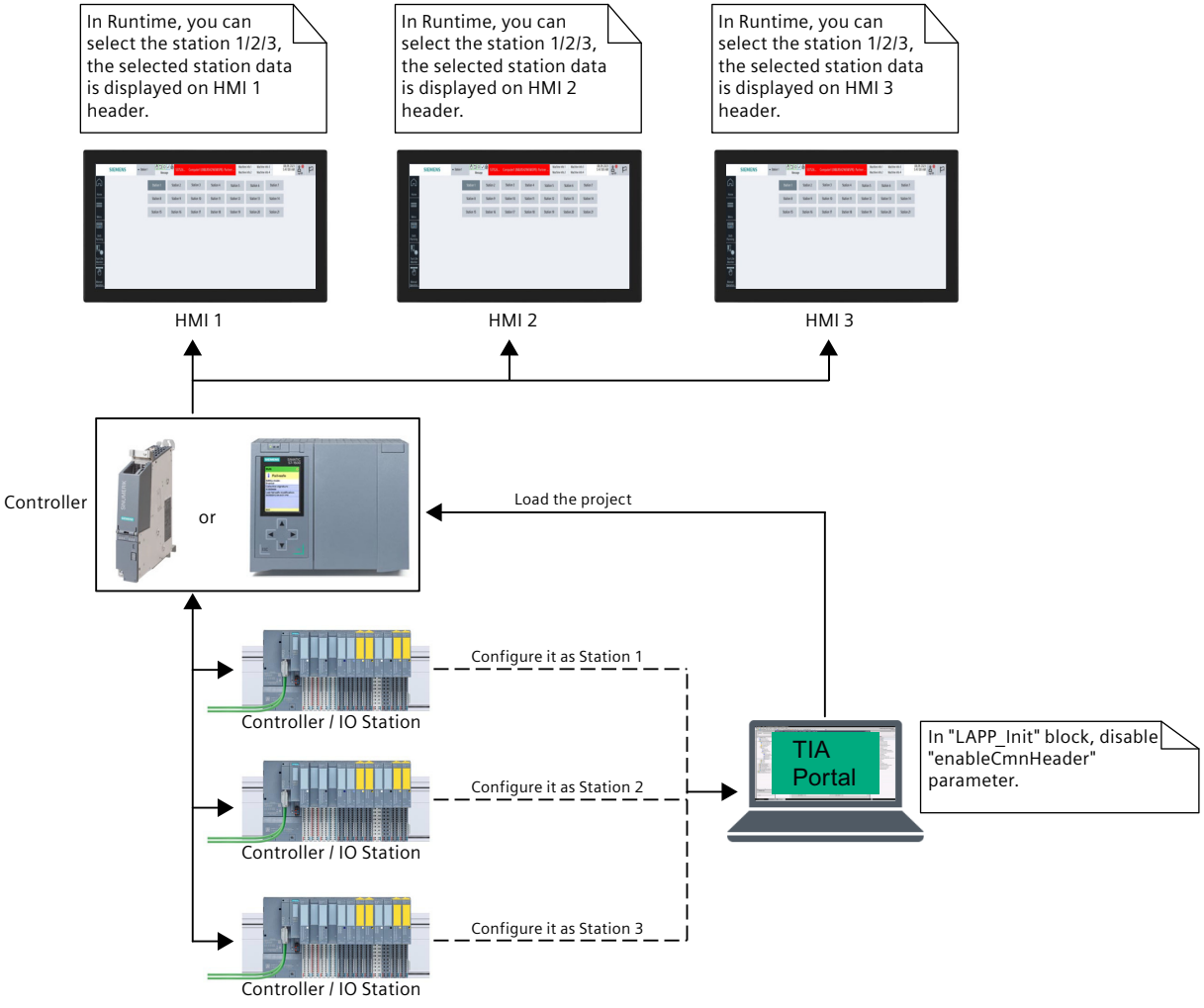


Figure 3-11 Scenario 1 - Common header disabled

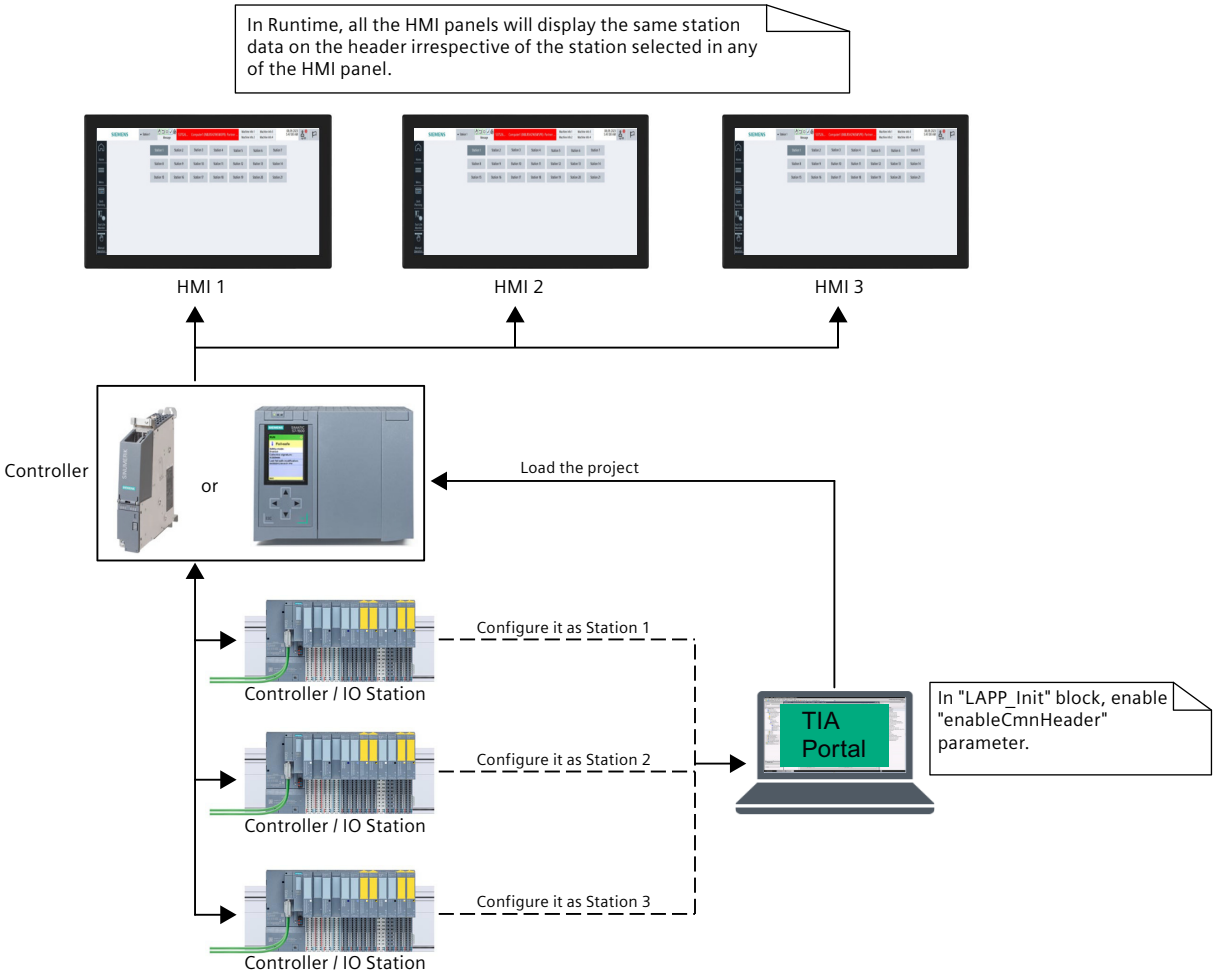


Figure 3-12 Scenario 2 - Common header enabled

Example

Scenario 1
If you are using SINUMERIK ONE embedded with additional ITC, then the HMI devices are always synchronized and one device can be operated at a time.

Scenario 2
If you are using SINUMERIK ONE embedded with additional IPC, and common header is enabled, then the HMI devices will display the same station data in the header.

Scenario 3
If you are using SINUMERIK ONE embedded with additional IPC, and common header is disabled, then the station data displayed on embedded display and IPC will be independent.

Scenario 4
If you are using PLC/SINUMERIK ONE with 3 HMI devices and common header is enabled, then all HMI devices will display same station data and can be operated independently.

Scenario 5
If you are using PLC/SINUMERIK ONE with 3 HMI devices and common header is disabled, then the HMI devices will display the selected station data respectively and can be operated independently.

Configuring number of stations

Configure the number of stations required by modifying the array value in "LAPP_StationData.station".

Name of the station and message can be modified in the textlist.

For example,
Scenario 1
To configure maximum stations to 10, then enter the value at "LAPP_StationData.station" as shown:

LAPP_StationData

	Name	Data type
1	Static	
2	station	Array[1..10] of LAPP_typeStation
3	station[1]	LAPP_typeStation
4	station[2]	LAPP_typeStation
5	station[3]	LAPP_typeStation
6	station[4]	LAPP_typeStation
7	station[5]	LAPP_typeStation
8	station[6]	LAPP_typeStation
9	station[7]	LAPP_typeStation
10	station[8]	LAPP_typeStation
11	station[9]	LAPP_typeStation
12	station[10]	LAPP_typeStation

Figure 3-13 "LAPP_StationData"

Note

By default, the "LAHP_StationDropDown" screen is configured with 21 buttons. Ensure to remove the unused station buttons.

Result
In runtime, you can view as below:

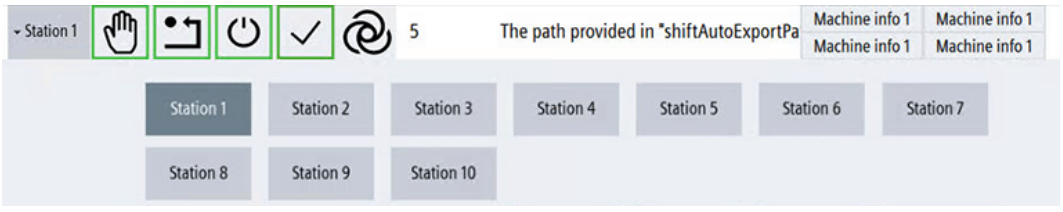


Figure 3-14 Header - Station list

Scenario 2
To configure maximum stations to 14, then enter the value at "LAPP_StationData.station" as shown:

LAPP_StationData

	Name	Data type
1	Static	
2	station	Array[1..30] of LAPP_typeStation
3	station[1]	LAPP_typeStation
4	station[2]	LAPP_typeStation
5	station[3]	LAPP_typeStation
6	station[4]	LAPP_typeStation
7	station[5]	LAPP_typeStation
8	station[6]	LAPP_typeStation
9	station[7]	LAPP_typeStation
10	station[8]	LAPP_typeStation
11	station[9]	LAPP_typeStation
12	station[10]	LAPP_typeStation
13	station[11]	LAPP_typeStation
14	station[12]	LAPP_typeStation
15	station[13]	LAPP_typeStation
16	station[14]	LAPP_typeStation
17	station[15]	LAPP_typeStation
18	station[16]	LAPP_typeStation
19	station[17]	LAPP_typeStation
20	station[18]	LAPP_typeStation
21	station[19]	LAPP_typeStation
22	station[20]	LAPP_typeStation
23	station[21]	LAPP_typeStation
24	station[22]	LAPP_typeStation
25	station[23]	LAPP_typeStation
26	station[24]	LAPP_typeStation
27	station[25]	LAPP_typeStation
28	station[26]	LAPP_typeStation
29	station[27]	LAPP_typeStation
30	station[28]	LAPP_typeStation
31	station[29]	LAPP_typeStation
32	station[30]	LAPP_typeStation

Figure 3-15 "LAPP_StationData"

Note

By default, the "LAHP_StationDropDown" screen is configured with 21 buttons. Ensure to add the required number of station buttons and configure its properties.

For more information, refer to Editing station buttons.

Result
In runtime, you can view as below:

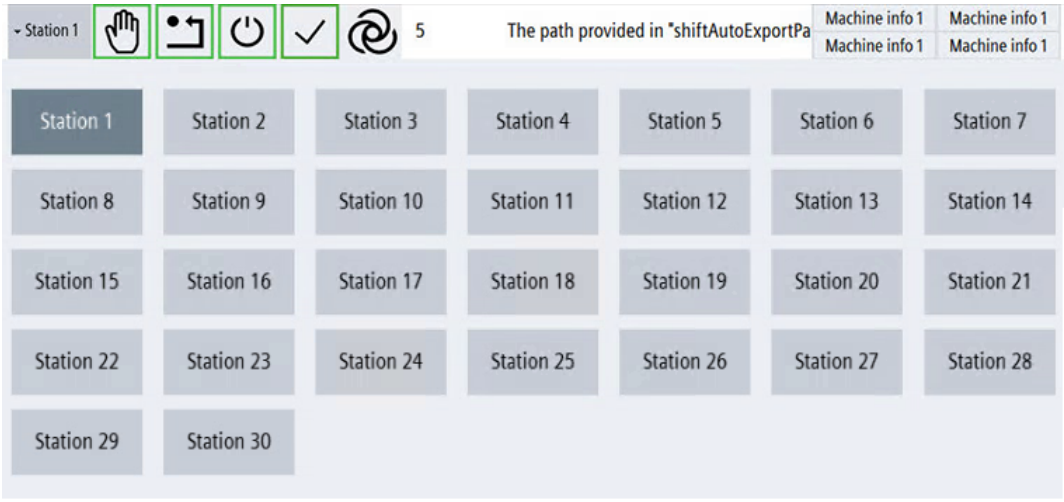


Figure 3-16 Header - Station list

3.5.2 Data type

LAPP_typeStation

Table 3-2 Structure of data type "LAPP_typeStation"

Name	Data type	Description
status	LAPP_typeStationStatus	You can configure the translation mode, initial position, plant, part ready, and MSO. For more information, refer to "Configuring Create MyHMI /Automotive > Project engineering > Using dynamic SVGs > Base elements" in TIA Portal online help.
Info	LAPP_typeInfo	You can configure the machine status information. For more information, refer to "Configuring Create MyHMI /Automotive > Project engineering > Automotive HMI plus screens > Header > Editing Machine Information" in TIA Portal online help.

3.6 Screen switching

3.6.1 Overview

The screen switching block can be configured to dynamically switch between the screens in the HMI.

Note

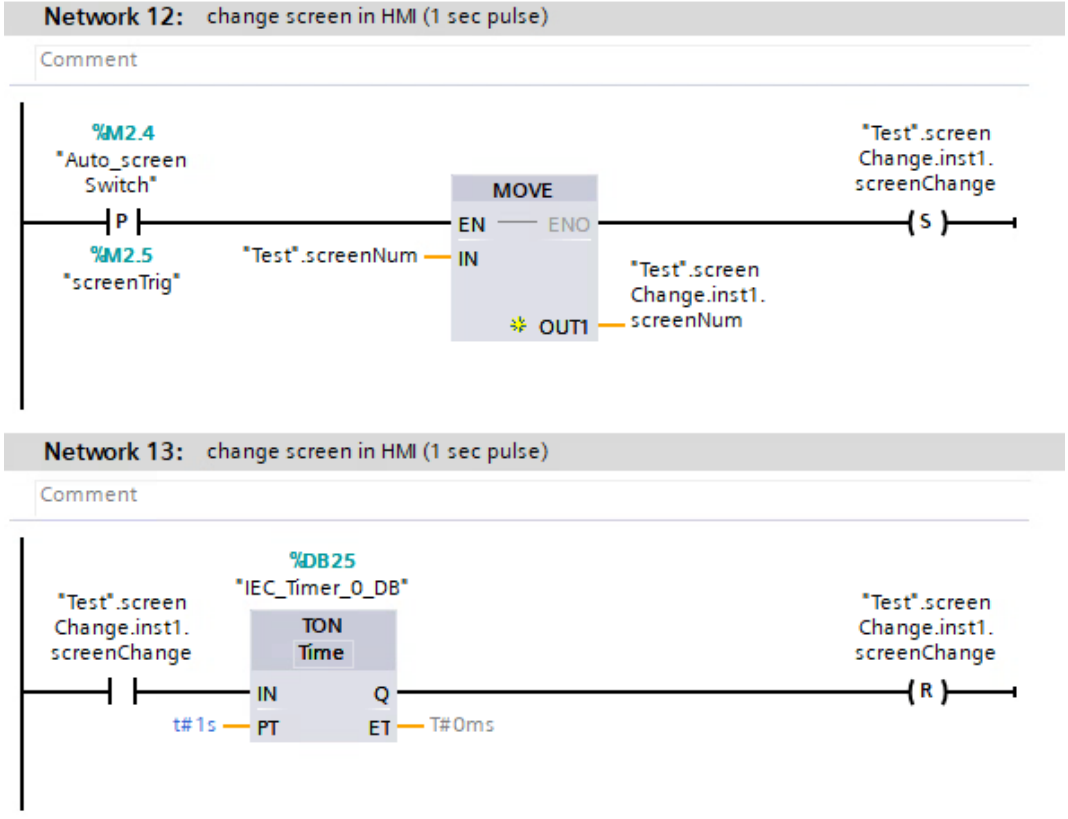
The block configuration for multiple instances (inst1 and inst2) is considered for Full HD resolution with display manager enabled.

In case of Non-HD, HD Ready, and Full HD without display manager enabled, the block configuration for single instance (inst1) is considered.

Procedure to configure the block

Follow the steps below for the screen switching feature to work as intended.

1. Configure the block "LAPP_ScreenSwitch (Page 41)" and interface the required input and output parameters.
2. To set/ reset the "screenChange" bit apply the logic shown in the figure below.



For more information on configuring the blocks refer AUTOHOTSPOT.

3.6.2 Block interface - LAPP_ScreenSwitch

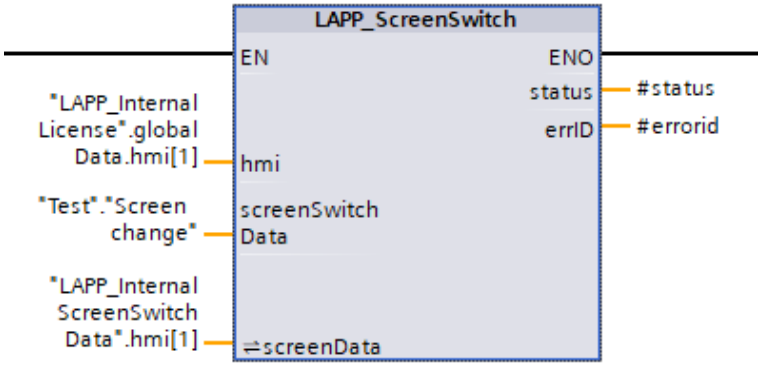


Figure 3-17 "LAPP_ScreenSwitch"

Parameters

Table 3-3 Parameters of "LAPP_ScreenSwitch"

Name	Declaration	Type	Description
hmi	Input	LAPP_typeIntlLicense	Configure the HMI Index which is defined while setting up the HMI device. **Note**: HMI index value cannot be 0. It should match with HMI index value used in License DB ("LAPP_InternalLicense").
screenSwitchData	Input	LAPP_typeScreenSwitchConfig	Screen switching data for the HMI.
screenData	InOut	LAPP_typeIntlHmiScreenSwitchData	Interface the HMI tag from "LAPP_InternalScreenSwitchData.hmi[y]". **Note**: "LAPP_InternalScreenSwitchData" is an internal DB. Modify the array value to increase the HMI device support. For more information, refer to Configuring PLC blocks for different HMI.
status	Output	Word	Displays the status codes of the block. For more information refer to Status codes (Page 42).
errID	Output	Word	Displays the error codes of the block. For more information, refer to "Information System > Programming a PLC > Instructions > Instructions (S7-1200, S7-1500) > Basic instructions (S7-1200, S7-1500) > Program control operations (S7-1500) > Runtime control (S7-1500) > GET_ERR_ID" instruction in TIA portal online help.

Data type

Structure of data type LAPP_typeScreenSwitchConfig

Name	Type	Description
inst1	LAPP_typeIntlScreenInst	Screen switching data for instance 1.
inst2	LAPP_typeIntlScreenInst	Screen switching data for instance 2.

Structure of data type LAPP_typeScreenInst

Name	Type	Description
screenChange	Bool	Screen change command for HMI (1 second pulse)
screenNum	USInt	The screen number to display the required screen in HMI. The value for the respective screens can be obtained from the text list.
funcGrpNum	UInt	Enter the function group number of the manual operation screen you wish to display in HMI. Note: Same function group cannot be configured for both the instances.

3.6.3 Status codes

List

The status codes for screen switching block are:

Status code	Description
16#7000	Block is in initialized state.
16#7002	Block is in execution state and has no errors.
16#8200	Faulty license.
16#8201	Invalid HMI index.
16#8400	No connection to HMI.

3.6.4 Example for manual operation screens

Configuring manual operation screen in both the instance

Set the parameter values as shown in the image for both instances:

Screen switch

	Name	Data type	Start value
1	Static		
2	Test	"LAPP_typeScreenSwitchConfig"	
3	inst1	"LAPP_typeScreenInst"	
4	screenChange	Bool	true
5	screenNum	USInt	33
6	funcGrpNum	UInt	100
7	inst2	"LAPP_typeScreenInst"	
8	screenChange	Bool	true
9	screenNum	USInt	33
10	funcGrpNum	UInt	500

Figure 3-18 Screen switch data block

Note

The value 33 entered for "screenNum" parameter is value of manual operation screen obtained from "LAHP_ScreenName" text list.

Result

In instance 1, you can view manual operation screen which belongs to function group 100.

In instance 2, you can view manual operation screen which belongs to function group 500.

3.7 Shift planning

3.7.1 Overview

Shift planning block is used to get a detailed report of the number of workpieces produced during different shifts in a day.

A shift is a scheduled period, during which the workpieces are set to production against a set target. These workpieces are further divided into good parts and bad parts following the quality assessment.

The "LAHP_ShiftPlanning" runtime screen is shown below:

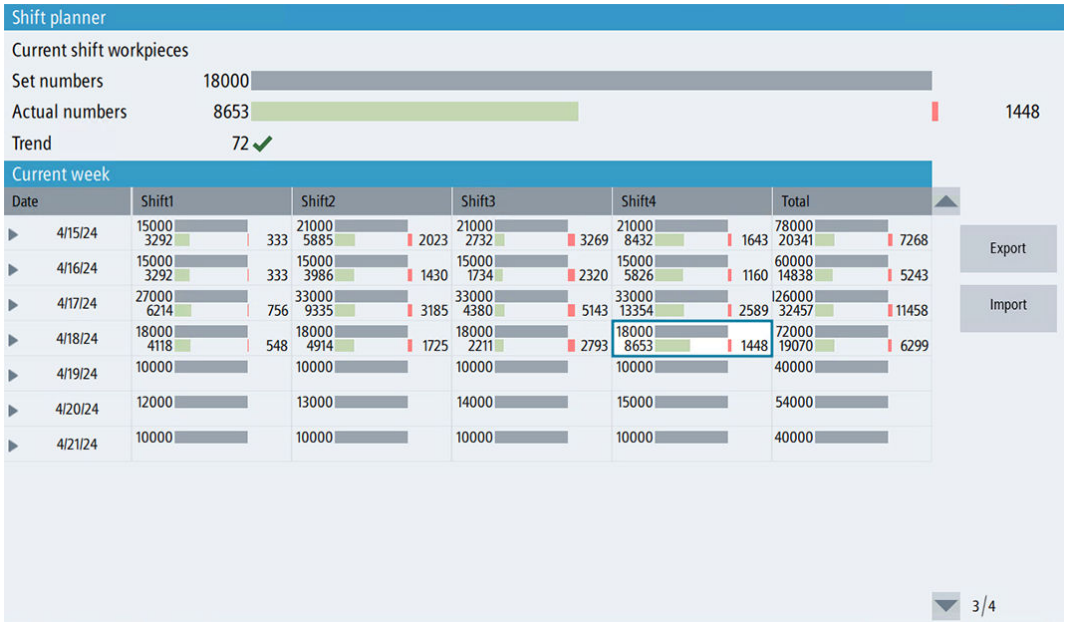


Figure 3-19 Shift Planner runtime screen

You need to configure the following for "LAHP_ShiftPlanning" screen to work as intended:

- Block configuration (Page 45) in Step 7.
  For more details on configuring the block, refer AUTOHOTSPOT.
  HMI tag configuration in WinCC Unified.

3.7.2 Block interface - LAPP_ShiftPlanning

This block is used for data handling and configurations.

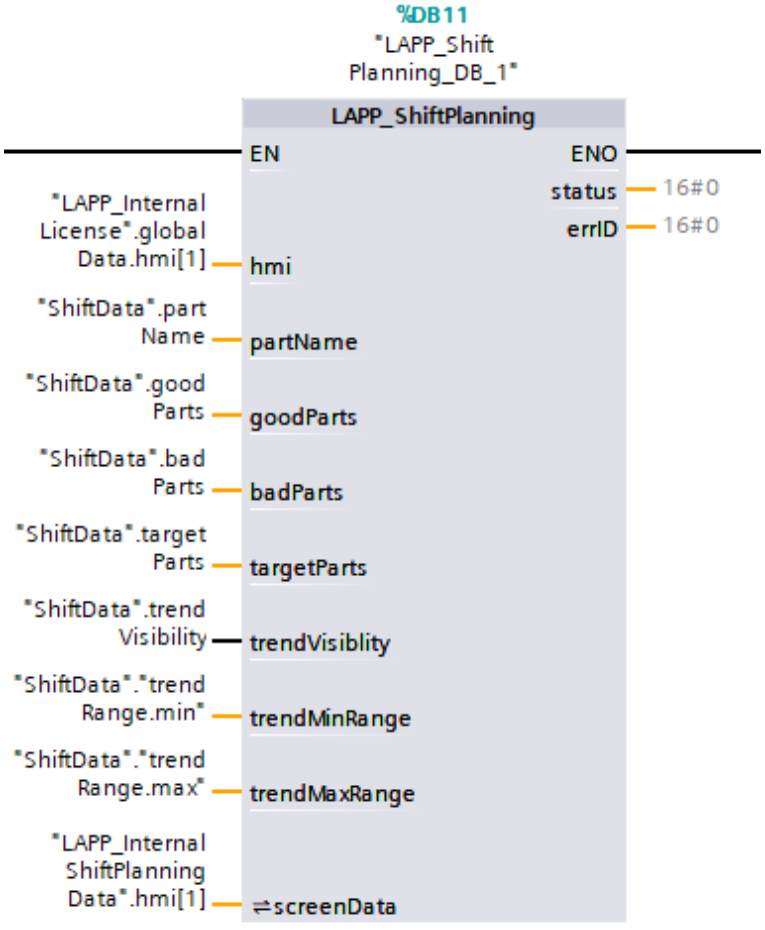


Figure 3-20 "LAPP_ShiftPlanning"

Parameters

Table 3-4 Parameters of "LAPP_ShiftPlanning"

Name	Declaration	Type	Description
hmi	Input	LAPP_typeIntl-License	Configure the HMI Index which is defined while setting up the HMI device. **Note**: HMI index value cannot be 0. It should match with HMI index value used in License DB ("LAPP_InternalLi-cense").
partName	Input	WString[16]	Actual part being manufactured. **Note**: The part name entered should match the part name entered in the csv file with the list of parts.
goodParts	Input	UInt	Actual number of good parts produced
badParts	Input	UInt	Actual number of bad parts produced

Name	Declaration	Type	Description
targetParts	Input	UInt	Actual number of parts to be produced
trendVisibility	Input	Bool	Visibility of "Trend" on HMI: • True = Visible • False = Invisible
trendMinRange	Input	USInt	Minimum threshold value for trend.
trendMaxRange	Input	USInt	Maximum threshold value for trend.
screenData	InOut	LAPP_type-IntlHmiShift-Planning	Interface the HMI tag from "LAPP_InternalShiftPlanning-Data.hmi[y]". **Note**: "LAPP_InternalShiftPlanningData" is an internal DB. Modify the array value to increase the HMI device support. For more information, refer to Configuring PLC blocks for different HMI.
status	Output	Word	Displays the status codes of the block. For more information refer to Status codes (Page 47).
errID	Output	Word	Displays the error codes of the block. For more information, refer to "Information System > Programming a PLC > Instructions > Instructions (S7-1200, S7-1500) > Basic instructions (S7-1200, S7-1500) > Program control operations (S7-1500) > Runtime control (S7-1500) > GET_ERR_ID" instruction in TIA portal online help.

Note

- All the actual data must be programmed by OEM.
- If you want to overwrite the "targetParts" from the PLC, then you must enable the parameter "overWritePartsTarget" in the "LAPP_InternalShiftPlanningData" block.
- To avoid data loss in case of PLC reaching stop mode or power failure, ensure to enable the "Retain" tag for "hmi" in "LAPP_InternalShiftPlanningData" block.
- Ensure that the PLC and the HMI are in the same time zone.
- You can use "LAPP_blPartCounter" business logic block along with "LAPP_Shiftplanning" block. For more information, refer "LAPP_blPartCnt (Page 199)".

3.7.3 Additional information

Soft limit on parts

A maximum of 160 parts can be configured per day. The OEM can set a soft limit on the target parts.

Follow the steps below to set a soft limit on the target parts:

1. In TIA portal, navigate to "PLC > Program blocks > CMHA blocks > ShiftPlanning" and open the "LAPP_InternalShiftPlanningData" block.
2. In the block parameters expand "hmi" and provide the soft limit value to the target parts in "maxPartsSoftLimit". For example, if the value given is 50 then maximum parts that can be configured per day will be 50.

Auto-Export

Previous two weeks data is automatically exported to the path mentioned in the start value of "LAHP_shiftAutoExportPath".

Follow the steps below to ensure the data is automatically exported:

1. In TIA portal, navigate to "PLC > Program blocks > CMHA Data blocks" and open the "LAPP_InternalShiftPlanningData" block.
2. In the block parameters expand "autoExport" and set the start value of "enable" to true for the automatic export of previous two weeks data.
3. Based on the requirement for the frequency of export set the value for "freqOfWeek".
   For example,
   If you set the value of "freqOfWeek" to 2, then the data is automatically exported once in two weeks.
   If you set the value of "freqOfWeek" to 1, then the data is automatically exported once a week.

3.7.4 Status codes

List

The status codes for Shift Planner:

Status code	Description
16#7000	Block is in initialized state.
16#7002	Block is in execution state and has no errors.
16#8200	Faulty license.
16#8201	Invalid HMI index.
16#8202	The input for the parameter "partName" = Null.
16#8203	Maximum number of parts per day reached.
16#8204	The file path for "List of parts" CSV mentioned in HMI tag "ShiftBackupPath" is wrong.
16#8205	The maximum part soft limit value is greater than 160.
16#8400	No connection to HMI.

3.8 Workpiece manager

3.8.1 Workpiece manager for PLC

3.8.1.1 Overview

Workpiece manager helps you in managing workpieces in PLC machines. It is used to tune the recipe and record the actual time taken to complete manufacturing of a part, so that the same time can be used in production to manage the quality and the workpieces produced.

It is also used for monitoring the state of workpiece under production.

Note

- You can import or export data for workpiece manager using the CSV file.
- In WinCC unified recipe management is possible through parameter control (in Wincc Advanced it is termed as recipe control).
- In a particular station, you can execute only one recipe at a time.

The "LAHP_WpieceManager_PLC" runtime screen is shown below:

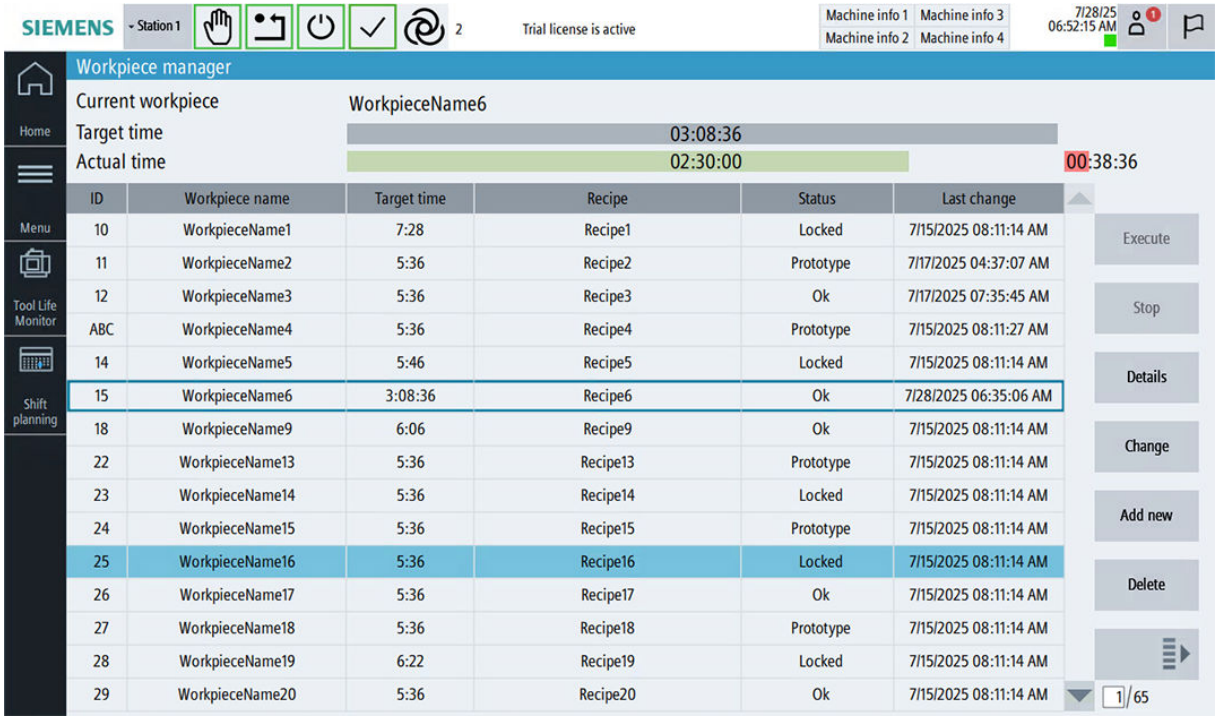


Figure 3-21 Workpiece manager screen for PLC

You need to configure the following for "LAHP_WpieceManager" screen to work as intended:

- Block configuration (Page 49) in Step 7
  For more details on configuring the block, refer AUTOHOTSPOT.
- Text list configuration (Page 49) in WinCC Unified.

3.8.1.2 Block interface - LAPP_WorkpieceManager_PLC

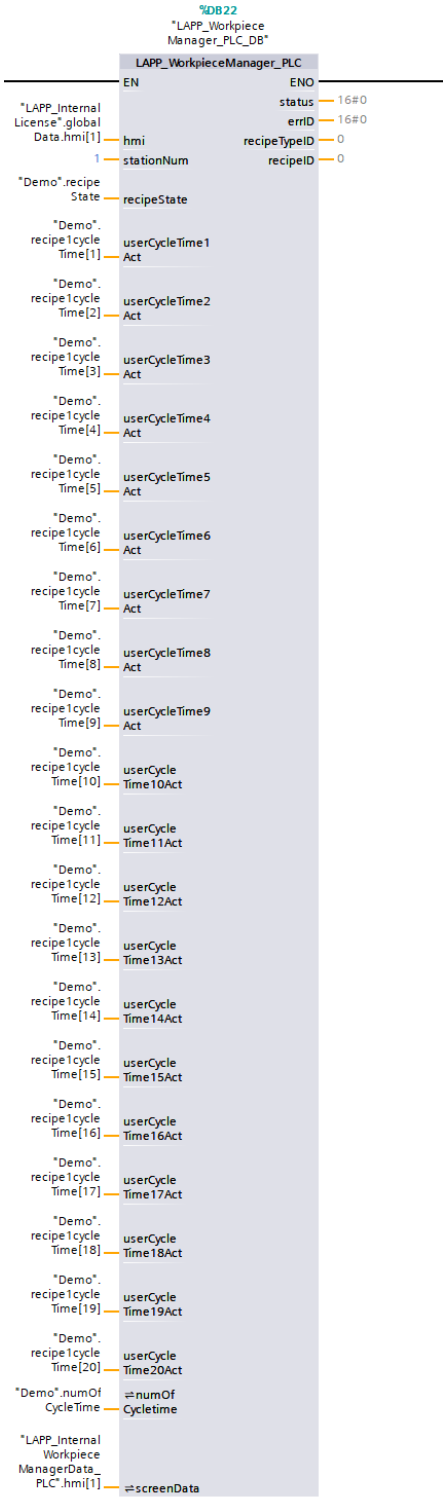


Figure 3-22 "LAPP_WorkpieceManager_PLC"

Parameters

Table 3-5 Parameters of "LAPP_WorkpieceManager_PLC"

Name	Declaration	Type	Description
hmi	Input	LAPP_typeIntlLicense	Configure the HMI Index which is defined while setting up the HMI device. **Note**: HMI index value cannot be 0. It should match with HMI index value used in License DB ("LAPP_InternalLicense").
stationNum	Input	USInt	Active station number. **Note:** Only one recipe can be executed at a time in each station. Maximum 10 stations can be configured. For more information on configuring the station refer to Configuring station data (Page 35).
recipeState	Input	USInt	Status of recipe: • 1 = Running • 2 = Stopped
userCycleTime1Act	Input	LTime	Mandatory columns for OEM-specific times for "userCycleTime1Act" to "userCycleTime5Act". **Note**: For more information refer to, Configuring text lists.
userCycleTime2Act	Input	LTime	
userCycleTime3Act	Input	LTime	
userCycleTime4Act	Input	LTime	
userCycleTime5Act	Input	LTime	
userCycleTime6Act	Input	LTime	Optional column for the OEM- specific times for "userCycleTime6Act" to "userCycleTime15Act".
userCycleTime7Act			
userCycleTime8Act			
userCycleTime9Act			
userCycleTime10Act			
userCycleTime11Act			
userCycleTime12Act			
userCycleTime13Act			
userCycleTime14Act			
userCycleTime15Act			
userCycleTime16Act			
userCycleTime17Act			
userCycleTime18Act			
userCycleTime19Act			
userCycleTime20Act			
numOfCycletime	InOut	USInt	Maximum number of cycle time for all recipe types. **Note**: The value depends on the number of "userCycleTimeAct" configured.
screenData	InOut	LAPP_typeIntlHmiRecipeManager	Interface the HMI tag from "LAPP_InternalWorkpieceManagerData_PLC.hmi[y]". **Note**: "LAPP_InternalWorkpieceManagerData_PLC" is an internal DB. Modify the array value to increase the HMI device support. For more information, refer to Configuring PLC blocks for different HMI.

Name	Declaration	Type	Description
status	Output	Word	Displays the status codes of the block. For more information refer to Status codes (Page 51).
errID	Output	Word	Displays the error codes of the block. For more information, refer to "Information System > Programming a PLC > Instructions > Instructions (S7-1200, S7-1500) > Basic instructions (S7-1200, S7-1500) > Program control operations (S7-1500) > Runtime control (S7-1500) > GET_ERR_ID" instruction in TIA portal online help.
recipeTypeID	Output	USInt	Selected recipe type ID.
recipeID	Output	UInt	Selected recipe ID.

Note

In WinCC Unified recipe management is possible through parameter control (in Wincc Advanced it is termed as recipe control). The "Recipe ID" indicates the parameter set and the "Recipe type ID" indicates the parameter set control.

3.8.1.3 Status codes

List

The status codes for Workpiece Manager:

Status code	Description
16#7000	Block is in initialized state.
16#7002	Block is in execution state and has no errors.
16#8200	Faulty license.
16#8201	Invalid HMI index.
16#8202	Recipe running state is invalid or recipe is not in execution.
16#8203	Incorrect input recipe type.
16#8204	Passed running recipe state is invalid.
16#8205	Selected station number not in range. (minimum = 1, maximum = 10)
16#8400	No connection to HMI.

3.8.2 Workpiece manager for CNC

3.8.2.1 Overview

Workpiece Manager helps you in managing workpieces. It is used to tune the part program created and record the actual time taken to complete manufacturing of a part, so that the same time can be used in production to manage the quality and the workpieces produced.

It is also used for monitoring the state of workpiece under production.

Note

- OPC UA communication is mandatory for the workpiece manager to work as intended. For more information, refer OPC UA communications.
- You can import or export data for workpiece manager using the CSV file.

The "LAHP_WpieceManager_CNC" runtime screen for SINUMERIK ONE is shown below:

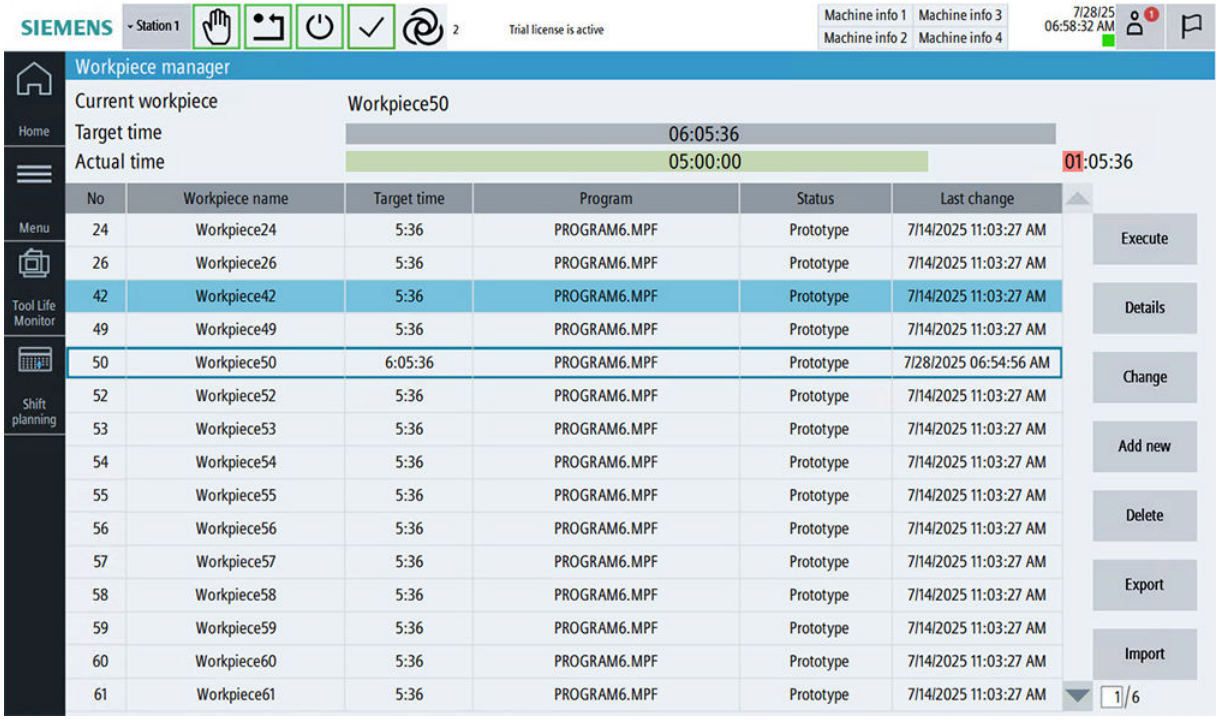


Figure 3-23 Workpiece manager screen for SINUMERIK ONE

You need to configure the following for "LAHP_WpieceManager_CNC" screen to work as intended:

- Block configuration (Page 53) in Step 7
  For more details on configuring the block, refer AUTOHOTSPOT.
- Text list configuration in WinCC Unified.

3.8.2.2 Block interface - LAPP_WorkpieceManager_CNC

This block is used for data handling and configurations.

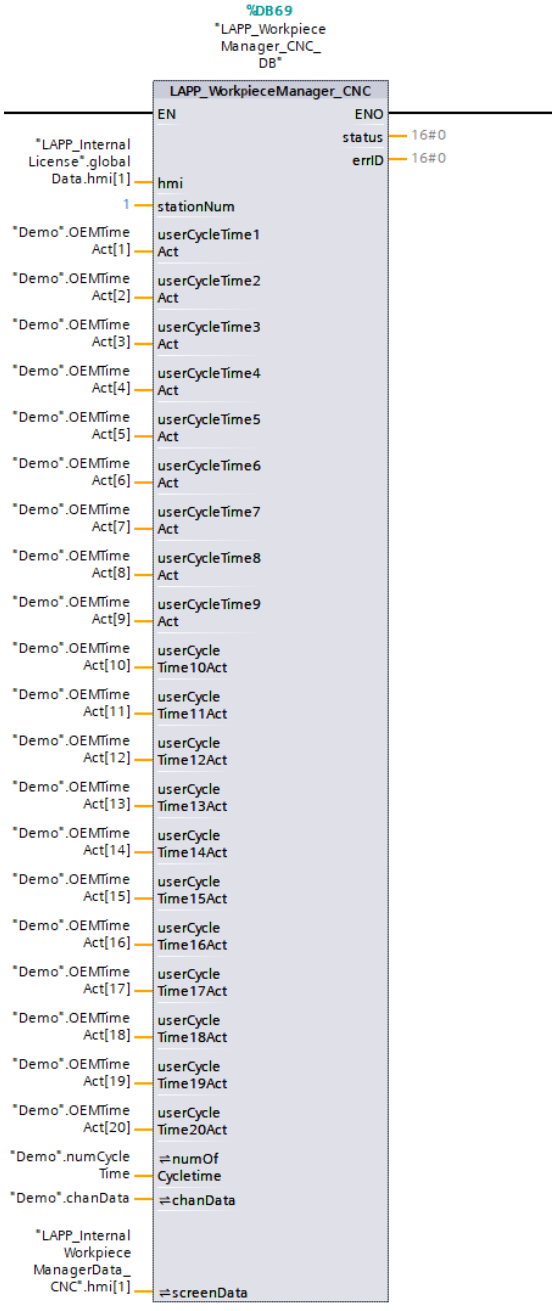


Figure 3-24 "LAPP_WorkpieceManager_CNC"

Parameters

Table 3-6 Parameters of "LAPP_WorkpieceManager_CNC"

Name	Declaration	Type	Description
hmi	Input	LAPP_typeIntlLicense	Configure the HMI Index which is defined while setting up the HMI device. **Note**: HMI index value cannot be 0. It should match with HMI index value used in License DB ("LAPP_InternalLicense").
stationNum	Input	USInt	Active station number.
userCycleTime1Act	Input	LTime	Mandatory columns for OEM-specific times for "userCycleTime1Act" to "userCycleTime5Act".
userCycleTime2Act	Input	LTime	
userCycleTime3Act	Input	LTime	
userCycleTime4Act	Input	LTime	
userCycleTime5Act	Input	LTime	
userCycleTime6Act	Input	LTime	Optional column for the OEM- specific times for "userCycleTime6Act" to "userCycleTime15Act".
userCycleTime7Act			
userCycleTime8Act			
userCycleTime9Act			
userCycleTime10Act			
userCycleTime11Act			
userCycleTime12Act			
userCycleTime13Act			
userCycleTime14Act			
userCycleTime15Act			
userCycleTime16Act			
userCycleTime17Act			
userCycleTime18Act			
userCycleTime19Act			
userCycleTime20Act			
numOfCycletime	InOut	USInt	Maximum number of cycle time for all workpiece types. **Note**: The value depends on the number of "userCycleTimeAct" configured.
chanData	InOut	NCK_typeChan	Current active channel displayed in the SINUMERIK Operate to select the part program. For more information, refer to Configuring chanData (Page 55).
screenData	InOut	LAPP_typeHmiWpieceManager	Interface the HMI tag from "LAPP_InternalWorkpieceManagerData.hmi[y]". **Note**: "LAPP_WorkpieceManagerData" is an internal DB. Modify the array value to increase the HMI device support. For more information, refer to Configuring PLC blocks for different HMI.

Name	Declaration	Type	Description
status	Output	Word	Displays the status codes of the block. For more information refer to Status codes (Page 56).
errID	Output	Word	Displays the error codes of the block. For more information, refer to "Information System > Programming a PLC > Instructions > Instructions (S7-1200, S7-1500) > Basic instructions (S7-1200, S7-1500) > Program control operations (S7-1500) > Runtime control (S7-1500) > GET_ERR_ID" instruction in TIA portal online help.

Note

- OEM must program and interface all the parameters related to the actual time with the block.
- A maximum of 10 channels can be configured.
- You can use "LAPP_blCycleTime" business logic block along with "LAPP_WorkpieceManager" block. For more information, refer "LAPP_blCycleTime (Page 202)".

3.8.2.3 Configuring chanData

Procedure

1. In Startup OB configure the register block "SINU_RegisterChan". For more information on configuring the block to register the channel, refer to "Information system > Configuring and programming SINUMERIK > Programming the SINUMERIK PLC > Programming SINUMERIK PLC (ONE) > Reference to the PLC Basic Program Plus > PLC instructions SINU_* > Registration > SINU_RegisterChan" in TIA Portal online help.
2. In Main OB configure the re-register block "SINU_ReRegisterChan". For more information on configuring the block to re-register the channel, refer to ""Information system > Configuring and programming SINUMERIK > Programming the SINUMERIK PLC > Programming SINUMERIK PLC (ONE) > Reference to the PLC Basic Program Plus > PLC instructions SINU_* > Registration > SINU_ReRegisterChan" in TIA Portal online help.
3. Give the same input given to the "chan" parameter of the "SINU_RegisterChan" and "SINU_ReRegisterChan" blocks to the "chanData" parameter of "LAPP_workPieceManager" block.

