

# SIEMENS

## SIMOTION

### SIMOTION SCOUT Communication

#### System Manual

---

#### Foreword

---

#### Introduction

1

---

#### Overview of the communication functions and services

2

---

#### PROFIdrive

3

---

#### PROFIBUS

4

---

#### Ethernet introduction (TCP/IP and UDP connections)

5

---

#### PROFINET IO

6

---

#### Routing - communication across network boundaries

7

---

#### SIMOTION IT

8

## Safety Guidelines

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.

 <b>DANGER</b>
indicates that death or severe personal injury <b>will</b> result if proper precautions are not taken.
 <b>WARNING</b>
indicates that death or severe personal injury <b>may</b> result if proper precautions are not taken.
 <b>CAUTION</b>
with a safety alert symbol, indicates that minor personal injury can result if proper precautions are not taken.
<b>CAUTION</b>
without a safety alert symbol, indicates that property damage can result if proper precautions are not taken.
<b>NOTICE</b>
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.

## Prescribed Usage

Note the following:

 <b>WARNING</b>
This device may only be used for the applications described in the catalog or the technical description and only in connection with devices or components from other manufacturers which have been approved or recommended by Siemens. Correct, reliable operation of the product requires proper transport, storage, positioning and assembly as well as careful operation and maintenance.

## 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.

# 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 V4.1:

- SIMOTION SCOUT V4.1 (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 SP2:

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

## Hotline and Internet addresses

### Technical support

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

	<b>Europe / Africa</b>
<b>Phone</b>	+49 180 5050 222 (subject to charge)
<b>Fax</b>	+49 180 5050 223
<b>Internet</b>	<a href="http://www.siemens.com/automation/support-request">http://www.siemens.com/automation/support-request</a>

	<b>Americas</b>
<b>Phone</b>	+1 423 262 2522
<b>Fax</b>	+1 423 262 2200
<b>E-mail</b>	<a href="mailto:techsupport.sea@siemens.com">mailto:techsupport.sea@siemens.com</a>

	<b>Asia / Pacific</b>
<b>Phone</b>	+86 1064 719 990
<b>Fax</b>	+86 1064 747 474
<b>E-mail</b>	<a href="mailto:adsupport.asia@siemens.com">mailto:adsupport.asia@siemens.com</a>

---

**Note**

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

<http://www.siemens.com/automation/service&support>

Calls are subject to charge, e.g. 0.14 €/min. on the German landline network. Tariffs of other phone companies may differ.

---

**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 63315
E-mail	<a href="mailto:docu.motioncontrol@siemens.com">mailto:docu.motioncontrol@siemens.com</a>

**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)
- Product support:
  - <http://support.automation.siemens.com/WW/view/en/10805436>

**Additional support**

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

Please contact your regional training center or our main training center at D-90027 Nuremberg, phone +49 (911) 895 3202.

Information about training courses on offer can be found at:

[www.sitrain.com](http://www.sitrain.com)



# Table of contents

	Foreword .....	3
<b>1</b>	<b>Introduction.....</b>	<b>13</b>
1.1	The communications subject in the SIMOTION documentation.....	13
<b>2</b>	<b>Overview of the communication functions and services.....</b>	<b>15</b>
2.1	Network options .....	15
2.1.1	Introduction .....	15
2.1.2	PROFINET .....	16
2.1.3	Industrial Ethernet.....	16
2.1.4	PROFIBUS.....	17
2.1.5	MPI (Multi-Point Interface) .....	17
2.1.6	Point-to-point communication (PtP).....	18
2.2	Communications services (or network functions) .....	18
2.2.1	Introduction .....	18
2.2.2	PG/OP communication services .....	19
2.2.3	S7 communication services .....	19
2.2.4	S7 basic communication services.....	20
2.2.5	"Global data" communication service .....	20
2.2.6	PROFINET communication services .....	21
2.2.7	Industrial Ethernet communication services .....	21
2.2.8	PROFIBUS communication services .....	22
2.3	Additional services for the exchange of information .....	23
<b>3</b>	<b>PROFIdrive.....</b>	<b>25</b>
3.1	Introduction .....	25
3.2	Why profiles? .....	25
3.3	Segmentation in application classes.....	26
3.4	PROFIdrive-specific data types .....	28
<b>4</b>	<b>PROFIBUS .....</b>	<b>33</b>
4.1	Cyclic communication .....	33
4.1.1	Cyclic communication (overview).....	33
4.2	DP V1 acyclic communication.....	34
4.2.1	Acyclic communication to slaves .....	34
4.2.2	Reading and writing data with DP V1 .....	34
4.2.3	Data set 47.....	37
4.2.4	Error assessment.....	39
4.2.5	Additional information for the parameters of a PROFIdrive drive.....	39
4.2.6	Structure of a read/write request.....	40
4.2.7	System commands in SIMOTION.....	41
4.2.7.1	_writeRecord/_readRecord SIMOTION system commands .....	41
4.2.7.2	_writeDrive.../_readDrive... SIMOTION system commands.....	42
4.2.7.3	Comparison of the system commands .....	43

4.2.7.4	Deleting _readDrive and _writeDrive jobs .....	44
4.2.8	Rules for using _readRecord and _writeRecord .....	45
4.2.8.1	Rule 1 - the job has its own job reference.....	45
4.2.8.2	Rule 2 - system functions for asynchronous programming.....	45
4.2.8.3	Rule 3 - read/write data record per PROFIDrive drive device .....	47
4.2.8.4	Rule 4 - the last call wins for SIMOTION .....	47
4.2.8.5	Rule 5 - a maximum of eight concurrent calls is possible in SIMOTION .....	49
4.2.9	Rules for SIMOTION _writeDrive.../_readDrive... commands.....	51
4.2.9.1	Scope for the rules .....	51
4.2.9.2	Rule 6 - repeated call of system function for asynchronous programming.....	52
4.2.9.3	Rule 7 - multiple concurrent calls per target device.....	53
4.2.9.4	Rule 8 - release the interlocking after the complete processing of a job .....	54
4.2.9.5	Rule 9 - canceling jobs for an asynchronous call .....	56
4.2.9.6	Rule 10 - management of sixteen jobs .....	59
4.2.9.7	Rule 11 - parallel jobs for different drive devices .....	59
4.2.10	Special features .....	61
4.2.10.1	Rule 12 - data buffering of up to 64 drive objects .....	61
4.2.10.2	Rule 13 - a mix of system functions can be used .....	61
4.2.10.3	Rule 14 - interlocking for the mixed use of commands .....	63
4.2.11	Program examples .....	64
4.2.11.1	Programming example .....	64
4.3	Communication with SIMATIC S7.....	67
4.3.1	Possible communication connections between SIMOTION and SIMATIC.....	67
4.3.2	SIMOTION as DP slave on a SIMATIC S7 .....	68
4.3.2.1	Introduction .....	68
4.3.2.2	Connecting SIMOTION as DP slave with the aid of a GSD file to a SIMATIC S7 .....	68
4.3.2.3	Connecting SIMOTION as i-slave to a SIMATIC S7.....	69
4.3.3	SIMATIC S7 as DP slave on a SIMOTION .....	71
4.3.3.1	Introduction .....	71
4.3.3.2	Connecting SIMATIC as DP slave with the aid of a GSD file to a SIMOTION device.....	71
4.3.3.3	Connecting SIMATIC S7 CPU as i-slave to a SIMOTION device.....	72
4.3.4	PROFIBUS master-master connection between SIMATIC and SIMOTION.....	74
4.3.4.1	Introduction .....	74
4.3.4.2	SIMATIC S7 system functions for a PROFIBUS connection .....	75
<b>5</b>	<b>Ethernet introduction (TCP/IP and UDP connections) .....</b>	<b>79</b>
5.1	Introduction .....	79
5.2	Configuring Ethernet subnets with SIMOTION .....	79
5.2.1	Features of the Ethernet subnets.....	79
5.3	Function overview and functional sequence of Ethernet communication via TCP/IP or UDP.....	80
5.3.1	Introduction .....	80
5.3.2	SIMOTION TCP/IP functions - modeling .....	80
5.3.3	SIMOTION TCP/IP functions - description.....	82
5.3.4	SIMOTION UDP functions - modeling .....	82
5.3.5	SIMATIC functions .....	84
5.3.6	General information.....	86
5.4	Preparations for the configuration of the connection between SIMOTION and SIMATIC S7 .....	87
5.5	Configuring a communication connection between a SIMATIC with Ethernet CP and a SIMOTION device .....	88
5.5.1	Configuring a communication connection between a SIMATIC with Ethernet CP and a SIMOTION device .....	88

5.5.2	TCP/IP connection .....	89
5.5.3	UDP connection .....	92
5.6	Creating a communication connection between a SIMATIC CPU with integrated Ethernet interface and a SIMOTION device .....	94
5.7	Using the functions and function blocks in the user program .....	95
5.7.1	Configuration flowchart and general information .....	95
5.7.2	S7 and SIMOTION functions for a TCP/IP connection when using an S7 station with Ethernet CP .....	99
5.7.2.1	Introduction .....	99
5.7.2.2	S7 functions .....	99
5.7.2.3	SIMOTION functions .....	100
5.7.3	S7 and SIMOTION functions for a UDP connection when using an S7 station with Ethernet CP .....	104
5.7.3.1	Introduction .....	104
5.7.3.2	S7 functions .....	104
5.7.3.3	SIMOTION functions .....	104
5.7.4	S7 function blocks and SIMOTION functions for a TCP/IP connection when using an S7 station with integrated Ethernet interface .....	105
5.7.4.1	Introduction .....	105
5.7.4.2	S7 function blocks .....	105
5.7.4.3	SIMOTION functions .....	110
5.7.5	Processing of TCP/IP data packets in the SIMOTION user program .....	110
5.8	Details of the SIMOTION TCP/IP system functions .....	113
5.8.1	_tcpOpenServer function .....	113
5.8.2	_tcpOpenClient function .....	113
5.8.3	_tcpReceive function .....	114
5.8.4	_tcpSend function .....	114
5.8.5	_tcpCloseConnection function .....	115
5.8.6	_tcpCloseServer function .....	115
5.9	Details of the SIMOTION UDP system functions .....	115
5.9.1	Function _udpSend .....	115
5.9.2	Function _udpReceive .....	116
<b>6</b>	<b>PROFINET IO .....</b>	<b>119</b>
6.1	PROFINET IO overview .....	119
6.1.1	PROFINET IO .....	119
6.1.2	Application model .....	120
6.1.3	IO controller .....	120
6.1.4	IO device .....	121
6.1.5	Sync domain .....	121
6.1.6	iDevice .....	121
6.1.7	Addressing of PROFINET IO devices .....	122
6.1.8	RT classes .....	122
6.1.8.1	RT classes for PROFINET IO .....	122
6.1.8.2	PROFINET IO with RT .....	126
6.1.8.3	PROFINET IO with IRT - Overview .....	127
6.1.8.4	PROFINET IO with IRT (High Flexibility) .....	128
6.1.8.5	PROFINET IO with IRT (High Performance) .....	129
6.1.9	Topology .....	130
6.1.10	Isochronous applications with PROFINET .....	133
6.1.11	Cycle clock scaling .....	135
6.1.11.1	Cycle clock scaling with PROFINET IO on SIMOTION devices .....	135
6.1.11.2	Cycle clock scaling for IO accesses .....	136

6.1.11.3	Bus cycle clocks that can be adjusted for cycle clock scaling to SIMOTION devices .....	137
6.1.12	Connection between sync domain and IO systems .....	137
6.1.13	Redundant sync master .....	138
6.1.14	Quantity structures .....	139
6.1.15	Acyclic communication via PROFINET .....	140
6.2	Specific properties of PROFINET IO with SIMOTION .....	141
6.2.1	Introduction .....	141
6.3	Configuring PROFINET IO with SIMOTION .....	142
6.3.1	New to SIMOTION V4.1.2 .....	142
6.3.2	Proceed as follows for configuring PROFINET IO .....	142
6.3.3	Adding and configuring a CBE30-PROFINET board .....	143
6.3.4	Inserting and configuring P350 .....	145
6.3.5	Inserting and configuring the C240 .....	147
6.3.6	Creating a sync domain .....	149
6.3.7	Configuring a topology .....	151
6.3.7.1	Topology .....	151
6.3.7.2	Interconnecting ports via the topology editor .....	152
6.3.7.3	Interconnecting ports via object properties .....	154
6.3.7.4	Topology editor (graphical view) .....	155
6.3.8	Defining send clock and refresh times .....	156
6.3.9	Creating an IO device .....	157
6.3.10	Inserting and configuring the SINAMICS S120 .....	159
6.3.11	Assigning device names and IP addresses to IO devices .....	161
6.4	Configuring direct data exchange between IO controllers .....	166
6.4.1	Introduction .....	166
6.4.2	Configuring the sender .....	167
6.4.3	Configuring the receiver .....	168
6.5	Configuring the iDevice .....	169
6.5.1	PROFINET iDevice .....	169
6.5.2	Configuring a PROFINET iDevice .....	170
6.5.3	Creating a substitute iDevice .....	173
6.5.4	Inserting an iDevice in the higher-level IO controller .....	175
6.6	Loading the communication configuration .....	179
6.6.1	Loading the PROFINET IO configuration .....	179
6.7	Data exchange between SIMATIC and SIMOTION via PROFINET .....	179
6.7.1	Data exchange through the use of iDevices .....	179
6.7.2	PN-PN coupler .....	180
6.7.3	Communication using standard protocols .....	181
6.8	Diagnostic and alarm behavior .....	182
6.8.1	PROFINET IO alarm and diagnostic messages to SIMOTION .....	182
6.8.2	Diagnostics model .....	183
6.8.3	Alarms on the IO controller .....	185
6.8.4	Alarms from the IO device to the IO controller .....	186
6.8.5	Alarms for direct data exchange between IO controllers .....	187
6.8.6	Alarms for SINAMICS S120 drives .....	188
6.8.7	System functions for the diagnostics for PROFINET or PROFIBUS .....	189
6.8.8	PROFINET device diagnosis in STEP 7 .....	190
<b>7</b>	<b>Routing - communication across network boundaries .....</b>	<b>191</b>
7.1	What does routing mean? .....	191
7.2	Configuration of S7 routing .....	192

	7.3 Routing for SIMOTION.....	192
	7.4 Routing for SIMOTION D with inserted PROFINET CBE30 board.....	194
	7.5 Routing for SIMOTION D to the SINAMICS integrated .....	196
	7.6 Routing for SIMOTION P350 .....	197
<b>8</b>	<b>SIMOTION IT.....</b>	<b>199</b>
	8.1 SIMOTION IT - overview.....	199
	8.2 Web access to SIMOTION.....	201
	8.3 SIMOTION IT DIAG .....	202
	8.4 SIMOTION IT OPC XML DA.....	205
	8.5 FTP data transfer .....	207
	<b>Index.....</b>	<b>209</b>



# 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 PROFI-safe 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 through 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

## 3.1 Introduction

### 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.

You will find the current version under [www.profibus.com](http://www.profibus.com); Order No.: 3.172

## 3.2 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.

## PROFdrive

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

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 PROFdrive profile supports both central as well as distributed motion control concepts.

The basic philosophy: – Keep it simple –

The PROFdrive 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.

## 3.3 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, PROFdrive 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.)
Category 2	Standard drives with technology functions
Category 3	Positioning drives
Category 4	Motion control drives with central, higher-level motion control intelligence
Category 5	Motion control drives with central, higher-level motion control intelligence and the patented "Dynamic Servo Control" position control concept
Category 6	Motion control drives with distributed, motion control intelligence integrated in the drives

PROFdrive 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, PROFdrive 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 over the so-called DP-V1 parameter channel (Base Mode Parameter Access).

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

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.

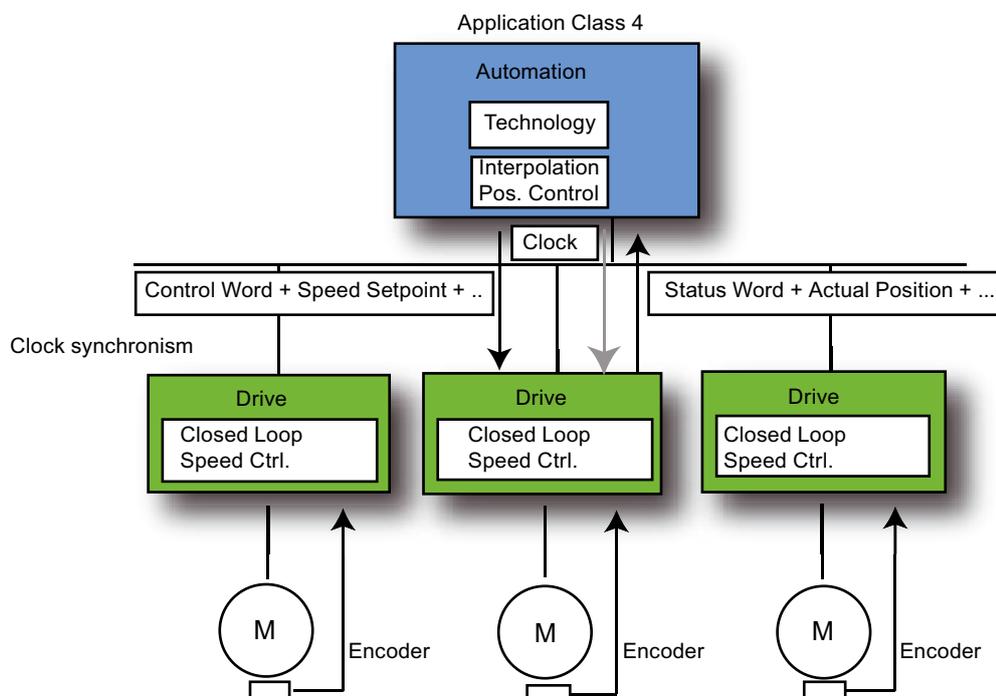


Figure 3-1 Utilization categories

The DSC (Dynamic Servo Control) function significantly improves the dynamic response and stiffness of the position control loop by minimizing the dead times which usually occur for speed setpoint interfaces with an additional, relatively simple feedback network in the drive. The position control loop is closed in the drive which enables very fast position control cycles (e.g. 125  $\mu$ s for SINAMICS S) and thus limits dead times exclusively to the control behavior.

#### Note

In SIMOTION, the drive interface is implemented according to the PROFdrive profile and utilization category 5 with and without DSC.

### 3.4 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 used, for example, by the function `_readDriveParameterDescription`.

#### 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
<b>Specific data types</b>	<b>See below for description</b>	
N2 (normalized value (16-bit))		113
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

Data types used in the PROFdrive profile	Definition	Coding (dec.)
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
$-200\% \leq i \leq (200-2^{-14})\%$	$2^{-12} = 0.0061\%$	113	1	SN	$2^0$	$2^{-1}$	$2^{-2}$	$2^{-3}$	$2^{-4}$	$2^{-5}$	$2^{-6}$
			2	$2^{-7}$	$2^{-8}$	$2^{-9}$	$2^{-10}$	$2^{-11}$	$2^{-12}$	$2^{-13}$	$2^{-14}$
$-200\% \leq i \leq (200-2^{-30})\%$	$2^{-28} = 9.3 \cdot 10^{-8}\%$	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
$-800\% \leq i \leq 800-2^{-12})\%$	$2^{-12}$	123	1	SN	$2^2$	$2^1$	$2^0$	$2^{-1}$	$2^{-2}$	$2^{-3}$	$2^{-4}$
			2	$2^{-5}$	$2^{-6}$	$2^{-7}$	$2^{-8}$	$2^{-9}$	$2^{-10}$	$2^{-11}$	$2^{-12}$
$-800\% \leq i \leq 800-2^{-28})\%$	$2^{-28}$	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
$-256 \cdot 2^{-7} \leq i \leq 256 \cdot 2^{-7}$	$2^{-7} = 0.0078125$	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
$-214,748.3648 \leq i \leq 214,748.3648$	$10^{-4} = 00001$	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 =  $16,384 * T_a / \text{internal value}$

*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
$1 * T_a \leq i \leq 16,384 * T_a$	$T_a$	117	2 octets

# PROFIBUS

## 4.1 Cyclic communication

### 4.1.1 Cyclic 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.

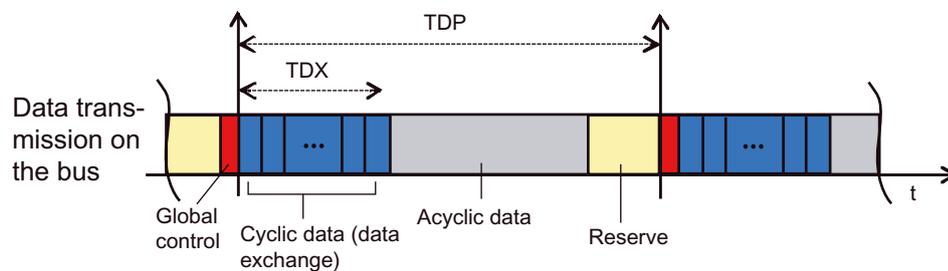


Figure 4-1 Data protocol on Profibus

## 4.2 DP V1 acyclic communication

### 4.2.1 Acyclic communication to slaves

#### 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 using so-called DP V1 services, "read data record" or "write data record".

These services are defined and provided by PROFIBUS; they can be used in parallel to the cyclical communication on the PROFIBUS. The PROFIdrive profile specifies precisely how the basic PROFIBUS mechanisms for the write access are used for parameters of a PROFIdrive-conform drive, see PROFIdrive profile.

#### Literature note

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

### 4.2.2 Reading and writing data with DP V1

#### Description

Data set 47, 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 **Acyclical Communication** section of the SINAMICS S120 Commissioning Manual.

A parameter access always consists of a pair of items consisting of "Write data record" and "Read data record" in this sequence, irrespective of whether a 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).

**Note**

For communication via PROFIBUS, the data set 47 is used to access parameters in PROFIdrive drives. When PROFINET is used, the basic processes do not change, although the data record number is then 0xB02F ("Base Mode Parameter Access Local"). PROFINET will not be discussed further in this document.

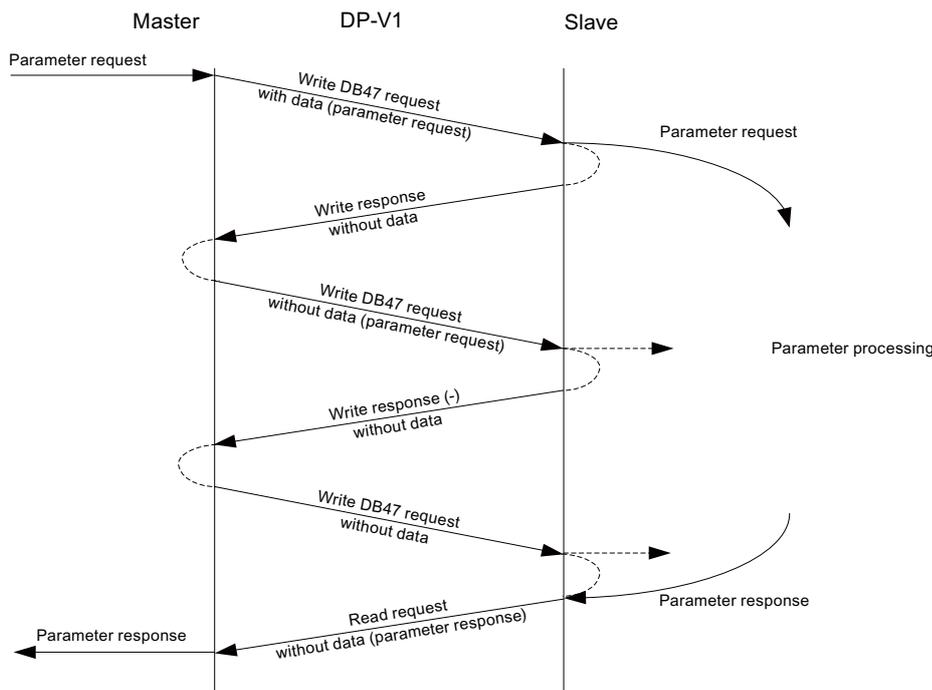


Figure 4-2 Read and write DP V1

The figure **Read and write DP V1** shows that both "Write data record" and "Read data record" consist of "Request" and "Response".

**Request Reference**

The "Write Request" is initiated by the controller and also contains in the data to be transferred information that allows the subsequent "Read Response" of the target device to be assigned (see below: the header of the Parameter Request data record contains a "Request Reference"). This information is mirrored by the target device with the Parameter Response (see "Request Reference mirrored" under Data record 47 (Page 37)) and thus enables an assignment.

This "Request Reference" is used not only for the assignment of request and response, but also for the assignment of "Write data record" to the following "Read data record", because the controller can process several actions in parallel to different target devices using the same PROFIBUS.

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

---

**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, several PROFIdrive drive devices are connected to a controller using PROFIBUS, a job can be processed in parallel for each of these drive devices. 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 49).

---

**Note**

The PROFIdrive standard does not specify how a PROFIdrive drive should behave when an attempt is made to process more than one "Read/write data record" action concurrently with the drive. Therefore, differences between various PROFIdrive drive products according to the implementation must be expected.

---

**Note**

The PROFIdrive standard does not specify how a controller should behave when the user attempts to process more than one "Read/write data record" action concurrently with the drive. Therefore, differences between various controllers according to the implementation must be expected.

---

### 4.2.3 Data set 47

#### Structure of data set 47

The data set 47 always consists of:

- A header (job reference, job identification, target axis / drive object, number of parameters in the job)
- User data (attribute, number of elements/indexes, parameter number and subindex), for write jobs, also the values.

The data to be transferred for a WRITE or READ request has the following structure.

	Job parameters			Offset
	Request Header	Request Reference	RequestID	0
		Axis	Number of parameters	2
	1. Parameter address	Attribute	Number of elements	4
		Parameter number		6
		Subindex		8
	...			
	nth parameter address	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.

	Parameter response			Offset
	Request Header	Request Reference mirrored	RequestID	0
		Axis-No / DO-ID mirrored	Number of parameters	2
Values only for read access	1. parameter value(s)	Format	Number of values	4
		Values or error values		6

	Parameter response		Offset
		...	
		...	
Error values for negative response only	nth parameter value(s)	Format	Number of values
		Values or error values	
		...	

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 Commissioning 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.

### Writing parameter records

To write parameter values (one or more), the data set 47 is first created (parameter number(s) and value(s)), and then transferred using "Write data record". A subsequent "Read data record" returns the information whether the writing of the values was successful. A successful completion of "Write data record" signals only the correct transfer of the data record over the communications path, but not the correct execution of the write action in the target device.

### Reading parameter records

To read parameter values, the data set 47 is created first with the parameter(s) to be read. This data record is transferred to the drive using "Write data record". A subsequent "Read data record" then returns the required values once (the same job reference will also be returned in the response).

The processes are also shown as diagram in the figure **Read and write DP V1**.

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 controller in certain cases has a different internal data representation, an explicit conversion must be performed for the grouping and processing of the data in data set 47.

A conversion may be required for SIMOTION, see Program example (Page 64).

## 4.2.4 Error assessment

### Description

Two different types of errors can occur in conjunction with DP V1 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 errors for PROFIdrive-conform drives are defined in the PROFIdrive standard and, for example, can be found in the SINAMICS S120 Commissioning Manual.

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

The error codes are returned in the response of the drive device in data set 47, see the above table, "Values or error values". The differentiation whether the queried parameter represents an error code or a "true" value can be made in the "Format" field in the Parameter Response, see the table Structure after Base Mode Parameter Access - Parameter Response, Offset 4, "Format".

The SINAMICS S120 Commissioning Manual 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.

## 4.2.5 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 differentiation is made for the transferred "Parameter Request" in data set 47 in the "Attribute" field:

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.

### 4.2.6 Structure of a read/write request

#### Description

The following section describes the structure of write or read requests, as it can be used when transmitting a parameter request or parameter response via PAP (Parameter Access Point).

#### Transmission of a parameter request in a write request

Block	Byte n + 1	Byte n
Write Header	Function_Num = 0x5F (Write)	Slot_Number = ...
	Index = 47	Length = (Data)
Data (Length)	Parameter request ... ...	

#### Confirmation of a parameter request with a write response (without data)

Block	Byte n + 1	Byte n
Write Header	Function_Num = 0x5F (Write)	Slot_Number = (mirrored)
	Index = (mirrored)	Length = (mirrored)

#### Request for a parameter response in a read request (without data)

Block	Byte n + 1	Byte n
Read Header	Function_Num = 0x5E (Read)	Slot_Number = ...
	Index = 47	Length = MAX

#### Transmission of the parameter response in a read response

Block	Byte n + 1	Byte n
Read Header	Function_Num = 0x5E (Read)	Slot_Number = (mirrored)
	Index = (mirrored)	Length = (Data)
Data (Length)	Parameter response ... ...	

## Meaning of the various transfer elements

Element	
Function_Num	ID of the service (Read, Write, Error)
Slot_Number	Addresses the DP slave in the request, is mirrored in the response
Index	Addresses the data block in the DP slave, is mirrored in the response. For PROFIdrive, Index defines the 47 parameter request and response for data set 47
Length	Length of transferred data in bytes

## 4.2.7 System commands in SIMOTION

### 4.2.7.1 \_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 611 universal 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")

#### 4.2.7.2 `_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 51)

**4.2.7.3 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	<ul style="list-style-type: none"> <li>• Generally usable, not just for DP V1 services for drives</li> <li>• Assumes only the knowledge of some I/O address <i>on the drive device</i> and the "DO-ID" or "Axis-No" on the drive device</li> </ul>	<ul style="list-style-type: none"> <li>• The user must create the data record</li> <li>• The user must program two calls for parameter accesses in a PROFIdrive drive</li> <li>• Users may need to perform the required data conversions themselves</li> <li>• "DO-ID" or "Axis-No" must be known</li> </ul>
_readDrive... _writeDrive...	<ul style="list-style-type: none"> <li>• Tailored for the typical communication with PROFIdrive drives</li> <li>• The user does not need to know the structure of data set 47</li> <li>• Reduced programming effort for the user for communication to drives</li> </ul>	<ul style="list-style-type: none"> <li>• Assumes the presence or knowledge of an I/O address <i>of the associated drive object</i></li> <li>• 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</li> <li>• The user must make any required data conversions</li> </ul>

Properties of the system commands

The use of the drive-specific \_write/\_readDrive... system commands on 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 64).

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.

