SIEMENS

SIMOTION

SIMOTION SCOUT Communication

System Manual

05/2009

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
with a safety alert symbol, indicates that minor personal injury can result if proper precautions are not taken.

CAUTION
without a safety alert symbol, indicates that property damage can result if proper precautions are not taken.

NOTICE
indicates that an unintended result or situation can occur if the corresponding information is not taken into account.

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 device/system may only be set up and used in conjunction with this documentation. Commissioning and operation of a device/system may only be performed by **qualified personnel**. Within the context of the safety notes in this documentation qualified persons are defined as persons who are authorized to commission, ground and label devices, systems and circuits in accordance with established safety practices and standards.

Proper use of Siemens products

Note the following:

⚠WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. 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 adhered to. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of the Siemens AG. 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 AG
Industry Sector
Postfach 48 48
90026 NÜRNBERG
GERMANY

℗ 06/2009

Foreword

Foreword

Content

This document is part of the **System and Function Descriptions** documentation package.

Scope of validity

This manual is valid for SIMOTION SCOUT product version V4.1 SP4:

- SIMOTION SCOUT V4.1 SP4 (Engineering System for the SIMOTION product range)

Chapters in this manual

This manual describes the communications possibilities for SIMOTION systems.

- **Communications functions and services overview**

  General information about the communications possibilities provided by SIMOTION.
- **PROFIdrive**

  Description of the PROFIdrive profile.
- **PROFIBUS**

  Information about the DPV1 communication, and the setup and programming of the communication between SIMOTION and SIMATIC devices.
- **Ethernet introduction (TCP/IP and UDP connections)**

  Information about the the setup and programming of the Ethernet communication between SIMOTION and SIMATIC devices.
- **PROFINET IO**

  Information about configuring PROFINET with SIMOTION
- **Routing - communication across network boundaries**

  General information about routing
- **SIMOTION IT**

  General information about the IT and Web functions provided by SIMOTION.
- **Index**

  Keyword index for locating information

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in a separate list of references.

This documentation is included as electronic documentation with the supplied SIMOTION SCOUT.

The SIMOTION documentation consists of 9 documentation packages containing approximately 80 SIMOTION documents and documents on related systems (e.g. SINAMICS).

The following documentation packages are available for SIMOTION V4.1 SP4:

- SIMOTION Engineering System
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P350
- SIMOTION D4xx
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

Siemens Internet address

The latest information about SIMOTION products, product support, and FAQs can be found on the Internet at:

- General information:
  - http://www.siemens.de/simotion (German)
  - http://www.siemens.com/simotion (international)
- Downloading documentation
  Further links for downloading files from Service & Support.
  http://support.automation.siemens.com/WW/view/en/10805436
- Individually compiling documentation on the basis of Siemens contents with the My Documentation Manager (MDM), refer to http://www.siemens.com/mdm

  My Documentation Manager provides you with a range of features for creating your own documentation.
- FAQs
  You can find information on FAQs (frequently asked questions) by clicking http://support.automation.siemens.com/WW/view/en/10805436/133000.

Additional support

We also offer introductory courses to help you familiarize yourself with SIMOTION.

For more information, please contact your regional Training Center or the main Training Center in 90027 Nuremberg, Germany.

Information about training courses on offer can be found at:

www.sitrain.com

Technical support

If you have any technical questions, please contact our hotline:

	Europe / Africa
Phone	+49 180 5050 222 (subject to charge)
Fax	+49 180 5050 223
€0.14/min from German wire-line network, mobile phone prices may differ.	
Internet	http://www.siemens.com/automation/support-request

	Americas
Phone	+1 423 262 2522
Fax	+1 423 262 2200
E-mail	mailto:techsupport.sea@siemens.com

	Asia / Pacific
Phone	+86 1064 757575
Fax	+86 1064 747474
E-mail	mailto:support.asia.automation@siemens.com

Note

Country-specific telephone numbers for technical support are provided under the following Internet address:

http://www.automation.siemens.com/partner

Questions about this documentation

If you have any questions (suggestions, corrections) regarding this documentation, please fax or e-mail us at:

Fax	+49 9131- 98 2176
E-mail	mailto:docu.motioncontrol@siemens.com

Table of contents

1 Introduction

1.1 The communications subject in the SIMOTION documentation

Overview

You can find information on the subject of communication in the individual Manuals, in the Programming Manuals and in this Communication Manual.

Communication manual

This communication manual provides, in particular, information that is important for the communication of SIMOTION devices with devices that are not part of the SIMOTION family, especially SIMATIC devices.

This manual contains descriptions of the required configuration steps that must be performed on both communication partners in order to obtain an error-free, functioning communication relationship.

Therefore, this manual deals very intensively with the settings and the programming of the SIMATIC S7 stations as communication partners of the SIMOTION devices.

Product manuals and programming manuals

The product manuals deal with the subject of communication from the point of view of the devices themselves, i.e. with respect to the electrical properties of the available interfaces as well as the setting options with the SIMOTION SCOUT engineering system.

You will also find further information in the manuals entitled **Modular Machine Concepts** and **Base Functions**, which are part of the SIMOTION documentation package.

There is no information here how the partner stations are set.

Overview of the communication functions and services 2

2.1 Network options

2.1.1 Introduction

As an integral part of "Totally Integrated Automation" (TIA), the SIMOTION and SIMATIC network solutions provide the necessary flexibility and performance characteristic for the communication requirements of your application, irrespective of how simple or complex it is.

Note

This section provides a general description of the communication functions and services included in Siemens' automation technology. This does not necessarily imply that all functions mentioned also are available for SIMOTION. You will find details concerning the functions supported by SIMOTION in chapters 4 - 8.

SIMOTION and SIMATIC networks for all applications

The SIMOTION products support a variety of network options. With these network solutions, you can combine the SIMOTION devices in accordance with the requirements of your application.

For further optimization of the network solutions, SIMOTION products provide integrated communication services and functions to extend the performance capability of the network protocol.

2.1.2 PROFINET

Overview

PROFINET is based on the open Industrial Ethernet standard for industrial automation for company-wide communication and extends the capability for data exchange of your automation components through to the office environment, so that your automation components, even the distributed field devices and drives, can be connected to your local area network (LAN).

Because PROFINET connects all levels of your organization – from the field devices through to the management systems – you can perform the plant-wide engineering using normal IT standards. As for all solutions based on Industrial Ethernet, PROFINET supports electrical, optical and wireless networks.

As PROFINET is based on Industrial Ethernet and not implemented as a derived form of "PROFIBUS for Ethernet", PROFINET can utilize the previously installed Ethernet-compatible devices. Even if PROFINET is not a master/slave system, the PROFINET IO and PROFINET CBA communication services provide the functionality required by automation systems:

- With PROFINET IO, you can connect distributed field devices (e.g. digital or analog signal modules) and drives directly to an Industrial Ethernet subnet.
- PROFINET CBA (Component-Based Automation) supports modular solutions for machine and plant construction. You define your automation system as autonomous components, whereby each component consists of independent, self-contained tasks.

Both communication services provide real-time functionality to make PROFINET a real-time implementation. PROFINET also enables the simultaneous existence of the real-time communication of your automation process and your other IT communication, at the same time in the same network, without the real-time behavior of your automation system being impaired.

The PROFIsafe profile communicates with the fail-safe devices via the PROFINET subnet for further support of fail-safe or "safety-relevant" applications.

2.1.3 Industrial Ethernet

Overview

As Industrial Ethernet provides a communication network for the connection of command levels and cell levels, you can extend the data exchange capability of your automation components into the office environment with Industrial Ethernet.

Industrial Ethernet is based on the standards IEEE 802.3 and IEEE 802.3u for communication between computers and automation systems and enables your system to exchange large data volumes over long distances.

2.1.4 PROFIBUS

Overview

PROFIBUS is based on the standards IEC 61158 and EN 50170 and provides a solution with open field bus for the complete production and process automation. PROFIBUS provides fast, reliable data exchange and integrated diagnostic functions. PROFIBUS supports manufacturer-independent solutions with the largest third-party manufacturer support worldwide. A variety of transmission media can be used for your PROFIBUS subnet: electrical, optical and wireless.

PROFIBUS provides the following communication services:

- PROFIBUS DP (Distributed Peripherals) is a communication protocol that is especially suitable for production automation.
  PROFIBUS DP provides a fast, cyclic and deterministic exchange of process data between a bus DP master and the assigned DP slave devices. PROFIBUS DP supports isochronous communication. The synchronized execution cycles ensure that the data is transmitted at consistently equidistant time intervals.
- PROFIBUS PA (Process Automation) expands PROFIBUS DP to provide intrinsically safe data and power transmission according to the IEC 61158-2 standard.
- PROFIBUS FMS (Fieldbus Message Specification) is for communication on the cell level, where the controllers communicate with one another. Automation systems from different manufacturers can communicate with one another by means of PROFIBUS FMS.
- PROFIBUS FDL (Fieldbus Data Link) has been optimized for the transmission of medium-sized data volumes to support error-free data transmission on the PROFIBUS subnet.

In addition, PROFIBUS uses profiles to provide communication options for the needs of specific applications, such as PROFIdrive (for the motion control) or PROFIsafe (for fail-safe or "safety-relevant" applications).

2.1.5 MPI (Multi-Point Interface)

Overview

MPIs are integrated interfaces for SIMOTION and SIMATIC products (SIMOTION devices, SIMATIC S7 devices, SIMATIC HMI as well as SIMATIC PC and PG).

MPI provides an interface for PG/OP communication. In addition, MPI provides simple networking capability using the following services: communication via global data (GD), S7 communication and S7 basic communication.

The electric transmission medium for MPI uses the RS 485 standard, which is also used by PROFIBUS.

2.1.6 Point-to-point communication (PtP)

Overview

SIMOTION devices can be programmed so that they exchange data with another controller in the network. Even if the point-to-point communication is not considered as a subnet, the point-to-point connection provides serial transmission (e.g. RS232 or RS485) of data between two stations, e.g. with a SIMATIC controller or even with a third-party device that is capable of communication.

CP modules (e.g. a CP340) or ET200 modules can be used for point-to-point communication to read and write data between two controllers. Point-to-point communication thus represents a powerful and cost-effective alternative to bus solutions, particularly when only a few devices are connected to the SIMOTION device.

Point-to-point communication provides the following capabilities:

- Using standard procedures or loadable drivers to adapt to the protocol of the communication partner
- Using ASCII characters to define a user-specific procedure
- Communication with other types of devices, such as operator panels, printers or card readers

Additional references

You will find additional references concerning point-to-point communication in the descriptions of the CP or ET200 modules.

2.2 Communications services (or network functions)

2.2.1 Introduction

SIMOTION and SIMATIC devices support a set of specific communication services, which control the data packets that are transmitted via the physical networks. Each communication service defines a set of functions and performance characteristics, e.g. the data to be transferred, the devices to be controlled, the devices to be monitored and the programs to be loaded.

Communication services of the SIMOTION and SIMATIC products

Communication services, also often referred to as network functions, are the software components that utilize the physical hardware of the networks. Software interfaces (e.g. S7 system functions) in the end device (e.g. SIMOTION device, SIMATIC S7 device or PC) provide access to the communication services. However, a software interface does not necessarily have all of the communication functions for the communication service. Such a service can be provided in the respective end system with different software interfaces.

2.2.2 PG/OP communication services

Overview

PG/OP services are the integrated communication functions with which SIMATIC and SIMOTION automation systems communicate with a programming device (e.g. STEP 7) and HMI devices. All SIMOTION and SIMATIC networks support the PG/OP communication services.

2.2.3 S7 communication services

Overview

S7 communication services provide data exchange using communication system function blocks (SFBs) and function blocks (FBs) for configured S7 connections.

All SIMOTION devices and SIMATIC S7 devices have integrated S7 communication services that allow the user program in the controller to initiate the reading or writing of data. These functions are independent of specific networks, allowing you to program S7 communication via any network (MPI, PROFIBUS, PROFINET or Industrial Ethernet).

For transferring data between the controllers, you must configure a connection between both controllers. The integrated communication functions are called up by the SFB/FB in the application. You can transfer up to 64 KB of data between SIMOTION and SIMATIC S7 devices.

You can access data in the controller with your HMI device, programming device (PG) or PC as the S7 communication functions are integrated in the operating system of the SIMOTION devices and SIMATIC S7 devices. This type of peer-to-peer link does not require any additional connection equipment. (However, if you configure a connection to one of these devices, you can access the data via the symbolic names.)

Note

SFBs may not be used with SIMOTION.

2.2.4 S7 basic communication services

Overview

S7 basic communication services provide data exchange using communication system functions (SFCs) for non-configured S7 connections. These SFCs (e.g. X_GET or X_PUT) read or write the data to a SIMATIC controller, so that small data volumes can be transferred via an MPI subnet to another S7 station (S7 controller, HMI or PC).

The SFCs for the S7 basic communication do not communicate with stations in other subnets. You do not need to configure connections for the S7 basic communication. The connections are established when the user program calls the SFC.

Note

You can only use the S7 basic communication services via an MPI connection between SIMATIC S7-300, S7-400 or C7-600 controllers.

2.2.5 "Global data" communication service

Overview

In addition to the other options for the network communication, you can configure a 'global data' communication connection (GD) to provide cyclic data transmission between SIMATIC controllers that are connected to an MPI network. The data exchange runs as part of the normal process image exchange, as the global data communication is integrated in the operating system of the SIMATIC controller.

As the global data communication is a process for transferring data, the receipt of the global data is not acknowledged. A publisher (data source) sends the data to one or several subscriber(s) (data sink) and subscribers receive the data. The publisher does not receive an acknowledgement from the subscribers that they have received the transmitted data.

Note

You can only use the global data communication via an MPI connection between SIMATIC S7-300, S7-400 or C7-600 controllers.

GD communication does not require any special programming or program blocks in your STEP 7 user program. The operating systems of the individual controllers process the global data exchange. Using STEP 7, you configure a global data (GD) table with the source path of the data to be transmitted to the subscribers. This GD table is downloaded with the hardware configuration for both the publisher and the subscribers.

Global data is not available for SIMOTION.

2.2.6 PROFINET communication services

Overview

PROFINET provides the following communication services:

- You can connect I/O devices and drives via a Ethernet physics to the SIMOTION or SIMATIC controller with the communication service PROFINET IO. The user program executed in the controller can process the input and output data of the I/O devices with PROFINET IO. You configure the addressing for PROFINET IO in STEP 7 or SIMOTION SCOUT.
- With PROFINET CBA, you can define your automation system as autonomous subunits or components. These components can be PROFINET IO, PROFIBUS DP or third-party devices or subnets.

If you want to use the PROFINET CBA communication services for a component-based solution, configure the SIMATIC controllers and the I/O devices in individual components in STEP 7. Then configure the communication between the various components with SIMATIC iMAP.

Both PROFINET IO and PROFINET CBA communication services provide the real-time communication required by automation systems.

Note

PROFINET CBA is only available for SIMATIC devices, but not yet for SIMOTION devices.

2.2.7 Industrial Ethernet communication services

Overview

Industrial Ethernet is based on the IEEE 802.3 and IEEE 802.3u standards and connects the automation systems with your business system, so that you also have access to the data in the office.

Industrial Ethernet provides the following communication services:

- The ISO transfer provides services for transmitting data via connections that support error-free data transmission. The ISO transfer is only possible with STEP7.
- TCP/IP allows you to exchange contiguous data blocks between the controllers and computers in PROFINET or Industrial Ethernet networks. With TCP/IP, the controller transmits contiguous data blocks.
- ISO-on-TCP (RFC 1006) supports error-free data transmission. For SIMOTION only when going though SCOUT ONLINE. If the communication is performed from the user program, an RFC must be programmed.
- UDP (User Datagram Protocol) and UDP multi-cast provide simple data transmission without acknowledgment. You can transmit contiguous data blocks from one station to another, such as between a SIMOTION and SIMATIC controller, a PC or a third-party system.
- Information technology (IT) communication allows you to share data using standard Ethernet protocols and services (such as FTP, HTTP and e-mail) via PROFINET or Industrial Ethernet networks.

2.2.8 PROFIBUS communication services

Overview

PROFIBUS provides the following communication services:

- PROFIBUS DP (Distributed Peripherals) supports the transparent communication with the distributed I/O. The SIMOTION/STEP 7 user program accesses the distributed I/O in the same manner as it accesses the I/O on the central rack of the controller (or the PLC). PROFIBUS DP enables the direct communication with the distributed I/O. PROFIBUS DP complies with the EN 61158 and EN 50170 standards.

- PROFIBUS PA (Process Automation) facilitates the direct communication with process automation (PA) instruments. This includes both cyclic access to I/O, typically with a PLC master, as well as acyclic access to the potentially large set of device operating parameters, typically with an engineering tool such as Process Device Manager (PDM). PROFIBUS PA complies with the IEC 61158 standard.
- PROFIBUS FMS (Fieldbus Message Specifications) enables the transmission of structured data (FMS variables). PROFIBUS FMS complies with the IEC 61784 standard.
- PROFIBUS FDL (Fieldbus Data Link) has been optimized for the transmission of medium-sized data volumes to support error-free data transmission on the PROFIBUS subnet. PROFIBUS FDL supports the SDA function (Send Data with Acknowledge).

Note

SIMOTION devices only support the PROFIBUS DP communication service.

For fail-safe communication, SIMOTION and SIMATIC devices use the PROFIsafe profile for PROFIBUS DP.

SIMOTION devices use the PROFIdrive profile for communication between SIMOTION devices through to the connected drives.

Additional references

You can find a comparison of the SIMATIC S7 and SIMOTION system functions in the 2_FAQ directory on the Utilities & Applications CD.

2.3 Additional services for the exchange of information

In addition to supporting the standard communication networks, SIMOTION and SIMATIC also provide additional means for sharing information via networks.

Sharing data with other applications via OPC (OLE for Process Control)

OPC (OLE for Process Control) allows Windows applications to access process data, making it easy to combine devices and applications produced by different manufacturers. OPC not only provides an open, manufacturer-independent interface, but also an easy-to-use client/server configuration for the standardized data exchange between applications (e.g. HMI or office applications) that do not require a specific network or protocol.

The OPC server provides interfaces for connecting the OPC client applications. You configure the client applications for access to data sources, e.g. addresses in the memory of a PLC. Because several different OPC clients can access the same OPC server at the same time, the same data sources can be used for any OPC-compliant application.

In addition to OPC servers, SIMATIC NET also provides applications for configuring and testing OPC connections: Advanced PC Configuration (APC) and OPC Scout (used to test and commission an OPC application or OPC server). You use these tools to connect SIMOTION and SIMATIC S7 products to other OPC-compliant applications.

The SIMATIC NET OPC servers support the following communication services:

- PROFINET IO (by means of PROFINET or Industrial Ethernet subnet)
- PROFINET CBA (by means of PROFINET or Industrial Ethernet subnet)
- TCP/IP (by means of PROFINET or Industrial Ethernet subnet)

- PROFIBUS DP (by means of PROFIBUS subnet)
- PROFIBUS FMS (by means of PROFIBUS subnet)
- S7 communication
- S5compatible communication

Using information technology (IT) for sharing data in an office environment

SIMOTION and SIMATIC use standard IT tools (such as e-mail, HTTP Web server, FTP and SNMP) with PROFINET and Industrial Ethernet networks to expand the data-sharing capabilities into the office environment.

For SIMOTION devices, the corresponding functions are made available through SIMOTION IT DIAG, see SIMOTION IT Ethernet-based HMI and Diagnostic Functions.

PROFIdrive 3

3.1 Why profiles?

Profiles used in automation technology define certain characteristics and responses for devices, device groups or whole systems which specify their main and unique properties. Only devices with manufacturer-independent profiles can behave in exactly the same way on a fieldbus and thus fully exploit the advantages of a fieldbus for the user.

Profiles are specifications defined by manufacturers and users for certain characteristics, performance features and behaviors of devices and systems. They aim to ensure a certain degree of interoperability of devices and systems on a bus which are part of the same product family due to "profile-compliant" development.

Different types of profiles can be distinguished such as so-called application profiles (general or specific) and system profiles.

- Application profiles mainly refer to devices, in this case drives, and contain an agreed selection of bus communication methods as well as specific device applications.
- System profiles describe system classes and include the master functionality, program interfaces and integration methods.

PROFIdrive

The PROFIdrive profile is a specific application profile. It contains a detailed description of how the communication functions "data exchange broadcast", "equidistance" and "isochronous operation" are used appropriately in drive applications. In addition, it specifies all device characteristics which influence interfaces connected to a controller over PROFIBUS or PROFINET. This also includes the State machine (sequential control), the encoder interface, the normalization of values, the definition of standard message frames, the access to drive parameters, the drive diagnostics, etc.

The PROFIdrive profile supports both central as well as distributed motion control concepts.

The basic philosophy: – Keep it simple –

The PROFIdrive profile tries to keep the drive interface as simple as possible and free from technology functions. This philosophy ensures that reference models as well as the functionality and performance of the PROFIBUS/PROFIDRIVE master have no or very little effect on the drive interface.

3.2 PROFIdrive overview

The PROFIdrive Profile

The PROFIdrive profile defines the device behavior and the access procedure to drive data for electrical drives on PROFIBUS and on PROFINET, from simple frequency converters up to high performance servo controllers.

PROFIdrive consists of a general part and a bus-specific part. The following properties are defined in the general part:

- Base model
- Parameter model
- Application model

The following assignments are made in the bus-specific part:

- PROFIdrive to PROFIBUS
- PROFIdrive to PROFINET

Details of where to find a precise description of the PROFIdrive profile are given below.

Literature note

PROFIdrive profile

PROFIBUS Profile PROFIdrive – Profile Drive Technology
Version V4.1, May 2006,
PROFIBUS User Organization e. V.
Haid-und-Neu-Strasse 7, 76131 Karlsruhe (Germany)
http://www.profibus.com
Order Number 3.172, specifically Chap. 6

Standards

IEC 61800 standard

3.3 PROFIdrive base/parameter model

Description

The PROFIdrive base model describes an automation system in terms of a number of devices and their interrelationships (application interfaces, parameter access). The base model distinguishes between the following device classes:

PROFIdrive	PROFIBUS DP	PROFINET IO
Controller (higher-level control or host of the automation system)	Class 1 DP master	IO controller
Peripheral device (P device)	DP slave (I slaves)	IO device
Supervisor (engineering station)	Class 2 DP master	IO Supervisor

PROFIdrive device classes

Example of a PROFIdrive automation concept

The graphic below shows a typical automation concept.

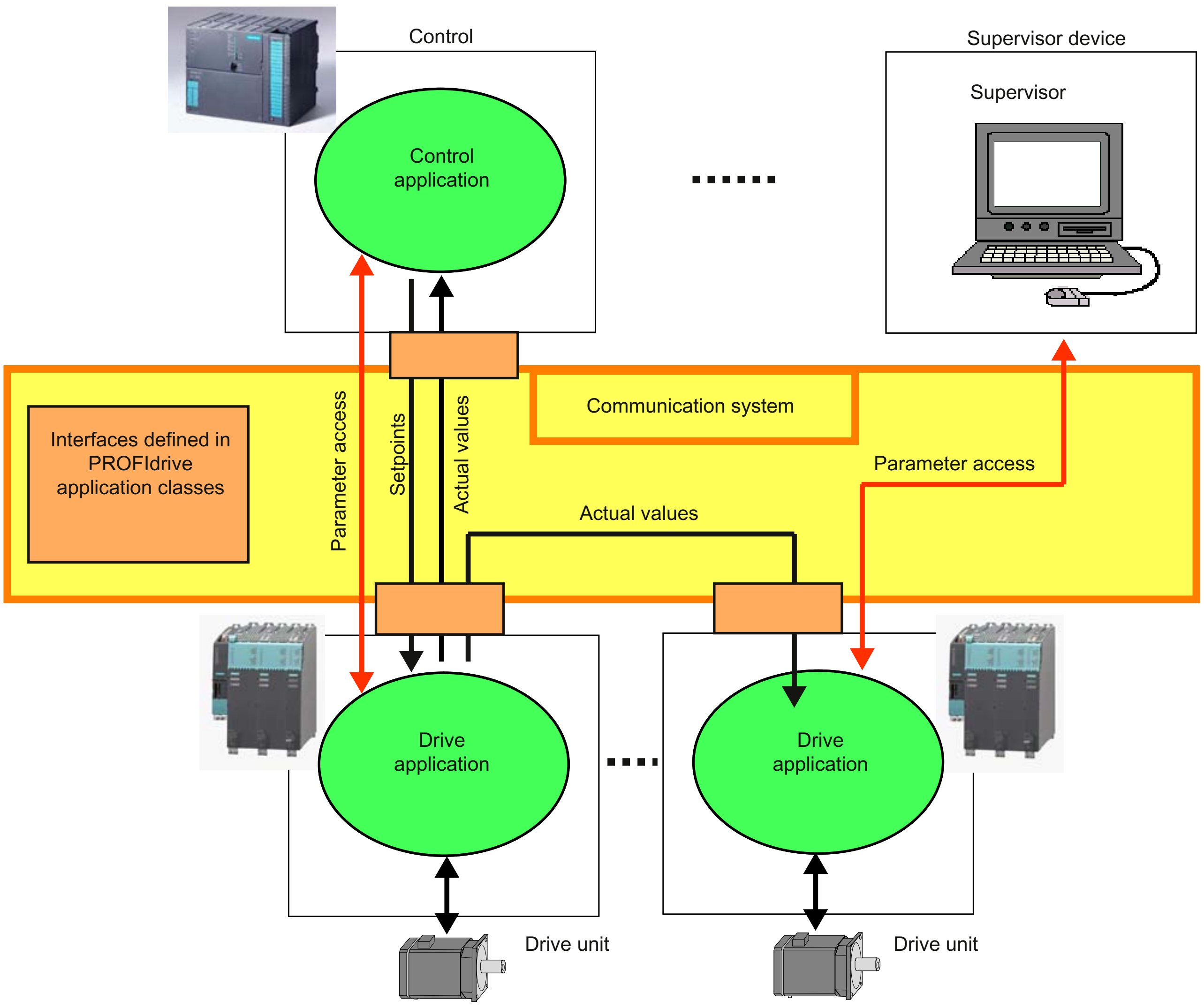


Figure 3-1 Automation concept

Communication services

Two communication services are defined in the PROFIdrive profile; namely, cyclic data exchange and acyclic data exchange.

- Cyclic data exchange via a cyclic data channel

  Motion control systems need cyclically updated data during operation for open- and closed-loop control purposes. This data must be sent to the drive units in the form of setpoints or transmitted from the drive units in the form of actual values, via the communications system. Transmission of this data is usually time-critical.

- Acyclic data exchange via an acyclic data channel

  In addition to cyclic data exchange, there is an acyclic parameter channel for exchanging parameters between the control/supervisor and drive units. Access to this data is not time-critical.

- Alarm channel

  Alarms are output on an event-driven basis, and show the occurrence and expiry of error states.

The graphic below shows the data model and data flow in the P device.

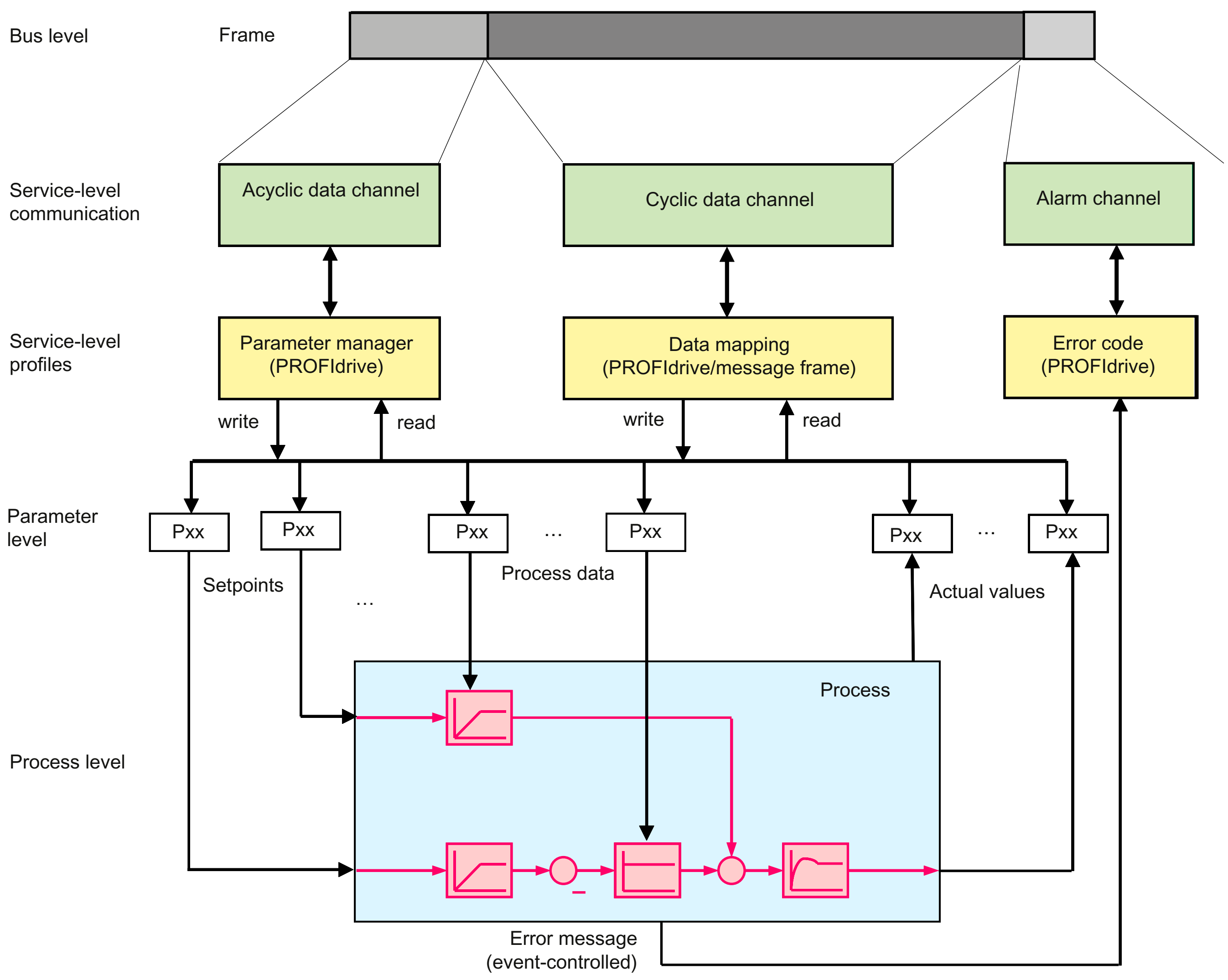


Figure 3-2 Data model and data flow in the P device

Alarms and error messages

Alarms are output on an event-driven basis, and show the occurrence and expiry of error states.

Parameter model

The parameter model in the PROFIdrive profile makes a distinction between profile parameters and manufacturer-specific parameters:

- Profile parameters are defined for objects derived from the device model of the PROFIdrive profile. These may include general functions such as drive identification, fault buffer, or drive control, for example. These parameters are the same for all drives.
- All other parameters are manufacturer-specific. The parameters are defined by the interface for the application process, rather than by the profile.

Access to a parameter's elements (values, parameter descriptions, text elements) essentially works on an acyclic basis (with the exception of G120 drives, which can exchange data cyclically with PIV). An independent request/response data structure is defined for this purpose.

3.4 Segmentation in application classes

Integration of drives in automation solutions

The integration of drives into automation solutions depends strongly upon the drive task. To cover the extensive range of drive applications from the most simple frequency converter up to highly dynamic, synchronized multi-axis systems with a single profile, PROFIdrive defines six application categories which cover most drive applications.

Table 3- 1 Table 3-1 Application/utilization categories

Category	Drive
Category 1	Standard drives (such as pumps, fans, agitators, etc.); implemented in SIMOTION and SINAMICS
Category 2	Standard drives with technology functions
Category 3	Positioning drives; implemented in SIMOTION and SINAMICS
Category 4	Motion control drives with central, higher-level motion control intelligence; implemented in SIMOTION and SINAMICS
Category 5	Motion control drives with central, higher-level motion control intelligence and the patented "Dynamic Servo Control" (DSC) position control concept; implemented in SIMOTION and SINAMICS
Category 6	Motion control drives with distributed, motion control intelligence integrated in the drives

PROFIdrive defines a device model based on function modules which cooperate in the device and generate the intelligence of the drive system.

Objects are assigned to these modules that are described in the profile and defined in terms of their function. The overall functionality of a drive is therefore described through the sum of its parameters.

In contrast to other drive profiles, PROFIdrive defines only the access mechanisms to the parameters as well as a subset of approx. 70 profile parameters such as the fault buffer, drive control and device identification.

All other parameters are manufacturer-specific which gives drive manufacturers great flexibility with respect to implementing control functions. The elements of a parameter are accessed acyclically using what is known as "Base Mode Parameter Access".

Note

Jobs involving Base Mode Parameter Access are coded in a way you may already know from data set 47 (DPV1 communication from PROFIBUS). For any differences, please refer to Specifications for PROFIBUS and PROFINET IO (Page 42).

PROFIdrive uses DP V0, DP V1, and the DP V2 expansions for PROFIBUS, and the slave data exchange broadcast and isochronous operation functions contained within them as the communication protocol.

PROFIdrive for PROFINET contains the functions for IO controller-to-IO device communication and isochronous operation.

Utilization categories

Utilization category 4 is the most important for highly dynamic and highly complex motion control tasks. This application category describes in detail the master/slave relationship between the controller and the drives which are connected to each other over PROFIBUS and PROFINET.

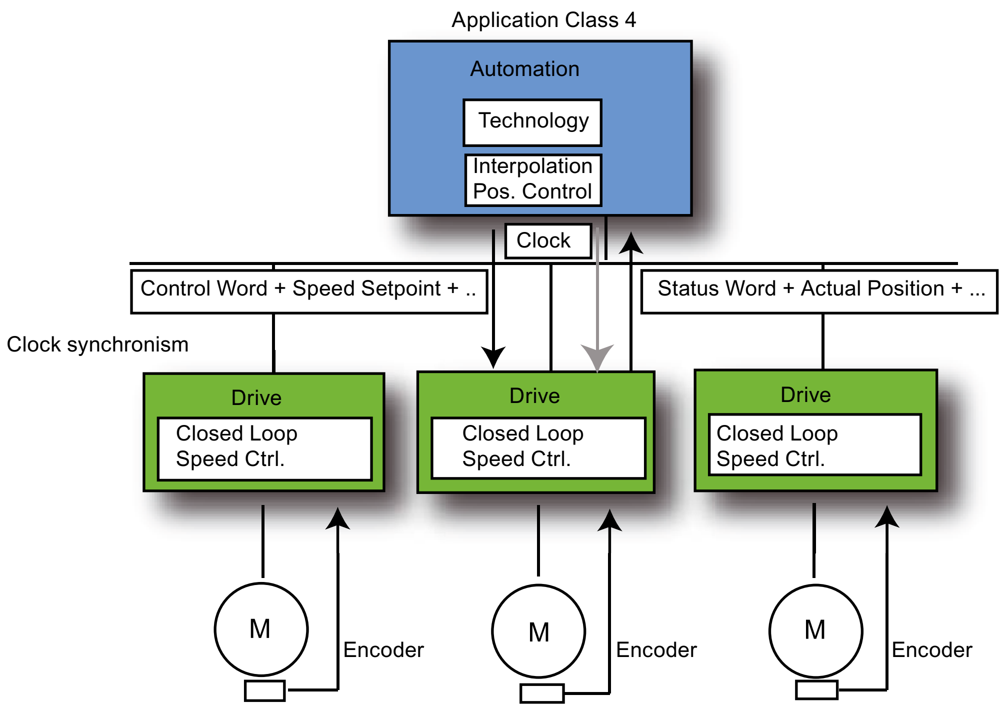


Figure 3-3 Utilization categories

The DSC (Dynamic Servo Control) function significantly improves the dynamic response and stiffness of the position control loop. With SIMOTION, this improvement usually relates to the dead times which occur for speed setpoint interfaces (transmission time, computing time for the controller and device), which are minimized by an additional, relatively simple feedback network in the drive. The position controller is pre-controlled in the drive by the SIMOTION controller using precontrol and position deviation, which enables very fast position control cycle clocks (e.g. 125 µs for servo in SINAMICS S), thereby restricting dead times to the control behavior alone.

3.5 PROFIdrive-specific data types

Description

A range of data types have been defined for the purpose of using communication that is compliant with PROFIdrive. You will find detailed information on this in the following standards:

- IEC 61800-7-203
- IEC 61800-7-303
- IEC 61158-5

These standards contain detailed descriptions of the data types. The most important data types are listed below. Data types are provided by the _readDriveParameterDescription function, for example. For more information, please also refer to .

Note

For S7 communication or communication with SINAMICS, you will have to use the *AnyType_to_BigByteArray* or *BigByteArray_to_AnyType* system functions to perform a type conversion for different data types (normalized value N2, N4; normalized value X2, X4; fixed-point value E2 and time constants T2 and T4).

PROFIdrive profile-specific data types

Data types used in the PROFIdrive profile	Definition	Coding (dec.)
Boolean	Boolean (IEC 61158-5)	1
Integer8	Integer8 (IEC 61158-5)	2
Integer16	Integer16 (IEC 61158-5)	3
Integer32	Integer32 (IEC 61158-5)	4
Unsigned8	Unsigned8 (IEC 61158-5)	5
Unsigned16	Unsigned16 (IEC 61158-5)	6
Unsigned32	Unsigned32 (IEC 61158-5)	7
FloatingPoint32	Float32 (IEC 61158-5)	8
FloatingPoint64	Float64 (IEC 61158-5)	15
VisibleString	VisibleString (IEC 61158-5)	9
OctetString	OctetString (IEC 61158-5)	10
TimeOfDay (with date indication)	TimeOfDay (IEC 61158-5)	11
TimeDifference	TimeDifference (IEC 61158-5)	12
Date	Date (IEC 61158-5)	13
TimeOfDay (without data indication)	TimeOfDay (IEC 61158-5)	52
TimeDifference (with data indication)	TimeDifference (IEC 61158-5)	53
TimeDifference (without data indication)	TimeDifference (IEC 61158-5)	54
Specific data types	**See below for description**	
N2 (normalized value (16-bit))		113

Data types used in the PROFIdrive profile	Definition	Coding (dec.)
N4 (normalized value (32-bit))		114
V2 bit sequence		115
L2 nibble		116
R2 reciprocal time constant		117
T2 time constant (16-bit)		118
T4 time constant (32-bit)		119
D2 time constant		120
E2 fixed-point value (16-bit)		121
C4 fixed-point value (32-bit)		122
X2 normalized value, variable (16-bit)		123
X4 normalized value, variable (32-bit)		124

Normalized value N2, N4

Linear normalized value, 0% corresponds to 0 (0x0), 100% corresponds to 2^{12} (0x4,000) for N2, or 2^{28} (0x40,000,000) for N4. The length is 2 or 4 octets.

Coding

Represented in two's complement; MSB (most significant bit) is the first bit after the sign bit (SN) of the first octet.

- SN = 0; positive numbers with 0
- SN = 1; negative numbers

Range of values N2, N4	Resolution N2, N4	Cod. N2, N4 (dec.)	Octet	Bit							
				8	7	6	5	4	3	2	1
$-200\% \le i \le (200-2^{-14})\%$	$2^{-12} = 0.0061$ %	113	1	SN	2^{0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
			2	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}
$-200\% \le i \le (200-2^{30})\%$	$2^{-28} = 9.3 * 10^{-8}\%$	114	3	2^{-15}	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}	2^{-21}	2^{-22}
			4	2^{-23}	2^{-24}	2^{-25}	2^{-26}	2^{-27}	2^{-28}	2^{-29}	2^{-30}

Normalized value X2, X4 (example X = 12/28)

Linear normalized value, 0% corresponds to 0 (0x0), 100% corresponds to 2^{x}. These structures are identical to N2 and N4, except that normalization is variable. Normalization can be determined from the parameter descriptions. The length is 2 or 4 octets.

Coding

Represented in two's complement; MSB (most significant bit) is the first bit after the sign bit (SN) of the first octet.

- SN = 0; positive numbers with 0
- SN = 1; negative numbers

Range of values X2, X4	Resolution X2, X4	Cod. X2, X4 (dec.)	Octet	Bit							
				8	7	6	5	4	3	2	1
-800% ≤ i ≤ 800-2^{-12})%	2^{-12}	123	1	SN	2^{2}	2^{1}	2^{0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}
			2	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}
-800% ≤ i ≤ 800-2^{-28})%	2^{-28}	124	3	2^{-13}	2^{-14}	2^{-15}	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}
			4	2^{-21}	2^{-22}	2^{-23}	2^{-24}	2^{-25}	2^{-26}	2^{-27}	2^{-28}

Fixed-point value E2

Linear fixed-point value with four places after the decimal point. 0 corresponds to 0 (0x0), 128 corresponds to 2^{14} (0x4,000). The length is 2 octets.

Coding

Represented in two's complement; MSB (most significant bit) is the first bit after the sign bit (SN) of the first octet.

- SN = 0; positive numbers with 0
- SN = 1; negative numbers

Range of values E2	Resolution	Cod. (dec.)	Octet	Bit							
				8	7	6	5	4	3	2	1
$-256+2^{-7} \leq i \leq 256-2^{-7}$	2^{-7} = 0.0078125	121	1	SN	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}
			2	2^{0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}

Fixed-point value C4

Linear fixed-point value with four places after the decimal point. 0 corresponds to 0 (0x0), 0.0001 corresponds to 2^{0} (0x0000 0001).

Coding

As with Integer32, the weighting of the bits has been reduced by a factor of 10,000.

Range of values	Resolution	Coding (dec.)	Length
-214,748.3648 ≤ i ≤ 214,748.3648	10^{-4} = 00001	122	4 octets

Bit sequence V2

Bit sequence for checking and representing application functions. 16 Boolean variables are combined to form 2 octets.

Range of values	Resolution	Cod. (dec.)	Octet	Bit							
				8	7	6	5	4	3	2	1
		115	1	15	14	13	12	11	10	9	8
			2	7	6	5	4	3	2	1	0

Nibble (half-byte) L2

Four associated bits make up a nibble. Four nibbles are represented by two octets.

Coding

Range of values	Resolution	Cod. (dec.)	Octet	Bit							
				8	7	6	5	4	3	2	1
-	-	116	1	Nibble 3				Nibble 2			
			2	Nibble 1				Nibble 0			

Time constants T2 and T4

Time data as a multiple of sampling time T_a. Interpreted value = internal value * T_a

Coding

- T2: As with Unsigned16, with a restricted range of values of $0 \leq x \leq 32,767$.
  When interpreted, internal values that fall outside this range of values are set to 0.
- T4: As with Unsigned32

The values for the time parameters of types D2, T2, T4, and R2 always relate to the specified, constant sampling time T_a. The associated sampling time (parameter p0962) is required to interpret the internal value.

Range of values	Resolution	Coding (dec.)	Length
$0 \leq i \leq 32,767 * T_a$	T_a	118	2 octets
$0 \leq i \leq 4,294,967,295 * T_a$	T_a	119	4 octets

Time constant D2

Time data as a fraction of the constant sampling time T_a. Interpreted value = internal value * T_a/16,348

Coding

- T2: As with Unsigned16, with a restricted range of values of $0 \leq x \leq 32,767$.
  When interpreted, internal values that fall outside this range of values are set to 0.

Range of values	Resolution	Coding (dec.)	Length
$0 \leq i \leq (2-2-14) * T_a$	T_a	120	2 octets

Time constant R2

Time data as a reciprocal multiple of the constant sampling time T_a. Interpreted value = 16,348 * T_a/internal value

Coding

- T2: As with Unsigned16, with a restricted range of values of $0 \leq x \leq 16,384$.
  When interpreted, internal values that fall outside this range of values are set to 16,384.

Range of values	Resolution	Coding (dec.)	Length
1 * Ta ≤ i ≤ 16,384 * T_a	T_a	117	2 octets

See also

Parameter request/response data set (Page 39)

3.6 Acyclic communication (Base Mode Parameter Access)

3.6.1 Acyclic communication

Description

PROFIdrive drive devices are supplied with control signals and setpoints by the controller and return status signals and actual values.

These signals are normally transferred cyclically (namely, continuous) between the controller and the drive.

In addition, PROFIdrive drive devices recognize parameters that contain other required data, such as error codes, warnings, control parameters, motor data. This data is normally not transferred cyclically (i.e. continuously), but "acyclically" when required. Commands for the drive can also be transferred using parameter accesses.

The reading/writing of parameters from PROFIdrive drives is always performed acyclically, using what is known as "Base Mode Parameter Access". "Base Mode Parameter Access" can be used with both PROFIBUS and PROFINET. For differences between PROFIBUS and PROFINET, please refer to Specifications for PROFIBUS and PROFINET IO (Page 42).

This service is defined and provided by PROFIdrive, and it can be used in parallel to the cyclic communication on the relevant bus. The PROFIdrive profile specifies precisely how this basic mechanism is used for write access to parameters of a PROFIdrive-compliant drive.

3.6.2 Reading and writing parameters with Base Mode Parameter Access

Description

Base Mode Parameter Access, whose structure is defined in the PROFIdrive profile, is always used for communicating the writing/reading parameters for PROFIdrive drives such as SINAMICS S120. The structure is also contained, for example, in the **Acyclic communication** section of the SINAMICS S120 Function Manual.

Under this arrangement, parameter access always consists of two elements:

- Write request ("Write data set")
- Read request ("Read data set")

This sequence must be observed, irrespective of whether read or write access is involved.

A "Write data record" is used to transfer the parameter job (for example, read parameter x). A "Read data record" is used to fetch the response for this parameter job (value of parameter x).

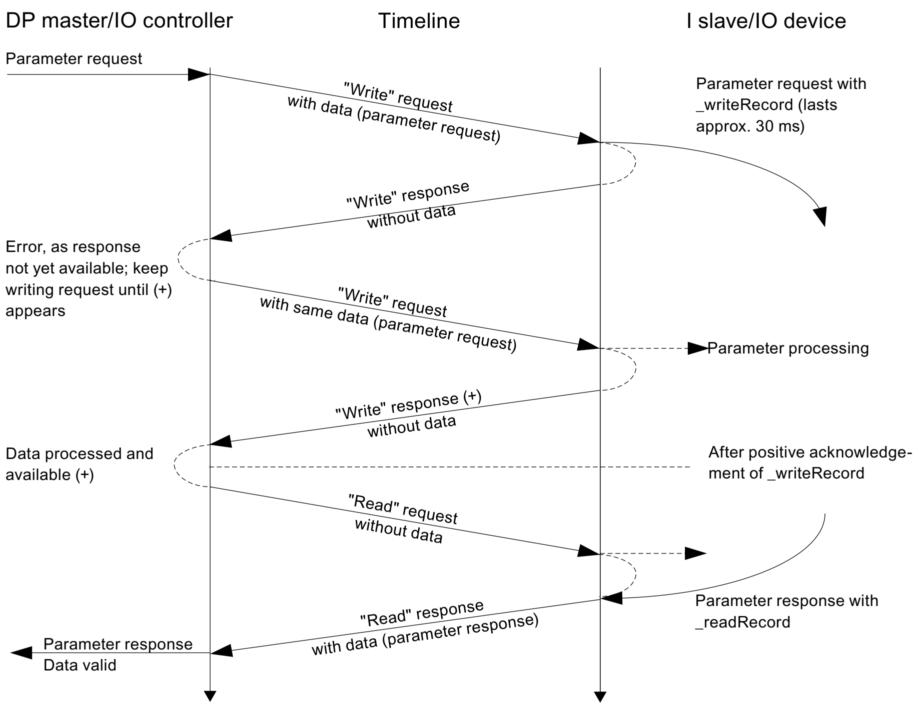

DP master/IO controller
Timeline
I slave/IO device
Parameter request
"Write" request with data (parameter request)
Parameter request with _writeRecord (lasts approx. 30 ms)
"Write" response without data
Error, as response not yet available; keep writing request until (+) appears
"Write" request with same data (parameter request)
Parameter processing
"Write" response (+) without data
Data processed and available (+)
After positive acknowledge-ment of _writeRecord
"Read" request without data
Parameter response with _readRecord
"Read" response with data (parameter response)
Parameter response
Data valid

Figure 3-4 Reading and writing acyclically

The figure **Reading and writing acyclically** shows that both "Write data set" and "Read data set" consist of the following elements:

- Request
- Response

The controller does not process this "Request Reference". However, the user program can or should process this reference.

Writing parameter records

Initially, data (P request/response data set) is transmitted to the job structure for the purpose of writing (one or more) parameter values. The data is subsequently transmitted with "Write data set" using _writeRecord. Repeated instances of "Read data set" (_writeRecord without data) can be used to monitor the status until a positive acknowledgment is given. Once this has been done, _readRecord ("Read data set") continues to be sent until the slave supplies the data.

Note

An instance of "Write data set" without data enables the status of "Write data set" with data to be determined until the positive acknowledgment is given.

If "Write data set" is successfully completed, this only signifies that the data set has been transmitted via the communication path without any errors; it does not signify that the action has been executed without any errors in the target device.

Reading parameter records

For the purpose of reading parameter values, the data block for determining which parameter(s) is/are to be read is created first. This data set is transmitted to the drive by means of the "Write data set"-"Read data set" pair of commands (first _writeRecord and then _readRecord). A subsequent "Read data set" then returns the required values **once** (the same job reference will also be returned in the response).

The processes are also represented above in the form of a diagram.

The PROFIdrive profile specifies how data larger than one byte is to be transferred. The so-called "Big Endian" format, the highest value parts are transferred first, is used:

WORD		High Byte (Byte 1)	Low Byte (Byte 2)
DOUBLE WORD	High Word	High Byte (Byte 1)	Low Byte (Byte 2)
	Low Word	High Byte (Byte 3)	Low Byte (Byte 4)

WORD and DWORD representation in Big Endian format

Since the control has a different internal data representation in certain cases, an explicit conversion must be performed when grouping and evaluating the data in the P request/response data block (data set 47).

A conversion may be required for SIMOTION, see Program example.

See also

Rule 5 - a maximum of eight concurrent calls is possible in SIMOTION (Page 54)

Parameter request/response data set (Page 39)

3.6.3 Parameter request/response data set

Structure of the P request/response data set

This always consists of:

- A header (job identifier, target axis/drive object, number of parameters in the job)
- A request reference; a reference for identifying jobs
- Job data (attribute, number of elements/indexes, parameter number and subindex), plus the values for write jobs
- Values transferred to the function

The data transmitted for a WRITE or READ request has the following structure.

	Job parameters	Byte n+1	Byte	Offset
Header	Request Header	Request Reference	RequestID	0
		Axis	Number of parameters	2
Details	1. parameter	Attribute	Number of elements	4
		Parameter number		6
		Subindex		8
	...			
	nth parameter	Attribute	Number of elements	
		Parameter number		
		Subindex		
Values only for write access	1. parameter value(s)	Format	Number of values	
		Values		
	...			
	nth parameter value(s)	Format	Number of values	
		Values		
		...		

Structure after Base Mode Parameter Access - Parameter Request

The following structure is defined for the subsequent Parameter Response. This must be retrieved with _readRecord.

	Parameter response	Byte n+1	Byte	Offset
	Request Header	Request Reference mirrored	RequestID mirrored or error	0
		Axis no./DO ID mirrored	Number of parameters	2

	Parameter response	Byte n+1	Byte	Offset
Values only for read access	1st parameter value(s)	Format	Number of values	4
		Values or error codes		6
		...		
	...			
Error values for negative response only	nth parameter value(s)	Format	Number of values	
		Values or error codes		
		...		

Structure after Base Mode Parameter Access - Parameter Response

The exact coding of the individual parts of the data structure can be obtained from the PROFIdrive profile or the SINAMICS S120 Function Manual. The assignment of "Request" and "Response", and "Write data record" and "Read data record" using the "Request Reference" job reference in the above table is important.

Request Reference

The "Request Reference" is used for the assignment of the write request to the following read request, because the control can, in principle, process several actions (as many as 8) in parallel for different target devices using the same fieldbus.

Description of fields in parameter job and response

Field	Data type	Values	Comments
Job reference	Unsigned8	0x01 to 0xFF	
	Unique identification of the job/response pair for the master. The master changes the job reference with each new job. The slave mirrors the job reference in its response.		
Job identifier	Unsigned8	0x01 0x02	Read job Write job
	Specifies the type of job. In the case of a write job, the changes are made in a volatile memory (RAM). A save operation is needed in order to transfer the modified data to the non-volatile memory (p0971, p0977).		
Response ID	Unsigned8	0x01 0x02 0x81 0x82	Read job (+) Write job (+) Read job (-) Write job (-)
	Mirrors the job identifier and specifies whether job execution was positive or negative. Negative means: Cannot execute part or all of job. The error values are transferred instead of the values for each subresponse.		
Drive object	Unsigned8	0x01 to 0xFE	Number

Field	Data type	Values	Comments
number	Setting for the drive object number on a drive unit with more than one drive object. Different drive objects with separate parameter number ranges can be accessed over the same DPV1 connection.		
Number of parameters	Unsigned8	0x01 to 0x27	No. 1 to 39 Limited by DPV1 message-frame length
	Defines the number of adjoining areas for the parameter address and/or parameter value for multi-parameter jobs. The number of parameters = 1 for single jobs.		
Attribute	Unsigned8	0x10 0x20 0x30	Value Description Text (not implemented in the case of SINAMICS)
	Type of parameter element accessed		
Number of elements	Unsigned8	0x00 0x01 to 0x75	Special function No. 1 to 117 Limited by DPV1 message-frame length
	Number of array elements accessed		
Parameter number	Unsigned16	0x0001 to 0xFFFF	No. 1 to 65535
	Addresses the parameter accessed		
Subindex	Unsigned16	0x0000 to 0xFFFE	No. 0 to 65534
	Addresses the first array element of the parameter to be accessed		
Format	Unsigned8	0x02 0x03 0x04 0x05 0x06 0x07 0x08 Other values 0x40 0x41 0x42 0x43 0x44	Data type integer8 Data type integer16 Data type integer32 Data type unsigned8 Data type unsigned16 Data type unsigned32 Data type floating point See PROFIdrive profile V3.1 Zero (without values as a positive subresponse to a write job) Byte Word Double word Error
	The format and number specify the adjoining space containing values in the message frame. Data types in conformity with PROFIdrive Profile shall be preferred for write access. Bytes, words, and double words are also possible as a substitute.		
Number of values	Unsigned8	0x00 to 0xEA	No. 0 to 234 Limited by DPV1 message-frame length
	Specifies the number of subsequent values.		

Field	Data type	Values	Comments
Error values	Unsigned16	0x0000 to 0x00FF	Meaning of error values --> see table 4-29
	The error values in the event of a negative response. If the values make up an odd number of bytes, a zero byte is appended. This ensures the integrity of the word structure of the message frame.		
Values	Unsigned16	0x0000 to 0x00FF	
	The values of the parameter for read or write access. If the values make up an odd number of bytes, a zero byte is appended. This ensures the integrity of the word structure of the message frame.		

For more information on coding PROFIdrive data types, see PROFIdrive-specific data types (Page 32).

See also

Programming example (Page 68)

Reading and writing parameters with Base Mode Parameter Access (Page 36)

3.6.4 Specifications for PROFIBUS and PROFINET IO

Global and local parameters

PROFIdrive makes a distinction between two parameter ranges:

- Global parameters; these are assigned to the drive unit as a whole. If you want to address different DOs on a drive unit, a global parameter will always show the same value.
- Local parameters; these parameters are specific to an axis or a DO. Axis-specific and DO-specific parameters can have different values for each axis/DO.

In view of this, there are two different types of access under Base Mode Parameter Access:

- Base Mode Parameter Access - local
- Base Mode Parameter Access - global

Specific properties of acyclic communication with PROFIBUS

For communication via PROFIBUS, data set 47 (0x002F) is used to access parameters in PROFIdrive drives.

- Base Mode Parameter Access – global; the drive unit's parameters (all DOs, global and local parameters) can be addressed via the drive unit's PAPs.

Note

The PROFIdrive standard specifies that in PROFIdrive drives no pipelining of jobs is supported, namely, only one "Read/write data record" is possible concurrently for a single drive device. If, however, more than one PROFIdrive drive unit is connected to a control via PROFIBUS, a job can be processed in parallel for each of these drive units. The maximum number then depends on the controller. The data for SIMOTION is specified in Rule 5 - a maximum of eight concurrent calls is possible in SIMOTION (Page 54).

Specific properties of acyclic communication with PROFINET IO

When PROFINET is used, the basic processes do not change, although the data set number is then 0xB02E ("Base Mode Parameter Access - local") and 0XB02F ("Base Mode Parameter Access - global").

- Base Mode Parameter Access - local; you can use any valid PAP of the drive unit to access a DO's global parameters. The local DO ID (axis ID) is not evaluated for local access. You will then be able to address these via the PAP's diagnostics address.
- Base Mode Parameter Access - global; you will need a valid DO ID to access a DO's local parameters. You can use any valid PAP to access a drive unit's global parameters.

3.6.5 Error assessment

Description

Two different types of errors can occur in conjunction with Base Mode Parameter Access services:

- Error in the communication (transfer of data)

  For example, the addressed device may not exist and is not switched on. This type of error is indicated with the return values of the system functions and is defined in the description of the system functions in the SIMOTION reference lists.
- Error during the processing of the jobs themselves

  For example, an attempt is made to write to a read-only parameter.

  Error codes for this second type of error are defined for PROFIdrive-compliant drives in the PROFIdrive standard and listed below.

  The ID 0x81 (hex) or 0x82 (hex) response indicates an error for the parameter access.

Error codes are returned in the drive unit's response in the P response/request data block (see table below). The "Format" field in the parameter response can be used to distinguish whether the queried parameter represents an error code or a "true" value. See the Structure after Base Mode Parameter Access - parameter response (Page 39) table, offset 4, "Format".

This table also contains the coding for the "Format" field. Code 0x44 (hex) indicates an error code in the "Values" field. Other "Format" values specify the number format (e.g. Bool, Byte, Integer8, etc.) with which the value in the "Values" field was returned.

Note

The error codes up to 0x19 correspond to the PROFIdrive profile. The error codes from 0x65 onwards are manufacturer-specific and may, therefore, vary from drive to drive.

Error codes in Base Mode Parameter Access responses

Error code	Meaning	Comments	Additional info
0x00	Illegal parameter number	Access to a parameter which does not exist	–
0x01	Parameter value cannot be changed.	Modification access to a parameter value which cannot be changed	Subindex
0x02	Lower or upper value limit exceeded	Modification access with value outside value limits	Subindex
0x03	Invalid subindex	Access to a subindex which does not exist	Subindex
0x04	No array	Access with subindex to an unindexed parameter	–
0x05	Wrong data type	Modification access with a value which does not match the data type of the parameter	–
0x06	Setting not allowed (only reset allowed)	Modification access with a value not equal to 0 in a case where this is not allowed	Subindex
0x07	Description element cannot be changed.	Modification access to a description element which cannot be changed	Subindex
0x09	No description data	Access to a description which does not exist (the parameter value exists)	–
0x0B	No operating priority	Modification access with no operating priority	–
0x0F	No text array exists	Access to a text array which does not exist (the parameter value exists)	–
0x11	Job cannot be executed due to operating mode.	Access is not possible temporarily for unspecified reasons.	–
0x14	Illegal value	Modification access with a value which is within the limits but which is illegal for other permanent reasons (parameter with defined individual values)	Subindex
0x15	Response too long	The length of the present response exceeds the maximum length that can be transferred.	–
0x16	Illegal parameter address	Impermissible or unsupported value for attribute, number of elements, parameter number, subindex, or a combination of these	–
0x17	Illegal format	Write job: illegal or unsupported parameter data format	–
0x18	No. of values inconsistent	Write job: a mismatch exists between the number of values in the parameter data and the number of elements in the parameter address.	–

Error code	Meaning	Comments	Additional info
0x19	Drive object does not exist.	You have attempted to access a drive object that does not exist.	–
0x65	Presently deactivated	You have tried to access a parameter that, although available, is currently inactive (e.g. n control set and access to parameter from V/f control).	–
0x6B	Parameter %s [%s]: no write access with enabled controller	–	–
0x6C	Parameter %s [%s]: unit unknown	–	–
0x6D	Parameter %s [%s]: Write access only in the commissioning state, encoder (p0010 = 4).	–	–
0x6E	Parameter %s [%s]: Write access only in the commissioning state, motor (p0010 = 3)	–	–
0x6F	Parameter %s [%s]: Write access only in the commissioning state, power unit (p0010 = 2)	–	–
0x70	Parameter %s [%s]: Write access only in the quick commissioning mode (p0010 = 1)	–	–
0x71	Parameter %s [%s]: Write access only in the ready state (p0010 = 0)	–	–
0x72	Parameter %s [%s]: Write access only in the commissioning state, parameter reset (p0010 = 30)	–	–
0x73	Parameter %s [%s]: Write access only in the commissioning state, safety (p0010 = 95)	–	–
0x74	Parameter %s [%s]: Write access only in the commissioning state, tech. application/units (p0010 = 5)	–	–
0x75	Parameter %s [%s]: Write access only in the commissioning state (p0010 not equal to 0)	–	–
0x76	Parameter %s [%s]: Write access only in the commissioning state, download (p0010 = 29)	–	–
0x77	Parameter %s [%s] may not be written in download.	–	–
0x78	Parameter %s [%s]: Write access only in the commissioning state, drive configuration (device: p0009 = 3)	–	–
0x79	Parameter %s [%s]: Write access only in the commissioning state, define drive type (device: p0009 = 2)	–	–
0x7A	Parameter %s [%s]: Write access only in the commissioning state, data set basis configuration (device: p0009 = 4)	–	–

Error code	Meaning	Comments	Additional info
0x7B	Parameter %s [%s]: Write access only in the commissioning state, device configuration (device: p0009 = 1)	–	–
0x7C	Parameter %s [%s]: Write access only in the commissioning state, device download (device: p0009 = 29)	–	–
0x7D	Parameter %s [%s]: Write access only in the commissioning state, device parameter reset (device: p0009 = 30)	–	–
0x7E	Parameter %s [%s]: Write access only in the commissioning state, device ready (device: p0009 = 0)	–	–
0x7F	Parameter %s [%s]: Write access only in the commissioning state, device (device: p0009 not equal to 0)	–	–
0x81	Parameter %s [%s] may not be written in download.	–	–
0x82	Transfer of the control authority (master) is inhibited by BI: p0806.	–	–
0x83	Parameter %s [%s]: requested BICO interconnection not possible	BICO output does not supply float values. The BICO input, however, requires a float value.	–
0x84	Parameter %s [%s]: parameter change inhibited (refer to p0300, p0400, p0922)	–	–
0x85	Parameter %s [%s]: access method not defined.	–	–
0xC8	Below the valid values	Modification job for a value that, although within "absolute" limits, is below the currently valid lower limit	–
0xC9	Above the valid values	Modification job for a value that, although within "absolute" limits, is above the currently valid upper limit (e.g. governed by the current inverter rating)	–
0xCC	Write access not permitted	Write access is not permitted because an access key is not available.	–

3.6.6 Additional information for the parameters of a PROFIdrive drive

Description

From a PROFIdrive drive device, not only the values of parameters, but also the descriptions of the parameters, can be read.

The P response/request data block in the "Attribute" field is used to express a preference when sending the "parameter request":

Attribute = 0x10 (hex)	Value
Attribute = 0x20 (hex)	"Parameter Description" parameter description
Attribute 0 0x30 (hex)	Parameter Text

If, rather than the value of a parameter, its "Parameter Description" is requested, the "Value" field in the "Parameter Response" contains the description (data type, possibly the number of indexes of the parameter, ...).

Note

Normally, parameter descriptions are read-only.

3.6.7 System commands in SIMOTION

3.6.7.1 _writeRecord/_readRecord SIMOTION system commands

Description

A "write data record" can be performed in SIMOTION using the _writeRecord() system command. A "read data record" can be performed in SIMOTION using the _readRecord() system command. This makes it also possible to read, write or fetch the description of parameters in a PROFIdrive drive.

The description of the system functions, their input parameters and return values is contained in the SIMOTION system documentation:

- C2xx reference list
- D4XX reference list
- P350 reference list

The _write/_readRecord system commands can be used universally, not just for PROFIdrive drives, but, for example, also for intelligent sensors on the PROFIBUS or other peripheral modules that support the so-called DP V1 services for PROFIBUS.

Note

For SIMATIC, the corresponding system functions are

SFB52 WR_REC Write data record

SFB53 RD_REC Read data record

The following is required to be able to use the SIMOTION system commands _write/_readRecord:

- PROFIBUS DP: Access is possible via a logical I/O address as well as a diagnostics address.
- PROFINET IO: Access is only possible via the diagnostics address of a Parameter Access Point (PAP).

Furthermore, the DO ID is only relevant for data set 47 (0x002f) and Global Access (PROFINET 0xb02f). The diagnostics address of the corresponding PAP is relevant for Local Access (PROFINET IO 0xb02e), the DO ID is not analyzed.