Result

The "chanData" parameter is configured.

3.8.2.4 Status codes

List

The status codes for Workpiece Manager:

Status code	Description
16#7000	Block is in initialized state.
16#7002	Block is in execution state and has no errors.
16#7003	Function is not executed.
16#7004	Waiting for free interface to the NCK.
16#7005	Task sent to NCK, waiting for response.
16#7006	Task sent to NCK, waiting for completion.
16#8200	Faulty license.
16#8201	Invalid HMI index.
16#8202	The number of columns is not within the range (minimum 5 and maximum 20).
16#8203	The input program name does not match with the running program name.
16#8204	Incorrect channel number.
16#8205	Channel state is not "Ready" or "Reset".
16#8400	No connection to HMI.
16#8401	PI Service - channel instance is not registered.
16#8402	PI Service - new job started before the completion of active job.
16#8403	PI Service - parameter too long.
16#8404	PI Service - program not found.
16#8405	PI Service - parameter outside the valid range.

3.8.3 Additional widget for workpiece data

An additional widget workpiece manager can be configured in the OEM defined runtime screens. The widget displays the name, target time and actual time of the running workpiece.

Note

The workpiece manager widget is available with both PLC and SINUMERIK ONE libraries.

Procedure to create data block

Follow the steps below to configure the workpiece manager widget.

1. Right-click "Program blocks" and click "Add new block"

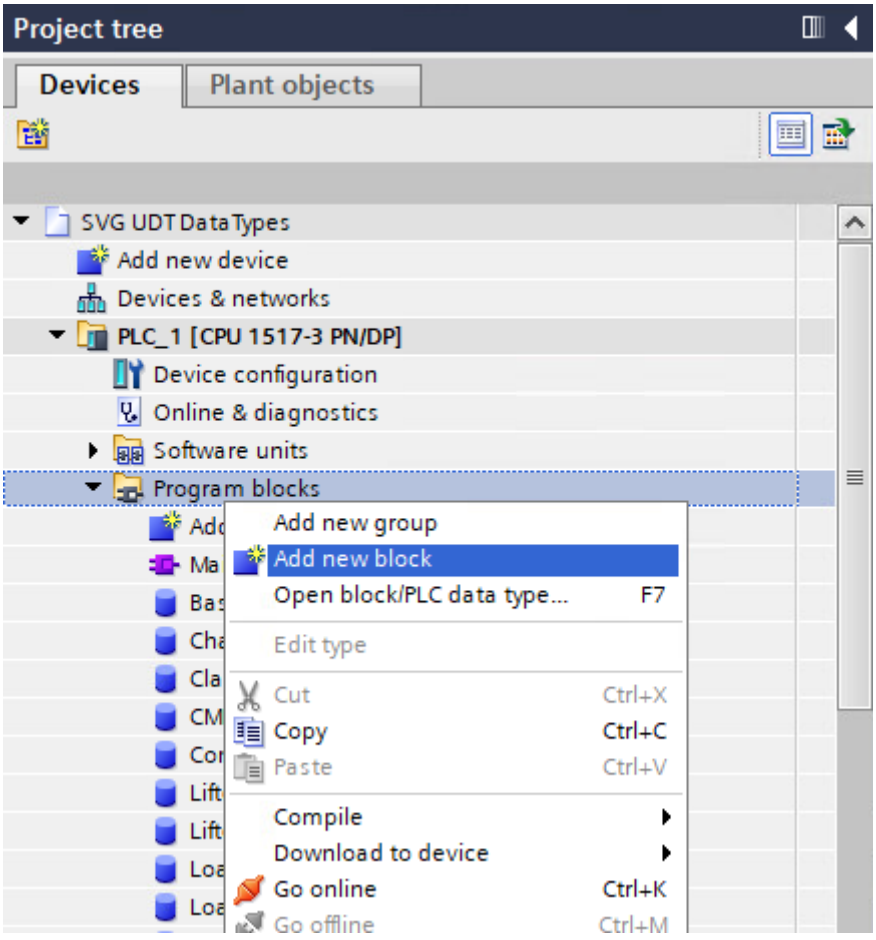


Figure 3-25 Add new block

"Add new block" window opens.

2. Click "Data block".
3. In "Type" field, select "LAPP_typeWidgetWpieceManager" to configure the workpiece details.
4. Click "OK".

Data type

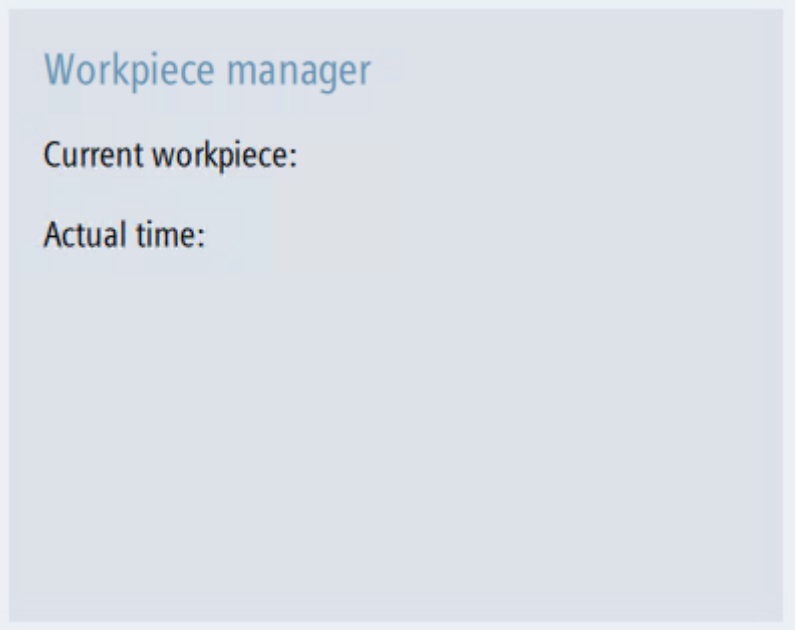
The parameters of the widget are:

Table 3-7 Structure of data type "LAPP_typeWidgetWpieceManager"

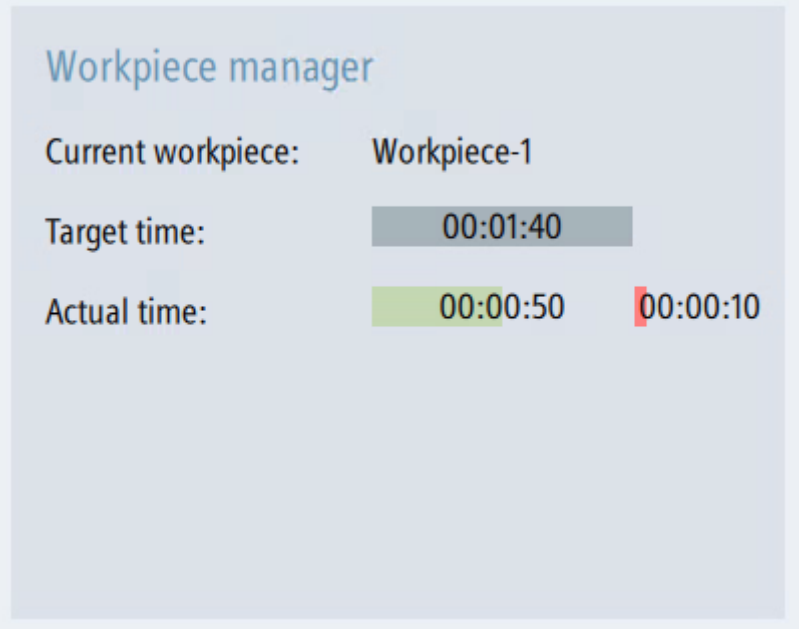
Name	Data type	Description
wPieceName	WString[24]	Name of the current workpiece in production
targetTime	LTime	Target time set for the current workpiece
actTime	LTime	Actual machining time of the current workpiece
residualTime	LTime	Remaining machining time of the current workpiece

The widget appears as shown below

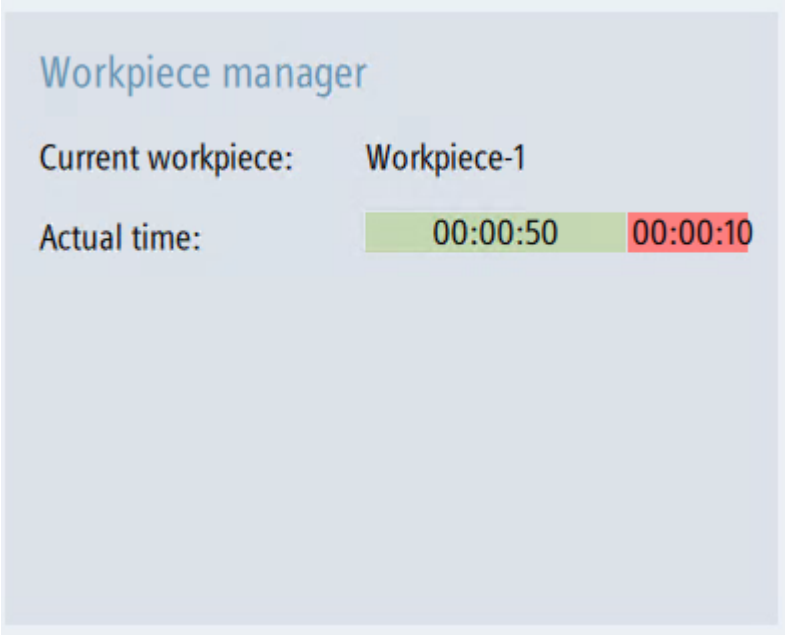
- If no input is provided:



- If all the inputs (workpiece name, target, actual and residual time) are provided:



- If the target time is not provided:



3.9 Manual operation

3.9.1 Manual Operation as a screen

3.9.1.1 Overview

Manual operation that can be handled in automotive manufacturing for production assembly lines and can be programmed using Manual Operation block. For example, you can program power-up conditions, drive movements, handling nut runners, handling cycles etc.

The below image gives the workflow details of manual operation.

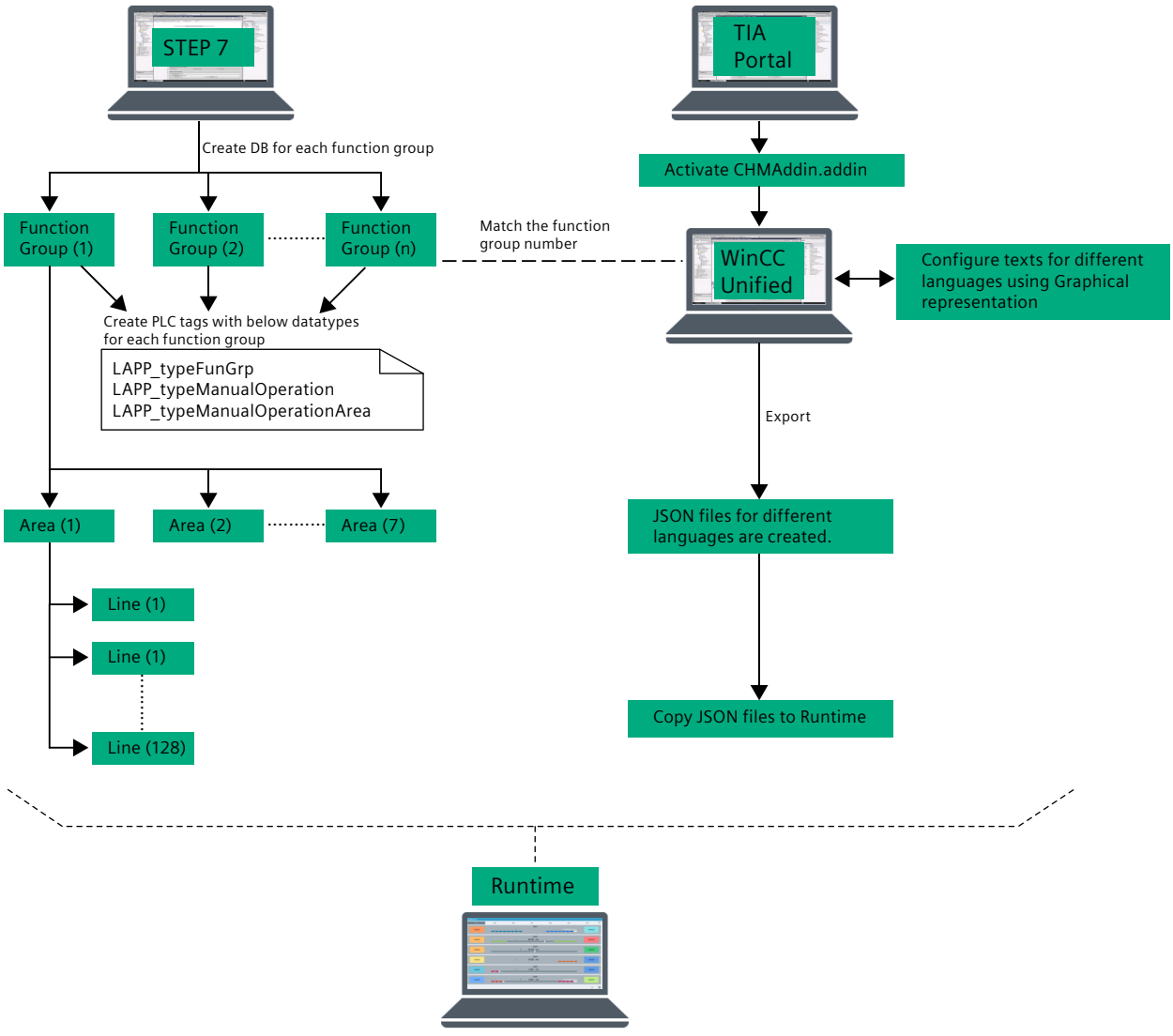


Figure 3-26 Manual Operation configuration workflow

Note

Maximum of 7 areas can be configured in individual function group (for each type of functions like nutrunner, cycles, powerup and so on).

Based on the requirement, you can create individual function groups for each program. You can group the manual operation lines into seven different areas under each function group.

You need to define the number of lines (Page 75) required for each function group. You can group the lines into different areas which can be represented and operated using the screens during runtime.

The following data types (Page 73) are used to configure manual operation block:

- "LAPP_typeFuncGrp"
- "LAPP_typeManualOperationArea"
- "LAPP_typeManualOperation"

You need to configure the following for manual operation screen to work as intended:

- Block configuration in Step 7
  - "LAPP_manualOperationMain" - configure HMI index.
  - "LAPP_funcGrpConfig (Page 66)" - configure the number of lines, areas, and the HMI index required for the function group.
  - "LAPP_manualOperation (Page 67)" - configure the mode of operation and the manual operation line parameters.

  For more details on configuring the blocks, refer to AUTOHOTSPOT.
- Text list and screen configuration in WinCC Unified - activate the "CHMAddin" add-in, configure the line properties.

In runtime, you can view the manual operation screen as shown:

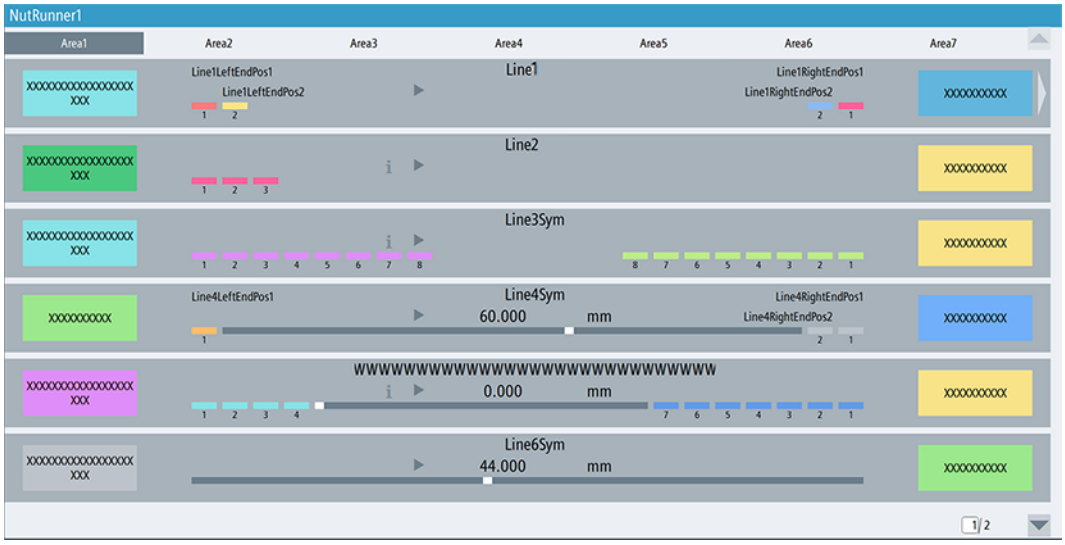


Figure 3-27 Manual operation screen

Note

All the buttons, end positions, position indicator and actor set is configurable, you can adapt the manual operation as per the requirement and configure the lines for powerup, movement, setup, cycle times, nut runners and so on.

For more information on runtime behavior, refer to "Configuring Create MyHMI /Automotive > Runtime screens > Manual operation" in TIA Portal online help.

3.9.1.2 Manual operation as a screen

For illustration purpose, the following PLC tag names are used:

- functionGrp
- Areas
- Lines

Procedure

1. Follow the steps from Configuring PLC blocks to configure the mandatory block and create DB. For example, for first function group create a DB "NutRunner1"

   Note

   You can create different data block for different function groups.

2. In the DB, create 3 tags with data types "LAPP_typeFunGrp", "LAPP_typeManualOperationArea", and "LAPP_typeManualOperation".
   For example, In "NutRunner1" DB, create the tags as shown:

   NutRunner1

	Name	Data type
1	Static	
2	FunctionGrp	"LAPP_typeFuncGrp"
3	Areas	Array[1..5] of "LAPP_typeManualOperationArea"
4	Lines	Array[1..100] of "LAPP_typeManualOperation"

   Figure 3-28 NutRunner1 data block

   Note

   Ensure to start array value from 1.

3. In "NutRunner1.Lines", define the number of lines required for this function group by modifying the array value. For more information, refer to Examples (Page 75).

   Note

   OEM can decide the number of lines required based on the type of the S7 1500 PLC.

4. In "NutRunner1.FunctionGrp.funcGrpNum", enter the function group ID. For example, 100 as shown below:

NutRunner1

	Name	Data type	Start value
1	Static		
2	FunctionGrp	"LAPP_typeFuncGrp"	
3	funcGrpNum	UInt	100
4	actvArea	UInt	0
5	pages	"LAPP_typePageDetails"	
6	Areas	Array[1..5] of "LAPP_typeManualOperationArea"	
7	Lines	Array[1..100] of "LAPP_typeManualOperation"	

5. In "NutRunner1.Areas", define the number of areas required for this function group by modifying the array value. For more information, refer to Examples (Page 75).
6. In "NutRunner1.Areas[x].displayedLines", enter the line numbers to be displayed as shown for each area:
x - line number

	Name	Data type	Start value
1	Static		
2	funcGrp	"LAPP_typeFuncGrp"	
3	areas	Array[1..7] of "LAPP_typeManualOper...	
4	areas[1]	"LAPP_typeManualOperationArea"	
5	areaNum	UInt	101
6	displayedLines	String	'1-3,0,4-10,3-1;'
7	hmiPage	"LAPP_typePageDetails"	
8	areas[2]	"LAPP_typeManualOperationArea"	
9	areaNum	UInt	102
10	displayedLines	String	'18-60,100-78,44;'
11	hmiPage	"LAPP_typePageDetails"	
12	areas[3]	"LAPP_typeManualOperationArea"	

Figure 3-29 NutRunner1 data block

Note

- Ensure to enter the value within single quotation mark and end it with semicolon -> '...;'
- You can enter multiple line numbers by segregating them using comma -> '1,10'
- You can enter the range of line numbers by using hyphen -> '1-10'
- You can enter the lines to be displayed in any order as mentioned below:
  - incremental range of line values -> '1 to 10'
  - decremental range of line values -> '100-7'
  - single line number -> 44
  - repeated line number -> '1-10, 3, 1;'
  - Separator - > 0
- Ensure to follow the rules to enter the line numbers for the block to work as intended. Else the PLC can go into error mode.

7. In "Instructions" task card, navigate to "Optional packages > Create MyHMI /Automotive" and drag the "LAPP_ManualOperationMain" block to the cyclic block or main block. Interface the input and output parameters.

Note

You need to call this mandatory block only once in cyclic block/ main block.

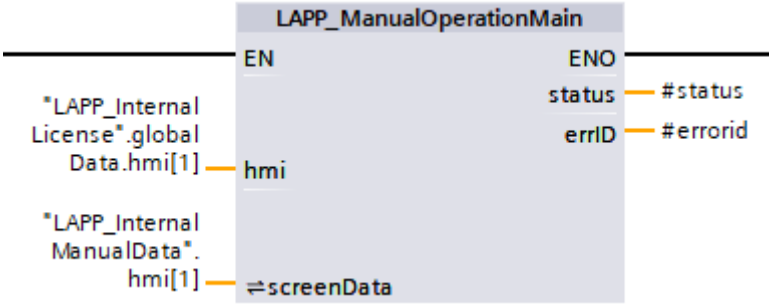


Figure 3-30 "LAPP_ManualOperationMain"

Name	Declaration	Type	Description
hmi	Input	LAPP_typeIntlLicense	Configure the HMI Index which is defined while setting up the HMI device. **Note**: HMI index value cannot be 0. It should match with HMI index value used in License DB ("LAPP_InternalLicense").
screenData	InOut	LAPP_typeIntl-FuncGrpHmiDisp	Interface the HMI tag from "LAPP_Internal-ManualData.hmi[y]". **Note**: "LAPP_InternalManualData" is an internal DB. Modify the array value to increase the HMI device support. For more information, refer to Configuring PLC blocks for different HMI.
status	Output	Word	Displays the status of the block. For more information, refer to Status codes (Page 80).
errID	Output	Word	Displays the error codes of the block. For more information, refer to "Information System > Programming a PLC > Instructions > Instructions (S7-1200, S7-1500) > Basic instructions (S7-1200, S7-1500) > Program control operations (S7-1500) > Runtime control (S7-1500) > GET_ERR_ID" instruction in TIA portal online help.

y- indicates HMI device number.

8. In "Instructions" task card, navigate to "Optional packages > Create MyHMI /Automotive" and drag the "LAPP_FuncGrpConfig" block to the cyclic block or main block.
9. Interface the input and output parameters for the "LAPP_FuncGrpConfig" block. For more information, refer to Block interface - LAPP_FuncGrpConfig (Page 66).

10. In "Instructions" task card, navigate to "Optional packages > Create MyHMI /Automotive" and drag the "LAPP_ManualOperation" block to the cyclic block or main block.

Note

- Refer to the subsequent topics to know more about the types of Manual Operation blocks in this section.
- It is mandatory to configure the manual operation lines using the types of Manual Operation blocks to view the line in runtime.

11. Interface the input and output parameter to the added block interface. For more information, refer to Block interface - LAPP_ManualOperation (Page 67).
12. Configure the text list and screen in WinCC Unified. For more information, refer to Configuring LAHP_ManualOperation (MCP/Touch).

3.9.1.3 Block interface - LAPP_FuncGrpConfig

Block interface

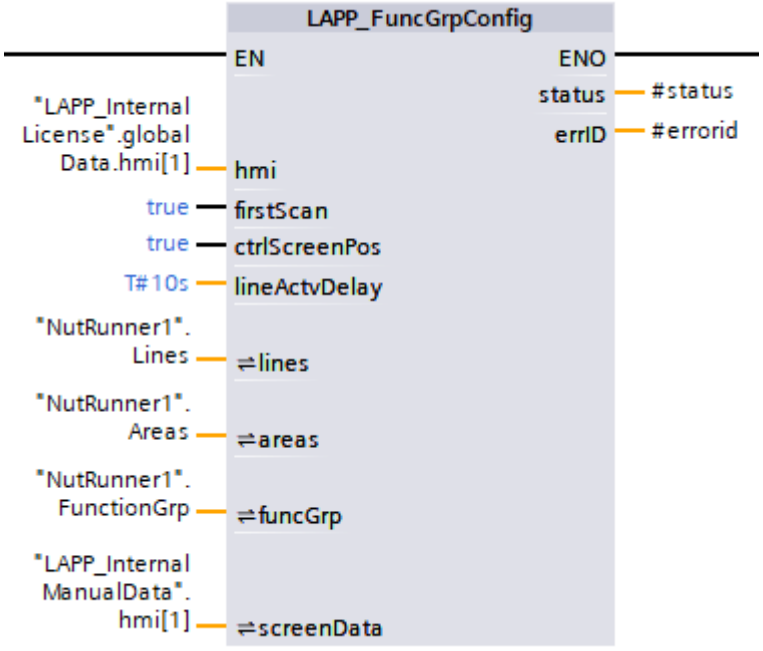


Figure 3-31 "LAPP_FuncGrpConfig"

Table 3-8 Parameters of "LAPP_FuncGrpConfig"

Name	Declaration	Type	Description
hmi	Input	LAPP_typeIntlLicense	Configure the HMI Index which is defined while setting up the HMI device. **Note**: HMI index value cannot be 0. It should match with HMI index value used in License DB ("LAPP_InternalLicense").
firstScan	Input	Bool	Startup bit.
ctrlScreenPos	Input	Bool	Determines the position of the control screen for all the lines in the function group: • 0 - displays on top of the line • 1 - displays below the line
lineActvDelay	Input	Time	Set the line active delay for the function group.
lines	InOut	Array of LAPP_typeManualOperation	Interface the total number of lines in the function group.
areas	InOut	Array of LAPP_typeManualOperationArea	Interface the total number of areas in the function group.
funcGrp	InOut	LAPP_typeFuncGrp	Interface the function group ID.
screenData	InOut	LAPP_typeIntlFuncGrpHmiDisp	Interface the HMI tag from "LAPP_InternalManualData.hmi[y]". **Note**: "LAPP_InternalManualData" is an internal DB. Modify the array value to increase the HMI device support. For more information, refer to Configuring PLC blocks for different HMI.
status	Output	Word	Displays the status codes of the block. For more information, refer to Status codes (Page 80).
errID	Output	Word	Displays the error codes of the block. For more information, refer to "Information System > Programming a PLC > Instructions > Instructions (S7-1200, S7-1500) > Basic instructions (S7-1200, S7-1500) > Program control operations (S7-1500) > Runtime control (S7-1500) > GET_ERR_ID" instruction in TIA portal online help.

y- indicates HMI device number.

Note

Ensure to interface the same HMI device number at "hmiIndex" and "screenData" value.

3.9.1.4 Block interface - LAPP_ManualOperation

"LAPP_ManualOperation" block is used to configure Touch Operated and MCP mode for Manual Operation.

In case of touch operated mode, you can move the device along the left (negative) and right (positive) direction of the axes using buttons.

The actual data comes from field and the touch operation signals are sent as feedback to PLC. You can also edit the text on these buttons. For more information, refer to Editing text.

In case of MCP operated mode. The actual data goes as output field to WinCC and the MCP signals comes as input to HMI.

Block details

This block is used for data handling and configurations.



Figure 3-32 "LAPP_ManualOperation"

Table 3-9 Parameters of "LAPP_ManualOperation"

Name	Declaration	Type	Description
hmi	Input	LAPP_typeIntlLicense	Configure the HMI Index which is defined while setting up the HMI device. **Note**: HMI index value cannot be 0. It should match with HMI index value used in License DB ("LAPP_InternalLicense").
mode	Input	USInt	Determines the operating mode of the line configured: • 1 - Touch operated mode • 2 - MCP operated mode
showLine	Input	Bool	Determines the line visibility in runtime: • 0 - Invisible • 1 - Visible For more information, refer to Example (Page 76). **Note**: The "showLine" parameter works only for the lines configured in screen.
enableMechAddr	Input	Bool	Enables mechanical addressing mode on the screen: • 0 - Disable • 1 - Enable **Note**: The mechanical addressing mode is optional. If enabled, then OEM needs to configure the mechanical address also. If not enabled, "Mech.x.x" is displayed if mechanical addressing is selected in runtime screen. You can modify the text in "LAHP_ManualMechanicalText" text list.
expandable	Input	Bool	Determines the visibility of "▶" icon on the line to view the expand area: • 0 - Invisible • 1 - Visible
disableLineSel	Input	Bool	Determines the line selection possibility to operate the commands: • 0 - Enables line selection • 1 - Disables line selection **Note**: If line selection is disabled, then "enableBigArea" is also disabled.
enableBigArea	Input	Bool	Enables the operating area of the line to whole screen for 5 seconds: • 0 - Disable • 1 - Enable For more information, refer to Example (Page 77).
execLeftCmd	Input	Bool	Status of left command execution. **Note**: • Only available for MCP mode. Set the value to "false" for touch operation mode. • If "disableLineSel" is set to 0, then select the line to execute the left command. • If "disableLineSel" is set to 1, then without line selection left command can be executed.

Name	Declaration	Type	Description
execRightCmd	Input	Bool	Status of right command execution. **Note**: • Only available for MCP mode. Set the value to "false" for touch operation mode. • If "disableLineSel" is set to 0, then select the line to execute the right command. • If "disableLineSel" is set to 1, then without line selection right command can be executed.
cnfrmExecLeft	Input	Bool	Confirms executability for the left button: • 0 - not executable • 1 - executable
cnfrmExecRight	Input	Bool	Confirms executability for the right button: • 0 - not executable • 1 - executable
disableLeft	Input	Bool	Disables left button. For example: Enable the button when the "cnfrmExecLeft" is executable.
disableRight	Input	Bool	Disables right button. For example: Enable the button when the "cnfrmExecRight" is executable.
leftBtnVis	Input	Bool	Determines the visibility of the left button: • 0 - Invisible • 1 - Visible
rightBtnVis	Input	Bool	Determines the visibility of the right button: • 0 - Invisible • 1 - Visible
cnfrmExecLeftCmd	Input	Bool	Confirms the execution of "execLeftCmd": • 0 - not confirmed • 1 - confirmed
cnfrmExecRightCmd	Input	Bool	Confirms the execution of "execRightCmd": • 0 - not confirmed • 1 - confirmed
fieldDevAckLeft	Input	Bool	Acknowledgement from field device for left command: • 0 - not acknowledged • 1 - acknowledged
fieldDevAckRight	Input	Bool	Acknowledgement from field device for right command: • 0 - not acknowledged • 1 - acknowledged
leftEndPosSens1	Input	Bool	Sensor input for left position 1 (field input) • 0 - not sensed • 1 - sensed

Name	Declaration	Type	Description
leftEndPosSens2	Input	Bool	Sensor input for left position 2 (field input): • 0 - not sensed • 1 - sensed
leftEndPosSens3	Input	Bool	Sensor input for left position 3 (field input): • 0 - not sensed • 1 - sensed
leftEndPosSens4	Input	Bool	Sensor input for left position 4 (field input): • 0 - not sensed • 1 - sensed
leftEndPosSens5	Input	Bool	Sensor input for left position 5 (field input) • 0 - not sensed • 1 - sensed
leftEndPosSens6	Input	Bool	Sensor input for left position 6 (field input) • 0 - not sensed • 1 - sensed
leftEndPosSens7	Input	Bool	Sensor input for left position 7 (field input): • 0 - not sensed • 1 - sensed
leftEndPosSens8	Input	Bool	Sensor input for left position 8 (field input) • 0 - not sensed • 1 - sensed
rightEndPosSens1	Input	Bool	Sensor input for right position 1 (field input) • 0 - not sensed • 1 - sensed
rightEndPosSens2	Input	Bool	Sensor input for right position 2 (field input) • 0 - not sensed • 1 - sensed
rightEndPosSens3	Input	Bool	Sensor input for right position 3 (field input) • 0 - not sensed • 1 - sensed
rightEndPosSens4	Input	Bool	Sensor input for right position 4 (field input): • 0 - not sensed • 1 - sensed
rightEndPosSens5	Input	Bool	Sensor input for right position 5 (field input): • 0 - not sensed • 1 - sensed
rightEndPosSens6	Input	Bool	Sensor input for right position 6 (field input): • 0 - not sensed • 1 - sensed

Name	Declaration	Type	Description
rightEndPosSens7	Input	Bool	Sensor input for right position 7 (field input): • 0 - not sensed • 1 - sensed
rightEndPosSens8	Input	Bool	Sensor input for right position 8 (field input): • 0 - not sensed • 1 - sensed
pos	Input	Real	Actual position of the axis.
maxPos	Input	Real	maximum limit of the position along the line.
minPos	Input	Real	minimum limit of the position along the line.
showPos	Input	USInt	Determines the position of the setup line visible: • 00 – both, graphical representation, and numeric value is invisible • 01 – both, graphical representation, and numeric value is visible • 02 – only the numeric value is visible You can set the value to 00 and use the line for Powerup condition.
line	InOut	LAPP_typeManualOperation	Assign manual operation line data. For example, "NutRunner1.lines[x]".
status	Output	Word	Displays the status codes of the block. For more information, refer to Status codes (Page 80).
errID	Output	Word	Displays the error codes of the block. For more information, refer to "Information System > Programming a PLC > Instructions > Instructions (S7-1200, S7-1500) > Basic instructions (S7-1200, S7-1500) > Program control operations (S7-1500) > Runtime control (S7-1500) > GET_ERR_ID" instruction in TIA portal online help.

x - line number

3.9.1.5 Data type

LAPP_typeFunGrp

Table 3-10 Structure of data type "LAPP_typeFunGrp"

Name	Data type	Description
funcGrpNum	UInt	Displays the specific ID for the function group.
actvArea	UInt	Displays the active area.

LAPP_typeManualOperationArea

Table 3-11 Structure of data type "LAPP_typeManualOperationArea"

Name	Data type	Description
areaNum	UInt	Displays the area number
displayedLines	String	The list of lines to be displayed in the area

LAPP_typeManualOperation

Refer to "LAPP_typeManualOperation (Page 67)" for more information on the data type.

3.9.1.6 Configuring addressing modes

The display text on the line can be configured in symbolic, electrical, and mechanical addressing modes.

Note

You can configure mechanical addressing mode by enabling "enableMechAddr" in the "LAPP_ManualOperation" block parameter.

You can configure the below parameters in "LAPP_InternalManualData" data block to set the default addressing mode and auto switching of addressing modes in runtime.

Input parameter	Data type	Description
hmi.hmi[1].addr-Mode.default	USint	Configured addressing mode is set as default in runtime. • 1 - Symbolic addressing mode • 2 - Electrical addressing mode • 3 - Mechanical addressing
hmi.hmi[1].addr-Mode.enableAutoSwitch	Bool	Enable this to toggle back to default addressing mode if switched to other addressing mode. • True - enables auto switching to default addressing mode • False - disables auto switching to default addressing mode
hmi.hmi[1].addr-Mode.autoSwitchTime	Time	Time delay to switch back to the default addressing mode. By default, the value is set to 5 seconds.

Note

- Auto switch timer works only when you enable auto switching of addressing modes.
- Add the names of addressing modes in "LAHP_manualOperationTemplate" screen.

3.9.1.7 Examples

Configuring number of lines

Example

Scenario 1
To configure maximum lines to 100, then enter the value at "NutRunner1.lines" as shown:

NutRunner1

	Name	Data type
1	Static	
2	Lines	Array[1..100] of "LAPP_typeManualOperation"

Figure 3-33 NutRunner1 data block

Scenario 2

To configure maximum lines to 128, then enter the value at "NutRunner1.lines" as shown:

NutRunner1

	Name	Data type
1	Static	
2	Lines	Array[1..128] of "LAPP_typeManualOperation"

Figure 3-34 NutRunner1 data block

Configuring number of areas

Example

Scenario 1
To configure 5 areas, enter the value at "NutRunner1.areas" as shown:

NutRunner1

	Name	Data type
1	Static	
2	Lines	Array[1..100] of "LAPP_typeManualOperation"
3	Areas	Array[1..5] of "LAPP_typeManualOperationArea"

Figure 3-35 NutRunner1 data block

Scenario 2

To configure 7 areas, enter the value at "NutRunner1.areas" as shown:

NutRunner1

	Name	Data type
1	Static	
2	Lines	Array[1..128] of "LAPP_typeManualOperation"
3	Areas	Array[1..7] of "LAPP_typeManualOperationArea"

Figure 3-36 NutRunner1 data block

Note

A maximum of only 7 areas can be configured.

Configuring showline

Example to Show line

To show the line in manual operation screen, set the parameter values as shown below:

Input parameter	Value	Description
"line"	"Line2"	In runtime, line 2 will be shown.
"showLine"	1	If value is 1, then the line is visible during runtime.

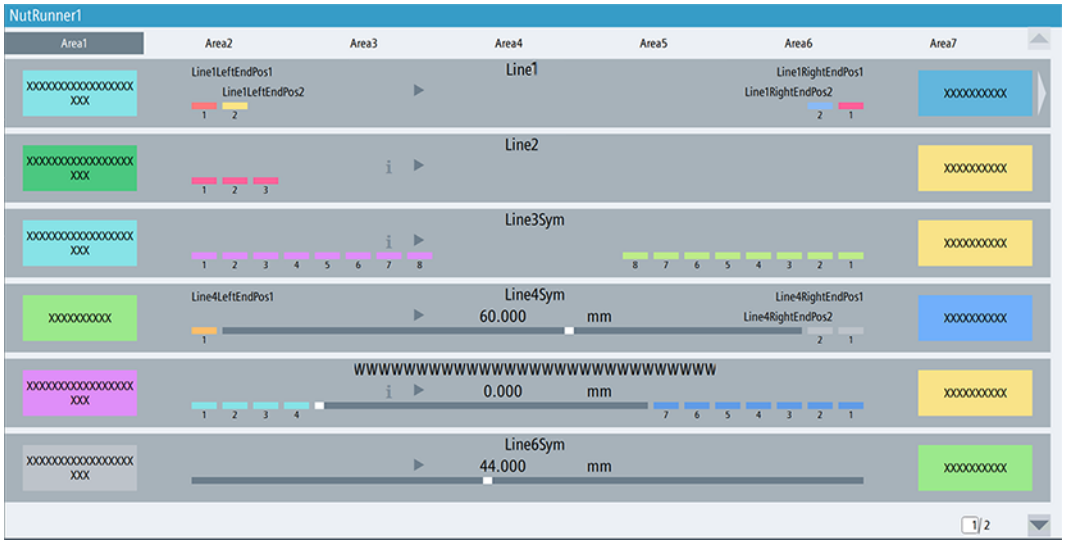


Figure 3-37 Line visible

Example to hide line

To hide the line in manual operation screen, set the parameter values as shown below:

Input parameter	Value	Description
"line"	"Line2"	In runtime, Line2 will be shown.
"showLine"	0	If value is 0, then the line is visible during runtime.

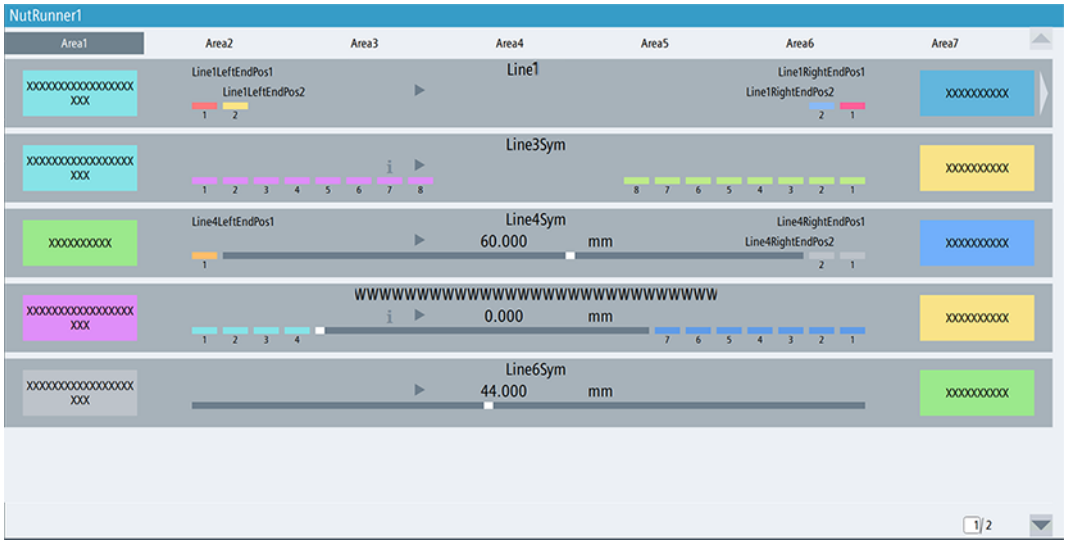


Figure 3-38 Line2 invisible

Note

You can hide more than one line.

Configuring big area

Example to enable big area

To enable the line operating area to full screen in manual operation screen, set the parameter values as shown below:

Input parameter	Value	Description
"line"	"Line2"	In runtime, line 2 will be shown.
"disableLineSel"	0	Line selection is enabled.
"enableBigArea"	1	If value is 1, then the line operating area is set to full screen.



Figure 3-39 "enableBigArea" enabled

Example to disable big area

To disable the full screen operating area of the line in manual operation screen, set the parameter values as shown below:

Input parameter	Value	Description
"line"	"Line2"	In runtime, line 2 will be shown.
"disableLineSel"	0	Line selection is enabled.
"enableBigArea"	0	If value is 0, then the line operating area is restricted to the button area.

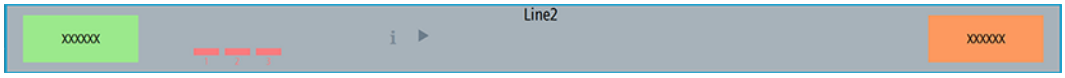


Figure 3-40 "enableBigArea" disabled

Configuring separator

Example

To add separator in between lines, add 0 while defining the displayed lines in DB.
Scenario 1

To define the 2nd line as separator, add 0 as shown,

	Name	Data type	Start value
1	Static		
2	funcGrp	"LAPP_typeFuncGrp"	
3	areas	Array[1..7] of "LAPP_typeManualOper...	
4	areas[1]	"LAPP_typeManualOperationArea"	
5	areaNum	UInt	101
6	displayedLines	String	'1,0,2-10,3-1;'
7	hmiPage	"LAPP_typePageDetails"	
8	areas[2]	"LAPP_typeManualOperationArea"	
9	areaNum	UInt	102
10	displayedLines	String	'18-60,100-78,44;'
11	hmiPage	"LAPP_typePageDetails"	
12	areas[3]	"LAPP_typeManualOperationArea"	

Figure 3-41 DB

In runtime you can view as follows,

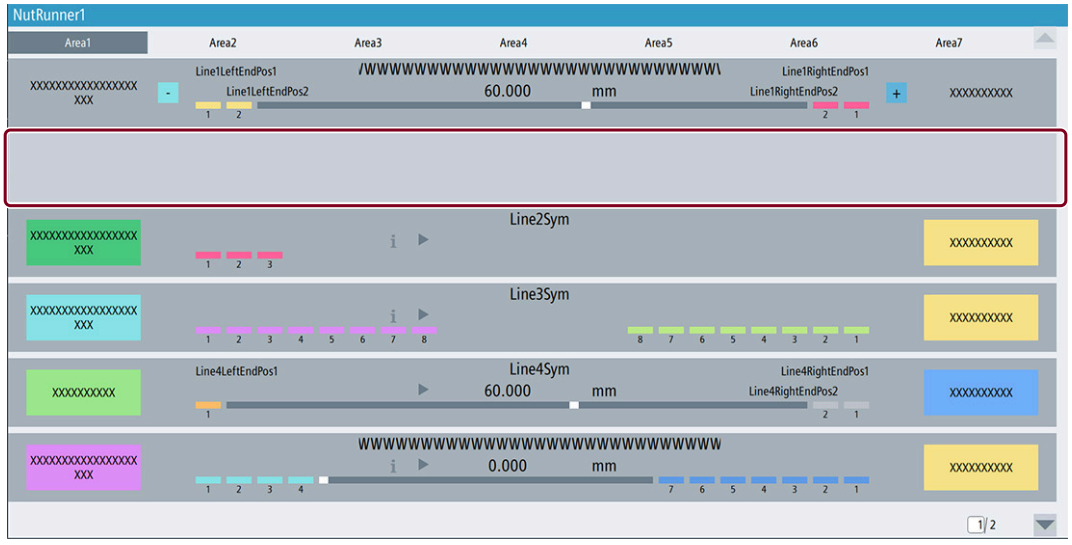


Figure 3-42 Separator

Scenario 2

To define the 4th line as separator, add 0 as shown,

		Name	Data type	Start value
1		Static		
2		funcGrp	"LAPP_typeFuncGrp"	
3		areas	Array[1..7] of "LAPP_typeManualOper...	
4		areas[1]	"LAPP_typeManualOperationArea"	
5		areaNum	UInt	101
6		displayedLines	String	'1-3,0,4-10,3-1;'
7		hmiPage	"LAPP_typePageDetails"	
8		areas[2]	"LAPP_typeManualOperationArea"	
9		areaNum	UInt	102
10		displayedLines	String	'18-60,100-78,44;'
11		hmiPage	"LAPP_typePageDetails"	
12		areas[3]	"LAPP_typeManualOperationArea"	

Figure 3-43 DB

In runtime you can view as follows,



Figure 3-44 Separator

3.9.1.8 Status codes

List

Following are the status codes for "LAPP_manualOperationMain".

Status code	Description
16#7000	Block is in initialized state.
16#7002	Block is in execution state and has no errors.
16#8400	No connection to HMI.
16#8200	Faulty license.
16#8201	Invalid HMI index.

Following are the status codes for "LAPP_funcGrpConfig".

Status code	Description
16#7000	Block is in initialized state.
16#7002	Block is in execution state and has no errors.
16#8400	No connection to HMI.
16#8200	Faulty license.
16#8201	Invalid HMI index.
16#8202	Input at displayed lines is empty.
16#8203	The specified range of lines are not in the required format.
16#8204	Incorrect line numbers entered in the input parameters of the displayed line.
16#8205	The number of active lines exceeds the maximum limit (128).

Following are the status codes for "LAPP_manualOperation".

Status code	Description
16#7002	Block is in execution state and has no errors.
16#8400	No connection to HMI.
16#8200	Faulty license.
16#8201	Invalid HMI index.

3.9.2 Manual Operation as faceplate

3.9.2.1 Manual Operation as faceplate

You can use the manual operation lines in any of the required screen using "LAPP_ManualOperationFaceplate" block. For each "LAPP_ManualOperationFaceplate" block you can add up to 6 lines. To add faceplate on multiple screens, add a separate block and trigger to execute the block.
For example,

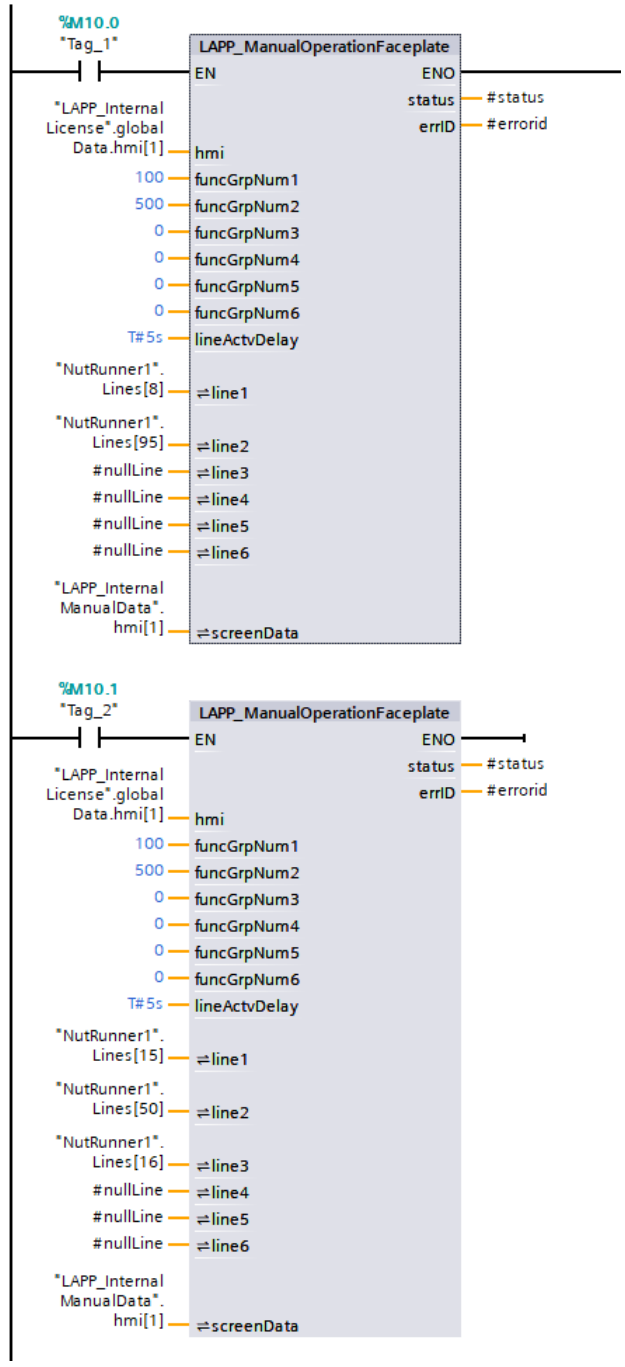


Figure 3-45 "LAPP_manualOpeartionFaceplate"

Procedure

1. In "Instructions" task card, navigate to "Optional packages > Create MyHMI /Automotive" and drag the "LAHP_ManualOperationFacePlate".
2. Interface the input and output parameters. For more information on block interface, refer to Block interface - LAPP_ManualOperationFaceplate (Page 84).

