SIEMENS

SIMOTION

Motion Control Communication

System Manual

Valid as of Version 4.4

01/2015

Legal information

Warning notice system

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

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

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

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

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

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

Qualified Personnel

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

Proper use of Siemens products

Note the following:

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

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

℗ 02/2015 Subject to change

Foreword

Foreword

Content

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

Scope of validity

This manual is valid for SIMOTION SCOUT product level V4.4:

- SIMOTION SCOUT V4.4 (Engineering System for the SIMOTION product family)
- The software configuration is described in this manual based on SIMOTION SCOUT and SIMATIC STEP 7 Version V5.x.
  Information on configuration in the Engineering Framework Totally Integrated Automation Portal (SIMOTION SCOUT in the TIA Portal), you will find in the configuration manual SIMOTION SCOUT TIA.

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.
- **PROFIBUS**
  Information about the DPV1 communication, and the setup and programming of the communication between SIMOTION and SIMATIC devices.
- **PROFINET IO**
  Information about configuring PROFINET with SIMOTION
- **Ethernet introduction (TCP/IP and UDP connections)**
  Information about the the setup and programming of the Ethernet communication between SIMOTION and SIMATIC devices.
- **Routing - communication across network boundaries**
  General information about routing
- **SIMOTION IT**
  General information about the IT and Web functions provided by SIMOTION.
- **PROFIsafe**
  General information on configuring failsafe controls.
- **PROFIdrive**
  Description of the PROFIdrive profile.
- **Index**
  Keyword index for locating information

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION V4.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

Additional information

Click the following link to find information on the following topics:

- Ordering documentation / overview of documentation
- Additional links to download documents
- Using documentation online (find and search manuals/information)

http://www.siemens.com/motioncontrol/docu

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

http://www.siemens.com/mdm

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

http://support.automation.siemens.com

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

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

Table of contents

Fundamental safety instructions

1

1.1 General safety instructions

⚠ WARNING

Risk of death if the safety instructions and remaining risks are not carefully observed

If the safety instructions and residual risks are not observed in the associated hardware documentation, accidents involving severe injuries or death can occur.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

⚠ WARNING

Danger to life or malfunctions of the machine as a result of incorrect or changed parameterization

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization (parameter assignments) against unauthorized access.
- Respond to possible malfunctions by applying suitable measures (e.g. EMERGENCY STOP or EMERGENCY OFF).

1.2 Industrial security

Note

Industrial security

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit http://www.siemens.com/industrialsecurity.

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit http://support.automation.siemens.com

⚠ WARNING

Danger as a result of unsafe operating states resulting from software manipulation

Software manipulation (e.g. by viruses, Trojan horses, malware, worms) can cause unsafe operating states to develop in your installation which can lead to death, severe injuries and/or material damage.

- Keep the software up to date.
  Information and newsletters can be found at:
  http://support.automation.siemens.com
- Incorporate the automation and drive components into a state-of-the-art, integrated industrial security concept for the installation or machine.
  For more detailed information, go to:
  http://www.siemens.com/industrialsecurity
- Make sure that you include all installed products into the integrated industrial security concept.

Introduction 2

2.1 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 explanations 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 3

3.1 Network options

3.1.1 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.

Note

Appropriate protective measures (among other things, IT security, e.g. network segmentation) are to be taken in order to ensure safe operation of the system. You can find more information on Industrial Security on the Internet at:

www.siemens.de/industrialsecurity.

3.1.2 PROFINET

Overview

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

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

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

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

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

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

3.1.3 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.

3.1.4 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).

3.1.5 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.

3.1.6 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.

3.2 Communications services (or network functions)

3.2.1 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.

Communicating with SIMATIC

For the communication functions of the SIMATIC controllers, refer to the SIMATIC documentation.

- SIMATIC S7-1500, ET 200MP, ET 200SP communication (http://support.automation.siemens.com/WW/view/en/59192925)
- SIMATIC communication with SIMATIC (http://support.automation.siemens.com/WW/view/en/25074283)

3.2.2 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 operator control and monitoring system. All SIMOTION and SIMATIC networks support the PG/OP communication services.

3.2.3 S7 communication services

Overview

S7 communication services provide data exchange using communication system function blocks (SFBs) and communication 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.

3.2.4 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.

3.2.5 "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.

3.2.6 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, not for SIMOTION devices.

3.2.7 Industrial Ethernet communication services

Overview

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

Industrial Ethernet provides the following communication services:

- The ISO transfer provides services for transmitting data via connections that support error-free data transmission. The ISO transfer is only possible with STEP7.
- TCP/IP allows you to exchange contiguous data blocks between the controllers and computers in PROFINET or Industrial Ethernet networks. With TCP/IP, the controller transmits contiguous data blocks.
- ISO-on-TCP (RFC 1006) supports error-free data transmission. For SIMOTION only when going though SCOUT ONLINE. If the communication is performed from the user program, an RFC must be programmed.
- UDP (User Datagram Protocol) and UDP multi-cast provide simple data transmission without acknowledgment. You can transmit related 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.

3.2.8 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.

3.3 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)

With OPC, Windows applications can access process data so that it is easy for devices and applications from different vendors to be combined with each other. 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 application 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, see SIMOTION IT Ethernet-based HMI and Diagnostic Functions.

PROFIBUS DP

4

4.1 PROFIBUS DP communication

4.1.1 Overview of PROFIBUS DP communication

Description

PROFIBUS DP (Decentralized Peripherals) is designed for fast data exchange at the field level. The communication is performed between a class 1 PROFIBUS DP master (e.g. a SIMOTION controller) and PROFIBUS DP 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.

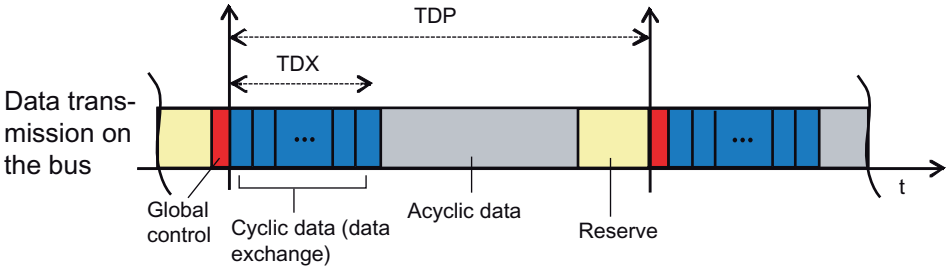


Figure 4-1 Data protocol on PROFIBUS DP

4.2 Cyclical data exchange between a SIMOTION and SIMATIC controller.

4.2.1 Description of cyclical SIMOTION and SIMATIC controller data exchange

The following section describes the options for having a SIMOTION and a SIMATIC controller communicating with one another via PROFIBUS DP.

There are various possibilities:

- A SIMOTION controller is connected to a PROFIBUS DP master system of a SIMATIC controller as a PROFIBUS DP slave.
- A SIMATIC controller is connected to a PROFIBUS DP master system of a SIMOTION controller as a PROFIBUS DP slave.
- A master-master communication is used between SIMOTION and SIMATIC controls.

Linking as DP slave

For linking as a DP slave, there are two options:

- The controller is connected to the DP master as a "standard" DP slave by means of a GSD file. This is essential if the two controllers are in different projects.
- The controller is connected to the DP master as a so-called I slave (intelligent DP slave). For this purpose, the two controls must be located in one project.

The two options are supported by SIMOTION and SIMATIC controllers.

Note

You will find further information on cyclic data exchange with controllers in the SIMATIC FAQs.

- Linking a S7-400 CPU to a non-Siemens master as a DP slave (http://support.automation.siemens.com/WW/view/en/36749973)
- DP linking CPU 315-2DP (slave) and S7-400 (master) (http://support.automation.siemens.com/WW/view/en/6518822)

4.2.2 Linking a SIMOTION controller to a SIMATIC controller as a PROFIBUS DP slave

4.2.2.1 Linking by means of a GSD file

Procedure

The GSD files for the various SIMOTION hardware platforms are automatically also installed by the SIMOTION SCOUT installation.

If there is no SIMOTION SCOUT on the engineering PC, the GSD files must first be installed in STEP7 HW Config. You will find the corresponding GSD files on the SIMOTION SCOUT DVD "Add-on" in the respective device directory under Firmware and Version.

Table 4-1 GSD file

Device	Name of the GSD file
SIMOTION C	Si0480aa.gsd
SIMOTION D	Si0280ab.gsd (This file can be used for all SIMOTION D)
SIMOTION P	Si0380fa.gsd

After the GSD files have been installed by selecting **Options > Install GSD file...** in the menu, the corresponding SIMOTION devices will appear in the Hardware Catalog under **PROFIBUS DP > Additional FIELD DEVICES > PLC > SIMOTION**. These can be inserted into a DP master system from there.

Note

On SIMOTION controllers that are linked to a higher-level controller via a GSD file (separate projects), access is not possible with SIMOTION SCOUT and the access point S7ONLINE via a routed connection.

The alternative access point **DEVICE** in the SIMOTION SCOUT must be used for this! This also has the advantage of being able to go online simultaneously with the two projects to SIMATIC (S7ONLINE) and SIMOTION controller (DEVICE). However, it is not possible to go online to the SINAMICS_Integrated of a SIMOTION D controller if the link with the SIMOTION has already been routed. Only one network transition can be configured (e.g. PC/PG <Ethernet> S7-CPU <PROFIBUS> SIMOTION, but not <PROFIBUS_Integrated> SINAMICS_Integrated in addition).

Note

It is possible to access SINAMICS drives that are linked to a controller via a GSD file with SIMOTION SCOUT (or STARTER) via a routed connection with the exception of the SINAMICS_Integrated drive.

To do this, it is possible to set a network transition point using the S7 subnet ID with the setting **Target device > Online access ...**. Through a network node it is also possible to route to drives that have been inserted as single drives.

4.2.2.2 Linking using an I slave

Requirement

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

If these requirements are fulfilled, the SIMOTION controller can also be connected to the PROFIBUS DP master system of the higher-level SIMATIC as an I slave.

Procedure

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

Below you will find 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 a DP slave, e.g. SIMOTION C2xx
   Double-clicking on the desired PROFIBUS interface (e.g. DP2/MPI) in HW Config opens its properties. Select the **DP slave** option from the **Operating Mode** tab.
2. Configuring the local I/O addresses
   You can set the local I/O addresses and the diagnostics address on the **Configuration** tab.
3. Switch to the configured SIMATIC station that is to be DP master for the SIMOTION.
4. Creating an I slave
   Drag the station type "C2xx/P3xx/D4xx/D4xx-2-I-Slave" from the **Hardware catalog** window, "Preconfigured stations" folder, and drop it on the DP master system of the SIMATIC controller.
5. Linking an I-slave
   Double-click the I slave proxy to open the properties window. On the **Link** tab, assign the station that is to represent the intelligent DP slave (I slave). This displays all the stations that are already available in the project and that are potential link partners.

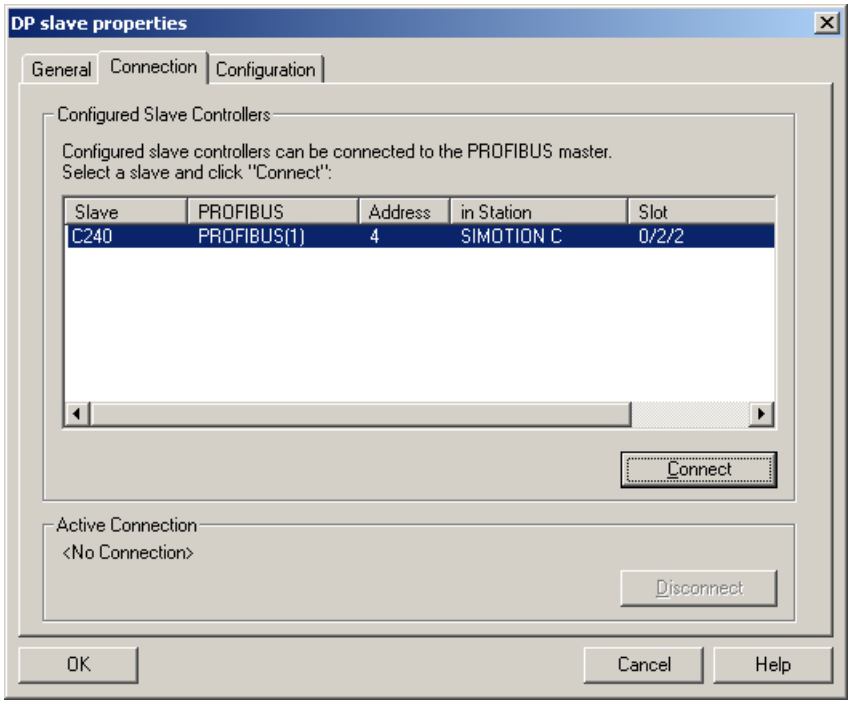


Figure 4-2 I-slave properties

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:

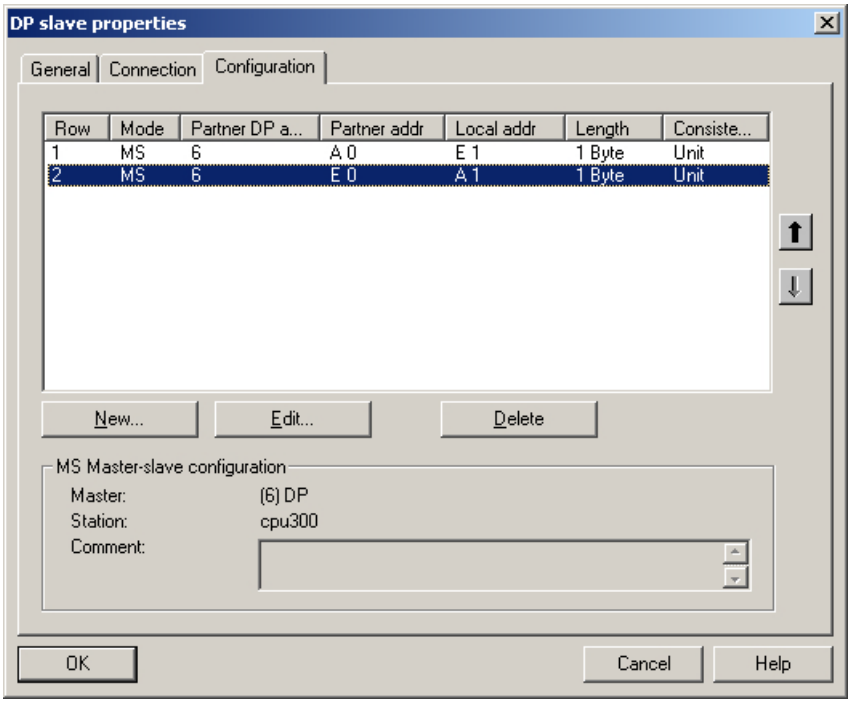


Figure 4-3 Properties - configuration

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

Note

For direct data exchange with a DP slave (slave-to-slave data exchange broadcast) or DP master (master-to-master data exchange broadcast), it would be necessary to select DX mode (Direct Data Exchange) mode here.

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

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

4.2.3 Linking a SIMATIC controller to a SIMOTION controller as a PROFIBUS DP slave

4.2.3.1 Linking by means of a GSD file

Procedure

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

Note

You will find the corresponding GSD files in Product Support: PROFIBUS GSD files for SIMATIC.

After the GSD files have been installed by selecting **Options > Install GSD file...** in the menu, the corresponding SIMATIC devices will appear in the Hardware Catalog under **PROFIBUS DP > Additional FIELD DEVICES > PLC > SIMATIC**. These can be inserted into a DP master system from there.

Note

SIMATIC controllers that have been connected to a higher-level controller by means of a GSD file, cannot be accessed with STEP 7 via a routed connection.

See also

PROFIBUS GSD files: SIMATIC (http://support.automation.siemens.com/ww/view/en/113652)

4.2.3.2 Linking using an I slave

Requirements

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

If these requirements are fulfilled, the SIMATIC controller can also be connected to the PROFIBUS DP master system of the SIMOTION controller as an I slave.

Procedure

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

Below you will find 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, for example, 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 I slave proxy to open the properties window. On the **Link** tab, assign the station that is to represent the intelligent DP slave (I slave). This displays all the stations that are already available in the project and that are potential link partners.

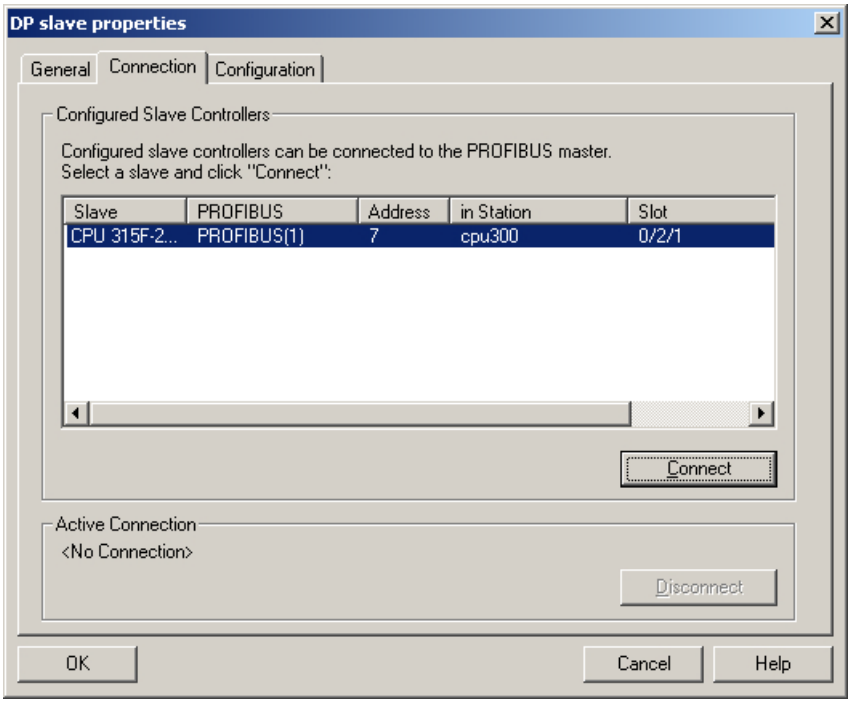


Figure 4-4 Properties - link

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

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

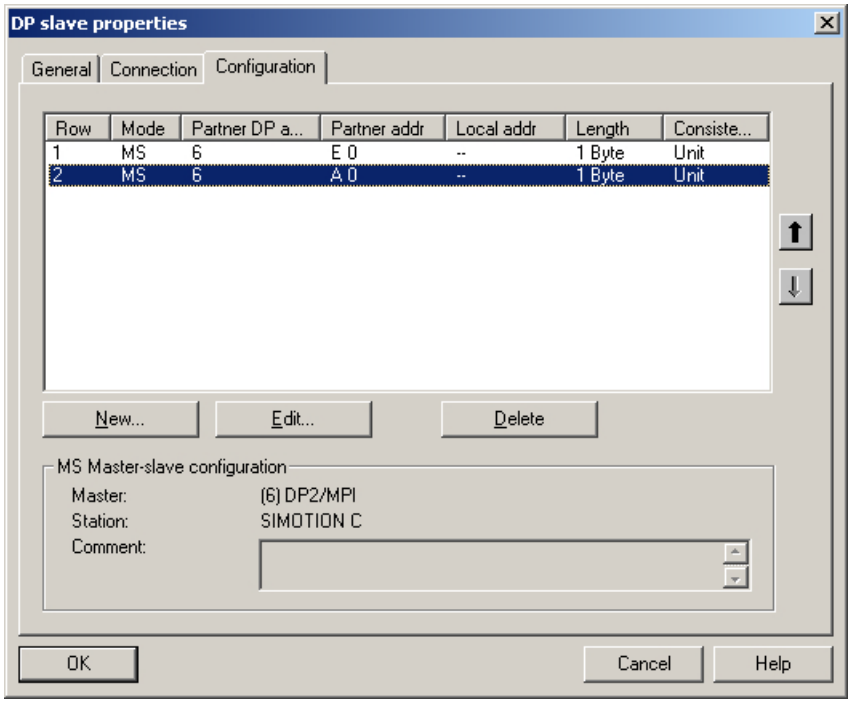


Figure 4-5 Configuration - address selection

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

Note

For direct data exchange with a DP slave (slave-to-slave data exchange broadcast) or DP master (master-to-master data exchange broadcast), it would be necessary to select DX mode (Direct Data Exchange) mode here.

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.

4.2.4 CPU-CPU communication using the S7 basic communication via PROFIBUS

4.2.4.1 Introduction

CPU-CPU communication

CPU-CPU communication using the S7 basic communication between a SIMATIC and a SIMOTION controller, is set up using the SFC65 (XSEND) and SFC66 (XRECEIVE) system functions on the SIMATIC side and the _Xsend() and _Xreceive() system functions on the SIMOTION side. This application communication is effected via PROFIBUS, or MPI.

Table 4-2 CPU-CPU communication

Protocol	SIMATIC device	Function	SIMOTION device	Function
PROFIBUS	S7-300 CPU S7-400 CPU	SFC65 (XSEND) SFC66 (XRCV)	SIMOTION C SIMOTION D SIMOTION P	_Xsend _Xreceive

PROFIBUS addresses are assigned in HW Config. This must also be passed as a parameter when the system function _Xsend() is called. That is, the parameters that are important for communication are determined by the user and passed in the function call. It is not necessary to configure the communication link in NetPro.

Note

For further information, refer to the online help of each system function.

You will find additional information in the FAQ for SIMATIC communication:

- CPU-CPU communication with SIMATIC controllers (http://support.automation.siemens.com/WW/view/en/20982954)
- S7 basic communication (http://support.automation.siemens.com/WW/view/en/28866132)

4.2.4.2 SIMOTION functions

The SIMOTION functions for a PROFIBUS connection are explained in more detail below. .

System functions Xsend() and Xreceive()

Table 4-3 Xsend()

```
RetVal_PB_Senden:=
    _xsend(
          communicationMode := PB_Senden_CommunicationMode,
          address := PB_Senden_Address,
          messageID := PB_Senden_MessageID,
          nextCommand := PB_Sender_NextCommand,
          commandID := PB_Senden_CommandID,
          data := PB_Sende_Daten,
          datalenght := PB_Sende_Daten_Laenge
          );
```

Example of the call of the SIMOTION function _xsend

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

Note

Application transmission and reception via S7 basic communication (_Xreceive() or _Xsend()) is only possible on a PROFIBUS interface of the SIMOTION controller if the **Programming, status/force, or other PG functions and non-configured communication connections possible** option is set.

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. For example, the value 1 is entered for interface X8 for a SIMOTION C2xx. The value 2 is entered for interface X9. If a SIMATIC S7 station is connected to X101 of a SIMOTION P, 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 D, 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 communication via MPI or PROFIBUS, a value of 0 is pre-assigned 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.