#### 4.2.7.4 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`.

## 4.2.8 Rules for using `_readRecord` and `_writeRecord`

### 4.2.8.1 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.

### 4.2.8.2 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.

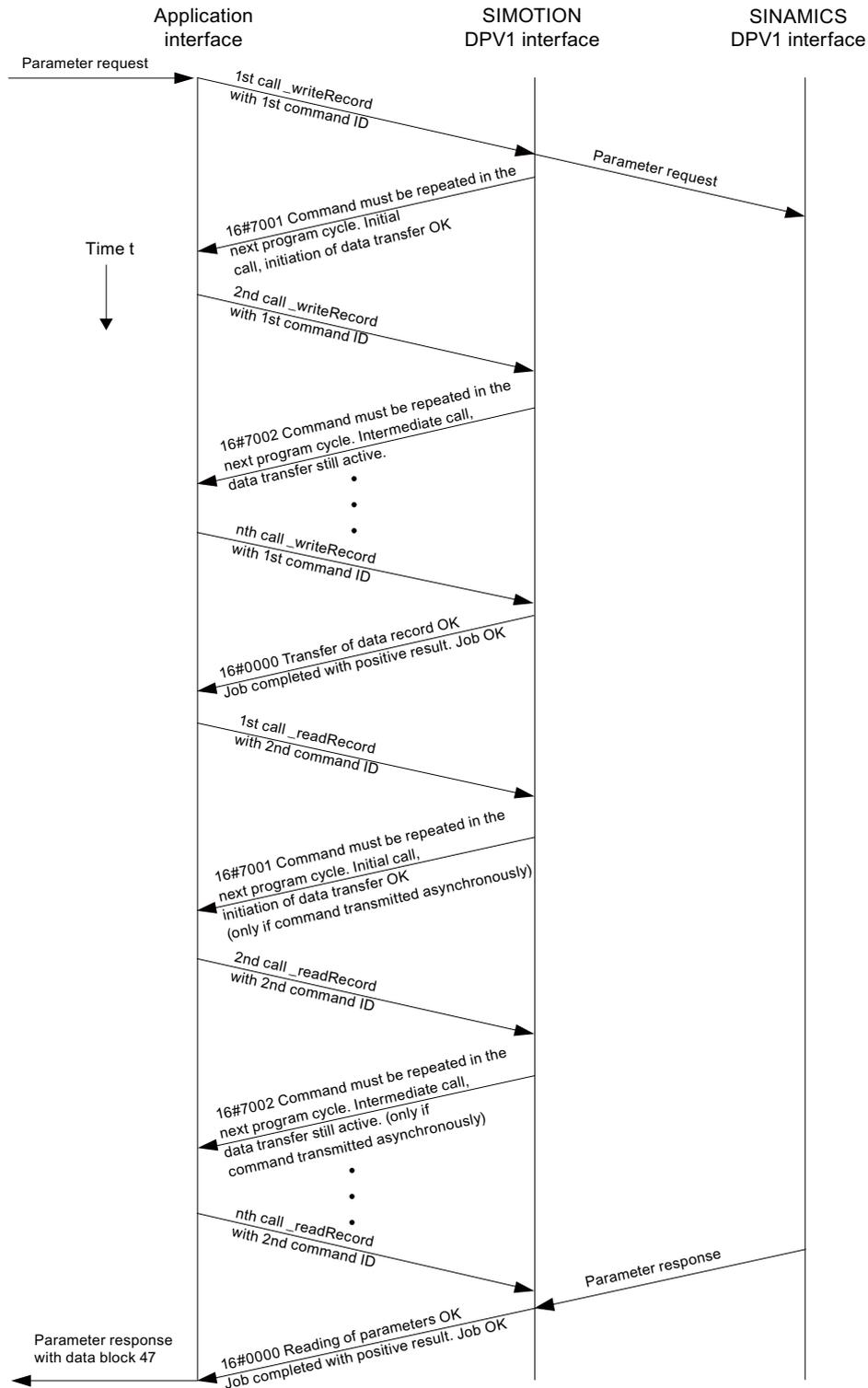


Figure 4-3 Correct processing with the \_readRecord and \_writeRecord system functions

### 4.2.8.3 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 53).

### 4.2.8.4 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.

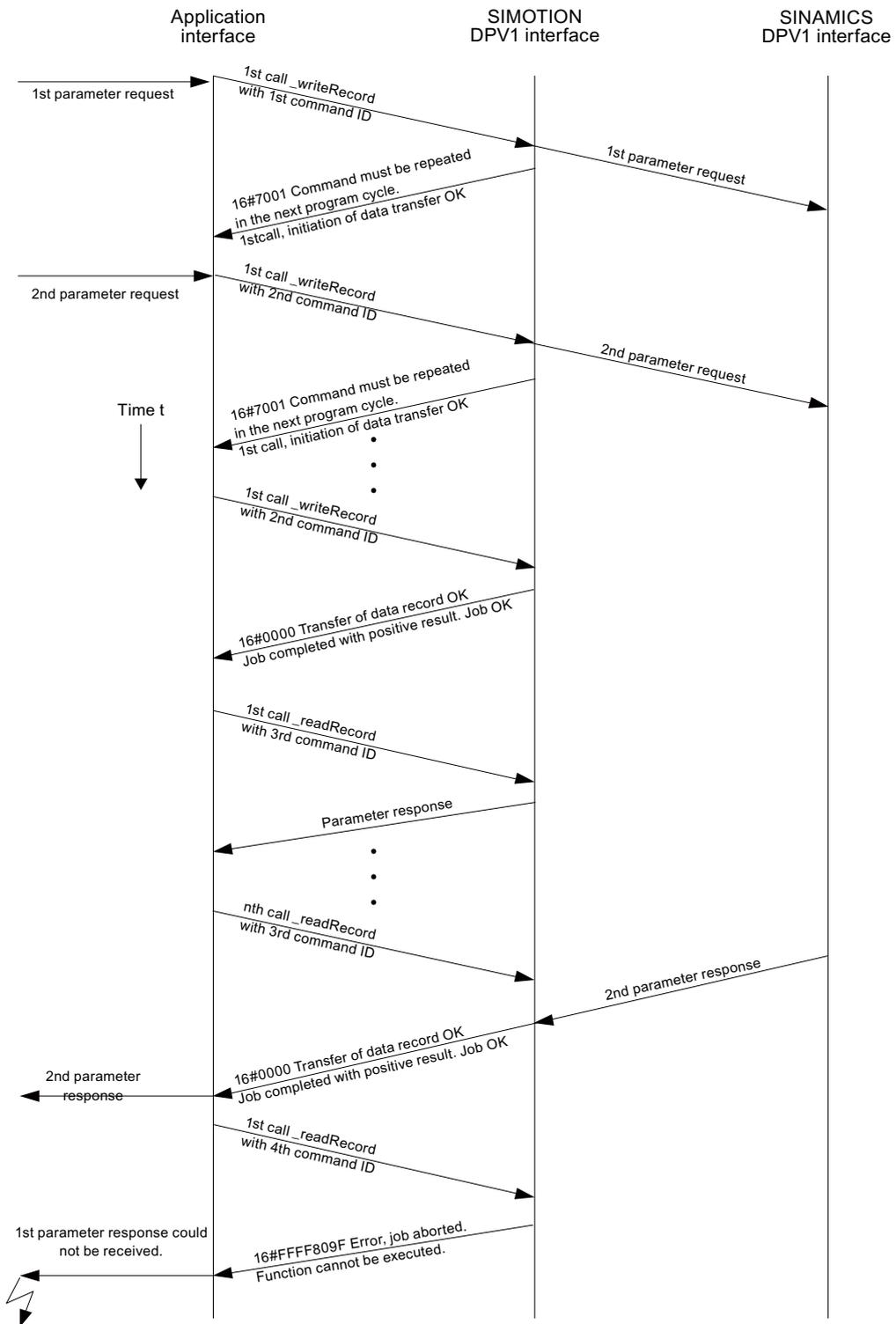


Figure 4-4 The second `_writeRecord` call wins in case of doubt

#### 4.2.8.5 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 47)) 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.

---

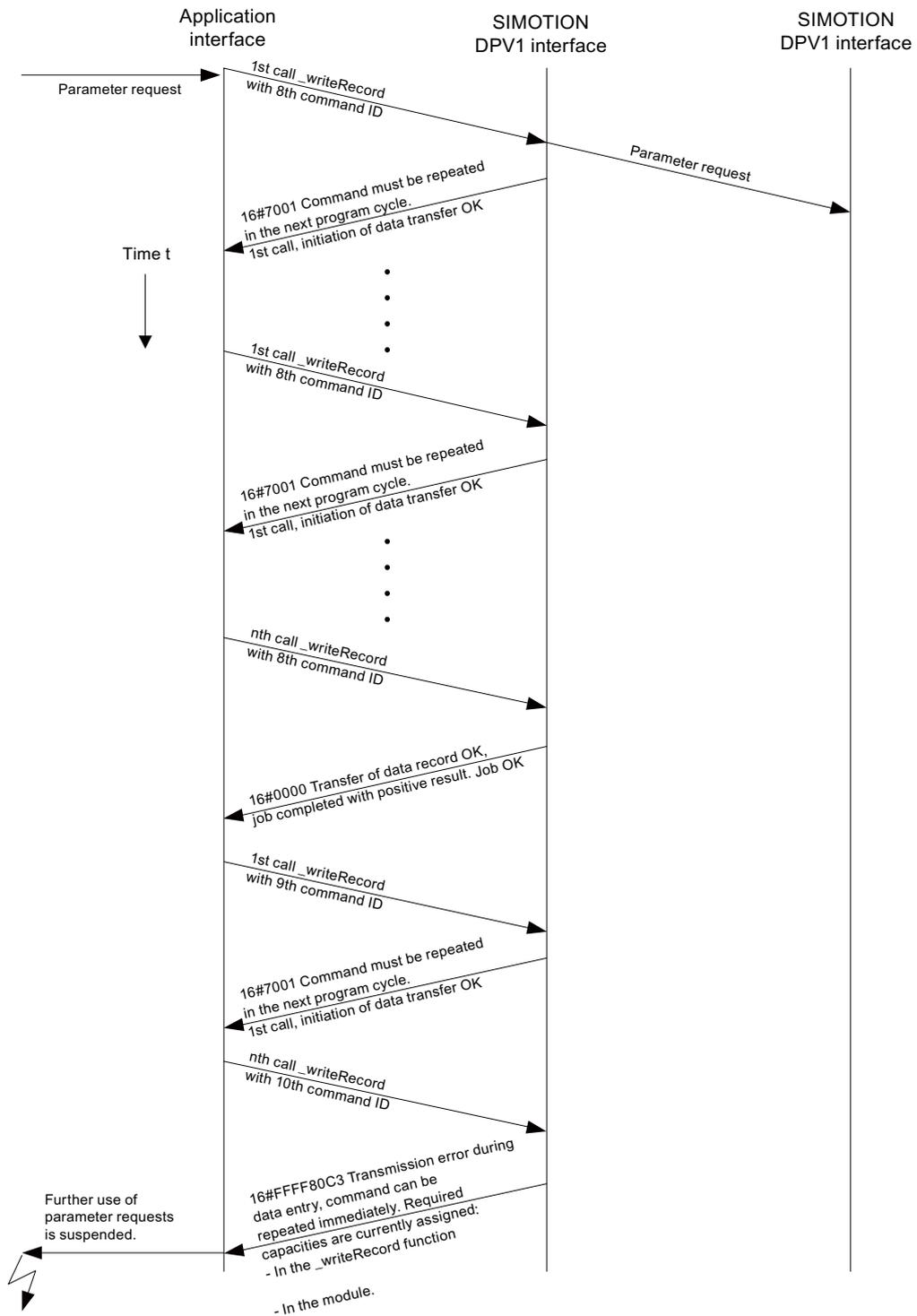


Figure 4-5 Managing 8 jobs simultaneously

**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.

---

## 4.2.9 Rules for SIMOTION `_writeDrive.../_readDrive...` commands

### 4.2.9.1 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.

### 4.2.9.2 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.

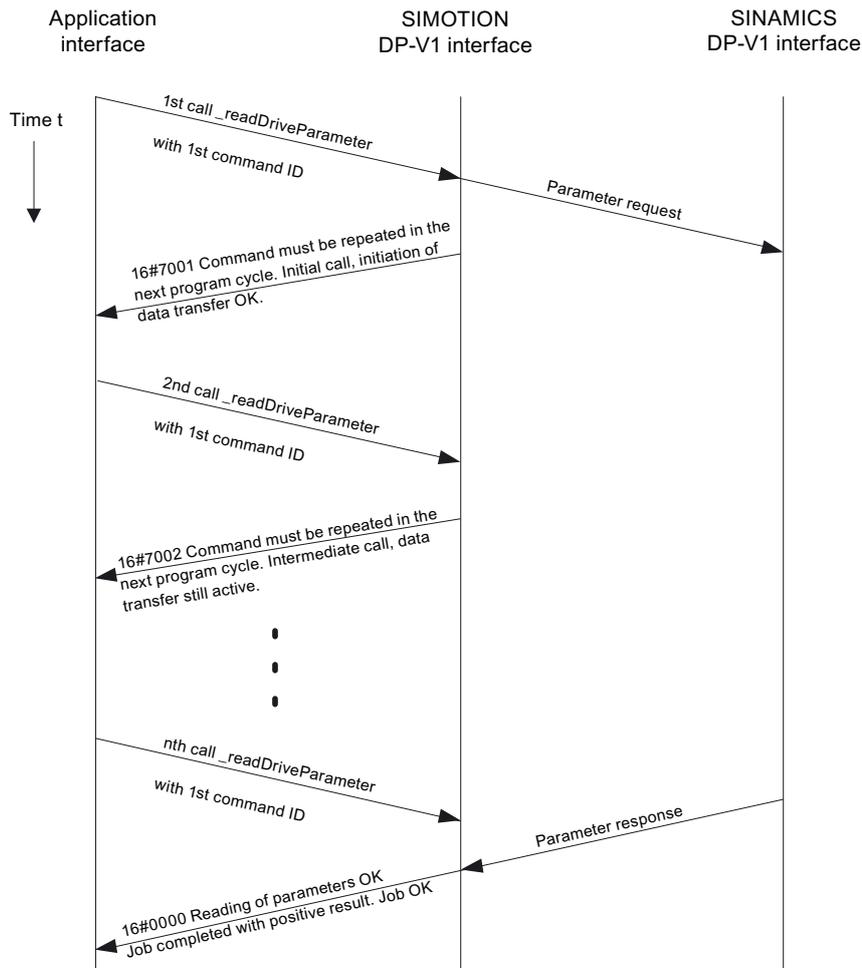


Figure 4-6 Correct processing with the `_writeDriveParameter` and `_readDriveParameter` system functions

### 4.2.9.3 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 54).

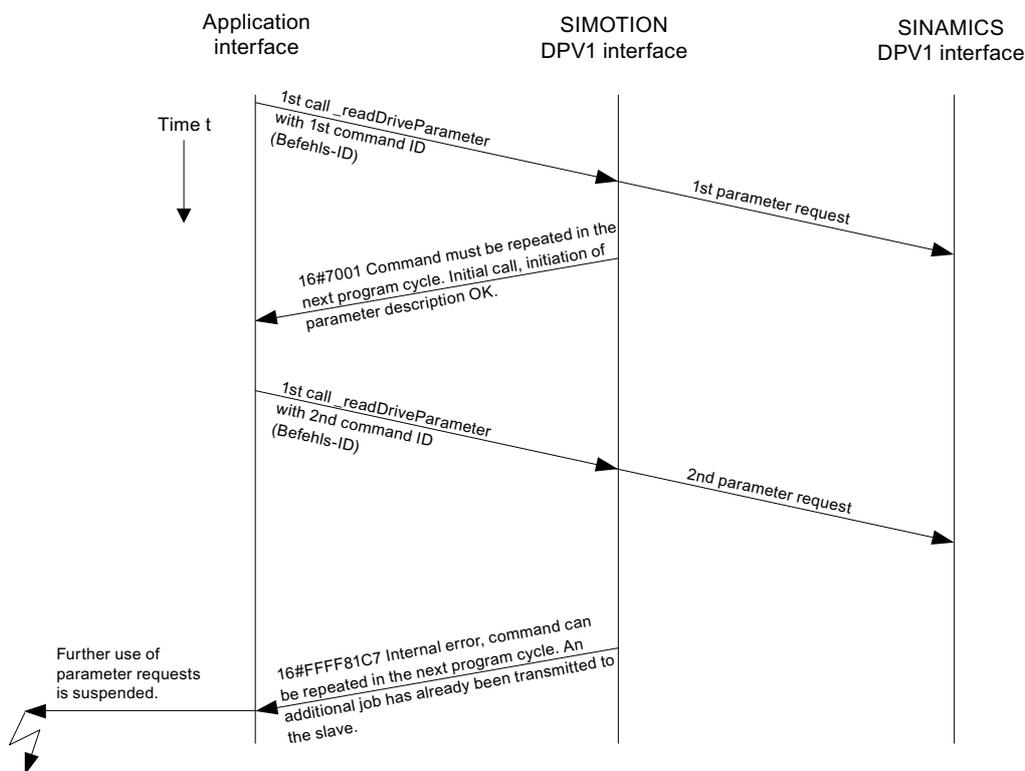


Figure 4-7 Interlocking of several `_readDriveParameter` jobs on a target device

#### 4.2.9.4 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.

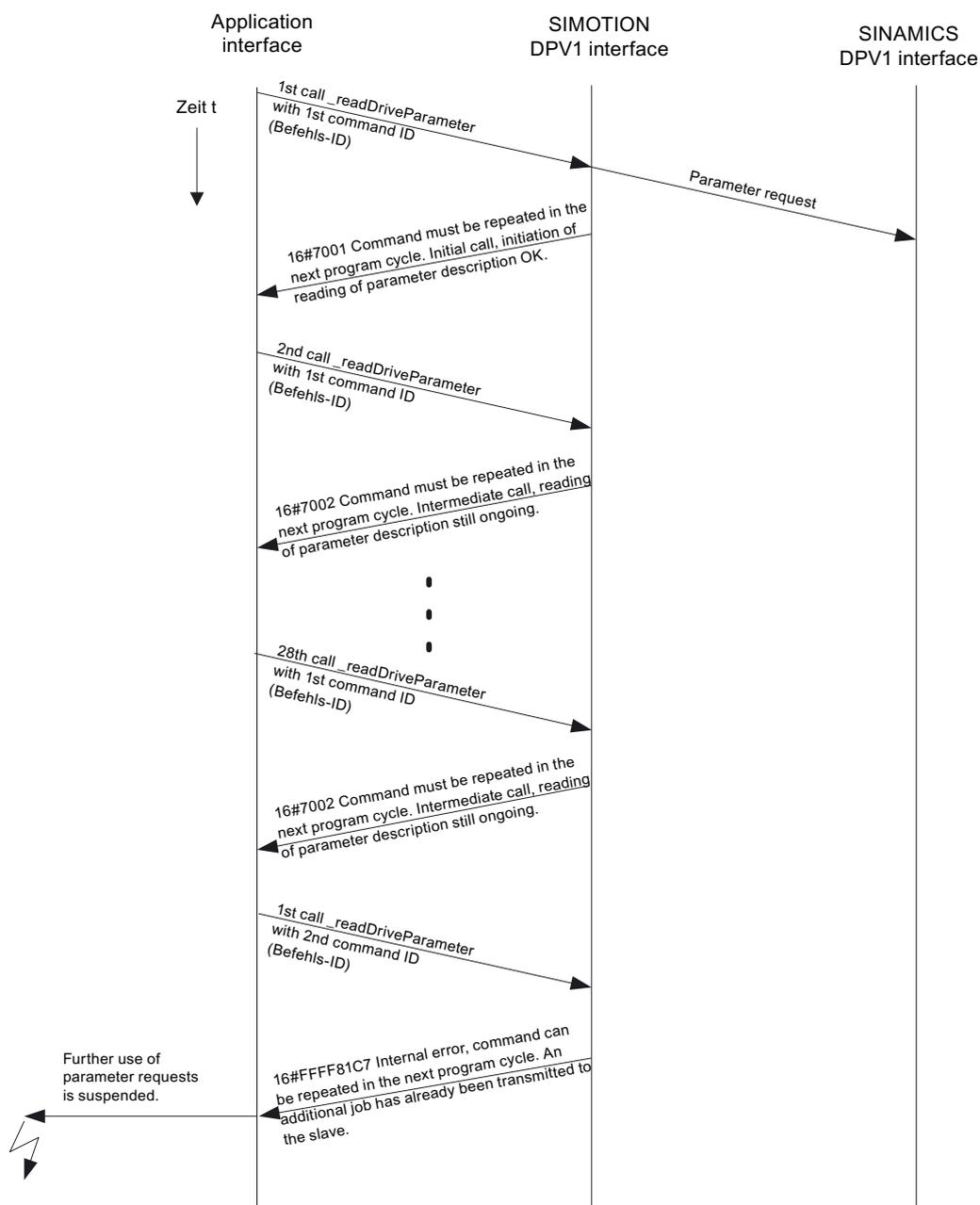


Figure 4-8 Complete processing of a `_readDriveParameter` required to release the interlock

#### 4.2.9.5 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 \(Page 44\)](#) .

---

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.

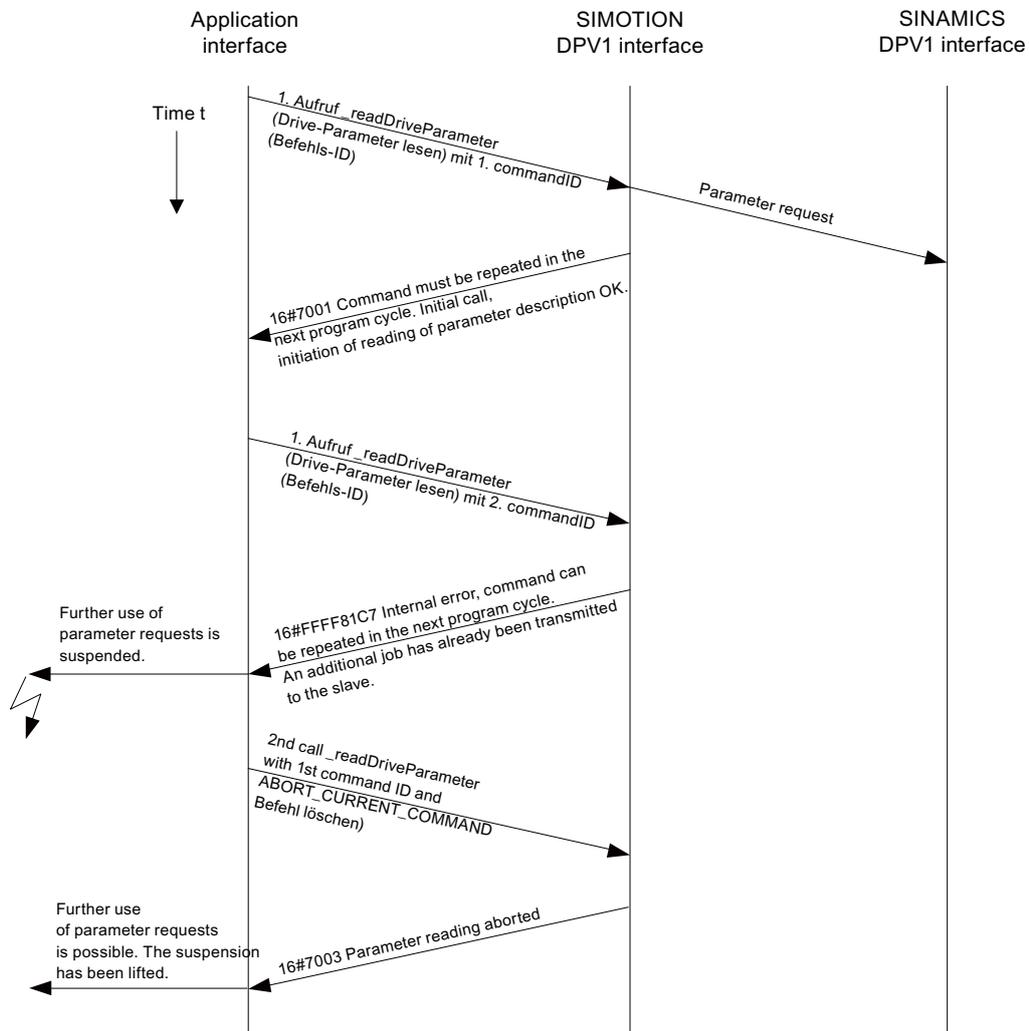


Figure 4-9 Canceling a \_readDriveParameter job with known commandID

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.

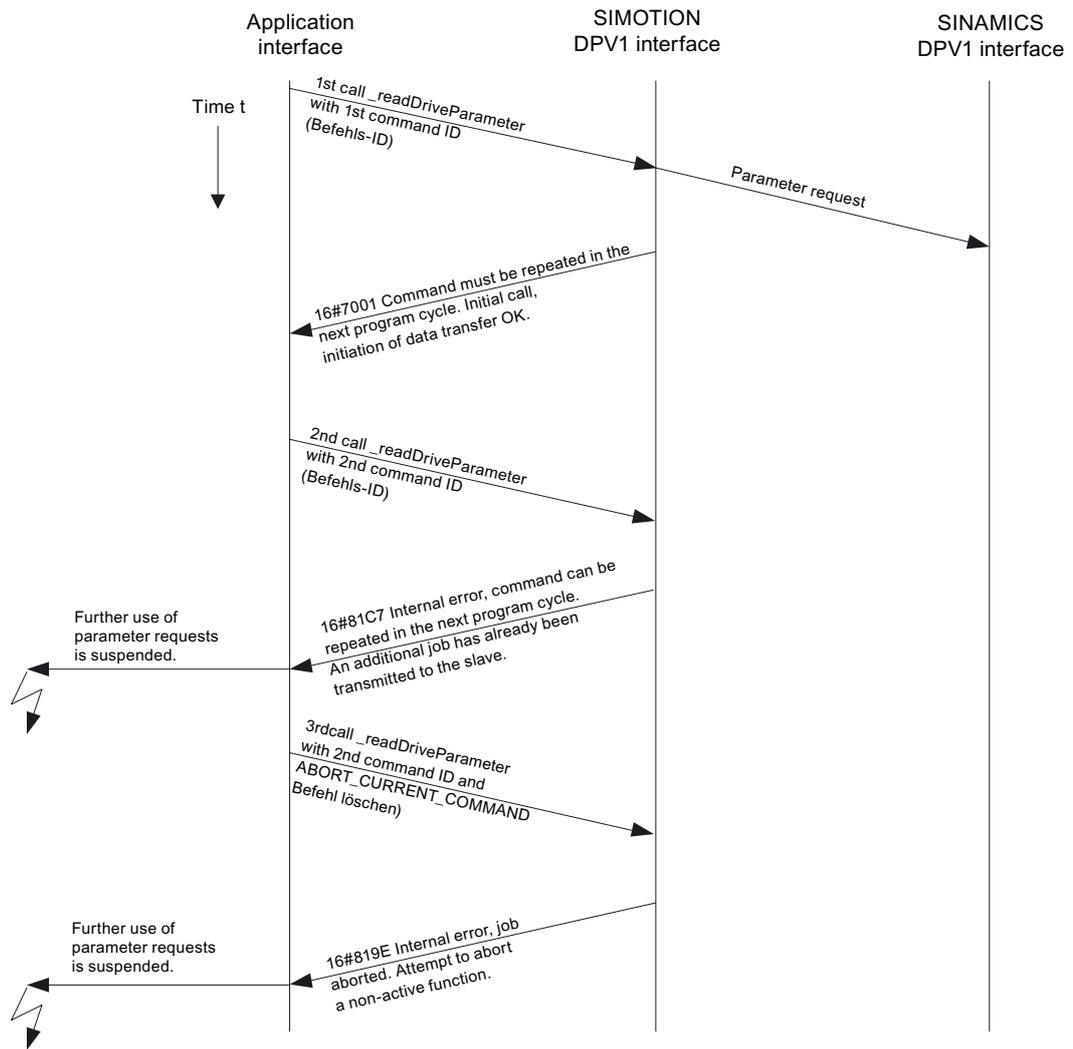


Figure 4-10 No cancelation of a `_readDriveParameter` job with new CommandID

**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.

#### 4.2.9.6 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 49).

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.

#### 4.2.9.7 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.

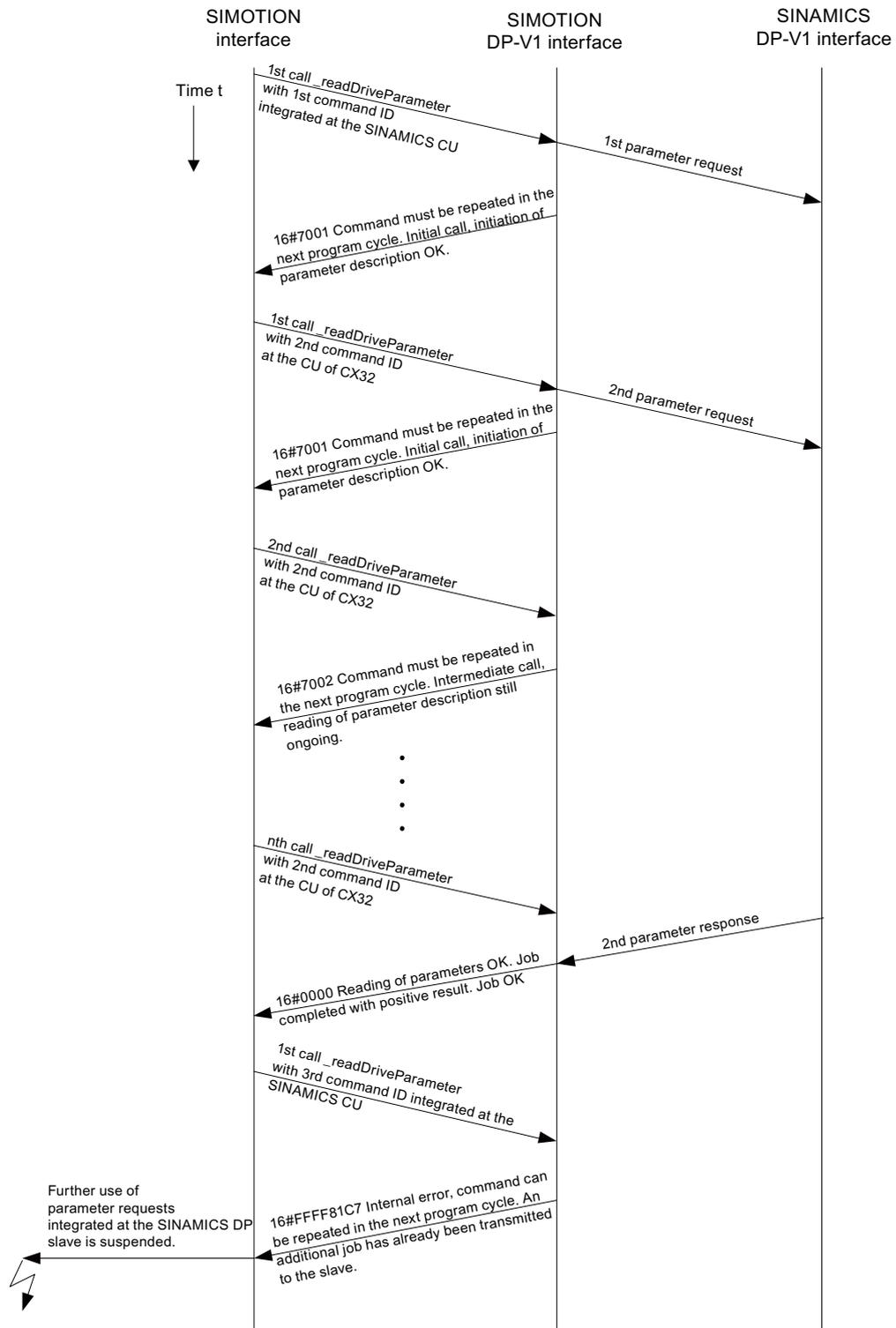


Figure 4-11 Parallel processing of `_readDriveParameter` jobs to different drive devices of a controller

## 4.2.10 Special features

### 4.2.10.1 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.

### 4.2.10.2 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.

4.2 DP V1 acyclic communication

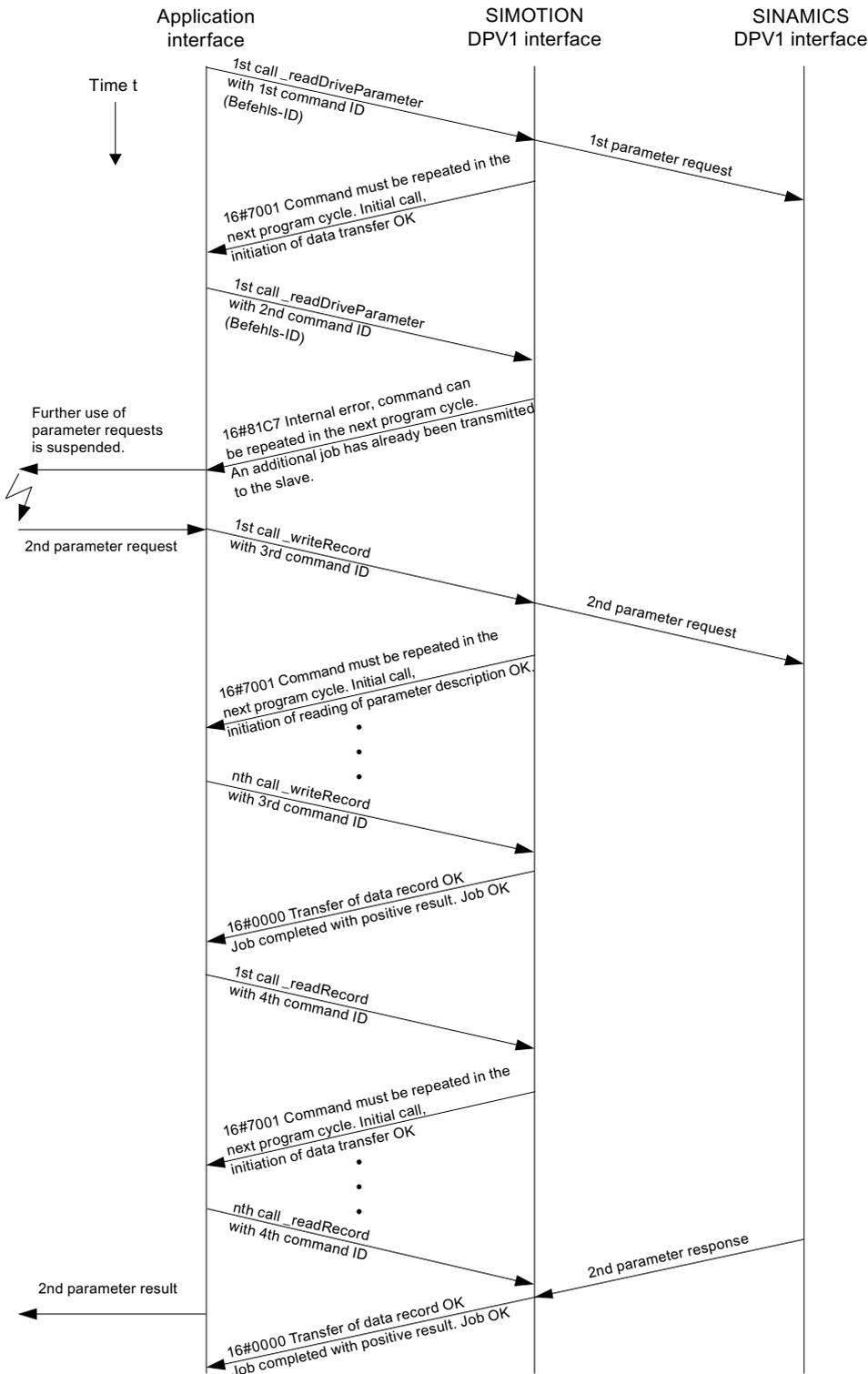


Figure 4-12 Mixed use of `_readDrive...` and `_read/_writeRecord`

#### 4.2.10.3 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 PROFdrive profile specifies that a PROFdrive drive does not perform any pipelining and consequently can process only one job at any given time.