Note

You can use indirect addressing for the faceplate to minimize the number of power tags in HMI. For more information, refer to Addressing tags indirectly on TIA Portal online help.

3.9.2.2 Block interface - LAPP_ManualOperationFaceplate

Block interface

This block is used for data handling and configurations.

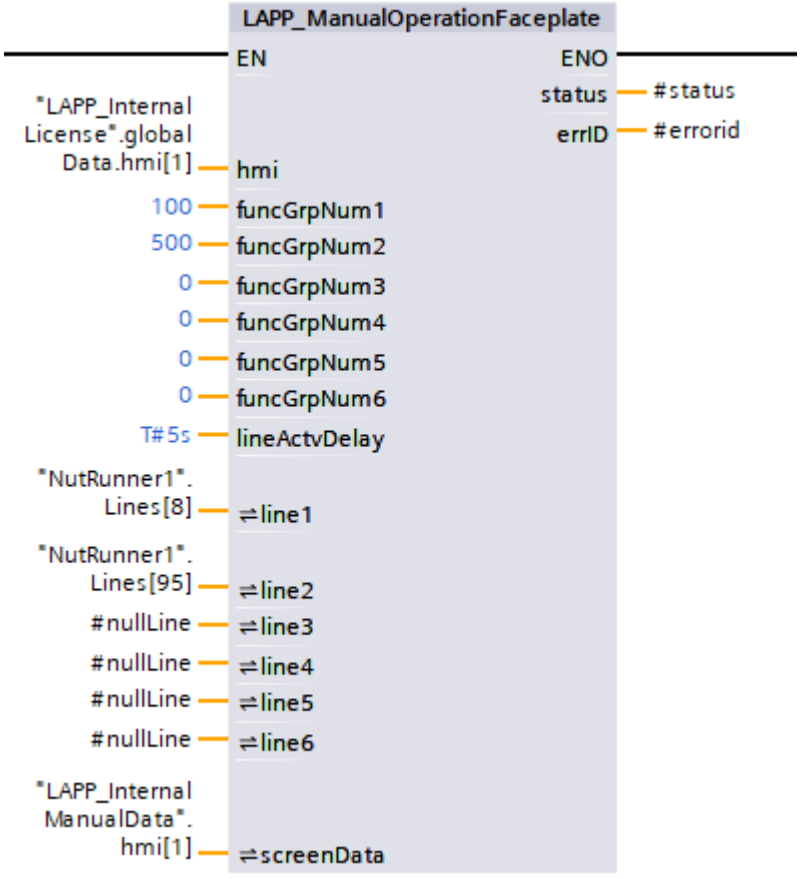


Figure 3-46 "LAPP_ManualOperationFaceplate"

Table 3-12 Parameters of "LAPP_ManualOperationFaceplate"

Name	Declaration	Type	Description
hmi	Input	LAPP_typeIntlLicense	Configure the HMI Index which is defined while setting up the HMI device. **Note**: HMI index value cannot be 0. It should match with HMI index value used in License DB ("LAPP_InternalLicense").
funcGrpNum1	Input	UInt	Interface the function group number of the line you want to interface at "line1" parameter.
funcGrpNum2	Input	UInt	Interface the function group number of the line you want to interface at "line2" parameter.
funcGrpNum3	Input	UInt	Interface the function group number of the line you want to interface at "line3" parameter.
funcGrpNum4	Input	UInt	Interface the function group number of the line you want to interface at "line4" parameter.
funcGrpNum5	Input	UInt	Interface the function group number of the line you want to interface at "line5" parameter.
funcGrpNum6	Input	UInt	Interface the function group number of the line you want to interface at "line6" parameter.
lineActvDelay	Input	Time	Set the line active delay for the function group.
line1	InOut	LAPP_typeManualOperation	Represents the number of manual operation lines configured. For example, Assign manual operation lines from "NutRunner1.lines[x]". **Note**: If the line is not configured, create a tag for null line with data type "LAPP_typeManualOperation" and interface.
line2	InOut	LAPP_typeManualOperation	Represents the number of manual operation lines configured. For example, Assign manual operation lines from "NutRunner1.lines[x]". **Note**: If the line is not configured, create a tag for null line with data type "LAPP_typeManualOperation" and interface.
line3	InOut	LAPP_typeManualOperation	Represents the number of manual operation lines configured. For example, Assign manual operation lines from "NutRunner1.lines[x]". **Note**: If the line is not configured, create a tag for null line with data type "LAPP_typeManualOperation" and interface.
line4	InOut	LAPP_typeManualOperation	Represents the number of manual operation lines configured. For example, Assign manual operation lines from "NutRunner1.lines[x]". **Note**: If the line is not configured, create a tag for null line with data type "LAPP_typeManualOperation" and interface.
line5	InOut	LAPP_typeManualOperation	Represents the number of manual operation lines configured. For example, Assign manual operation lines from "NutRunner1.lines[x]". **Note**: If the line is not configured, create a tag for null line with data type "LAPP_typeManualOperation" and interface.

Name	Declaration	Type	Description
line6	InOut	LAPP_typeManualOperation	Represents the number of manual operation lines configured. For example, Assign manual operation lines from "NutRunner1.lines[x]". **Note**: If the line is not configured, create a tag for null line with data type "LAPP_typeManualOperation" and interface.
screenData	InOut	LAPP_typeIntlFuncGrpHmiDisp	Interface the HMI tag from "LAPP_InternalManualData.hmi[y]". **Note**: "LAPP_InternalManualData" is an internal DB. Modify the array value to increase the HMI device support. For more information, refer to Configuring PLC blocks for different HMI.
status	Output	Word	Displays the status codes of the block. For more information, refer to Status codes (Page 86).
errID	Output	Word	Displays the error codes of the block. For more information, refer to "Information System > Programming a PLC > Instructions > Instructions (S7-1200, S7-1500) > Basic instructions (S7-1200, S7-1500) > Program control operations (S7-1500) > Runtime control (S7-1500) > GET_ERR_ID" instruction in TIA portal online help.

x- indicates line number.

y- indicates HMI device number.

Note

- All the actual data must be programmed by OEM.
- Ensure to configure null line if the line is not configured for the block, else the block will not work as intended.
- Ensure to interface the same HMI device number at "hmiIndex" and "screenData" value.

3.9.2.3 Status codes

List

Following are the status codes for "LAPP_manualOperationFaceplate".

Status code	Description
16#7002	Block is in execution state and has no errors.
16#8400	No connection to HMI.
16#8200	Faulty license.
16#8201	Invalid HMI index.

3.10 Tool life monitor

3.10.1 Overview

Tool life monitor is used to indicate the life expectancy of the tools used in machining. It is a database provided to monitor tool used in production. You can also view the status of the tool - if the tool is in active state, if there are any alarms on the tool and/or if the life of the tool has expired.

The "LAHP_toolLifeMonitor" runtime screen is shown below:

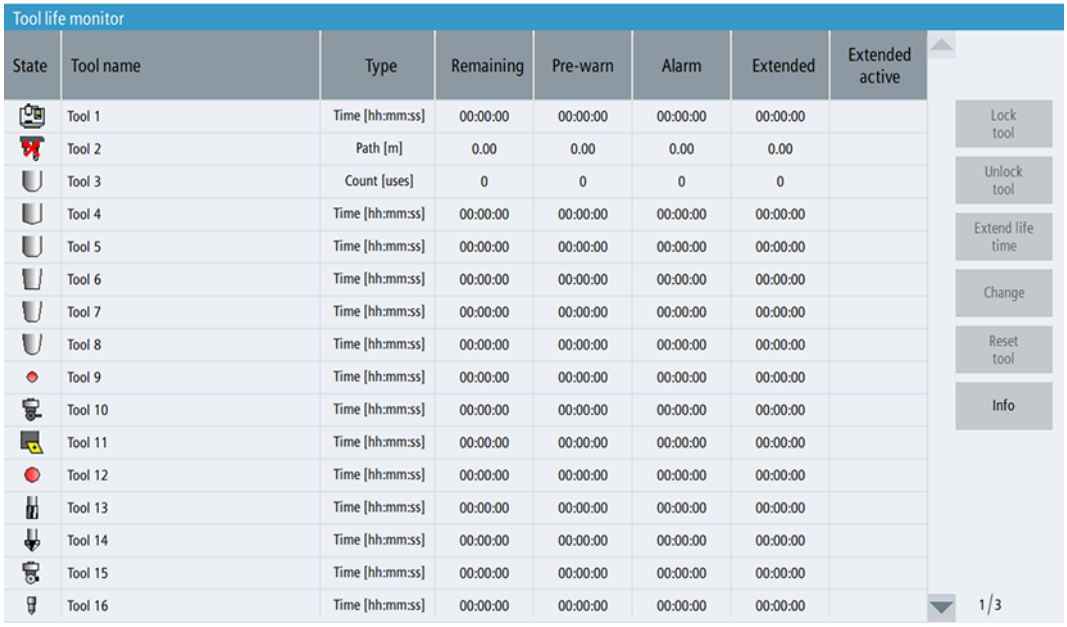


Figure 3-47 Tool life monitor screen

Tool life monitor is available in three modes.

- Time - The tool can be monitored based the duration for which it is operated.
- Length - The tool can be monitored based on the length of the tool.
- Count - The tool can be monitored based the number of times it is used, or the number pieces it manufactures.

Procedure to configure the blocks

Follow the steps below for the tool life screen to work as intended.

1. Configure the block "LAPP_ToolLifeMain" (Page 89) once in the cyclic block or main block and interface the number of tools and the data to be fetched to display on the screen.
2. **Note**
   - Perform this step only for the tools to be configured for the length or count mode.
   - Ensure to add this logic before you configure the "LAPP_toolLifeMonitor" block.

   If you reset the tool from the runtime screen using the "Reset tool" button i.e. if the "notifier" value in "LAPP_ToolLifeMonitor (Page 91)" block is 11, then you must ensure that the actual value is reset to the initial value. For this reset operation to take place use the logic shown in the figures below.

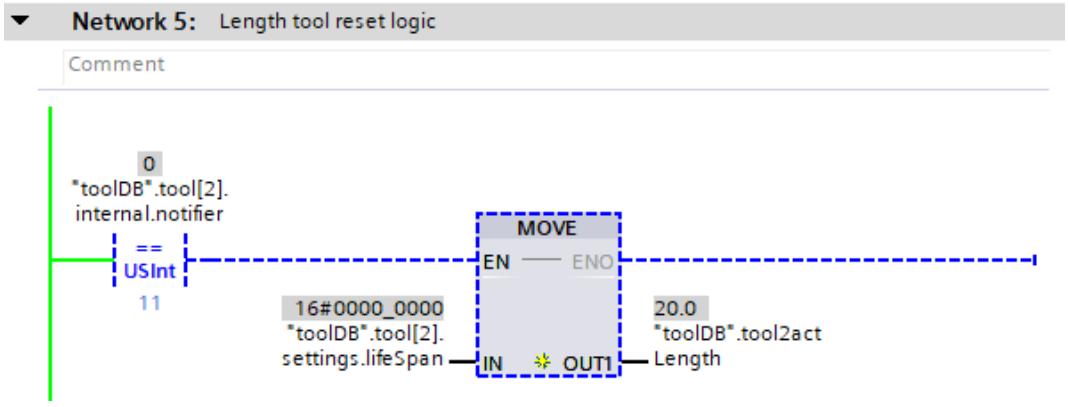


Figure 3-48 Reset logic for tool in length mode

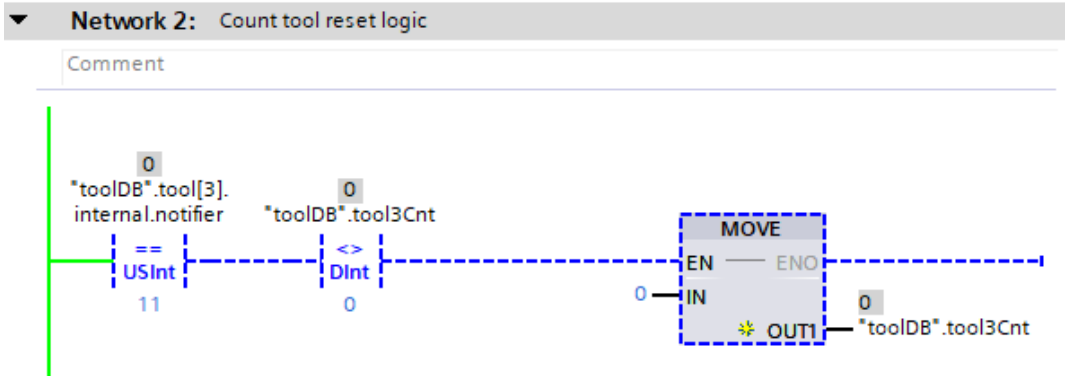


Figure 3-49 Reset logic for tool in count mode

3. Configure the block "LAPP_toolLifeMonitor (Page 91)" for each tool and interface the required input and output parameters.

For more information on the configuring the block, refer to AUTOHOTSPOT.

To modify the displayed tool name and the tool information in WinCC Unified, refer to Text and graphic list configuration.

3.10.2 Configuring number of tools

Procedure

1. Create new data block. For example: Demo
2. Create a tag with array[1.....X] of "LAPP_typeToolLifeMonitor" as data type.
   X - enter the maximum tools in the array value
3. Interface the tag to "tools" parameter of the "LAPP_ToolLifeMain" block.

For example,
Scenario 1

To configure maximum tools to 30, create a tag with array value as [1..30] as shown:

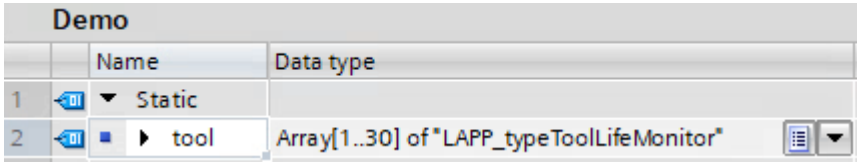


Figure 3-50 Demo DB

Scenario 2

To configure maximum tools to 300, create a tag with array value as [1..300] as shown:

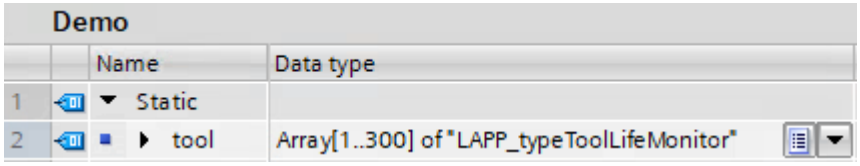


Figure 3-51 Demo DB

Result

You have configured the number of tools for monitoring the tool life.

3.10.3 Block interface - LAPP_ToolLifeMain

The "LAPP_ToolLifeMain" block is used to configure number of tools used.

Note

You need call this mandatory block only once in cyclic block/ main block.

Block interface

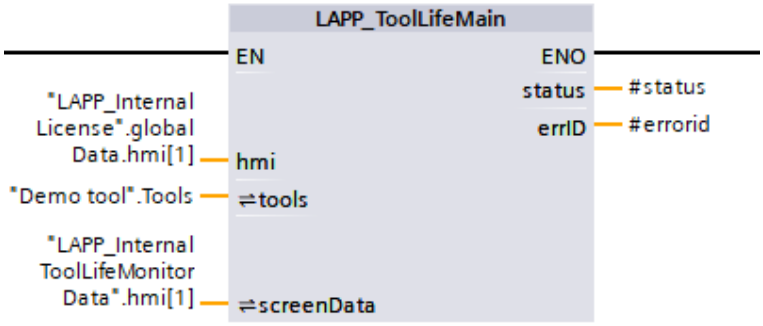


Figure 3-52 "LAPP_ToolLifeMain"

Parameters

Table 3-13 Parameters of "LAPP_ToolLifeMain"

Name	Declaration	Type	Description
hmi	Input	LAPP_typeIntlLicense	Configure the HMI Index which is defined while setting up the HMI device. **Note**: HMI index value cannot be 0. It should match with HMI index value used in License DB ("LAPP_InternalLicense").
tools	InOut	Array[*] of LAPP_typeToolLifeMonitor	Interface the PLC tag with "LAPP_toolLifeMain" data type. You are interfacing number of tools defined for use.
screenData	InOut	LAPP_typeIntlHmiToolScreenData	Interface the HMI tag from "LAPP_InternalToolLifeMonitorData.hmi[y]". **Note**: "LAPP_InternalToolLifeMonitorData" is an internal DB. Modify the array value to increase the HMI device support. For more information, refer to Configuring PLC blocks for different HMI.
status	Output	Word	Displays the status codes of the block. For more information, refer to Status codes (Page 95).
errID	Output	Word	Displays the error codes of the block. For more information, refer to "Information System > Programming a PLC > Instructions > Instructions (S7-1200, S7-1500) > Basic instructions (S7-1200, S7-1500) > Program control operations (S7-1500) > Runtime control (S7-1500) > GET_ERR_ID" instruction in TIA portal online help.

y- indicates HMI device number.

Note

To view the data in multiple HMI, you need to configure multiple blocks with same configuration and different "hmi" value.

3.10.4 Block interface - LAPP_ToolLifeMonitor

Based on the mode selection, interface the parameters of the "LAPP_ToolLifeMonitor" block.

You need to configure this block for each tool.

For example, if you need to configure 10 tools, use this block ten times to configure 10 tools. Each block is configured with one tool data.

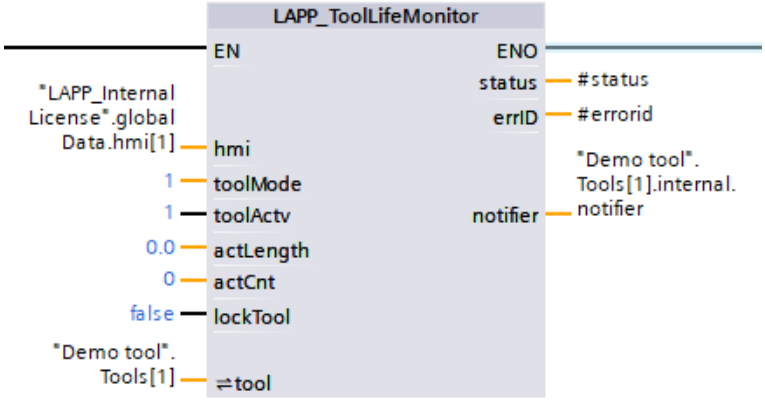


Figure 3-53 "LAPP_ToolLifeMonitor" time mode

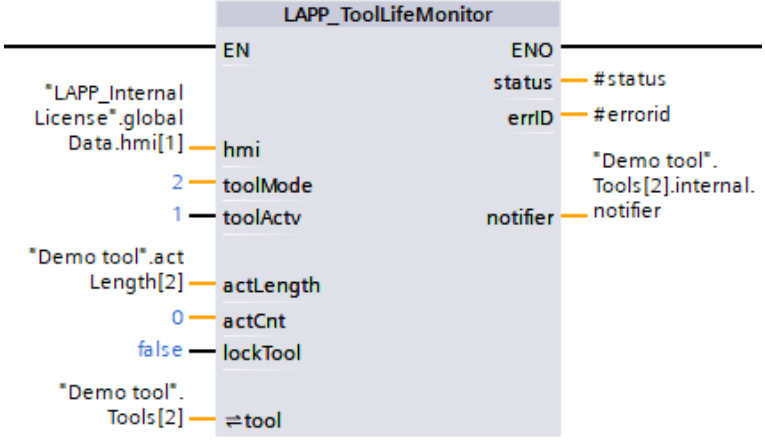


Figure 3-54 "LAPP_ToolLifeMonitor" length mode

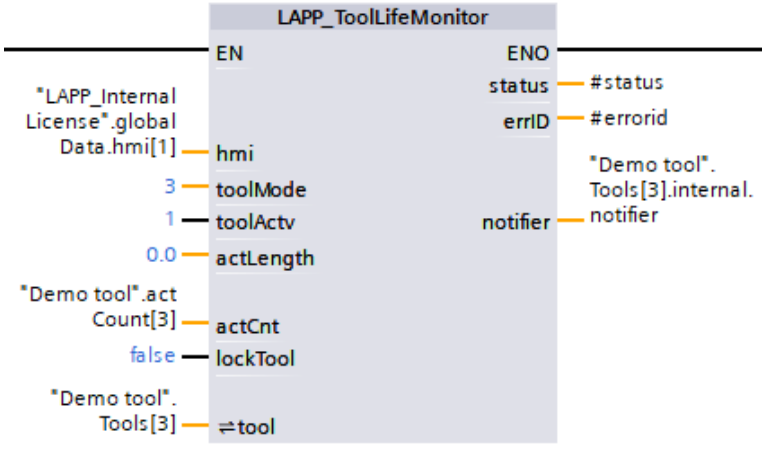


Figure 3-55 "LAPP_ToolLifeMonitor" count mode

⚠ WARNING

Dynamization of the modes

Do not change the mode in runtime. If you change, you need to reconfigure the tool from UI and reactivate the tool from PLC.

Parameters

Table 3-14 Parameters of "LAPP_ToolLifeMonitor"

Name	Declaration	Type	Description
hmi	Input	LAPP_typeIntlLicense	Configure the HMI Index which is defined while setting up the HMI device. **Note**: HMI index value cannot be 0. It should match with HMI index value used in License DB ("LAPP_InternalLicense").
mode	Input	USInt	Operating mode of the tool: • 1 - Time • 2 - Length • 3 - Count
toolActv	Input	Bool	Enable to activate and start the timer of the tool: • 0 - inactive • 1 - active
actLength	Input	Real	Actual length of the tool (to be entered if the tool is in length mode)

Name	Declaration	Type	Description
actCnt	Input	DInt	Actual count of the tool (to be entered if the tool is in count mode)
lockTool	Input	Bool	Enable to lock the tool. "lockTool" can be used in case of an emergency. **Note:** The tool cannot be unlocked by disabling the parameter.
tool	InOut	LAPP_typeToolLifeMonitor	Tool life structure.
status	Output	Word	Displays the status codes of the block. For more information, refer to Status codes (Page 95).
errID	Output	Word	Displays the error codes of the block. For more information, refer to "Information System > Programming a PLC > Instructions > Instructions (S7-1200, S7-1500) > Basic instructions (S7-1200, S7-1500) > Program control operations (S7-1500) > Runtime control (S7-1500) > GET_ERR_ID" instruction in TIA portal online help.
notifier	Output	USInt	The value displayed indicates the state of the tool. For more information, refer to Notifier data (Page 94).

3.10.5 Data type

LAPP_typeToolLifeMonitor

This data type is used to configure the details of the tool in use.

Table 3-15 Structure of data type "LAPP_typeToolLifeMonitor"

Name	Type	Description
mode	USInt	Operating mode of the tool: • 1 - Time • 2 - Length • 3 - Count
toolIndex	UInt	Tool index number
state	USInt	State of the tool: • 0 - Ok • 1 - Prewarn • 2 - End of life • 3 - Locked

Name	Type	Description
extndLifeActv	Bool	Extended life: • 0 - Not active • 1 - Active
edgeNum	UInt	Edge tool number
sisterToolNum	UInt	sister tool number
settings	LAPP_typeIntlToolLife-Settings	Limits of tool settings

Table 3-16 Structure of data type "LAPP_typeIntlToolLifeSettings"

Name	Type	Description
lifeSpan	DWord	Life of the tool
remaining	DWord	Remaining measure of the tool
preWarn	DWord	Pre-warning limit to measure the tool operation
alarm	DWord	Alarm limit to measure the tool operation
extndAlarm	DWord	Extended alarm limit to measure the tool operation

3.10.6 Notifier data

List

The table below lists the notifier value and its description for the "notifier" parameter in "LAPP_ToolLifeMonitor" block.

Notifier value	Description
1	The tool life is not extended and the tool is locked.
2	The tool reaches end of life and gets locked before the life is extended. If you unlock this tool and extend the life time then, the tool will be in alarm state.
3	The tool unlocked using "Unlock tool" button from the screen is in active state.
4	The tool is locked while its life is extended.
5	The tool is locked from the block after end of life.
6	The tool is unlocked using the "Unlock tool" button in the screen.
7	The actual value of the tool is not reset to the initial value after it becomes zero.
8	The actual value of the tool is reset to the initial value after it becomes zero.
9	The tool is locked using the "lockTool" parameter in the block.
10	The tool life is extended and it is locked using the "lockTool" parameter in the block.
11	Command for resetting the tool from the screen.

3.10.7 Status codes

List

The following are the status codes for "LAPP_ToolLifeMain" block:

Status Code	Description
16#7000	Block is in initialized state.
16#7002	Block is in execution state and has no errors.
16#8200	Faulty license.
16#8201	Invalid HMI index.
16#8400	No connection to HMI.

The following are the status codes for "LAPP_ToolLifeMonitor" block:

Status Code	Description
16#7000	Block is in initialized state.
16#7002	Block is in execution state and has no errors.
16#8200	Faulty license.
16#8201	Invalid HMI index.
16#8202	Incorrect mode of the tool.
16#8203	Actual value is not within the limit.
16#8204	The actual value of the tool is not reset/ does not match extended value in case it is in alarm mode and extended. Note: This is applicable only in length mode.
16#8205	Tool has to be replaced or recommissioned.
16#8400	No connection to HMI.

3.10.8 Modifying the actual value

The actual value (length) must be modified if the "LAPP_toolLifeMonitor" block triggers the status (16#8204).

Procedure

1. Check if the value of "actLength" is zero.
2. Check if the value of "extendedLifeActive" is "TRUE".
3. Pass the value of "extendedAlarm" to "actLength". The data type of "extendedAlarm" (DWord) must be converted to REAL before passing the value.

Note

DWORD_TO_REAL (LAPP_toolLifetool[i].settings.extendedAlarm) - Read this value based on the condition: tool life is extended, actual value = 0 and state of the tool = 2.

LAPP_toolLife

	Name	Data type	Start value	Monitor value
1	Static			
2	tool	Array[1..15] of "LAPP_typ...		
3	tool[1]	"LAPP_typeToolLifeMonit...		
4	mode	USInt	1	2
5	toolIndex	UInt	0	1
6	state	USInt	0	2
7	extndLifeActv	Bool	false	TRUE
8	edgeNum	UInt	0	0
9	sisterToolNum	UInt	0	0
10	settings	"LAPP_typeToolLifeSettin...		
11	lifeSpan	DWord	16#0	16#41A0_0000
12	remaining	DWord	16#0	16#41A0_0000
13	preWarn	DWord	16#0	16#40C0_0000
14	alarm	DWord	16#0	16#4080_0000
15	extndAlarm	DWord	16#0	16#4000_0000

Figure 3-56 "LAPP_toolLife"

You can also modify the parameter "actLength" by directly entering the value in the "LAPP_ToolLifeMonitor" block (Page 91). This entered value must match the value mentioned in "Extended" column in the runtime.

Result

The actual length is modified to the extended limit.

3.11 Process Dashboard

3.11.1 Overview

The process dashboard is used to view the quality metrics, check details of the workpiece, current workpiece in production, tool life details, and add notes to next shift operator.

The "LAPP_TypeWidgetPartDetails", "LAPP_TypeWidgetProductionStatus", "LAPP_TypeWidgetQualityMetrics", "LAPP_TypeWidgetWpiece" UDT and "LAPP_typeWidgetToolLife" block is used to parametrize the process dashboard screen during runtime.

You need to configure the following for process dashboard screen to work as intended:

- Create tags using the UDT and configure the "LAPP_typeWidgetToolLife" (Page 100) block in Step 7.
  For more information on creating a data block to use in "LAHP_ProcessDashboard" screen, refer to Creating data block (Page 97).
  For more information on the configuring the block, refer to AUTOHOTSPOT.
- HMI tag creation and text list configuration to modify the displayed tool name and the tool information in WinCC Unified.

3.11.2 Creating data block

To configure the process dashboard block, you need to create data block.

Procedure

To create data block for process dashboard:

1. Right-click "Program blocks" and click "Add new block"

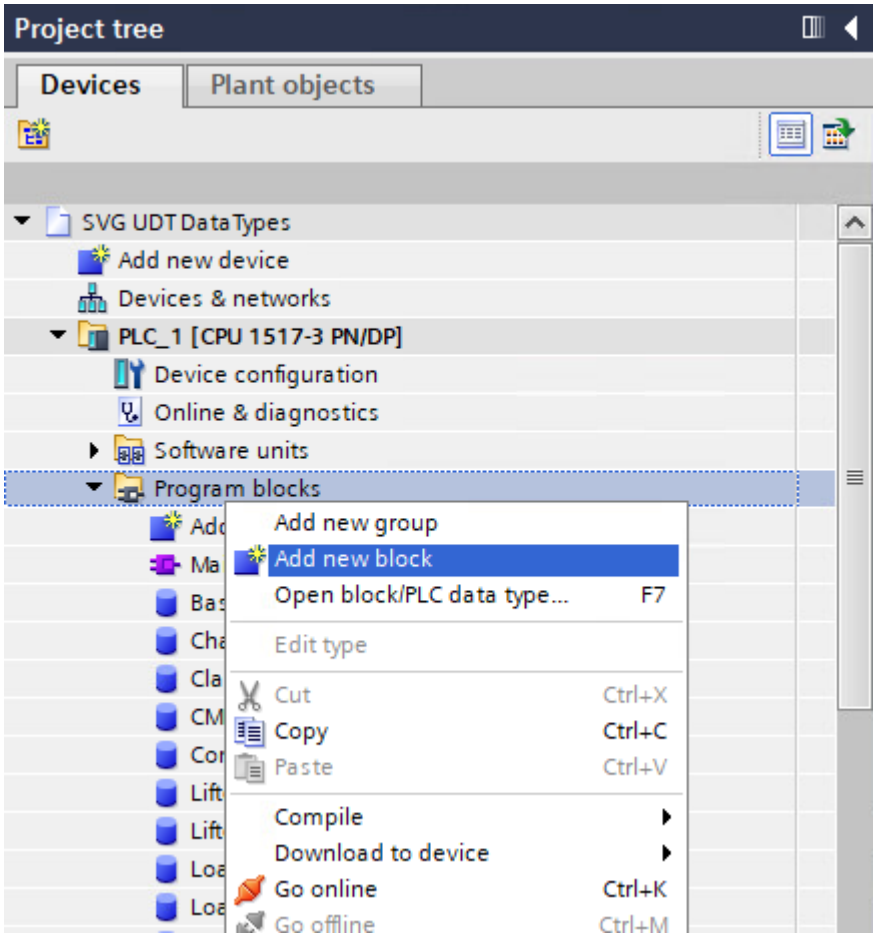


Figure 3-57 Add new block

"Add new block" window opens.

2. Click "Data block".

3. In "Type" field, select "LAPP_typeWidgetPartDetails", "LAPP_typeWidgetProductionStatus", "LAPP_typeWidgetQualityMetrics" or "LAPP_typeWidgetWpiece" to configure the part details, production status, quality metrics or the workpiece details respectively.

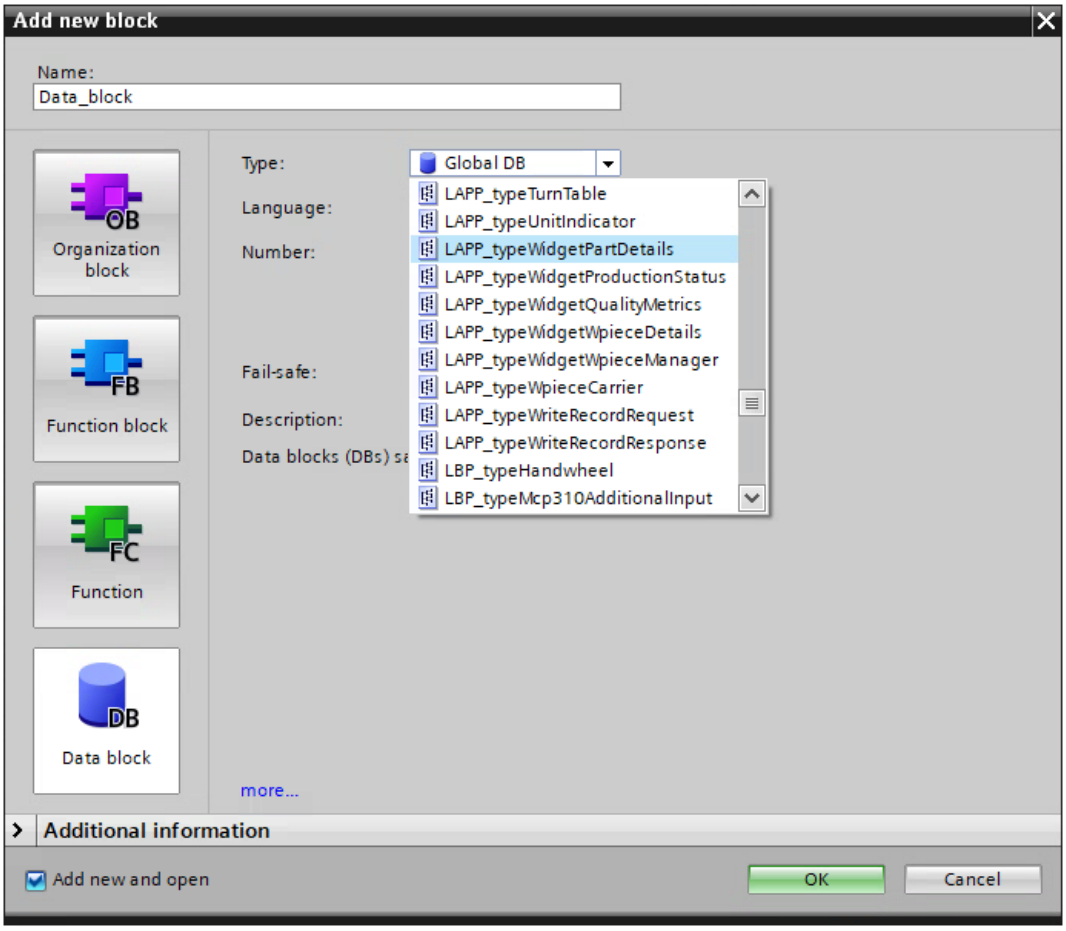


Figure 3-58 Data block creation

4. Click "OK".

Result

Data block for dashboard gets created.

3.11.3 Block interface - LAPP_WidgetToolLife

Block interface

This block is used to configure the tool life widget in the "LAHP_ProcessDashboard" runtime screen.

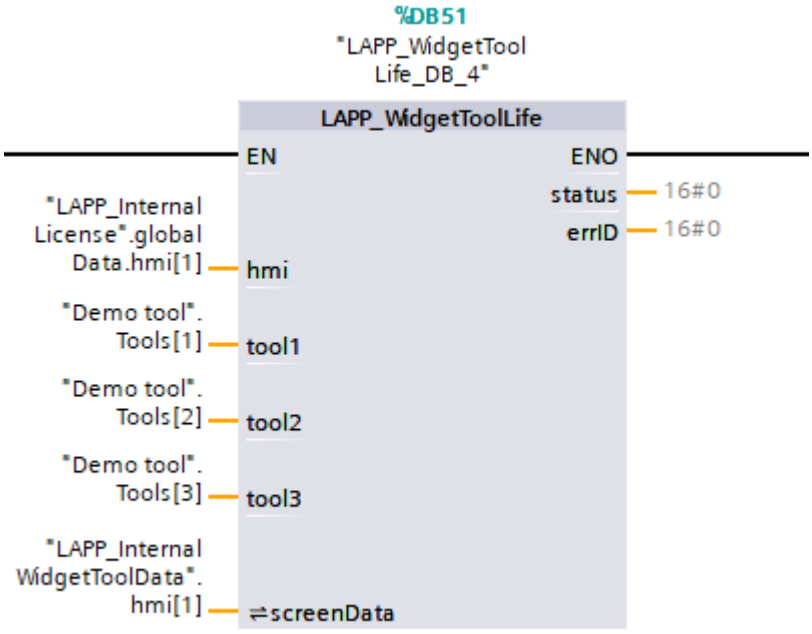


Figure 3-59 "LAPP_WidgetToolLife"

Parameters

Table 3-17 Parameters of "LAPP_WidgetToolLife"

Name	Declaration	Type	Description
hmi	Input	LAPP_typeIntlLicense	Configure the HMI Index which is defined while setting up the HMI device. **Note**: HMI index value cannot be 0. It should match with HMI index value used in License DB ("LAPP_InternalLicense").
tool1	Input	LAPP_typeToolLifeMonitor	Details of tool 1 to be displayed on the dashboard screen
tool2	Input	LAPP_typeToolLifeMonitor	Details of tool 2 to be displayed on the dashboard screen
tool3	Input	LAPP_typeToolLifeMonitor	Details of tool 3 to be displayed on the dashboard screen

Name	Declaration	Type	Description
screenData	InOut	LAPP_InternalWidgetToolData	Interface the HMI tag from "LAPP_InternalWidgetToolData.hmi[y]". **Note**: "LAPP_InternalWidgetToolData" is an internal DB. Modify the array value to increase the HMI device support. For more information, refer to Configuring PLC blocks for different HMI.
status	Output	Word	Displays the status codes of the block. For more information, refer to Status codes (Page 103).
errID	Output	Word	Displays the error codes of the block. For more information, refer to "Information System > Programming a PLC > Instructions > Instructions (S7-1200, S7-1500) > Basic instructions (S7-1200, S7-1500) > Program control operations (S7-1500) > Runtime control (S7-1500) > GET_ERR_ID" instruction in TIA portal online help.

y- indicates HMI device number.

LAPP_typeToolLifeMonitor

This data type is used to configure the details of the tool in use. For more information on the structure, refer to Data type (Page 93).

3.11.4 Data type

LAPP_typeWidgetWpiece

This data type is used to fetch the details of the workpiece in progress.

Table 3-18 Structure of data type "LAPP_typeWidgetWpiece"

Name	Data type	Description
currWpieceName	String	Name of the current workpiece in production
currWpiecePartID	String	Part ID of the current workpiece in production
currWpieceProgram	String	Name of the program used for current workpiece
setTime	LTime	Runtime set for the current workpiece
actTime	LTime	Actual machining time of the current workpiece
residualTime	LTime	Residual runtime of the current workpiece

LAPP_typeWidgetPartDetails

This data type is used to fetch the details of the workpiece count.

Table 3-19 Structure of data type "LAPP_typeWidgetPartDetails"

Name	Data type	Description
actWpieceTarget	UInt	Number of workpieces set for production
actWpieceGoodParts	UInt	Actual number of good parts produced
actWpieceBadParts	UInt	Actual number of bad parts produced
actShiftTarget	UDInt	Number of workpieces set for production in the current shift
actShiftGoodParts	UDInt	Actual number of good parts produced in the current shift
actShiftBadParts	UDInt	Actual number of bad parts produced in the current shift
dayTarget	UDInt	Number of workpieces set for production in a day
dayGoodParts	UDInt	Actual number of good parts produced in a day
dayBadParts	UDInt	Actual number of bad parts produced in a day

LAPP_typeWidgetQualityMetrics

This data type is used to fetch the details on the quality of the workpiece.

Table 3-20 Structure of data type "LAPP_typeWidgetQualityMetrics"

Name	Data type	Description
lastWpieceStatus	Bool	Status of the quality of the workpiece checked: • 0 = Not ok • 1 = ok
nextWpieceCheck	UInt	Program the counter to check for the next workpiece quality.

LAPP_typeWidgetProductionStatus

This data type is used to fetch the details on the quality of the workpiece.

Table 3-21 Structure of data type "LAPP_typeWidgetProductionStatus"

Name	Data type	Description
status1	Bool	Production status 1
status2	Bool	Production status 2
status3	Bool	Production status 3
status4	Bool	Production status 4

Name	Data type	Description
status5	Bool	Production status 5
stacklightCount	USInt	Stacklight count for production status

3.11.5 Status codes

List

Table 3-22 Description of status codes "LAPP_WidgetToolLife".

Status code	Description
16#7000	Block is in initialized state.
16#7002	Block is in execution state and has no errors.
16#8200	Faulty license.
16#8201	Invalid HMI index.
16#8202	The mode of tool entered is incorrect.
16#8203	The tool number is zero or invalid.
16#8400	No connection to HMI.

3.12 Hardware diagnostics

3.12.1 Camera

3.12.1.1 Overview

You can view live image from camera.

- The camera image allows you to track remote processes and monitor difficult-to-access areas.
- You can create and configure the required number of cameras.

The "LAHP_Camera" runtime screen is shown below.

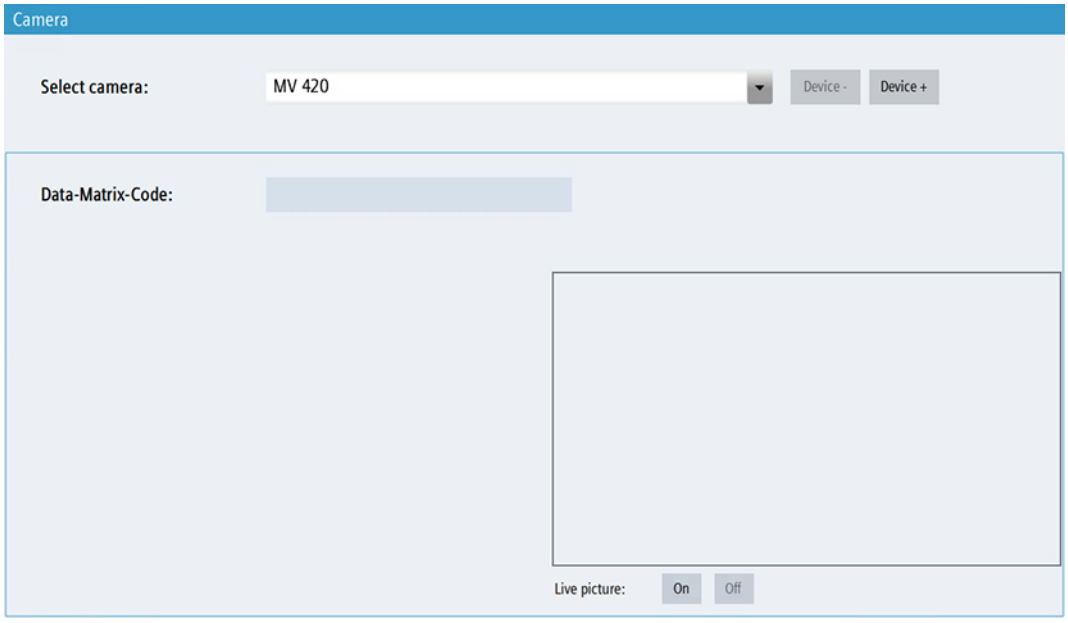


Figure 3-60 Camera

The "LAPP_typeCameraIPConfig" data type is used to configure the camera block.

You need to configure the following for "LAHP_Camera" screen to work as intended:

- Block configuration (Page 104) in Step 7
  For more information on configuring the block, refer to AUTOHOTSPOT.
- Text list configuration in WinCC Unified

3.12.1.2 Block interface - LAPP_Camera

This block is used for data handling and configurations.

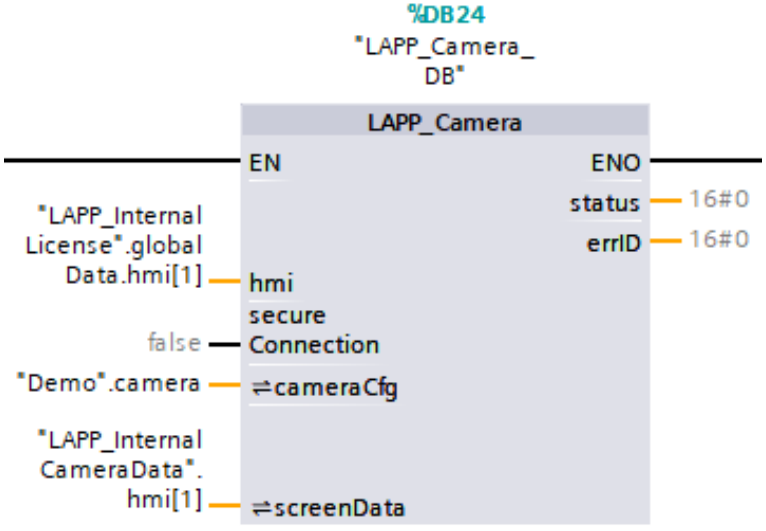


Figure 3-61 "LAPP_Camera"

Parameters

Table 3-23 Parameters of "LAPP_Camera"

Name	Declaration	Type	Description
hmi	Input	LAPP_typeIntlLicense	Configure the HMI Index which is defined while setting up the HMI device. **Note**: HMI index value cannot be 0. It should match with HMI index value used in License DB ("LAPP_InternalLicense").
secureConnection	Input	Bool	URL format for the camera: • 0 - "http://" • 1 - "https://"
cameraCfg	InOut	Array of LAPP_typeCameraIP-Config	Interface the PLC tag with data type "LAPP_typeCameraIPConfig"
screenData	InOut	LAPP_typeIntlHmiCamera	Interface the HMI tag from "LAPP_InternalCameraData.hmi[y]". Note: "LAPP_InternalCameraData" is an internal DB. Modify the array value to increase the HMI device support. For more information, refer to Configuring PLC blocks for different HMI.
status	Output	Word	Displays the status codes of the block. For more information, refer to Status codes (Page 106).
errID	Output	Word	Displays the error codes of the block. For more information, refer to "Information System > Programming a PLC > Instructions > Instructions (S7-1200, S7-1500) > Basic instructions (S7-1200, S7-1500) > Program control operations (S7-1500) > Runtime control (S7-1500) > GET_ERR_ID" instruction in TIA portal online help.

y- indicates HMI device number.

3.12.1.3 Data type

LAPP_typeCameraIPConfig

This data type is used for configuring the IP address.

Table 3-24 Structure of data type "LAPP_typeCameraIPConfig"

Name	Type	Description
cameraIP	String[15]	Configure the IP address of the camera

3.12.1.4 Status codes

List

Table 3-25 Description of status codes of "LAPP_Camera"

Status code	Description
16#7000	Block is in initialized state.
16#7002	Block is in execution state and has no errors.
16#8200	Faulty license.
16#8201	Invalid HMI index.
16#8202	Invalid IP address of the camera.
16#8400	No connection to HMI.

3.12.2 Energy efficiency

3.12.2.1 Overview

The energy block is used for measuring the energy consumption for different machine states. The "LAHP_EnergyEfficiency" screen is shown below.

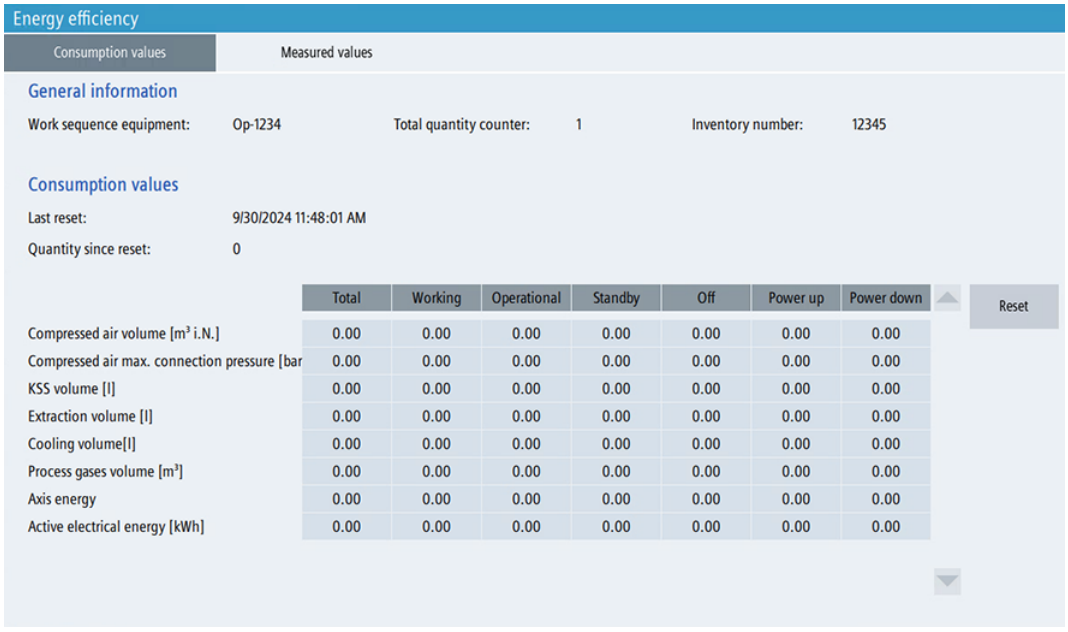


Figure 3-62 Energy efficiency

You need to configure the following for "LAHP_EnergyEfficiency" screen to work as intended:

- Block configuration (Page 109) in Step 7
  For more information on the configuring the block, refer to AUTOHOTSPOT.
- Text list configuration in WinCC Unified

Runtime interface (LAPP_EETransfer)

In the PLC, four blocks are provided for the use of **EE@TRANSLINE**. The **"LAPP_EE**" energy efficiency data block is the interface to the energy efficiency screens.