Table 4-4 Xreceive()

```
RetVal_PB_Empfangen:=
    _xreceive(
          messageID := PB_Empfangen_MessageID,
          nextCommand := PB_Empfangen_NextCommand,
          commandID := 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 useful data bytes after a successful call of the _xreceive function. The received useful data can be accessed via the data parameter.

PROFINET IO 5

5.1 PROFINET IO overview

5.1.1 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 many years of experience with the successful PROFIBUS DP and combines the normal user operations with the simultaneous use of innovative Ethernet technology concepts. This ensures the smooth migration of PROFIBUS DP into the PROFINET IO world.

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

Note

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

5.1.2 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.

5.1.3 PROFINET IO system

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

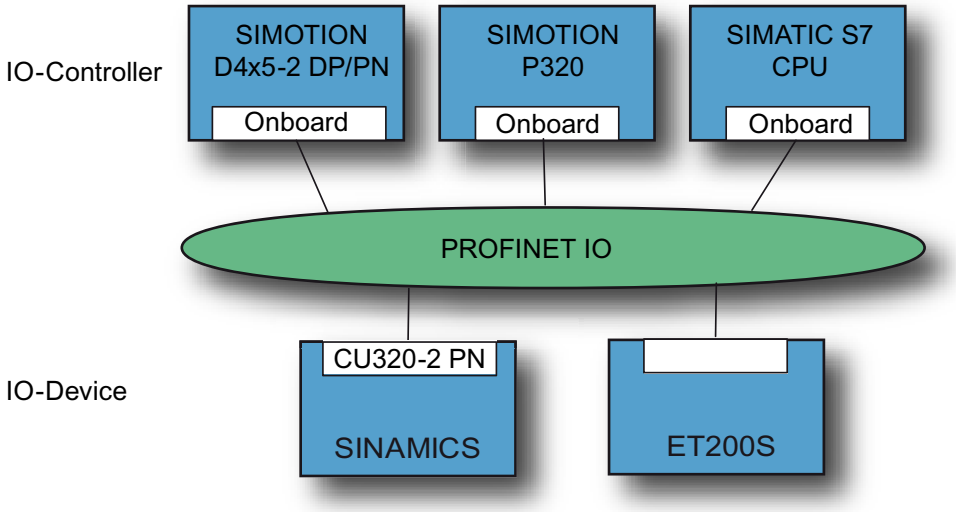


Figure 5-1 PROFINET overview IO Controller - IO Device

5.1.4 IO controller

The PROFINET IO controller has the same functions as the PROFIBUS DP master. The IO controller of, for example, a D4x5-2 DP/PN with onboard PROFINET interface exchanges data cyclically with the I/O devices assigned to it (PROFINET IO devices), such as the SINAMICS S120.

5.1.5 IO device

Distributed field devices such as I/O components (e g. ET200) or drives (e.g. SINAMICS S120 with CU320-2 PN) are referred to as IO devices. The function is comparable to a PROFIBUS DP slave.

See also

Creating an IO device (Page 143)

5.1.6 RT classes

5.1.6.1 RT classes for PROFINET IO

Description

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

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

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

RT

PROFINET RT uses the option of prioritizing Ethernet telegrams (as described in the Ethernet standard). For more detailed information, see PROFINET IO with RT (Page 47).

IRT

PROFINET IRT uses a time slot procedure (bandwidth reservation); i.e. 2 time slots occur. The IRT telegrams are transmitted in the first time slot; the standard Ethernet and RT telegrams in the second. By reserving the transmission bandwidth, the transmission of the IRT data is guaranteed for all load/overload situations. For all participating devices to know when the time slot begins, all the participating devices must be synchronized with PROFINET IRT.

In addition to the bandwidth reservation, a schedule for the cyclic telegrams is developed with consideration given to the topology. This makes it possible for the engineering system to determine the required bandwidth for each individual cable.

In addition to the synchronization of the transmission network with PROFINET IRT, the application can also be synchronized in the devices with IRT High Performance (isochronous application; e.g. position controller in SIMOTION or ServoSynchronousTask). This is an essential requirement for closing control loops across the network and isochronous switching of inputs and outputs in the network.

Note

Only PROFINET IRT High Performance is used for SIMOTION. Whenever PROFINET IRT is referred to in the rest of this document, this means IRT High Performance.

For more detailed information, see PROFINET IO with IRT (High Performance) (Page 49).

Comparing RT and IRT

Table 5-1 The major differences between RT and IRT

Property	RT	IRT (High Performance)
Real-time class	Real-time class 1	Real-time class 3
Transfer mode	Prioritization of cyclic RT telegrams using the so-called VLAN tag.	Bandwidth reservation optimized by the engineering system on the basis of topology information
Determinism	No Send and receive time points of the cyclical RT data are not precise; they can be delayed by standard Ethernet telegrams.	Yes Transmission and receiving times for cyclic IRT data are precisely defined and guaranteed for all kinds of topologies.
Isochronous application	Not supported	Supported
Hardware support using special Ethernet controller	No	Special hardware necessary (e.g. SCALANCE X200 IRT)

5.1.6.2 Send clock and update time

Description

With PROFINET the send clock and update time are distinguished. The send clock is the basic cycle clock for cyclic communication. The update time indicates the cycle in which a device is supplied with data.

Send clock

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

Update time

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

Relationship between the update time and send clock

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

5.1.6.3 Adjustable send clocks and update times

Description

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

Note

The following versions generally relate to the use of the position control cycle clock with servo or IPO task, unless otherwise specified. When using the Servo_fast with Servo_fast or IPO_fast Task, other restrictions apply under certain circumstances. If Servo_fast and IPO_fast are used, then the PROFINET must be operated isochronously.

Table 5-2 Adjustable send clocks and update times when using servo or Servo_fast

Send clock		**Factors (update time = factor * send clock)**	
		RT	**IRT**
"Even" range	125 µs *Note 4)*	IO devices cannot be used with RT	1 *Note 2)*
	250, 500, 1,000 µs	1, 2, 4, 8, 16, 32, 64, 128, 256, 512	
	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)	Cannot be used in front of SIMOTION V4.4 IO devices with RT *Note 1*)	1
Note 1)	**Mixed operation RT / IRT with SIMOTION < V4.4** Send cycles from the "odd" range can be used only if there is no IO device in the IO systems involved in the sync domain. As soon as there are IO devices with RT in a sync domain, it is only possible to set send clocks from the "even" range.		
Note 2)	**Settable factors and isochronous operation (isochronous application)** Some IO devices with IRT support the factors 2, 4, 8, and 16 in addition to factor 1. Where IO devices (e.g. ET200S IM151-3 PN HS, SINAMICS S120) are operated isochronously, it is usually only possible to set a factor of 1. In these cases, the mode for the update time must always be set to **Fixed factor** to ensure STEP 7 does not automatically adapt the update time. The update time than always matches the send clock.		

Send clock		Factors (update time = factor * send clock)	
		RT	IRT
Note 3)	Mixed operation RT / IRT as of SIMOTION V4.4 As of SIMOTION V4.4, send cycles from the "odd" range can be used only if there is no IO device with RT in the PROFINET IO systems involved in the sync domain. However, this mixed operation is only possible with SIMOTION D4x5-2 devices on the onboard PROFINET interface. The same factors are available as from the "even" range. The factors that can actually be set depends specifically on the set send clock. Restriction of Shared Device with RT / IRT as of SIMOTION V4.4: If a Shared Device is simultaneously used with PROFINET IO with RT and PROFINET IO with IRT, PROFINET IO with IRT can only be operated with send clocks from the "even" range.		
Note 4)	The send clock can only be used with D455-2 DP/PN on the onboard PROFINET interface together with Servo_fast as of SIMOTION V4.4. If this send clock is used, only the first port of the onboard PROFINET interface will be active, all other ports are deactivated and cannot be used.		

If there is no sync master configured in a PROFINET IO system (no PROFINET IRT), the send cycle clock for the respective PROFINET IO system can be set individually on the relevant IO controller. You can do this in the properties **<PROFINET Interface>** on the **PROFINET** tab under **Send cycle** or in the properties **PROFINET IO System** on the **Update time** tab. There is a default setting for the send clock of 1 ms. On the **IO cycle** tab, the down-scaling for the update time can be set via the mode **fixed factor** or **fixed update time** and the factor.

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

Possible modes for setting the update time

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

Note

It is recommended that you work with the **Fixed factor** setting.

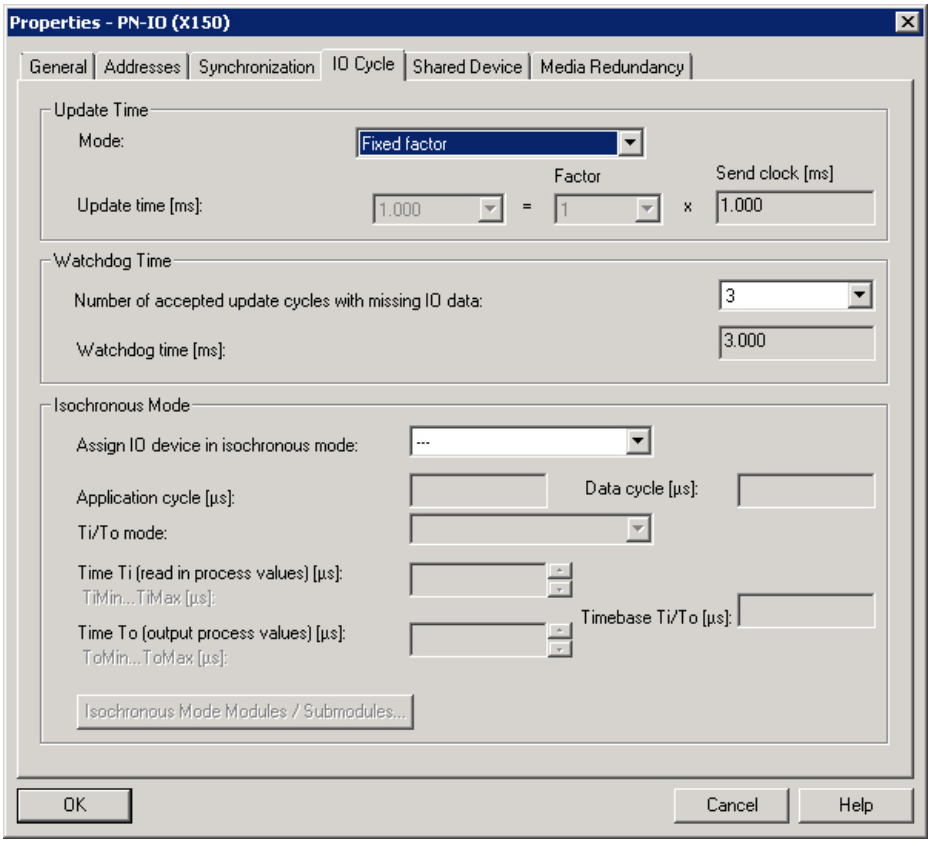


Figure 5-2 The influence of isochronous mode on setting the update time

5.1.6.4 PROFINET IO with RT

PROFINET IO with RT is the optimal solution for the integration of I/O systems without particular requirements in terms of performance and isochronous mode. This is a solution that uses standard Ethernet and commercially available industrial switches as infrastructure components. A special hardware support is not required.

Not isochronous

Although standard Ethernet and PROFINET IO with RT do not offer any synchronization mechanisms for devices, this does not mean that such arrangements are impossible. However, it does mean that isochronous data transmission is impossible, and there is no isochronous application for motion control as a result.

Data exchange

Communication via PROFINET IO with RT and IRT is based on the Ethernet frame and the MAC address. This means that cross-network communication via a router using RT and IRT is not possible. PROFINET IO message frames have priority over IT message frames in accordance with IEEE802.1Q. This ensures the availability of the real-time properties required in automation applications (e.g. for standard IOs).

Update time

The adjustable update time is in the range of 0.25 - 512 ms. The selected update time depends on the process requirements, the number of devices, and the amount of IO data. Given the improved performance offered by PROFINET compared to field buses, the bus cycle is generally no longer the variable which determines the system cycle.

5.1.6.5 PROFINET IO with IRT - Overview

Overview

PROFINET IO with IRT satisfies communication requirements which go beyond the sending of standard signals. As far as IRT is concerned, the jitters which may still be encountered with RT during communication are significantly reduced by synchronizing the network.

A time-slot procedure is required at a level above that of the Ethernet network. One time slot is reserved for the IRT telegrams and another slot is reserved for the RT and IP-based telegrams, in which the standard Ethernet communication (NRT (optional)) also runs. This type of approach requires all devices involved in IRT communication to be synchronized.

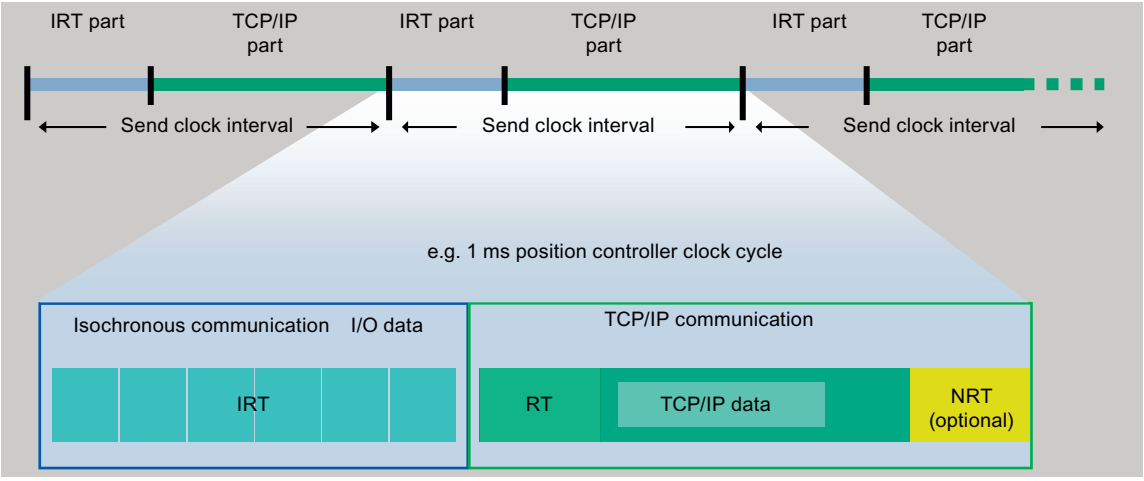


Figure 5-3 IRT Communication - Overview

See also IRT High Performance (Page 49) with optimized bandwidth reservation and scheduled IRT communication.

For PROFINET IO with IRT, all IRT devices are synchronized on a shared sync master. See also Isochronous operation and isochronous mode with PROFINET (Page 51).

5.1.6.6 PROFINET IO with IRT (High Performance)

The performance capability of motion control applications is significantly increased with PROFINET IO IRT (High Performance).

PROFINET is based on Ethernet technology, which is in turn based on point-to-point connections. Point-to-point connections support a significantly higher transmission rate when compared with arrangements based on parallel wiring. PROFINET uses 100 Mbps. Using this in conjunction with switching technology means that all connecting cables are decoupled from each other, enabling each cable to both transmit and receive at the same time.

Scheduling the telegram traffic for IRT High Performance enables data traffic to be optimized to a considerably higher degree, as only the bandwidth which is actually required is reserved.

IRT (High Performance) is particularly suitable for:

- The control and synchronization of axes via PROFINET IO
- A fast, isochronous I/O integration with short terminal-terminal times

Send clock

The send clocks that can be set are described in the table in Section Adjustable send clocks and update times (Page 45).

The actual send clock that can be used depends on various factors:

- The process; communication should be no faster than required. This reduces the bus and CPU loads.
- The bus load (number of devices and the amount of IO data per device)

- The computing power available in the controller
- The supported send cycles on the PROFINET devices involved in a sync domain.

The supported send clocks can be found in the corresponding manuals of the respective SIMOTION controls.

Isochronous application

Isochronous data transmission and an application synchronized with the bus system satisfy the requirements associated with demanding motion control applications. This makes it possible to close control loops via the bus system and achieve minimum guaranteed response times (terminal-to-terminal time response). In addition, a high-performance and isochronous connection to the application with low load on the application CPU is ensured.

Unlike standard Ethernet and PROFINET IO with RT, transmission of telegrams for PROFINET IO with IRT High Performance is scheduled.

Time-scheduled data transmission

Scheduling is the specification of the communication paths and the exact transmission times for the data to be transferred. The bandwidth can be optimally utilized through communication scheduling and therefore the best possible performance achieved. This requires the network topology to be configured, with the engineering system automatically calculating the communication schedule from the configured topology (see also Topology (Page 138)). The data relevant to PROFINET IO is transmitted by means of a download from HW Config to the IO controller. The highest determinism quality is achieved through the scheduling of the transmission times which is especially advantageous for an isochronous application connection.

Data exchange

PROFINET IO with IRT (high performance) only runs within a sync domain. These must not however be mixed in one IO system. In other words, a sync domain may consist of two or more IO systems which can all be synchronized with one another. Within the IO system, PROFINET IO with IRT (high performance) is then used.

5.1.7 Sync domain

A sync domain is a group of PROFINET devices synchronized to a common cycle clock. The sync slaves synchronize themselves with the cycle clock set by the sync master. The role of a sync master can assume a control system (IO controller) or a SCALANCE X200 IRT (IO device). A sync domain has just one sync master.

See also

Creating a sync domain (Page 128)

5.1.8 Isochronous operation and isochronous mode with PROFINET

Description

PROFINET IO with IRT is based on Ethernet with a higher-level time-slot procedure. This arrangement requires all bus interfaces involved in communication to be synchronized. In the case of PROFINET IO with IRT High Performance, a sync master transfers a synchronization message frame to which all sync slaves synchronize themselves.

Sync master, slaves, and domain

The sync master and sync slave device roles are assigned during the configuration. An IO controller or SCALANCE 200 IRT switches can be assigned a sync master role.

A sync domain can comprise PROFINET devices with IRT High Performance. PROFINET devices with RT may also be located at the ends (spur lines), but not between two PROFINET IO devices with IRT (High Performance).

A line (network) may only comprise IRT High Performance, and these must always be connected to each other directly.

Compatibility

Communication between and through different sync domains via PROFINET IO with RT is possible.

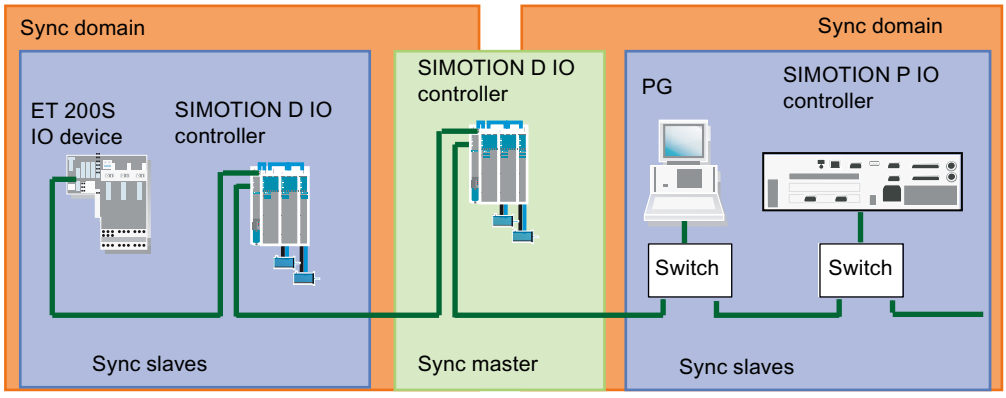


Figure 5-4 PROFINET isochronous mode

IRT-compatible switches, such as SCALANCE X204 IRT, are used in the structure. All devices involved in IRT communication are connected to each other directly. As the programming device in the center of the network is connected via a spur line, it does not interrupt the IRT path.

See also

Isochronous applications with PROFINET (Page 58)

5.1.9 Addressing of PROFINET IO devices

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

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

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

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

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

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

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

See also

Assigning device names and IP addresses to IO devices (Page 151)

5.1.10 Planning and topology for a PROFINET network

Planning guidance

Fundamental questions must be resolved before implementing the design of a PROFINET network. In this chapter, you will find broad guidelines to support you in defining requirements and creating planning documentation. Planning is an iterative process, i.e. some requirements influence each another and therefore necessitate changes in the overall plan.

Content of the planning documentation

Once the requirements have been specified and the planning process has come to an end, the following information should be available to you:

- System configuration
- Topology
- Selection of components
- Selection of transmission medium
- Connector selection
- Communication relationships
- Estimate of the data volumes to be transferred

Preliminary considerations and analysis during the planning stage

1. **Selecting the devices**
   Create a list of devices. The selection of devices is based, among other things, on the application class (conformance class), the time and communication requirements, the function, environmental influences, and the degree of protection.
2. **Position of the devices in the machine**
   The position of the devices in the machine has implications for the degree of protection, EMC, device dimensions and the cables used, e.g. whether fiber-optic cable should be used rather than copper cable. This in turn influences device selection.
3. **Defining the communication properties**
   The time requirements concerning the application (isochronous/cyclic) and communication via PROFINET must be defined, i.e. does communication take place in real-time (IRT/RT) or is it acyclic via TCP/IP or UDP/IP.

4. **Network planning (topology)**
   Specify the network topology (ring, star, line). Depending on the type of topology selected, the following must be taken into account: switches (with IRT capability), EMC, extent of network, WLAN (IRT not possible) and, if applicable, media redundancy MRP (possible with SIMOTION V4.3 and higher, and SINAMICS V4.5).
   - Set up your PROFINET in a point-to-point architecture where this is useful (for example, use a switch to branch off into a point-to-point topology downstream of a CPU).
   - In the case of PROFINET with IRT, a line structure with 64 IRT devices is permissible. The amount of data transmitted has certain implications. If longer message frame lengths are configured for each device, the possible number of devices per line may be reduced. However, this is detected early on during configuration with HW Config and signaled by means of an error message. The 64 IRT devices in the line are only applicable to PROFINET after V2.2.
   - Maintain a low interconnection depth for the switches. This will reduce the effect of a worst-case jitter scenario with RT communication.

Topology	
Star	If you connect communication nodes to a switch, you automatically create a star-shaped network topology. With this structure (unlike with other structures), if an individual PROFINET device fails, this does not automatically lead to the failure of the entire network. Only the failure of a switch causes the failure of devices downstream of the switch.
Tree	If you interconnect several star-shaped structures, you obtain a tree network topology.
Line	All the communication nodes are connected in series as a bus. If a switch fails, communication downstream of the failed switch is no longer possible. Devices with a 2-port switch must be used in order to set up a linear structure. Linear network structures require the least amount of cabling.
Ring	With V4.3 or higher, you can establish a ring topology for MRP or MRPD. For the ring topology you must define a redundancy manager and redundancy clients.

5. **Defining the communication relationships (logical assignment of partners)**
   Specify which communications partners are connected as well as the spatial and functional assignment of these partners.

6. **Defining the data volumes**
   Define the possible data volumes of the network nodes and those at the communication junctions.

   Note

   Communication within a PROFINET network should only be as fast as the technology of the system requires it to be and not as fast as technically possible.

7. **Defining the sending cycle and update time**
   The sending cycle is determined by the PROFINET device which has to be updated the most frequently. The update time is a multiple of the sending cycle and determines the PROFINET network load. The network load generated by PROFINET should always remain below 50% to allow reserve capacity for peak loads. Please note that the PROFINET network load increases in linear relation to the number of PROFINET devices and the sending cycle.

   Note

   Always configure the update times as required by the process, even if the bus system allows for a very large number of shorter update times. This reduces the PROFINET network load and the load on the PROFINET controller to what is strictly necessary.

8. **Checking the network load**
   In order to determine the network load, you must take into account the PROFINET network load as well as the network load generated by standard Ethernet devices. These can take the form of video cameras for monitoring the system or data servers for production data, for example. Ethernet devices with a low data volume such as engineering workstations or HMIs, for example, are normally non-critical. To prevent the PROFINET RT data stream from being compromised, you should, if necessary, adapt the network topology in cases where high network utilization by Ethernet nodes is anticipated. However, you also need to make sure you have sufficient bandwidth reserves for future expansion.

   Note

   Devices which exert a high IP load on the network should be located, where possible, in a separate area of the network. The diagnostics server and HMI server are examples of such devices.

9. **Connection to the company network**
   Various points must be considered if the automation system is to be connected to the company network. Normally, you should establish the connection via a router or a SCALANCE S with firewall to prevent unauthorized access. PROFINET IRT or RT communication via routers is not possible. You should only establish further connections (such as remote access or VPN, for example) in individual cases following consultation with the IT department.

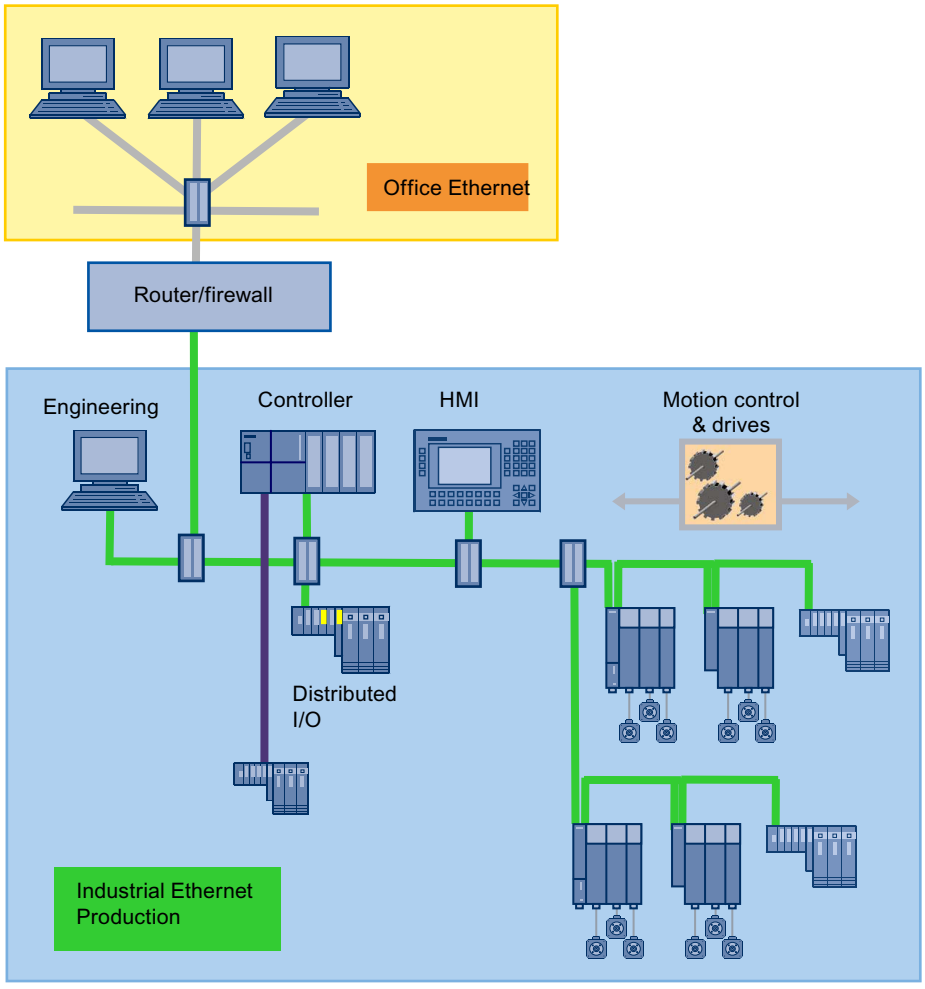


Figure 5-5 Optimized topology of a company network connection

Configuring the topology

Communication scheduling requires knowledge of the network topology. This includes information about interconnecting the individual devices to create a communication network.

Topology scheduling is only relevant for IRT High Performance. The topology editor that has been integrated in the hardware configuration enables user-friendly configuration of the network topology.

5.1.11 Isochronous applications with PROFINET

As with PROFIBUS, in the case of PROFINET IO with IRT High Performance the application can be synchronized with the transmission network's cycle clock. Distributed Motion Control applications and terminal-to-terminal response times of less than a millisecond require all PROFINET devices with IRT High Performance to be synchronized with a common time base.

Note

Isochronous mode for the application on the bus is only possible for PROFINET IO with IRT High Performance.

Configuration model for isochronous mode, V4.4 and higher

The configuration model for isochronous mode has changed as of V4.4. A selection must be made in the controller properties to specify whether isochronous IO devices are being operated. The isochronous task, the clock generator, must be selected. You can find a more detailed outline of the procedure for making settings in isochronous applications in the chapter titled Inserting and configuring the SINAMICS S120 (Page 145)

Procedure

When configuring isochronous applications, proceed as follows in HW Config:

1. Set the synchronization role for the IO controller to **Sync Master**. For SIMOTION (controller) V4.2 or higher, the RT class is automatically set to **IRT** and **High Performance**.
   (Double-click in HW Config on the PN interface of the IO controller. You can set the **Synchronization role** in the **Synchronization** tab of the Properties window.)

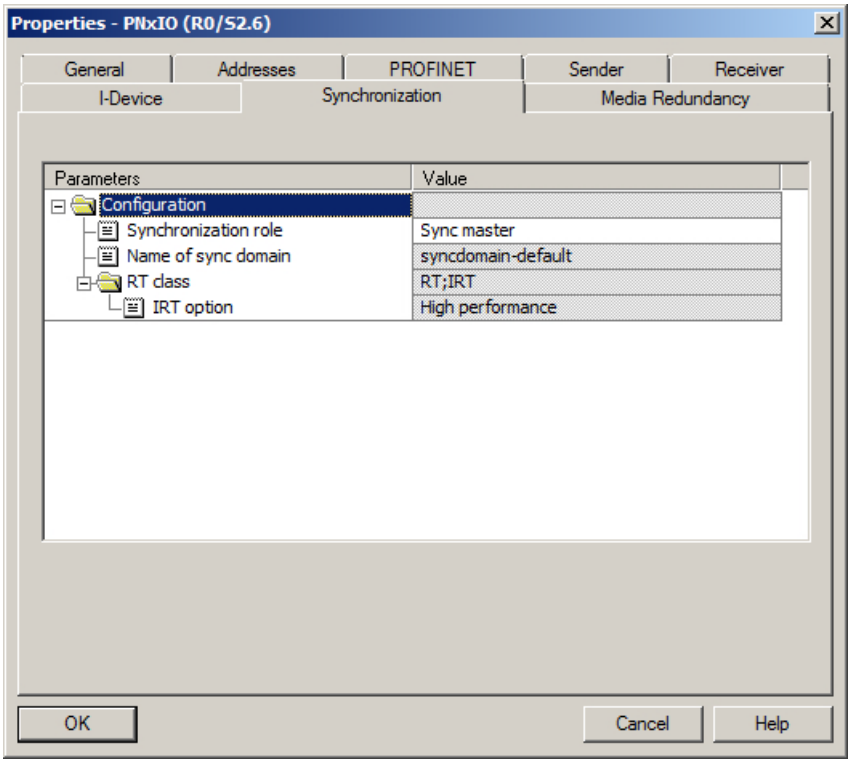


Figure 5-6 Setting the RT class

2. Set the synchronization role on the IO devices to **sync slave**. The RT class must be set to **IRT** and **High Performance**.
   (Double-click in HW Config on the PN interface of the IO devices. You can set the **Synchronization role** in the **Synchronization** tab of the Properties window.)

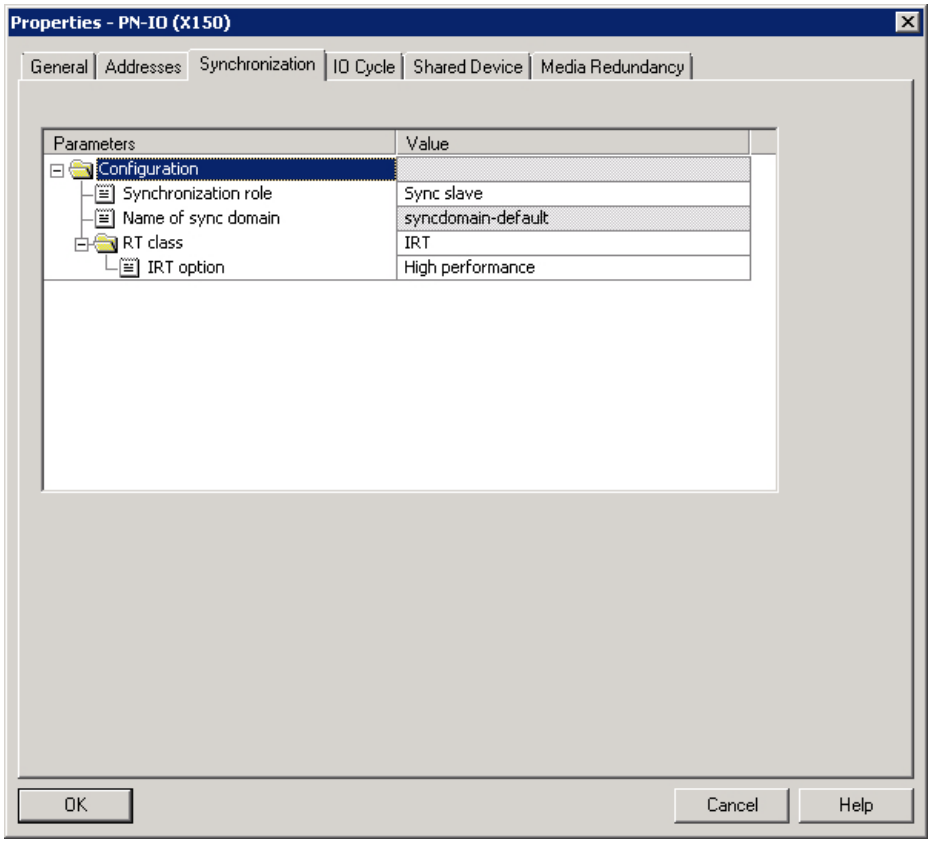


Figure 5-7 Setting IO device synchronization to sync slave and setting IRT High Performance.

3. Select the task (**Servo_fast** or **Servo**) in the **Isochronous tasks** tab in the controller properties. In general, **Servo** is used. The reference cycle clock is shown grayed out. The checkbox must be set for the following configurations:
   - Operation of isochronous IO devices (e.g. drives).
   - Configuration of exchange broadcast data. This is always isochronous.
   - Use of an iDevice with isochronous IO data.

   If two PN interfaces (e.g. X150 and X1400) of the controller are used with isochronous IO data, i.e. both checkboxes in the **Servo** field are activated, this is referred to as "coupled I/O" or "coupled IO systems". The two IO systems are then synchronized with one another.

Note

The checkbox must **not** be set if no isochronous IO data is configured. During compilation, inconsistent settings are displayed with an error message.

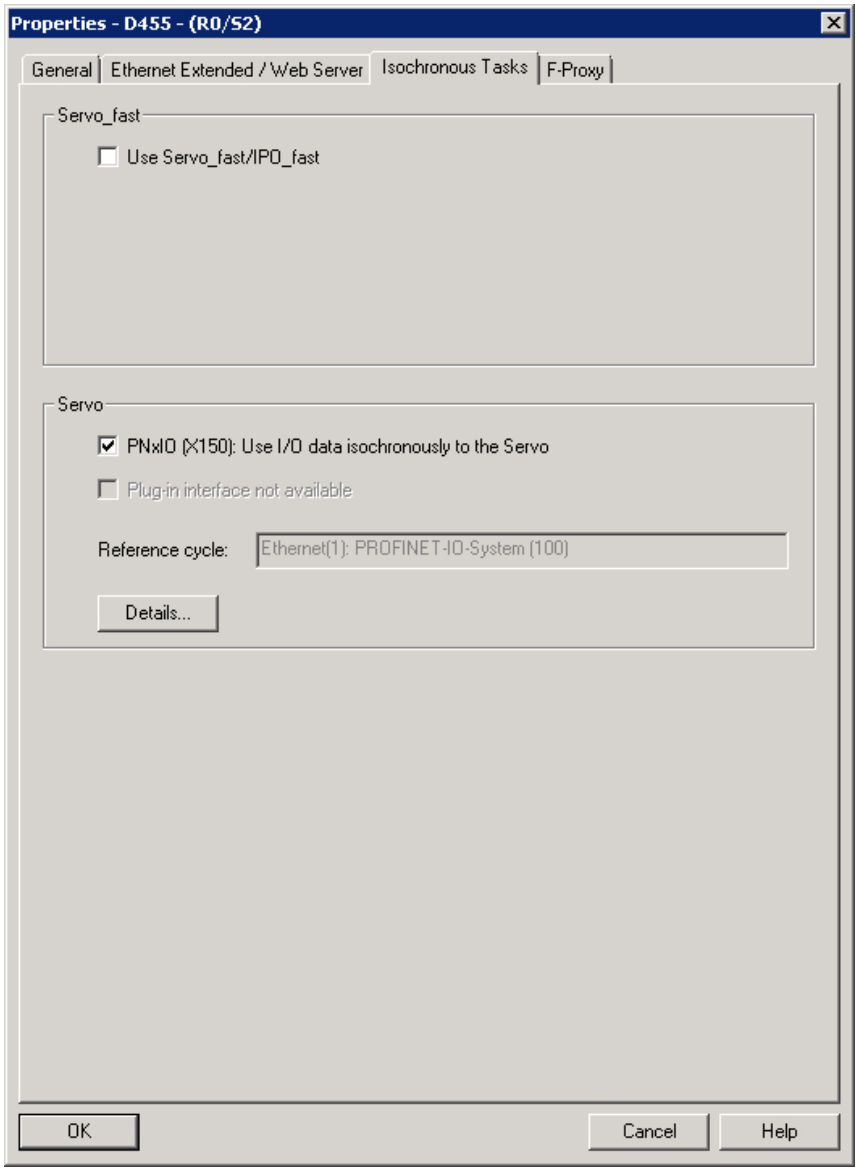


Figure 5-8 Isochronous IO devices are operated on the controller

4. On the devices, set the **Update time** mode to **Fixed factor** and select the cycle clock (servo or Servo_fast) under **Assign IO device isochronously**. Here, you can select the cycle clock that was set under "Isochronous tasks" in the previous step. **Servo** is used in the example. (Double-click in HW Config on the PN interface of the IO devices. You can configure the settings in the **IO Cycle** tab of the Properties window.)

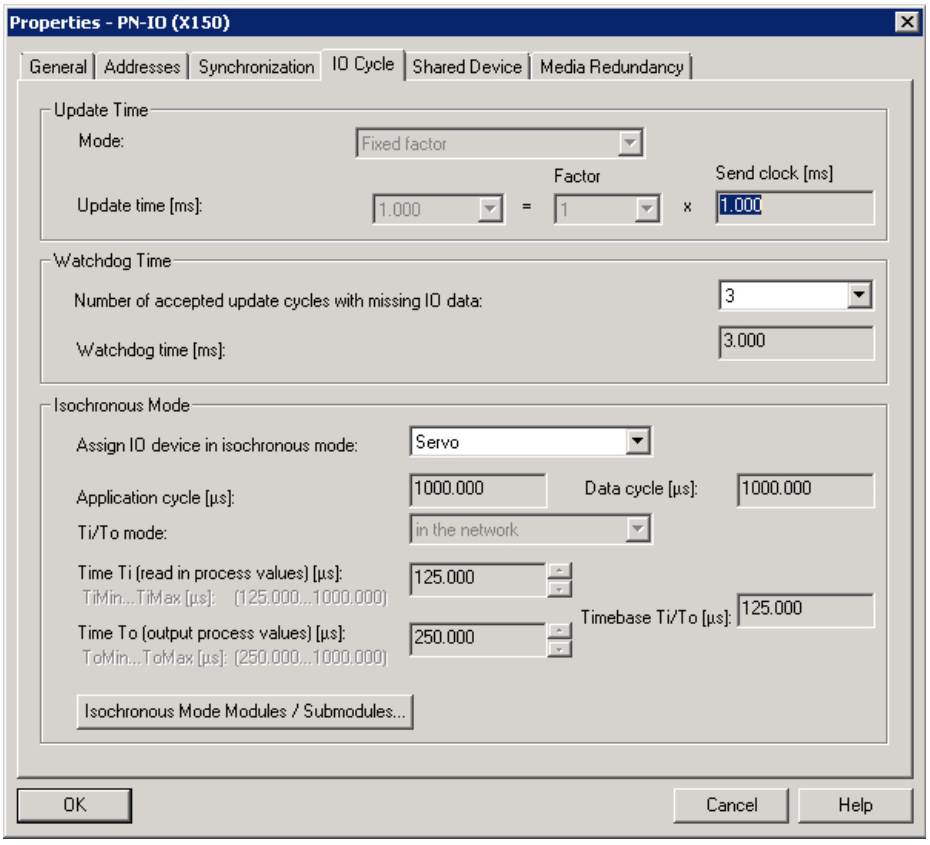


Figure 5-9 Assigning IO cycle parameters

5.1.12 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 356).

5.1.13 Shared device

Shared device functionality

Large systems or systems spread out over a large area frequently make use of numerous IO controllers. It can, therefore, happen that sensors that are physically near one another must supply data to different IO controllers. Previously, this could be solved using multiple IO devices which were assigned to the different IO controllers. The shared device functionality makes it possible to divide up the submodules of an IO device between different IO controllers in order to save one or more interface modules.

Further application

Safety engineering is required in a system for certain system sections. Therefore, in addition to the standard CPU, an F-CPU is used which ensures that the critical system sections are shut down safely. With the shared device function, it is possible to assemble an IO device from F modules and standard modules and to assign the individual modules to either the F-CPU or the standard CPU accordingly.

The application with an F-CPU is described briefly in the PROFIsafe chapter (Page 303).

Principle

Access to the submodules of the shared device is divided between the individual IO controllers. Each submodule of the shared device can be assigned exclusively to one IO controller. The individual submodules are assigned in HW Config.

5.1.14 iDevice

iDevice functionality

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

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

Shared iDevice (V4.4 or higher)

As of SIMOTION V4.4, an iDevice of the PN-IO interfaces can be operated with two higher-level controllers. This functionality is referred to as "shared iDevice" and is comparable with the shared device. A configuration example for a shared iDevice can be found in the chapter Shared iDevice and PROFIsafe (Page 310).

See also

PROFINET IO and I device (Page 176)

5.2 Properties and functions of PROFINET IO with SIMOTION

5.2.1 Introduction

Requirement

To work with SIMOTION using PROFINET IO, you must have at least one PN interface available. This can either already be integrated within the controller or inserted via an (additional) option board.

Connection possibilities:

- SIMOTION D4x5 with Option Board CBE30
- SIMOTION D4x5-2 DP/PN
- SIMOTION D4x5-2 DP/PN with CBE30-2 (from V4.3)
- SIMOTION P350 PN or SIMOTION P320-4
- SIMOTION D410 PN/SIMOTION D410-2 DP/PN
- SIMOTION C240 PN

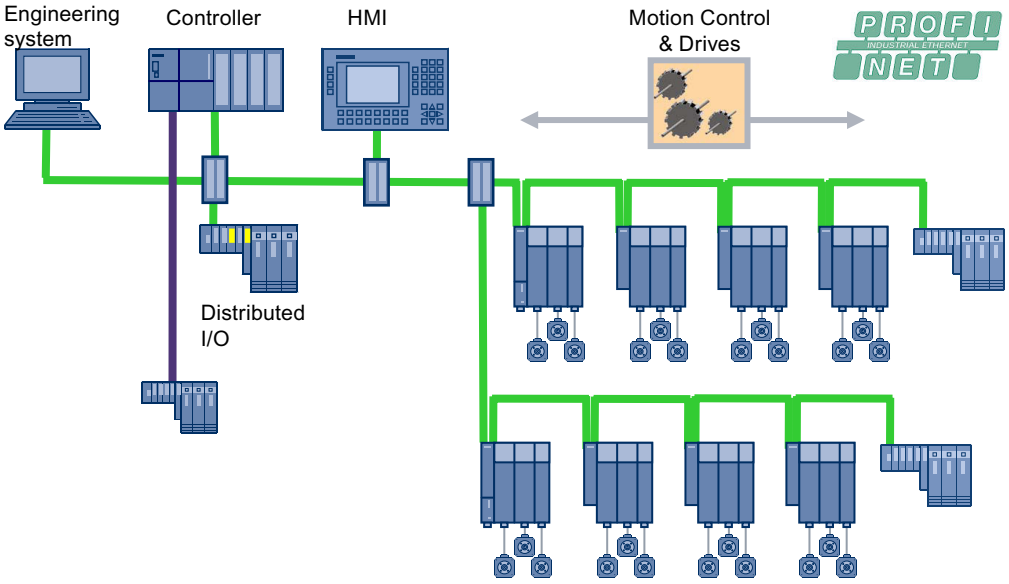


Figure 5-10 System topology with PROFINET

The SIMOTION controllers as PROFINET IO controllers support the simultaneous operation of:

- IRT High Performance - isochronous realtime Ethernet
  - Operation of IRT I/O (e.g. ET200S HS for IRT High Performance)
  - Operation of a SINAMICS S120 as an IRT IO device
  - Data exchange between IO controllers via IRT High Performance (e.g. distributed synchronous operation)
- RT - realtime Ethernet
  - Operation of RT - peripherals (e.g. ET 200S, ET 200pro)
  - ASi link via IE/AS interface link PN IO for the PROFINET IO gateway to AS interface
  - SINAMICS as RT IO device
- TCP/IP, UDP, HTTP, … standard Ethernet services

Note

With a mixed operation of PROFINET IRT and PROFINET RT, one must be aware that the IRT IO devices have to be connected together directly; i.e. no IO devices which PROFINET IRT does not support are allowed to be connected between two IRT devices The IO device that is not IRT-compatible must be located at the end of the IRT line so that IRT operation is not interrupted.

Note

With SIMOTION SCOUT, it is possible to access a maximum of 10 nodes ONLINE simultaneously. If you have installed SIMATIC NET, it is possible to establish more connections. The exact number is based on the available resources of the network adapter.

Note

For a PROFINET IO device that is connected to a SIMOTION controller, max. submodule sizes are defined. This limit is especially important if a SIMATIC controller is configured as an I-device for a SIMOTION controller . Therefore, please note the possible quantity structures (Page 94).

PROFINET V2.2/V2.3

SIMOTION SCOUT supports PROFINET V2.2 and is compatible with V2.3. Older versions are no longer supported as standard in the hardware and must be changed over.

When inserting SIMOTION controllers and/or SINAMICS drives, only PROFINET V2.2 versions are inserted in SIMOTION SCOUT. If you want to configure older versions, you need to explicitly add the hardware as PROFINET V2.1 in HW Config.

PROFINET V2.3 for SIMOTION D455-2 V4.4

SIMOTION D455-2 DP/PN supports PROFINET V2.3. In particular, the following functions are available:

- Fast Send Clock
  Isochronous mode with send clock ≥ 125 µs
- Dynamic Frame Packing (DFP)
  With DFP, the cyclic data of multiple IO devices is summarized in one Ethernet frame.

See also

PROFINET V2.3 Performance Upgrade (Page 107)

5.2.2 Cycle clock scaling

5.2.2.1 Cycle clock scaling with PROFINET IO on SIMOTION devices

Description (PROFINET IO with IRT High Performance)

An isochronous application (e.g. position controller) on an IO controller must be synchronized with the send clock for IRT High Performance. It can, however, be synchronized with a multiple of the send clock of the data. This multiple is designated as CACF (Controller Application Cycle Factor). Cycle clock scaling is configured in SCOUT with the execution system via **Expert > Set system cycle clocks** in the shortcut menu or via **Set system cycle clocks** from the shortcut menu of the highlighted SIMOTION controller in the project navigator.

Example: The data on the network is transferred with a send clock time of 1 ms. However, the servo should run with 2 msec. Therefore, the application cycle must be 2. For this purpose, set the ratio 2 at **Set system cycle clocks** for the servo.

The CACF is set on the IO device, see e.g. Inserting and configuring the SINAMICS S120 (Page 145).

V4.2 and higher, two servo cycle clocks (servo, Servo_fast)

With V4.2 and higher, Servo_fast is introduced as a second position control cycle clock for fast applications. The second servo cycle clock enables you to operate two bus systems in different application cycle clocks (PROFINET and PROFIBUS). An assigned servo cycle clock and IPO cycle clock are available for each of the two application cycle clocks. This enables you to divide your application into a slow section (servo and IPO) and a fast section (Servo_fast and IPO_fast). If a Servo_fast is configured, it is linked to the send clock of PROFINET IRT at a ratio of 1:1. The servo cycle clock must be set as a multiple of Servo_fast (at least 2x Servo_fast). In this case, the CACF continues to be the factor between the send clock/ Servo_fast and servo.

The functions described below mainly relate to configurations without the option of a second servo (Servo_fast).

Note

You can find a detailed description of the Servo_fast option in the *SIMOTION SCOUT Basic Functions Manual*, Chapter 6.

With V4.3 or higher, two PROFINET IO interfaces with different cycle clocks (SIMOTION D4x5-2 DP/PN with CBE30-2)

With V4.3 or higher, you can operate two PROFINET IO interfaces with different cycle clocks. With V4.2, it was only possible to operate one interface with PROFINET IO and one with PROFIBUS DP in different cycles (see above).

For the rules that apply when using Servo_fast, see Two PROFINET IO interfaces (Page 77).

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 SIMOTION D and an isochronous DP master interface always run in accordance with the servo cycle clock.
- For a SIMOTION P350/C240, the isochronous DP master interface always runs simultaneously with the servo cycle clock.
- For drives (e.g. S120) which are connected via SINAMICS Integrated or as an external drive unit, the servo cycle clock must **always** be counted in the first send clock. This means that down-scaling is possible, although the position controller must be counted in a send clock.

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

- The servo cycle clock/Servo_fast cycle clock is only synchronous with the send clock if the following conditions apply:
  - At least one IO device is operated isochronously.
  - At least one I-device is operated isochronously.
  - One controller-controller communication is configured.

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.
- If a Servo_fast is configured, then the CACF = 1 for Servo_fast and Servo is also fixed and cannot be changed (not configurable).
- Cycle clocks for SINAMICS Integrated and isochronous DP master interfaces must run simultaneously with the servo cycle clock.

5.2.2.2 Cycle clock scaling for IO accesses

Description of PROFINET IRT data transmission

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

- PROFINET IO IRT data is always read at the beginning of the servo cycle clock and written at the end of the servo cycle clock.
- PROFINET IO RT data is read at the beginning of the IPO or IPO2 cycle clock and written at the end of the IPO cycle clock.