## 4.2.11 Program examples

### 4.2.11.1 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;
ParameterAddress : 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

```

4.2 DP V1 acyclic communication

```

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
```

## 4.3 Communication with SIMATIC S7

### 4.3.1 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.

---

### 4.3.2 SIMOTION as DP slave on a SIMATIC S7

#### 4.3.2.1 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.

#### 4.3.2.2 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 4-1 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.

---

### 4.3.2.3 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.

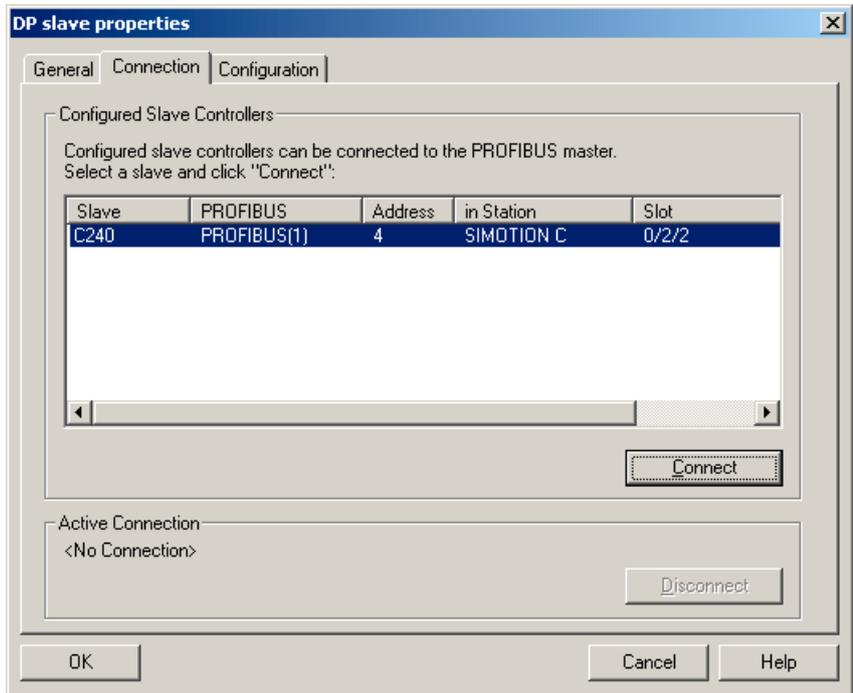


Figure 4-13 DP slave properties

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:

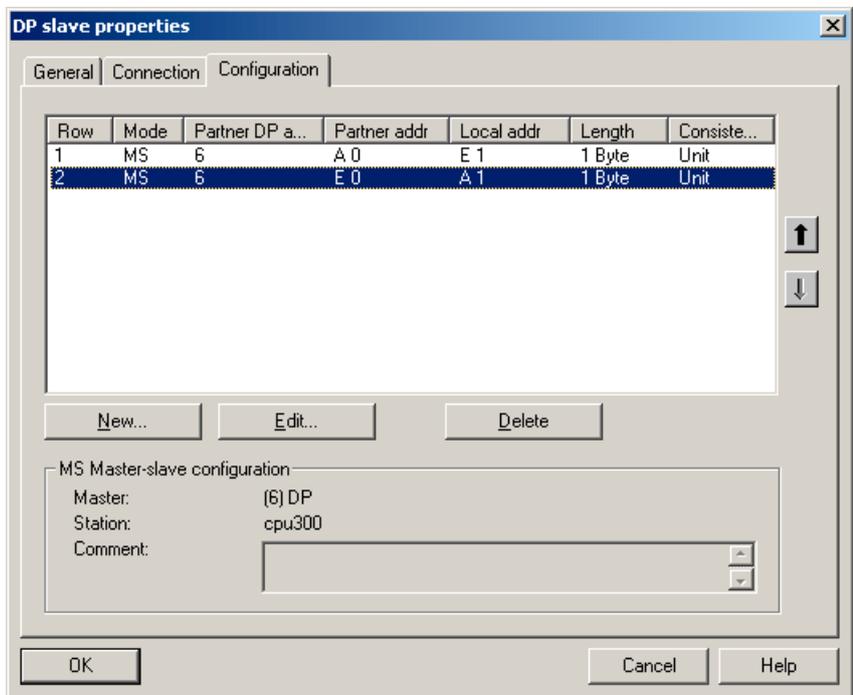


Figure 4-14 Properties - configuration

- 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.

### 4.3.3 SIMATIC S7 as DP slave on a SIMOTION

#### 4.3.3.1 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.

#### 4.3.3.2 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.

### 4.3.3.3 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.

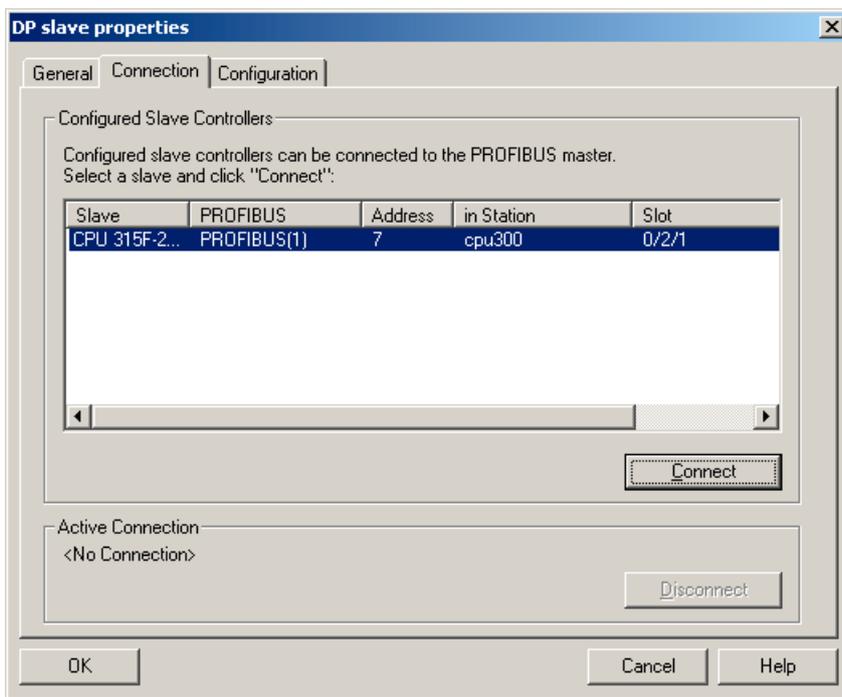


Figure 4-15 Properties - link

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:

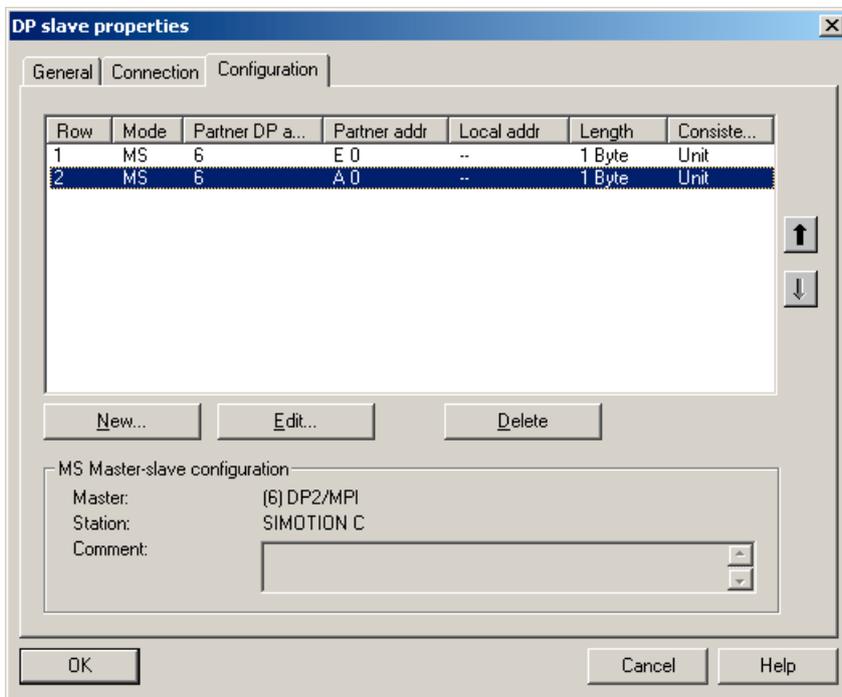


Figure 4-16 Configuration - address selection

- 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.

### 4.3.4 PROFIBUS master-master connection between SIMATIC and SIMOTION

#### 4.3.4.1 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 4-2 Master-master communication

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

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.

#### 4.3.4.2 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_Empfangen:=
    _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)

### 5.1 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.

### 5.2 Configuring Ethernet subnets with SIMOTION

#### 5.2.1 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)

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

### 5.3.1 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.

### 5.3.2 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)

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

- 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)

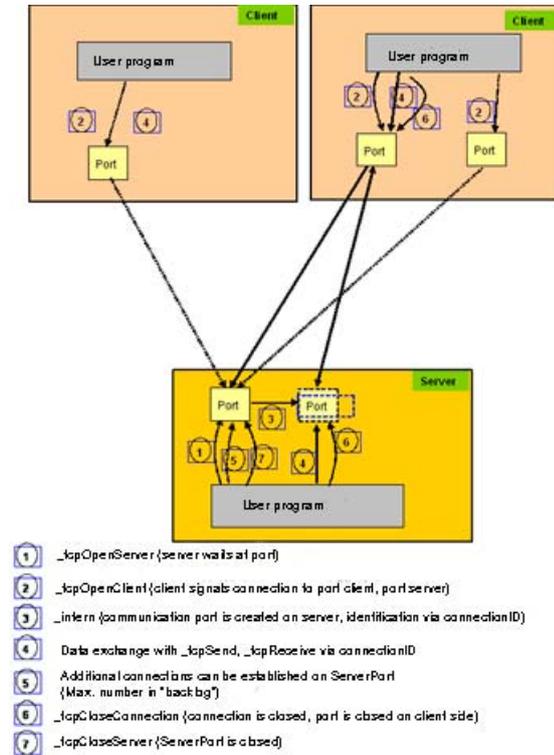


Figure 5-1 Principle communication sequence of TCP/IP communication

### 5.3.3 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 5-1 Communication between a sender (client) and a receiver (server)

	Function
Establish the connection <ul style="list-style-type: none"> <li>• Receiver/server waits for communication request</li> <li>• Sender/client requests connection to be established to the receiver</li> <li>• Receiver/server has established communication request</li> <li>• No further connection is required</li> </ul>	_tcpOpenServer _tcpOpenClient  _tcpCloseServer
Communicating <ul style="list-style-type: none"> <li>• Sender sends data to the receiver</li> <li>• Receiver receives data from the sender</li> </ul>	_tcpSend _tcpReceive
Terminating communication connection <ul style="list-style-type: none"> <li>• Sender no longer sends data and closes the connection</li> </ul>	_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.

### 5.3.4 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

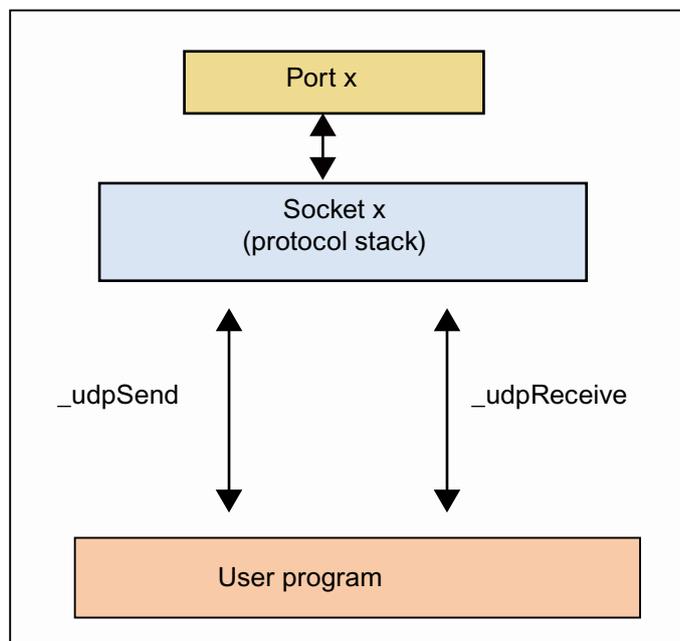


Figure 5-2 UDP communication model

## See also

Function `_udpSend` (Page 115)

Function `_udpReceive` (Page 116)

UDP connection (Page 92)

### 5.3.5 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 5-2 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, _tcpCloseConnection, _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  $\leq 240$  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 5-3 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	1400 bytes	
SFC65 (XSEND), SFC66 (XRCV) (S7-300, S7-400)			76 bytes
_Xsend, _Xreceive (SIMOTION C2xx, D4xx, P350)			200 bytes

### 5.3.6 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 84).
- 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.

## 5.4 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.

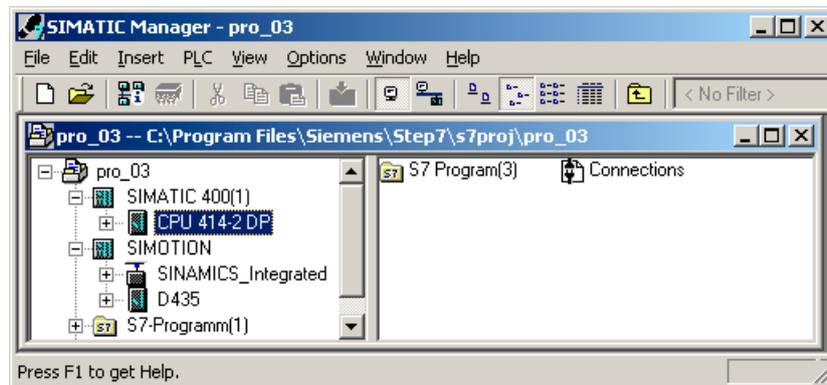


Figure 5-3 Display of the "Connections" object in the SIMATIC Manager

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

### 5.5.1 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.

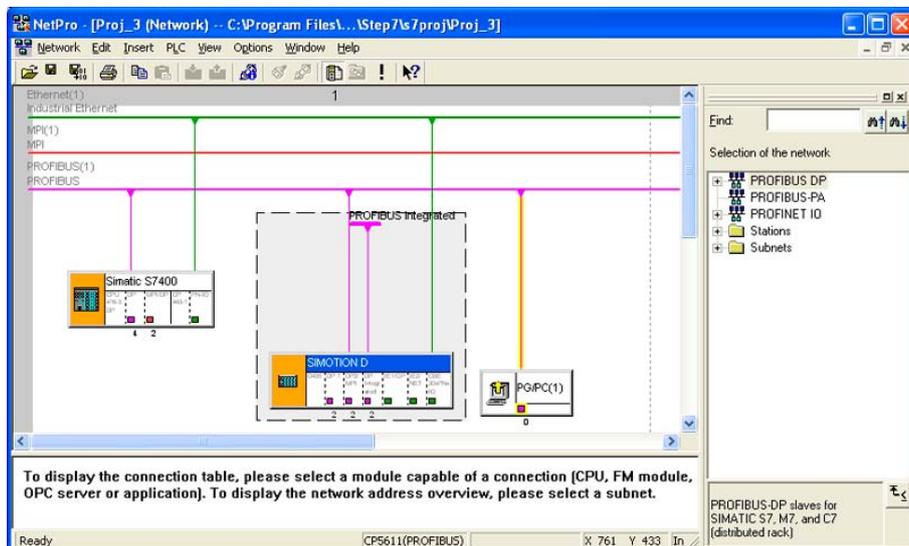


Figure 5-4 Selected S7 CPU and the associated connection table

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.

## 5.5.2 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.

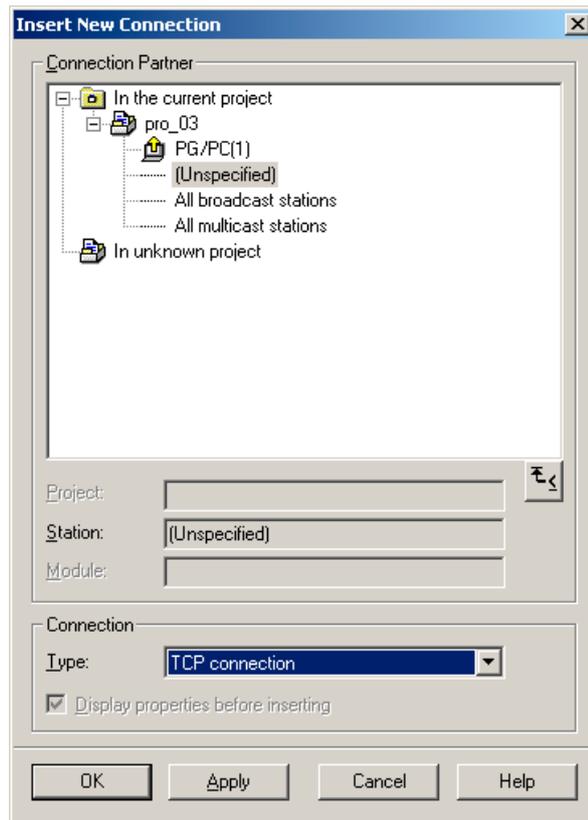


Figure 5-5 **Insert New Connection** dialog box with selected TCP/IP connection

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.

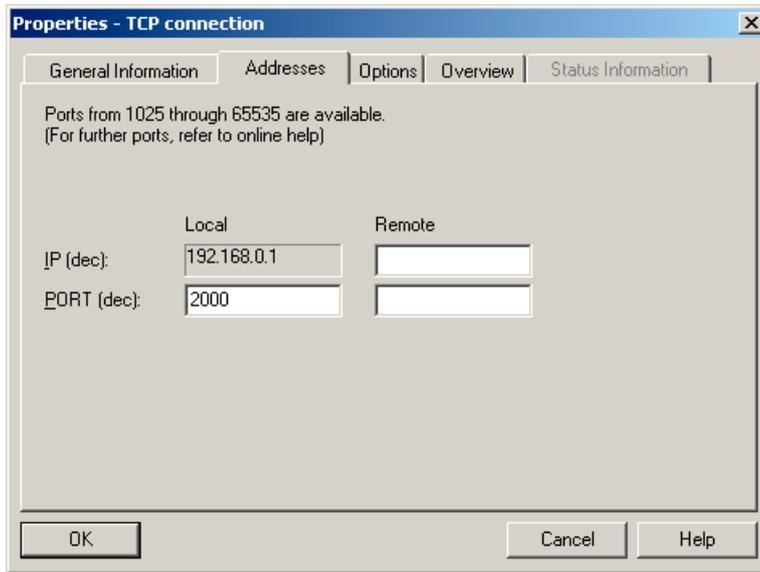


Figure 5-6 "Properties - TCP Connection" dialog box - "Addresses" tab

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.

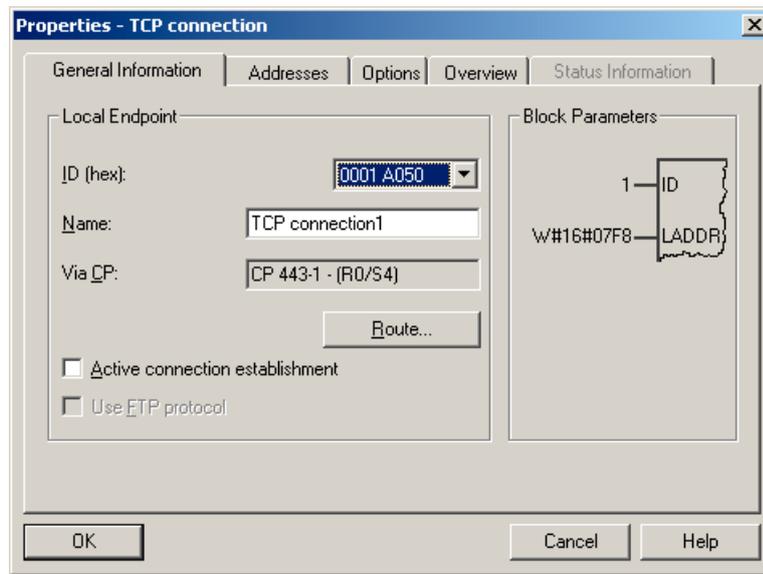


Figure 5-7 Properties - TCP Connection dialog box - General tab

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.

### 5.5.3 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.

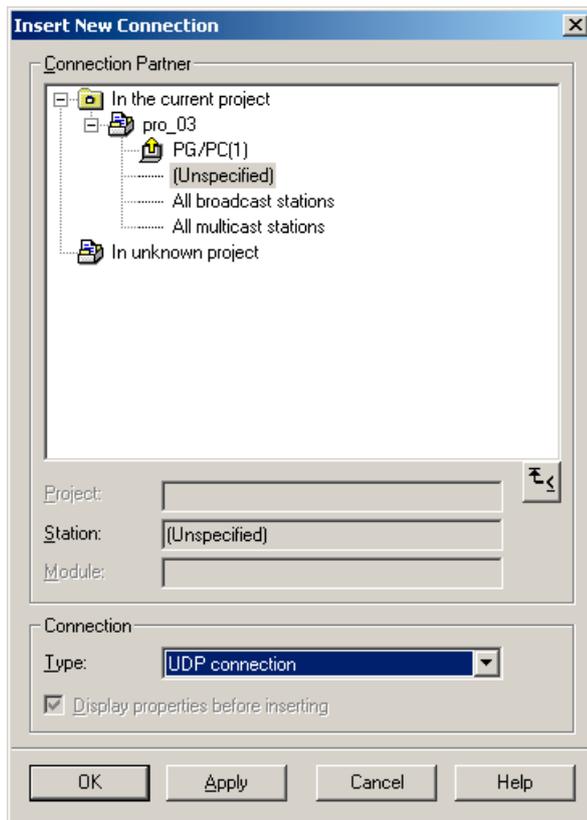


Figure 5-8 **Insert New Connection** dialog box with selected UDP connection

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.

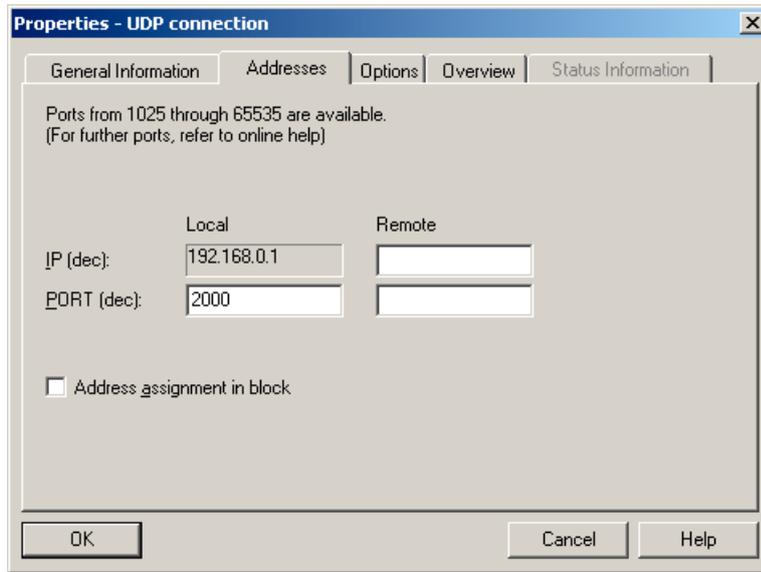


Figure 5-9 Properties - UDP Connection dialog box - Addresses tab

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".

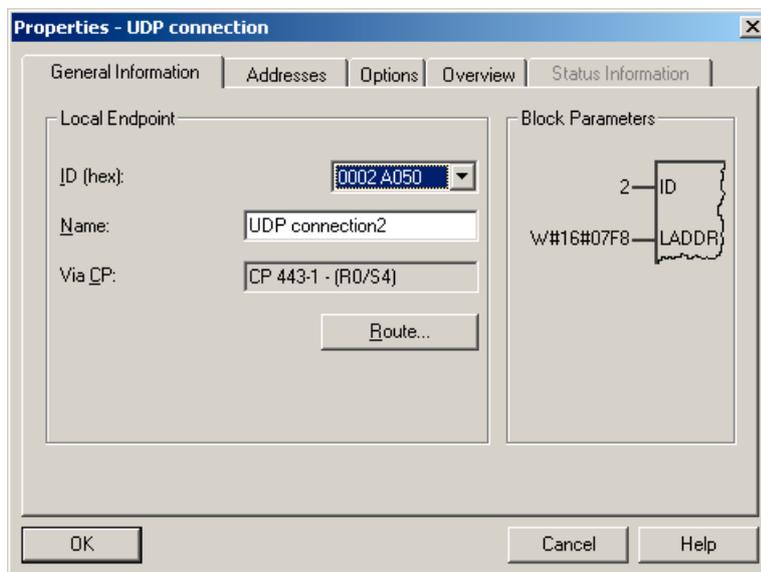


Figure 5-10 Properties - UDP Connection dialog box - General tab

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

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

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.

## 5.6 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 105).

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

## 5.7 Using the functions and function blocks in the user program

### 5.7.1 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.

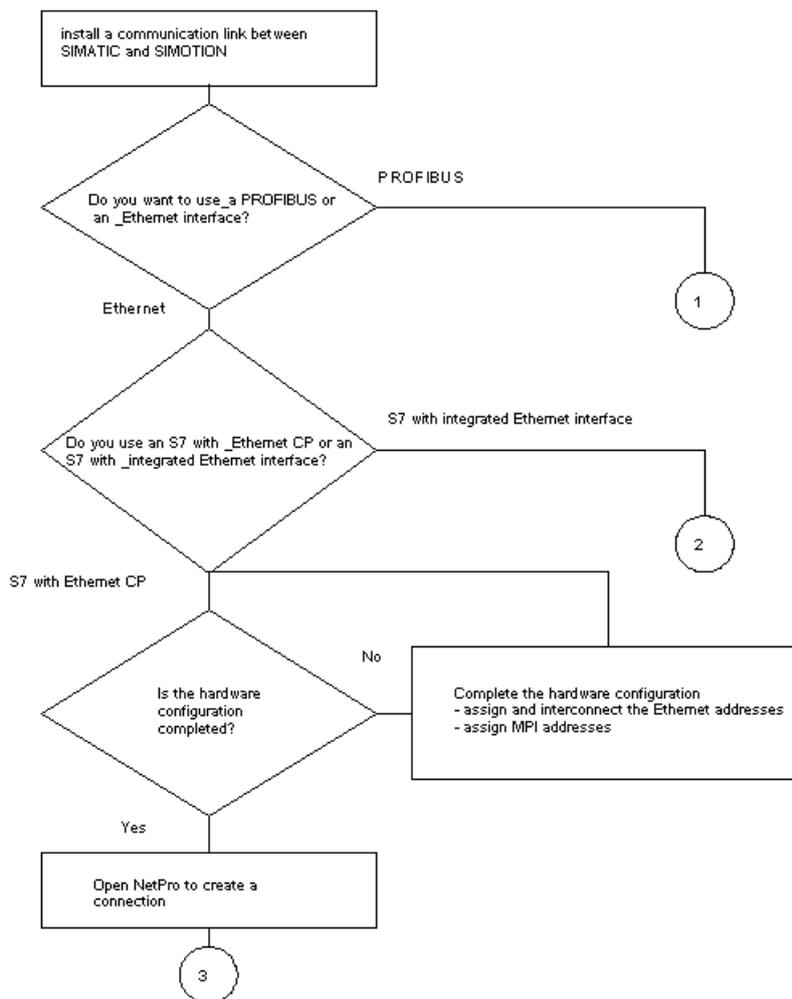


Figure 5-11 Flowchart for the configuration of a communication connection: selection of the communications protocol and the SIMATIC station

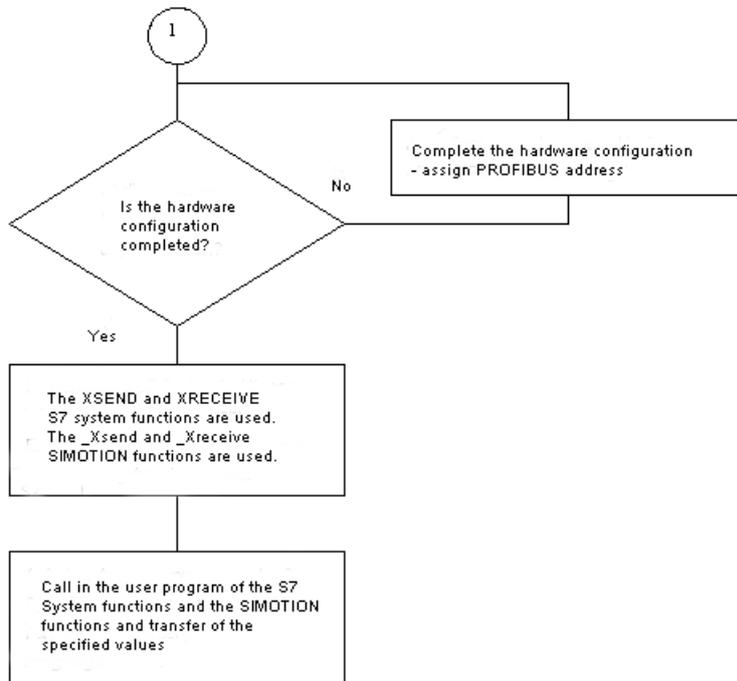


Figure 5-12 Flowchart for the configuration of a communication connection - continued: PROFIBUS connection

**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 75).

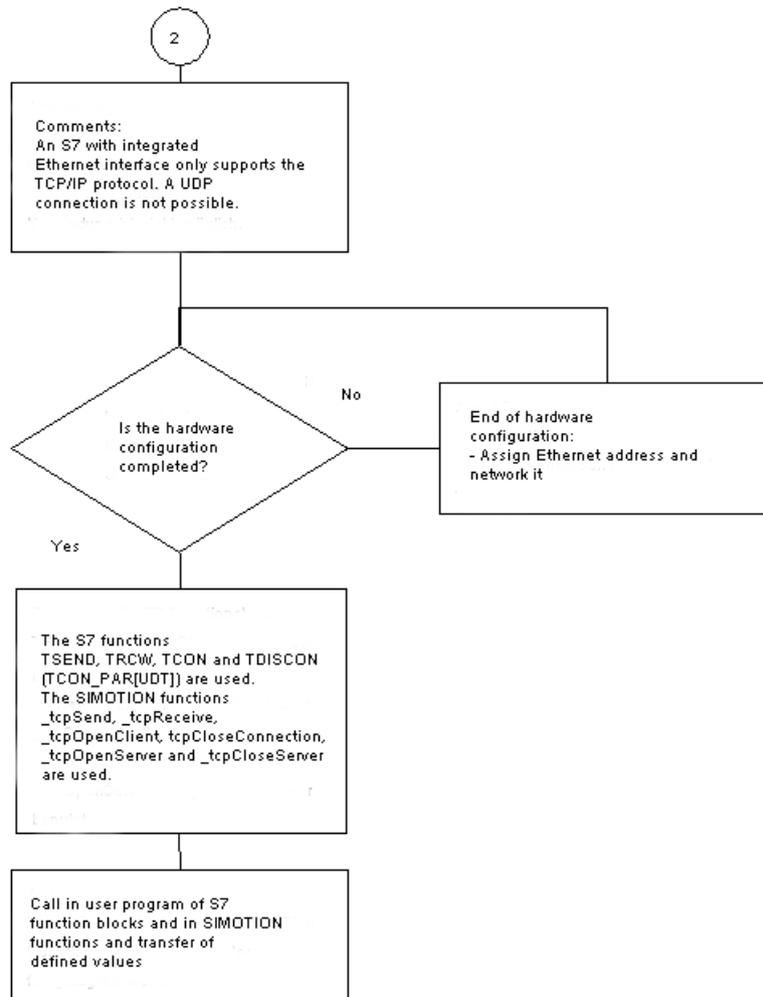


Figure 5-13 Flowchart for the configuration of a communication connection - continued: TCP/IP connection (S7 with integrated Ethernet interface)

---

**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 94).

---

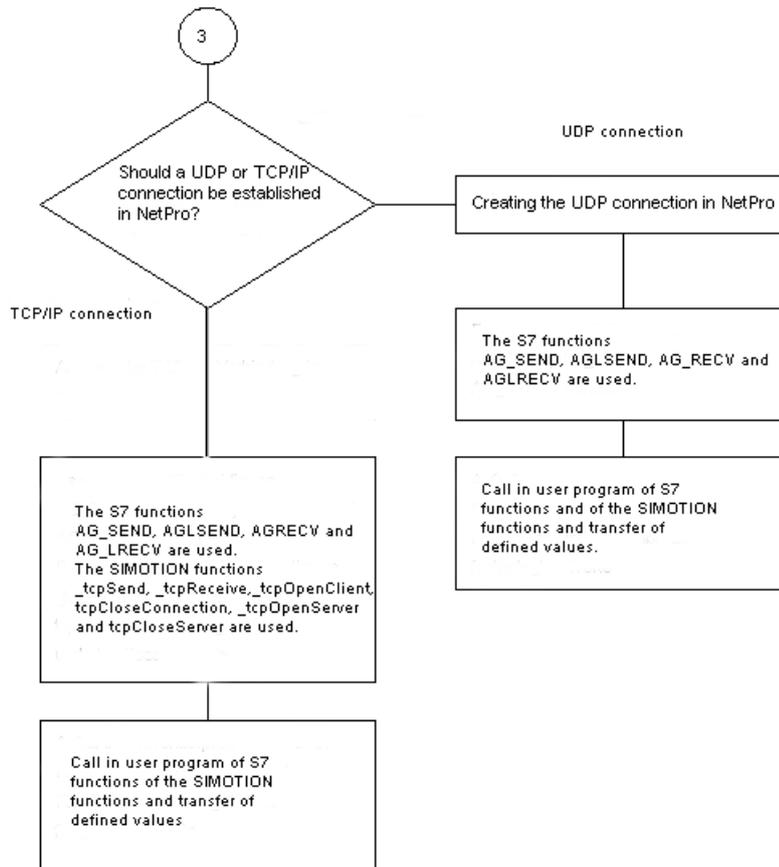


Figure 5-14 Flowchart for the configuration of a communication connection - continued: TCP/IP and UDP connection (S7 with Ethernet CP)

**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 88) )
- 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 94) )

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

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

### 5.7.2.1 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.

### 5.7.2.2 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 5-4 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 5-5 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.

### 5.7.2.3 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 `_tcp`send and `_tcp`receive 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 `_tcp`openclient 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 `_tcp`openclient 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 `_tcp`openclient 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 `_tcp`send, `_tcp`receive and `_tcp`closeconnection 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.ConnectionID,
   nextcommand       :=WHEN_COMMAND_DONE,
   datalength       :=Soll_Sende_Datenlaenge,
   data              :=TCP_Sende_Daten);
```

Call example of the SIMOTION `_tcp`send function

The `_tcp`send 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 `_tcp`openclient or `_tcp`openserver 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  
  &#1921; 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.