As a result, for example in connection with PROFIdrive drives, the message frame start address of the PROFIdrive message frame exchanged cyclically with the device is required.

If a drive has several axes (with a shared PROFIBUS interface connection) on a drive device, to differentiate the axes in the same device, the "Axis-No." or "DO-ID" in data set 47 is also required. SIMODRIVE 611universal and SINAMICS S120 are examples for such multi-axis drives. To determine the "DO-ID" for SINAMICS S120, refer to the **Acyclical Communication** section in the SINAMICS S120 Commissioning Manual.

"Axis-No." or "DO-ID" = 0 can be used to access the so-called "global parameters". Examples of such "global parameters" are:

- P0918: PROFIBUS address
- P0964: Device identification (manufacturer, version, number of axes, etc.)
- P0965: Profile number (the implemented PROFIdrive version)
- P0978: List of the DO Ids (the set "Axis-No." or "DO-ID")

3.6.7.2 _writeDrive.../_readDrive... SIMOTION system commands

Description

Whereas the _readRecord and _writeRecord system functions can be used universally for all devices on PROFIBUS that support the so-called "read/write data record" DP V1 services, the following commands are specially tailored to PROFIdrive drives using the PROFIdrive profile:

- _read/writeDriveParameter (reads/writes a, possibly indexed, drive parameter)
- _read/writeDriveMultiParameter (reads/writes several, possibly indexed, drive parameters for a drive or drive object)
- _readDriveFaults (reads the current fault buffer entry of a drive or drive object)
- _readDriveParameterDescription (reads the descriptive data of a parameter from the drive or drive object)
- _readDriveParameterDescription (reads the descriptive data of several parameters from the drive or drive object)

The commands create internally the data set 47 required for the individual functions in accordance with PROFIdrive profile using the parameters transferred by the user when the system functions are called, and independently handle the communication to the PROFIdrive drive using "read/write data record".

The commands are described in the SIMOTION system documentation, refer to the reference lists for the associated platform.

See also

Scope for the rules (Page 56)

3.6.7.3 Comparison of the system commands

Description

The following table shows the most important differences between the two groups of system commands:

Command group	Advantage	Disadvantages
_readRecord _writeRecord	• Generally usable, not just for DP V1 services for drives • Assumes only the knowledge of some I/O address *on the drive device* and the "DO-ID" or "Axis-No" on the drive device	• The user must create the data record • The user must program two calls for parameter accesses in a PROFIdrive drive • Users may need to perform the required data conversions themselves • "DO-ID" or "Axis-No" must be known
_readDrive... _writeDrive...	• Tailored for the typical communication with PROFIdrive drives • The user does not need to know the structure of data set 47 • Reduced programming effort for the user for communication to drives	• Assumes the presence or knowledge of an I/O address *of the associated drive object* • An I/O address for a drive object exists only for cyclical communication (with PROFIBUS) to the drive object, possibly, for example, not for TB30 and TMxx I/O expansion modules used exclusively in the drive • The user must make any required data conversions

Properties of the system commands

The use of the drive-specific _write/_readDrive... system commands one the one hand makes it easier for you than using general _write/_readRecord commands, since you do not need to know the structure of data set 47 and do not need to program the successive _writeRecord and _readRecord calls in sequencers. Because the general usability of these system functions means the structure of the transferred data records is not known to the system, you may need to perform the required conversion into the representation in accordance with the PROFIdrive profile for sending and receiving yourself, see Program example (Page 68).

On the other hand, the use of the _write/_readDrive... commands is restricted to those cases for which there is a cyclical data traffic to the associated drive object, because this is required as input parameter. In contrast, _write/_readRecord can also be used to access drive objects even when no cyclical data traffic exists (or when the I/O address is not known in the application). This succeeds with _write/_readRecord because the explicit knowledge of the "DO-ID" or "Axis-No." and the knowledge of some I/O address on the device suffices to construct the data set 47. This can be advantageous, for example, when individual drive objects are used only drive-internal (namely, without cyclical message frame traffic for control) or they are not generally known for "generic programming".

From V4.1 and up, you can also access drive objects using the _write/_readDrive... commands, when there is no cyclic data traffic, since you can transfer the "DO ID" or "Axis No" as a parameter.

3.6.7.4 Deleting _readDrive and _writeDrive jobs

Description

You can use the following functions to cancel or delete incorrect read or write jobs, which, for example, were called with the _readDriveParameter:

- _abortReadWriteRecordJobs, for the _readRecord or _writeRecord functions
- _abortAllReadWriteDriveParameterJobs, for the following functions:
  - _readDrive(Multi)ParameterDescription
  - _readDrive(Multi)Parameter
  - _writeDrive(Multi)Parameter
  - _readDriveFaults

You can call the functions without needing to know or read the CommandID.

3.6.8 Rules for using _readRecord and _writeRecord

3.6.8.1 Rule 1 - the job has its own job reference

Each job has its own job reference

This is required so that different jobs can be assigned. The job reference can be reused when the assignment is clear because of some other characteristic, such as the chronological sequence.

3.6.8.2 Rule 2 - system functions for asynchronous programming

Description

R2: For asynchronous programming, you must repeatedly call the system function with the same IDs until the function is terminated ("longrunner"). The correct use of the system functions _writeRecord and _readRecord based on communication with SINAMICS S120 is shown in the figure **Correct processing with the _readRecord and _writeRecord system functions**.

The communication for reading and writing parameters for the SINAMICS S120 is always performed using data set 47, whose structure is described in the documentation for the SINAMICS S120, refer to the Acyclical Communication section in the SINAMICS S120 Commissioning Manual.

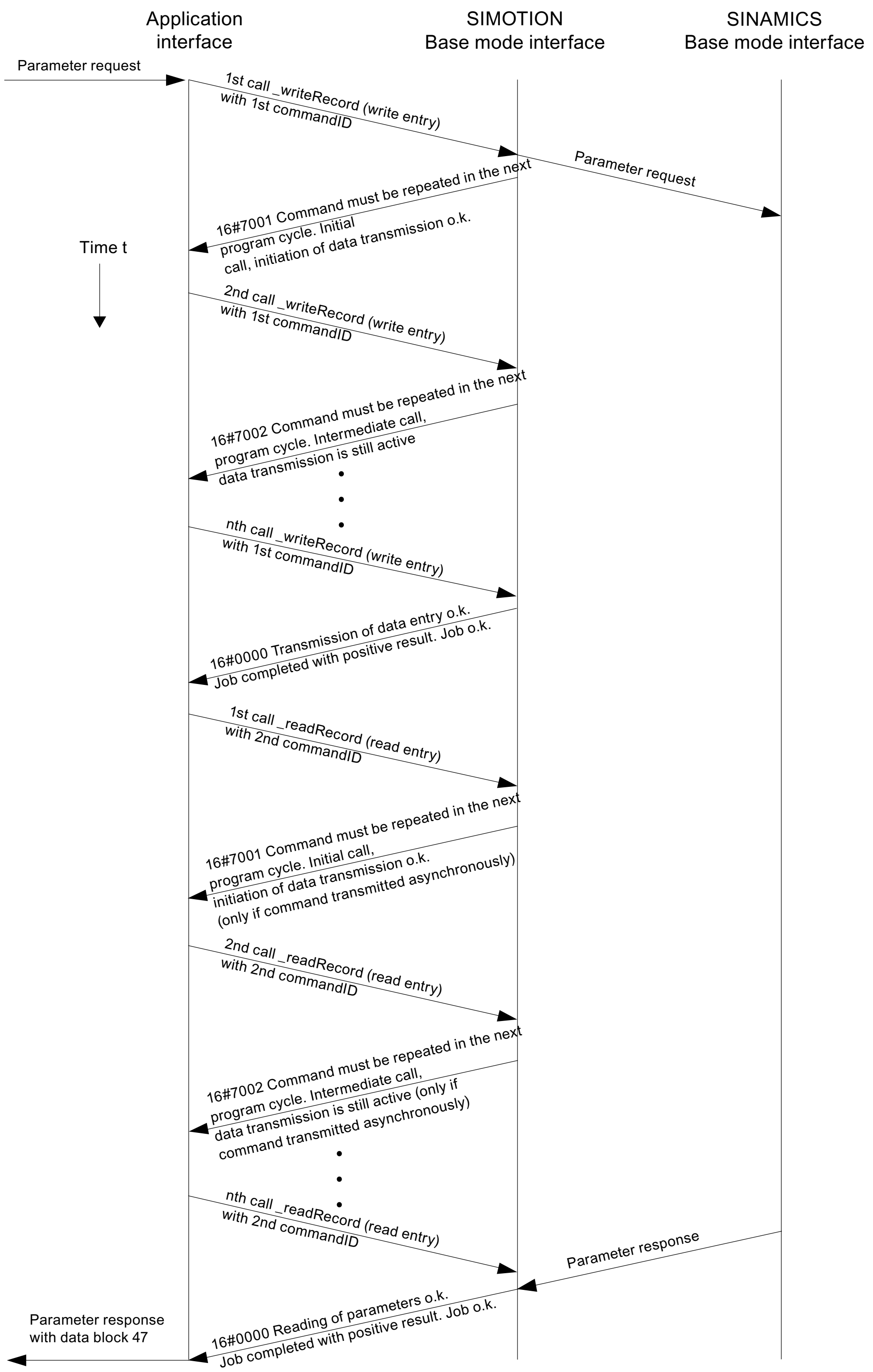

Application interface
SIMOTION Base mode interface
SINAMICS Base mode interface
Parameter request
1st call _writeRecord (write entry) with 1st commandID
Parameter request
16#7001 Command must be repeated in the next program cycle. Initial call, initiation of data transmission o.k.
Time t
2nd call _writeRecord (write entry) with 1st commandID
16#7002 Command must be repeated in the next program cycle. Intermediate call, data transmission is still active
nth call _writeRecord (write entry) with 1st commandID
16#0000 Transmission of data entry o.k. Job completed with positive result. Job o.k.
1st call _readRecord (read entry) with 2nd commandID
16#7001 Command must be repeated in the next program cycle. Initial call, initiation of data transmission o.k. (only if command transmitted asynchronously)
2nd call _readRecord (read entry) with 2nd commandID
16#7002 Command must be repeated in the next program cycle. Intermediate call, data transmission is still active (only if command transmitted asynchronously)
nth call _readRecord (read entry) with 2nd commandID
Parameter response
Parameter response with data block 47
16#0000 Reading of parameters o.k. Job completed with positive result. Job o.k.

Figure 3-5 Correct processing with the _readRecord and _writeRecord system functions

3.6.8.3 Rule 3 - read/write data record per PROFIDrive drive device

Only one read/write data record per PROFIdrive drive device concurrently

The PROFIdrive profile specifies that PROFIdrive drives do not perform any pipelining and consequently only one job will be processed at any one time. Consequently, this is also described for SINAMICS S120 in the Commissioning Manual.

Note

It does not matter which system functions are used for the transmission in the controller. A PROFIdrive drive can process only one job at any one time.

Note

It is certainly possible for other devices on the PROFIBUS that they support several "read/write data record" in parallel.

Note

Because the _write/_readRecord system functions can be used universally, *no* interlock is performed on the controller side to limit only one "read/write data record" per PROFIdrive drive to be initiated at any one time.

Consequence for the application on the controller:

An interlock must be set to prevent the application or different parts of the application from sending overlapping jobs to the same PROFIdrive drive device, also refer to section Interlocking of several calls (Page 57).

3.6.8.4 Rule 4 - the last call wins for SIMOTION

In case of doubt, the last call "wins" for SIMOTION

If Rule 3 "Only one read/write data record per PROFIdrive drive device concurrently" is violated by a second _writeRecord command being issued to the same drive in the meantime, the response of the first job can then no longer be read. The attempt to read the drive response to the first job can no longer be processed by the drive and will be acknowledged with an error and terminated. The chronological sequence is shown in the figure **The second _writeRecord call wins in case of doubt**.

To differentiate between the jobs at the controller, a separate commandID was used for each of the calls of the _writeRecord and _readRecord system functions.

To also differentiate between the jobs at the drive, unique job references for the first and second job were assigned in data set 47.

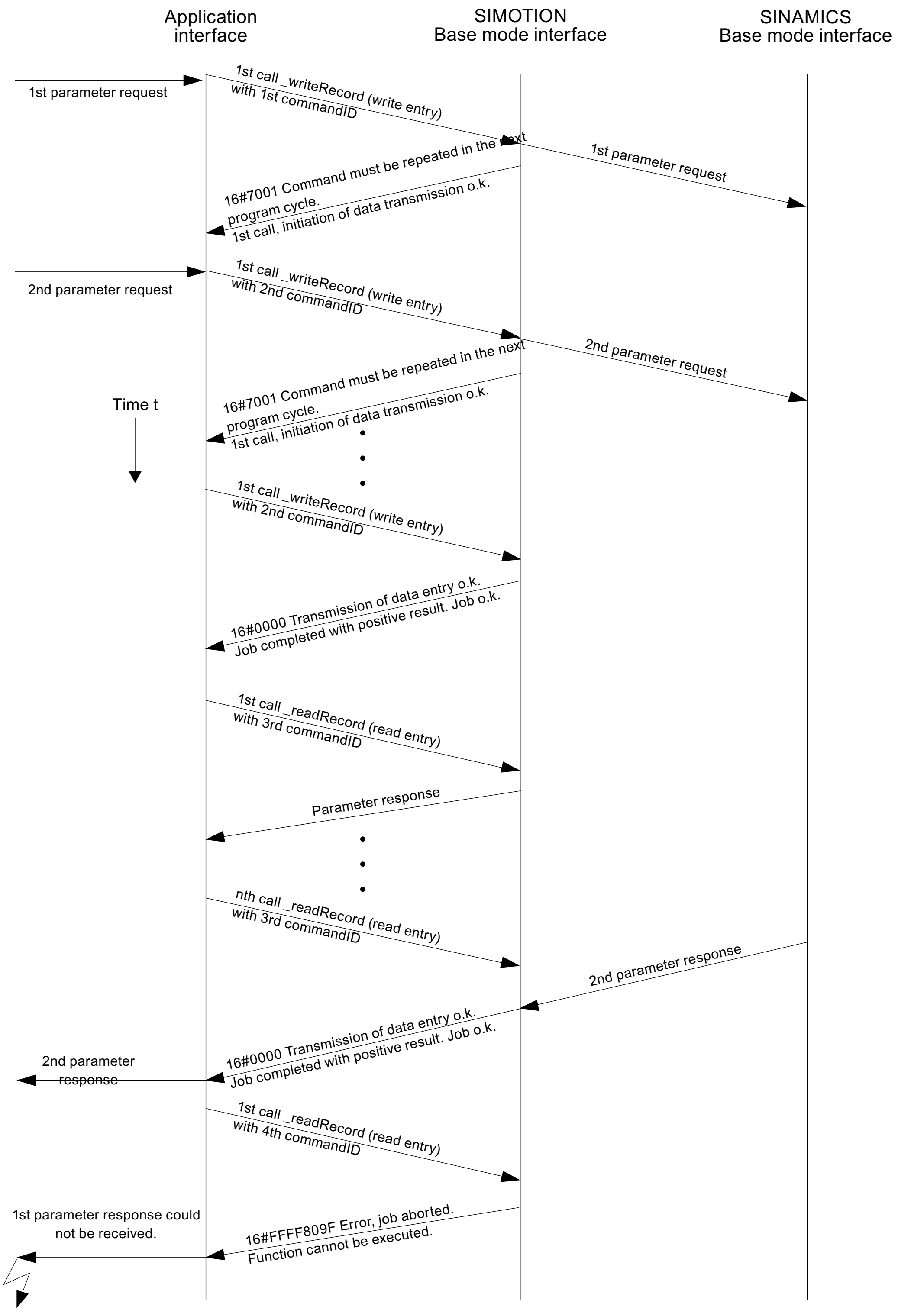

Application interface
SIMOTION Base mode interface
SINAMICS Base mode interface
1st parameter request
1st call _writeRecord (write entry) with 1st commandID
1st parameter request
16#7001 Command must be repeated in the next program cycle.
1st call, initiation of data transmission o.k.
2nd parameter request
1st call _writeRecord (write entry) with 2nd commandID
2nd parameter request
16#7001 Command must be repeated in the next program cycle.
1st call, initiation of data transmission o.k.
Time t
1st call _writeRecord (write entry) with 2nd commandID
16#0000 Transmission of data entry o.k.
Job completed with positive result. Job o.k.
1st call _readRecord (read entry) with 3rd commandID
Parameter response
nth call _readRecord (read entry) with 3rd commandID
2nd parameter response
16#0000 Transmission of data entry o.k.
Job completed with positive result. Job o.k.
2nd parameter response
1st call _readRecord (read entry) with 4th commandID
1st parameter response could not be received.
16#FFFF809F Error, job aborted.
Function cannot be executed.

Figure 3-6 The second _writeRecord call wins in case of doubt

3.6.8.5 Rule 5 - a maximum of eight concurrent calls is possible in SIMOTION

SIMOTION can manage a maximum of eight _write/_readRecord calls concurrently

Although according to rule 3 (see Rule 3 (Page 52)) only a single job can be processed at any given time for a *single* PROFIdrive drive device, it is still possible for the control program to issue several jobs in parallel.

Although this does not make any sense for a *single* PROFIdrive drive, it can be sensible for communication to *several* drives in parallel (or possibly for other devices that support this).

For SIMOTION, resources are reserved to permit a maximum of eight _write/_readRecord calls to be managed. The _write/_readRecord commandID is used to differentiate between the calls. If an attempt is made to issue a ninth concurrent call, this will be acknowledged by the controller with an error and suppressed.

The chronological sequence is shown in figure **Managing 8 jobs simultaneously**.

Initially seven _writeRecord jobs are initiated but not completed (no further _writeRecord calls to complete the jobs). The eighth _writeRecord job will be initiated and further processed until completion. It is then possible to issue a ninth call (which, however, is not further processed by the user program). The SIMOTION _writeRecord system function then acknowledges the attempt to issue the tenth job with error 16#80C3, because this would have been the ninth "open" job.

Note

The upper limit applies to each SIMOTION controller, not to each bus segment on the controller. This means it does not matter whether the addressed target devices operate on a single PROFIBUS segment or are assigned to several PROFIBUS segments.

Note

Because the _write/_readRecord system functions can be used universally, *no* interlock is performed on the controller side to limit only one "read/write data record" per PROFIdrive drive to be initiated at any one time.

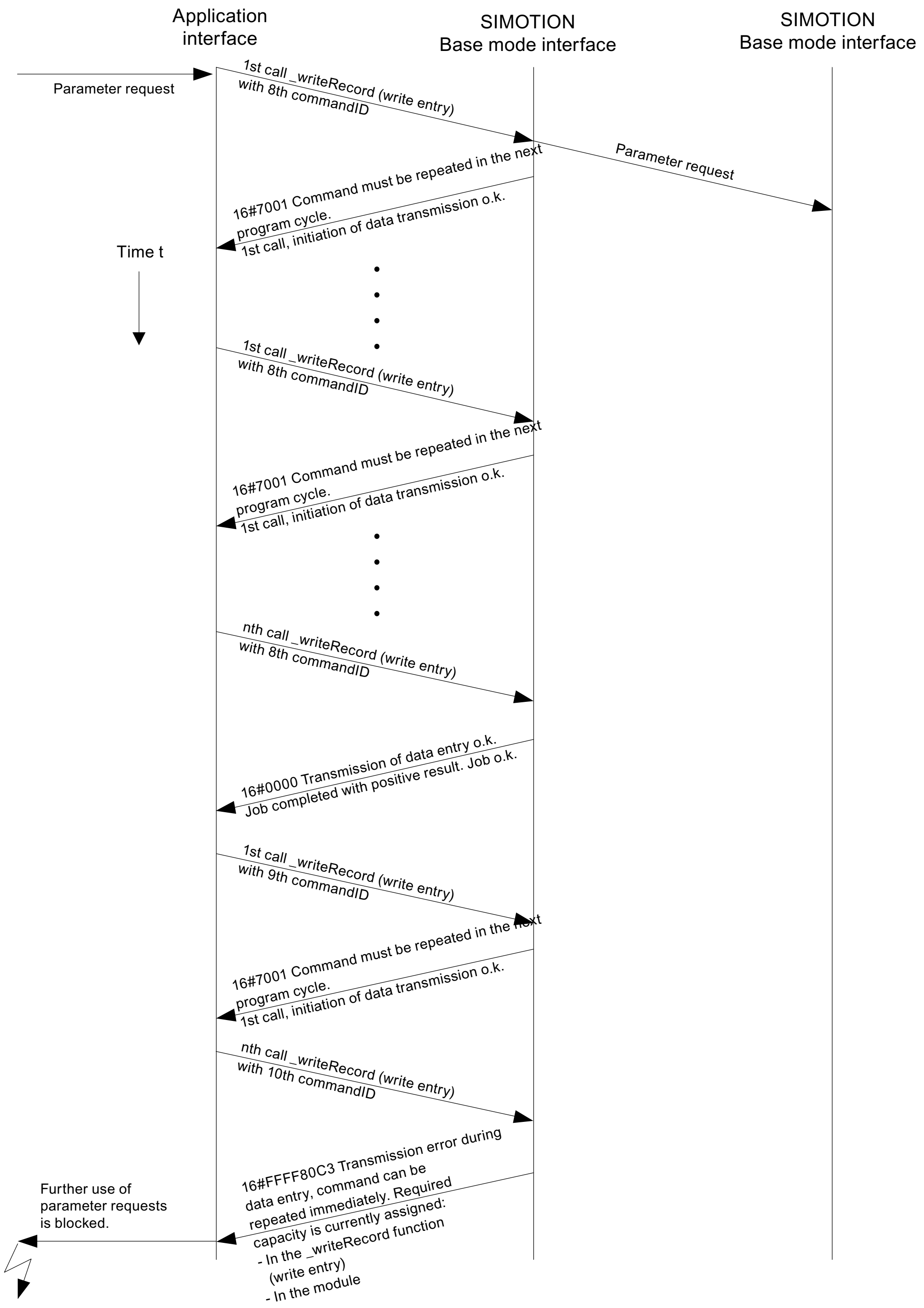

Application interface
SIMOTION Base mode interface
SIMOTION Base mode interface
Parameter request
1st call _writeRecord (write entry) with 8th commandID
Parameter request
16#7001 Command must be repeated in the next program cycle.
1st call, initiation of data transmission o.k.
Time t
1st call _writeRecord (write entry) with 8th commandID
16#7001 Command must be repeated in the next program cycle.
1st call, initiation of data transmission o.k.
nth call _writeRecord (write entry) with 8th commandID
16#0000 Transmission of data entry o.k.
Job completed with positive result. Job o.k.
1st call _writeRecord (write entry) with 9th commandID
16#7001 Command must be repeated in the next program cycle.
1st call, initiation of data transmission o.k.
nth call _writeRecord (write entry) with 10th commandID
16#FFFF80C3 Transmission error during data entry, command can be repeated immediately. Required capacity is currently assigned:
- In the _writeRecord function (write entry)
- In the module
Further use of parameter requests is blocked.

Figure 3-7 Managing 8 jobs simultaneously

Note

If the error 16#80C3 occurs, you must set the CPU to STOP and then back to RUN. This deletes the job buffer. In order to prevent the error, you should end the job with an abort command, if you are unable to end the job.

3.6.9 Rules for SIMOTION _writeDrive.../_readDrive... commands

3.6.9.1 Scope for the rules

Description

The following examples are shown using the _readDriveParameter system function . The descriptions also apply similarly for the previously mentioned _writeDrive.../_readDrive... system functions.

3.6.9.2 Rule 6 - repeated call of system function for asynchronous programming

Description

For asynchronous programming, the user must call repeatedly the system function with the same IDs until the function is terminated ("longrunner").

The following figure shows the correct use of the _readDriveParameter system function.

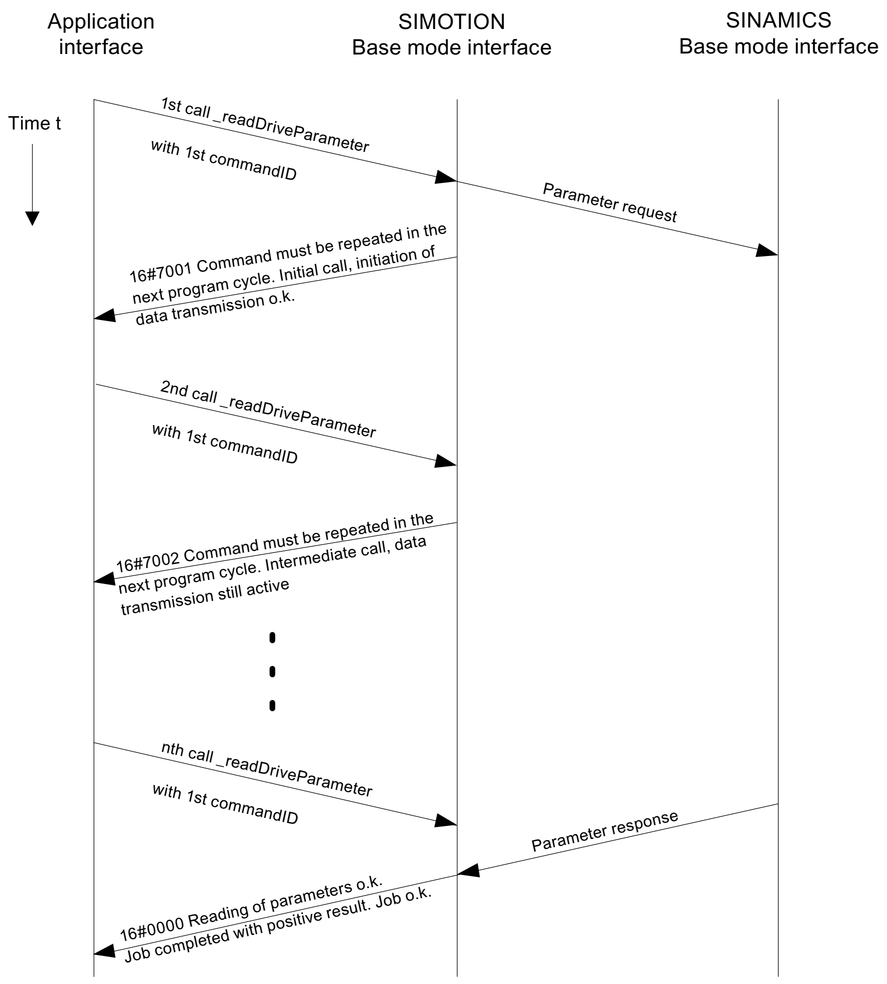

Application interface
SIMOTION Base mode interface
SINAMICS Base mode interface
Time t
1st call _readDriveParameter with 1st commandID
Parameter request
16#7001 Command must be repeated in the next program cycle. Initial call, initiation of data transmission o.k.
2nd call _readDriveParameter with 1st commandID
16#7002 Command must be repeated in the next program cycle. Intermediate call, data transmission still active
nth call _readDriveParameter with 1st commandID
Parameter response
16#0000 Reading of parameters o.k. Job completed with positive result. Job o.k.

Figure 3-8 Correct processing with the _writeDriveParameter and _readDriveParameter system functions

3.6.9.3 Rule 7 - multiple concurrent calls per target device

Description

The PROFIdrive standard specifies that PROFIdrive drives do not perform any pipelining and consequently only one job will be processed at any one time. Consequently, this is also documented for SINAMICS S120 in the SINAMICS S120 Commissioning Manual.

Because the SIMOTION _write/_readDrive... system commands have been created for the frequent use with PROFIdrive drives, this is already handled by the controller.

Note

It does not matter which system functions are used for the transmission in the controller. A PROFIdrive drive can process only one job at any one time.

Consequence for the application on the controller:

An interlock must be set to prevent the application or different parts of the application from sending overlapping jobs to the same PROFIdrive drive device.

The figure below shows the behavior when this is not handled. The attempt to issue a second job (with unique commandID) to the same target device will be acknowledged with an error. A further job to the same target device can then be issued only when the first job has completed or has been canceled, see Section Releasing the Interlocking (Page 58).

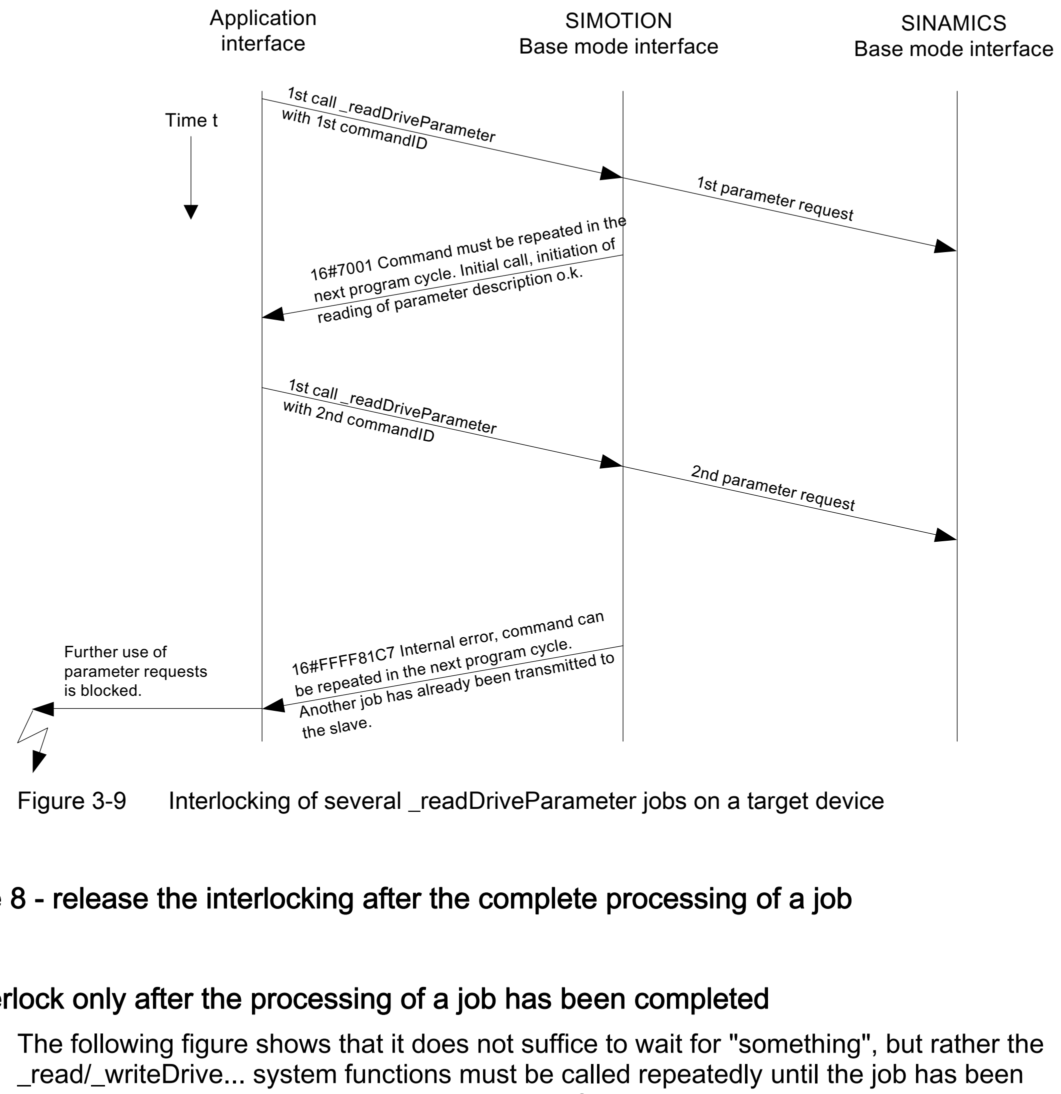

Application interface
SIMOTION Base mode interface
SINAMICS Base mode interface
Time t
1st call _readDriveParameter with 1st commandID
1st parameter request
16#7001 Command must be repeated in the next program cycle. Initial call, initiation of reading of parameter description o.k.
1st call _readDriveParameter with 2nd commandID
2nd parameter request
Further use of parameter requests is blocked.
16#FFFF81C7 Internal error, command can be repeated in the next program cycle. Another job has already been transmitted to the slave.

Figure 3-9 Interlocking of several _readDriveParameter jobs on a target device

3.6.9.4 Rule 8 - release the interlocking after the complete processing of a job

Release the interlock only after the processing of a job has been completed

The following figure shows that it does not suffice to wait for "something", but rather the _read/_writeDrive... system functions must be called repeatedly until the job has been processed completely. The interlock will not be freed and the internal management resources released beforehand.

The number of calls has been selected so that the SIMOTION DP V1 interface answers each subsequent call for the first job with 16#7002 and thus is not processed completely. Depending on the loading of the bus and the drive, this can also be necessary very frequently (>25 times). This means an estimate cannot be given.

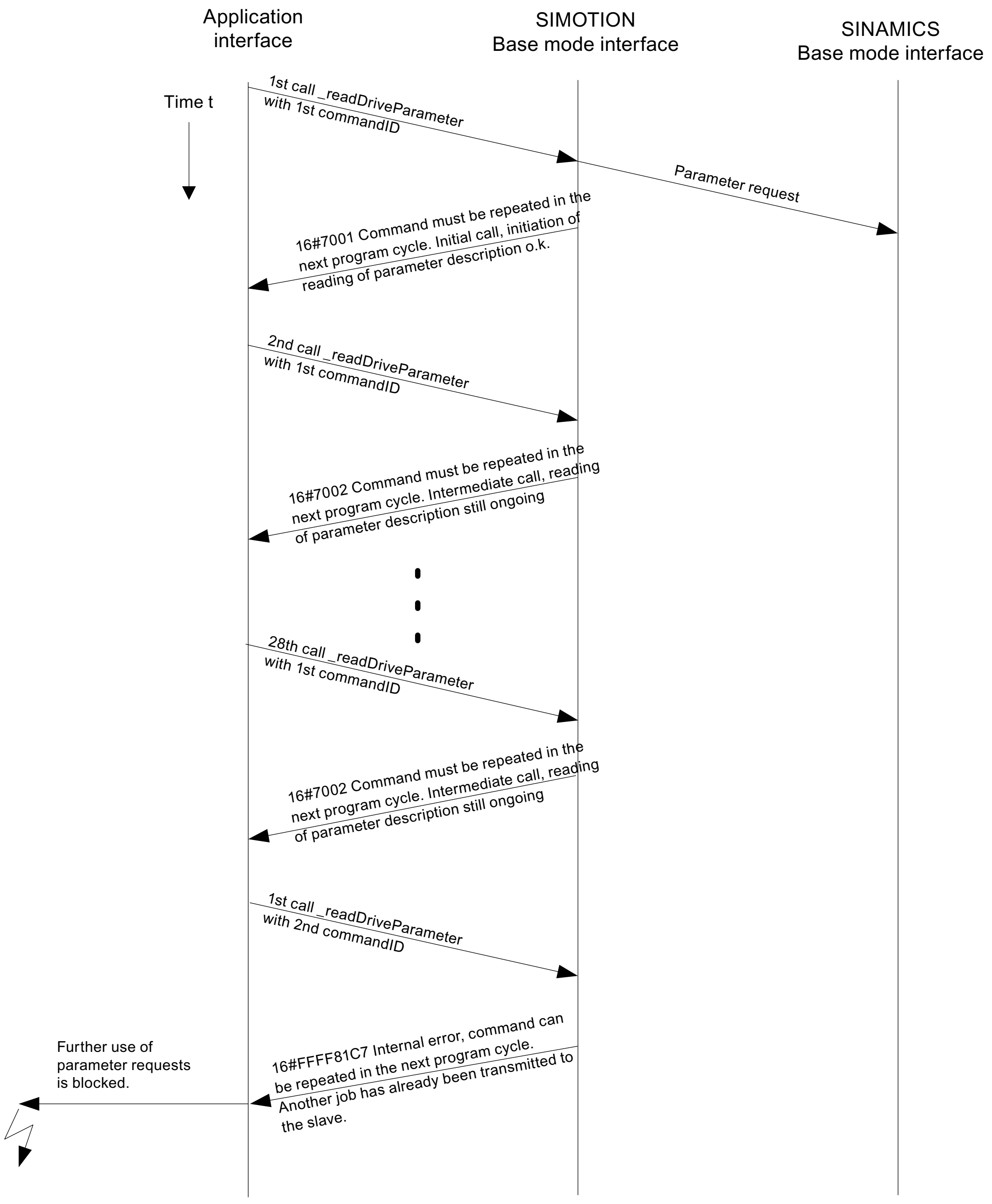

Application interface
SIMOTION Base mode interface
SINAMICS Base mode interface
Time t
1st call _readDriveParameter with 1st commandID
Parameter request
16#7001 Command must be repeated in the next program cycle. Initial call, initiation of reading of parameter description o.k.
2nd call _readDriveParameter with 1st commandID
16#7002 Command must be repeated in the next program cycle. Intermediate call, reading of parameter description still ongoing
28th call _readDriveParameter with 1st commandID
16#7002 Command must be repeated in the next program cycle. Intermediate call, reading of parameter description still ongoing
1st call _readDriveParameter with 2nd commandID
16#FFFF81C7 Internal error, command can be repeated in the next program cycle. Another job has already been transmitted to the slave.
Further use of parameter requests is blocked.

Figure 3-10 Complete processing of a _readDriveParameter required to release the interlock

3.6.9.5 Rule 9 - canceling jobs for an asynchronous call

CommandID is needed to cancel jobs for an asynchronous call

To re-enable the DP V1 service for the target device,

- either the first job must have completed (repeated calls with the commandID of the first job)
- or cancelled (again a call of the _readDriveParameter function with the same commandID as for the first initiation of the job. In addition, the nextCommand input parameter must have the ABORT_CURRENT_COMMAND value).

Note

From V4.1 and up, it is possible to cancel without knowing the commandID, see Deleting _readDrive and _writeDrive jobs (Page 50) .

A sample call of the _readDriveParameter function with the first commandID (id1) and ABORTED_CURRENT_COMMAND has the following form:

```
Return_Par_read_delete :=
    readDriveParameter(
    ioId:=INPUT,
    logAddress := 256,
    parameterNumber := number,
    numberOfElements := 0,
    subIndex:= 0,
    nextCommand :=
    ABORT_CURRENT_COMMAND,
    commandId := id1);
```

The figure below shows the chronological sequence.

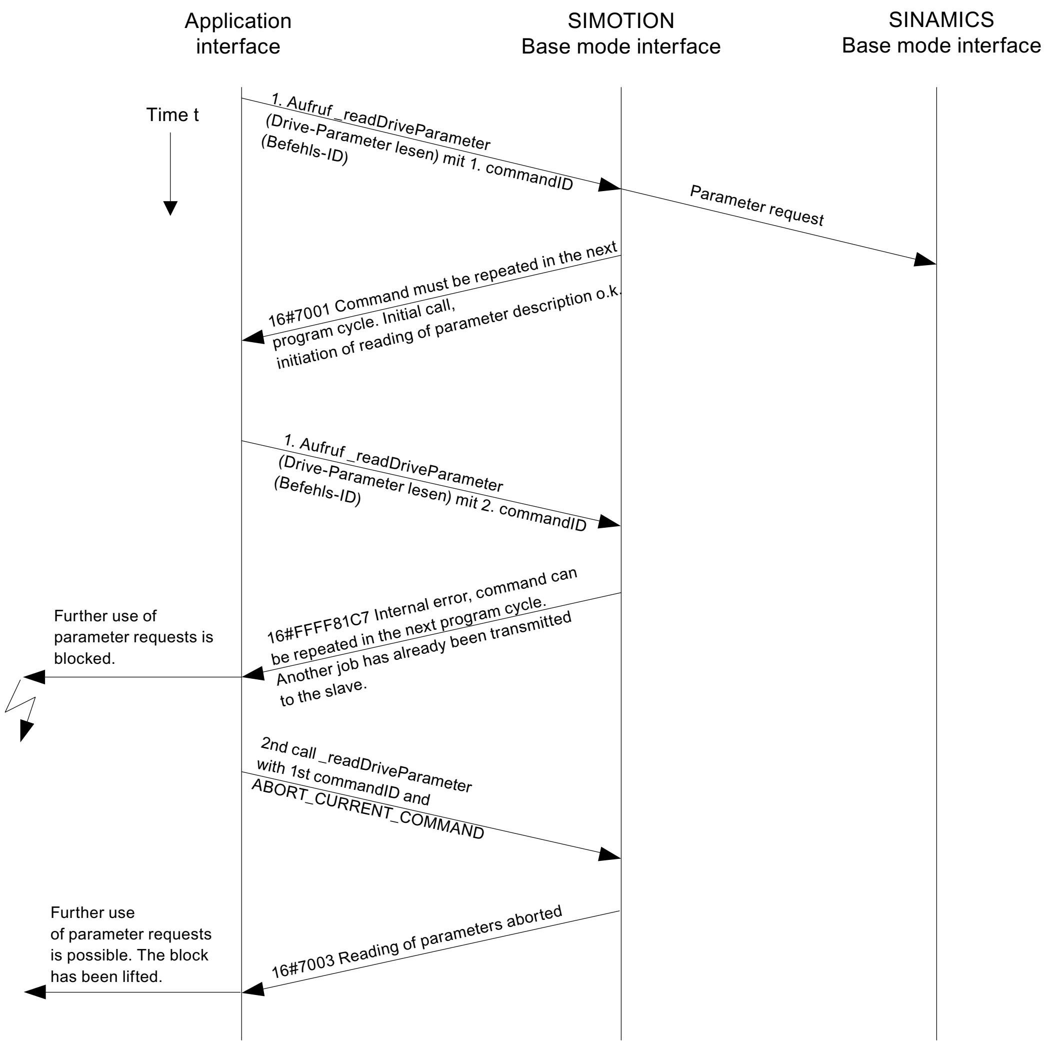

Application interface
SIMOTION Base mode interface
SINAMICS Base mode interface
Time t
1. Aufruf _readDriveParameter (Drive-Parameter lesen) mit 1. commandID (Befehls-ID)
Parameter request
16#7001 Command must be repeated in the next program cycle. Initial call, initiation of reading of parameter description o.k.
1. Aufruf _readDriveParameter (Drive-Parameter lesen) mit 2. commandID (Befehls-ID)
16#FFFF81C7 Internal error, command can be repeated in the next program cycle. Another job has already been transmitted to the slave.
Further use of parameter requests is blocked.
2nd call _readDriveParameter with 1st commandID and ABORT_CURRENT_COMMAND
16#7003 Reading of parameters aborted
Further use of parameter requests is possible. The block has been lifted.

Figure 3-11 Canceling a _readDriveParameter job with known commandID

The process in the following figure shows that it is not possible to cancel a job without knowledge of the original commandID. Not the first job, but rather the cancel attempt will be canceled. The reason is that the commandID is used for managing the various jobs in the system.

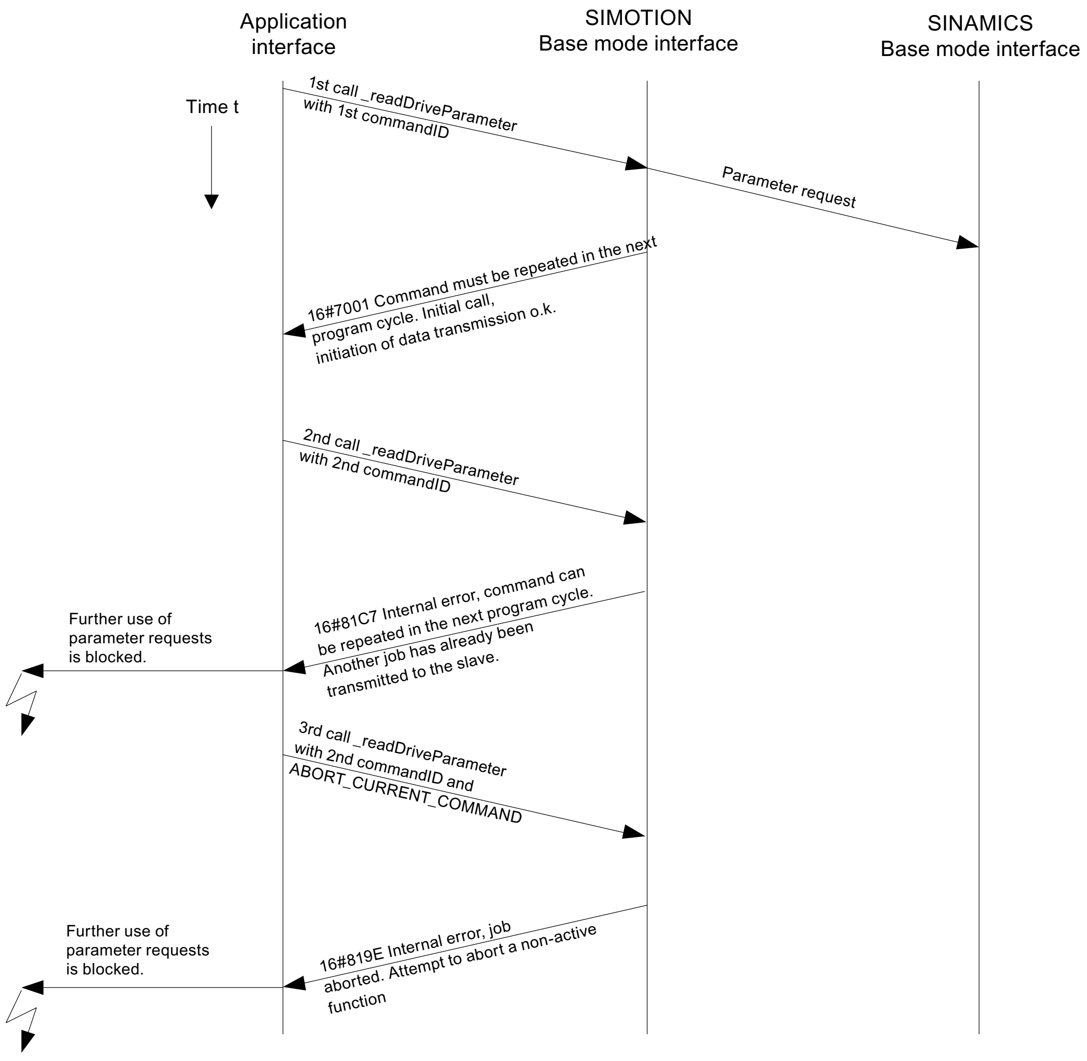

Application interface
SIMOTION Base mode interface
SINAMICS Base mode interface
Time t
1st call _readDriveParameter with 1st commandID
Parameter request
16#7001 Command must be repeated in the next program cycle. Initial call, initiation of data transmission o.k.
2nd call _readDriveParameter with 2nd commandID
16#81C7 Internal error, command can be repeated in the next program cycle. Another job has already been transmitted to the slave.
Further use of parameter requests is blocked.
3rd call _readDriveParameter with 2nd commandID and ABORT_CURRENT_COMMAND
16#819E Internal error, job aborted. Attempt to abort a non-active function
Further use of parameter requests is blocked.

Figure 3-12 No cancelation of a _readDriveParameter job with new CommandID

Note

It is therefore important that the user program retains the commandID of the jobs until the job has completed or has been canceled.

Note

Take particular care, for example, through control by other conditions, that in the user program the processing of the _write/_readDrive... functions is not bypassed before they have completed.

3.6.9.6 Rule 10 - management of sixteen jobs

SIMOTION manages a maximum of 16 calls in parallel for different devices

The controller has limited resources (memory space) available for storing the management data for _write/_readDrive... system function calls. If too many calls are issued in parallel, an error message will be issued, similar to the limit for _read/_writeRecord in Section Maximum Number of Calls (Page 54).

For SIMOTION, resources are reserved to permit a maximum of sixteen calls of _writeDrive.../_readDrive... system functions to be managed. The commandID is used to differentiate between the calls. If an attempt is made to issue a seventeenth concurrent call, this will be acknowledged by the controller with an error and suppressed.

3.6.9.7 Rule 11 - parallel jobs for different drive devices

Parallel jobs to different drive devices are possible

The figure **Parallel processing of _readDriveParameter jobs to different drive devices of a controller** shows that parallel jobs can be processed with different drive devices.. The SIMOTION D445 controller uses the SINAMICS Integrated of a D445 (for example) as first PROFIdrive drive device and the CX32 expansion module as second PROFIdrive drive device.

In the example, a total of three read jobs (two jobs to the first drive device (SINAMICS Integrated) and one job to the second drive device (CX32)) are issued with the _readDriveParameter system function.

- The first read job for the SINAMICS Integrated is intentionally called just once so that the interlock acts.
- The second read job is then issued to the second PROFIdrive drive device (CX32). This job is processed successfully.
- The third read job is addressed again to the first drive device (SINAMICS Integrated) and can no longer be executed successfully because the first job is still running.

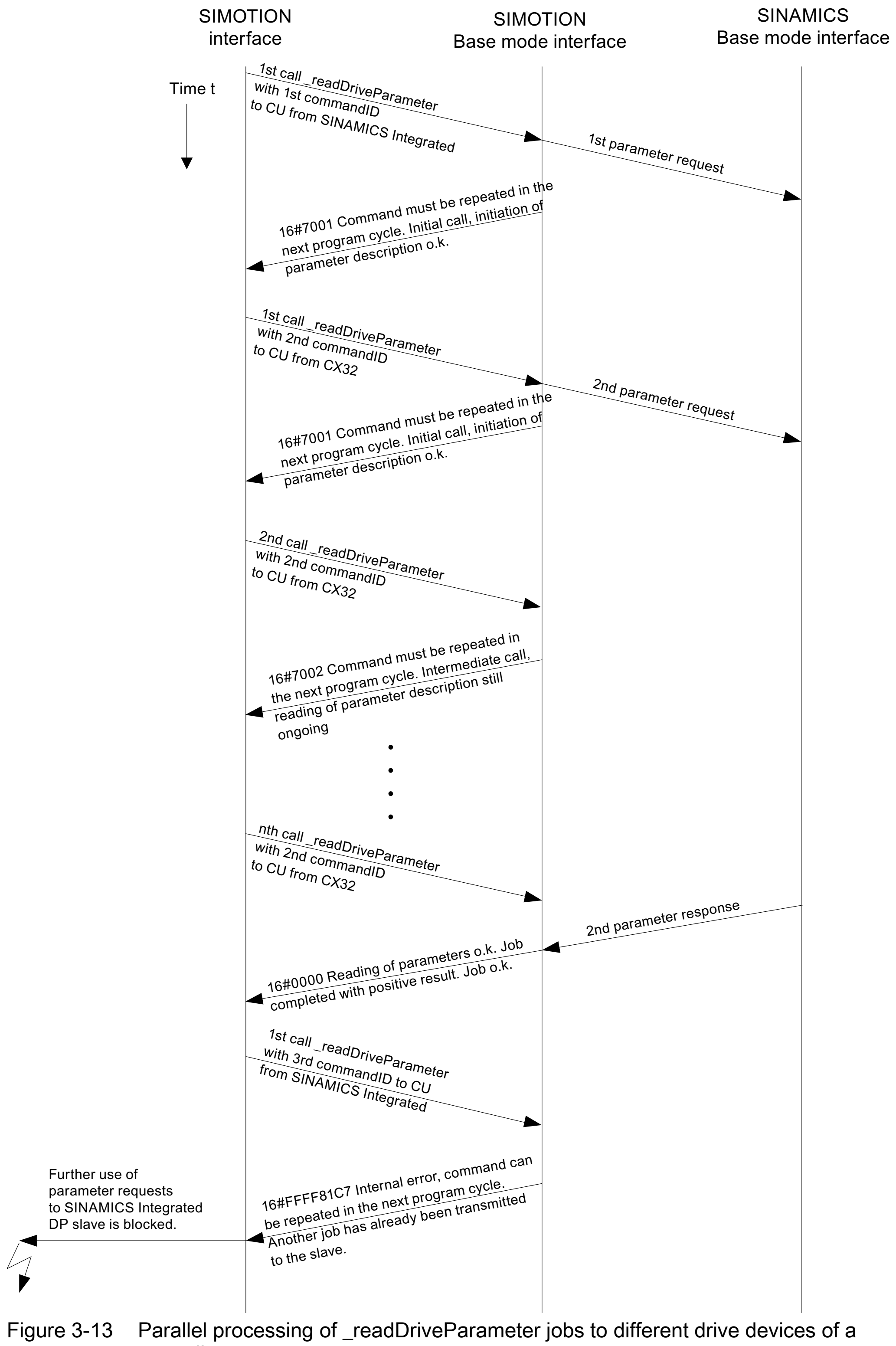

SIMOTION interface
SIMOTION Base mode interface
SINAMICS Base mode interface
Time t
1st call _readDriveParameter with 1st commandID to CU from SINAMICS Integrated
1st parameter request
16#7001 Command must be repeated in the next program cycle. Initial call, initiation of parameter description o.k.
1st call _readDriveParameter with 2nd commandID to CU from CX32
2nd parameter request
16#7001 Command must be repeated in the next program cycle. Initial call, initiation of parameter description o.k.
2nd call _readDriveParameter with 2nd commandID to CU from CX32
16#7002 Command must be repeated in the next program cycle. Intermediate call, reading of parameter description still ongoing
nth call _readDriveParameter with 2nd commandID to CU from CX32
2nd parameter response
16#0000 Reading of parameters o.k. Job completed with positive result. Job o.k.
1st call _readDriveParameter with 3rd commandID to CU from SINAMICS Integrated
Further use of parameter requests to SINAMICS Integrated DP slave is blocked.
16#FFFF81C7 Internal error, command can be repeated in the next program cycle. Another job has already been transmitted to the slave.

Figure 3-13 Parallel processing of _readDriveParameter jobs to different drive devices of a controller

3.6.10 Special features

3.6.10.1 Rule 12 - data buffering of up to 64 drive objects

SIMOTION buffers the data of up to 64 drive objects

The first call to the functions _write/_readDrive... after system power-up runs considerably longer than subsequent calls to the same drive object.

- The system must first set up internal management information that can be accessed faster in subsequent calls to the same drive object.

In SIMOTION, the data of up to 64 drive objects can be stored for use with _write/_readDrive... The distinction is made using the I/O address.

3.6.10.2 Rule 13 - a mix of system functions can be used

A mix of the _writeRecord/_readRecord and _writeDrive.../_readDrive... system functions can be used

A mixed use of the following system commands is generally possible:

- _writeRecord/_readRecord SIMOTION system commands
- _writeDrive.../_readDrive... SIMOTION system commands

Note

However, it is important to appreciate that a missing interlock of the system commands from the two command groups means several jobs could be issued to a PROFIdrive drive (see following section), which a PROFIdrive drive cannot process. Handling the system-internal interlocking for _write/_readDrive....

The figure **Mixed use of _readDrive... and _read/_writeRecord** shows that the _write/_readRecord functions, in particular, can be used for the same target device when, because of a running _readDriveParameter job, further jobs with the same command are suppressed by the system – this situation must be blocked by the user because it cannot be processed by a PROFIdrive.

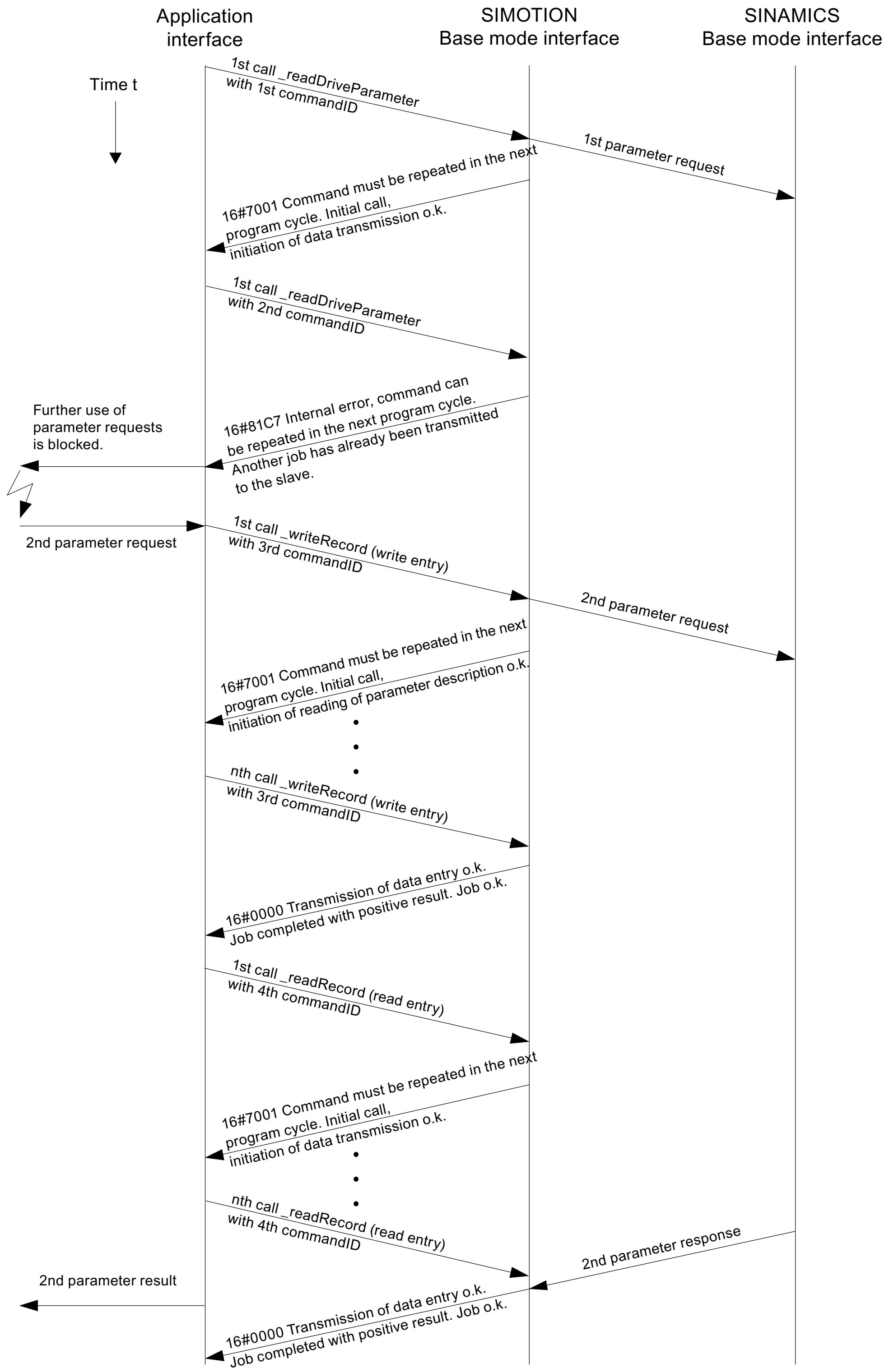

Application interface
SIMOTION Base mode interface
SINAMICS Base mode interface
Time t
1st call _readDriveParameter with 1st commandID
1st parameter request
16#7001 Command must be repeated in the next program cycle. Initial call, initiation of data transmission o.k.
1st call _readDriveParameter with 2nd commandID
Further use of parameter requests is blocked.
16#81C7 Internal error, command can be repeated in the next program cycle. Another job has already been transmitted to the slave.
2nd parameter request
1st call _writeRecord (write entry) with 3rd commandID
2nd parameter request
16#7001 Command must be repeated in the next program cycle. Initial call, initiation of reading of parameter description o.k.
nth call _writeRecord (write entry) with 3rd commandID
16#0000 Transmission of data entry o.k. Job completed with positive result. Job o.k.
1st call _readRecord (read entry) with 4th commandID
16#7001 Command must be repeated in the next program cycle. Initial call, initiation of data transmission o.k.
nth call _readRecord (read entry) with 4th commandID
2nd parameter response
2nd parameter result
16#0000 Transmission of data entry o.k. Job completed with positive result. Job o.k.

Figure 3-14 Mixed use of _readDrive... and _read/_writeRecord

3.6.10.3 Rule 14 - interlocking for the mixed use of commands

The user must interlock for the mixed use of the commands from the two command groups

When the following system commands are used together, it is possible that more than one "read/write data record" is issued concurrently to a single device because for SIMOTION interlocking and buffering is performed only within the command groups but not between command groups.

- _writeRecord/_readRecord SIMOTION system commands

and

- _writeDrive.../_readDrive... SIMOTION system commands

If necessary, this must be interlocked by the user to prevent data loss/overlapping because the PROFIdrive profile specifies that a PROFIdrive drive does not perform any pipelining and consequently can process only one job at any given time.

3.6.11 Program examples

3.6.11.1 Programming example

Description

The following example shows how the _writeRecord and _readRecord system commands can be used to fetch the error code from parameter p0945 of a SINAMICS drive (drive object DO3, I/O address 256).

Example

The sample program can be called, for example, in the BackgroundTask, because so-called "asynchronous programming" is used.

```
//===========================================================================
// demonstrate reading parameter 945 (fault code) via data set 47
// using SIMOTION system functions _write/_readRecord (asynchronous call)
// INPUT address 256 is assumed to address the SINAMICS
// drive is DO3 in SINAMICS S120
//===========================================================================
INTERFACE
 PROGRAM record;
 // declare request type
 TYPE
 // declare struct of header request
 Header_Type_Request : STRUCT
    Request_Reference : USINT;
    Request_Id : USINT;
```

```
     Axis : USINT;
     Number_Of_Parameter : USINT;
  END_STRUCT;

  // declare struct of parameter address request
  Parameter_Address_Request : STRUCT
     Attribute : USINT;
     Number_Of_Elements : USINT;
     Parameter_Number : UINT;
     SubIndex : UINT;
  END_STRUCT;

  // declare struct of request
  Request : STRUCT
     Header : Header_Type_Request;
     ParameterAddress : Parameter_Address_Request;
  END_STRUCT;
 // declare struct of header response
  Header_Type_Response : STRUCT
  Response_Reference : USINT;
  Response_Id : USINT;
  Axis : USINT;
  Number_Of_Parameter : USINT;
  END_STRUCT;

  // declare struct of parameter address response
  Parameter_Address_Response : STRUCT
  Format : USINT;
  Number_Of_Elements : USINT;
  Value_Or_Error_Value : DWORD; // dependent on format
  END_STRUCT

  // declare struct of response
  Response : STRUCT
  Header : Header_Type_Response;
  ParameterAdress : Parameter_Address_Response;
  END_STRUCT;
  END_TYPE
  // declare global variables
  VAR_GLOBAL
  // declare variable, that represents the dataset 47 request
  myRequest : Request;
  // declare variable, that represents the dataset 47 response
  myResponse : Response;
  // declare variable, that returns a value after calling _writeRecord
  myRetDINT : DINT;
  // declare variable, that returns a struct after calling _readRecord
  myRetstructretreadrecord : StructRetReadRecord;
  // declare array of byte,
  // which helps to create the request/response
  // with marshalling function
 bytearray : ARRAY[0..239] OF BYTE;
 // declare array of USINT,
 // because the systemfunctions _writeRecord and _readRecord
 // use this array
 usintarray : ARRAY[0..239] OF USINT;
 // declare command ids
 id_write, id_read : commandidtype;
 // declare the variable, to control step by step execution
 // start cycle with setting to 0 by user
 program_step : USINT := 3; // initially idle;
 END_VAR
END_INTERFACE
```

Implementation

```
// ====================================================================
IMPLEMENTATION
PROGRAM record
CASE program_step OF
// initialize -------------------------------------------------------
0:
 // get command ids for calling system functions
 id_write := _getcommandid();
 id_read := _getcommandid();
 // header from the request
 // here: Axis-No / DO-ID is 3
 // read Parameter 945 (drive fault code)
 myRequest.Header.Request_Reference := 16#10; // arbitrary no.
 myRequest.Header.Request_Id := 16#1; // read request
 myRequest.Header.Axis := 16#3; // axis no 3
 myRequest.Header.Number_Of_Parameter := 16#1; // one parameter

 // parameter address from the request
 myRequest.ParameterAddress.Attribute := 16#10; // read value
 myRequest.ParameterAddress.Number_Of_Elements := 16#1; // one index
 myRequest.ParameterAddress.Parameter_Number := 945; // parameter no.
 myRequest.ParameterAddress.SubIndex := 0;

 // convert myRequest to a BIBBYTEARRAY to use the marshalling functions
 // two step conversion from user defined data type
 // to usintarray type required by system functions
 bytearray := ANYTYPE_TO_BIGBYTEARRAY(myRequest,0);
 usintarray := BIGBYTEARRAY_TO_ANYTYPE(bytearray,0);

 // next step
 program_step := 1;

// execute _writeRecord ----------------------------------------------
1:
 // the systemfunctions _writeRecord and _readRecord
 // have to be called in sequence.
 // the functions occur always as pair.
 // call systemfunction _writeRecord to send the request
 myRetDINT := _writerecord(
    ioid := INPUT,
    logaddress := 256, // io address
    recordnumber := 47, // data set 47 for DPV1
    offset := 0,
    datalength := 240,
    data := usintarray, //
    nextcommand := IMMEDIATELY, // use asynchronous
    commandid := id_write // use known commandID
 );
 // check the return value
 // keep calling until _writeRecord ready
 IF(myRetDINT = 0)THEN
    // next step
    program_step := 2;
 END_IF;
// wait for requested data -----------------------------------------------
// execute _readRecord
2:
 // call systemfunction _readRecord to receive the data
 myRetstructretreadrecord := _readrecord(
    ioid := INPUT,
    logaddress := 256, // io address
    recordnumber := 47, // data set 47 for DPV1
    offset := 0,
    datalength := 240,
    nextcommand := IMMEDIATELY, // use asynchronous
    commandid := id_read // use known commandID
 );
 // check the return value
 // keep calling until _readRecord ready
    IF(myRetstructretreadrecord.functionresult = 0)THEN
        // next step
        program_step := 3; // --> done
        // get data
        // two step conversion into user defined data type
        // from usintarray type given by system functions
        bytearray :=ANYTYPE_TO_BIGBYTEARRAY(
        myRetstructretreadrecord.data,0);
        myResponse := BIGBYTEARRAY_TO_ANYTYPE(bytearray,0);
        // received data can now be read from myResponse...
    END_IF;
END_CASE;
END_PROGRAM
END_IMPLEMENTATION
```

4 PROFIsafe

4.1 Communication relationships for drive-based safety

Description

The drive-based safety functions in the drive can be controlled either by using safe terminals directly on the drive or from a fail-safe control (F control) via PROFIBUS/PROFINET.