- At the end of a IPOSynchronousTask, the process image is output with the next possible servo (Data Out) (= response-time-optimized). If the position control cycle clock is down-scaled to the IPO cycle clock (position control < IPO), this can lead to the data being output one or more position control cycle clocks earlier or later within an IPO cycle clock, if the I/O accesses are performed via the IPOSynchronousTask. This is the case if the runtime for the IPO cycle clock is not constant and, as a result, data is transmitted earlier or later on the bus with a faster position control cycle clock.
- At the end of the position control execution level, the process image of the ServoSynchronousTask is output with the next possible bus cycle clock (= response-time-optimized).
- If the PROFINET cycle clock is down-scaled to the position control cycle clock (PROFINET <= servo), the data is always output in the first PROFINET cycle clock.
- For PROFIBUS, the data is always output with the first bus clock cycle, since the servo priority class must always be finished with the first bus clock cycle. In case of a different runtime of the servo priority class in the individual cycles, the terminal-terminal time may vary as a result.

If an always constant response time is to be achieved instead of a response-time-optimized behavior, the following must be set:

- For PROFIBUS:
  - A reduction ratio servo: IPO = 1 : 1 so that the I/O accesses from the IPOSynchronousTask are always implemented in isochronous mode.
  - Comment: IO accesses from the ServoSynchronousTask are always isochronous for PROFIBUS
- For PROFINET:
  - A reduction ratio bus clock cycle: Servo: IPO = 1 : 1 : 1 so that the I/O accesses from the IPOSynchronousTask are always implemented in isochronous mode
  - A reduction ratio bus clock cycle: servo = 1 : 1 so that the I/O accesses from the ServoSynchronousTask are always implemented in isochronous mode

Note

With I/O access, cycle clock down-scaling can result in an offset (fixed dead time) for the IPO of a send cycle clock.

5.2.2.3 Bus cycle clocks that can be adjusted for cycle clock scaling to SIMOTION devices

Overview of the possible bus cycle clocks

	PROFIBUS[2)]	PROFINET IRT High Performance	PROFINET IRT High Performance	Servo Servo_fast
	Minimum	Minimum	Maximum	Minimum
SINAMICS S120 CU310-2 PN	1.0 ms	0.25 ms	4.0 ms	0.25 ms
SINAMICS S120 CU320-2 PN	1.0 ms	0.25 ms	4.0 ms	0.25 ms
SINAMICS S110 CU305 PN	1.0 ms	1.0 ms	4.0 ms	1.0 ms
SINAMICS S120 CU320-2 DP with CBE20	1.0 ms	0.25 ms	4.0 ms	0.25 ms
SINAMICS S120 CU320-2 PN with CBE20	1.0 ms	0.25 ms	4.0 ms	0.25 ms
SINAMICS S120 CU320-2 DP with CBE25	1.0 ms	0.125 ms	4.0 ms	0.125 ms
SINAMICS S120 CU320-2 PN with CBE25	1.0 ms	0.125 ms	4.0 ms	0.125 ms
C230-2	1.5 ms	-	-	1.5 ms
C240 PN	1.0 ms	0.5 ms	4.0 ms	0.5 ms
C240	1.0 ms	-	-	0.5 ms
D410 PN	-	0.5 ms	4.0 ms	2.0 ms
D410 DP	2.0 ms	-	-	2.0 ms
D410-2 DP	1.0 ms	-	-	0.5 ms[3)]
D410-2 DP/PN	1.0 ms	0.25 ms	4.0 ms	0.5 ms[3)]
D425	2.0 ms	0.5 ms	4.0 ms	2.0 ms
D425-2 DP	1.0 ms	-	-	0.5 ms
D425-2 DP/PN	1.0 ms	0.25 ms	4.0 ms	0.5 ms
D435	1.0 ms	0.5 ms	4.0 ms	1.0 ms
D435-2 DP	1.0 ms	-	-	0.5 ms
D435-2 DP/PN	1.0 ms	0.25 ms	4.0 ms	0.25 ms[1)]
D445/D445-1	1.0 ms	0.5 ms	4 ms	0.5 ms
D445-2 DP/PN	1.0 ms	0.25 ms	4.0 ms	0.25 ms[1)]
D455-2 DP/PN	1.0 ms	0.125 ms [4)]	4.0 ms	0.125 ms [1)]
P350-3	1.0 ms	0.25 ms	4.0 ms	0.25 ms
P320-3	-	0.25 ms	4.0 ms	0.25 ms
P320-4 PN	-	0.25 ms	4.0 ms	0.25 ms
P320-4 DP/PN	1.0 ms	0.25 ms	4.0 ms	0.25 ms

	PROFIBUS[2)]	PROFINET IRT High Performance	PROFINET IRT High Performance	Servo Servo_fast
	Minimum	Minimum	Maximum	Minimum
ET200S High Speed	-	0.25 ms	4.0 ms	0.25 ms
ET200SP High Speed	-	0.125 ms [4)]	4 ms	0.125 ms

1) Explanation:

- 0.5 ms in conjunction with SINAMICS S120 (incl. SINAMICS Integrated/CX32-2)
- 0.25 ms in conjunction with SERVO_fast and IPO_fast (D455-2 DP/PN: 0.125 ms)

2) Explanation:

- The specifications apply to "external" PROFIBUS interfaces and not to PROFIBUS Integrated for SIMOTION D. PROFIBUS Integrated for SIMOTION D: minimum 0.5 ms

3) Explanation:

- 1 ms when using the TO axis and the integrated drive control

4) Explanation:

- A general release of the functionality is planned for SIMOTION D455-2 DP/PN, as soon as the release for the I/O system ET200SP HighSpeed and the drive system SINAMICS S120 (CU320-2 with CBE25) is complete.

Cycle clock scaling with PROFINET IO

As of V4.3, it is possible to down-scale the PROFIBUS cycle clock to the position control cycle clock. Down-scaling is only permitted if PROFINET with IRT has not been configured. It is also possible to down-scale the PROFINET send clock to the PROFIBUS cycle clock. For example, PROFINET send clock = 0.5 ms and PROFIBUS cycle clock = position control cycle clock = 1 ms.

The PROFIBUS cycle clock can be operated relative to the PROFINET send clock at a ratio of 1:1 to 16:1. The table below shows the possible ratio settings for the system cycle clocks based on the DP cycle of the SINAMICS_Integrated or PROFINET send clock.

Table 5-3 Ratios of the system cycle clocks (when one servo is used)

Cycle clock name	Adjustable factors	Reference cycle clock
PROFIBUS DP bus cycle clock	1, 2, 3, 4, 6, 8, 10, 12, 14, 16	PROFINET IO send clock
Servo	As of V4.3: 1, 2, 3, 4, 8[1)] V4.2: 1	PROFIBUS DP bus cycle clock
IPO	1, 2, 3, 4, 5, 6	Servo
IPO_2	2, 3, 4, 5, …, 64	IPO

[1)] Always "1" if PROFINET with IRT has been configured

Table 5-4 Ratios of system cycle clocks (two servos)

Cycle clock name	Adjustable factors	Reference cycle clock
Servo_fast	1	Send clock of the onboard PROFINET IO interface (X150)
IPO_fast	1, 2, 4	Servo_fast
Servo	1	PROFIBUS DP bus cycle clock and send clock of the optional second PROFINET IO interface (CBE30-2)
IPO	1, 2, 4	Servo
IPO_2	2, 3, 4, 5, ... 64	IPO

Ratios of the system cycle clocks when using Servo_fast and servo

- PROFIBUS DP bus cycle clock = N x send cycle clock of the onboard PROFINET IO interface X150
  N = 2, 4, 8, 16, 32, 64 (N = 2 only for send cycles > 250 µs)
  (for < V4.4: N = 2, 4, 8, 16 for all send cycle clocks)
- Send cycle clock of an optional second PROFINET IO interface (CBE30-2) = PROFIBUS DP bus cycle clock
- IPO ≥ IPO_fast

5.2.3 Task system and time response

5.2.3.1 Overview of SIMOTION task system and system cycle clocks

Overview

If IRT data is transmitted via the bus using PROFINET IO, the cycle clock execution times fall between reading and writing the data (e.g. axis data), depending on which task in the execution system the application is executed in. You can find examples of applications in different tasks (execution levels) in the chapters that follow.

5.2.3.2 BackgroundTask, MotionTask, and IPOSynchronousTask

MotionTask/BackgroundTask

The data is transmitted via the bus using PROFINET IO with IRT High Performance, and accepted by the communication interface at the start of the position control cycle clock. The logic signals are generally evaluated in a MotionTask or BackgroundTask. Here, a distinction is made as to which machine function is activated; for example, "position-controlled traversing of axis". The traversing profile required is counted in the next IPO cycle clock. Based on the position setpoints determined here, the speed setpoints for controlling the axis are calculated in the next position control cycle clock. These are transmitted to the drive in the next cycle, via PROFINET IO with IRT High Performance.

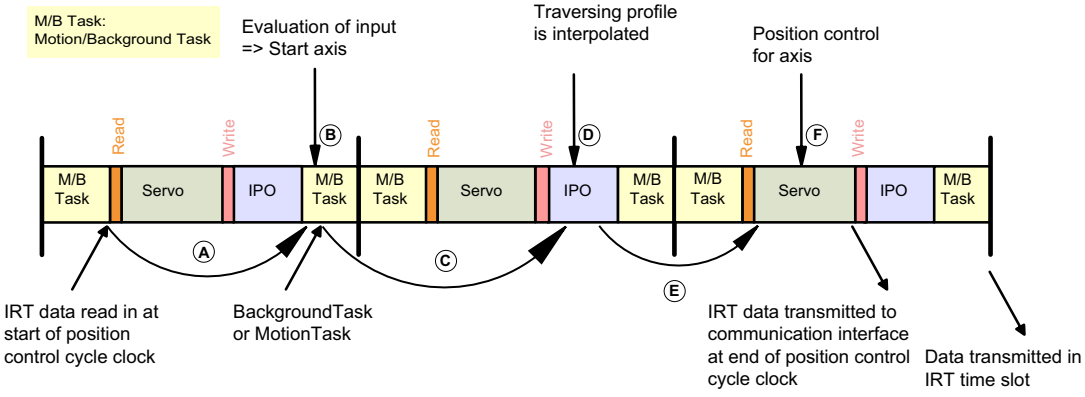


Figure 5-11 Logic evaluation for an axis in the BackgroundTask or MotionTask

Constraints

The option of a second servo clock is not activated, i.e. no Servo_fast and IPO_fast. The bus cycle clock, servo clock and IPO cycle clock are set to a 1:1:1 ratio. Other ratios may result in longer response times. With a 1:1:2 ratio, the IPO execution may be extended to two position control cycle clocks, which can lead to the response time increasing by one position control cycle clock.

Additionally, the BackgroundTask may be processed over several position control cycle clocks, meaning that the system is unable to make sure the data is evaluated in the first position control cycle clock. This may also lead to an increased response time.

Assigning the variables to a process image has an impact on the response time as well. The process images are made available to other tasks or to the communication interface at the end of the respective task, rather than after the variables are updated.

IPOSynchronousTask

In order to optimize the time response and enable synchronous triggering of actions (e.g. starting axes simultaneously), in the IPOSynchronousTask it is possible to process the part of the application that triggers axis commands. If this option is used, it is counted before the IPO. In this way, the axis command can be issued before the IPO is executed, and the resulting position setpoint then calculated in the IPO. Based on this, the speed setpoint for the drive is calculated in the next position control cycle clock. Once the position control cycle clock has finished, the data is passed on to the communication interface and transmitted in the next PROFINET IRT send clock. Unlike processing in a MotionTask/BackgroundTask, where the response time equals the maximum BackgroundTask runtime + one IPO cycle clock + one position control cycle clock, in this case you can be assured that the response time will be one IPO cycle clock + one position control cycle clock until new data is output.

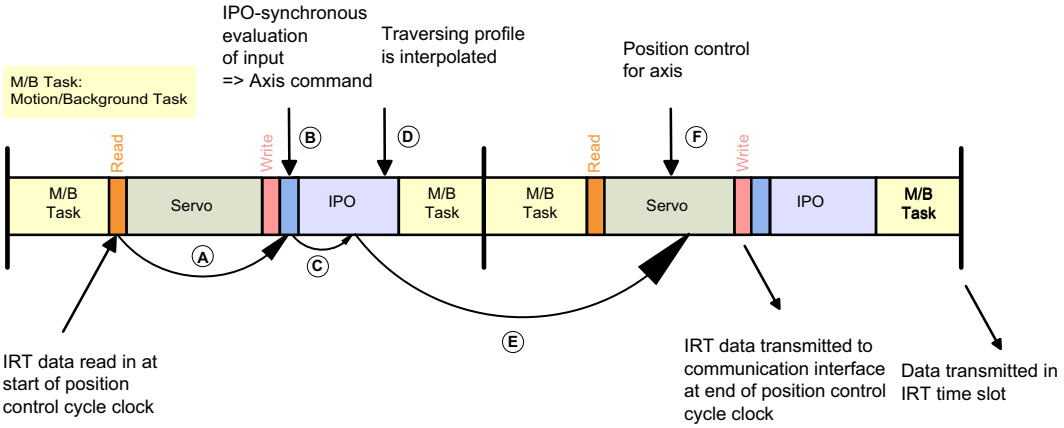


Figure 5-12 Logic evaluation for an axis in the IPOSynchronousTask

5.2.3.3 ServoSynchronousTask

ServoSynchronousTask

It is possible to optimize the time response even further and reduce the response time to one servo cycle clock. This option can be used for high-speed actual-value synchronous operations, e.g. flying knife/shear. Within this context, the part of the application that triggers axis commands for selected axes is processed in the ServoSynchronousTask. Additionally, the IPO part of the system for the axes involved is counted before the position controller in the servo task. In this way, the speed setpoints may be transmitted as soon as the next IRT time slot.

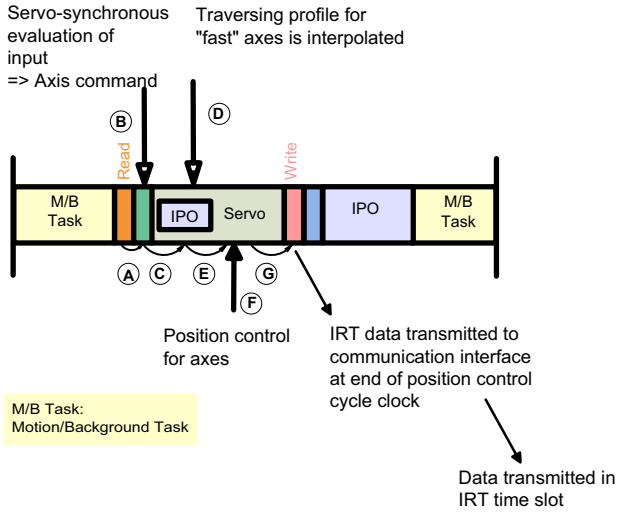


Figure 5-13 Logic evaluation for an axis in the ServoSynchronousTask

Boundary conditions

Using this function increases the CPU load and, therefore, the position control cycle clock; for this reason it should only be used when necessary.

Activating

This feature must be activated explicitly for the axes in SIMOTION SCOUT as part of axis configuration.

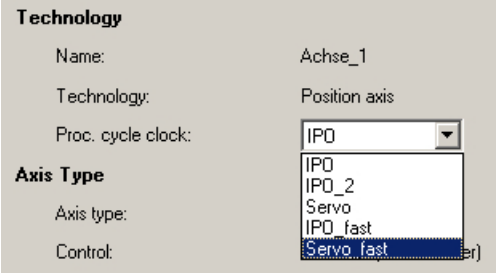


Figure 5-14 Determining the processing cycle clock for the axis

If the option of a second servo cycle clock is configured, then the axis can also be assigned to the Servo_fast processing cycle clock if this is linked to the faster bus.

Position control

The position controller responds within one position control cycle clock. The data is read out from the communication interface at the start of the position control cycle clock. The position controller is counted in the position control cycle clock. The new speed setpoints are copied to the communication interface at the end of the position control cycle clock and, therefore, transmitted in the next IRT time slot.

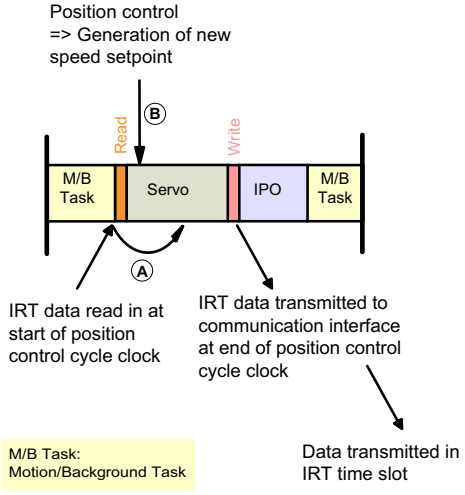


Figure 5-15 Position control time response with ServoSynchronousTask

5.2.3.4 Fast I/O processing in the ServoSynchronousTask

Fast I/O processing in the ServoSynchronousTask

The evaluation of quick I/Os – e.g. ET200S High Speed – is performed in the ServoSynchronousTask. resulting in a system response time of one cycle. Due to the terminal-terminal response, there is a delay of Ti + servo cycle clock + To.

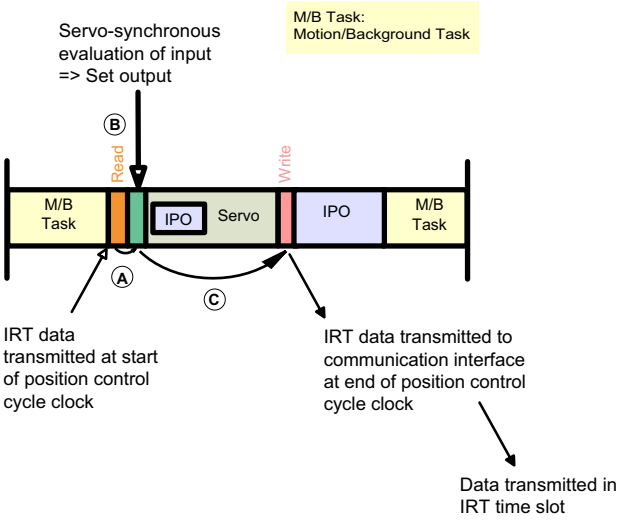


Figure 5-16 Fast IOs in the ServoSynchronousTask

If the **second servo clock** is configured, the fast I/O processing can also be performed in the Servo_fast clock (i.e. in the ServoFastSynchronousTask). This requires the I/O peripherals to be configured on the faster bus.

5.2.4 SIMOTION controllers with two PROFINET IO interfaces

5.2.4.1 Two PROFINET IO interfaces

SIMOTION controllers with two PROFINET IO interfaces

With the SIMOTION D4x5-2 DP/PN controllers, optionally a second PROFINET IO interface is available in addition to the onboard PROFINET interface (X150) with the Ethernet communication board (CBE30-2, X1400).

The CBE30-2 can only be operated in D4x5-2 DP/PN control units. It is impossible to use it in D4x5-2 DP control units.

Application description

A detailed application description explaining how to operate two different PROFINET networks on one SIMOTION controller at the same time is available to download in the Service and Support Portal.

PROFINET & SIMOTION: Use of Two PROFINET Interfaces with SIMOTION (http://support.automation.siemens.com/WW/view/de/59396321)

See also

Example configuration for "controlled" sync master (Page 91)

Cycle clock scaling with PROFINET IO on SIMOTION devices (Page 67)

Use of Safety with MRRT (Page 105)

5.2.4.2 Basic rules for using two PN IO interfaces

Introduction

The following rules must be considered in use of two **isochronous** PROFINET interfaces. The most important basic rules for the PN interfaces and the use of Servo_fast are sorted thematically there.

Definition of isochronous PN interfaces

A PN interface is isochronous (i.e. assigned to servo or Servo_fast) if the following conditions apply.

- The PN interface is the IO controller and one of the IO devices is isochronous.
- The I-device is configured isochronously.
- Controller-controller data exchange broadcast is configured.

Note

IO controller and I-device cannot simultaneously be configured isochronously (IRT) on a PN interface. One PN interface can be configured as the IRT controller and the second PN interface as the IRT I-device.

If a PN interface is configured as IRT and does not meet the conditions for isochronous operation, it will work **asynchronously** with respect to the Servo or Servo_fast clocks and therefore also asynchronously with respect to the second isochronous PN interface.

General rules when using the onboard PN interface (X150) and CBE30-2 PN interface (X1400)

- A higher-level, isochronous controller must always be connected via the CBE30-2 PN interface. This applies both to controller-controller data exchange broadcast communication and to I-device communication with a higher-level controller.
- Lower-level isochronous drives or IO devices should be connected via the onboard PN interface.
- Both PN interfaces can be configured as a PROFINET IO I-device and/or as an IO controller.
- If a PN interface is used as an IRT I-device, it cannot be configured as a redundant sync master. Only the sync slave and sync master roles are possible.

Constraints on Sync master and Sync slave

- The CBE30-2 PN interface can be configured as the sync master, sync slave, or redundant sync master.
- The onboard PN interface can only be configured as the sync master.
- In the Sync domain of the onboard PN interface, there must be no redundant sync master if the CBE30-2 is configured as a Sync slave or IRT I-device.

Failsafe I/Os for using two PN interfaces

An F-CPU can only be connected to one of the two PN interfaces, because fail-safe I/O transfer areas can only be configured either on the I-device on the onboard PN interface **or** on the CBE30-2 PN interface (I-device F proxy).

The PROFIsafe telegrams are routed to the following drives:

- Drives on the SINAMICS Integrated and CX32-2
- Drives on the external PROFIBUS
- Drives on the onboard PROFINET interface (X150 interface)
- Drives on the CBE30-2 (X1400 interface)
  The maximum quantity structure for PROFIsafe on PROFINET is therefore not increased by using a second PROFINET interface.

Use of Servo_fast

For the use of Servo_fast, the following rules apply.

- If Servo_fast is not used, either one or both PN interfaces can be assigned to the servo.
- If Servo_fast is used, the onboard PN interface is assigned to Servo_fast and the CBE30-2 PN interface can be assigned to the servo.

Constraints on the use of Servo_fast

- Only the onboard PN interface can be assigned to Servo_fast.
- If Servo_fast is used, only one CACF=MACF=1 (Controller Application Cycle Factor=Master Application Cycle Factor) is permitted Servo and Servo_fast, i.e. the servo clock is taken over from the set send clock in HW Config and does not have to be set especially.
  The following applies:
  - Servo clock = send clock CBE30-2 PN interface X1400
  - Servo_fast cycle clock = send clock onboard PN interface X150
- The onboard PN interface can only be operated as the sync master (not sync slave or redundant sync master).
  This means that **no** controller-controller data exchange broadcast can be configured between the two SIMOTION controllers if the onboard PN interfaces of both SIMOTION controllers are assigned to Servo_fast.
- The onboard PN interface can be an IRT IO controller but not an IRT I-device if assigned to Servo_fast.
- If the Servo_fast is used, it is essential that isochronous IO devices are assigned to it.

Constraints on the clock ratio and cycle time of servo/Servo_fast

- Rules for the cycle clock ratio **servo = N * Servo_fast**
  - Servo_fast > 250 µsec: N = 2, 4, 8, 16, 32, …
  - Servo_fast <= 250 µsec: N = 4, 8, 16, 32, …(For 32: Servo_fast = 125 µs, Servo = 4000 µs)
- The minimum send clock for Servo_fast depends on the SIMOTION device (see also "Adjustable cycle clocks")
  - D455-2 DP/PN: 125 µs
  - Other: 250 µs
- The maximum send clock for the servo depends on the source of the IO data.
  - PROFINET is the source of the IO data: 4 ms
  - SINAMICS_Integrated or PROFIBUS is the source of the IO data: 8 ms

5.2.4.3 Applications for devices with two PROFINET IO interfaces

Applications for two PN IO interfaces

Possible applications

The PROFINET interface of a CBE30-2 can be used, for example, for the following applications:

- The second PROFINET interface is used
  - to increase the maximum number of connectable IO devices
  - to increase the available I/O address space
- IO devices are to be operated with different send cycle clocks and assigned different isochronous tasks (Servo and Servo_fast).
- IO devices should have an independent IP address range or NameOfStation. Devices with identical device configuration at the local interface can thus be connected to each other independently via the other interface in the plant network (e.g. higher-level network as the plant network; local network for machine module; identical machine modules; participants in the "local network" can be addressed independently of nodes in the "plant network").
- I-device and IO controller are to be operated isochronously at the same time.
  If one PROFINET interface is operated simultaneously as an I-device and IO controller, either only the I-device or the IO controller can be operated isosynchronously (i.e. with IRT). With two PROFINET interfaces, it is possible to operate a PROFINET IO controller and an I-device isosynchronously on a D4x5-2 DP/PN at the same time.

Application examples

There are, for example, the following applications for two PROFINET IO interfaces:

1. Increase the number of devices on PROFINET from 64 to 128, see Application 1 - increase in the maximum possible number of IO devices on PROFINET (Page 82).
2. Fast and slow clock cycle for data to isochronous devices, see Application 2 - fast I/O processing and slower electrical drives on one module (Page 83).
3. Distributed synchronous operation independent of isochronous devices, see Application case 3 - distributed synchronism over multiple devices and independent local systems (Page 84).
4. The same project for the machine modules
5. Machine modules are individualized by assigning the IP address and NameOfStation on site, see Application 4 – Modular Machine: Independent IP address and NameOfStation for the iDevice and IO controller (Page 85).
6. Two independent I-devices, see Application 5 - iDevice on both interfaces (Page 87).

Application 1 - increase in the maximum possible number of IO devices on PROFINET

Description

The following functions are illustrated in the topology:

- Increased number of possible devices on SIMOTION D4x5 2 DP/PN by 2 subnets from 64 to up to 128.
- IRT and RT in each subnet

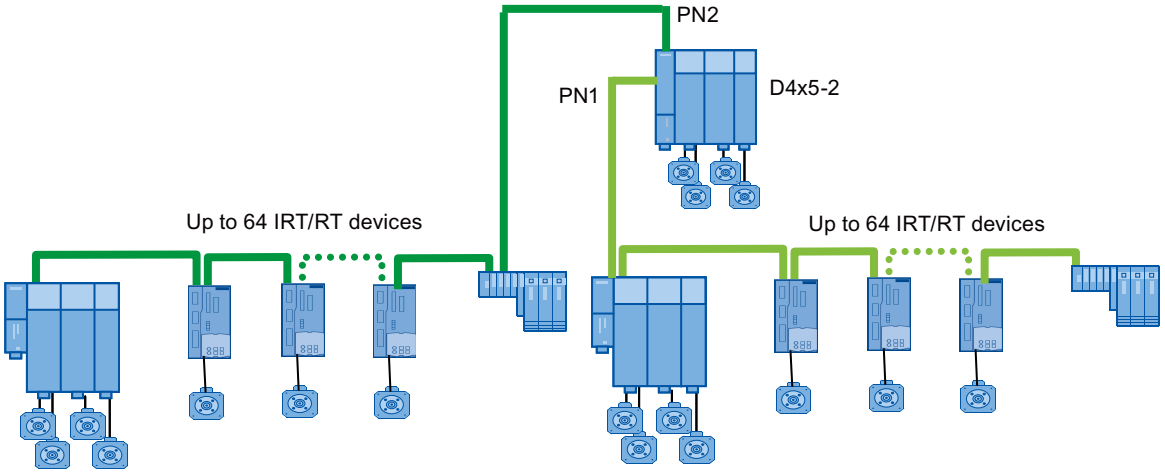


Figure 5-17 Application 1 for two PROFINET IO interfaces

PN1	Integrated PROFINET interface
PN2	CBE30-2

See also

Applications for two PN IO interfaces (Page 81)

Application 2 - fast I/O processing and slower electrical drives on one module

Description

The following functions are illustrated in the topology:

- Fast I/O application and slower drive functions in one controller.
- Control of slower SINAMICS drives via PN2 (e.g. servo clock of 1 ms).
- Fast I/O processing (e.g. ET200S High Speed) via PN1 (e.g. Servo_fast clock of 250 µs).

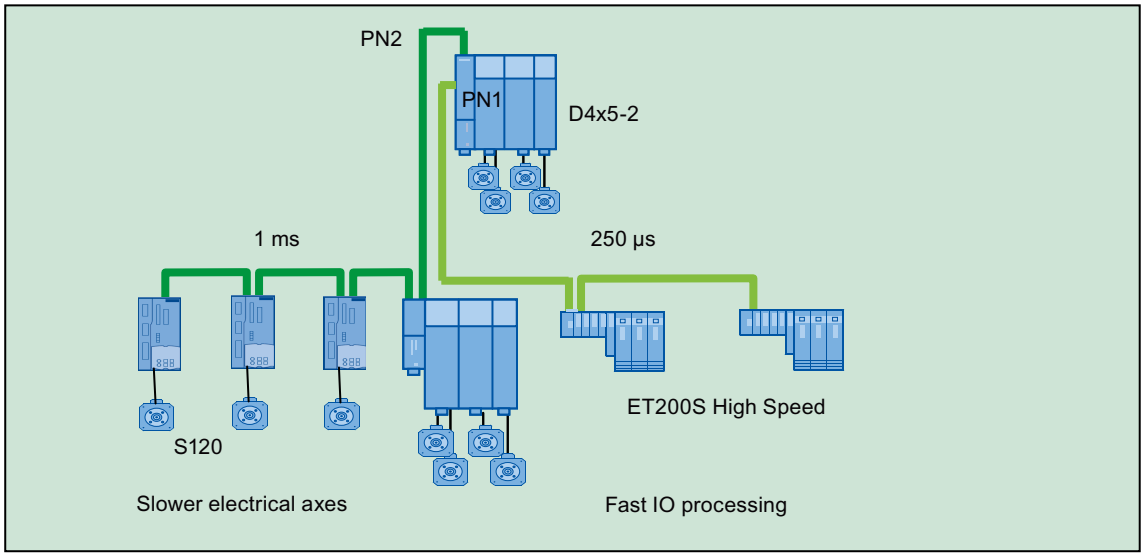


Figure 5-18 Application 2 for two PROFINET IO interfaces

PN1	Integrated PROFINET interface
PN2	CBE30-2

See also

Applications for two PN IO interfaces (Page 81)

Application case 3 - distributed synchronism over multiple devices and independent local systems

Description:

The following functions are illustrated in the topology:

- Distributed synchronous operation via controller-controller communication between the SIMOTION controllers of different machine modules in subnet 1.
- Local I/O on each SIMOTION controller of a machine module in separate subnet independent of subnet 1.
  - Independent adjustable communication parameters (send clock, IRT, IP address, NameOfStation, etc.), adapted to the technical requirements.
  - Addresses can be set identically to the other modules, thus facilitating simpler establishment of modular machine concepts
- Use of Servo and Servo_fast

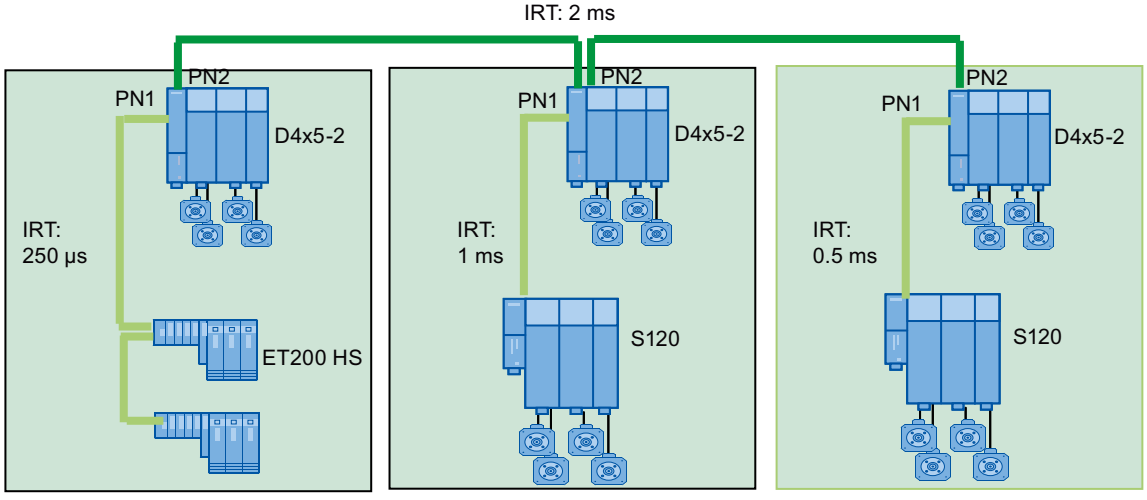


Figure 5-19 Application 3 for two PROFINET IO interfaces

PN1	Integrated PROFINET interface
PN2	CBE30-2

See also

Applications for two PN IO interfaces (Page 81)

Application 4 – Modular Machine: Independent IP address and NameOfStation for the iDevice and IO controller

Description

The following functions are illustrated in the topology:

- 2 PN interfaces in the SIMOTION controllers disconnect the subnet of the base machine from the machine modules.
- IP address and NameOfStation of the interface of PN2 can be granted independently of IP address and NameOfStation of the interface of PN1.

  Note

  The IP addresses of the two PN interfaces must be in different IP subnets.

- IP addresses and NameOfStation of the PN interfaces that are operated as an IO controller can be identical in the different machine modules.
- In the base machine project the SIMOTION controllers are added as I devices via a GSD file (I device substitute).

- CBE30-2 as IRT I device and onboard PN interface as IRT IO controller.
- Distributed synchronous operation between the base machine controller and the IRT I devices.
  - Isochronous coupling of the subnets, if PN axes
  - IRT communication on iDevice and controller interface

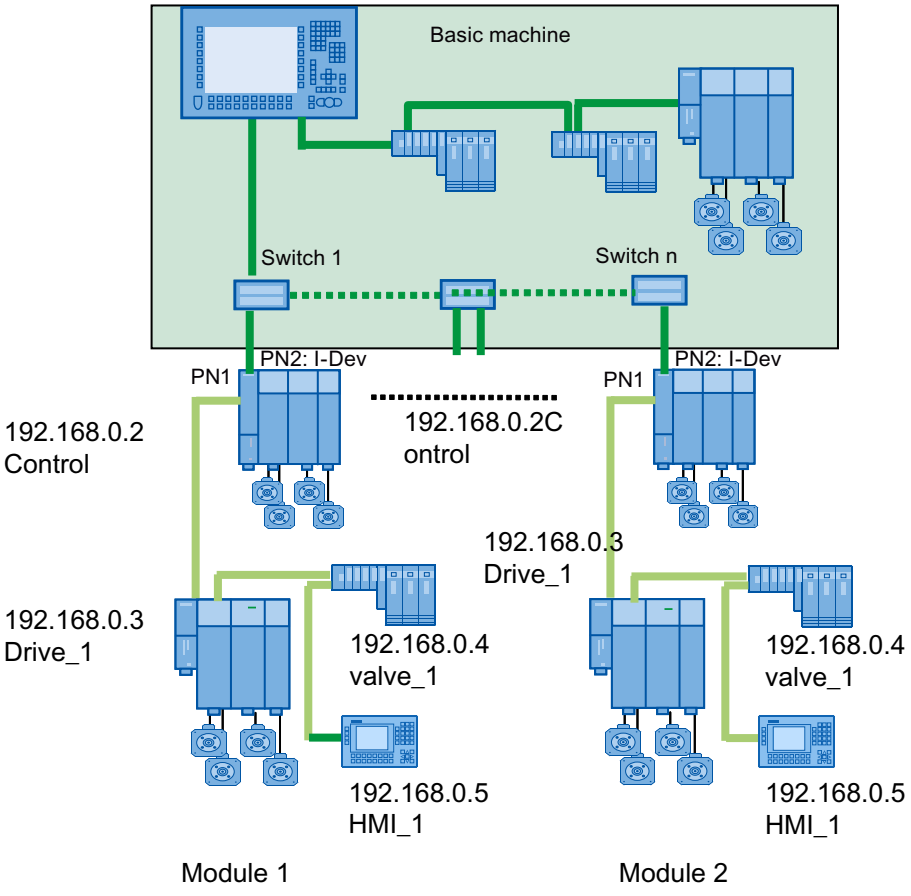


Figure 5-20 Application 4 for two PROFINET IO interfaces

PN1	Integrated PROFINET interface
PN2	CBE30-2

See also

Applications for two PN IO interfaces (Page 81)

Application 5 - iDevice on both interfaces

Description

The following functions are illustrated in the topology:

- Distributed synchronous operation between higher-level controller and lower-level SIMOTION controllers via CBE30-2 as an IRT I device
- RT communication between additional higher-level controllers and lower-level SIMOTION controllers via onboard PN interfaces as an RT I device (e.g. for PROFIsafe, diagnostics, process value recording). IRT communication is not possible here.

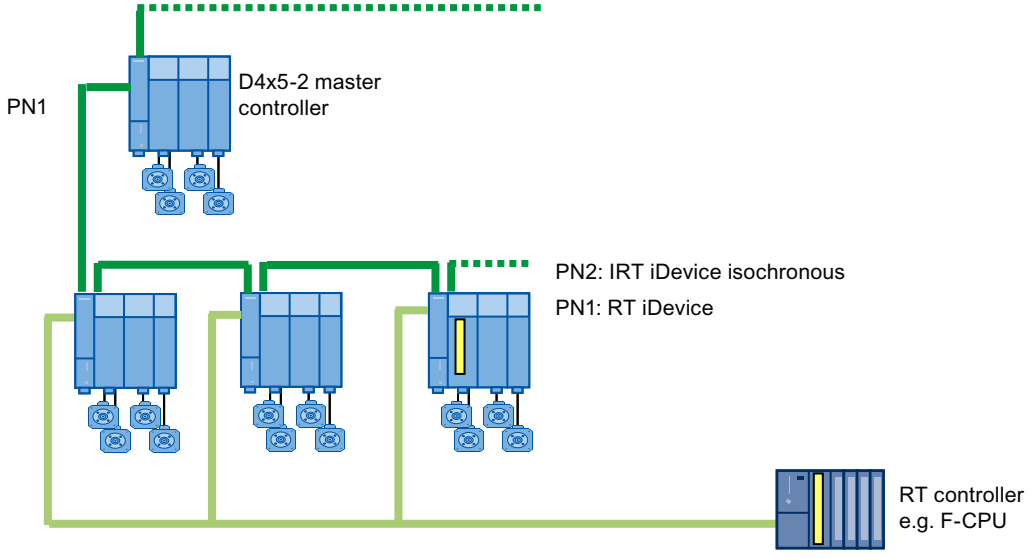


Figure 5-21 Application 5 for two PROFINET IO interfaces

PN1	Integrated PROFINET interface
PN2	CBE30-2

See also

Applications for two PN IO interfaces (Page 81)

5.2.4.4 "Controlled" sync master

"Controlled" sync master

The "controlled" sync master synchronizes the PROFINET IRT cycle of the two PROFINET IO interfaces with the bus cycle of Servo/Servo_fast of the higher-level controller. The property "Controlled Sync Master" does not have to be explicitly configured. In the engineering, set "Sync Master" for this. The property "Controlled Sync Master" is automatically activated at the integrated PN interface if isochronous data has been configured on both PN interfaces, i.e. the send cycle clocks of both interfaces are synchronized to servo or Servo_fast.

A "controlled" sync master cannot be used together with a redundant sync master in a sync domain.

Application-specific synchronization is necessary analogously to PROFIBUS if the synchronization is lost and the system must be restarted.

Synchronization example with two PROFINET interfaces (IRT i device and IRT IO controller)

The following illustration shows the synchronization if the CBE30-2 is configured as an IRT I device for a higher-level controller and the onboard PN interface is configured as an IRT IO controller for the lower-level local peripherals.

Requirements for synchronization:

- CBE30-2 (X1400): Sync slave (synchronizes with the higher-level IO controller).
- Onboard PN interface (X150): "Controlled" sync master synchronizes the lower-level local peripherals.
- You can achieve application-specific synchronization with the _synchronizeDPInterfaces function, for example. See also *Clock generation and synchronization in PROFINET IO* in the Function Manual *Basic Functions for Modular Machines*.

  Note

  The application-specific synchronization can also be achieved in the StartupTask using the system function _enableDpInterfaceSynchronization.

- An example configuration can be found at Example configuration for "controlled" sync master (Page 91).

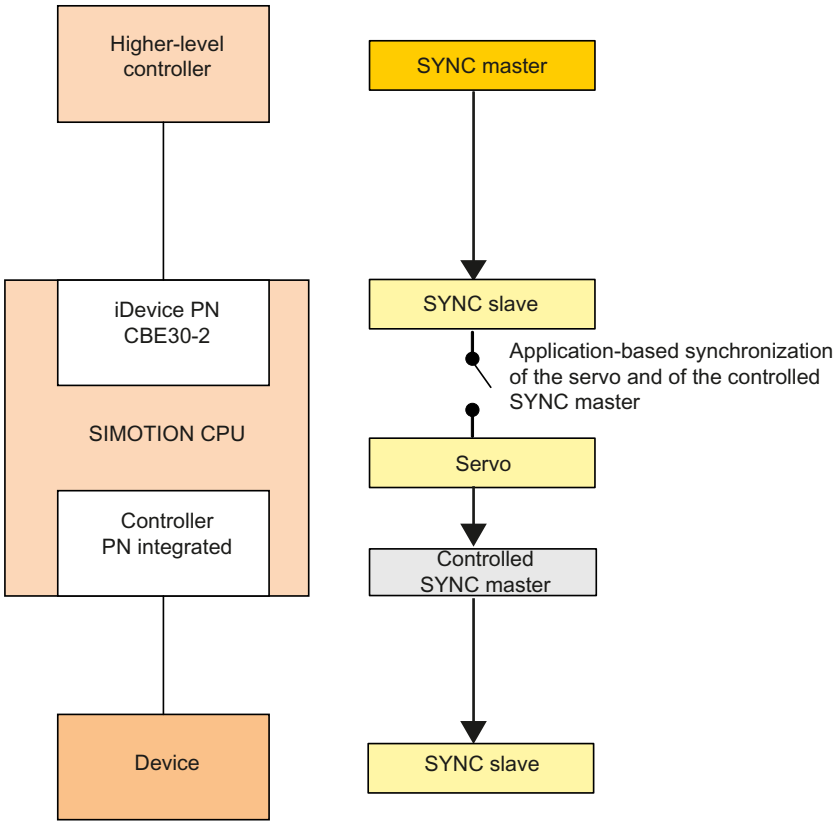


Figure 5-22 "Controlled" sync master

Synchronization example with two PROFINET interfaces (IRT IO controller on both interfaces)

The following illustration shows the synchronization if the CBE30-2 is configured as an IRT IO controller for direct controller-controller communication and the onboard PN interface is configured as an IRT IO controller for the lower-level local peripherals.

Requirements for synchronization:

- Onboard PN interface (X150): "Controlled" sync master synchronizes the lower-level local peripherals.
- CBE30-2 (X1400): Sync master (synchronizes the IO controller configured as a sync slave which is involved in direct data exchange) or sync slave (synchronizes with the IO controller configured as a sync master which is involved in direct data exchange).
- You can achieve application-specific synchronization with the _synchronizeDPInterfaces function, for example. See also *Clock generation and synchronization in PROFINET IO* in the Function Manual *Basic Functions for Modular Machines*.

Note

The application-specific synchronization can also be achieved in the StartupTask using the system function _enableDpInterfaceSynchronization.

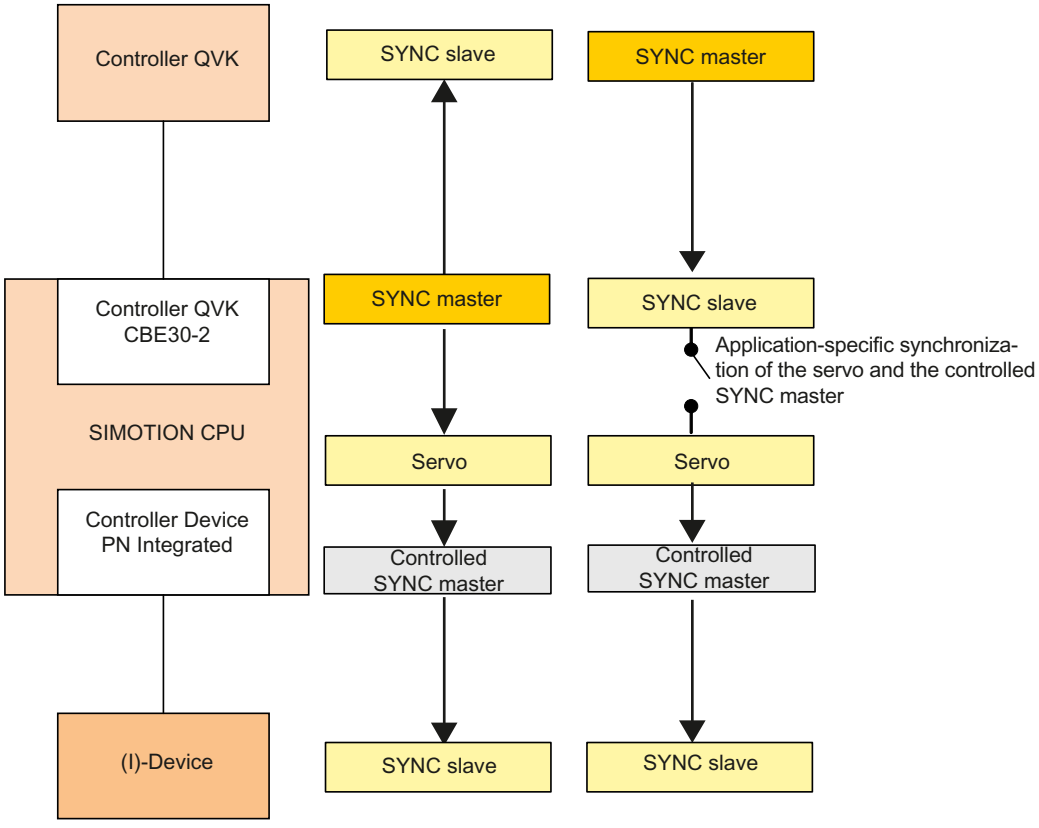


Figure 5-23 Controller direct data exchange with "controlled" sync master

5.2.4.5 Example configuration for "controlled" sync master

Distributed synchronous operation via two PROFINET interfaces

The example shows which application requires the use of two PROFINET interfaces for SIMOTION controllers. The PROFINET network "1" is configured as the MRPD ring for the distributed synchronous operation. The SIMOTION controller which has been configured as a sync master specifies the bus cycle clock as well as the servo and IPO cycle clock. The two other SIMOTION controllers are integrated into the MRPD ring as sync slaves via the CBE30-2 and synchronize with the sync master. Through their onboard PROFINET interface, they each set up their own PROFINET network and transmit the bus cycle clock as a "controlled" sync master to the lower-level local periphery (SINAMICS drives).

See also "Controlled" sync master (Page 87) and Two PROFINET IO interfaces (Page 77).

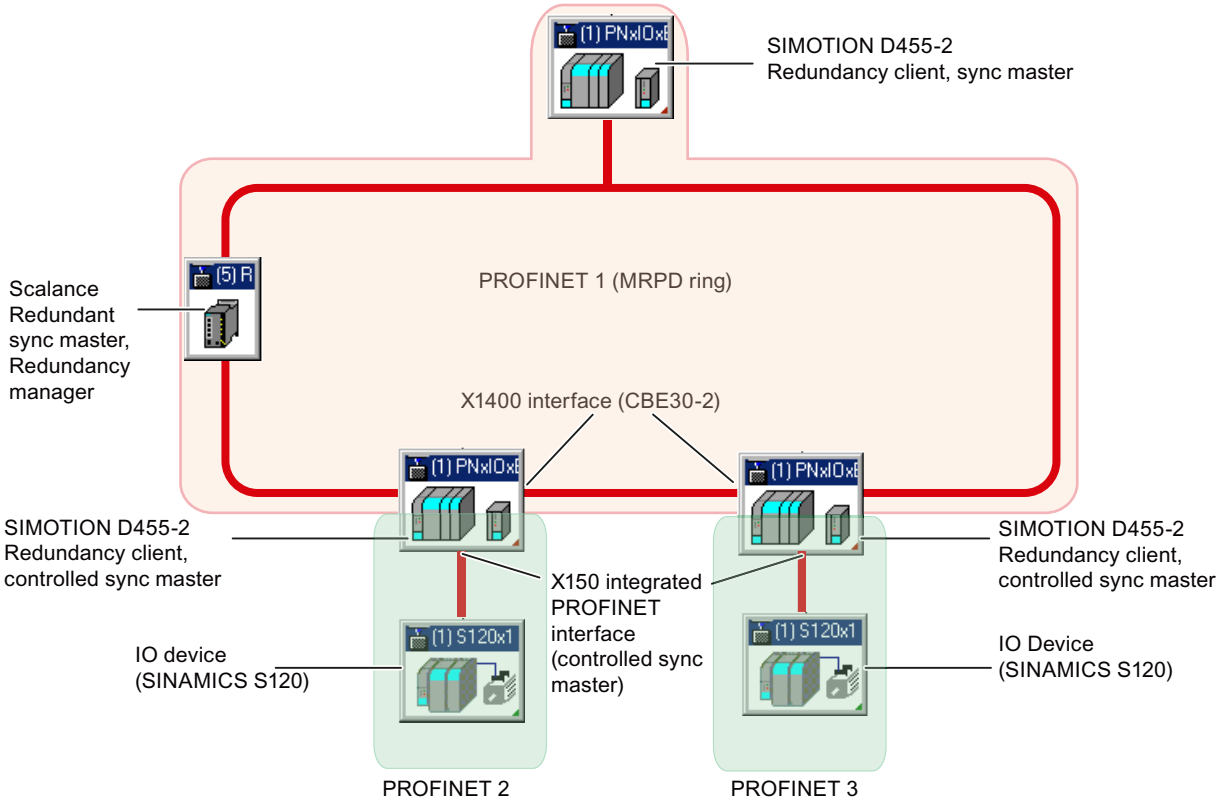


Figure 5-24 Example configuration for controlled sync master

If several sync domains are operated on a physical Ethernet, you should ensure that:

- either all sync domains have different names
- or configure this explicitly as the end of the sync domain at the ports that form the end of the sync domain.
  To do this click on the port that is to be the end of the sync domain and select **Properties** from the context menu. Choose the **Options** tab from the properties dialog box that appears. Activate the option **End of sync domain** and confirm with **OK**.

5.2.5 Relationship between sync domain and PROFINET IO systems

Introduction

The devices of several PROFINET 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, a PROFINET IO system may only belong to a single sync domain.

Devices with two PROFINET interfaces

The two PROFINET interfaces must belong to different sync domains. Synchronization of the two sync domains then takes place via a "controlled" sync master, see also "Controlled" sync master (Page 87).

To find out how to create a second sync domain or change the name of an existing sync domain, go to Creating a sync domain (Page 128).

5.2.6 Redundant sync master

Description

For certain plants, e.g. printing machines in "tandem configuration", it is necessary for the two plant sections to be operated in stand-alone mode or together synchronously. If the entire system has only one sync master, the other component would not be capable of functioning independently. The redundant sync master function was developed for this very reason. Under this arrangement, a sync master is defined for every component. One of these is defined as the sync master, and the other as the redundant sync master. Provided the system components are combined during operation, the redundant sync master will synchronize itself with the sync master.

Applications

- Tandem machine
  Tandem machines are large machines whose machine sections can be operated independently of each other. The machine can thus be operated as a total machine or as a machine section, see also Media redundancy for SIMOTION (Page 97).
- Redundant sync master in the case of MRPD
  A redundant sync master is also possible in the case of a ring topology with media redundancy, see also Sample configurations for MRPD rings (Page 103) and Information on PROFINET with MRPD (Page 101). For this, an additional SCALANCE switch must be configured in the ring as a redundant sync master.

Limitations of use

- There can physically be up to a maximum of two additional modules between two sync masters.
- If the transmission link between the sync master and redundant sync master fails, leaving two subnets with one sync master each, both subnets remain synchronized with the remaining sync master in each case. This results in two independent synchronized subnets that drift apart due, among other things, to the temperature drift of the quartzes.
- Once the transmission link has been reestablished, the redundant sync master cannot synchronize with the sync master. The redundant sync master must be switched off and on again (power OFF/ON) before the synchronization can take place.
- If the plant section with the sync master fails as the result of a fault or shutdown, the other plant section with the redundant sync master will continue operating independently. The sync slaves assigned to the sync master will lose their synchronization and cease to operate.

Configuring the redundant sync master

1. Add a second SIMOTION module or a SCALANCE Switch (if MRPD) and configure the PROFINET communication in accordance with your requirements.
2. Right-click on the PROFINET interface to open the **Properties - <PROFINET interface> -- (R0/S2.6)** dialog box.
3. Select the **Sync master (redundant)** entry under **Synchronization role** on the **Synchronization** tab.

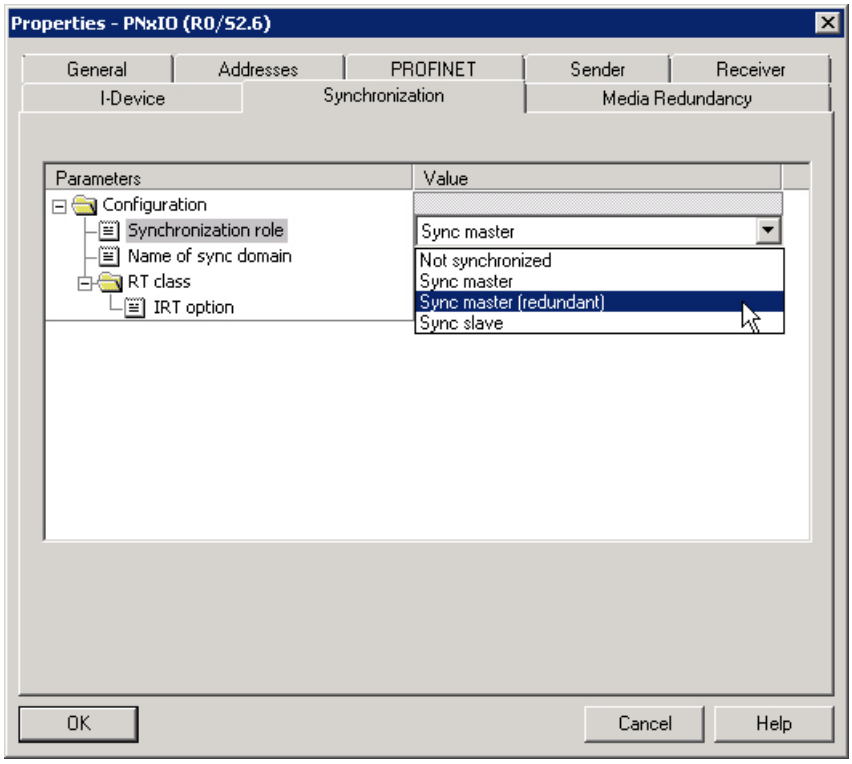


Figure 5-25 Configuring the redundant sync master

Deployment of a redundant sync master using the example of a tandem machine

The switchover behavior of the sync master and the redundant sync master is subject to several rules that must observed during the project creation and for applicative solutions.

- The topological design of the machine should provide the connection between the two machine parts via the sync master and the redundant sync master.
- If possible, no further device should be present between the sync master and the redundant sync master (e.g. IRT switch).

Note

Wait time for switching the sync master to the redundant sync master

A wait time of 60 seconds must be observed for switching from the sync master to the redundant sync master. Only after this interval is it guaranteed that the redundant sync master has assumed its final status rather than the sync master resuming its role after a return.

Boundary conditions

- If the redundant sync master is switched on after the sync master, all sync slaves remain synchronized to the sync master.
- If the redundant sync master is switched on first and then the sync master after a wait time (> 60 seconds), all sync slaves remain synchronized to the redundant sync master.
- If both sync masters are switched on simultaneously, all sync slaves are synchronized to the sync master (this behavior is guaranteed only for an approximately equal startup time).
- If the connection between the sync master and the redundant sync master is interrupted, both sync masters accept their appropriate role and the topologically assigned sync slaves are synchronized accordingly.
- If the sync master and the redundant sync master are switched on simultaneously without a direct connection, the connection will be restored after a time > 60 seconds. The sync slaves remain synchronized to the sync master and the redundant sync master with which they are connected topologically. The two sync domains are not joined automatically. A diagnostic message will be generated.
- If the sync master is switched off, the redundant sync master handles the synchronization of the topologically-linked sync slaves. The sync slaves remain synchronized, i.e. no failure occurs.
- If the redundant sync master is switched off, the sync slaves connected topologically with the sync master remain synchronized.

5.2.7 Quantity structures

The following maximum values apply for IO controllers of the SIMOTION platform.

A maximum of 64 communication relationships are possible; these can be divided up as follows:

- Connection of up to 64 IO devices.
- Maximum 64 RT IO devices.

- Maximum 64 IRT IO devices (High Performance).
- A maximum of up to 64 controller-controller data exchange broadcast relationships may be set up between SIMOTION controllers.
- If a SIMOTION controller has been configured as an I-device, it counts as an IO device; in other words, only another 63 IO devices can be connected to this controller.
- A SIMOTION controller can receive data from up to 64 other SIMOTION controllers as part of controller-controller data exchange broadcast yet can send data to any number of SIMOTION controllers.
- The IRT telegrams in controller-controller data exchange broadcast are broadcast telegrams. This is why any number of SIMOTION controllers can access the data of one SIMOTION controller.
- In controller-controller data exchange broadcast, the data volume also has to be considered. A data exchange broadcast only counts as a connection up to a certain data volume. Where a second telegram is required, data exchange broadcast does not require two connections.

Note

For MRPD, halve the quantity structures, since an MRPD IO device requires the resources of two IO devices. The same applies to controller-controller data exchange broadcast.

The IO area doubles if two PROFINET IO interfaces are used.

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:

```
RT + IRT High Performance IO device + data exchange broadcast frames
<= 64
```

Note

In a data exchange broadcast relationship, it is not the number of configured slots (lines on the **Receiver** tab - see Configuring the receiver (Page 175)) that is intended for IRT High Performance direct communication configuration, but rather the number of Ethernet frames received for the data exchange broadcast. An Ethernet frame can contain up to 768 bytes of data exchange broadcast net data.

One slot has up to 768 bytes and 3,072 bytes of net data can be exchanged during data exchange broadcast (divided into 4 frames of 768 bytes each). The value will depend on the sizes of the slot chosen. This means a transmitter can receive more than one slot, although these must be transmitted within a single telegram.

Each sender sends its data exchange broadcast data in an Ethernet frame. Every other SIMOTION controller can read the data in this frame. This results in a counting connection to each transmitting SIMOTION controller.

For data exchange between an IO controller and an IO device, a frame has a useful data size of 1,440 bytes for each transmitting SIMOTION controller.

During the compilation of the project, HW Config verifies the configured quality structure based on the formulas mentioned above.

Address space

A maximum of 4 KB each may be assigned for PROFINET IO data for the input and output data in the logical address space of an IO controller. The rest of the 16 KB large address space can be used, for example, for PROFIBUS data or diagnostic data.

Submodule size from SIMOTION V4.4

In PROFINET, data for devices (and I-devices) and the controller-controller data exchange broadcast is structured in the form of submodules. Only data from a submodule can be read or written consistently when "non-synchronized" with PROFINET RT or IRT. The max. submodule size of SIMOTION controllers for PROFINET interfaces has been increased for the I-device and for the IO device from 254 bytes to 1024 bytes. The size of the submodule depends on the hardware and the PROFINET interface.

Maximum submodule size and consistency range

The following table defines the maximum submodule size, depending no the SIMOTION controller and the PROFINET interface. The submodule size stated applies to operation of a PROFINET interface with the following function:

- IO controller (the IO controller can operate IO devices with submodules up to the specified size)
- I-device (on the local I-device, submodules up to the specified size can be set up)

SIMOTION module	Onboard PROFINET interface [bytes]	CBE30-2 [bytes]
D455-2 DP/PN	1024	254
D445-2 DP/PN	1024	254
D435-2 DP/PN	1024	254
D425-2 DP/PN	1024	254
D410-2 DP/PN	1024	-
C240 PN	254	-
P320-4 PN	254	-
P320-4 DP/PN	254	-

Note

Controller-Controller data exchange broadcast

For controller-controller data exchange broadcast, the max. submodule size of 254 bytes is retained.

Rule for use of a SIMOTION or SIMATIC I-device on a SIMOTION IO controller

If a SIMOTION controller is configured as an IO controller and a SIMOTION or SIMATIC controller is configured as an I-device, submodules can be larger than 254 bytes subject to the following constraints:

- The I-device must support submodules up to 1024 bytes.
- The IO controller must support submodules up to 1024 bytes.

Note

For SIMATIC controllers, a max. submodule size of 1024 bytes applies.

Potential problems when using interfaces with different consistency lengths

If, for example, you configure an I-device that contains a submodule with a size of 500 bytes on the onboard PROFINET interface of a SIMOTION D4x5-2, the I-device cannot be operated on a SIMOTION C240 PN that is a higher-level controller. An error message is displayed in STEP 7, stating that the output/input net data length of 254 bytes has been exceeded.

See also

Data exchange through the use of iDevices (Page 198)

5.2.8 Media redundancy (MRP and MRPD)

5.2.8.1 Media redundancy for SIMOTION

Media Redundancy Protocol

It is possible to establish redundant networks via the so-called Media Redundancy Protocol (MRP). Redundant transmission links (ring topology) ensure that an alternative communication path is made available when a transmission link fails. The PROFINET devices that are part of this redundant network form an MRP domain.

Note

Devices that support MRP

An overview of all devices that support media and/or system redundancy can be found in the Support area (http://support.automation.siemens.com/WW/view/en/67364686).

MRP (Media Redundancy Protocol for PROFINET RT, as of SIMOTION V4.3)

MRP guarantees media redundancy in the event of a problem in the ring. The switchover of the ring is performed by the Redundancy Manager.

The switchover times depend on the following parameters:

- The actual topology
- The devices used and
- The network load in the relevant network.

The typical reconfiguration time of the communication paths for TCP/IP and RT frames in the event of a fault is < 200 ms.

In most systems, the switchover time of MRP is far above the PROFINET update time for cyclic data, so that a failure for cyclic data is detected. The PROFINET connection therefore fails and is re-established after the switchover by the Redundancy Manager. In this way, an error can be corrected in the network, while the system continues to run *with bumps*.

The figures below show a sample topology with and without fault:

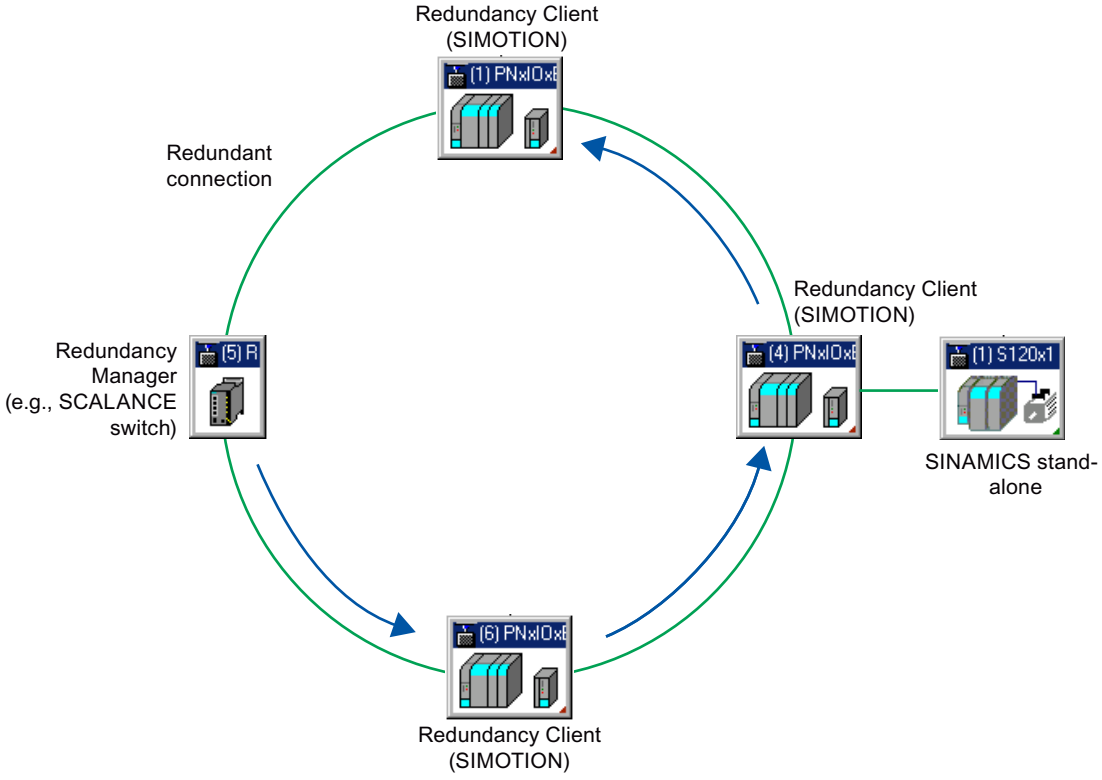


Figure 5-26 Ring topology for media redundancy without fault

After the occurrence of a fault, the Redundancy Manager reconfigures the communication of the ring. In contrast with MRPD, the PROFINET RT telegrams are only sent in one direction in the ring with MRP. Therefore, the redundant communication path is only closed by the redundancy manager in the event of a fault.

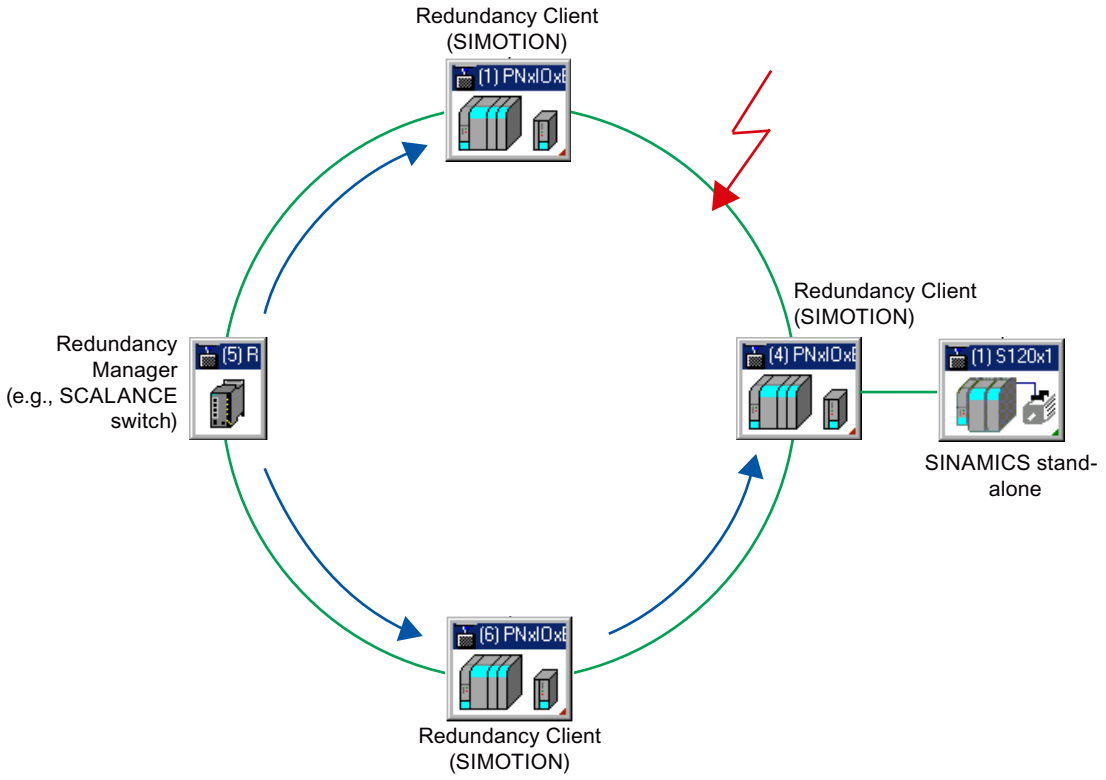


Figure 5-27 Ring topology for media redundancy with a fault

Ring ports

A SIMOTION controller / SINAMICS drive may only be inserted into an MRP ring as a node with MRP-capable ports. For SIMOTION D, the first two ports of the PROFINET IO interfaces are designed as ring ports.

These two ports are marked with an "R" in the module rack in HW Config.

Note

Only devices with MRP-capable ports may be inserted in an MRP ring. If MRP-capable ports are not used, the reconfiguration times can be in the multi-digit seconds range.

For SINAMICS S120 drives (SINAMICS S120 CU320-2/CU310-2), these are ports P1 and P2. However, they are not separately designated as ring ports in the HW Config.

MRPD (Media Redundancy for Planned Duplication for PROFINET IRT, from SIMOTION V4.3)

MRPD (Media Redundancy for Planned Duplication) is a method for smooth media redundancy with PROFINET IRT. MRPD also requires MRP.

The combination of MRP with MRPD provides smooth PROFINET operation for short cycle times in the event of a fault in the ring. MRPD is based on IRT and ensures bumpless operation by the sender sending the cyclic data in both directions which the recipient then receives twice. The first received telegram is used by the recipient. The second telegram is discarded

automatically. If the ring is interrupted at one position (e.g., through the failure of a ring node), receipt of the cyclic data via the uninterrupted side of the ring is still guaranteed. The following figure shows a sample configuration:

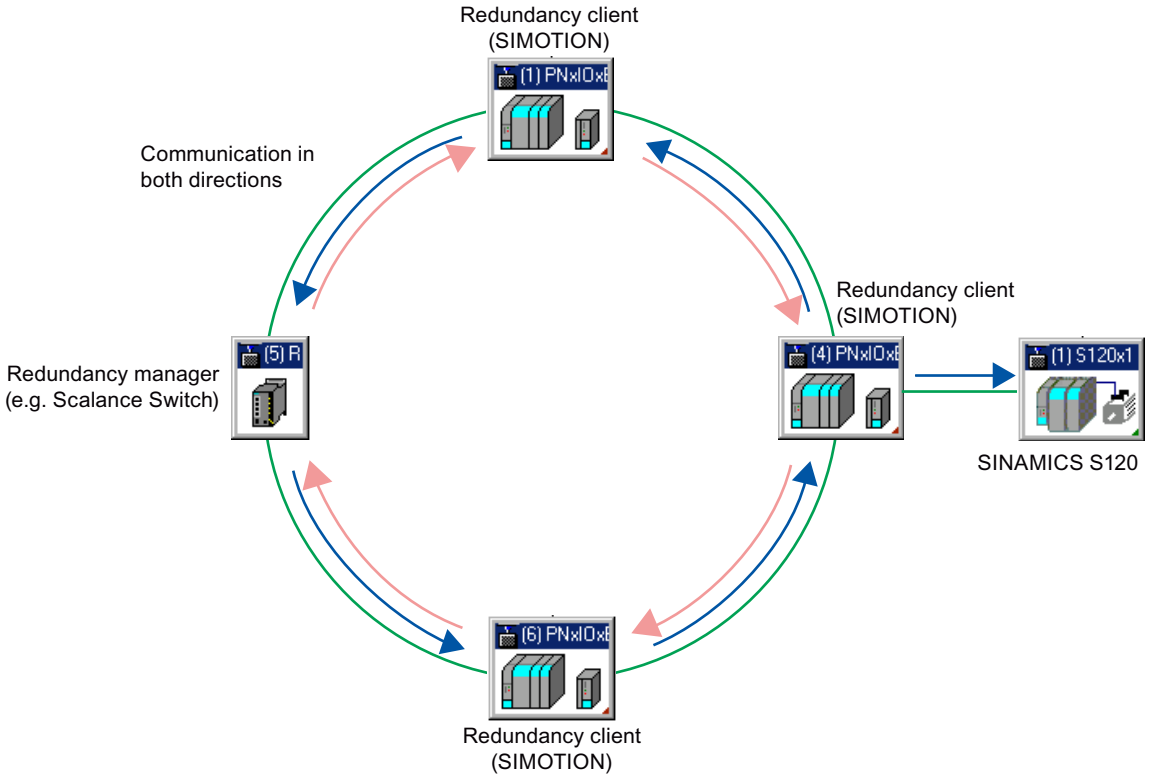


Figure 5-28 Ring communication in MRPD

Smooth media redundancy MRPD always requires the activation of MRP in the individual rings.

Note

Too high a network load or too rapid incoming/outgoing faults caused by delayed or continuous switchovers of the MRP, may lead to failure of the PROFINET connection even with activated MRPD.

Devices that support MRPD

Note

An overview of all devices that support media and/or system redundancy can be found in the Support area (http://support.automation.siemens.com/WW/view/en/67364686).

If the device prevents MRPD, an **error message** is issued in HW Config:

- "The <xxxx> device does not support sync domain with MRPD area."

SIMATIC controllers, such as a SIMATIC 300 or an ET200, currently do not support MRPD.

See also

Servo_fast, scaling down of cycle clocks to the servo at the PROFINET interface (Page 136)

Redundant sync master (Page 92)

5.2.8.2 Information on PROFINET with MRPD

Smoothness of MRPD

Delayed or incomplete switchovers of MRP/MRPD caused, for example, by a network load that is too high or the too rapid incoming/outgoing of faults, under unfavorable conditions, can lead to the PROFINET connection failing even with activated MRPD.

For example, in the event of two consecutive faults at various parts of the ring, smooth operation is only ensured when the two faults are separated by approx. three seconds (if a fault has "gone", an additional fault may "come" only after a short pause).

Operation of MRPD in conjunction with a redundant sync master

There can be a maximum of two Ethernet nodes between one sync master and one redundant sync master.

If the redundant sync master is used together with MRPD, it is recommended that the redundant sync master (e.g. a SCALANCE IRT switch) be connected directly to the sync master and the two nodes positioned in a common switch cabinet so that the cable connection between both nodes is protected.

The insertion or withdrawal of a connection in an MRPD ring does not have any effect on the synchronization. Only one sync master remains.

In case of an interruption in the distance between sync master and redundant sync master, the system initially continues to run smoothly, however faults may occur when you switch off and then restart the system if the devices between the sync master and the redundant sync master have too great a difference in ramp-up time.

The fault is that the sync domain breaks down into two sub-domains, and one part of the sync slave synchronizes to the sync master, and the other part synchronizes to the redundant sync master. The PN diagnosis "Remote mismatch / Peer PTCP mismatch 0x8008" is present.

If the sync master or the redundant sync master in an MRPD ring is switched off, all sync slaves remain synchronized, because unlike the line, all sync slaves retain their connection to the remaining sync master.

Configuration instructions

You must comply with the following rules to operate PROFINET IRT in connection with MRPD:

- The rules for configuring MRP and PROFINET IRT must be followed.
- All devices in the ring must support MRPD.

- If devices at the stub are in redundant communication via the ring, all devices at the stub must support up to the last MRPD node of the MRPD stub.
- Nodes in the stub that only communicate locally in the stub and not over the ring, do not have to support MRPD. They do, however, require the IRT property "StartupMode=Advanced" so you can operate them in a sync domain together with MRPD nodes. In V4.3, the devices described in Media redundancy for SIMOTION (Page 97) have this IRT property.

See also

Media redundancy for SIMOTION (Page 97)

5.2.8.3 Sample configurations for MRPD rings

MRPD configuration with a ring from SCALANCE switches

The ring consists of SCALANCE switches (redundancy manager and redundancy clients), SIMOTION controllers and IO devices at the stubs. In this configuration, several IO controllers can fail or be deactivated with their IO devices. The rest of the system continues to run smoothly. In addition, it is also possible for a SCALANCE switch to fail, or for the ring to be interrupted.

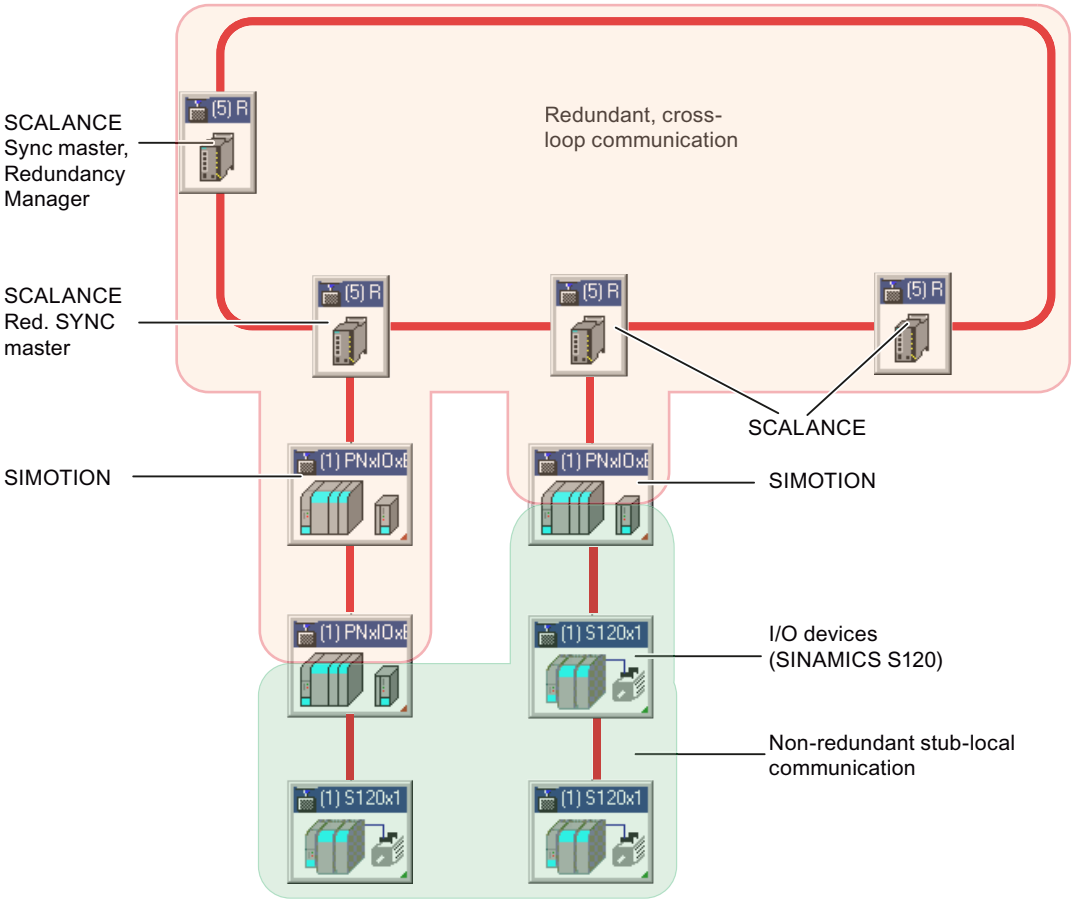


Figure 5-29 Ring configuration with SCALANCES

MRPD configuration with a ring from SIMOTION controllers and a SCALANCE switch

The ring contains a SCALANCE switch as a redundancy manager and SIMOTION controllers as redundancy clients. The SIMOTION controllers with their IO devices are connected with the ring via spur lines.

In this configuration, an IO controller can fail or can be deactivated with its IO devices (at the stub). The rest of the system continues to run smoothly.

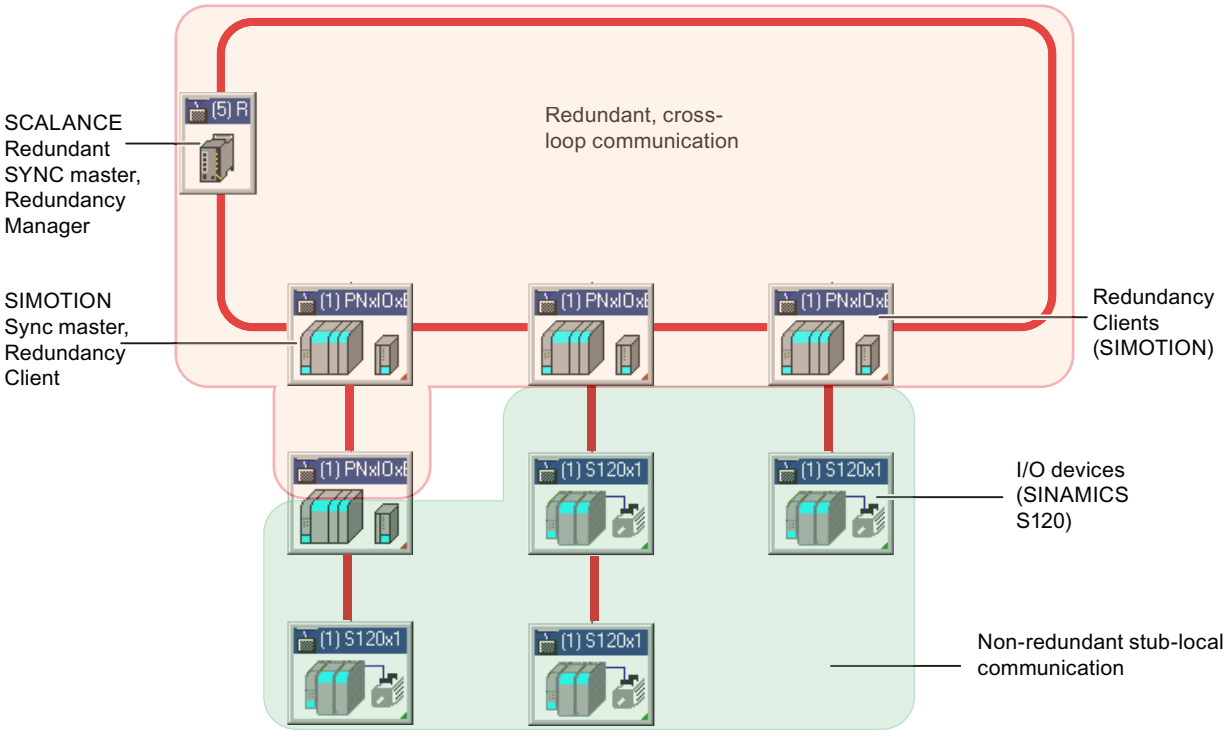


Figure 5-30 Ring configuration with SCALANCE and SIMOTION

MRPD configuration with SCALANCE, SIMOTION controller and SINAMICS I/O devices

The ring contains a SCALANCE switch, a SIMOTION controller and SINAMICS IO devices. An IO device can fail or can be deactivated. The rest of the system continues to run smoothly.

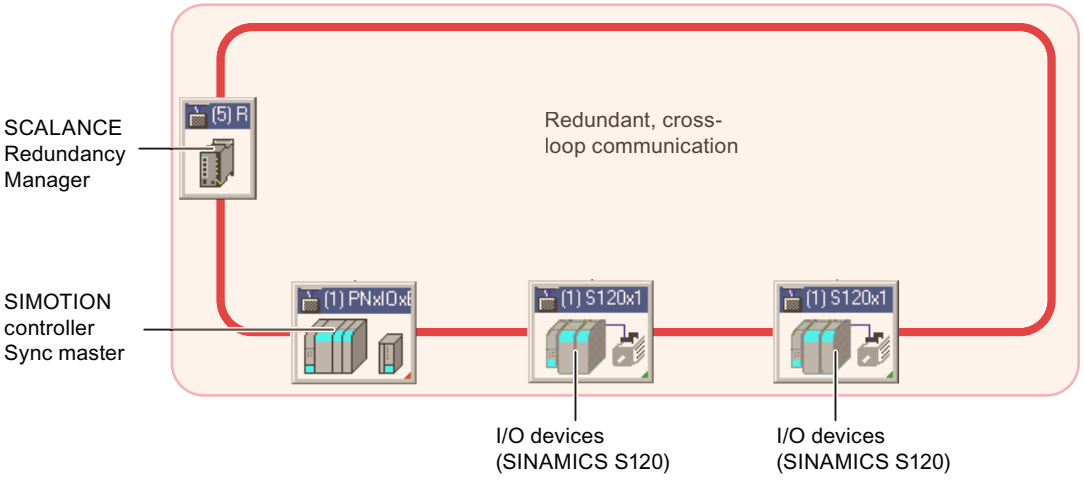


Figure 5-31 Ring configuration with SINAMICS I/O devices

Hierarchical multi-ring system

If, for example, the number of stations per ring or cycle times is exceeded, you can establish hierarchical multi-ring systems and divide the functionality into sub-rings. The sub-rings can continue to operate independently of each other.

For an example, see also Redundant sync master (Page 92).

5.2.8.4 Use of Safety with MRRT

No smoothness for Safety via RT and shared device

MRRT offers no smooth redundancy for RT data (but only for IRT data).

In order to guarantee smooth media redundancy with Safety, the safety data within the MRRT ring must be transferred as IRT data. To make sure that this is the case, the SIMOTION controller F-proxy must be used for communication with the F-CPU. Please note that a direct connection must be established between the F-CPU and the SIMOTION controller. Either a free port or the second PROFINET interface, which is not part of the MRPD ring, can be used for this purpose. The Safety RT data of the F-CPU is then transferred internally from the SIMOTION controller into the IRT data of the MRPD ring.

Note

The use of the shared device (the F-CPU communicates directly with the SINAMICS drive) is not recommended for MRPD, because the RT data deactivates temporarily in the event of a fault and so the PROFIsafe modules are passivated. Use of the I-device F-Proxy is recommended, because the PROFIsafe data is packed together in the IRT data.

Safety via I-device F-Proxy

When using the I-device F-Proxy, the safety data is transmitted from the F-Proxy to the drives in the MRPD ring via PROFINET IRT. The PROFIsafe communication between the F-CPU and the drives is thus also secured in case of a fault with the ring.

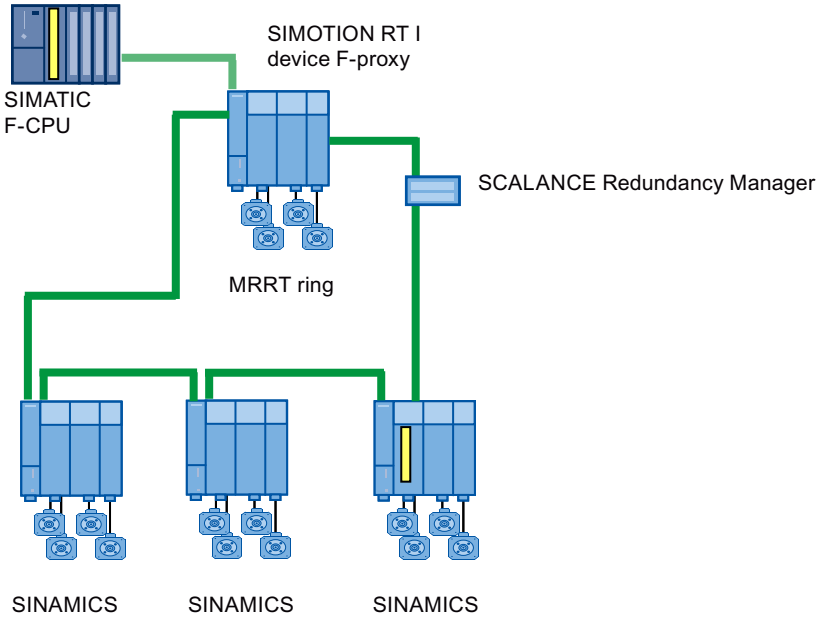


Figure 5-32 Safety via MRRT

In order for the safety data to be transferred smoothly, the communication paths between the F-CPU and the SIMOTION I-device F-Proxy may not go via the ring. As seen in the illustration above, the F-CPU should either be connected directly with an open port of the PN interface of the SIMOTION controller, which has been configured as an I-device F-Proxy, or the second PN interface be used as an I-device F-Proxy. For an exact description of how to configure an I-device F-proxy, please refer to section Principles of I device failsafe proxy (Page 276).

You can find a description of how to configure Safety via PROFINET with an F-CPU in the FAQ (http://support.automation.siemens.com/WW/view/en/50207350).

See also

Two PROFINET IO interfaces (Page 77)

PROFIsafe via PROFINET with an F-CPU (Page 281)

5.2.9 PROFINET V2.3 Performance Upgrade

5.2.9.1 Optimized data transfer / 125 µs PROFINET send clock

PROFINET V2.3 Performance Upgrade

All SIMOTION control units support PROFINET in accordance with the PROFINET International Specification PN V2.3. The optional **Performance Upgrade** functionality, which provides an extended range of functions, is available for SIMOTION D455-2 DP/PN.

- Reduction of the minimum PROFINET send clock of 250 µs to 125 µs
- Optimization of data transfer (e.g. by Dynamic Frame Packing - DFP (Page 108))
- Fast forwarding
- Fragmentation

Note

You will find further information on this on the web site of the PI Organization (http://www.profibus.com) and the following technical specifications:

- Technical Specification for PROFINET, Version 2.3 – Date: October 2010, Order No.: 2.722
- Application Layer services for decentralized periphery and distributed automation, Technical Specification for PROFINET, Version 2.3 – Date: October 2010, Order No.: 2.712

With optimized data transfer, the following options are possible:

- More devices can be operated with the same cycle time.
- The cycle time can be reduced for the same number of devices.

Requirements

General conditions for using the 125 µs send clock:

- the 125 µs send clock is only supported by the onboard PROFINET interface X150 in conjunction with the Servo_fast/IPO_fast
- only X150 Port 1 can be used; Ports 2 and 3 are deactivated (for TCP/IP and UDP communication too)
- X150 can only be used as a PROFINET IO controller (not as an I-device)
- no media redundancy MRP/MRPD (only 1 port is available on X150)
- Use of STEP 7 V5.5 SP4; for other rated conditions, see also *Readme* for SIMOTION SCOUT V4.4

Components that can be used

The 125 µs send clock is only supported by selected PROFINET nodes. (e.g. by the ET 200SP I/O system). For the 125 µs send clock, no SCALANCE switches are currently available.

- SIMOTION D455-2 DP/PN as of V4.4
- SINAMICS S120 CU320-2 PN + CBE25 as of V4.7
- SINAMICS S120 CU320-2 DP + CBE25 as of V4.7

Note

For the PROFINET nodes that support a send clock of 125 µs and the conditions that have to be met, see the relevant product documentation.

5.2.9.2 Dynamic Frame Packing - DFP

Description

For PROFINET V2.2, a separate Ethernet frame is used for each IO device for the cyclic transfer of the IO data. This leads to a large overhead due to the Ethernet header, especially in the case of IO devices with a small amount of IO data.

With the PN V2.3 Performance Upgrade, SIMOTION D455-2 DP/PN (as of V4.4) supports Dynamic Frame Packing (DFP) via the onboard PROFINET interface x150. With Dynamic Frame Packing, the cyclic data of multiple IO devices is summarized in one Ethernet frame. DFP significantly reduces the bandwidth required to exchange the cyclic data, especially in the case of IO devices with a small amount of IO data. This makes it possible to either operate more IO devices with the same cycle time or to reduce the cycle time in order to achieve a shorter response time.

In the outbound direction (from the IO controller to the last IO device), the Ethernet frame is automatically reduced. This is achieved by having each device remove its own data from the telegram when forwarding. In the inbound direction (from the last IO device to the IO controller), each IO device adds its own data to the Ethernet frame.

Dynamic Frame Packing is automatically used when send clocks < 250 µs are set and the requirements (see "Requirements" section) are met.

The figure below shows examples of how package groups can be formed. The IO devices (IO-D) with red backgrounds are automatically grouped into package groups.

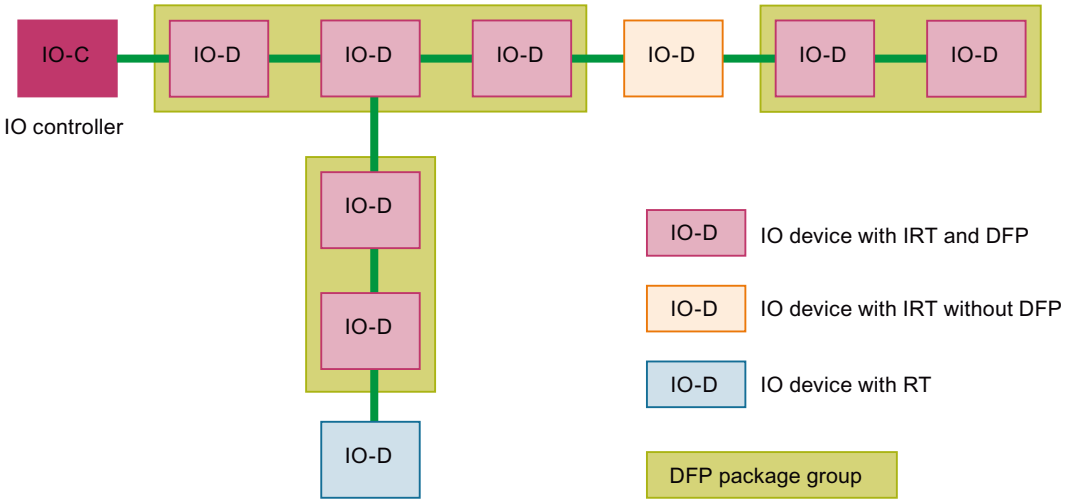


Figure 5-33 Example of DFP package group formation

Requirements

DFP can only be used with the following restrictions:

- DFP can only be used on the onboard PROFINET interface X150 of the SIMOTION D455-2 DP/PN and the IO devices SINAMICS S120 CBE25 and ET200SP High Speed.
- DFP is supported by SIMOTION only as an IO controller. The SIMOTION I-device does not support DFP.
- A DFP device can be used by a second IO controller as a shared device; the shared device restrictions apply. One IO controller communicates with DFP via IRT; the other via RT.
- In the case of MRPD, the DFP devices must be present in a spur line. IO devices in the ring are not supported.
- IO devices that do not support DFP finish a package group; i.e. the grouped IO devices must form a line with no gaps.
- A maximum of 63 IO devices can be packed in one group.
- The size of a package frame is limited to 128 bytes in the outbound direction and 256 bytes in the inbound direction.

Note

Delivery release

At the time of the delivery release of SIMOTION V4.4, all IO devices that support the Performance Upgrade for PROFINET with 125 µs cycle time are still under development. A general release of the functionality is planned for SIMOTION D455-2 DP/PN, as soon as the release for the I/O system ET200SP and the drive system SINAMICS S120 (CU320-2 with CBE25) is complete.

Until the general delivery release, configuration in STEP 7 HW Config is disabled. If you wish to use the function before this time in test or pilot applications, please ask your Siemens contact.

Please also note the product Information for SIMOTION, SINAMICS S120, and SIMATIC ET 200SP High Speed.

See also

Optimized data transfer / 125 µs PROFINET send clock (Page 107)

5.3 Configuring PROFINET IO with SIMOTION

5.3.1 New in SIMOTION SCOUT as of V4.4

Shared iDevice

As of SIMOTION V4.4, an iDevice of the PN-IO interfaces can be operated with two higher-level IO controllers. This functionality is referred to as "shared iDevice" and is comparable with the shared device.

See also Shared iDevice and PROFIsafe (Page 310)

PROFINET V2.3 Performance Upgrade

The Control Unit SIMOTION D455-2 DP/PN supports the PROFINET Performance Upgrade in accordance with the PROFINET specification V2.3. The PROFINET send clock can now be set to a minimum of 125 µs.

See also Performance Upgrade PN V2.3 (Page 107)

Support for PROFIenergy

PROFIenergy is a data interface based on PROFINET. It allows loads to be shut down during idle time in a controlled, centralized manner, and irrespective of the manufacturer and device.

See also PROFIenergy (Page 213)

Setting IP address and device name via user program

In the case of SIMOTION, the IP configuration can be set via the user program for the PN-IE and PN-IO interfaces.

See also Setting IP address and device name via UP (Page 154)

Identification and maintenance (I&M) data

As of V4.4, SIMOTION supports the data sets 1-4 in addition to the I&M data set 0 (electronic rating plate) as a PROFINET iDevice.

See also I&M data (Page 169)

PROFINET IO with IRT

As of SIMOTION V4.4, an additional configuration step has been added for the configuration of PROFINET IO with IRT. In the HW Config of the controller CPU, under **Object properties** in the **Isochronous tasks** tab, the setting is made as to whether the servo task is to be assigned to the PROFINET interface as the application level for the processing of isochronous data:

- If PROFINET IO devices with isochronous data are to be configured on PROFINET, the PROFINET interface must first be assigned to the servo under **Object properties** of the SIMOTION CPU in the **Isochronous tasks** tab (Use "..**I/O data isochronously to the servo**" checkbox). Only then can servo be assigned for the device in the **Object properties** of the PROFINET interface in the **I/O cycle** tab (**Assign I/O device isochronous** list box).
- If no devices are configured with isochronous data on the PROFINET interface, the PROFINET interface must also not be assigned to the SERVO.

As of SIMOTION V4.4, it is therefore also possible to configure the PROFINET interface of SIMOTION devices with IRT without the data being transferred isochronously. This means that the data is transferred synchronously via IRT on the bus, but the applications in SIMOTION and in the device do not accept the data isochronously. For example, an ET200HF can be configured with IRT, but the I/O data is not transferred isochronously to the physical input or output. Or a SIMOTION I-device can be connected via IRT to a higher-level controller, but the input and output data of the I-device interface is not processed isochronously by the CPU.

Up to and including V4.3, SIMOTION always performed the communication with PROFINET IO with IRT isochronously to the servo, even if isochronous mode was not configured on any PROFINET IO device.

5.3.2 New in SIMOTION SCOUT as of V4.3

Media redundancy protocols MRP and MRPD

Via the so-called Media Redundancy Protocol (MRP or MRPD), it is possible to establish redundant networks, see Media redundancy for SIMOTION (Page 97).

Second PROFINET I/O interface

With SIMOTION D4x5-2 DP/PN, is it possible to have two PROFINET I/O interfaces (integrated interface and CBE30-2) on one device, see Two PROFINET IO interfaces (Page 77). If two interfaces are available, one interface can be operated in the servo cycle clock and the other interface in Servo_fast cycle clock (integrated interface).

Position control cycle clock and Servo_fast cycle clock

Position control cycle clock and Servo_fast both run via PROFINET (slow and fast bus cycle clock), see Servo_fast, scaling down of cycle clocks to the servo at the PROFINET interface (Page 136).

Standard Ethernet interfaces

The standard Ethernet interfaces now include basic PROFINET services, Properties of the SIMOTION Ethernet interfaces (Page 219).

More message frames for PROFIsafe

Additional message frames are provided for PROFIsafe, see also Message frames and signals in drive-based safety (Page 269).

See also

Media redundancy for SIMOTION (Page 97)

Two PROFINET IO interfaces (Page 77)

Servo_fast, scaling down of cycle clocks to the servo at the PROFINET interface (Page 136)

Properties of the SIMOTION Ethernet interfaces (Page 219)

Message frames and signals in drive-based safety (Page 269)

5.3.3 New with SIMOTION SCOUT V4.2 or higher

IRT synchronization process

SIMOTION with IRT High Performance. With PROFINET IRT, this synchronization process is set automatically.

Controllers with integrated PN interface

With SIMOTION D4x5-2 DP/PN, new SIMOTION D controllers with integrated PN interfaces are available.

Isochronous operation for applications has been simplified.

The typical isochronous applications for Motion Control have been made much more straightforward. The Ti/To time constants can be calculated automatically for all IO devices. The configuration in HW Config has been enhanced to allow the object properties of isochronous tasks to be set to automatic on the controller.

iDevice

The I device functionality has been improved with the release of Step 7 V5.5. If SIMOTION SCOUT projects use a lower version than V4.2, certain points must be considered when upgrading. You can find more information in the I device chapter.

iDevice F-Proxy

Using the I device F-Proxy, you can produce a PROFIsafe configuration with an F host (F-CPU SIMATIC) on PROFINET with SIMOTION devices (SIMOTION D, SIMOTION P3xx, SIMOTION C240 PN) for the lower-level drives.

Second servo cycle clock (Servo_fast)

The second servo cycle clock enables you to operate two bus systems in different application cycle clocks in the case of SIMOTION D4x5-2 DP/PN. An assigned servo cycle clock and IPO cycle clock are available for each of the two application cycle clocks. This allows you to divide your application into a slow part (SINAMICS_Integrated with servo and IPO) and a fast part (IO devices on the PROFINET interface with Servo_fast and IPO_fast).

5.3.4 Procedure for configuring PROFINET IO with IRT High Performance

Procedure

The following steps need to be performed when configuring PROFINET IO with IRT High Performance:

1. Insert the SIMOTION module.
   Select the **Device** followed by the **Device version**. A PROFINET interface is already integrated in devices labeled with PN.
   - With SIMOTION C and SIMOTION P, you only select the SIMOTION version, e.g. V4.4.
   - With SIMOTION D, the modules have up to two PROFINET IO interfaces (onboard and CBE30-2) depending on the version. Depending on the version, select SIMOTION Version V4.4, the option module, and the SINAMICS drive and version.
     You can select the CBE30-2 option module for PROFINET here or enter it later in HW Config from the hardware catalog under **SIMOTION Drive-based > SIMOTION D4x5 xxx > 6AUxxxxxx > Vx.x SINAMICS > CBE30-x-PN-IO**.
2. Insert IO devices:
   Insert IO devices from the hardware catalog in HW Config into the IO system. The IO devices can be found in the hardware catalog under **PROFINET IO**.
3. Configure sync domain and specify send clock:
   Configure the PROFINET IO node as sync master (clock generator) and define the associated sync slaves. Specify the send clock.
4. Generate the topology:
   Define the topology, i.e. specify how the individual ports of the PROFINET devices (IO controller and IO device) are interconnected with one another. The topology only needs to be configured for PROFINET IO with IRT High Performance. If topology-based initialization is to be used, the topology will need to be configured for all PROFINET devices.
5. As an option, you can still configure the controller-controller data exchange broadcast:
   Specify which address areas are to be used for sending and receiving.

5.3.5 Inserting and configuring SIMOTION D

5.3.5.1 General information on inserting and configuring SIMOTION D