The "**LAPP_EETransfer"** block supplies the **"LAPP_EE"** block with the data relevant for the screen.
The measuring points for the energy efficiency (e.g. electrical energy and compressed air) are transferred to **"LAPP_EETransfer**". Measuring points that are independent of the energy state (e.g. supply pressure), are not processed via the **EnS_EEm_Calc** block, but instead are written directly to the **"LAPP_EE**" block. "**LAPP_EETransfer**" also calls the **EnS_EEm_Calc** block internally. **EnS_EEm_Calc** is responsible for the scaling, overflow check, measurements as well as for the distribution of the measuring points according to energy states. The database for the **EnS_EEm_Calc** block is the **"LAPP_EEmMachine"** data block.

All configurations are performed and operations controlled in the **"LAPP_EEmMachine**" block. The measuring points are already pre-assigned according to the **EE@TRANSLINE** concept. For this reason, the following variables of the "**LAPP_EEmMachine"** block must not be changed:

- machineConfiguration.eStateName
- measurementsConfiguration[0-7, 9].name
- measurementsOperation[0-7, 9].in

For more information refer to "Programming a PLC > Instructions > Instructions (S7-1200, S71500) > Optional packages (S7-1200, S7-1500) > Energy efficiency instructions" in TIA Portal online help.

Information on the pre-assigned measuring points of the "LAPP_EE" data block can be found in the following table:

Table 3-26 **Pre-assigned measuring points of the energy efficiency**

No.	Energy type	Measured variable	Unit in the interface data block	CMHA
0	Electrical energy	Active energy	kWh	Yes
1	Electrical energy	Apparent energy	kVAh	No
2	Electrical energy	Reactive energy	kvarh	No
3	Compressed air	Volume	kvarh	Yes
4	Compressed air	Maximum volume flow	normally m^3	No
5	Compressed air	Supply pressure	bar	No
6	Compressed air	Maximum supply pressure	bar	No
7	Cooling lubricant	Volume	l	Yes
8	Cooling lubricant	Supply pressure	bar	No
9	Extraction	Volume	l	Yes

No.	Energy type	Measured variable	Unit in the interface data block	CMHA
10	Cooling	Volume	l	Yes
11	Process gases	Volume flow	m^3	Yes
12	Axis energy		kWh	Yes

Note

The **CMHA** column specifies which measurements are displayed in CMHA

3.12.2.2 Block interface - LAPP_EETransfer

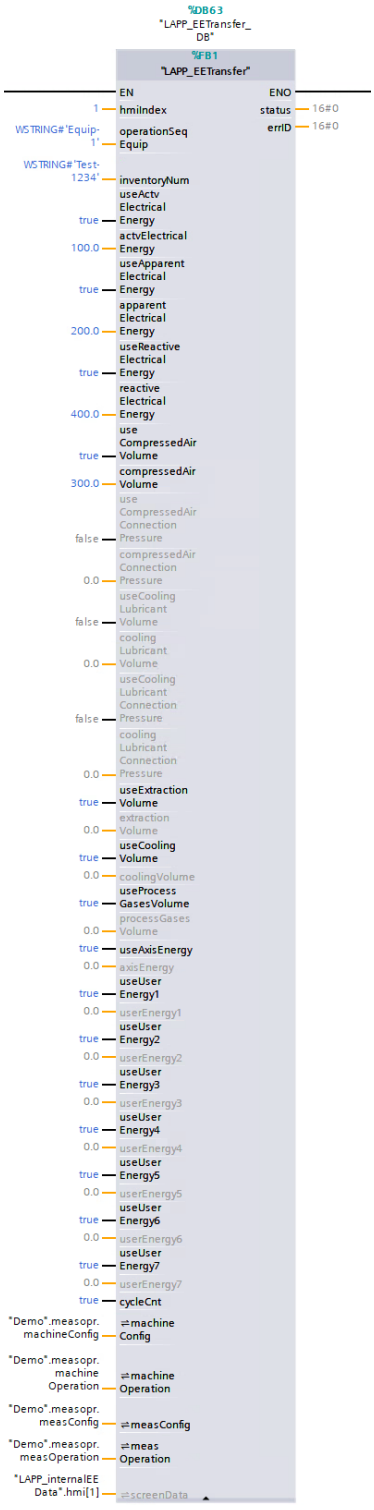


Figure 3-63 "LAPP_EETransfer"

Parameters

Table 3-27 Parameters of "LAPP_EETransfer"

Name	Declaration	Type	Standard	Description
hmi	Input	LAPP_typeIntlLicense	Configure the HMI Index which is defined while setting up the HMI device. **Note**: HMI index value cannot be 0. It should match with HMI index value used in License DB ("LAPP_InternalLicense").	Configure the HMI Index which is defined while setting up the HMI device. **Note**: HMI index value cannot be 0. It should match with HMI index value used in License DB ("LAPP_license").
operationSeqEquip	Input	WString[10]	---	Operation sequence of the equipment
inventoryNum	Input	WString[20]	---	Inventory number
useActvElectricalEnergy	Input	Bool	---	Activate electrical energy - active energy
actvElectricalEnergy	Input	LReal	---	Electrical energy - active energy [kWh]
useApparentElectricalEnergy	Input	Bool	---	Activate electrical energy - apparent energy
apparentElectricalEnergy	Input	LReal	---	Electrical energy - apparent energy [kVAh]
useReactiveElectricalEnergy	Input	Bool	---	Activate electrical energy - reactive energy
reactiveElectricalEnergy	Input	LReal	---	Electrical energy - reactive energy [kvarh]
useCompressedAirVolume	Input	Bool	---	Activate compressed air - volume
compressedAirVolume	Input	Real	---	Compressed air - volume [normally m^3]
useCompressedAirConnectionPressure	Input	Bool	---	Activate compressed air - supply pressure
compressedAirConnectionPressure	Input	Real	---	compressed Air Connection Pressure
useCoolingLubricantVolume	Input	Bool	---	Activate cooling lubricant - volume
coolingLubricantVolume	Input	LReal	---	Cooling lubricant - volume [l]
useCoolingLubricantConnectionPressure	Input	Bool	---	Activate cooling lubricant - supply pressure
coolingLubricantConnectionPressure	Input	Real	---	Cooling lubricant - supply pressure [bar]

Name	Declaration	Type	Standard	Description
useExtractionVolume	Input	Bool	---	Activate extraction - volume
extractionVolume	Input	LReal	---	Extraction - volume [l]
useCoolingVolume	Input	Bool	---	Activate cooling - volume
coolingVolume	Input	LReal	---	Cooling - volume [l]
useProcessGasesVolume	Input	Bool	---	Activate process gases - volume
processGasesVolume	Input	LReal	---	Process gases - volume [m³]
useAxisEnergy	Input	Bool	---	Activate axis energy
axisEnergy	Input	LReal	---	Axis energy
useUserEnergy1-7	Input	Bool	---	Activate user specified energy form 1-7
userEnergy1-7	Input	Bool	---	User specified energy form 1-7
cycleCnt	Input	Bool	---	Increase cycle clock for rising edge
machineConfig	InOut	EnS_ EEm_ typeConfig	EEmMachine.machine Configuration	Configuration data of the machine Note: You should create a data block with data type "LAPP_typeEEmachineConfig" and assign the parameter from this block.
machineOperation	InOut	EnS_ EEm_typeOperation	EEmMachine.machineOperation	Operating data of the machine Note: You should create a data block with data type "LAPP_typeEEmachineConfig" and assign the parameter from this block.
measConfig	InOut	Array of EnS_EEm_typeMeasConfig	EEmMachine.measurements Configuration	Operating data of the machine Note: You should create a data block with data type "LAPP_typeEEmachineConfig" and assign the parameter from this block.
measOperation	InOut	Array of EnS_EEm_typeMeasOperation	EEmMachine.measurements Operation	Operating data of the measuring points Note: You should create a data block with data type "LAPP_typeEEmachineConfig" and assign the parameter from this block.

Name	Declaration	Type	Standard	Description
screenData	InOut	LAPP_typeIntlH-miEEData	---	Interface the HMI tag from "LAPP_InternalEEDa-ta.hmi[y]". **Note**: "LAPP_InternalEEDa-ta" is an internal DB. Modi-fy the array value to in-crease the HMI device sup-port. For more informa-tion, refer to Configuring PLC blocks for different HMI.
status	Output	Word	---	Displays the status codes of the block. For more in-formation, refer to Status codes (Page 112).
errID	Output	Word	---	Displays the error codes of the block. For more infor-mation, refer to "Informa-tion System > Program-ming a PLC > Instructions > Instructions (S7-1200, S7-1500) > Basic instruc-tions (S7-1200, S7-1500) > Program control opera-tions (S7-1500) > Runtime control (S7-1500) > GET_ERR_ID" instruction in TIA portal online help.

y- indicates HMI device number.

For more information on the description of all other values "Programming a PLC > Instructions > Instructions (S7-1200, S71500) > Optional packages (S7-1200, S7-1500) > Energy efficiency instructions" in TIA Portal online help.

3.12.2.3 Status codes

List

Table 3-28 Description of status codes of "LAPP_EETransfer"

Status code	Description
16#7000	Block is in initialized state.
16#7002	Block is in execution state and has no errors.
16#8200	Faulty license.
16#8201	Invalid HMI index.
16#8202	Invalid number of energy types.
16#8203	Invalid array of lower bounds.
16#8204	Invalid array of upper bounds.
16#8400	No connection to HMI.

Note

For more information on additional status codes of "LAPP_EETransfer" refer to Parameter status of EnS_EEm_Calc in TIA Portal online help.

3.12.3 Interface

3.12.3.1 Overview

You can use "LAHP_Interface" screen to view the status of the inputs and outputs for the selected interface as shown:

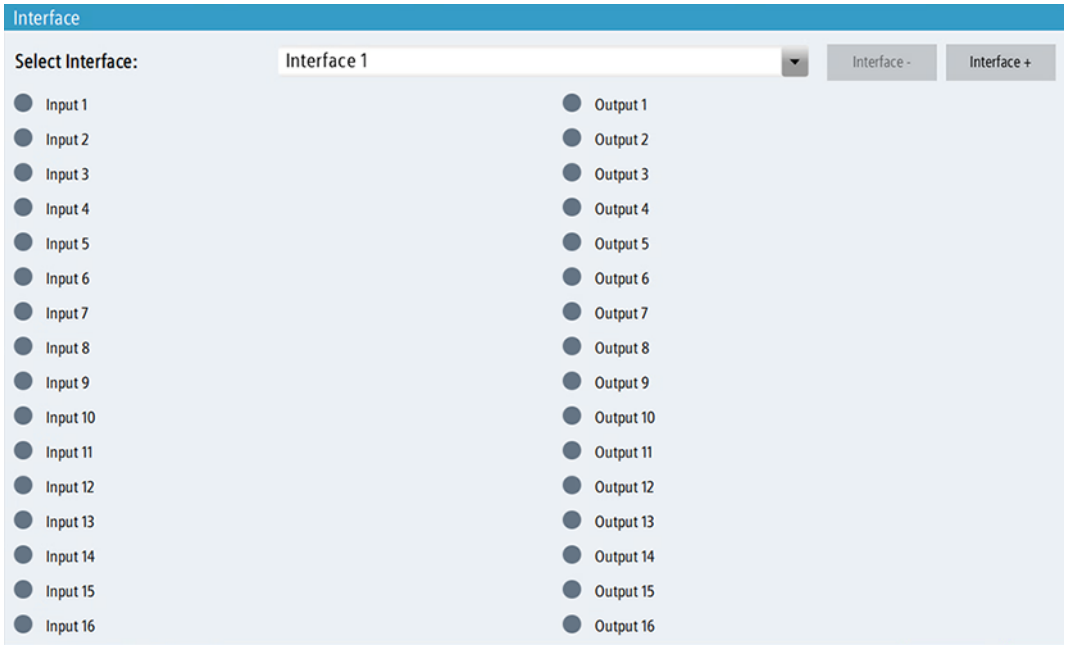


Figure 3-64 LAHP_Interface

You need to map the IO parameters to "LAPP_Interface" block for the "LAHP_Interface" screen to work as intended.

You need to configure the following for "LAHP_Interface" screen to work as intended:

- Block configuration (Page 113) in Step 7

For more information on the configuring the block, refer to AUTOHOTSPOT.

- Text list configuration in WinCC Unified

3.12.3.2 Block interface - LAPP_Interface

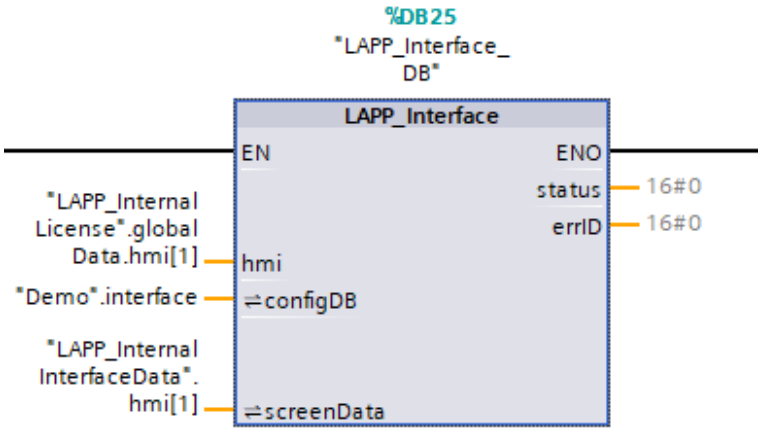


Figure 3-65 "LAPP_Interface"

Parameters

Table 3-29 Parameters of "LAPP_Interface"

Name	Declaration	Type	Description
hmi	Input	LAPP_typeIntlLicense	Configure the HMI Index which is defined while setting up the HMI device. **Note**: HMI index value cannot be 0. It should match with HMI index value used in License DB ("LAPP_InternalLicense").
ConfigDB	InOut	Array of LAPP_typeIntf-Config	Interface the PLC tag with LAPP_typeIntfConfig data type.
screenData	InOut	LAPP_typeIntlHmiIntf	Interface the HMI tag from "LAPP_InternalInterfaceData.hmi[y]". **Note**: "LAPP_InternalInterfaceData" is an internal DB. Modify the array value to increase the HMI device support. For more information, refer to Configuring PLC blocks for different HMI.
status	Output	Word	Displays the status codes of the block. For more information, refer to Status codes (Page 116).
errID	Output	Word	Displays the error codes of the block. For more information, refer to "Information System > Programming a PLC > Instructions > Instructions (S7-1200, S7-1500) > Basic instructions (S7-1200, S7-1500) > Program control operations (S7-1500) > Runtime control (S7-1500) > GET_ERR_ID" instruction in TIA portal online help.

y- indicates HMI device number.

3.12.3.3 Data type

LAPP_typeIntfConfig

Table 3-30 Structure of data type "LAPP_typeIntfConfig"

Name	Type	Description
in	Word	Input interface
out	Word	Output interface

Example

In the below image, "interface" tag is created with "LAPP_typeIntfConfig" with array value [0..9].

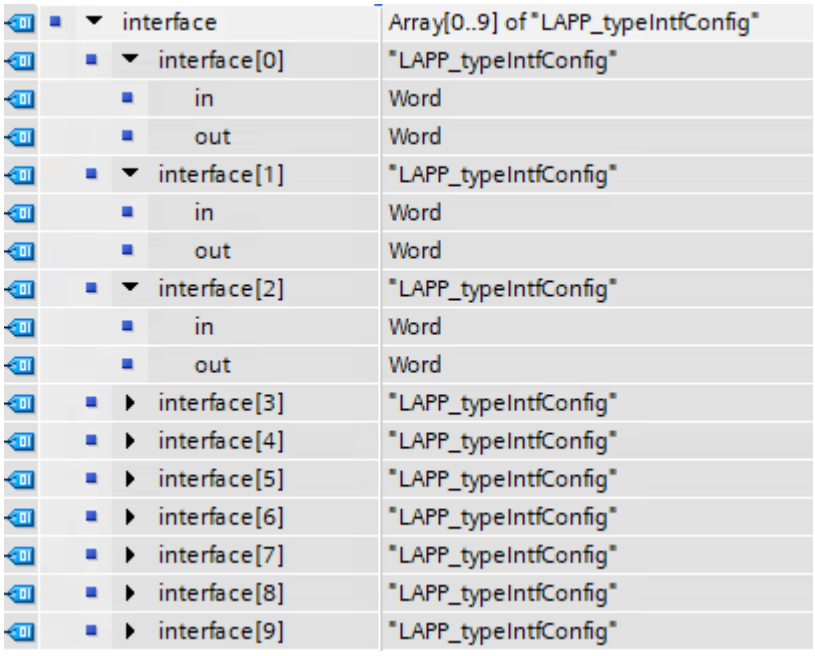
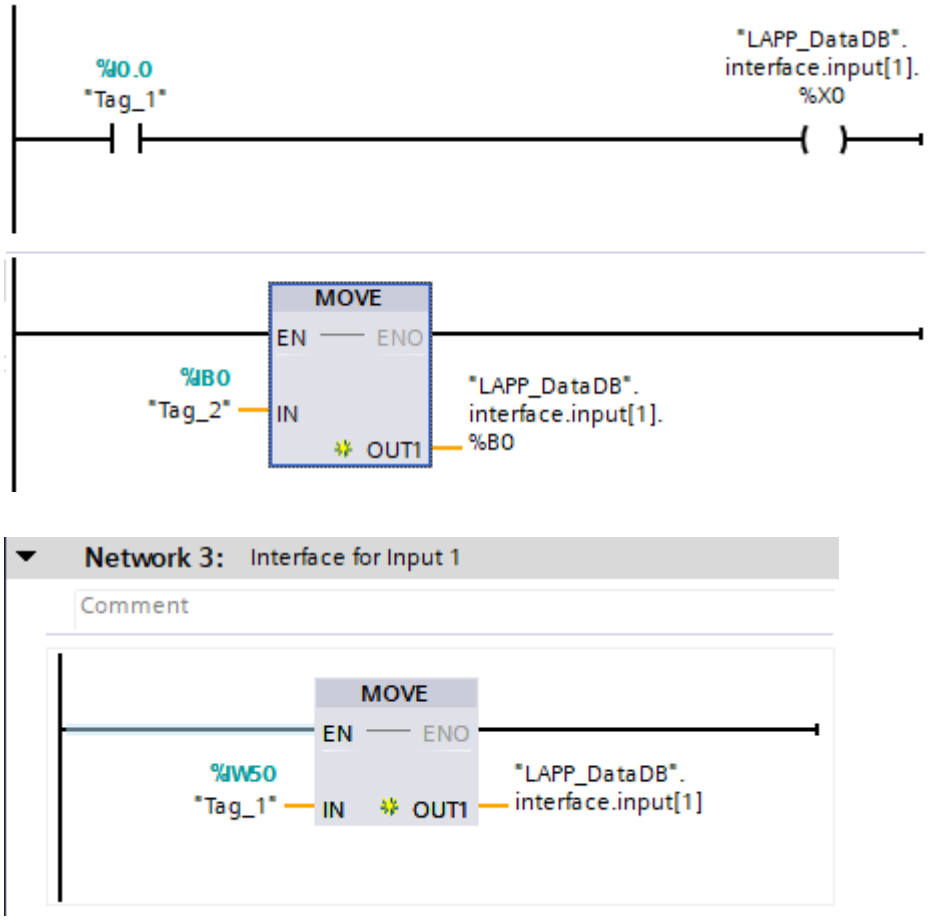


Figure 3-66 Interface tag

- interface[0] - Input and output for interface 1
- interface[1] - Input and output for interface 2
  .
  .
- interface[9] - Input and output for interface 10

3.12.3.4 Example

You can assign the inputs to the interface screen as shown



3.12.3.5 Status codes

List

Table 3-31 Description of status codes of "LAPP_Interface"

Status code	Description
16#7000	Block is in initialized state.
16#7002	Block is in execution state and has no errors.
16#8200	Faulty license
16#8201	Invalid HMI index.
16#8400	No connection to HMI.

3.12.4 Enhanced interface

3.12.4.1 Overview

You can use "LAHP_EnhancedInterface" screen to view the status of the parameters being monitored for the selected interface as shown:

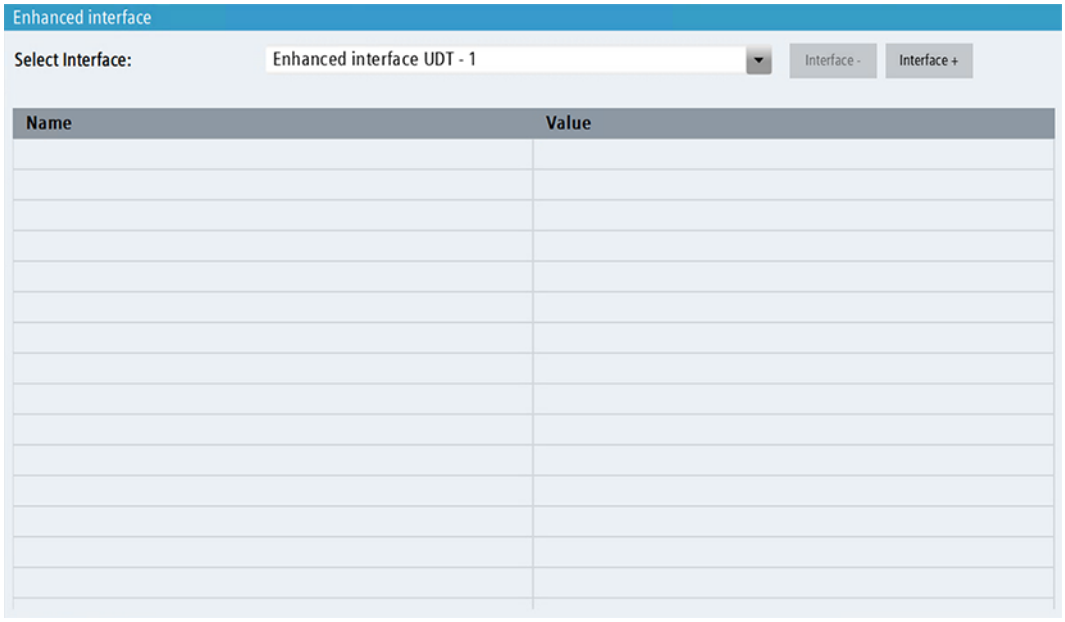


Figure 3-67 "LAHP_EnhancedInterface"

The "LAHP_EnhancedInterface" can be configured by the user to display and monitor the parameters that they require.

Configuration of the interface screen involves the following steps:

1. Creation of user defined datatype (Page 118)
2. Creation of a tags (Page 122)
3. Creation of parameter set types (Page 126)

3.12.4.2 Creation of UDT

Procedure

1. Right click on "PLC data types" and click on "Add new data type".

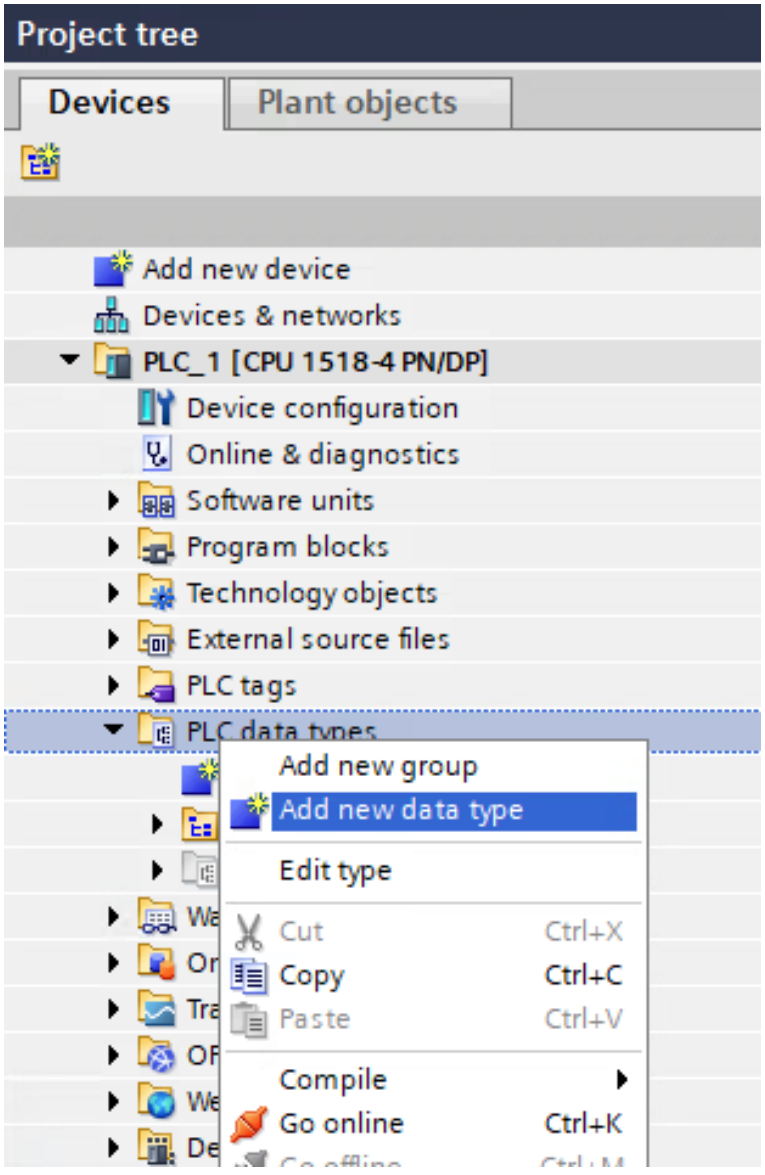


Figure 3-68 Add datatype

2. A data type with default name "User_data_type_1" is created. Rename the data type based on the requirement.

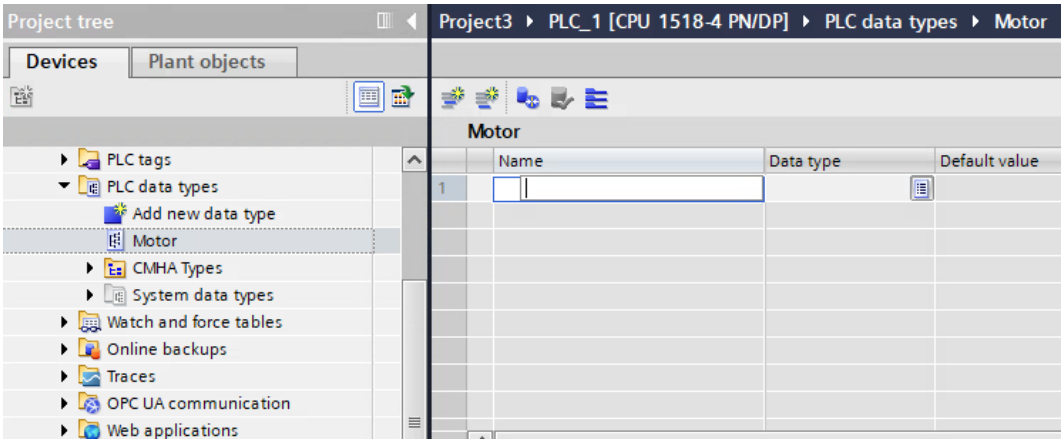


Figure 3-69 Data type created

3. Add the required input parameters to be monitored and displayed on the runtime screen.

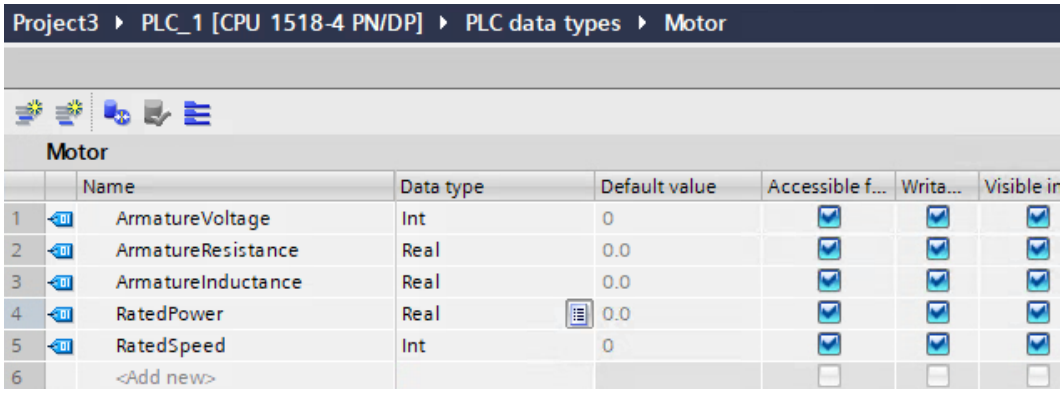


Figure 3-70 Input parameters

4. Drag and drop the created UDTs to "Libraries > Project library > Types".

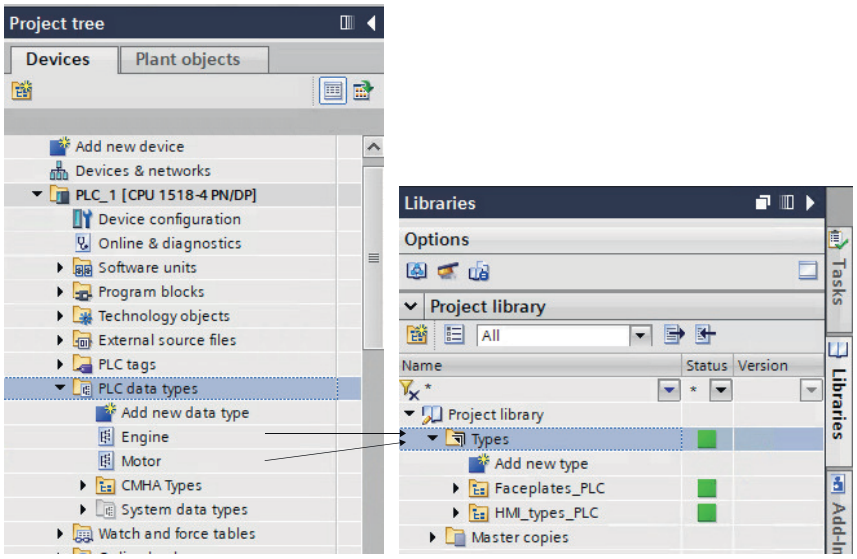


Figure 3-71 Project library

"Add type" pop-up opens. Enter the necessary details and click "OK".

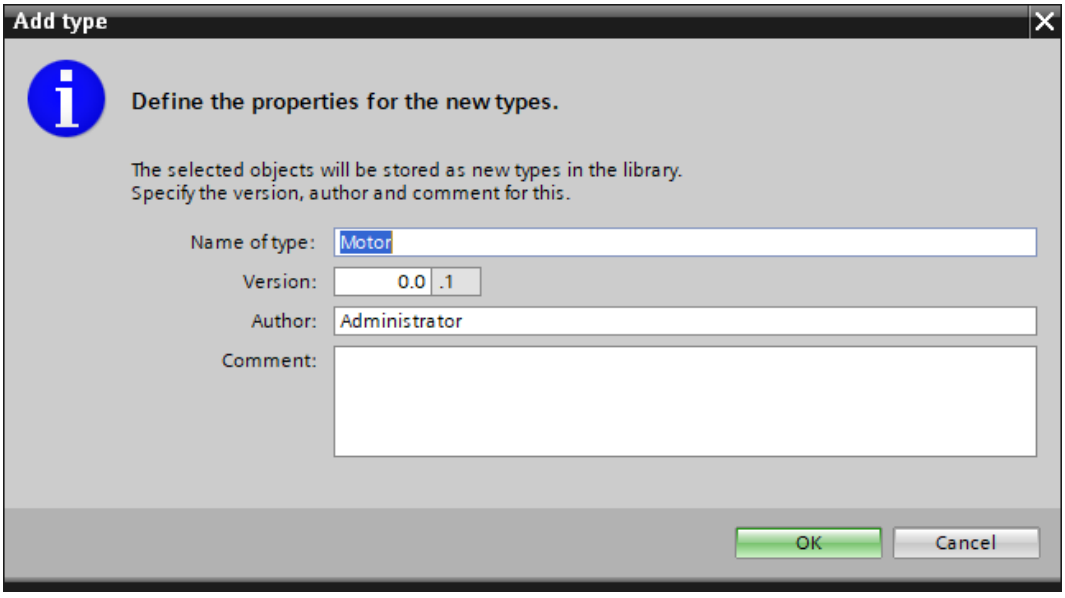


Figure 3-72 Add type

Result

User defined data type is created.

3.12.4.3 Creation of tags

Procedure

PLC data block

1. Right-click "Program blocks" and click "Add new block"
   "Add new block" window opens

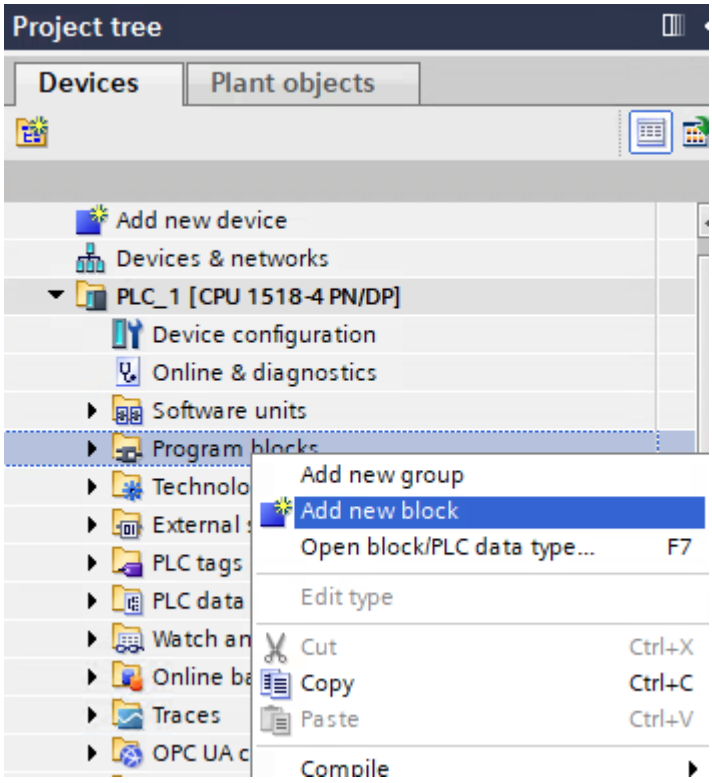


Figure 3-73 Add new block

2. Click "Data block" and enter the name required for data block.

3. In "Type" field, select "Global DB"

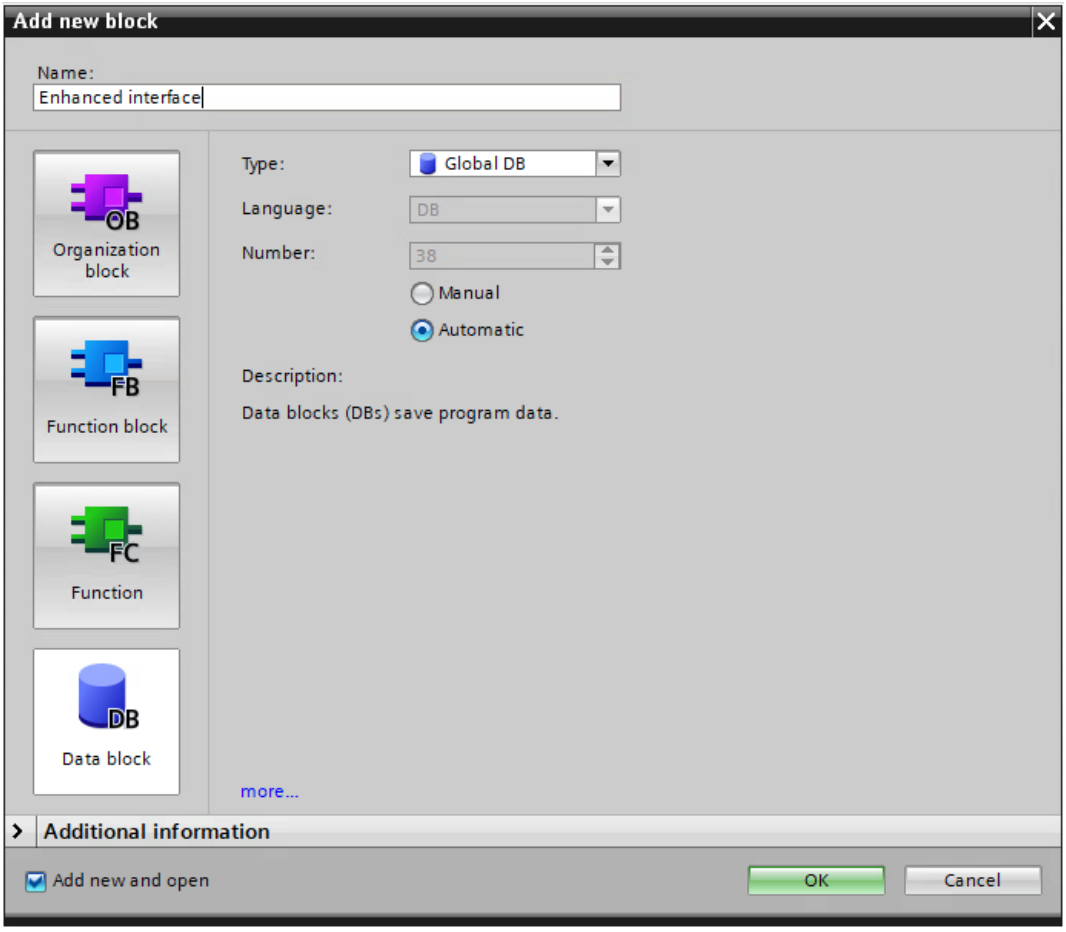


Figure 3-74 Global DB

4. Click "OK"

5. Provide the required name for the parameter and data type created in "Creation of UDT (Page 118)" as shown below.

Enhanced interface

	Name	Data type	Start value	Retain	Accessible f...	Writa...	Visible in ...
1	Static						
2	MotorParameter	"Motor"					
3	ArmatureVoltage	Int	0				
4	ArmatureResistance	Real	0.0				
5	ArmatureInductance	Real	0.0				
6	RatedPower	Real	0.0				
7	RatedSpeed	Int	0				
8	EngineParts	"Engine"					
9	EngineCount	Int	0				
10	PistonName	WString	WSTRING#''				
11	EngineValveOpen	Bool	false				
12	<Add new>						

Figure 3-75 Data block

HMI tag

1. Navigate to "IPC device > HMI tags" folder.
2. Select "Default tag table".
3. Click "Add new" and drag and drop the data block created in the above procedure.

Tag table_1

	Name	Data type	Connection	PLC name	PLC tag
	Enhanced interface_EngineParts	Engine	S7Connection	PLC_1	"Enhanced interface".EngineParts
	EngineCount	Int	S7Connection	PLC_1	"Enhanced interface".EngineParts.EngineCount
	PistonName	WString	S7Connection	PLC_1	"Enhanced interface".EngineParts.PistonName
	EngineValveOpen	Bool	S7Connection	PLC_1	"Enhanced interface".EngineParts.EngineValveOpen
	Enhanced interface_MotorParameter	Motor	S7Connection	PLC_1	"Enhanced interface".MotorParameter
	ArmatureVoltage	Int	S7Connection	PLC_1	"Enhanced interface".MotorParameter.ArmatureVolt...
	ArmatureResistance	Real	S7Connection	PLC_1	"Enhanced interface".MotorParameter.ArmatureResi...
	ArmatureInductance	Real	S7Connection	PLC_1	"Enhanced interface".MotorParameter.ArmatureInd...
	RatedPower	Real	S7Connection	PLC_1	"Enhanced interface".MotorParameter.RatedPower
	RatedSpeed	Int	S7Connection	PLC_1	"Enhanced interface".MotorParameter.RatedSpeed

Figure 3-76 HMI tag table

Parameter set ID and Job ID

You can find the default Parameter set IDs and Job IDs under "IPC device > HMI tags > CMHA HMI tags > HardwareDiagnostic > LAHP_EnhancedInterface". You can use them or create new IDs.

LAHP_EnhancedInterface

Name	Data type	Connection	PLC name	PLC tag
currParameterSetID	UInt	<Internal tag>		<Undefined>
currParameterSetTypeID	UInt	<Internal tag>		<Undefined>
intfScreenActv	Bool	<Internal tag>		<Undefined>
jobID1	Int	<Internal tag>		<Undefined>
jobID2	Int	<Internal tag>		<Undefined>
jobID3	Int	<Internal tag>		<Undefined>
parameterID1	Int	<Internal tag>		<Undefined>
parameterID2	Int	<Internal tag>		<Undefined>
parameterID3	Int	<Internal tag>		<Undefined>
parameterSetTypeName	WString	<Internal tag>		<Undefined>
selectedIntf	USInt	<Internal tag>		<Undefined>
<Add new>				

Figure 3-77 Default parameter set ID and job ID

Follow the steps below to create new Parameter set IDs and Job IDs.

1. Navigate to IPC device >> “HMI tags” folder.
2. Select “Default tag table”.
3. Click “Add new” and drag and create HMI tags "jobID" and "parameterID" as shown below.

Tag table_2

	Name	Data type	Connection	PLC name	PLC tag
	jobID4	Int	<Internal tag>		<Undefined>
	jobID5	Int	<Internal tag>		<Undefined>
	jobID6	Int	<Internal tag>		<Undefined>
	jobID7	Int	<Internal tag>		<Undefined>
	parameterID4	Int	<Internal tag>		<Undefined>
	parameterID5	Int	<Internal tag>		<Undefined>
	parameterID6	Int	<Internal tag>		<Undefined>
	parameterID7	Int	<Internal tag>		<Undefined>
	<Add new>				

Figure 3-78 Job ID and parameter ID

Note

Ensure to follow the format shown below for job and parameter IDs

- Parameter ID - parameterID4, parameterID5, parameterID6
- Job ID - jobID4, jobID5, jobID6

The numbering for the newly created IDs should be from 4 as 1-3 are provided for the defaults IDs.

Result

HMI tags are created to be used in the Parameter set types.

3.12.4.4 Creation of parameter set types

Procedure

1. Right click on "Parameter set types" and click on "Add new parameter set type".

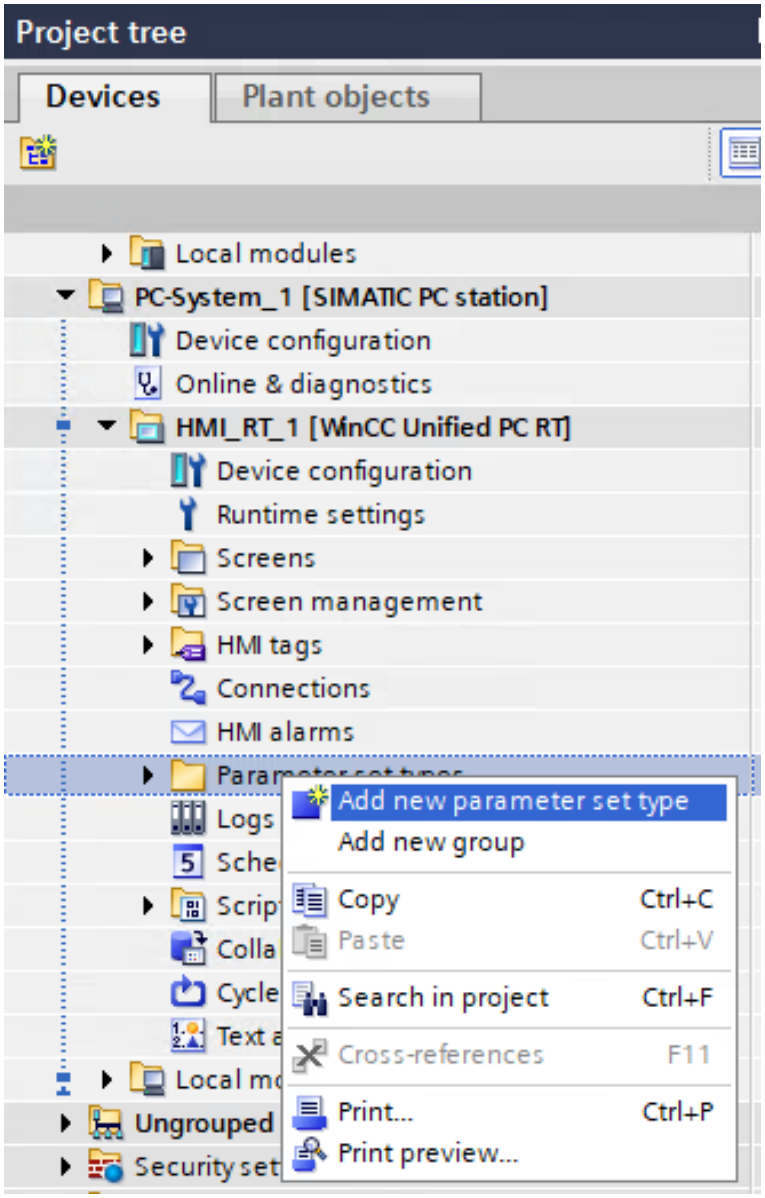


Figure 3-79 Add parameter set type

2. A data type with default name "Parameter set type_1" is created. Rename the data type based on the requirement. Ensure that the name matches with the name provided in the textlist "LAHP_Interface".

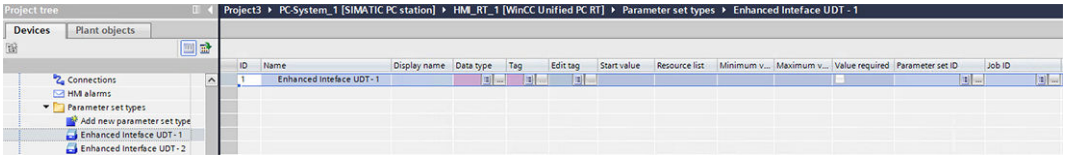


Figure 3-80 Parameter set type

3. Provide the "Data type", "Tag", "Parameter set ID" and "Job ID" as shown below.

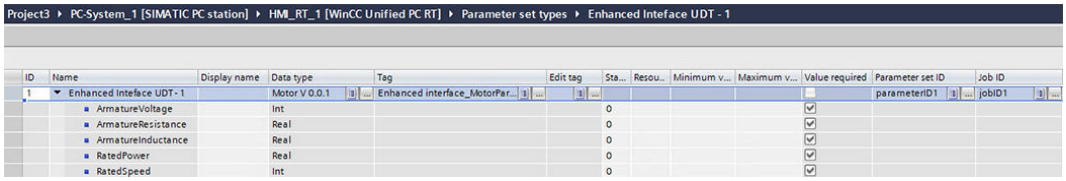


Figure 3-81 Enhanced interface UDT-1

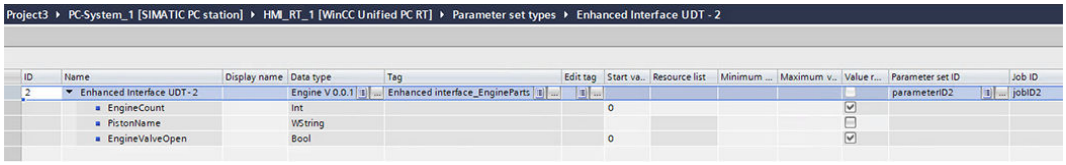


Figure 3-82 Enhanced interface UDT-2

Note

- "Parameter set ID" and "Job ID" are mandatory in parameter set types to read the values from PLC and display them in the HMI.
- Ensure that the parameter set ID and job ID are unique to each Parameter set type.
- Ensure to follow the restrictions while creating Parameter set types. For more information refer to "Configuring parameter sets" in SIMATIC HMI WinCC Unified V19. (https://support.industry.siemens.com/cs/mdm/109828368?c=177361948043&lc=en-US)

4. To add translated texts, in the Parameter set type, add the "Display name".
Select the "Display name" and navigate to "Properties > Texts" and add the translated texts for the respective languages.

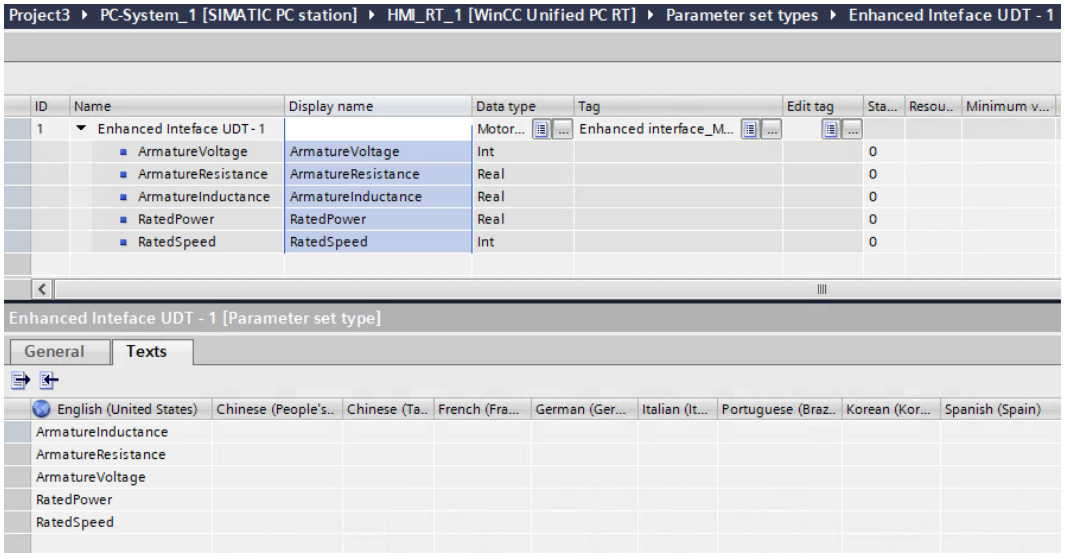


Figure 3-83 Translated text

3.12.5 Motor starter

3.12.5.1 Overview

The motor starter blocks writes data to WinCC screens for the motor starter diagnostics. You must call the "LAPP_MotorStarter" block.