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

#### 5.7.3.1 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.

#### 5.7.3.2 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.

#### 5.7.3.3 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.

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

### 5.7.4.1 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.

### 5.7.4.2 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
+13.0		BYTE		
+28.0	rem_subnet_id	ARRAY[1..6]	B#16#0	
+29.0		BYTE		
+34.0	rem_staddr	ARRAY[1..6]	B#16#0	xxx Remote IP-Adr.
+35.0		BYTE		
+40.0	rem_tsap_id	ARRAY[1..16]	D#16#0	
+41.0		BYTE		
+56.0	next_staddr	ARRAY[1..6]	B#16#0	
+57.0		BYTE		
+62.0	spare	WORD	W#16#0	
+64.0		END_STRUCT		

Figure 5-15 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.

#### 5.7.4.3 SIMOTION functions

The same functions (`_tcpopenclient`, `_tcpsend`, `_tcpreceive`, `_tcpcloseconnection`, `_tcpopenserver`, `_tcpclosesserver`) 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.

#### 5.7.5 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

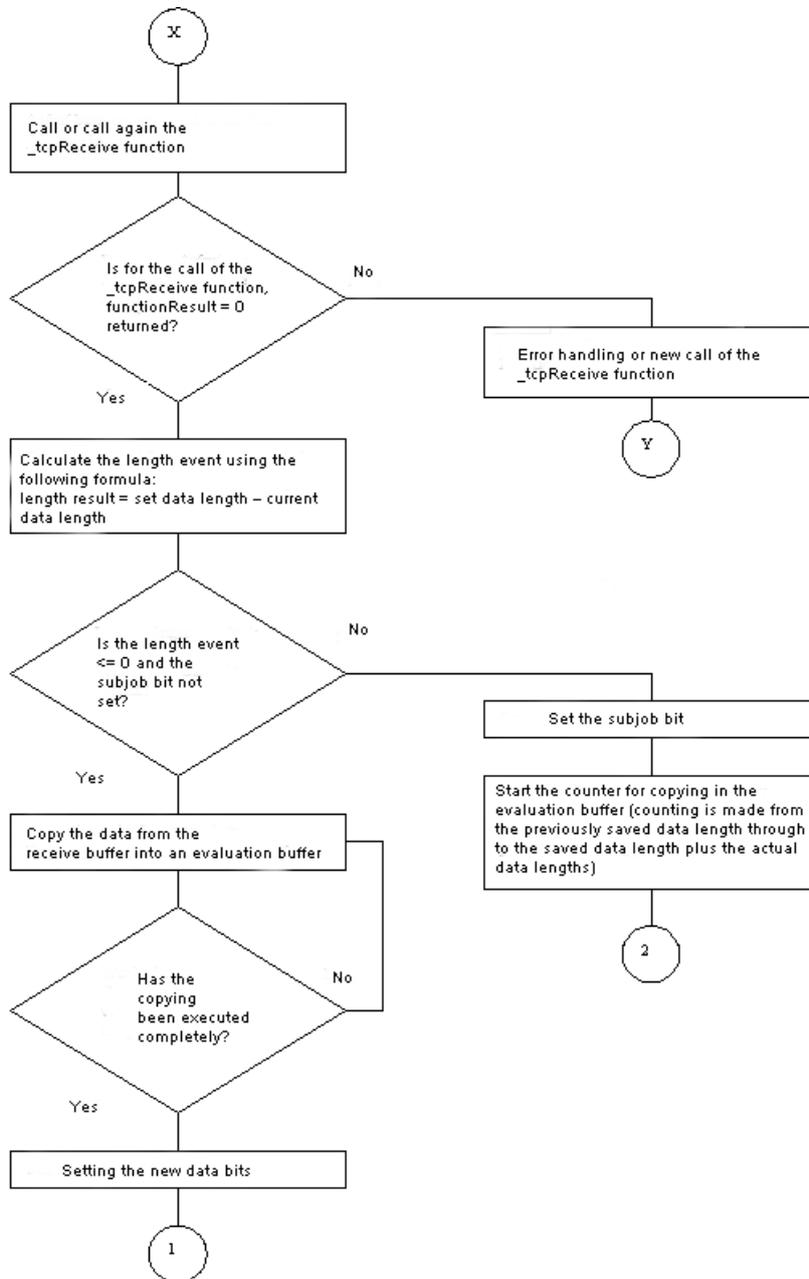


Figure 5-16 Flowchart for receipt at SIMOTION end using \_tcpReceive

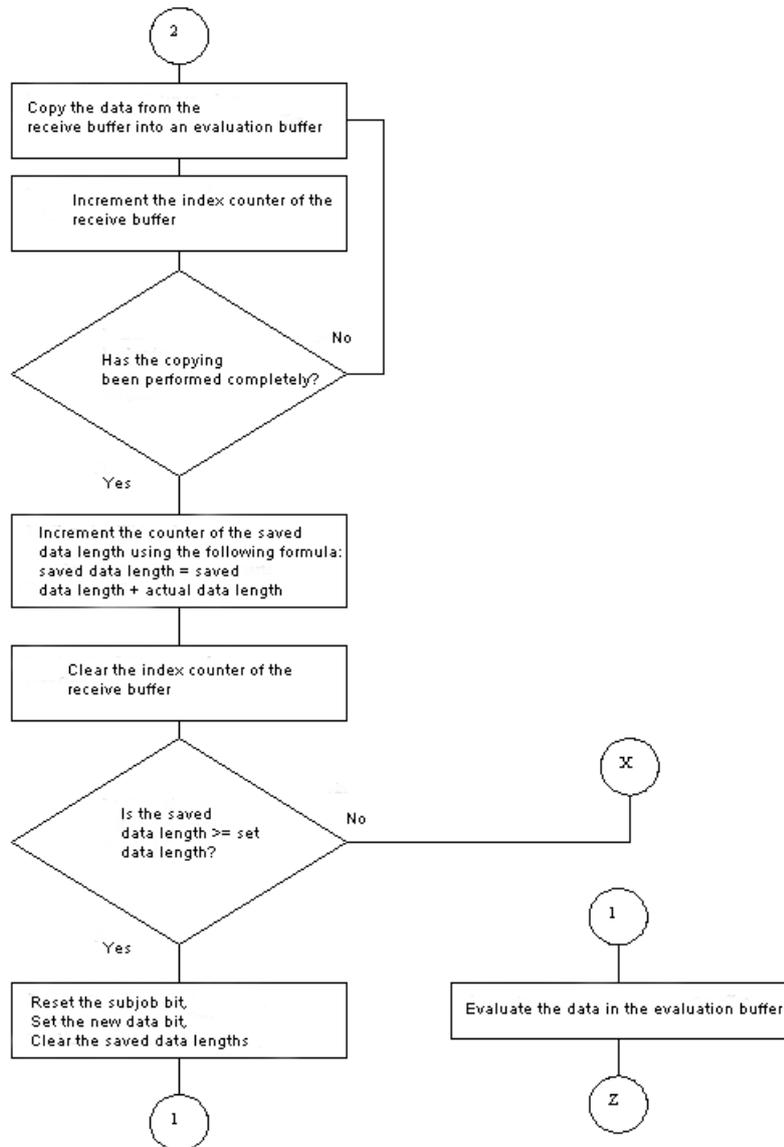


Figure 5-17 Flowchart for receipt at SIMOTION end using \_tcpReceive - continued

## 5.8 Details of the SIMOTION TCP/IP system functions

### 5.8.1 `_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,  
    nextCommand :EnumTcpNextCommandMode  
);
```

### 5.8.2 `_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  
);
```

### 5.8.3 `_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  
);
```

### 5.8.4 `_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]  
);
```

### 5.8.5 **\_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  
);
```

### 5.8.6 **\_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  
);
```

## 5.9 **Details of the SIMOTION UDP system functions**

### 5.9.1 **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.

**5.9.2 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

### 6.1 PROFINET IO overview

#### 6.1.1 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. Only one node can access the bus at any one time (half-duplex operation). With PROFINET IO a switching technology is implemented that allows all stations to access the network at any time. In this way, the network can be used much more efficiently through the simultaneous data transfer of several nodes. Simultaneous sending and receiving is enabled through the full-duplex operation of Switched Ethernet. PROFINET IO is based on Switched Ethernet full-duplex operation and a bandwidth of 100 Mbit/s.



---

#### More information

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

---

### 6.1.2 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.

### 6.1.3 IO controller

The IO controller provides an IO system for the I/O data communication of the distributed field devices. The IO controller is usually the communication interface of a SIMOTION device, for example, in this case SIMOTION D with CBE30-PN. The function is comparable to a PROFIBUS DP master class 1.

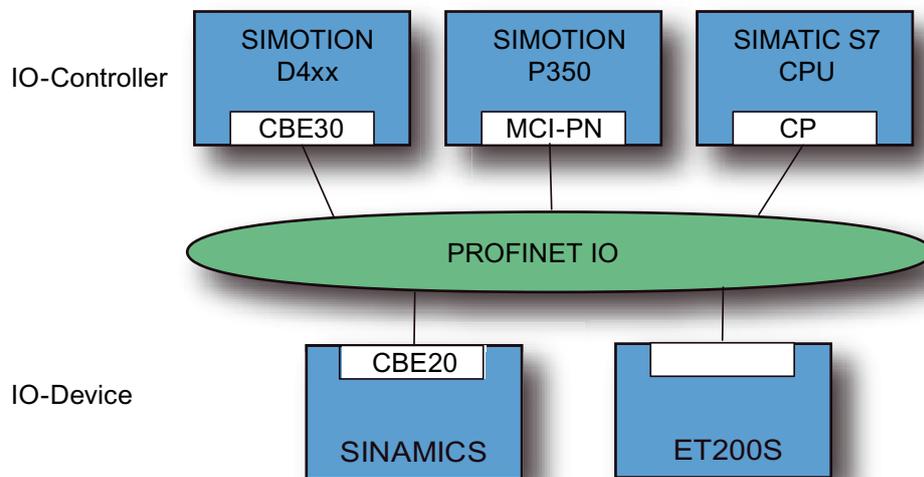


Figure 6-1 Examples of IO controllers and IO devices

#### 6.1.4 IO device

Distributed field devices such as I/O, drives, e.g. SINAMICS S120 with CBE20-PN, or control terminals are called IO devices. The function is comparable to a PROFIBUS DP slave.

##### See also

Creating an IO device (Page 157)

#### 6.1.5 Sync domain

##### Description

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.

##### See also

Creating a sync domain (Page 149)

#### 6.1.6 iDevice

##### Description

The functionality of an iDevice under PROFINET is comparable with that of an iSlave for PROFIBUS. This means that a SIMOTION CPU can assume the role of an IO device and, as a result, exchange data with a different IO controller.

Whereas for PROFIBUS an interface can be either only master or only slave, for PROFINET it is possible to be concurrently both IO controller and IO device on the same PN interface.

##### See also

PROFINET iDevice (Page 169)

### 6.1.7 Addressing of PROFINET IO devices

A worldwide unique MAC (Media Access Control) address is used for the data exchange at the hardware level. IP (Internet Protocol)-based services, such as HTTP (Web applications) or FTP (file transfer), use IP addresses to address the devices. Both addresses are made up of several bytes.

PROFINET therefore uses a unique device name (NameOfStation) to identify PROFINET devices. The device name is a string that fulfills the requirements of a DNS (Domain Name Service) name.

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 (so-called 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.

If configured, the IO controller automatically assigns each station a unique IP address via the device name (so-called IP address assignment) during ramp-up. The station can then be accessed via IP services. The IP address can be taken from a configured sequence of numbers or configured individually. If you do not use the automatic IP address assignment, you must assign an IP address to the IO device manually, see Assigning device names and IP addresses to IO devices (Page 161) .

### 6.1.8 RT classes

#### 6.1.8.1 RT classes for PROFINET IO

##### Description

PROFINET IO is a scalable realtime communications system based on Ethernet technology. The scalable approach is expressed in two realtime classes.

##### RT

The RT communication is based on standard Ethernet. The data is transferred via prioritized Ethernet message frames. For more detailed information, see PROFINET IO with RT (Page 126).

##### IRT

Where the RT class is IRT, a distinction is made between two different formats: High Flexibility and High Performance.

## IRT - High Flexibility

The message frames are exchanged in a bandwidth reserved by the hardware. The bandwidth can only be reserved if the transmission network has been synchronized. One IRT time interval and one standard Ethernet time interval are created for each cycle. For more detailed information, see PROFINET IO with IRT (High Flexibility) (Page 128).

## 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 results in even better data exchange performance, as well as ensuring that cyclic data is transmitted in a precise and deterministic manner for any kind of topology. The IRT time interval can thus be optimized or minimized to a greater extent when compared with IRT High Flexibility.

In addition to transmission network synchronization, IRT even allows applications (position control cycle, IPO cycle) to be synchronized in the devices (isochronous application). This is an essential requirement for axis control and isochronous switching of inputs and outputs in the network.

For more detailed information, see PROFINET IO with IRT (High Performance) (Page 129).

## Comparing RT and IRT

Table 6-1 The major differences between RT and IRT

Property	RT	IRT (High Flexibility)	IRT (High Performance)
<b>Transfer mode</b>	Prioritization of cyclic RT data using Ethernet-Prio (VLAN tag)	Bandwidth reservation, i.e. reservation of a time domain in which only cyclic RT data (but no TCP/IP frames) are transmitted	In addition to bandwidth reservation, scheduling of cyclic RT data with consideration given to the topology
<b>Determinism</b>	Variance of the transmission duration for cyclic RT data using TCP/IP message frames	Guaranteed transmission of cyclic RT data within the reserved IRT time interval	Transmission and receiving times for cyclic RT data are precisely defined and guaranteed for all kinds of topologies.
<b>Isochronous application</b>	Not supported	Not supported	Supported
<b>Hardware support using special Ethernet controller</b>	No	Yes	Yes

Table 6-2 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 <i>Note 2)</i>
	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	375, 625, 750, 875, 1,125, 1,250 µs ... 3,875 µs (increment 125 µs)	Not supported	1

**Note 1) Mixed operation, RT/IRT High Performance**

Odd send clocks can only be used if there is no RT or IRTflex 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.

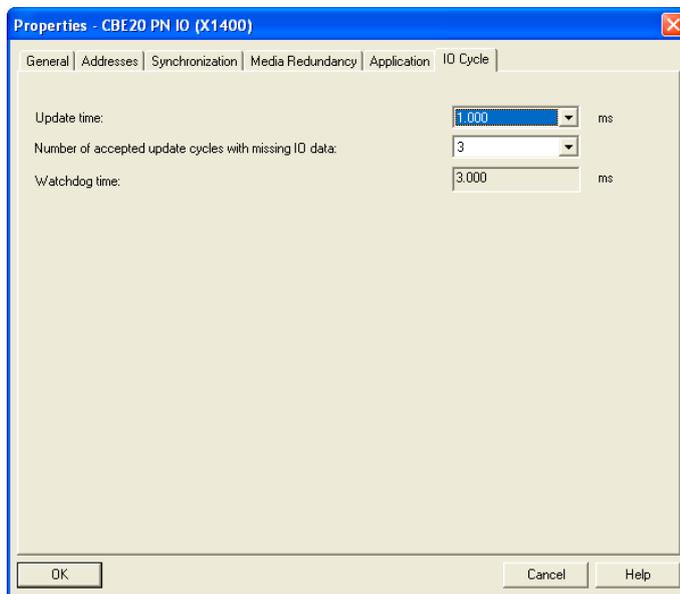
**Note 2) Scaling and isochronous applications**

With IRT High Performance, some IO devices support scalings 2, 4, 8, and 16, as well as scaling 1.

Where IO devices (e.g. ET200S IM151-3 PN HS, SINAMICS S) are operated with an isochronous application, it is usually only possible to set scaling 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.



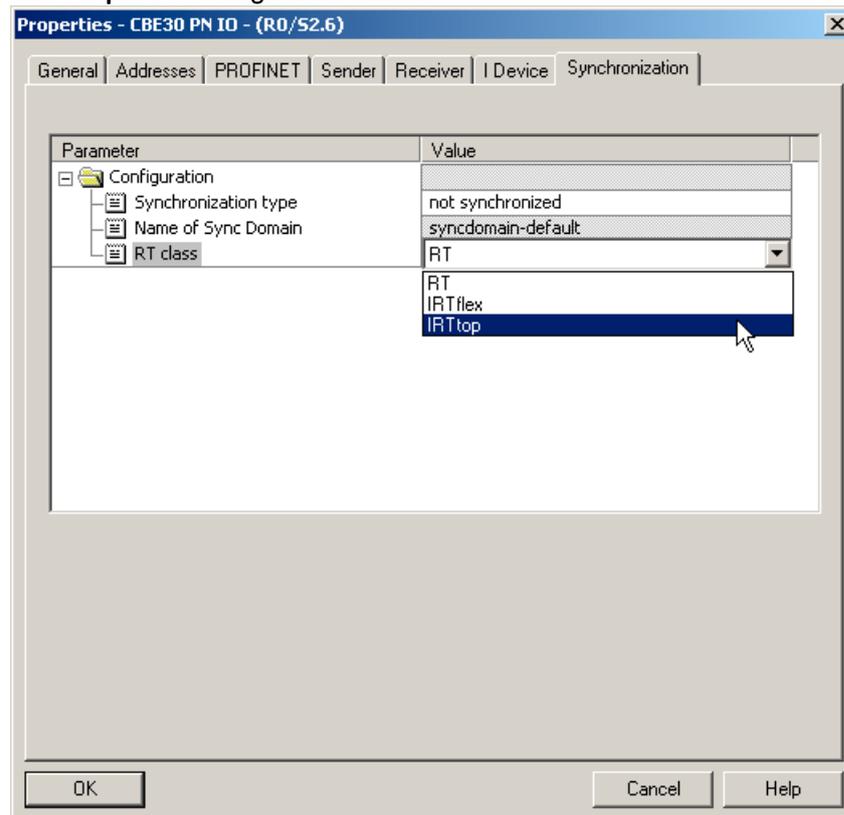
## Set RT class

The IO controller determines which RT class its IO system supports, by setting the real time class at its controller interface. If IRT High Performance is set, it is not possible to operate any devices with IRT High Flexibility on the IO controller, and vice versa. RT devices can always be operated, even if IRT classes are set.

You can set the RT class in the HW Config for the associated PROFINET device.

1. Double-click the PROFINET board entry in the module in HW Config.

The **Properties** dialog box is called.



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.

### 6.1.8.2 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 also uses standard Ethernet in the devices and commercially available industrial switches as infrastructure components. A special hardware support is not required.

#### Not isochronous

Because standard Ethernet does not support any synchronization mechanisms, isochronous operation is not possible with PROFINET IO with RT!

The realtime capability is comparable with the present PROFIBUS DP solutions with 12 MBaud, whereby a sufficiently large bandwidth portion is available for the parallel transmission of IT services on the same line.

PROFINET IO message frames have priority over IT message frames in accordance with IEEE802.1Q. This ensures the automation technology has the required real-time properties.

#### Data exchange

Communication is possibly only within a network (subnet).

#### Refresh time

The adjustable update time is in the range of 0.25 - 512 ms. The real refresh time depends on the bus load, the devices used and the quality structure of the I/O data. The refresh time is a multiple of the send clock.

### 6.1.8.3 PROFINET IO with IRT - Overview

#### Overview

PROFINET IO with IRT distinguishes itself through the separate time domains for IRT as well as RT and TCP/IP communication. This is guaranteed by highly precise hardware-supported cycle monitoring.

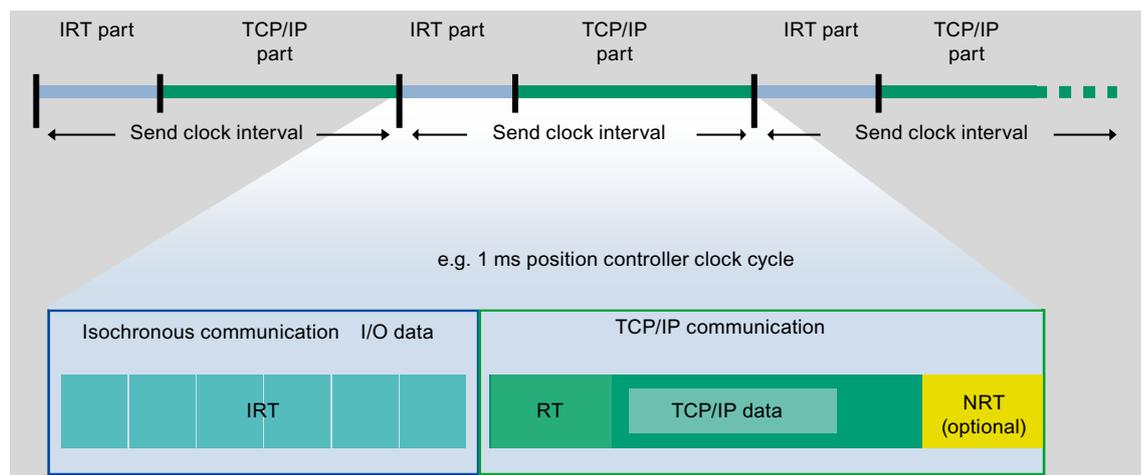


Figure 6-2 IRT Communication - Overview

For PROFINET IO with IRT, all devices must be synchronized on a shared sync master. The sum of all synchronized devices form a sync domain.

PROFINET IO with IRT is available in two versions:

- IRT High Flexibility (Page 128) with fixed bandwidth reservation.
- IRT High Performance (Page 129) with fixed bandwidth reservation and scheduled IRT communication

What is possible for PROFIBUS DP with equidistant bus cycles and isochronous operation is also possible for PROFINET IO with IRT High Performance.

In PROFIBUS DP, all nodes are synchronized in isochronous mode via a global control message frame generated by the DP master.

#### Note

Equidistance and isochronous mode is not possible for PROFINET IO with IRT High Flexibility.

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. The synchronization mechanisms are performed by the communication ASIC. This ensures a synchronization accuracy of less than one microsecond. Synchronization of all PROFINET devices with IRT High Performance on a common time base is a requirement for scheduled communication control, bandwidth reservation, and isochronous mode.

The sync master and sync slave device roles are assigned by the user during the configuration. The role of a sync master can currently only be assigned to a IO controller.

Sync master and sync slaves belong to a sync domain which is assigned a name via configuration. A sync domain has exactly one sync master.

A sync domain can consist of both PROFINET devices with IRT High Performance and PROFINET devices with IRT High Flexibility. There may also be PROFINET devices with RT at the ends of a line, but not between two PROFINET IO devices with IRT (High Flexibility or High Performance). otherwise scheduled communication is not possible.

**Compatibility**

Communication between and through different sync domains via PROFINET IO with RT is possible. If no topology has been configured, it is not necessary to observe a topology when assembling the devices. This is not the case for IRT, where devices must be connected one to another according to the configured topology.

A line containing PROFINET devices with IRT High Flexibility or IRT High Performance must not be interrupted by a PROFINET device with RT, as communication with IRT High Performance or IRT High Flexibility would be impossible.

**6.1.8.4 PROFINET IO with IRT (High Flexibility)**

**Description**

For PROFINET IO with IRT (High Flexibility), the largest IRT bandwidth requirement of a device plus a reserve for the complete network is reserved. It is not defined when which message frame in the IRT time window over which port will be transferred.

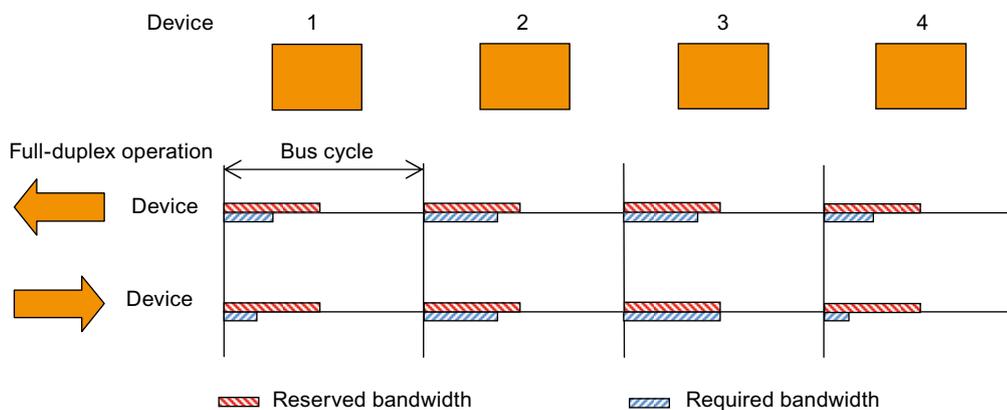


Figure 6-3 Overview of communication with IRT (High Flexibility)

**Send clock**

0.5, 1.0, 2.0 and 4.0 ms can be selected as send clock.

**Note**

Isochronous applications are supported with IRT (High Performance) only.

### 6.1.8.5 PROFINET IO with IRT (High Performance)

The performance capability of motion control applications is significantly increased with PROFINET IO IRT (High Performance). A hardware support enables a significant increase in performance compared with the present field bus solutions. By scheduling the message frame traffic for IRT High Performance, data traffic is considerably more optimized when compared with IRT High Flexibility.

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

All cyclic and acyclic data (IRT High Performance data) is transmitted within this time. Acyclic data is NRT data. The send clock of 250  $\mu$ s (P350) - 4 ms is the maximum range in which the send clock can be set. The actual send clock that can be set depends on various factors:

- Bus load
- Type of devices used
- 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  $\mu$ s grid within the limits of 250  $\mu$ s to 4 ms.

The supported send clocks can be found in the corresponding manuals of the respective SIMOTION devices. A minimum cycle time of 250  $\mu$ s is only supported by selected components (SIMOTION P350-3 and the fast ET 200S modules).

#### Isochronous application

In addition, a high-performance and isochronous connection to the application with low load on the application CPU is ensured. Isochronous data transfer with cycle times well below one millisecond and with a deviation in the cycle start (jitter) of less than a microsecond provide sufficient performance reserves for demanding motion control applications.

In contrast to standard Ethernet and PROFINET IO with RT, 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. The highest determinism quality is achieved through the scheduling of the transmission times which is especially advantageous for an isochronous application connection.

**Data exchange**

TCP/IP communication is generally also possible via network limits using routers. However, PROFINET IO with IRT High Performance and PROFINET IO with IRT High Flexibility only run within a sync domain.

**6.1.9 Topology**

**General**

Below, you will find an overview of various options for setting up a PROFINET network with SIMOTION.

Table 6-3 Possible topology for SIMOTION

Topology	
Star	<p>If you connect communication nodes to a switch, you automatically create a star-shaped network topology.</p> <p>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 part of the communication network.</p>
Tree	<p>If you interconnect several star-shaped structures, you obtain a tree network topology.</p>
Linear	<p>All the communication nodes are connected in series as a bus.</p> <p>If a coupling element (e.g. switch) fails, communication downstream of the failed coupling element is no longer possible. The network is then divided into two subsegments.</p> <p>In PROFINET, the linear topology is implemented by switches that are already integrated in the PROFINET devices. Therefore, the linear topology in PROFINET is simply a special form of the tree-/star-shaped structure.</p> <p>Linear network structures require the least amount of cabling.</p>

**Production topology examples**

The following example shows various topologies combined.

### General information on optimization

PROFINET allows you to set up communication with both high-performance and a high degree of uniformity. By keeping to the following guidelines, you can improve performance even further.

1. Connect a router or a SCALANCE S between the office network and PROFINET system. Use the router to define access privileges for your PROFINET system.
2. Where possible, set up your PROFINET system star-shaped (e.g. in the control cabinet).
3. Keep the interconnection depth of the switches as low as possible. This increases clarity of your PROFINET system architecture.
4. Connect your PG/PC close to the communication partner (e.g.: PG and communication partner connected to the same switch).

### Sample topology - company network - production network

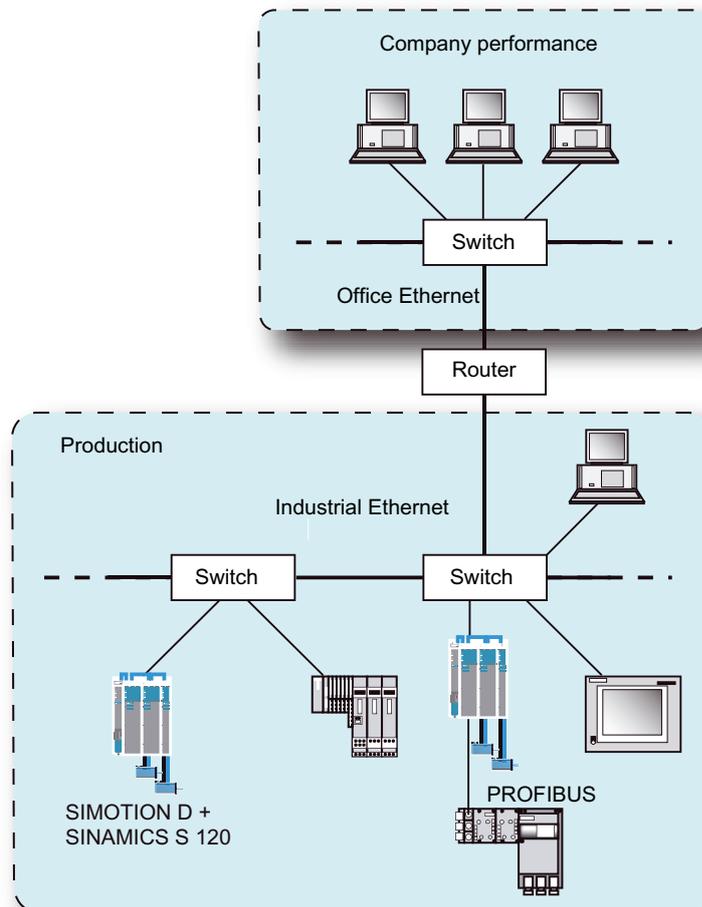


Figure 6-4 Optimized Topology

---

**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. Of importance are also the delay times through the transmission on the Ethernet cables between the switches and the delay times in the switches. The cable propagation times must be taken into account in the topology configuration.

Since no congestion situations can occur in the switches because of the communication scheduling, the retention time in PROFINET devices with IRT High Performance is constant. This is a device characteristic which is described in the GSD file. 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.

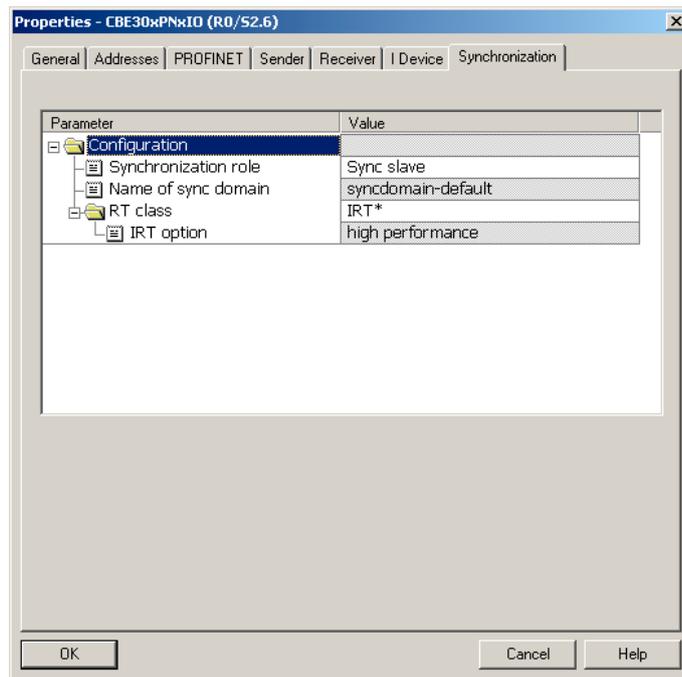
### 6.1.10 Isochronous applications with PROFINET

As with PROFIBUS, the application can also be synchronized to the transmission network's cycle clock with PROFINET.