General information

You have created a project and want to configure a SIMOTION D using a PROFINET interface.

With SIMOTION D, the control units feature an onboard PROFINET interface (D410 PN, D410-2 DP/PN, D4x5-2 DP/PN) or an option module CBE30 for PROFINET (D4x5), depending on the version concerned.

With V4.3 or higher, you can use a CBE30-2 as a second PROFINET interface with SIMOTION D4x5-2 DP/PN.

You can select the CBE30-2 when inserting the SIMOTION D controller in the SIMOTION SCOUT project or, alternatively, you can insert the relevant version for the SIMOTION D controller later in HW Config from the HW catalog. You will find the CBE30-2, for example, under **SIMOTION Drive-based > SIMOTION D4x5 xxx > 6AUxxxxxx > Vx.x SINAMICS > CBE30-x-PN-IO**

Implementing the PROFINET interface:

- Already integrated in the case of D410 PN, D410-2 DP/PN, and D4x5-2 DP/PN
- With D4x5-2 DP/PN, a second PROFINET interface can be implemented with a CBE30-2.
- Available as an option in the case of D4x5 (option module CBE30)

The procedure is explained in the following sections.

5.3.5.2 Inserting and configuring SIMOTION D4x5-2 DP/PN/D410 PN/D410-2 DP/PN

General information

You have created a project and want to configure a SIMOTION D4x5-2 DP/PN, SIMOTION D410 PN, or SIMOTION D410(-2) DP/PN using the internal PROFINET interface. The procedure for this will be explained using the example of a SIMOTION D4x5-2 DP/PN.

Procedure

1. Click on **Insert SIMOTION device** in the project navigator to open the device selection dialog box.
2. Select SIMOTION D under **Device** and click on a **Device version** e.g. D455-2 DP/PN.

3. Select the current **SIMOTION Version** , e.g. **V4.4** and **SINAMICS S120 Integrated.**

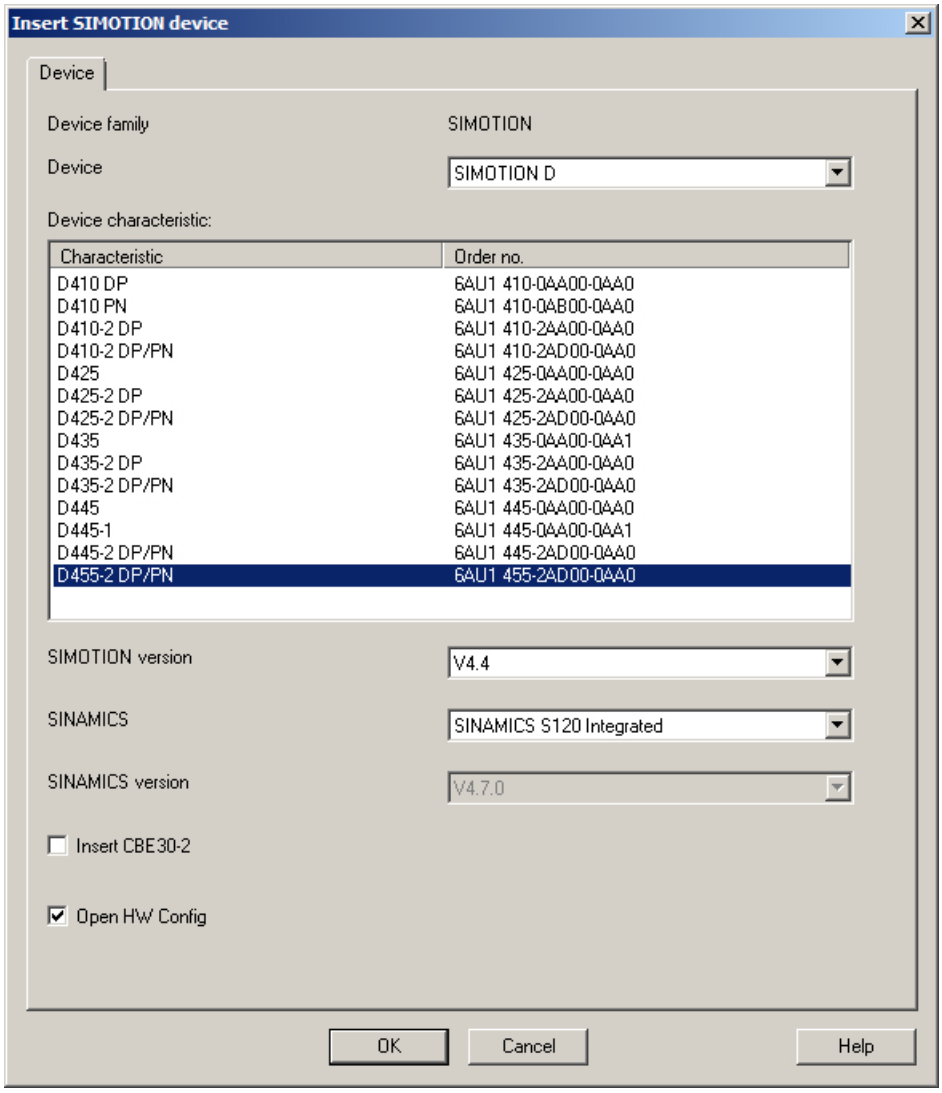


Figure 5-34 Inserting a SIMOTION device

4. Activate the **Open HW Config** check box to add e.g. an option module or an IO system.
5. Click **OK** to confirm.

6. The dialog box for creating a PROFINET subnet will be displayed. Create a new subnet using **New...** and enter the required **IP address** and **subnet mask**. Click **OK** to confirm.

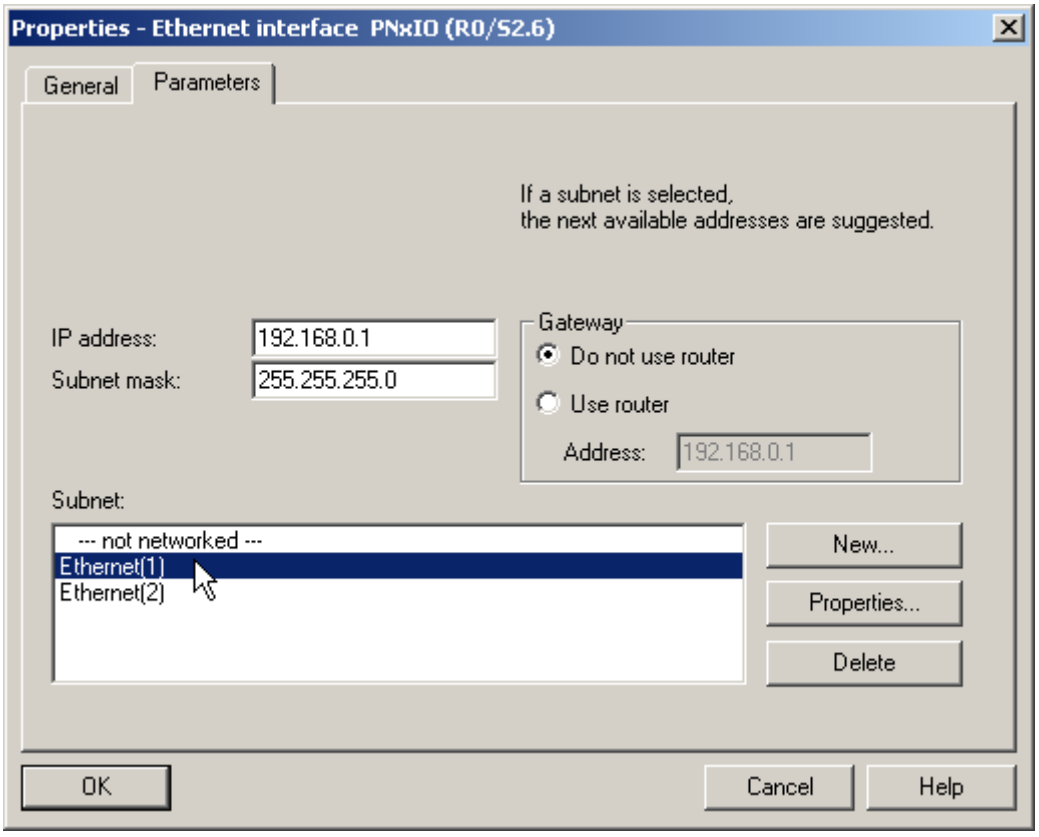


Figure 5-35 Creating a new Ethernet subnet

7. Select the PG/PC interface from the next dialog box and click OK to confirm. The device is inserted, HW Config opens, and the module with the configured PROFINET subnet is displayed.

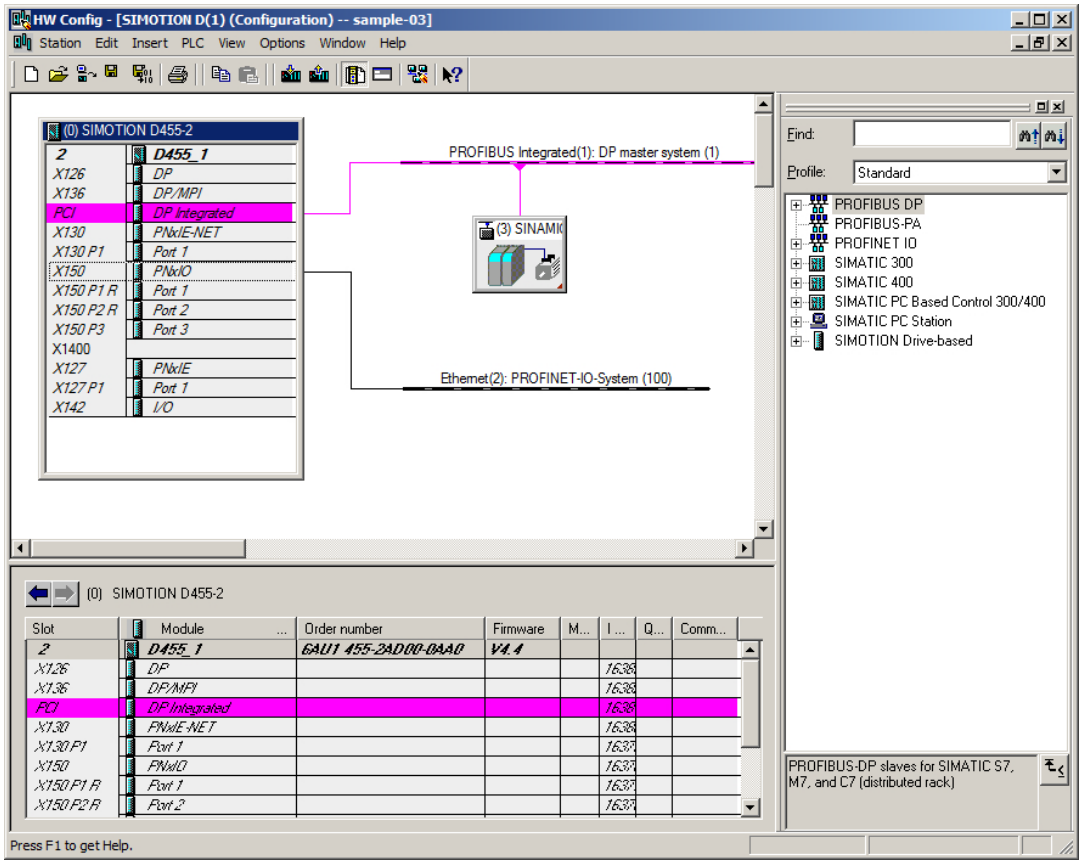


Figure 5-36 SIMOTION D455-2 DP/PN in HW Config

See also

Add and configure PROFINET interface CBE30-2 (Page 120)

5.3.5.3 Insert and configure a SIMOTION D4x5-2 incl. CBE30-2

Requirement

You have already created a project and now want to insert a SIMOTION D4x5-2 with option module CBE30-2 for PROFINET.

Procedure:

1. Click on **Insert SIMOTION device** in the project navigator to open the device selection dialog box.
2. Select SIMOTION D under **Device** and click on the **Device version** e.g. D455-2 DP/PN.

3. Select the **SIMOTION Version**, e.g. **V4.4** and activate the option module **CBE30-2**.
   You can also insert the option module CBE30-2 in HW Config at a later point (see Add and configure PROFINET interface CBE30-2 (Page 120)).

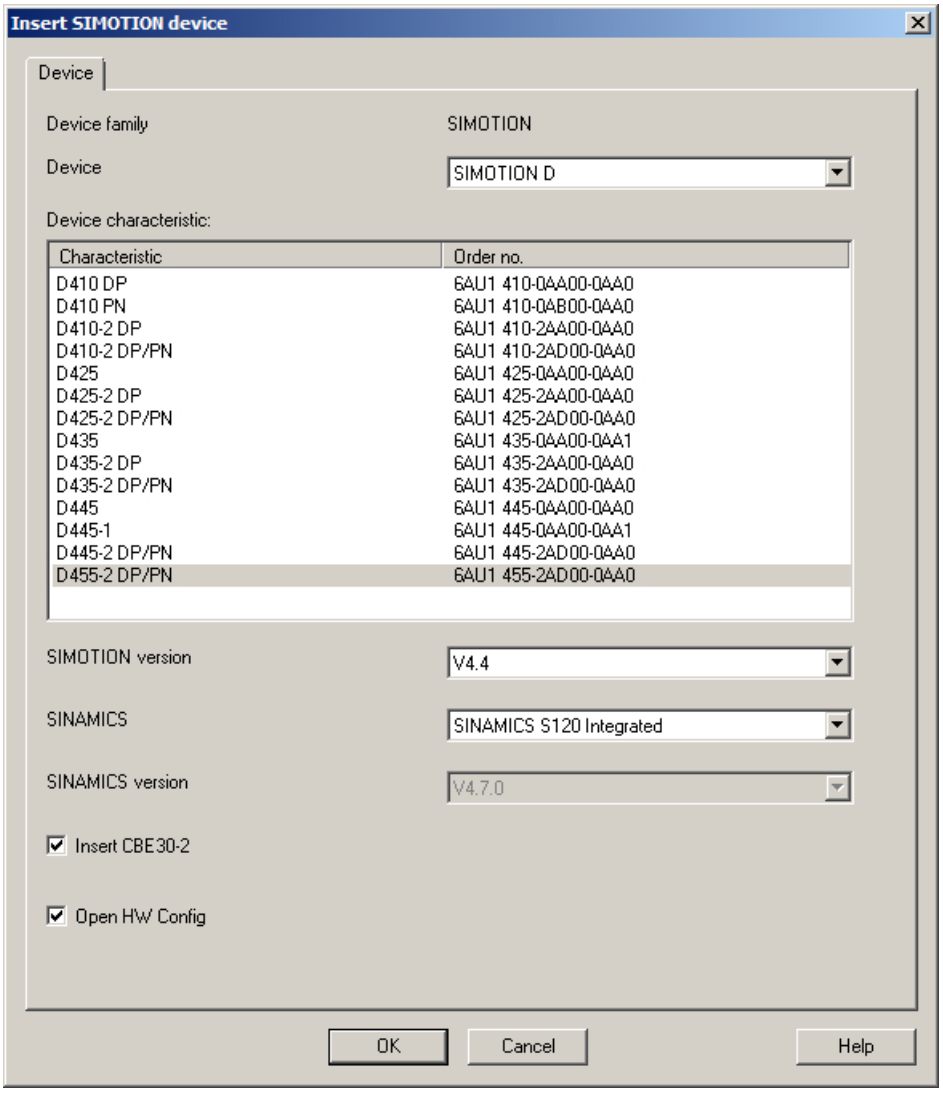


Figure 5-37 Creating a SIMOTION D4x5-2 with CBE30-2 for PROFINET

4. Activate the **Open HW Config** checkbox to insert an IO system, for example.

5. Click **OK** to confirm. The dialog box for creating a PROFINET subnet will be displayed. Create a new subnet using **New...** and enter the required **IP address** and **subnet mask**. Press **OK** to confirm.
6. Select the PG/PC interface from the next dialog box and click **OK** to confirm. The device is inserted, HW Config opens, and the module with the configured PROFINET subnet is displayed.

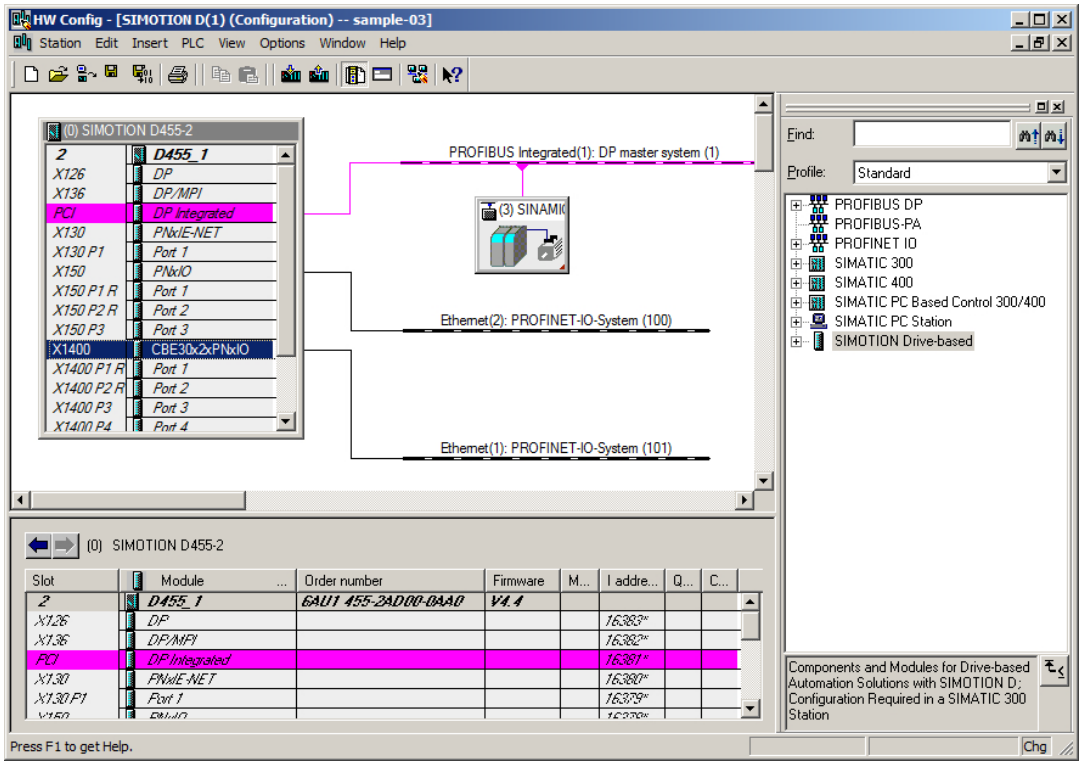


Figure 5-38 SIMOTION D4x5-2 in HW Config

5.3.5.4 Add and configure PROFINET interface CBE30-2

Requirement

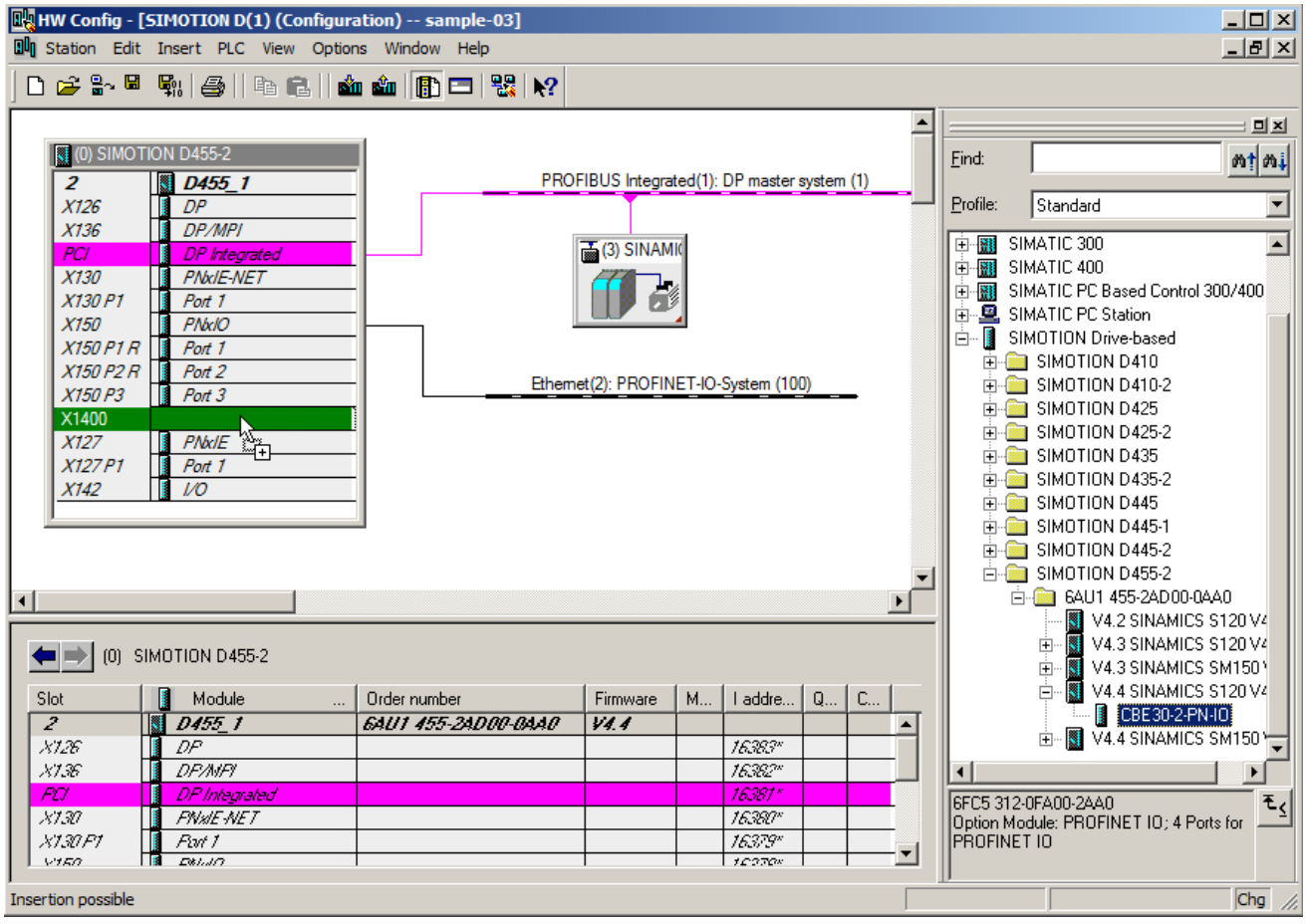
You have already created a project and inserted a SIMOTION D4x5-2 device.

When adding a device in SIMOTION SCOUT, you can add a CBE30-2 as standard. You can also insert the option module in HW Config at a later point.

Procedure

1. In the project navigator, double-click the module (in this case D455-2 DP/PN). HW Config is displayed with the corresponding module.
2. In the hardware catalog, select the appropriate option module under **SIMOTION Drive-based > SIMOTION D4x5-2 > 6AUxx > Vx.x SINAMICSxx > CBE30-x-PN-IO**. Note the CPU type and version.

3. Click PROFINET module CBE30-2-PN. As soon as the appropriate CBE30-PN is selected, the interface **X1400** in the rack turns green.

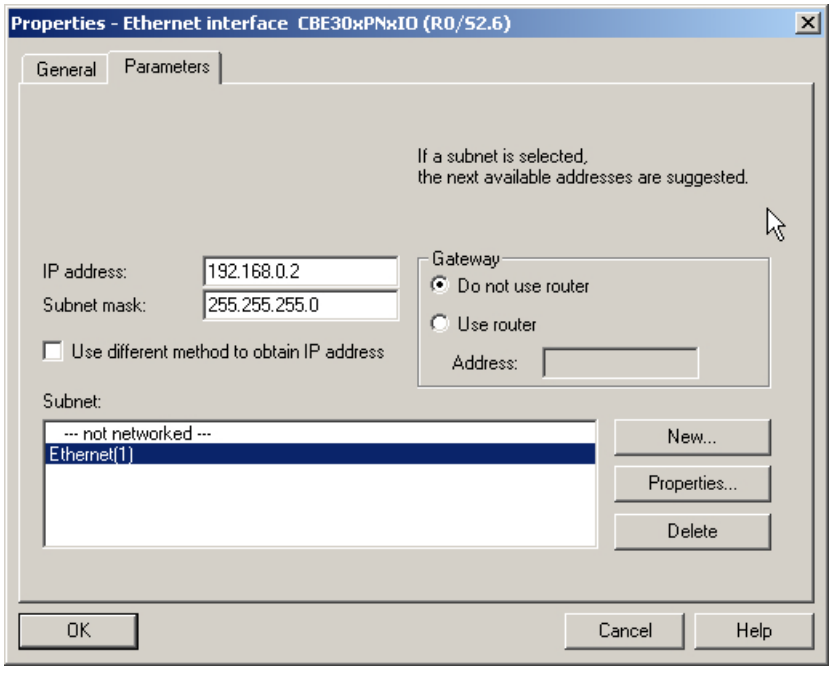


4. Drag the CBE30-2-PN to the corresponding interface of the SIMOTION module (X1400). The **Properties - Ethernet Interface CBE30-2-PN (R0/S2.6)** window opens.
5. Click **New** to create a new subnet. The **Properties – New subnet Industrial Ethernet** dialog box is displayed.

6. Click **OK** to confirm these entries. A new Ethernet subnet will be created, e.g. Ethernet(4).

Note

The IP addresses of the SIMOTION interfaces must be in different address areas.



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

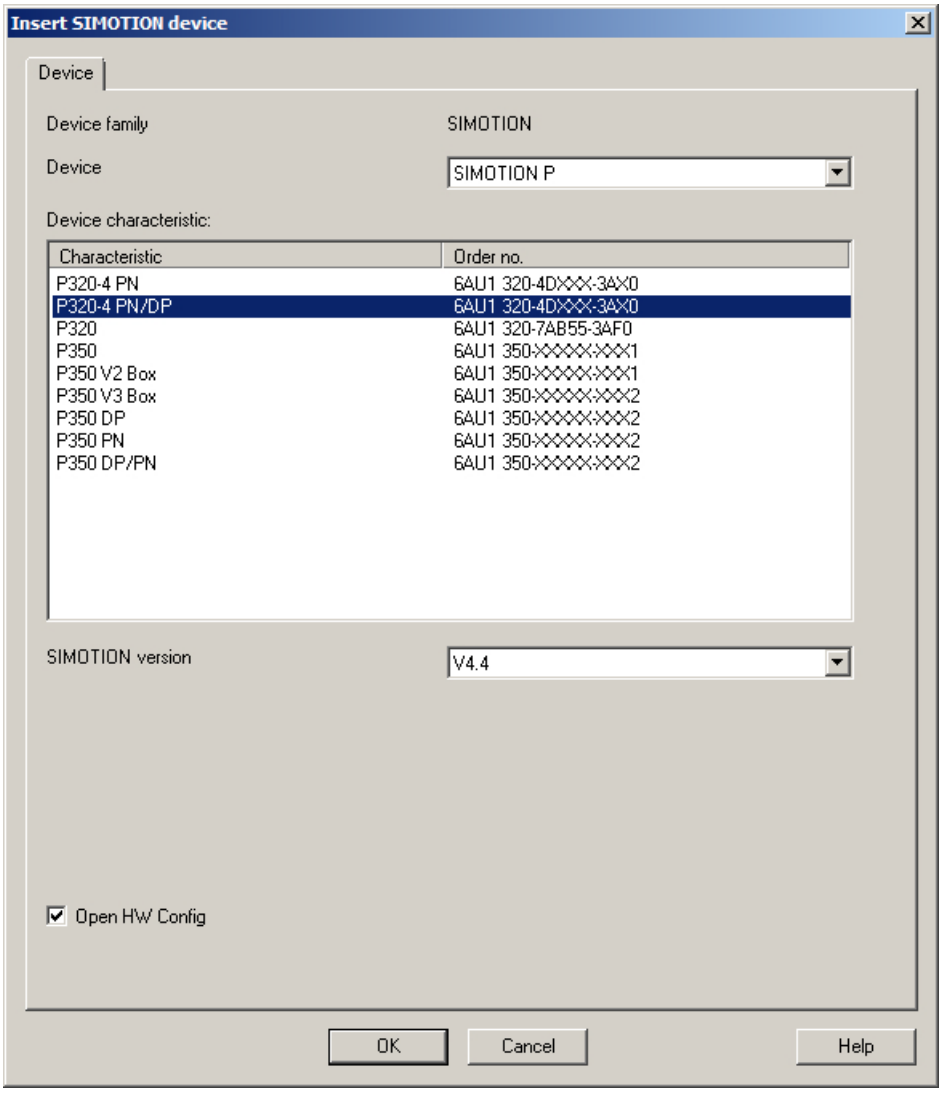
5.3.6 Adding and configuring SIMOTION P

Requirement

You have already created a project and want to insert a SIMOTION P with PROFINET. The procedure for this will be explained using the example of a SIMOTION P320.

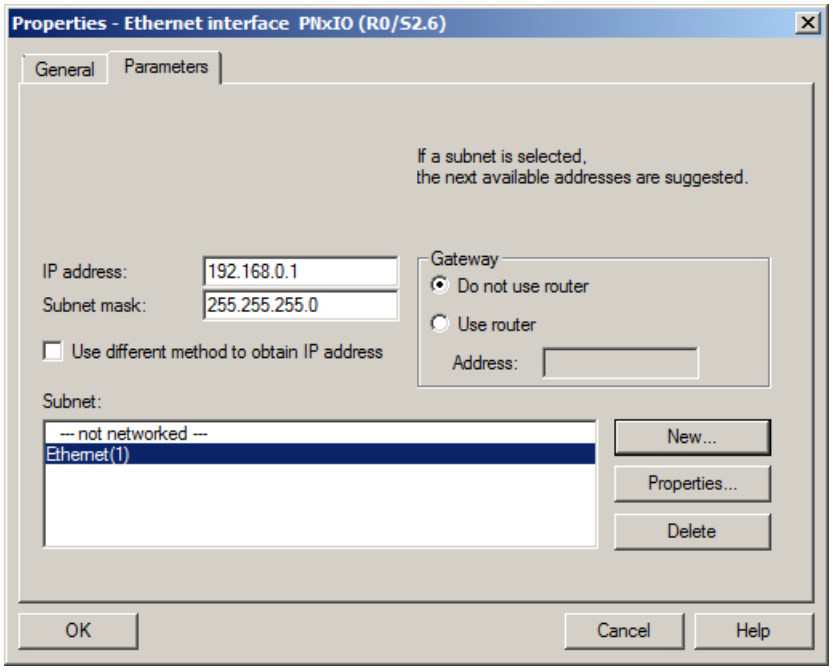
Procedure

1. Click on **Insert SIMOTION device** in the project navigator to open the device selection dialog box.
2. Under **Device**, select SIMOTION P, then click on a **Device version** (e.g. P320-4), and select the **SIMOTION Version**, e.g. V4.4.

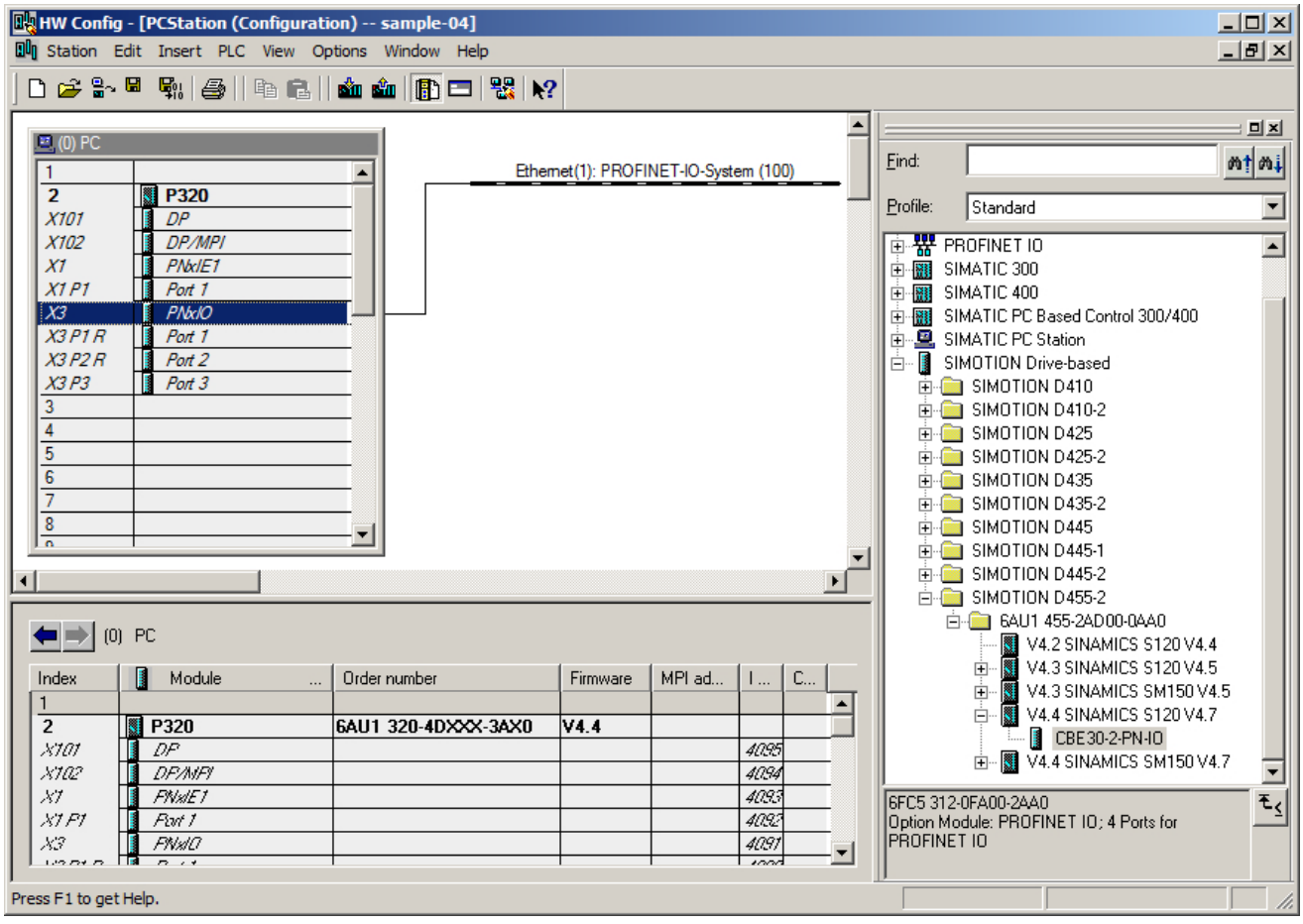


3. Activate the **Open HW Config** checkbox to insert an IO system, for example.
4. Click **OK** to confirm. The dialog box for creating a PROFINET subnet will be displayed.

5. Enter the required **IP address** and the **subnet mask** here. Click **OK** to confirm.



6. 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.



5.3.7 Adding and configuring SIMOTION C

Requirement

You have already created a project and now want to insert a SIMOTION C with PROFINET. The procedure for this will be explained using the example of a SIMOTION C240 PN.

Procedure

1. Click on **Insert SIMOTION device** in the project navigator to open the device selection dialog box.
2. Under Device, select **SIMOTION C**, then click on a **Device version** (e.g. **C240 PN**), and select the **SIMOTION version**, e.g. V4.4.

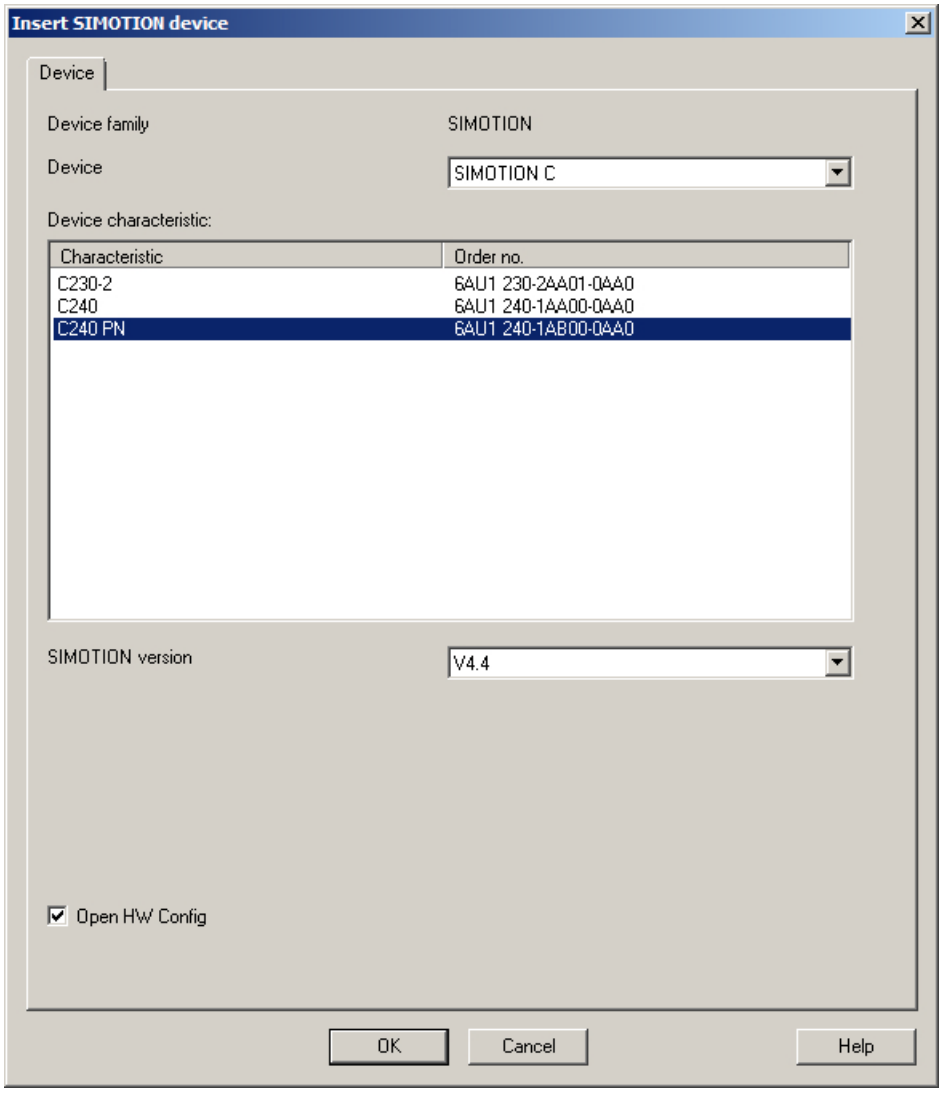


Figure 5-39 Creating a new C240 PN device

3. Activate the **Open HW Config** checkbox to insert an IO system, for example.

4. The dialog box for creating a PROFINET subnet will be displayed. Enter the required **IP address** and the **subnet mask** here. Click **OK** to confirm.

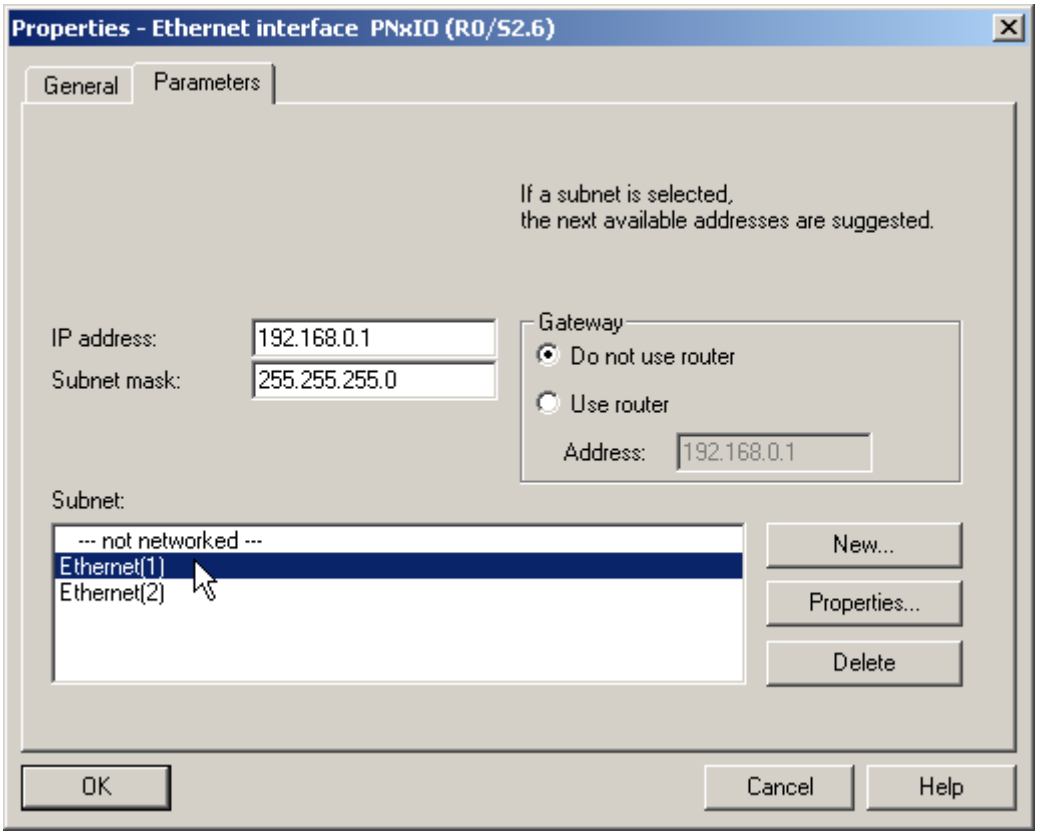


Figure 5-40 Creating a new Ethernet for C240 PN

5. 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.

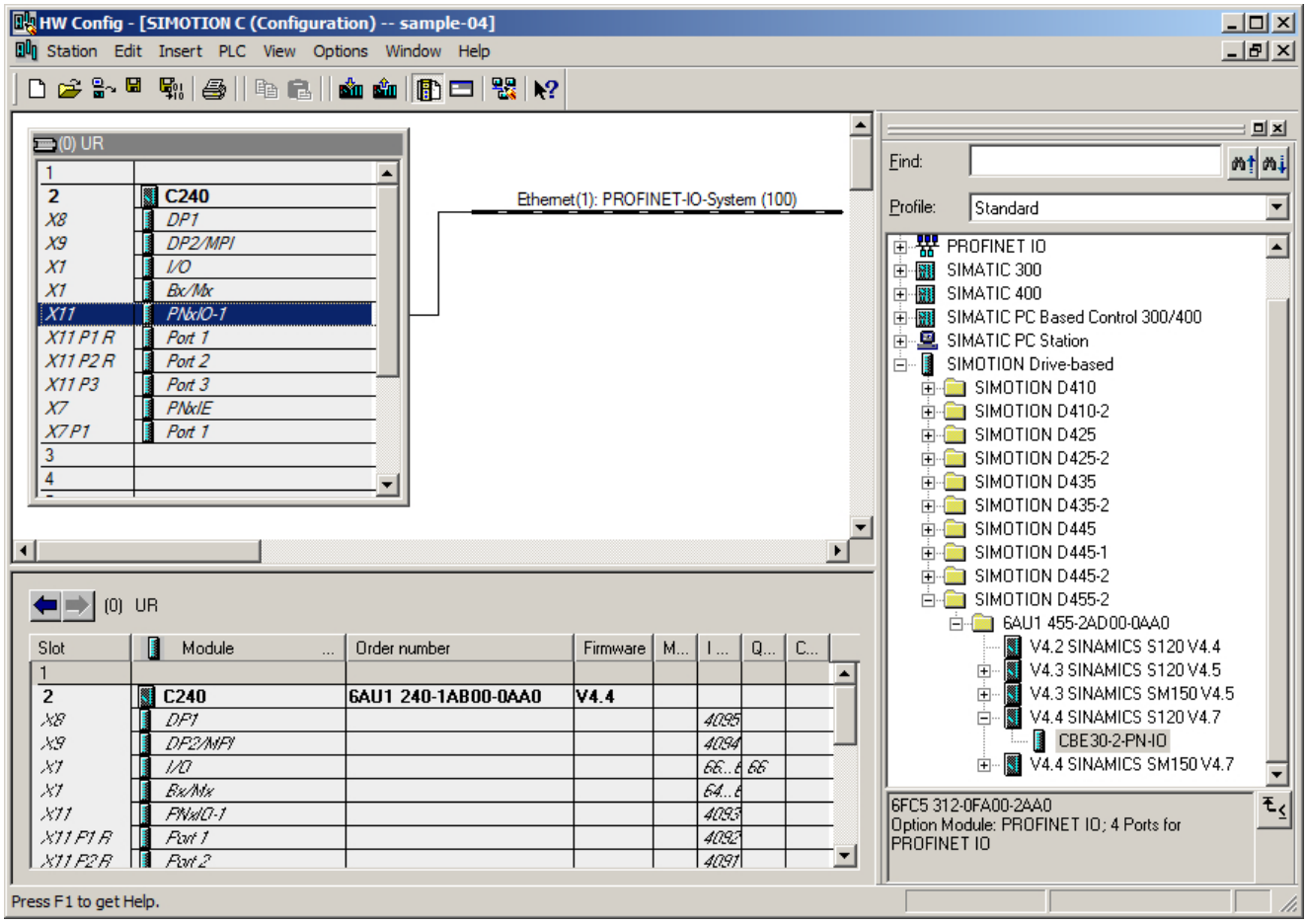


Figure 5-41 HW Config with PROFINET for C240 PN

5.3.8 Creating a sync domain

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

Note

All devices that exchange data via PROFINET IRT must belong to a single sync domain and be directly connected to one another. The connection may not be interrupted by devices that do not support PROFINET IRT, because otherwise the synchronization information cannot be transferred.

Procedure

1. In HW Config, open the station with the PROFINET devices to be involved in IRT communication and select, for example, the PROFINET interface **PNxIO** in the case of a SIMOTION D455-2 DP/PN.
2. Select the **Edit > PROFINET IO > Domain Management** menu command. A dialog tab with the list of all the devices opens. A default sync domain is created and the devices are already assigned.
3. Select the station in the upper field and in the lower field double-click the device that is to be configured as the Sync-Master, e.g. SIMOTION D. This opens the Properties dialog box for the device.

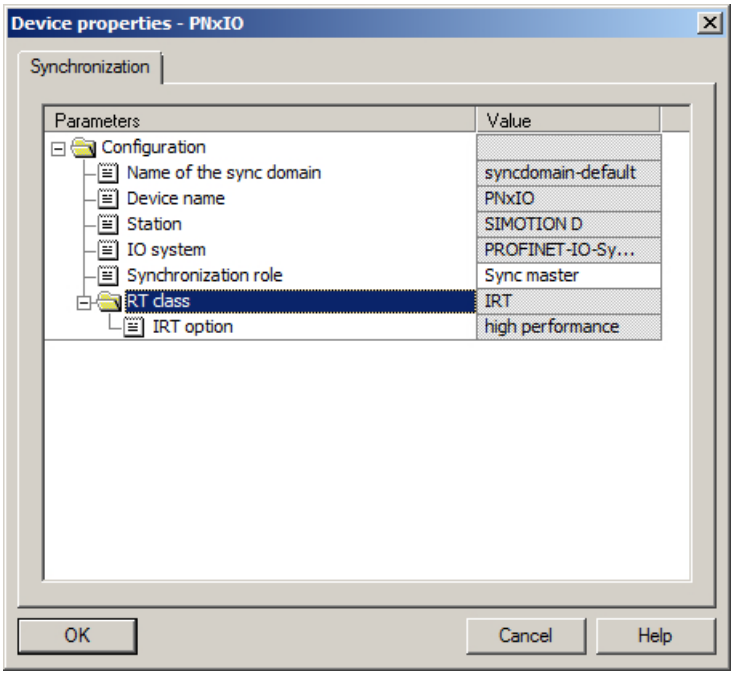


Figure 5-42 Selecting synchronization

4. Set the synchronization type to **Sync-Master**. IRT High Performance is the standard setting on SIMOTION controllers.
5. Confirm the settings by clicking **OK**.
6. Then, select all devices which are to be configured as sync slaves (keep the Ctrl key depressed and select the devices one after the other).
7. Then, click on the **Properties device** button.
8. Set the synchronization type to **Sync-Slave** in the dialog box.
9. Confirm the settings by clicking **OK**.

Note

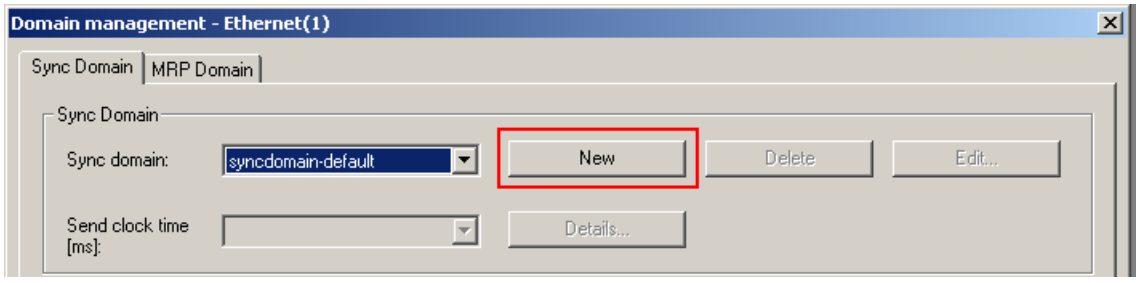
Any devices for which **not synchronized** is selected will not be involved in IRT communication, but will automatically take part in RT communication.

Creating a sync domain and changing the name of a sync domain

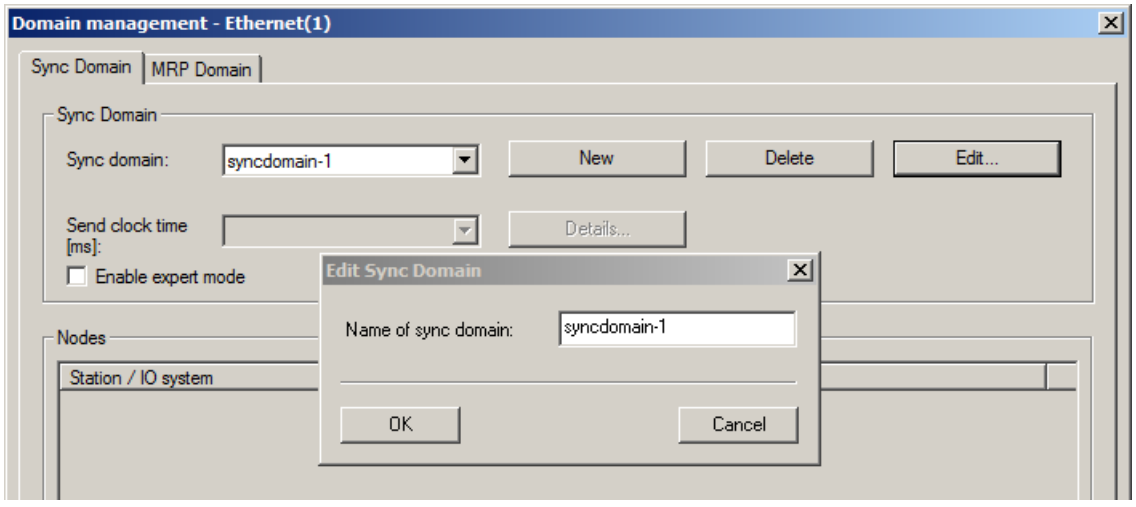
If you work with two PROFINET IO interfaces, they must be in different sync domains. The two sync domains must not have the same name.

To create a new sync domain and change the name of a sync domain, proceed as follows:

1. Open HW Config and right-click on a device of the sync domain.
2. In the shortcut menu, select **PROFINET IO Domain Management**.
   The dialog **Domain Management - PN-xxxx** opens.

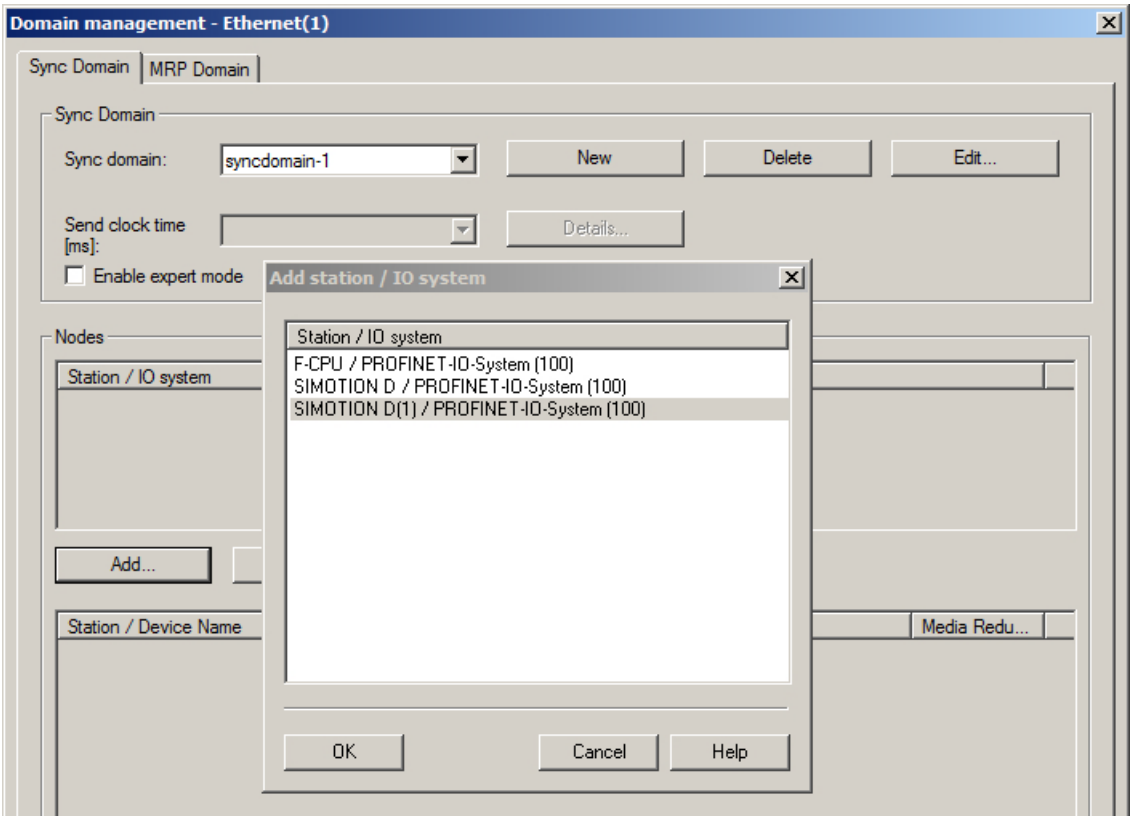


3. Click **New** to create a new sync domain.
4. Select the newly created sync domain in the **Sync Domain** drop-down list box.
5. Click the **Edit** button to open the **Edit Sync Domain** dialog box.



6. Enter a new name for the sync domain and click on **OK**.
   The name of the sync domain is changed.

7. Click on **Add** to add a station or an I/O system to the sync domain.



8. Click **OK** to add the station / IO system.

5.3.9 Defining send clock and refresh times

PROFINET RT and IRT are forms of cyclic communication; the basic clock is the send clock. The update time is a multiple (2^n) of the send clock and the devices are supplied with data in this cycle clock. Individual update times can be set for each device.

Note

With IRT High Performance, the devices are supplied with data during each sending cycle (sending cycle = update time). However, the data is only evaluated every nth cycle in the SIMOTION Servo cycle clock (Servo_fast runs scaled-down to the servo cycle clock) or SINAMICS_Integrated.

As far as SINAMICS_Integrated is concerned, the controller can be used to define an application cycle in such a way that it is only supplied with new data every n cycles.

Configuring a send clock for PROFINET IRT

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

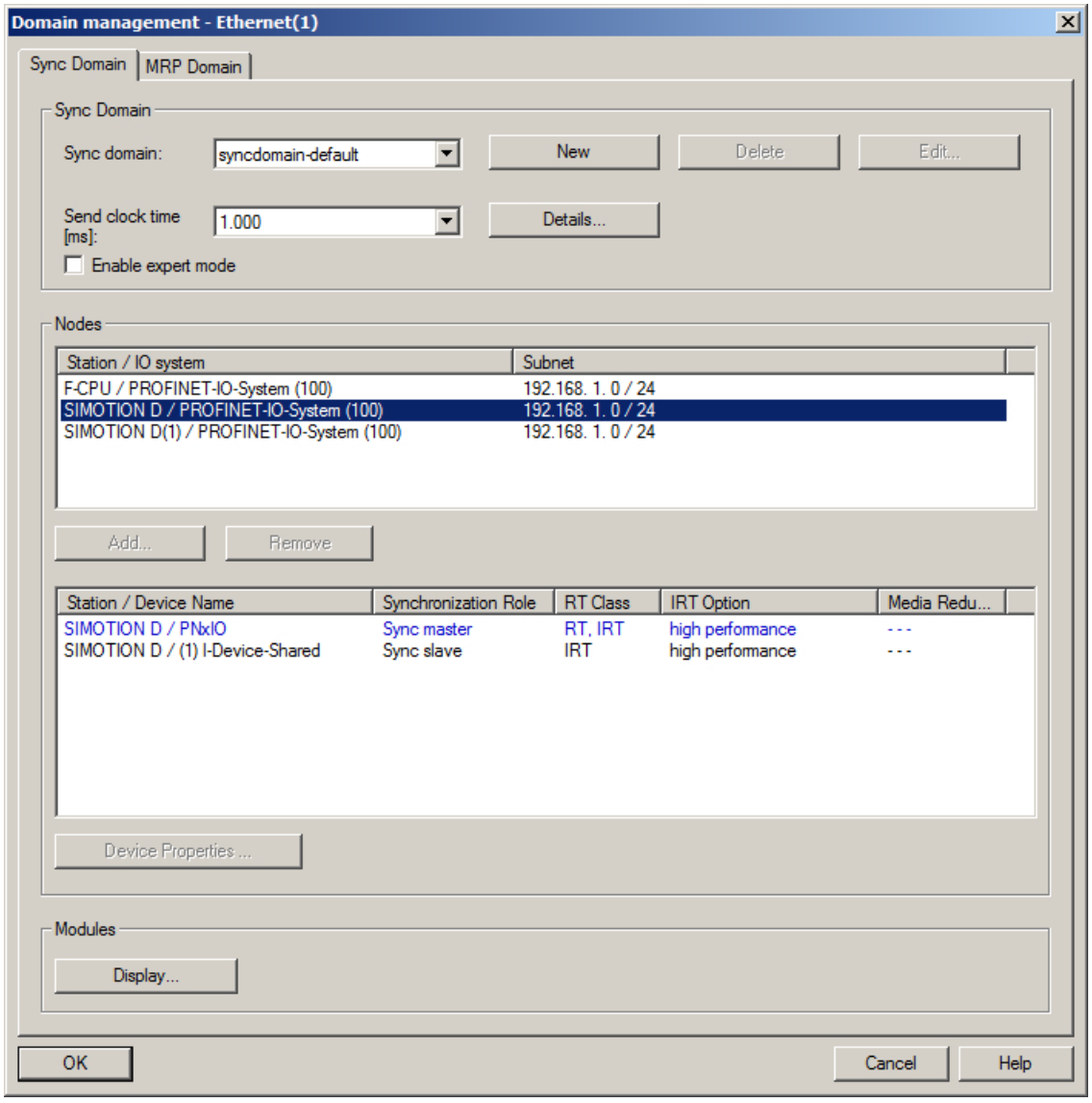


Figure 5-43 Domain Management

2. Select an appropriate send clock for the process. The transmission cycle clock is the smallest possible transmission interval. The send clock is preset to 1 ms.

Note

Communication should be no faster than required, irrespective of what the maximum communication speed may be. This will reduce the bandwidth requirement and the relieve the load that the devices need to support.

Configuring a send clock for PROFINET RT

1. Double-click on the PN interface. The **Properties** dialog box opens.
2. Switch to the PROFINET tab and select a **send clock**. The send clock can only be set if IO controllers and IO devices are not synchronized.

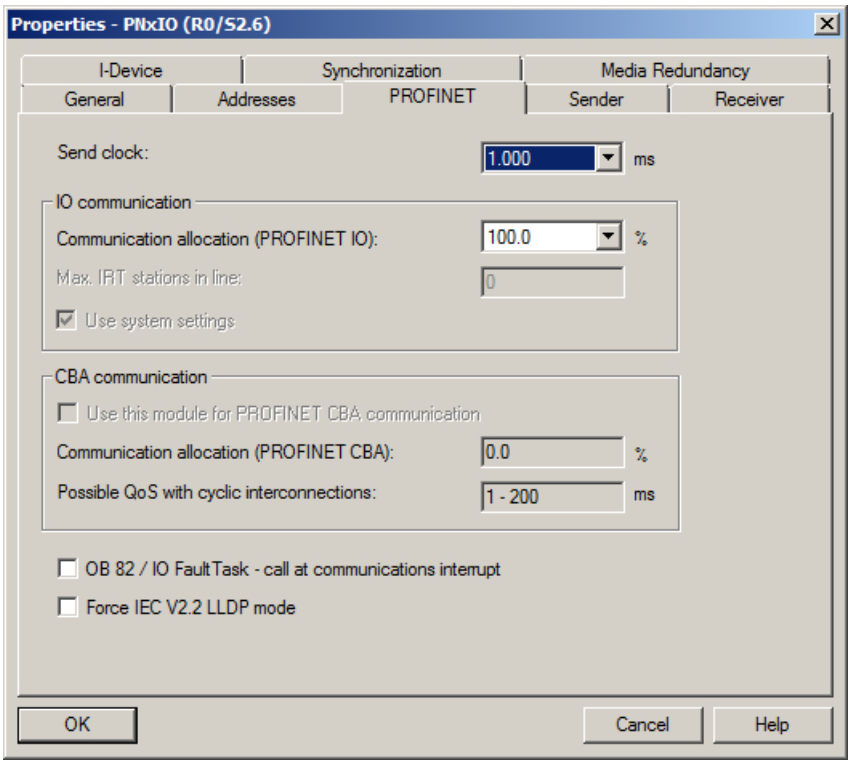


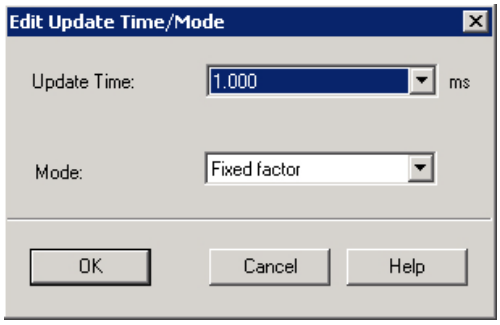
Figure 5-44 Configuring a send clock for PROFINET RT

Defining the update times

The update times for the data exchange of the IO devices is set in the **Properties PROFINET IO System** dialog box.

1. Click the path for the PROFINET IO system in HW Config and select **Object properties** from the context menu. The dialog is displayed.
2. Switch to the **Update time** tab and highlight the respective IO device for which you want to set the update time.

3. Click **Edit**. You can select the refresh time in the **Edit refresh time** dialog.



4. Click **OK** to confirm.

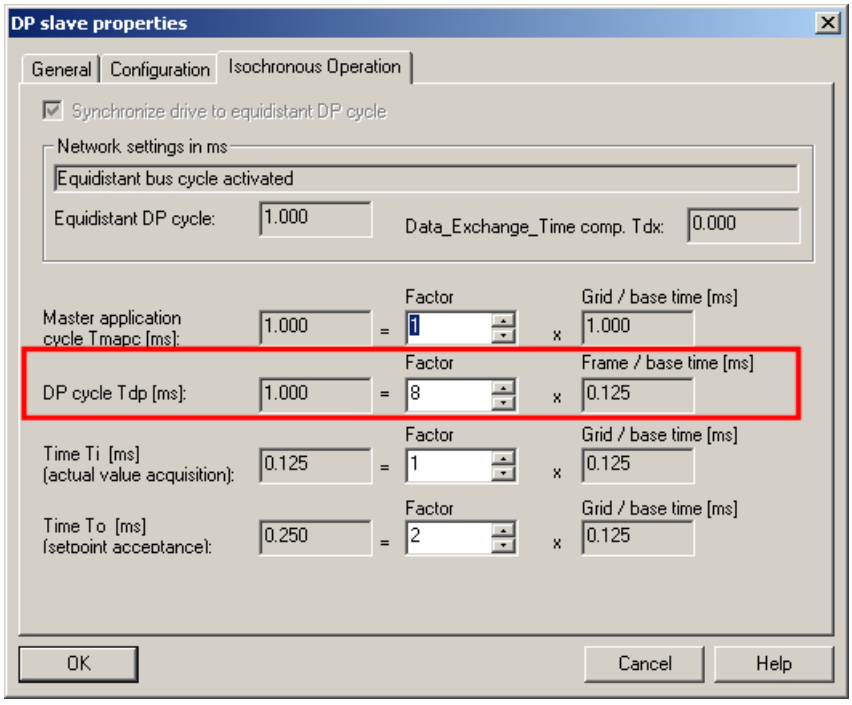
Ratio of PROFINET IRT send clock and DP cycle on SINAMICS_Integrated (SIMOTION D)

If a SINAMICS drive is being operated on a SIMOTION D via PROFINET IO IRT High Performance, the DP cycle of the SINAMICS_Integrated must be the same as the position control cycle clock. The DP cycle is set to 3 ms by default. In most cases, this will not be the same as the servo cycle clock selected for PROFINET applications. If the DP cycle does not correspond to the position control cycle clock, an error message is generated in SIMOTION SCOUT during the consistency check.

You set the DP cycle in the DP slave properties:

1. Double-click SINAMICS_Integrated in HW Config. The **DP slave properties** window appears.
2. Switch to the **Isochronous Operation** tab and activate the "Synchronize drive to equidistant DP cycle" checkbox.

3. Set the factor for **DP cycle Tdp**. The DP cycle must be the same as the position control cycle clock. For example, if the position control cycle clock is 1 ms, you will need to enter a factor of 8 (8 x [base time of 0.125 ms] = 1 ms).



4. Click **OK** to confirm.

When IO devices are operated on PROFINET the position control cycle clock must always correspond to the PROFIBUS cycle clock. The position control cycle clock and the PROFIBUS cycle clock can be scaled to the PROFINET cycle clock.

Example:

PROFINET send clock = 0.5 ms

PROFIBUS cycle clock = position control cycle clock = 1 ms

The PROFIBUS cycle clock can be operated relative to the PROFINET cycle clock at a ratio of 1:1 to 1:64 (max. 8 ms).

5.3.10 Servo_fast, scaling down of cycle clocks to the servo at the PROFINET interface

Cycle clock scaling to the Servo using Servo_fast

The second position control cycle clock enables you to operate two bus systems in different application cycle clocks. An assigned servo cycle clock and IPO cycle clock are available for each of the two application cycle clocks. This enables you to divide your application into slow and fast sections (Servo_fast and IPO_fast).

- The I/O on the fast bus system are used isochronously in the fast Servo_fast/IPO_fast.
- The I/O on the slow bus system are used isochronously in the slow Servo/IPO/IPO_2.
- Servo_fast is only intended to be used with fast IOs and drives.

Options available for isochronous use

For isochronous use, the application cycle is set in HW Config on the I/O module:

- The MAPC (Master Application Cycle) is set for DP slave. Only MAPC=1 is supported if you are also using PROFINET IO. All PROFIBUS devices run in the slow position control cycle clock. The master application cycle describes the reduction ratio between the DP cycle and the cycle of a higher-level controller, e.g. the servo clock of a SIMOTION controller.
- The CACF (Controller Application Cycle Factor) is set for I/O devices. In V4.2 or higher you can also select the position control cycle clock (Servo_fast) on I/O devices. If two servos are configured, only CACF = 1 is still allowed as a setting; i.e. send clock onboard PN interface = Servo_fast cycle clock, send clock CBE30-2 = servo clock. The servo must be scaled down to the Servo_fast at a factor of 2, 4, 8, 16, or 32 (as of SIMOTION V4.4).

The following settings are possible:

Product version	Property	Application cycle of the devices on		
		PROFINET IO Integrated (X150)	PROFINET IO (CBE30-2)	PROFIBUS DP
Before V4.2	1 position control cycle clock	--	--	Servo
Additionally for V4.2 or higher	2 servo clocks	Servo_fast	--	Servo
Additionally for V4.3 and higher	2 servo clocks	Servo_fast	Servo	Servo

Configuring the Servo_fast using the example of a D455-2 DP/PN

A precondition is that a D455-2 DP/PN must be configured with PROFINET IO system and PROFIBUS DP. In V4.3 or higher, you can also use a D455-2 DP/PN with two PN interfaces (onboard PN interface and CBE30-2).

1. In HW Config, select the SIMOTION device module, e.g. D455, and choose **Edit > Object properties** in the menu.
2. Activate the checkbox **Use Servo_fast/IPO_fast** on the **Isochronous Tasks** tab. Therefore the used PROFINET IO system is operated isochronously to the Servo_fast.

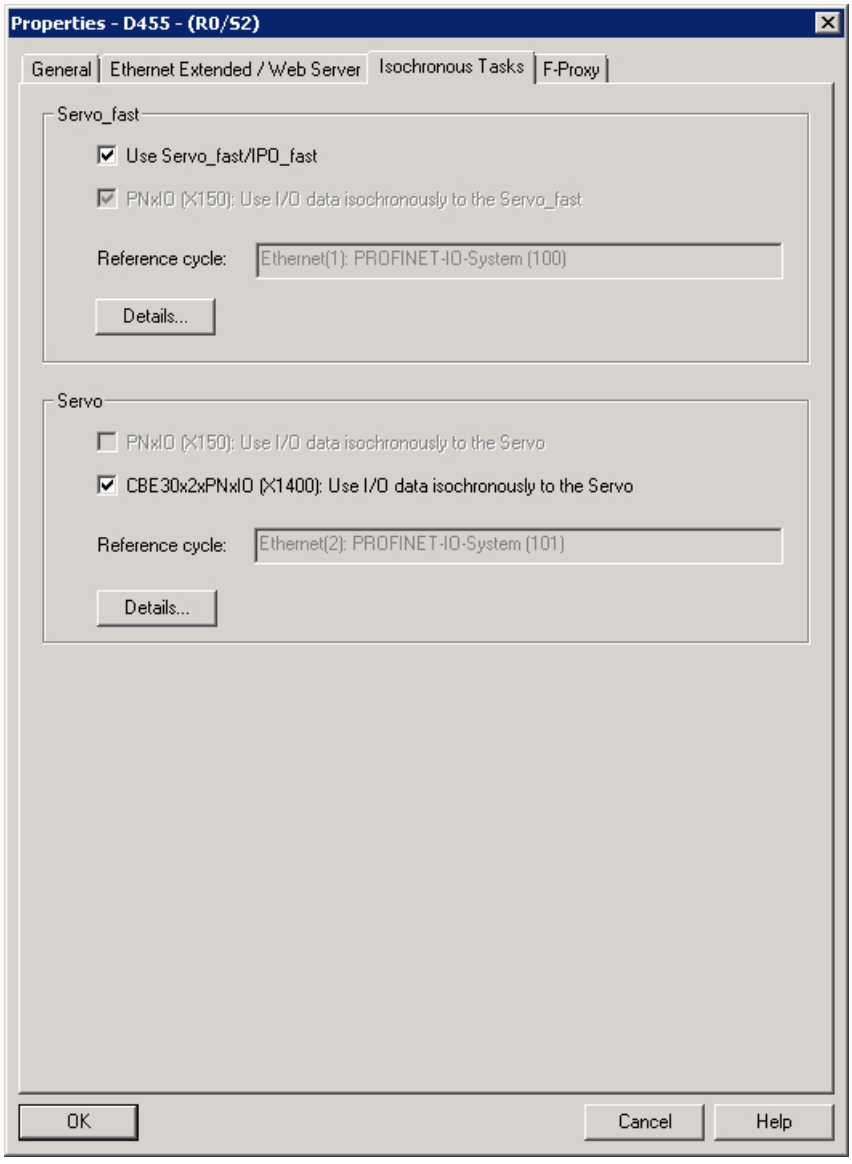


Figure 5-45 Configuring Servo_fast in HW Config

3. Click on the **Details...** button next to the **Servo** field.
4. On the **Isochronous operation** tab of the PROFIBUS DP, enter a factor for the DP cycle Tdp. The servo clock and DP cycle must always be configured the same when using PROFINET.

5. Confirm by selecting **OK** and save and compile the project in HW Config.
6. Switch to SIMOTION SCOUT and select **Set system cycle clocks** in the context menu of the SIMOTION CPU. This displays the values for the servo and for Servo_fast.

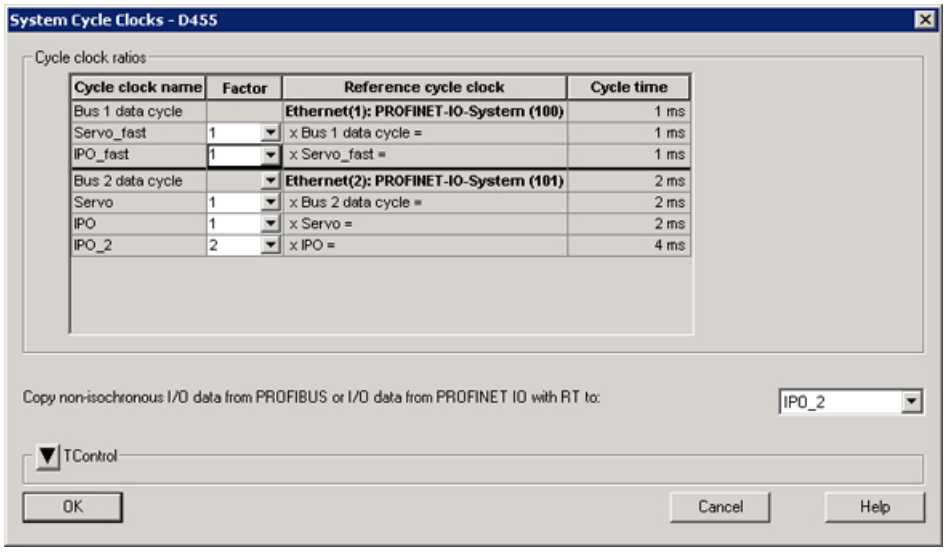


Figure 5-46 Servo and Servo_fast system cycle clocks

Note

For further information, refer to the *SIMOTION SCOUT Basic Functions* Function Manual.

5.3.11 Configuring a topology

5.3.11.1 Topology

Introduction

With IRT High Performance, the topology must be configured and settings made to determine which device is to be connected via which port to which other devices.

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

Using the topology editor (Page 141)

Using the object properties (Page 142)

5.3.11.2 Topology editor (graphical view)

Procedure

With the topology editor you have an overview of all ports in the project and can interconnect them centrally.

The topology editor is started with the **Edit > PROFINET IO > Topology** menu command in HW Config or NetPro (PROFINET device must be selected).

The topology editor offers the user two options for displaying the topology graphically (STEP7 V5.4 SP2 or higher) or in a tabular format. The graphic view is more suitable for situations involving interconnection.

Description

In the topology editor, you can:

- Interconnect ports
- Modify the properties of the interconnection
- Add passive components
- Arrange for an offline/online comparison to be displayed in online mode

Procedure

1. In SCOUT, double-click on the SIMOTION module in order to access HW Config.
2. Select the PROFINET module, e.g. PNxIO in the case of a D445-2 DP/PN.
3. Perform **Edit > PROFINET IO > Topology**. The topology editor opens.

4. Click **Graphical view** to bring the tab into the foreground.

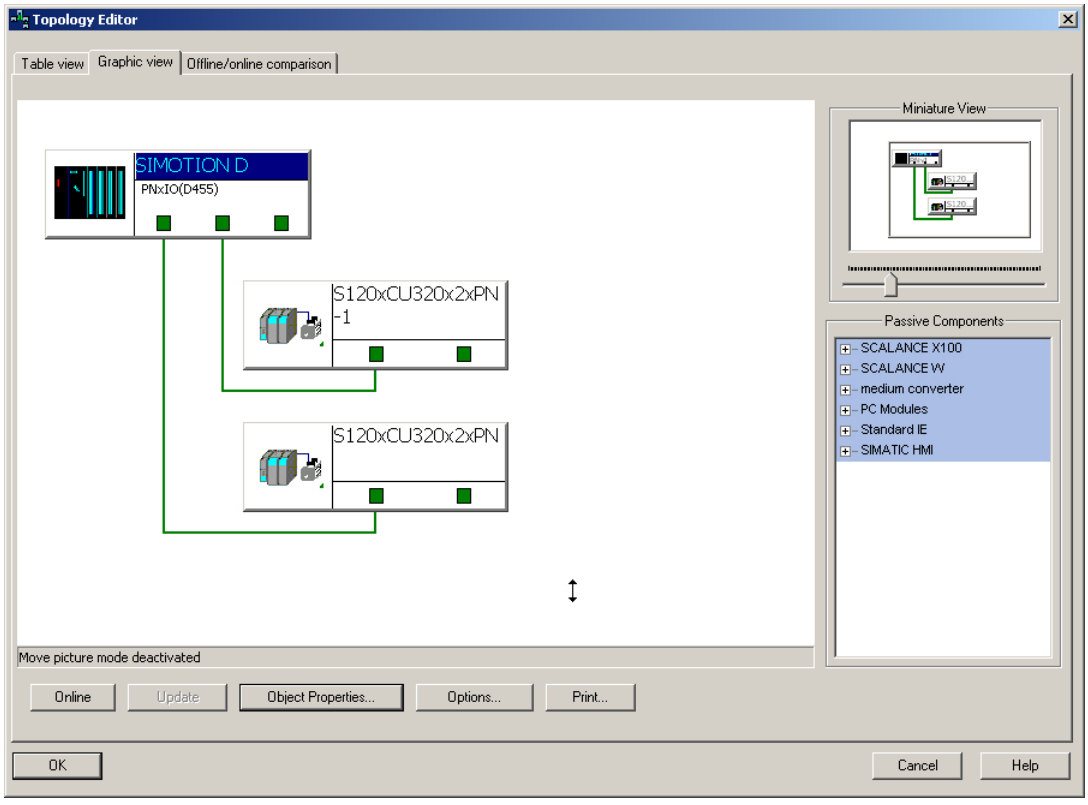
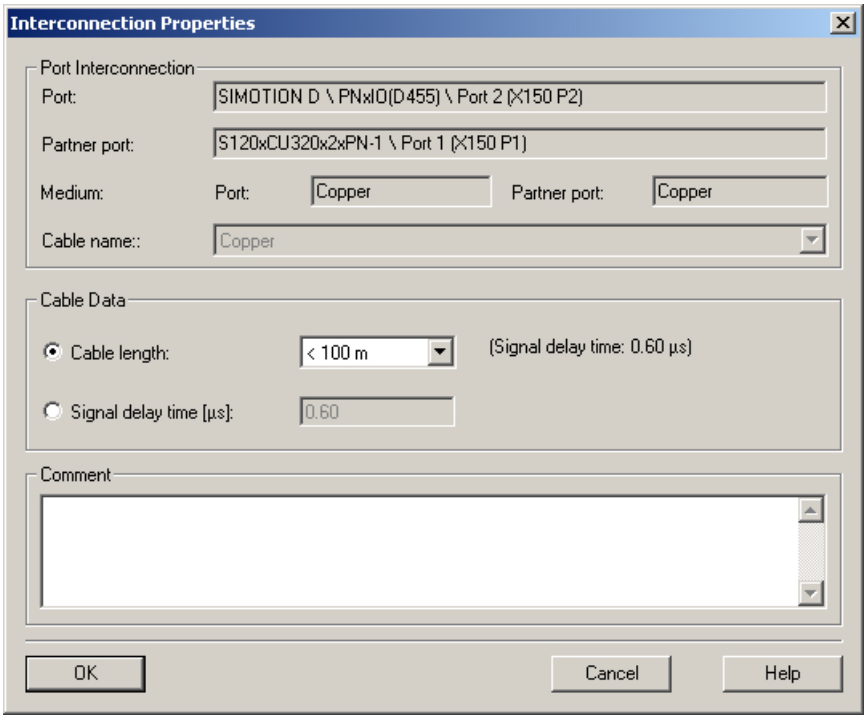


Figure 5-47 Topology editor (graphical view)

5. Establish connections between ports by pressing and holding down the left mouse button and drawing a line between the two ports to be connected. The **Interconnection Properties** window opens.

6. The port interconnection is displayed: You can configure the cable data: A cable length of <100 m is set as standard. This should not be modified under normal circumstances. Alternatively, you have the option of specifying a signal propagation delay; e.g. the latency time when using slip rings.



7. Click **OK** to confirm.

Offline/online comparison

When you switch to online mode, the topology in the editor is compared with the actual topology. Any components which are not recognized are highlighted by a question mark, while connections and components in RUN are highlighted in green.

5.3.11.3 Interconnecting ports via the topology editor (table view)

Procedure

Ports can be interconnected in the **Tabular view** of the topology editor.

The topology editor is started with the **Edit > PROFINET IO > Topology** menu command in HW Config or NetPro (PROFINET device must be selected).

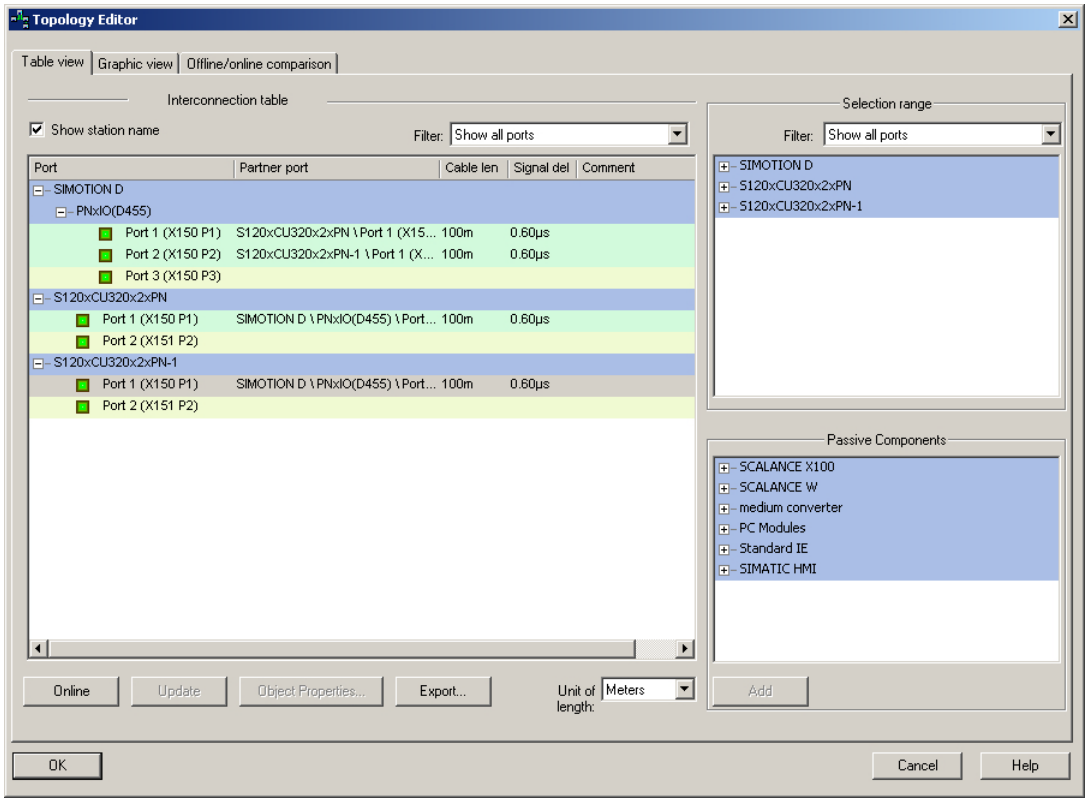


Figure 5-48 Topology editor

All configured PROFINET IO devices with their ports are listed in the interconnection table on the left-hand side. You can use the Filter dialog to select whether all ports, only the ports that have not yet been interconnected or only the ports that have already been interconnected are to be displayed.

Interconnecting ports in the tabular overview

1. To interconnect ports of different devices, select the port of a device that you want to interconnect in the right-hand field.
2. Drag this port to the desired port of a device in the interconnection table. The "Interconnection Properties" dialog opens.
3. Configure the cable data. A cable length of < 100 m should be set by default.
4. Confirm your entries with **OK**.

5.3.11.4 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.

Procedure

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.

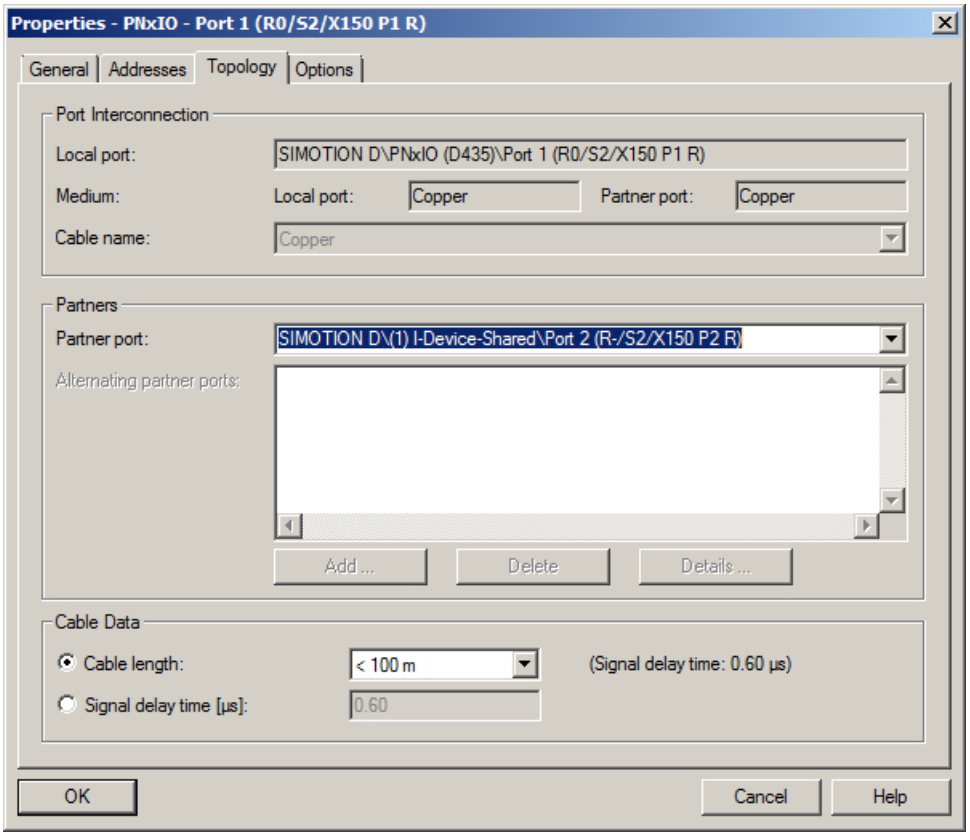


Figure 5-49 Object properties Topology

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 currently selected port.
4. Confirm your entries with **OK**.

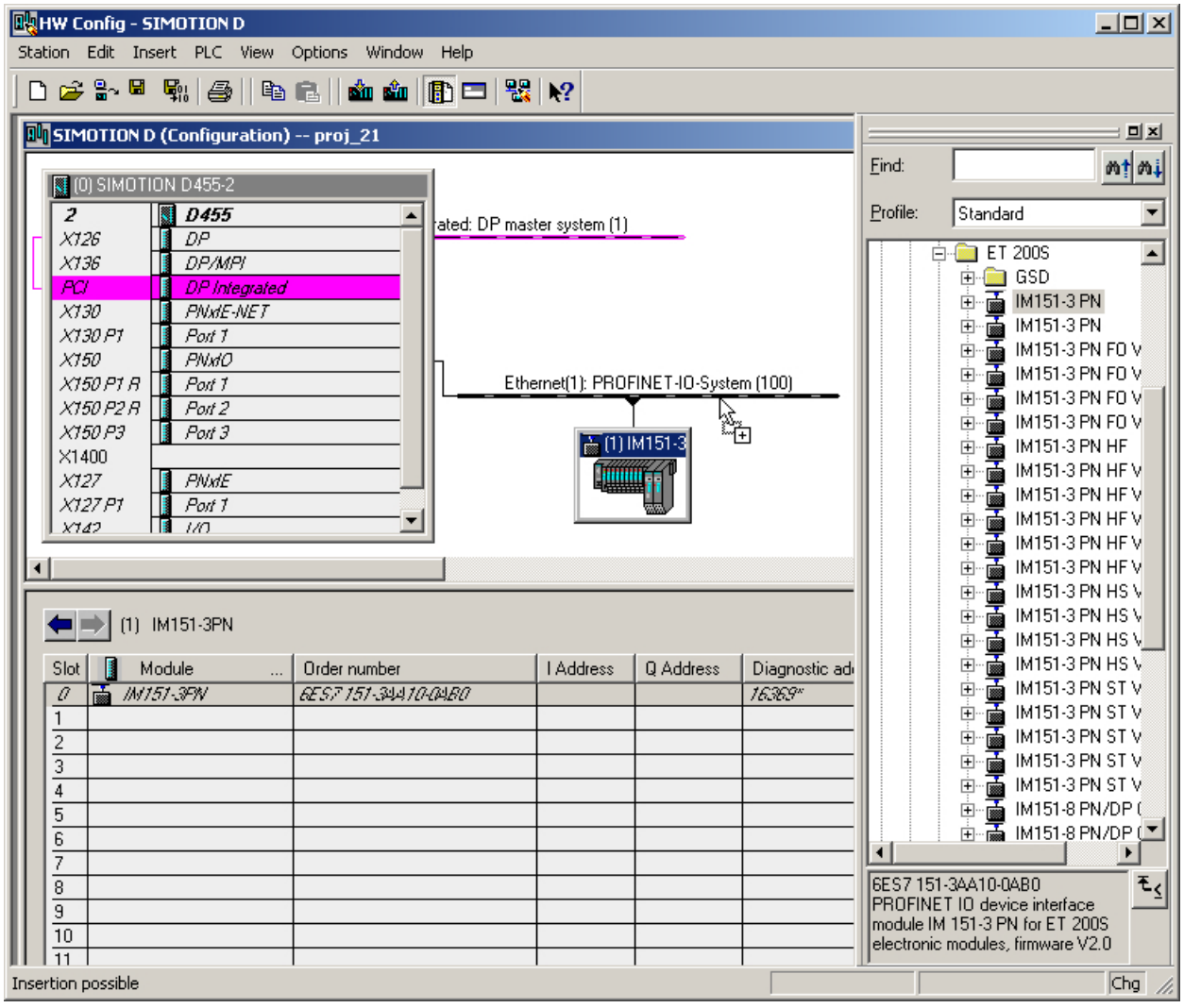
5.3.12 Creating an IO device

Requirement

You have already created a PROFINET IO system and configured a PROFINET IO module, e.g. SIMOTION D455-2 PN (see Inserting and configuring SIMOTION D4x5-2 DP/PN/D410 PN/D410-2 DP/PN (Page 115)).

Procedure for PROFINET IO devices using the hardware catalog

1. Double-click the corresponding module in SIMOTION SCOUT to open HW Config.
2. Under PROFINET IO in the hardware catalog, select the module you wish to connect to the PROFINET IO system.
3. Drag the module to the path of the PROFINET IO system. The IO device is inserted.



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.

5.3.13 Inserting and configuring the SINAMICS S120

Requirement

You have inserted a SIMOTION D4x5-2 with integrated PROFINET interface in your project, a PROFINET IO subnet has already been created and the Sync-Master is configured.

Note

The configuration is almost identical for all SINAMICS S120 and SIMOTION D devices.

Procedure

1. Select **PROFINET IO > Drives > SINAMICS** in the HW Config hardware catalog. Then select the module, e.g. **SINAMICS S120 CU320-2 PN**.
2. Click the entry and drag the drive, e.g. **V4.4** , to the PROFINET IO subnet. The **Properties - Ethernet Interface SINAMICS-S120xCU320x2xPN** window opens.
   A suggested IP address will already be displayed here and the subnet will be selected.
3. Click **OK** to accept the setting.
4. In HW Config, select **Station > Save and Compile Changes**.

Setting a telegram in SIMOTION SCOUT

For V4.2 and higher, the telegrams are assigned automatically in SIMOTION SCOUT by means of symbolic assignment. For isochronous communication, a telegram must be configured (e.g. telegram 105) which enables the drive to be synchronized with PROFINET. The telegram is thus automatically set in SCOUT during the configuration of the drive unit and following the assignment of an axis.

1. If you have not yet configured a supply and drive, click **Configure drive unit** in the SCOUT project navigator and drive unit.
2. Run the drive unit configuration wizard.
3. After you have closed the wizard, click **Communication > Telegram configuration** under the drive unit in the project navigator. Tab **IF1: PROFIdrive PZD telegrams** contains a list of telegrams. A telegram is not yet configured following commissioning of the drive since the drive has not yet been symbolically assigned.
4. Therefore, insert a new axis TO and run through the axis wizard. In the wizard, interconnect the axis to the corresponding drive object of the S120 and a corresponding telegram will automatically be created (symbolic assignment).

5. Select **Project > Save and compile all** from the menu. This means that the addresses will be set up automatically with symbolic assignment. You can also trigger symbolic assignment of addresses using the menu item **Project > Set up addresses**.
6. Once the axis configuration process is complete, click **Communication > Telegram configuration** under the drive unit in the project navigator. Tab **IF1: PROFIdrive PZD telegrams** now lists the telegram for the drive (e.g. SIEMENS telegram 105).

Telegram in HW Config

Alternatively, you can assign the telegram in HW Config. This is only possible if the drive unit and the drive have already been configured in SCOUT. You will then be able to proceed as follows in HW Config.

Note

Symbolic assignment in SIMOTION SCOUT is recommended for telegram interconnection. For this purpose, make sure that a check mark is applied for **Project > Use symbolic assignment** in the menu.

If you are using the symbolic assignment, SIMOTION SCOUT is entirely responsible for managing the telegram between SIMOTION and the drive DOs. In other words, SIMOTION SCOUT independently creates telegrams that all contain the necessary signals according to the technological configuration. SIMOTION SCOUT automatically manages the positioning of signals in the telegram and its size; the user is no longer able to influence this process or make any changes.

1. Select the inserted SINAMICS drive and double-click the entry **SIEMENS / Standard telegram xx** in the lower table for the drive.
   The **Properties SIEMENS / Standard telegram xx** dialog box is called.
2. Select the corresponding telegram. After it is saved, the telegram can be also be selected in SCOUT, under **<"Drive_device_xx"> - Communication > Telegram configuration** in the project navigator. Alignment with HW Config is possible.

Settings for isochronous mode (V4.4 or higher)

For isochronous mode, the drive unit must be synchronized with the PROFINET cycle clock. You can make these settings at the PN interface of the drive unit and on the SIMOTION device (Sync-Master).

Settings on SIMOTION device (Sync-Master)

1. In HW Config, select the SIMOTION device module, e.g. D455, and choose **Edit > Object properties** in the menu.
2. Click the **Details...** button on the **Isochronous tasks** tab.
   This button only appears if you select the **...Assign I/O device isochronous** check box.

3. Select the **Automatic** setting in the **Details for servo** dialog box under **Ti/To mode**. The cycle clocks and time constants for all IO devices (Sync-Slaves) on the PROFINET are then calculated automatically by the system.

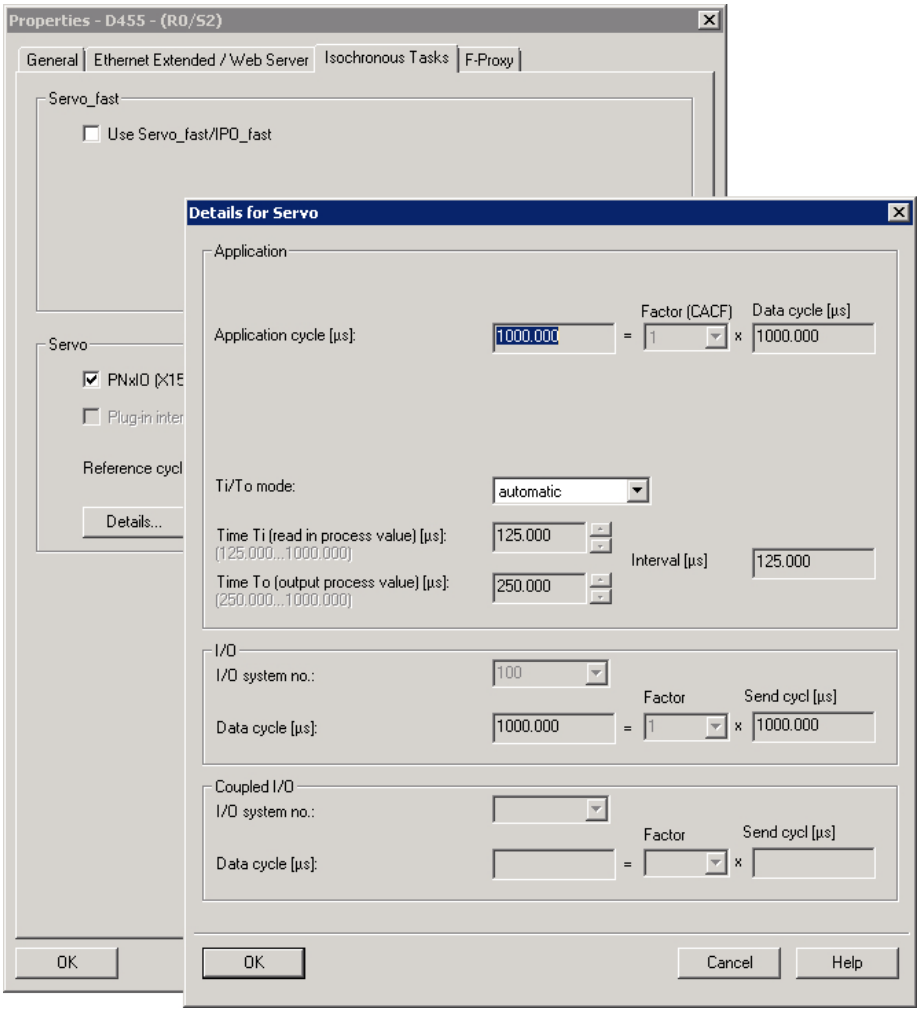


Figure 5-50 Configuring an isochronous task on the SIMOTION device.

4. Click **OK** to close both dialog boxes.

Settings on the SINAMICS drive (Sync-Slave)

1. Select the SINAMICS drive on the PROFINET IO system and double-click on the entry of the PROFINET interface in the lower table, e.g. PN-IO.
   The **Properties PN--IO** dialog box is displayed.
2. Select the **Servo** entry on the **IO cycle** tab under **Assign IO device isochronously**. The drive is operated isochronously. The time constants are calculated automatically.

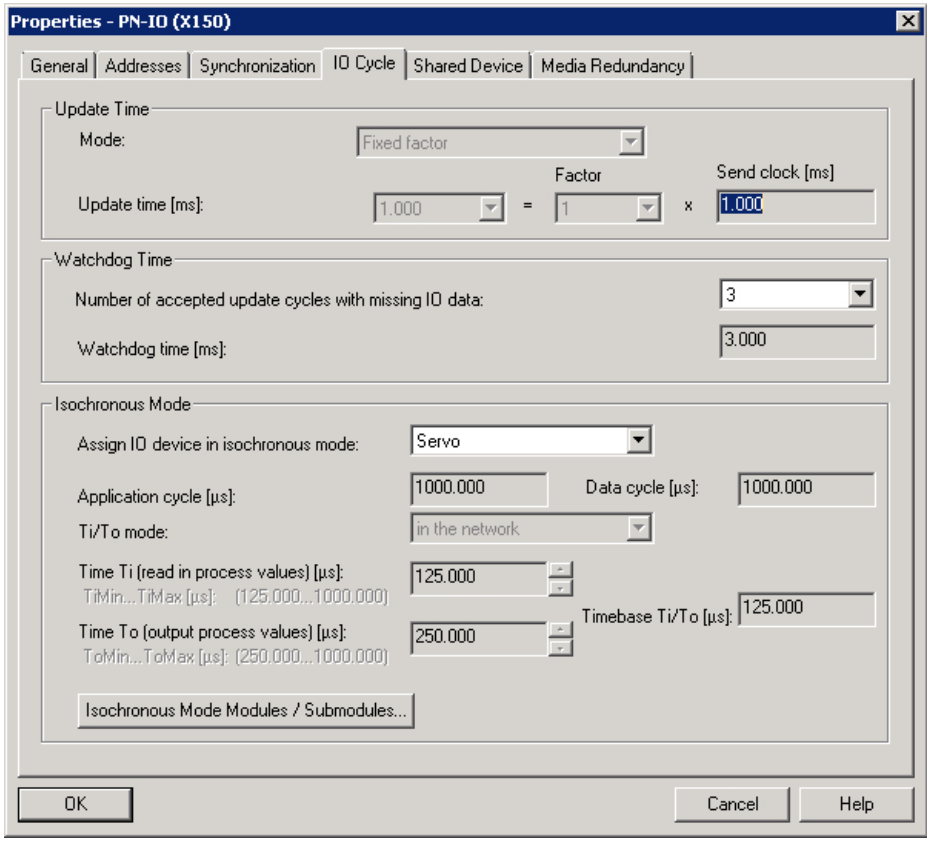


Figure 5-51 Configuring an isochronous task on the PROFINET IO device

3. You can see the current cycle clock in **Application cycle**. To configure cycle clock scaling, you need to configure down-scaling by selecting **Set system cycle clocks** in the **execution system**. This is required if the servo cycle clock should be slower than the PROFINET IRT send clock, for example. However, the servo cycle clock must already have been taken into account in the PROFINET IRT send clock.
4. 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**.
5. Confirm the entries with **OK**.

Note

You must carry out further steps in order to include the drive in the sync domain (see Creating a sync domain (Page 128)), load the IP address to the drive (see Assigning device names and IP addresses to IO devices (Page 151)), and interconnect the ports (see Interconnecting ports via the topology editor (table view) (Page 141)).

5.3.14 IP address and device name

Introduction

Devices and controllers have a fixed MAC address, a configurable IP address, and a device name. The IP address, subnet mask, and device name are defined within the Ethernet interface properties in the engineering software. This ensures the devices within the project have their own unique assignment. The device name is referred to as **NameOfStation** in accordance with the standards or, in the case of devices with multiple interfaces, **NameOfInterface**. The device name must be unique for the communication.

You can find the device name in the **Properties - Interface** dialog by double-clicking on the device or the controller. Click on **Properties** in the dialog to define the IP address. See also Add and configure PROFINET interface CBE30-2 (Page 120).

Device name guidelines

In the case of an IO device, the device name is, for example, the drive name *Drive1*. In the case of controllers, the device name is the name of the PROFINET interface, e.g. *PNxIO* for an integrated interface of SIMOTION D. A controller can also have multiple interfaces and, therefore, multiple device names.