The "LAHP_MotorStarter" runtime screen is shown below.

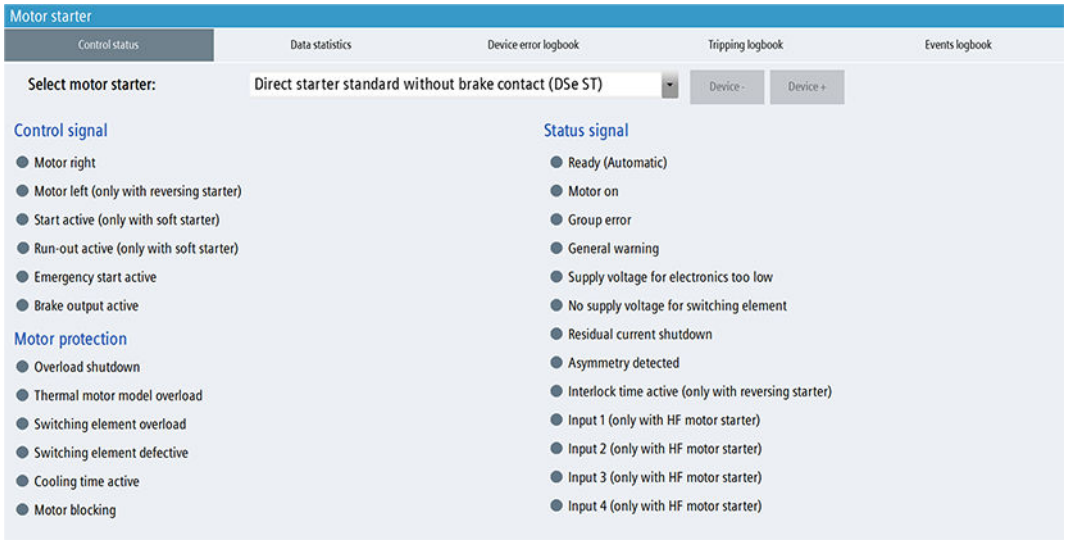


Figure 3-84 Motor starter

The "LAPP_typeStarterConfig" data type is used to configure the motor starter blocks.

You need to configure the following for "LAHP_MotorStarter" screen to work as intended:

- Block configuration (Page 130) in Step 7
  For more information on the configuring the block, refer to AUTOHOTSPOT.
- Text list configuration in WinCC Unified

3.12.5.2 Block interface - LAPP_MotorStarter

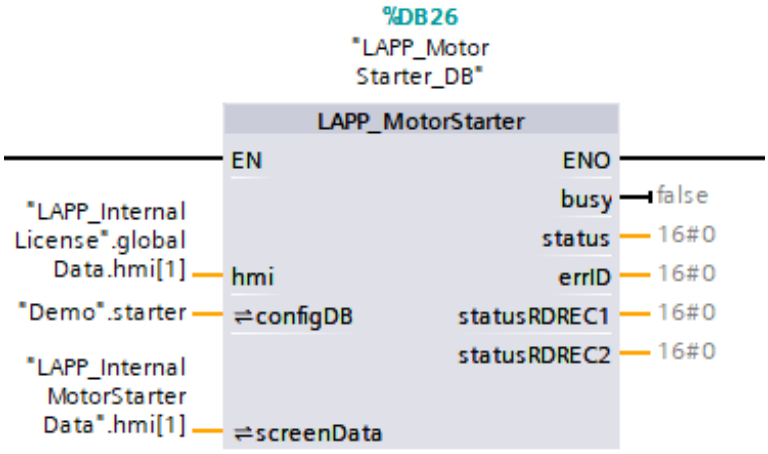


Figure 3-85 "LAPP_MotorStarter"

Parameters

Table 3-32 Parameters of "LAPP_MotorStarter"

Name	Declaration	Type	Description
hmi	Input	LAPP_typeIntlLicense	Configure the HMI Index which is defined while setting up the HMI device. **Note**: HMI index value cannot be 0. It should match with HMI index value used in License DB ("LAPP_InternalLicense").
configDB	InOut	Array of LAPP_typeStarterConfig	The database interfaced to configure the motorstater device
screenData	InOut	LAPP_typeIntlHmiMotorStarter	Interface the HMI tag from "LAPP_InternalMotorStarterData.hmi[y]". **Note**: "LAPP_InternalMotorStarterData" is an internal DB. Modify the array value to increase the HMI device support. For more information, refer to Configuring PLC blocks for different HMI.
busy	Output	Bool	"TRUE" communication with the drive
status	Output	Word	Displays the status codes of the blocks. For more information, refer to Status codes (Page 131).

Name	Declaration	Type	Description
errID	Output	Word	Displays the error codes of the block. For more information, refer to "Information System > Programming a PLC > Instructions > Instructions (S7-1200, S7-1500) > Basic instructions (S7-1200, S7-1500) > Program control operations (S7-1500) > Runtime control (S7-1500) > GET_ERR_ID" instruction in TIA portal online help.
statusRDREC1	Output	DWord	Status of the RDREC instruction. For more information, refer to RDREC: Read data record TIA Portal online help.
statusRDREC2	Output	DWord	Status of the RDREC instruction. For more information, refer to RDREC: Read data record TIA Portal online help.

y- indicates HMI device number.

3.12.5.3 Data type

LAPP_typeStarterConfig

You must specify the hardware address of the motor starter that you want to diagnose in the "LAPP_MotorStarter" block. Up to 128 motor starters can be entered. The index of the field corresponds to the index of the text list in WinCC.

Name	Type	Description
deviceType	Byte	Motor starter type. 0 - ET200pro 1 - ET200SP
hwID	HW_IO	Hardware identification of the DP slave, taken from "Devices & networks"

3.12.5.4 Status codes

List

Table 3-33 Description of the status codes of "LAPP_MotorStarter" block

Status code	Description
16#7000	Block is in initialized state.
16#7002	Block is in execution state and has no errors.
16#8200	Faulty license.
16#8201	Invalid HMI index.

Status code	Description
16#8202	Invalid device type.
16#8203	Control status screen is not active.
16#8204	Invalid input array configuration.
16#8600	Status of the read record block.
16#8601	Status of the read record block.
16#8400	No connection to HMI.

3.12.6 PLC system data

3.12.6.1 Overview

The "LAPP_PLCSystemData" block reads the PLC system data on the first scan of PLC startup.

The "LAHP_PLCSystemData" runtime screen is shown below.



Figure 3-86 PLC system data

For more information on the configuring the block, refer to AUTOHOTSPOT.

3.12.6.2 Block interface - LAPP_PLCSystemData

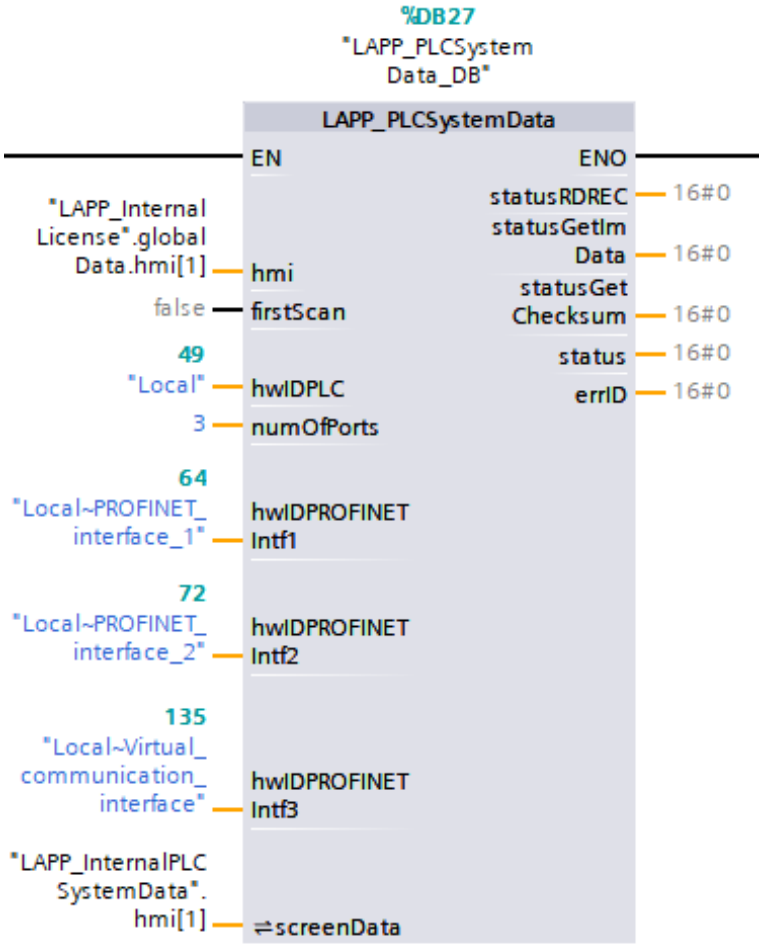


Figure 3-87 "LAPP_PLCSystemData"

Parameters

Table 3-34 Parameters of "LAPP_PLCSystemData"

Name	Declaration	Type	Description
hmi	Input	LAPP_typeIntlLicense	Configure the HMI Index which is defined while setting up the HMI device. **Note**: HMI index value cannot be 0. It should match with HMI index value used in License DB ("LAPP_InternalLicense").
firstScan	Input	Bool	Startup bit. For more information to enable first scan, refer to Enabling first scan (Page 134).
hwIDPLC	Input	HW_IO	PLC hardware ID

Name	Declaration	Type	Description
numOfPorts	Input	Int	Number of ports configured (Minimum 1 and maximum 3)
hwIDProfinetIntf1	Input	HW_IO	Hardware ID of the PROFINET interface 1
hwIDProfinetIntf2	Input	HW_IO	Hardware ID of the PROFINET interface 2
hwIDProfinetIntf3	Input	HW_IO	Hardware ID of the PROFINET interface 3
screenData	InOut	LAPP_typeIntlHmiPLC-SystemData	Interface the HMI tag from "LAPP_Internal-PlcSystemData.hmi[y]". **Note**: "LAPP_InternalPlcSystemData" is an internal DB. Modify the array value to increase the HMI device support. For more information, refer to Configuring PLC blocks for different HMI.
statusRDREC	Output	DWord	Status of the RDREC instruction. For more information on RDREC instruction, refer to RDREC: Read data record in TIA Portal online help.
statusGetImData	Output	Word	Status of the GET_IM_DATA instruction. For more information on GET_IM_DATA instruction, refer to Get_IM_Data: Reading identification and maintenance data in TIA Portal online help.
statusGetChecksum	Output	Word	Status of the GetChecksum instruction. For more information on GetChecksum instruction, refer to GetChecksum: Read out checksum in TIA Portal online help.
status	Output	Word	Displays the status codes of the block. For more information, refer to Status codes (Page 135).
errID	Output	Word	Displays the error codes of the block. For more information, refer to "Information System > Programming a PLC > Instructions > Instructions (S7-1200, S7-1500) > Basic instructions (S7-1200, S7-1500) > Program control operations (S7-1500) > Runtime control (S7-1500) > GET_ERR_ID" instruction in TIA portal online help.

y- indicates HMI device number.

3.12.6.3 Enabling first scan

Procedure

To activate first scan bit, follow the below steps:

1. In the "Network view", click PLC.
2. Navigate to "Properties > General" tab in the inspector window.

3. Click "System and clock memory".
4. Enable the "Enable the use of system memory byte" checkbox in "System memory bits" as shown below:

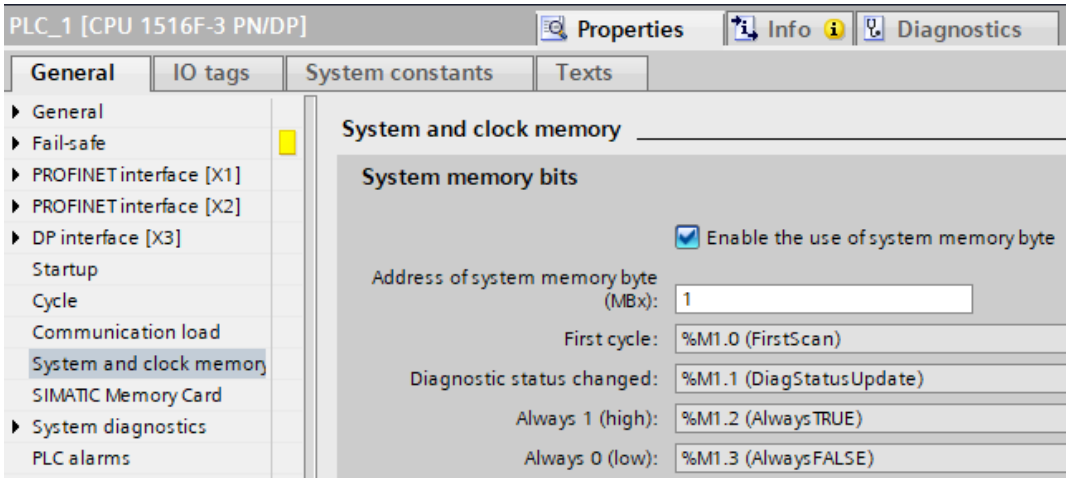


Figure 3-88 System memory bits

3.12.6.4 Status codes

List

Table 3-35 Description of status codes "LAPP_PLCSystemData"

Status code	Description
16#7000	Block is in initialized state.
16#7002	Block is in execution state and has no errors.
16#8200	Faulty license.
16#8201	Invalid HMI index.
16#8202	PLC model not found.
16#8203	PLC configuration port error.
16#8400	No connection to HMI.

3.12.7 RFID

3.12.7.1 Overview

The "LAPP_RFID" block supplies the WinCC screens for the RFID diagnostics. The displayed data is read in via the interface. You need to call this block cyclically once for each configured ident device, whereby the selected device parameter corresponds to the corresponding values from the WinCC text list "LAHP_IdentDeviceName".

The "LAHP_RFID" runtime screen is shown below.

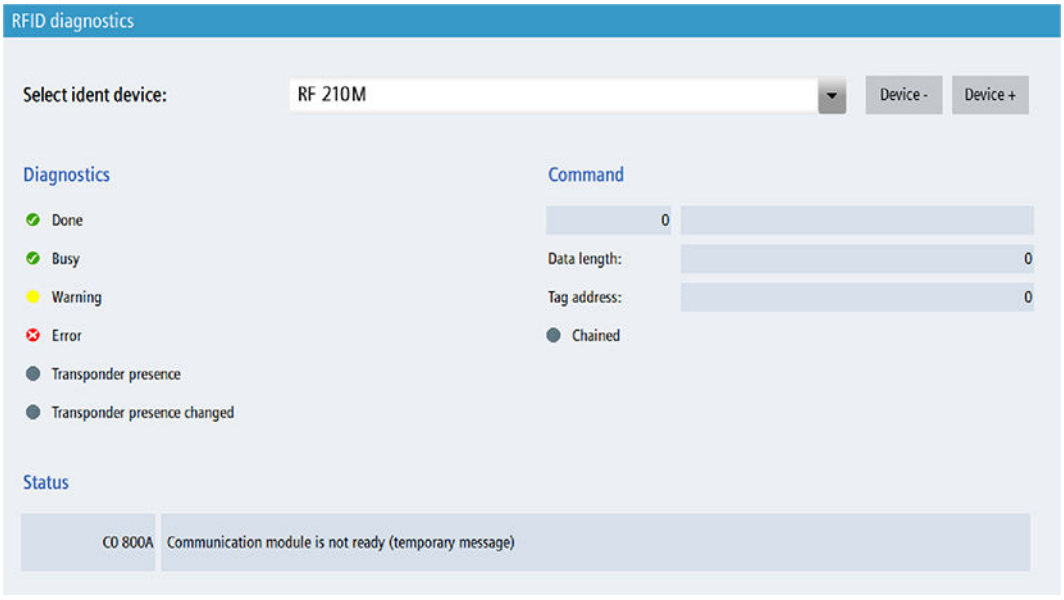


Figure 3-89 RFID diagnostics

You need to configure the following for "LAHP_RFID" screen to work as intended:

- Block configuration (Page 137) in Step 7
  For more information on the configuring the block, refer to AUTOHOTSPOT.
- Text list configuration in WinCC Unified.

3.12.7.2 Block interface - LAPP_RFID

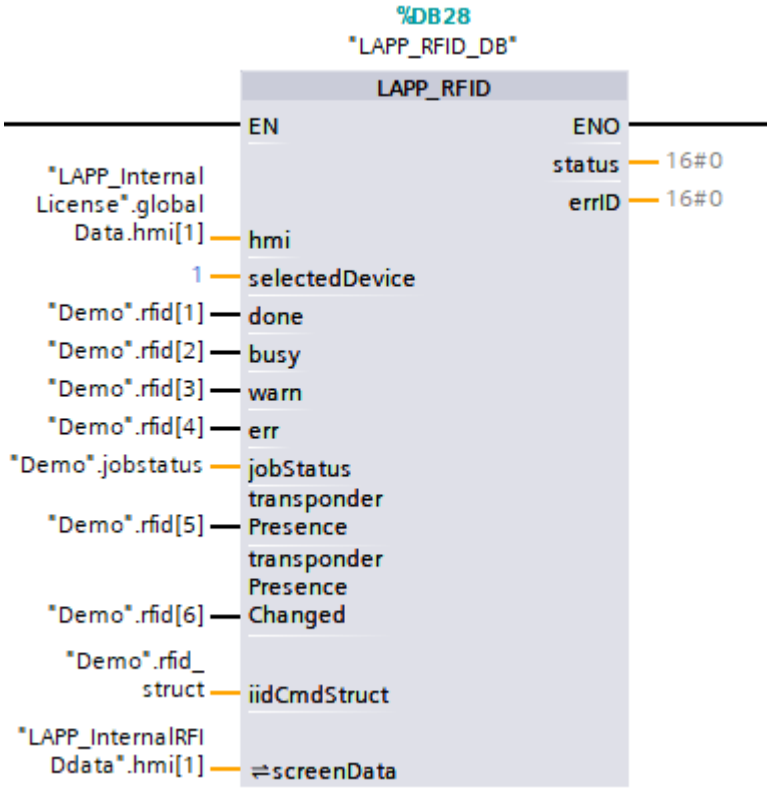


Figure 3-90 "LAPP_RFID"

Parameters

Table 3-36 Parameters of "LAPP_RFID"

Name	Declaration	Type	Description
hmi	Input	LAPP_typeIntlLicense	Configure the HMI Index which is defined while setting up the HMI device. **Note**: HMI index value cannot be 0. It should match with HMI index value used in License DB ("LAPP_InternalLicense").
selectedDevice	Input	UInt	Selecting the ident device: Value from the text list LAHP_IdentDeviceName
done	Input	Bool	Assign the output parameter "done" of the ident blocks or of the ident profile
busy	Input	Bool	Assign the output parameter "busy" of the ident blocks or of the ident profile
warn	Input	Bool	Assign the output parameter "warning" of the ident blocks or of the ident profile
err	Input	Bool	Assign the output parameter "error" of the ident blocks or of the ident profile

Name	Declaration	Type	Description
jobStatus	Input	DWord	Assign the output parameter "status" of the ident blocks or of the ident profile
transponderPresence	Input	Bool	Assign the output parameter "presence" of the ident blocks or of the ident profile
transponderPresence-Changed	Input	Bool	Assign the output parameter "tpc" of the ident blocks or of the ident profile
iidCmdStruct	Input	IID_CMD_ STRUCT	Assign the input parameter of the current command of the ident blocks or of the ident profile
screenData	InOut	LAPP_typeIntlHmiRFID	Interface the HMI tag from "LAPP_InternalRFIDdata.hmi[y]". **Note**: "LAPP_InternalRFIDdata" is an internal DB. Modify the array value to increase the HMI device support. For more information, refer to Configuring PLC blocks for different HMI.
status	Output	Word	Displays the status codes of the block. For more information, refer to Status codes (Page 138).
errID	Output	Word	Displays the error codes of the block. For more information, refer to "Information System > Programming a PLC > Instructions > Instructions (S7-1200, S7-1500) > Basic instructions (S7-1200, S7-1500) > Program control operations (S7-1500) > Runtime control (S7-1500) > GET_ERR_ID" instruction in TIA portal online help.

y- indicates HMI device number.

3.12.7.3 Status codes

List

Table 3-37 Description of status codes of "LAPP_RFID"

Status code	Description
16#7000	Block is in initialized state.
16#7002	Block is in execution state and has no errors.
16#8200	Faulty license.
16#8201	Invalid HMI index.
16#8202	Device in the screen not selected.
16#8400	No connection to HMI.

3.12.8 Safety

3.12.8.1 Overview

Safety block is used for monitoring the failsafe functions of the PLC.

You need to configure the following for "LAHP_Safety" screen to work as intended:

- Block configuration (Page 138) in Step 7
  For more information on the configuring the block, refer to AUTOHOTSPOT.

3.12.8.2 Block interface - LAPP_Safety

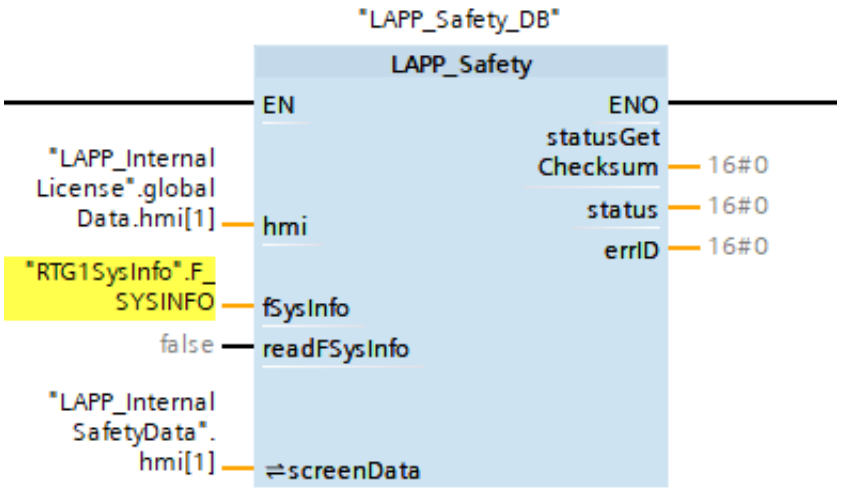


Figure 3-91 "LAPP_Safety"

NOTICE
Data loss Ensure to activate "retain" for safety DB to retain the current values of the DB after every restart. If not retained, DB displays the values since last start up.

Note

TIA Portal Project protection

Recommended to protect the TIA Portal projects against unauthorized access and assign roles with permissions to users or user groups. For more information on how to protect project, Refer section 2.3.4 Import UMC user groups and users to TIA Portal (https://cache.industry.siemens.com/dl/files/337/109780337/att_1105238/v2/109780337_UMC_TIAPortal_WinCCRTAdvanced_ComfortPanels_en_V11.pdf).

Runtime Project protection

- Recommended to protect project against unauthorized access by using PLC access level and Protect confidential data.
- For UCP, ensure to protect access of the control panel by enabling password protection for control panel. For more information, refer section 5.6.3 Control panel access (https://cache.industry.siemens.com/dl/files/257/109773257/att_1019074/v1/hmi_unified_comfort_panels_operating_instructions_enUS_en-US.pdf).

Parameters of "LAPP_Safety"

Name	Declaration	Type	Standard	Description
hmi	Input	LAPP_typeIntl-License	-	Configure the HMI Index which is defined while setting up the HMI device. **Note**: HMI index value cannot be 0. It should match with HMI index value used in License DB ("LAPP_InternalLicense").
fSysInfo	Input	F_SYSINFO	RTG1SysInfo.F_ SYSINFO	S_SYSINFO parameter of the F-runtime group info DB For more information, refer to F-runtime group information DB (S7-1200, S7-1500) in TIA Portal online help. You can view "Step 7 Safety" blocks only when safety function is configured for PLC.
readFSysInfo	Input	Bool	-	Positive edge copies the current data to the acceptance data
screenData	InOut	LAPP_type-IntlHmiSafety	-	Interface the HMI tag from "LAPP_InternalSafetyData.hmi[y]". **Note**: "LAPP_InternalSafetyData" is an internal DB. Modify the array value to increase the HMI device support. For more information, refer to Configuring PLC blocks for different HMI.
statusGetChecksum	Output	Word	-	Status of the GetChecksum instruction

Name	Declaration	Type	Standard	Description
status	Output	Word	-	Displays the status codes of the block. For more information, refer to Status codes (Page 141).
errID	Output	Word	-	Displays the error codes of the block. For more information, refer to "Information System > Programming a PLC > Instructions > Instructions (S7-1200, S7-1500) > Basic instructions (S7-1200, S7-1500) > Program control operations (S7-1500) > Runtime control (S7-1500) > GET_ERR_ID" instruction in TIA portal online help.

y- indicates HMI device number.

3.12.8.3 Status codes

List

Following are the status codes for "LAPP_Safety".

Status code	Description
16#7000	Block is in initialized state.
16#7002	Block is in execution state and has no errors.
16#8200	Faulty license.
16#8201	Invalid HMI index.
16#8202	Checksum provided by the user is not identical to the actual checksum.
16#8400	No connection to HMI.

3.12.9 SINAMICS

3.12.9.1 Overview

The SINAMICS blocks supplies the details for the SINAMICS screens. There are two types of SINAMICS diagnostic blocks:

- SINAMICS diagnostic blocks using EPOS (Page 143): This block is used to read the drive parameters and actual values directly from drive.
  The "LAPP_typeSINAMICSConfig" data type is used to configure the SINAMICS EPOS blocks.
  The SINAMICS EPOS runtime screen is shown below.

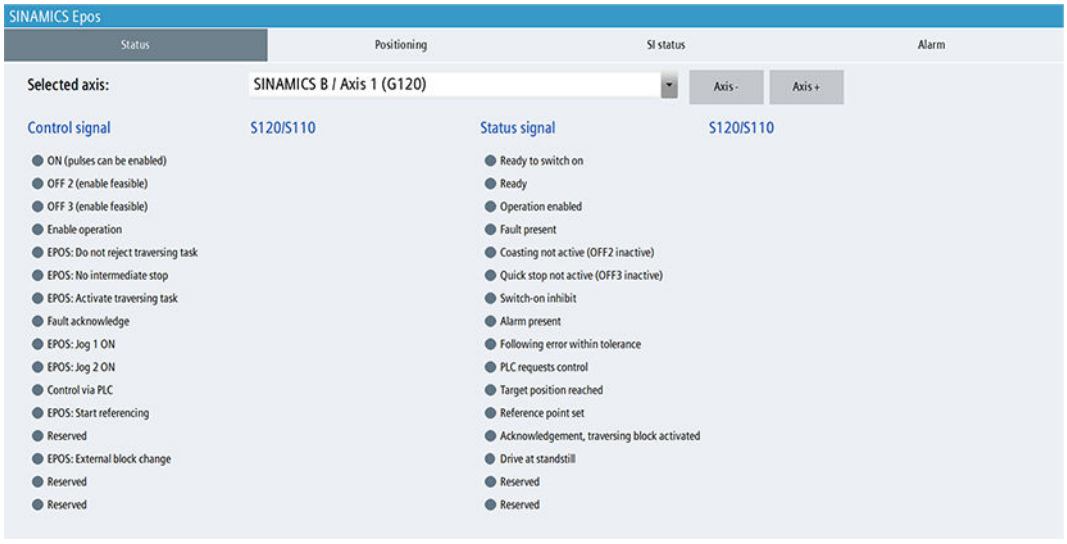


Figure 3-92 SINAMICS EPOS

- SINAMICS diagnostic block using technology objects (Page 145): This block is used to read data of the technology objects. The technology objects that can be used are speed axis, position axis and synchronous axis. The SINAMICS TO runtime screen is shown below.

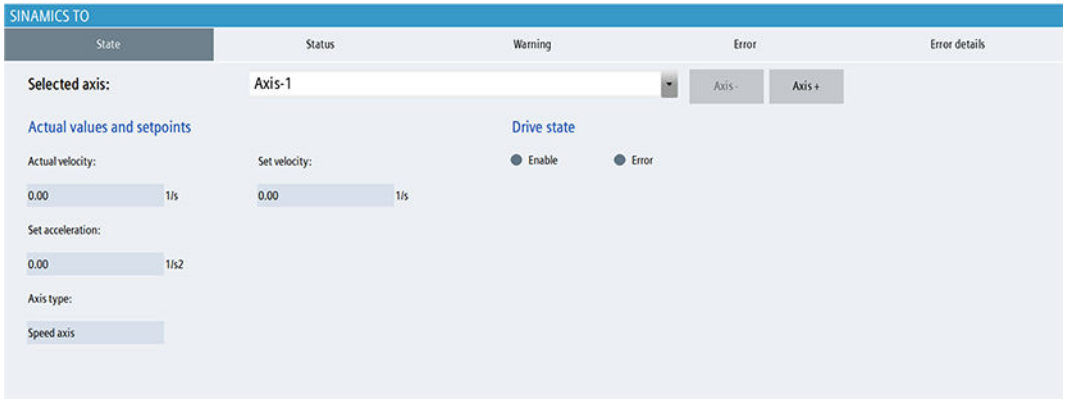


Figure 3-93 SINAMICS TO

The function block must be called once cyclically. The FB call must be enabled via the "driveEnable" parameter.

For more details on configuring the block, refer to AUTOHOTSPOT.

For more information on modifying the Axes names in WinCC Unified refer to Configuring text lists.

3.12.9.2 SINAMICS diagnostic blocks using EPOS

Block interface - LAPP_SINAMICSEpos

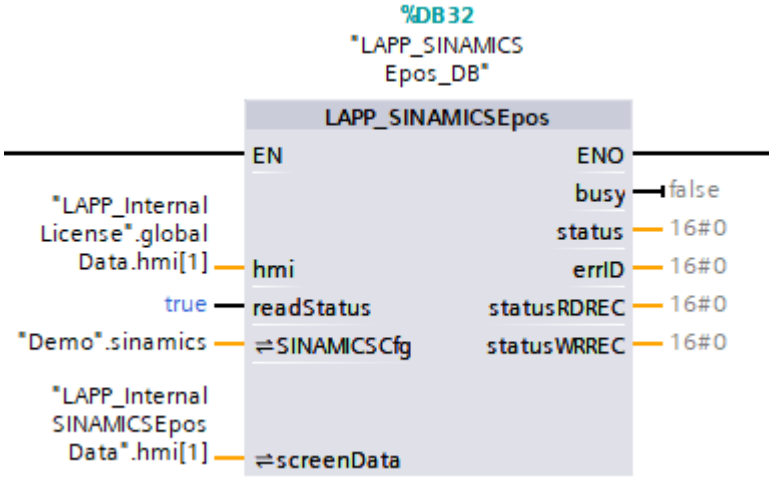


Figure 3-94 "LAPP_SINAMICSEpos"

Table 3-38 Parameters of "LAPP_SINAMICSEpos"

Name	Declaration	Type	Description
hmi	Input	LAPP_typeIntlLicense	Configure the HMI Index which is defined while setting up the HMI device. **Note**: HMI index value cannot be 0. It should match with HMI index value used in License DB ("LAPP_InternalLicense").
readStatus	Input	Bool	"TRUE" enables the communication of the block with the drive.
SINAMICSCfg	InOut	Array of LAPP_typeSINAMICSConfig	The drive objects are configured.
screenData	InOut	LAPP_typeIntlHmiSINAMICSEpos	Interface the HMI tag from "LAPP_InternalSINAMICSEposData.hmi[y]". Note: "LAPP_InternalSINAMICSEposData" is an internal DB. Modify the array value to increase the HMI device support. For more information, refer to Configuring PLC blocks for different HMI.
busy	Output	Bool	"TRUE" communication with the drive.
status	Output	Word	Displays the status codes of the block. For more information, refer to Status codes (Page 144).

Name	Declaration	Type	Description
errID	Output	Word	Displays the error codes of the block. For more information, refer to "Information System > Programming a PLC > Instructions > Instructions (S7-1200, S7-1500) > Basic instructions (S7-1200, S7-1500) > Program control operations (S7-1500) > Runtime control (S7-1500) > GET_ERR_ID" instruction in TIA portal online help.
statusRDREC	Output	DWord	Status of the RDREC instruction. For more information on instruction, refer to RDREC: Read data record in TIA Portal online help.
statusWRREC	Output	DWord	Status of the WRREC instruction. For more information on instruction, refer to WRREC: Write data record in TIA Portal online help.

y- indicates HMI device number.

Data type

LAPP_typeSINAMICSConfig

Table 3-39 Structure of data type "LAPP_typeSINAMICSConfig"

Name	Type	Description
doType	BYTE	Drive object type. 0 = SINAMICS S110/S120 with positioning functionality (EPOS) 1 = SINAMICS S110/S120 without positioning functionality (EPOS) 2 = SINAMICS G110/G120 3 = SINAMICS S210 without position 4 = SINAMICS S210 with position
hwID	HW_IO	Hardware identification of the DP slave, taken from "Devices & networks"
axis	BYTE	Device number from the hardware configuration

Status codes

List

Table 3-40 Description of status codes of "LAPP_SINAMICS"

Status code	Description
16#7000	Block is in initialized state.
16#7002	Block is in execution state and has no errors.
16#8200	Faulty license.
16#8201	Invalid HMI index.
16#8202	Invalid drive object type.
16#8203	Invalid input array configuration.
16#8204	Block disabled.

Status code	Description
16#8600	Status of the write record block.
16#8601	Status of the read record block.
16#8602	Invalid order reference.
16#8400	No connection to HMI.

3.12.9.3 SINAMICS diagnostic block using technology objects

Block interface - LAPP_SINAMICSTO

Prerequisite

Use the compatible version of technology object blocks.

Note

If older version TO block is used, you get a compilation error.
Use the correct version for the blocks to work as intended.

You need to select the compatible version of technology block in the area highlighted in the below screen.

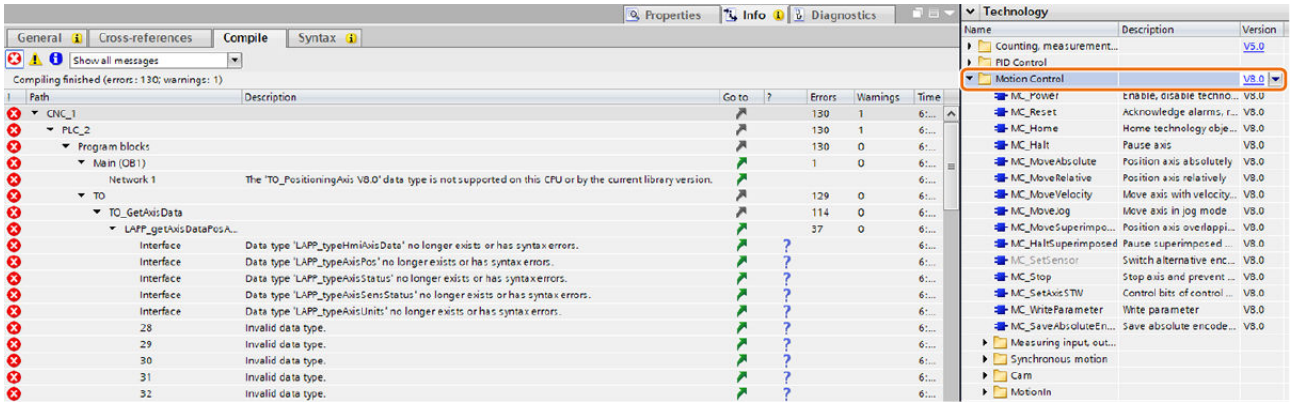


Figure 3-95 Compile error and TO version selection

Block details

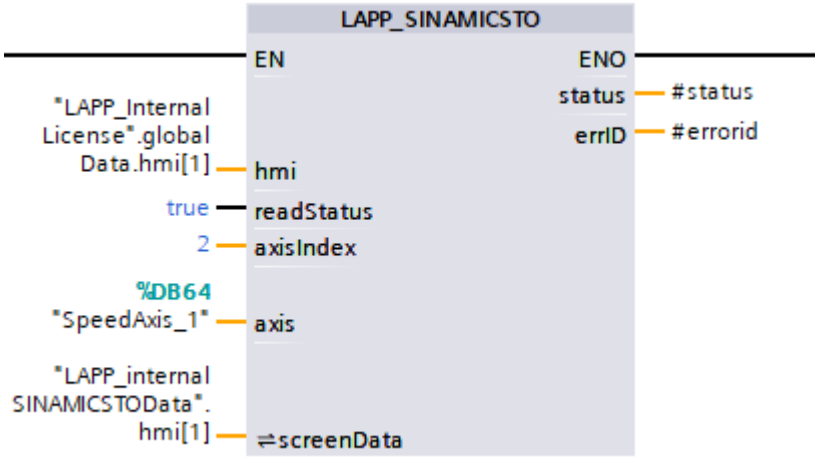


Figure 3-96 "LAPP_SINAMICSTO"

Parameters

Table 3-41 Parameters of "LAPP_SINAMICSTO"

Name	Declaration	Type	Description
hmi	Input	LAPP_typeIntlLicense	Configure the HMI Index which is defined while setting up the HMI device. **Note**: HMI index value cannot be 0. It should match with HMI index value used in License DB ("LAPP_InternalLicense").
readStatus	Input	Bool	"TRUE" enables the communication of the block with the technology object
axisIndex	Input	Int	HMI index for displaying the selected technology object data in the screen
axis	Input	DB_ANY	Interface for the technology object data block. Note: Ensure to interface the correct TO data-block, and verify if the type of axis (speed, position, or synchronous axis) for the entered "axisIndex" in the text list is same.
screenData	InOut	LAPP_typeIntlHmiAxisData	Interface the HMI tag from "LAPP_InternalSINAMICSTOData.hmi[y]". **Note**: "LAPP_InternalSINAMICSTOData" is an internal DB. Modify the array value to increase the HMI device support. For more information, refer to Configuring PLC blocks for different HMI.

Name	Declaration	Type	Description
status	Output	Word	Displays the status codes of the block. For more information, refer to Status codes (Page 147).
errID	Output	Word	Displays the error codes of the block. For more information, refer to "Information System > Programming a PLC > Instructions > Instructions (S7-1200, S7-1500) > Basic instructions (S7-1200, S7-1500) > Program control operations (S7-1500) > Runtime control (S7-1500) > GET_ERR_ID" instruction in TIA portal online help.

y- indicates HMI device number.

Status codes

List

Table 3-42 Description of status codes of "LAPP_SINAMICSTO"

Status code	Description
16#7002	Block is in execution state and has no errors.
16#8200	Faulty license.
16#8201	Invalid HMI index.
16#8202	Block is disabled.
16#8203	Axis is not selected.
16#8204	The selected axis type is not supported.
16#8400	No connection to HMI.

3.12.10 EKS

3.12.10.1 Overview

The "LAPP_EKS_PLC" or "LAPP_EKS_CNC" instruction is used to read the key data of Electronic Key System (EKS). The instruction has to be called in the cyclic program once per EKS adapter (key recording) with different instance of DB.

- LAPP_EKS_PLC: The key information is read from the PLC Profibus/ Profinet protocol. Depending on the output protection level from the PLC, OEM should program WinCC Unified objects to enable or disable them in runtime. The key data is validated based on the input provided to "validateEKSData".
  - If "validateEKSData" = 0, No validation
  - If "validateEKSData" = 1, Access level validation is from local (EKS key).
  - If "validateEKSData" = 2, Access level validation is from server (CSV file).
- LAPP_EKS_CNC: EKS is integrated with SINUMERIK Operate via WinCC Unified runtime. The key information is read from the PLC Profibus/ Profinet protocol. OEM should configure the access levels and passwords in TIA portal. The access level for SINUMERIK Operate is set with or without validation based on the input provided to "validateEKSData".
  - If "validateEKSData" = 0, No validation
  - If "validateEKSData" = 1, Access level validation is from local (EKS key).
  - If "validateEKSData" = 2, Access level validation is from server (CSV file).

Note

Access level validation from CSV file (EKS Server)

- Ensure to copy the CSV file in the path mentioned in HMI tag "EKSBackupPath". For more information refer to Configuring tags for LAHP_EKS in TIA Portal online help.
- Ensure that the path is a shared drive folder and ensure it is encrypted.

Note

TIA Portal Project protection

You need to protect the TIA Portal projects against unauthorized access and assign roles with permissions to users or user groups. For more information on how to protect project, Refer section 2.3.4 Import UMC user groups and users to TIA Portal (https://cache.industry.siemens.com/dl/files/337/109780337/att_1105238/v2/109780337_UMC_TIAPortal_WinCCRTAdvanced_ComfortPanels_en_V11.pdf).

Runtime Project protection

- Recommended to protect project against unauthorized access by using PLC access level and Protect confidential data.
- For UCP, ensure to protect access of the control panel by enabling password protection for control panel. For more information, refer section 5.6.3 Control panel access (https://cache.industry.siemens.com/dl/files/257/109773257/att_1019074/v1/hmi_unified_comfort_panels_operating_instructions_enUS_en-US.pdf).

The "LAHP_EKS" runtime screen is shown below.

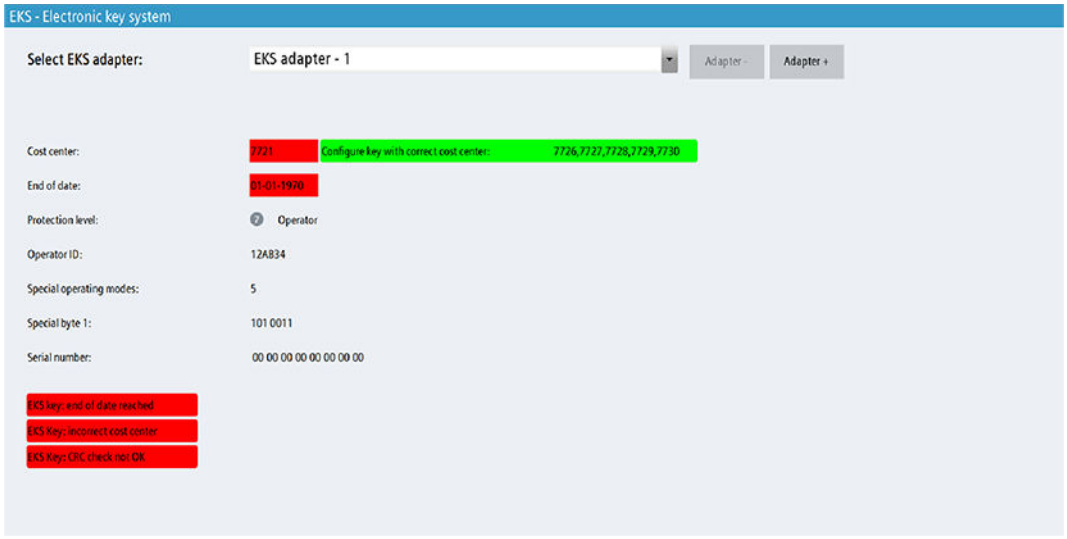


Figure 3-97 EKS - Electronic key system

The Instruction includes the following functions:

- Reading out the EKS key
- CSV validation
- Calculating the checksum of the key and comparing it with the checksum of the key
- Checking the expiry date of the key
- Checking the cost center of the key

Note

The expiry date of the key is compared with the system clock of the S7 CPU. Ensure that the date and time of the S7-CPU are set correctly.

The cost center of the key is compared with the cost center parameterized at the "CostCenter" input parameter.

If all checks have been carried out successfully, the protection level of the key is output at the "ProtectionLevel" output parameter. The key data is deleted when the key is removed.

The "LAPP_typeEKSIntf" data type is used to configure the EKS blocks.

You need to configure the following for "LAHP_EKS" screen to work as intended:

- Block configuration (Page 151) in Step 7
  For more information on the configuring the block, refer to AUTOHOTSPOT.
- Tag creation, text and graphic list (Page 148) configuration in WinCC Unified

3.12.10.2 EKS for CNC

Block interface - LAPP_SetAccessLevel

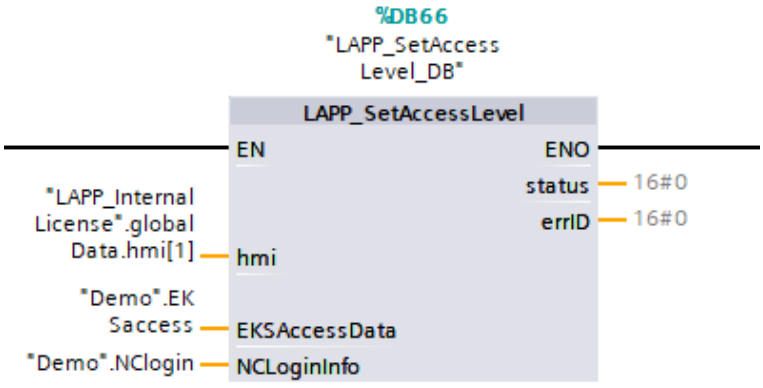


Figure 3-98 LAPP_SetAccessLevel

Parameters

Table 3-43 Parameters of "LAPP_SetAccessLevel"

Name	Declaration	Type	Description
hmi	Input	LAPP_typeIntlLicense	Configure the HMI Index which is defined while setting up the HMI device. **Note**: HMI index value cannot be 0. It should match with HMI index value used in License DB ("LAPP_InternalLicense").
EKSAccessData	InOut	LAPP_typeEKSAccessLevel	Interface to access the EKS data; PLC tag with "LAPP_typeEKSAccessLevel" data type.
NCLoginInfo	InOut	LAPP_typeNCloginInfo	Stores the user level password for SINUMERIK Operate.
status	Output	Word	Displays the status codes of the block. For more information, refer to Status codes (Page 154).
errID	Output	Word	Displays the error codes of the block. For more information, refer to "Information System > Programming a PLC > Instructions > Instructions (S7-1200, S7-1500) > Basic instructions (S7-1200, S7-1500) > Program control operations (S7-1500) > Runtime control (S7-1500) > GET_ERR_ID" instruction in TIA portal online help.

y- indicates HMI device number.

Block interface - LAPP_EKS_CNC

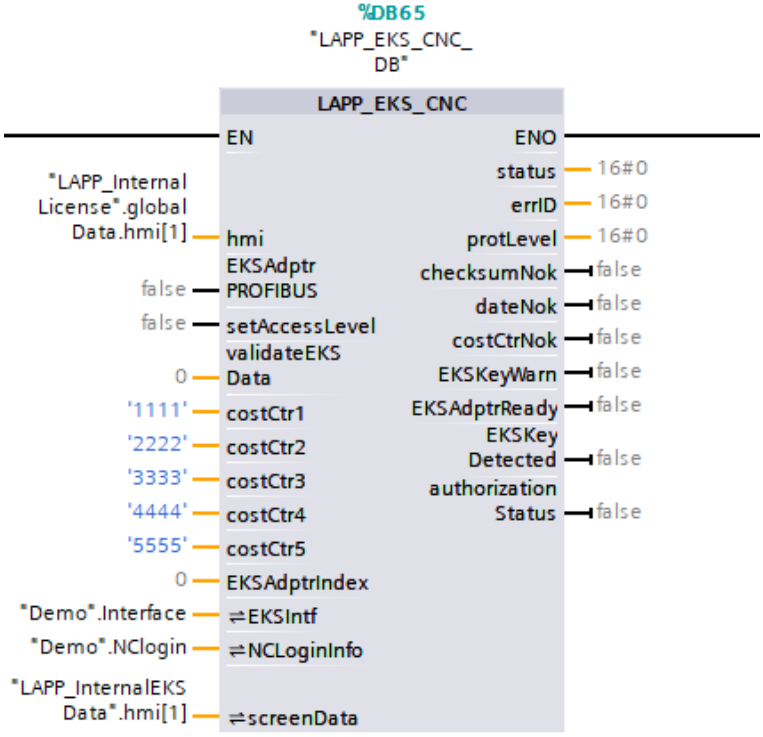


Figure 3-99 "LAPP_EKS_CNC" for CNC

Parameters

Table 3-44 Parameters of "LAPP_EKS_CNC"

Name	Declaration	Type	Description
hmi	Input	LAPP_typeIntlLicense	Configure the HMI Index which is defined while setting up the HMI device. **Note**: HMI index value cannot be 0. It should match with HMI index value used in License DB ("LAPP_InternalLicense").
EKSAdptrPROFIBUS	Input	Bool	Interface of the EKS adapter: PROFINET=FALSE PROFIBUS=TRUE
setAccessLevel	Input	Bool	Automatic login to or logout from SINUMERIK Operate. • True = Automatic login and logout enabled • False = Automatic login and logout disabled

Name	Declaration	Type	Description
validateEKSData	Input	USInt	Validation of EKS data. • 0 = No validation • 1 = Access level validation from EKS key • 2 = Access level validation from CSV file
costCtr1	Input	STRING[4]	Cost center number
costCtr2	Input	STRING[4]	Cost center number
costCtr3	Input	STRING[4]	Cost center number
costCtr4	Input	STRING[4]	Cost center number
costCtr5	Input	STRING[4]	Cost center number
EKSAdptrIndex	Input	UInt	HMI index for displaying the selected key data in the screen "EKS - electronic"
EKSIntf	InOut	LAPP_typeEKSIntf	Interface to the EKS adapter; PLC tag with "LAPP_typeEksIntf" data type.
NCLoginInfo	InOut	LAPP_typeNCloginInfo	Stores the user level password for SINUMERIK Operate.
screenData	InOut	LAPP_typeIntlHmiEKSData	Interface the HMI tag from "LAPP_Internal-EKSData.hmi[y]". **Note**: "LAPP_InternalEKSData" is an internal DB. Modify the array value to increase the HMI device support. For more information, refer to Configuring PLC blocks for different HMI.
status	Output	Word	Displays the status codes of the block. For more information, refer to Status codes (Page 154).
errID	Output	Word	Displays the error codes of the block. For more information, refer to "Information System > Programming a PLC > Instructions > Instructions (S7-1200, S7-1500) > Basic instructions (S7-1200, S7-1500) > Program control operations (S7-1500) > Runtime control (S7-1500) > GET_ERR_ID" instruction in TIA portal online help.
protLevel	Output	Byte	Output of the protection level
checksumNOK	Output	Bool	Checksum of the key not OK
dateNOK	Output	Bool	Expiry date of the key elapsed
costCtrNOK	Output	Bool	Cost center does not match parameterized cost center
EKSKeyWarn	Output	Bool	EKS key will become invalid in less than 30 days
EKSAdptrReady	Output	Bool	EKS adapter ready to operate

Name	Declaration	Type	Description
EKSKeyDetected	Output	Bool	EKS key recognized
authorizationStatus	Output	Bool	Authorization status • "authorizationStatus" is **TRUE** if the EKS keydata compared with the CSV file are identical. • "authorizationStatus" is **FALSE** if the EKS keydata compared with the CSV file are not identical.

y- indicates HMI device number.