The control signals for the drive-based safety functions, as well as the feedback relating to the safety function status, are safety-oriented and must, therefore, be transmitted via a communication channel that is secured by means of a PROFIsafe protocol. The figure below contains a diagram providing a general overview of how interaction between the various control and drive processes works, as well as the communication relationships between them that are required for this purpose.

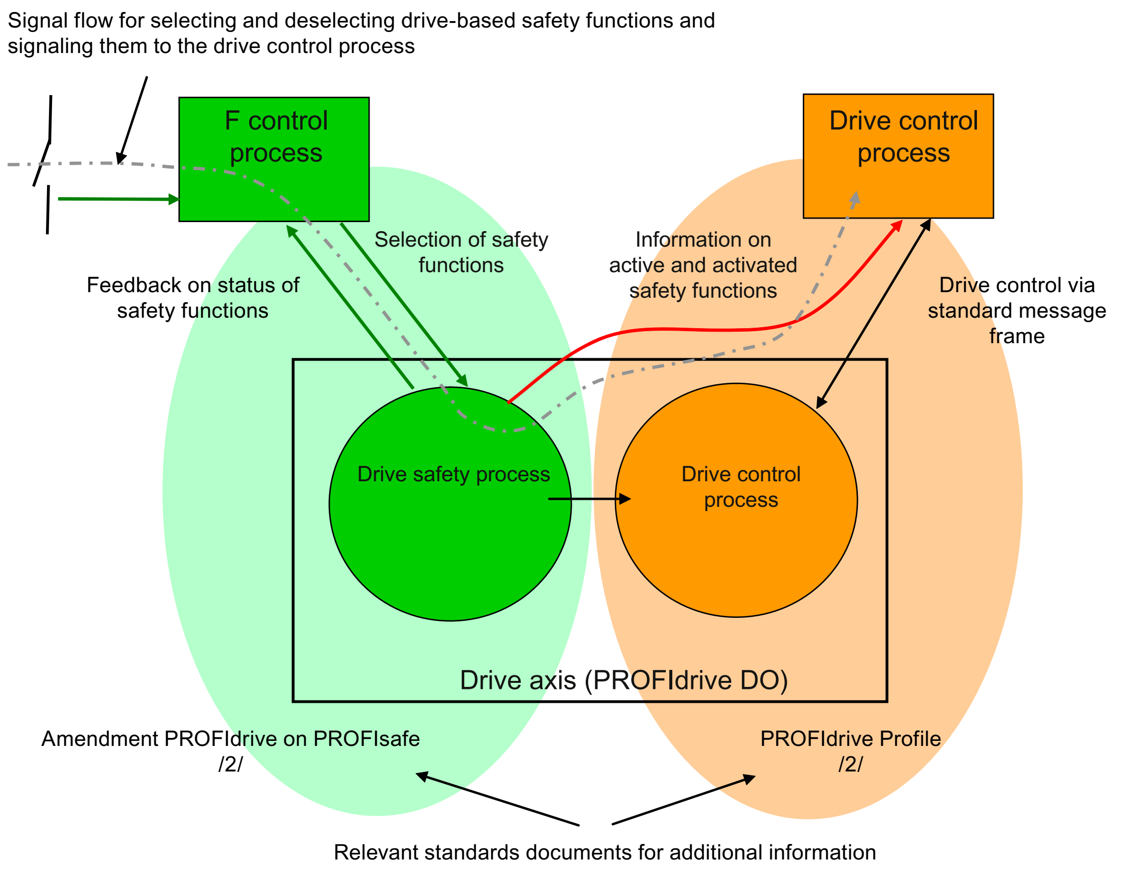

Signal flow for selecting and deselecting drive-based safety functions and signaling them to the drive control process
F control process
Drive control process
Selection of safety functions
Information on active and activated safety functions
Feedback on status of safety functions
Drive control via standard message frame
Drive safety process
Drive control process
Drive axis (PROFIdrive DO)
Amendment PROFIdrive on PROFIsafe /2/
PROFIdrive Profile /2/
Relevant standards documents for additional information

Figure 4-1 Communication relationships for drive-based safety

The "drive safety process" interfaces to the F control and drive control are PROFIdrive interfaces; their functions are defined in /1/ and /2/.

The F control introduces and monitors a drive-based safety function via the PROFIsafe-secured transmission channel between the F control and the drive (drive axis). The respective statuses of the drive-based safety functions in the drive also have repercussions on the drive interface to the drive control, since the drive priority switches between the drive control and drive safety process in the case of some drive-based safety functions.

In order for the F control and drive control to be coordinated effectively, therefore, an information channel from the "drive safety process" to the drive control is also required so that the drive control can respond to the required or activated drive-based safety functions accordingly.

4.2 Message frames and signals in drive-based safety

Description

Since Simotion does not have a safe logic function, it cannot attend to the F control process. Instead, this is carried out using a second controller featuring F functionality (usually a SIMATIC F-CPU). The image below shows how this setup works, using the example of a Sinamics axis. A PROFIsafe-secured communication channel runs between the DO drive and the SIMATIC F-CPU. Standard message frame 30, consisting of a safety control word and status word, is normally available for this communication channel. With the user program on the F-CPU, the safety control word is used to select or deselect the configured drive-based safety functions in the drive (drive safety process). The feedback from the active safety functions is sent in the input data to the F-CPU, via the safety status word. A safe process in the drive is used to send the feedback from the active safety functions via a secured communication channel; therefore, the feedback may be used for activating protection zones and doors via the F-CPU.

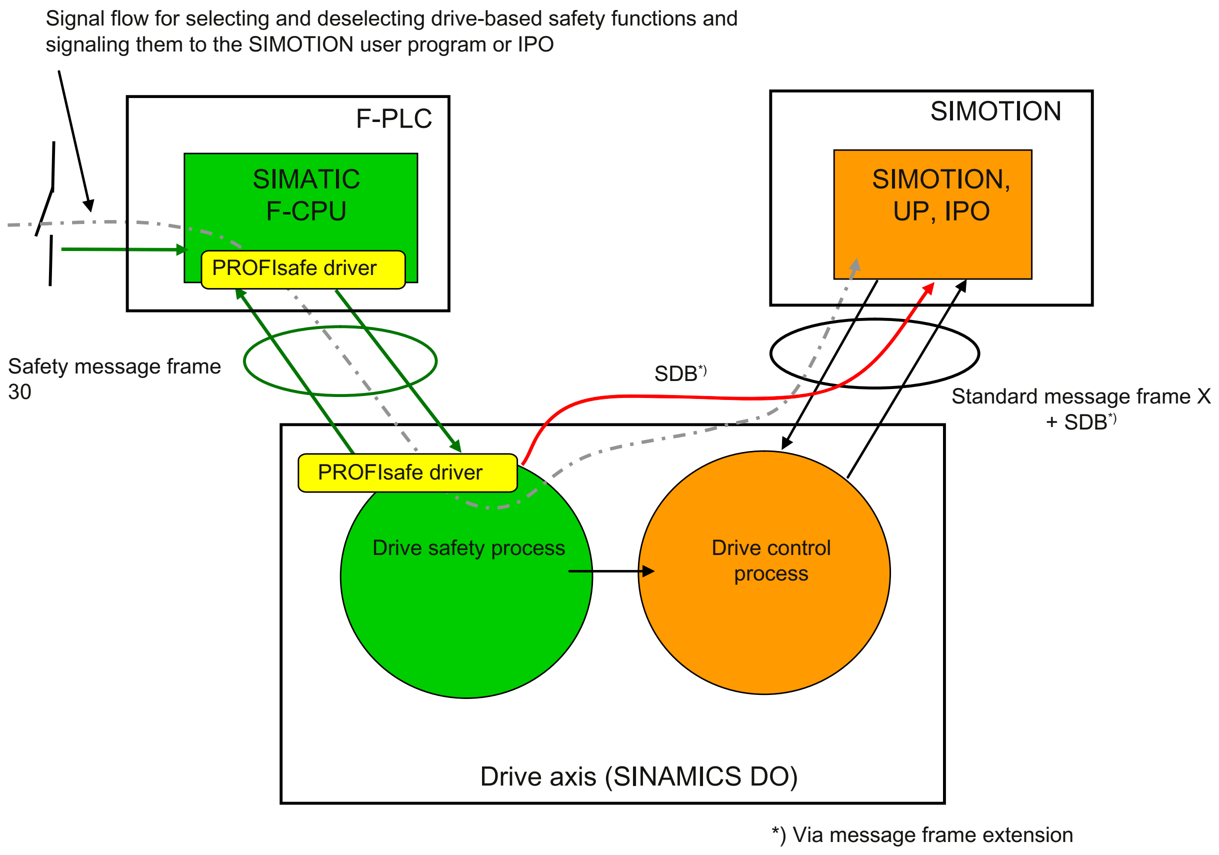

Signal flow for selecting and deselecting drive-based safety functions and signaling them to the SIMOTION user program or IPO
F-PLC
SIMATIC F-CPU
PROFIsafe driver
SIMOTION
SIMOTION, UP, IPO
Safety message frame 30
SDB*)
Standard message frame X + SDB*)
PROFIsafe driver
Drive safety process
Drive control process
Drive axis (SINAMICS DO)
*) Via message frame extension

Figure 4-2 Message frames and signals in drive-based safety

In addition to controlling safety functions in the drive via the PROFIsafe channel, you have the option of transmitting statuses and activation statuses of the drive-based safety functions from the drive to SIMOTION via a safety information channel (SDB). This information channel is not secured and is put into effect in practice by extending the standard message frame to include the system data block (SDB). The purpose of the safety information channel is to provide the option of integrating the drive motion control (IPO, SERVO) and the entire user program (UP) into the higher-priority execution of both drive-based safety functions and the F-control user program. Typical Simotion responses include:

- Recognition of safety-related, autonomous handling of the drive (e.g. braking ramp for SS1 and SS2, and switching to follow-up mode)
- Recognition that a safety function has been selected, and the Simotion response associated with this for the purpose of introducing the safety function (e.g. control-based braking ramp with reduction in velocity for SOS and SLS)

4.3 SIMOTION F proxy functions

Description

Simotion features integrated F proxy functionality for the purpose of PROFIsafe connection of integrated Simotion D drives, as well as SINAMICS drives that are controlled by Simotion but are in a different communication domain from the F-CPU. The F proxy functionality enables transparent routing of IO data from the Simotion I slave or I device interface to the respective Simotion master or controller interface on which the drive is configured. As the figure below illustrates, the path of communication is secured by the PROFIsafe drivers in the F-CPU and in the drive as a whole.

In order to use F proxy functionality, the two paths of communication - from the F-CPU to Simotion and from Simotion to the drive - need to be configured separately.

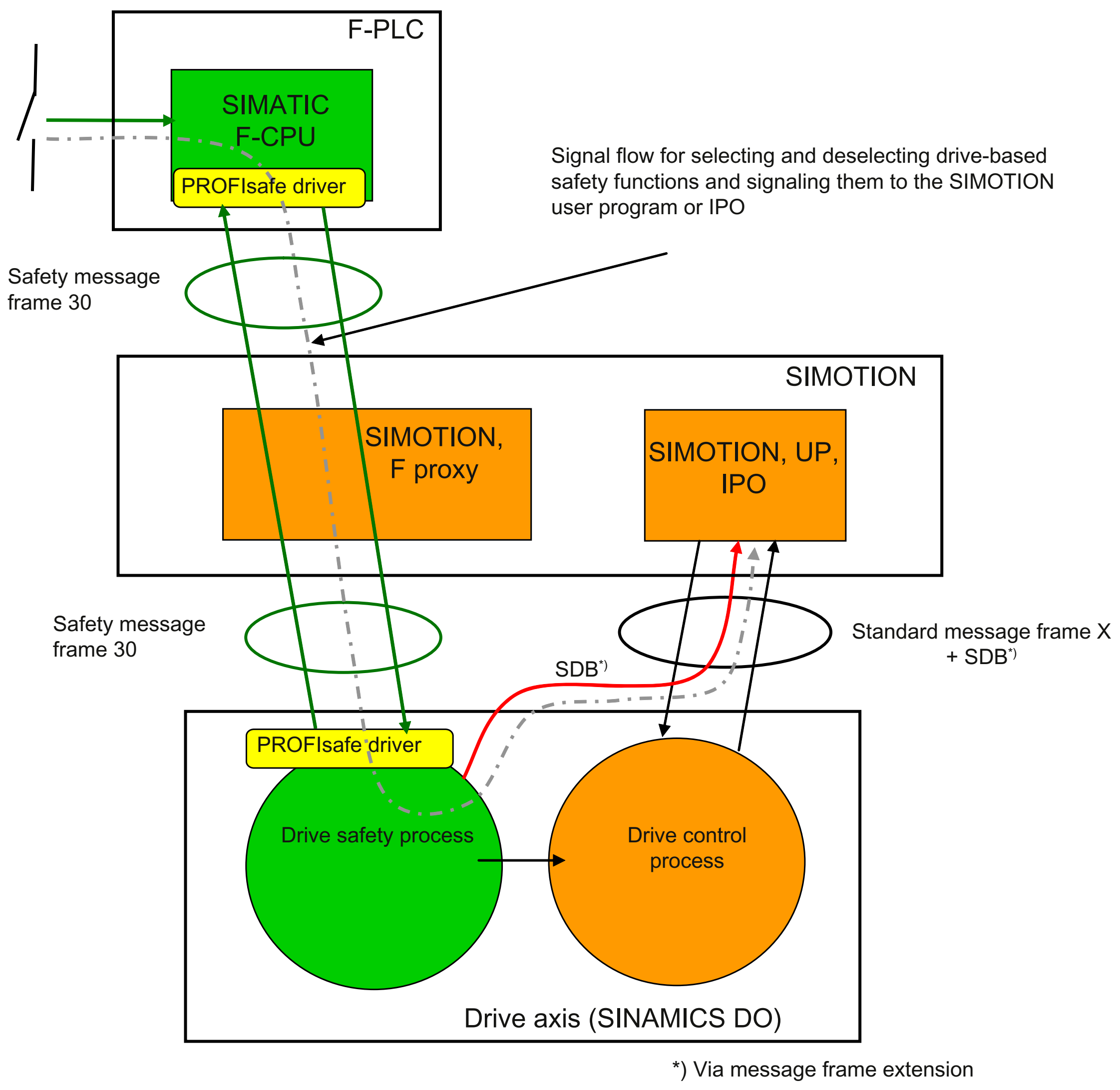


Figure 4-3 Routing the PROFIsafe channel with F proxy functionality

4.4 Additional information on SIMOTION and PROFIsafe

Description

Additional information on the subject of PROFIsafe is available in the following documents:

- For information on how to connect an axis to a SINAMICS drive with Safety Integrated, please refer to the TO Axis / External Encoder Function Manual.
- For information on how to configure a SINAMICS S120 drive or SINAMICS S110 drive with Safety Integrated, please refer to the following:
  – *SINAMICS S120 Function Manual*
  – *SINAMICS S120 Safety Integrated Function Manual*
  – *SINAMICS S110 Function Manual*.

PROFIBUS 5

5.1 PROFIBUS communication

5.1.1 PROFIBUS communication (overview)

Description

PROFIBUS DP (Decentralized Peripherals) is designed for fast data exchange at the field level. The communication is performed in a class 1 PROFIBUS master (e.g. a SIMOTION controller) and PROFIBUS slaves (e.g. a SINAMICS S120 drive). The data exchange with decentralized devices is mainly performed cyclically (DP V0 communication). In this case, the central controller (SIMOTION controller) reads the input information cyclically from the slaves and writes the output information cyclically to the slaves. Moreover, diagnostics functions are made available through the cyclic services. The following figure shows the data protocol on PROFIBUS DP.

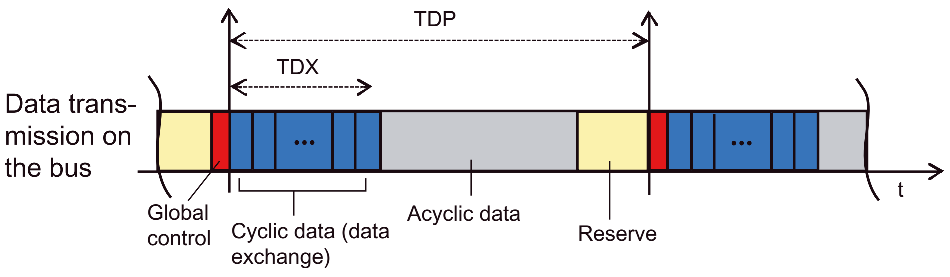


Figure 5-1 Data protocol on Profibus

5.2 Communication with SIMATIC S7

5.2.1 Possible communication connections between SIMOTION and SIMATIC

The following section describes how a SIMOTION and a SIMATIC S7 device can communicate with one another via PROFIBUS.

There are various possibilities:

- A SIMOTION device is connected as DP slave to a DP master system of a SIMATIC S7.
- A SIMATIC S7 device is connected as DP slave to a DP master system of a SIMOTION.
- A master-master communication is used between SIMOTION and SIMATIC S7.

There are two additional variants for the connection as DP slave:

- Connection as standard slave by means of a GSD file.
- Connection as intelligent DP slave (i-slave).

An i-slave is a station that has a separate intelligence, and whose range of functions as DP slave is specified through dedicated programming.

This means that these stations have to be completely configured first with respect to their communication structure, before they can be used as i-slave.

The available i-slaves can be found in the HW catalog of HW Config in the "Already configured stations" folder.

Difference: "Normal" DP slave (standard slave) - intelligent DP slave (I slave)

With a "normal" DP slave such as a compact (ET 200eco) or modular (ET 200M) DP slave, the DP master accesses the distributed inputs/outputs.

With an intelligent DP slave, the DP master does not access the connected inputs/outputs of the intelligent DP slave, but accesses instead a transfer area in the input/output address area of the "preprocessing CPU". The user program of the preprocessing CPU must handle the data exchange between the operand area and inputs/outputs.

Note

The configured I/O areas for the data exchange between the master and slaves must not be "occupied" by I/O modules.

5.2.2 SIMOTION as DP slave on a SIMATIC S7

5.2.2.1 Introduction

The following section describes how a SIMOTION device can be connected as PROFIBUS DP slave to a PROFIBUS network.

There are two possibilities:

- The SIMOTION device is connected as standard slave to a DP master system by means of a GSD file.
- The SIMOTION device is integrated as so-called intelligent DP slave (i-slave) in the DP master system.

5.2.2.2 Connecting SIMOTION as DP slave with the aid of a GSD file to a SIMATIC S7

Proceed as follows

The GSD files for the various SIMOTION platforms must first be imported into STEP 7 HW Config.

You will find the corresponding GSD files on the SIMOTION SCOUT CD "Add-on" in the respective device directory under Firmware and Version.

Table 5- 1 GSD file

Device	Name of the GSD file
SIMOTION C	Si0380aa.gsd
SIMOTION D4xx	Si0180ab.gsd (This file can be used for all SIMOTION D 4xx)
SIMOTION P	Si0280fa.gsd

After these GSD files have been imported from the Options - Install GSD file menu into the STEP7 HW Config, the devices appear in the HW catalog under Additional field devices - PLC - SIMATIC - SIMOTION and can be inserted from there into a DP master system of a S7 station.

Note

SIMOTION devices that have been connected to a SIMATIC S7 by means of a GSD file, cannot be accessed with SIMOTION SCOUT via a routed connection.
The name of the GSD file depends on the version, e.g. S10180AA and S10280AA.

Note

Through a network node it is also possible to route to drives that have been inserted as single drives.

It is thus also possible to route to SIEMENS drives that have been configured in SCOUT/STARTER, if these are configured as GSD slave / GSDML device in HW Config. However, the limitation that a network transition point can be set using the subnet ID, by setting the online access parameter (**Target device->Online access**), applies.

Moreover, the GSD file is named according to the version.

5.2.2.3 Connecting SIMOTION as i-slave to a SIMATIC S7

Requirement

- SIMOTION SCOUT and thus STEP 7 must have been installed on the engineering PC.
- The SIMATIC S7 and the SIMOTION station must be in the same project.

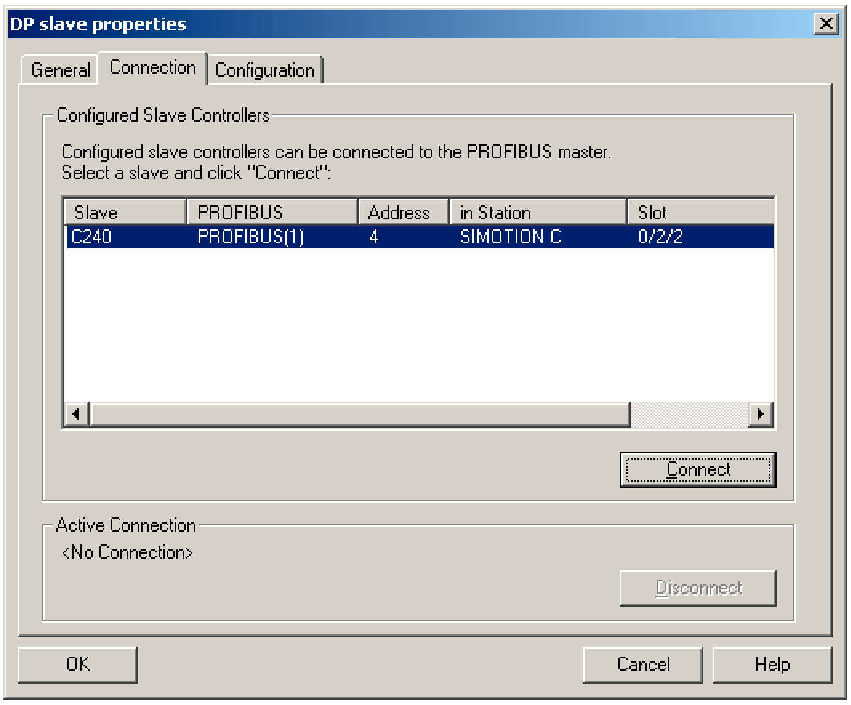
If these requirements are fulfilled, the SIMOTION can also be connected as i-slave to the PROFIBUS DP network of the SIMATIC.

Proceed as follows

It is recommended that the SIMOTION station is first completely configured as DP slave before it is placed as slave on the DP line of the SIMATIC CPU.

The following is a description of the procedure for a SIMOTION C. The procedure is identical apart from the selection of the SIMOTION platform.

1. Configuring a station as DP slave, e.g. SIMOTION C-2xx
   Double-click the desired interface (e.g. DP2/MPI) in the configuration table and select the DP slave option in the Operating mode tab.
2. Configuring the local I/O addresses
   You can set the local I/O addresses and the diagnostics address in the **Configuration** tab.
3. Switch to the configured SIMATIC station that is to be DP master for the SIMOTION.
4. Creating an iSlave
   Drag the station type "C2xx/P350/D4xx i-slave" from the Hardware catalog window (folder of already configured stations) and drop it on the symbol for the DP master system of the SIMATIC station.
5. Specifying the intelligent DP slave
   Double-click the symbol for the intelligent SIMOTION DP slave and select the **Link** tab. In this tab, assign the station that is to represent the intelligent DP slave. This dialog box displays all the stations that are already available in the project and that are potential link partners.


DP slave properties

General | Connection | Configuration

Configured Slave Controllers

Configured slave controllers can be connected to the PROFIBUS master.
Select a slave and click "Connect":

Slave	PROFIBUS	Address	in Station	Slot
C240	PROFIBUS(1)	4	SIMOTION C	0/2/2

Connect

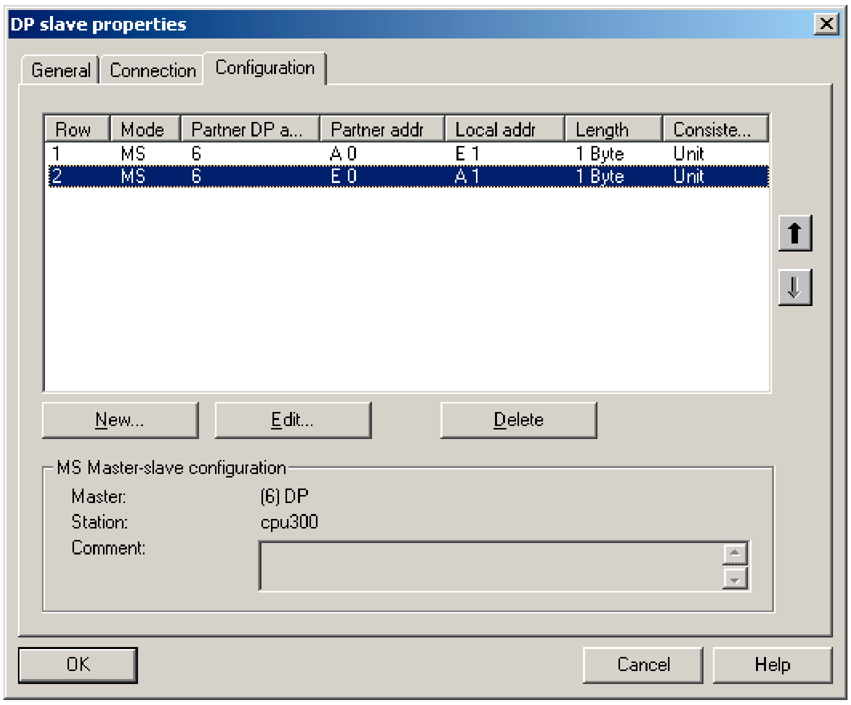
Active Connection
<No Connection>

Disconnect

OK | Cancel | Help

Figure 5-2 DP slave properties

6. Select the appropriate SIMOTION and click **Connect**. The configured SIMOTION station is now connected as intelligent DP slave to the SIMATIC.
7. Select the **Configuration** tab and assign the addresses:



Row	Mode	Partner DP a...	Partner addr	Local addr	Length	Consiste...
1	MS	6	A 0	E 1	1 Byte	Unit
2	MS	6	E 0	A 1	1 Byte	Unit

MS Master-slave configuration

Master:	(6) DP
Station:	cpu300
Comment:	

Figure 5-3 Properties - configuration

- For the data exchange with the DP master via I/O areas, select the **MS** (Master-Slave) mode
- For the direct data exchange with a DP slave or DP master, select the **DX** (Direct Data Exchange) mode

1. Confirm the settings by clicking **OK**.

The configuration of the SIMOTION station as intelligent DP slave on the SIMATIC station is now completed and data can be exchanged via the specified I/O addresses.

5.2.3 SIMATIC S7 as DP slave on a SIMOTION

5.2.3.1 Introduction

The following section describes how a SIMATIC station can be connected as PROFIBUS DP slave to a PROFIBUS network.

There are two possibilities:

- The SIMATIC station is connected as standard slave to the DP master system of a SIMOTION by means of a GSD file.
- The SIMATIC station is integrated as a so-called i-slave in the DP master system of a SIMOTION.

5.2.3.2 Connecting SIMATIC as DP slave with the aid of a GSD file to a SIMOTION device

Procedure

The GSD files for the various SIMATIC stations must first be imported into STEP 7 HW Config.

You will find the corresponding GSD files in Product Support under:
http://support.automation.siemens.com/ww/view/en/113653.

After these GSD files have been imported from the Options - Install GSD file menu into the STEP7 HW Config, the devices appear in the HW catalog under Additional field devices - PLC - SIMATIC and can be inserted from there into a DP master system of a SIMOTION station.

SIMATIC S7 devices that have been connected to a SIMOTION by means of a GSD file, cannot be accessed with STEP 7 via a routed connection.

5.2.3.3 Connecting SIMATIC S7 CPU as i-slave to a SIMOTION device

Prerequisites

- SIMOTION SCOUT and thus SIMATIC STEP 7 have been installed on the engineering PC.
- The SIMATIC S7 and the SIMOTION station must be in the same project.

If these requirements are fulfilled, the SIMATIC can also be connected as i-slave to the PROFIBUS DP network of the SIMOTION.

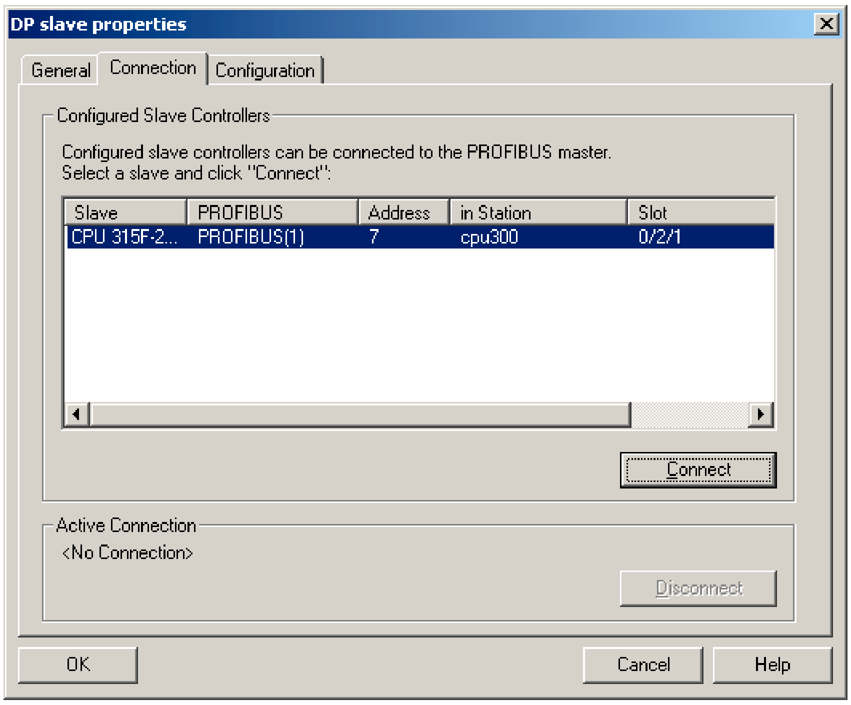
Proceed as follows

It is recommended that the SIMATIC station is first completely configured as DP slave before it is placed as slave on the DP line of the SIMOTION.

The following is a description of the procedure for a CPU 315-2 D. The procedure is identical apart from the selection of the CPU types, also for an S7-400.

1. Configure a station, e.g. with the CPU 315-2 DP as DP slave. Double-click on line 2.1 (interface) in the configuration table and select the DP slave option in the **Operating mode** tab.
2. You can set the local I/O addresses and the diagnostics address in the **Configuration** tab.
3. Switch to the configured SIMOTION station that is to be DP master for the SIMATIC.
4. Drag the appropriate station type, CPU 31x or CPU 41x, from the Hardware catalog window (folder of already configured stations) and drop it on the symbol for the DP master system of the SIMOTION station.
5. Double-click the icon for the intelligent SIMOTION DP slave and select the Link tab. In this tab, assign the station that is to represent the intelligent DP slave. This dialog box

displays all the stations that are already available in the project and that are potential link partners.


DP slave properties

General | Connection | Configuration

Configured Slave Controllers

Configured slave controllers can be connected to the PROFIBUS master.
Select a slave and click "Connect":

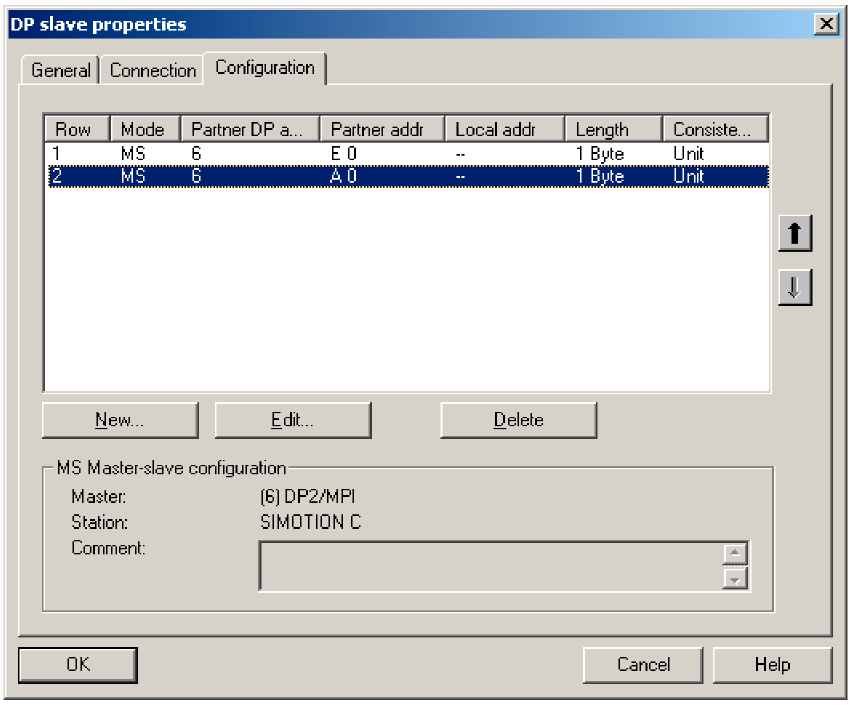
Slave	PROFIBUS	Address	in Station	Slot
CPU 315F-2...	PROFIBUS(1)	7	cpu300	0/2/1

Active Connection
<No Connection>

Figure 5-4 Properties - link

6. Select the appropriate S7 station and click **Connect**. The configured S7 station is now connected as intelligent DP slave to the SIMOTION.

7. Select the **Configuration** tab and assign the addresses:


DP slave properties

General | Connection | Configuration

Row	Mode	Partner DP a...	Partner addr	Local addr	Length	Consiste...
1	MS	6	E 0	--	1 Byte	Unit
2	MS	6	A 0	--	1 Byte	Unit

New... | Edit... | Delete

MS Master-slave configuration

Master: (6) DP2/MPI
Station: SIMOTION C
Comment:

OK | Cancel | Help

Figure 5-5 Configuration - address selection

– For the data exchange with the DP master via I/O areas, select the **MS** (Master-Slave) mode

– For the direct data exchange with a DP slave or DP master, select the **DX** (Direct Data Exchange) mode

8. Confirm the settings by clicking **OK**.

The configuration of the SIMATIC station as intelligent DP slave on the SIMOTION station is now completed and data can be exchanged via the specified I/O addresses.

5.2.4 PROFIBUS master-master connection between SIMATIC and SIMOTION

5.2.4.1 Introduction

Master-master communication

A master-master communication connection between a SIMATIC S7 and a SIMOTION device via PROFIBUS is created using the SFC65 (XSEND) and SFC66 (XRECEIVE) system functions on the SIMATIC side and the _Xsend and _Xreceive system functions on the SIMOTION side. It is not necessary to configure a communication connection in NetPro.

Table 5- 2 Master-master communication

Log	SIMATIC device	Function	SIMOTION device	Function
PROFIBUS	S7-300 CPU S7-400 CPU	SFC65 (XSEND) SFC66 (XRCV)	C2xx D4xx P350	_Xsend _Xreceive

The PROFIBUS addresses are assigned in HW Config. All further block parameters are specified for the connection by the user and also transferred when the function is called. The PROFIBUS connection between SIMATIC and SIMOTION is therefore similar to a TCP/IP connection between a SIMATIC station with integrated Ethernet interface and a SIMOTION device and vice versa. The parameters important for the communication are specified by the user and transferred with the block or function call.

The following section describes the parameterization of the system functions on the SIMATIC S7 side and the functions on the SIMOTION side in more detail.

5.2.4.2 SIMATIC S7 system functions for a PROFIBUS connection

Introduction

The PROFIBUS connection between a SIMATIC S7 station and a SIMOTION device was introduced in the previous section. The following contains a detailed explanation of the parameterization of the SIMATIC S7 system functions and the SIMOTION functions for a PROFIBUS connection.

SIMATIC S7 system functions

On the SIMATIC S7 side, the two system functions SFC65 X_SEND and SFC66 X_RCV are used for the communication between a SIMATIC S7 station and a SIMOTION device. SIMOTION functions:

```
CALL "X_SEND"
    REQ:=M1.0
    CONT:=FALSE            //This is the DP address of the
    DEST_ID:=W#16#2        //communication partner (SIMOTION P350)
    REQ_ID:=DW#16#2        //The REQ_ID must match the MessageID on
    SD:=P#DB100.0DBX0.0 BYTE 10 //the SIMOTION receive side!
    //
    RET_VAL:=MW64
    BUSY:=M1.1
```

Parameterization of the SFC65 X_SEND system function

The SFC65 X_SEND system function is called on the SIMATIC S7 side to send data via a PROFIBUS connection from a SIMATIC S7 station to a SIMOTION device.

The data transfer is controlled via the REQ parameter, i.e. when the parameter is set to 1, the data transfer is started. If there is no connection to the communication partner at this time, it is established before the data is sent.

The CONT parameter is used for the parameterization of the connection behavior after completion of the data transfer. If value 1 is entered in the CONT parameter, the connection is maintained after completion of the data transfer. If 0 is entered as value, the connection is cleared after the data transfer.

The DEST_ID parameter contains the PROFIBUS address of the SIMOTION device. It is specified in STEP 7 HW Config.

REQ_ID identifies the send data, i.e. the sent data can be uniquely assigned to the S7 station in the SIMOTION device via the value in the REQ_ID parameter. The value assigned here is confirmed in the messageid parameter in the receive function on the SIMOTION side.

SD specifies the area from where the send data originates.

The RET_VAL and BUSY parameters are used to monitor the status of the transmission process. BUSY indicates that the send job is still running or has already been completely executed. RET_VAL can be used for a detailed diagnosis when an error occurs.

```
CALL "X_RCV"
    EN_DT:=M0.0
    RET_VAL:=MW50
    REQ_ID:=MD52
    NDA:=M0.1
    RD:=P#DB110.DBX0.0 BYTE 10
```

Call example of the system function SFC66 X_RCV

If data from a SIMOTION device is to be received on a SIMATIC S7 station, the SFC66 X_RCV system function must be called in the S7 program.

The "EN_DT" input of the system function specifies:

- Whether the function should only check if new data is received (EN_DT=0) or
- Whether the received data should be copied from the queue to the area specified by "RD" (EN_DT=1).

The user can monitor the status of the function call with the RET_VAL parameter. If an error occurs, the user receives detailed information on the cause.

REQ_ID identifies the receive data, i.e. the received data can be uniquely assigned to a SIMOTION device via the REQ_ID parameter. The value received here corresponds to the value in the messageid parameter in the relevant send function on the SIMOTION side.

The NDA parameter indicates whether new data has been received. If NDA is 1, new data is available and can be transferred to the receive data area. If NDA is 0, no new data is available.

The RD parameter specifies where the received data is stored.

SIMOTION functions

```
RetVal_PB_Senden:=
    _xsend(PB_Senden_CommunicationMode, PB_Senden_Address,
    PB_Senden_MessageID, PB_Sender_NextCommand, PB_Senden_CommandID,
    PB_Sende_Daten, PB_Sende_Daten_Laenge);
```

Example for calling the SIMOTION _xsend function

If the SIMATIC S7 station and the SIMOTION device communicate via PROFIBUS, the _xsend function is called on the SIMOTION side for the transmission.

The "communicationmode" parameter informs the called function of what is to happen to the connection after the successful data transfer. The function data type can assign the ABORT_CONNECTION or HOLD_CONNECTION values. If ABORT_CONNECTION is assigned to the parameter, the connection will be removed after the data transfer. The HOLD_CONNECTION value is used to parameterize the function so that the connection will be retained after a successful data transfer.

The address parameter contains a structure of the StructXsendDestAddr data type, which also consists of various parameters. This structure contains all the information about the communication partner address of the SIMOTION device.

Parameter structure "StructXsendDestAddr

The individual parameters of the structure are listed and explained in the following.

The deviceid parameter is used for the respective SIMOTION hardware. The physical connection point is specified with the parameter. The value 1 is entered for interface X8 for a SIMOTION C2xx. The value 2 is entered for interface X9. If a SIMOTION P350 is connected to the SIMATIC S7 station on X101, the value 1 is assigned in the deviceid parameter. The value 2 is written in the deviceid parameter for the X102 interface. For the SIMOTION D4x5, the value 1 is entered for the X126 interface and the value 2 for the X136 interface in the deviceid parameter.

Because no subnet mask is specified for the communication via MPI or PROFIBUS, the value 0 is preassigned to the remotesubnetidlength parameter. Consequently, the assignment of the remotesubnetid parameter is irrelevant.

The value 1 is set in the remotestaddrlength parameter for the MPI or PROFIBUS communication.

The nextstaddrlength parameter specifies the length of the router address. As a router is not used for the MPI or PROFIBUS communication between the SIMATIC S7 station and the SIMOTION device, the value 0 is assigned for this parameter. Consequently, the nextstaddr parameter is also irrelevant (see below).

The following remotesubnetid parameter identifies the subnet mask and has, as already mentioned above, no significance for the communication via MPI or PROFIBUS.

The remotestaddr parameter specifies the actual destination address. The parameter is an array. However, only the first index is used for the MPI or PROFIBUS communication. The other five indices have no significance.

The nextstaddr parameter is used to specify the router address. The same applies for this parameter as for the remotesubnetid parameter. Its assignment is also irrelevant for the communication via MPI or PROFIBUS.

The messageid parameter is assigned by the user for the identification of the SIMOTION on the receive side. The value entered enables an assignment on the SIMATIC S7 station via the REQ_ID parameter. The value can be fetched there from the messageid parameter.

The behavior of this function with respect to the advance when called is parameterized with the nextcommand parameter. There are two setting options: IMMEDIATELY and WHEN_COMMAND_DONE. With the first value, the advance is immediately and with the second value, after completion of the command.

When the function is called, a system-wide unique number is assigned in the commandid parameter to allow tracking of the command status.

The send data is specified with the data variable when the function is called.

The datalength parameter specifies the length of the data to be transferred from the send area.

The return value of the _xsend function to the user program is of data type DINT. The various return values indicate any problems that occurred during the execution of the function. There is also a confirmation when the data has been successfully sent.

```
RetVal_PB_Empfanen:=
    _xreceive(PB_Empfangen_MessageID,
    PB_Empfangen_NextCommand,PB_Empfangen_CommandID);
```

Call example of the SIMOTION _xreceive function

The example shows the use of the _xreceive function. The function is used when data from a SIMATIC S7 station is to be received via PROFIBUS.

The messageid parameter is transferred to the _xreceive function for the identification of the S7 station from which the data is to be received. The entered value is that what was assigned on the S7 page in the REQ_ID parameter of the corresponding _xsend system function.

The behavior of this function with respect to the advance when called is parameterized with the nextcommand parameter. There are two setting options: IMMEDIATELY and WHEN_COMMAND_DONE. With the first value, the advance is immediately and with the second value, after completion of the command.

When the function is called, a system-wide unique number is assigned in the commandid parameter to allow tracking of the command status.

The structure returned from the function to the user program contains the functionresult, datalength and data parameters. The receive status can be queried via the functionresult parameter. The datalength parameter returns the number of received user data bytes after a successful call of the _xreceive function. The received user data can be accessed via the data parameter.

Ethernet introduction (TCP/IP and UDP connections) 6

6.1 Introduction

The following section describes how open TCP/IP and UDP Ethernet connections can be set up between a SIMOTION device and a SIMATIC S7 device.

All the necessary steps that have to be prepared and the required function calls are explained using a programming example.

6.2 Configuring Ethernet subnets with SIMOTION

6.2.1 Features of the Ethernet subnets

Depending on the device, SIMOTION has one or two onboard Ethernet interfaces. You can connect an Industrial Ethernet with a transmission rate of 10/100 Mbit/s to the 8-pin RJ45 sockets.

Alternatively, you can also connect an Industrial Ethernet through the PROFINET modules, such as e.g. CBE30 of SIMOTION D4x5.

You can use a PG/PC to communicate with STEP 7, SIMOTION SCOUT, and SIMATIC NET OPC.

You can also communicate with other devices such as SIMOTION devices, SIMATIC S7 devices or PCs via TCP/IP.

There is no HUB/switch functionality, i.e. message frames are not forwarded from one interface to the other, for modules with two Ethernet interfaces. The interfaces belong to separate Ethernet subnets. The SIMOTION devices do not have an IP router functionality, they do not forward the message frames from one subnet to another.

With two interfaces, the TCP/IP timeout parameters can be set once for both interfaces. The transmission rate / duplex can be set separately for the two interfaces.

"Utilities via TCP" are supported for both Ethernet interfaces. This enables S7 routing from the Ethernet interfaces to the PROFIBUS interfaces. "Utilities via TCP" are not routed from one Ethernet interface to the other.

The MAC addresses can be seen on the outside of the housing.

Use

Industrial Ethernet can be used with SIMOTION as follows:

- For communication with STEP 7, SIMOTION SCOUT and SIMATIC NET OPC via a PG/PC
- For communication via UDP (User Datagram Protocol) with other components, e.g. other SIMOTION devices, SIMATIC devices or PCs

- For communication via TCP (Transfer Control Protocol) with other components, e.g. other SIMOTION devices, SIMATIC S7 stations or PC
- For the connection of SIMATIC HMI devices such as MP277, MP370 or PC-based HMIs
- For communication by means of SIMOTION IT DIAG and SIMOTION IT OPC XML-DA (separate license required for each)
- For communication by means of SIMOTION VM (separate license required)

6.3 Function overview and functional sequence of Ethernet communication via TCP/IP or UDP

6.3.1 Introduction

The following section describes which system or communication functions are available for a configured Ethernet communication connection, and how these functions are used in the correct execution sequence.

6.3.2 SIMOTION TCP/IP functions - modeling

The communication sequence is shown and explained based on the SIMOTION system functions in the figure **Principle communication sequence of TCP/IP communication**.

These system functions must be performed by the corresponding S7 system function blocks for communication connections with a SIMATIC S7. A corresponding comparison table is contained in **SIMATIC Functions**.

The modeling explains the individual steps shown in the sequence.

- Server waits at port (1)
- Client announces connection request at this port (2). If a port is not announced on the server, wait with TimeOut (system setting)
- Server creates internal communication port with connection announcement and releases server port for new connection. The internal communication port is identified via the connectionId (3)
- Possible to send/receive data via this connection not only from the client, but also from the server (4)
- Further connections can be established at the server port (5)

- An existing connection can be closed on the client or server side with _tcpCloseConnection (6)
- Server port to establish connection is closed with _tcpCloseServer (7)

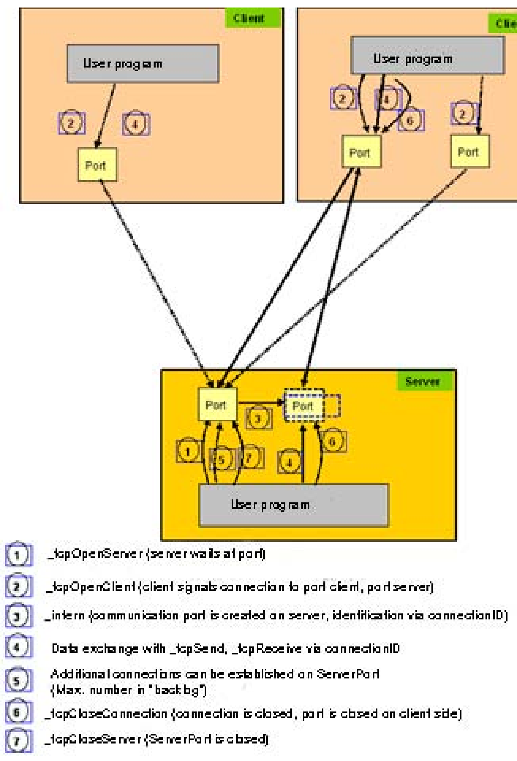

Client
User program
Port
Client
User program
Port
Port
Server
Port
Port
User program
1 _tcpOpenServer (server waits at port)
2 _tcpOpenClient (client signals connection to port client, port server)
3 _intern (communication port is created on server, identification via connectionID)
4 Data exchange with _tcpSend, _tcpReceive via connectionID
5 Additional connections can be established on ServerPort (Max. number in "backlog")
6 _tcpCloseConnection (connection is closed, port is closed on client side)
7 _tcpCloseServer (ServerPort is closed)

Figure 6-1 Principle communication sequence of TCP/IP communication

6.3.3 SIMOTION TCP/IP functions - description

The three calls of _tcpOpenClient in the modeling all refer to the same server (IP address / port). However, internally a separate port is assigned on the server. External communication is performed with the connectionId.

Port assignment:

- The port number is in the range 1024 to 65535.
- The port on the client can be the same as the port on the server.
- The port on the client can be different from the port on the server.

The sequence shows a simple example for the function execution sequence with two partners:

Table 6- 1 Communication between a sender (client) and a receiver (server)

	Function
Establish the connection • Receiver/server waits for communication request • Sender/client requests connection to be established to the receiver • Receiver/server has established communication request • No further connection is required	 _tcpOpenServer _tcpOpenClient _tcpCloseServer
Communicating • Sender sends data to the receiver • Receiver receives data from the sender	 _tcpSend _tcpReceive
Terminating communication connection • Sender no longer sends data and closes the connection	 _tcpCloseConnection

A sender or receiver can be a client as well as a server when establishing a connection. There must be at least one client and one server when establishing a TCP/IP connection.

The client-server relationship is only valid until the connection is established. After the connection has been established, both communication partners are equivalent, i.e. each of the two can send or receive or close the connection at any time.

6.3.4 SIMOTION UDP functions - modeling

Description of UDP (User Datagram Protocol)

UDP (User Datagram Protocol) makes a procedure available to send and receive data over Ethernet from the user program with a minimum of protocol mechanism. No information concerning the transferred data is returned in case of communication via UDP.

The communication takes place via ports on both the send and receive sides.

As opposed to TCP/IP, you do not need to program any connection buildup or closing.

UDP communication model

- For reception, in the command you address the port that you want to use on your component for the communication job.
- When sending data you specify the IP address of the target system, the port number for the data on the target system and the port number of your component (see above).
- You can specify whether the port should remain reserved on your end after the communication job has been executed.

- UDP is not a secured model. Therefore, data may be lost during transfer. A secured data transfer must be programmed in your application, e.g. by acknowledging the receipt of the data.
- Function _udpReceive allows you to transfer the data of a transfer protocol in the return structure, if several data protocols have been returned with _udpReceive, the "oldest" data protocol is returned.

The following figure shows the UDP communication model at the SIMOTION end

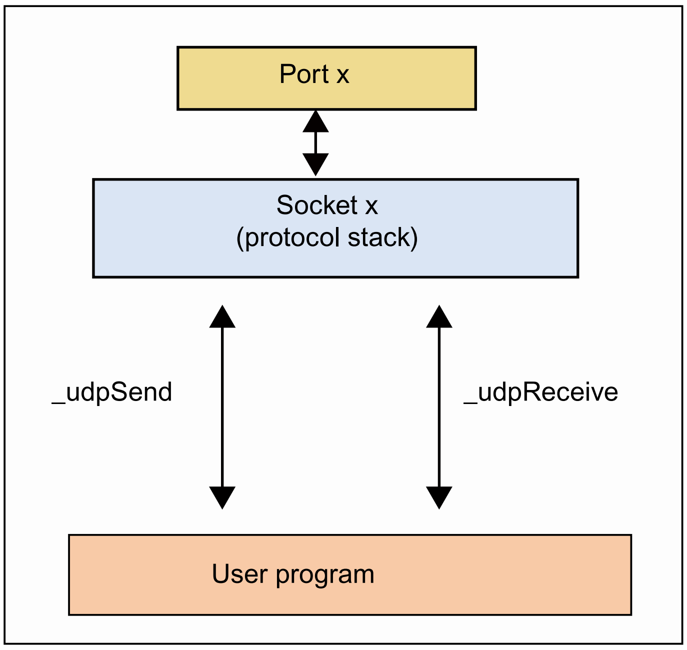


Figure 6-2 UDP communication model

See also

Function _udpSend (Page 128)

Function _udpReceive (Page 129)

UDP connection (Page 105)

6.3.5 SIMATIC functions

The following is a comparison of the SIMOTION system functions with the corresponding SIMATIC S7 functions, which are required to establish Ethernet communication between a SIMOTION and a SIMATIC S7 device.

The assignment of the communication functions to the protocols and the individual devices is specified in the following table.

Table 6- 2 Overview of the protocols, devices and communication functions

Log	SIMATIC device	Function	SIMOTION device	Function
TCP/IP	S7 300 CPU with Ethernet CP (CP343-1)	FC5 AG_SEND, FC6 AG_RECV	C2xx D4xx P350	_tcpOpenClient, _tcpSend, _tcpReceive, _tcpCloseConnecti on, _tcpOpenServer, _tcpCloseServer
	S7 300 CPU with integrated Ethernet interface	FB63 TSEND, FB64 TRCV, FB65 TCON, FB66 TDISCON, UDT65 TCON_PAR		
	S7 400 CPU	FC50 AG_LSEND, FC60 AG_LRECEIVE		
UDP	S7 300 CPU with Ethernet CP (CP343-1)	FC5 AG_SEND, FC6 AG_RECEIVE	C2xx D4xx P350	_udpSend, _udpReceive
	S7 300 CPU with integrated Ethernet interface	This protocol is not supported by CPU modules of the S7 300 series with integrated Ethernet interface!		
	S7 400 CPU	FC50 AG_LSEND, FC60 AG_LRECEIVE		
PROFIBUS	S7 300 /S7 400 CPU	SFC65 (XSEND) SFC66 (XRCV)	C2xx D4xx P350	_Xsend _Xreceive

The following applies for S7-300 with Ethernet CP:

Only the AG_SEND/AG_RECV functions are used with the current versions of the Ethernet CP; the data length can be up to 8192 bytes (see table). With older versions of the Ethernet CP, the data length is limited to <= 240 bytes per job (is valid up to block version V3.0 of AG_SEND/AG_RECV); with later versions of the Ethernet CP, longer data (up to 8192 bytes) can be transferred with the AG_LSEND or AG_LRECV function.

Therefore, it is important to know and take into consideration the version of the CP and the version of the used blocks.

The following applies for S7-400:

The AG_SEND/AG_RECV functions can also be used for the S7-400. However, the transferable data length is generally limited to <= 240 bytes per job!

Longer data records (maximum 8192 bytes; see table) can be transferred using the AG_LSEND/AG_LRECV functions. It is also important to know which data length the CP supports. This can be taken from the description of the CP.

The table provides an overview of the data volumes that can be exchanged between SIMATIC S7 and SIMOTION with the different transmission methods.

Table 6- 3 Maximum transferable length per job with the communication functions

Function	TCP/IP protocol	UDP protocol	PROFIBUS protocol
FC5 AG_SEND, FC6 AG_RECV (S7-300)	8192 bytes	2048 bytes	
FC50 AG_LSEND, FC60 AG_LRECV (S7-400)	8192 bytes	2048 bytes	
FB63 TSEND, FB64 TRCV (S7-300 CPU with integrated Ethernet interface)	1460 bytes	This protocol is not supported by CPU modules of the S7-300 series with integrated Ethernet interface!	
_tcpSend, _tcpReceive, udpSend, udpReceive (SIMOTION C2xx, D4xx, P350	4096 bytes	1,470 bytes	
SFC65 (XSEND), SFC66 (XRCV) (S7-300, S7-400)			76 bytes
_Xsend, _Xreceive (SIMOTION C2xx, D4xx, P350)			200 bytes

6.3.6 General information

Communication via Ethernet is connection-oriented, i.e. data can only be transferred when a connection has been established to the partner station.

TCP/IP communication is performed via data packets that are sent from the sender in a certain size. However, these can arrive at the receiver in various data packet sizes.

The following scenarios are possible on the receiver side:

- Subpackets:
  received data packet < sent data packet
- Several packets combined into a large data packet:
  received data packet > sent data packet

The order of the data is maintained. Users must ensure that these data packets are restored to the length of the sent data packet in their SIMOTION program. Details can be found in the appropriate configuration sections.

Ethernet communication of SIMOTION with TCP/IP or UDP is possible:

- With a SIMATIC S7 module with Ethernet connection (integrated or with extra Ethernet CP). Which SIMATIC S7 module is capable of TCP/IP can be found in the technical specifications of the respective module. The essential module types are specified in the table **Overview of the protocols, devices and communication functions** under SIMATIC Functions (Page 95).
- With a PC. An appropriate software that supports TCP/IP communication (e.g. Perl, Visual Basic or C++) is required on the PC.
- Between the SIMOTION Cxx, Dxx and P350 modules.

The TCP/IP system functions of SIMOTION may only be called in the BackgroundTask or in a MotionTask.

Useful data in the stack for data exchange

TCP and UDP use the IP protocol. For communication via TCP or UDP, the useful data received is stored in the stack of the SIMOTION CPU and must be fetched from here by the application. The size of the useful data stored is limited for each connection:

- UDP communication: 1,470 bytes of useful data
  (The useful data size per message frame is limited by the system function:
  < V4.1 SP4: 1,400 bytes
  V4.1 SP4 and higher: 1,470 bytes.)
- TCP/IP communication: 8,192 bytes of useful data

Note

Therefore, during UDP communication the application must ensure that the useful data is fetched in good time ahead of the stack discarding the new data. For example, if 11 message frames of 148 bytes each were sent, the last message frame would disappear if the application failed to fetch any of the others (i.e. in good time).

During TCP communication, the application must ensure that the useful data is fetched promptly; otherwise, it will not be possible for any other data to be received. However, by contrast none of the useful data is discarded, as TCP has a flow control facility. In this state, the communications partner does not send any additional data and alerts its application to this fact.

UDP and TCP communication

The main features of a communication connection are two end points. An end point is an ordered pair consisting of an IP address and port; generally speaking, one end point represents the server and the other represents the client. The server provides a service via a port. The client uses the service that the server provides.

Maximum number of possible TCP connections

The table below contains examples of the number of possible communication connections for a SIMOTION CPU acting as a client. The values relate to a local network without any other external load sources, and a SIMOTION D435 acting as a server. The number of possible communication connections may vary depending on the configuration, hardware, and network utilization/topology.

Table 6- 4 Communication connections in relation to a SIMOTION CPU

SIMOTION CPU (client)	Number of communication connections
C240	45
D410	45
D435	75
P350	40

Note

In order to ensure secure cyclic communication, you should have recourse to standard mechanisms such as those existing under PROFIBUS DP or PROFINET IRT in order to avoid the kinds of critical incidents that can arise with UDP or TCP communication.

6.4 Preparations for the configuration of the connection between SIMOTION and SIMATIC S7

Prerequisites

Before a TCP/IP or a UDP communication connection can be created between a SIMATIC S7 station with an Ethernet CP and a SIMOTION device, it is necessary that the SIMATIC S7 station and the SIMOTION device have been created in the same project. (Multiple projects are not supported in Version V4.1 of SIMOTION.)

Further requirements are that an Ethernet CP has been configured in the SIMATIC station and that an Ethernet network is present in the project. The two communication stations must also be connected to the network and have been assigned addresses.

The communication connection is configured in NetPro. NetPro can be accessed in several ways:

- NetPro can be started via a menu button in the SIMATIC Manager, SIMOTION SCOUT and HW Config.
- NetPro can also be opened in SIMOTION SCOUT via the menu Project -> Open NetPro or in the SIMATIC Manager via the menu Options -> Configure network.
- Another option is to open NetPro via the Connections object within an S7 CPU (see following figure). The advantage of this method is that the connection table of the appropriate SIMATIC S7 station is opened immediately.

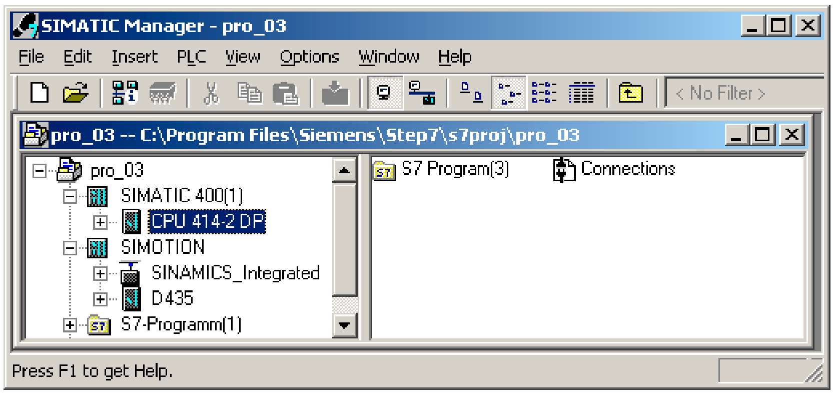


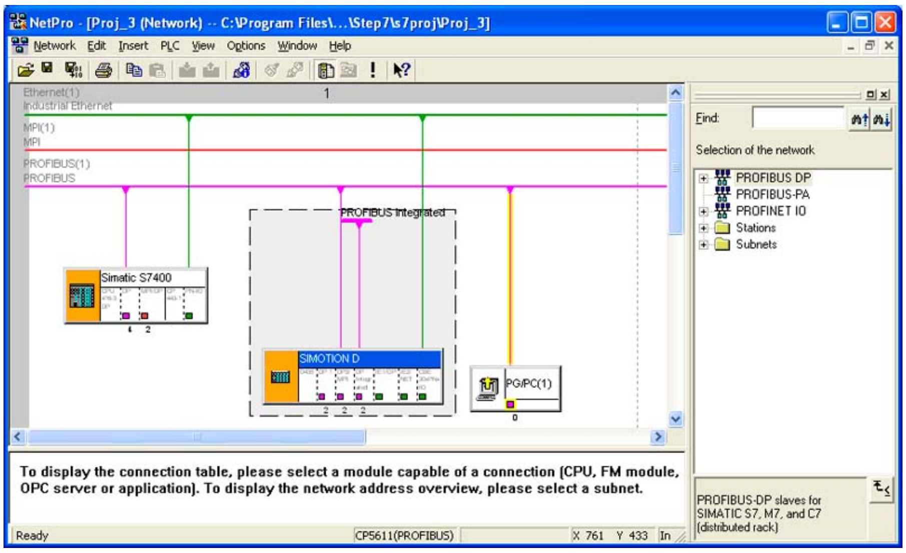
Figure 6-3 Display of the "Connections" object in the SIMATIC Manager

6.5 Configuring a communication connection between a SIMATIC with Ethernet CP and a SIMOTION device

6.5.1 Configuring a communication connection between a SIMATIC with Ethernet CP and a SIMOTION device

Proceed as follows

The connection table of the S7 station must be displayed in NetPro in order to create and configure the communication connection. To do this, the S7 CPU within the S7 station is selected. The connection table is then displayed in the lower working area of NetPro. A connection table cannot be displayed for the SIMOTION device in NetPro.



To display the connection table, please select a module capable of a connection (CPU, FM module, OPC server or application). To display the network address overview, please select a subnet.

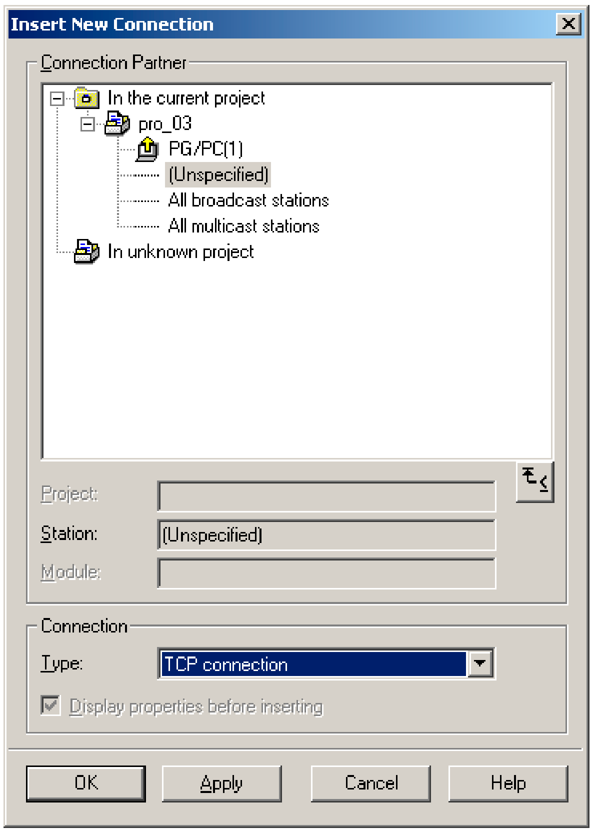
Figure 6-4 Selected S7 CPU and the associated connection table

Double-clicking an empty line in the connection table opens the **Insert new connection** dialog box for adding a new communications connection. The dialog box is the same for TCP/IP and UDP connections. The dialog box can also be opened via the context menu for the selected S7 CPU, via the menu "Insert" - "New connection…" or by clicking the button in the menu bar.