Device names are subject to certain syntax rules, i.e. they must always be DNS-compliant. For SIMOTION and SIMATIC, there are additional boundary conditions for the device names.

- Letters a-z and numbers 0-9 may be used.
- No German umlauts (ä, ö, ü)
- The special characters minus (-) and period (.) may be used
- No other special characters or spaces ! " § $ % & / ( ) = ? * ' _ : ; > < , # + | ~ \ }] [{
- The period is used for structuring, i.e. the device name can comprise several parts (labels)
  - Label.Label.Label.Label
  - Periods are used as separators.
  - A label must not begin with a hyphen (-) or a period (.)
  - A label must not exceed 63 characters
- The maximum total length of the name is 240 characters (including hyphen and period)
- The device name must start with a letter, but not with the character string "port-xyz-" (x, y, z = 0..9).
- **No** distinction is made between upper-case and lower-case letters. The configured device name can also contain upper-case letters but these will be saved on the device as lower-case letters.
- Up to SIMOTION SCOUT V4.3, the device name must not include hyphens (-), i.e. the name must comply with the ST naming convention. As of SIMOTION SCOUT V4.3, this restriction no longer applies.

Initialization of the controller and devices in online mode

As the controllers and devices do not have a device name or IP address when delivered, these will need to be assigned at the outset. When assigning addresses (i.e. the IP address and device name - a process also referred to as initialization), a distinction is made between controllers and devices. You can adopt different approaches when initializing devices and controllers.

Controller initialization

- Download the application
- Engineering software
  - HW Config, NetPro, SCOUT
  - Primary Setup Tool (PST)
  - PRONETA
- Via the application (the _setPnNameOfStation system function for SIMOTION (as of V4.4)

Note

When using engineering software for initialization, and particularly where larger systems are involved, you should establish direct connections with the device to ensure the device to be initialized is clearly identifiable. Alternatively, the Flash feature can be used, whereby devices can be identified by a flashing LED.

Device initialization

- Engineering software
  - HW Config, NetPro
  - SCOUT, STARTER
  - Primary Setup Tool (PST)
  - PRONETA
- Write to the MMC or CF card beforehand and then plug in.

Topology-based initialization for devices

Devices can also be initialized without an MMC or CF card. This method is known as topology-based initialization

Note

For topology-based initialization, the PN interface of the device must be in its factory setting.

See also

Creating an I device (Page 183)

5.3.15 Assigning device names and IP addresses to IO devices

Introduction

You can only go online with the engineering software of a device (or controller) if it has been assigned an IP address in the engineering software. Devices must also be assigned a device name which is unique across the network. This enables the controller to identify the devices assigned to it.

You can specify the IP address in the **Properties - Ethernet interface….**dialog (double-click on the device to open this). Moreover, a default name is entered that you can modify. The default setting **Assign IP address through Controller** is active. In other words, when powering up, the controller identifies the devices assigned to it via the device names and then assigns the IP address defined in Engineering to them. We recommend leaving this function activated.

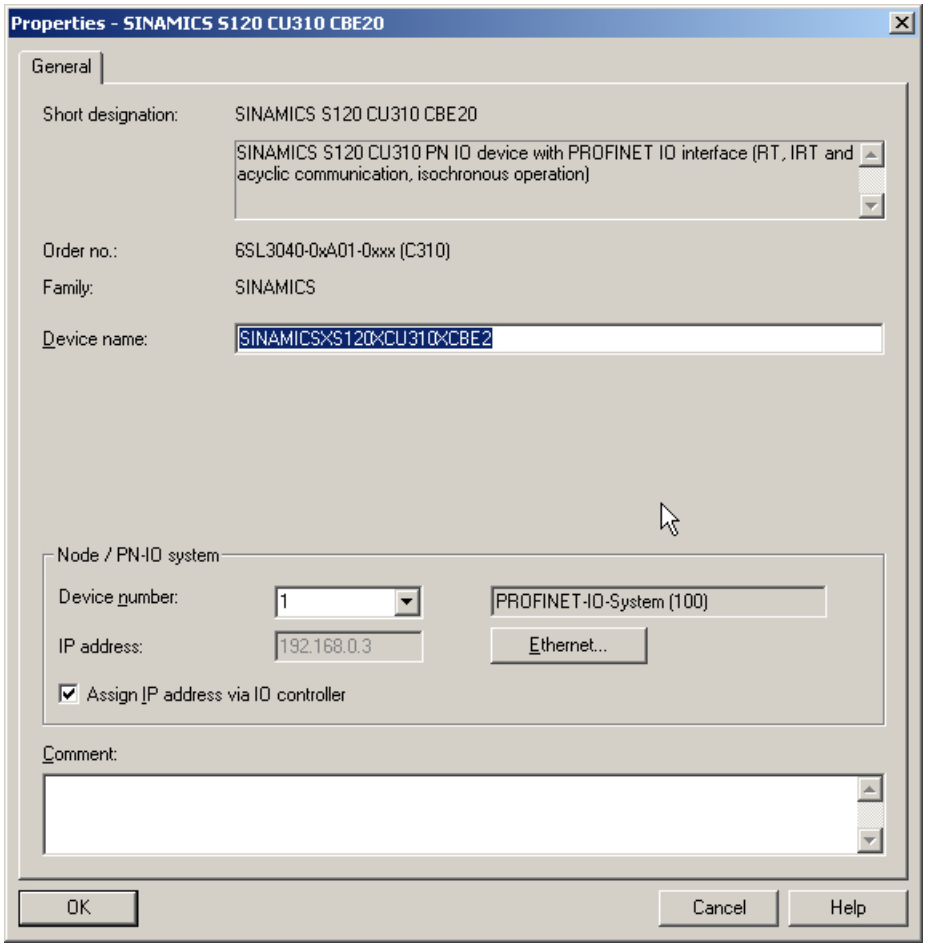


Figure 5-52 Properties SINAMICS S120

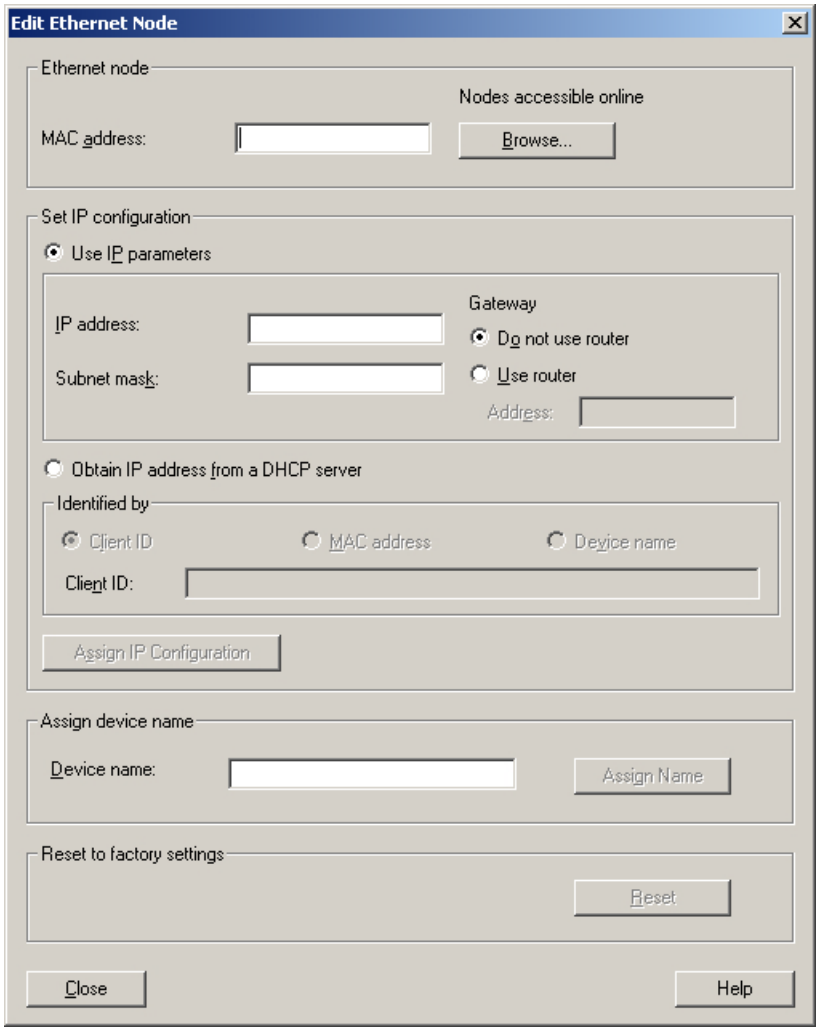
IO device initialization

Note

If you are connecting the programming device/PC directly to the device's PROFINET interface, you can use a patch or crossover cable. During commissioning, we recommend that the device to be initialized is connected directly to the programming device/PC.

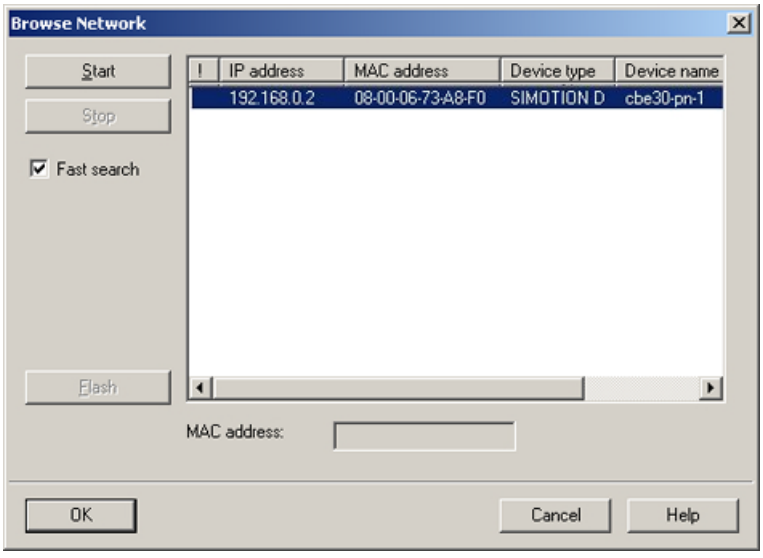
Alternatively, the Flash feature can be used, whereby the IO device can be identified by a flashing LED.

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



2. Click the **Browse** button.

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



4. Click the device to be initialized and confirm with **OK**.
5. Enter the IP address and subnet mask you specified in the **Properties – Ethernet interface … ** dialog.
6. The default setting (**Do not use router**) for the gateway remains unchanged.
7. Click on the **Assign IP configuration button**. The IP address is then assigned to the device online.
8. Enter the device name that you have defined in HW Config, see figure **Properties SINAMICS S120**.
9. Click on 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.

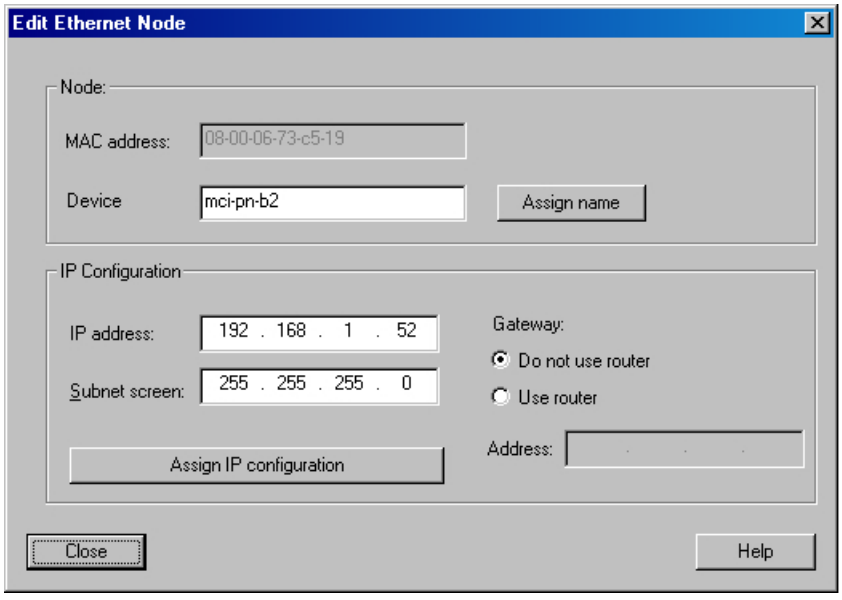


Figure 5-53 Edit Ethernet nodes

- 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.

5.3.16 Assigning device name and IP address via user program/DCP

5.3.16.1 IP address and device name via UP/DCP (Mini-IP-Config)

Description

It was previously only for I devices that SIMOTION supported the allocation of the IP address and device name (NameOfStation) from the user program or via the Discovery and Configuration Protocol (DCP). With SIMOTION V4.2 and Step 7 V5.5 or higher, this must be configured in HW Config and applies to all IO devices and IO controllers. To do this, you need to select the checkboxes for free allocation in the Properties of the Ethernet interface and the PROFINET interface. Previously in the case of SIMOTION, the IP configuration could only be set for the PN-IE interfaces via the user program; not for the PN-IO interfaces. As of V4.4, the IP address can be set from the user program for all PN interfaces.

A PN-IE/PN-IO interface is addressed in the system function via its diagnostics address (for the interface of the PN interface). You can also specify whether the IP configuration (device

name, IP address) is to be set temporarily (until the power is next switched OFF/ON) or permanently (retained after the power is switched OFF/ON). If it is a temporary setting, the device name is deleted when the power is next switched OFF/ON and will contain a blank string.

With the new system functions, the IP addresses and the device names of the PN IE interfaces and the PN IO interfaces can be issued from the user program. An additional reboot is no longer necessary for the PN interfaces, as the set name does not also have to be activated via _activateNameOfStation.

This mechanism allows the IP addresses and device names to be adjusted without making changes to the project. The IP settings can be changed locally, especially for series machines.

New system functions as of V4.4

New system functions have been introduced as of SIMOTION V4.4 for assigning the IP address and device name. They expand the range of functions already available. The functionality of the existing system functions remains unchanged and they can still be used as before. You are advised to use the new system functions.

System functions up to V4.4	New system functions as of V4.4
_setNameOfStation/_activateNameOfStation	_setPnNameOfStation
_getActiveNameOfStation	_getPnNameOfStation
_getPnInterfacePortNeighbour	_getPnPortNeighbour (only PN IO interfaces)
_setIpConfig	_setPnIpConfig
_getIpConfig	_getPnIpConfig

Note

A detailed description of the system functions can be found in the online help and in the *SIMOTION System Functions/Variables Devices List Manual*.

Comparison of system functions for obtaining the IP address and device name via a different method

With the SIMOTION system functions, the following operations are possible:

	Assignment of device name via UP		Assignment of IP address via UP	
	(_setNameOfStation etc.)	(_setPnNameOfStation etc.) As of V4.4	(_setIPConfig/ _getIPConfig)	(_setPnIPConfig/ _getPnIPConfig) As of V4.4
PN IE interfaces	No	Yes	Yes	Yes
PN IO interfaces	Yes	Yes	No	Yes

Diagnostics when an error occurs

For the purpose of error diagnostics, a diagnostics buffer entry is created on the SIMOTION I device CPU if a DCP request to set the device name or IP address for the I device could not be carried out because the relevant checkboxes are not activated in HW Config.

5.3.16.2 Configuring device names via system function

Introduction

The device name (NameOfStation or NameOfInterface) can be assigned via user program. This option must be explicitly enabled when configuring the hardware for this version.

Obtaining the device name using a different method

1. In HW Config, open the Properties dialog box for the PROFINET interface.
2. Activate the **Obtain device name using a different method** check box on the General tab and confirm the selection by clicking **OK**.

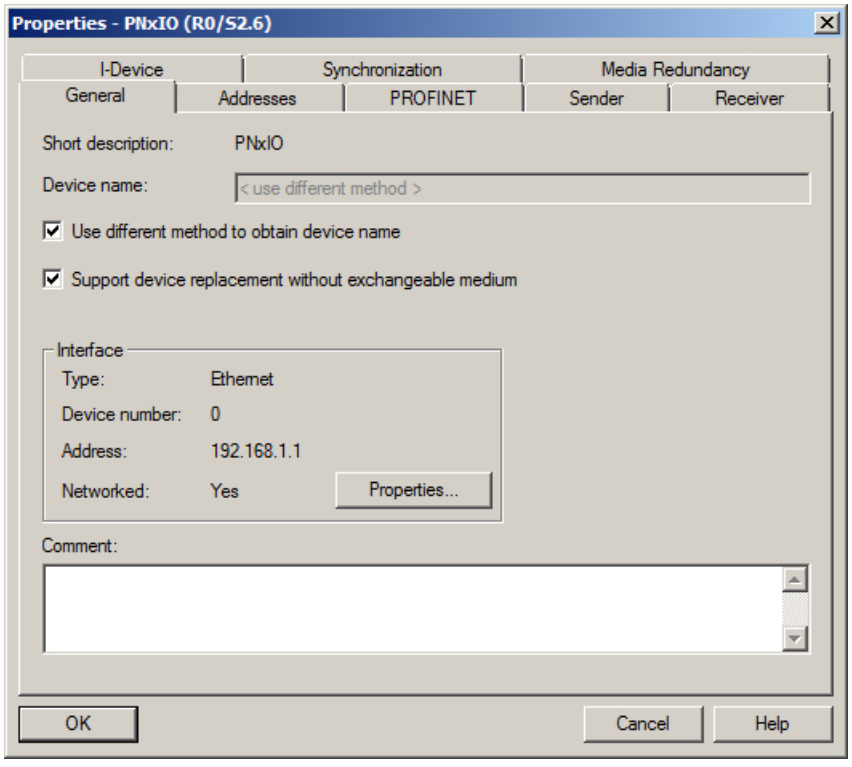


Figure 5-54 Obtaining the device name using a different method

Overview of the possible configuration methods

Configuration method		Response
Device name from project The option "Obtain device name using a different method" is **not** set.	System function _setPnNameOfStation / primary setup tool via DCP	**The device name cannot be set.** The configured device name from the project is active. Following a restart the device name from the project is still active.
Device name not from project The option "Obtain device name using a different method" is set.	System function **_setPnNameOfStation** with parameter storeNameOfStationPermanent := YES	After fault-free performance of the system function, the set device name is active. Following a restart the device name set by the system function is still active. **The device name from the project is no longer active.**
	System function **_setPnNameOfStation** with parameter storeNameOfStationPermanent := NO	After fault-free performance of the system function, the set device name is active. Following a restart, the device name set by the system function is no longer active. The device name contains a blank string. **The device name from the project is no longer active.**
	STEP7 / primary setup tool via DCP	The device name set via STEP7 / DCP is active. Following a restart the device name set by STEP7 or the primary setup tool via DCP is still active. **The device name from the project is no longer active.**
	Through PROFINET controller via DCP (topology-based initialization)	Following a restart the device name set by the PROFINET controller is still active. Topology-based initialization only occurs when there is no existing device name (interface in factory setting). **The device name from the project is no longer active.**
	Through PRONETA via DCP with the "temporary" option (PRONETA supports the "temporary" option for tests)	Following a restart the device name set by PRONETA is no longer active. The device name contains a blank string. **The device name from the project is no longer active.**

General information on device names

- If the setting "Use different method to obtain device name" is activated in the project, the last device name provided on the interface becomes active once the project is downloaded.
  This means:
  - The device name from the last project is active if a device name was configured in the previous project.
  - A blank string is set if the device name was in its factory setting (delivery state).
  - A blank string "" can be set via user program and DCP
  - A new device name can also be assigned via DCP and user program if a PROFINET connection (controller or iDevice) is active. Changing the name terminates the connection.
- In certain situations, the module is reset when downloading from SIMOTION SCOUT ("Target system reset" displayed in SCOUT log). This reset has the same effect on the NameOfStation as switching the power OFF/ON.

You will find an overview of the possible functions in the chapter IP address and device name via UP (Page 154).

Note

If the option "Use different method to obtain device name" has been activated, the option "Use different method to obtain IP address" must also be activated.

5.3.16.3 Configuring an IP address via system function

Introduction

The IP address can be assigned via user program. This option must be explicitly enabled when configuring the hardware for this version.

Obtaining the IP address using a different method

1. In HW Config, open the Properties dialog box for the PROFINET interface and click on the **Properties** button on the General tab.
2. Switch to the **Parameter** tab in the Properties dialog box that opens for the Ethernet interface.
3. Activate the **Use different method to obtain IP address** check box and confirm the selection by clicking **OK**.

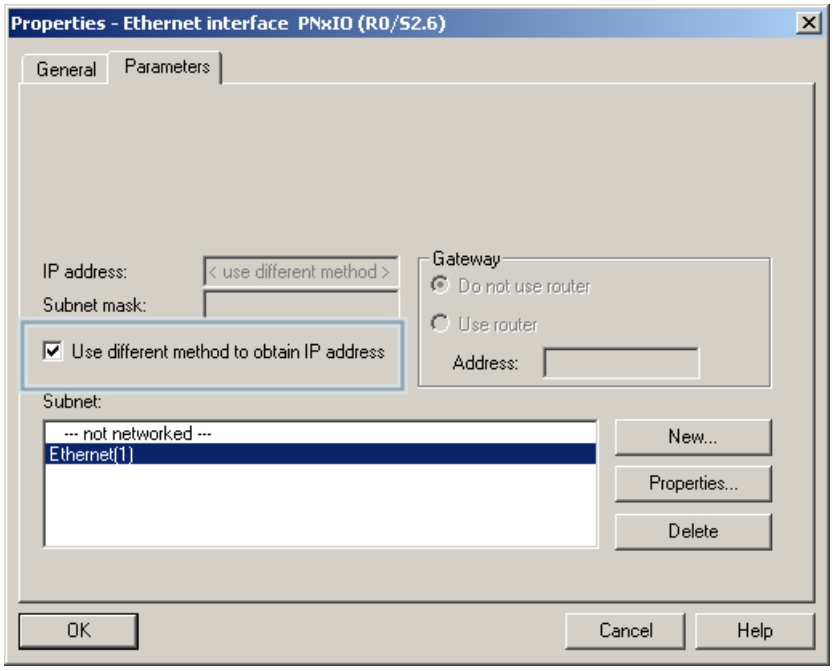


Figure 5-55 Obtaining the IP address using a different method

Overview of the possible configuration methods

Configuration method		Response
Project not present in the device	DCP (DCP BlockQualifier = permanent)	The IP address set via DCP is active. After a restart the IP address set via DCP is still active.
	DCP (DCP BlockQualifier = temporary)	The IP address set via DCP is active. After a restart the IP address 0.0.0.0 is active.

Configuration method		Response
Project present in the device or not	Download project; in the project the "Obtain IP address using a different method" **option is not** set	The IP address loaded via the project is active. After a restart the IP address set via the project is still active.
	Download project; in the project the "Obtain IP address using a different method" option is set	The IP address last active on the interface is still valid. • The IP address from the last project, if a project with a valid IP address was previously loaded. • If a project with a valid IP address has not previously been loaded, then the IP address previously set via DCP / _setPnIPConfig or the default IP address as delivered.
Project present in the device The option "Obtain IP address using a different method" is **not** set	Opening of **_setPnIPConfig** to set an emergency IP address	The IP address loaded via the project is not temporarily overwritten by a so-called emergency IP address. Following a restart the IP address from the project is active again.
	DCP (DCP BlockQualifier = temporary or permanent) for setting an emergency IP address	
Project present in the device The option "Obtain IP address using a different method" is set	System function **_setPnIpConfig** with storeIp ConfigPermanent = YES	The IP address set via system function is active. After a restart the IP address set via the system function is still active.
	System function **_setPnIpConfig** with storeIp ConfigPermanent = NO	The IP address set via system function is active. After a restart the IP address 0.0.0.0 is active.
	STEP7 or primary setup tool via DCP (DCP BlockQualifier = permanent)	The IP address set via STEP7 / primary setup tool is active. After a restart the set IP address is still active.
	PROFINET controller via DCP (DCP BlockQualifier = temporary, only possible for PROFINET IO interfaces)	The IP address set via DCP is active. After a restart the IP address 0.0.0.0 is active.

General information on modifying the IP address

- If a PROFINET connection is active, DCP cannot be used to assign a new IP address. For safety reasons the setting of an IP address via DCP with a project present **in RUN and STOP operating state** is only permitted if **Obtain IP address using a different method** is set. If **Obtain IP address using a different method** is not set, the setting of an IP address is only permitted **in the STOP operating state**. This is intended to prevent an IP configuration set via a project, e.g. via the function "Edit Ethernet node," being changed easily from outside, by means of which a PROFINET communication of a SIMOTION device can be disrupted.
- An IP address can also be assigned via system function if a PROFINET connection (IO controller or I-device) is active. Changing the IP address via the system function leads to the breaking of the PROFINET connection.
- The IP address 0.0.0.0 can be set via the user program and DCP.
- In certain situations, the module is restarted when downloading with SIMOTION SCOUT ("Target system reset" displayed in SCOUT log). Such a restart has an impact on when IP addresses come into force.
- Emergency IP address:
  The emergency IP allows the user to temporarily overwrite the IP configured in the project via DCP or system function, e.g. in order to load a project with a corrected IP address if access is no longer possible via the IP address configured in the project. When downloading the new project, make sure that the emergency IP address assigned via "Edit Ethernet node" agrees with the IP address used in the project, because in certain cases SIMOTION performs restarts during downloading, and the IP address from the project subsequently becomes effective.

You will find an overview of the system functions in the chapter IP address and device name via UP (Page 154).

5.3.16.4 ResetToFactorySettings via DCP

Description

A resetting of the device names or the IP address to the factory settings (ResetToFactorySettings) via DCP to one of the PN IO or PN IE interfaces sets the parameters as follows:

- IP address: 0.0.0.0
- NameOfStation: Blank string

Boundary condition

Due to security restrictions, DCP ResetToFactorySettings is only permitted when an iDevice is configured with **Parameter assignment for the PN interface and its ports on the higher-level IO controller** and, at the same time, **Use different method to obtain device name** and **Use different method to obtain IP address** are both set. The interface must not be used as an IO controller, i.e. no IO device is configured.

For an interface with a default IP address, the response after a ResetToFactory is as follows:

- When a ResetToFactorySettings is performed, the IP address is immediately set to 0.0.0.0.
- If the power is then switched OFF/ON, the IP address remains 0.0.0.0.
- The default IP address is not reactivated until the NVRAM is erased.

5.3.17 Configuring media redundancy (V4.3 and higher)

5.3.17.1 Creating ring topology

Setting up ring topology

In order to use MRP, you must establish a ring topology. The ring nodes must support media/ system redundancy.

Note

Devices that support MRP

An overview of all devices that support media and/or system redundancy can be found in the Support area (http://support.automation.siemens.com/WW/view/en/67364686).

To establish the ring, bring together the ends of the line topology into one device.

Requirement

You have created a project in SCOUT and added a SIMOTION module.

Setting up devices in HW Config

1. Open HW Config.
2. Set up a PROFINET IO system in the SIMOTION module.

3. Add a module, such as a SCALANCE switch, that can take the role of Redundancy Manager.
4. Then add the other required SIMOTION modules (or other modules). The modules assume the role of redundancy clients.

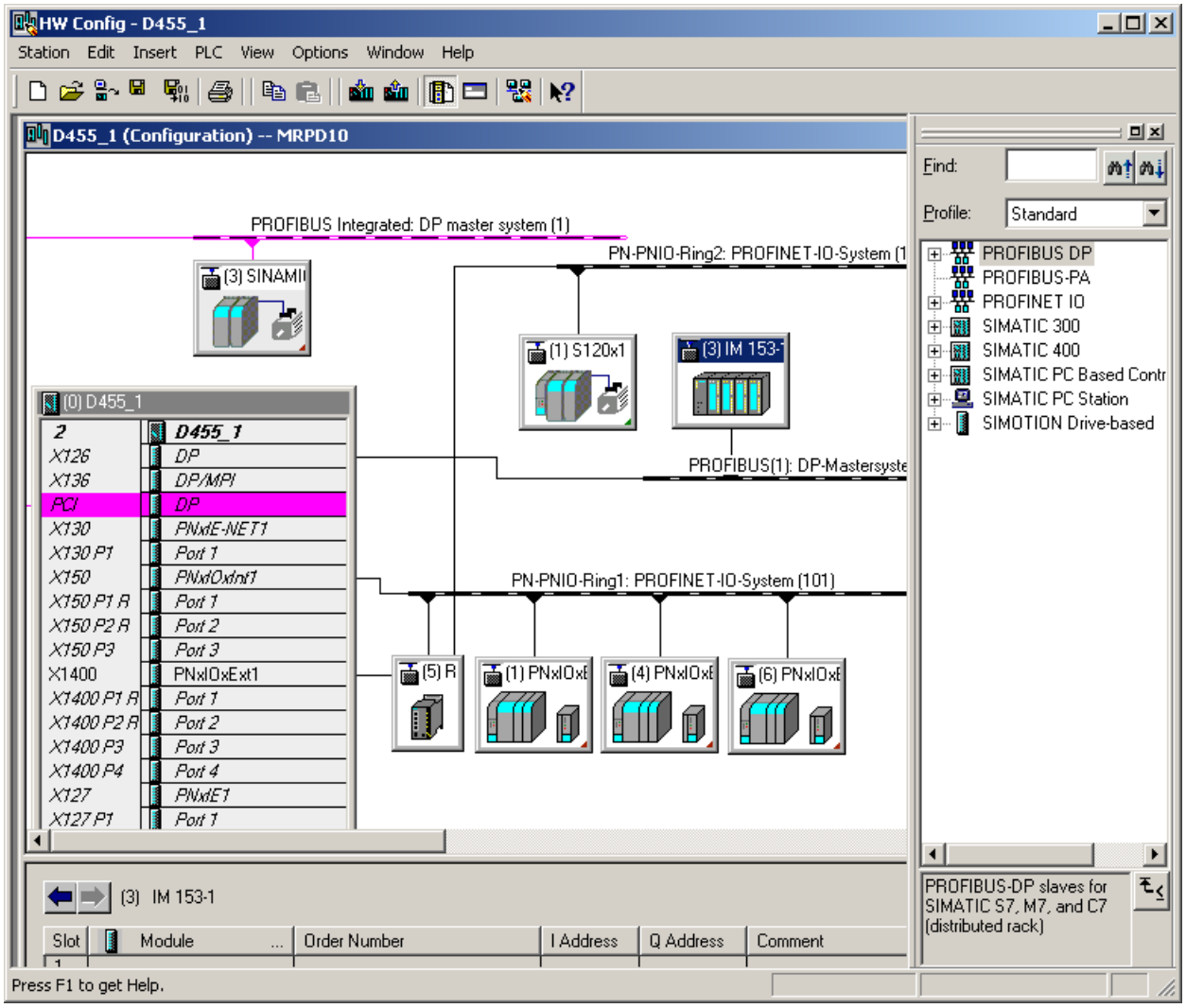


Figure 5-56 Ring project in HW Config

Establishing ring topology in the Topology Editor

1. Open the Topology Editor.
2. Interconnect the individual ports of the devices.

3. If necessary, change to graphic view to drag the connections with the mouse.
4. Connect the two end points of the line topology with each other in order to close the ring, e.g. the redundancy manager with a redundancy client.

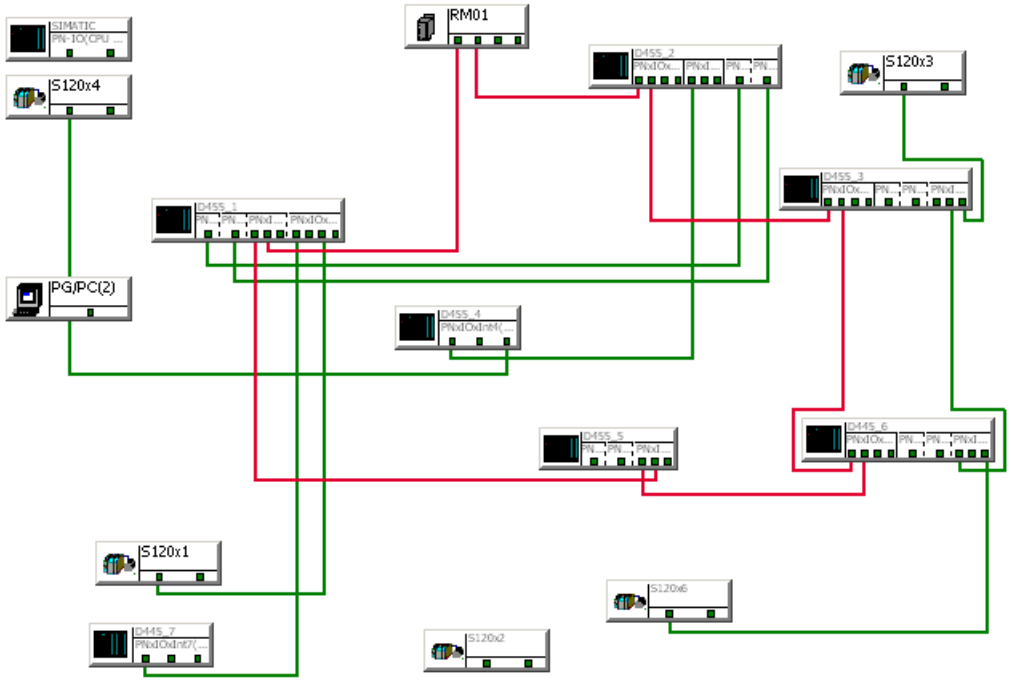


Figure 5-57 Ring topology in the Topology Editor

5.3.17.2 Setting up MRP domain

MRP domain

An MRP domain containing all ring nodes is required for media redundancy. If you are using several ring topologies, you must set up the corresponding number of MRP domains.

Setting up MRP domain

In order to set up an MRP domain, proceed as follows:

1. Select the PROFINET IO system in the working area of HW Config.
2. Go to **Edit > PROFINET IO > Domain Management**.
   The **Domain Management** dialog opens.

3. Switch to the **MRP domain** tab.

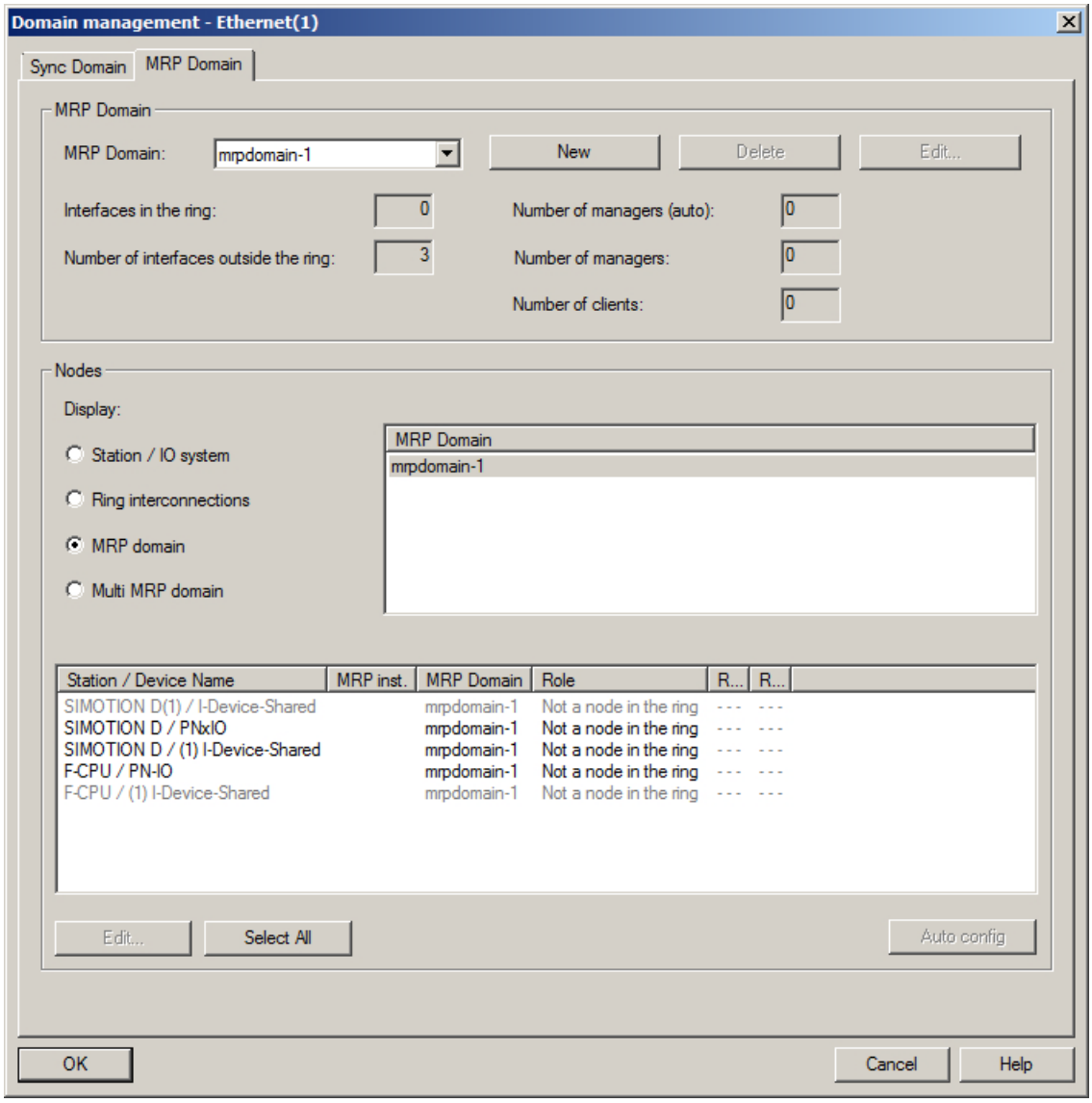


Figure 5-58 Domain Management MRP domain

4. Click **New** to create a new MRP domain. By default, the domain mrpdomain-1 will be created.

Assign the individual devices to the domain according to their roles.

1. Select the device under **Station/device name**and click on **Edit**.
   The "Edit Media Redundancy" dialog box opens.

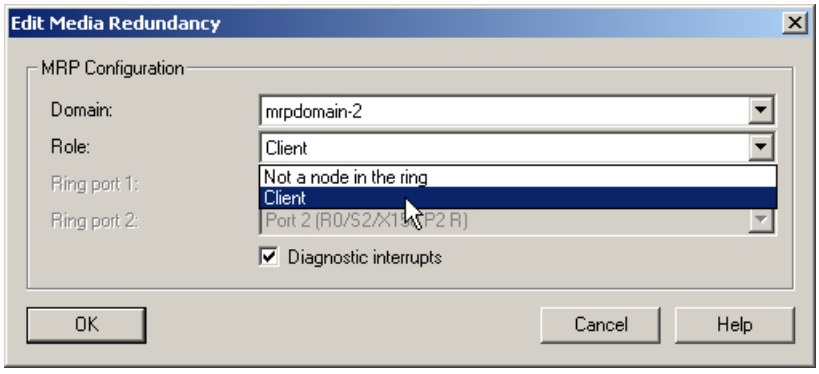


Figure 5-59 Edit media redundancy

2. Select the role intended for the device, e.g. Client, under **Role**.
3. Click OK to accept the settings.
4. Repeat the steps for all MRP domains and devices.

Result

The MRP domains have been created and the corresponding roles have been assigned to the members of the individual domains.

5.3.17.3 Configuring media redundancy

Entering settings for media redundancy

For media redundancy, you must assign the various roles and an MRP domain to the nodes.

Configure MRPD

If all participants within the MRP domain work in IRT mode (IRT with High Performance), MRPD is automatically activated for all nodes of the IRT domain that communicate via the ring.

Requirements

You have set up the devices in HW Config.

Procedure for the Redundancy Manager

To configure the Redundancy Manager, proceed as follows. Only SCALANCE switches can be Redundancy Managers.

1. Select the SCALANCE switch in the working area of HW Config.
2. Double-click on the PN interface in the detailed view; for example, on X1 in a SCALANCE switch.
   The **Properties - PN (xx)** dialog opens.

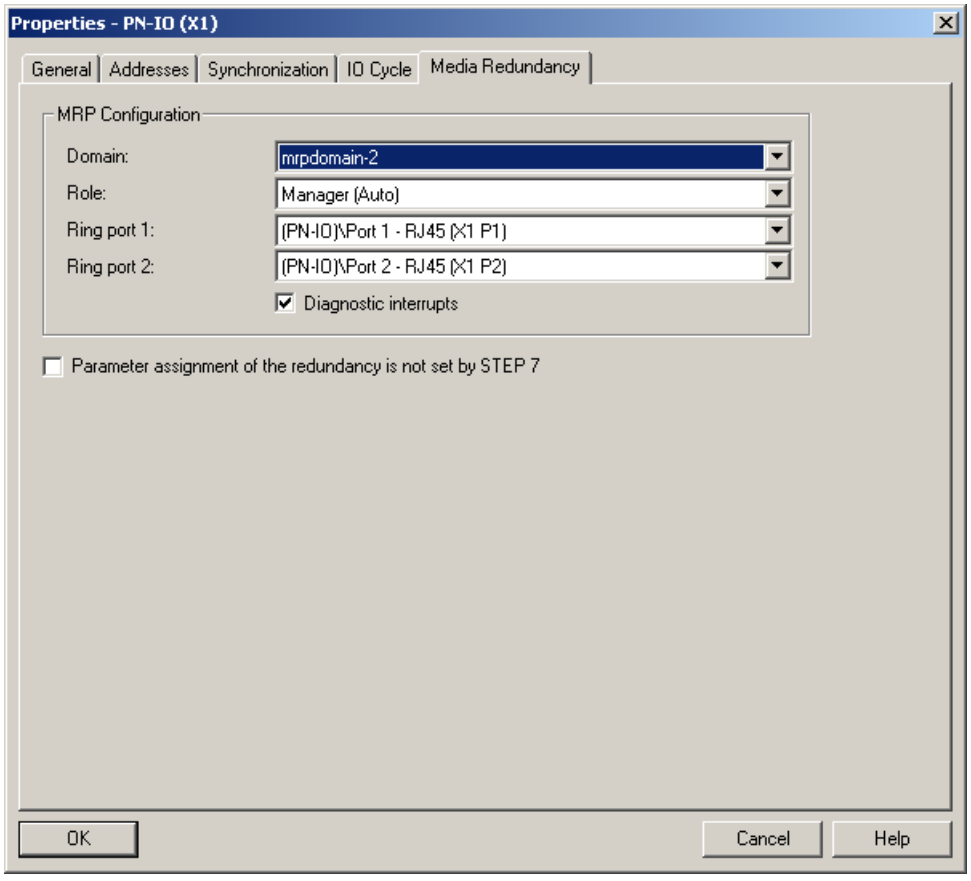


Figure 5-60 Properties Redundancy Manager

3. Switch to the **Media Redundancy** tab.
4. Select the MRP domain under **Domain**. If you are using only one domain, this is pre-assigned.
5. Under **Role** select the entry **Manager (Auto)**.
6. Under **Ring port 1** and **Ring port 2**, select the ports which are to be used in the ring (only with the SCALANCE switch).
7. Activate the **Diagnostic interrupts** option if you want to see ring communication faults in the diagnostics buffer and in the device diagnostics.

Procedure for the redundancy client

In order to configure devices as redundancy clients, proceed as follows:

1. Select the device in the work area of HW Config.
2. In the detailed view of the device, double-click on the interface line, e.g. on X1400 for a SIMOTION controller.
   The **Properties - Interface (xx)** dialog opens.

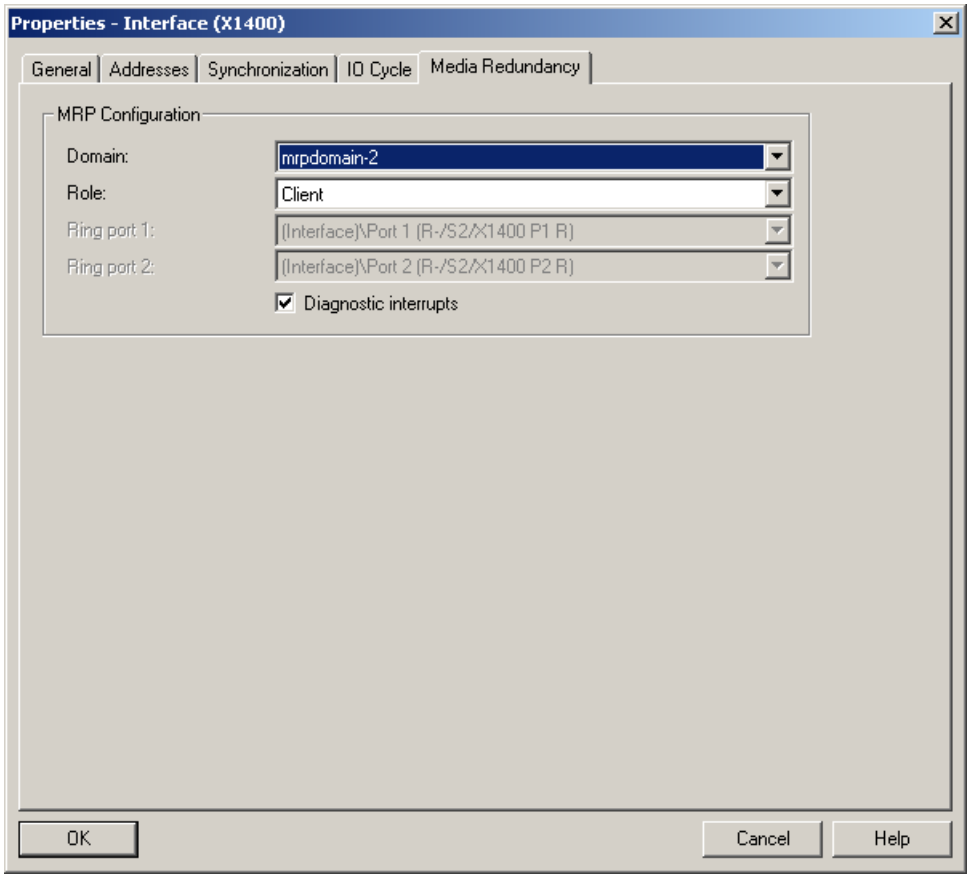


Figure 5-61 Properties of redundancy client

3. Switch to the **Media Redundancy** tab.
4. Select the MRP domain under **Domain**. If you are using only one domain, this is pre-assigned.
5. Activate the **Diagnostic interrupts** option if you want to see ring communication faults in the diagnostics buffer and in the device diagnostics.

5.3.18 Identification and maintenance (I&M) data

Introduction

For the identification and maintenance data (I&M), data structures are defined in PROFINET which can be implemented for all devices. These data structures are used to uniquely identify the IO device. The data set for the unique identification of an IO device, the identification and maintenance data set 0 (I&M0), is supported by all PROFINET IO devices. Further information can be stored in a standardized format as I&M 1-4 if required. As of V4.4, SIMOTION supports all I&M data.

PROFINET I-device I&M data

The I&M data is defined for a device (I-device in this case) and is readable and writable when the I-device is used by a higher-level controller. As of V4.4, SIMOTION supports the data sets 1-4 in addition to the I&M data set 0 (electronic rating plate) as a PROFINET I-device. Reading/ writing via PROFINET data set services is supported. Access from the user program via the SIMOTION user interface is not possible.

I&M data sets:

- I&M0: Electronic rating plate
- I&M1: Location designation/Plant ID
- I&M2: Installation date
- I&M3: Description
- I&M4: Signature/additional information

Structure of the I&M data

The following description provides an overview of the I&M data structure. A detailed description can be found in the PROFINET specification.

- 0xAFF0 I&M0 (read) - electronic rating plate
  - Vendor ID (Unsigned16)
  - Order ID (VisibleString[20])
  - Serial Number (VisibleString[16])
  - Hardware Revision (Unsigned16)
  - Software Revision (“Vx.y.z”)
  - Revision Counter (Unsigned16)
  - Profile ID (Unsigned16)
  - Profile Specific Type (Unsigned16)
  - I&M Version (OctetString[2])
  - I&M Supported (Bit Sequence V2)
- 0xAFF1 I&M1 (read/write) - location designation/plant ID
  - Function (submodule’s function or task; “AKZ”; VisibleString[32])
  - Location (submodule’s location; “OKZ”; VisibleString[22])
- 0xAFF2 I&M2 (read/write) - installation date
  - Installation Date (date of installation or commissioning of a device or module; VisibleString[16])
- 0xAFF3 I&M3 (read/write) - description
  - Descriptor (allows storing any individual additional information and annotation; VisibleString[54])
- 0xAFF4 I&M4 (read/write) - signature disabled for PROFIsafe purposes, not changed by ResetToFactory
  - Signature ("security" code, e.g. for Safety; OctetString[54])

Constraints on the I-device I&M data

Please note the following constraints.

- The I&M 1-4 data can only be accessed if the option **Parameter assignment for the PN interface and its ports on the higher-level IO controller** has been activated on the **I-device** tab of the PN interface.
- In the case of a SIMOTION CPU with two PN-IO interfaces, each I-device has its own writable I&M data.

- The second SIMOTION PN-IO interface is a plug-in interface that has its own article number. Both I-devices, therefore, also have different I&M 0 data.
  - Onboard PN interface: SIMOTION CPU article number
  - CBE30-2 PN interface: CBE30-2 article number
- The rack for the I-device I&M data is the first subslot of the transfer areas (subslot 1000). The I&M 1-4 data can only be written via this subslot. The I&M data for the other subslots is also shown when reading.

Reading and writing I&M data

You can display the I&M data in SIMATIC Manager or in HW Config. If you have an online connection, you can also change the data.

Setting I&M data in SIMATIC Manager

The PG/PC is connected online with the SIMOTION I-devices. They are connected to the same physical Ethernet subnet.

1. Select the menu command **PLC > Display accessible nodes**.
2. Select the SIMOTION I-device that is found
3. Select menu command **PLC > Change module identification**. A window containing the I&M data appears.
4. Change the required I&M data. The **Take into account** checkbox must be activated for the change to be transferred to the module.
5. Click **OK** to confirm. The data is changed.

Changing I&M data in HW Config

1. In HW Config, double-click the SIMOTION I-device symbol in the working area. The **Properties window** is displayed.
2. You can view the I&M data in the **Identification** tab.
3. If you have an online connection, you can display a comparison of the offline and online data via the menu command **PLC > Load module identification**.
4. In order to change the data, you must activate the **Take into account** checkbox and confirm the new data with **OK**.

5.4 Configuring direct data exchange (data exchange broadcast) between IO controllers

5.4.1 Introduction

Introduction

I/O data areas can be exchanged cyclically between two or more SIMOTION controllers via IRT High Performance. This is also referred to as controller-controller data exchange broadcast. Controller-controller data exchange broadcast is only possible between SIMOTION controllers via PROFINET IO with IRT High Performance.

For data exchange to take place, the devices must be located in a common sync domain and configured accordingly as sync master and sync slaves.

Note

This function is not available for SIMATIC CPUs.

There are in fact two types of data exchange broadcast. One is automatically created by the system (e.g. distributed synchronous operation), while the other can be applied by the user in his or her application. You can configure this second type of data exchange broadcast.

Note

The user must not use the engineering tools to make changes (e.g. amending the address areas) to the data exchange broadcast which has been automatically configured by the system. Doing so will result in error states.

Note

The controller-controller data exchange broadcast can only be configured if all the SIMOTION controllers involved in slave-to-slave communication are together in one project. If this is not the case, alternatively the SIMOTION I-device can be used for cross-project data exchange broadcast.