For more information, refer to AUTOHOTSPOT

Note

- "LAPP_EKS_CNC" is only compatible with GIV 6.25 and its relevant hotfixes.
- While using EKS, ensure to encrypt the SD card to protect user data.

Data type

LAPP_typeEKSIntf

Table 3-45 Structure of data type "LAPP_typeEKSIntf"

Name	Type	Description
keyData	Array [0..127] of Byte	Configure the key data to display on screen.

LAPP_typeNCloginInfo

Table 3-46 Structure of data type "LAPP_typeNCloginInfo"

Name	Type	Description
password	Array [0..127] of String	SINUMERIK Operate login password.

LAPP_typeEKSAccessLevel

Table 3-47 Structure of data type "LAPP_typeEKSAccessLevel"

Name	Type	Description
execute	Bool	
protLevel	USInt	

Status codes

List

Description of status codes of "LAPP_SetAccessLevel"

Status code	Description
16#7002	Block is in execution state and has no errors.
16#8200	Faulty license.
16#8201	Invalid HMI index.
16#8017	"Execute" is set to true before the completion of active job.
16#820a	Password is lengthy.
16#820b	Incorrect or weak password.
16#820c	Password value not within the range.

Description of status codes of "LAPP_EKS_CNC"

Status code	Description
16#7002	Block is in execution state and has no errors.
16#8200	Faulty license.
16#8201	Invalid HMI index.
16#8202	Cost center is not configured.
16#8400	No connection to HMI.
16#8203	No free interface for the NCK.
16#8204	Job sent to NCK, waiting for response.
16#8205	Job sent to NCK, waiting for completion.
16#8206	New job started prior to completion of the active job.
16#8207	Data transmission not possible due to huge volume.
16#8208	Access not possible.
16#8209	Parameter value type is not compatible.
16#8210	Parameter value type is not supported.
16#8211	The length and type of the data in the variable token do not match.
16#8212	Data is too huge for the parameter value.
16#8213	Zero at parameter value is not permitted.
16#8214	Indicates invalid or incorrect path.
16#8215	Indicates that the backup file is empty.
16#8216	Indicates backup failure.
16#8217	Invalid serial number.
16#8218	Invalid cost center.
16#8219	Incorrect position of key switch.

3.12.10.3 EKS for PLC

Block interface - LAPP_EKS_PLC

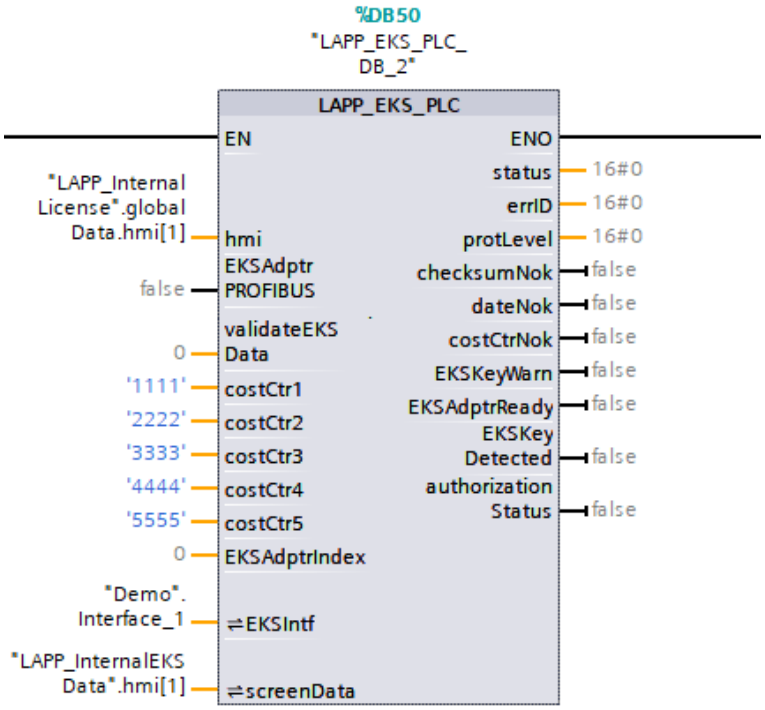


Figure 3-100 "LAPP_EKS_PLC" for PLC

Parameters

Table 3-48 Parameters of "LAPP_EKS_PLC"

Name	Declaration	Type	Description
hmi	Input	LAPP_typeIntlLicense	Configure the HMI Index which is defined while setting up the HMI device. **Note**: HMI index value cannot be 0. It should match with HMI index value used in License DB ("LAPP_InternalLicense").
EKSAdptrPROFIBUS	Input	Bool	Interface of the EKS adapter: • PROFINET=FALSE • PROFIBUS=TRUE
validateEKSData	Input	USInt	Validation of EKS data. • 0 = No validation • 1 = Access level validation from EKS key • 2 = Access level validation from CSV file
costCtr1	Input	STRING[4]	Cost center number

Name	Declaration	Type	Description
costCtr2	Input	STRING[4]	Cost center number
costCtr3	Input	STRING[4]	Cost center number
costCtr4	Input	STRING[4]	Cost center number
costCtr5	Input	STRING[4]	Cost center number
EKSAdptrIndex	Input	UInt	HMI index for displaying the selected key data in the screen "EKS - electronic"
EKSIntf	InOut	LAPP_typeEKSIntf	Interface to the EKS adapter; PLC tag with "LAPP_typeEksIntf" data type.
screenData	InOut	LAPP_typeHmiEKSData	Interface the HMI tag from "LAPP_Internal-EKSData.hmi[y]". **Note**: "LAPP_InternalEKSData" is an internal DB. Modify the array value to increase the HMI device support. For more information, refer to Configuring PLC blocks for different HMI.
status	Output	Word	Displays the status codes of the block. For more information, refer to Status codes (Page 157).
errID	Output	Word	Displays the error codes of the block. For more information, refer to "Information System > Programming a PLC > Instructions > Instructions (S7-1200, S7-1500) > Basic instructions (S7-1200, S7-1500) > Program control operations (S7-1500) > Runtime control (S7-1500) > GET_ERR_ID" instruction in TIA portal online help.
protLevel	Output	Byte	Output of the protection level
checksumNOK	Output	Bool	Checksum of the key not OK
dateNOK	Output	Bool	Expiry date of the key elapsed
costCtrNOK	Output	Bool	Cost center does not match parameterized cost center
EKSKeyWarn	Output	Bool	EKS key will become invalid in less than 30 days
EKSAdptrReady	Output	Bool	EKS adapter ready to operate
EKSKeyDetected	Output	Bool	EKS key recognized
authorizationStatus	Output	Bool	Authorization status • "authorizationStatus" is **TRUE** if the EKS keydata compared with the CSV file are identical. • "authorizationStatus" is **FALSE** if the EKS keydata compared with the CSV file are not identical.

y- indicates HMI device number.

For more information, refer to AUTOHOTSPOT

Note

While using EKS, ensure to encrypt the SD card to protect user data.

Data type

LAPP_typeEKSIntf

Table 3-49 Structure of data type "LAPP_typeEKSIntf"

Name	Type	Description
keyData	Array [0..127] of byte	Configure the key data to display on screen.

Status codes

List

Description of status codes of "LAPP_EKS_PLC"

Status code	Description
16#7002	Block is in execution state and has no errors.
16#8200	Faulty license.
16#8201	Invalid HMI index.
16#8202	Cost center is not configured.
16#8400	No connection to HMI.
16#8214	Indicates invalid or incorrect path.
16#8215	Indicates that the backup file is empty.
16#8216	Indicates backup failure.
16#8217	Invalid serial number.
16#8218	Invalid cost center.
16#8219	Incorrect position of key switch.

3.12.10.4 Authorization levels concept

The following table shows the authorization levels concept when the EKS (Electronic Key Systems) is used:

Table 3-50 Authorization levels concept

Standard Siemens authorization levels	Authorization	Your EKS
Authorization level 1 (machine manufacturer)	Manufacturer, service, maintenance engineer	Authorization level 1 Default: Red key
Authorization level 2 (commissioning engineer, service)	Not used	Not used
Authorization level 3 (end user)	Not used	Not used

Standard Siemens authorization levels	Authorization	Your EKS
Authorization level 4 (programmer, machine setter)	Programmer, machine setter	Authorization level 4 Default: Green key
Authorization level 5 (qualified operator)	Not used	Not used
Authorization level 6 (trained operator)	Operator	Authorization level 6 Default: Black key
Authorization level 7 (semi-skilled operator)	No authorization	No key Default: Grey key

Note

- You can modify authorization names as per the requirement in "LAHP_EksProtectionLevel" text list.
  For more information, refer to Editing EKS authorization names.
- You can modify the default key as per the requirement in "LAHP_User_FullHD" or "LAHP_User_HDReady" or "LAHP_User_NonHD" graphic list.
  For more information, refer to Editing graphics.

3.12.10.5 Format of the EKS key

Use suitable software to read and write to the EKS key, e.g. Electronic Key Manager of the Euchner company.

Data structure

Write to the EKS key, considering the following data structure:

- Memory size of the EKS key: 124 bytes
- Area that can be written to: Bytes 0 - 115
- Area with serial number that cannot be changed:
  Bytes 116 - 123

The function block described here "LAPP_EKS" verifies the checksum starting at byte 84. Since the key allocation described here is also used for SINUMERIK-based machines with HMI PRO, some of the written data is configured for machine tools. To use an EKS key with this function block, it has to be written in the following format as of byte 84:

Table 3-51 Data of the EKS key

Data area of the EKS key	Size	Data format	Content	Further information
84 - 87	4 bytes	ASCII	Cost center	
88 - 95	8 bytes	ASCII	End date	End of the validity of the EKS key
96	1 byte	Hexadecimal	Authorization level	Protection levels 1-7
97 - 103	7 bytes	ASCII	Machine operator identification	

Data area of the EKS key	Size	Data format	Content	Further information
104	1 byte	Hexadecimal	Safe operating modes	MSO 1-5
105.0	1 bit	BOOL	Special bit, reworking	1=rework is enabled
105.1	1 bit	BOOL	Special bit, operating system	1=access to PC OS enabled
105.2	1 bit	BOOL	Special bit, quality data	0 = not OK 1 = OK
105.3 - 105.7	5 bits	BOOL		Spare
106 - 113	8 bytes	Hexadecimal	Special bytes	Spare

3.12.10.6 Examples

CNC as a controller

The CSV file is provided with the data and the EKS key is inserted in the EKS adapter.

- If the input provided to "validateEKSData" is 0, then EKS data is not validated.
- If the input provided to "validateEKSData" is 1,
  - If the "setAccessLevel" is set to "TRUE" automatic login or logout from SINUMERIK Operate takes place.
  - If the "setAccessLevel" is set to "FALSE" automatic login or logout from SINUMERIK Operate does not take place.
- If the input provided to "validateEKSData" is 2 and then the key data is compared with the CSV file.
  - If the compared data is identical then "authorizationStatus" is set to TRUE and if the "setAccessLevel" is set to "TRUE" automatic login or logout from SINUMERIK Operate takes place.
  - If the compared data is identical then "authorizationStatus" is set to TRUE and if the "setAccessLevel" is set to "FALSE" automatic login or logout from SINUMERIK Operate does not take place.
  - If the compared data is not identical then "authorizationStatus" is set to FALSE.

3.13 Machine control panel

3.13.1 LAPP_Powerride

3.13.1.1 Overview

Note

Instruction available from:

CNC-SW ≥ V6.24 and library version instruction ≥ V14.1

Description

Instruction "LAPP_Powerride" integrates a rotary/push control dial (SINUMERIK Powerride) into the SINUMERIK environment.

With the Powerride you change the feedrate in percent with immediate effect on the machine. To do this, multiply the override factor of the Powerride by the programmed feed rates.

Interfaces for the feedrate change:

- Individual in the channel instance
- Individual in the axis instance

More information on the hardware of the Powerride can be found in the *Equipment Manual SINUMERIK ONE MCP*.

You can query data and set the behavior of the Powerride at the parameters of the instruction.

You can immediately use Powerride after you have configured your own global data block (in the example "MyPowerrideConfig (Page 164)") to store configuration data. Transfer the required configuration to the block at parameter "powerrideConfig".

If you want to program the behavior of the Powerride yourself, you can query the signals directly with the parameters "powerrideInputs" or control them with "powerrideOutputs".

Call time

- Once per PLC cycle for each instance, e.g. in the main OB

3.13.1.2 Block interface - LAPP_Powerride

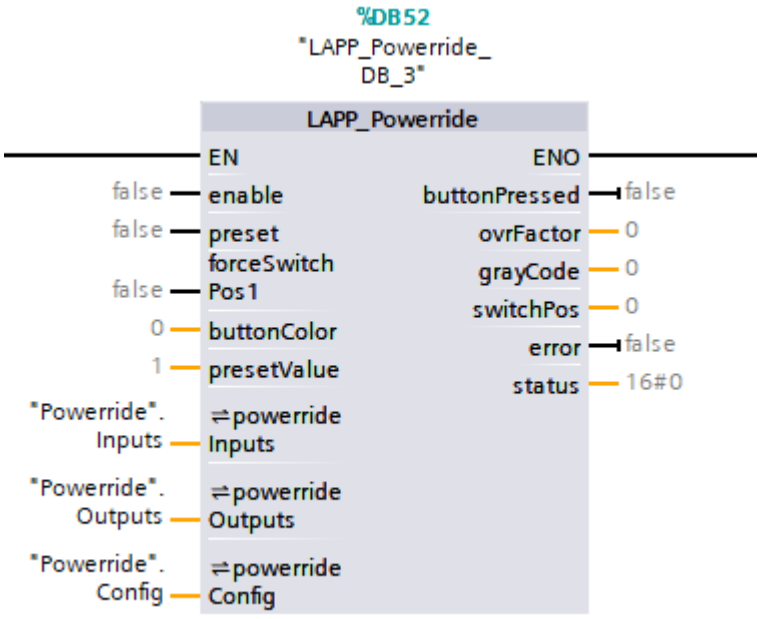


Figure 3-101 "LAPP_Powerride"

Parameters

Table 3-52 Parameters of "LAPP_Powerride"

Parameter	Declaration	Data type	Description
enable	Input	Bool	Activate Powerride function The parameter may only be set to TRUE after the Power On (control startup) has been completed. TRUE: Configuration data are checked and processing of the FBs is started if the check is successful. With Power On, the last "switchPos" becomes active if the instance DB is retentive. FALSE: No processing of the FB; output values are retained.
preset	Input	Bool	Change "switchPos" With the positive edge, "presetValue" is taken over as the current "switchPos". The "presetValue" parameterization must be valid for this purpose. Override factor "powerrideConfig.switchPos[1..36].value" is output. The displays in "grayCode" and "ovrFactor" are refreshed.

Parameter	Declaration	Data type	Description
forceSwitchPos1	Input	Bool	Set switch to position 1. With the positive edge, the switch is set to position 1 and the override factor "powerrideConfig.switchPos[1..36].value" is output. The same result can be achieved with presetValue = 1 and preset = TRUE. In this respect, this parameter is provided for convenience. If at the same time "presetValue" contains a valid value and "preset" is set, the function of "forceSwitchPos1" prevails. The displays in "grayCode" and "ovrFactor" are refreshed.
buttonColor	Input	USInt	Defining the color of the switch LED. At present, only the color green is possible if bit 1 is set. Value range: 0, 1, 4, 5: no color selected: 2, 3, 6, 7: green
presetValue	Input	USInt	New value for "switchPos" Value range: 1..36 With a positive edge of "preset", the value is accepted as the new "switchPos" **Special case 1** "presetValue" is between "powerrideConfig.pos100" and "powerrideConfig.posMax" and also "switchPos" (current position) is less than "powerrideConfig.pos100" For safety reasons, the value of "powerrideConfig.pos100" is taken over with "preset" in this case. **Special case 2** "presetValue" is greater than "powerrideConfig.posMax" and less than 36 With "preset" the value of "powerrideConfig.posMax" is taken over.
powerrideInputs	InOut[1)]	LAPP_typePowerrideInputs	Signals from the Powerride to the PLC
powerrideOutputs	InOut[1)]	LAPP_typePowerrideOutputs	Signals from the PLC to the Powerride
powerrideConfig	InOut[1)]	LAPP_typePowerrideConfig	Configuration data
buttonPressed	Output	Bool	Indication whether the switch is pressed.
ovrFactor[2)]	Output	USInt	The value parameterized under "powerrideConfig.switchPos[1..36].value" is output. Value range: 0..200
grayCode[2)]	Output	USInt	The value parameterized under "powerrideConfig.switchPos[1..36].value" is output as gray code.
switchPos	Output	USInt	Display of the current switch position of the Powerride Value range: 1..powerrideConfig.posMax

Parameter	Declaration	Data type	Description
error	Output	Bool	Indication whether an error is present The cause of the error is displayed in the "status" parameter. TRUE: An error has occurred, the other outputs are reset FALSE: If "enable" = FALSE or no error is present
status	Output	Word	Status number when the job is executed. Error number if an error occurred.

1) These parameters could also have been defined as pure inputs (powerrideInputs, powerrideConfig) or outputs (powerrideOutputs). However, using InOut has runtime advantages at this point because InOut parameters are transferred as pointers instead of copies.

2) You set the interpretation of the override factor in the machine data (e.g.: MD12020 $MN_OVR_FEED_IS_GRAY_CODE).

Parameter

The following table shows the parameters of the instruction:

Pulse diagram

Basic information on processing the instruction can be found in "Information system > Configuring and programming SINUMERIK > Programming SINUMERIK PLC > Programming SINUMERIK PLC (ONE) > PLC instructions SINU_* > Execution of instructions and blocks > Continuous asynchronous processing" in TIA Portal online help.

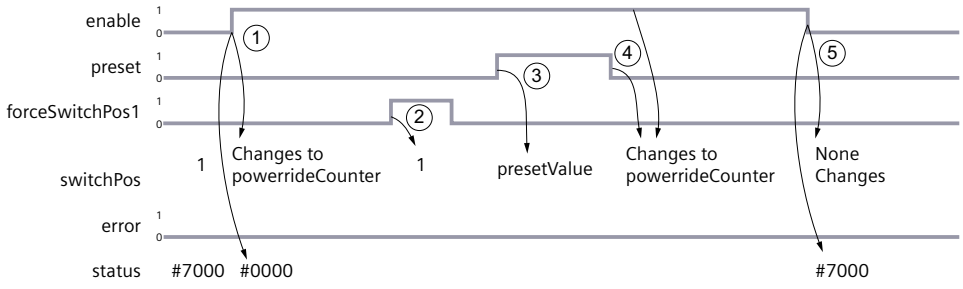


Figure 3-102 LAPP_Powerride: Signal curve with a change at the rotary switch

1. Parameter "enable" is set:
   After the configuration has been checked successfully, the instruction transfers the changes to the rotary knob ("powerrideCounter") to the "switchPos". The outputs "ovrFactor" and "grayCode" are also set according to the transferred configuration.
2. Parameter "forceSwitchPos1" is set:
   Output "switchPos" is set to 1. The outputs "ovrFactor "and "grayCode" are set according to the configuration for "switchPos[1..36]".
3. Parameter "preset" is set, parameter "forceSwitchPos" is 0:
   At the output, "switchPos" is output according to "presetValue" and "ovrFactor" and "grayCode" are set.

4. Parameters "forceSwitchPos1" and "preset" are both 0:
   The instruction again transfers the changes to the rotary knob to the "switchPos" and "ovrFactor" and "grayCode" are set according to the configuration.
5. Parameter "enable" is reset:
   The outputs keep their last values.

3.13.1.3 Create data block MyPowerrideConfig

Description

Data block "MyPowerrideConfig", type "LAPP_typePowerrideConfig" contains the preconfigured settings for Powerride. You can transfer one of the listed configurations "configFeed" or "configSpindle" to instruction "LAPP_Powerride" at parameter "powerrideConfig".

Alternatively, you can create your own configuration using type "LAPP_typePowerrideConfig" with your configuration settings.

You create the data block by importing an external source in the TIA Portal project.

This is how you created data block "MyPowerrideConfig"

1. Copy the complete source code from the following Section "Copy template for the data block" into a text editor.

   Note

   Copy from the online help in the TIA Portal

   You copy the complete source code from the TIA Portal online help if you click on button in the header of the program section.

2. Save the file as "MyPowerrideConfig.db".
3. In the project tree of the TIA Portal project, navigate to folder "CNC_1 > PLC_1 > External sources".
4. Click on "Add new external file".
5. In the dialog that is displayed, select file "MyPowerrideConfig.db". The file is imported as external source and displayed in the project navigation.
6. Click on the imported file and in the context menu, select "Generate block from source". Observe the message that is displayed.
7. Confirm with "OK". The data block is created and after compilation is displayed under "Program blocks". Observe the messages in tab "Compile".

The data block created contains the preconfigured settings for Powerride. You transfer the configurations to the instruction"LAPP_Powerride" at parameter "powerrideConfig".

External source template

Copy template for the data block

```
DATA_BLOCK "MyPowerrideConfig"
TITLE = Powerride Configuration Parameters
{ DB_Accessible_From_OPC_UA := 'FALSE' ;
  DB_Accessible_From_Webserver := 'FALSE' ;
  S7_Optimized_Access := 'TRUE' }
AUTHOR : SINUMERIK
FAMILY : SINUMERIK_ONE
VERSION : 10.0
//Template with two predefined configurations for a Powerride
   VAR RETAIN
      configFeed : "LAPP_typePowerrideConfig";   // Feed configuration %
      configSpindle : "LAPP_typePowerrideConfig";   // Spindle configuration %
   END_VAR

BEGIN
   configFeed.pos100 := 19;
   configFeed.posMax := 23;
   configFeed.delay100 := T#500MS;
   configFeed.switchPos[1].vibrationIndex := 1;
   configFeed.switchPos[2].value := 1;
   configFeed.switchPos[2].ledColor := 7;
   configFeed.switchPos[3].value := 2;
   configFeed.switchPos[3].ledColor := 7;
   configFeed.switchPos[4].value := 4;
   configFeed.switchPos[4].ledColor := 7;
   configFeed.switchPos[5].value := 6;
   configFeed.switchPos[5].ledColor := 7;
   configFeed.switchPos[6].value := 8;
   configFeed.switchPos[6].ledColor := 7;
   configFeed.switchPos[7].value := 10;
   configFeed.switchPos[7].ledColor := 7;
   configFeed.switchPos[8].value := 20;
   configFeed.switchPos[8].ledColor := 7;
   configFeed.switchPos[9].value := 30;
   configFeed.switchPos[9].ledColor := 7;
   configFeed.switchPos[10].value := 40;
   configFeed.switchPos[10].ledColor := 7;
   configFeed.switchPos[11].value := 50;
   configFeed.switchPos[11].ledColor := 7;
   configFeed.switchPos[12].value := 60;
   configFeed.switchPos[12].ledColor := 7;
   configFeed.switchPos[13].value := 70;
   configFeed.switchPos[13].ledColor := 7;
   configFeed.switchPos[14].value := 75;
   configFeed.switchPos[14].ledColor := 7;
   configFeed.switchPos[15].value := 80;
   configFeed.switchPos[15].ledColor := 7;
   configFeed.switchPos[16].value := 85;
   configFeed.switchPos[16].ledColor := 7;
   configFeed.switchPos[17].value := 90;
   configFeed.switchPos[17].ledColor := 7;
   configFeed.switchPos[18].value := 95;
   configFeed.switchPos[18].ledColor := 7;
   configFeed.switchPos[19].value := 100;
```

Copy template for the data block

```
configFeed.switchPos[19].ledColor := 2;
configFeed.switchPos[19].vibrationIndex := 2;
configFeed.switchPos[20].value := 105;
configFeed.switchPos[20].ledColor := 3;
configFeed.switchPos[21].value := 110;
configFeed.switchPos[21].ledColor := 3;
configFeed.switchPos[22].value := 115;
configFeed.switchPos[22].ledColor := 3;
configFeed.switchPos[23].value := 120;
configFeed.switchPos[23].ledColor := 1;
configFeed.switchPos[23].vibrationIndex := 3;
configFeed.switchPos[24].value := 120;
configFeed.switchPos[24].ledColor := 1;
configFeed.switchPos[25].value := 120;
configFeed.switchPos[25].ledColor := 1;
configFeed.switchPos[26].value := 120;
configFeed.switchPos[26].ledColor := 1;
configFeed.switchPos[27].value := 120;
configFeed.switchPos[27].ledColor := 1;
configFeed.switchPos[28].value := 120;
configFeed.switchPos[28].ledColor := 1;
configFeed.switchPos[29].value := 120;
configFeed.switchPos[29].ledColor := 1;
configFeed.switchPos[30].value := 120;
configFeed.switchPos[30].ledColor := 1;
configFeed.switchPos[31].value := 120;
configFeed.switchPos[31].ledColor := 1;
configFeed.switchPos[32].value := 120;
configFeed.switchPos[32].ledColor := 1;
configFeed.switchPos[33].value := 120;
configFeed.switchPos[33].ledColor := 1;
configFeed.switchPos[34].value := 120;
configFeed.switchPos[34].ledColor := 1;
configFeed.switchPos[35].value := 120;
configFeed.switchPos[35].ledColor := 1;
configFeed.switchPos[36].value := 120;
configFeed.switchPos[36].ledColor := 1;
configSpindle.pos100 := 11;
configSpindle.posMax := 15;
configSpindle.delay100 := T#500MS;
configSpindle.switchPos[1].value := 50;
configSpindle.switchPos[1].ledColor := 7;
configSpindle.switchPos[2].value := 55;
configSpindle.switchPos[2].ledColor := 7;
configSpindle.switchPos[3].value := 60;
configSpindle.switchPos[3].ledColor := 7;
configSpindle.switchPos[4].value := 65;
configSpindle.switchPos[4].ledColor := 7;
configSpindle.switchPos[5].value := 70;
configSpindle.switchPos[5].ledColor := 7;
configSpindle.switchPos[6].value := 75;
configSpindle.switchPos[6].ledColor := 7;
configSpindle.switchPos[7].value := 80;
configSpindle.switchPos[7].ledColor := 7;
configSpindle.switchPos[8].value := 85;
configSpindle.switchPos[8].ledColor := 7;
```

Copy template for the data block

```
configSpindle.switchPos[9].value := 90;
configSpindle.switchPos[9].ledColor := 7;
configSpindle.switchPos[10].value := 95;
configSpindle.switchPos[10].ledColor := 7;
configSpindle.switchPos[11].value := 100;
configSpindle.switchPos[11].ledColor := 2;
configSpindle.switchPos[11].vibrationIndex := 2;
configSpindle.switchPos[12].value := 105;
configSpindle.switchPos[12].ledColor := 3;
configSpindle.switchPos[13].value := 110;
configSpindle.switchPos[13].ledColor := 3;
configSpindle.switchPos[14].value := 115;
configSpindle.switchPos[14].ledColor := 3;
configSpindle.switchPos[15].value := 120;
configSpindle.switchPos[15].ledColor := 1;
configSpindle.switchPos[15].vibrationIndex := 3;
configSpindle.switchPos[16].value := 120;
configSpindle.switchPos[16].ledColor := 1;
configSpindle.switchPos[17].value := 120;
configSpindle.switchPos[17].ledColor := 1;
configSpindle.switchPos[18].value := 120;
configSpindle.switchPos[18].ledColor := 1;
configSpindle.switchPos[19].value := 120;
configSpindle.switchPos[19].ledColor := 1;
configSpindle.switchPos[20].value := 120;
configSpindle.switchPos[20].ledColor := 1;
configSpindle.switchPos[21].value := 120;
configSpindle.switchPos[21].ledColor := 1;
configSpindle.switchPos[22].value := 120;
configSpindle.switchPos[22].ledColor := 1;
configSpindle.switchPos[23].value := 120;
configSpindle.switchPos[23].ledColor := 1;
configSpindle.switchPos[23].vibrationIndex := 3;
configSpindle.switchPos[24].value := 120;
configSpindle.switchPos[24].ledColor := 1;
configSpindle.switchPos[25].value := 120;
configSpindle.switchPos[25].ledColor := 1;
configSpindle.switchPos[26].value := 120;
configSpindle.switchPos[26].ledColor := 1;
configSpindle.switchPos[27].value := 120;
configSpindle.switchPos[27].ledColor := 1;
configSpindle.switchPos[28].value := 120;
configSpindle.switchPos[28].ledColor := 1;
configSpindle.switchPos[29].value := 120;
configSpindle.switchPos[29].ledColor := 1;
configSpindle.switchPos[30].value := 120;
configSpindle.switchPos[30].ledColor := 1;
configSpindle.switchPos[31].value := 120;
configSpindle.switchPos[31].ledColor := 1;
configSpindle.switchPos[32].value := 120;
configSpindle.switchPos[32].ledColor := 1;
configSpindle.switchPos[33].value := 120;
configSpindle.switchPos[33].ledColor := 1;
configSpindle.switchPos[34].value := 120;
configSpindle.switchPos[34].ledColor := 1;
configSpindle.switchPos[35].value := 120;
```

Copy template for the data block

```
    configSpindle.switchPos[35].ledColor := 1;
    configSpindle.switchPos[36].value := 120;
    configSpindle.switchPos[36].ledColor := 1;

END_DATA_BLOCK
```

3.13.1.4 LED color palette

The following table shows the possible color combinations depending on the individual colors blue, green and red.

Note

For the switch (input "buttonColor" of "LAPP_Powerride") only the color green can be selected at present.

Combined color	Blue (bit 2)	Green (bit 1)	Red (bit 0)
-	-	-	-
Red	-	-	x
Green	-	x	-
Yellow	-	x	x
Blue	x	-	-
Magenta	x	-	x
Cyan	x	x	-
White	x	x	x

3.13.1.5 Example: Preparing to integrate a Powerride via an MCP

Requirement

The following data block has been created in the project:

- MyPowerrideConfig (see "Create data block MyPowerrideConfig (Page 164)")

The following data types are available in the project:

- LBP_typeModularMcpBaseInput, LBP_typeModularMcpUniversalInput, LAPP_typeHandwheelData
- LBP_typeModularMcpBaseOutput, LBP_typeModularMcpUniversalOutput

Integrating a Powerride via an MCP

The example uses a SINUMERIK ONE MCP with two Powerride modules connected via Industrial Ethernet (IE).

Global data block "OneMcpData" is available for the input and output images. This block is not included in the library. The example of "OneMcpData" shows how the input and output

image can be mapped for a SINUMERIK ONE MCP using data types supplied in the libraries, and how this DB can be used.

OneMcpData

	Name	Data type
1	Static	
2	fromMcp	Struct
3	base	"LBP_typeModularMcpBaseInput"
4	keypad1	"LBP_typeModularMcpUniversalInput"
5	keypad2	"LBP_typeModularMcpUniversalInput"
6	keypad3	"LBP_typeModularMcpUniversalInput"
7	keypad4	"LBP_typeModularMcpUniversalInput"
8	powerride1	LAPP_typePowerrideInputs
9	powerride2	LAPP_typePowerrideInputs
10	toMcp	Struct
11	base	"LBP_typeModularMcpBaseOutput"
12	keypad1	"LBP_typeModularMcpUniversalOutput"
13	keypad2	"LBP_typeModularMcpUniversalOutput"
14	keypad3	"LBP_typeModularMcpUniversalOutput"
15	keypad4	"LBP_typeModularMcpUniversalOutput"
16	powerride1	LAPP_typePowerrideOutputs
17	powerride2	LAPP_typePowerrideOutputs
18	handwheel1	LAPP_typeHandwheelData
19	handwheel2	LAPP_typeHandwheelData

Figure 3-103 Data block: OneMcpData

The following table shows an example of the input and output addresses:

	Byte numbers of the data
Input data	"OneMcpData".fromMcp.powerride1 and "OneMcpData".fromMcp.powerride2
Output data	"OneMcpData".toMcp.powerride1 and "OneMcpData".toMcp.powerride2

Note

The addresses of the Powerride data in the input and output image of the MCP can change with the firmware version. The actual address assignment is described in the *Equipment Manual SINUMERIK ONE MCP*.

3.13.1.6 Example: calling LAPP_Powerride

Description

The example shows "LAPP_Powerride" being called. The input and output image of the SINUMERIK ONE MCP must be assigned beforehand with the parameters "fromMcp" and "toMcp" of the block "LAPP_UpdateEthernetMcp".

Global data block "OneMcpData" contains the input and output image in the example. You can also find information on this in the *Equipment Manual SINUMERIK ONE MCP*.

The input and output parameters of "LAPP_Powerride" are supplied using data block "PowerrideDb".

PowerrideDb

	Name	Data type	Start value
1	Static		
2	pr1enable	Bool	false
3	pr1disable	Bool	false
4	pr1preset	Bool	false
5	pr1forceSwitchPos1	Bool	false
6	pr1buttonColor	USInt	0
7	pr1presetValue	USInt	0
8	pr1error	Bool	false
9	pr1buttonPressed	Bool	false
10	pr1ovrFactor	USInt	0
11	pr1grayCode	USInt	0
12	pr1switchPos	USInt	0
13	pr1status	Word	16#0

Figure 3-104 Data block: PowerrideDb

For parameter "powerrideConfig", configuration "configFeed" is used, which was created from data block "MyPowerrideConfig".

The following code shows the call of "LAPP_Powerride" in "Main [OB1]". The instruction "LAPP_UpdateEthernetMcp" is called first. This instruction is used for communication with operator components via Industrial Ethernet. The Powerride block itself is only activated when cyclic communication to the MCP is taking place ("mcpReady" = TRUE) and should not be explicitly disabled otherwise ("pr1disable")

Code example

SCL

```
// InstUpdateEthernetMcp : LAPP_UpdateEthernetMcp
// InstPowerride1 : LAPP_Powerride

#InstUpdateEthernetMcp(enable := TRUE,
                       address := 198,
                       udpPort := 16001,
                       connectionId := 1001,
                       interfaceId := 262,
                       mcpReady => #tempMcpReady,
                       newData => #tempNewData,
                       identified => #tempIdentified,
                       mcpType => #tempMcpType,
                       lengthIn => #tempLengthIn,
                       lengthOut => #tempLengthOut,
                       busy => #tempBusy,
                       warning => #tempWarning,
                       error => #tempError,
                       status => #tempStatus,
                       fromMcp := "OneMcpData".fromMcp,
                       toMcp := "OneMcpData".toMcp);

"PowerrideDb".pr1enable := #tempMcpReady
                           AND NOT "PowerrideDb".pr1disable;

"InstPowerride1"(enable := "PowerrideDb".pr1enable,
                 preset := "PowerrideDb".pr1preset,
                 forceSwitchPos1 := "PowerrideDb".pr1forceSwitchPos1,
                 buttonColor := "PowerrideDb".pr1buttonColor,
                 presetValue := "PowerrideDb".pr1presetValue,
                 error => "PowerrideDb".pr1error,
                 buttonPressed => "PowerrideDb".pr1buttonPressed,
                 ovrFactor => "PowerrideDb".pr1ovrFactor,
                 grayCode => "PowerrideDb".pr1grayCode,
                 switchPos => "PowerrideDb".pr1switchPos,
                 status => "PowerrideDb".pr1status,
                 powerrideInputs := "OneMcpData".fromMcp.powerride1,
                 powerrideOutputs := "OneMcpData".toMcp.powerride1,
                 powerrideConfig := "MyPowerrideConfig".configFeed);
```

Error, status, and remedy

Status/error code* (W#16#...)	Description	Remedy
0000	No error	-
7000	Function is not executed	Set "enable" to TRUE
7101	Warning[1]: "buttonColor" invalid	Use 0..7 See LED color palette, no reset of "enable" necessary after correction

Status/error code* (W#16#...)	Description	Remedy
7102	Warning: powerrideConfig.switchPos[1..36].ledColor invalid	Use 0..7 See LED color palette (Page 168), perform a reset of "enable" to read in the corrected configuration again
7103	Warning: powerrideConfig.switchPos[1..36].vibrationIndex invalid[4)]	Use 0..7 Perform a reset of "enable" to read in the corrected configuration again
8101	Positive "preset" edge, but "presetValue"[3)] is greater than 36	Use 1..36 Reset of "preset"
8102	Positive "preset" edge, but "presetValue" is less than 1	Use 1..36 Reset of "preset"
8212	Configuration error[2)]: "posMax" is greater than 36	Use 1..36 Reset of "enable"
8221	Configuration error: "pos100" is less than 1	Use 1..36 Reset of "enable"
8222	Configuration error: "posMax" is less than 1	Use 1..36 Reset of "enable"
8223	Configuration error: "pos100" is greater than "posMax"	Use pos100 ≤ posMax Reset of "enable"
8231	Configuration error: "switchPos[1..36].value" is outside the valid range (0..200 %) in one or more data sets	Use 0..200 Correct data sets, reset of "enable"
* Status code: 7xxx (information, alarm), parameter error = FALSE; error code: 8xxx (error), parameter error = TRUE		

Notes regarding the table

1) If a warning occurs, the other functionality of the Powerride is not restricted.

2) If a configuration error occurs, the switchPos[1..36] is output for safety reasons. After eliminating configuration errors, the signal at the "enable" input must be set to FALSE and then back to TRUE in order to re-read the configuration data and reset the error.

3) If errors occur at the "presetValue" parameter, the "preset" input must be set to FALSE after correction and then set to TRUE again in order to accept the now correct "presetValue" with the positive edge.

4) The selectable vibration patterns are described in the *Equipment Manual SINUMERIK ONE MCP Part 1: MCP xxxx* in Chapter "Commissioning" in the "Vibration patterns Powerride" section.

More information

A detailed description of the signal transmission between Powerride and PLC can be found in the *Equipment Manual SINUMERIK ONE MCP Part 1: MCP xxxx* in Chapter "Commissioning" in the "Input images" and "Output images" sections.

For a description of the LED colors, see the "LED Color Palette (Page 168)" section.

See also

Create data block MyPowerrideConfig (Page 164)

3.13.2 LAPP_UpdateEthernetMcp

3.13.2.1 Overview

Note

- This function is not available in the virtual SINUMERIK
- The "LAPP_UpdateEthernetMcp" instruction is not supported for PLC-UCP combination.

The instruction is used for communication with operator components via Industrial Ethernet (IE), e.g. machine control panels (MCP), handheld terminals (HT), and direct key modules.

For each of these operator components, the instruction "LAPP_UpdateEthernetMcp" must be called cyclically using a separate instance every time.

The connection parameters MCP address, connection ID, interface ID and local UDP port form part of the input parameters. It is also specified at the input whether this is a direct key module. The connection parameters and the input "directKeys" can only be changed if enable = FALSE. The input "disableDataToMcp" (no data can be sent to the operator component) can be changed cyclically.

If the process of establishing the connection is started with "enable", the first step is to identify the operator component. If the "identified" output parameter is TRUE, the MCP type and the length of the required input and output data can be read. A list of known and supported types is provided in Section "Device types". If identification was successful, the cyclic connection is then established and the output mcpReady is set to TRUE.

In the cyclic data exchange, the data in parameter "toMcp" is sent to the operator component and the data received from the operator component is written to parameter "fromMcp". Data are sent and received at defined intervals. Output "newData" is set if a new data package is available.

To create the input and output images at parameters "fromMcp" and "toMcp", the supplied data types of SINUMERIK ONE MCP or other operator panels can be used. User-defined types or variables can also be parameterized.

If an operator panel fails or a connection to the operator panel is lost, error information is output in "error" and "status". In addition, if the connection is lost, the input image is deleted and the output "newData" is set to TRUE once to allow a response to the failure. If "enable" stays TRUE, there is an automatic attempt to restore the connection.

To end the connection, "enable" must be set to FALSE. The active Ethernet connection is ended and the instruction is not then processed any further. The input image contains the most recent status and is not deleted as it is when an error occurs.

More information

For details on the configuration and use, refer to "Information system > Configuring and programming SINUMERIK > Programming SINUMERIK PLC > Programming SINUMERIK PLC (ONE) > PLC Basic Program plus > Quick introduction > Communication configuration Ethernet-MCP and HT 10" in TIA Portal online help.

Call time

- Once per PLC cycle for each instance, e.g. in the main OB

3.13.2.2 Block interface - LAPP_UpdateEthernetMcp

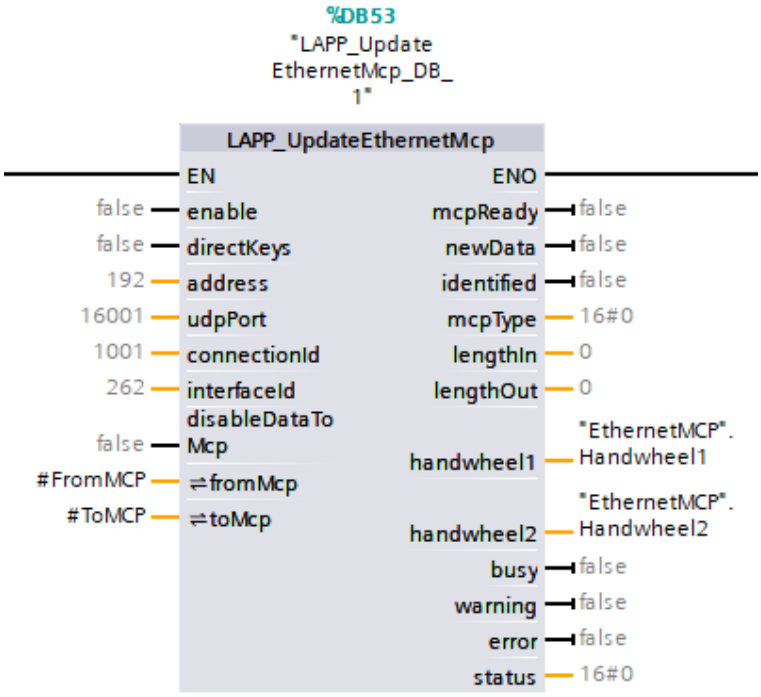


Figure 3-105 "LAPP_UpdateEthernetMcp"

Parameter

The following table shows the parameters of the instruction:

Table 3-53 Parameters of "LAPP_UpdateEthernetMcp"

Parameter	Declaration	Data type	Description
enable	Input	Bool	TRUE: Establish a connection and communicate cyclically
directKeys	Input	Bool	TRUE: Establish a connection to direct keys instead of the MCP
address	Input	USInt	Device address according to the DIP switch at the MCP
udpPort	Input	UInt	Local UDP port on the PLC Value range: 1...49151 Must not be simultaneously used for other instructions. Use a separate UDP port for each connection. The system has reserved the following ports and these may not be used: 20, 21, 80, 102, 135, 161, 162, 443, 34962, 34963, 34964.

Parameter	Declaration	Data type	Description
connectionId	Input	CONN_OUC	Connection ID on the PLC Value range: 1...4095 Must not be simultaneously used for other instructions. Use a separate connection ID for each connection.
interfaceId	Input	HW_ANY	Interface ID of the network interface via which the MCP is connected, see hardware configuration. (System constant "Local~CP~Ethernet interface_GBIT,,," can be used or the numerical value, set as default to 262). Default: 262
disableData-ToMcp	Input	Bool	TRUE: Do not send data to the MCP
mcpReady	Output	Bool	TRUE: Cyclic communication taking place with the operator component
newData	Output	Bool	TRUE: New data package is available
identified	Output	Bool	TRUE: Identification successful
mcpType	Output	Byte	Type identifier of the MCP (see section "Device types")
lengthIn	Output	USInt	Length of the data from the MCP
lengthOut	Output	USInt	Length of the data to the MCP
handwheel1	Output	LAPP_typeHandwheel-Data	Data for handwheel 1: active, recent, stopped, pulses
handwheel2	Output	LAPP_typeHandwheel-Data	Data for handwheel 2: active, recent, stopped, pulses
busy	Output	Bool	TRUE: Connection established to the operator component
warning	Output	Bool	TRUE: Data reception taking longer than expected
error	Output	Bool	TRUE: Error during cyclic communication when the connection is established. The cause of the error is displayed in the "status" parameter.
status	Output	Word	Instruction status or error information. For the interpretation of the "status" parameter, see section "Errors, status and remedies"
fromMcp	InOut	Variant	Data from MCP
toMcp	InOut	Variant	Data to MCP

Pulse diagrams

Basic information on processing the instruction can be found in "Information system > Configuring and programming SINUMERIK > Programming SINUMERIK PLC > Programming SINUMERIK PLC (ONE) > PLC instructions SINU_* > Execution of instructions and blocks > Continuous asynchronous processing" in TIA Portal online help.

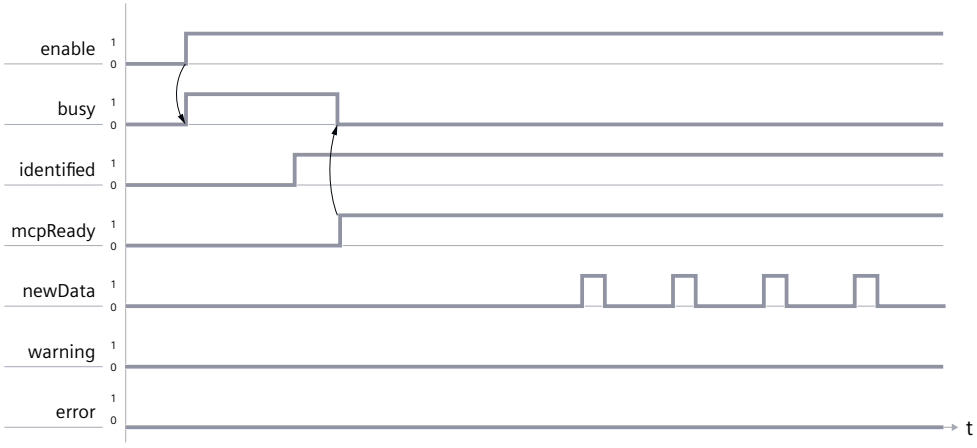


Figure 3-106 LAPP_UpdateEthernetMcp: Establishing the cyclic connection to a SINUMERIK ONE MCP

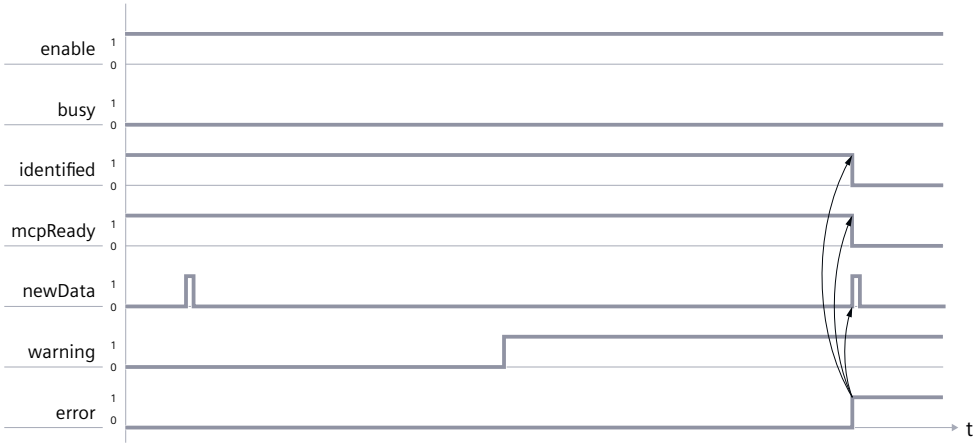


Figure 3-107 LAPP_UpdateEthernetMcp: Warning and error timeout

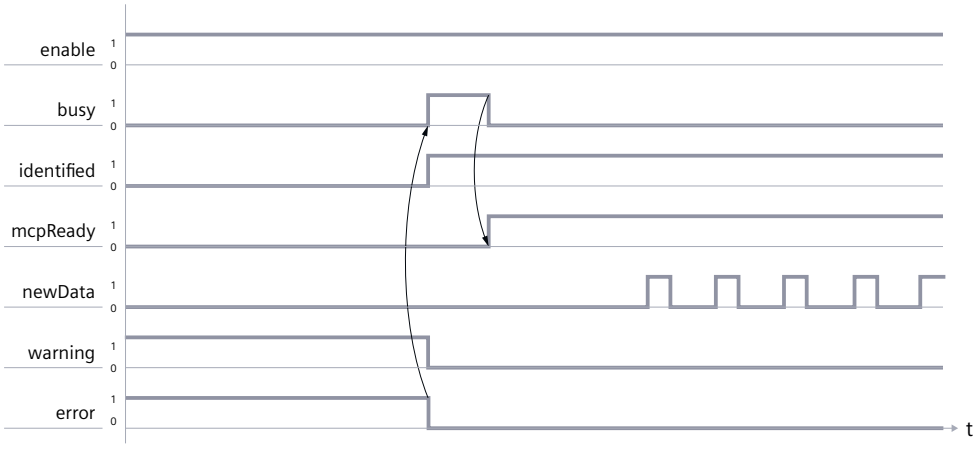


Figure 3-108 LAPP_UpdateEthernetMcp: Connection established after timeout

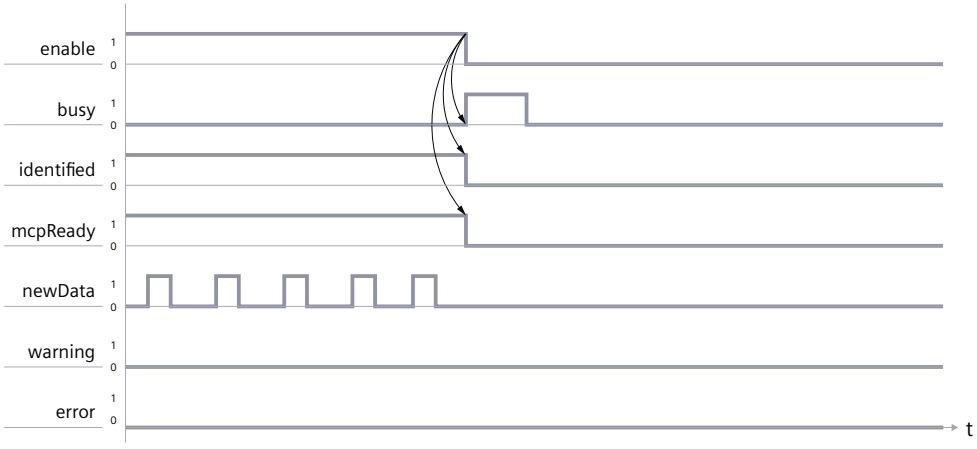


Figure 3-109 LAPP_UpdateEthernetMcp: enable = "False", connection ended

Device types

The SINUMERIK ONE (NC 63) (https://mall.industry.siemens.com/mall/en/WW/Catalog/Products/10028455) catalog contains the supported operator components for the SINUMERIK ONE which are related to PLC Basic Program plus.