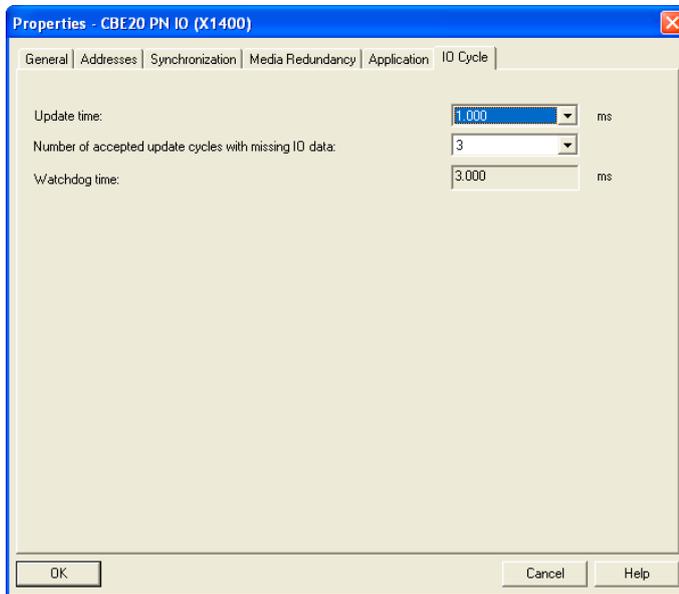
#### Proceed as follows

When configuring isochronous applications, proceed as follows:

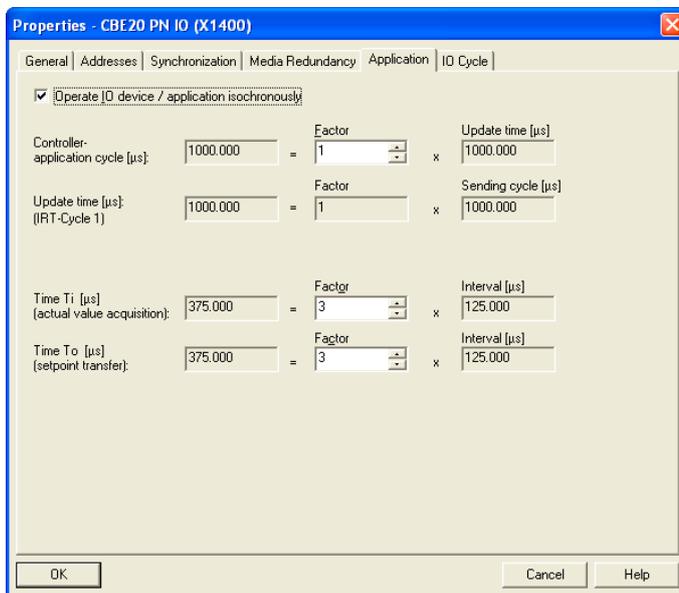
1. Set "IRT High Performance" on the controller and devices.



2. Set the refresh time mode on the devices to a fixed factor.



3. Activate the "Operate IO device/application in isochronous mode" checkbox on the devices.



## 6.1.11 Cycle clock scaling

### 6.1.11.1 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 of the data transferred by 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

Equidistance and 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 159).

#### 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 (e.g. a SINAMICS S120) runs scaled to a send clock in the servo cycle clock, the servo cycle clock can extend across n send clocks (n = scaling of the servo to the send clock), and must not be completely processed within the send clock time.

#### The following general conditions apply to cycle clocks and cycle clock scalings for a SIMOTION controller

- If IRT High Performance data is configured for a SIMOTION device, the PROFINET interface is always the cycle clock source in the system, which means the servo cycle clock is synchronized with the send clock. If IRT High Performance is configured for the SIMOTION device, but no cyclic data is created, the servo cycle clock runs asynchronously in relation to 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

### 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.

#### 6.1.11.2 Cycle clock scaling for IO accesses

##### Description

The following must be observed for cycle clock scaling (PROFINET and PROFIBUS):

- At the end of a IPOSynchronousTask, the process image is output with the next possible servo (Data Out) (= response-time-optimized). In case the servo clock cycle is scaled to the IPO clock cycle, this can lead to the data being output one/several servo clock cycles earlier or later within an IPO clock cycle, if the IO accesses are performed via the IPOSynchronousTask.
- At the end of the servo priority class, the process image of the ServoSynchronousTask is output with the next possible bus clock cycle (= response-time-optimized).
- For PROFINET and with PROFINET cycle clock scaled to the servo cycle clock, this can lead to the data to be output one/several bus clock cycles earlier or later within the servo clock cycle, if the IO accesses are performed via the ServoSynchronousTask and the runtime of the servo priority class fluctuates over one bus clock cycle.
- 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

### 6.1.11.3 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		0.5 ms	4.0 ms	0.5 ms
C240 DP	1 ms			
D410	1 ms	0.5 ms	4.0 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	1 ms	0.5 ms	4.0 ms	0.5 ms
P350-3	1 ms	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

### 6.1.12 Connection between sync domain and IO systems

It is important that sync domains do not need to be restricted to a single PROFINET IO system. 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.

### 6.1.13 Redundant sync master

#### Description

You can use two sync masters for isochronous operation with IRT High Performance in a sync domain (primary and secondary sync master). At any one time, only one sync master can be used for the synchronization of the sync slaves.

The switch is performed as soon as the primary sync master fails. If both sync masters are operational, the sync master configured as primary handles the synchronization of the sync slaves.

#### Limitations of use

If the transmission link between the primary and secondary sync master fails, so that there are 2 subnets with one sync master each, both subnets remain synchronized with the corresponding remaining sync master. As a result, there are two independent synchronized subnets that drift apart due to the temperature drift of the quartzes. Once the data transmission link has been reestablished, no smooth switching to the primary sync master is possible, i.e. the drives in the network would lose the synchronization and fail for a short time, since a renewed synchronization is necessary due to the temperature drift.

#### 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.

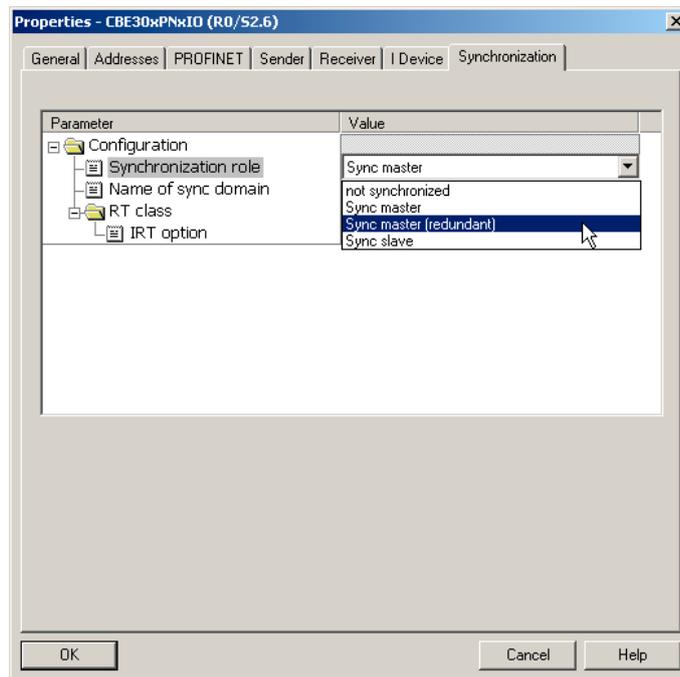


Figure 6-5 Configuring the second sync master

### 6.1.14 Quantity structures

The following maximum values apply for IO controllers of the SIMOTION platform (SIMOTION D and SIMOTION P):

- Connection of up to 64 IO devices.
- Up to 64 controller-controller data exchange broadcast relationships may be set up between IO controllers

#### 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:

$$(\text{RT/IRT High Flexibility IO device}) + (\text{IRT High Performance IO device}) \leq 64$$

and at the same time

$$(\text{IRT High Performance IO device}) + \text{data exchange broadcast frame consumer} \leq 64$$

---

**Note**

In a data exchange broadcast relationship, it is not the number of lines in the lug receiver (see Configuring the receiver (Page 168)) that is intended for IRT High Performance data exchange broadcast configuration, but rather the number of Ethernet frames received for the data exchange broadcast.

Each provider sends their data exchange broadcast data in an Ethernet frame. Any other SIMOTION can read this data in this frame. This means there is a counting connection to each transmitting SIMOTION.

---

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 diagnostics data

**6.1.15 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 34).

## 6.2 Specific properties of PROFINET IO with SIMOTION

### 6.2.1 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.

PROFINET devices support the simultaneous operation of:

- IRT High Performance or IRT High Flexibility - Isochronous Realtime Ethernet
  - Operation of IRT peripherals (e.g. ET200S)
  - Operation of a SINAMICS S120 as an IO device
- RT - realtime Ethernet
  - Operation of RT - peripherals (e.g. ET 200S, ET 200pro)
  - 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 form a sync domain. In other words, there must not be any non-IRT (High Performance or High Flexibility) PROFINET devices on the transmission link between the PROFINET devices with IRT (High Performance or High Flexibility).

---

#### Note

With SIMOTION SCOUT, it is possible to access a maximum of 10 PROFINET nodes ONLINE simultaneously.

---

## 6.3 Configuring PROFINET IO with SIMOTION

### 6.3.1 New to SIMOTION V4.1.2

#### Description

New hardware, in which a new synchronization process has been implemented, is being introduced into SIMOTION V4.1.2. The synchronization process used up until this point is, however, still supported. The old synchronization process will continue to be used as standard. This is offered as an option in HW Config with **IRT\***.

If you wish to work with the new synchronization process, you must first upgrade the SIMOTION firmware. The relevant data set can be found on the Utilities CD.

The new synchronization process offers both IRT versions (IRT High Performance and IRT High Flexibility), with the old IRT\* continuing to be supported.

### 6.3.2 Proceed as follows for configuring PROFINET IO

#### Procedure

To configure PROFINET IO you must perform the following steps:

1. Insert the SIMOTION module.
  - SIMOTION P350 PN, SIMOTION D410 PN and C240 PN can be selected and inserted.
  - For SIMOTION D4x5, you must drag the CBE30 module from the hardware catalog in HW Config and drop it onto the corresponding interface of the SIMOTION module.
2. Insert IO devices: Insert IO devices from the hardware catalog in HW Config into the I/O system.
3. Create the topology: Specify the topology, i.e. how the individual ports of the PROFINET IO devices are interconnected one to another.
4. Configure the sync domain: Define which PROFINET IO nodes are sync master (clock generator) or sync slave.
5. Specify the send clock: Describes the time within which a PROFINET IO device exchanges the user data with the PROFINET IO controller.
6. Controller-controller cross-traffic: Specify which address areas are to be used for sending and receiving, respectively.

### 6.3.3 Adding and configuring a CBE30-PROFINET board

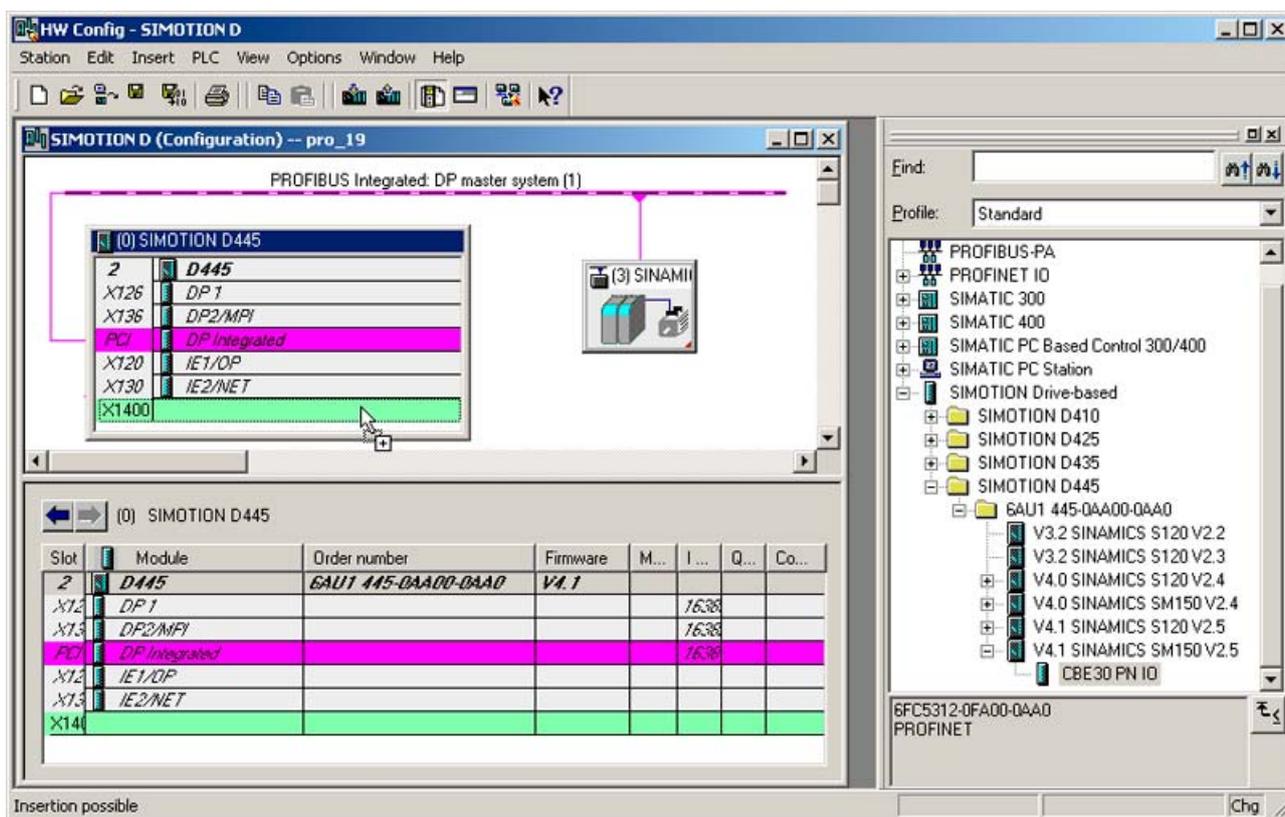
#### Requirement

You have created a project and have already inserted a SIMOTION device.

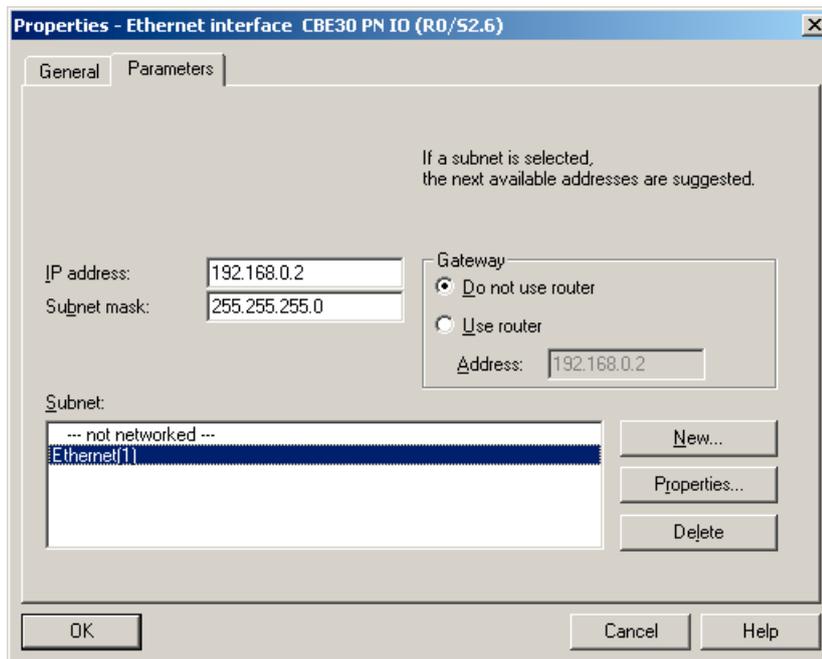
A PROFINET interface does not have a valid IP address when delivered. Therefore, you must assign an IP address (node initialization) and load it into the module.

#### Proceed as follows

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.



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. The subnet ID is preselected.
6. Click **OK** to confirm these entries. A new Ethernet subnet is created, e.g. Ethernet(1).



7. Select the subnet.
8. Assign the desired IP address.
9. Accept the settings by clicking **OK**.

## 6.3.4 Inserting and configuring P350

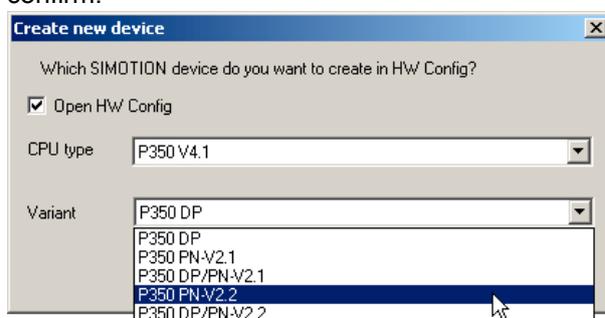
### Requirement

You have already created a project and now want to insert a P350 with PROFINET.

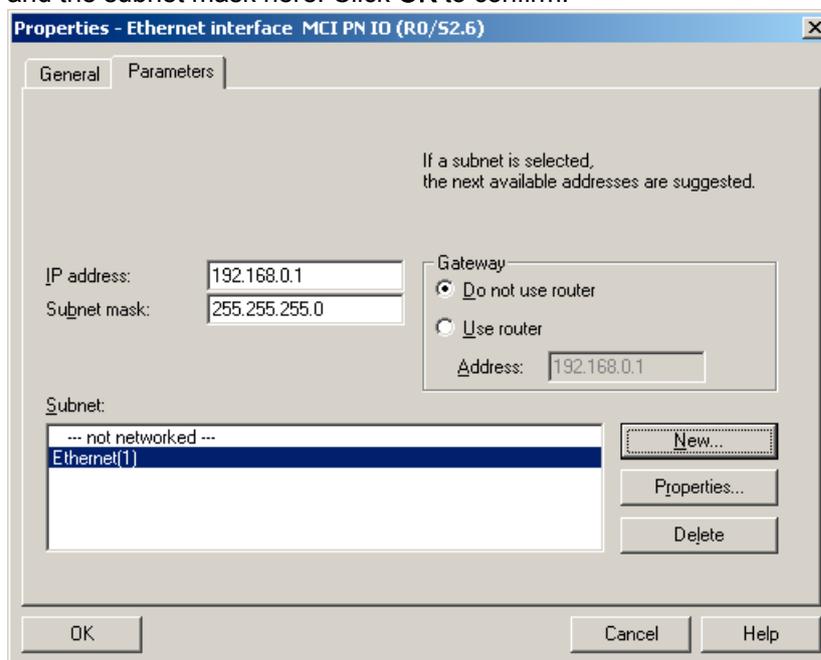
A PROFINET interface does not have a valid IP address when delivered. Therefore, you must assign an IP address (node initialization) and load it into the module.

### Proceed as follows

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.



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.



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.

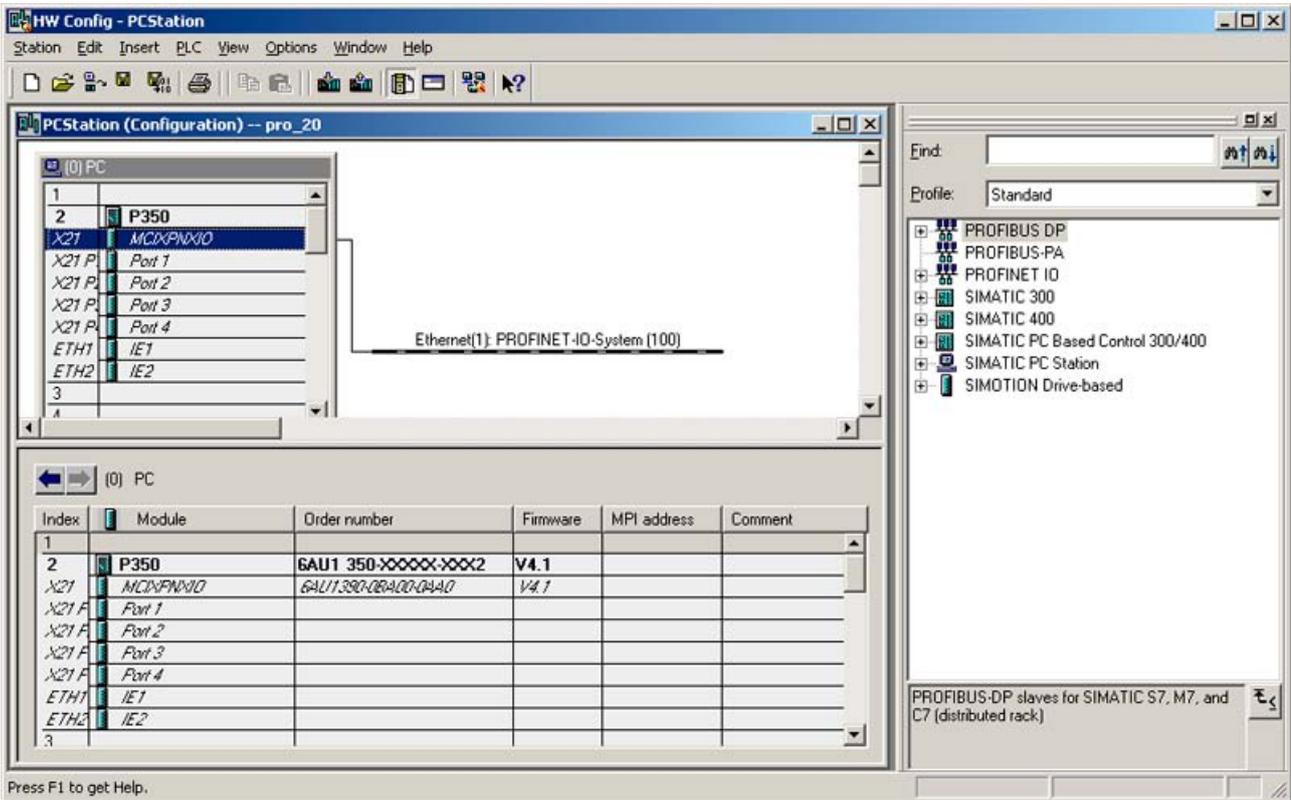


Figure 6-6 HW Config with PROFINET for P350

### 6.3.5 Inserting and configuring the C240

#### Requirement

You have already created a project and now want to insert a C240 with PROFINET.

A PROFINET interface does not have a valid IP address when delivered. Therefore, you must assign an IP address (node initialization) and load it into the module.

#### 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.

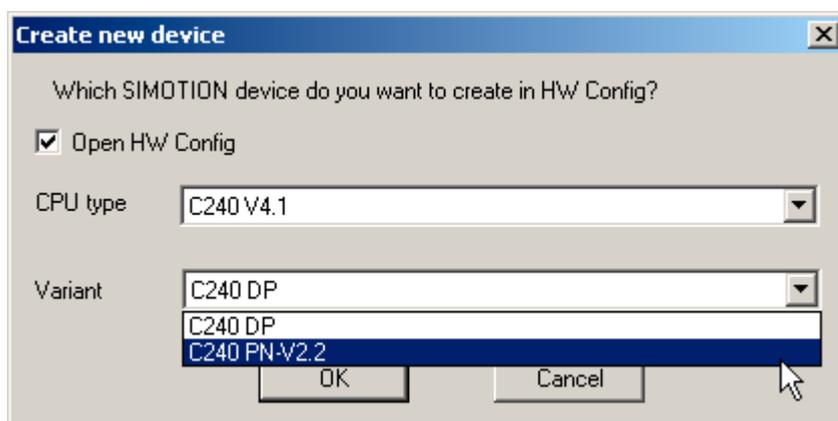


Figure 6-7 Creating a new C240 PN device

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.

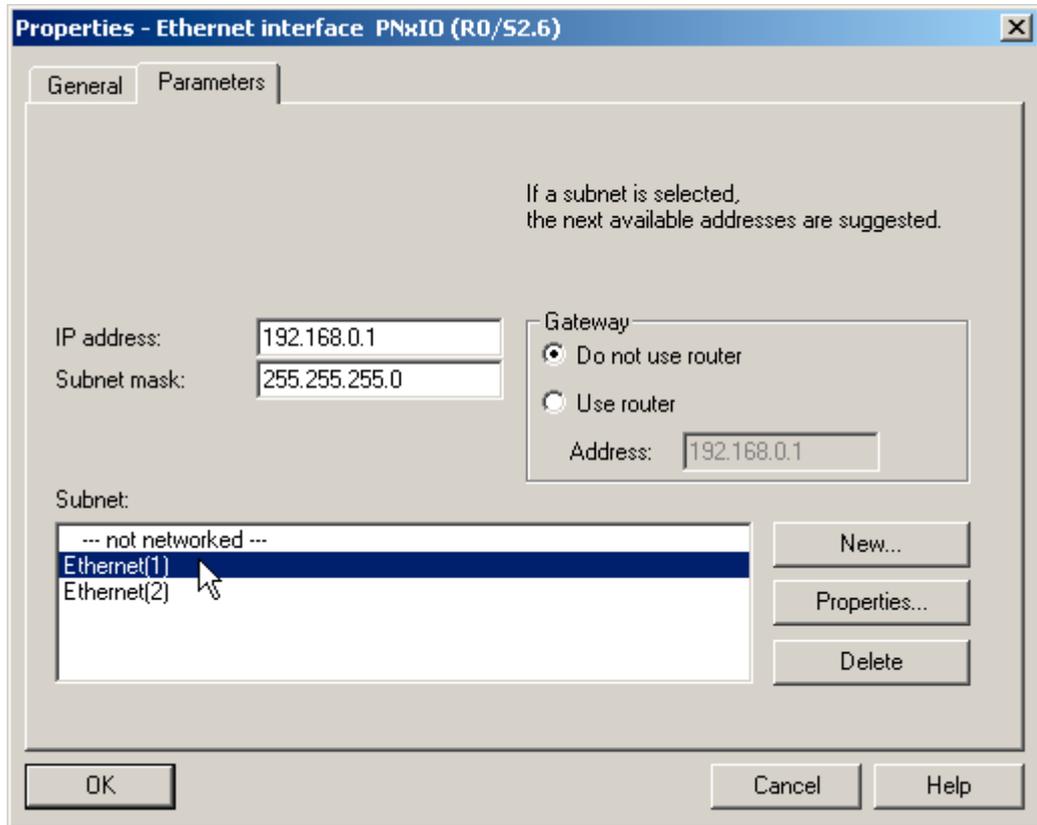


Figure 6-8 Creating a new Ethernet for C240 PN

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.

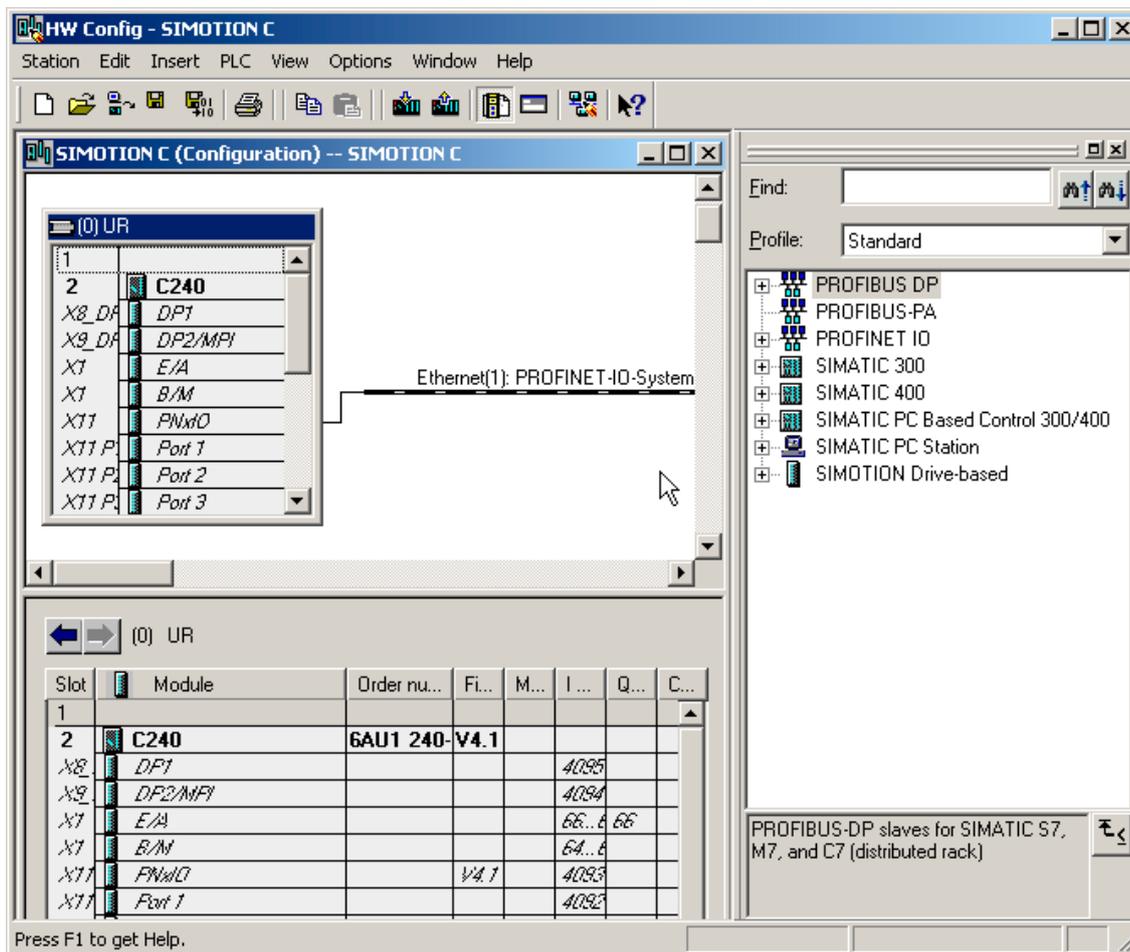


Figure 6-9 HW Config with PROFINET for C240 PN

### 6.3.6 Creating a sync domain

A sync domain is a group of PROFINET devices synchronized to a common cycle clock. Exactly one device has the role of the sync master (clock generator), all other devices have the role of a sync slave.

---

#### Note

All components that exchange data via IRT must belong to a single sync domain. Operation steps 3 and 4 are skipped when the preset sync domain is kept.

---

**Proceed as follows**

1. In HW Config, open the station with PROFINET devices which are to take part in the IRT communication.
2. Select the **Edit > PROFINET IO > Domain Management** menu command. A dialog tab with the list of all devices supporting synchronization is opened. A default sync domain is created and the devices are already assigned.
3. Click **New** to create a new sync domain.
4. Click Add to assign the device concerned to the sync domain.
5. 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.

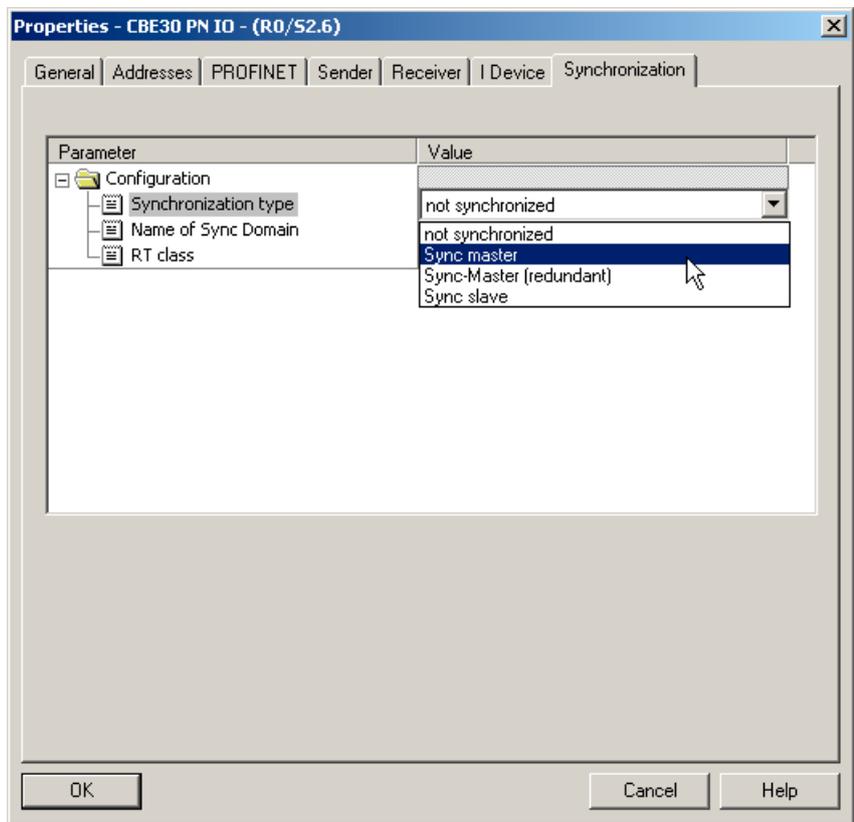


Figure 6-10 Selecting synchronization

6. Set the synchronization type to sync master.
7. Confirm the settings with **OK**.
8. Then, select all devices, first in the upper field then in the lower field, which are to be configured as sync slaves (keep the Ctrl key pressed and select the devices one after the other).
9. Then, click on the **Properties device** button.
10. Set the synchronization type to Sync slave in the dialog box.
11. 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.