Recommendation

We recommend, initially configure the send areas for all PROFINET devices and then the receive areas. Adopting this procedure will enable you to assign the previously defined send areas when defining the receive areas. This prevents invalid inputs.

Data volume

Approx. 3 KB (4 frames of 768 bytes each) can be transferred. 24 bytes are needed for each configured synchronous operation relationship. In other words, if 5 following axes were defined for one master axis, the system needs 5 * 24 bytes. The remaining data is available for application-specific data exchange broadcast. In controller-controller data exchange broadcast, the max. submodule size is limited to 254 bytes.

Note

An FAQ section on the subject of PROFINET configuration is provided in SIMOTION Utilities & Applications. SIMOTION Utilities & Applications is provided as part of the SIMOTION SCOUT scope of delivery.

This FAQ section deals with the subjects of distributed gearing and controller-controller data exchange broadcast (http://support.automation.siemens.com/WW/view/en/38486079).

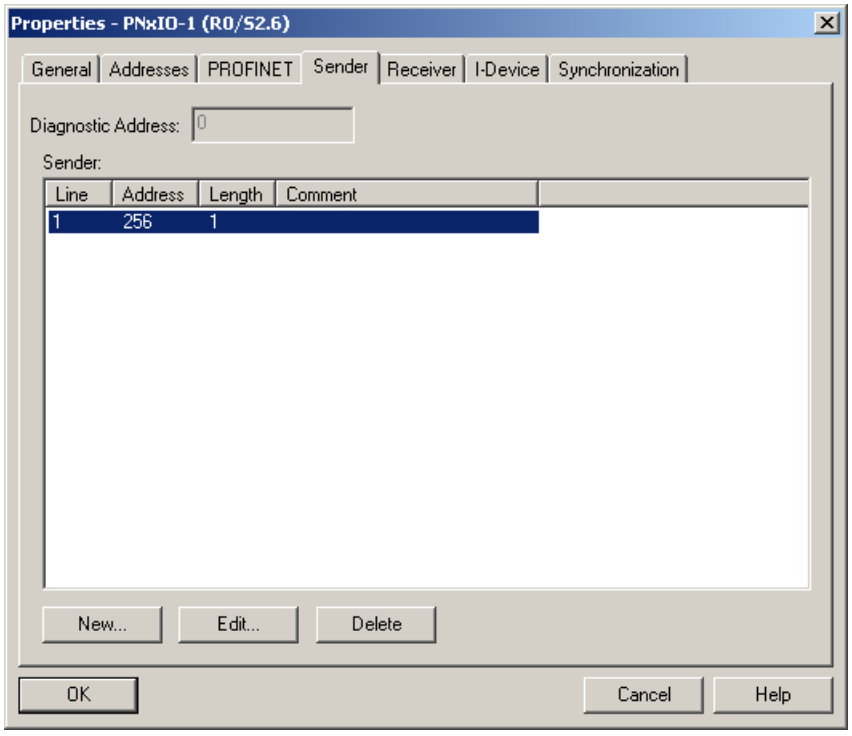
See also

Quantity structures (Page 94)

5.4.2 Configuring the sender

Procedure

1. Open the Properties dialog of the PROFINET interface (double-click the corresponding row in the configuration table of HW Config).
2. Select the **Sender** tab.



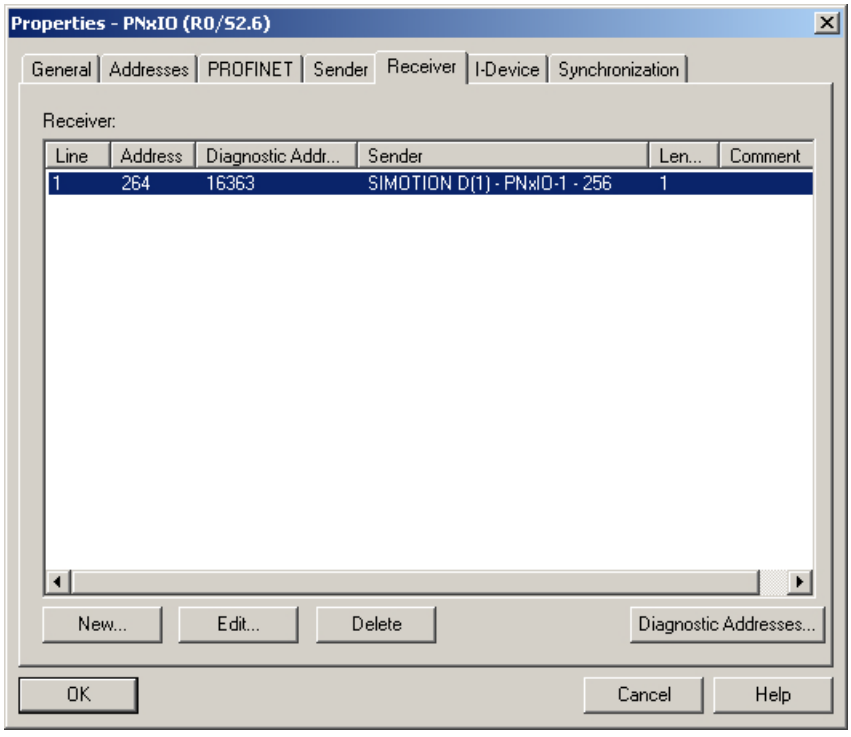
3. Click the **New** button.
4. Enter in the Properties dialog box of the sender, the start address from the I/O area and the length of the address area to be used for sending. Comment the data area so that you will be able to identify the data transmitted via this area later on. The maximum size of a variable is limited to 254 bytes (in the case of controller-controller data exchange broadcast).
5. Confirm the settings by clicking **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**.

Note

If a PROFINET interface of the SIMOTION controller is configured as a sender for controller-controller data exchange broadcast, this sender will have precisely one diagnostics address. This also applies if several slots are created for the data exchange broadcast.

5.4.3 Configuring the receiver

Procedure



1. Open the Properties dialog of the PROFINET interface (double-click the corresponding row in the configuration table of HW Config).
2. Select the **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. Enter in the Properties dialog of the receiver, the start address of the address area to be used for receiving. The length of the address area must not be changed as it is automatically adapted to the length of the send area. The configuration can only be compiled if the send and receive areas have identical lengths!

   Note

   Unlike the sender, each slot is assigned its own diagnostics address in the configuration of the receiver, through which the receiver can detect sender failure, for example. Click the Diagnostics addresses button if you want to edit this address.

8. Repeat steps 3 to 7 for further receive areas.
9. Confirm your entries with **OK**.

5.5 Configuring the iDevice

5.5.1 PROFINET IO and I device

Introduction

Up to SIMOTION 4.0, direct coupling (of SIMATIC and SIMOTION via PROFINET, for example) was only possible via TCP or UDP, or via additional hardware (PN/PN coupler, SIMATIC CP). With SIMOTION V 4.1.1.6 or higher, the direct coupling of controls - a familiar feature with PROFIBUS - has been introduced for PROFINET IO. It is possible, for example, to connect the SIMOTION as an I slave to the SIMATIC CPU via PROFIBUS. A similar function, referred to as "I-device", is also available for PROFINET IO. This supports data exchange between the controls via I/O areas, with the communication programming required for TCP or UDP being replaced by configuring and system functionality. In addition, the costs associated with the hardware solutions used previously no longer apply (PN/PN coupler, SIMATIC-CP).

The "I-device" (intelligent IO device) functionality of a controller enables exchange of data with an IO controller. Here the I-device is connected as an IO device to a higher-level IO controller. Communication can be established in both directions (bidirectional) via I/O areas using the I-device functionality. Further, this function allows both controls to operate in separate projects.

When operating as an I-device, a SIMOTION device can be used for data exchange with a SIMATIC station, for example. A SIMOTION device acting as an I-device may also be used as a feeder for a modular machine. Please see the *description of functions titled Motion Control basic functions for modular machines*.

Another application for a SIMOTION device acting as an I-device involves providing distributed synchronous operation beyond the limits of a project (see the *Motion Control Technology Objects Synchronous Operation, Cam* Function Manual).

Note

An I-device can only be created using SIMOTION V4.1.1.6 or higher.

Properties of an I-device

As well as performing the role of an IO device on a higher-level IO controller, an I-device can set up its own local PROFINET IO system with built-in local IO devices, thereby acting as an IO controller itself. Both these functions are implemented via the same PROFINET interface on the device.

With SIMOTION, the I-device is available for PROFINET IO with RT and IRT High Performance.

The following constraint applies to the options for combining functions:

Table 5-5 RT and IRT I-device combination options with SIMOTION

SIMOTION function	Possible additional functions			
	RT I-device	RT controller	IRT I-device	IRT controller
RT I-device		X	-	X
RT controller	X	-	X*	X*
IRT I-device	-	X	-	-
IRT controller	X	X	-	

*Either an IRT I-device or an IRT controller

As with any other IO device, an I-device's PROFINET interface requires parameter assignment data in order to operate. With IO devices, this data is usually loaded via the associated IO controller in the form of parameter assignment data sets.

Two options are available for I-devices.

An I-device's interface and PROFINET interface ports can either be parameterized by the higher-level IO controller or by the I-device itself on a local level. The preferred option can be selected as part of the I-device's configuration.

- The higher-level IO controller does **not** need be used for parameterizing the PROFINET interface of the I-device.
  This option should be used if the I-device is to be operated as an RT I-device. The same interface can then additionally be operated as an RT controller or an IRT controller.
- The IO controller loads parameter assignment data sets for the PROFINET interface to the I-device.
  This option should be used if the I-device is to be operated as an RT I-device. The same interface can then also only be operated as an RT controller (IRT controller only possible on a second PROFINET interface).

If the I-device is being operated with IRT, the I-device's send clock must be set to match the send clock for the sync domain of the higher-level IO controller's PROFINET IO system. If the I-device is being operated with RT, the I-device's update time must be either set to match the send clock for the sync domain of the higher-level IO controller's PROFINET IO system, or down-scaled by a multiple of this send clock.

The table below contains details of the send clocks and update times which must be set, along with their possible combinations.

Send clocks/update times of an I-device
Higher-level IO controller and I-device with IRT, no local PROFINET IO system or local PROFINET IO system with IO devices with RT • I-device send clock: – Must be the same as the send clock for the higher-level IO controller. – To be set on the I-device in <PROFINET Interface> properties using the **Send clock** drop-down list box on the **PROFINET** tab

Higher-level IO controller with IRT and I-device with RT, local PROFINET IO system with IRT • I-device update time: – Must be an integral multiple of the send clock for the higher-level IO controller and the send clock for the IO controller on the I-device – To be set on the I-device proxy in <PROFINET Interface> properties, under **Update Time** on the **IO Cycle** tab
Higher-level IO controller and I-device with RT, no local PROFINET IO system • I-device update time: – Any of the possible update times for the I-device may be set. – To be set on the I-device proxy in <PROFINET Interface> properties, under **Update Time** on the **IO Cycle** tab
Higher-level IO controller and I-device with RT, local PROFINET IO system with IRT: • I-device update time: – Must be greater than or equal to the send clock for the IO controller in the I-device. This means that, for example, if an IRT controller with a 2 msec send clock is configured on the CPU of the RT I-device, the following condition applies to the higher-level controller: Update time of the higher-level controller for access to the I-device >= send clock on the CPU of the I-device. This means that update times of 2, 4, 8, ...ms are possible. – To be set on the I-device proxy in <PROFINET Interface> properties, under **Update Time** on the **IO Cycle** tab

The figure below shows how an I-device can be configured on a higher-level IO controller. The higher-level IO controller sets up a PROFINET IO system containing the I-device.. The I-device is able to set up a local PROFINET IO system. Each of these PROFINET IO systems can belong to its own sync domain. However, the I-device must only be assigned to one of the possible sync domains, as a PROFINET interface can only belong to a single sync domain.

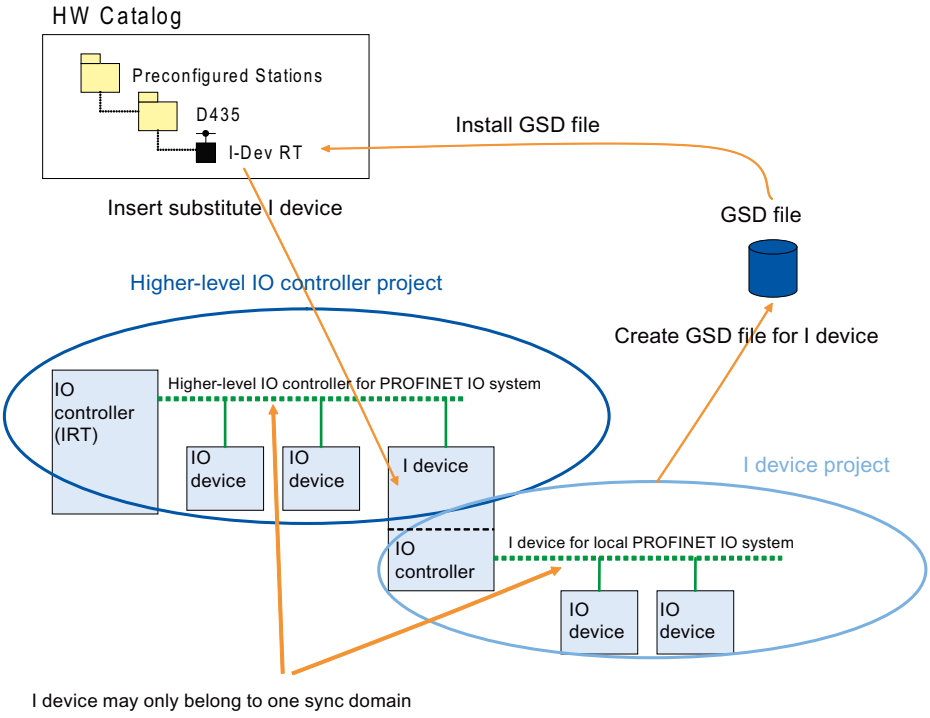


Figure 5-62 I-device configuration

Configuration procedure

- The I-device itself and the IO controller on which it is to be operated should be created in different projects.
- The PROFINET interface's I-device mode must be active for the I-device. In addition, the input and output ranges in the I-device must be configured for data exchange with the higher-level IO controller.
- After an I-device has been created and configured, a GSD file needs to be created and installed for its I-device proxy. The I-device proxy will then be available in the hardware catalog under "Preconfigured Stations."
- The next step involves inserting the I-device proxy from "Preconfigured Stations" in the hardware catalog into the higher-level IO controller's PROFINET IO system.

Since a manual process in the hardware catalog is required to create an I-device proxy, there is no automatic alignment between the project with the I-device and the corresponding I-device proxy in the GSD file. As a result, no subsequent modifications can be made to the I-device configuration. If a modification is made, however, a new GSD file will have to be created and installed. Where numerous modifications have subsequently been made to the configuration of a particular I-device, with multiple GSD files created and installed by the I-device, the version shown under "Preconfigured Stations" in the hardware catalog will always be the most recent one. The version will only be updated, however, if the identifier used for the I-device proxy when creating and installing the GSD file remains the same. Only the input and output addresses for data exchange may be modified in the project for the higher-level IO controller.

Since I-device connected to their higher-level IO controller and IO devices connected on the PROFINET IO system of a single I-device are connected via one and the same PROFINET

interface, they will also be located in one and the same Ethernet subnet. This means that the device names and IP addresses of all these devices must be different from each other, and the subnet masks must be identical. It is particularly important to bear this in mind if the higher-level IO controller and the I-device are in different projects, as HW Config cannot check that device names, IP addresses, and subnet masks are consistent across different projects.

Device name (NameOfStation) for the I-device

As with all IO devices on PROFINET IO, a device name also has to be defined for the I-device in the configuration. As the device name (NameOfStation) for the I-device is set in the properties for its PROFINET interface, it is identical to the device name of the IO controller in the I-device. The name set is written to the GSD file for the I-device proxy when this file is created and installed.

When the I-device proxy is inserted into the PROFINET IO system of the higher-level IO controller, the device name previously assigned in the GSD file will be accepted into the configuration. The **device name will not be accepted** if you insert the GSD into the same project on a different IO controller. This is often the case with connections between a SIMOTION controller and a SIMATIC CPU. In this case the device name is automatically manipulated by the engineering system (devicename"-1") and has to be manually adjusted to the original device name. Message 13:5144 ("There is already...") appears and must be confirmed.

Note

In all cases, it is important to ensure that the **device name in the configuration** of the higher-level **IO controller** is the **same** as the **device name** defined for the **I-device**. As a result, device names must not be amended after the higher-level IO controller has been added to the PROFINET IO system.

If the device names are different, the higher-level IO controller will be unable to identify the relevant I-device and, consequently, will not commence cyclic exchange of input/output data.

Applications of device names for the I-device

- Since a device name must not be used twice within an Ethernet subnet, any device name which already exists will be amended when an I-device proxy is inserted into the PROFINET IO system of its higher-level IO controller. In view of this, it is important to ensure that the device name previously assigned in the GSD file is not used at this stage.
- An I-device proxy can be used multiple times if multiple SIMOTION controllers with the same configuration are to be connected to a higher-level IO controller. You only have to ensure that each SIMOTION controller has a unique device name.

See also

Creating an I device (Page 183)

Insert the iDevice substitute in the higher-level IO controller (Page 189)

Configuration example for SIMOTION D435 and SINAMICS S120 via PROFINET (Page 287)

iDevice (Page 64)

5.5.2 I device functionality with SIMOTION SCOUT V4.2 or higher

Description

With Step 7 5.5 or higher, SIMATIC CPUs can also be configured as I devices. The I device functionality of SIMOTION CPUs and SIMATIC CPUs has been consistently standardized. In the context of this standardization process, the GSD version has been set as V2.25. With SIMOTION projects that use a lower version than V4.2, the iDevice must therefore be re-exported / reinstalled and reintegrated into the project of the higher-level IO controller when upgrading to V4.2. If you edit projects without re-exporting and reinstalling the GSD file, a diagnostics buffer entry will be created on the CPU of the I device when establishing a connection via PROFINET.

Note

If you import, restore or insert a project that contains a module configured via a GSD file onto a computer, and then open with SIMOTION SCOUT, the GSD file used within it is installed automatically, if this is no longer present on the computer. The hardware catalog is not updated automatically and must be manually updated. Execute **Tools > Install catalog** in HW Config to do this.

Note

You can scan the logical address of a proxy iDevice using the function _getLogDiagnosticAddressFromDpStationAddress. The function _getnextlogaddress is not intended for this purpose.

Note

Boundary conditions for I device V4.2 or higher

STEP 7 V5.5 must be used for configuration on the higher-level IO controller, as this is the minimum version required to import the GSD file V2.25 file.

When upgrading older versions of the SIMOTION controller to V4.2, the iDevice becomes "incompatible" and it is absolutely essential to export/import the GSD file.

Below you will find a list of various scenarios which illustrate I device compatibility.

Use case 1: Old project involving SIMOTION devices lower than V4.2 without modification to the I device.

1. Open an old project involving SCOUT V4.2/Step 7 5.5. The project contains RT I devices for communication with a SIMATIC CPU.
2. Change the project, but do not make **any** changes to the I device configuration.

3. Compile the SIMOTION project, including HW Config, and save it as SIMOTION SCOUT V4.1.
4. The project can be reloaded without any problems.

Use case 2: Old project involving SIMOTION devices lower than V4.2 with modification to the I device interface.

1. Open an old project involving SCOUT V4.2/Step 7 5.5. The project contains RT I devices for communication with a SIMATIC CPU.
2. Change the project and make changes to the I device configuration. Changes are automatically implemented on the I device interface and an IO slot is added.
3. Create a new GSD from the iDevice.
4. Install the exported iDevice and replace it on the higher-level SIMATIC CPU.
5. Compile the SIMOTION project, including HW Config, and save it as SIMOTION SCOUT V4.1.
6. The project can be reloaded without any problems.

Use case 3: Higher-level SIMATIC CPU and SIMOTION I device lower than V4.2 and upgrade to V4.2.

1. Open the project with a SIMOTION CPU and higher-level SIMATIC CPU. The SIMATIC CPU communicates as a PN controller with the I device of the SIMOTION CPU.
2. Upgrade the SIMOTION CPU to SIMOTION V4.2.
3. Export the I device of the SIMOTION CPU.
4. Diagnostics addresses and station numbers have changed compared to what they were in the previous I device. If these are used as absolute values in system calls in the SIMOTION application, the application program will need to be adapted accordingly.
5. Delete the previous I device of the SIMOTION CPU in the project of the higher-level SIMATIC CPU. Make sure that the input and output addresses of the new I device are identical to those of the old I device, so that the previous S7 application can be used.
6. Import the new GSD file into the project of the higher-level SIMATIC CPU.
7. Compile the SIMOTION project, including HW Config, and save it as SIMOTION SCOUT V4.2. The project has now been upgraded to V4.2.

Use case 4: Higher-level SIMOTION CPU and several SIMOTION I devices lower than V4.2 and upgrade to V4.2.

1. Open the project with several SIMOTION CPUs as I devices and a higher-level SIMATIC CPU. The higher-level SIMOTION CPU communicates as a PN controller with the I device of the SIMOTION CPU.
2. Upgrade the SIMOTION CPUs to SIMOTION V4.2.
3. Diagnostics addresses and station numbers have changed compared to what they were in the previous I device. If these are used as absolute values in system calls in the SIMOTION application, the application program will need to be adapted accordingly.

4. Export the I device of the SIMOTION CPUs and import the GSD file into the project of the higher-level SIMOTION CPU.
5. Compile the SIMOTION project, including HW Config, and save it as SIMOTION SCOUT V4.2. The project has now been upgraded to V4.2.

5.5.3 Creating an I device

Requirement

You have already created a project and created a station with rack or a SIMOTION controller in HW Config (SIMATIC Manager or SIMOTION SCOUT). You have configured the PROFINET IO system and now want to configure the I-device.

Note

When configuring the I-device, pay attention to the possible settings for the RT class, see PROFINET IO and I device (Page 176).

Procedure

1. Double-click on the PROFINET interface of the SIMOTION controller that you want to configure as an I-device. The **Properties** dialog box opens.
2. Switch to the **General** tab and, if necessary, change the device name. This must comply with the DNS conventions (Page 154).
3. Switch to the **I-Device** tab.

4. Select the **I-device mode** check box.

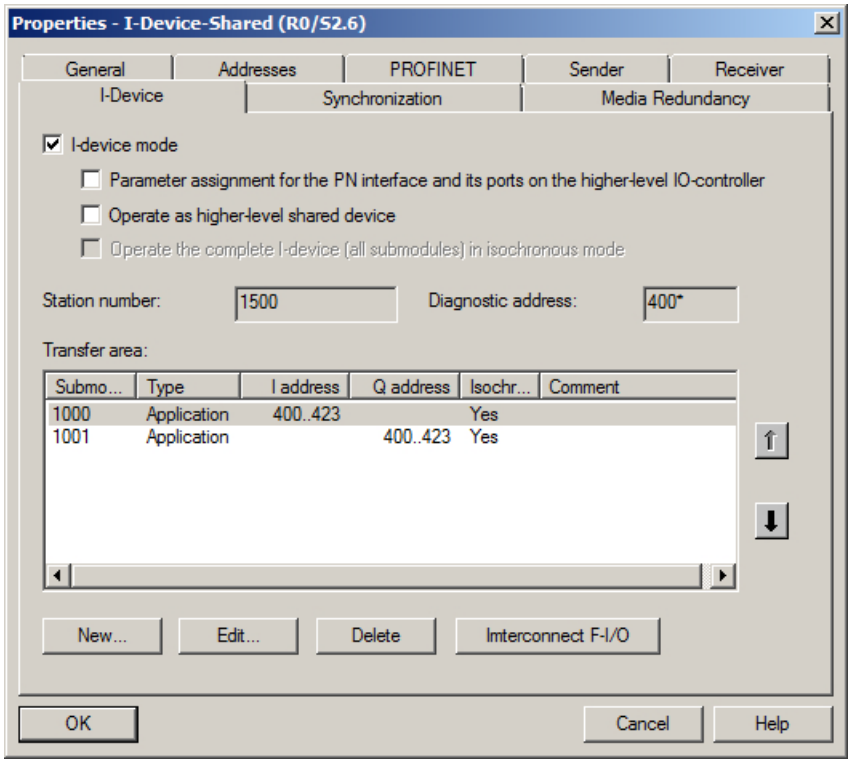


Figure 5-63 I-device properties dialog box

5. You have to activate different check boxes depending on whether I-device communication is to take place on the higher-level controller using RT or IRT.

   – **I-device using RT:**
   Select the **I-device mode** checkbox only.

   – **I-device using IRT**:
   If the I-device is to be operated on the higher-level IO controller isochronously, you also have to activate the **Parameter assignment for the PN interface and its ports on the higher-level IO controller** and **Operate the complete I-device (all submodules) in isochronous mode** checkboxes. This will also result in ports being created in the GSD file and parameter assignment data sets being loaded to the I-device's controller on start-up. If you do not select this check box cyclic communication between the higher-level IO controller and the I-device can only take place via RT. On the I-device proxy, this selection causes the **IO Cycle** tab to appear in the **Properties for the PROFINET interface** dialog box. It will then be possible to select the **Servo** entry under **Assign I-device in isochronous mode** on this tab so that the I-device can be operated isochronously. If an I-device is to be operated with IRT, the PN interface and its ports have to be parameterized on the higher-level IO controller and **Assign I-device isochronously** must be set.

6. You configure the **transfer areas** in the next step. Click **New**... each time to create the virtual subslots (input and output address transfer area), and configure these according to your requirements. By doing this, you are configuring the I/O range for the I-device via which data is exchanged with the higher-level IO controller. You do not need to configure any more settings in the **Sender** and **Receiver** tabs.
   Depending on the CPU used, the following settings are available for the transfer area type:
   - **Application**
   - **I/O**
   - **F-I/O**
     Note:
     You will find more information on the transfer area in the next section.
7. For the **transfer areas**, you can specify whether each individual area is to be operated isochronously, e.g. the fail-safe I/O cannot be operated isochronously. In order to select isochronous mode separately for each transfer area, the check box **Operate the complete I-device (all submodules) in isochronous mode** must be deactivated (from SIMOTION V4.4).
8. Click **OK** to accept these settings and save the project.
9. Continue by creating the I-device proxy (Page 187).

Creating transfer areas for an I-device

How to create an application transfer area:

1. Select **Application** for an application transfer area. STEP 7 automatically assigns the values of the transfer area on the higher-level IO controller (slot and subslot); the fields cannot be edited.
2. Specify whether the transfer area is to be a local input or output transfer area. To do this, select the corresponding **Address type**. STEP 7 automatically assigns the address type of the higher-level IO controller. If the transfer area is to appear as an output in the higher-level IO controller, it must be an input in the I-device, and vice versa.
3. For isochronous application areas, activate the **Isochronous** checkbox.
4. As with any other submodule, a transfer area requires an address space in order to be addressed by the user program; specify the start address, length, and the process image of the input/output.
5. Enter further information in the comment area as required and click **OK** to close the dialog box.

How to create an F-I/O transfer area (or I/O transfer area):

1. Select **F-I/O** (or **I/O**) for an I/O transfer area. STEP 7 automatically assigns the values of the transfer area on the higher-level IO controller (slot and subslot); the fields cannot be edited.

   Note

   The I/O transfer areas are established automatically for the entire configured fail-safe I/O using the **Interconnect fail-safe I/O** button. You no longer have to set up each fail-safe I/O individually.

2. Now specify which modules/submodules of the I-device are to be made available to the higher-level IO controller as I/O transfer areas. Click the **Select I/O** button: The **I/O transfer area - Select I/O** dialog box opens.
3. Select a module/submodule and click **OK** to close the dialog.
4. As with any other submodule, a transfer area requires an address space in order to be addressed by the user program. You must, therefore, specify the start address of the input/output. The length is determined automatically from the selected module/submodule.
5. Enter further information in the comment area as required and click **OK** to close the dialog box.

See also

Insert the iDevice substitute in the higher-level IO controller (Page 189)

5.5.4 Exporting the GSD file for the I device

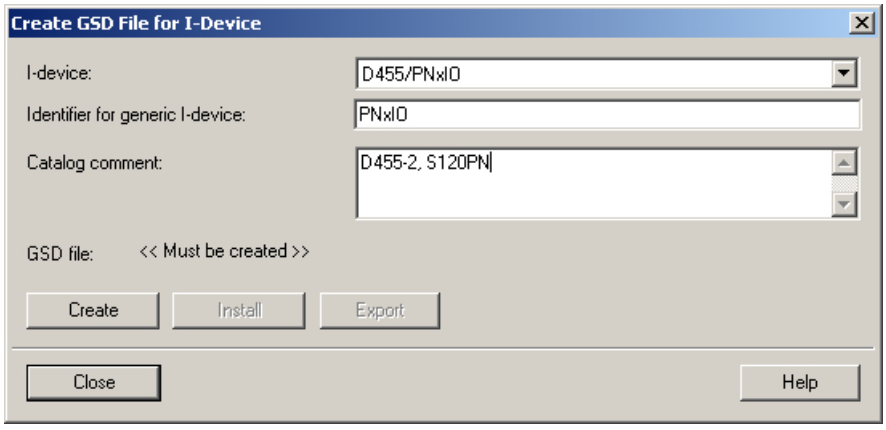
A GSD file must always be exported if an iDevice is to be used in a project on another PG/PC.

Requirement

You have already configured the module to be used as an I device.

1. First save the project.
2. Open HW Config and select **Extras > Create GSD file for iDevice...** from the menu. The **Create GSD file for I device** dialog opens.

3. Select the I device and enter a name for the substitute I device. The substitute I device will appear under this name in **Preconfigured Stations** in the HW catalog.



4. Click **Create** and then **Export**. The **Find folder** dialog opens.
5. Select the path in which the GSD file of the substitute I device is to be stored and click on **OK**.

5.5.5 Creating a substitute I device

There are two different ways of creating a substitute iDevice. The first, **Options > Install GSD files ...** , involves a previously exported GSD file. The second involves the **Create GSD file for I device** dialog.

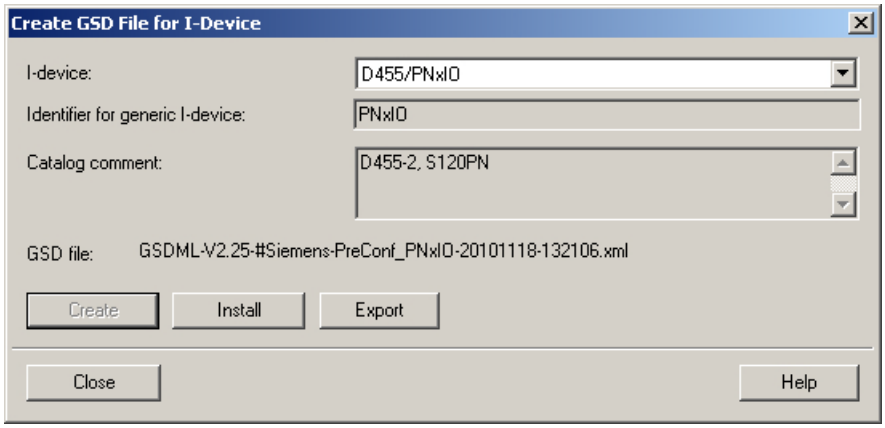
Requirement

You have already configured the module to be used as an I device and exported the GSD file from this.

Procedure 1:

1. First save the project.
2. Create an iDevice as described in Chapter Exporting the GSD file for the I device (Page 186). There is no need to export the file; instead, it is installed immediately.

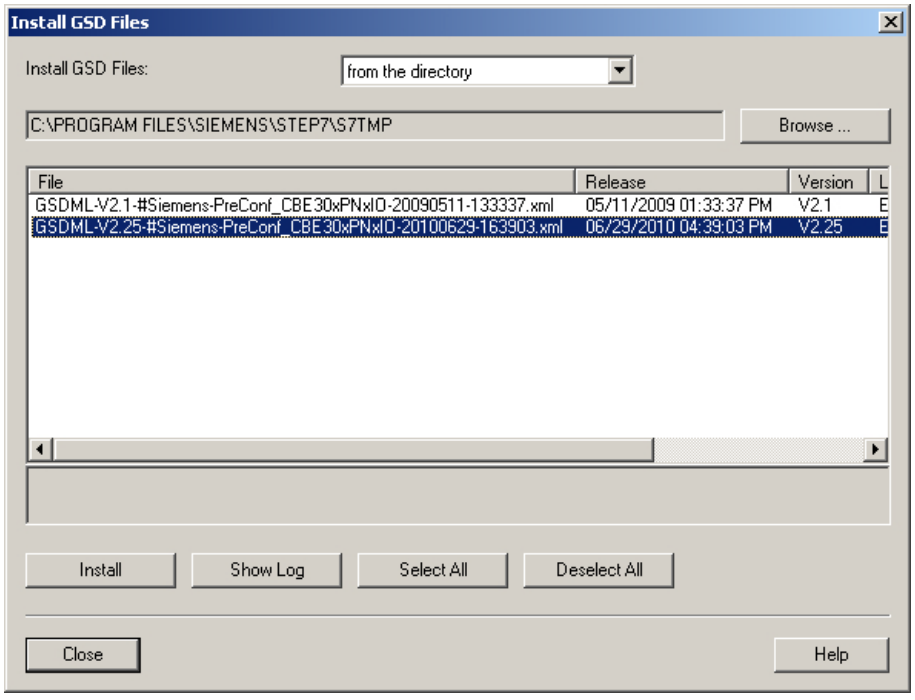
3. Click **Create** and then **Install**.



4. Click **Close**. The substitute I device will now be available under "Preconfigured Stations".

Procedure 2:

1. Select **Extras > Install GSD files....** . The "Install GSD files" dialog box opens.
2. Click **Browse...** . The **Find folder** dialog opens.
3. Select the path in which the GSD files of the substitute iDevice are stored and click on **OK.**



4. Select the required GSD files and click **Install**.
5. Click **Close**. The substitute I devices will now be available under "Preconfigured Stations".

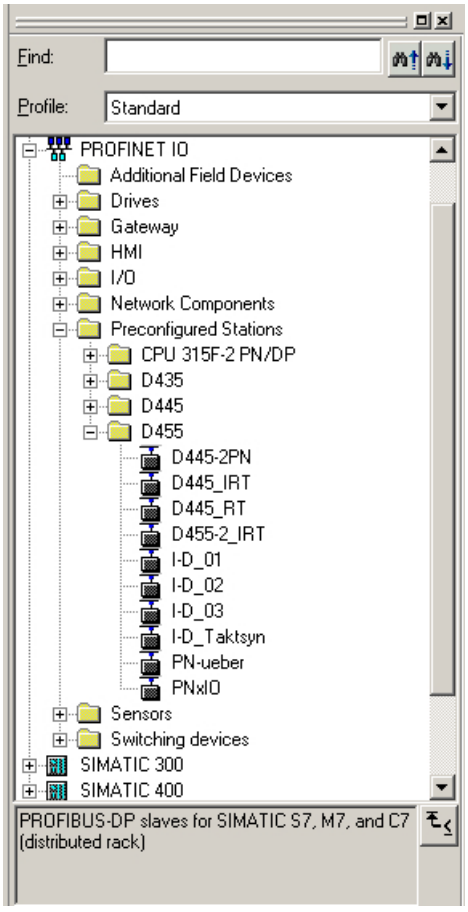


Figure 5-64 iDevice entry in the hardware catalog

5.5.6 Insert the iDevice substitute in the higher-level IO controller

Requirement

You have already created an I-device proxy. A project is open and an IO controller with a PROFINET IO system has already been configured.

Inserting an I-device proxy

1. Open the hardware catalog in HW Config.
2. Drag the relevant I-device proxy from the hardware catalog (**PROFINET IO > Preconfigured stations**) to the PROFINET IO system of the higher-level IO controller. The I-device proxy is displayed as a normal IO device on the PROFINET IO system. Depending on whether the option **Parameter assignment for the PN interface and its ports on the higher-level IO controller** has been selected for the I-device configuration, the PN interface and the individual ports of the I-device are also displayed here.

   Note

   Check whether the device name of the I-device proxy matches that of the PN interface of the corresponding controller. For more information, see the chapter PROFINET IO and I device (Page 176).

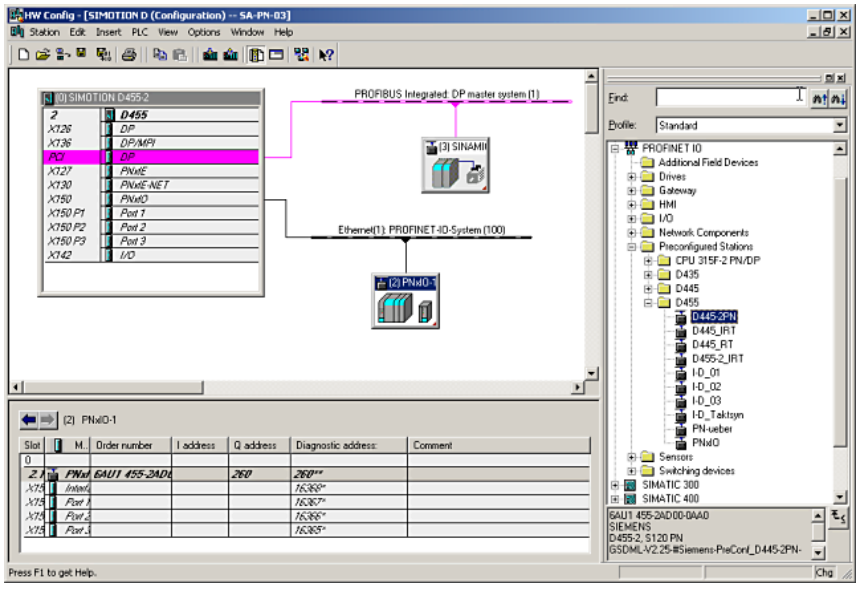


Figure 5-65 I-device on the IO controller

The number of submodules corresponds to the number of the configured submodules of the I-device in the GSD file. Modules and submodules (virtual subslots) cannot be deleted.

3. You can interconnect the I-device ports with the IO controller ports in the Topology Editor.

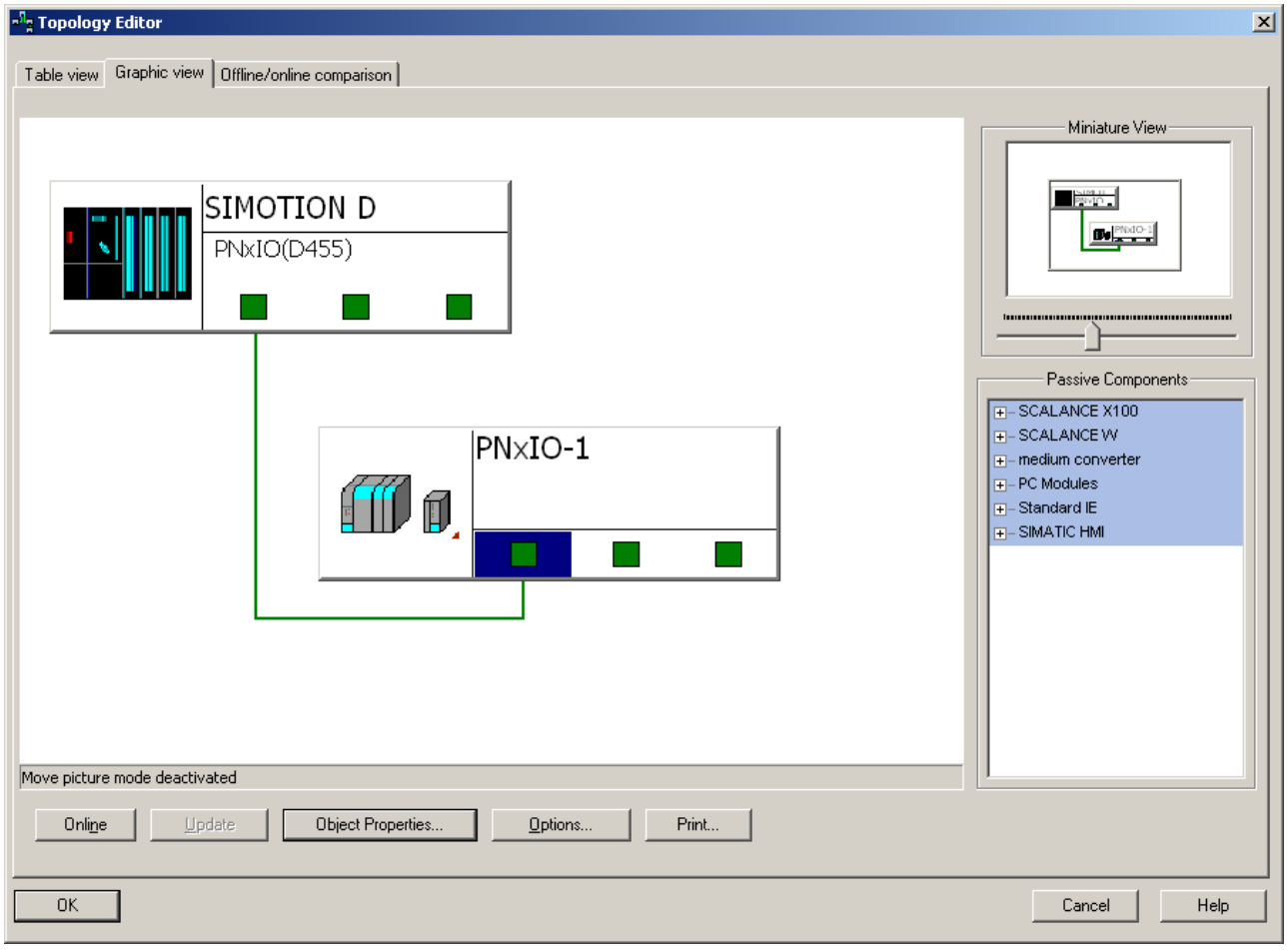


Figure 5-66 Interconnecting I-device ports

Assigning an IP address for the I-device proxy

1. Double-click the I-device to open the **Properties** dialog box. This displays how the IP address is assigned.