Apart from the operator components listed in the catalog, you can identify more operator components with the instruction "LAPP_UpdateEthernetMcp". The list below contains the known operator components with the associated type identifier (corresponding to output "mcpType").

Type identifier	Designation
16#80	MCP 483C
16#81	MCP 483
16#82	MCP 310
16#83	MCP OEM
16#84	MCP DMG

Type identifier	Designation
16#85	HT 8
16#86	TCU DT
16#87	MCP MPP
16#88	HT 2
16#89	OP 08T
16#8A	MCP 310C PN
16#8B	MCP Index OEM
16#8C	RCU/TCU30
16#8D	MCP Interface PN
16#8E	VDCK (virtual direct control keys)
16#90	MCP 398C
16#91	HT 10
16#A0	SINUMERIK ONE MCP

Example

In the following example, call "LAPP_UpdateEthernetMcp" is programmed with instance DB "InstLAPP_UpdateEthernetMcpOneMcp". The control and status data are created in a global data block "dataMcp". The structures of the in/out parameters "toMcp" and "fromMcp" are selected for a SINUMERIK ONE MCP with 1 basic module, 4 key modules and 2 Powerrides.

To create the input and output image of the SINUMERIK ONE MCP, the PLC data types provided in the SINUMERIK STEP 7 Toolbox were used. For more information, refer to "Information system > Configuring and programming SINUMERIK > Programming SINUMERIK PLC > Programming SINUMERIK PLC (ONE) > PLC Basic Program plus > Preparing the PLC project > Inserting PLC data types Basic Program".

The monitor value in the image shows the outputs at the stage of cyclic communication following successful identification of the SINUMERIK ONE MCP.

OneMcpData

	Name	Data type	Default value	Start value
1	Static			
2	fromMcp	Struct		
3	base	"LBP_typeModularMcpBaseInput"		
4	keypad1	"LBP_typeModularMcpUniversalInput"		
5	keypad2	"LBP_typeModularMcpUniversalInput"		
6	keypad3	"LBP_typeModularMcpUniversalInput"		
7	keypad4	"LBP_typeModularMcpUniversalInput"		
8	powerride1	LAPP_typePowerrideInputs		
9	powerride2	LAPP_typePowerrideInputs		
10	toMcp	Struct		
11	base	"LBP_typeModularMcpBaseOutput"		
12	keypad1	"LBP_typeModularMcpUniversalOutput"		
13	keypad2	"LBP_typeModularMcpUniversalOutput"		
14	keypad3	"LBP_typeModularMcpUniversalOutput"		
15	keypad4	"LBP_typeModularMcpUniversalOutput"		
16	powerride1	LAPP_typePowerrideOutputs		
17	powerride2	LAPP_typePowerrideOutputs		
18	handwheel1	LAPP_typeHandwheelData		
19	active	Bool	false	TRUE
20	recent	Bool	false	TRUE
21	stopped	Bool	false	FALSE
22	pulses	Int	0	0
23	handwheel2	LAPP_typeHandwheelData		
24	enable	Bool	false	TRUE
25	directKeys	Bool	false	FALSE
26	disableDataToMcp	Bool	false	FALSE
27	mcpReady	Bool	false	TRUE
28	newData	Bool	false	TRUE
29	identified	Bool	false	TRUE
30	mcpType	Byte	16#0	16#A0
31	lengthIn	USInt	0	26
32	lengthOut	USInt	0	30
33	busy	Bool	false	FALSE
34	warning	Bool	false	FALSE
35	error	Bool	false	FALSE
36	status	Word	16#0	16#0000

Figure 3-110 Control and status variables in the data block "dataMcp"

Call LAPP_UpdateEthernetMcp

SCL

```
// InstLAPP_UpdateEthernetMcpOneMcp : LAPP_UpdateEthernetMcp
#InstLAPP_UpdateEthernetMcpOneMcp(enable := TRUE,
                                  address := 193,
                                  udpPort := 16001,
                                  connectionId := 1001,
                                  interfaceId := 262,
                                  // "Local~CP~Ethernet interface_GBIT,,,"
                                  mcpReady => "dataMcp".OneMcpData.mcpReady,
                                  newData => "dataMcp".OneMcpData.newData,
                                  identified => "dataMcp".OneMcpData.identified,
                                  mcpType => "dataMcp".OneMcpData.mcpType,
                                  lengthIn => "dataMcp".OneMcpData.lengthIn,
                                  lengthOut => "dataMcp".OneMcpData.lengthOut,
                                  handwheel1 => "dataMcp".OneMcpData.handwheel1,
                                  handwheel2 => "dataMcp".OneMcpData.handwheel2,
                                  busy => "dataMcp".OneMcpData.busy,
                                  warning => "dataMcp".OneMcpData.warning,
                                  error => "dataMcp".OneMcpData.error,
                                  status => "dataMcp".OneMcpData.status,
                                  fromMcp := "dataMcp".OneMcpData.fromMcp,
                                  toMcp := "dataMcp".OneMcpData.toMcp);
```

3.13.2.3 Status codes

Error, status, and remedy

Status/error code* (W#16#...)	Description	Remedy
0000	No error	---
7000	Waiting for start of job	---
7131	Connection established to the MCP	---
7132	Data sent to the MCP	---
7133	Data received from the MCP	---
7134	Connection to the MCP ended	---
7135	The area at the "fromMcp" parameter is too small. Data loss.	Make the area at the "fromMcp" parameter at least as large as "lengthIn".
8202	Area too long at the "fromMcp" or "toMcp" parameter	Check the length of the "fromMcp" and "toMcp" parameters. Use variables with maximum 224 bytes.
8203	Impermissible memory area at the "fromMcp" or "toMcp" parameter	Check "fromMcp" and "toMcp" parameters. Use data area E, A, D or instruction interface (Input, Output, Static, Temp).
8116	MCP did not respond within the specified time during identification	Check the specified MCP address and the connection to the MCP
8117	MCP did not respond within the specified time during parameterization	Check connection to the MCP

Status/error code* (W#16#...)	Description	Remedy
8118	MCP did not respond within the specified time during cyclic communication	Check connection to the MCP
8119	Error when connecting: A connection with this ID is already established	Use an unused connection ID, check the "connectionId" parameter
811a	Error when connecting: Port is already being used or is reserved by the system	Use an unused UDP port, check parameter "udpPort". The system has reserved the following ports and these may not be used: 20, 21, 80, 102, 135, 161, 162, 443, 34962, 34963, 34964.
8234	Error when connecting: Incorrect interface ID at the "interfaceId" parameter	Check "interfaceId" parameter
8235	Error when connecting: Incorrect connection ID at the "connectionId" parameter	Check "connectionId" parameter
8240	NULL at the "toMcp" parameter	Check "toMcp" parameter
8241	NULL at the "fromMcp" parameter	Check "fromMcp" parameter
8242	Error when sending: Incorrect MCP address	Check "address" parameter and connection to the MCP
8243	The operator component does not have an IPv4 address	Check whether this operator component is supported
8244	The parameterization data for this operator component could not be determined	Check whether this operator component is in the list of known types
8245	The MCP parameterization data are invalid	Check whether the MCP parameterization data defined for the determined type are correct. Regenerate an instance for "LAPP_UpdateEthernetMcp".
8246	The MCP parameterization has been rejected by the operator component	Check whether the MCP parameterization data defined for the determined type are correct. Regenerate an instance for "LAPP_UpdateEthernetMcp".
86xx	Internal errors	Please contact Siemens.
* Status code: 7xxx (information, alarm), parameter error = FALSE; error code: 8xxx (error), parameter error = TRUE		

More information

LAPP_WriteRecordOverEthernet (Page 182)

3.13.3 LAPP_WriteRecordOverEthernet

3.13.3.1 Overview

Description

The instruction transfers a data set (record) to an MCP which is connected to the SINUMERIK ONE by Ethernet.

Note

This function is not available in the virtual SINUMERIK.

More information

For an MCP connected by PROFINET IO, the instruction "WRREC" (write data set) is available on the PLC.

For more information about "WRREC" refer to "Information system > Programming a PLC > Instructions > Instructions (S7-1200, S7-1500) > Extended instructions > Distributed I/O > WRREC: Write data record" in TIA Portal online help.

Call time

- Once per PLC cycle for each instance, e.g. in the main OB

3.13.3.2 Block interface - WriteRecordOverEthernet

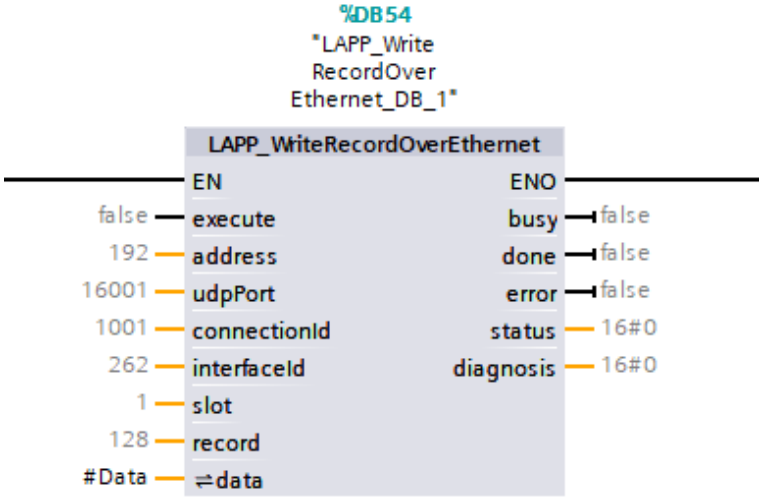


Figure 3-111 "LAPP_WriteRecordOverEthernet"

Parameter

The following table shows the parameters of the instruction:

Table 3-54 Parameters of "LAPP_WriteRecordOverEthernet"

Parameter	Declaration	Data type	Description
execute	Input	Bool	TRUE: Transfer the data set, see the pulse diagram
address	Input	USInt	Device address according to the DIP switch at the MCP Default: 192
udpPort	Input	UInt	Local UDP port on the PLC Must not be simultaneously used for other instructions. Value range: 1...49151. Default: 16001 The system has reserved the following ports and these may not be used: 20, 21, 80, 102, 135, 161, 162, 443, 34962, 34963, 34964.
connectionId	Input	CONN_OUC	Connection ID on the PLC Must not be simultaneously used for other instructions. Value range: 1...4095 Default: 1001
interfaceId	Input	HW_ANY	Interface ID of the network interface over which the MCP is connected, see also hardware configuration. (System constant "Local~CP~Ethernet interface_GBIT,,," can be used or the numerical value, set as default to 262). Default: 262
slot	Input	UInt	Slot of the SINUMERIK ONE MCP key module to which the sent data set RECORD refers. Default: 1
record	Input	UInt	Data set number of the data set to be written. 128 for the configuration data of the MCP Default: 128
busy	Output	Bool	TRUE: The write operation is active/not yet complete.
done	Output	Bool	TRUE: The data record was successfully transferred/written.
error	Output	Bool	TRUE: An error occurred while writing. The cause of the error is displayed in the "status" parameter.
status	Output	Word	Instruction status or error information. For the interpretation of the "status" parameter, see the Status and error codes table.
diagnosis	Output	Word	Diagnostic output for additional information about the status
data	InOut	Variant	Data set to be transferred.

Pulse diagram

Basic information on processing the instruction can be found in "Information system > Configuring and programming SINUMERIK > Programming SINUMERIK PLC > Programming SINUMERIK PLC (ONE) > PLC instructions LAPP_* > Execution of instructions and blocks > Continuous asynchronous processing" IN TIA Portal online help.

Detailed description of parameter "slot"

The configuration of the MCP module or the input/output images results from the arrangement of the key modules in a fixed sequence with variable number.

Fixed sequence of modules:

1. Basis (always available, slot 1)
2. Key modules (1 to 4, from slot 2)
3. Powerrides (1 to 3)
4. 2 handwheels (always available)

Basic module	**Key module 1**	**Key module 2**	**Key module 3**	**Key module 4**	Powerride 1	Powerride 2	Powerride 3	Handwheel 1	Handwheel 2
Slot 1	**Slot 2**	**Slot 3**	**Slot 4**	**Slot 5**					

Structure of the data set 128 (16#80) for parameter "data"

Note

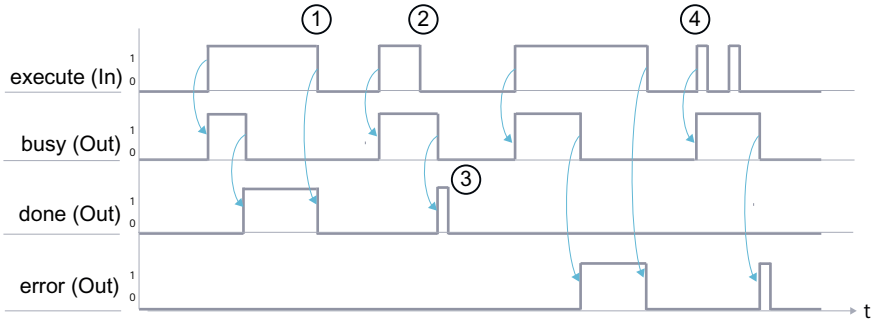
Use the data types "Modular MCP data types" provided in the PLC basic program library to structure the RECORD data set.

- SINUMERIK ONE: Data type "LBP_typeModularMcpKeypadRecord"

The values for a maximum of 21 LED keys can be adapted in a data set. For each key module, create an instance for data set 128, if you wish to change the color values of the LED keys in the PLC user program.

More information about the structure of data set 128 is provided in the Equipment Manual of the SINUMERIK ONE MCP in the "Structure of parameter data set 128 for the complete module" section.

Pulse diagram



The job is started on a rising edge at parameter "execute". The values at the input parameters are applied and remain valid for the entire duration of the job.

The instruction must be called until it reports "done" or "error".

The job cannot be aborted.

Once the job has been completed, a positive edge at "execute" is required to start a new job.

Multiple, parallel calls with different instances of "LAPP_WriteRecordOverEthernet" can be started. The instruction processes the calls internally one after the other, not in parallel.

① "done" and "error" are reset at "execute" with a falling edge.

② The functionality of the FB is not stopped with a falling edge on "execute". If input "execute" is reset before output "done" is set, output "done" is only set for one cycle.

③ If "execute" is already FALSE, "done" and "error" are only active for one cycle.

④ A new job is requested with a positive edge at "execute" while the instruction is still being processed ("busy" = TRUE).
The old job is ended with the parameters active at the beginning of the job.

Figure 3-112 Pulse diagram LAPP_WriteRecordOverEthernet

Example: Setting the key color of a SINUMERIK ONE MCP

In the following example, the colors of the LED keys of a 3x5 key module for their two states - active and inactive - should be changed with respect to the standard assignment.

The key module is part of a SINUMERIK ONE MCP1900, configured as machine control panel 1 (called with "LAPP_UpdateEthernetMcp"). Keys 4 and 12 of key module 2, customer key block (see the following diagram) are adapted.



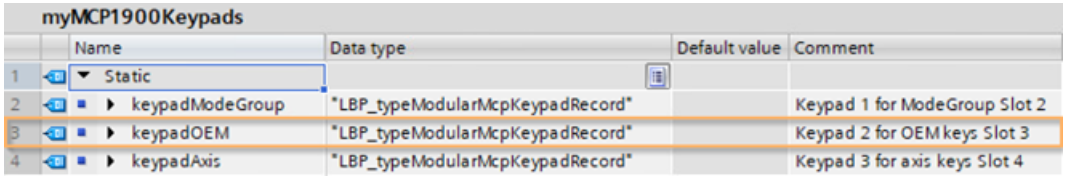
Figure 3-113 Key module 2, slot 3

The following values should apply to the two keys:

Key number	Color code for the active status	Color code for the inactive status
4	(0,0,255) blue	(255,255,0) yellow
12	(0,0,255) blue	(255,255,0) yellow

Preparations in the PLC user project

1. Create an instance for the data set of the required key module 2 of the SINUMERIK ONE MCP in a global data block "myMCP1900keypads".

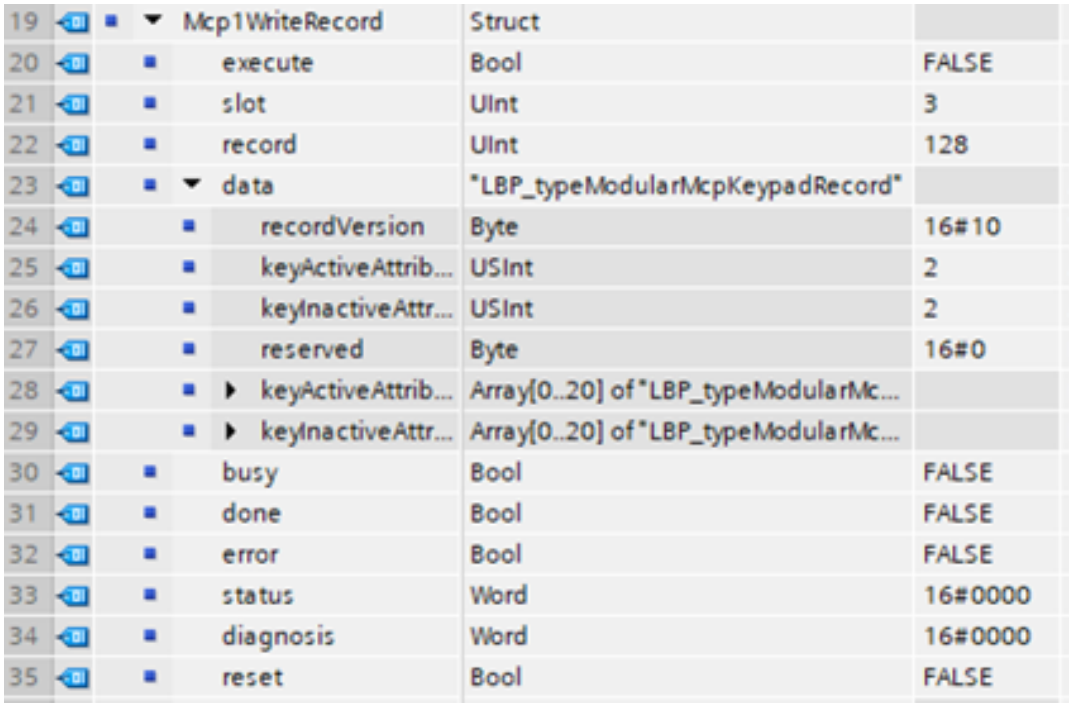


myMCP1900Keypads

	Name	Data type	Default value	Comment
1	Static			
2	keypadModeGroup	"LBP_typeModularMcpKeypadRecord"		Keypad 1 for ModeGroup Slot 2
3	keypadOEM	"LBP_typeModularMcpKeypadRecord"		Keypad 2 for OEM keys Slot 3
4	keypadAxis	"LBP_typeModularMcpKeypadRecord"		Keypad 3 for axis keys Slot 4

Figure 3-114 Data block with symbolic name "myMCP1900Keypads"

2. Create the following control and status variables in a data block "dataMcp".



19	Mcp1WriteRecord	Struct	
20	execute	Bool	FALSE
21	slot	UInt	3
22	record	UInt	128
23	data	"LBP_typeModularMcpKeypadRecord"	
24	recordVersion	Byte	16#10
25	keyActiveAttrib...	USInt	2
26	keyInactiveAttr...	USInt	2
27	reserved	Byte	16#0
28	keyActiveAttrib...	Array[0..20] of "LBP_typeModularMc...	
29	keyInactiveAttr...	Array[0..20] of "LBP_typeModularMc...	
30	busy	Bool	FALSE
31	done	Bool	FALSE
32	error	Bool	FALSE
33	status	Word	16#0000
34	diagnosis	Word	16#0000
35	reset	Bool	FALSE

Figure 3-115 Control and status variables in the DB "dataMcp"

3. Adapt the values for keys 4 and 12 of key module 2 (slot 3) in the data block ("myMCP1900Keypads".keypadOEM) that has been created as follows. You can either assign these in your user program or directly in the data block.

Name	Data type	Value to be set	Comment
Header information			
keypadOEM.recordVersion	Byte	B#16#10	Version 1.0
keypadOEM.keyActiveAttribCount	USInt	2	number of valid entries in keyActiveAttributes
keypadOEM.keyInactiveAttribCount	USInt	2	number of valid entries in keyInactiveAttributes
Parameter block 1 for active status for maximum 21 LEDs			
keypadOEM.keyActiveAttributes[0].key-Index	USInt	4	Index of key
keypadOEM.keyActiveAttributes[0].red	USInt	0	intensity of red LED
keypadOEM.keyActiveAttri-butes[0].green	USInt	0	intensity of green LED
keypadOEM.keyActiveAttributes[0].blue	USInt	255	intensity of blue LED
keypadOEM.keyActiveAttributes[1].key-Index	USInt	12	Index of key
keypadOEM.keyActiveAttributes[1].red	USInt	0	intensity of red LED
keypadOEM.keyActiveAttri-butes[1].green	USInt	0	intensity of green LED
keypadOEM.keyActiveAttributes[1].blue	USInt	255	intensity of blue LED
Parameter block 2 for inactive status for maximum 21 LEDs			
keypadOEM.keyInactiveAttri-butes[0].keyIndex	USInt	4	Index of key
keypadOEM.keyInactiveAttributes[0].red	USInt	255	intensity of red LED
keypadOEM.keyInactiveAttri-butes[0].green	USInt	255	intensity of green LED
keypadOEM.keyInactiveAttri-butes[0].blue	USInt	0	intensity of blue LED
keypadOEM.keyInactiveAttri-butes[1].keyIndex	USInt	12	Index of key
keypadOEM.keyInactiveAttributes[1].red	USInt	255	intensity of red LED
keypadOEM.keyInactiveAttri-butes[1].green	USInt	255	intensity of green LED
keypadOEM.keyInactiveAttri-butes[1].blue	USInt	0	intensity of blue LED
...			

Note

The index in the array field of your instance does not correspond to the key number. The key number is specified under "keyIndex".

Example

In the first storage space (index 0), key 4 is to be configured.

KeypadOEM.keyActiveAttributes[0].keyIndex := 4

4. The "slot" and "record" parameters were explained under point 3. The connection parameters "address", "udpPort", "connectionId" and "InterfaceID" have default values in the example and may need to be changed. The write operation is started when the "dataMcp".Mcp1WriteRecord.execute" bit is set in the DB "dataMcp". "busy" indicates whether the operation is still active. "done" is set if the write operation is successful. If an error occurs, the "error" bit changes to TRUE and the error status is output in "status". If either of the "done" or "error" bits is TRUE, "execute" can be reset (see the pulse diagram).

Call LAPP_WriteRecordOverEthernet

SCL

```
// Write data set 128 to customer key module 2 (slot 3) of the SINUMERIK ONE MCP
#InstWriteRecordOverEthernet(execute := "dataMcp".Mcp1WriteRecord.execute,
                             address := 192,
                             udpPort := 16001,
                             connectionId := 1001,
                             interfaceId := 262,
                             slot := "dataMcp".Mcp1WriteRecord.slot, // Slot 3
                             record := "dataMcp".Mcp1WriteRecord.record, // Record No. 128
                             busy => "dataMcp".Mcp1WriteRecord.busy,
                             done => "dataMcp".Mcp1WriteRecord.done,
                             error => "dataMcp".Mcp1WriteRecord.error,
                             status => "dataMcp".Mcp1WriteRecord.status,
                             diagnosis=>"dataMcp".Mcp1WriteRecord.diagnosis,
                             data := "dataMcp".Mcp1WriteRecord.data);

IF "dataMcp".Mcp1WriteRecord.done THEN         // Write job completed without error
  "dataMcp".Mcp1WriteRecord.execute := FALSE;  // Reset "execute"
END_IF;
IF "dataMcp".Mcp1WriteRecord.error AND "dataMcp".Mcp1WriteRecord.reset THEN
  // Write job completed with error
  // Evaluate the error using output parameters "status" and "diagnosis"
  // Reset "execute" with "error" using the reset signal "reset"
  "dataMcp".Mcp1WriteRecord.execute := FALSE;
END_IF;
```

3.13.3.3 Status codes

Error, status, and remedy

Status/error code* (W#16#...)	Description	Remedy
0000	No error	---
7000	Waiting for start of job	---
7132	Record sent to MCP	---
7133	Waiting for reply from MCP	---
8017	Write operation already running/not completed yet	Only set execute when "busy" is FALSE
8116	MCP did not respond within the specified time	Check connection to the MCP
8119	Error when connecting: A connection with this ID is already established	Use a different connection ID, "connectionId" parameter
811a	Error when connecting: Port is already being used or is reserved by the system	Check parameter "udpPort". The system has reserved the following ports and these may not be used: 20, 21, 80, 102, 135, 161, 162, 443, 34962, 34963, 34964.
8202	Record data too long at the "data" parameter	Check length of data package "data"
8203	Impermissible memory area at the "data" parameter	Check "data" parameter
8234	Error when connecting: Incorrect interface ID at the "interfaceId" parameter	Check "interfaceId" parameter
8235	Error when connecting: Incorrect connection ID at the "connectionId" parameter	Check "connectionId" parameter
8236	Impermissible record version at the "data" parameter (header information)	Check record version in the "data" parameter
8237	Impermissible record number at the "record" parameter	Use a correct record number at the "record" parameter
8238	For "record" :=128: impermissible configuration for key, Impermissible key index is output at the "diagnosis" output.	Parameterize the correct key at the "data" parameter
8239	Impermissible configuration, check parameter	Check "slot", "record" and "data" parameters
8240	NULL at the "data" parameter	Check "data" parameter
8242	Error when sending: Incorrect MCP address	Check "address" parameter
86xx	Internal errors	Please contact Siemens.
* Status code: 7xxx (information, alarm), parameter error = FALSE; error code: 8xxx (error), parameter error = TRUE		

Business logic blocks

4

4.1 GRAPH for manual operation

4.1.1 Overview

Machining step programming using GRAPH is suitable for programming sequential machine sequences in PLC-controlled machines.
This chapter describes using GRAPH blocks in manual operation screens for manual operation via GRAPH.

4.1.2 PLC data types for manual operation using GRAPH blocks

Data types and storage in the project navigation

The following data types are required for manual operation using GRAPH blocks.

- **LAPP_typeBlGraphRowDataConfig**
  Operating manual operation lines using the GRAPH block
- **LAPP_typeBlGraphExtnStatParam**
  Internal processing
- **LAPP_typeBlGraphExtnInParam**
  Internal processing
- **LAPP_typeBlGraphExtnOutParam**
  Internal processing

You can find the data types under: control [controlType] > PLC data types > Libraries > CMH Automotive Program Vxx.x.x.x > BusinessLogic Types > GraphManualOperation.

LAPP_typeBlGraphRowDataConfig

It is specified as an array[1..x] in a data block.

Name	Type	Description
left.instanceName	Wstring[80]	Name of the instance DB for operation on the left side. Specify the instance DB in quotation marks.
left.stepNumber	Int	Step number for operation on the left side

Name	Type	Description
left.activationPossible	Bool	True: Step can be executed. This parameter can be assigned to input parameter **fieldDevAckLeft** of block **LAPP_ManualOperation** The operator value is set using **LAPP_blGraphManExtn**.
left.activateStepOFF	Bool	True: Left step is deactivated. The operator value is set using **LAPP_blGraphManExtn**.
left.activateStepON	Bool	True: Left step is activated. The operator value is set using **LAPP_blGraphManExtn**.
right.instanceName	Wstring[80]	Name of the instance DB for operation on the right side. Specify the instance DB in quotation marks.
right.stepNumber	Int	Step number for operation on the right side.
right.activationPossible	Bool	True: Step can be executed. This parameter can be assigned to input parameter **fieldDevAckRight** of block **LAPP_ManualOperation.** The operator value is set using **LAPP_blGraphManExtn**.
right.activateStepOFF	Bool	True: Right step is deactivated. The operator value is set using **LAPP_blGraphManExtn**.
right.activateStepON	Bool	True: Right step is activated. The operator value is set using **LAPP_blGraphManExtn**.

4.1.3 Program blocks for manual operation using GRAPH blocks

4.1.3.1 Program blocks and storage in the project navigation

In the PLC, the following blocks are used for manual operation using GRAPH blocks:

- LAPP_blGraphManConfigData
- LAPP_blGraphManExtn
- LAPP_blGraphManInParam
- LAPP_blGraphManOutParam

The blocks can be found under: Control [controlType] > Program blocks > Libraries > CMH Automotive Program Vxx.x.x.x > BusinessLogic Blocks > GraphManualOperation.

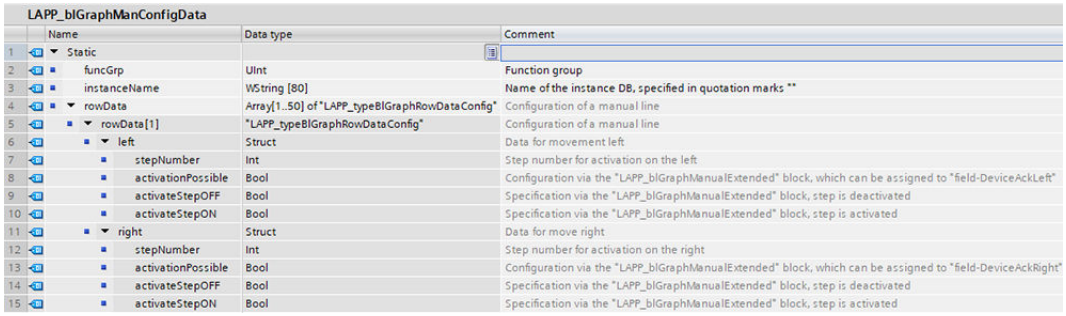
4.1.3.2 LAPP_blGraphManConfigData

Overview

This block is used for the configuration of the control of individual manual operation lines with the help of GRAPH blocks.

Every function group needs its own structure of the "LAPP_blGraphManConfigData" block. This can be achieved by either one block with multiple data structures or multiple blocks with one or multiple data structures.

This block is used to configure the line (instance of the graph and step number) for the corresponding movement. The index of the array element is the number of the manual operation line.
Example: rowData[1] corresponds to manual operation line 1.



LAPP_blGraphManConfigData

	Name	Data type	Comment
1	Static		
2	funcGrp	UInt	Function group
3	instanceName	WString [80]	Name of the instance DB, specified in quotation marks ""
4	rowData	Array[1..50] of "LAPP_typeBlGraphRowDataConfig"	Configuration of a manual line
5	rowData[1]	"LAPP_typeBlGraphRowDataConfig"	Configuration of a manual line
6	left	Struct	Data for movement left
7	stepNumber	Int	Step number for activation on the left
8	activationPossible	Bool	Configuration via the "LAPP_blGraphManualExtended" block, which can be assigned to "field-DeviceAckLeft"
9	activateStepOFF	Bool	Specification via the "LAPP_blGraphManualExtended" block, step is deactivated
10	activateStepON	Bool	Specification via the "LAPP_blGraphManualExtended" block, step is activated
11	right	Struct	Data for move right
12	stepNumber	Int	Step number for activation on the right
13	activationPossible	Bool	Configuration via the "LAPP_blGraphManualExtended" block, which can be assigned to "field-DeviceAckRight"
14	activateStepOFF	Bool	Specification via the "LAPP_blGraphManualExtended" block, step is deactivated
15	activateStepON	Bool	Specification via the "LAPP_blGraphManualExtended" block, step is activated

Figure 4-1 "LAPP_blGraphManConfigData"

Data type

LAPP_blGraphManConfigData parameters

Name	Data type	Description
funcGrp	UInt	Function group number used.
instanceName	Wstring[80]	Number of the instance DB. Specify the instance DB in quotation marks.
left.stepNumber	Int	Step number for the left activation.
left.activationPossible	Bool	True: Step can be executed This parameter can be assigned to input parameter fieldDevAckLeft of block LAPP_ManualOperation. The operator value is set using LAPP_blGraphManExtn.
left.activateStepOFF	Bool	The operator value is deactivated by LAPP_blGraphManExtn.
left.activateStepON	Bool	The operator value is activated by LAPP_blGraphManExtn.
right.stepNumber	Int	Step number for the right activation.
right.activationPossible	Bool	True: Step can be executed This parameter can be assigned input parameter fieldDevAckRight of block LAPP_ManualOperation. The operator value is set using LAPP_blGraphManExtn.
right.activateStepOFF	Bool	The operator value is deactivated by LAPP_blGraphManExtn.
right.activateStepON	Bool	The operator value is activated by LAPP_blGraphManExtn.

Error codes

List

Following are the error codes for "LAPP_blGraphManConfigData".

Error code	Description
W#16#7002	No error
W#16#8201	GRAPH version not correct
W#16#8202	GRAPH not in manual operation

4.1.3.3 LAPP_blGraphManExtn

Overview

It is not permissible that this block is called in the PLC program. It has no parameters. If you wish to execute manual operation via GRAPH, then you need to take into account the following property settings:

- GRAPH version: V6.0
- Extension block: LAPP_blGraphManExtn

Properties

GRAPH version

In the properties of your GRAPH block under "General > General > Block > Version" you need to select GRAPH version V6.0. It is mandatory for GRAPH blocks.

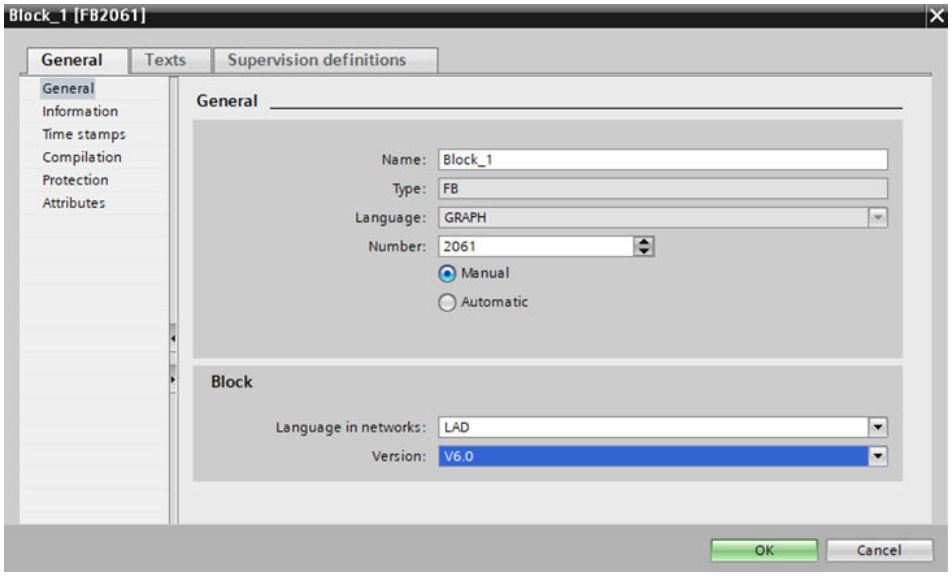


Figure 4-2 GRAPH version

Extension block

In the properties of your GRAPH block, enter the **LAPP_blGraphManExtn** block under General > Attributes > Attributes > Extension block name.

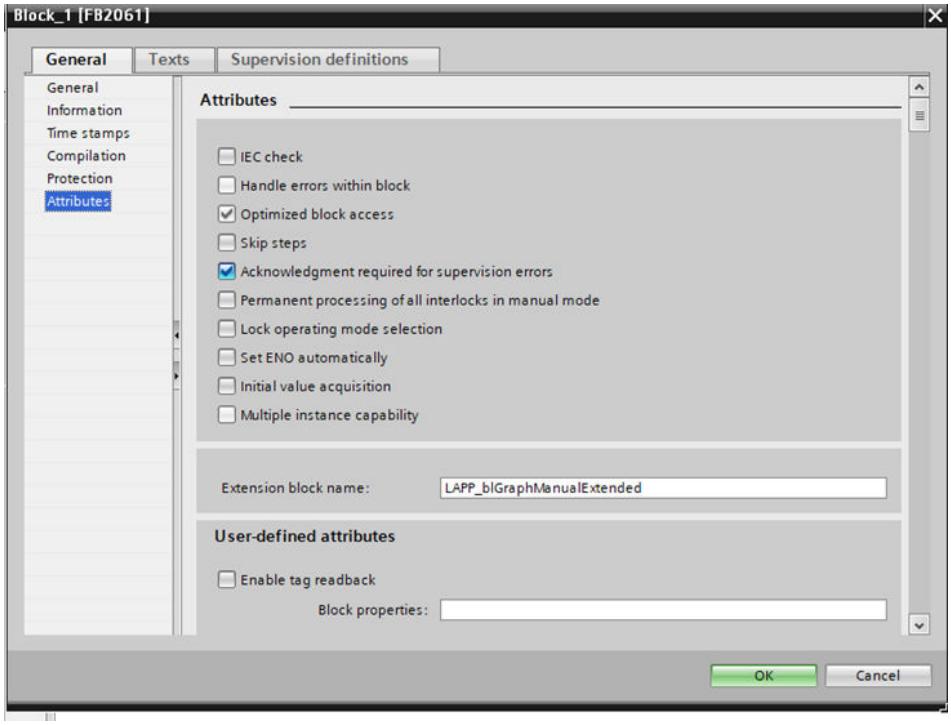


Figure 4-3 GRAPH attributes

Extend the GRAPH FB interface

For this purpose, in the interface of the GRAPH block, either a single-instance of the FB must be defined as I/O parameter – or a multi-instance in the static area of the GRAPH block.

GRAPH_Extn

Name	Data type	Comment
ERR_FLT	Bool	Interlock or supervision group error
AUTO_ON	Bool	Automatic mode is active
TAP_ON	Bool	Semiautomatic mode/step with transition enabled
TOP_ON	Bool	Semiautomatic mode/ignore transition enabled
MAN_ON	Bool	Manual mode is active
InOut		
extendedFunction	"LAPP_blGraphManExtn"	for to use as GRAPH extended FB for manual operation on mode setup
<Hinzufügen>		
Static		
RT_DATA	G7_RTDataPlus_V6	Internal data area
Trans1	G7_TransitionPlus_V6	Transition structure
Step1	G7_StepPlus_V6	Step structure

Figure 4-4 Extending the GRAPH FB interface via single instance

GRAPH_Extn_Multi

	Name	Data type	Comment
	TOP_ON	Bool	Semiautomatic mode/ignore transition enabled
	MAN_ON	Bool	Manual mode is active
	InOut		
	<Hinzufügen>		
	Static		
	RT_DATA	G7_RTDataPlus_V6	Internal data area
	Trans1	G7_TransitionPlus_V6	Transition structure
	Trans2	G7_TransitionPlus_V6	
	Trans3	G7_TransitionPlus_V6	
	Step1	G7_StepPlus_V6	Step structure
	Step2	G7_StepPlus_V6	
	Step3	G7_StepPlus_V6	
	extendedFunction	"LAPP_blGraphManExtn"	for to use as GRAPH extended FB for manual operation on mode setup
	<Hinzufügen>		
	Temp		
	<Hinzufügen>		
	Constant		

Figure 4-5 Extending the GRAPH FB interface via single instance

4.1.3.4 LAPP_blGraphManInParam

Overview

This block must be called in the preceding permanent instruction of the GRAPH block. It transfers the required input parameters to LAPP_blGraphManExtn.

The GetInstanceName instruction determines the instance name of the GRAPH and transfers it to the GRAPH interface.

Permanent pre-instruction

Block details

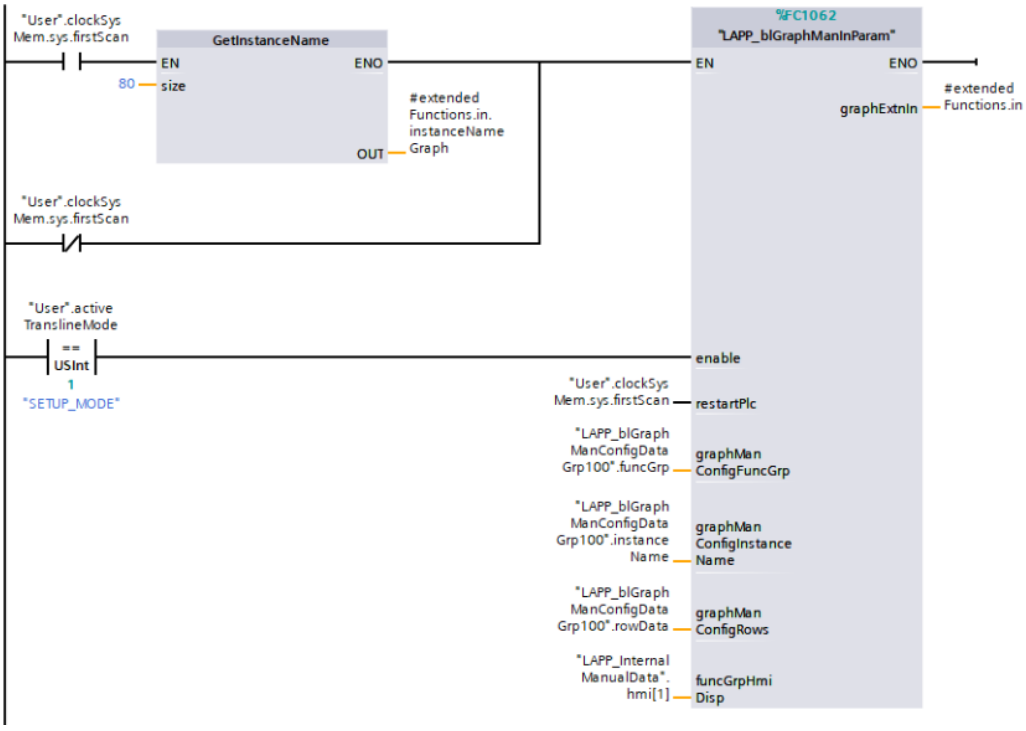


Figure 4-6 "LAPP_blGraphManInParam"

Parameters

Name	Declaration	Type	Description
enable	Input	Bool	Enable signal to process the block for manual operation. • True: Enabled • False: Not enabled
restartPLC	Input	Bool	Executing the block when restarting • True: Block is executed • False: Block is not executed
graphManConfigFuncGrp	Input	UInt	Function group number used.
graphManConfigInstance-Name	Input	WString[80]	Number of the instance DB. Specify the instance DB in quotation marks.
graphManConfigRows	Input	Array[*] of LAPP_type-BlGraphRowDataConfig	Configuration data for assigning the manual operation line number to GRAPH. The array index must correspond to the manual operation line number

Name	Declaration	Type	Description
funcGrpHmiDisp	Input	LAPP_typeFuncGrpHmiDisp	Internal line number.
graphExtnIn	Output	LAPP_typeBlGraphExtnInParam	Data for the GRAPH extension block.

4.1.3.5 LAPP_blGraphManOutParam

Overview

This block must be called in the permanent post-instruction of the GRAPH block.

"LAPP_blGraphManOutParam" transfers the calculated results to "LAPP_DataDB".

Permanent post-instruction

Block details

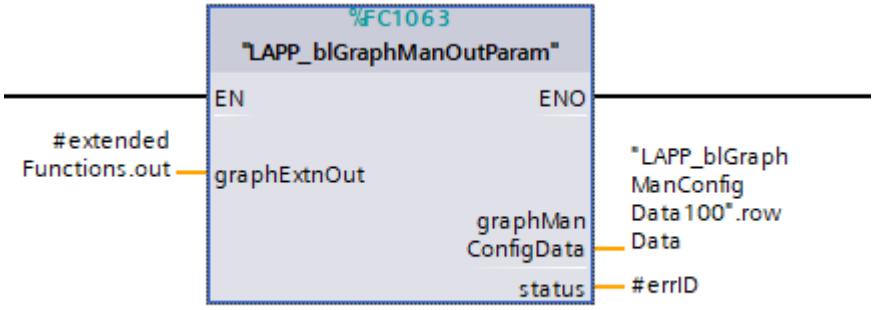


Figure 4-7 "LAPP_blGraphManOutParam"

Parameters

Name	Declaration	Type	Description
graphExtnOut	Input	LAPP_type-BlGraph Extende-dOutputPar-am	Extension block data.
graphManConfigData	Output	Array[*] of "LAPP_type-BlGraphRow-DataConfig"	Configuration data for the return to the user program.
status	Output	Word	Displays the status and error codes of the extended block. For more information, refer to Error codes (Page 199).

Error codes

List

Following are the error codes.

Error code	Description
W#16#0000	No error
W#16#8001	GRAPH version not correct
W#16#8002	GRAPH not in manual operation

4.2 Process data

4.2.1 LAPP_blPartCnt

4.2.1.1 Overview

The part counter block is used for counting all produced workpieces in a plant.
For recording the workpiece state, the data block requires data type "LAPP_typeBlPartState"; for the workpiece counting, it requires data type "LAPP_typeBlPartCnt".

4.2.1.2 Data type

LAPP_typeBlPartState

This data type is used for recording the availability and the state of a workpiece.

Depending on the number of workpiece types being produced simultaneously, either a variable of data type "LAPP_typeBlPartState" can be configured directly as parameter at the block or a variable as an ARRAY[1..x] of data type "LAPP_typeBlPartState" can be configured as parameter at the block.

Table 4-1 Structure of data type "LAPP_typeBlPartState"

Name	Data type	Description
PartExists	Bool	Workpiece is available
partIsGood	Bool	• TRUE: Workpiece is a good part • FALSE: Workpiece is a bad part

The counting is performed as follows:

- Workpiece is counted as a good part:
  PartExists=TRUE and PartIsGood=TRUE
- Workpiece is counted as a bad part:
  PartExists=TRUE and PartIsGood=FALSE
- Workpiece is not counted:
  PartExists=FALSE

LAPP_typeBlPartCnt

This data type is used for counting the workpieces.

If only one type of workpieces is produced, a variable of data type "LAPP_typeBlPartCnt" can be configured directly as parameter at the block. If several types of workpieces are produced, a variable as an Array[1..x] of data type "LAPP_typeBlPartCounter" can be configured as parameter at the block.