---

**Note**

A second sync master can be inserted for media redundancy (secondary sync master).

---

## 6.3.7 Configuring a topology

### 6.3.7.1 Topology

#### Introduction

Topology configuration is a requirement for communication scheduling of IRT High Performance.

---

**Note**

You must only interconnect IRT High Performance devices via ports. Devices operated with IRT High Flexibility or RT must not be connected in this way. A topology can be configured for IRT High Flexibility: when present, this is also monitored. However, this is not a requirement for operating IRT High Flexibility.

---

There are two options for defining the properties of the cables between the ports of the switches:

Using the topology editor (Page 152)

Using the object properties (Page 154)

6.3.7.2 Interconnecting ports via the topology editor

Proceed as follows

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).

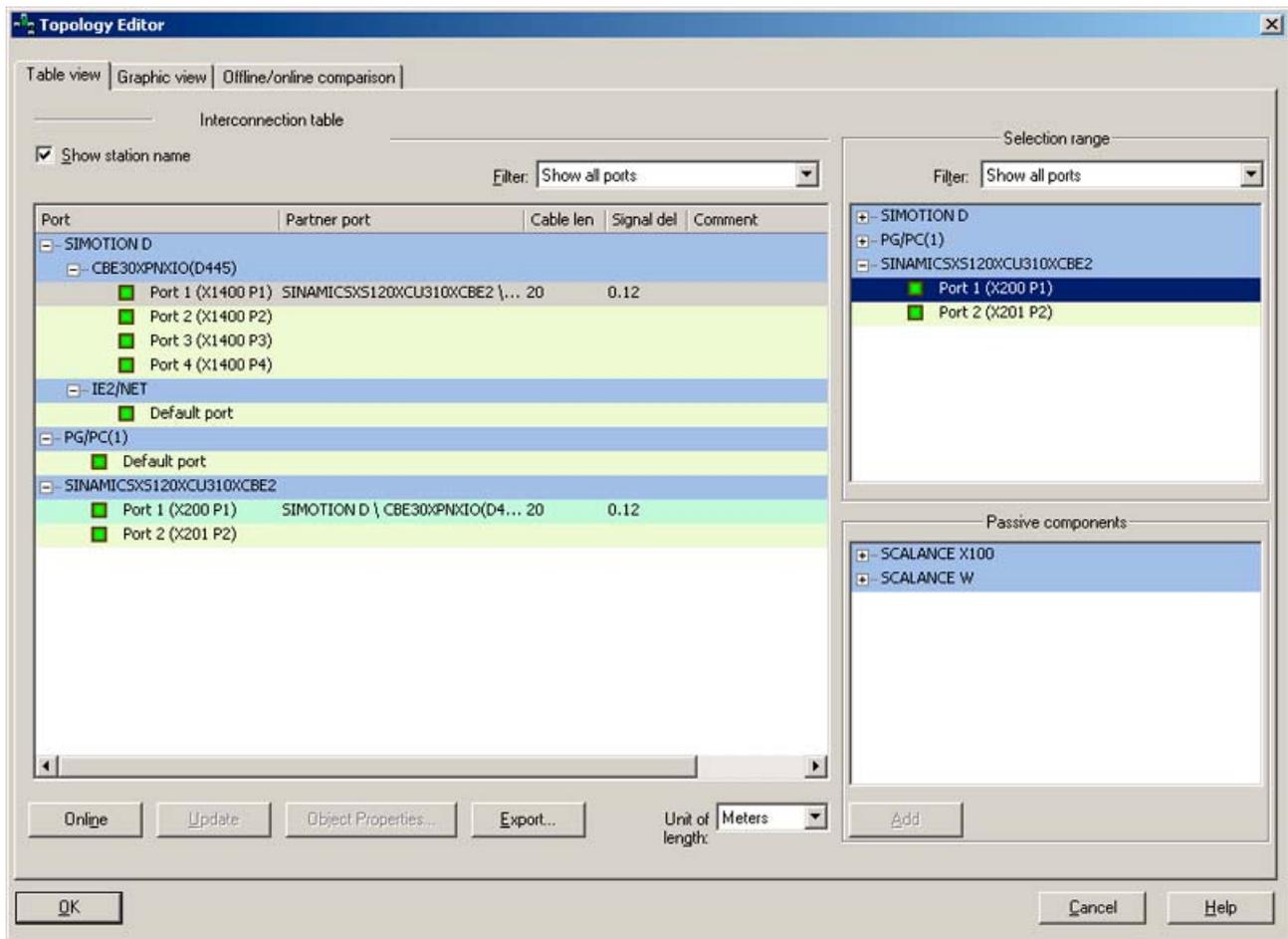


Figure 6-11 Topology editor

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.

To interconnect ports of different devices, select the port of a device that you want to interconnect in the right-hand field. Drag this port to the desired port of a device in the interconnection table. The following dialog box then opens:

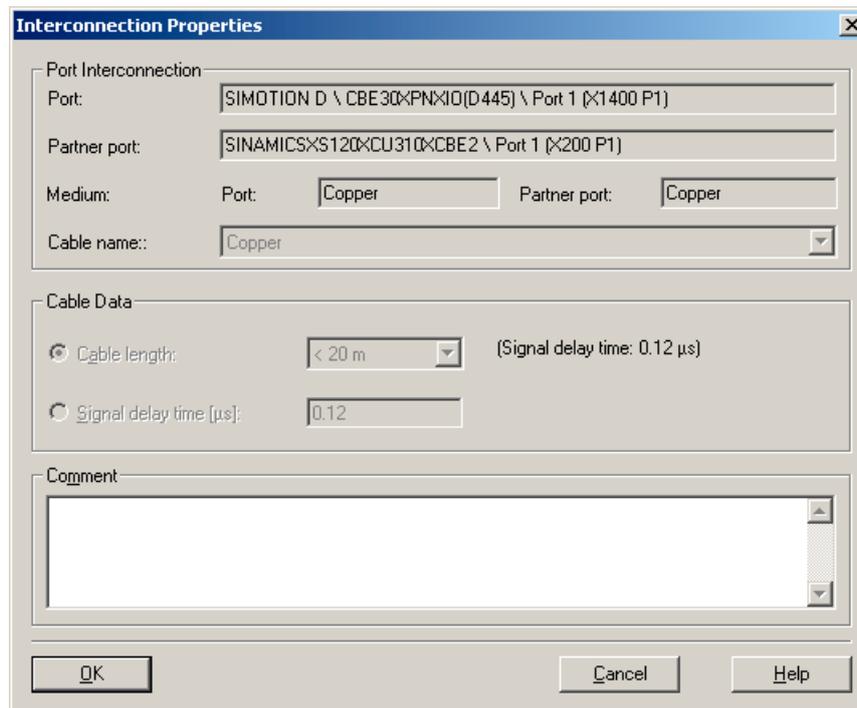


Figure 6-12 Properties - Topology

The port interconnection is displayed: You can configure the cable data: A cable length < 20 m is set by default. Alternatively, you can configure the signal propagation delay:

- Confirm your entries with **OK**.

### 6.3.7.3 Interconnecting ports via object properties

Alternatively, a partner port can be selected via the properties of a port. Thus, the cable between two ports is defined and the properties of this cable can be edited.

#### Proceed as follows

1. The dialog box is opened in HW Config by selecting a port on the module and selecting the **Edit > Object properties** menu command or double-clicking on the port.

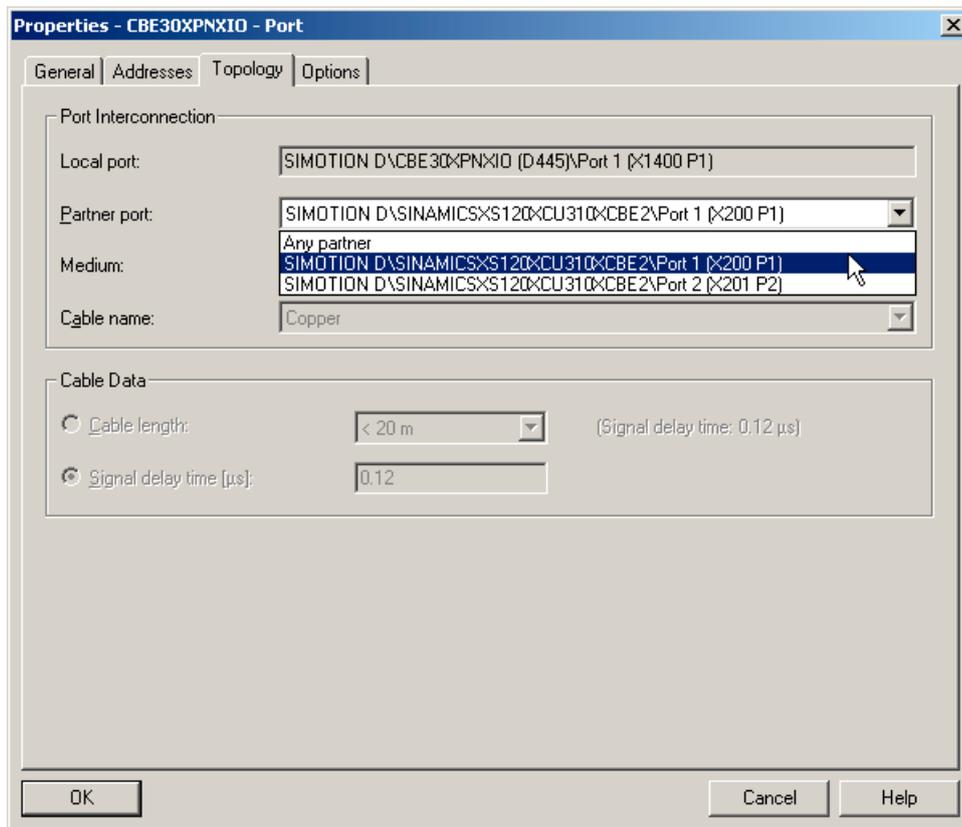


Figure 6-13 Object properties Topology

2. Then, select the **Topology** tab in the **Properties port...** dialog.
3. In the **Partner port** list, select the port with which you want to interconnect the device.
4. Confirm your entries with **OK**.

### 6.3.7.4 Topology editor (graphical view)

#### Description

In addition to the tabular view of the topology, you can also use a graphical view (from STEP7 V5.4 SP2). Some of the tasks you can perform in this view:

- Interconnect ports
- Modify the properties of the interconnection
- Add passive components

#### Proceed as follows

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.

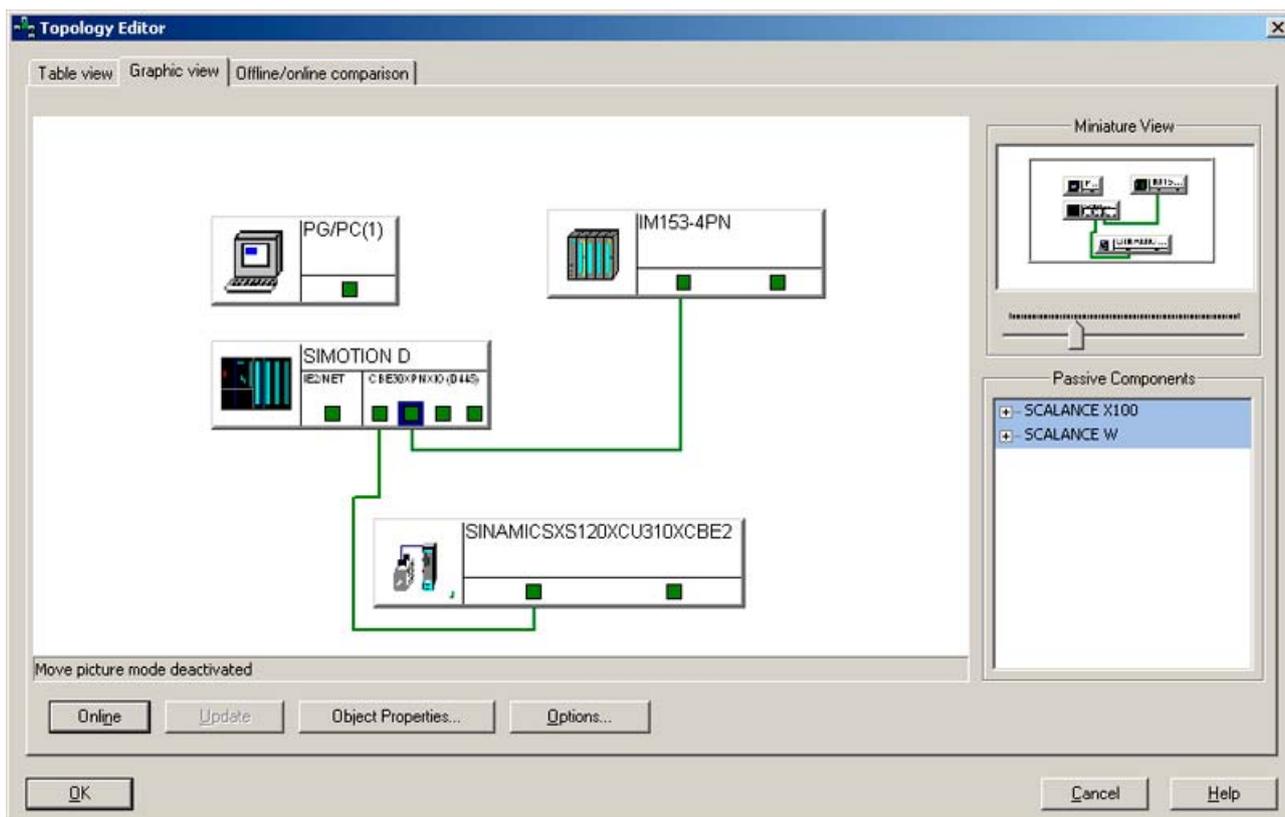


Figure 6-14 Topology editor (graphical view)

### 6.3.8 Defining send clock and refresh times

The refresh time applies to all devices that participate not synchronized (with RT class: RT) in the PROFINET IO system.

The send clock defines the period between two consecutive intervals for IRT (High Performance and High Flexibility) or RT communication. This applies to all devices in a sync domain.

#### How to set the send clock

1. In HW Config, open the **Domain Management** dialog box.

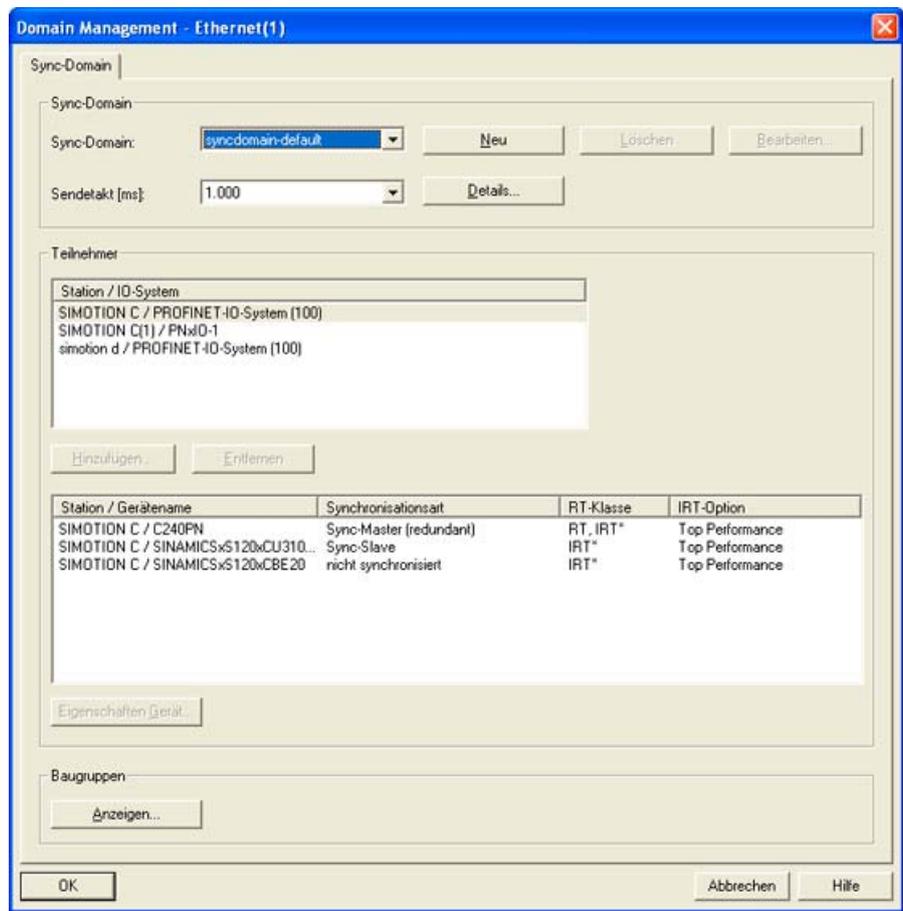


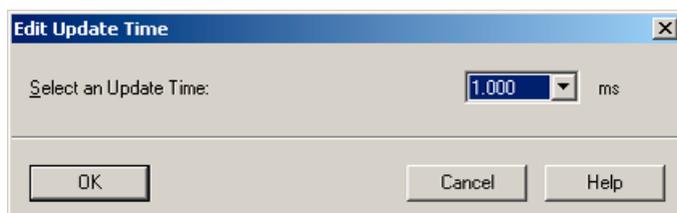
Figure 6-15 Domain Management

2. Select a suitable send clock. The transmission cycle clock is the smallest possible transmission interval. The send clock is preset to 1 ms.

### Defining update times for PROFINET IO using PROFINET devices with RT

Update times for IO data exchange of PROFINET IO using PROFINET devices with RT are set in the **Properties PROFINET IO System** dialog box.

1. Select the device for which you want to set refresh times.
2. Click **Edit**. You can select the refresh time in the **Edit refresh time** dialog.



3. Confirm the settings with **OK**.

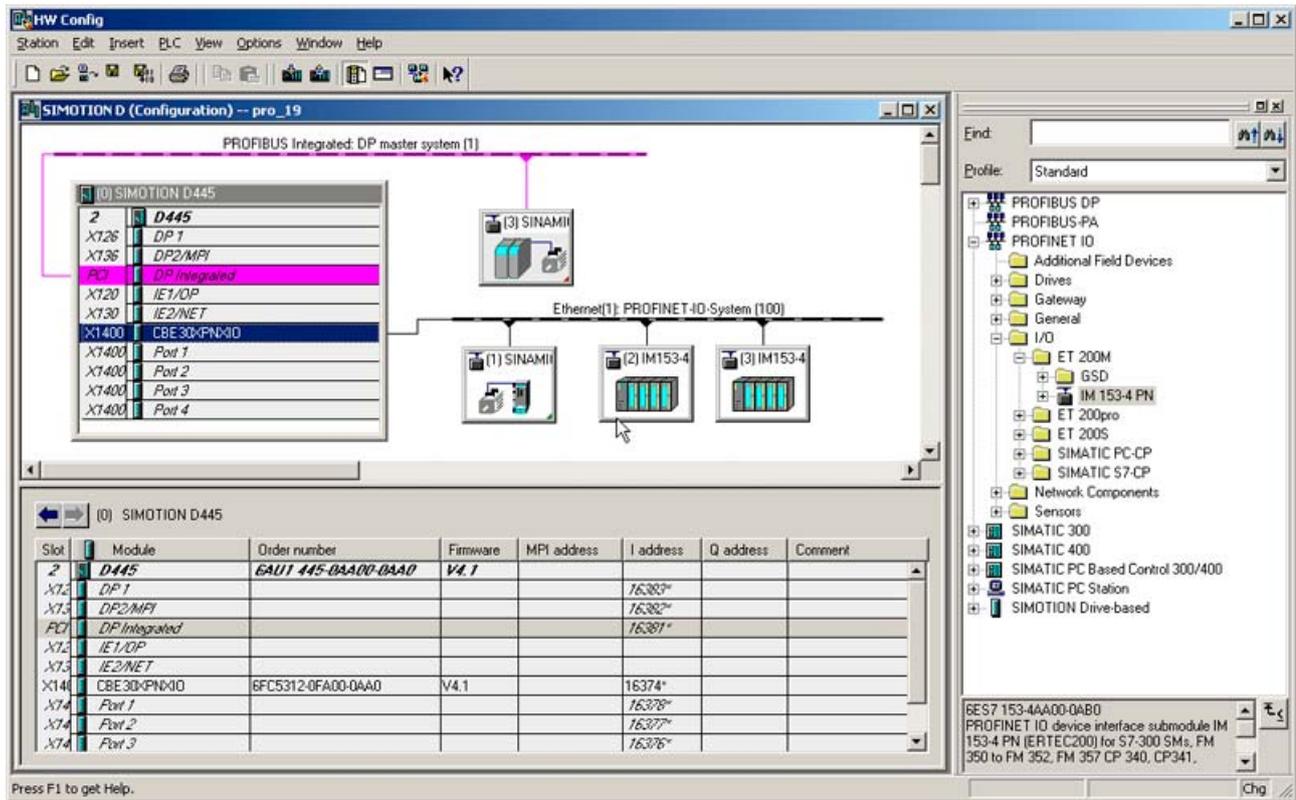
### 6.3.9 Creating an IO device

#### Requirement

You have already created a PROFINET IO system and configured a PROFINET IO module, e.g. SIMOTION D445 with CBE30-PN.

#### Procedure for PROFINET IO devices using the hardware catalog

1. Double-click the corresponding module 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.



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.

### 6.3.10 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.

#### Note

Please note that for each parameter in the SIMOTION device in use, a setting must be made specifying whether PROFINET V2.1 or PROFINET V2.2 is being used.

#### Proceed as follows

1. Select the module from the PROFINET IO > Drives > SINAMICS entry in the hardware catalog, e.g. SINAMICS-S120-CBE20 (not GSD).
2. Click the entry of the PROFINET module SINAMICS S120 CBE20.
3. Drag the drive 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.
4. Click **OK** to accept the settings.  
The dialog box **Properties SINAMICS** is displayed.
5. Select the device version (firmware version).

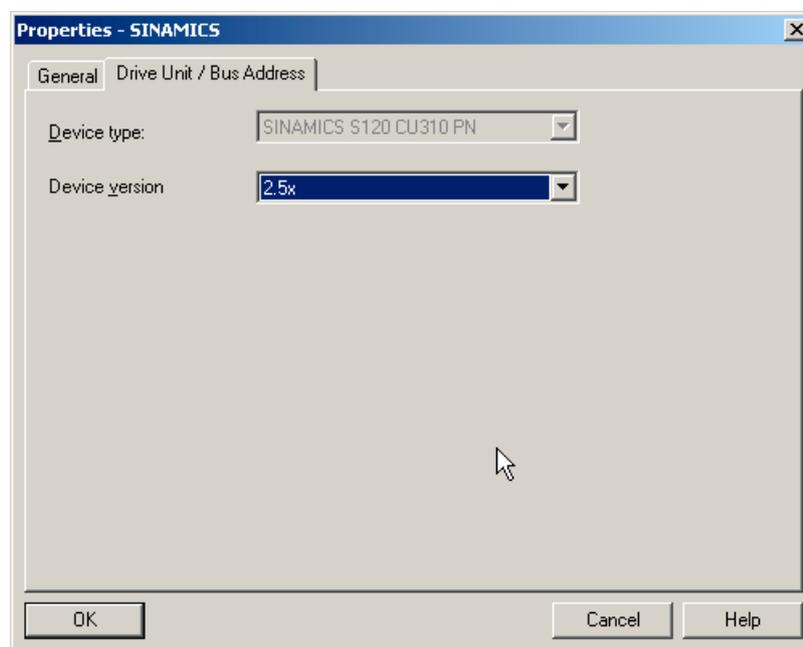


Figure 6-16 SINAMICS properties

6. Click **OK** to confirm these entries.

### Set message frame

If you want to use isochronous communication, you must configure a message frame, with which this is possible (e.g. message frame 105), as otherwise, for example, input fields Ti or To in the **Properties SINAMICS CBE20 PN-IO** dialog box will not be active or empty.

In other words, if you want to configure in another message frame, proceed with the following steps:

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 in the project navigator under **<"Drive\_device\_xx"> - Configuration**. 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 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.

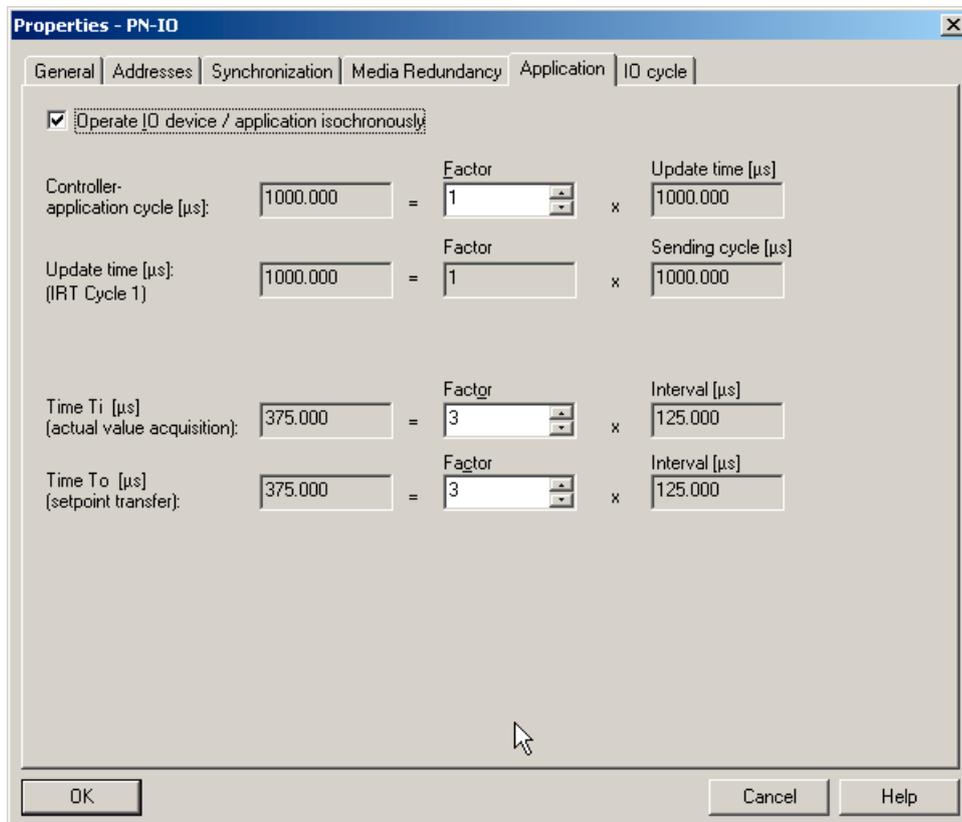


Figure 6-17 Properties SINAMICS CBE20 PN-IO

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.
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**

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 161)), and interconnect the ports (see Interconnecting ports via the topology editor (Page 152) ).

---

**See also**

Creating a sync domain (Page 149)

### 6.3.11 Assigning device names and IP addresses to IO devices

**Requirement**

A PG/PC is directly connected to the PROFINET device.

**Introduction**

For it to be possible to access an IO device ONLINE (e.g. a SINAMICS S120) the devices must first be assigned an IP address. This is done via the so-called node initialization or IP address assignment ONLINE on the device.

---

**Note**

If you connect the PG/PC directly to the device's PROFINET interface, you do not require a crossover cable.

---

As IP address, use the address you specified in the **Properties - Ethernet Interface ...** dialog box (can be opened by double-clicking the device). The default setting **Assign IP address through Controller** is active. Moreover, a default name is entered that you can modify.

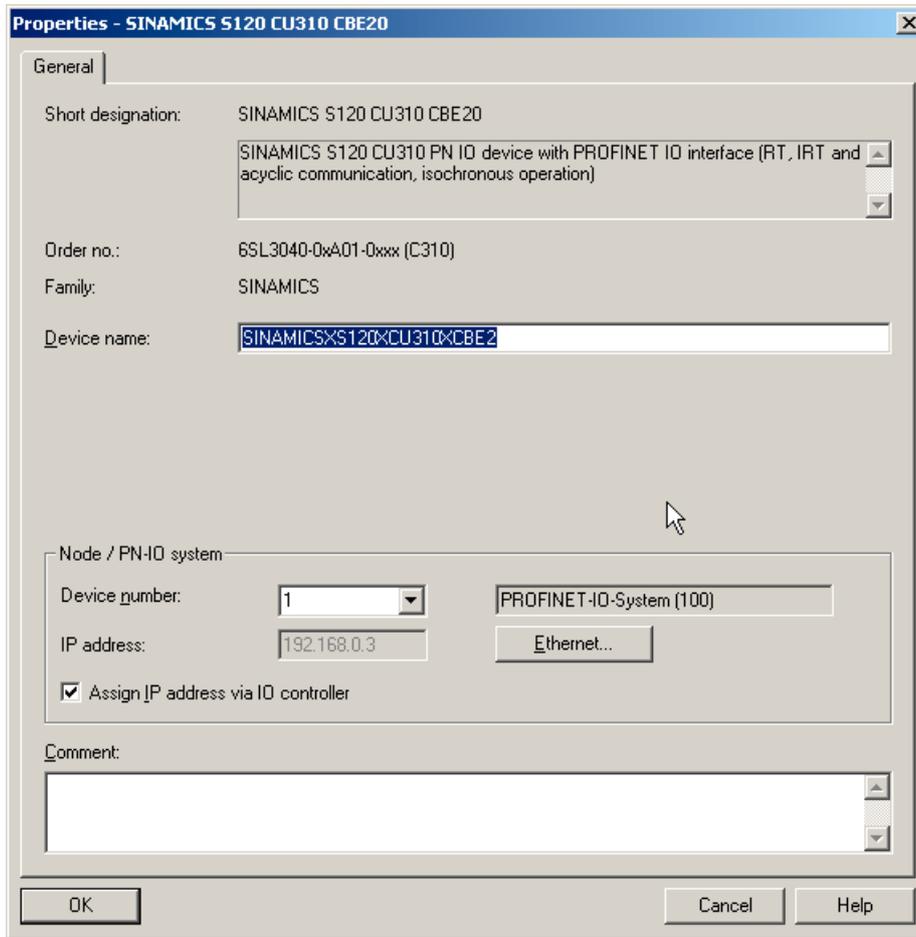


Figure 6-18 Properties SINAMICS S120

**Proceed as follows**

1. In HW Config, select the **Target system - Ethernet - Edit Ethernet node** menu item. The **Edit Ethernet node** dialog box is displayed.

**Edit Ethernet Node**

Ethernet node

MAC address:   Nodes accessible online

Set IP configuration

Use IP parameters

IP address:  Gateway

Subnet mask:   Do not use router

Use router

Address:

Obtain IP address from a DHCP server

Identified by

Client ID  MAC address  Device name

Client ID:

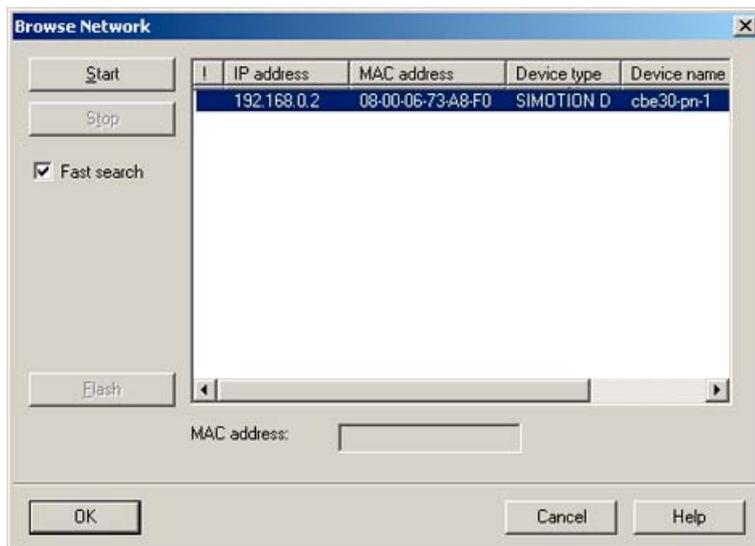
Assign device name

Device name:

Reset to factory settings

2. Click the **Browse** button.

3. The **Browse Network** dialog box opens. The connected nodes are displayed.



4. To make sure that the selected node is the correct node, click the Flash button. The LED then flashes on the selected device.
5. Click the device to be initialized and confirm with **OK**.
6. Enter the address you specified in the **Properties – Ethernet Interface ...** dialog box.
7. Select **No router used** as the router.
8. Click the **Assign IP configuration** button. The IP address is then assigned to the device online.
9. Enter the device name that you have defined in HW Config, see figure **Properties SINAMICS S120**.
10. 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.

The screenshot shows a dialog box titled "Edit Ethernet Node". It contains the following fields and controls:

- Node:**
  - MAC address: 08-00-06-73-c5-19
  - Device: mci-pn-b2
  - Assign name button
- IP Configuration:**
  - IP address: 192 . 168 . 1 . 52
  - Subnet screen: 255 . 255 . 255 . 0
  - Gateway:  Do not use router,  Use router
  - Address: [Field with dotted pattern]
  - Assign IP configuration button
- Close button
- Help button

Figure 6-19 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.

## 6.4 Configuring direct data exchange between IO controllers

### 6.4.1 Introduction

I/O data areas can be exchanged cyclically between two or more SIMOTION controllers.

For the data exchange between SIMOTION controllers, the devices must be located in a common sync domain and configured accordingly as sync master and sync slaves. Data exchange is performed using IRT High Performance communication.