6.5.2 TCP/IP connection

Proceed as follows

In the **Insert new connection** dialog box, the setting "(unspecified)" is maintained for the communication between an S7 CP and a SIMOTION device in the **Connection partner** field, as the communication partner – the SIMOTION device – is not available for selection. The desired connection type **TCP connection** is selected in the **Connection** field.


Insert New Connection

Connection Partner
- In the current project
  - pro_03
    - PG/PC(1)
    - (Unspecified)
    - All broadcast stations
    - All multicast stations
- In unknown project

Project:
Station: (Unspecified)
Module:

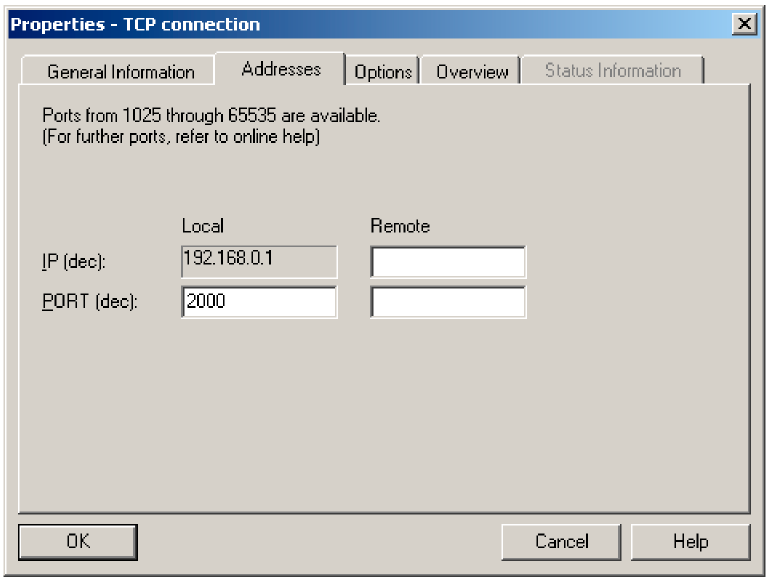
Connection
Type: TCP connection
Display properties before inserting

Figure 6-5 **Insert New Connection** dialog box with selected TCP/IP connection

If the **Insert New Connection** dialog box is exited with **OK** or **Accept**, a prompt appears to inform you that connections are also possible via subnets and the router addresses may have to be checked. After acknowledging this prompt, the **Properties - TCP - Connection** dialog box for a TCP/IP connection opens.

The IP address and the port for the local communication partner are already pre-assigned in the **Addresses** tab. The settings still have to be made for the remote communication partner. The IP address of the SIMOTION device must be entered in the IP (DEZ) field.

A port, specified by the user on the SIMOTION device for this communication connection, must be entered in the Port (DEZ) field. Supplementary conditions must be met for the port on the S7 side, i.e. a port between 2000 and 5000 must be selected on the S7 side. It is best to keep the proposed port.



Ports from 1025 through 65535 are available.
(For further ports, refer to online help)

	Local	Remote
IP (dec):	192.168.0.1	
PORT (dec):	2000	

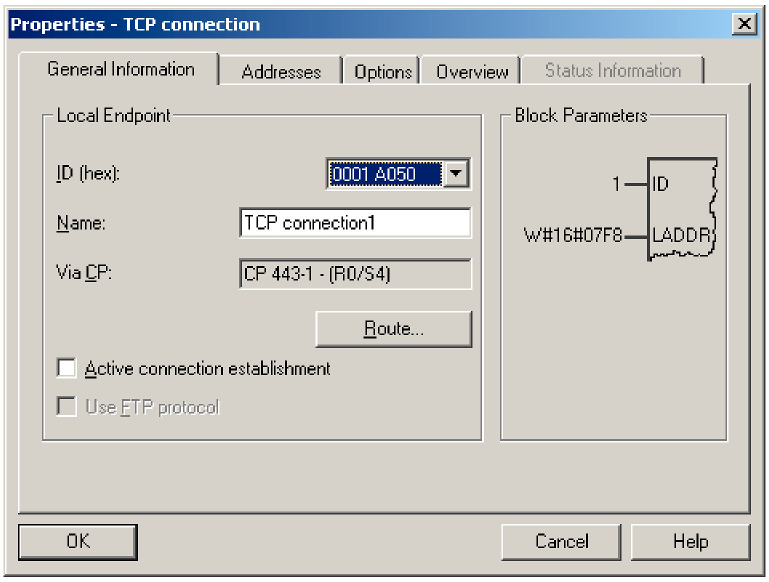
Figure 6-6 "Properties - TCP Connection" dialog box - "Addresses" tab

The important parameters for the parameterization of the SEND/RECEIVE interface, via which the connection in the user program can be referenced, can be taken from the **Block parameters** field in the **General** tab. The communication connection is assigned a unique reference with **ID**. The address of the CP is also specified as "LADDR".

The **Active connection buildup** checkbox can be used to specify whether the connection is to be established from the S7 station. If an active connection buildup is selected on the S7 side, the calls _tcpOpenServer and _tcpCloseServer must be used in the user program on the SIMOTION side to establish and close the connection.

If, however, an active connection buildup is not selected on the S7 side, the calls _tcpOpenClient and _tcpCloseConnection must be used in the user program on the SIMOTION side to establish and close the connection.

Once a connection is established – irrespective by which communication partner – both communication partners can use the connection to send and receive.



Local Endpoint

ID (hex): 0001 A050

Name: TCP connection1

Via CP: CP 443-1 - (R0/S4)

☐ Active connection establishment

☐ Use FTP protocol

Figure 6-7 **Properties - TCP Connection** dialog box - **General** tab

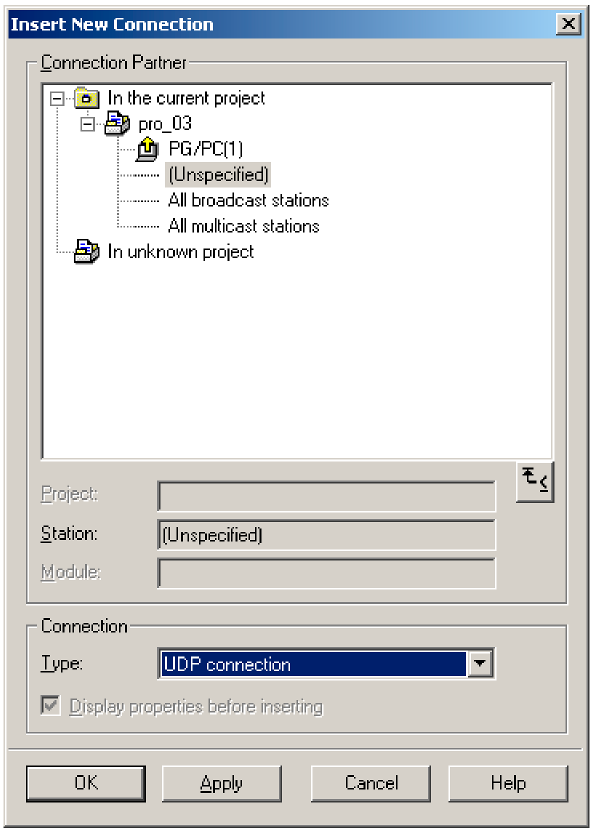
If the **Properties - TCP Connection** dialog box is exited with **OK**, the **Insert New Connection** dialog box must also be closed by clicking the **Close** button to finish the connection configuration. It is possible to configure further connections by selecting the desired connection type and then clicking the **Accept** button.

When the configuration of the communication connection is complete, the parameters for the call of the communication functions are defined in the S7 and SIMOTION user programs. The block parameters specified in the **General** tab are required for the S7 user program. The IP address of the SIMATIC CP, the port on the S7 side (local port in the **Address** tab) and the port on the SIMOTION side (partner port in the **Address** tab) are required for the SIMOTION user program.

6.5.3 UDP connection

Proceed as follows

In the **Insert new connection** dialog box the setting "(unspecified)" is maintained for the communication via a UDP connection between the SIMATIC S7 CP and a SIMOTION device in the **Connection partner** field, as the communication partner – the SIMOTION device – is not available for selection The desired connection type **UDP connection** is selected in the **Connection** field.



Station: (Unspecified)

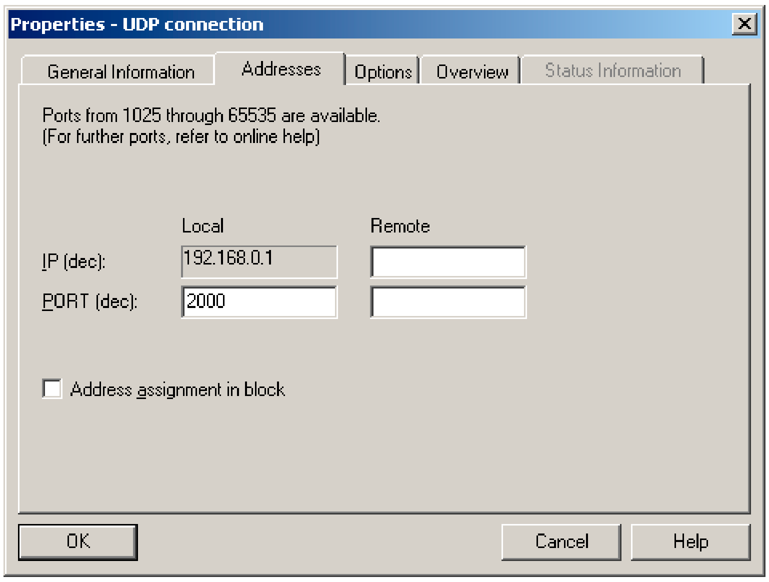
Type: UDP connection

Figure 6-8 **Insert New Connection** dialog box with selected UDP connection

If the **Insert New Connection** dialog box is exited with **OK** or **Accept** a prompt appears to inform you that connections are also possible via subnets and the router addresses may have to be checked. After acknowledging this prompt, the Properties dialog box for a UDP connection opens (see **Properties - UDP connection dialog box - Addresses tab**).

The IP address and also the port for the local communication partner are already pre-assigned in the **Addresses** tab. The settings still have to be made for the remote communication partner. The IP address of the SIMOTION device must be entered in the **IP (DEZ)** field. A port, specified by the user on the SIMOTION device for this communication connection, must be entered in the **Port (DEZ)** field.

Supplementary conditions must be met for the port on the S7 side, i.e. a port greater than 2000 must be selected on the S7 side. It is best to keep the proposed port. The **Address assignment on the block** checkbox is not activated.



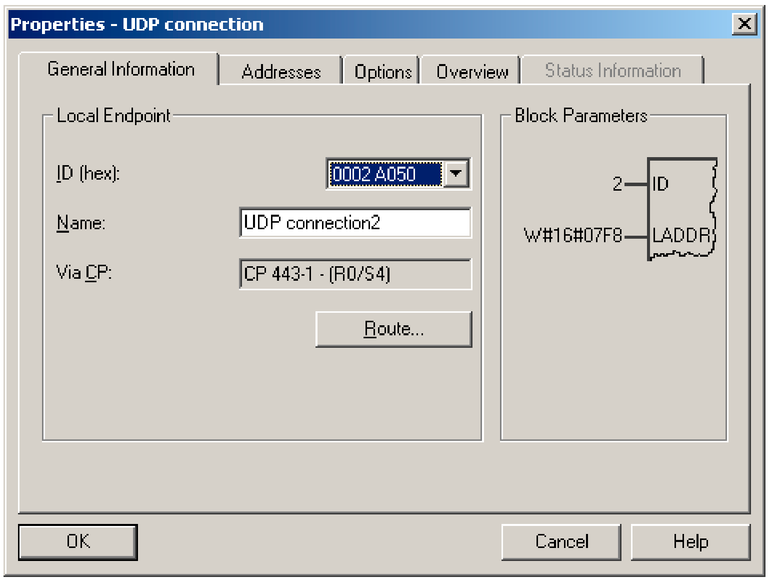
Ports from 1025 through 65535 are available.
(For further ports, refer to online help)

	Local	Remote
IP (dec):	192.168.0.1	
PORT (dec):	2000	

☐ Address assignment in block

Figure 6-9 **Properties - UDP Connection** dialog box - **Addresses** tab

The important parameters for the parameterization of the SEND/RECEIVE interface, via which the connection in the user program can be referenced, can be taken from the **Block parameters** field in the **General** tab. The communication connection is assigned a unique reference with **ID**. The address of the CP is also specified as "LADDR".



Local Endpoint

ID (hex): 0002 A050
Name: UDP connection2
Via CP: CP 443-1 - (R0/S4)

Figure 6-10 **Properties - UDP Connection** dialog box - **General** tab

If the **Properties - UDP Connection** dialog box is exited with **OK**, the **Insert New Connection** dialog box must also be closed by clicking the **Close** button to finish the connection configuration. It is possible to configure further connections by selecting the desired connection type and then clicking the **Accept** button.

When the configuration of the communication connection is complete, the parameters for the call of the communication functions are defined in the S7 and SIMOTION user programs. As already mentioned, the block parameters specified in the **General** tab are required for the S7 user program. The IP address of the SIMATIC CP, the port on the S7 side (local port in the **Address** tab) and the port on the SIMOTION side (partner port in the **Address** tab) are required for the SIMOTION user program.

6.6 Creating a communication connection between a SIMATIC CPU with integrated Ethernet interface and a SIMOTION device

Because of the different principle, the creation of a communication connection between a SIMATIC CPU with integrated Ethernet interface and a SIMOTION device differs from the configuration of a communication connection between a SIMATIC CPU with Ethernet CP and a SIMOTION device.

A connection cannot be inserted via NetPro for the communication between a SIMOTION and a SIMATIC CPU with integrated Ethernet interface – analogous to the communication between two SIMOTION devices.

The SIMATIC CPU and the SIMOTION device are only assigned IP addresses in HW Config.

The additional parameters required to establish the communication connection are specified by the user for the two communication partners and transferred during the block call on the S7 side and during the communication function call on the SIMOTION side.

On the S7 side, the parameters required to establish the communication connection are transferred by means of a data block which has a specific structure. The structure of the data block is explained in section SIMATIC S7 function blocks and SIMOTION functions to establish a TCP/IP connection (Page 118).

Communication between a SIMATIC CPU with integrated Ethernet interface and a SIMOTION device via UDP is not possible!

6.7 Using the functions and function blocks in the user program

6.7.1 Configuration flowchart and general information

Proceed as follows

The following figures show how a communication connection between a SIMATIC station and a SIMOTION station is configured. The flowchart also shows in which sections the individual steps are described in detail.

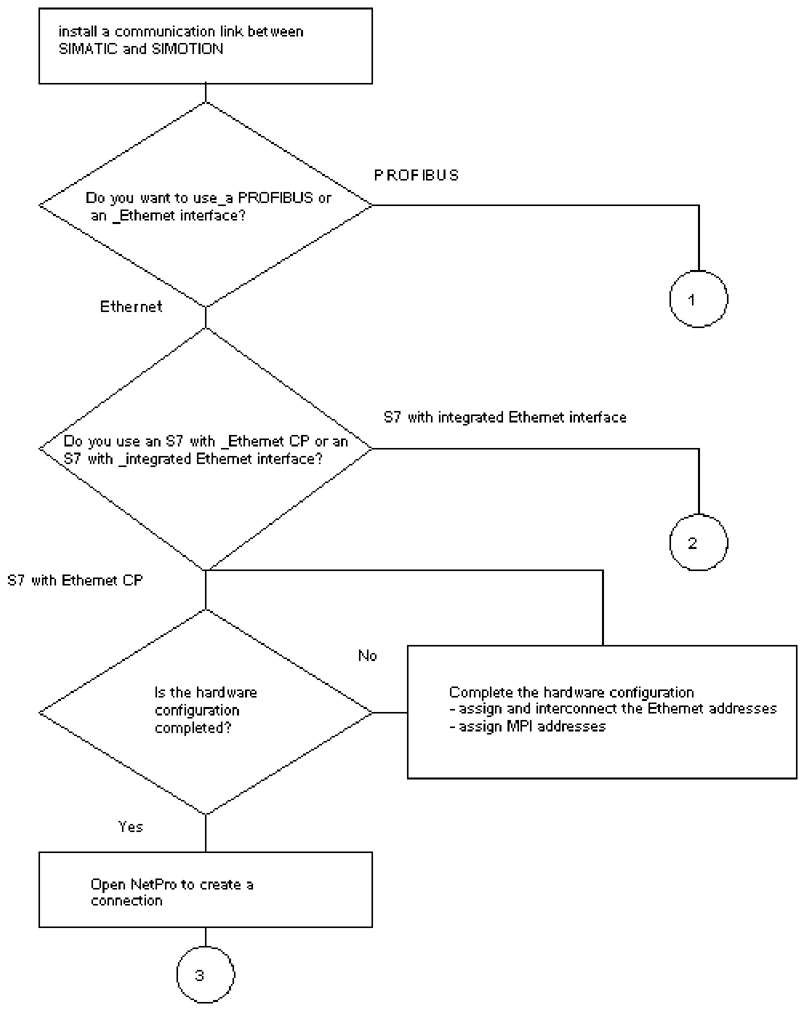

install a communication link between SIMATIC and SIMOTION
Do you want to use_a PROFIBUS or an _Ethernet interface?
PROFIBUS
1
Ethernet
Do you use an S7 with _Ethernet CP or an S7 with _integrated Ethernet interface?
S7 with integrated Ethernet interface
2
S7 with Ethernet CP
Is the hardware configuration completed?
No
Complete the hardware configuration
- assign and interconnect the Ethernet addresses
- assign MPI addresses
Yes
Open NetPro to create a connection
3

Figure 6-11 Flowchart for the configuration of a communication connection: selection of the communications protocol and the SIMATIC station

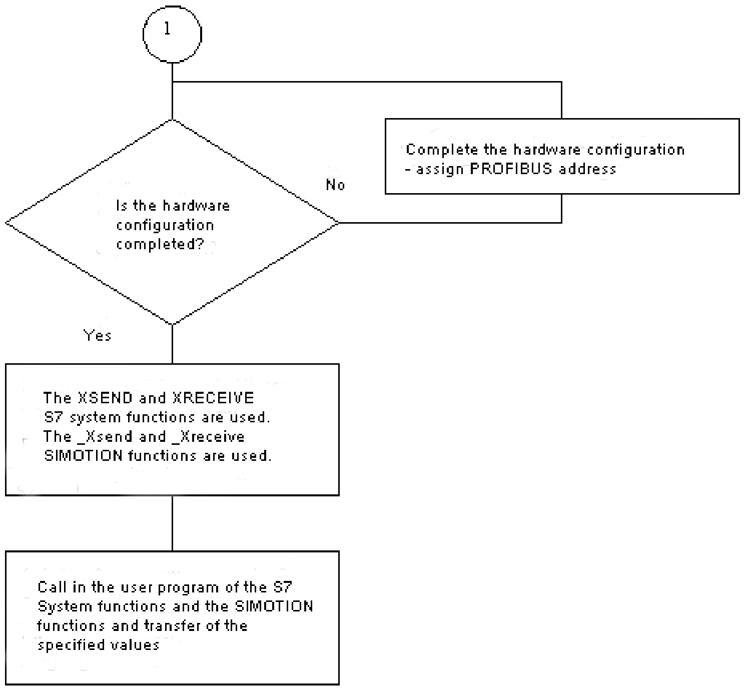

1
Is the hardware configuration completed?
No
Complete the hardware configuration - assign PROFIBUS address
Yes
The XSEND and XRECEIVE S7 system functions are used. The _Xsend and _Xreceive SIMOTION functions are used.
Call in the user program of the S7 System functions and the SIMOTION functions and transfer of the specified values

Figure 6-12 Flowchart for the configuration of a communication connection - continued: PROFIBUS connection

Note

The descriptions for the S7 functions and the SIMOTION functions are contained in section S7 system functions and SIMOTION functions for a PROFIBUS connection (Page 87).

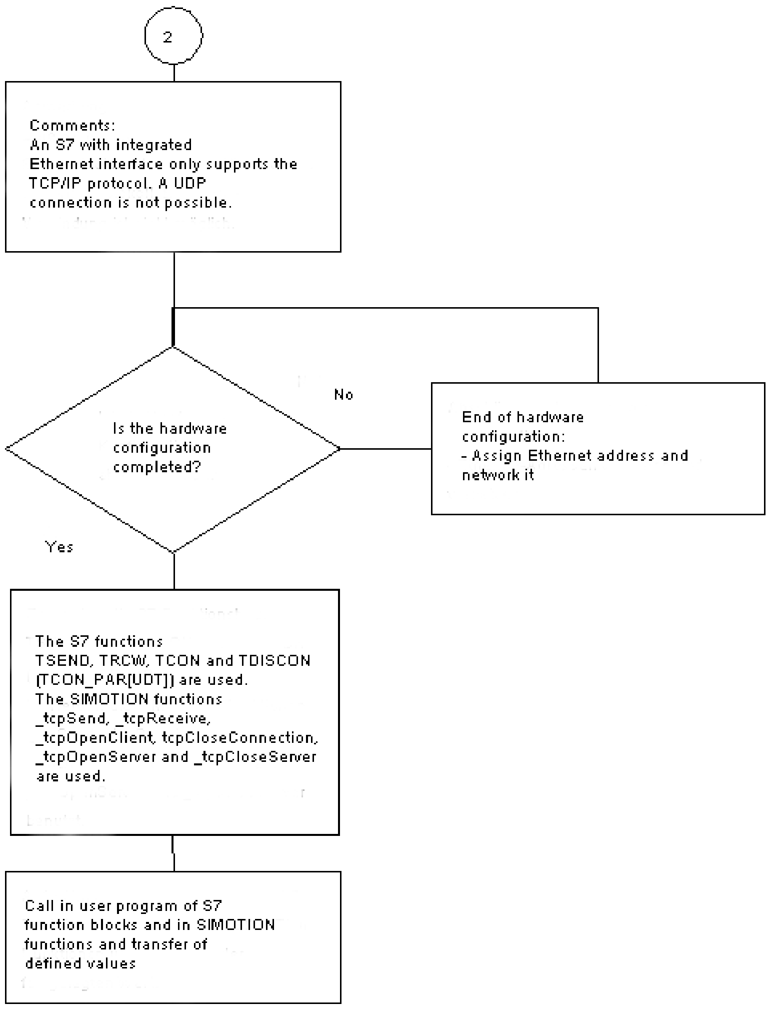

2
Comments:
An S7 with integrated Ethernet interface only supports the TCP/IP protocol. A UDP connection is not possible.
Is the hardware configuration completed?
No
End of hardware configuration:
- Assign Ethernet address and network it
Yes
The S7 functions TSEND, TRCW, TCON and TDISCON (TCON_PAR[UDT]) are used.
The SIMOTION functions _tcpSend, _tcpReceive, _tcpOpenClient, tcpCloseConnection, _tcpOpenServer and _tcpCloseServer are used.
Call in user program of S7 function blocks and in SIMOTION functions and transfer of defined values

Figure 6-13 Flowchart for the configuration of a communication connection - continued: TCP/IP connection (S7 with integrated Ethernet interface)

Note

The descriptions for the S7 functions and the SIMOTION functions are contained in section S7 function blocks and SIMOTION functions for a TCP/IP connection when using an S7 station with integrated Ethernet interface (Page 107).

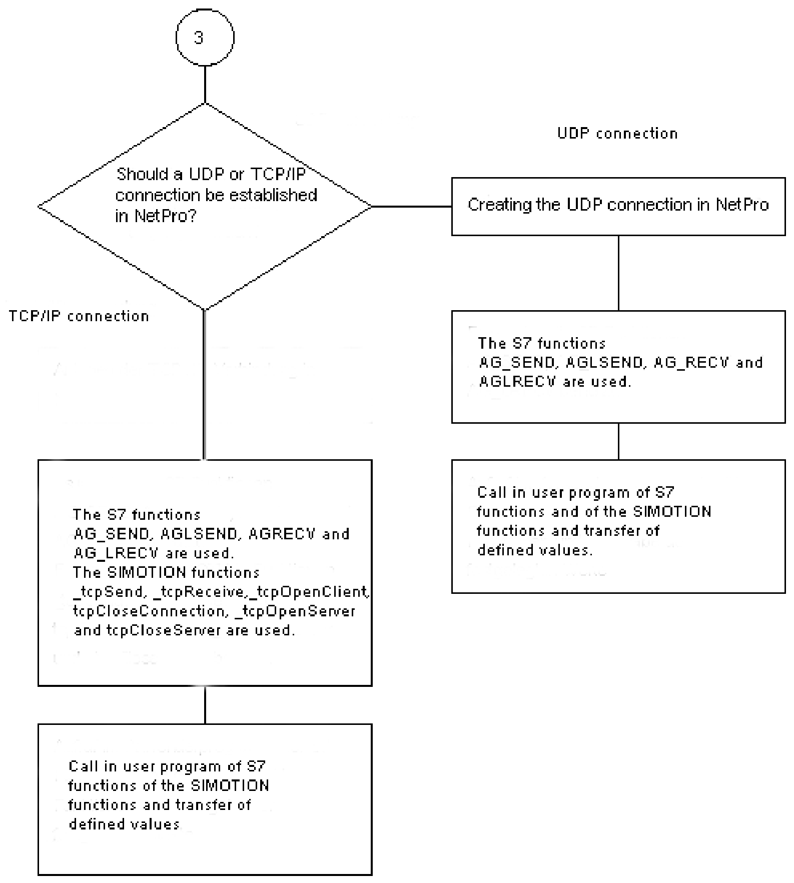

3
Should a UDP or TCP/IP connection be established in NetPro?
UDP connection
Creating the UDP connection in NetPro
TCP/IP connection
The S7 functions AG_SEND, AGLSEND, AG_RECV and AGLRECV are used.
Call in user program of S7 functions and of the SIMOTION functions and transfer of defined values.
The S7 functions AG_SEND, AGLSEND, AGRECV and AG_LRECV are used. The SIMOTION functions _tcpSend, _tcpReceive,_tcpOpenClient, tcpCloseConnection, _tcpOpenServer and tcpCloseServer are used.
Call in user program of S7 functions of the SIMOTION functions and transfer of defined values

Figure 6-14 Flowchart for the configuration of a communication connection - continued: TCP/IP and UDP connection (S7 with Ethernet CP)

Note

The descriptions for the S7 functions and the SIMOTION functions are contained in section *S7 and SIMOTION functions for a TCP/IP connection when using an S7 station with integrated Ethernet-CP*.

You must then use the specified or determined parameters during the parameterization of the interface in the user program, once you have performed the following:

- You have configured a communication connection between a SIMATIC CPU with Ethernet-CP and a SIMOTION device (see Configuring a communication connection between a SIMATIC with Ethernet CP and a SIMOTION device (Page 101))
- You have determined the parameters for the buildup of a communication connection for each block call (see Creating a communication connection between a SIMATIC CPU with integrated Ethernet interface and a SIMOTION device (Page 107))

Also the parameters specified for communication via PROFIBUS should be used for the parameterization of the user interface.

6.7.2 S7 and SIMOTION functions for a TCP/IP connection when using an S7 station with Ethernet CP

6.7.2.1 Introduction

The following section describes the parameterization of the SIMATIC S7 and the SIMOTION functions for a TCP/IP connection used with a SIMATIC S7 station with Ethernet CP.

6.7.2.2 S7 functions

Depending on the series of the S7 station (S7-300 or S7-400), two functions each are available for the send and receive direction for the described application case.

In the send direction, these are the FC5 (AG_SEND) and FC50 (AG_LSEND) functions. The following section shows the form of a call in the user program and the associated parameterization.

Table 6- 5 Program example

```
CALL "AG_Send"
 Act :=M0.0
 ID :=1
 LADDR :=W#16#3FFD
 SEND :=P#DB100.DBX0.0 BYTE 1000
 LEN :=1000
 DONE :=M0.1
 ERROR :=M0.2
 STATUS :=MW10
CALL "AG_LSEND"
 ACT :=M0.0
 ID :=1
 LADDR :=W#16#3FFD
 SEND :=P#DB100.DBX0.0 BYTE 1000
 LEN :=1000
 DONE :=M0.1
 ERROR :=M0.2
 STATUS :=MW1.0
```

The structure and thus the parameterization are identical. Therefore, the important parameters for both functions are explained together.

The ID and LADDR parameters are displayed when creating the connection in NetPro and must be transferred for this connection when the functions are called. Transmission is triggered via ACT. The SEND and LEN parameters define the send data and the associated length, respectively. The DONE, ERROR and STATUS parameters are used for the diagnostics or to return the status of the send job.

There are also two functions for the receive direction, FC6 (AG_RECV) and FC60 (AG_LRECV). The following section shows the form of a call in the user program and the associated parameterization.

Table 6- 6 Program example

```
Call "AG_RECV"
 ID :=1
 LADDR :=W#16#3FFD
 RECV :=P#DB110.DBX0.0 BYTE 1000
 NDR :=M2.0
 ERROR :=M2.1
 STATUS :=MW14
 LEN :=MW16
Call "AG_LRECV"
 ID :=1
 LADDR :=W#16#3FFD
 RECV :=P#DB110.DBX0.0 BYTE 1000
 NDR :=M2.0
 ERROR :=M2.1
 STATUS :=MW14
 LEN :=MW16
```

It can also be seen in the receive direction that the structure and therefore the parameterization of the functions are identical. Therefore, the important parameters for both functions are explained together.

The ID and LADDR parameters are specified by NetPro when creating the connection in NetPro and must be transferred for this connection when the functions are called. The RECV parameter specifies the data area in which the received data is stored. NDR informs the user when new data has been received (1: new data). The LEN parameter specifies the length in bytes of the new received data. The ERROR and STATUS parameters return a diagnosis or the status of the receive call.

6.7.2.3 SIMOTION functions

For the application case of a TCP/IP connection between an S7 station with Ethernet CP and a SIMOTION device, there are a total of six functions available on the SIMOTION side, which however are not all required at the same time.

Depending on the configuration of the connection buildup on the S7 side, different functions are used on the SIMOTION side to establish and close the connection.

If an active established connection has been specified on the S7 side, the connection is established with the _tcpopenserver function and closed with the _tcpcloseserver function on the SIMOTION side.

If an active established connection has not been configured on the S7 side, the connection is established with the _tcpopenclient function and closed with the _tcpcloseconnection function on the SIMOTION side.

The sending and receiving is performed via the _tcpsend and _tcpreceive functions independent of the configured connection buildup and close. However, a connection must be established before sending or receiving.

```
RetVal_TCPOpenClient  :=_TCPOpenClient
    (port             :=D435_Port,
    serveraddress     :=S7_IP_Adresse,
    serverport        :=S7_Port,
    nextcommand       :=WHEN_COMMAND_DONE);
```

Call example of the SIMOTION _tcpopenclient function

If a connection is to be established to an S7 station for which no active connection buildup has been selected on the S7 side, the _tcpopenclient function is called in the SIMOTION user program. When called, the locally assigned SIMOTION port is transferred to the function for the port parameter. The serveraddress parameter is the IP address of the S7 station which is transferred in an array. The port number designated as the local port number is transferred to the function in the serverport parameter. The behavior of this function with respect to the advance when called is parameterized with the nextcommand parameter. There are two setting options: IMMEDIATELY and WHEN_COMMAND_DONE. With the first value the advance is immediate and with the second value it is after completion of the command.

When the _tcpopenclient function is called, a structure is returned to the user program that contains the following parameters. The status of the connection buildup can be queried via the functionResult parameter. The connectionid parameter is used as (input) parameter for the call of the _tcpsend, _tcpreceive and _tcpcloseconnection functions and assigns a unique TCP/IP connection to these functions. This return value is referred to in the following call examples.

```
RetVal_TCPSend      := _TCPSend
    (connectionid   :=RetVal_TCPOpenClient.ConnctionID,
    nextcommand     :=WHEN_COMMAND_DONE,
    datalength      :=Soll_Sende_Datenlaenge,
    data            :=TCP_Sende_Daten);
```

Call example of the SIMOTION _tcpsend function

The _tcpsend function is called in the SIMOTION user program to send data from SIMOTION to the SIMATIC.

The parameters that have to be transferred when the function is called are described in the following. The connectionid return value of the _tcpopenclient or _tcpopenserver functions is transferred for the connectionid parameter - depending on which station the active connection request was started, in order to uniquely define the connection to be used for sending. The behavior of this function with respect to the advance when called is also parameterized with the nextcommand parameter. There are two setting options: IMMEDIATELY and WHEN_COMMAND_DONE. With the first value, the advance is immediately and with the second value, after completion of the command. The datalength parameter informs the function of the user data length in bytes to be transferred. The data parameter specifies the location of the user data area for the send data to be transferred with the function.

The return value of the function to the user program is of DINT data type. The various return values indicate any problems during the execution of the function. There is also a confirmation when the data has been successfully sent.

```
RetVal_TCPReceive    := _TCPReceive
    (connectionid    :=RetVal_TCPOpenclient.ConnectionID,
     nextcommand     :=IMMEDIATELY,
     receivevariable :=TCP_Empfangs_Daten);
```

Call example of the SIMOTION _tcpreceive function

The _tcpreceive function is called in the SIMOTION user program to receive data from a SIMATIC station on the SIMOTION side.

Various parameters are transferred when the function is called. The connectionid return value of the _tcpopenclient or _tcpopenserver functions is transferred for the connectionid parameter - depending on which station the active connection request was started, in order to uniquely define the connection to be used for receiving. The behavior of the _tcpreceive function with respect to the advance when called is also parameterized with the nextcommand parameter. There are two setting options: IMMEDIATELY and WHEN_COMMAND_DONE. With the first value, the advance is immediately and with the second value, after completion of the command. Typically during the transfer of various user data lengths between SIMATIC and SIMOTION, starting from a user data length of more than 240 bytes, the user data is transferred in unpredictable packet sizes.

In this case, it must also be ensured that the user data is stored in the correct sequence before the evaluation and processing on the SIMOTION side. To do this, the nextcommand parameter should be set to IMMEDIATELY. The receivevariable parameter informs the function in which user data area the data received from the SIMATIC side is to be stored.

When the _tcpreceive function is called, a structure is returned to the user program that contains the following parameters. The receive status can be queried via the functionresult parameter. The datalength parameter returns the number of received user data bytes after successful call of the _tcpreceive function.

```
RetVal_TCPCloseConnection    := _TCPCloseConnection
    (connectionid            :=RetVal_TCPOpenclient.ConnectionID
    );
```

Call example of the SIMOTION _tcpcloseconnection function

The _tcpcloseconnection function is called on the SIMOTION side to close a connection for which no active connection buildup has been selected in the connection configuration on the S7 side (i.e. the Active connection buildup checkbox has not been activated).

Only the connectionID return value of the _tcpopenclient function is transferred to the function in the connectionid parameter to uniquely specify the connection to be closed.

The return value of the function to the user program has DINT data type and indicates any problems during the execution of the function or signals if the connection has been closed successfully.

```
RetVal_TCPOpenServer                 := _TCPOpenServer
    &#i921; port:=D435_Port, backlog   := 5,
    nextcommand                        := WHEN_COMMAND_DONE);
```

Call example of the SIMOTION _tcpopenserver function

The _tcpopenserver function is called on the SIMOTION side to open the connection for the data exchange if an active connection buildup has been specified in the connection configuration on the S7 side (i.e. the Active connection buildup checkbox has not been activated).

To parameterize the function, the locally assigned SIMOTION port is transferred for the port parameter. The maximum number of parallel connection requests for this port that are to be permitted from other controllers is also specified as a further parameter for backlog. The behavior of this function with respect to the advance when called is also parameterized with the nextcommand parameter. There are two setting options: IMMEDIATELY and WHEN_COMMAND_DONE. With the first value the advance is immediate and with the second value it is after completion of the command.

When the _tcpopenserver function is called, the structure returned to the user program contains the following parameters. The status of the connection buildup can be queried via the functionResult parameter. The connectionid parameter is used as (input) parameter for the call of the _tcpsend and _tcpreceive functions and assigns a unique TCP/IP connection to these functions. This return value is referred to in the above call examples.

The two following parameters that are returned to the user program in the structure are configured by the user in NetPro and therefore known. However, for completeness, they should still be specified. The clientAddress parameter returns as array the IP address of the S7 station from which the connection is activated. The port number designated as the local port number of the S7 station is specified in the clientPort parameter.

```
 RetVal_TCPCloseServer:= _TCPCloseServer(port:= D435_Port);
```

Call example of the SIMOTION _tcpcloseserver function

The _tcpcloseserver function is called for the connection on the SIMOTION side if a connection is to be closed for which an active connection buildup has been selected in the connection configuration on the S7 side (i.e. the Active connection buildup checkbox has been activated).

Only the locally assigned SIMOTION port is transferred to the function for the port parameter.

The return value of the function to the user program has DINT data type and indicates any errors in the parameterization of the function or signals if the port has been closed successfully.

6.7.3 S7 and SIMOTION functions for a UDP connection when using an S7 station with Ethernet CP

6.7.3.1 Introduction

The following section describes the parameterization of the S7 and the SIMOTION functions for a UDP connection that was created for use with an S7 station with Ethernet CP.

6.7.3.2 S7 functions

The functions used for this application case are also the S7 FC5 (AG_SEND), FC50 (AG_LSEND), FC6 (AG_RECV) and FC60 (AG_LRECV) functions described previously.

The ID and LADDR parameters assigned for the connection by NetPro are shown using an example. All further parameters for the parameterization of the above functions for a UDP connection between an S7 station with Ethernet CP and a SIMOTION device have already been described and can be referenced.

6.7.3.3 SIMOTION functions

Two functions are used on the SIMOTION side for the application case of a UDP connection between an S7 station with Ethernet CP and a SIMOTION device.

The sending of data is performed via _udpSend. If data is to be received on the SIMOTION side, the _udpReceive function is used. The following program examples show the call and parameterization.

```
RetVal_UDPSend          :=_UDPSend(sourceport:= P350_Port,
    destinationaddress  :=S7 IP address,
    destinationport     :=S7_Port,
    communicationmode   :=CLOSE_ON_EXIT,
    datalength          :=UDPDatalength_Send,
    data                :=UDPSendData);
```

Sample call of the SIMOTION _udpSend function

When the _udpSend function is called, the port assigned on the SIMOTION side is transferred for the sourceport parameter. The destinationaddress parameter is an array that specifies the IP address of the S7 station. The IP address of the S7 station can be configured and read out in HW Config. The port specified as "local port" on the S7 side is transferred as destinationport. The user can specify with communicationmode whether the communication resources are to be released after sending (CLOSE_ON_EXIT) or not (DO_NOT_CLOSE_ON_EXIT). The datalength and data parameters specify the data length to be sent or the area where the sent data is stored.

The status of the send job can be checked via the return value of the function.

```
RetVal_UDPReceive       :=_udpreceive(port:=P350_Port,
    communicationmode   :=CLOSE_ON_EXIT,
    nextcommand         :=WHEN_COMMAND_DONE,
    receivevariable     :=UDPReceiveData);
```

Call example of the SIMOTION _udpReceive function

When the _udpReceive function is called, the port designated as "Partner port" on the SIMOTION side is also specified for the port parameter. The user can also specify with communicationmode whether the communication resources are to be released after receipt (CLOSE_ON_EXIT) or not (DO_NOT_CLOSE_ON_EXIT).

The behavior of this function with respect to the advance when called is parameterized with the nextcommand parameter. There are three setting options for this parameter: IMMEDIATELY, WHEN_COMMAND_DONE and ABORT_CURRENT_COMMAND.

With the first two values advance is either immediately or after completion of the command. With the third value, if the same port number as in the previous function call is transferred, the active function is aborted. The receivevariable parameter specifies the buffer in which the receive data is stored.

When the _udpReceive function is called, the structure returned to the user program contains the following parameters. The call status of the receive function can be queried in the functionResult parameter. The sourceAddress parameter is an array that contains the IP address of the S7 station. The sourceport parameter of the structure also contains the port of the S7 station designated as local port. The number of received user data bytes after a successful call of the _udpReceive function can be fetched in the datalength parameter.

6.7.4 S7 function blocks and SIMOTION functions for a TCP/IP connection when using an S7 station with integrated Ethernet interface

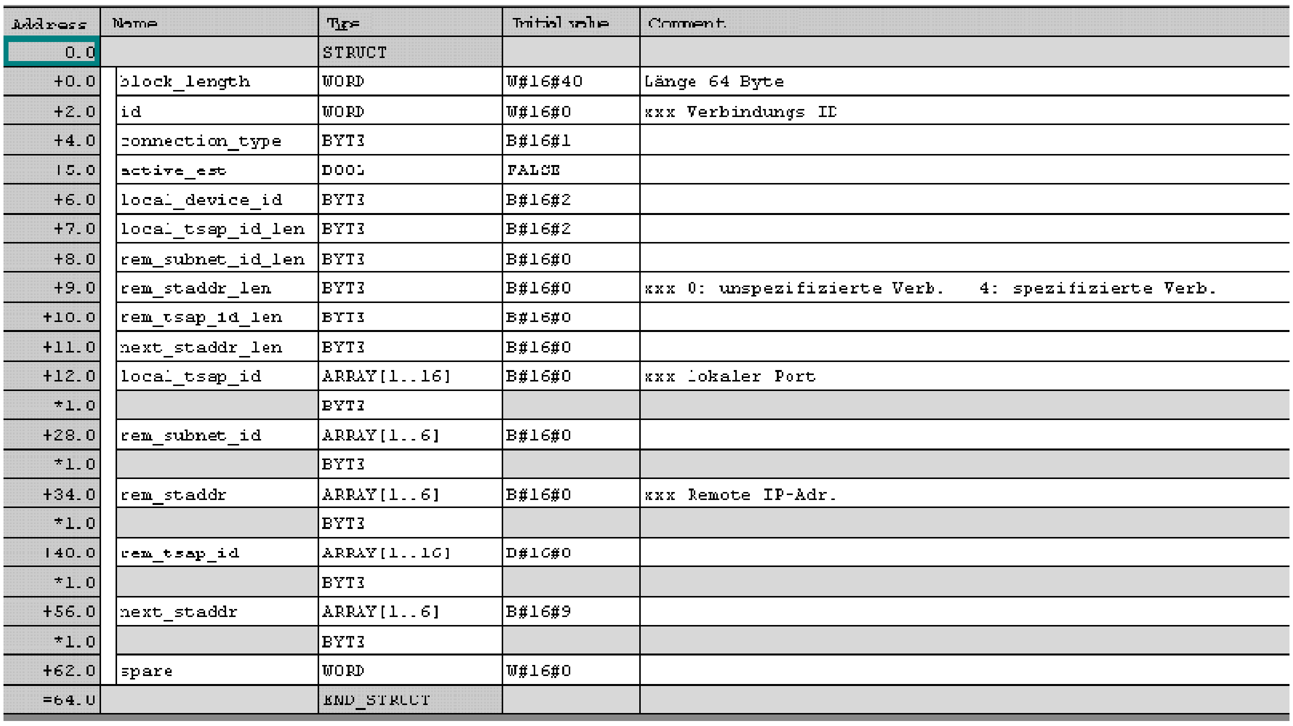
6.7.4.1 Introduction

The following section describes the parameterization of the S7 function blocks and the SIMOTION functions for a TCP/IP connection created for use with an S7 station with integrated Ethernet interface.

6.7.4.2 S7 function blocks

Various function blocks and a UDT are available on the SIMATIC side for the communication between a SIMATIC station with integrated Ethernet interface and a SIMOTION device. The FB65 TCON is called to establish the connection. The FB66 TDISCON function block is used to close the connection. The sending and receiving of data is performed using the FB63 TSEND and FB64 TRCV blocks.

For the communication between a SIMATIC station with integrated Ethernet interface and a SIMOTION device, the connection is not configured in NetPro. Instead, the connection is configured in the user program. On the SIMATIC side, this is implemented with data blocks derived from UDT65 TCON_PAR (see figure below). This means that the data blocks must contain the data structure from the UDT65.



Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	block_length	WORD	W#16#40	Länge 64 Byte
+2.0	id	WORD	W#16#0	xxx Verbindungs ID
+4.0	connection_type	BYTE	B#16#1	
+5.0	active_est	BOOL	FALSE	
+6.0	local_device_id	BYTE	B#16#2	
+7.0	local_tsap_id_len	BYTE	B#16#2	
+8.0	rem_subnet_id_len	BYTE	B#16#0	
+9.0	rem_staddr_len	BYTE	B#16#0	xxx 0: unspezifizierte Verb. 4: spezifizierte Verb.
+10.0	rem_tsap_id_len	BYTE	B#16#0	
+11.0	next_staddr_len	BYTE	B#16#0	
+12.0	local_tsap_id	ARRAY[1..16]	B#16#0	xxx lokaler Port
*1.0		BYTE		
+28.0	rem_subnet_id	ARRAY[1..6]	B#16#0	
*1.0		BYTE		
+34.0	rem_staddr	ARRAY[1..6]	B#16#0	xxx Remote IP-Adr.
*1.0		BYTE		
+40.0	rem_tsap_id	ARRAY[1..16]	B#16#0	
*1.0		BYTE		
+56.0	next_staddr	ARRAY[1..6]	B#16#9	
*1.0		BYTE		
+62.0	spare	WORD	W#16#0	
=64.0		END_STRUCT		

Figure 6-15 Declaration view of the UDT65 TCON_PAR in the LAD/STL/FBD editor

The structure and parameterization of the UDT65 is described in detail in the following. In principle, there are two cases for the parameterization of the UDT65 or the data block derived from this:

1. The connection is established actively from the S7 station.
2. The S7 station passively waits for the connection to be established from the communication partner.

Before the special parameterization of the two basic types is considered, the parameters and their parameterization, which remain the same irrespective of the role of the S7 station during the connection buildup, are described first.

The block_length parameter contains the length of a parameterization block and is permanently set to 64. The parameter must not be changed.

The connection type is set via the connection_type parameter. The value 1 is entered permanently. This means that the connection type is "TCP/IP native". This value must also not be changed.

A further parameter that may not be changed and that is permanently assigned is local_device_id. The value 2 (meaning Industrial Ethernet) must be entered here and must not be changed. The rem_subnet_id_len and next_staddr_len parameters must be assigned the value 0 and must also not be changed.

There are also the rem_subnet_id and next_staddr parameters and a reserved area designated as spare. The parameters or reserved area are not relevant for the connection buildup or the communication via a TCP/IP connection. However, the parameters or reserved area should still be specified with 0 and the value should be retained.

The parameterizations depending on the connection buildup are now described separately in the following.

- For 1:

If the active connection is to be established from the S7 station, the active_est parameter must be set to TRUE. This specifies that the connection is to be actively established from the S7 station.

The connection is assigned a unique number, the id parameter, that can be used to reference this connection. This reference is required for the parameterization of the TCON, TSEND, TRCV and TDISCON function blocks.

If the connection is actively established from the S7 side, the local port number is irrelevant. Therefore, the local_tsap_id_len parameter is set to 0.

The rem_staddr_len parameter is set to 4, as a valid IP address can be expressed as four numbers displayable with a byte - separated by a dot in written form.

The port number length of the communication partner (SIMOTION device) is specified in the rem_tsap_id_len parameter. Because the port number can be displayed with two bytes, this parameter is set to 2.

Because the port number on the S7 side does not play a role in this case and the local_tsap_id_len parameter has the value 0, the local_tsap_id parameter does not have to be assigned.

The rem_staddr parameter structured as an array of byte variables specifies the IP address of the SIMOTION device. The first four bytes of the array are assigned. Whereby the positions of the IP address are entered in the array from right to left in ascending order. This means that the right number is entered as hexadecimal number in the first index of the array. The second number from the right is then entered in the second index, etc.

To complete the parameterization, the port number of the SIMOTION device is specified in the rem_tsap_id parameter, which is also structured as an array of byte variables. The first index contains the low-order byte of the port number converted to a hexadecimal number. The second index contains the high-order byte of the port number converted to a hexadecimal number.

- For 2:

If no active connection is to be established from the S7 station, the active_est parameter must be set to FALSE. This specifies that the connection is not to be actively established from the S7 station.

Also in this case, the connection on the S7 side is assigned a unique number, the id parameter, that can be used to reference this connection. The unique number is transferred to the function blocks TCON, TSEND, TRCV and TDISCON during the function block call.

As already mentioned above, a port number can be displayed with two bytes. The local port number on the S7 is relevant and therefore the local_tsap_id_len parameter must be set to 2.

As in the previous case a, rem_staddr_len is set to 4, because a valid IP address consists of four numbers displayable with a byte - separated by a dot in written form.

The port number length of the communication partner (SIMOTION device) is specified in the rem_tsap_id_len parameter. However, in this case, the port number on the SIMOTION side is not relevant. For this reason, the rem_tsap_id_len parameter is set to 0.

The port number of the S7 station is entered in ascending order in the local_tsap_id parameter - an array of byte variables. The first index contains the low-order byte of the port number converted to a hexadecimal number. The second index contains the high-order byte of the port number converted to a hexadecimal number.

The IP address of the SIMOTION device is specified in the rem_staddr parameter (an array of byte variables). The first four bytes of the array are assigned. Whereby the positions of the

IP address are entered in the array from right to left in ascending order. This means that the right number is entered as hexadecimal number in the first index of the array. The second number from the right is then entered in the second index, etc.

Because the port number on the SIMOTION device side does not play a role in this case and the rem_tsap_id_len parameter has the value 0, the rem_tsap_id parameter does not have to be assigned.

Then follows the description of the function blocks with which a connection to a SIMOTION device can be established or closed and with which data can be sent to or received from the SIMOTION device.

```
CALL "TCON" , DB66
    REQ       :=M1.0
    ID        :=W#16#1
    DONE      :=M2.0
    BUSY      :=M3.0
    ERROR     :=M4.0
    STATUS    :=MW100
    CONNECT   :=P#DB1.DBX0.0 BYTE 64
```

Call example of the FB65 (TCON) function block

If data is to be received from a SIMOTION device on an S7 station with integrated Ethernet interface or data sent from an S7 station with integrated Ethernet interface to a SIMOTION device, then first of all a connection between the S7 station and the SIMOTION device must be established via the function block FB65 TCON.

The above program example shows a sample call to the FB65 TCON function block. The connection buildup is controlled via the REQ parameter. If the parameter is set to 1 and therefore an edge created, the data (connection description) from the area specified under CONNECT is transferred to the function block in order to establish the connection.

A reference to the desired connection to be established is specified via the ID parameter.

The DONE, BUSY and ERROR parameters can be used to query the execution status of the function block. In addition to the information that an error has occurred (ERROR = 1), the user also receives detailed information about the type of error via the STATUS parameter.

As already mentioned above, the CONNECT parameter contains the addresses and length of the connection description. This address refers to a data block area whose structure corresponds to the UDT65.

```
Call "TSEND" , DB63
    REQ        :=M5.0
    ID         :=W#16#1
    LEN        :=10
    DONE       :=M6.0
    BUSY       :=M7.0
    ERROR      :=M8.0
    STATUS     :=MW200
    DATA       :=DB10.DBBO
```

Call example of the FB63 (TSEND) function block

Once a communication connection has been established, it can be used to send data from the S7 station with integrated Ethernet interface to the SIMOTION device. This is performed by calling the FB63 TSEND function block.

The transmission is activated with a rising edge at the REQ parameter. When called for the first time, the data from the area specified with the DATA parameter is transferred to the function block.

The ID parameter is used to reference the communication connection over which the data is to be sent. The LEN parameter specifies the length of the data to be sent in bytes.

The DONE, BUSY and ERROR parameters also display the execution status of the function block. In addition to the information that an error has occurred (ERROR = 1), the user also receives detailed information about the type of error via the STATUS parameter.

As already mentioned above, the DATA parameter contains the address and length of the send area.

```
CALL "TRCV" , DB64
    EN_R      :=M8.0
    ID        :=W#16#1
    LEN       :=10
    NDR       :=M9.0
    BUSY      :=10.0
    ERROR     :=11.0
    STATUS    :=MW300
    RCVD_LEN  :=MW310
    DATA      :=DB20.DBB0
```

Call example of the FB64 (TRCV) function block

Data sent from a SIMOTION device can also be received on the S7 station with integrated Ethernet interface via an established connection. The FB64 TRCV function block is called for this purpose.

Receiving is controlled with the EN_R parameter. This means, if the EN_R parameter is assigned the value 1, data can be received.

The ID is used to select a specific communication connection to be used to receive the data.

There are two principle parameterization settings for the LEN parameter. If the parameter is assigned the value 0, the length of the expected receive data is implicitly specified via an ANY pointer on the DATA block input. As soon as data is received, the data is provided in the receive buffer and this is signalled via the NDR parameter. The length of the received data can be taken from the RCVD_LEN parameter and it can also be less than the size stored in the DATA parameter. If the LEN parameter is assigned a value other than 0, the received data is temporarily stored in the receive buffer and only provided when the configured length is reached. The NDR parameter also signals when the data has been completely received.

The NDR parameter signals the partial or complete reception of data.

For receiving, the DONE, BUSY and ERROR parameters indicate the execution status of the function block. In addition to the information that an error has occurred (ERROR = 1), the user also receives detailed information about the type of error via the STATUS parameter.

The meaning of the RCVD_LEN parameter has already been explained above. If the LEN parameter has been assigned the value 0, the RCVD_LEN parameter specifies the number of data bytes contained in the most recently received data block. If a value other than 0 has been assigned in the LEN parameter, the same value is present in RCVD_LEN.

The DATA parameter contains the address and length of the send area. The received data can be taken from here for further processing.

```
CALL    "TDISCON" , DB66
    REQ     :=M12.0
    ID      :=W#16#1
    DONE    :=M13.0
    BUSY    :=M14.0
    ERROR   :=M15.0
    STATUS  :=MW400
```

Call example of the FB66 (TDISCON) function block

The FB66 TDISCON function block is used to close an existing connection. To close the connection, the input parameter REQ is set to 1. The closing of the connection is therefore triggered by the rising edge.

The ID parameter informs the function block which connection is to be closed. This parameter specifies a reference to an already established connection defined by means of a structure of type TCON_PAR.

The DONE, BUSY and ERROR parameters can be used to query the execution status of the function block. In addition to the information that an error has occurred (ERROR = 1), the user also receives detailed information about the type of error via the STATUS parameter.

6.7.4.3 SIMOTION functions

The same functions (_tcpopenclient, _tcpsend, _tcpreceive, _tcpcloseconnection, _tcpopenserver, _tcpcloseserver) are used for this application case on the SIMOTION side as described previously.

However, as already mentioned, the port numbers are not assigned in NetPro, but specified by the user in the block or function parameterization.

6.7.5 Processing of TCP/IP data packets in the SIMOTION user program

A special feature of the communication via the TCP/IP protocol is that the data, even when the maximum transferable data length has not been exceeded, is sent in individual packets of unpredictable size.

On the S7 side, calling the FC6 (AG_RECV) or FC60 (AG_LRECV) function ensures that the user data arriving in packets is provided to the user in the length specified during the function call.

However, on the SIMOTION side, the function call _tcpReceive does not transfer the entire data sent by the communication partner to the user all at once. The user must ensure that the arriving data packets of unknown length are written to a separate buffer without gaps and in the correct sequence.

The following flowchart shows a possible solution for the receipt in SIMOTION, with known total data length ("specified data length") and unknown length of the individual data packets (each corresponds to the "actual data length").

Receipt at the SIMOTION end using _tcpReceive

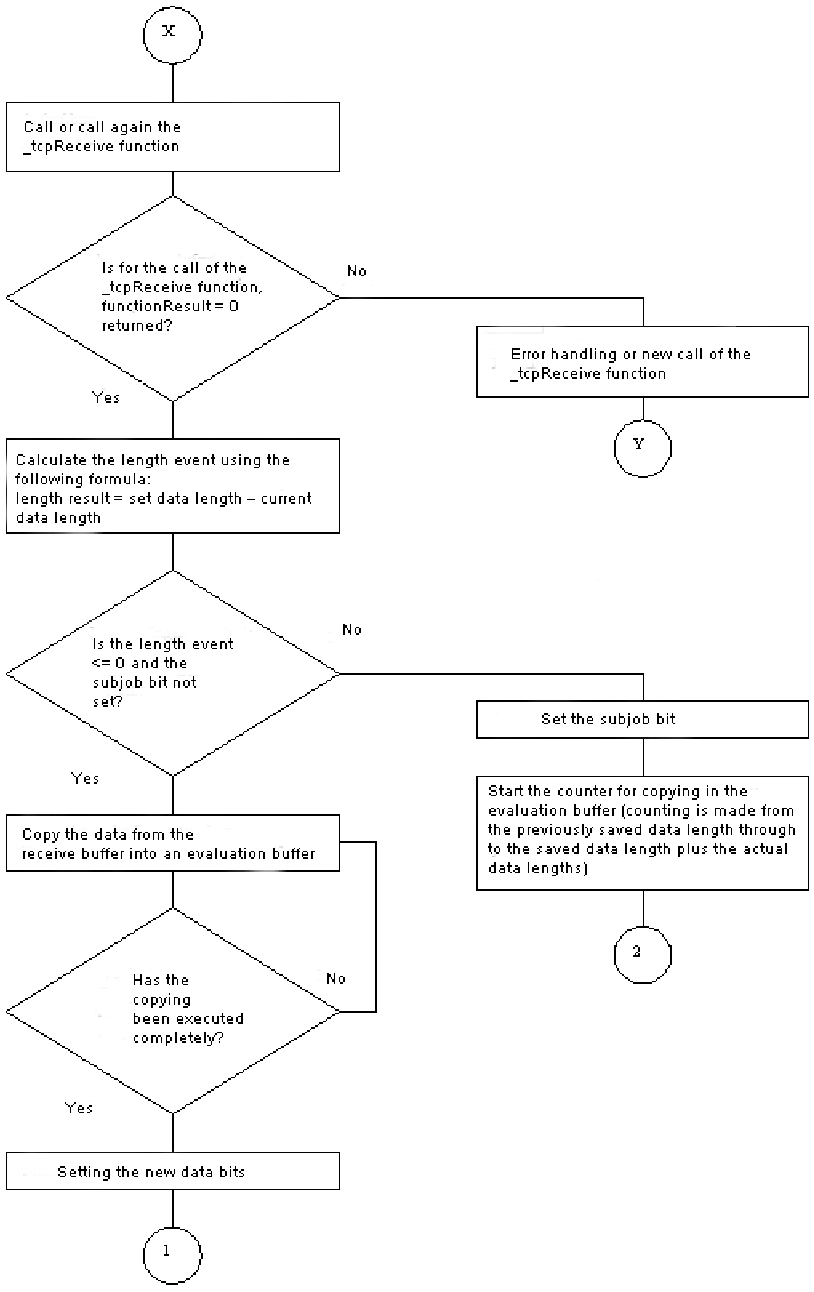

X
Call or call again the _tcpReceive function
Is for the call of the _tcpReceive function, functionResult = 0 returned?
No
Error handling or new call of the _tcpReceive function
Y
Yes
Calculate the length event using the following formula: length result = set data length – current data length
Is the length event <= 0 and the subjob bit not set?
No
Set the subjob bit
Start the counter for copying in the evaluation buffer (counting is made from the previously saved data length through to the saved data length plus the actual data lengths)
2
Yes
Copy the data from the receive buffer into an evaluation buffer
Has the copying been executed completely?
No
Yes
Setting the new data bits
1

Figure 6-16 Flowchart for receipt at SIMOTION end using _tcpReceive

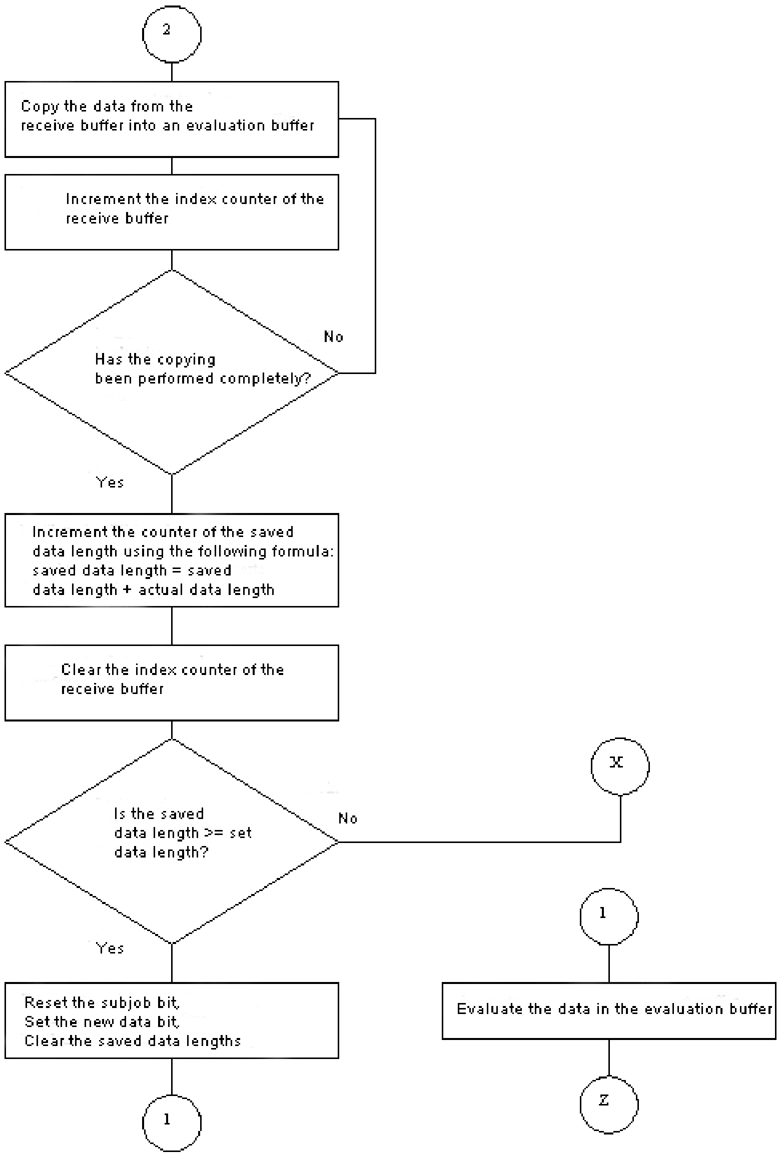

2
Copy the data from the receive buffer into an evaluation buffer
Increment the index counter of the receive buffer
Has the copying been performed completely?
No
Yes
Increment the counter of the saved data length using the following formula: saved data length = saved data length + actual data length
Clear the index counter of the receive buffer
Is the saved data length >= set data length?
No
X
Yes
Reset the subjob bit, Set the new data bit, Clear the saved data lengths
1
1
Evaluate the data in the evaluation buffer
Z

Figure 6-17 Flowchart for receipt at SIMOTION end using _tcpReceive - continued

6.8 Details of the SIMOTION TCP/IP system functions

6.8.1 _tcpOpenServer function

The _tcpOpenServer function implements the server functionality of a TCP/IP connection. Once it is called, _tcpOpenServer waits for the connection requests from the communication nodes (clients) at the port specified by the "port" parameter.

The returned connectionId is required for the following write and read calls (_tcpSend, _tcpReceive).

After this, the function is available again for the establishment of other connections. If no further connection is to be established, the resources used by _tcpOpenServer can be released again by calling _tcpCloseServer. This does not automatically close any previously established connections; they have to be closed by calling _tcpCloseConnection.

The "backlog" parameter describes the maximum number of connection requests that the server can backlog while processing a currently running connection request.

The function must have return value = 16#0 before the data transfer with TCP/IP can start.

The function may only be called in the BackgroundTask or in a MotionTask.

```
_tcpOpenServer:StructRetTcpOpenServer
    (
     port :UINT,
     backlog :DINT,
     nectCommand :EnumTcpNextCommandMode
     );
```

6.8.2 _tcpOpenClient function

The _tcpOpenClient function implements the client side of a TCP/IP connection. When the function is called, there is a connection request to the server addressed by serverAddress and serverPort. The returned connectionId is required for the following write and read calls (_tcpSend, _tcpReceive).

The function must have return value = 16#0 before the data transfer with TCP/IP can start.

The function may only be called in the BackgroundTask or in a MotionTask.

```
_tcpOpenClient:StructRetTcpOpenClient
    (
     port            :UINT,
     serverAddress   :ARRAY [0 ... 3] of USINT;
     serverPort      :UINT;
     nextCommand     :EnumTcpNextCommandMode
    );
```

6.8.3 _tcpReceive function

_tcpReceive waits for data at the active connection to the communication partner and fetches this data. The function receives data via a connection previously established with _tcpOpenServer or _tcpOpenClient. Data can be received in arbitrary packet sizes. Generally the packet sizes do not correspond to that of the transmitting side.

(i.e. a send packet can be split up into several receive packets. However, several send data packets can also be combined into one receive packet.)