Name	Data type	Description
partName	WString[16]	Name of the workpiece
goodParts	Int	Actual number of good parts being produced since the last reset
badParts	Int	Actual number of bad parts being produced since the last reset

4.2.1.3 Block interface

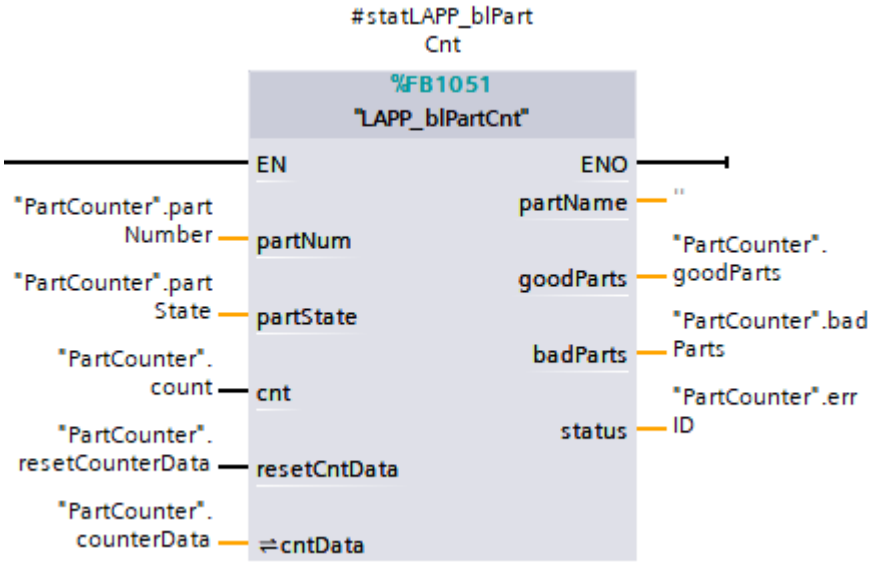


Figure 4-8 "LAPP_blPartCnt" block

Parameters

Name	Declaration	Type	Description
partNum	Input	Int	Array number of the workpiece type to be counted or array number of the workpiece type whose counters are to be reset • 1: Always the first element of the array, independent on the array boundaries. • 0: Counters of all workpiece types will be reset.
partState	Input	Variant	State of simultaneously produced parts of a workpiece type as "LAPP_typeBlPartState" or array [1..x] of "LAPP_typeBlPartState"
cnt	Input	Bool	The positive edge counts the number of produced parts, correspondent to "partNumber" and "partState" of the produced workpiece types.
resetCntData	Input	Bool	Resets the counter of the workpiece type, specified in "partNumber" If "partNumber" = 0, then counters of all workpiece types will be reset.
cntData	InOut	Variant	Storage for the workpieces to be produced Counts per feasible workpiece type as "LAPP_typeBlPartCnt" or array [1..x] of "LAPP_typeBlPartCnt"
partName	Output	WString [16]	Name of the counted workpiece
goodParts	Output	Int	Actual number of good parts since the last reset
badParts	Output	Int	Actual number of bad parts since the last reset
status	Output	Word	Displays error codes of the block. For more information, refer to Error codes (Page 202).

4.2.1.4 Error codes

List

Following are the error codes for "LAPP_typeBLPartState".

Error code	Description
16#7002	No error
16#8201	Parameter "partState" incorrect
16#8202	Parameter "partCounter" incorrect
16#8204	Parameter "partNumber" incorrect
16#8208	Parameter "partName" incorrect

4.2.1.5 Example

Connection to the LAPP_ShiftPlanning block

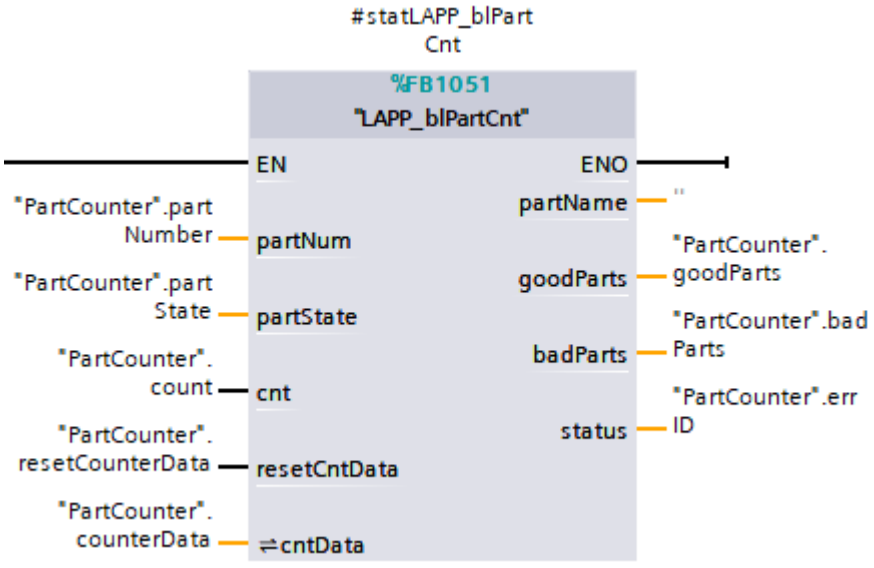


Figure 4-9 LAPP_blPartCnt block

4.2.2 LAPP_blCycleTime

4.2.2.1 Overview

This block calculates the cycle time of a processing cycle for a maximum of 10 stations. The cycle is specified via parameter "startTimeMeas" and "stopTimeMeas" or via parameter "startStopTimeMeas". For internal processing, the data block requires data type "LAPP_typeBlCycleTime".

4.2.2.2 Data type

This data type is used for internal processing of the cycle time of a station.

4.2.2.3 Block interface

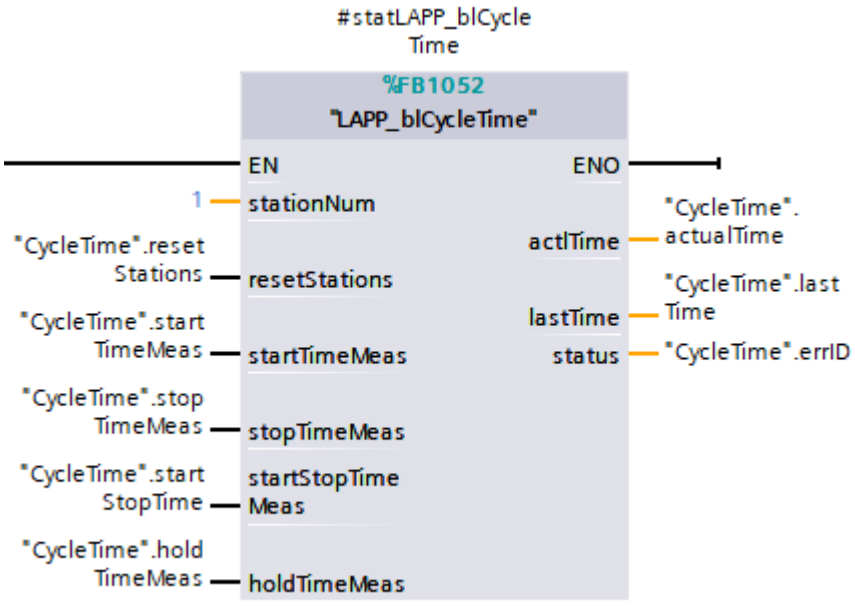


Figure 4-10 "LAPP_blCycleTime" block

Parameters

Name	Declaration	Type	Description
stationNum	Input	Int	Number of the working station for which the cycle time measurement applies to (range: 1...10)
resetStations	Input	Bool	Reset of internal data of all stations
startTimeMeas	Input	Bool	Start of time measurement via start. The time measurement is started via "startTimeMeas" with positive edge
stopTimeMeas	Input	Bool	Stop of time measurement The time measurement is stopped via "stopTimeMeas" with negative edge
startStopTimeMeas	Input	Bool	Starting/stopping time measurement via positive/ negative impulse. The time measurement is started with positive edge and stopped with negative edge.
holdTimeMeas	Input	Bool	Interrupt time measurement via hold The time measurement is interrupted while the signal is TRUE.

Name	Declaration	Type	Description
actlTime	Output	Time	Actual cycle time of the current measurement
lastTime	Output	Time	Time of the last production cycle Last measured cycle time of the station
status	Output	Word	Displays error codes of the block. For more information, refer to Error codes (Page 204).

4.2.2.4 Error codes

List

The error codes for "LAPP_blCycleTime" are:

Error code	Description
16#7002	No error
16#8201	Parameter "stationNumber" incorrect
16#8202	Measurement already running via "startTimeMeas" or "startStopTimeMeas"

4.2.2.5 Cycle time chart

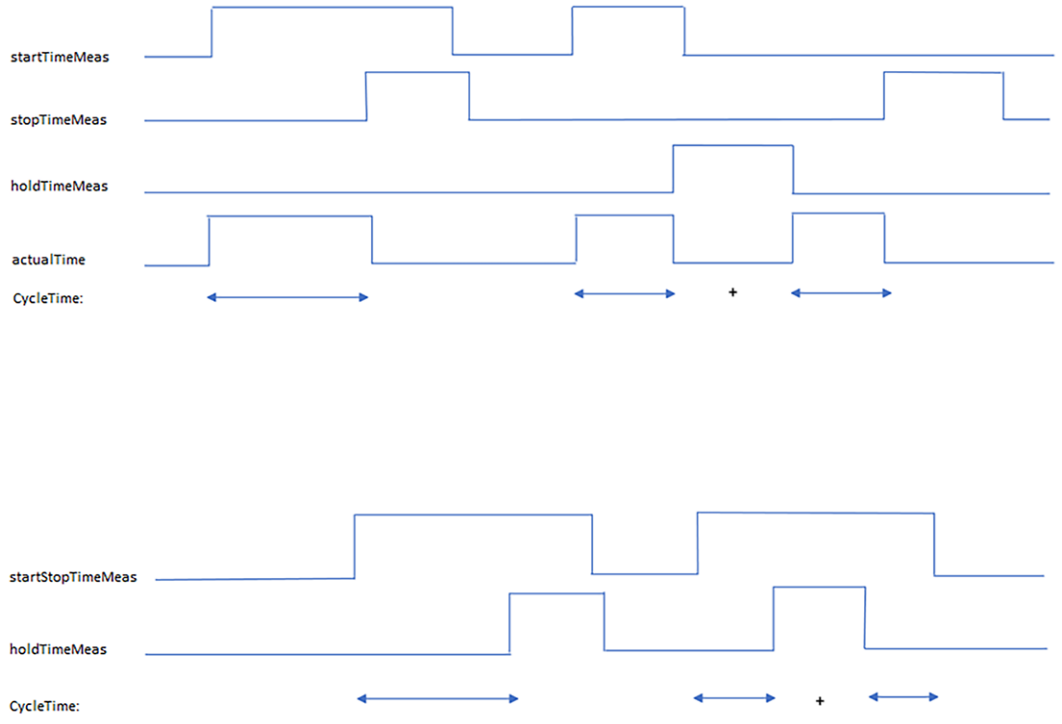


Figure 4-11 Cycle time chart

4.3 Modes of operation

4.3.1 Overview

A function mode can be modified using the operator controls once the preconditions of the specific transline function mode to be selected are fulfilled. The different transline modes are:

- Single setup mode (SET)
- Single step mode (SSM)
- Single mode (SM)
- Automatic mode (AUT)

It must be ensured that there is a safe transition from the active to the selected function mode. For more information, refer to the document TRANSLINE Guide Industry-Specific Specifications (SINUMERIK ONE, SINUMERIK 840D sl and SIMATIC S7-1500).

The blocks under machine operating modes aid the OEMs while using safe operating modes. The operating mode blocks are included in the SINUMERIK ONE Create MyHMI /Automotive library under "BusinessLogic Blocks > ModeOfOperation". The OEMs can use the following blocks for safe operating modes:

- LAPP_blModularMCP (Page 206)
- LAPP_blFooter (Page 211)
- LAPP_blOperatingMode (Page 213)
- LAPP_blFctMode (Page 216)

The data blocks provided along with the above blocks are listed below:

- LAPP_blDevIntfData
- LAPP_blFooterIntfData
- LAPP_blMcpClrDefData
- LAPP_blMcpClParamData
- LAPP_blMcpData
- LAPP_blOperatingModeData
- LAPP_blTlIntfData
- LAPP_blNCintfData
- LAPP_blNCKspecData

Each control panel (SINUMERIK ONE MCP) has a function block and parameterization data block that share a common interface which is "LAPP_blDevInftData".

The operating mode blocks have a similar interface for each device for the below mentioned keys and LEDs:

- Transline, machine function keys and LEDs.
- Axis selection keys and LEDs.
- Keys and LEDs that OEMs can use.

To use the freely definable OEM functions, the OEM must evaluate the signals from the input of the data block "LAPP_blDevIntfData" and write the feedback to the output of the same data block.

Following blocks are provided with Mode of Operation as block call templates:

Note

- These blocks are included in SINUMERIK ONE Create MyHMI /Automotive library under "UserTemplate blocks".
- These blocks are included only for SINUMERIK ONE library.

- LAUP_blAutomotiveToNC
- LAUP_blAutomotiveToNCchan
- LAUP_blAutomotiveToNCaxis
- LAUP_blNCtoAutomotive
- LAUP_blNCaxisToAutomotive
- LAUP_blNCchanToAutomotive

4.3.2 LAPP_blModularMCP

4.3.2.1 Overview

The block is required when you use a MCPxx00 as machine control panel.

MCPxx00 stands for the customer-specific machine control panels SINUMERIK ONE MCP.

- MCP1500
- MCP1900
- MCP2200
- MCP2400

4.3.2.2 Data type

LAPP_blMcpData

The keys, LEDs and powerrides are assigned using "LAPP_blMcpData" data block.

This data block is derived from "LAPP_typeBlParamCtrlMcpxx00" which already contains the configuration for the standard assignment for an MCPxx00.

Note

xx - 15, 19, 22 or 24.

For example, if the "LAPP_blMcpData" is derived from the data type "LAPP_typeBlParamCtrlMcp2200" then this data type already contains the configuration for the standard assignment for MCP 2200. If you use MCP1500, MCP1900 or MCP2400 instead of MCP2200 then the parameterized data types "LAPP_typeBlParamCtrlMcp1500", "LAPP_typeBlParamCtrlMcp1900" or "LAPP_typeBlParamCtrlMcp2400" are already stored in the library. From these you can then create the "LAPP_blMcpData" and transfer its parameters to the respective parameters of "LAPP_blModularMcp".

Additional data types

The following additional data types are required to use MCPxx00. Ensure to copy the below mentioned data types into the "PLC data types" directory of your project.

- In “Libraries” task card, navigate to “Global Libraries > CMH PLC Automotive Program Vxx.xx.xx.xx > Master copies > CMHA PLC Program > BusinessLogic Types > ModeOfOperation" and copy the "ModularMcp" data types.
- In “Libraries” task card, navigate to “Global Libraries > SINUMERIK ONE PLC Basic Program Vxx.xx.xx.xx > Master copies > SINUMERIK ONE PLC Basic Program > SINUMERIK ONE PLC BP data types" and copy the "Modular MCP data types".

Note

Ensure to select the latest version of SINUMERIK ONE PLC Basic Program.

4.3.2.3 Block interface

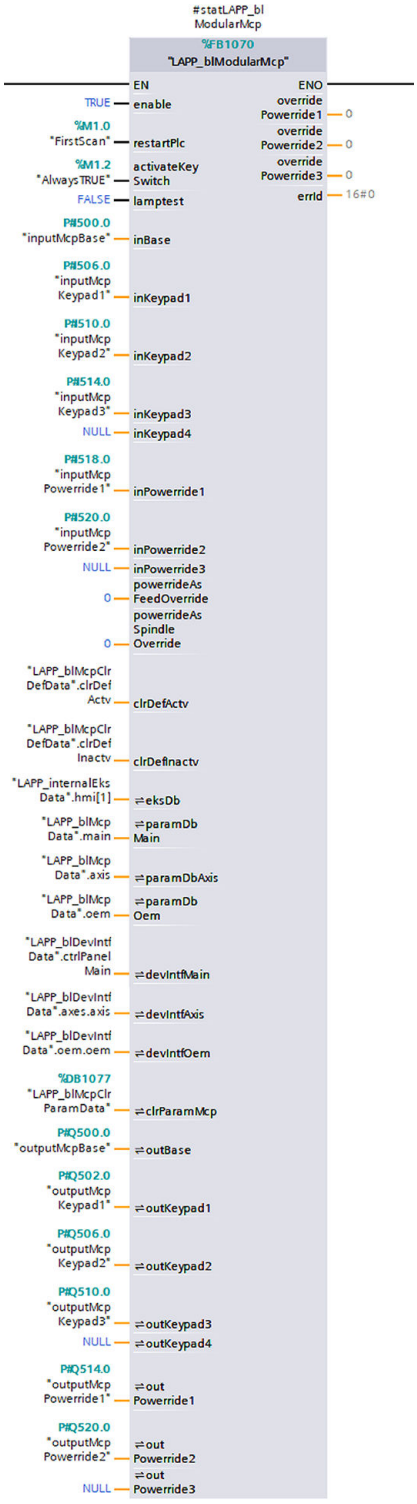


Figure 4-12 "LAPP_blModularMcp"

Parameters

Name	Declaration	Type	Description
enable	Input	Bool	Global release for reading in the function and key signals of the MCPxx00. • TRUE = Active control panel • FALSE = Inactive control panel
restartPlc	Input	Bool	First block cycle following a CPU restart. According to the TRANSLINE standard, the FirstScan [M1.0] memory bit is specified here.
activateKeySwitch	Input	Bool	Transfers the position of the key-operated switch of the basic module to "LAPP_blDevIntfData" when set to true.
lamptest	Input	Bool	• TRUE: Activates all the LEDs on the MCPxx00 • FALSE: Deactivates all the LEDs on the MCPxx00
inBase	Input	LBP_typeModularMcpBaseInput	Input data of the base module from SINUMERIK ONE MCP
inKeypad1	Input	LBP_typeModularMcpUniversalInput	Input data of the key module 1 from SINUMERIK ONE MCP
inKeypad2	Input	LBP_typeModularMcpUniversalInput	Input data of the key module 2 from SINUMERIK ONE MCP
inKeypad3	Input	LBP_typeModularMcpUniversalInput	Input data of the key module 3 from SINUMERIK ONE MCP
inKeypad4	Input	LBP_typeModularMcpUniversalInput	Input data of the key module 4 from SINUMERIK ONE MCP
inPowerride1	Input	LBP_typePowerrideInput	Input data of the Powerride 1 from SINUMERIK ONE MCP
inPowerride2	Input	LBP_typePowerrideInput	Input data of the Powerride 2 from SINUMERIK ONE MCP
inPowerride3	Input	LBP_typePowerrideInput	Input data of the Powerride 3 from SINUMERIK ONE MCP
powerrideAsFeedOverride	Input	Byte	0 = feed override of the basic module is transferred number of the Powerrides (1 to 3): the feed override is generated via the specified Powerride
powerrideAsSpindleOverride	Input	Byte	0 = feed override of the basic module is transferred number of the Powerrides (1 to 3): the feed override is generated via the specified Powerride
clrDefActv	Input	Variant	Color definitions for the display of the LED in active state.
clrDefInactv	Input	Variant	Color definitions for the display of the LED in inactive state.
eksDb	InOut	LAPP_typeHmiEksData	Data block for displaying the key data.
paramDbMain	InOut	LAPP_typeBlParamCtrlMcp	Data block for parameterization of the transline and machine function keys.
paramDbAxis	InOut	Array of LAPP_typeBlParamKeyCtrlMcp	Data block for parameterization of the axis function keys
paramDbOem	InOut	Array of LAPP_typeBlParamKeyCtrlMcp	Data block for parameterization of the OEM function keys
devIntfMain	InOut	LAPP_typeBlDevIntfMain	The main signal interface between the devices and the blocks for acquiring the operating modes, transline and machine functions.

Name	Declaration	Type	Description
devIntfAxis	InOut	Array of LAPP_typeBlDevIntfSingle	The axis signal interface between the devices and the blocks for acquiring the operating modes, transline and machine functions.
devIntfOem	InOut	Array of LAPP_typeBlDevIntfSingle	The OEM signal interface between the devices and the blocks for acquiring the operating modes, transline and machine functions.
clrParamMcp	InOut	LAPP_typeBlClrParamMcp	Transfers the color parameter to the MCPxx00
outBase	InOut	LBP_typeModularMcpBaseOutput	Output data of the base module from SINUMERIK ONE MCP
outKeypad1	InOut	LBP_typeModularMcpUniversalOutput	Output data of the key module 1 from SINUMERIK ONE MCP
outKeypad2	InOut	LBP_typeModularMcpUniversalOutput	Output data of the key module 2 from SINUMERIK ONE MCP
outKeypad3	InOut	LBP_typeModularMcpUniversalOutput	Output data of the key module 3 from SINUMERIK ONE MCP
outKeypad4	InOut	LBP_typeModularMcpUniversalOutput	Output data of the key module 4 from SINUMERIK ONE MCP
outPowerride1	InOut	LBP_typePowerrideOutput	Output data of the Powerride 1 from SINUMERIK ONE MCP
outPowerride2	InOut	LBP_typePowerrideOutput	Output data of the Powerride 2 from SINUMERIK ONE MCP
outPowerride3	InOut	LBP_typePowerrideOutput	Output data of the Powerride 3 from SINUMERIK ONE MCP
overrridePowerride1	Output	Int	Calculated override value of the Powerride 1.
overrridePowerride2	Output	Int	Calculated override value of the Powerride 2.
overrridePowerride3	Output	Int	Calculated override value of the Powerride 3.
errID	Output	Word	Displays the error codes of the block. For more information, refer to Error codes (Page 210).

4.3.2.4 Error codes

List

The error codes for "LAPP_blModularMcp" are:

Error code	Description
16#7002	No error.
16#8201	Key definition for base is wrong.
16#8202	Key definition for keypad is wrong.
16#8203	Key definition for powerride is wrong.
16#8204	Parameter for powerride is wrong.
16#8205	Output key definition color active is wrong.
16#8206	Output key definition color inactive is wrong.
16#8220	The data type for base module parameter is not "LBP_typeModularMcpBaseInput".

Error code	Description
16#8221	The data type for keypad 1 parameter is not "LBP_typeModularMcpUniversalInput".
16#8222	The data type for keypad 2 parameter is not "LBP_typeModularMcpUniversalInput".
16#8223	The data type for keypad 3 parameter is not "LBP_typeModularMcpUniversalInput".
16#8224	The data type for keypad 4 parameter is not "LBP_typeModularMcpUniversalInput".
16#8225	The data type for powerride 1 is not "LBP_typePowerrideInput".
16#8226	The data type for powerride 2 is not "LBP_typePowerrideInput".
16#8227	The data type for powerride 3 is not "LBP_typePowerrideInput".
16#8228	The data type for handwheel 1 is not "SINU_typeHandwheelData".
16#8229	The data type for handwheel 2 is not "SINU_typeHandwheelData".
16#8230	The data type for base module parameter is not "LBP_typeModularMcpBaseOutput".
16#8231	The data type for keypad 1 parameter is not "LBP_typeModularMcpUniversalOutput".
16#8232	The data type for keypad 2 parameter is not "LBP_typeModularMcpUniversalOutput".
16#8233	The data type for keypad 3 parameter is not "LBP_typeModularMcpUniversalOutput".
16#8234	The data type for keypad 4 parameter is not "LBP_typeModularMcpUniversalOutput".
16#8235	Parameter for Powerride 1 not of data type "LBP_typePowerrideOutput".
16#8236	Parameter for Powerride 2 not of data type "LBP_typePowerrideOutput".
16#8237	Parameter for Powerride 3 not of data type "LBP_typePowerrideOutput".
16#8238	Parameters at input and output for base module not plausible.
16#8239	Parameters at input and output for keypad 1 not plausible.
16#8240	Parameters at input and output for keypad 2 not plausible.
16#8241	Parameters at input and output for keypad 3 not plausible.
16#8242	Parameters at input and output for keypad 4 not plausible.
16#8243	Parameters at input and output for Powerride 1 not plausible.
16#8244	Parameters at input and output for Powerride 2 not plausible.
16#8245	Parameters at input and output for Powerride 3 not plausible.
16#8246	Array sizes of "paramDbAxis" and "devIntfAxis" are different.
16#8247	Array sizes of "paramDbOem" and "devIntfOem" are different.
16#8248	Parameter for "clrDefActiv" is wrong.
16#8249	Parameter for "clrDefInactiv" is wrong.

4.3.3 LAPP_blFooter

4.3.3.1 Overview

The block is required when you use the SVG footer elements. The footer widgets have to be used in a group with a button, to be able to use the press and release event of the button.

For more information on the SVG footer elements refer to, Footer elements.

4.3.3.2 Data type

The "LAPP_blFooterIntfData" block is required to provide the footer signals for the main, axis and OEM functions to the "LAPP_blFooter".

The "LAPP_blDevIntfData" block is required to provide the device interface data for the main, axis and OEM functions to the "LAPP_blFooter".

Additional data types

The following additional data types are required to use MCPxx00. Ensure to copy the below mentioned data types into the "PLC data types" directory of your project.

In “Libraries” task card, navigate to “Global Libraries > CMH Automotive Program Vxx.xx.xx.xx > Master copies > CMH PLC Program > BusinessLogic Types > ModeOfOperation" and copy the "Footer" data types.

4.3.3.3 Block interface

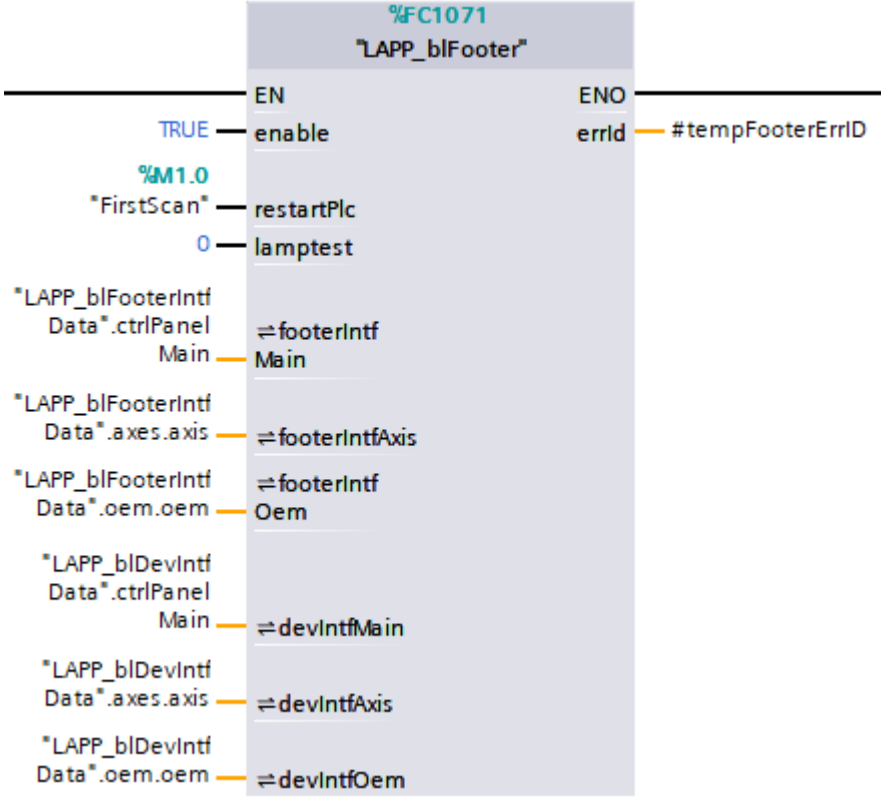


Figure 4-13 "LAPP_blFooter"

Parameters

Name	Declaration	Type	Description
enable	Input	Bool	Enables the evaluation of the footer signals
restartPlc	Input	Bool	First block cycle following a CPU restart. According to the TRANSLINE standard, the FirstScan [M1.0] memory bit is specified here.
lamptest	Input	Bool	Sets all output signals of the footer to true.
footerIntfMain	InOut	LAPP_typeBlFooterIntfMain	Interface for the footer signals for the main functions.
footerIntfAxis	InOut	Array of LAPP_typeBlFooter-IntfSingle	Interface for the footer signals for the axis functions.
footerIntfOem	InOut	Array of LAPP_typeBlFooter-IntfSingle	Interface for the footer signals for the OEM functions
devIntfMain	InOut	Array of LAPP_typeBlDevIntf-Main	Device interface of the main functions.
devIntfAxis	InOut	Array of LAPP_typeBlDevIntf-Single	Device interface of the axis functions.
devIntfOem	InOut	Array of LAPP_typeBlDe-vIntfSingle	Device interface of the OEM functions.
errId	Output	Word	Displays error codes of the block. For more information, refer to Error codes (Page 213).

4.3.3.4 Error codes

List

The error codes for "LAPP_blFooter" are:

Error code	Description
16#7002	No error.
16#8201	Array sizes on "parDbAxis" and "deviceInterfaceAxis" different.
16#8202	Array sizes on "parDbOem" and "deviceInterfaceOem" are different.

4.3.4 LAPP_blOperatingMode

4.3.4.1 Overview

The "LAPP_blOperatingMode" block is called cyclically between the device blocks "LAPP_blModularMcp" or "LAPP_blFooter" and the transline function block "LAPP_blFctModeTl".

Based on the selected mode of safe operation, "LAPP_blOperatingMode" preselects a transline function mode. It always attempts to retain the existing function. The block influences only the transline function mode selection. You must program the activation of transline function and the feedback signal.

4.3.4.2 Data types

LAPP_blOperatingModeData

You can use this data block to lock the key functions at the device interface "LAPP_blDevIntfData". This data block defines the permitted key functions by considering the safe operating modes and the transline functions.

4.3.4.3 Block interface



Figure 4-14 "LAPP_blOperatingMode"

Parameters

Name	Declaration	Type	Description
autoMode	Input	Bool	Set it to TRUE for mode of operation 1 (MO1) - standard mode (traversing with closed protection area).
setupMode	Input	Bool	Set it to TRUE for mode of operation 2 (MO2) - setup mode (traversing with open protection area).
processMon	Input	Bool	Set it to TRUE for mode of operation 3 (MO3) - process monitoring mode (traversing with open protection area under restricted operating conditions).
mtOperatingmode	Input	Bool	Set it to TRUE for maintenance mode of operation (MO MT) - safety related operating mode to conduct machine servicing (traversing with open protection area).
disableFctmodeSwitch	Input	Bool	Set it to TRUE to prevent the automatic selection of the TRANSLINE function mode when the operating mode is changed.
keyLocked	InOut	Bool	Sets the signal if the operator presses a locked key.
tlIntf	InOut	LAPP_typeBlTlIntf	Transline signals to or from the user.
devIntfMain	InOut	LAPP_typeBlDevIntf-Main	The main signal interface between the devices and the blocks for acquiring the operating modes, transline and machine functions.
devIntfAxis	InOut	Array of LAPP_typeBlDevIntfSingle	The axis signal interface between the devices and the blocks for acquiring the operating modes, transline and machine functions.
devIntfOem	InOut	Array of LAPP_typeBlDevIntfSingle	The OEM signal interface between the devices and the blocks for acquiring the operating modes, transline and machine functions.
relFctsMain	InOut	LAPP_typeBlRelFcts-Main	Supplies the interlock interface of the main signals.
relFctsAxes	InOut	Array of LAPP_type-BlRelFctsSingle	Supplies the interlock interface of the axes signals.
relFctsOem	InOut	Array of LAPP_type-BlRelFctsSingle	Supplies the interlock interface of the OEM signals.
errID	Output	Word	Displays the error codes of the block. For more information, refer to Error codes (Page 215).

4.3.4.4 Error codes

List

The error codes for "LAPP_blOperatingMode" are:

Error code	Description
16#7002	No error.
16#8201	Operating mode error.
16#8202	Transline function error.

Error code	Description
16#8203	Array sizes of parameters "devIntfAxes" and "relFctsAxes" do not match.
16#8204	Array sizes of parameters "devIntfOem" and "relFctsOem" do not match.

4.3.5 LAPP_blFctMode

4.3.5.1 LAPP_blFctModeTl

Overview

"LAPP_blFctModeTl" must be called cyclically after "LAPP_blOperatingMode". Based on the transline specifications "LAPP_blFctModeTl" will read and write the signals.

The block reads the input signals from "LAPP_blDevIntfData" and provides them as input to "LAPP_blTlIntfData". Further, it reads the signals from "LAPP_blTlIntfData" and writes them to the output signals of "LAPP_blDevIntfData".

Furthermore, the block supplies the states of the widgets in the header of Configuring Create My HMI /Automotive.

Block interface

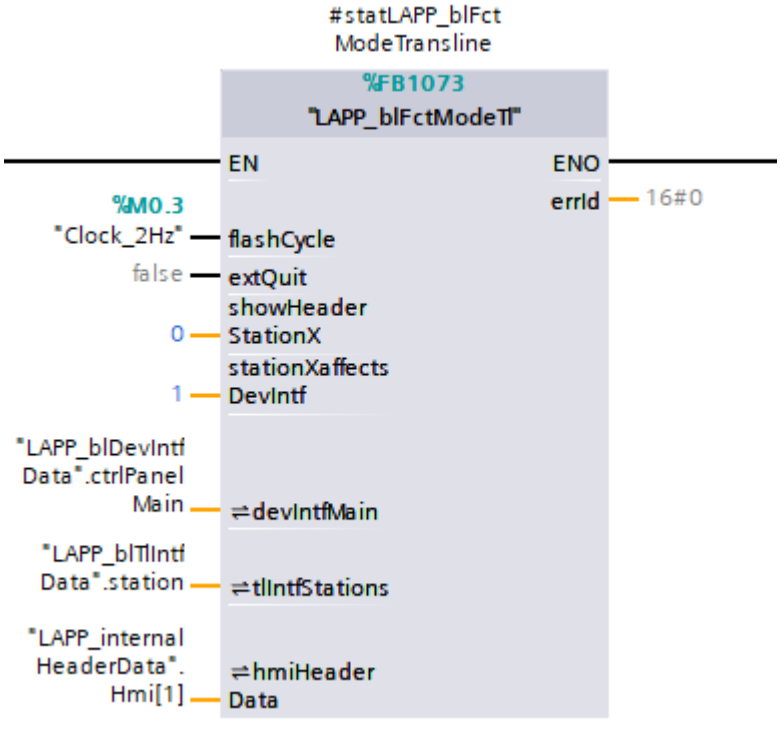


Figure 4-15 "LAPP_blFctModeTl"

Parameters

Name	Declaration	Type	Description
flashCycle	Input	Bool	Input for flashing bit memory cycle
extQuit	Input	Bool	The logic OR function is executed for the acknowledge key of the MCP or Footer button when acknowledging via MCP/Footer.
showHeaderStationX	Input	USInt	Displays the header for station X: • 0 = Active station • 1 = Main station • 2 = Do not write header
stationXaffectsDevIntf	Input	USInt	Displays the station that affects the device interface: • 0 = Active station • 1 = Main station • 2 = Do not write header
devIntfMain	InOut	LAPP_typeBlDevIntfMain	The main signal interface between the devices and the blocks for acquiring the operating modes, transline and machine functions.
tlIntfStations	InOut	Array of LAPP_typeBlTlIntf	Transline signals to or from the user
hmiHeaderData	InOut	LAPP_typeHmiHeaderData	Connection to the header of the corresponding HMI. If required, displays the actively selected station number and the status of "translineMode", "basicPos" etc., which are linked in the header.
errID	Output	Word	Displays the error codes of the block. For more information, refer to Error codes (Page 217).

Error codes

List

The error codes for "LAPP_blFctModeTl" are:

Error code	Description
16#7002	No error.
16#8201	Station for header binding and array boundaries of "tlIntfStations" does not match.
16#8202	Station for connection to device interface and array limits of "tlIntfStations" does not match.

4.3.5.2 LAPP_blFctModeNC

Overview

"LAPP_blFctModeNC" is only required for NC controllers and not for SIMATIC S7-1500 PLC controllers. It must be called in the cyclic sequence after LAPP_blOperatingMode and LAPP_blFctModeTl.

"LAPP_blFctModeNC" is used to convert TRANSLINE functions to the corresponding NC functions. The implementation is performed indirectly and does not directly set the interface signals of the NC. The corresponding signals are included in the parameter "nckSpecData".

Before calling "LAPP_blFctModeNC" its inputs must be supplied with the interface signals of the NC and the outputs of BAG, channel and axis data must be transferred to their corresponding NC data.

After the call the output signals must be passed on to the corresponding interface signals and the inputs of BAG, channel and axis data must be supplied with the corresponding NC signals.

Block call templates

LAPP_blNCtoAutomotive, LAPP_blNCaxisToAutomotive, LAPP_blNCchanToAutomotive

- These blocks are only available in the SINUMERIK ONE library of CMH /Automotive and can only be used with an NC controller in conjunction with "LAPP_blFctModeNC".
- The function blocks serve as templates to copy and expand. In this case the local parameters need to be replaced by the project-specific parameters and adapt the blocks to the project-specific number of mode groups (BAGs), axes and channels.
- "LAPP_blNCtoAutomotive" needs to be called in the cyclic sequence before "LAPP_blFctModeNC".
- The blocks are used to supply TRANSLINE with NC interface signals. "LAPP_blNCtoAutomotive" transfers the interface signals to TRANSLINE. Then it calls "LAPP_blNCaxisToAutomotive" and "LAPP_blNCchanToAutomotive" to supply the axis and channel interfaces.

LAPP_blAutomotiveToNC, LAPP_blAutomotiveToNCaxis, LAPP_blAutomotiveToNCchan

- These blocks are only available in the SINUMERIK ONE library of CMH /Automotive and can only be used with an NC controller in conjunction with "LAPP_blFctModeNC".
- The function blocks serve as templates to copy and expand. In this case the local parameters need to be replaced by the project-specific parameters and adapt the blocks to the project-specific number of mode groups (BAGs), axes and channels.
- "LAPP_blAutomotiveToNC" needs to be called in the cyclic sequence after "LAPP_blFctModeNC".
- The blocks are used to supply the NC interface with TRANSLINE data. "LAPP_blAutomotiveToNC" transfers the TRANSLINE data to the NC interfaces. Then it calls "LAPP_blAutomotiveToNCaxis" and "LAPP_blAutomotiveToNCchan" to supply the axis and channel interfaces.

Block interface



Figure 4-16 "LAPP_blFctModeNC"

Parameters

Name	Declara-tion	Type	Description
restartPLC	Input	Bool	Specifies that the PLC restarted, internal data is initial-ized.
mcsWsc	Input	Bool	If the parameter is set to • True = Displays in WCS • False = Displays in MCS

Name	Declaration	Type	Description
translineMode	Input	Bool	If the parameter is set to • True = The NC operating modes are influenced by the TRANSLINE function modes – AUT (automatic mode) = NC automatic mode – SM (single mode) = NC automatic mode – SSM (single step) = NC automatic mode, optional single block – SET (setup mode) = JOG, with switchover to setup - switch is made to NC JOG • False = MCP functions are always enabled, i.e. the user must establish the relationship.
disableNCsingleStep	Input	Bool	If the parameter is set to true then the TRANSLINE function mode SSM is not automatically accepted in the NC function single block while switching over the operating mode.
flashCycle	Input	Bool	Input for flashing bit memory cycle. According to the TRANSLINE standard, here, the memory bit clock_2Hz (M0.3) is specified.
axisSelectForWcs	Input	Bool	If the parameter is set to • True = Axis selection for GEO axes in the WCS representation is possible at MCP or HT10 • False = Axis selection is not possible in the WCS representation at MCP or HT10
bagNo	Input	Byte	Operating mode group in which the operating mode signals are transferred.
chanNo	Input	Byte	Channel number, for which the channel signals are determined. This must match the channel displayed at the control.
feedHold	InOut	Bool	The feed is stopped for a switchover and the previous state is maintained when the parameter "feedHold" is enabled.
spindleHold	InOut	Bool	The spindle is stopped for a switchover and the previous state is maintained when the parameter "spindleHold" is enabled.
nckSpecData	InOut	LAPP_typeBlNCKspecData	Contains all of the necessary NCK-specific data, which must be provided by the user with the corresponding interface signals.
devIntfAxes	InOut	LAPP_typeBlDevIntfAxes	Interface which holds the data of the axes.
devIntfMain	InOut	LAPP_typeBlDevIntfMain	Interface between the HMI devices and the blocks for acquiring the operating modes, TRANSLINE and NC functions.
ncIntfData	InOut	LAPP_typeBlNCintfData	Interface between CMH /Automotive and the PLC user program.
errId	Output	Word	Displays the error codes of the block. For more information, refer to Error codes (Page 221).

Error codes

List

The error codes for "LAPP_blFctModeNC" are:

Error code	Description
16#7002	No error.
16#8201	Channel number does not match the actual channel for MCS.
16#8202	Channel number = 0 for WCS.

4.3.6 Protection against manipulation

4.3.6.1 Overview

The purpose of the manipulation protection is to minimize the incentive to manipulate. The "ManipulationProtection" block package functions only in conjunction with the block "LAPP_EKS". It monitors the complete data flow, from reading the data set in the PLC up to inserting the EKS key. The function of the block package does not initiate a direct operating mode (MO) or machine movement.

The block package comprises of General TIA blocks (OB1 cycle).

Note

"ManipulationProtection" is not in conformance with the TIA style guide because of the special data access involved for manipulation protection.
The principle of operation of the blocks is not explained any further as these blocks are intended to provide manipulation protection. All blocks are encrypted. It is not permissible to change the block names; the block numbers may be changed.

4.3.6.2 Block interface - LAPP_blMoCtrl

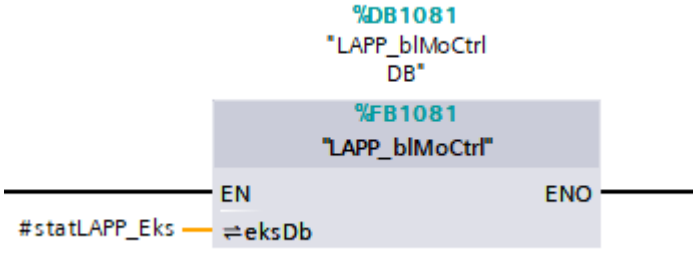


Figure 4-17 "LAPP_blMoCtrl"

Parameters

Name	Declaration	Type	Description
eksDb	InOut	LAPP_EKS	Instance data of the EKS block.

Abbreviations

5

A	
ACK	Acknowledgement
ACTV	Active
ACTVCHN	Active Channel
ACT	Actual
ADPTR	Adapter
ASYM	Asymmetry
ATTR	Attribute
ATTRIB	Attributes
AUTHZ	Authorization
AUTO	Automatic
AVG	Average

B	
BCD	Binary Coded Decimals
BI	Bidirectional
BL	Business logic

C	
CALC	Calculate
CF/CONFIG	Configuration
CLK	Clock
CLR	Color
CMD	Command
CMH	Create MyHMI
CMHA	Create MyHMI /Automotive
CMPLT	Complete
CNFRM	Confirm
CNT	count
COMM	Communication
CORR	Corrected
COST CTR	Cost Center
CT	Cycletime
CTR	Center
CTRL	Control
CURR	Current (electrical)

D	
D4	Display 4
DB	Data Block
DEF	Definition
DESCR	Description
DIAG	Diagnostics
DIFF	Difference
DISBL	Disable
DISP	Display
DM	Data Matrix
DO	Drive Object
DTL/DT	Date and Time

E	
EE	Energy Efficiency
EKS	Electronic Key Switch
EM	Emergency
EMSTOP	Extended Stop
ENBL	Enable
ENC	Encoder
EPOS	Basic positioner
EQUIP	Equipment
ERR	Error
ERRID	ErrorID
ET	Elapsed time
EXEC	Execute
EXP	Expiry
EXTN	Extension
EXTND	Extended

F	
FCT	Function
FCTS	Functions
FP	Flag pulse
FREQ	Frequency

G	
GRP	Group

H	
HDR	Header
HMI	Human Machine Interface
HR	Hour
HW	Hardware

I	
ID	Identity
IM	Identification and Maintenance
IMM	Immediate
IN	Input
INC	Increment
IND	Index
INFO	Information
INST	Instance
INTF	Interface
IP	Internet Protocol
IPC	Industrial PC

L	
LAHP	Library automotive HMI plus
LAPP	Library automotive PLC program plus
LAUP	Library automotive user program plus
LWR	Lower

M	
MACH	Machine
MAN	Manual
MAX	Maximum
MEAS	Measurement
MECH	Mechanical
MIN	Minimum
MLFB	Machine Readable Product Identifier
MO	Mode of Operation
MON	Monitoring

N	
NAV	Navigation
NCU	Numerical Control Unit

N	
NOK	Not Ok
NUM	Number

O	
OBJ	Object
OEM	Original Equipment Manufacturer
OPHRS	Operating Hours
OPTR	Operator
OSD	Online Software Delivery
OUT	Output
OVLD	Overload

P	
PARAM	Parameterization
PaCo	Parameter Control
PCT	Percent/Percentage
POS	Position
PRV/PREV	Previous
PRIO	Priority
PN	PROFINET
PLC	Programmable Logic Controller
PROT	Protection

R	
RFID	Radio Frequency Identification
RDREC	ReadRecord
REC	Record
REG	Registry
REL	Release
REQ	Request
RST	Reset
RT	Runtime

S	
SEL	Selection/select
SENS	Sensor
SIGN	Signature
SW	Software
STD	Standard
STATS	Statistics

S	
SPC	Statistical process control
SCHELE	Switching element
SVG	Scalable vector graphics
SSM	Single step mode
SYS	System

T	
TO	Technology Object
TOT	Total
TL	Transline
TRG	Trigger

U	
UCP	Unified Comfort Panel
URL	Uniform Resource Locator
UPR	Upper
USR	User

V	
VAR	Variable
VIB	Vibration

W	
WARN	Warning
WPIECE	Work Piece
WRREC	WriteRecord

Appendix A

A

A.1 Rulesets

This topic is about the deviations from programming guidelines. Following are the list of deviated rules:

Settings in TIA portal

Rule No.	Standard Rulesets	Deviated Rulesets
8	R: tab characters are not permitted in the source text. Indentations must be realized with two space characters.	Indentations are realized with four space characters.

Identifier

Rule No.	Standard Rulesets	Deviated Rulesets
14	R: only meaningful identifiers are used.	Identifiers used are based on recommendation, as specific review is difficult.

PLC programming

Rule No.	Standard Rulesets	Deviated Rulesets
26	R: within a block, only local tags are used.	In exceptional cases, for security reasons local tags are not used everywhere.
34	R: data exchange between blocks is performed exclusively via the block interfaces.	In exceptional cases, for security reasons data exchange between blocks is not via the block interfaces.

For more information on programming guidelines, refer to Programming Guidelines and Programming Styleguide for SIMATIC S7-1200 and S7-1500 and WinCC (TIA Portal) (https://support.industry.siemens.com/cs/document/81318674/programming-guidelines-and-programming-styleguide-for-simatic-s7-1200-and-s7-1500-and-wincc-(tia-portal)?dti=0&lc=en-WW).

Index

B

Block interface
    Data handling, Configurations, 53
    LAPP_ToolLifeMonitor, 91
Blocks
    Hardware Diagnostics, (LAPP_Config)
    LAPP_ShiftPlanning, (Shift Planner)
    Manual Operation, (LAPP_ManualOperation)
    Shift Planner, (Shift Planning)
    Tool life monitor, (LAPP_toolLifeMonitor), (PLC)
    Workpiece Manager, 51

C

Configurations, 104
Configure, (LAPP_Init)
Configure"; "LAPP_ShiftPlanning, 45

D

Data handling, 104

E

EKS (Electronic Key System, 157
Electronic Key Manager, 158

L

LAHP_IdentDeviceName, 135
LAPP_blCycleTime, 202
    Data type, 203
LAPP_blCycleTime block, 203
LAPP_blGraphManExtn, 194
LAPP_blGraphManOutParam, 198
LAPP_blGraphManualOutputParam, 199
LAPP_blPartCnt, 199
LAPP_blPartCnt block, 201
LAPP_blPartCounter
    LAPP_typeBlPartState, 199
LAPP_camera, 103
LAPP_EKS, 151, 155
    Electronic Key System, 148
LAPP_ManualOperationTouch
    Manual operation, Touch operated, 67
LAPP_Motorstarter, 129
LAPP_PLCSystemData, 133
    First scan, 134
    PLC system, 132
    Status codes, 135
LAPP_Powerride, 160, 161, 171
    Example Powerride via MCP, 168
    Example: Calling LAPP_Powerrride, 170
    LED color palette, 168
LAPP_RFID, 135
    Status codes, 138
LAPP_Safety, 139
LAPP_ShiftPlanning block, 202
LAPP_SINAMICSEpos, 143
LAPP_SINAMICSTO, 145
LAPP_typeBlPartCnt, 200
LAPP_typeToolLifeMonitor, 93
LAPP_typeWidgetPartDetails, 101
LAPP_typeWidgetProductionStatus, 102
LAPP_typeWidgetQualityMetrics, 102
LAPP_typeWidgetWpiece, 101
LAPP_UpdateEthernetMcp, 173
LAPP_WriteRecordOverEthernet, 182

M

Manual Operation
    As a screen, 66
MyPowerrideConfig, 164

S

Status
    LAPP_MotorStarter, 131
    LAPP_Safety, 141
    LAPP_SINAMICS, 144
    LAPP_SINAMICSTO, 147
    LAPP_ToolLifeMonitor, 95
    Manual Operation, 80
    Screen switching, 42
    Shift Planner, 47
    Workpiece manager, 51
    Workpiece Manager, 56

W

Workpiece Manager, (CNC), (PLC)
    Configure, (LAPP_WorkpieceManager)