---

#### **Note**

This function is not available for SIMATIC CPUs.

---

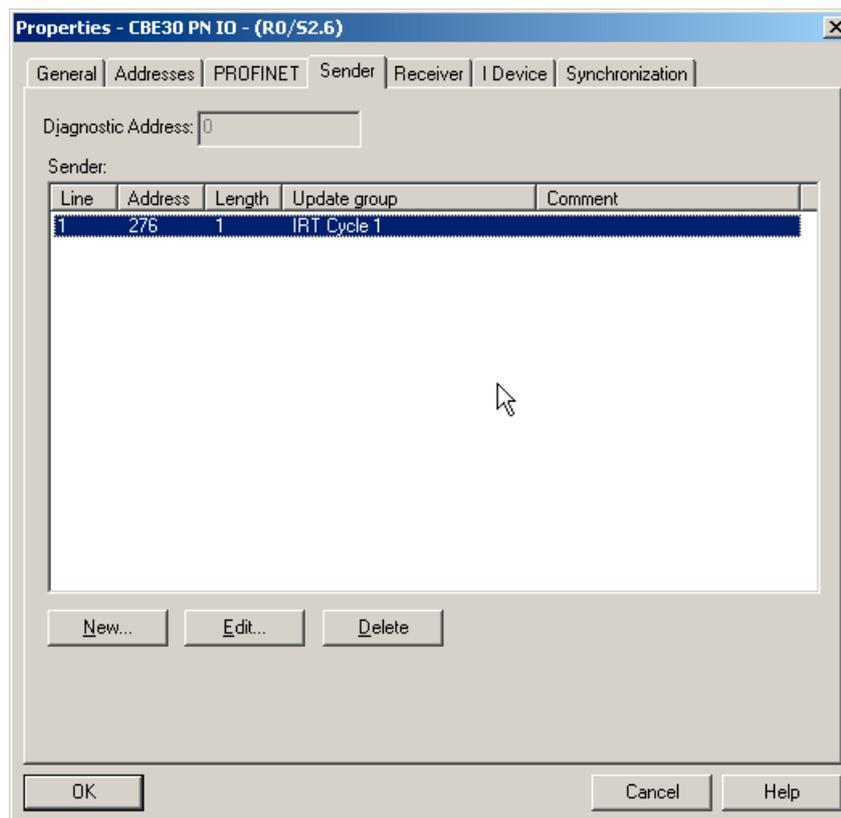
### Recommendation

We recommend, initially configure the send areas for all PROFINET devices and then the receive areas.

## 6.4.2 Configuring the sender

### Proceed as follows

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.

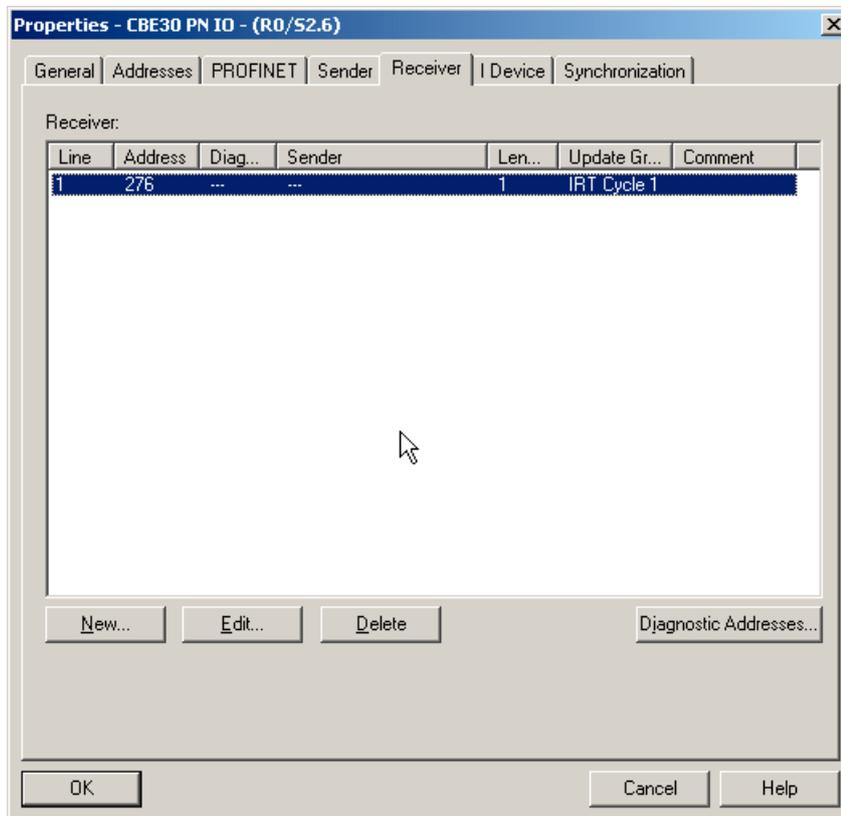


3. Click the **New** button.
4. Enter in the Properties dialog of the sender, the start address from the I/O area and the length of the address area to be used for sending. Comment the data area so that you will be able to identify the data transmitted via this area later on.
5. Confirm the settings 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.

### 6.4.3 Configuring the receiver

Proceed as follows



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 should 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**.

## 6.5 Configuring the iDevice

### 6.5.1 PROFINET iDevice

#### iDevice description

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

iDevices can, for example, be used in the process communication between SIMATIC and SIMOTION devices or in modular machines.

For example, iDevices can be used for distributed synchronous operation: See the Function Manual titled *Motion Control Technology Objects: Synchronous Operation, Cam*.

---

#### Note

An iDevice can only be created using SIMOTION V4.1.1 or higher.

---

#### Properties of an iDevice

In this case, the following must be observed:

- If the IO device in the iDevice is operated with IRT, the IO controller in the iDevice can only be operated with RT. In other words, if a SIMOTION controller is operated as an iDevice with IRT on a higher-level controller, the SIMOTION controller can only operate its lower-level IO devices with RT. The configuration data for ports, the synchronization and the IRT scheduling data of the IO device in the iDevice are loaded from the higher-level IO controller.
- If the IO controller in the iDevice is operated with IRT, the IO device in the iDevice can only be operated with RT.
- The send clock of the IO device in the iDevice is the same as the send clock in the sync domain of the higher-level IO system.
- The IO controller that is of a higher-level than the iDevice provides an IO system, in which the iDevice is an IO device. The IO controller in the iDevice provides its own IO system.

The following figure shows how you can operate an iDevice with a higher-level IO controller.

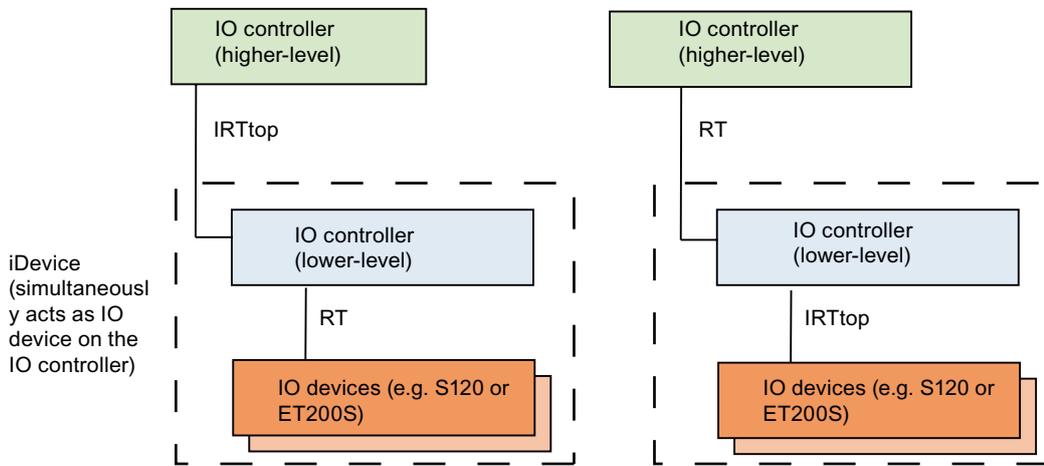


Figure 6-20 Configuration possibilities with iDevices

**Procedure for the configuring**

- The iDevice itself and the higher-level station on which it is operated (IO controller) should be contained in different projects in HW Config.
- Following successful configuration, a GSD file is generated from the iDevice by means of an export from HW Config.
- Once the GSD file has been imported, a substitute iDevice is created on the higher-level IO controller.
- Because no matching is performed on the configuration of the iDevice and of the substitute iDevice, the configuration becomes inconsistent after one of the components has been changed. It is also impossible to perform a consistency check as the iDevice is imported via a GSD file.

**6.5.2 Configuring a PROFINET iDevice**

**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 iDevice.

**Note**

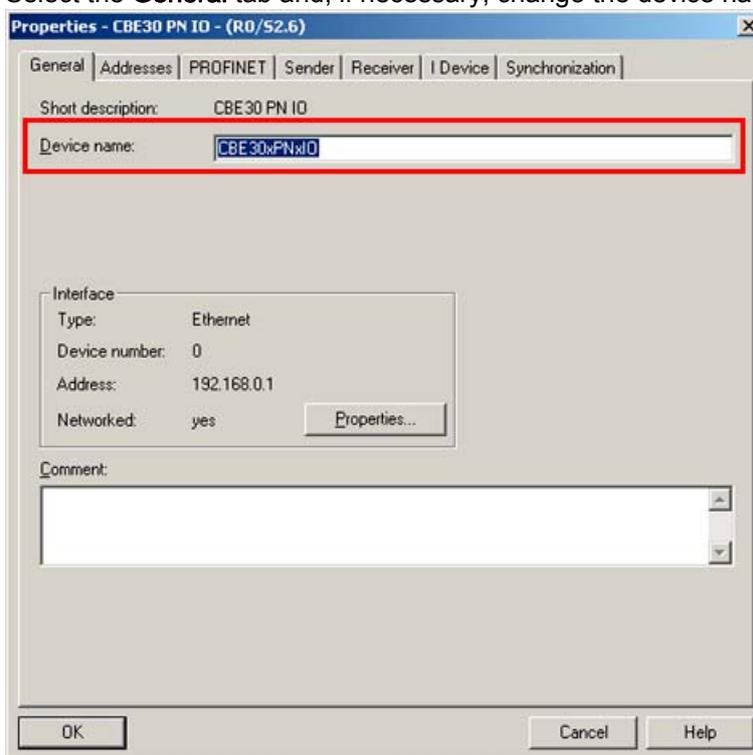
When configuring the iDevice, observe the possible settings for the RT class, see PROFINET iDevice (Page 169) .

**NOTICE**

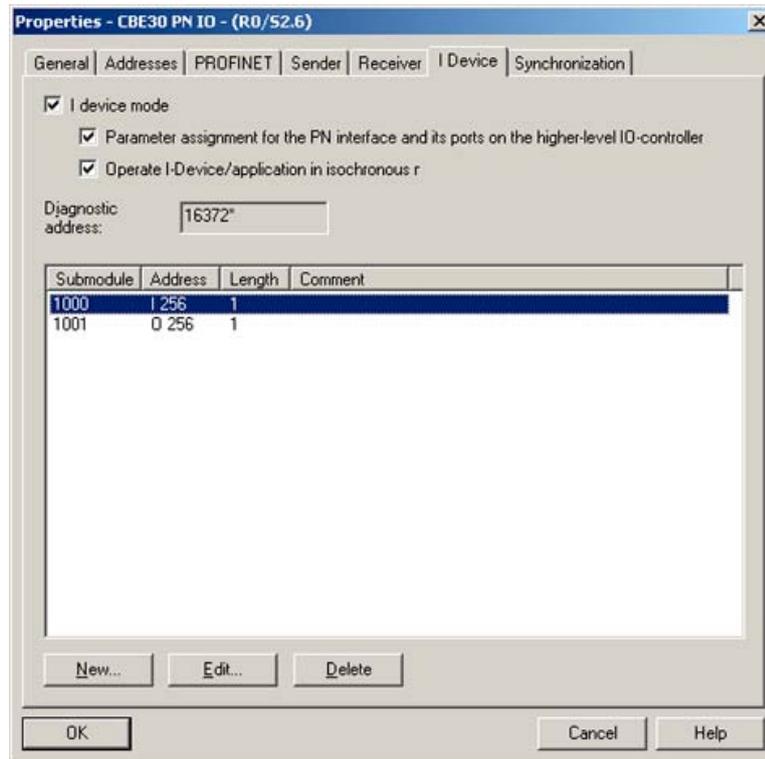
If you use a SIMOTION controller as higher-level IO controller, the device name of the iDevice (NameOfStation) may not contain any "-". They will be changed into "x" when you insert the iDevice in the IO system. Because the device name has then changed, you can no longer control the iDevice in the PROFINET IO system.

**Proceed as follows**

1. Double-click the interface module of the CPU. The **Properties** dialog box opens.
2. Select the **General** tab and, if necessary, change the device name (without any "-").



3. Select the **iDevice** tab.



4. Select the **iDevice mode**.
5. Select whether the **Parameterization of the PN interface and of its ports on higher-level IO controller** should be performed.

You can thus operate communication between the higher-level controller and the controller in the iDevice via IRT (ports are created in the GSD, and the parameterization data records are loaded to the controller of the iDevice on startup).

If you do not select this option, you can only operate communication between the higher-level IO controller and the IO device of the iDevice via RT.

6. Select **Operate I-Device/application in isochronous r** if you want to operate communication isochronously. The additional **Application** tab is then displayed in the **Properties** dialog on the iDevice. If you have not selected this option, the iDevice will not operate isochronously.
7. Click **New** each time to create the virtual subslots (I and O address) and configure these according to requirements. You thus configure the IO area of the iDevice. Do not perform any further settings in the **Sender** and **Receiver** tabs.
8. Click **OK** to accept these settings and save the project.
9. Continue with **Creating a substitute iDevice** (Page 173).

### Manually configuring the send clock/update time in the iDevice

- **IO device in iDevice with IRT/IO controller using PROFINET with RT.**  
In this case, specify the send clock in the **Properties <Profinet Interface>** dialog box under **PROFINET**. Possible times are 500, 1000, 2000 and 4000  $\mu$ s. Also set this value on the higher-level IO controller.
- **IO device in iDevice with RT/IO controller using PROFINET using IRT.**  
In this case, specify the send clock in the **Properties <Profinet Interface>** dialog box under **PROFINET**. You must then set the updating time on the higher-level IO controller to be equal to or an integer multiple of this value.

---

**Note**

You must ensure that the send clocks and update times on the iDevice and the higher-level IO controller are the same, or that one is an integer multiple of the other (see above).

---

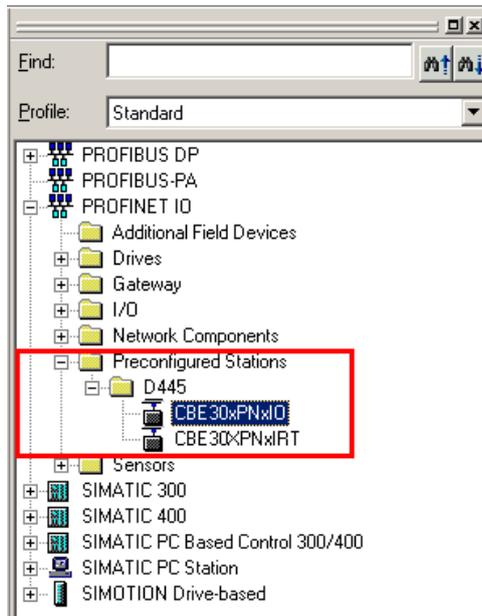
### 6.5.3 Creating a substitute iDevice

#### Requirements

You have already configured the module to be used as iDevice.

#### Proceed as follows

1. First save the project.
2. Perform **Options > Create GSD file for iDevice**.  
The **Create GSD file** dialog box will be displayed.
3. Select the iDevice and enter a name for the substitute iDevice.
4. Click on **Export**. The **Save as** dialog is displayed.
5. Select the path in which the GSD file of the substitute iDevice is to be created and click **Save**.
6. Click **Install** to install the substitute iDevice.  
The substitute iDevice is now present in the hardware catalog under **PROFINET IO > Preconfigured Stations** and can be selected.



## 6.5.4 Inserting an iDevice in the higher-level IO controller

### Requirement

You have already created a substitute iDevice. The IO controller and the corresponding IO system have already been configured.

### Proceed as follows

1. Open the hardware catalog.
2. Drag the iDevice substitute from the hardware catalog (**PROFINET IO > Preconfigured Stations**) to the IO system.

The substitute iDevice is displayed as a normal IO device. Depending on whether the iDevice is controlled using RT or IRT, ports will (IRT iDevice) or will not (RT iDevice) be displayed.

The following figure shows a higher-level IO controller with an RT iDevice.

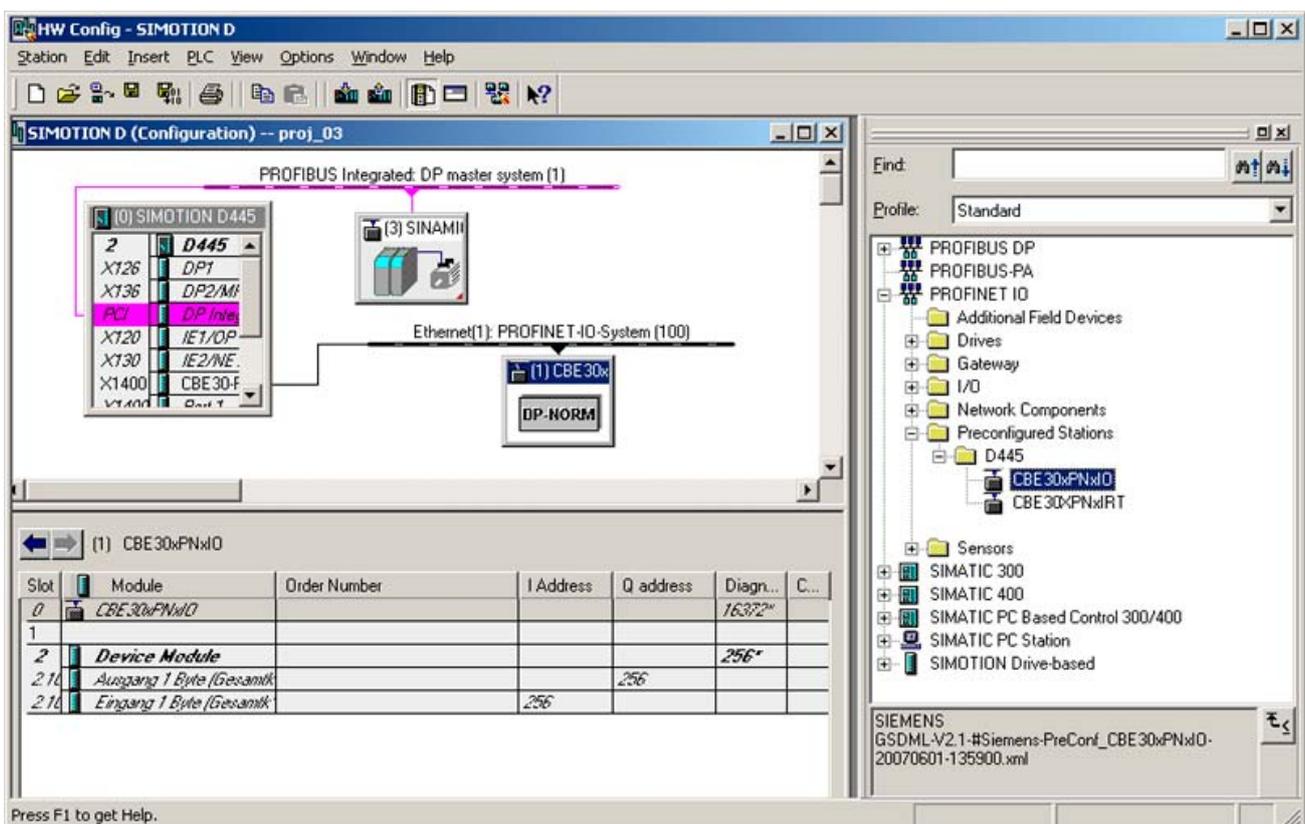


Figure 6-21 RT iDevice on the IO controller

6.5 Configuring the iDevice

The following figure shows a higher-level IO controller with an IRT iDevice.

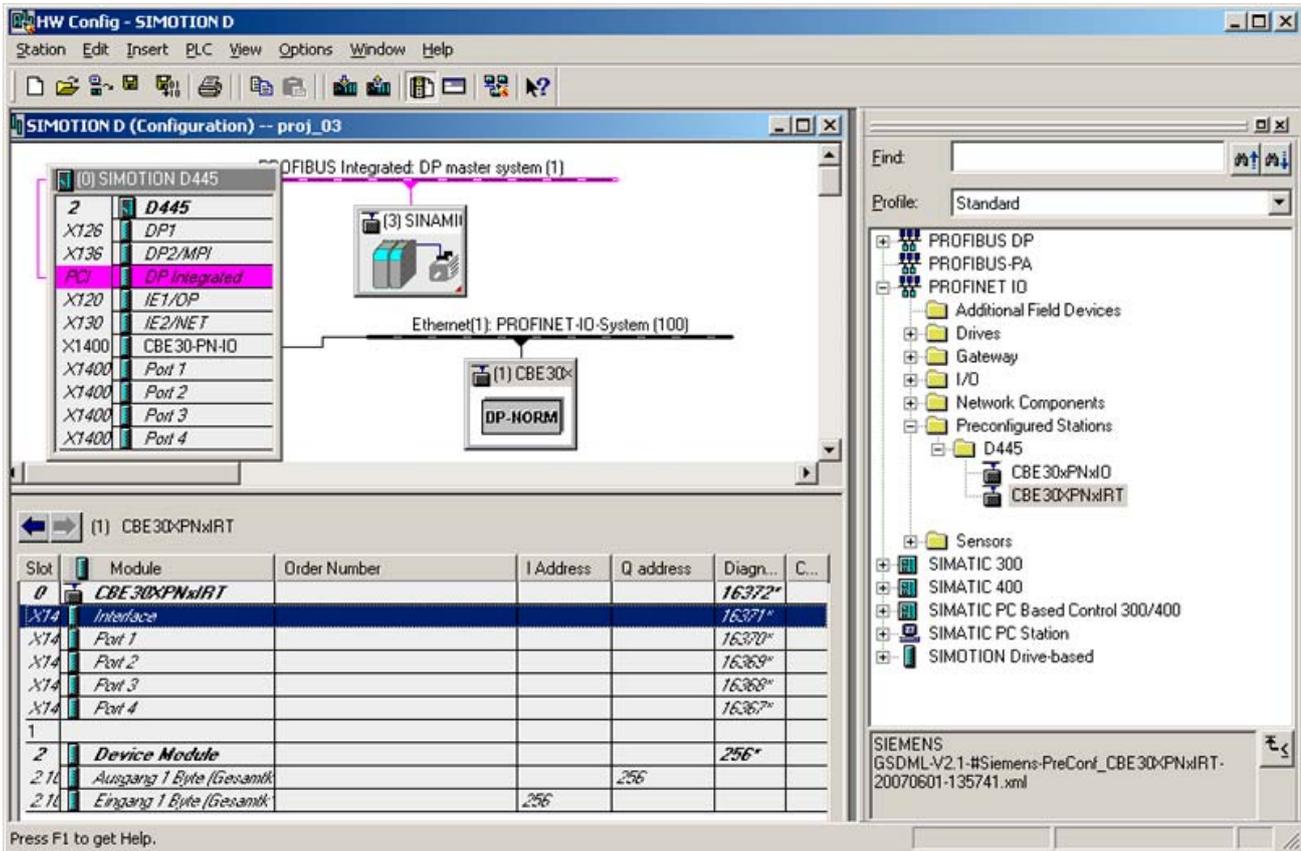


Figure 6-22 iDevice on the IO controller

The number of submodules corresponds to the number of the configured submodules of the iDevice in the GSD file. The module and the submodules cannot be deleted.

### Interconnecting IRT iDevice ports

As with an IO device, you must interconnect the ports between the higher-level controller and the iDevice for an IRT iDevice.

1. Select the iDevice and perform **Edit > PROFINET IO > Topology**.  
The topology editor opens.
2. Interconnect the ports and click **OK** to accept these settings.

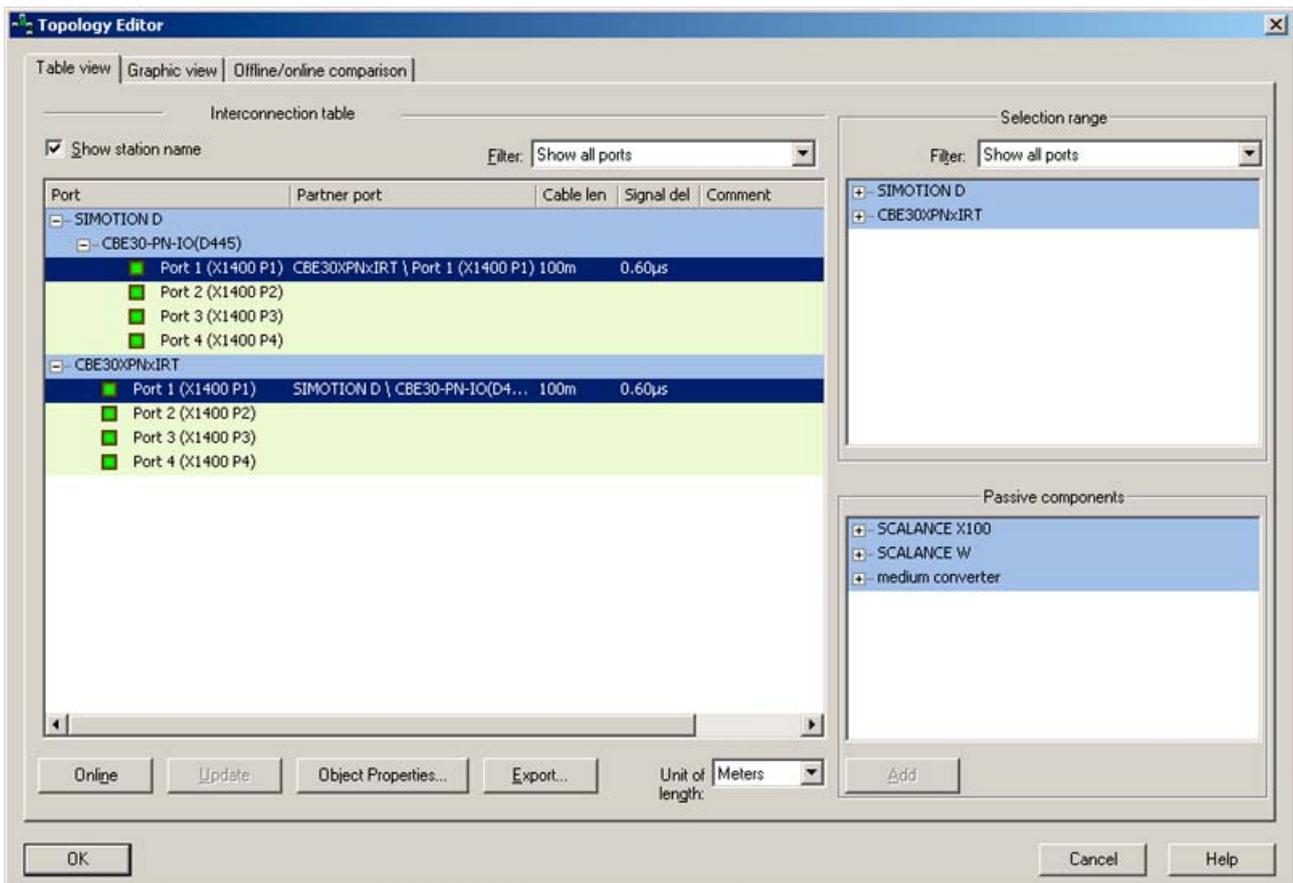


Figure 6-23 Interconnect iDevice ports

### Assigning the IP address of the iDevice

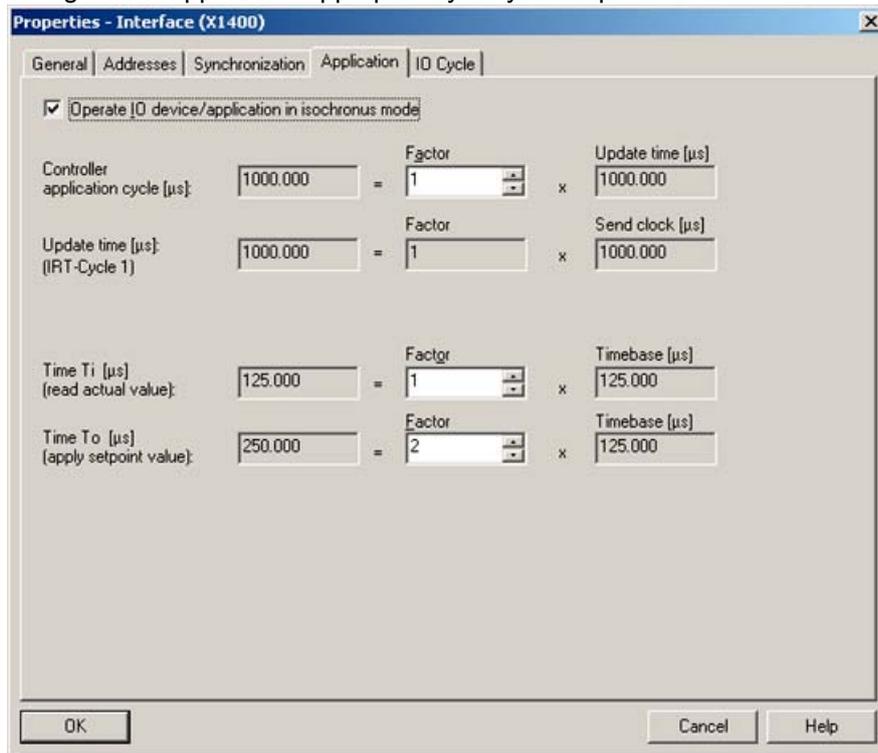
1. Double-click the iDevice to display the **Properties** dialog.
2. Disable the **Assign IP address via IO controller** option.

The IP address should not be assigned by the higher-level IO controller because it already assigns this in the Step7 project of the iDevice.

Exceptions exist, for example, if you cannot access the Step7 project of the iDevice.

**Edit the synchronization and the application (iDevice and higher-level IO controller isochronous)**

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 isochronized** option in the **Application** tab and configure the application appropriately for your requirements.



**Note**

T<sub>i</sub>, T<sub>o</sub> and CACF have no significance for iDevices and do not need to be entered.

**Setting the update time and send clock on the IO controller**

- **IO device in the iDevice with RT (update time)**

If the iDevice is operated in RT mode (the higher-level IO controller is the IRT), you must set the update time in the higher-level project as equal to or as an integer multiple of the update time of the iDevice in the lower-level project (iDevice project).

Double-click the PROFINET IO system and select the **Update time** tab in the **PROFINET subnet properties** dialog box. Set the updating time there.

- **IO device in the iDevice with IRT (send clock)**

The send clock must be identical in both projects. You can set the send clock in the higher-level project using **Edit > PROFINET IO > Domain Management**. Possible times are 500, 1,000, 2,000 and 4,000 µs.

**Note**

You must ensure that the send clocks and update times on the iDevice and the higher-level IO controller are the same, or that one is an integer multiple of the other (see above).

---

## 6.6 Loading the communication configuration

### 6.6.1 Loading the PROFINET IO configuration

**Requirement**

A PG/PC with which you can go ONLINE is connected.

**Proceed as follows**

The configuration data must be loaded in all participating SIMOTION devices after the successful configuration of PROFINET IO.

1. In NetPro, select the Ethernet subnet and then select the **Target system > Loading in current project > Nodes on the subnet** menu command.

## 6.7 Data exchange between SIMATIC and SIMOTION via PROFINET

### 6.7.1 Data exchange through the use of iDevices

**Description**

With STEP7 5.4 SP2, the data exchange is also possible through the use of iDevices. You can configure a SIMOTION controller as iDevice and use it in the sync domain of the respective partner.

For further information on the configuration of iDevices, see PROFINET iDevice (Page 169)

### 6.7.2 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 configuring, two IO devices are derived from a PN/PN coupler which means that there is one IO device for each station 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.

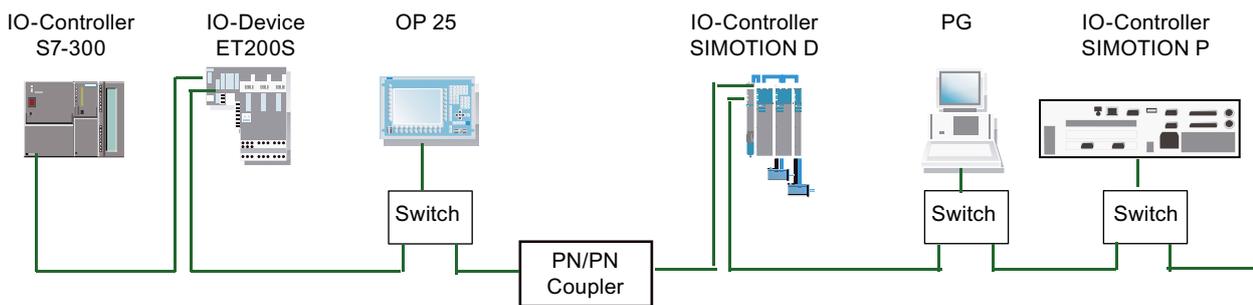


Figure 6-24 Coupling two PROFINET subnets with one PN-PN coupler

---

#### Note

Detailed information about the PN-PN coupler is contained in the appropriate device documentation.