The received data is available in the receiveVariable parameter in the datalength if the return value in the function result = 16#0. At the next function call, the receiveVariable parameter is overwritten in the datalength with new data.

Negative values in functionResult indicate an error in the data transfer. In this case, the connection must be disconnected by calling _tcpCloseConnection.

The function may only be called in the BackgroundTask or in a MotionTask.

```
_tcp_Receive:StructRetTcpReceive
    (
     connectionId      :DINT,
     nextCommand       :EnumNextCommandMode,
     receiveVariable   :ARRAY [0 ... 4095] of BYTE
    );
```

6.8.4 _tcpSend function

The _tcpSend function is used to send data to a communication partner via a connection previously established with _tcpOpenServer or _tcpOpenClient.

Both clients and server can send data via the active connection.

With an asynchronous call (nextCommand = IMMEDIATELY), the function must be called until it returns the value 0 (or a negative value in case of an error).

Negative values in functionResult indicate an error in the data transfer. In this case, the connection must be disconnected by calling _tcpCloseConnection.

The function may only be called in the BackgroundTask or in a MotionTask.

```
_tcpSend:DINT
   (
    connectionId :DINT,
    dataLength   :UDINT,
    data         :ARRAY [0 ... 4095] of BYTE
    nextCommand  :EnumTxpNextCommandMode [IMMEDIATELY | WHEN_COMMAND_DONE]
);
```

6.8.5 _tcpCloseConnection function

An active connection is closed by calling the function _tcpCloseConnection, which was previously established _tcpOpenServer and _tcpOpenClient and thus releases the occupied communication resources again.

The function may only be called in the BackgroundTask or in a MotionTask.

```
_tcpCloseConnection:DINT
    (
     connectionId:DINT
    );
```

6.8.6 _tcpCloseServer function

The _tcpCloseServer function terminates the waiting state for a connection request of a communication partner (client) started with _tcpOpenServer.

The function may only be called in the BackgroundTask or in a MotionTask.

```
_tcpCloseServer:DINT
   (
    port:UINT
   );
```

6.9 Details of the SIMOTION UDP system functions

6.9.1 Function _udpSend

Description

The _udpSend function sends a UDP (User Datagram Protocol) message frame to the receiver identified by the IP address and port number.

At least the following data is required for sending:

- IP address of communication partner
- "Own" port number
- Port number of communication partner

Syntax

```
_udpSend                :DINT
     (
     sourcePort          :UINT;
     destinationAddress  :ARRAY[0...3] of USINT;
     destinationPort     :UINT;
```

```
         communicationMode   :EnumUdpCommunicationMode
                              [ CLOSE_ON_EXIT |
                             DO_NOT_CLOSE_ON_EXIT ]
                              (default setting: DO_NOT_CLOSE_ON_EXIT );
           dataLength         : UDINT;
           data               : ARRAY  of Byte;
        )
```

- The command is synchronous concerning the data transfer at the port, but not for the communication.
- UDP is not a secured transfer protocol. You must program a feedback concerning the success of the data transfer in the user program yourself.

For a detailed description of the transfer parameters, please refer to the SIMOTION system documentation.

6.9.2 Function _udpReceive

Description

The _udpReceive function receives a UDP message frame at a port specified via a transfer parameter.

Syntax

```
_udpReceive    :StructRetUdpReceive
    (
    port              :UINT; // (specification of the port to be read)
    communicationMode :EnumUdpCommunicationMode (cf. _readRecord)
                       [ CLOSE_ON_EXIT |
                      DO_NOT_CLOSE_ON_EXIT ]
                       (default setting: DO_NOT_CLOSE_ON_EXIT);
     nextCommand        :EnumNextCommandMode
                        [ IMMEDIATELY |
                        WHEN_COMMAND_DONE |
                        ABORT_CURRENT_COMMAND ]
                        (default setting: IMMEDIATELY  );
     receiveVariable    :ARRAY of BYTE;
    )

    StructRetUdpReceive
        functionResult  :DINT;
        sourceAddress   :ARRAY[0...3] of USINT;
        sourcePort      :UINT;
        dataLength      :UDINT;
    END_STRUCT;
```

The data is returned in the variable specified in "receiveVariable".

- You do not have to specify any commandID in the function, since the status of the data transfer can be queried via the port.
- A call to the UDP functions from the IPO synchronous task should be avoided, in order to prevent level overflow in case the IPO cycle has not been set too generously.

For a detailed description of the transfer parameters, please refer to the SIMOTION system documentation.

PROFINET IO 7

7.1 PROFINET IO overview

7.1.1 PROFINET IO

In machine construction, there is a clear trend toward distributed machine concepts and mechatronic solutions. This increases the demands on the drive networking. A large number of drives and shorter cycle times as well as the use of IT mechanisms are increasingly gaining in importance.

The two successful solutions, PROFIBUS DP and Ethernet, are combined under PROFINET IO. PROFINET IO is based on 15 years of experience with the successful PROFIBUS DP and combines the normal user operations with the simultaneous use of innovative concepts of the Ethernet technology. This ensures the smooth migration of PROFIBUS DP into the PROFINET world.

PROFIBUS DP is a bus system where only one node can have "send" access to the bus at any one time (half-duplex operation). PROFINET IO uses the switching technology which is also found with Ethernet. This involves separating all the network segments, thereby enabling sending and receiving to be performed on all lines at the same time (full-duplex operation). In this way, the network can be used much more efficiently through the simultaneous data transfer of several nodes. The bandwidth has also been increased to 100 Mbps.

More information

Detailed descriptions on the subject of PROFINET can be found in the *SIMATIC PROFINET System Description* System Manual.

7.1.2 Application model

During the development of PROFINET IO, special emphasis was placed on the protection of investment for users and device manufacturers. The application model is retained for the migration to PROFINET IO. Compared with PROFIBUS DP, the process data view remains unchanged for:

- I/O data (access to the I/O data via logical addresses)
- Data records (storage of parameters and data) and
- Connection to a diagnostic system (reporting of diagnostic events, diagnostics buffer)

This means that the familiar view for access to the process data is used in the user program. Existing programming know-how can continue to be used. This also applies to device profiles, such as PROFIdrive, which is also available with PROFINET IO.

The engineering view also has a familiar "look and feel". The engineering of the distributed I/O is performed in the same way and with the same tools, as already used for PROFIBUS.

7.1.3 IO controller

The PROFINET IO controller has the same functions as the PROFIBUS DP master. The IO controller of, for example, a SIMOTION D with CBE30 (PROFINET IO expansion card) exchanges data cyclically with the I/O devices assigned to it (PROFINET IO devices), such as the SINAMICS S120.

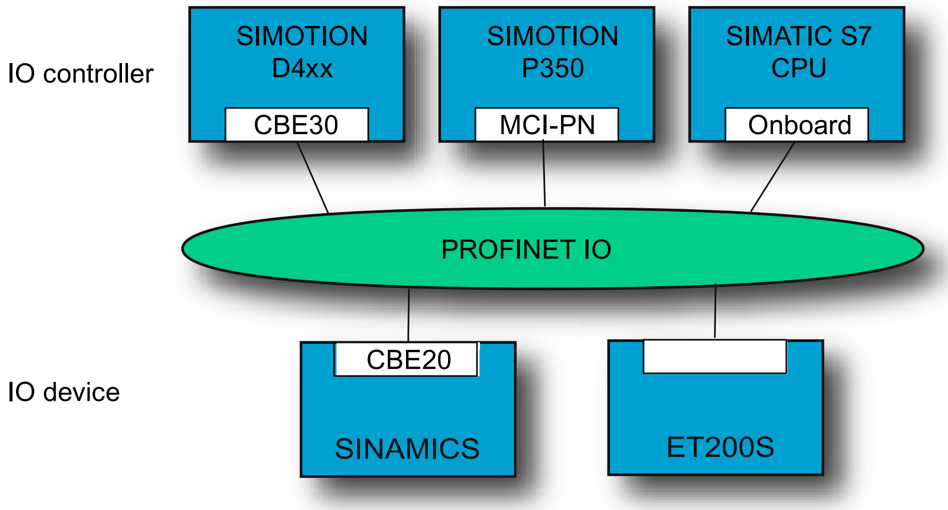


Figure 7-1 Examples of IO controllers and IO devices

7.1.4 IO device

Distributed field devices such as I/O components (e g. ET 200) or drives (e.g. SINAMICS S120 with CBE20-PN) are referred to as IO devices. The function is comparable to a PROFIBUS DP slave.

See also

Creating an IO device (Page 181)

7.1.5 PROFINET IO system

A PROFINET IO system consists of a controller and the devices assigned to it.

7.1.6 Sync domain

A sync domain is a group of PROFINET devices synchronized to a common cycle clock. The sync master sets the cycle clock. The sync slave synchronizes itself with the cycle clock set by the sync master. A sync domain has one sync master.

See also

Creating a sync domain (Page 171)

7.1.7 iDevice

The PROFINET I device functionality is comparable with that of the I slave for PROFIBUS, i.e. a SIMOTION CPU can accept the role of an IO device and so exchange data with a different IO controller.

Whereas with PROFIBUS an interface can be either a master or slave only, with PROFINET it is possible to be both an IO controller and IO device at the same time on a single PROFINET interface.

7.1.8 Addressing of PROFINET IO devices

A globally unique MAC (Media Access Control) address is used for data exchange via Ethernet and forms part of the Ethernet message frame. The MAC address is linked to the hardware and cannot be modified.

Ethernet-based protocols such as HTTP (Web applications) or FTP (file transfer) use the IP protocol. Addressing is based on the IP address. This is a logical address, which can be assigned by the user.

PROFINET uses a device name (NameOfStation) to identify PROFINET devices, in addition to the two items of address information already known in connection with Ethernet. The device name is a string that fulfills the requirements of a DNS (Domain Name Service) name. This device name, also known as the communication name, must be unique across the PROFINET network.

During the commissioning phase, each PROFINET device (identified via the MAC address) is assigned a device name once via the configuration tool and this is stored retentively in the PROFINET device (a process known as node initialization). A device is referenced in the configuration via the device name. If a device is replaced, e.g. because of a defect, the new device has another MAC address. If it is initialized with the same device name as the replaced device (e.g. by reconnecting a removable medium that stores the device name retentively), it can take over the function of the replaced device without any changes in the configuration.

Alternatively, the device can be initialized automatically by the controller on the basis of topology information. This is only possible if topology information (who is wired to whom) is stored in the engineering system. During ramp-up, the controller identifies the connected devices using the device names, before assigning the IP address defined in the engineering system to the devices. The station can then be accessed via IP services. The IP address can be taken from a configured sequence of numbers or configured individually.

A PROFINET device has the following addresses by which it can be addressed:

- MAC address (part of the Ethernet message frame, stored on the device, and cannot be modified)
- IP address (IP-based communication such as engineering access, must be assigned to all devices)
- Device name, communication name (devices identified by the controller during ramp-up)

See also

Assigning device names and IP addresses to IO devices (Page 187)

7.1.9 RT classes

7.1.9.1 RT classes for PROFINET IO

Description

PROFINET is based on the Ethernet standard. This means that all standard Ethernet-based protocols (e.g. HTTP, FTP, TCP, UDP, IP, etc.) can be transferred via the PROFINET network.

In addition to the protocols associated with the types of office-based applications known throughout the world, PROFINET offers two protocols (transmission modes) adapted to the requirements of the automation sector. These are PROFINET IO with RT and PROFINET IO with IRT.

Both these transmission modes are optimized for cyclic IO communication within a network involving small amounts of data.

RT

RT communication uses the option of prioritizing message frames (as described in the Ethernet standard). This mechanism is also used for Voice over IP, for example. For more detailed information, see PROFINET IO with RT (Page 139).

IRT

As far as PROFINET IO with IRT is concerned, a time-slot procedure at a level above Ethernet is used. This involves 2 slots, with IRT message frames being transmitted to the first, and RT and IP message frames being transmitted to the second. Under this arrangement, a transmission bandwidth guaranteed to cope with any load/overload situation is reserved for the IRT data. IRT requires devices to be synchronized so that all devices involved know when the time slot begins.

The IRT transmission mode makes a distinction between two RT classes: High Flexibility and High Performance.

IRT - High Flexibility

The IRT High Flexibility RT class corresponds to the IRT transmission mode which has just been described. A uniform IRT slot is defined for the entire network in the engineering system. For more detailed information, see PROFINET IO with IRT (High Flexibility) (Page 140).

IRT - High Performance

In addition to the bandwidth reservation, a schedule for the cyclic message frames is developed with consideration given to the topology. This makes it possible for the engineering system to determine the required bandwidth for each individual cable. As a result, the IRT time interval can be minimized and transmission optimized to a greater extent when compared with IRT High Flexibility.

As well as synchronizing the IRT transmission network, IRT High Performance enables the application (e.g. SIMOTION position controller and interpolator) to be synchronized in the devices (isochronous application). This mirrors the behavior of the isochronous PROFIBUS.

This is an essential requirement for closing control loops across the network and isochronous switching of inputs and outputs in the network.

For more detailed information, see PROFINET IO with IRT (High Performance) (Page 141).

Comparing RT and IRT

Table 7- 1 The major differences between RT and IRT

Property	RT	IRT (High Flexibility)	IRT (High Performance)
Real-time class	Real-time class 1	Real-time class 2	Real-time class 3
Transfer mode	Prioritization of cyclic RT data using Ethernet-Prio (VLAN tag)	Bandwidth reservation, i.e. reservation of a time range in which only cyclic IRT data (but no RT or IP message frames) is transmitted	Bandwidth reservation optimized by the engineering system on the basis of topology information
Determinism	Variance of the transmission duration for cyclic RT data using TCP/IP message frames	Guaranteed transmission of cyclic IRT data within the reserved IRT time interval	Transmission and receiving times for cyclic IRT data are precisely defined and guaranteed for all kinds of topologies.
Isochronous application	Not supported	Not supported	Supported
Hardware support using special Ethernet controller	No	Yes	Yes

7.1.9.2 Send clock and update time

Description

The PROFINET system makes a distinction between two cycle clocks known as the send clock and update time. The send clock is the basic cycle clock for cyclic communication. The update time indicates the cycle in which a device is supplied with data.

Send clock

This is the period between two successive intervals for IRT or RT communication. The send clock is the shortest possible transmit interval for exchanging data. The send clock therefore corresponds to the shortest possible update time. During this time, IRT data and non-IRT data (RT, TCP/IP) is transmitted. All devices within a sync domain work with the same send clock.

Update time

The update time can be configured separately for each IO device and determines the interval at which data is sent from the IO controller to the IO device (outputs) as well as from the IO device to the IO controller (inputs). The calculated/configured update times are always a multiple (2^n) of the send clock.

Relationship between the update time and send clock

The calculated update times are multiples (1, 2, 4, 8, ..., 512) of the send clock. The minimum possible update time thus depends on the minimum send clock for the IO controller that can be set and the efficiency of the IO controller and the IO device.

7.1.9.3 Adjustable send clocks and update times

Description

The table below describes the send clocks which can be set for SIMOTION devices with PROFINET IO, as well as the down-scalings which are dependent on them and can be set for IRT and RT. The adjustable send clocks are divided into two ranges: the "even" range and the "odd" range. Update times are obtained by multiplying down-scalings by the send clock.

Table 7- 2 Adjustable send clocks and update times

Send clock		Scaling (update time = scaling * send clock)	
		RT IRT High Flexibility	IRT High Performance
"Even" range	250, 500, 1,000 µs	1,2,4,8,16, 64,128,256,512	1 *Note 2)*
	2,000 µs	1,2,4,8,16,32,64,128,256	
	4,000 µs	1,2,4,8,16,32,64,128	
"Odd" range *Note 1*)	375, 625, 750, 875, 1,125, 1,250 µs ... 3,875 µs (increment 125 µs)	Not supported	1

If there is no sync master configured in a PROFINET IO system (no PROFINET IRT), the send clock for the PROFINET IO system concerned can be set on an individual basis at the relevant IO controller using the **<PROFINET interface>** properties on the PROFINET tab under the send clock or using the **PROFINET IO system** properties on the tab for the update time. There is a default setting for the send clock of 1 ms. On the IO cycle tab, the fixed factor or fixed update time can be set using the mode and the down-scaling for the update time can be set using the factor.

As soon as a sync master is configured in the PROFINET IO system, the send clock is defined under the sync domain properties. The controllers assigned to the sync domain accept this value. Update times can be set independently for each IO device.

Note 1) Mixed operation, RT/IRT High Performance
Odd send clocks can only be used if there is no RT or IRT High Flexibility IO device in the IO systems involved in the sync domain. If there are IO devices with RT class "RT" in a sync domain, it is only possible to set send clocks from the "even" range.

Note 2) Down-scaling and isochronous application
A number of IO devices support down-scalings of 2, 4, 8, and 16 with IRT High Performance, as well as a down-scaling of 1.

Where IO devices (e.g. ET 200S IM151-3 PN HS, SINAMICS S) are operated with an isochronous application, it is usually only possible to set a down-scaling of 1.
In these cases, the mode for the update time must always be set to **fixed factor** to ensure STEP 7 does not automatically adapt the update time to always match the send clock. Please refer to the image below.

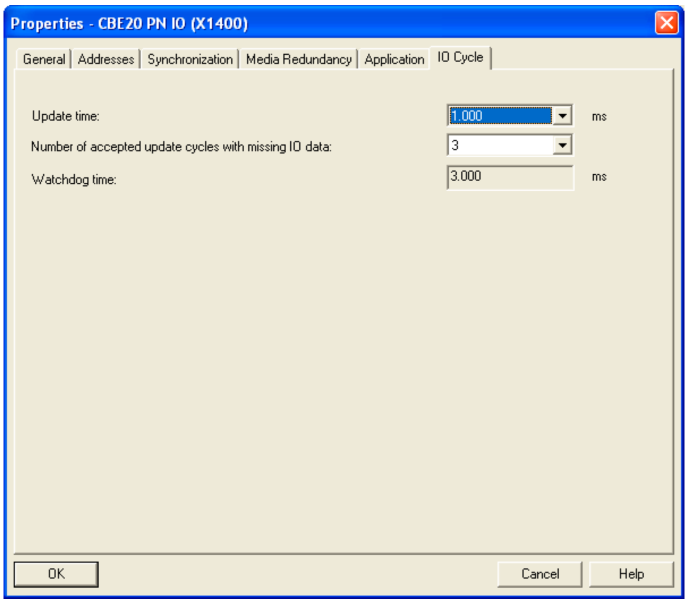
Send clock mode for update time

- Fixed factor,
  fixed send clock down-scaling for the update time
- Fixed update time,
  update time is set
- Automatic
  STEP 7 automatically adjusts the down-scaling if the one selected is too low

---

Note

It is recommended that you work with the **Fixed update time** setting.


Properties - CBE20 PN IO (X1400)

General | Addresses | Synchronization | Media Redundancy | Application | IO Cycle

Update time: 1.000 ms
Number of accepted update cycles with missing IO data: 3
Watchdog time: 3.000 ms

Figure 7-2 Send clocks

7.1.9.4 Setting RT classes

RT classes

The IO controller determines which RT class its IO system supports, by setting the real time class at its controller interface.

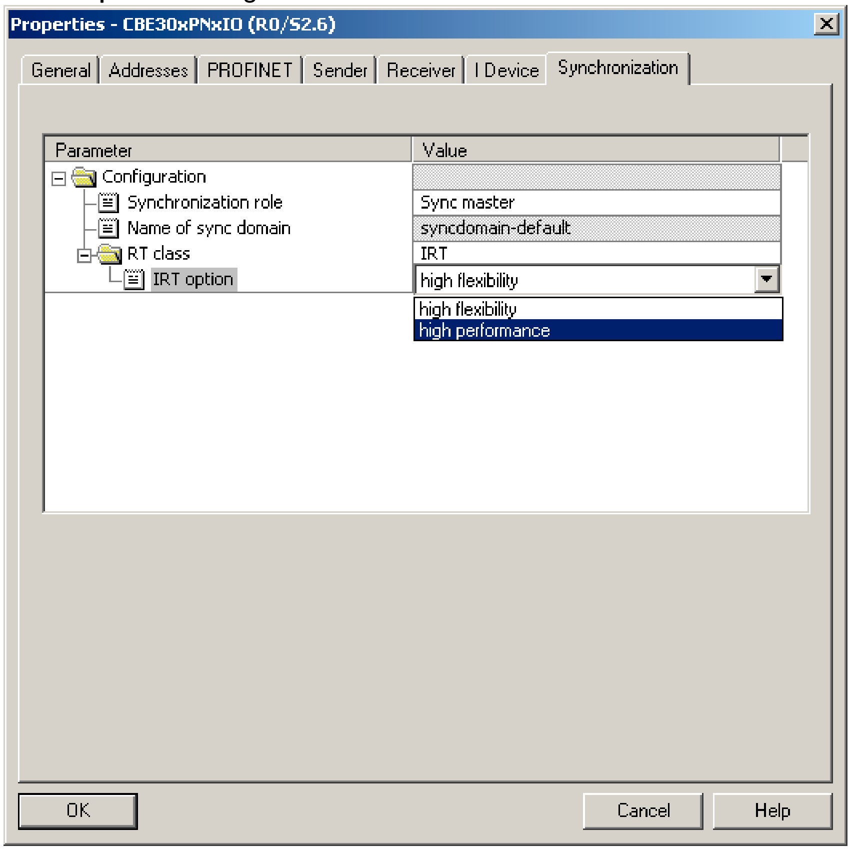
The IRT High Performance and IRT High Flexibility RT classes can be used in a network, although they are not suitable for use in an IO system. Mixed operation is **not** supported. RT devices can always be operated, even if IRT classes are set.

Setting the RT class

You can set the RT class in the HW Config for the associated PROFINET device.

1. In HW Config, double-click the PROFINET interface in the module.

   The **Properties** dialog box is called.



Properties - CBE30xPNxIO (R0/S2.6)

Parameter	Value
Configuration	
Synchronization role	Sync master
Name of sync domain	syncdomain-default
RT class	IRT
IRT option	high flexibility

high flexibility
high performance

2. Select the realtime class for RT class in the **Synchronization** tab.

3. **High Flexibility** and **High Performance** can be selected as options.
4. Click **OK** to confirm.

Note

Only IRT High Performance is used for motion control applications with SIMOTION and SINAMICS.

7.1.9.5 PROFINET IO with RT

PROFINET IO with RT is the optimal solution for the integration of I/O systems without particular requirements in terms of performance and isochronous mode. This is a solution that uses standard Ethernet IC (Ethernet Controller) and commercially available industrial switches as infrastructure components. A special hardware support is not required.

Not isochronous

Although standard Ethernet and PROFINET IO with RT do not offer any synchronization mechanisms for devices, this does not mean that such arrangements are impossible. However, it does mean that isochronous data transmission is impossible, and there is no isochronous application for motion control as a result.

Data exchange

Communication via PROFINET IO with RT and IRT is based on the Ethernet frame and the MAC address. This means that cross-network communication via a router using RT and IRT is not possible. PROFINET IO message frames have priority over IT message frames in accordance with IEEE802.1Q. This ensures the availability of the real-time properties required in automation applications (e.g. for standard IOs).

Update time

The adjustable update time is in the range of 0.25 - 512 ms. The selected update time depends on the process requirements, the number of devices, and the amount of IO data. Given the improved performance offered by PROFINET compared to field buses, the bus cycle is generally no longer the variable which determines the system cycle.

7.1.9.6 PROFINET IO with IRT - Overview

Overview

PROFINET IO with IRT satisfies communication requirements which go beyond the sending of standard signals. As far as IRT is concerned, the jitters which may still be encountered with RT during communication are significantly reduced by synchronizing the network.

A time-slot procedure is required at a level above that of the Ethernet network. One time slot is reserved for the IRT message frames and another for the RT and IP-based message frames. This type of approach requires all devices involved in IRT communication to be synchronized.

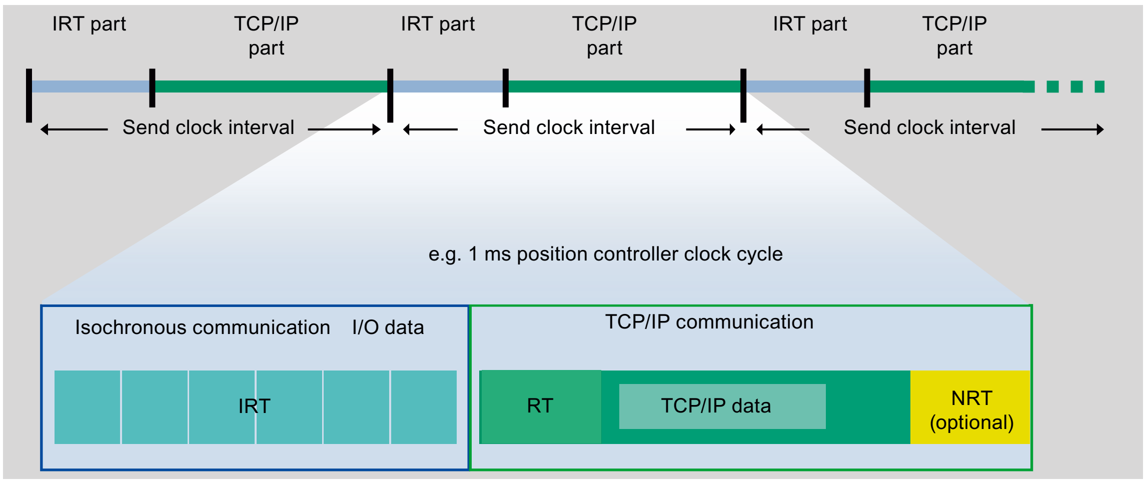


Figure 7-3 IRT Communication - Overview

PROFINET IO with IRT is available in two versions:

- IRT High Flexibility (Page 140) with fixed bandwidth reservation.
- IRT High Performance (Page 141) with optimized bandwidth reservation and scheduled IRT communication

For PROFINET IO with IRT, all IRT devices are synchronized on a shared sync master. See also Isochronous operation and isochronous mode with PROFINET (Page 145).

7.1.9.7 PROFINET IO with IRT (High Flexibility)

Description

For PROFINET IO with IRT (High Flexibility), the largest IRT bandwidth requirement of a device (typically a controller) is determined, and an across-the-board bandwidth is reserved for the entire IRT network. All IRT devices must be synchronized on a shared sync master.

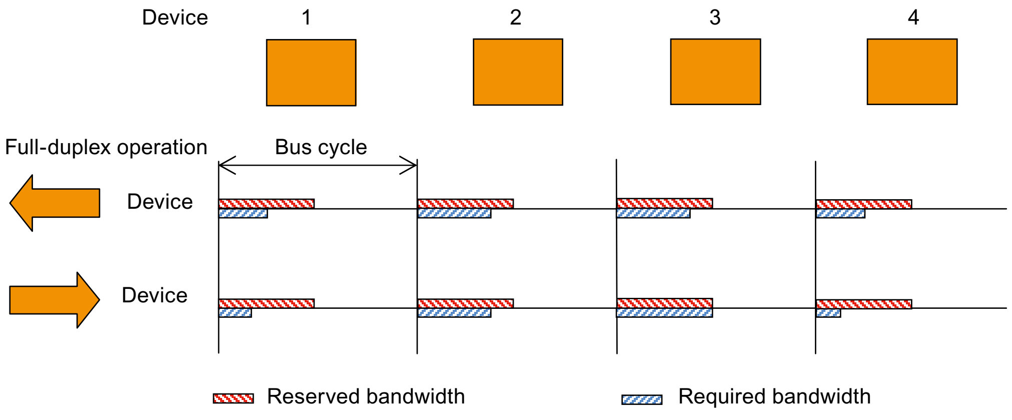


Figure 7-4 Overview of communication with IRT (High Flexibility)

Send clock

For mixed operation using RT and IRT High Flexibility, or where only IRT High Flexibility is used in conjunction with a SIMATIC CPU, the send clock can only have values of 0.25 (319PN only), 0.5, or 1.0. With SIMOTION CPUs, values of 0.5 or 1.0 can be set as the send clock for IRT High Flexibility.

7.1.9.8 PROFINET IO with IRT (High Performance)

The performance capability of motion control applications is significantly increased with PROFINET IO IRT (High Performance). If field buses such as PROFIBUS are used, devices are connected in parallel to the bus. This has a number of consequences. Firstly, only 1 device can transmit at any one time. Secondly, the parallel connection of all devices means that the physical limit is reached at approximately 12 Mbps.

PROFINET is based on Ethernet technology, which is in turn based on point-to-point connections. Point-to-point connections support a significantly higher transmission rate when compared with arrangements based on parallel wiring. PROFINET uses 100 Mbps. Using this in conjunction with switching technology means that all connecting cables are decoupled from each other, enabling each cable to both transmit and receive at the same time.

Scheduling the message frame traffic for IRT High Performance enables data traffic to be optimized to a considerably higher degree when compared with IRT High Flexibility, as only the bandwidth which is actually required is reserved.

IRT (High Performance) is particularly suitable for:

- The control and synchronization of axes via PROFINET IO
- A fast, isochronous I/O integration with short terminal-terminal times

Send clock

The send clock can be set between 250 µs (P350) and 4 ms for PROFINET IRT High Performance. In mixed operation involving RT and IRT High Performance, only values of 0.5, 1.0, 2.0, or 4.0 can be set.

The actual send clock used depends on various factors:

- The process; communication should be no faster than required. This reduces the bus and CPU loads.
- The bus load (number of devices and the amount of IO data per device)
- Computing power available in the controller
- Supported send clocks in the participating PROFINET devices of a sync domain

A typical send clock is, for example, 1 ms. However, it can be set in a 125 µs grid within the limits of 250 µs to 4 ms. See also Adjustable send clocks and update times (Page 136).

The supported send clocks can be found in the corresponding manuals of the respective SIMOTION devices. A minimum cycle time of 250 µs is only supported by selected components (SIMOTION P350-3 and ET 200S HS modules).

Isochronous application

Isochronous data transmission and an application synchronized with the bus system satisfy the requirements associated with demanding motion control applications. This makes it possible to close control loops via the bus system and achieve minimum guaranteed response times (terminal-to-terminal time response). In addition, a high-performance and isochronous connection to the application with low load on the application CPU is ensured.

In contrast to standard Ethernet, PROFINET IO with RT, and PROFINET IO with IRT High Flexibility, the transmission of message frames for PROFINET IO with IRT High Performance is scheduled.

Time-scheduled data transmission

Scheduling is the specification of the communication paths and the exact transmission times for the data to be transferred. The bandwidth can be optimally utilized through communication scheduling and therefore the best possible performance achieved. This requires the network topology to be configured, with the engineering system automatically calculating the communication schedule from the configured topology (see also Topology (Page 143)). The data relevant to PROFINET IO is transmitted by means of a download from HW Config to the IO controller. The highest determinism quality is achieved through the scheduling of the transmission times which is especially advantageous for an isochronous application connection.

Data exchange

PROFINET IO with IRT High Performance and PROFINET IO with IRT High Flexibility only run within a sync domain. They must not, however, be mixed in an IO system. In other words, a sync domain can consist of 2 or more IO systems which can all be synchronized with each other. Within the IO system, either IRT High Flexibility or IRT High Performance is used.

7.1.10 Topology structure

General

Below, you will find an overview of various options for setting up a PROFINET network with SIMOTION.

Table 7- 3 Possible topology for SIMOTION

Topology	
Star	If you connect communication nodes to a switch, you automatically create a star-shaped network topology. If an individual PROFINET device fails, this does not automatically lead to failure of the entire network, in contrast to other structures. Only the failure of a switch causes the failure of devices downstream of the switch.
Tree	If you interconnect several star-shaped structures, you obtain a tree network topology.
Linear	All the communication nodes are connected in series as a bus. If a switch fails, communication downstream of the failed switch is no longer possible. All devices (apart from 300 SIMATIC CPUs) have an integrated 2-port switch to support implementation of the linear topology. Linear network structures require the least amount of cabling.

Production topology examples

The following example shows various topologies combined.

General installation information

PROFINET allows you to set up communication with both high-performance and a high degree of uniformity. We recommend you observe the following installation guidelines:

1. Connect a router or a SCALANCE S between the office network and PROFINET system. You can use the router to prevent third-party access or unauthorized access to the production network's PROFINET system.
2. Set up your PROFINET in a point-to-point architecture where this is useful (for example, use a switch to branch off into a point-to-point topology downstream of a CPU.
3. In the case of PROFINET with IRT, a line structure with 64 IRT devices is permissible. Dependencies exist between the amounts of data transmitted: If longer message frame lengths are configured for each device, the possible number of devices per line may be reduced. However, this is detected early on during configuration with HW Config and signaled by means of an error message. The 64 IRT devices in the line only apply to PROFINET V2.2.

4. Keep the interconnection depth of the switches as low as possible. This will reduce the effect of a worst-case jitter scenario with RT communication. With IRT High Flexibility, the bandwidth requirement determined by the engineering software is reduced.
5. Devices with a high IP load on the network should be located, where possible, in a separate area of the network. The diagnostics server and HMI server are two such devices.

Sample topology - company network - production network

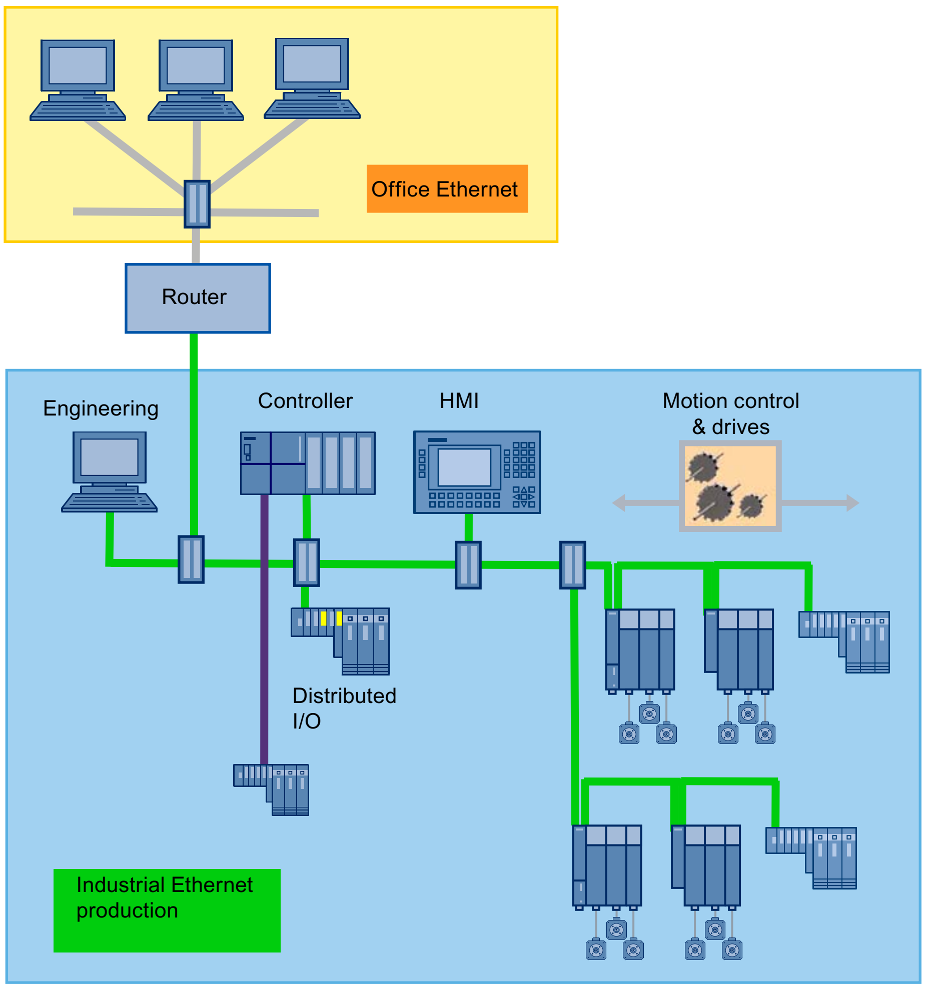


Figure 7-5 Example of a topology

Note

Further information on commissioning as well as on the topology structure can be found in the SIMOTION commissioning manuals of SIMOTION D and SIMOTION P.

Configuration

Prerequisite for the communication scheduling is knowledge of the network topology. This includes information on the interconnection of the individual devices to form a communication network.

The topology scheduling is only relevant for IRT High Performance. The network topology can be configured for ease of use with the aid of a topology editor integrated in the hardware configuration.

7.1.11 Isochronous operation and isochronous mode with PROFINET

Description

PROFINET IO with IRT is based on Ethernet with a higher-level time-slot procedure. This arrangement requires all bus interfaces involved in communication to be synchronized. In PROFINET IO with IRT High Performance and IRT High Flexibility, a sync master generates a synchronization message to which all sync slaves synchronize themselves.

Sync master, slaves, and domain

The sync master and sync slave device roles are assigned during the configuration. An IO controller or SCALANCE 200 IRT switches can be assigned a sync master role.

A sync domain can consist of both PROFINET devices with IRT High Performance and PROFINET devices with IRT High Flexibility. PROFINET devices with RT may also be located at the ends (spur lines), but not between two PROFINET IO devices with IRT (High Flexibility or High Performance).

A line (network) must not contain a mixture of IRT High Flexibility or IRT High Performance, as these must always be connected to each other directly.

Compatibility

Communication between and through different sync domains via PROFINET IO with RT is possible.

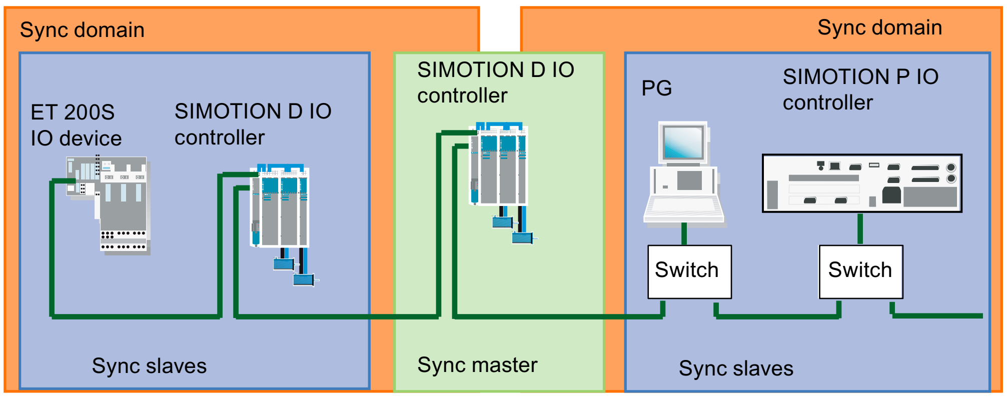


Figure 7-6 PROFINET isochronous mode

IRT-compatible switches, such as SCALANCE X204 IRT, are used in the structure. All devices involved in IRT communication are connected to each other directly. As the programming device in the center of the network is connected via a spur line, it does not interrupt the IRT path.

See also

Isochronous applications with PROFINET (Page 146)

7.1.12 Isochronous applications with PROFINET

As with PROFIBUS, in the case of PROFINET IO with IRT High Performance the application can be synchronized with the transmission network's cycle clock. Distributed motion control applications and terminal-to-terminal response times of less than a millisecond require all PROFINET devices with IRT High Performance to be synchronized with a common time base.

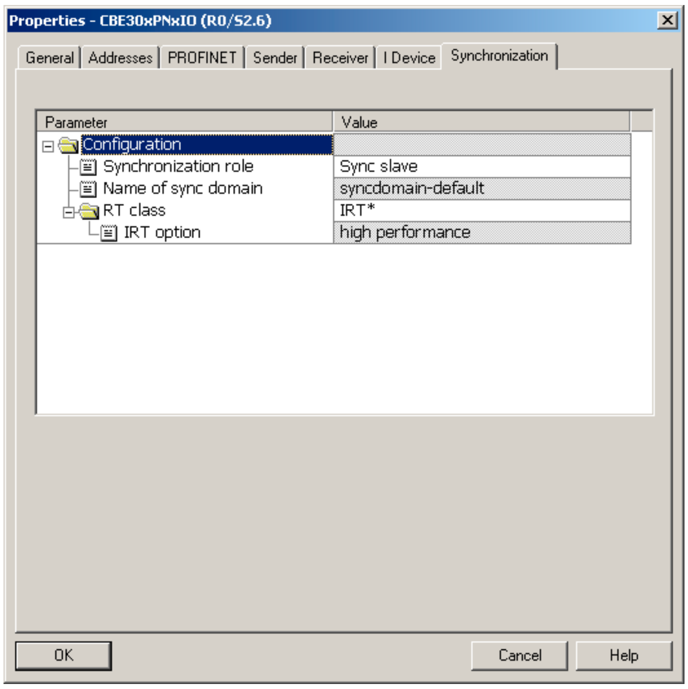
Note

Isochronous mode for the application on the bus is only possible for PROFINET IO with IRT High Performance.

Procedure

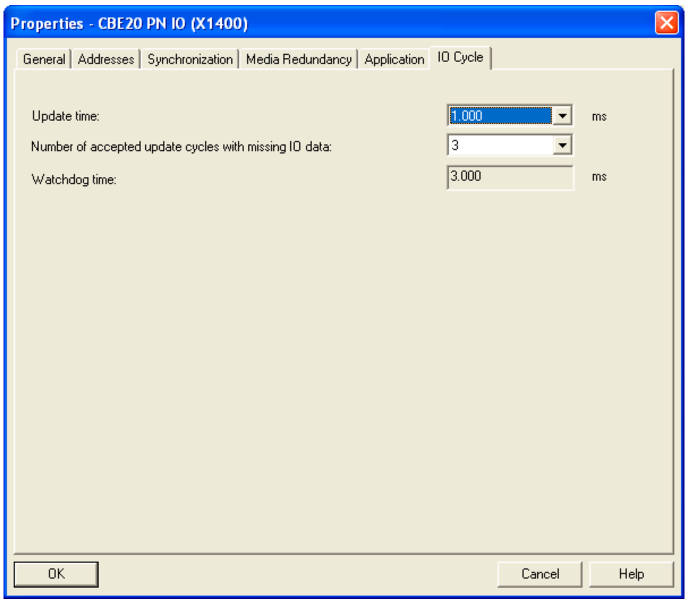
When configuring isochronous applications, proceed as follows:

1. Set "IRT High Performance" on the controller and devices.



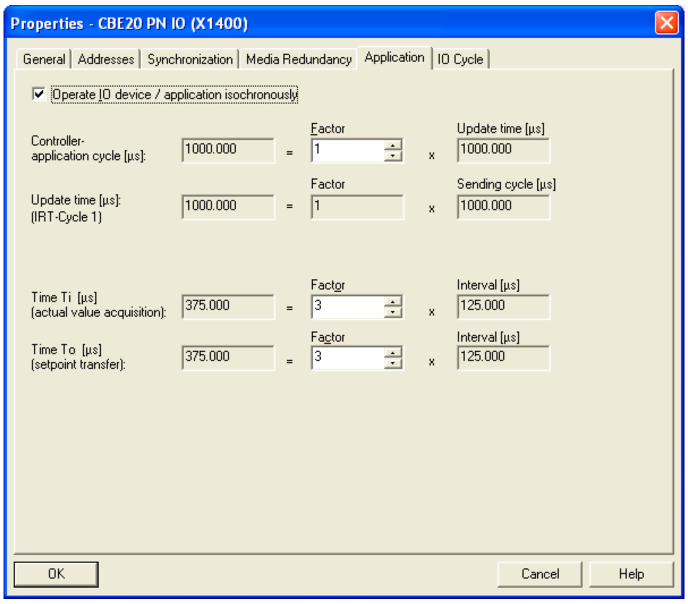
Parameter	Value
Configuration	
Synchronization role	Sync slave
Name of sync domain	syncdomain-default
RT class	IRT*
IRT option	high performance

2. Set the update time mode on the devices to a fixed factor.



Update time:	1.000	ms
Number of accepted update cycles with missing IO data:	3	
Watchdog time:	3.000	ms

3. Activate the "Operate IO device/application in isochronous mode" checkbox on the devices.


Properties - CBE20 PN IO (X1400)

General | Addresses | Synchronization | Media Redundancy | Application | IO Cycle

☑ Operate IO device / application isochronously

			Factor		
Controller-application cycle [µs]:	1000.000	=	1	x	Update time [µs]: 1000.000
Update time [µs]: (IRT-Cycle 1)	1000.000	=	1	x	Sending cycle [µs]: 1000.000
Time Ti [µs] (actual value acquisition):	375.000	=	3	x	Interval [µs]: 125.000
Time To [µs] (setpoint transfer):	375.000	=	3	x	Interval [µs]: 125.000

7.1.13 Cycle clock scaling

7.1.13.1 Cycle clock scaling with PROFINET IO on SIMOTION devices

Description (PROFINET IO with IRT High Performance)

An isochronous application (e.g. position controller) on an IO controller must be synchronized with the send clock for IRT High Performance. It can, however, be synchronized with a multiple of the send clock of the data. This multiple is designated as CACF (Controller Application Cycle Factor). The clock cycle scaling is set on the servo for PROFIBUS (set system clocks).

Example: The data on the network is transferred with a send clock time of 1 ms. However, because the servo should run with 2 msec, Therefore, the CACF must be equal to 2 and set on the corresponding drive.

Note

Isochronous mode is not possible for PROFINET IO with IRT High Flexibility.

The CACF is set on the IO device, see e.g. Inserting and configuring the SINAMICS S120 (Page 182).

Description

Scaling to the send clock with SIMOTION controllers is possible in the case of PROFINET with IRT High Performance under the following conditions:

- The SINAMICS Integrated of a D4xx and an isochronous DP master interface always run simultaneously to the servo cycle clock.
- For a SIMOTION P350 the isochronous DP master interface always runs simultaneously with the servo cycle clock.
- When a SIMOTION D drive, using SINAMICS Integrated, runs down-scaled to the send clock in the position control cycle clock, the position control cycle clock can extend across n send clocks (n = down-scaling of the position control cycle clock to the send clock) and must not be completely processed within the send clock.
- For drives which are not connected via SINAMICS Integrated (e.g. a SINAMICS S120 used as an external drive unit), the position control cycle clock must **always** be counted in the first send clock. This means that down-scaling is possible, although the position controller must be counted in the first send clock.

The following general conditions apply to cycle clocks and cycle clock scalings for a SIMOTION controller

- If IRT High Performance is configured for a SIMOTION device, but the SIMOTION device itself neither sends nor receives any IRT data (e.g. there is only a router for IRT data), then the position control cycle clock will not be synchronized with the send clock. Only the PROFINET interface is synchronized with the send clock. For example:
  - Only TCP/IP via PROFINET interface
  - Only RT devices on the PROFINET interface
  - PROFINET interface, only *router* for IRT High Performance data to other devices

Note

Only the SIMOTION device acting as the sync master synchronizes itself automatically with the bus. If other SIMOTION devices are configured as sync slaves, the device will have to be synchronized by means of the application. This application is programmed in the StartupTask via the _enableDPInterfaceSynchronizationMode command.

Combinations of cycle clocks and cycle clock sources for PROFIBUS and PROFINET IO

- Servo can be scaled in integral multiples (1, 2, ...n) of the send clock.
- Cycle clocks for SINAMICS Integrated and isochronous DP master interfaces must run simultaneously with the servo cycle clock.

7.1.13.2 Cycle clock scaling for IO accesses

Description of PROFINET IRT data transmission

The following must be observed for cycle clock scaling (PROFINET and PROFIBUS):

- PROFINET IO IRT data is always read at the beginning of the position control cycle clock and written at the end of the position control cycle clock.
- PROFINET IO RT data is read at the beginning of the IPO cycle clock and written at the end of the IPO cycle clock.
- At the end of a IPOSynchronousTask, the process image is output with the next possible servo (Data Out) (= response-time-optimized). If the position control cycle clock is down-scaled to the IPO cycle clock (position control < IPO), this can lead to the data being output one or more position control cycle clocks earlier or later within an IPO cycle clock, if the I/O accesses are performed via the IPOSynchronousTask. This is the case if the runtime for the IPO cycle clock is not constant and, as a result, data is transmitted earlier or later on the bus with a faster position control cycle clock.
- At the end of the position control execution level, the process image of the ServoSynchronousTask is output with the next possible bus cycle clock (= response-time-optimized).
- As far as PROFINET is concerned, and where the PROFINET cycle clock is down-scaled to the position control cycle clock (PROFINET < position control), this can lead to data being output one or more bus cycle clocks earlier or later within a position control cycle clock, if the I/O accesses are performed via the ServoSynchronousTask and the runtime of the position-control execution level fluctuates over one bus cycle clock.
- For PROFIBUS, the data is always output with the first bus clock cycle, since the servo priority class must always be finished with the first bus clock cycle. In case of a different runtime of the servo priority class in the individual cycles, the terminal-terminal time may vary as a result.

If an always constant response time is to be achieved instead of a response-time-optimized behavior, the following must be set:

- For PROFIBUS:
  - A reduction ratio servo: IPO = 1 : 1 so that the I/O accesses from the IPOSynchronousTask are always implemented in isochronous mode.
  - Comment: IO accesses from the ServoSynchronousTask are always isochronous for PROFIBUS
- For PROFINET:
  - A reduction ratio bus clock cycle: Servo: IPO = 1 : 1 : 1 so that the I/O accesses from the IPOSynchronousTask are always implemented in isochronous mode
  - A reduction ratio bus clock cycle: servo = 1 : 1 so that the I/O accesses from the ServoSynchronousTask are always implemented in isochronous mode

7.1.13.3 Bus cycle clocks that can be adjusted for cycle clock scaling to SIMOTION devices

Overview of the possible bus cycle clocks

	PROFIBUS	PROFINET IRT High Performance	PROFINET IRT High Performance	Servo
	Minimum	Minimum	Maximum	Minimum
SINAMICS S120 CU320	1 ms	0.5 ms	4.0 ms	0.5 ms
SINAMICS S120 CU310	1 ms	0.5 ms	4.0 ms	0.5 ms
C230-2	1.5 ms	-	-	1.5 ms
C240 PN	1 ms	0.5 ms	4.0 ms	0.5 ms
C240	1 ms	-	-	0.5 ms
D410 PN	-	0.5 ms	4.0 ms	2.0 ms
D410 DP	1 ms	-	-	2.0 ms
D425	1 ms	0.5 ms	4.0 ms	1.0 ms
D435	1 ms	0.5 ms	4.0 ms	1.0 ms
D445/D445-1	1 ms	0.5 ms	4.0 ms	0.5 ms
P350-3	1 ms	0.25 ms	4.0 ms	0.25 ms
ET 200S HS	-	0.25 ms	4.0 ms	0.25 ms

Cycle clock scaling with PROFINET IO

Task	Servo		IPO		IPO2	
	Min	Max	Min	Max	Min	Max
Cycle clock	1 x bus	16 x bus	1 x servo	6 x servo	2 x IPO	64 x IPO

7.1.14 Task system and time response

7.1.14.1 Overview of SIMOTION task system and system cycle clocks

Overview

If IRT data is transmitted via the bus using PROFINET IO, the cycle clock execution times fall between reading and writing the data (e.g. axis data), depending on which task in the execution system the application is executed in. You can find examples of applications in different tasks (execution levels) in the chapters that follow.

7.1.14.2 BackgroundTask, MotionTask, and IPOSynchronousTask

MotionTask/BackgroundTask

The data is transmitted via the bus using PROFINET IO with IRT High Performance, and accepted by the communication interface at the start of the position control cycle clock. The logic signals are generally evaluated in a MotionTask or BackgroundTask. Here, a distinction is made as to which machine function is activated; for example, "position-controlled traversing of axis". The traversing profile required is counted in the next IPO cycle clock. Based on the position setpoints determined here, the speed setpoints for controlling the axis are calculated in the next position control cycle clock. These are transmitted to the drive in the next cycle, via PROFINET IO with IRT High Performance.

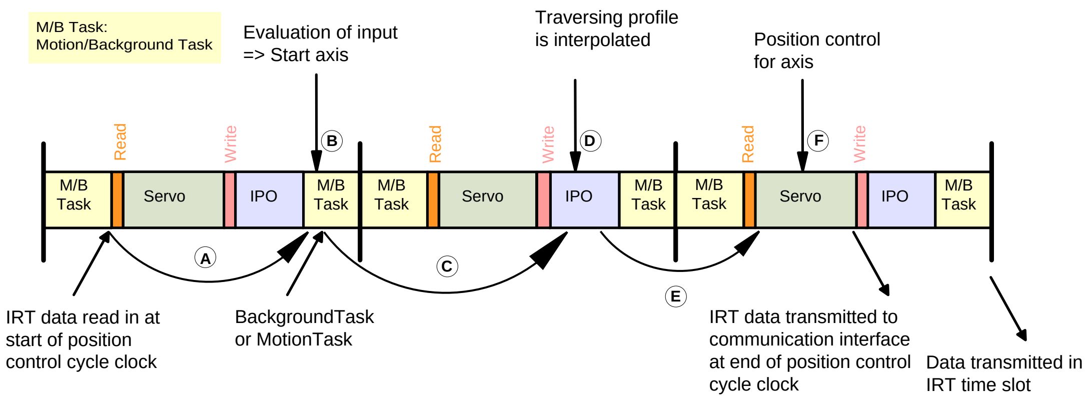

M/B Task:
Motion/Background Task
Evaluation of input
=> Start axis
Traversing profile
is interpolated
Position control
for axis
Read
Write
B
D
F
M/B Task
Servo
IPO
A
C
E
IRT data read in at start of position control cycle clock
BackgroundTask or MotionTask
IRT data transmitted to communication interface at end of position control cycle clock
Data transmitted in IRT time slot

Figure 7-7 Logic evaluation for an axis in the BackgroundTask or MotionTask

Boundary conditions

The bus cycle clock, position control cycle clock, and IPO cycle clock have a 1:1:1 ratio. Other ratios may result in longer response times. With a 1:1:2 ratio, the IPO execution may be extended to two position control cycle clocks, which can lead to the response time increasing by one position control cycle clock.

Additionally, the BackgroundTask may be processed over several position control cycle clocks, meaning that the system is unable to make sure the data is evaluated in the first position control cycle clock. This may also lead to an increased response time.

Assigning the variables to a process image has an impact on the response time as well. The process images are made available to other tasks or to the communication interface at the end of the respective task, rather than after the variables are updated.

IPOSynchronousTask

In order to optimize the time response and enable synchronous triggering of actions (e.g. starting axes simultaneously), in the IPOSynchronousTask it is possible to process the part of the application that triggers axis commands. If this option is used, it is counted before the IPO. In this way, the axis command can be issued before the IPO is executed, and the resulting position setpoint then calculated in the IPO. Based on this, the speed setpoint for the drive is calculated in the next position control cycle clock. Once the position control cycle clock has finished, the data is passed on to the communication interface and transmitted in the next PROFINET IRT send clock. Unlike processing in a MotionTask/BackgroundTask, where the response time equals the maximum BackgroundTask runtime + one IPO cycle clock + one position control cycle clock, in this case you can be assured that the response time will be one IPO cycle clock + one position control cycle clock until new data is output.

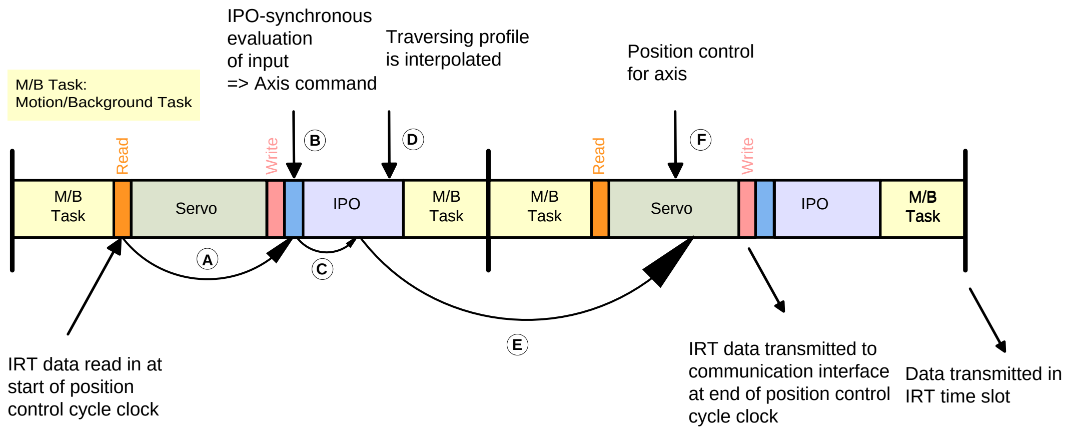

M/B Task:
Motion/Background Task
IPO-synchronous evaluation of input => Axis command
Traversing profile is interpolated
Position control for axis
Read
Write
B
D
F
M/B Task
Servo
IPO
A
C
E
IRT data read in at start of position control cycle clock
IRT data transmitted to communication interface at end of position control cycle clock
Data transmitted in IRT time slot

Figure 7-8 Logic evaluation for an axis in the IPOSynchronousTask

7.1.14.3 ServoSynchronousTask

ServoSynchronousTask

It is possible to optimize the time response even further and reduce the response time to one position control cycle clock. This option can be used for high-speed actual-value synchronous operations, e.g. flying knife/shear. Within this context, the part of the application that triggers axis commands for selected axes is processed in the ServoSynchronousTask. Additionally, the IPO part of the system for the axes involved is counted before the position controller in the servo task. In this way, the speed setpoints may be transmitted as soon as the next IRT time slot.

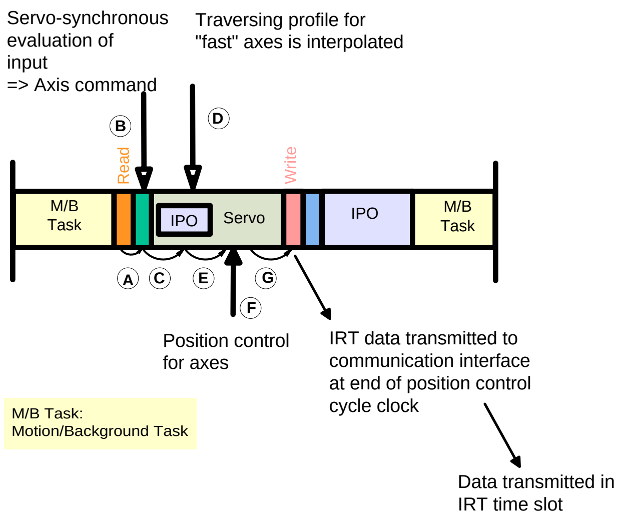

Servo-synchronous evaluation of input => Axis command
Traversing profile for "fast" axes is interpolated
B
D
Read
Write
M/B Task
IPO
Servo
IPO
M/B Task
A
C
E
G
F
Position control for axes
IRT data transmitted to communication interface at end of position control cycle clock
M/B Task: Motion/Background Task
Data transmitted in IRT time slot

Figure 7-9 Logic evaluation for an axis in the ServoSynchronousTask

Boundary conditions

Using this function increases the CPU load and, therefore, the position control cycle clock; for this reason it should only be used when necessary.

Activating

This feature must be activated explicitly for the axes in SIMOTION SCOUT as part of axis configuration.

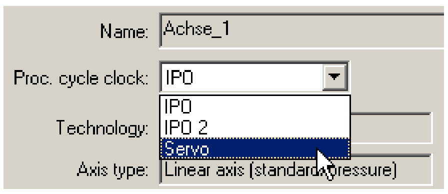

Name: Achse_1
Proc. cycle clock: IPO
IPO
IPO 2
Servo
Technology:
Axis type: Linear axis (standard pressure)

Figure 7-10 Determining the processing cycle clock for the axis

Position control

The position controller responds within one position control cycle clock. The data is read out from the communication interface at the start of the position control cycle clock. The position controller is counted in the position control cycle clock. The new speed setpoints are copied to the communication interface at the end of the position control cycle clock and, therefore, transmitted in the next IRT time slot.

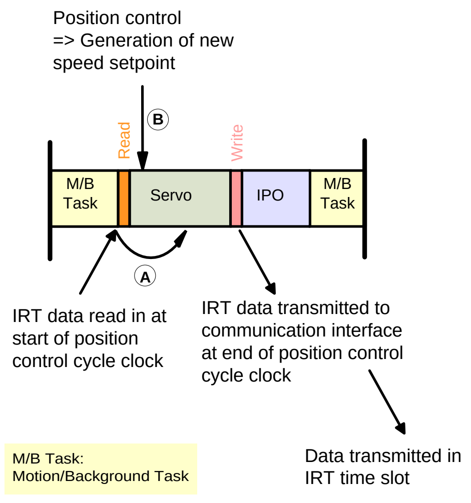

Position control
=> Generation of new
speed setpoint
B
Read
Write
M/B Task
Servo
IPO
M/B Task
A
IRT data read in at start of position control cycle clock
IRT data transmitted to communication interface at end of position control cycle clock
Data transmitted in IRT time slot
M/B Task:
Motion/Background Task

Figure 7-11 Position control time response with ServoSynchronousTask

7.1.14.4 Fast IOs in the ServoSynchronousTask

Fast IOs in the ServoSynchronousTask

Fast IOs (e.g. an ET200S HS) are evaluated in the ServoSynchronousTask, resulting in a system response time of one cycle. Due to the terminal-terminal response, there is a delay of Ti + position control cycle clock + To.

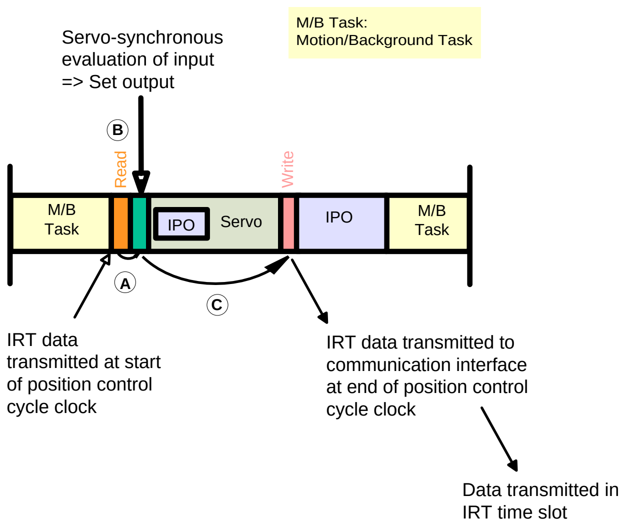

M/B Task:
Motion/Background Task
Servo-synchronous evaluation of input => Set output
B
Read
Write
M/B Task
IPO
Servo
IPO
M/B Task
A
C
IRT data transmitted at start of position control cycle clock
IRT data transmitted to communication interface at end of position control cycle clock
Data transmitted in IRT time slot

Figure 7-12 Fast IOs in the ServoSynchronousTask

7.1.15 Connection between sync domain and IO systems

The devices of several IO systems can be synchronized by a single sync master, provided they are connected to the same Ethernet subnet and belong to a sync domain.

Conversely, an IO system may only belong to a single sync domain.

7.1.16 Redundant sync master

Description

With certain systems (e.g. printing machines), it must be possible to operate system components on either a stand-alone basis or while connected to another component as part of a synchronous arrangement. If the system as a whole has only one sync master, the other component would not be capable of functioning independently. The "redundant sync master" function was developed for this very reason. Under this arrangement, a sync master is defined for every component. One of these is defined as the "sync master", and the other as the "sync master (redundant)". Provided the system components are combined during operation, the "sync master (redundant)" will synchronize itself with the sync master.

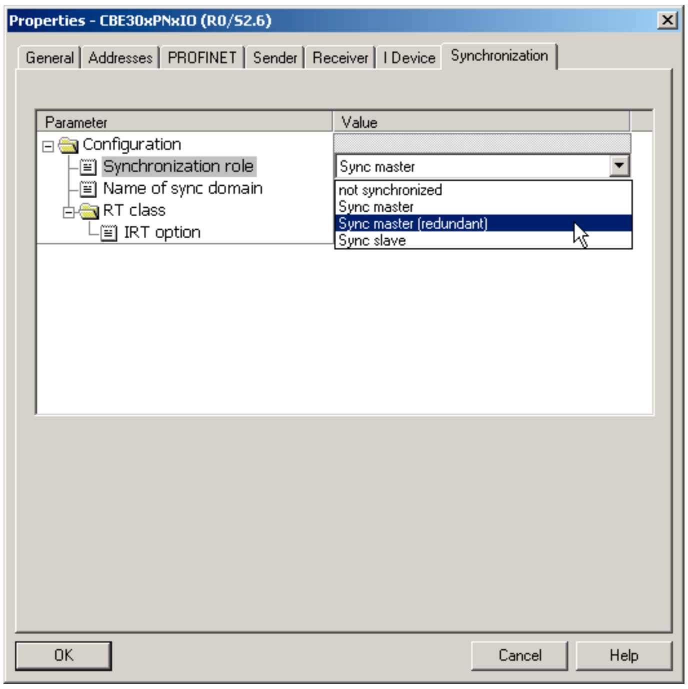
If the sync master ceases to function, the sync master (redundant) will continue operating independently and synchronize itself with its assigned sync slaves. The sync slaves assigned to the sync master will lose their synchronization and cease to operate.

Limitations of use

The two sync masters must be connected directly via a single cable with no switch. If the transmission link between the sync master and sync master (redundant) fails, leaving 2 subnets with one sync master each, both subnets remain synchronized with the remaining sync master in each case. This results in two independent synchronized subnets that drift apart due, among other things, to the temperature drift of the quartzes. Once the transmission link has been reestablished, there can be no smooth synchronization of the sync master (redundant) with the sync master, i.e. the drives assigned to the sync master (redundant) would lose synchronization and fail for a short time. Synchronization can only be reestablished after stopping the application.

Configuring the second sync master

1. Add a second SIMOTION module and configure PROFINET to satisfy your requirements.
2. Right-click with the mouse on the PROFINET board to open the **Properties - <PROFINET board> -- (R0/S2.6)** dialog.
3. Select the **Sync master (redundant)** entry under **Synchronization type** on the **Synchronization** tab.


Properties - CBE30xPNxIO (R0/S2.6)

General | Addresses | PROFINET | Sender | Receiver | I Device | Synchronization

Parameter	Value
Configuration	
Synchronization role	Sync master
Name of sync domain	
RT class	
IRT option	

Figure 7-13 Configuring the second sync master

7.1.17 Quantity structures

The following maximum values apply for IO controllers of the SIMOTION platform.

A maximum of 64 communication relationships are possible; these can be divided up as follows:

- Connection of up to 64 IO devices.
- Up to 64 RT or RT High Flexibility devices
- Up to 64 IRT High Performance devices
- Up to 64 controller-controller data exchange broadcast relationships may be set up between IO controllers.
- If a SIMOTION has been configured as an I device, it counts as a device; in other words, only another 63 devices can be connected.
- A SIMOTION device can receive data from up to 64 other SIMOTION devices and can send data to any number of SIMOTION devices.
- The amount of data involved must be taken into consideration where the data exchange broadcast is between controllers. A data exchange broadcast only counts as a connection if the amount of data involved does not exceed a certain level. Where a second message frame is required, the data exchange broadcast does not require two connections.

Mixed operation of IO devices and controller-controller data exchange broadcast

You can calculate the possible number of devices in mixed operation using the following formula:

IRT High Performance

```
RT + IRT High Performance IO device + data exchange broadcast frame
<= 64
```

IRT High Flexibility

```
RT + IRT High Performance IO device <= 64
```

Note

In a data exchange broadcast relationship, it is not the number of configured slots (lines in the lug receiver - see Configuring the receiver (Page 194)) that is intended for IRT High Performance data exchange broadcast configuration, but rather the number of Ethernet frames received for the data exchange broadcast. An Ethernet frame can contain up to 768 bytes of exchange broadcast useful data.

One slot has up to 254 bytes and 3,072 bytes of useful data can be exchanged during data exchange broadcast (divided into 4 frames of 768 bytes each). The value will depend on the sizes of the slot chosen. This means a transmitter can receive more than one slot, although these can be transmitted within a single message frame.

Each provider sends its exchange broadcast data in an Ethernet frame. Every other SIMOTION device can read the data in this frame. This means there is a counting connection to each transmitting SIMOTION.

Where data exchange broadcast is taking place between a controller and a device, the frame contains 1,440 bytes of useful data.

During the compilation of the project, HW Config verifies the configured quality structure based on the formulas mentioned above.

Address space

A maximum of 4 KB each may be assigned for PROFINET IO data for the input and output data in the logical address space of an IO controller. The rest of the 16 KB large address space can be used, for example, for PROFIBUS data or diagnostic data.

7.1.18 Acyclic communication via PROFINET

Description

Similarly to PROFIBUS DP, it is also possible for PROFINET IO to operate acyclic communication (Base Mode Parameter Access). You will find a detailed description hereof under DP V1 communication (Page 36).

7.2 Specific properties of PROFINET IO with SIMOTION

7.2.1 Introduction

Requirement

For it to be possible to work with SIMOTION using PROFINET IO, either the CBE30 option board must be inserted in the option slot of the SIMOTION D 4x5 devices, or a SIMOTION P350 PN, SIMOTION D410 PN, or C240 PN must be used.

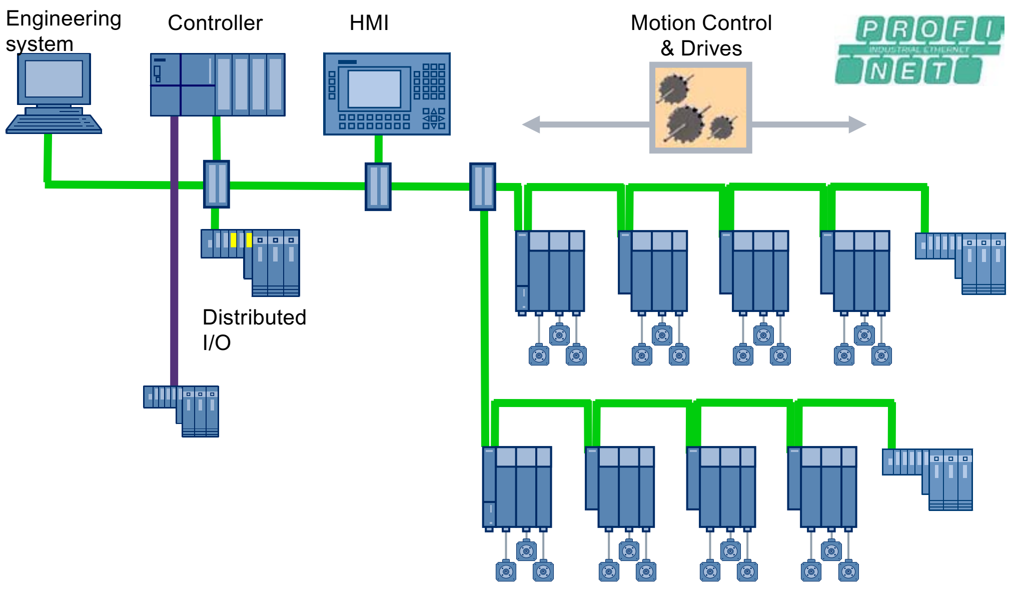


Figure 7-14 System topology with PROFINET

PROFINET devices support the simultaneous operation of:

- IRT High Performance - isochronous realtime Ethernet
  - Operation of IRT I/O (e.g. ET 200S HS for IRT High Performance)
  - Operation of a SINAMICS S120 as an IO device
  - Data exchange between controllers via IRT High Performance (e.g. distributed synchronous operation)
- RT - realtime Ethernet
  - Operation of RT - peripherals (e.g. ET 200S, ET 200pro)
  - ASi link via IE/AS interface link PN IO for the PROFINET IO gateway to AS interface
  - SINAMICS as PROFINET IO with an RT device
- TCP/IP, UDP, HTTP, … standard Ethernet services

Note

For mixed operation of IRT High Performance and RT, or IRT High Flexibility and RT, it must be ensured that the IRT-compatible (High Performance or High Flexibility) devices are directly connected with one another. In other words, there must not be any IRT-compatible device between the IRT devices.

Note

With SIMOTION SCOUT, it is possible to access a maximum of 10 PROFINET nodes ONLINE simultaneously. If you have installed SIMATIC NET, you will be able to establish more than 10 connections.

7.3 Configuring PROFINET IO with SIMOTION

7.3.1 New to SIMOTION V4.1.2

Description

A new synchronization process is supported with SIMOTION V4.1.2 and higher. This is defined in the PROFINET V2.2 specification. As of STEP7 5.4 SP4 and SCOUT V4.1.2, the PROFINET IRT previously described as "IRT top" is referred to as **IRT*** in HW Config.

PROFINET V2.1

IRT top (now IRT*) up to STEP7 5.4 SP3.1 and SCOUT < V4.1.2

PROFINET V2.2

IRT High Performance as of STEP7 5.4 SP4 and SCOUT V4.1.2

IRT High Flexibility as of STEP7 5.4 SP4 and SCOUT V4.1.2

Note

All IRT nodes must comply with either the PROFINET V2.1 or PROFINET V2.2 standard. Mixed configurations are not permissible. RT nodes are not affected by the switch to PROFINET V2.2.

The new hardware supplied as standard is PROFINET V2.2. If you are upgrading existing systems or downgrading new hardware, please get in touch with Siemens Support.

PROFINET V2.2 and higher offers both IRT variants (IRT High Performance and IRT High Flexibility), with the old IRT* (PROFINET V2.1) continuing to be supported.

Configuration overview for PROFINET V2.1 and V2.2 with STEP7 5.4 SP4 and SCOUT V4.1.2

Table 7- 4 PROFINET configuration

Configuration step	IRT according to PN V2.1	IRT according to PN V2.2
Configuration of the RT class in HW Config	IRT*	IRT with option: • High Performance • High Flexibility
Configuration of SINAMICS S120 via GSD	GSD V2.0, GSD V2.1, GSD V2.2	GSD V2.2
Configuration of SINAMICS S120 via DeviceOM	Supported	Supported
SIMOTION	-	4.1.2 and higher
SINAMICS	-	2.5.1.10 and higher
SCALANCE X200 IRT switch	FW 3.1.x	V4.1 and higher

7.3.2 PROFINET V2.2 and PROFINET V2.1

PROFINET V2.1 and V2.2

The standardization of PROFINET for IEC 61158 requires a transition from Version V2.1 to V2.2. The contents relating to PROFINET with IRT have changed.

When using the RT class "IRT" in the PROFINET network, versions V2.1 and V2.2 are not compatible. This means that when using PROFINET with IRT in both SIMOTION and SINAMICS, it is important to ensure that all devices that participate in IRT communication support the standard according to either V2.1 or V2.2. Communication with the real-time class "RT" remains compatible across the two PROFINET versions. All current versions of Siemens devices support PROFINET V2.2.

As of the following device versions, PNV2.2 is supported as the default setting:

- SIMOTION D4xx, P350-3, C240 PN >= V4.1.2
- SINAMICS S120 >= 2.6.1
- SCALANCE X200IRT >= V4.1
- ET200 HS >= V2.0
- SIMATIC NET CP1616/1604 >= 2.3

This list contains those devices that support IRT High Performance and whose previous versions are no longer compatible with respect to IRT.

PROFINET V2.2 is supported as of the following engineering tool versions:

- SIMOTION SCOUT/Starter >=V4.1.2
- SIMATIC Step 7 >= V5.4 SP4

However, if the need to use devices that do not support PROFINET V2.2 for IRT communication arises during machine configuration, SIMOTION and SINAMICS can both be converted to PROFINET V2.1. In each case, the default setting is V.2.2. In order to revert SIMOTION to V2.1, the relevant firmware must be loaded to the device. The appropriate firmware versions for SIMOTION D4xx can be found on the SCOUT DVD under Firmware\3_D4xx\Firmware_D4xx\V4.1.2.3\PN_V2.1and_DP\d4xx_pn21.zip. If downgrading is necessary for SIMOTION P350, please get in touch with your Siemens contact. In the case of SINAMICS S120, it is possible to do this by setting parameter p8835 to 0. SIMOTION C240 PN only supports PROFINET V2.2 and cannot be downgraded to PROFINET V2.1.

Note

Where possible, all projects involving the versions mentioned above should be carried out with PROFINET V2.2, as future versions of SIMOTION and SINAMICS will no longer support PROFINET with IRT Version V2.1.

7.3.3 Procedure for configuring PROFINET IO with IRT High Performance

Procedure

The following steps need to be performed when configuring PROFINET IO with IRT High Performance V2.2:

1. Insert the SIMOTION module.
   Select the **CPU type** followed by the **variant**. Be sure to select the variant with PROFINET V2.2.
   - Select the CPUs in the insertion dialog in SIMOTION SCOUT, e.g. CPU type D445-1 and variant D445 PN-V2.2, SINAMICS S120 Integrated V2.5.
   - For SIMOTION D4x5, you will also have to drag the CBE30 module from the hardware catalog in HW Config and drop it onto the relevant interface of the SIMOTION module. The CBE30 modules can be found in the hardware catalog under **SIMOTION Drive Based > SIMOTION D4xx > 6AUxx > V4.1 - PN-V2.2xx**.
2. Insert IO devices:
   Insert IO devices from the hardware catalog in HW Config into the I/O system. The IO devices can be found in the hardware catalog under **PROFINET IO**.
3. Configure sync domain and specify send clock:
   Specify which PROFINET IO node is to be the sync master (clock generator) and define the sync slaves. Specify the send clock.
4. Generate the topology:
   Specify the topology, i.e. how the individual ports of the PROFINET IO devices are interconnected with one another. The topology only needs to be configured for PROFINET IO with IRT High Performance. If topology-based initialization is to be used, the topology will need to be configured for all PROFINET devices.
5. Controller-controller data exchange broadcast:
   Specify which address areas are to be used for sending and receiving.

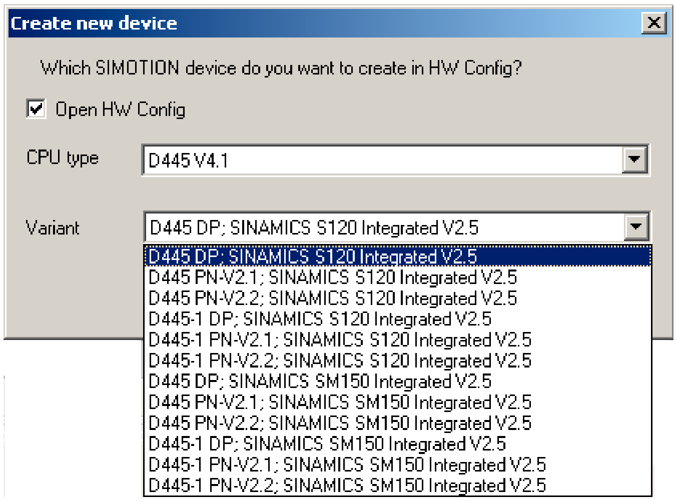
7.3.4 Inserting and configuring SIMOTION CPU D4x5

General information

You have created a project and want to configure a SIMOTION D4x5 using a PROFINET interface. With D4x5, the PROFINET interface is implemented via a CBE30. With D410 PN, the PROFINET interface is already integrated.

Procedure

1. Click "Create new device" in the project navigator to open the device selection dialog.
2. Select the **CPU type** (e.g. D445 V4.1). All CPUs under D4x5 V4.1 support PROFINET V2.2. You can also, however, select variant PN-V2.1 for PROFINET V2.1.
   Select the PROFINET and drive version under **Variant.** For PROFINET V2.2, you will have to select a variant with PN-V2.2 (e.g. D445 PN-V2.2 SINAMICS Integrated V2.5).



Which SIMOTION device do you want to create in HW Config?

☑ Open HW Config

CPU type: D445 V4.1

Variant: D445 DP; SINAMICS S120 Integrated V2.5

D445 DP; SINAMICS S120 Integrated V2.5
D445 PN-V2.1; SINAMICS S120 Integrated V2.5
D445 PN-V2.2; SINAMICS S120 Integrated V2.5
D445-1 DP; SINAMICS S120 Integrated V2.5
D445-1 PN-V2.1; SINAMICS S120 Integrated V2.5
D445-1 PN-V2.2; SINAMICS S120 Integrated V2.5
D445 DP; SINAMICS SM150 Integrated V2.5
D445 PN-V2.1; SINAMICS SM150 Integrated V2.5
D445 PN-V2.2; SINAMICS SM150 Integrated V2.5
D445-1 DP; SINAMICS SM150 Integrated V2.5
D445-1 PN-V2.1; SINAMICS SM150 Integrated V2.5
D445-1 PN-V2.2; SINAMICS SM150 Integrated V2.5

3. Click **OK** to confirm. HW Config opens.
4. You will have to insert a CBE30 for the relevant drive unit in HW Config (see Adding and configuring a CBE30-PROFINET board (Page 165)).

7.3.5 Adding and configuring a CBE30-PROFINET board

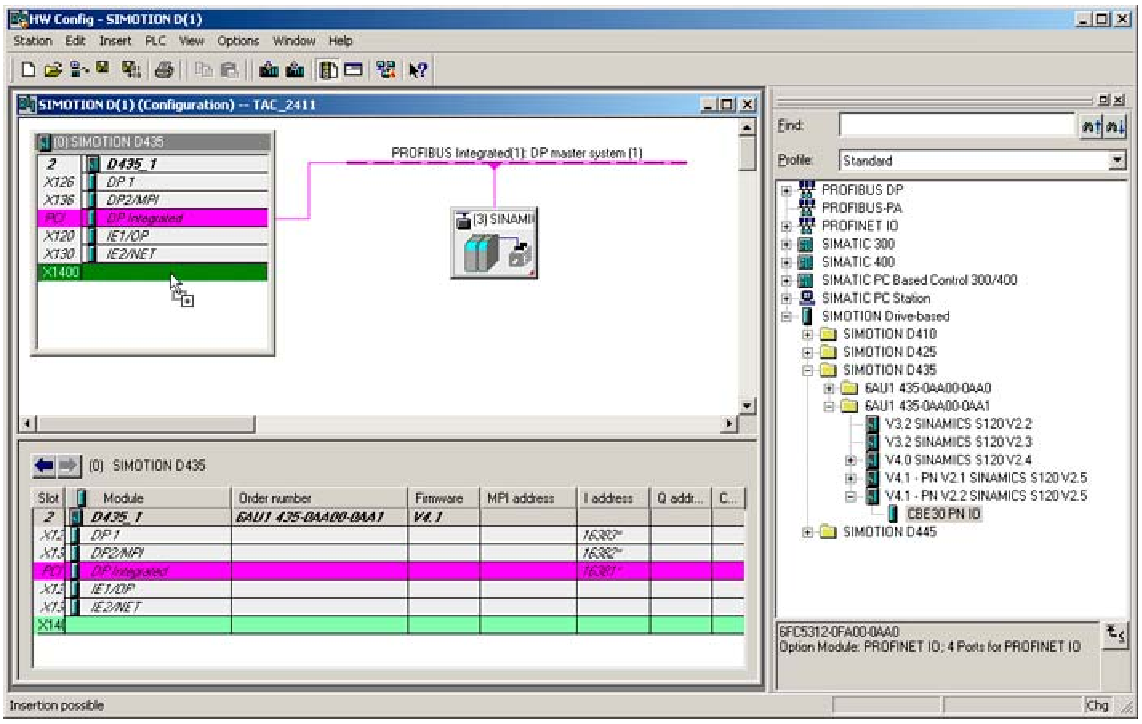
Requirement

You have created a project and have already inserted a SIMOTION device.

Procedure

1. In the project navigator, double-click the module (in this case D445). HW Config is displayed with the corresponding module.
2. In the hardware catalog, click the module entry, e.g. SIMOTION D445.

3. Click the entries for the order number and the version.
   The PROFINET module CBE30-PN is displayed below the version. As soon as the CBE30-PN is selected, X1400 becomes green.


HW Config - SIMOTION D(1)
Station Edit Insert PLC View Options Window Help
SIMOTION D(1) (Configuration) -- TAC_2411
PROFIBUS Integrated(1): DP master system (1)
(3) SINAMI
Find:
Profile: Standard
Chg

(0) SIMOTION D435

2	D435_1
X126	DP 1
X136	DP2/MPI
PCI	DP Integrated
X120	IE1/OP
X130	IE2/NET
X1400	

(0) SIMOTION D435

Slot	Module	Order number	Firmware	MPI address	I address	Q addr...	C...
2	D435_1	6AU1 435-0AA00-0AA1	V4.1				
X12	DP 1				16383*		
X13	DP2/MPI				16382*		
PCI	DP Integrated				16381*		
X12	IE1/OP						
X13	IE2/NET						
X14							

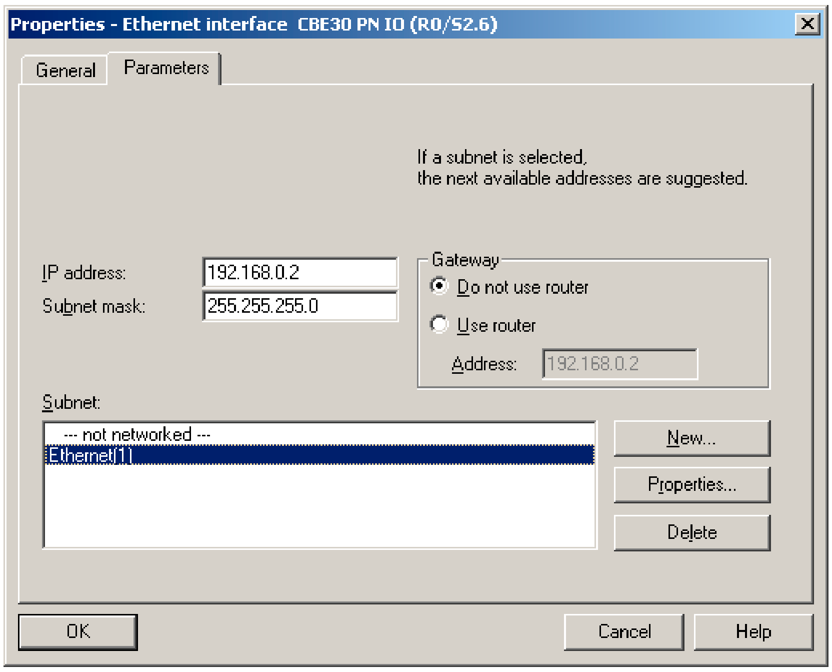
- PROFIBUS DP
- PROFIBUS-PA
- PROFINET IO
- SIMATIC 300
- SIMATIC 400
- SIMATIC PC Based Control 300/400
- SIMATIC PC Station
- SIMOTION Drive-based
  - SIMOTION D410
  - SIMOTION D425
  - SIMOTION D435
    - 6AU1 435-0AA00-0AA0
    - 6AU1 435-0AA00-0AA1
      - V3.2 SINAMICS S120 V2.2
      - V3.2 SINAMICS S120 V2.3
      - V4.0 SINAMICS S120 V2.4
      - V4.1 - PN V2.1 SINAMICS S120 V2.5
      - V4.1 - PN V2.2 SINAMICS S120 V2.5
        - CBE30 PN IO
  - SIMOTION D445

6FC5312-0FA00-0AA0
Option Module: PROFINET IO; 4 Ports for PROFINET IO

Insertion possible

4. Drag the CBE30-PN to the corresponding interface of the SIMOTION module (X1400).
   The **Properties - Ethernet Interface CBE30-PN (R0/S2.6)** window opens.
5. Click **New** to create a new subnet. The **Properties – New subnet Industrial Ethernet** dialog box is displayed.

6. Click **OK** to confirm these entries. A new Ethernet subnet is created, e.g. Ethernet(1).


Properties - Ethernet interface CBE30 PN IO (R0/S2.6)

General | Parameters

If a subnet is selected,
the next available addresses are suggested.

IP address: 192.168.0.2
Subnet mask: 255.255.255.0

Gateway
(● Do not use router)
○ Use router
Address: 192.168.0.2

Subnet:
--- not networked ---
Ethernet(1)

7. Select the subnet.
8. Assign the desired IP address.
9. Accept the settings by clicking **OK**.

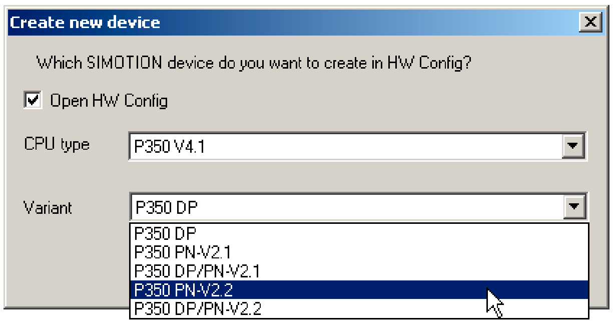
7.3.6 Inserting and configuring P350

Requirement

You have already created a project and now want to insert a P350 with PROFINET.

Procedure

1. Click **Create new device** to open the device selection dialog box.
2. Select the used variant of the P350 (i.e. P350 PN or P350 DP/PN), then click **OK** to confirm.


Create new device

Which SIMOTION device do you want to create in HW Config?

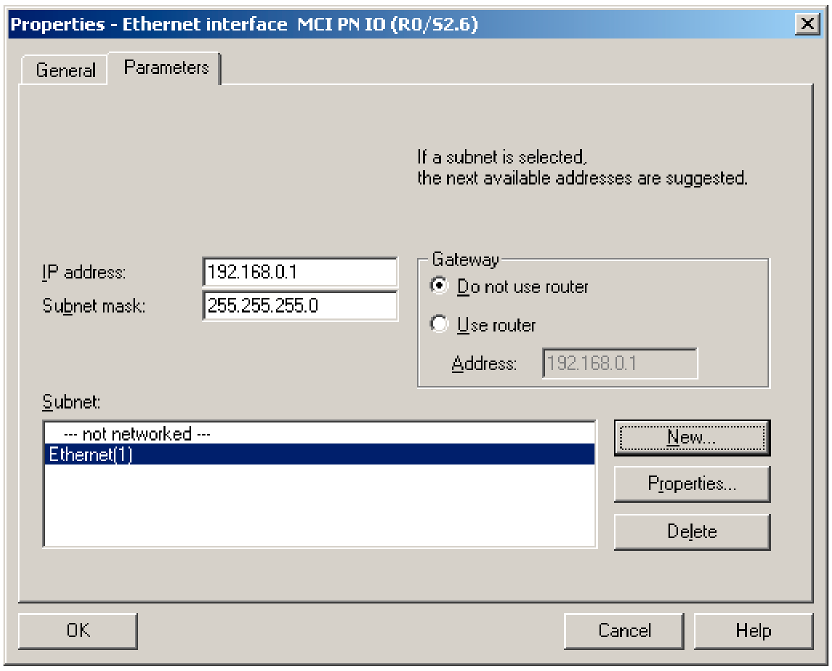
☑ Open HW Config

CPU type: P350 V4.1

Variant: P350 DP

P350 DP
P350 PN-V2.1
P350 DP/PN-V2.1
P350 PN-V2.2
P350 DP/PN-V2.2

3. The dialog box for creating a PROFINET subnet will be displayed. Enter the IP address and the subnet mask here. Click **OK** to confirm.



If a subnet is selected,
the next available addresses are suggested.

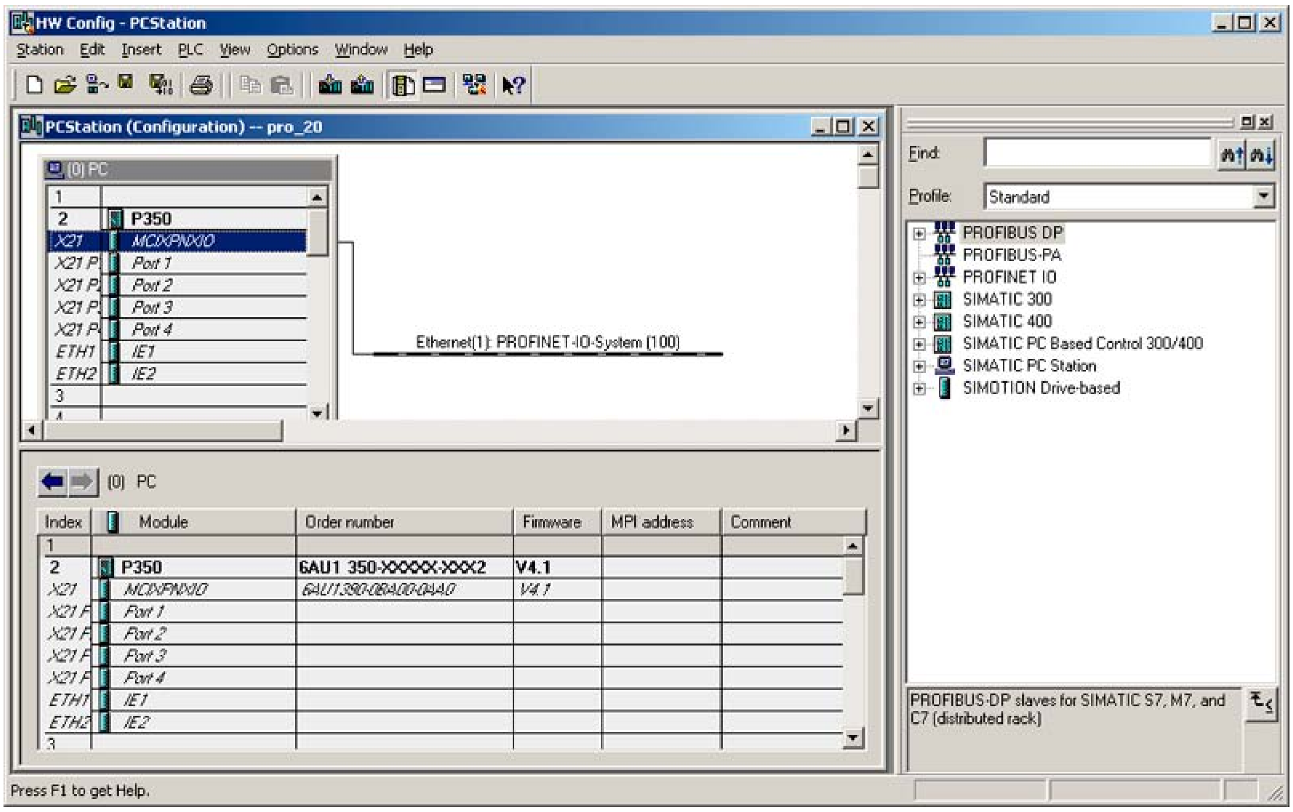
IP address: 192.168.0.1
Subnet mask: 255.255.255.0

Gateway: [✓ Do not use router] Use router
Address: 192.168.0.1

Subnet:
--- not networked ---
Ethernet(1)

4. Select the PG/PC interface from the next dialog box and click **OK** to confirm.

   The HW Config opens and displays the module with the configured PROFINET subnet.



1	
2	P350
X21	MCIXPNXIO
X21 P	Port 1
X21 P	Port 2
X21 P	Port 3
X21 P	Port 4
ETH1	IE1
ETH2	IE2
3	

Ethernet(1): PROFINET-IO-System (100)

(0) PC

Index	Module	Order number	Firmware	MPI address	Comment
1					
2	P350	6AU1 350-XXXXX-XXX2	V4.1		
X21	MCIXPNXIO	6AU1390-0BA00-0AA0	V4.1		
X21 F	Port 1				
X21 F	Port 2				
X21 F	Port 3				
X21 F	Port 4				
ETH1	IE1				
ETH2	IE2				
3					

PROFIBUS-DP slaves for SIMATIC S7, M7, and C7 (distributed rack)

Press F1 to get Help.

Figure 7-15 HW Config with PROFINET for P350

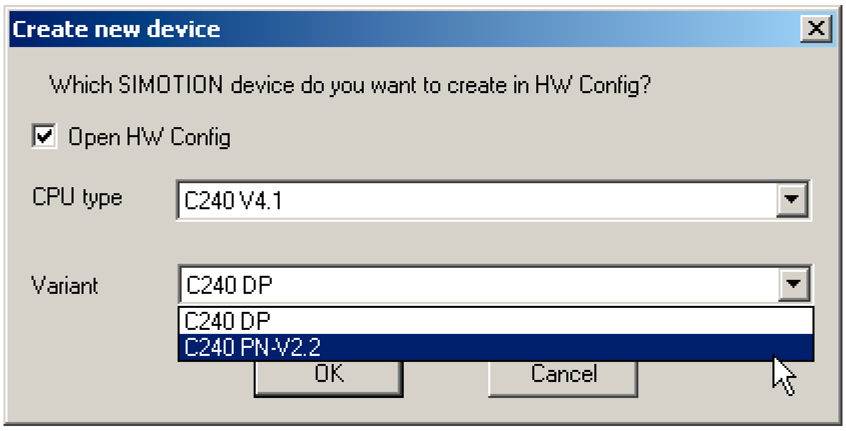
7.3.7 Inserting and configuring the C240

Requirement

You have already created a project and now want to insert a C240 with PROFINET.

Procedure

1. Click **Create new device** to open the device selection dialog box.
2. Select the version of the C240 PN in use and click **OK** to confirm.


Create new device

Which SIMOTION device do you want to create in HW Config?

☑ Open HW Config

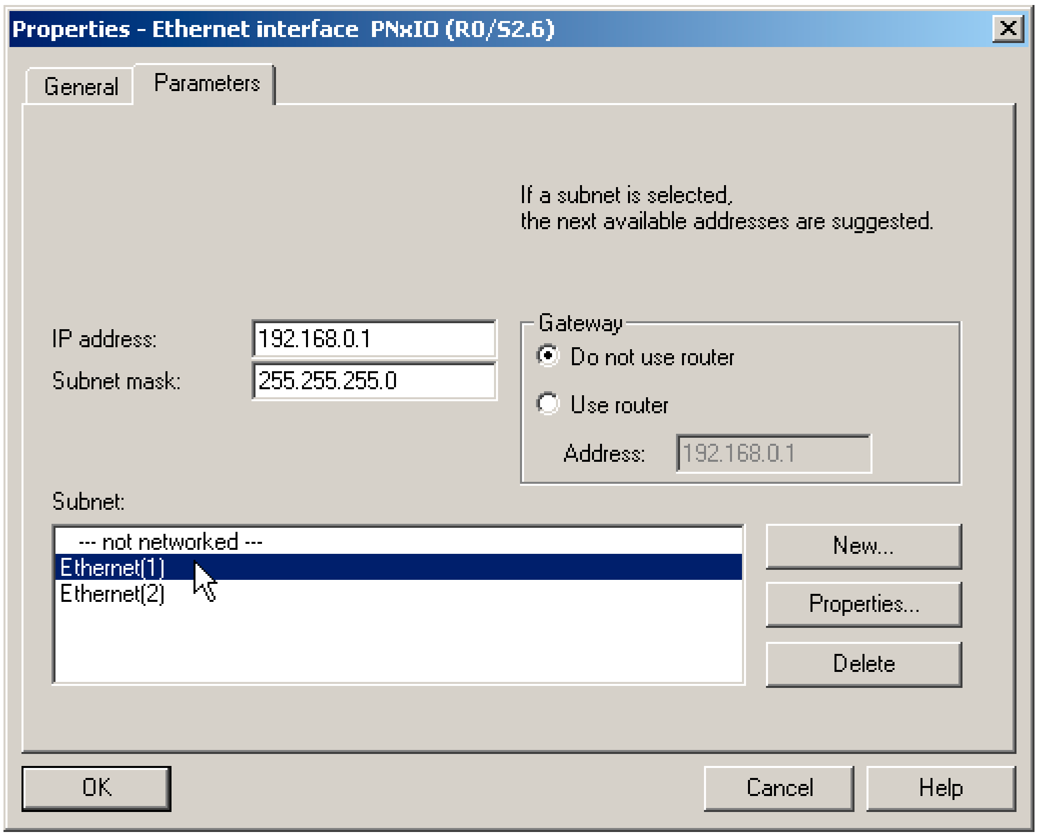
CPU type: C240 V4.1

Variant: C240 DP

C240 DP
C240 PN-V2.2

Figure 7-16 Creating a new C240 PN device

3. The dialog box for creating a PROFINET subnet will be displayed. Enter the IP address and the subnet mask here. Click **OK** to confirm.



If a subnet is selected,
the next available addresses are suggested.

IP address: 192.168.0.1
Subnet mask: 255.255.255.0

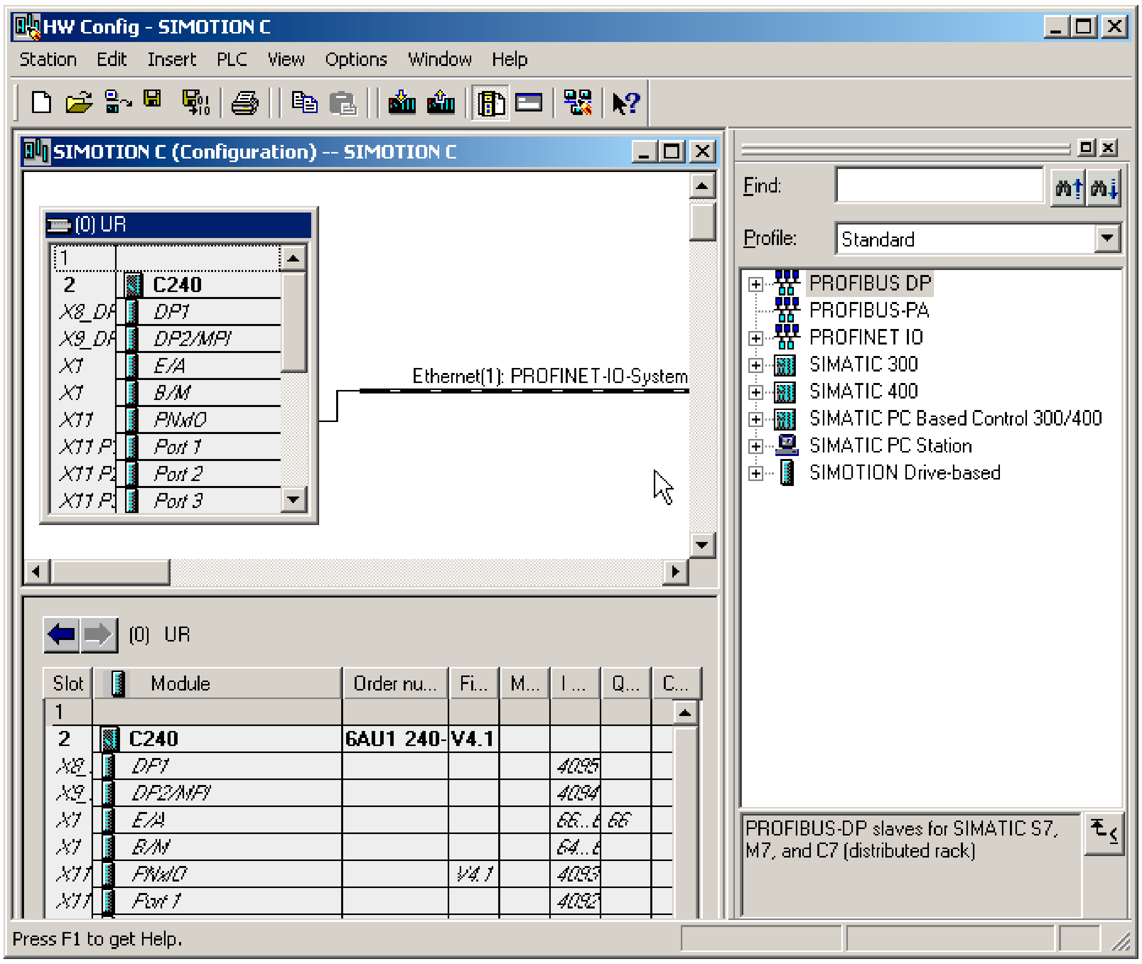
Gateway: [✓ Do not use router] Use router
Address: 192.168.0.1

Subnet:
--- not networked ---
Ethernet(1)
Ethernet(2)

Figure 7-17 Creating a new Ethernet for C240 PN

4. Select the PG/PC interface from the next dialog box and click **OK** to confirm.

   The HW Config opens and displays the module with the configured PROFINET subnet.



1	
2	C240
X8_DP	DP1
X9_DP	DP2/MPI
X1	E/A
X1	B/M
X11	PNxIO
X11 P	Port 1
X11 P	Port 2
X11 P	Port 3

(0) UR

Slot	Module	Order nu...	Fi...	M...	I ...	Q...	C...
1							
2	C240	6AU1 240-	V4.1				
X8	DP1				4095		
X9	DP2/MPI				4094		
X1	E/A				66...6	66	
X1	B/M				64...6		
X11	PNxIO		V4.1		4093		
X11	Port 1				4092		

PROFIBUS-DP slaves for SIMATIC S7, M7, and C7 (distributed rack)

Figure 7-18 HW Config with PROFINET for C240 PN

7.3.8 Creating a sync domain

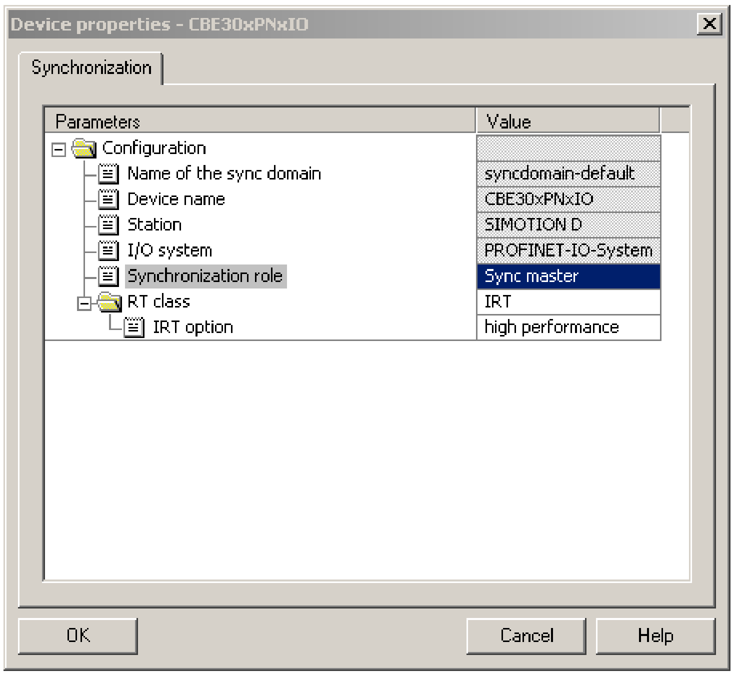
A sync domain is a group of PROFINET devices synchronized to a common cycle clock. One device has the role of the sync master (clock generator), all other devices have the role of sync slave.

Note

All devices that exchange data via IRT must belong to a single sync domain and be directly connected to one another; i.e. the connection must not be interrupted by devices that do not support IRT as this will prevent the synchronization information from being passed on.

Procedure

1. In HW Config, open the station with the PROFINET devices to be involved in IRT communication and select, for example, the CBE30 PROFINET interface in the station.
2. Select the **Edit > PROFINET IO > Domain Management** menu command. A dialog tab with the list of all the devices opens. A default sync domain is created and the devices are already assigned.
3. Select the station in the upper field and in the lower field double-click the device that is to be configured as sync master, e.g. CBE30. The Properties dialog box of the device opens.



Parameters	Value
Configuration	
Name of the sync domain	syncdomain-default
Device name	CBE30xPNxIO
Station	SIMOTION D
I/O system	PROFINET-IO-System
Synchronization role	Sync master
RT class	IRT
IRT option	high performance

Figure 7-19 Selecting synchronization

4. Set the synchronization type to sync master.
5. Confirm the settings with **OK**.
6. Then, select all devices which are to be configured as sync slaves (keep the Ctrl key depressed and select the devices one after the other).
7. Then, click on the **Properties device** button.
8. Set the synchronization type to Sync slave in the dialog box.
9. Confirm the settings with **OK**.

Note

Any devices for which **not synchronized** is selected will not be involved in IRT communication, but will automatically take part in RT communication.

7.3.9 Defining send clock and refresh times

PROFINET RT and IRT are forms of cyclic communication; the basic cycle clock is the send clock. The update time is a multiple (2^n) of the send clock and the devices are supplied with data in this cycle clock. Individual update times can be set for each device.

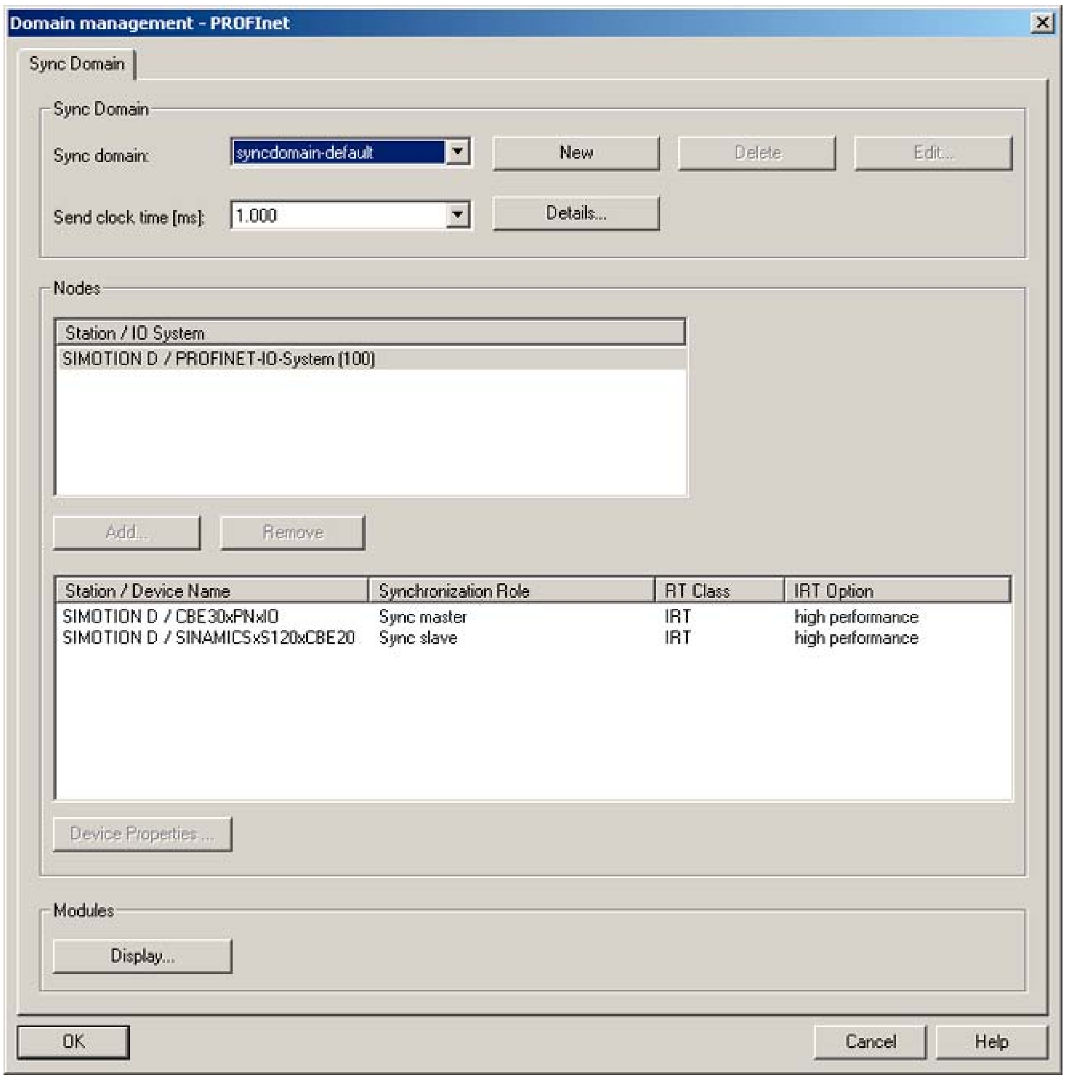
Note

The update time is adjustable for PROFINET RT and IRT High Flexibility. With IRT High Performance, the devices are supplied with data during each send clock (send clock = update time).

As far as SINAMICS is concerned, however, the controller can be used to define an application cycle whereby SINAMICS is only supplied with new data every n cycles.

How to set the send clock

1. In HW Config, open the **Domain Management** dialog box.


Domain management - PROFInet

Sync Domain

Sync domain: syncdomain-default

Send clock time [ms]: 1.000

Station / IO System
SIMOTION D / PROFINET-IO-System (100)

Station / Device Name	Synchronization Role	RT Class	IRT Option
SIMOTION D / CBE30xPNxIO	Sync master	IRT	high performance
SIMOTION D / SINAMICSxS120xCBE20	Sync slave	IRT	high performance

Figure 7-20 Domain Management

2. Select an appropriate send clock for the process. The transmission cycle clock is the smallest possible transmission interval. The send clock is preset to 1 ms.

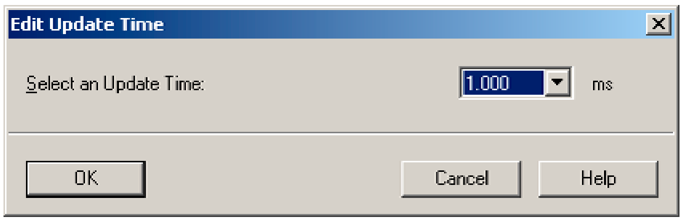
Note

Communication should be no faster than required, irrespective of what the maximum communication speed may be. This will reduce the bandwidth requirement and the relieve the load that the devices need to support.

Defining update times for PROFINET IO using PROFINET devices with RT or IRT High Flexibility

Update times for IO data exchange of PROFINET IO involving PROFINET devices with RT or IRT High Flexibility are set in the **Properties PROFINET IO System** dialog.

1. Click the path for the PROFINET IO system in HW Config and select **Object properties** from the context menu. The dialog is displayed.
2. Switch to the **Update time** tab and select the device from the overview containing all the IO devices.
3. Click **Edit**. You can select the refresh time in the **Edit refresh time** dialog.



Select an Update Time: 1.000 ms

4. Click **OK** to confirm.

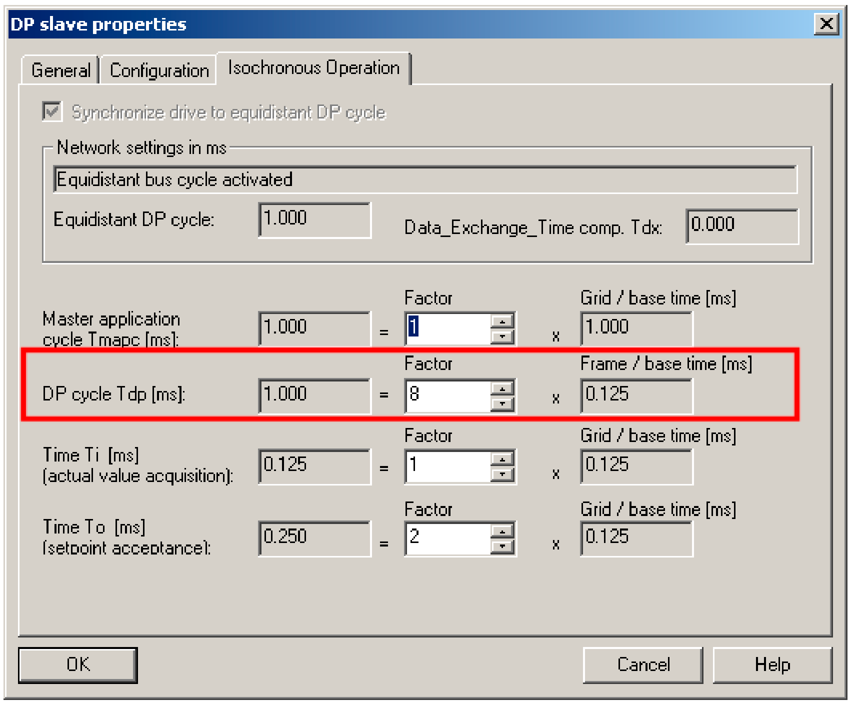
Send clock/DP cycle ratio for SINAMICS_Integrated (SIMOTION D) with IRT High Performance

If a SINAMICS drive is being operated on a SIMOTION D via PROFINET IO IRT High Performance, the DP cycle for the integrated PROFIBUS must be the same as the position control cycle. The DP cycle is set to 3 ms by default. In most cases, this will not be the same as the position control cycle clock selected for PROFINET applications.

You set the DP cycle in the DP slave properties:

1. Double-click SINAMICS_Integrated in HW Config. The **DP slave properties** window appears.
2. Switch to the **Isochronous Operation** tab and activate the "Synchronize drive to equidistant DP cycle" checkbox.

3. Set the factor for **DP cycle Tdp**. The DP cycle must be the same as the position control cycle clock. For example, if the position control cycle clock is 1 ms, you will need to enter a factor of 8 (8 x [base time of 0.125 ms] = 1 ms).



☑ Synchronize drive to equidistant DP cycle

Network settings in ms

Equidistant bus cycle activated

Equidistant DP cycle: 1.000　Data_Exchange_Time comp. Tdx: 0.000

			Factor		Grid / base time [ms]
Master application cycle Tmapc [ms]:	1.000	=	1	x	1.000
			Factor		Frame / base time [ms]
DP cycle Tdp [ms]:	1.000	=	8	x	0.125
			Factor		Grid / base time [ms]
Time Ti [ms] (actual value acquisition):	0.125	=	1	x	0.125
			Factor		Grid / base time [ms]
Time To [ms] (setpoint acceptance):	0.250	=	2	x	0.125

4. Click **OK** to confirm.

When PROFINET is operated isochronously, the position control cycle clock must always correspond to the PROFIBUS cycle clock. The position control cycle clock and the PROFIBUS cycle clock can be scaled to the PROFINET cycle clock.

Example:

PROFINET send clock = 0.5 ms

PROFIBUS cycle clock = position control cycle clock = 1 ms

The PROFIBUS cycle clock can operated relative to the PROFINET cycle clock at a ratio of 1:1 to 1:16.

7.3.10 Configuring a topology

7.3.10.1 Topology

Introduction

With IRT High Performance, the topology must be configured and settings made to determine which device is to be connected via which port to which other devices.

Note

With IRT High Flexibility or RT, topology configuration is optional. It may be required for topology-based initialization or diagnostics purposes, for example.

There are two options for defining the properties of the cables between the ports of the switches:

Using the topology editor (Page 179)

Using the object properties

7.3.10.2 Topology editor (graphical view)

Procedure

With the topology editor you have an overview of all ports in the project and can interconnect them centrally.

The topology editor is started with the Edit > PROFINET IO > Topology menu command in HW Config or NetPro (PROFINET device must be selected).

The topology editor offers the user options for displaying the topology graphically (STEP7 V5.4 SP2 or higher) or in a tabular format. The graphic view is more suitable for situations involving interconnection.

Description

In the topology editor, you can:

- Interconnect ports
- Modify the properties of the interconnection
- Add passive components
- Arrange for an offline/online comparison to be displayed in online mode

Procedure

1. In SCOUT, double-click the SIMOTION module in order to access HW Config.
2. Select the PROFINET module, e.g. a CBE30-PN.
3. Perform **Edit > PROFINET IO > Topology**. The topology editor opens.
4. Click **Graphical view** to bring the tab into the foreground.

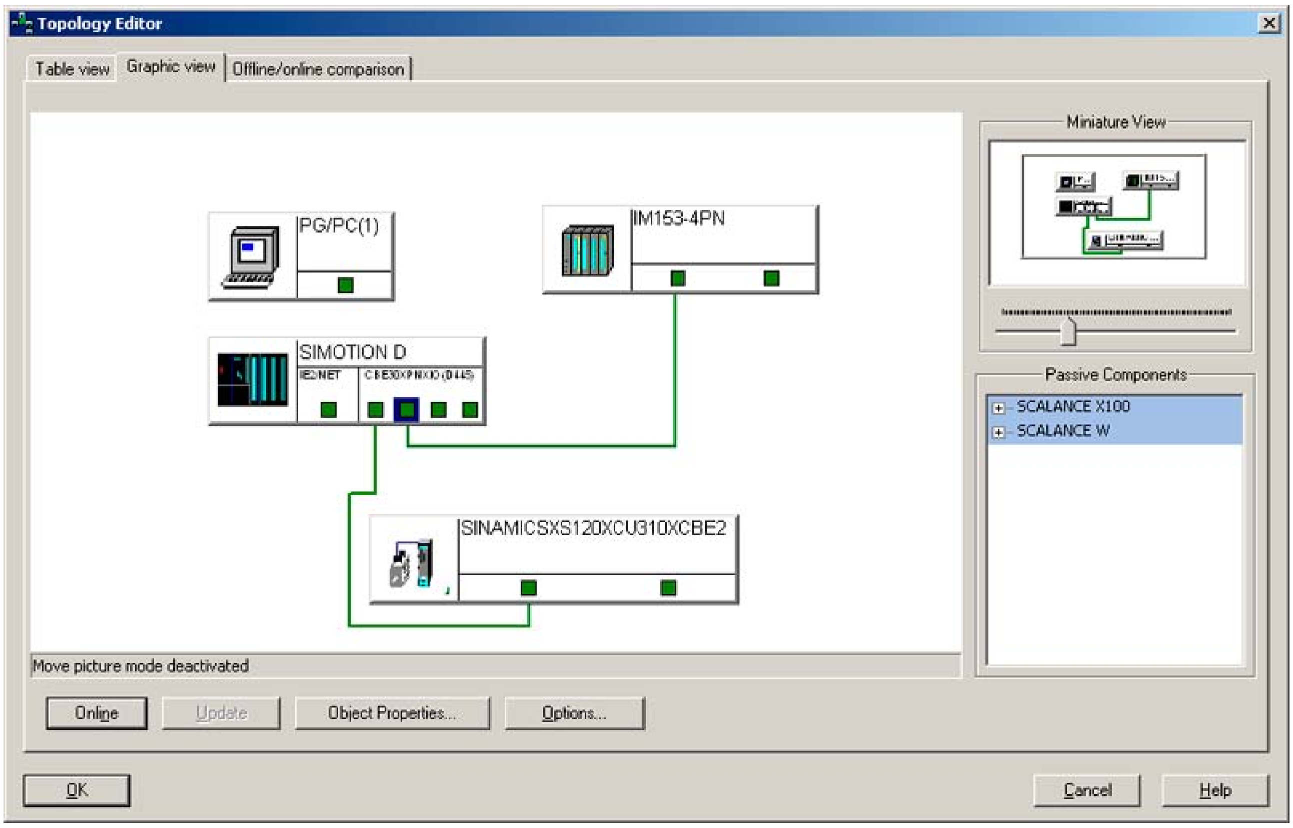
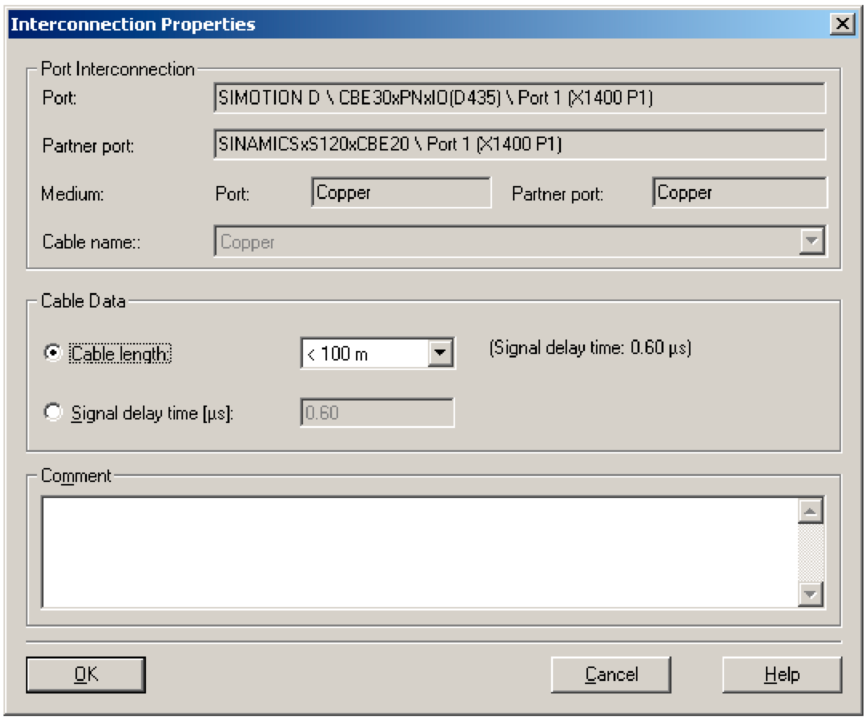


Figure 7-21 Topology editor (graphical view)

5. Establish connections between ports by pressing and holding down the left mouse button and drawing a line between the two ports to be connected. The **Interconnection Properties** window opens.

6. The port interconnection is displayed: You can configure the cable data: A cable length of < 100 m is set by default (recommended length). Alternatively, you can configure the signal propagation delay:



Interconnection Properties

Port Interconnection

Port: SIMOTION D \ CBE30xPNxIO(D435) \ Port 1 (X1400 P1)

Partner port: SINAMICSxS120xCBE20 \ Port 1 (X1400 P1)

Medium: Port: Copper Partner port: Copper

Cable name:: Copper

Cable Data

● Cable length: < 100 m (Signal delay time: 0.60 µs)

○ Signal delay time [µs]: 0.60

Comment

7. Click **OK** to confirm.

Offline/online comparison

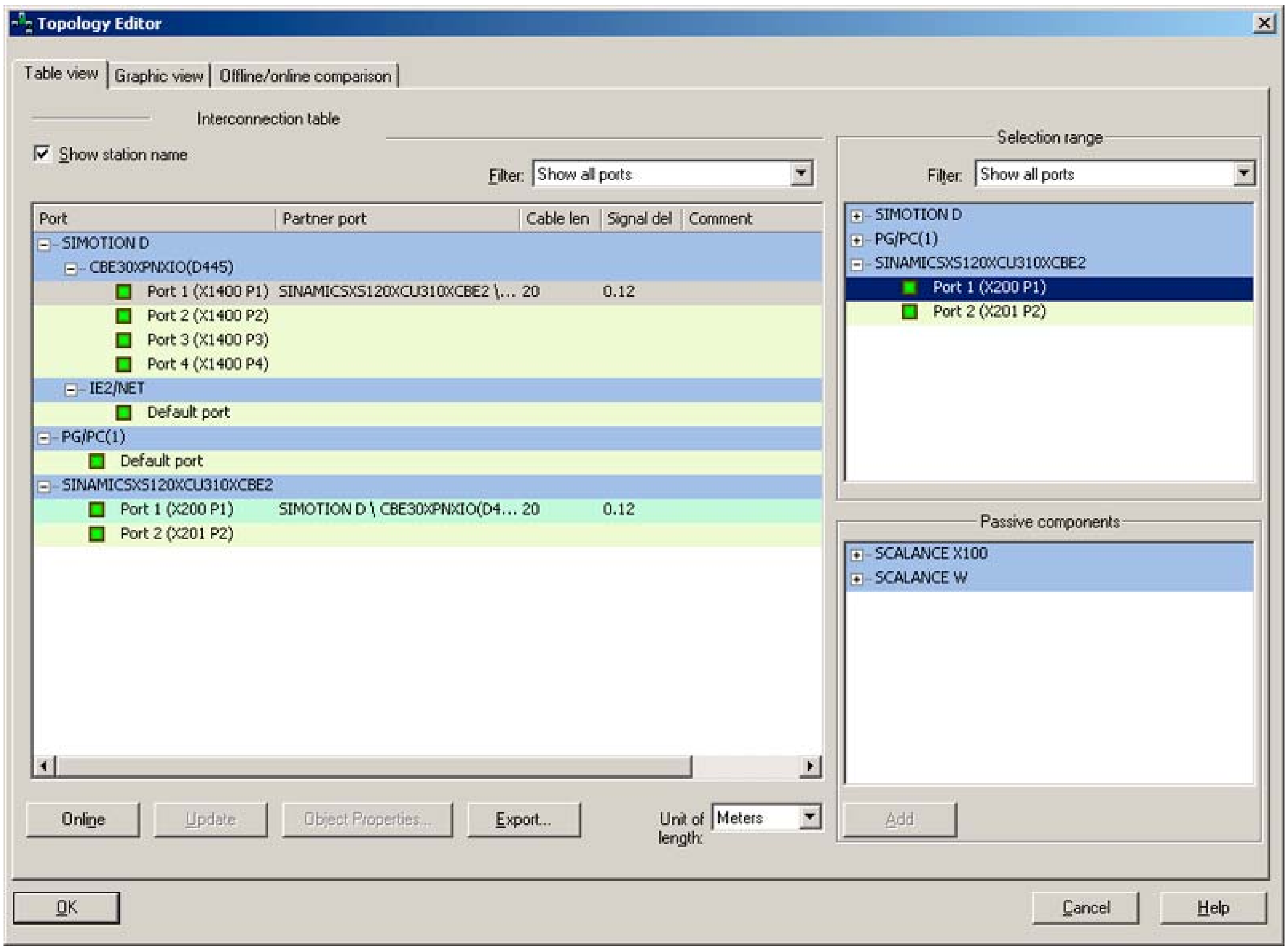
When you switch to online mode, the topology in the editor is compared with the actual topology. Any components which are not recognized are highlighted by a question mark, while connections and components in RUN are highlighted in green.

7.3.10.3 Interconnecting ports via the topology editor (table view)

Procedure

Ports can be interconnected in the tabular overview of the topology editor.

The topology editor is started with the **Edit > PROFINET IO > Topology** menu command in HW Config or NetPro (PROFINET device must be selected).