---

#### Configuring the PN-PN coupler

You use STEP 7 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.

### 6.7.3 Communication using standard protocols

#### Description

Since the PROFINET interface supports standard Ethernet services (amongst other things), you can exchange data on an individual basis with SIMATIC modules via TCP and UDP. Your user program must handle the management of the communications connection. You can use this connection, for example, to exchange data between a SIMATIC CPU and a SIMOTION controller via PROFINET.

This includes:

- 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 \(Page 113\)](#)

[\\_tcpOpenClient function \(Page 113\)](#)

[\\_tcpReceive function \(Page 114\)](#)

[\\_tcpSend function \(Page 114\)](#)

[\\_tcpCloseConnection function \(Page 115\)](#)

[\\_tcpCloseServer function \(Page 115\)](#)

## 6.8 Diagnostic and alarm behavior

### 6.8.1 PROFINET IO alarm and diagnostic messages to SIMOTION

#### Description

For PROFINET IO there is an alarm and diagnostic functionality for PROFINET devices.

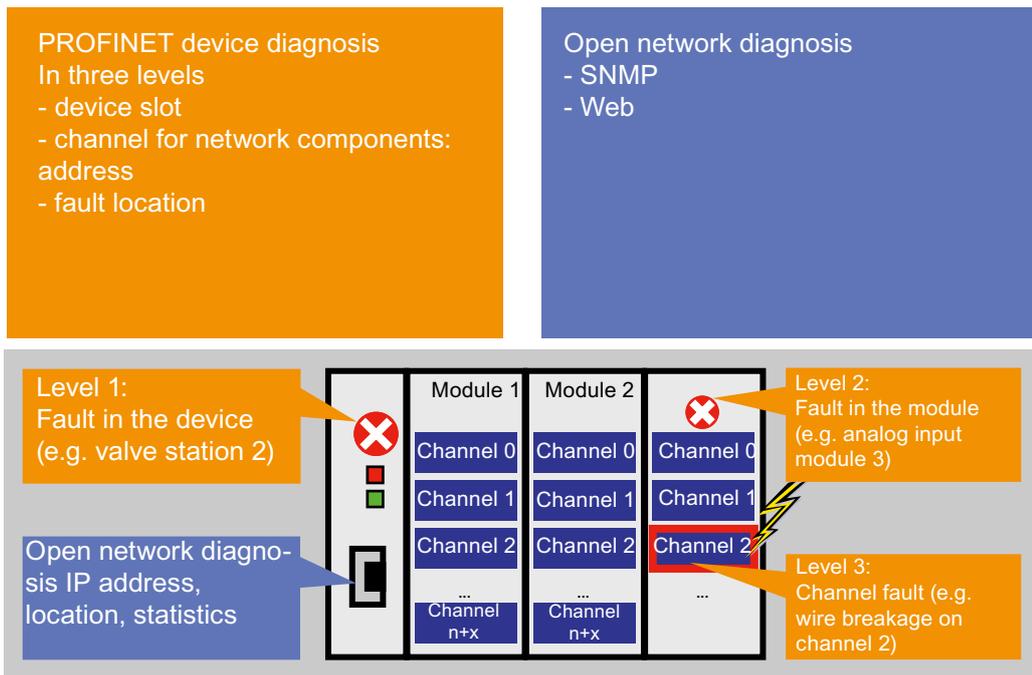


Figure 6-25 Diagnostics overview

#### Device diagnostics

The device diagnosis can be divided into three levels. For detailed information, see "Diagnostic model".

## 6.8.2 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.

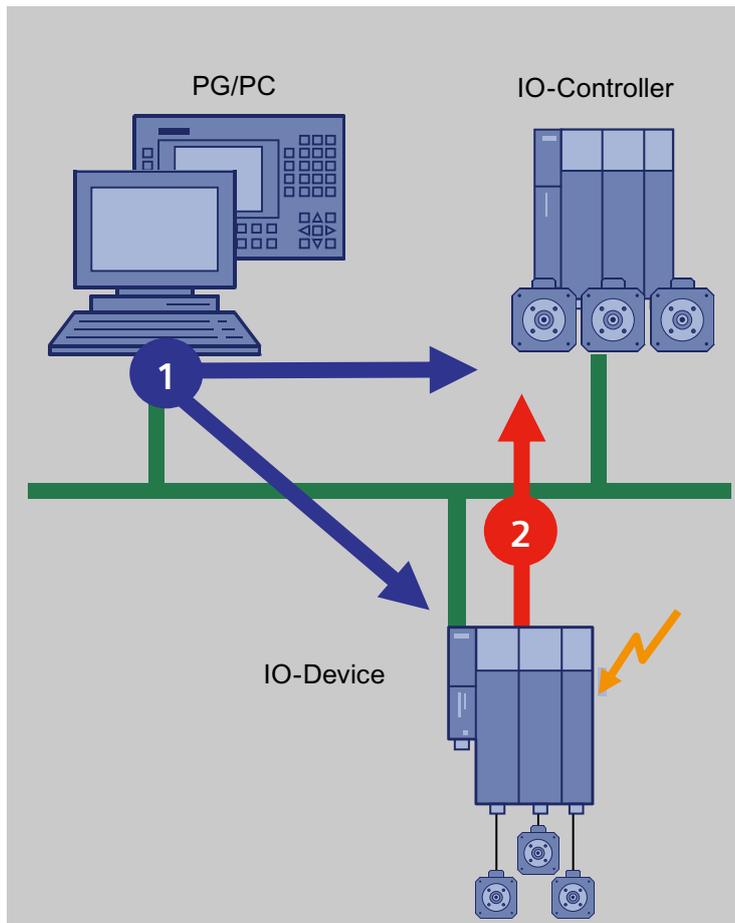


Figure 6-26 Access to the diagnostic data

### 6.8.3 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 (TSl#InterruptId)	TSl#eventClass	TSl#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.

### 6.8.4 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)	16#39	16#42	Incoming diagnostic interrupt
Diagnosis disappears (outgoing)	16#38	16#42	Outgoing diagnostic interrupt
Multicast Communication Mismatch			
Port Data Change Notification			
Sync Data Changed Notification			
Isochronous Mode Problem Notification			
Network component problem notification (_SC_DIAGNOSTIC_INTERRUPT (=201))			
Process interrupt (_SC_PROCESS_INTERRUPT (= 200))	16#11	16#41	Process interrupt
Pull Alarm Plug Alarm Plug Wrong Submodule Alarm Return of Submodule Alarm (_SC_PULL_PLUG_INTERRUPT (=216))	16#39	16#51	PROFINET IO module has been removed or cannot be addressed.
		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.

## 6.8.5 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).

### 6.8.6 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 185)
Sync Data Changed Notification	
Isochronous mode problem notification	
Multicast Communication Mismatch	

#### Technology/application alarms

Not yet sent to the controller as standard PROFINET alarms

## 6.8.7 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 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 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 _getStateOfDiagnosticDataC ommand	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)

## 6.8.8 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

### 7.1 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).

## 7.2 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.

## 7.3 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.

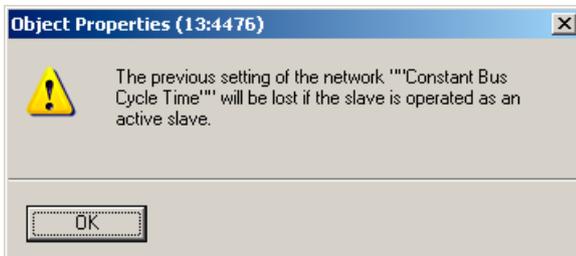
### S7 routing

S7 routing is supported by SIMOTION, i.e. information 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 (from 4.1.2 or 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.



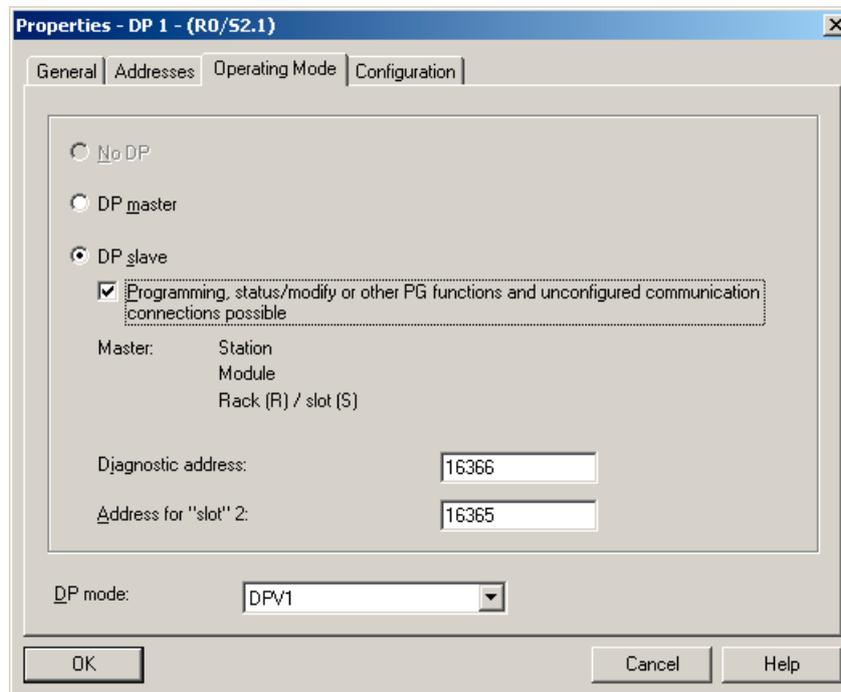


Figure 7-1 DP slave mode: Active station: Testing, commissioning, routing

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.

---

## 7.4 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)

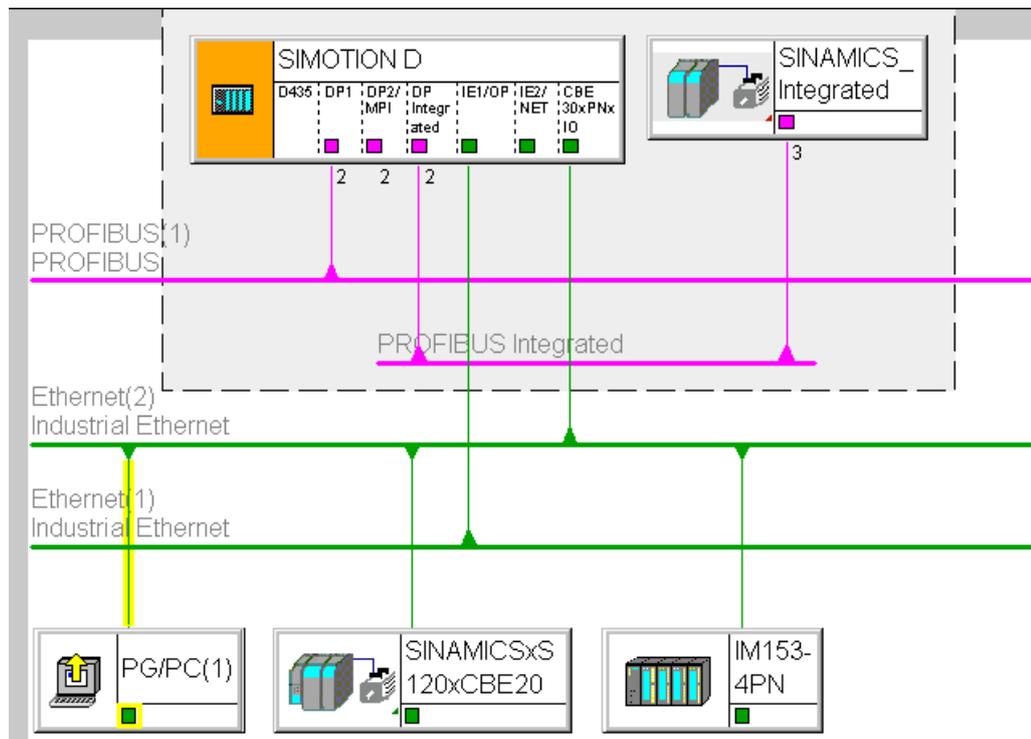


Figure 7-2 Example for PG/PC to CBE30

- S7 routing to the (master) PROFIBUS interfaces (only if configured)
- S7 routing to the standard Ethernet interfaces ET1/ET2 (X120, X130) (V.4.1.2 and higher)
- Access to the components on the same subnet (CBE30) via the switch functionality

## Engineering system / HMI to PROFIBUS

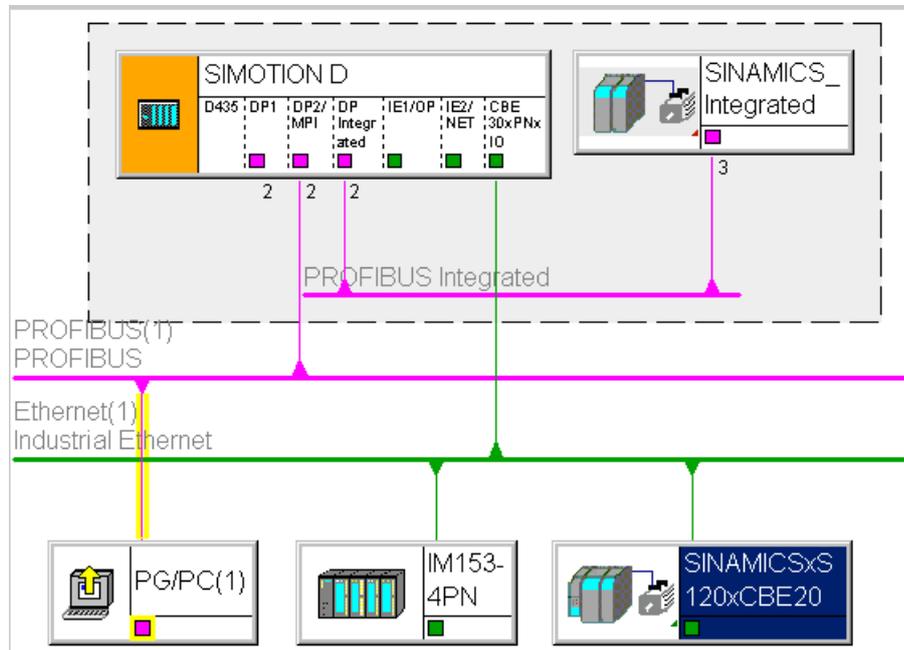


Figure 7-3 Example for PG/PC to PROFIBUS

- 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) (V.4.1.2 and higher)

Engineering system / HMI to Ethernet

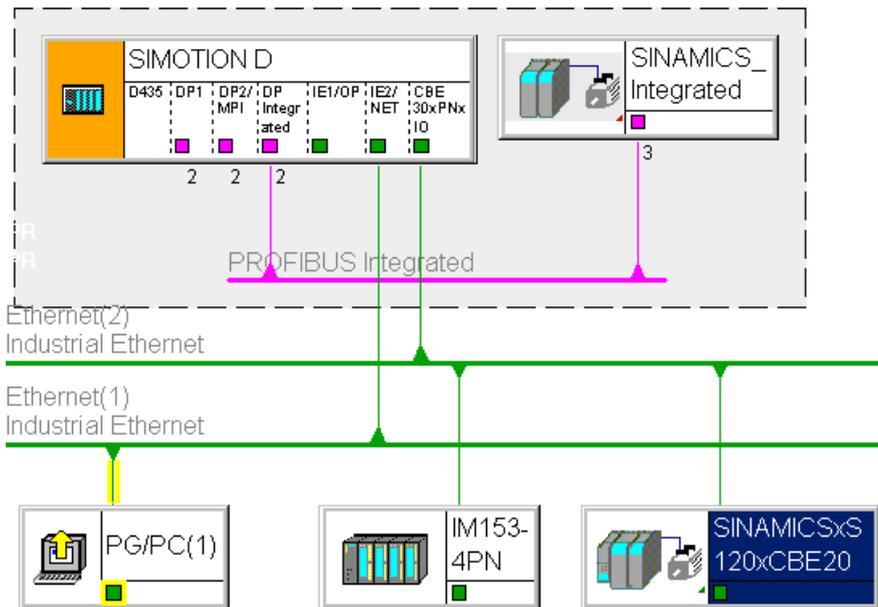


Figure 7-4 Example for PG/PC to Ethernet X120, X130

- S7 routing to the other (master) PROFIBUS interfaces (only if configured)
- No S7 routing to X1400 on the CBE30

## 7.5 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.

## 7.6 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

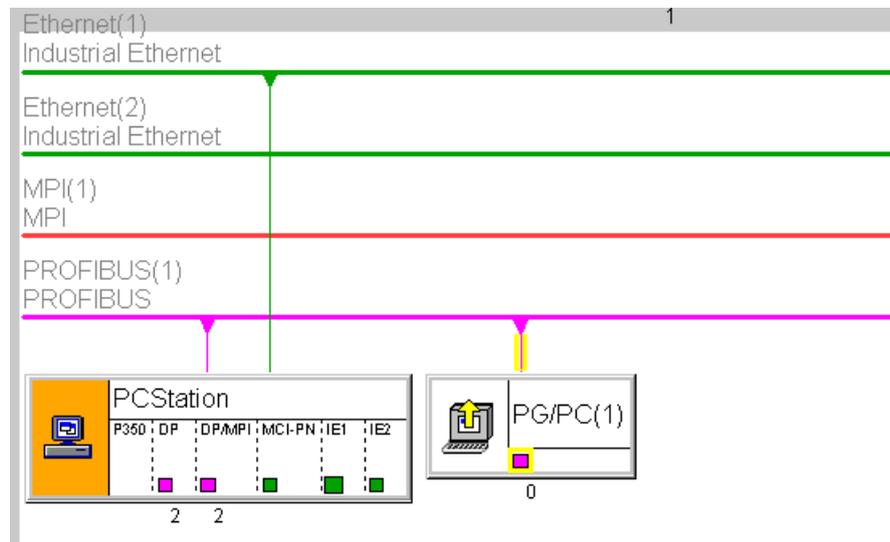


Figure 7-5 Example for P350 routing from PROFIBUS to PROFINET

### Routing from PROFINET on PROFIBUS

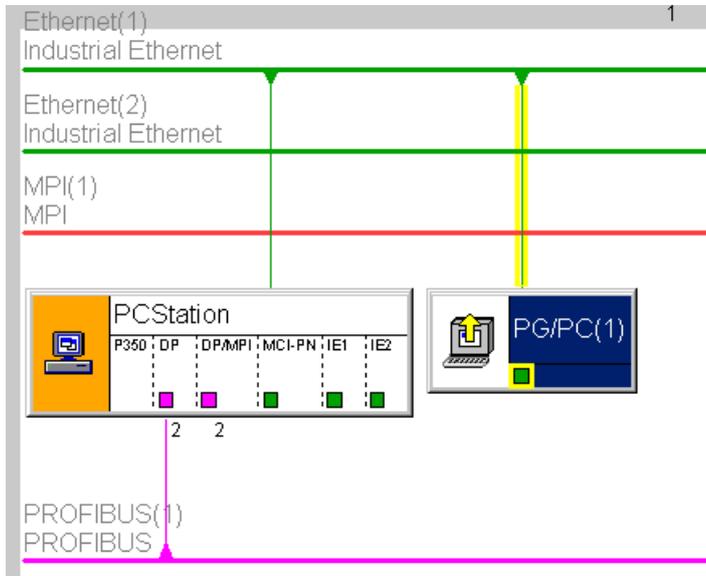


Figure 7-6 Example for P350 routing from PROFINET to PROFIBUS

# SIMOTION IT

## 8.1 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

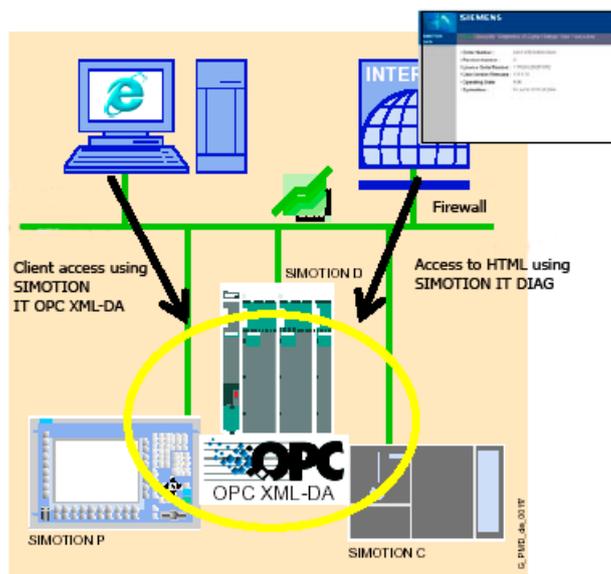


Figure 8-1 SIMOTION IT overview

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 201)

SIMOTION IT DIAG (Page 202)

SIMOTION IT OPC XML DA (Page 205)

## 8.2 Web access to SIMOTION

### Description

The following figure shows the various possibilities to access the data in a SIMOTION module.

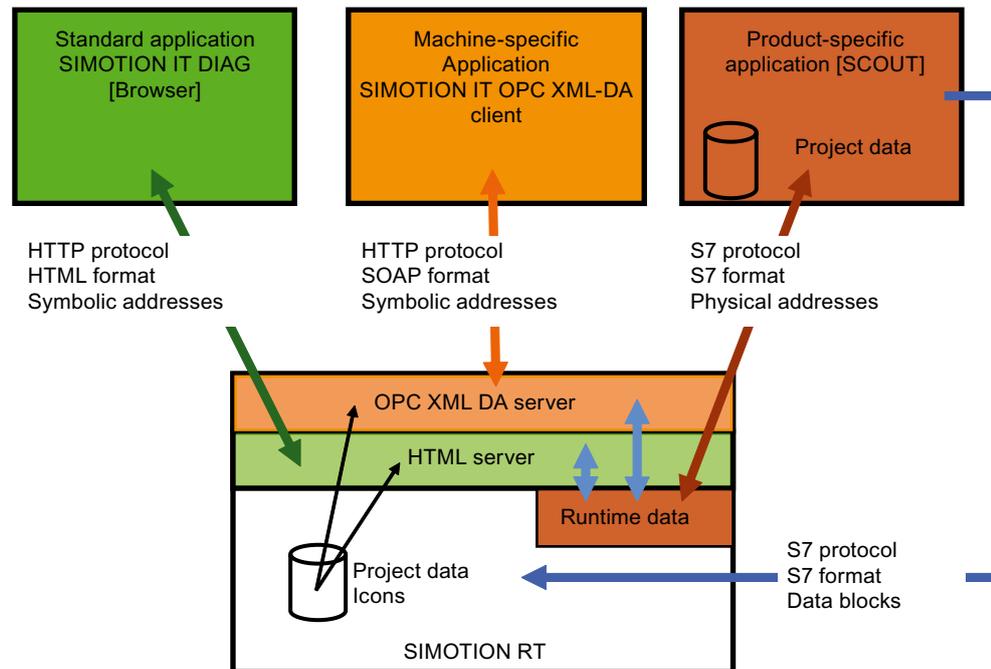


Figure 8-2 Access to SIMOTION

### See also

SIMOTION IT DIAG (Page 202)

SIMOTION IT OPC XML DA (Page 205)

## 8.3 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 loading, memory use, operating status, task run time display)
- **Diagnostic buffer** (messages from the diagnostic buffer are displayed)
- **IP-Config** (data of the SIMOTION device interface)
- **Settings** (firmware upload, project upload, project download, configuration upload, set the time zone, alter operating states)
- **File structure** (access to the SIMOTION file system, upload and download of files, create folders and store additional data, e.g. documentation)
- **Access protection** (access to the diagnostic pages can be restricted by the assignment of user identification and password)
- **HTTPS - access using encrypted communication** (encrypted messages)

## User-defined pages

You can create HTML pages yourself and so display SIMOTION variable values. The access to the SIMOTION system variable is realized using a special syntax in HTML code, so-called Server Side Includes. The following figure shows an example access to a user-defined page:

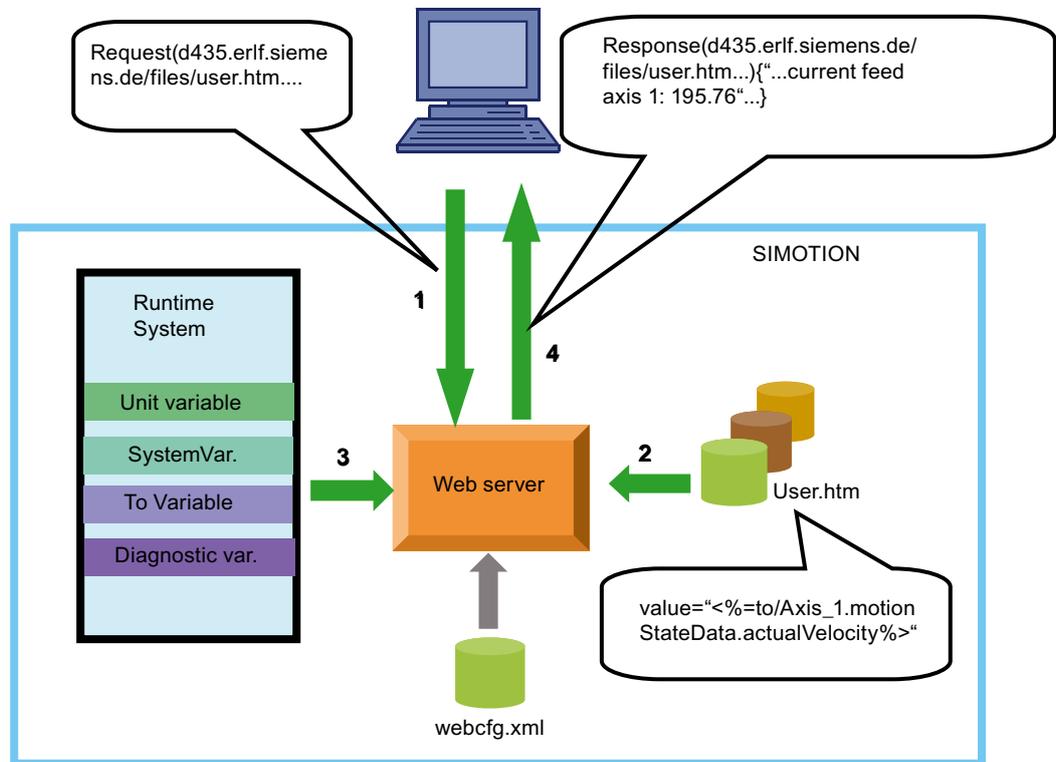


Figure 8-3 User-defined HTML pages

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.

## HTML code examples

The examples show how you can use HTML code to access variables and display figures.

Table 8-1 Read variable

```
<td>unit/Bedienen.controller_on</td>
<td align="right">
  <%= unit/Bedienen.controller_on %>
</td>
```

Table 8-2 Write variable

```
<form method="post" action="/VarApp">
  <td>
    <input type="hidden" value="TRUE" name="unit/Bedienen.start_prog"/>
    <input type="submit" value="Start" />
  </td>
</form>
```

Table 8-3 Simple graphic

```
<script language="JavaScript">
<!--
  if ('<%=unit/Bedienen.controller_on%>' == 'true'){
    document.writeln('<td style="background-color:#00FF00; font-size:1px;
width:12; height:12; border-style:solid; border-color:black; border-
width:1px">&nbsp;&nbsp;&nbsp;</td>');
  }
  else {
    document.writeln('<td style="background-color:#FF0000; font-size:1px;
width:12; height:12; border-style:solid; border-color:black; border-
width:1px">&nbsp;&nbsp;&nbsp;</td>');
  }
//-->
</script>
```

### Access to variables

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

## 8.4 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

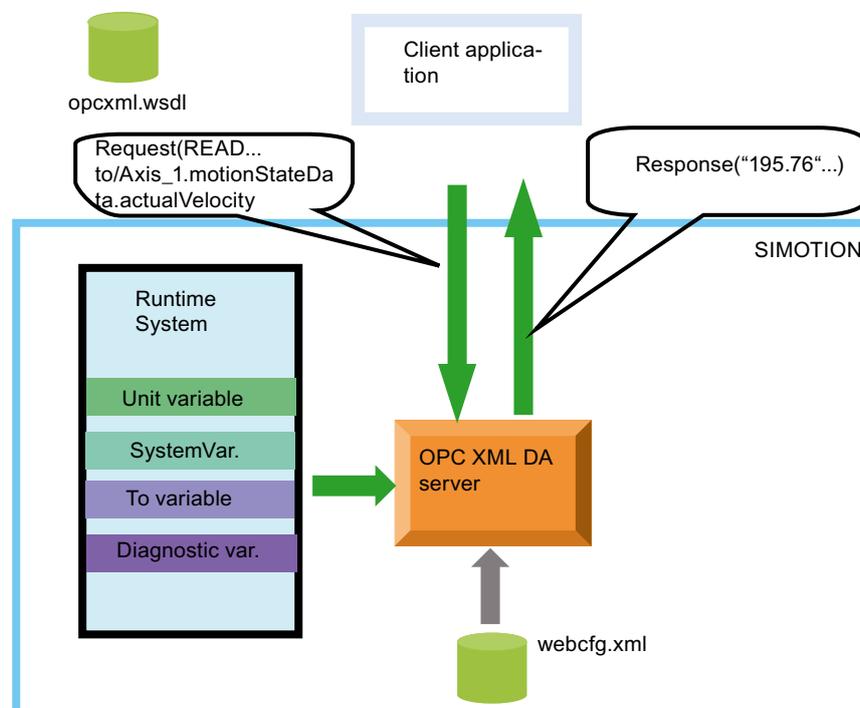


Figure 8-4 Access to the OPC server

**C# code example**

Table 8-4 Establishing a connection

```
OpcXmlDa_R1_0.Service MyServer = new OpcXmlDa_R1_0.Service();  
MyServer.Url = "http://" + this.ServerAddress.Text + "/soap/opcxml";
```

Table 8-5 Write variable

```
WriteServer.Write(RequestOptions,WriteItemList,true, out RItemList,out  
WriteErrorList);
```

Table 8-6 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);
```

**Access to variables**

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

## **8.5 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

–

- \_readRecord
  - Application, 45
- \_tcpCloseConnection, 115
- \_tcpCloseServer, 115
- \_tcpOpenClient, 113
- \_tcpOpenServer, 113
- \_tcpReceive, 110, 114
- \_tcpSend, 114
- \_writeRecord
  - Application, 45
- \_xreceive, 78
- \_xsend, 76

## C

- Communication
  - Between SIMOTION and SIMATIC, 67
  - SIMATIC as DP slave, 71
  - SIMATIC as iSlave, 72
  - SIMOTION as DP slave, 68
  - SIMOTION as i-slave, 69
- Configuring the sender, 167
- Connection configuration
  - Ethernet, 87
- Constant bus cycle time, 127
- Controller Application Cycle Factor, 135
- Cycle clock scaling, 135

## D

- Data set 47, 37
- Diagnostics model, 183
- DP slave
  - SIMOTION, 68
- DP V1 communication, 34
  - Program example, 64

## E

- Ethernet

- Properties of the subnets, 79
- Ethernet communication
  - Configuring a connection, 87
  - Modeling, 80
  - SIMATIC S7 functions, 84
  - SIMATIC with integrated interface, 94
  - TCP/IP connection, 89
  - UDP connection, 92
  - Use of \_tcpReceive, 110

## F

- FTP file transfer, 207

## I

- iDevice
  - Configuring, 169
- Insert PROFINET board, 143
- IRT (High Flexibility), 128
- IRT High Performance, 129
- iSlave
  - SIMATIC, 72
- i-Slave
  - SIMOTION, 69
- Isochronous mode, 127

## N

- Network topology, 130

## P

- PN-PN coupler, 180
- PROFIBUS
  - Acyclic communication, 34
  - Cyclic services, 33
  - DPV1 communication, 34
- PROFIBUS master-master
  - S7 system functions, 75
  - SIMOTION functions, 76
- PROFIdrive
  - Application classes, 26

- Profile, 25
- PROFINET IO
  - IO controller, 120
  - IO device, 121
  - IRT (High Flexibility), 128
  - Network topology, 130
  - RT, 126
  - SIMATIC - SIMOTION data exchange, 181

## R

- Recipients
  - configuring, 168
- References, 4
- Refresh time, 156
- Routing
  - For SIMOTION, 192
  - S7 routing, 192

## S

- Send clock, 156
- SIMATIC S7-300
  - Features of the Ethernet communication, 84
- SIMATIC S7-400
  - Features of the Ethernet communication, 85
- SIMOTION IT
  - Access to variables, 204, 206
  - IT-DIAG, 202
  - OPC XML DA, 205
- SP slave
  - SIMATIC, 71
- Sync domain, 121
  - creating, 149
- Sync master, 127
  - Redundant, 138
- Sync slave, 127

## T

- TCP/IP communication
  - SIMOTION functions, 100
- TCP/IP connection, 89
  - SIMATIC S7 function blocks, 105
  - SIMATIC S7 functions, 99
  - SIMOTION functions, 110
- Topology
  - Configuring, 151

## U

- UDP connection, 92
  - SIMATIC S7 functions, 104
  - SIMOTION functions, 104

## X

- X\_RCV, 75
- X\_SEND, 75