Interconnection table

[✓] Show station name

Filter: Show all ports

Port	Partner port	Cable len	Signal del	Comment
⊟ SIMOTION D				
⊟ CBE30XPNXIO(D445)				
🟩 Port 1 (X1400 P1)	SINAMICSXS120XCU310XCBE2 \...	20	0.12	
🟩 Port 2 (X1400 P2)				
🟩 Port 3 (X1400 P3)				
🟩 Port 4 (X1400 P4)				
⊟ IE2/NET				
🟩 Default port				
⊟ PG/PC(1)				
🟩 Default port				
⊟ SINAMICSXS120XCU310XCBE2				
🟩 Port 1 (X200 P1)	SIMOTION D \ CBE30XPNXIO(D4...	20	0.12	
🟩 Port 2 (X201 P2)				

Unit of length: Meters

Selection range

Filter: Show all ports

- ⊞ SIMOTION D
- ⊞ PG/PC(1)
- ⊟ SINAMICSXS120XCU310XCBE2
  - 🟩 Port 1 (X200 P1)
  - 🟩 Port 2 (X201 P2)

Passive components

- ⊞ SCALANCE X100
- ⊞ SCALANCE W

Figure 7-22 Topology editor

All configured PROFINET IO devices with their ports are listed in the interconnection table on the left-hand side. You can use the Filter dialog to select whether all ports, only the ports that have not yet been interconnected or only the ports that have already been interconnected are to be displayed.

Interconnecting ports in the tabular overview

1. To interconnect ports of different devices, select the port of a device that you want to interconnect in the right-hand field.
2. Drag this port to the desired port of a device in the interconnection table. The "Interconnection Properties" dialog opens.
3. Configure the cable data. A cable length of < 100 m should be set by default.
4. Confirm your entries with **OK**.

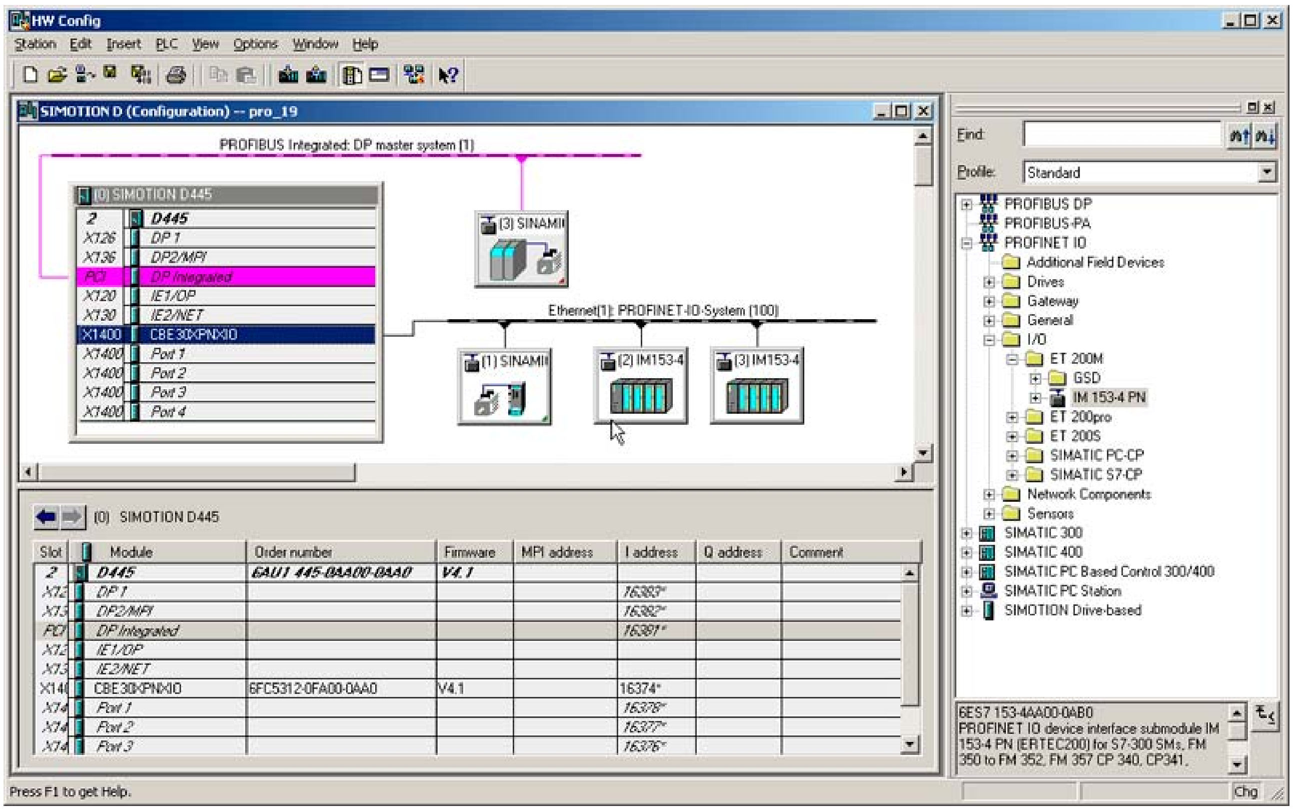
7.3.11 Creating an IO device

Requirement

You have already created a PROFINET IO system and configured a PROFINET IO module, e.g. SIMOTION D445 with CBE30-PN (see Inserting and configuring SIMOTION CPU D4x5 (Page 164)).

Procedure for PROFINET IO devices using the hardware catalog

1. Double-click the corresponding module in SIMOTION SCOUT to open HW Config.
2. Under PROFINET IO in the hardware catalog, select the module you wish to connect to the PROFINET IO system.
3. Drag the module to the path of the PROFINET IO system. The IO device is inserted.



(0) SIMOTION D445

Slot	Module	Order number	Firmware	MPI address	I address	Q address	Comment
2	D445	6AU1 445-0AA00-0AA0	V4.1				
X12	DP 1				16383*		
X13	DP2/MPI				16382*		
PCI	DP Integrated				16381*		
X12	IE1/OP						
X13	IE2/NET						
X14(	CBE30xPNxIO	6FC5312-0FA00-0AA0	V4.1		16374*		
X14	Port 1				16378*		
X14	Port 2				16377*		
X14	Port 3				16376*		

6ES7 153-4AA00-0AB0
PROFINET IO device interface submodule IM 153-4 PN (ERTEC200) for S7-300 SMs, FM 350 to FM 352, FM 357 CP 340, CP341.

4. Save and compile the settings in HW Config.

Procedure for third-party manufacturer PROFINET IO devices

1. Double-click the corresponding module to open HW Config.
2. Select the **Options > Install GSD files** menu command.
3. Select the GSD file to be installed in the Install GSD Files dialog box.
4. Click the **Install** button.

5. Close the dialog box by clicking the **Close** button.
6. Under PROFINET IO in the hardware catalog, select the module you wish to connect to the PROFINET IO system.
7. Drag the module to the path of the PROFINET IO system. The IO device is inserted.
8. Save and compile the settings in HW Config.

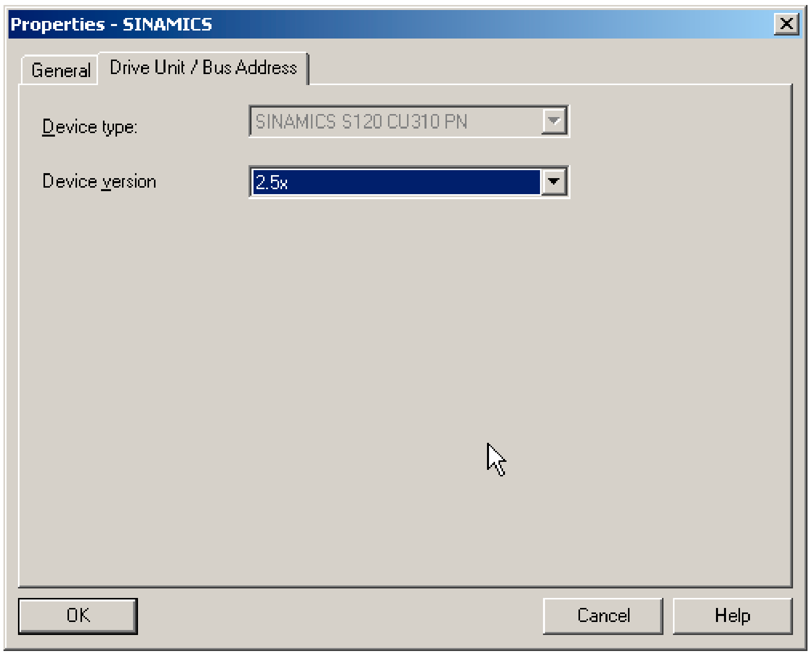
7.3.12 Inserting and configuring the SINAMICS S120

Requirement

You have inserted a SIMOTION device and a CBE30 in your project and a PROFINET IO subnet has already been created.

Procedure

1. Select **PROFINET IO > Drives > SINAMICS** in the HW Config hardware catalog. Then select the module (e.g. **SINAMICS S120 CU320 CBE20**).
2. Click the entry and drag the drive (e.g. **V2.5 PN-V2.2**) to the PROFINET IO subnet. The **Properties - Ethernet Interface SINAMICS-S120-CBE20** window opens.
   A suggested IP address will already be displayed here and the subnet will be selected.
3. Click **OK** to accept the settings.
   The dialog box **Properties SINAMICS** is displayed.
4. Select the device version (firmware version).



Device type: SINAMICS S120 CU310 PN
Device version 2.5x

Figure 7-23 SINAMICS properties

5. Click **OK** to confirm these entries.
6. In HW Config, select **Station > Save and Compile All**.

Set message frame

If you want to use isochronous communication, you must configure a message frame which supports this (e.g. message frame 105). This may be necessary, for example, to ensure the drive can be synchronized with PROFINET. The Ti or To input fields in the **Properties - SINAMICS CBE20 PN-IO** dialog are only active with isochronous drives.

The message frame is automatically set in SCOUT when the drive unit is configured.

1. If you have not yet configured a supply and drive, click **Configure drive unit** in the SCOUT project navigator and drive unit.
2. Run the drive unit configuration wizard. With PROFINET, one onboard interface is available and is assigned two interfaces named PZD IF1 and PZD IF2. You can define these during configuration. We recommend the **Automatic** setting is left unchanged. You should choose SIEMENS message frames numbered XXX (e.g. message frame 105 for the drive).
3. After you have closed the wizard, click **Communication > Message frame configuration** under the drive unit in the project navigator. Tab **IF1: PROFIdrive PZD message frames** contains a list of message frames.
4. Click **Transfer to HW Config** to align the addresses. Once alignment is complete, a check mark will appear after the address areas.
5. Select **Project > Save** in the menu.

Message frame in HW Config

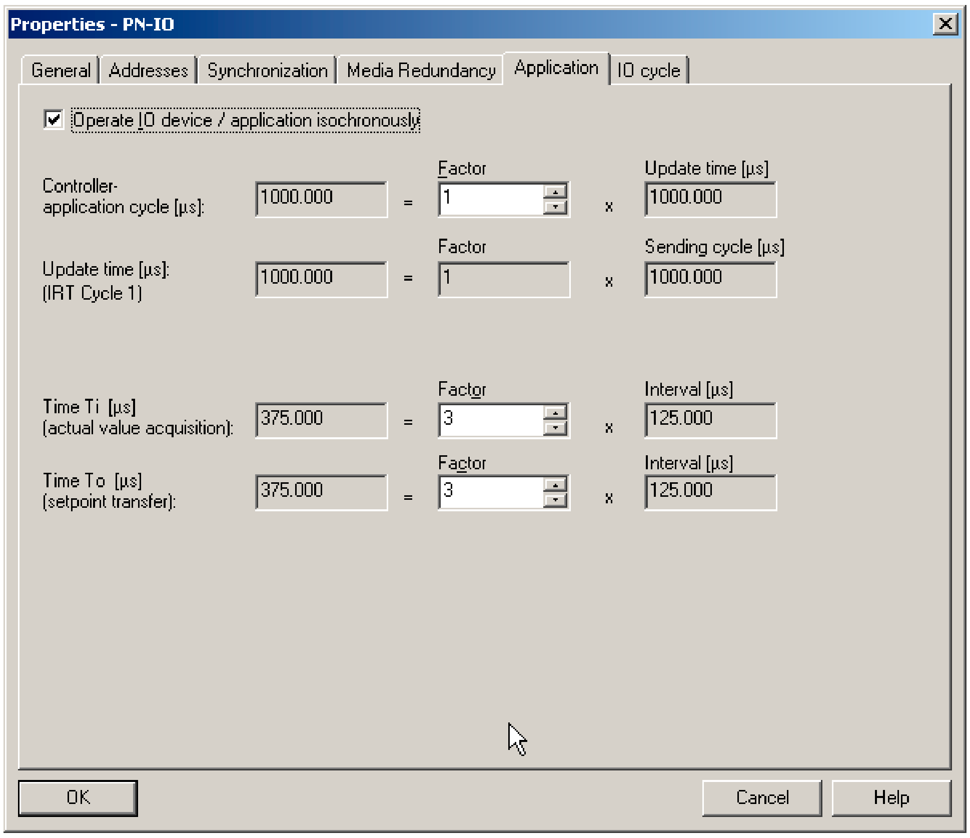
Alternatively, you can assign the message frame in HW Config. This is only possible if the drive unit and the drive have already been configured in SCOUT. You will then be able to proceed as follows in HW Config:

1. Select the inserted SINAMICS drive and double-click the entry **SIEMENS / Standard message frame xx** in the lower table.
   The **Properties SIEMENS / Standard message frame xx** dialog box is called.
2. Select the corresponding message frame. After it is saved, the message frame can be also be selected in SCOUT, under **<"Drive_device_xx"> - Communication > Message frame configuration** in the project navigator. Tuning with HW Config is possible.

Settings on the SINAMICS PROFINET interface

In order for the inserted SINAMICS S120 drive to run in isochronous mode in PROFINET, a number of additional settings must be made on the SINAMICS.

1. Select the SINAMICS drive on the PROFINET IO system and double-click the entry of the PROFINET interface in the lower table, e.g. CBE20-PN-IO.
   The **Properties CBE20-PN--IO** dialog box is displayed.
2. Select the **Application** tab.



☑ Operate IO device / application isochronously

			Factor		
Controller-application cycle [µs]:	1000.000	=	1	x	Update time [µs]: 1000.000
Update time [µs]: (IRT Cycle 1)	1000.000	=	1	x	Sending cycle [µs]: 1000.000
Time Ti [µs] (actual value acquisition):	375.000	=	3	x	Interval [µs]: 125.000
Time To [µs] (setpoint transfer):	375.000	=	3	x	Interval [µs]: 125.000

Figure 7-24 Properties SINAMICS CBE20 PN-IO

3. Click the entry **Operate IO device/application in isochronous mode**.
   The drive now takes part in the isochronous communication.
4. If necessary, enter a value > 1 under **Controller-application cycle**, in order to configure clock cycle scaling. This is required if the position control cycle clock is slower than the IRT cycle clock, although the position control cycle clock must already have been taken into account in the IRT cycle clock.

5. If necessary, switch to the **Synchronization** tab to select the **Synchronization type**, **Sync slave** in this case. This can also be set under **Domain Management**.
6. Confirm the entries with **OK**.

Note

Isochronous mode is only possible if PROFINET IRT High Performance has been configured.

During the next steps, you must include the drive in the sync domain (see Creating a sync domain), load the IP address to the drive (see Assigning device names and IP addresses to IO devices (Page 187)), and interconnect the ports (see Interconnecting ports via the topology editor (table view) (Page 179)).

See also

Creating a sync domain (Page 171)

7.3.13 IP address and communication name

Introduction

Devices and controllers have a fixed MAC address, a configurable IP address, and a communication name. The IP address, subnet mask, and communication name are defined within the Ethernet interface properties in the engineering software. This ensures the devices within the project have their own unique assignment.

You can find the communication name (device name) in the Properties - Interface dialog by double-clicking the device or the controller. Click Properties in the dialog to define the IP address. See also Adding and configuring a CBE30-PROFINET board (Page 165).

Communication name guidelines

The communication name for the device is the same as the device name (e.g. *Drive1* for a drive). The communication name for a controller is the name of the PROFINET interface (e.g. *CBE30xPNxIO* for a CBE30 on SIMOTION D).

Communication names are subject to certain syntax rules, i.e. they must always be DHCP-compliant. Additional boundary conditions also apply to SIMOTION and SIMATIC.

- Letters a-z and numbers 0-9 may be used.
- Special characters are not permitted: these include ! " § $ % & / ( ) = ? * ' _ : ; > < , # + | ~ \ }] [{

- The name may consist of more than one part.
  – Label.Label.Label.Label
  – Periods are used as separators.
  – The label must start and end with a letter.
  – Labels must not exceed 63 characters.
  – Labels must not begin with "xn-".
- The total maximum length for a name is 240 characters.
- Reserved names "port-xyz" or "port-xyz-abcde"
- No distinction is made between upper and lower case. However, since all names are shown in lower case in the engineering system, you should use lower-case letters only.
- Minus signs "-" must not be used for SCOUT.
- The engineering software replaces impermissible characters with an x.

Initialization of the controller and devices in online mode

As the controller and devices do not have a communication name or IP address when delivered, these will need to be assigned at the outset. When assigning addresses (i.e. the IP address and communication name - a process also referred to as initialization), a distinction is made between controllers and devices. You can adopt different approaches when initializing devices and controllers.

Controller initialization

- Download the application
- Engineering software
  – HW Config, NetPro, SCOUT
  – Primary Setup Tool (PST)
- Via the application (the _setNameofStation system function for SIMOTION)

Note

When using engineering software for initialization, and particularly where larger systems are involved, you should establish direct connections with the device to ensure the device to be initialized is clearly identifiable. Alternatively, the Flash feature can be used, whereby devices can be identified by a flashing LED.

Device initialization

- Engineering software
  – HW Config, NetPro
  – SCOUT, STARTER
  – Primary Setup Tool (PST)
- Write to the MMC or CF card beforehand and then plug in.

Topology-based initialization for devices

Devices can also be initialized without an MMC or CF card. This method is known as topology-based initialization and is only supported as of certain software versions.

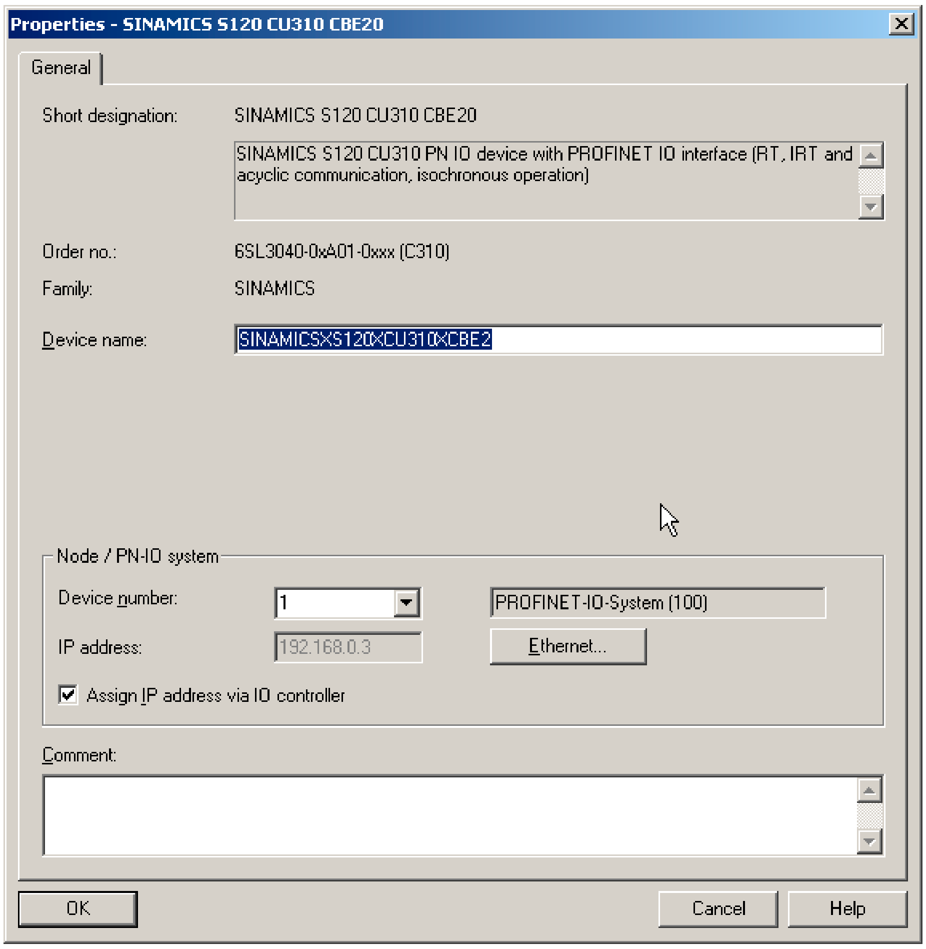
- SIMATIC S7-300 FW ≥ V 2.7
- SIMATIC S7-400 FW ≥ V 5.2
- SIMOTION FW ≥ V4.1.2 with PN V2.2
- SINAMICS FW ≥ V2.5.1.10 with PN V2.2
- ET 200S FW ≥ 6.0
- ET 200S HS FW ≥ 2.0

7.3.14 Assigning device names and IP addresses to IO devices

Introduction

You can only go online with the engineering software of a device (or controller) if it has been assigned an IP address in the engineering software. Devices must also be assigned a communication name which is unique across the network. This enables the controller to identify the devices assigned to it.

You can specify the IP address in the **Properties - Ethernet interface…**. dialog (double-click the device to open this). Moreover, a default name is entered that you can modify. **Assign IP address via IO controller** is active as the default setting, whereby the controller identifies its assigned devices during ramp-up by their communication names and assigns them the IP address specified in the engineering software. We recommend leaving this function activated.



Short designation: SINAMICS S120 CU310 CBE20

SINAMICS S120 CU310 PN IO device with PROFINET IO interface (RT, IRT and acyclic communication, isochronous operation)

Order no.: 6SL3040-0xA01-0xxx (C310)

Family: SINAMICS

Device name: SINAMICSXS120XCU310XCBE2

Node / PN-IO system

Device number: 1   PROFINET-IO-System (100)

IP address: 192.168.0.3

[✓] Assign IP address via IO controller

Comment:

Figure 7-25 Properties SINAMICS S120

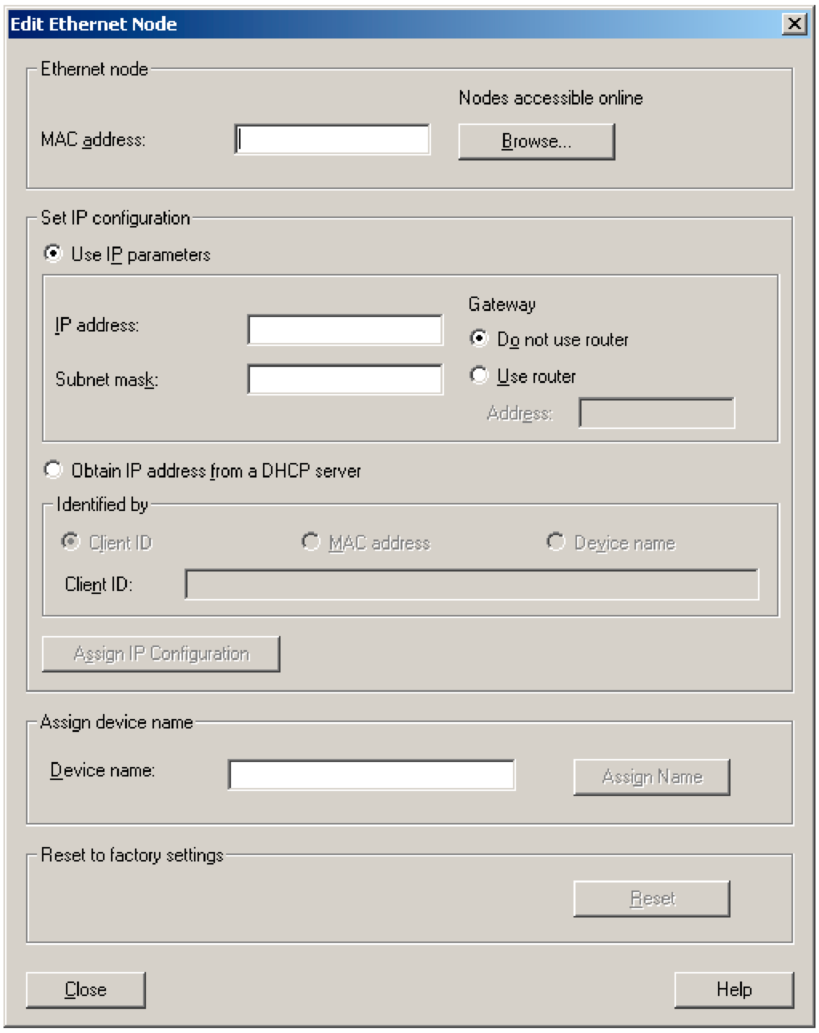
IO device initialization

Note

If you are connecting the programming device/PC directly to the device's PROFINET interface, you can use a patch or crossover cable.

During commissioning, we recommend that the device to be initialized is connected directly to the programming device/PC.

1. In HW Config or NetPro, select the **Target system - Ethernet - Edit Ethernet Node** menu item. The **Edit Ethernet node** dialog box is displayed.


Edit Ethernet Node

Ethernet node

Nodes accessible online

MAC address:

Set IP configuration

(●) Use IP parameters

IP address:

Subnet mask:

Gateway

(●) Do not use router

( ) Use router

Address:

( ) Obtain IP address from a DHCP server

Identified by

(●) Client ID ( ) MAC address ( ) Device name

Client ID:

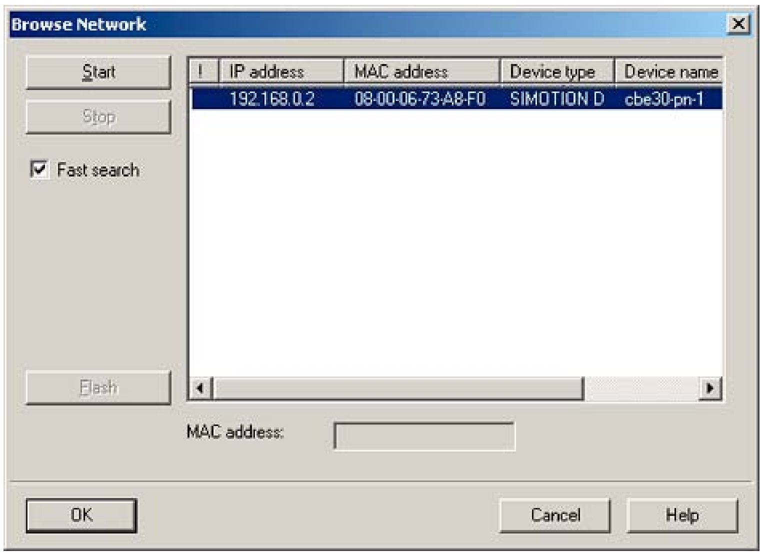
Assign device name

Device name:

Reset to factory settings

2. Click the **Browse** button.

3. The **Browse Network** dialog box opens. The connected nodes are displayed.



!	IP address	MAC address	Device type	Device name
	192.168.0.2	08-00-06-73-A8-F0	SIMOTION D	cbe30-pn-1

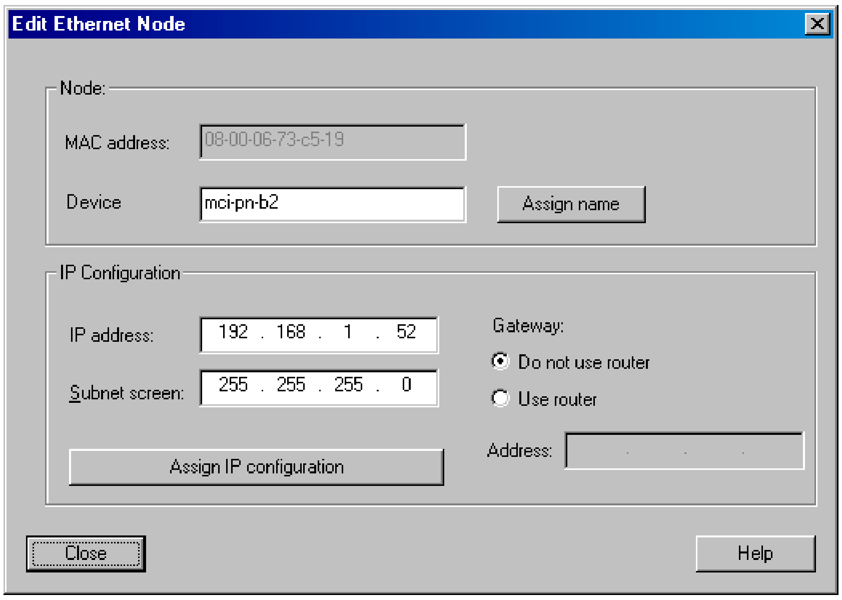
MAC address:

4. Click the device to be initialized and confirm with **OK**.
5. Enter the IP address and subnet mask you specified in the **Properties – Ethernet interface** … dialog.
6. The default setting (**Do not use router**) for the gateway remains unchanged.
7. Click the **Assign IP configuration** button. The IP address is then assigned to the device online.
8. Enter the device name that you have defined in HW Config, see figure **Properties SINAMICS S120**.
9. Click the **Assign name** button. The device name is assigned to the device.

As an alternative, you can perform node initialization in SIMOTION SCOUT.

You can also perform the node initialization in SCOUT.

- In SCOUT, execute **Reachable nodes** and, in the dialog box displayed, right-click the device that you want to edit.
- Execute **Edit Ethernet nodes.** The corresponding dialog box is displayed.



Edit Ethernet Node

Node:
MAC address: 08-00-06-73-c5-19
Device: mci-pn-b2

IP Configuration
IP address: 192 . 168 . 1 . 52
Subnet screen: 255 . 255 . 255 . 0
Gateway: [✓ Do not use router] Use router
Address: . . .

Figure 7-26 Edit Ethernet nodes

- Enter a device name, a subnet mask and an IP address.
- Confirm your entries.

The device name and IP address are transferred to the device and stored there.

7.4 Configuring direct data exchange between IO controllers

7.4.1 Introduction

I/O data areas can be exchanged cyclically between two or more SIMOTION controllers via IRT High Performance. This is also referred to as controller-controller data exchange broadcast. Controller-controller data exchange broadcast is only possible between SIMOTION controllers via PROFINET IO with IRT High Performance.

For data exchange to take place, the devices must be located in a common sync domain and configured accordingly as sync master and sync slaves.

Note

This function is not available for SIMATIC CPUs.

There are in fact two types of data exchange broadcast. One is automatically created by the system (e.g. distributed synchronous operation), while the other can be applied by the user in his or her application. You can configure this second type of data exchange broadcast.

Note

The user must not use the engineering tools to make changes (e.g. amending the address areas) to the data exchange broadcast which has been automatically configured by the system. Doing so will result in error states.

Recommendation

We recommend, initially configure the send areas for all PROFINET devices and then the receive areas. Adopting this procedure will enable you to assign the previously defined send areas when defining the receive areas. This prevents invalid inputs.

Data volume

Around 3 KB of data can be transmitted. Each synchronous relationship configured requires 24 bytes. For example, this means the system would require 5 * 24 bytes if 5 following axes were defined for a master axis. The remaining data is available for application-specific data exchange broadcast.

Note

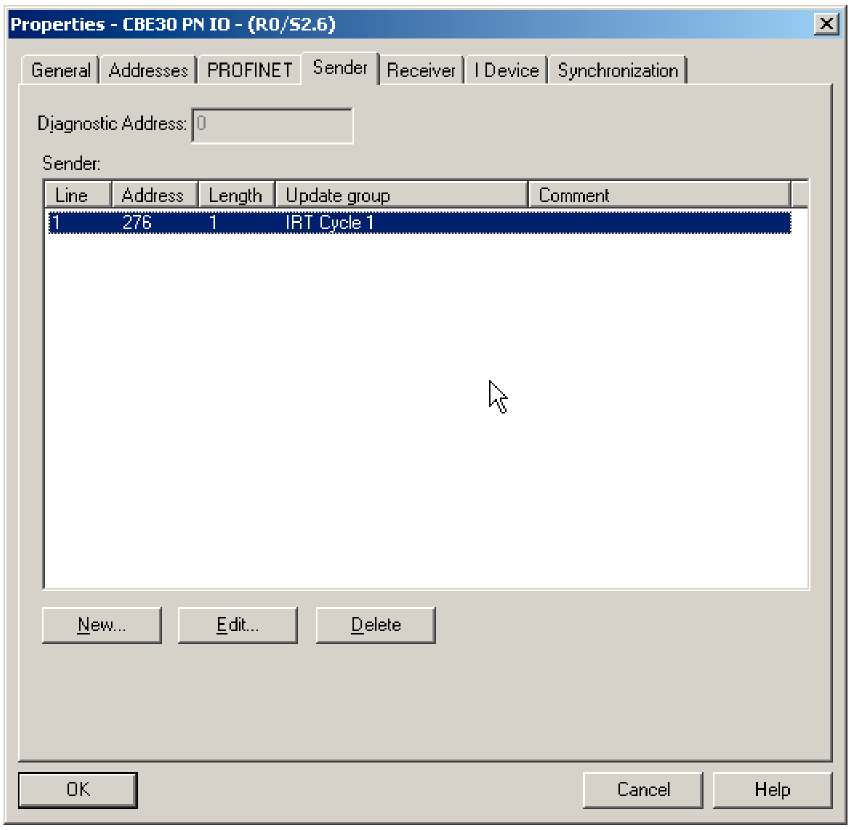
An FAQ section on the subject of PROFINET configuration is provided in SIMOTION Utilities & Applications. SIMOTION Utilities & Applications is provided as part of the SIMOTION SCOUT scope of delivery.

This FAQ section deals with the subjects of distributed gearing and controller-controller data exchange broadcast.

7.4.2 Configuring the sender

Procedure

1. Open the Properties dialog of the PROFINET interface (double-click the corresponding row in the configuration table of HW Config).
2. Select the **Sender** tab.



Properties - CBE30 PN IO - (R0/S2.6)

Diagnostic Address: 0

Sender:

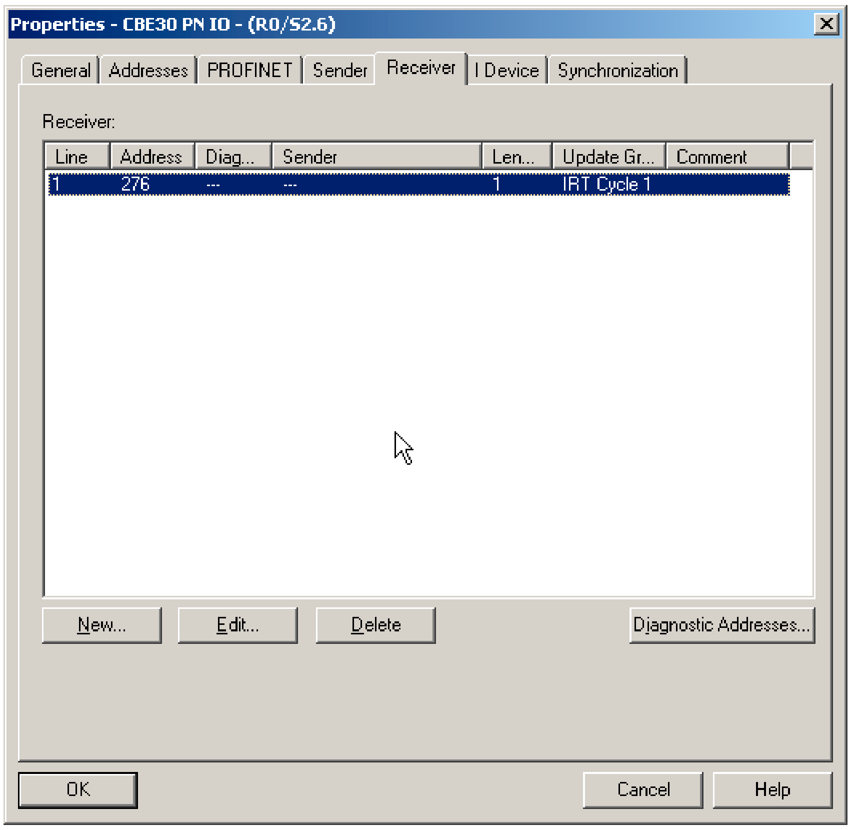
Line	Address	Length	Update group	Comment
1	276	1	IRT Cycle 1	

3. Click the **New** button.
4. Enter in the Properties dialog of the sender, the start address from the I/O area and the length of the address area to be used for sending. Comment the data area so that you will be able to identify the data transmitted via this area later on. A variable may not exceed 254 bytes in size.
5. Confirm the settings with **OK**.
6. Repeat steps 3 to 5 for further send areas.
7. Change the preset diagnostics address for the send areas, if required.
8. Confirm your entries with **OK**.

A single diagnostics address must be assigned for the communication relationship in which a PROFINET interface is the transmitter for direct data exchange.

7.4.3 Configuring the receiver

Procedure



Line	Address	Diag...	Sender	Len...	Update Gr...	Comment
1	276	---	---	1	IRT Cycle 1	

1. Open the Properties dialog of the PROFINET interface (double-click the corresponding row in the configuration table of HW Config).
2. Select the **Receiver** tab.
3. Click the **New** button.
4. Click the **Assign sender** button in the **Properties receiver** dialog.
5. In the **Assign sender** dialog, select the data area of the desired node which is to be received by the local controller.
6. Confirm your selection with **OK**.
7. In the Properties dialog box of the receiver, enter the start address of the address area via which the reception is to be implemented. The length of the address area must not be changed as it is automatically adapted to the length of the send area. The configuration can only be compiled if the send and receive areas have identical lengths!
8. Repeat steps 3 to 7 for further receive areas.
9. A diagnostics address is reserved for each assigned sender via which the receiver can detect a failure of the sender.
10. Click the **Diagnostics addresses** button if you want to edit these addresses.
11. Confirm your entries with **OK**.

7.5 Configuring the iDevice

7.5.1 PROFINET IO and I device

Introduction

Up to SIMOTION 4.0, direct coupling (of SIMATIC and SIMOTION via PROFINET, for example) was only possible via TCP or UDP, or via additional hardware (PN/PN coupler, SIMATIC CP). With SIMOTION V 4.1.1.6, direct coupling of controls - a familiar feature with PROFIBUS - has been introduced for PROFINET IO. It is possible, for example, to connect the SIMOTION as an I slave to the SIMATIC CPU via PROFIBUS. A similar function, referred to as "I device", is also available for PROFINET IO. This supports data exchange between the controls via I/O areas, with the communication programming required for TCP or UDP being replaced by configuring and system functionality. In addition, the costs associated with the hardware solutions used previously no longer apply (PN/PN coupler, SIMATIC-CP).

An I device is a controller which also assumes the function of an IO device. The term "I device" is used to refer to an intelligent IO device. Intelligent I devices are characterized by the fact that their input/output data is pre-processed in the I device, rather than being made directly available to the higher-level IO controller by actual inputs/outputs.

When operating as an I device, a SIMOTION device can be used for data exchange on a SIMATIC station, for example. A SIMOTION device acting as an I device may also be used as a feeder for a modular machine. Please see the *description of functions titled Motion Control basic functions for modular machines*.

Another application for a SIMOTION device acting as an I device involves providing distributed synchronous operation beyond the limits of a project (see the Motion Control Technology Objects Synchronous Operation, Cam Function Manual).

Note

An I device can only be created using SIMOTION V4.1.1.6 or higher.

Properties of an I device

As well as performing the role of an IO device on a higher-level IO controller, an I device can set up its own local PROFINET IO system with built-in local IO devices, thereby acting as an IO controller itself. Both these functions are implemented via the same PROFINET interface on the device.

With SIMOTION, the I device is available for PROFINET IO with RT and IRT High Performance.

The following boundary condition applies in terms of the options available for combining functions:

Table 7- 5 RT and IRT I device combination options with SIMOTION

SIMOTION function	Possible additional functions			
	RT I device	RT controller	IRT I device	IRT controller
RT I device		X	-	X
RT controller	X	-	X*	X*
IRT I device	-	X	-	-
IRT controller	X	X	-	

*Either an IRT I device or an IRT controller

As with any other IO device, an I device's PROFINET interface requires parameter assignment data in order to operate. With IO devices, this data is usually loaded via the associated IO controller in the form of parameter assignment data sets. Two options are available for I devices. An I device's interface and PROFINET interface ports can either be parameterized by the higher-level IO controller or by the I device itself on a local level. The preferred option can be selected as part of the I device's configuration.

If parameters are being assigned locally, the required data is loaded to the I device while the Engineering System is being downloaded. Parameter assignment data for the PROFINET interface is contained in the device download data. The higher-level IO controller does not need be used for parameterizing the PROFINET interface of the I device. However, this option should be used if the I device is to be operated with RT.

If the higher-level IO controller is being used for parameter assignment, the parameter assignment data for the I device's PROFINET interface must be loaded by the IO controller together with the remaining parameter assignment data. For this purpose, the IO controller loads parameter assignment data sets for the PROFINET interface to the I device. If the I device is to be operated with IRT, the parameter assignment data will need to be loaded by the IO controller.

If the I device is being operated with IRT, the I device's send clock must be set to match the send clock for the sync domain of the higher-level IO controller's PROFINET IO system. If the I device is being operated with RT, the I device's update time must be either set to match the send clock for the sync domain of the higher-level IO controller's PROFINET IO system, or down-scaled by a multiple of this send clock.

The table below contains details of the send clocks and update times which must be set, along with their possible combinations.

Table 7- 6 Send clocks/update times of an I device

Higher-level IO controller and I device with IRT, no local PROFINET IO system or local PROFINET IO system with IO devices with RT • I device send clock: – Must be the same as the send clock for the higher-level IO controller – To be set on the I device in <Profinet Interface> properties using the "Send clock" drop-down list box on the PROFINET tab
Higher-level IO controller with IRT and I device with RT, local PROFINET IO system with IRT • I device update time: – Must be an integral multiple of the send clock for the higher-level IO controller and the send clock for the IO controller on the I device – To be set on the substitute I device in <Profinet Interface> properties, under the update time on the "IO Cycle" tab
Higher-level IO controller and I device with RT, no local PROFINET IO system • I device update time: – Any of the possible update times for the I device may be set. – To be set on the substitute I device in <Profinet Interface> properties, under the update time on the "IO Cycle" tab
Higher-level IO controller and I device with RT, local PROFINET IO system with IRT: • I device update time: – Must be less than or equal to the send clock for the IO controller in the I device – To be set on the substitute I device in <Profinet Interface> properties, under the update time on the "IO Cycle" tab

The figure below shows how an I device can be configured on a higher-level IO controller. The higher-level IO controller sets up a PROFINET IO system containing the I device. The I device is able to set up a local PROFINET IO system. Each of these PROFINET IO systems can belong to its own sync domain. However, the I device must only be assigned to one of the possible sync domains, as a PROFINET interface can only belong to a single sync domain.

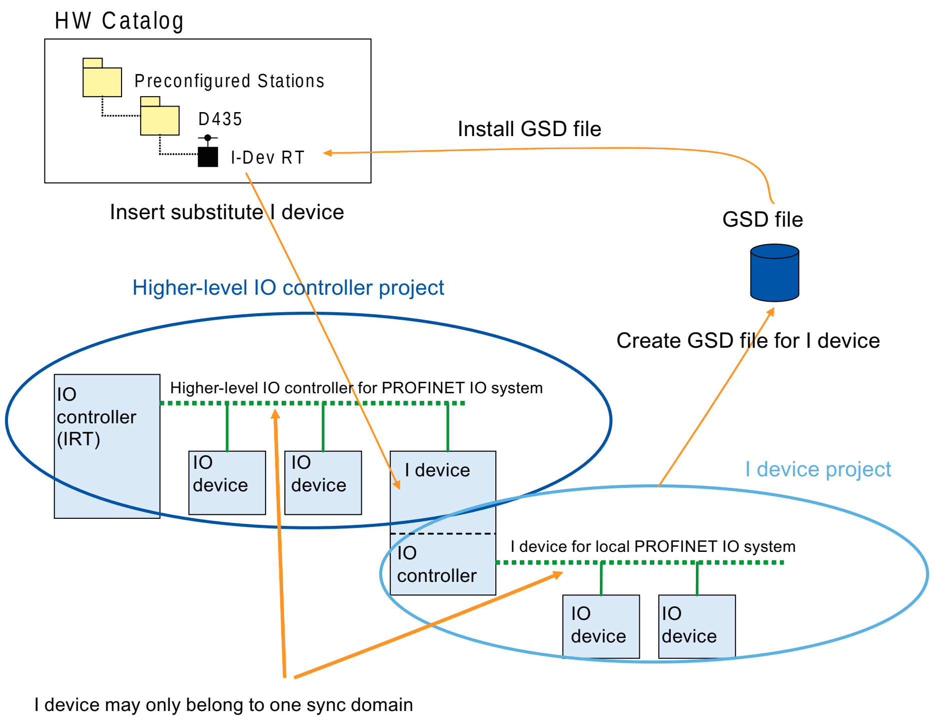

HW Catalog
Preconfigured Stations
D435
I-Dev RT
Install GSD file
Insert substitute I device
GSD file
Higher-level IO controller project
Create GSD file for I device
IO controller (IRT)
Higher-level IO controller for PROFINET IO system
IO device
IO device
I device
I device project
IO controller
I device for local PROFINET IO system
IO device
IO device
I device may only belong to one sync domain

Figure 7-27 I device configuration

Configuration procedure

- The I device itself and the IO controller on which it is to be operated should be created in different projects.
- The PROFINET interface's I device mode must be active for the I device. In addition, the input and output ranges in the I device must be configured for data exchange with the higher-level IO controller.
- After an I device has been created and configured, a GSD file needs to be created and installed for its substitute I device. The substitute I device will then be available in the hardware catalog under "Preconfigured Stations".
- The next step involves inserting the substitute I device from "Preconfigured Stations" in the hardware catalog into the higher-level IO controller's PROFINET IO system.

Since a manual process in the hardware catalog is required to create a substitute I device, there is no automatic alignment between the project with the I device and the corresponding substitute I device in the GSD file. As a result, no subsequent amendments can be made to the I device configuration. If an amendment is made, however, a new GSD file will have to be created and installed. Where numerous amendments have subsequently been made to the configuration of a given I device, with multiple GSD files created and installed by the I device, the version shown under "Preconfigured Stations" in the hardware catalog will always be the most recent one. The version will only be updated, however, if the identifier used for the substitute I device when creating and installing the GSD file remains the same. Only the input and output addresses for data exchange may be amended in the project for the higher-level IO controller.

Since I devices connected to their higher-level IO controller and IO devices connected on the PROFINET IO system of a single I device are connected via one and the same PROFINET interface, they will also be located in one and the same Ethernet subnet. This means that the device names and IP addresses of all these devices must be different from each other, and the subnet masks must be identical. It is particularly important to bear this in mind if the higher-level IO controller and the I device are in different projects, as HW Config cannot check that device names, IP addresses, and subnet masks are consistent across different projects.

Device name (NameOfStation) for the I device

As with all IO devices on PROFINET IO, a device name also has to be defined for the I device in the configuration. As the device name (NameOfStation) for the I device is set in the properties for its PROFINET interface, it is identical to the device name of the IO controller in the I device. The name set is written to the GSD file for the substitute I device when this file is created and installed. When the substitute I device is inserted into the higher-level IO controller's PROFINET IO system, the device name previously assigned in the GSD file will be accepted into the configuration. In all cases, it is important to ensure that the device name in the configuration of the higher-level IO controller is the same as the device name defined for the I device. As a result, device names must not be amended after the higher-level IO controller has been added to the PROFINET IO system.

If the device names are different, the higher-level IO controller will be unable to power up the relevant I device and, consequently, be unable to commence cyclic exchange of input/output data.

The following scenarios result in different device names and must therefore be avoided:

- If you use a SIMOTION device as the higher-level IO controller, the device name (NameOfStation) of the I device may not contain any "-" marks, as these will be changed into "x" when you insert the I device into the PROFINET IO system.
- Since a device name must not be used twice within an Ethernet subnet, any device name which already exists will be amended when a substitute I device is inserted into the PROFINET IO system of its higher-level IO controller. In view of this, it is important to ensure that the device name previously assigned in the GSD file is not used at this stage.
- If more than one substitute I device from the same "Preconfigured Stations" entry is inserted into the higher-level IO controller's PROFINET IO system, the device name previously assigned in the GSD file will be amended. With this in mind, a substitute I device must also be created for every I device to be used in a PROFINET IO system.

7.5.2 Creating an I device

Requirement

You have already created a project and created a station with rack or a SIMOTION controller in HW Config (SIMATIC Manager or SIMOTION SCOUT). You have already configured the PROFINET IO system and now want to configure the I device.

Note

When configuring the I device, observe the possible settings for the RT class (see PROFINET I device).

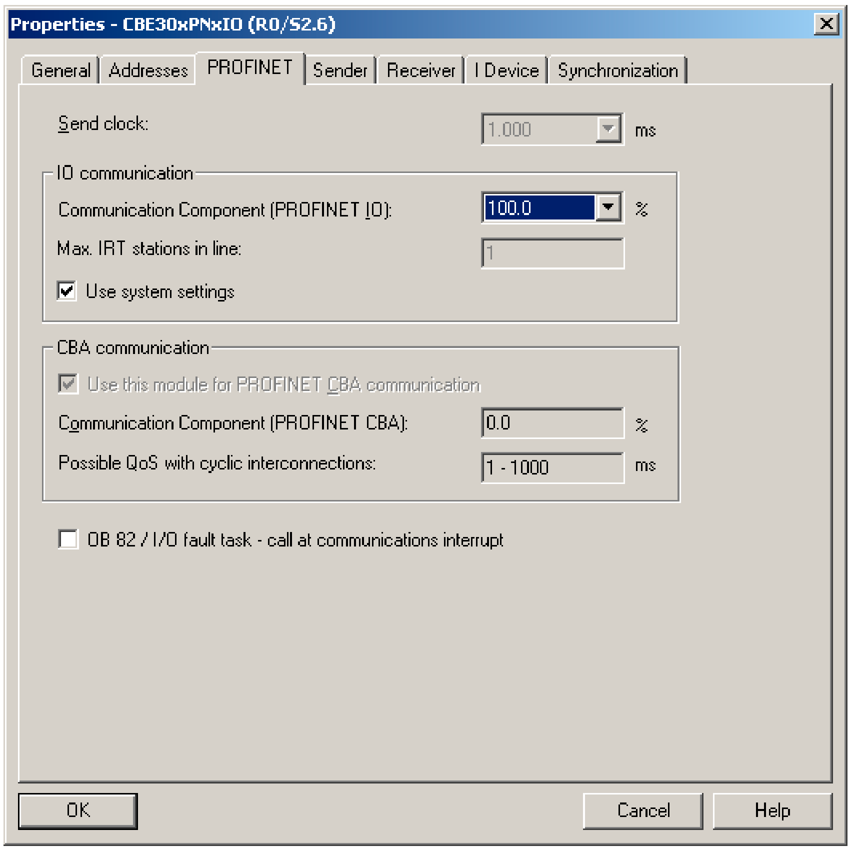
Procedure

1. Double-click the interface module of the CPU. The Properties dialog opens.
2. Select the General tab and, if necessary, change the device name (without using any "-").

   Figure: General tab in dialog
3. Select the I device tab.

   Figure: I Device tab in dialog
4. Select the I device mode.
5. Select whether "Parameterization of the PN interface and of its ports on higher-level IO controller" should be performed. You should always select this option if cyclic communication between the higher-level IO controller and the I device is to operate with IRT. This will also result in ports being created in the GSD file and parameter assignment data sets being loaded to the I device's controller on start-up. If you do not select this option, cyclic communication between the higher-level IO controller and the I devices can only take place via RT.
6. Select Operate I device/application in isochronous mode if you want to implement communication with IRT. On the substitute I device, this selection also causes the Application tab in the "Properties" dialog for the substitute I device's PROFINET interface to be shown. It will then be possible to select Operate I device/application in isochronous mode on this tab so that the I device can be operated isochronously. If an I device is to be operated with IRT, the parameter assignment for the PN interface and its ports will also need to be set as Operate I device/application in isochronous mode on the higher-level IO controller.
7. If the I device is being operated with IRT, its send clock will have to be set. To do this, select the PROFINET tab and set the send clock as appropriate.
8. Click New..., each time to create the virtual subslots (input and output address), and configure these according to your requirements. By doing this, you are configuring the I/O range for the I device via which data is exchanged with the higher-level IO controller. Do not perform any further settings in the Sender and Receiver tabs.

9. Click **OK** to accept these settings and save the project.
10. Continue by creating the substitute I device.



Send clock: 1.000 ms

IO communication
Communication Component (PROFINET IO): 100.0 %
Max. IRT stations in line: 1
☑ Use system settings

CBA communication
☑ Use this module for PROFINET CBA communication
Communication Component (PROFINET CBA): 0.0 %
Possible QoS with cyclic interconnections: 1 - 1000 ms

☐ OB 82 / I/O fault task - call at communications interrupt

Figure 7-28 Setting the I device send clock

7.5.3 Exporting the GSD file for the I device

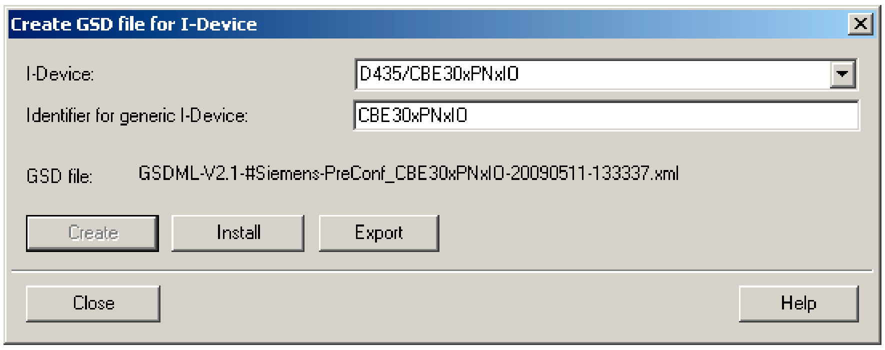
The GSD file always needs to be exported so that an I device in a project can be used on another PC.

Requirement

You have already configured the module to be used as an I device.

1. First save the project.
2. Select **Options > Create GSD file for I device...**. The "Create GSD file for I device" dialog opens.

3. Select the I device and enter a name for the substitute I device. The substitute I device will appear under this name in "Preconfigured Stations" in the HW catalog.



Create GSD file for I-Device

I-Device: D435/CBE30xPNxIO

Identifier for generic I-Device: CBE30xPNxIO

GSD file: GSDML-V2.1-#Siemens-PreConf_CBE30xPNxIO-20090511-133337.xml

4. Click **Create** and then **Export**. The **Find folder** dialog opens.
5. Select the path in which the GSD file of the substitute I device is to be stored and click **OK**.

7.5.4 Creating a substitute I device

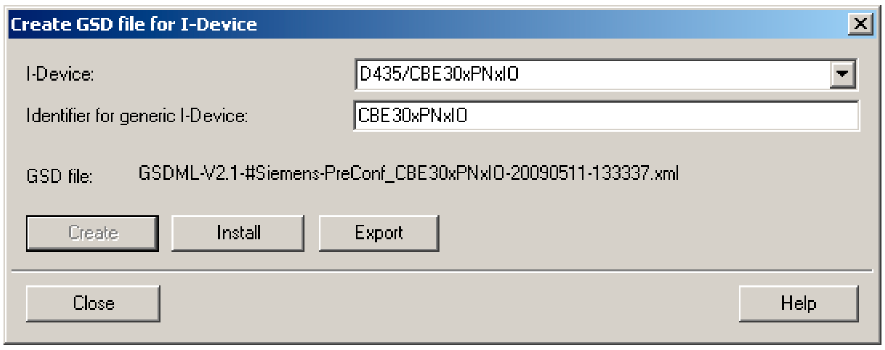
There are two different ways of creating a substitute I device. The first, **Options > Install GSD files ...** , involves a previously exported GSD file. The second involves the **Create GSD file for I device** dialog.

Requirement

You have already configured the module to be used as an I device and exported the GSD file from this.

Procedure 1:

1. First save the project.
2. Create an I device as described in Chapter Exporting the GSD file for the I device (Page 201). There is no need to export the file; instead, it is installed immediately.
3. Click **Create** and then **Install**.



Create GSD file for I-Device

I-Device: D435/CBE30xPNxIO

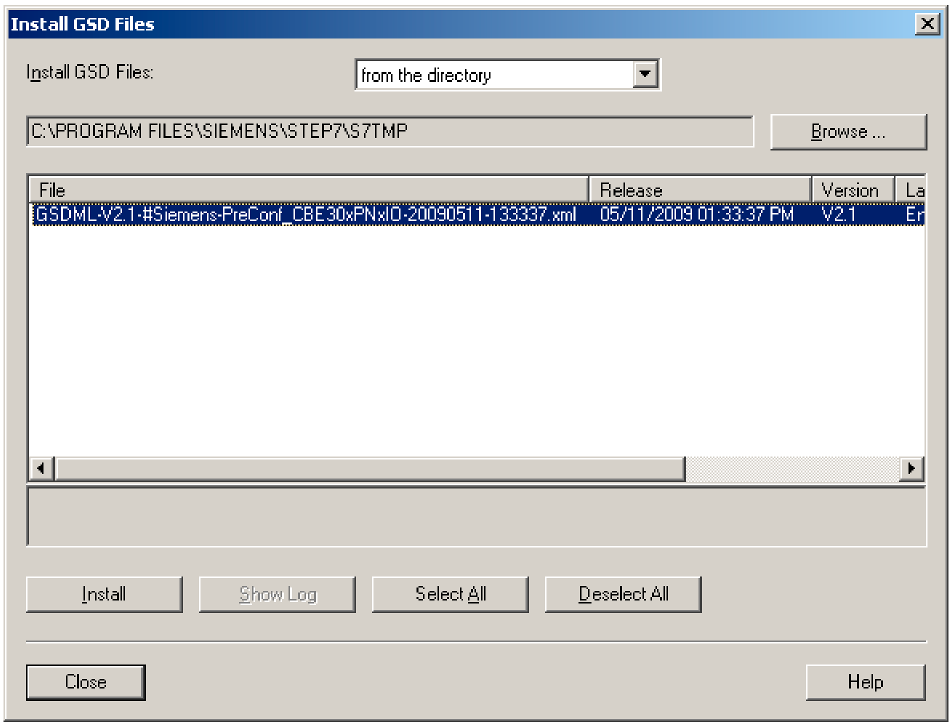
Identifier for generic I-Device: CBE30xPNxIO

GSD file: GSDML-V2.1-#Siemens-PreConf_CBE30xPNxIO-20090511-133337.xml

4. Click **Close**. The substitute I device will now be available under "Preconfigured Stations".

Procedure 2:

1. Select **Options > GSD files...**. The "Install GSD files" dialog box opens.
2. Click **Browse...** . The **Find folder** dialog opens.
3. Select the path in which the GSD file of the substitute I device is stored and click **OK**.



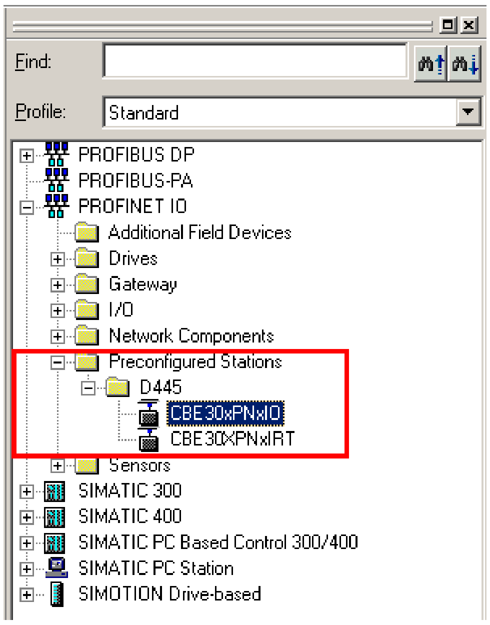
Install GSD Files

Install GSD Files: from the directory

C:\PROGRAM FILES\SIEMENS\STEP7\S7TMP

File	Release	Version	La
GSDML-V2.1-#Siemens-PreConf_CBE30xPNxIO-20090511-133337.xml	05/11/2009 01:33:37 PM	V2.1	En

4. Select the required GSD files and click **Install**.
5. Click **Close**. The substitute I devices will now be available under "Preconfigured Stations".



Find:
Profile: Standard

```
PROFIBUS DP
PROFIBUS-PA
PROFINET IO
    Additional Field Devices
    Drives
    Gateway
    I/O
    Network Components
    Preconfigured Stations
        D445
            CBE30xPNxIO
            CBE30XPNxIRT
    Sensors
SIMATIC 300
SIMATIC 400
SIMATIC PC Based Control 300/400
SIMATIC PC Station
SIMOTION Drive-based
```

Figure 7-29 I device entry in the hardware catalog

7.5.5 Inserting a substitute I device in the higher-level IO controller

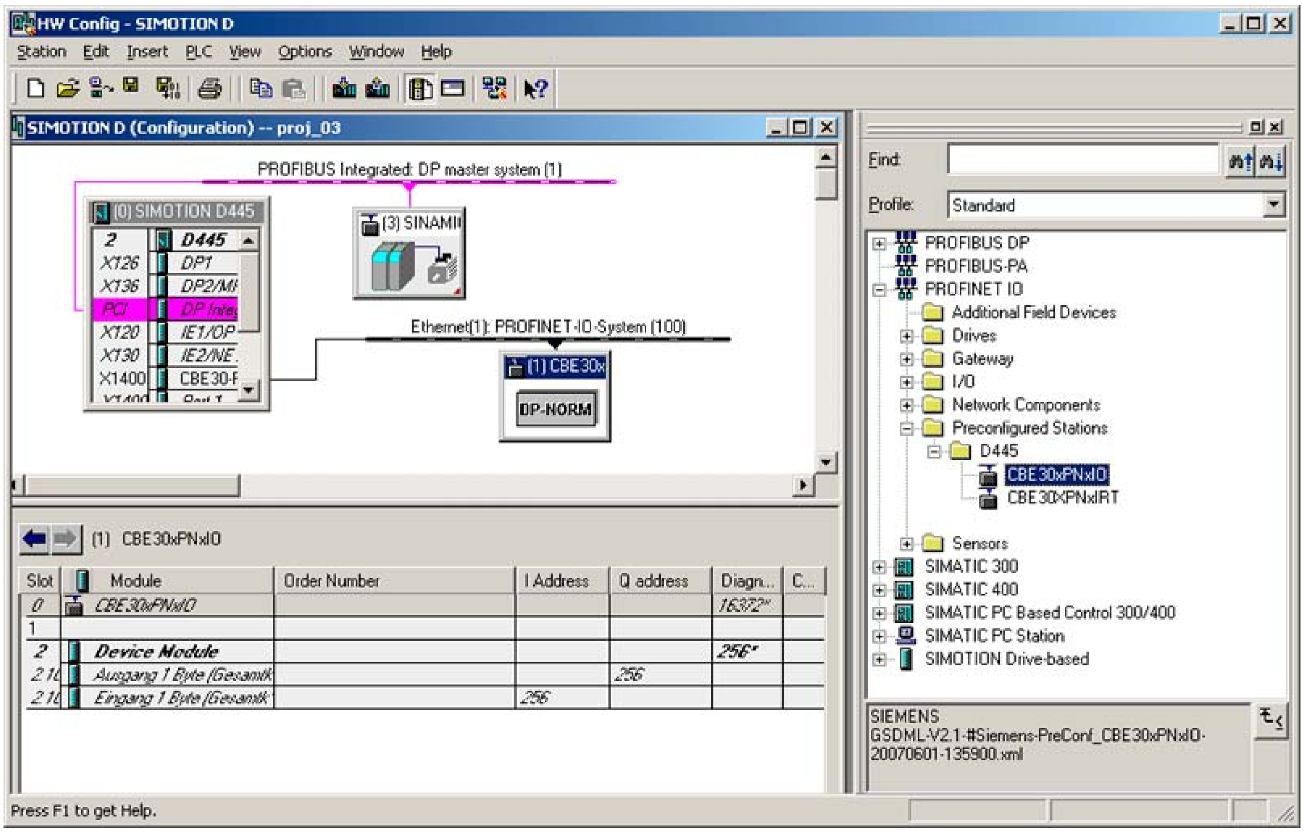
Requirement

You have already created a substitute I device. A project is open and an IO controller with a PROFINET IO system has already been configured.

Inserting a substitute I device

1. Open the hardware catalog.
2. Drag the relevant substitute I device from the hardware catalog (PROFINET IO > Preconfigured Stations) to the PROFINET IO system. The substitute I device is displayed as a normal IO device on the PROFINET IO system. Whether or not ports are displayed depends on whether **Parameter assignment of the PN interface and its ports on higher-level IO controller** is selected.

The figure below shows a higher-level IO controller with an RT I device.



(1) CBE30xPNxIO

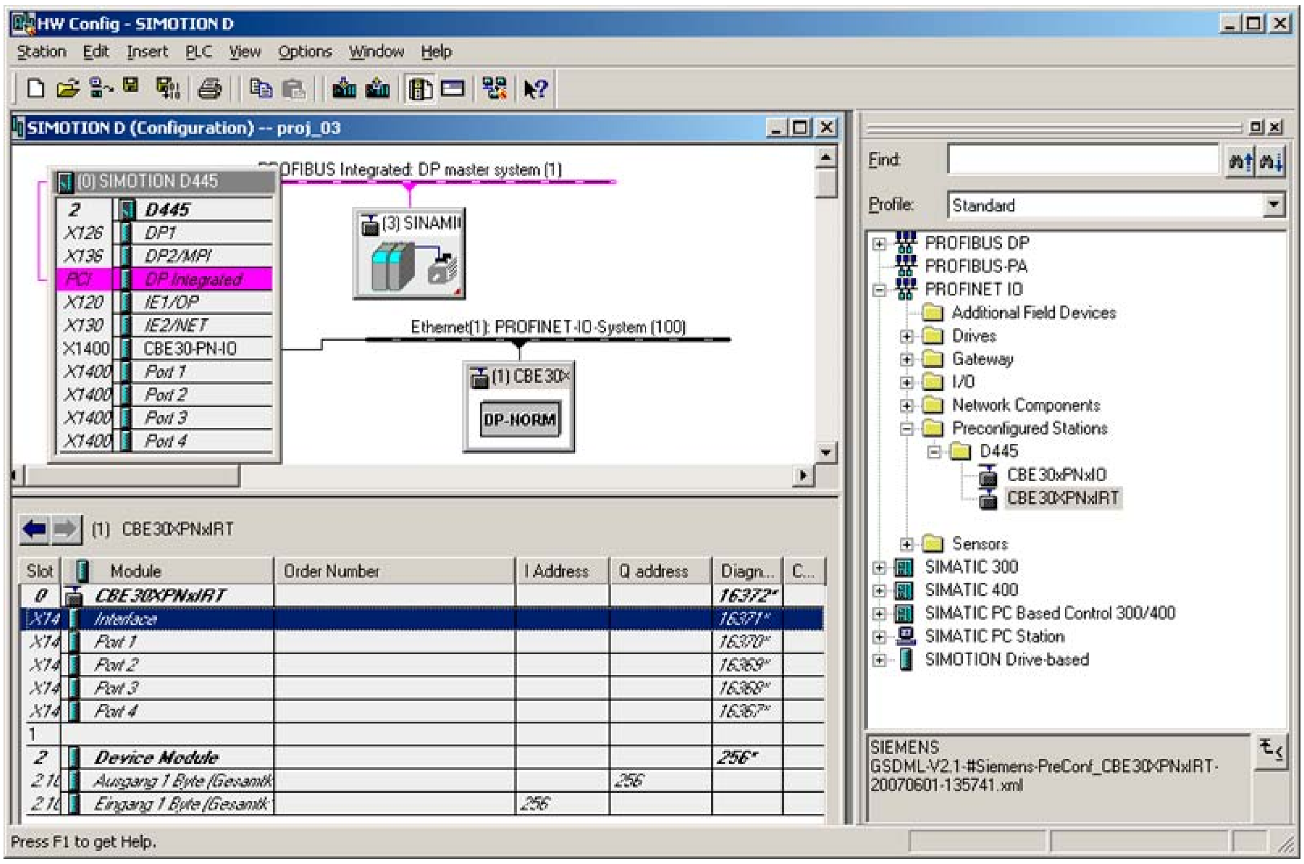
Slot	Module	Order Number	I Address	Q address	Diagn...	C...
0	CBE30xPNxIO				16372*	
1						
2	**Device Module**				256*	
2.1(	Ausgang 1 Byte (Gesamtk			256		
2.1(	Eingang 1 Byte (Gesamtk		256			

2	D445
X126	DP1
X136	DP2/MI
PCI	DP Inte
X120	IE1/OP
X130	IE2/NE
X1400	CBE30-F

SIEMENS
GSDML-V2.1-#Siemens-PreConf_CBE30xPNxIO-20070601-135900.xml

Figure 7-30 RT I device on the IO controller

The figure below shows a higher-level IO controller with an IRT I device.



(0) SIMOTION D445

2	D445
X126	DP1
X136	DP2/MPI
PCI	DP Integrated
X120	IE1/OP
X130	IE2/NET
X1400	CBE30-PN-IO
X1400	Port 1
X1400	Port 2
X1400	Port 3
X1400	Port 4

(1) CBE30XPNxIRT

Slot	Module	Order Number	I Address	Q address	Diagn...	C...
0	CBE30XPNxIRT				16372*	
X14	Interface				16371*	
X14	Port 1				16370*	
X14	Port 2				16369*	
X14	Port 3				16368*	
X14	Port 4				16367*	
1						
2	Device Module				256*	
2 1	Ausgang 1 Byte (Gesamtk			256		
2 1	Eingang 1 Byte (Gesamtk		256			

SIEMENS
GSDML-V2.1-#Siemens-PreConf_CBE30XPNxIRT-20070601-135741.xml

Figure 7-31 I device on the IO controller

The number of submodules corresponds to the number of the configured submodules of the I device in the GSD file. The module and the submodules (virtual subslots) cannot be deleted.

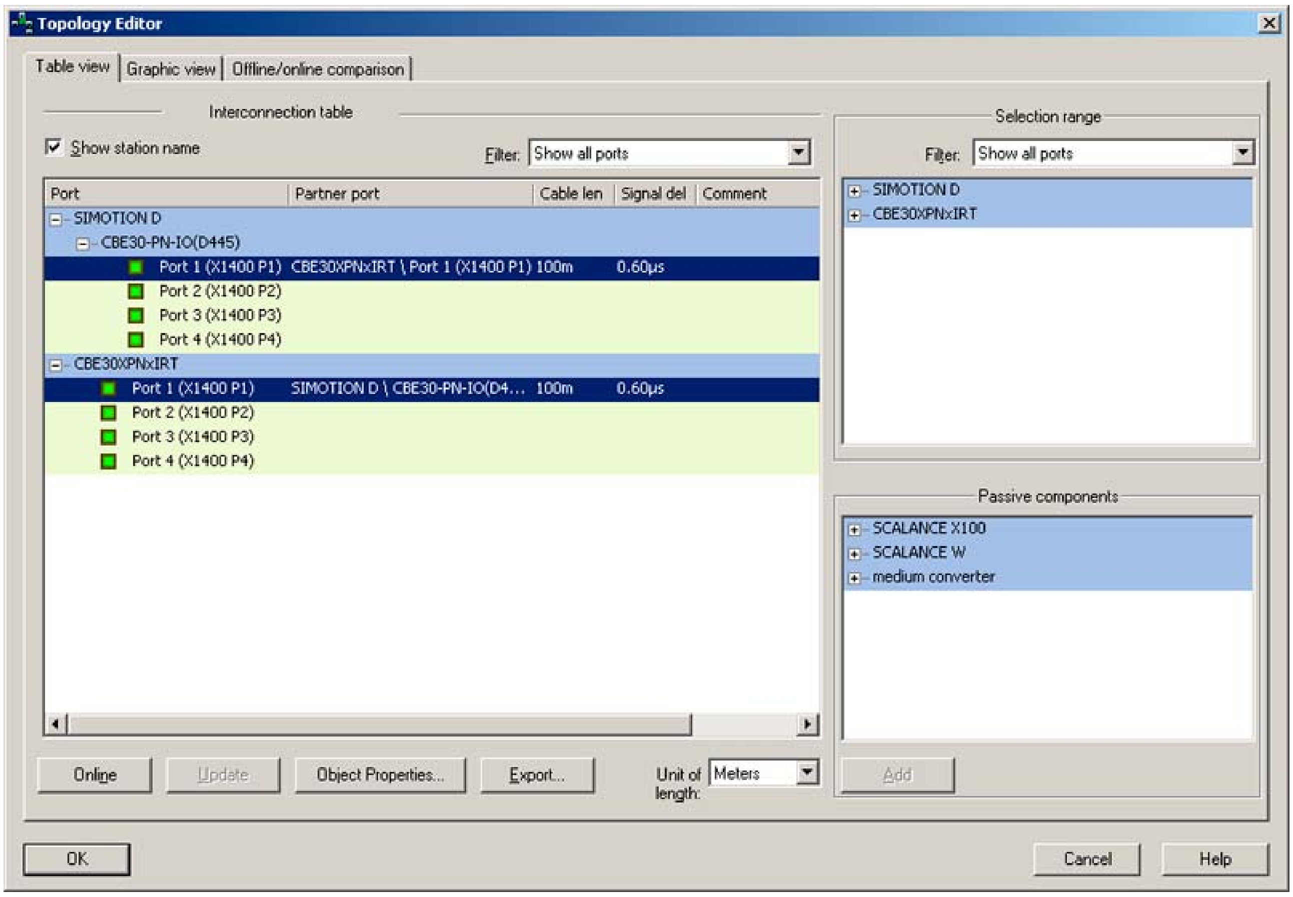

Topology Editor

Table view | Graphic view | Offline/online comparison

Interconnection table

☑ Show station name

Filter: Show all ports

Port	Partner port	Cable len	Signal del	Comment
⊟ SIMOTION D				
⊟ CBE30-PN-IO(D445)				
🟩 Port 1 (X1400 P1)	CBE30XPNxIRT \ Port 1 (X1400 P1)	100m	0.60µs	
🟩 Port 2 (X1400 P2)				
🟩 Port 3 (X1400 P3)				
🟩 Port 4 (X1400 P4)				
⊟ CBE30XPNxIRT				
🟩 Port 1 (X1400 P1)	SIMOTION D \ CBE30-PN-IO(D4...	100m	0.60µs	
🟩 Port 2 (X1400 P2)				
🟩 Port 3 (X1400 P3)				
🟩 Port 4 (X1400 P4)				

Selection range

Filter: Show all ports

- ⊞ SIMOTION D
- ⊞ CBE30XPNxIRT

Passive components

- ⊞ SCALANCE X100
- ⊞ SCALANCE W
- ⊞ medium converter

Unit of length: Meters

Figure 7-32 Interconnecting I device ports

Assigning the IP address for the substitute I device

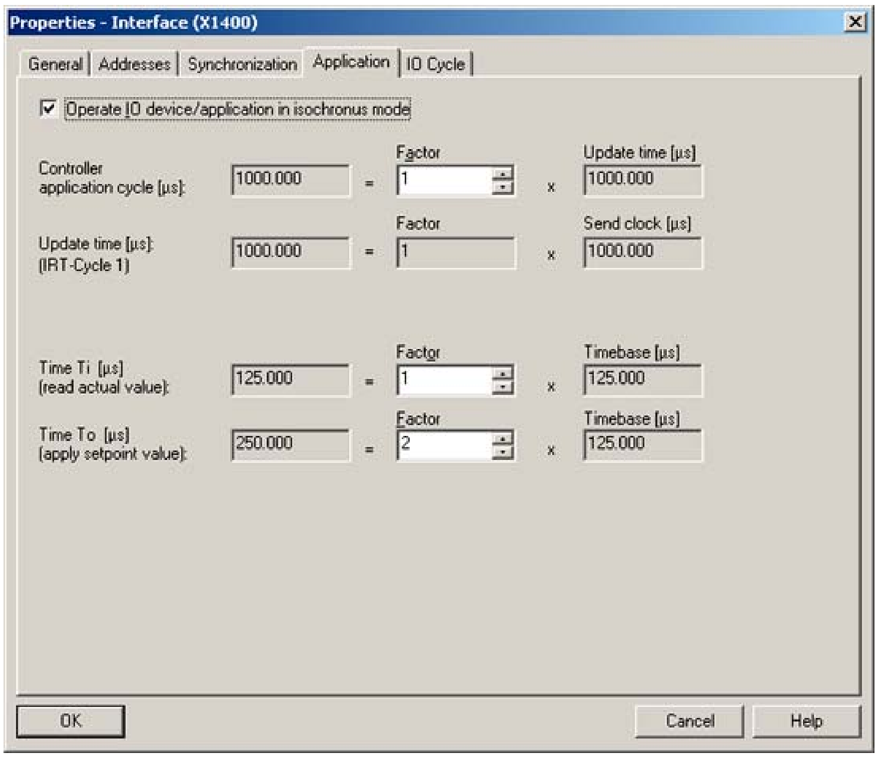
1. Double-click the I device to display the **Properties** dialog.
2. Disable the **Assign IP address via IO controller** option.

   The IP address should not be assigned by the higher-level IO controller because it already assigns this in the Step7 project of the I device.

Setting the synchronization type and isochronous mode for an I device with IRT

1. Double-click the interface entry (X1400) to display the **Interface properties** dialog.
2. In the **Synchronization** tab, select **Sync-Slave** and **IRT** as synchronization type and RT class respectively.

   Only then are the fields in the **Application** tab active.
3. Select the **Operate IO device/application in isochronous mode** option in the **Application** tab and configure the application appropriately for your requirements.



Properties - Interface (X1400)

General | Addresses | Synchronization | Application | IO Cycle

[✓] Operate IO device/application in isochronus mode

			Factor		
Controller application cycle [µs]:	1000.000	=	1	x	Update time [µs] 1000.000
Update time [µs]: (IRT-Cycle 1)	1000.000	=	1	x	Send clock [µs] 1000.000
Time Ti [µs] (read actual value):	125.000	=	1	x	Timebase [µs] 125.000
Time To [µs] (apply setpoint value):	250.000	=	2	x	Timebase [µs] 125.000

Figure 7-33 I device application

Setting the update time and send clock

I device with RT

- The update time needs to be set for I devices with RT. Double-click the PROFINET IO system and select the **Update time** tab in the **PROFINET subnet properties** dialog. Set the update time there.

I device with IRT

- The update time needs to be set for I devices with IRT. The setting made for the send clock in the I device's project must be the same as for the send clock in the higher level IO controller's project. Set the send clock for the higher-level project in HW Config using **Edit > PROFINET IO > Domain Management**.

7.5.6 Deleting a substitute I device

The GSD files for the substitute I device can be found under "Preconfigured Stations" in the following directory:

<Program Files>\Siemens\Step7\S7DATA\GSD,
e.g. GSDML-V2.2-#Siemens-PreConf_**I-Dev RT**-20080704-095947.xml.

Here, **I-Dev RT** is the name of the substitute I device. You can delete the substitute I device by deleting the corresponding XML files. The substitute I devices displayed under "Preconfigured Stations" are only updated when a new GSD file is created and installed.

7.6 Loading the communication configuration

7.6.1 Loading the PROFINET IO configuration

Requirement

A PG/PC with which you can go ONLINE is connected.

Procedure

The configuration data must be loaded to all participating controllers once PROFINET IO configuration has been successfully completed.

1. In NetPro, select the Ethernet subnet and then select the **Target system > Loading in current project > Nodes on the subnet** menu command.

7.7 Data exchange between SIMATIC and SIMOTION via PROFINET

7.7.1 Data exchange through the use of iDevices

Description

SIMATIC and SIMOTION can be linked using the following functions:

- TCP/UDP*) user communication
- PROFINET IO/RT, via S7-300 CP as a device

- PROFINET IO/RT, via PN/PN coupler
- PROFINET IO/RT, via I device

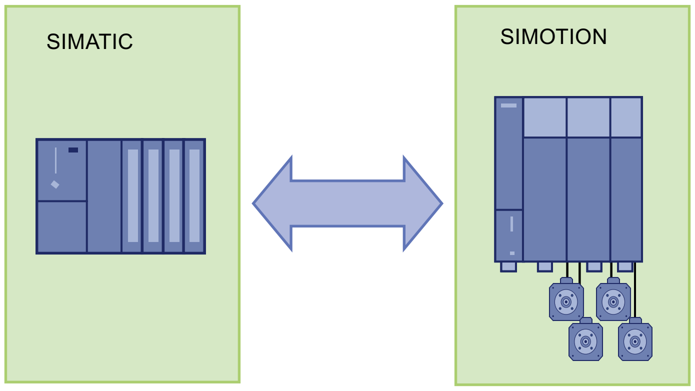


Figure 7-34 Data exchange between SIMOTION and SIMATIC

More information

An FAQ section on the subject of coupling a PROFINET RT I device between a SIMOTION control and a SIMATIC control is provided in SIMOTION Utilities & Applications. SIMOTION Utilities & Applications is provided as part of the SIMOTION SCOUT scope of delivery.

It looks at the following three application cases:

Case A: SIMOTION and SIMATIC as an I device in a project; SIMOTION as a controller in a second, separate project

Case B: One project for all components

Case C: Multiple use of an I device

For more detailed information on the configuration of I devices, please refer to Chapter Configuring the iDevice (Page 195).

Please also refer to the FAQ on configuration I device FAQ (http://support.automation.siemens.com/WW/view/en/29578823).

7.7.2 PN-PN coupler

Description

The PN/PN coupler is used to link two PROFINET IO system with one another and to exchange data between them. The maximum size of the data which can be transferred is 256-byte input data and 256-byte output data.

As a device, the PN/PN coupler has two PROFINET interfaces, each of which is linked to another subnet.

Note

The PN/PN coupler can only be implemented as a device with RT class **RT**.

During configuration, two IO devices are derived from a PN/PN coupler which means that there is one IO device for each controller with its own subnet. The other part of PN/PN coupler in each case is known as the bus node. Once configuring is complete, the two parts are joined.

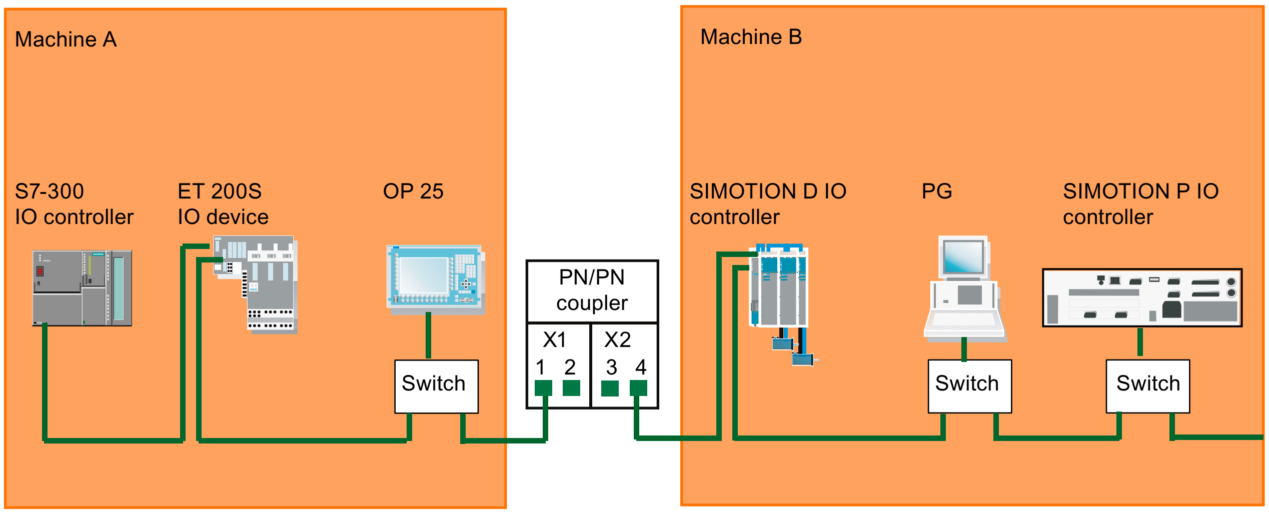


Figure 7-35 Coupling two PROFINET subnets with one PN/PN coupler

The two machines in the figure are isolated via the PN/PN coupler. If it is also possible, for example, to use the programming device of machine B to go online on machine A, a jumper can be inserted between ports 2 and 3 in the PN/PN coupler. This removes the isolation.

Note

Detailed information about the PN/PN coupler is contained in the appropriate device documentation.

Configuring the PN-PN coupler

Two PROFINET devices are created for the purpose of configuring the PN/PN coupler: one on the left (X1) and one on the right (X2). You use HW Config to configure the PN/PN coupler. Once both subnets in a project have been configured, you can use STEP 7 to configure the PN/PN coupler for both subnets. Once the subnets in various projects have been configured, you must configure the coupler in each project.

Note

The PN/PN coupler is used for data exchange between SIMOTION and SIMATIC. It is also, however, the preferred solution for fast exchange of F signals between SIMATIC CPUs.

7.7.3 Communication using standard protocols

Description

TCP and UDP can be used to exchange data between SIMOTION, SIMATIC, other controls, and third-party systems. The PROFINET interface is a standard Ethernet interface and supports these protocols. The user program must handle the management of the connection. You can use this connection, for example, to exchange data between a SIMATIC CPU and SIMOTION controller via PROFINET.

UDP is a connectionless protocol; this means transmission is not followed by feedback indicating whether the receiver actually received the message. TCP is a connection-based protocol; this means a logical connection is established at the outset. Transmission only begins once this connection is present. The connection (i.e. message receipt) is monitored.

Data exchange via TCP:

- Establishing a connection
- Data management
- Connection monitoring
- Connection termination

The use of the system commands is described in detail in the **Introduction to Ethernet (TCP/IP and UDP connections)** section.

See also

_tcpOpenServer function (Page 125)

_tcpOpenClient function (Page 126)

_tcpReceive function (Page 126)

_tcpSend function (Page 127)

_tcpCloseConnection function (Page 127)

_tcpCloseServer function (Page 128)

Function _udpSend (Page 128)

Function _udpReceive (Page 129)

7.8 Diagnostic and alarm behavior

7.8.1 PROFINET IO alarm and diagnostic messages to SIMOTION

Description

For PROFINET IO there is an alarm and diagnostic functionality for PROFINET devices.

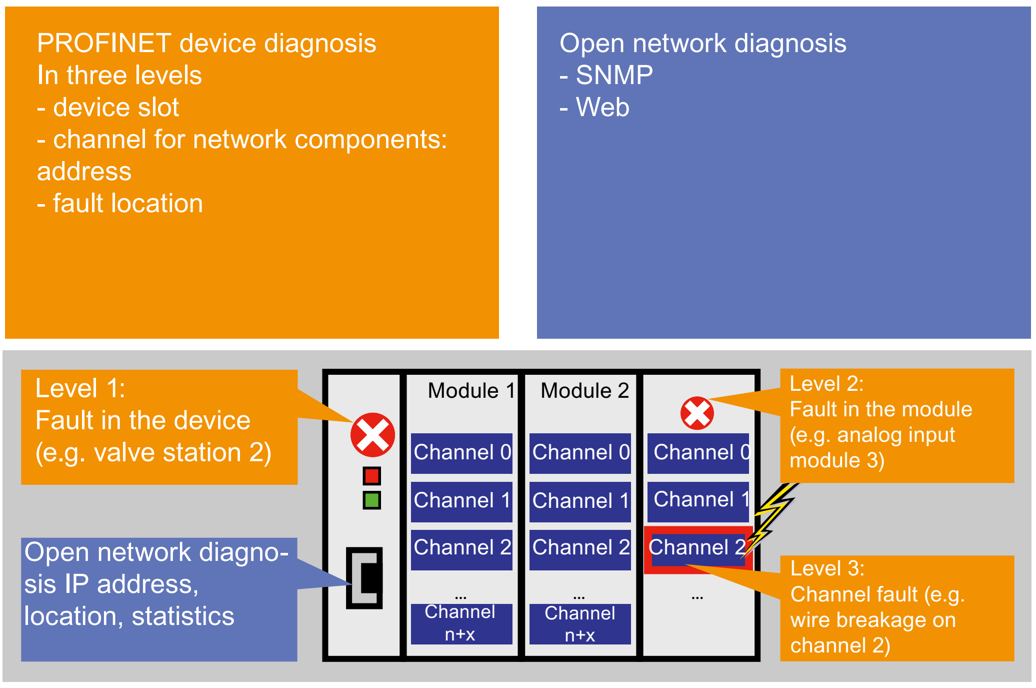

PROFINET device diagnosis
In three levels
- device slot
- channel for network components: address
- fault location
Open network diagnosis
- SNMP
- Web
Level 1: Fault in the device (e.g. valve station 2)
Open network diagnosis IP address, location, statistics
Module 1
Module 2
Channel 0
Channel 1
Channel 2
...
Channel n+x
Level 2: Fault in the module (e.g. analog input module 3)
Level 3: Channel fault (e.g. wire breakage on channel 2)

Figure 7-36 Diagnostics overview

Device diagnostics

The device diagnosis can be divided into three levels. For detailed information, see "Diagnostic model".

7.8.2 Diagnostics model

With PROFIBUS DP, a diagnostics message frame is transferred to the master for diagnostic and status messages of a slave. A diagnostics message frame contains the entire diagnostic status of a slave. Only the representation of parameterization and configuration errors is standardized in a DP diagnostics message frame. Further diagnostic and status messages can be added, but these are coded manufacturer-specific.

Right from the start, PROFINET IO uses completely standardized diagnostics mechanisms. This is especially helpful for manufacturer-wide device and system diagnostics.

Because of the large quantity structures, it is not possible to keep the status information of all stations in the IO controller. Therefore only the current diagnostic events are transferred to the IO controller via the standardized alarms.

The use of an acknowledged service enables the transfer of the diagnostic events in causal sequence. The status of a station is saved by this and can be read out by a diagnostic system at any time and directly via standardized data records, see corresponding STEP7 documentation.

Access to the alarm and diagnostic data

For PROFINET IO, a differentiation is made between the following alarm and diagnostic messages:

- Alarms sent from IO devices to the IO controller
- Alarms that occur in the IO controller

The following figure shows the access possibilities to the diagnostic data:

1. Diagnostics on PG
   The PG reads the diagnostics directly from the IO device. Visualization takes place in the PG.
2. Diagnostics on controller
   The IO device sends the diagnostics to the IO controller, the response to the fault takes place in the controller.

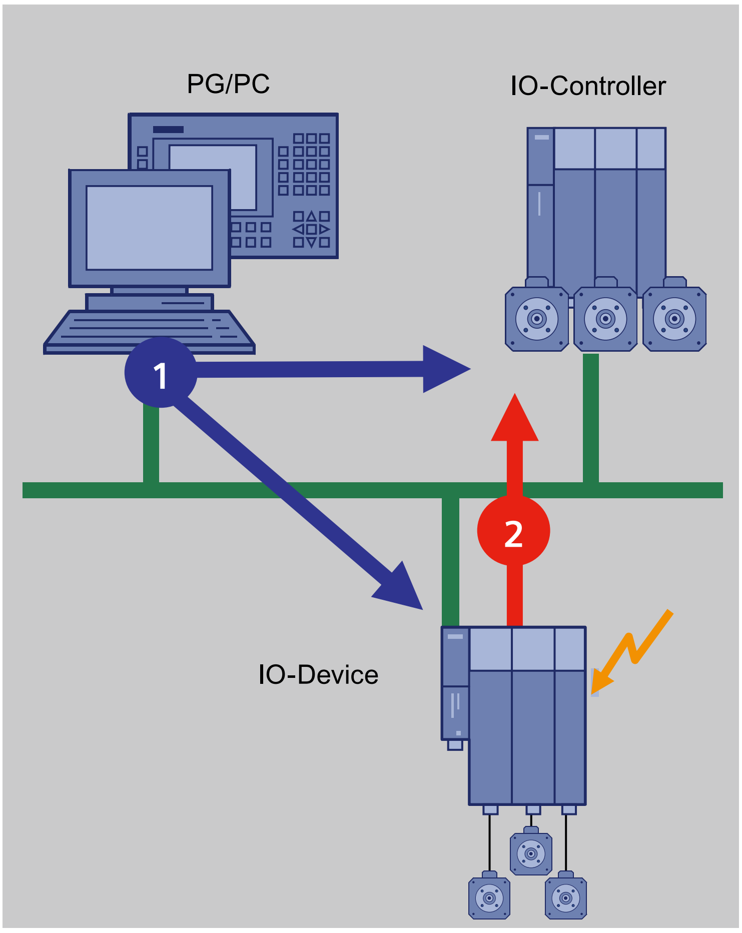


Figure 7-37 Access to the diagnostic data

7.8.3 Alarms on the IO controller

Description

A number of alarms are issued on the IO controller. Occurring alarms are listed with the corresponding EventID in the diagnostics buffer of SIMOTION. The following alarms are possible:

- Alarms for direct data exchange between IO controllers
- Station alarms reported by the PROFINET interface

The following table shows PROFINET IO alarms as they are represented in SIMOTION:

Alarm (TSI#InterruptId)	TSI#eventClass	TSI#faultId	Meaning
Station failure (_SC_STATION_DISCONNECTED (= 202))	16#39	16#CA	PROFINET IO system error: in this case there is only an incoming event; an outgoing event is represented on 16#38 - 16#CB for each IO device present.
		16#CB	Station failure of an IO device
		16#CC	IO device fault present. Channel diagnostics or manufacturer-specific diagnostics pending.
Station reconnection (_SC_STATION_RECONNECTED (= 203)	16#38	16#CB	An IO device has been reconnected without errors
		16#CC	IO device error corrected
		16#CD	An IO device has been reconnected, but with an error: set configuration <> actual configuration
		16#CE	An IO device has been reconnected, but error during module parameterization

Use of the TaskStartInfo

Information concerning the TaskStartInfo for the PeripheralFaultTask is contained in the **Base Functions** manual.

7.8.4 Alarms from the IO device to the IO controller

Description

The alarms are transferred using the PROFINET alarm mechanism from the IO device to its associated IO controller. The alarms are entered in the diagnostic buffer and can be evaluated using the PeripheralFaultTask. The following table shows how alarms are represented as PeripheralFaultTask.

Alarm (TSI#InterruptId)	TSI#event Class	TSI#faultId	Meaning
Diagnosis (incoming) Diagnosis disappears (outgoing) Multicast Communication Mismatch Port Data Change Notification Sync Data Changed Notification Isochronous Mode Problem Notification Network component problem notification (_SC_DIAGNOSTIC_INTERRUPT (=201))	16#39	16#42	Incoming diagnostic interrupt
	16#38	16#42	Outgoing diagnostic interrupt
Process interrupt (_SC_PROCESS_INTERRUPT (= 200)	16#11	16#41	Process interrupt
Pull Alarm Plug Alarm	16#39	16#51	PROFINET IO module has been removed or cannot be addressed.

Alarm (TSI#InterruptId)	TSI#event Class	TSI#faultId	Meaning
Plug Wrong Submodule Alarm Return of Submodule Alarm (_SC_PULL_PLUG_INTERRUPT (=216))		16#54	PROFINET IO submodule has been removed or cannot be addressed.
	16#38	16#54	PROFINET IO module or submodule has been inserted, module type OK (actual configuration = set configuration)
		16#55	PROFINET IO module or submodule has been inserted, but wrong module type (actual configuration <> set configuration)
		16#56	PROFINET IO module or submodule has been inserted, but error during module parameterization
		16#58	IO status of a module has changed from BAD to GOOD
State			Not Supported
Update			Not supported
Time data changed notification			Not supported
Upload and storage notification			Not supported
Pull module			Not supported
Manufacturer-specific			Not supported
Profile-specific			Not Supported

Alarm types indicated as "not supported" are acknowledged by the SIMOTION controller with "not supported" and not entered in the diagnostic buffer.

Use of the TaskStartInfo

Information concerning the TaskStartInfo for the PeripheralFaultTask is contained in the Base Functions manual.

Transfer diagnostic data

The exact reason for the alarm is provided as diagnostic data. The _readDiagnosticData function can be used to fetch this data. The length is restricted to 255 bytes.

7.8.5 Alarms for direct data exchange between IO controllers

Description

For PROFINET IO with IRT, communication monitoring takes place between IO controllers. If this establishes that IRT data is no longer being received (either there is no data arriving, or it is arriving too late) a station failure alarm is generated. If communication is re-established, a station reconnection alarm is generated. If IRT data arrives late on three occasions, a station failure alarm is reported.

The following table shows PROFINET IO alarms between IO controllers involved in direct data exchange as they are represented in SIMOTION:

Alarm (TSI#InterruptId)	TSI#eventClass	TSI#faultId	Meaning
Station failure (_SC_STATION_DISCONNECTED (= 202))	16#39	16#F3	The receiver in the direct data exchange is no longer receiving data.
Station reconnection (_SC_STATION_RECONNECTED (= 203)	16#38	16#F0	The transmitter in the direct data exchange has started up and is able to transmit.
		16#F1	The receiver in the direct data exchange has started up and is receiving without errors, or the receiver is receiving data again (all receiving areas are available).
		16#F2	The receiver in the direct data exchange has started up and is receiving with errors, or the receiver is receiving data again (at least one receiving area not available).

7.8.6 Alarms for SINAMICS S120 drives

Description

Alarms initiated by the SINAMICS S120 CU320/CBE20 or SINAMICS S120 CU310 PN are issued using the PROFINET alarm channel. Two types are possible for alarms:

- Alarms issued by the PROFINET interface that directly concern PROFINET.
- Alarms issued by the application/technology in the drive.

PROFINET alarms

The following alarms are supported via the SINAMICS module that can be used with PROFINET:

Alarm	Description
Port data change notification	A detailed description can be found under Alarms on the IO controller (Page 214)
Sync Data Changed Notification	
Isochronous mode problem notification	
Multicast Communication Mismatch	

Technology/application alarms

Alarms are not sent to the controller as standard PROFINET alarms, but instead are mapped as PROFIdrive alarms. They have to be fetched by means of a parameter job.

7.8.7 System functions for the diagnostics for PROFINET or PROFIBUS

Overview of system and diagnostics functions

The following table provides an overview of the various system and diagnostics functions for PROFINET IO. Differences with PROFIBUS DP are also indicated. You will find detailed information on the respective functions in the reference lists of the SIMOTION controller.

Function	Note	PROFIBUS	PROFINET
_getStateOfSingleDpSlave	This function returns the status data of a single DP slave / IO device:	Logical diagnostic address of the DP slave	Logical diagnostic address of the station substitute of the IO device
_getStateOfAllDPStations	This function returns the status data of all DP slaves / IO devices:	Logical diagnostic address of the DP slave	Logical diagnostic address of the station substitute of the IO device
_activateDpSlave _deactivateDpSlave _getStateOfDpSlave	_getStateOfDpSlave supplies information on whether the slave is activated or deactivated.	Logical diagnostic address DP slave	Logical diagnostic address of the station substitute of the IO device
_readDiagnosticData _getStateOfDiagnosticDataCommand	This function is used to output diagnostic data for a DP slave via the user program. The diagnostic data is read in the form specified by EN 50170, Volume 2, PROFIBUS. Structure of data for PROFINET is not identical to PROFIBUS. The diagnostics are specific to a subslot	Logical diagnostic address DP slave	Logical diagnostic address of IO address of subslot
_readDriveFaults	This function is used to read the current fault buffer entry in the drive.	Logical start address of drive (slot).	Each valid logical I/O address of the subslot concerned or diagnostic address of the PAP (for subslots without user data)

7.8.8 PROFINET device diagnosis in STEP 7

Device diagnosis in STEP 7

In SCOUT, HW Config can be used to perform an online device diagnosis via PROFINET. The diagnosis supplies not only the slot and the channel number, but also the error type. The diagnosis operates similar to that for PROFIBUS.

Proceed as follows

1. Go online and open the HW Config for the appropriate SIMOTION device.
2. Select **Target system > Diagnose, monitor/control Ethernet node**.
   HW Config searches for all network nodes. The **(Diagnosis) ONLINE** window opens and displays the network nodes.
3. Right-click the required node and select **Properties**. The detailed diagnosis is displayed. The associated fault is displayed here.

Routing - communication across network boundaries 8

8.1 What does routing mean?

Routing is the transfer of information from Network x to Network y.

There is a fundamental difference between intelligent, self-learning routing (e.g. IP routing in the Internet) and routing according to previously specified routing tables (e.g. S7 routing).

IP routing

IP routing is a self-learning routing procedure (which can also be performed manually), used exclusively in Ethernet communication networks which operate with the IP protocol, such as the Internet.

The function is performed by special routers that pass on the information to adjacent networks based on the IP address, when the IP address is not detected in the own network.

S7 routing

S7 routing is a routing procedure based on previously configured routing tables, but which can also exchange information between different communication networks, e.g. between Ethernet, PROFIBUS and MPI. These routing tables can be created as interconnection tables in NetPro.

S7 routing does not work with the IP address, but with what are known as subnet IDs within the S7 protocol.

- Information transfer from Ethernet to MPI and vice versa
- Information transfer from Ethernet to PROFIBUS and vice versa
- Information transfer from MPI to PROFIBUS and vice versa
- Information transfer from Ethernet to Ethernet (only SIMOTION, including PROFINET)

PG / PC assignment

Modification of the PG assignment may be required for S7 routing. You can do this now in the toolbar in SIMOTION SCOUT above the **Assign PG** button. This calls the properties window for PG assignment, where you modify the assignment and "activate" it (S7ONLINE access).

8.2 Configuration of S7 routing

S7 routing is configured in STEP 7 / SIMOTION SCOUT with the aid of the "NetPro" network configuration.

All stations contained in the network configuration can exchange information between one another. Connection tables must be created in NetPro for this purpose. The required routing tables are automatically generated during the compilation of the project, but must then be loaded to all the participating stations.

8.3 Routing for SIMOTION

Routing makes it possible, for example, to access devices connected to subnets ONLINE via a PG/PC.

IP routing

IP routing is fundamentally NOT supported by SIMOTION even when there are several Ethernet interfaces on some SIMOTION devices, such as SIMOTION D4xx.

If you want to connect different Ethernet networks with one another, please set a separate router for the IP routing.

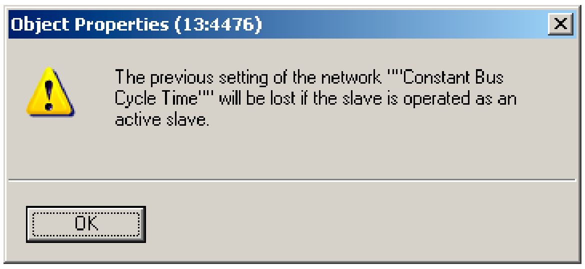
S7 routing

S7 routing is supported by SIMOTION, i.e. information (engineering accesses) can be routed by a SIMOTION device from higher-level networks such as Ethernet and MPI to lower-level networks such as PROFIBUS or PROFINET/Ethernet (4.1.2 and higher).

Supplementary conditions

The following supplementary conditions must be taken into account in the "DP slave" mode when routing information on an isochronously operated PROFIBUS.

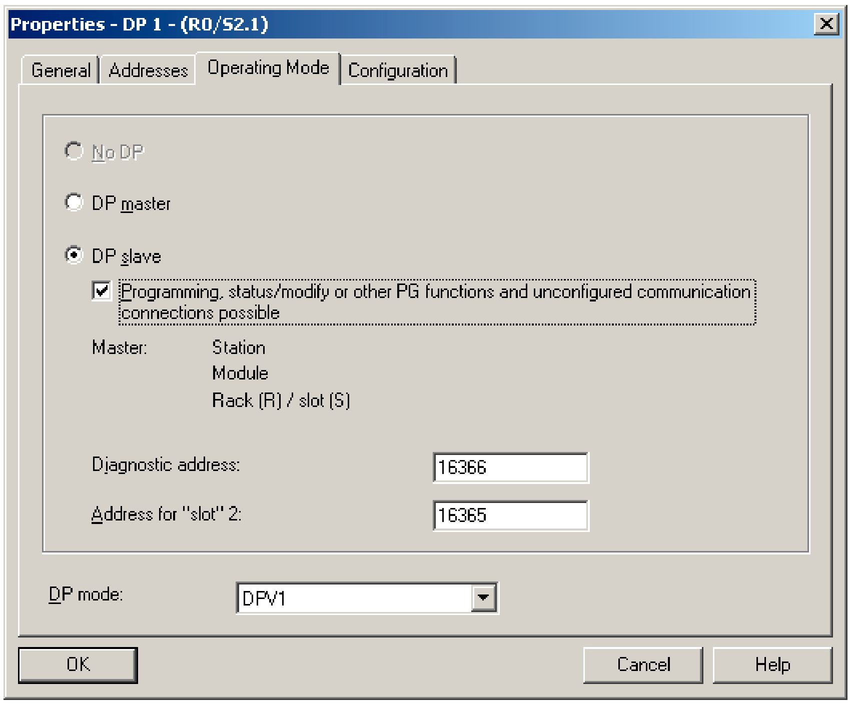
- The functions "Equidistant bus cycle" (requirement for isochronous applications) and "Active station" (requirement for routing to a lower-level network segment) mutually exclude each other.
- It is not possible to operate an active I slave on the isochronous bus.



Object Properties (13:4476)

The previous setting of the network ""Constant Bus Cycle Time"" will be lost if the slave is operated as an active slave.

OK



Properties - DP 1 - (R0/S2.1)

General | Addresses | Operating Mode | Configuration

( ) No DP

( ) DP master

[● DP slave]

[✓] Programming, status/modify or other PG functions and unconfigured communication connections possible

Master:	Station	
	Module	
	Rack (R) / slot (S)	

Diagnostic address: 16366

Address for "slot" 2: 16365

DP mode: DPV1

OK | Cancel | Help

Figure 8-1 DP slave mode: Active station: Testing, commissioning, routing

The "Programming, status/modify or other PG functions …" checkbox must be activated if, for example, you frequently want to perform PG functions required for commissioning and testing via this interface, or if you want to access (S7 route) SINAMICS drives on the cascaded, lower-level DP master interface of the SIMOTION with PG functions (e.g. Starter).

If the "Programming, status/modify or other PG functions..." option is activated, the interface becomes the active node on the PROFIBUS (i.e. the interface participates in the token rotation of the routing PROFIBUS). The following functions are then possible:

- Programming (e.g. loading)
- Test (status/control)
- S7 routing (I-slave as gateway)

The bus cycle time can be prolonged. Therefore, this option should not be activated for time-critical applications and when S7 routing and the client functionality are not required for the communication.

Note

When the "Programming, status/control or other PG functions …" checkbox is not activated, the server only operates as server for communication services, i.e. S7 routing is not possible.

8.4 Routing for SIMOTION D with inserted PROFINET CBE30 board

Routing between the different interfaces

The two standard Ethernet interfaces X120 and X130 of the SIMOTION D each form a separate subnet, all ports on the CBE30 also form a common subnet.

- Routing from subnet to subnet (IP routing) is not supported. You can use an external IP router for this
- The S7 routing from a PROFINET/Ethernet subnet to a PROFIBUS is possible.

There are three options for connecting a PG/PC or HMI via S7 routing to a SIMOTION D with CBE30.

Engineering system to PROFINET (CBE30)

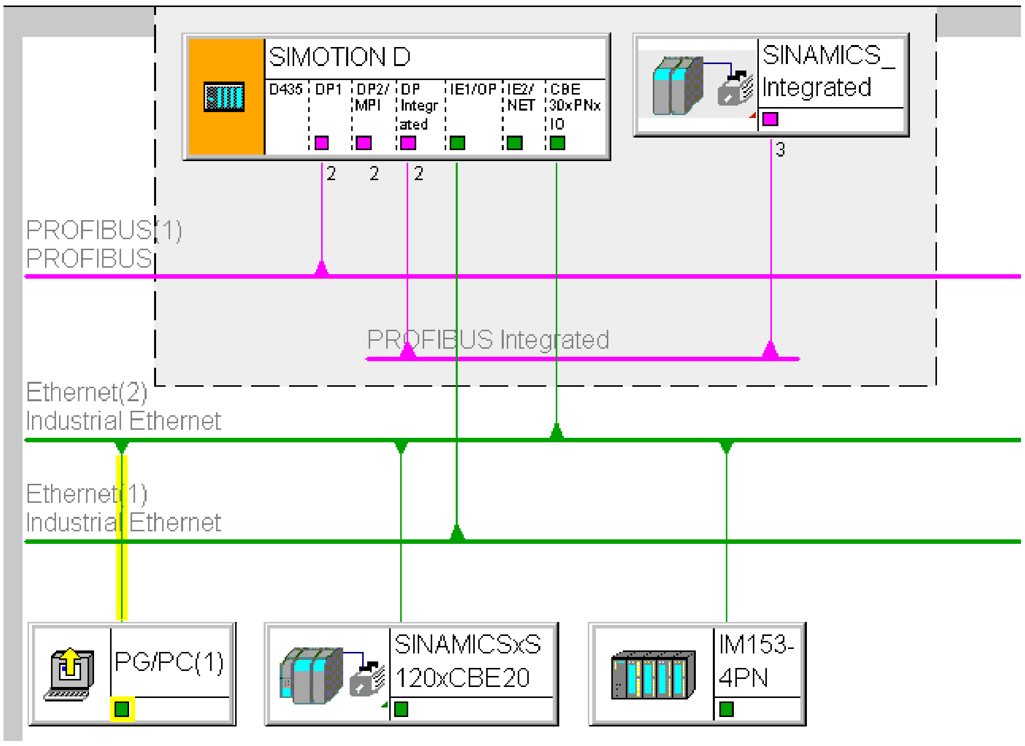


Figure 8-2 Example for PG/PC to CBE30

- S7 routing to the (master) PROFIBUS interfaces (only if configured)
- S7 routing to PROFIBUS Integrated
- S7 routing to the standard Ethernet interfaces ET1/ET2 (X120, X130) (V4.1.2 and higher)
- Access to the components on the same subnet (CBE30) via the switch functionality

Engineering system / HMI to PROFIBUS

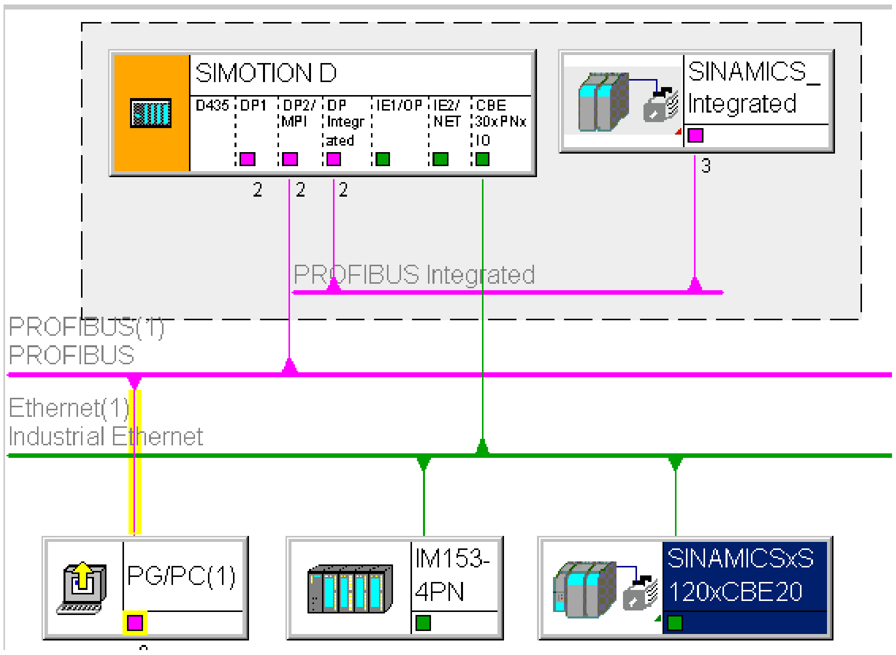


Figure 8-3 Example for PG/PC to PROFIBUS

- S7 routing to the other (master) PROFIBUS interfaces (only if configured)
- S7 routing to PROFIBUS Integrated
- S7 routing to X1400 on the CBE30
- S7 routing to the standard Ethernet interfaces (X120, X130) (V4.1.2 and higher)

Engineering system / HMI to Ethernet

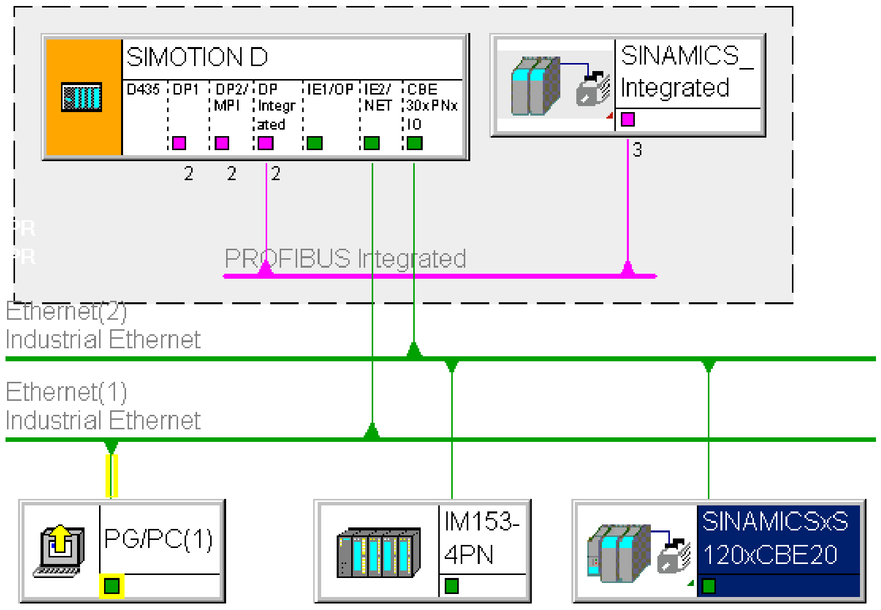


Figure 8-4 Example for PG/PC to Ethernet X120, X130

- S7 routing to the other (master) PROFIBUS interfaces (only if configured)
- S7 routing to PROFIBUS Integrated
- S7 routing to X1400 on the CBE30

8.5 Routing for SIMOTION D to the SINAMICS integrated

S7 routing to the internal PROFIBUS on SINAMICS Integrated

All SIMOTION D have an integrated SINAMICS drive control. In order to be able to access drive parameters, the message frames must be routed from the external SIMOTION D interfaces to the internal PROFIBUS DP. S7 routing can be used to access the integrated PROFIBUS. Here, the internal PROFIBUS DP forms a separate subnet. This must be especially taken into account for the communication to several routing nodes.

8.6 Routing for SIMOTION P350

Description

S7 routing is possible:

- From PROFIBUS (ISO board) on PROFINET subnet to MCI-PN board
- From PROFINET subnet to MCI-PN board on PROFIBUS (ISO PROFIBUS board)
- From SCOUT on SIMOTION P via softbus through the runtime on PN devices on the MCI-PN board

Routing is not possible from onboard Ethernet interfaces on PROFIBUS (ISO PROFIBUS board). IP routing is not possible via the Ethernet interfaces of the P350.

Routing from PROFIBUS to PROFINET

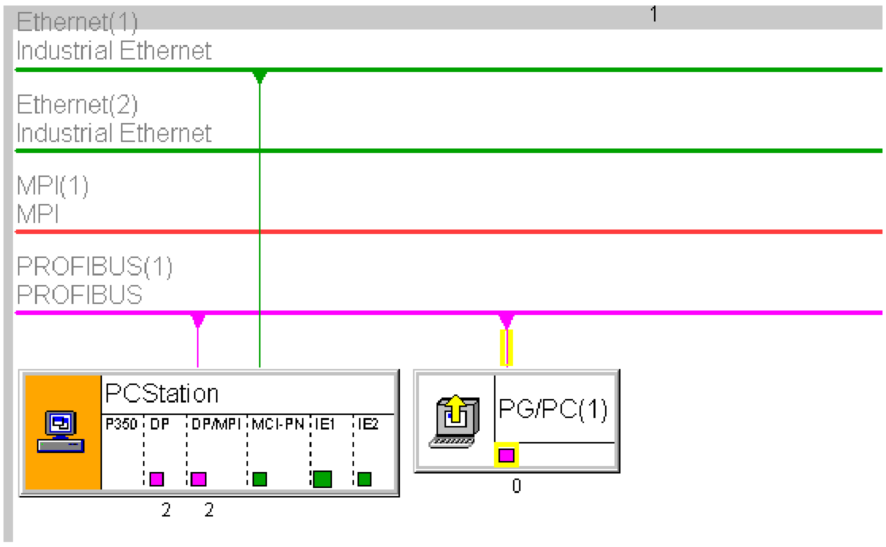


Figure 8-5 Example for P350 routing from PROFIBUS to PROFINET

Routing from PROFINET on PROFIBUS

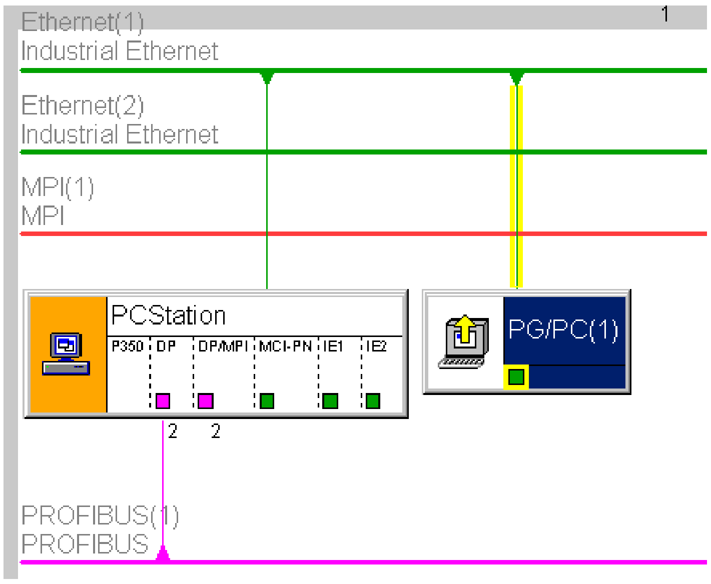


Figure 8-6 Example for P350 routing from PROFINET to PROFIBUS

SIMOTION IT 9

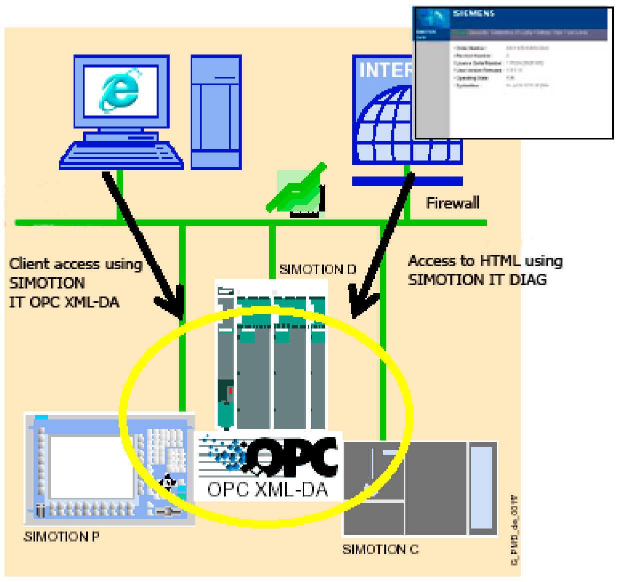
9.1 SIMOTION IT - overview

Description

SIMOTION IT allows you to use standard Internet mechanisms (HTTP) to access SIMOTION via Ethernet and so perform diagnosis and process monitoring.

This provides the following advantages.

- Location-independent open diagnosis / process monitoring
- Use of standard mechanisms (IP, TCP/IP, HTTP)
- Client device independent of the operating system (Windows, Linux, ...)
- Independent of manufacturer-specific tool
- Decoupled from the engineering
- No version conflict between client tool and runtime
- Series commissioning without engineering tool



Firewall
Client access using SIMOTION IT OPC XML-DA
SIMOTION D
Access to HTML using SIMOTION IT DIAG
OPC
OPC XML-DA
SIMOTION P
SIMOTION C

Figure 9-1 SIMOTION IT overview

SIMOTION provides various services:

- SIMOTION IT DIAG
- SIMOTION IT OPC XML DA

- Access to TRACE (extension of SIMOTION OPC XML - DA)
- File download using FTP (File Transfer Protocol)

Further references

A detailed description of the SIMOTION IT products is contained in the **SIMOTION IT Ethernet-based HMI and Diagnostic Functions** Product Information on the SIMOTION SCOUT Documentation CD.

See also

Web access to SIMOTION (Page 232)

SIMOTION IT DIAG (Page 233)

SIMOTION IT OPC XML DA (Page 235)

9.2 Web access to SIMOTION

Description

The following figure shows the various possibilities to access the data in a SIMOTION module.

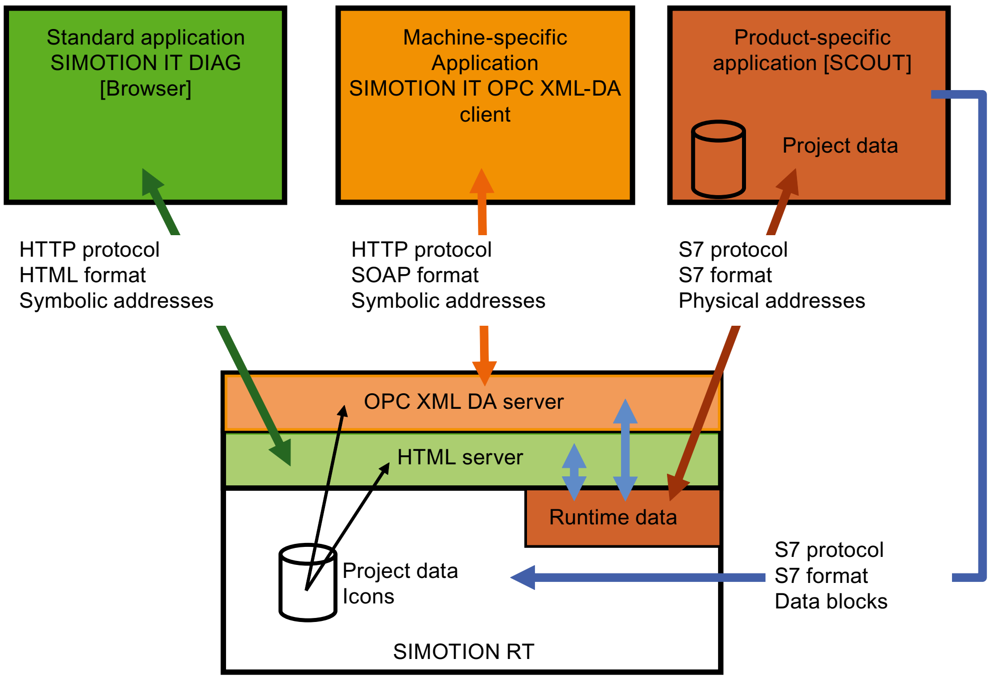


Figure 9-2 Access to SIMOTION

See also

SIMOTION IT DIAG (Page 233)

SIMOTION IT OPC XML DA (Page 235)

9.3 SIMOTION IT DIAG

Description

SIMOTION IT DIAG allows a PC to use any Internet browser to access the HTML pages in SIMOTION.

A separate license is required for SIMOTION IT DIAG.

Standard diagnostic pages

SIMOTION provides the following standard diagnostic pages:

- Start page
- Device Info (information about the firmware, devices, device components and technology objects)
- Diagnostics (CPU utilization, memory use, operating mode, display of task runtimes, SIMOTION alarms and drives, trace, diagnostics buffer, service overview)
- IP-Config (data of the SIMOTION device interface)
- Settings (setting the time zone, switching operating modes, changing the display of user-defined pages)
- Manage Config (loading IT DIAG configurations, device updates, saving device data)
- Files (accessing the SIMOTION file system, uploading and downloading files, creating folders, and storing additional data, e.g. documentation)

Simplified standard pages

To enable the best possible display of IT DIAG pages on devices such as cell phones or PDAs, a set of special pages is provided for version 4.1.3 and higher. These contain a simplified representation of information from the standard pages. You can access these pages via the address http://<IPAddr>/BASIC.

Configuration via WebCfg.xml

The WEBCFG.XML file is used to configure the web server. This involves, for example, defining changes for standard pages, converting default pages to user-defined pages, and defining configuration areas and the user database.

User-defined pages

SIMOTION IT DIAG offers the option of creating individually designed web pages. Two different variants are available:

- JavaScript libraries opcxml.js and appl.js
- MiniWeb Server Language (MWSL)

The "User's Area" of the standard diagnostics pages is reserved for user-defined HTML pages. In this area you can store user-defined HTML pages in the SIMOTION CPU using the flash file system.

Trace Interface

The "Trace Interface via SOAP" function package enables variable values to be written to a buffer. The values are packed in files and can be retrieved asynchronously via a HTTP request. This interface can only be used by client applications. The client enables the time characteristic of variables to be traced.

Variable access

The variable access for the SIMOTION IT applications is implemented using a variable provider. This makes it possible to access the following variables:

- Device system variables
- TO system variables
- Program interface variables
- Configuration data
- Drive parameters
- Setting of the operating state, execute RamToRom, execute ActiveToRom
- Technological alarms
- Diagnostics buffer

Secure HTTPS connection

The Secure Socket Layer protocol (SSL) enables encrypted data transmission between a client and SIMOTION. HTTPS access between the browser and the SIMOTION CPU is based on the Secure Socket Layer protocol. Encrypted access to SIMOTION can take place via both SIMOTION IT OPC XML DA and SIMOTION IT DIAG.

9.4 SIMOTION IT OPC XML DA

Description

A customer-specific application created on a client PC, which, for example, is programmed with the C#, Visual Basic, or Java programming language, uses the SIMOTION IT OPC XML-DA services and properties:

A separate license is required for SIMOTION IT OPC XML-DA.

- Open communication using HTTP, SOAP, OPC-XML (Ethernet) between client device and SIMOTION (Web services, Remote Procedure Call)
- Uses the OPC XML DA 1.0 specification of the OPC Foundation
- Access to SIMOTION process variable
  - Read and write variables
  - Cyclical reading of variables using subscriptions
  - Browse variables
- Trace interface using SOAP; this function is an extension of the OPC specification
- Clients on any hardware with various operating systems (Windows, Linux, etc.)
- Creating client applications using C#, Java, C++. You must implement yourself the application that you want to access on the SIMOTION OPC server.
- Access protection with user ID and password

The following figure shows schematically the access to the OPC server

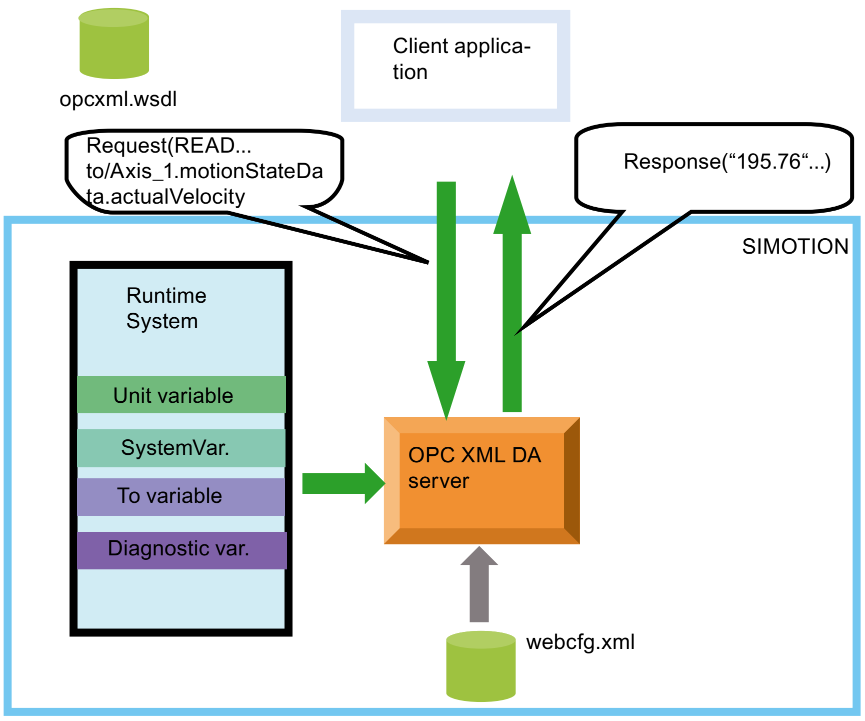


Figure 9-3 Access to the OPC server

C# code example

Table 9- 1 Establishing a connection

```
OpcXmlDa_R1_0.Service MyServer = new OpcXmlDa_R1_0.Service();
MyServer.Url = "http://" + this.ServerAddress.Text + "/soap/opcxml";
```

Table 9- 2 Write variable

```
WriteServer.Write(RequestOptions,WriteItemList,true, out RItemList,out
WriteErrorList);
```

Table 9- 3 Subscription

```
OpcXmlDa_R1_0.ReplyBase Result = SubscribeServer.Subscribe(RequestOptions,
SubscribeItemList, false, 0, out SubscribeReplyItemList, out
SubscribeErrorList, out ServerSubHandle);
SubscribeServer.SubscriptionPolledRefresh(RequestOptions,
ServerSubHandleList,
Result.ReplyTime.AddMilliseconds(System.Double.Parse("100")),true,
System.Int32.Parse("10000"), false, out InvalidServerSubHandles, out
SubscribePolledRefreshReplyItemListArray, out SubscribeErrorList, out
DataBufferOverflow);
```

Variable access

The variable access for the SIMOTION IT applications is implemented using a variable provider. This makes it possible to access the following variables:

- Device system variables
- TO system variables
- Program interface variables
- Configuration data
- Drive parameters
- Setting of the operating mode, executing RamToRom, executing ActiveToRom
- Technological alarms
- Diagnostics buffer

9.5 FTP data transfer

File access using FTP

You can access specific data of the SIMOTION memory cards. For this purpose, you can access the FTP server integrated in SIMOTION. You must create the users and the associated passwords on the FTP server. FTP is protected through access protection.

You can use FTP, for example, to perform firmware updates or load user defined HTML pages.

The FTP service does not require its own license.

Index