2. If the option **Obtain IP address using a different method** is activated for the SIMOTION controller that is configured as an I-device, **no** IP address will be saved in the project. Then the IP address must be assigned from the IO controller. If the option is **not** activated, then the IP address is already saved in the project.

Set synchronization role and isochronous mode for IRT I-device

1. Double-click on the PROFINET interface in the rack of the SIMOTION I-device to open its properties.
2. In the **Synchronization** tab, select **Sync-Slave** and **IRT** as synchronization role and RT class respectively.
3. Select Servo under **Assign IO device in isochronous mode** on the **IO Cycle** tab.

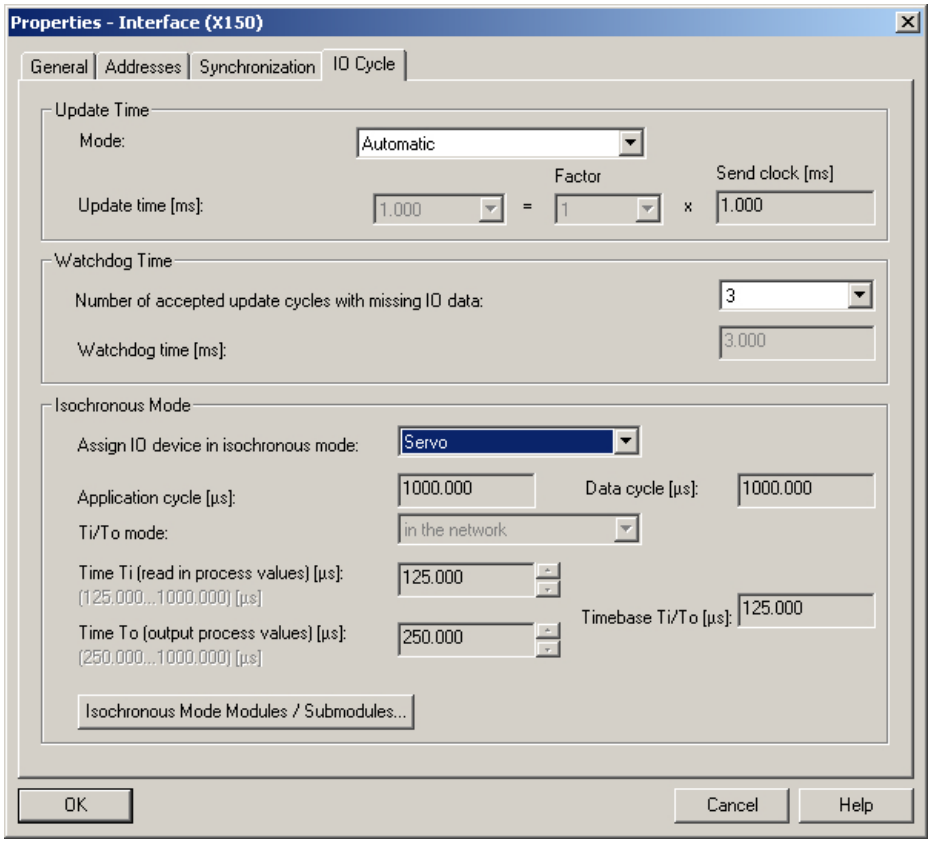


Figure 5-67 Setting isochronous mode on the I-device

Setting the update time and send clock

RT I-device

- The update time is to be set up for the RT I-devices. Double-click the PROFINET IO system and select the **Update time** tab in the **PROFINET subnet properties** dialog. Set the update time there.

IRT I-device

- The send clock needs to be set for IRT I-device. The setting made for the send clock in the I-device's project must be the same as for the send clock in the higher level IO controller's project. Set the send clock for the higher-level IO controller in HW Config using **Edit > PROFINET IO > Domain Management**.

See also

IP address and device name via UP/DCP (Mini-IP-Config) (Page 154)

5.5.7 Deleting a substitute I device

The GSD files for the substitute I device can be found under "Preconfigured Stations" in the following directory:

<Program Files>\Siemens\Step7\S7DATA\GSD,
e.g. GSDML-V2.25-#Siemens-PreConf_**D455-2_IRT**-20100830-132044.xml.

Here, **D455-2_IRT** is the name of the substitute I device. You can delete the substitute I device by deleting the corresponding XML files. The substitute iDevices displayed under "Preconfigured Stations" are only updated when a new GSD file is created and installed or when the HW catalog is manually updated in HW Config.

5.5.8 Shared iDevice

Introduction

As of V4.4, the iDevice of the PN-IO interfaces can be operated with two higher-level IO controllers as a shared iDevice. The properties and configuration are comparable with a shared device. Configuration is similar to that of a shared device.

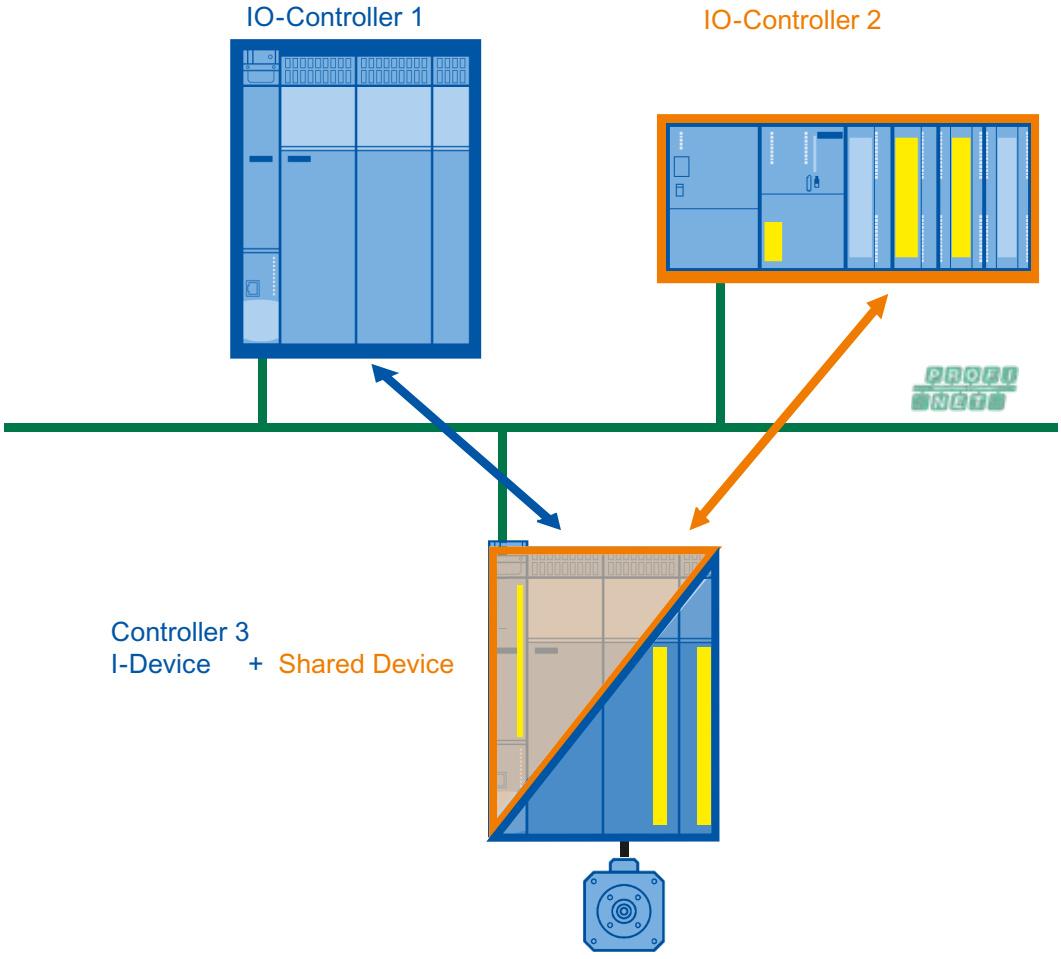


Figure 5-68 iDevice as a shared device

Boundary conditions

When using the shared iDevice, certain boundary conditions must be observed.

- Only one of the two higher-level IO controllers can communicate with the iDevice via PROFINET IRT. The interface module and the port submodule of the iDevice must be assigned to this IO controller.
- The second higher-level IO controller can only communicate with the iDevice via PROFINET RT.
- Each submodule of the iDevice can be assigned to one of the two IO controllers.

- Here a submodule can only be assigned to one of the two higher-level IO controllers.
- In the SIMOTION SCOUT the shared iDevice (shared master) appears only once in the project navigator.
  (The shared master: First device added directly to the network via drag&drop. The second device is created by copying the shared masters and clicking **Shared paste** in the context menu.)

Station failure and station reconnection

A shared iDevice communicates with two higher-level IO controllers which share the submodules of the iDevice. Using the diagnostic events "Partial station reconnection" and "Partial station failure", one can check on the CPU of the iDevice in an application-specific way whether only one or both higher-level IO controllers are connected to the iDevice. However, this is only possible if each submodule of the iDevice has also been assigned to one of the two IO controllers.

The establishment or removal of an application relationship between IO controller and iDevice is reported by opening the PeripheralFaultTask. A distinction can be thus made in an application-specific manner between the different cases using the reported event ID.

An overview of the evaluation with the PeripheralFaultTask can be found in the chapter Alarms on the IO controller (Page 201).

Shared iDevice send clocks with IRT

If a shared iDevice is operated in connection with PROFINET IRT, then only "even" send clocks (125, 250, 500, 1,000, 2,000, 4,000 µsec) can be used on the higher-level IO controllers for PROFINET RT and IRT.

Background:
As only "even" send clocks are permitted for PROFINET RT and the shared iDevice physically represents a module, only "even" send clocks may also be set on the higher-level IO controller which communicates with the iDevice via PROFINET IRT.

Shared iDevice and PROFIsafe

The shared iDevice function can be used to implement a PROFIsafe configuration with the automation CPU and safety CPU. You can find more information on this subject from the chapter Shared iDevice and PROFIsafe (Page 310) onward.

5.6 Loading the communication configuration

5.6.1 Loading the PROFINET IO configuration

Requirement

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

Procedure

The configuration data must be loaded to all participating controllers once PROFINET IO configuration has been successfully completed.

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

5.7 Communication connections between SIMOTION and SIMATIC

5.7.1 Communication connections overview

Description

The following options are available for establishing a communication connection between SIMOTION and SIMATIC:

- PROFINET RT (or IRT) via iDevice
- PROFINET RT (or IRT) via PN/PN coupler
- PROFINET RT via S7-300 CP as an IO device
- TCP/UDP user communication

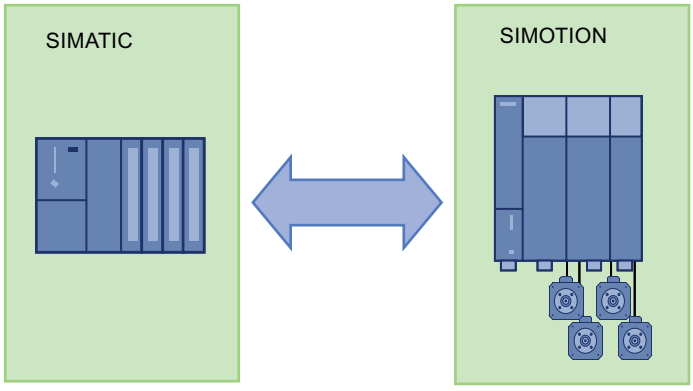


Figure 5-69 Data exchange between SIMOTION and SIMATIC

See also

Data exchange through the use of iDevices (Page 198)

5.7.2 Data exchange through the use of iDevices

Description

With PROFINET IO/RT, data can be exchanged via an I-device.

Note

A PROFINET device connected to a SIMOTION CPU may have a maximum submodule size of 1,024 bytes.

This limit is especially important if a SIMATIC CPU is configured as an I-device for a SIMOTION CPU. Depending on the hardware and the PN interface, the submodule size may vary from 254 to 1024. Therefore, please note the possible quantity structures (Page 94).

I-device FAQ

An FAQ section on the subject of coupling a PROFINET RT I-device between a SIMOTION control and a SIMATIC control is provided in SIMOTION Utilities & Applications. SIMOTION Utilities & Applications is provided as part of the SIMOTION SCOUT scope of delivery.

It looks at the following three application cases:

Case A: Two separate projects: SIMATIC and the SIMOTION as an I-device in a project; SIMOTION as a controller in a second, separate project

Case B: One project for all components

Case C: Multiple use of an I-device

For more detailed information on the configuration of I-devices, please refer to the chapter Configuring the iDevice (Page 176).

Also note the FAQ for **Configuring RT communication between SIMATIC and SIMOTION (I-device)** I device FAQ (http://support.automation.siemens.com/WW/view/en/38716528)

See also

Communication connections overview (Page 197)

5.8 Diagnostic and alarm behavior

5.8.1 Device model for PROFINET

Definition of device model

The device model of PROFINET describes the structure of modular and compact field devices. Similar to a PROFIBUS DP slave, a PROFINET IO device is structured in a modular way. In this way, modules are attached to slots, and submodules to subslots. Channels are located on the modules and submodules, through which process signals can be read and output. In principle it is possible to divide one slot into further subslots, into which the submodules can be plugged.

5.8.2 Diagnostics levels for PROFINET

Diagnostics levels

Each occurring fault in an IO device is transmitted from it to the IO controller. The range and depth of information of a diagnosis can vary depending at which diagnostics level the diagnostics data is evaluated.

In principle there is the option of evaluating diagnostics information at the following levels of address.

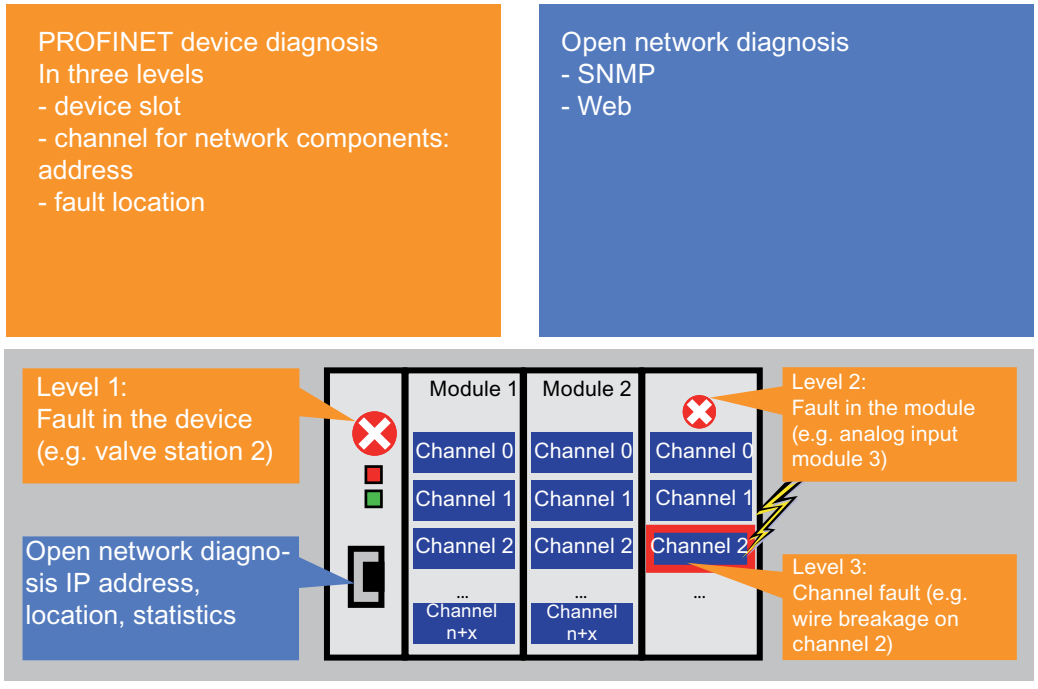


Figure 5-70 Diagnostics overview using the example of ET200

There is a group of standardized diagnostics data sets for PROFINET for each level of address with which the diagnostics information of the IO device can be read (e.g. via the SIMOTION system function _readRecord).

Note

You will find further information on the _readRecord system function in the online help or in the *"SIMOTION System Functions/Variables Devices List Manual"*.

You will find an overview of the data sets available with PROFINET, including the structure of data sets, from section 5 in the *Programming Manual "From PROFIBUS DP to PROFINET IO."*You can find the Programming Manual in the Service&Support portal as a download (http://support.automation.siemens.com/WW/view/en/19289930).

5.8.3 Using TaskStartInfo

TaskStartInfo

In addition to the option of reading the diagnostics information of IO devices through standardized data sets, with SIMOTION the so-called "TaskStartInfo" can also be evaluated for fundamental fault diagnostics.

Important information on the start of each task in the execution system of the SIMOTION controller is saved in the "TaskStartInfo". The respective "TaskStartInfo" can be requested within a task (for diagnostics events that concern peripherals connected to the SIMOTION controller, this request must be made in the "PeripheralFaultTask").

In this process the event that occurs is reported through the system variable "TSI#interruptId" (e.g. CPU stop, process alarm, etc.). The system variable "TSI#eventClass" specifies whether the fault that occurs is appearing or disappearing. On this basis a more precise specification of the event is performed using the system variable "TSI#faultId" (e.g. channel diagnostics pending, station reconnection of an IO device with / without faults, etc.).

Note

Further information:

- Example application SIMOTION: Diagnose CPU stops and module statuses (http://support.automation.siemens.com/WW/view/en/53705461).
- *Function Manual "SIMOTION Runtime Basic functions*" (section "Use Taskstartinfo").

5.8.4 Alarms on the IO controller

Description

A number of alarms are issued on the IO controller. Occurring alarms are listed with the corresponding EventID in the diagnostics buffer of SIMOTION. The following alarms are possible:

- Alarms for direct data exchange between IO controllers
- Station alarms reported by the PROFINET interface

The following table shows PROFINET IO alarms as they are represented in SIMOTION:

Alarm (TSI#InterruptId)	TSI#eventClass	TSI#faultId	Meaning
Station failure (_SC_STATION_DISCONNECTED (= 202))	16#39	16#CA	PROFINET IO system error: In this case there is only an incoming event; an outgoing event is represented on 16#38 - 16#CB for each IO device present.
		16#CB	Station failure of an IO device. Shared I-device: No submodule of the I-device is currently subscribed to by the higher-level IO controller.
		16#CC	IO device fault present. Channel diagnostics or manufacturer-specific diagnostics pending. **Note:** 16#39:16#CB is always used for the station failure; 16#38:16#CC is used for incoming with fault.
		16#F8	Partial station failure Shared I-device: Submodules of the I-device are partially subscribed to by the higher-level IO controller.
Station reconnection (_SC_STATION_RECONNECTED (= 203))	16#38	16#CB	An IO device has been reconnected with an error or warning. Shared I-device: All submodules of the I-device are currently subscribed to by the higher-level IO controller.
		16#CC	An IO device has been reconnected with an error or warning.
		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.
		16#F8	Partial station reconnection Shared I-device: Submodules of the I-device are partially subscribed to by the higher-level IO controller.

Use of TaskStartInfo

Information about using the TaskStartInfo for the PeripheralFaultTask can be found in the *"SIMOTION Runtime Basic Functions"* Function Manual.

5.8.5 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#eventClass	TSI#faultId	Meaning
Diagnosis (incoming)	16#39	16#42	Incoming diagnostic interrupt
Diagnosis disappears (outgoing) Multicast Communication Mismatch Port Data Change Notification Sync Data Changed Notification Isochronous Mode Problem Notification Network component problem notification (_SC_DIAGNOSTIC_INTERRUPT (=201))	16#38	16#42	Outgoing diagnostic interrupt
Process interrupt (_SC_PROCESS_INTERRUPT (= 200)	16#11	16#41	Process interrupt
Pull Alarm	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.
Plug Alarm Plug Wrong Submodule Alarm Return of Submodule Alarm (_SC_PULL_PLUG_INTERRUPT (=216))	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 <> preset 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
Status			Not Supported
Update			Not supported
Time data changed notification			Not supported
Upload and storage notification			Not supported
Pull module			Not supported

Alarm (TSI#InterruptId)	TSI#even-tClass	TSI#faultId	Meaning
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 TaskStartInfo

Information about using the TaskStartInfo for the PeripheralFaultTask is contained in the *SIMOTION Basic Functions* manual.

Transfer diagnostic data

The exact reason for the alarm is provided as diagnostic data. The _readDiagnosticData() function can be used to read out this data. The length is restricted to 240 bytes.

With V4.2 or higher, you can read out station/module diagnostic data of up to 65,535 bytes with the function block _readVariableDiagnosticData() and the user program.

5.8.6 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.

Note

During communication monitoring, only the receiver slot is monitored.

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 without errors and the receiver is receiving data again (all receiving areas are available).
		16#F2	The receiver in the direct data exchange has started up with errors and the receiver is receiving data again (at least one receiving area not available).

5.8.7 SINAMICS drives alarms

Description

In SINAMICS firmware Version 4.5 and higher, SINAMICS drives (S and G) have a PROFIBUS/PROFINET diagnostics channel. Alarms can be forwarded to a higher-level control via this channel.

The drive is controlled via a TO

If the drive is controlled via a TO, drive alarms are issued via the alarm mechanisms of the technology object.

The drive is controlled directly by means of a user program

If you are controlling the drive directly by means of a user program, you must program the alarm response.

See also

Alarms on the IO controller (Page 201)

5.8.8 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 about each of the functions in the reference list for functions *List Manual, SIMOTION system functions/variables for devices*.

Function	Note	PROFIBUS	PROFINET
_getStateOfSingleDpSlave()	The function determines the status of communication with a cyclic communications partner (device - PROFINET, slave - PROFIBUS, transmitter or receiver controller-controller data exchange broadcast - PROFINET).	Logical diagnostic address of the DP slave	Logical diagnostic address of the IO device
_getStateOfDpSlave()	_getStateOfDpSlave provides information concerning whether the PROFIBUS DP slave or the PROFINET IO device is activated or deactivated.	Logical diagnostic address DP slave	Logical diagnostic address of the IO device
_getStateOfAllDPStations()	The system function determines the status of the communication with cyclic communications partners (device - PROFINET, slave - PROFIBUS, transmitter or receiver controller-controller data exchange broadcast - PROFINET). In addition to the activation status, information on availability is also provided.	Logical diagnostics address of the interfaces	Logical diagnostics address of the interfaces 4.2 and higher With PROFINET IO devices, this system function forms a group signal from the modules present in the device. This means that the function can also tell the user program when modules have been removed or are faulty by using the device feedback values for this purpose.
_getStateOfIO()	This function provides the user program with information on the status of the DP stations, modules, and submodules. It also provides this information in detail for modules and submodules. The following are supported: • Interfaces such as PROFIBUS DP or PROFINET • Stations (slaves, devices), modules, or submodules This function provides a logical diagnostics address or logical I/O address to identified modules, and a list of assigned submodules.		

Function	Note	PROFIBUS	PROFINET
_getNextLogAddress()	All configured logical addresses of a segment can be determined using this function.	Logical diagnostics address	Logical diagnostics address
_readDiagnosticData() _getStateOfDiagnosticDataCommand()	This function is used to output diagnostic data for a DP slave via the user program. For PROFIBUS, the diagnostics for the slave are read out, i.e. the slave supplies the complete diagnostic information. The structure of the diagnostic data is described in IEC 61158-6-3. For PROFINET, the subslot-specific diagnostics are read (i.e. the data set 0x800A). The diagnostics for a subslot are supplied. The structure of the diagnostic data is described in IEC 61158-6-10.	Logical diagnostic address DP slave	Logical diagnostics address of the IO device, of the module/slot, or IO address of the channel/subslot
_readVariableDiagnosticData()	This function block is used to read out diagnostic data of a station or module via the user program. The diagnostics format for PROFINET IO V2.2 and PROFIBUS DP is described in the IEC 61158-6 standard.	-	Data records with a total length up to 4 KB can be read with the FB _readVariableRecord. This function can be used for PROFIBUS and PROFINET. Data sets up to 4 KB are returned from the device only with PROFINET.
_readDriveFaults() SINAMICS only (ET200FC, S120, etc.)	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 diagnostics address of the PAP (for modules without cyclic data). **Note:** In the case of PROFINET, parameter access takes place via the MAP (Module Access Point) of a DO. The MAP submodule is the substitute DO and hosts the alarm channel along with one or more PAPs (Parameter Access Point). The PAP is a data record in the MAP submodule. MAP is displayed in the GSD file.

5.8.9 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.

Procedure

1. Go online and open 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.

5.8.10 PROFINET IO and DS0 diagnostic interrupts

5.8.10.1 Diagnostic interrupt PROFINET IO maintenance concept

Description

A device or module of an automation system can essentially be in one of two states: either 'good' or 'failure.' With a view to increasing the availability of sensors/actuators, devices, and modules, these same components also supply information concerning necessary maintenance work in addition to these two states. This additional information includes details about the maintenance state, state of wear, and remaining life time, etc. This constitutes what is known as an 'extended maintenance concept'. The aim of the extended maintenance concept is to detect and eliminate potential faults early on - before they lead to production outages. For this purpose, the 'good' and 'failure' states are supplemented by states called 'maintenance required' and 'maintenance demanded'.

The above states can be depicted on the following maintenance state model.

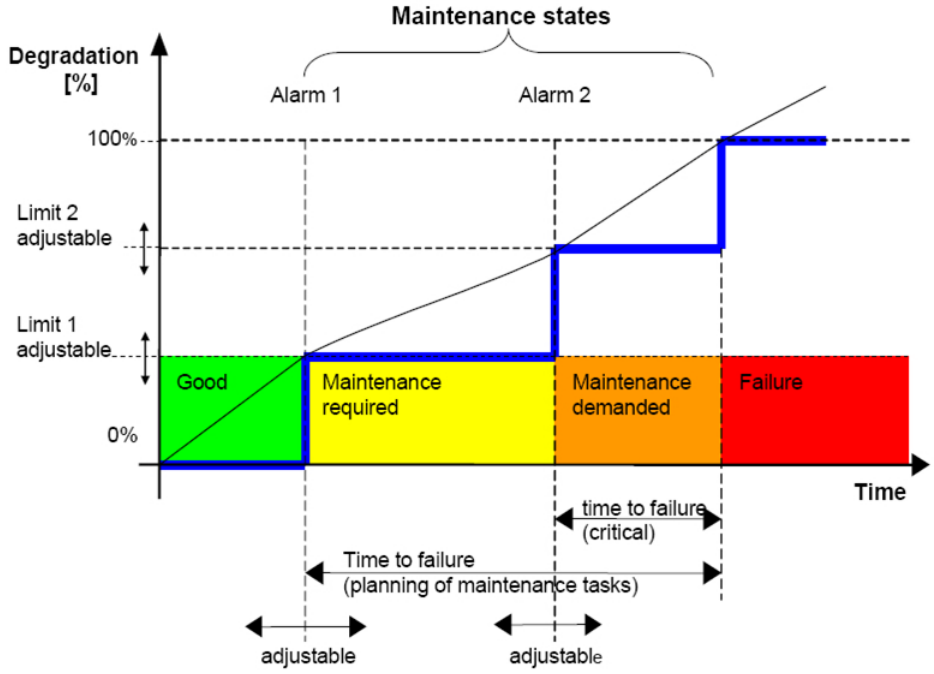


Figure 5-71 PROFINET IO maintenance state model

Maintenance state model

- Good (green): no action required
- Maintenance required (yellow): maintenance is to be scheduled
- Maintenance demanded (orange): maintenance is to be carried out
- Failure (red): fault

5.8.10.2 Device model for IO device

Description

The PROFINET IO device model stipulates that an IO device is to be divided into slots and subslots. A slot represents a module (= physical assembly) and a subslot represents a submodule (= physical subassembly).

Every submodule can be used for the following objects:

- Cyclic data (I/O interface useful data for process control)
- Alarms (e.g. diagnostic interrupt)
- Data sets

- Parameters
- Diagnostics

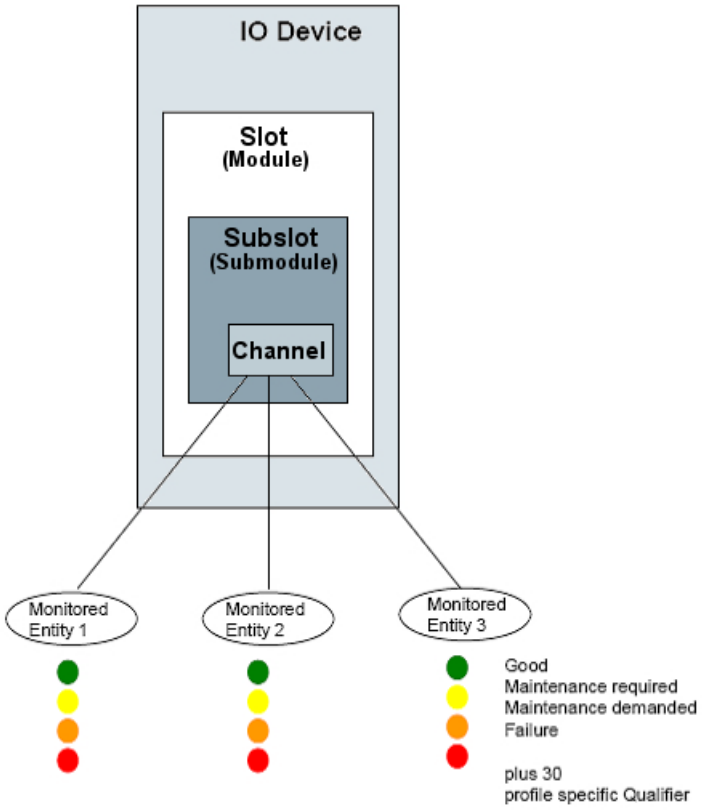


Figure 5-72 Classification of channel, diagnostic states and group information

In the PROFINET IO device model, channels are defined below submodules for the diagnostics. A channel is a logical substructure of a submodule. A submodule can contain up to 65,536 channels. Defined diagnostic functions (e.g. short-circuit, wire breakage, overtemperature) are monitored within the channels. A channel can also monitor several diagnostic functions in parallel.

This monitoring results in channel diagnostics. Several different sets of channel diagnostics can occur in a submodule at the same time. Channel diagnostics from different submodules can, of course, also occur. Channel diagnostics are signaled via a diagnostic interrupt. The diagnostic interrupt message is issued for each submodule in which channel diagnostics occur.

5.8.10.3 PROFINET IO and DS0 diagnostic interrupts

Description

If a diagnostics-related event occurs (e.g. failure or maintenance required) in an IO device, a diagnostic interrupt is generated and sent to the associated IO controller. The SIMOTION device acting as the IO controller then issues a diagnostic interrupt via the PeripheralFaultTask using TSI#interruptId = _SC_DIAGNOSTIC_INTERRUPT (= 201).

Data set 0 (DS0) for the diagnostic interrupt can be found under the TSI#InterruptId. Data set 0 (DS0) supplies the group states for the channels of a submodule.

Note

The DS0 contains only one selection of the possible diagnostics elements and is filled with all PROFINET-compliant I/O devices.

A diagnostic interrupt can occur as an incoming diagnostic interrupt with TSI#eventClass = 0x38, TSI#faultId = 0x42 and as an outgoing diagnostic interrupt with TSI#eventClass = 0x39, TSI#faultId = 0x42.

A diagnostic interrupt contains the following group states in data set 0 (DS0) in the form of the group bits below. The information relates to the sum total of all diagnostic functions of the channels within a submodule:

- Group failure: DS0.Byte0.Bit 0
- Group maintenance requirement: DS0.Byte1.Bit 7
- Group maintenance demand: DS0.Byte2.Bit7

The group bits are formed by the logical ORing of the respective individual bits of all diagnostic functions of the channels for a submodule. The three group bits are independent of each other and do not influence each other. The group bits are therefore set and/or reset independently of each other.

If just one modification is made to the maintenance states, individual maintenance requirements or maintenance demands are still pending in the submodule, and there are no faults, then an incoming diagnostic alarm is signaled and the following group bits are set accordingly in DS0: group maintenance requirement/group maintenance demand. If all maintenance states have disappeared and there is no failure, an outgoing diagnostic interrupt is signaled and the two group bits (group maintenance requirement/group maintenance demand) are set to 0. Normally, this means that the rule which applies is that an incoming diagnostic interrupt is signaled as soon as at least one of the group bits (group failure or group maintenance requirement or group maintenance demand) is set to 1. However, when all of the group bits (group failure or group maintenance requirement or group maintenance demand) are no longer set to 1, an outgoing diagnostic interrupt is signaled.

Explanation of the DS0 data set bits

Table 5-6 Table: DS0 byte 0

Bit	Description	Comment
Bit 0	Module Fault/OK	Group failure (diagnostics) for a submodule: • 0: No group failure (diagnostics) pending • 1: Group failure (diagnostics) pending
Bit 1	Internal error	Always 0
Bit 2	External error existent	Has the same content as bit 0
Bit 3	Channel error existent	Has the same content as bit 0
Bit 4	External auxiliary power missing	Always 0
Bit 5	Front connector missing	Always 0
Bit 6	Module not parameterized	Always 0
Bit 7	Wrong parameters in module	Always 0

Table 5-7 Table: DS0 byte 1

Bit	Description	Comment
Bit 0-3	Type class of module	3: Type class 3 is to be understood as a distributed I/O and as such also includes PROFINET IO.
Bit 4	Channel information existent	0: No readable channel information present 1: Readable channel information is present in the interrupt data of the diagnostic interrupt. The interrupt data can be read out using the _readDiagnosticData system function.
Bit 5	User information existent	0: No channel diagnostics or manufacturer-specific diagnostics available 1: At least one set of channel diagnostics and/or manufacturer-specific diagnostics is available
Bit 6	Diagnosis alarm from proxy	Always 0
Bit 7	Maintenance Required	Group maintenance requirement for a submodule: • 0: No group maintenance requirement pending • 1: Group maintenance requirement pending

Table 5-8 Table: DS0 byte 2

Bit	Description	Comment
Bit 0	-	Always 0
Bit 1	-	Always 0
Bit 2	-	Always 0
Bit 3	-	Always 0

Bit	Description	Comment
Bit 4	-	Always 0
Bit 5	-	Always 0
Bit 6	-	Always 0
Bit 7	Maintenance Demanded	Group maintenance demand for a submodule: • 0: No group maintenance demand pending • 1: Group maintenance demand pending

Table 5-9 Table: DS0 byte 3

Bit	Description	Comment
Bit 0	-	Always 0
Bit 1	-	Always 0
Bit 2	-	Always 0
Bit 3	-	Always 0
Bit 4	-	Always 0
Bit 5	-	Always 0
Bit 6	-	Always 0
Bit 7	-	Always 0

If detailed information on the channels of a submodule is required in the diagnostic interrupt, the interrupt data for the diagnostic interrupt must be read out and evaluated accordingly using the _readDiagnosticData system function.

Additional information

For more information, please refer to the FAQs in "SIMOTION Diagnostics Device Failure" and the SIMATIC "From PROFIBUS DP to PROFINET IO" Programming Manual.

5.9 PROFIenergy

5.9.1 Overview of PROFIenergy

Saving energy with PROFIenergy

PROFIenergy is a data interface based on PROFINET. It allows loads to be shut down during idle time in a controlled, centralized manner, and irrespective of the manufacturer and device. This means that the process is given only the energy it actually requires. As well as enabling non-operational states to be introduced, the interface also makes it possible to request consumption values and PROFIenergy statuses from the device. The majority of the energy is saved by the process itself; the PROFINET device only makes a small contribution to the saving potential.

Basic principles and functionality

The PROFINET devices are shut down using special commands in the user program of the PROFINET IO controller. No additional hardware is required; the PROFIenergy commands are interpreted directly by the PROFINET devices.

At the start and end of pauses, the plant operator activates or deactivates the pause function of the plant, after which the IO controller sends the PROFIenergy "START_Pause"/"END_Pause" command to the PROFINET devices. The device then interprets the content of the PROFIenergy command and switches to PROFIenergy mode PAUSE or back to READY TO RUN mode. Further PROFIenergy functions can be used to fetch device information.

PROFIenergy profile

Further information and the basic principles of the PROFIenergy profile can be found in the *Common Application Profile PROFIenergy; Technical Specification for PROFINET; Version 1.1; January 2012; Order No: 3.802*. This profile lists all possible measured values in table form, among other things.

5.9.2 Example of PROFIenergy with SIMOTION

PROFIenergy cell concept

As far as PROFIenergy is concerned, production plants are generally made up of independent cells. Each cell has a "head" controller, e.g. SIMOTION, which has a communication relationship with the higher-level controller. The higher-level energy management system (e.g. SIMATIC controller) communicates with the cell with the help of the PROFINET I-device functionality. In the example, cell 1 contains a SIMOTION controller which serves as a PROFIenergy controller for the lower-level devices (ET200, G120, and S120). At this point, SIMOTION is a PROFINET device for the higher-level, centralized energy management, but also a controller and PROFIenergy controller for its lower-level PN devices.

There may be several cells within a production plant. The higher-level controller records the state of the lower-level cells and shuts them down during idle time in a coordinated manner. A production plant can also consist of n cells.

In this example, the higher-level controller sends the PROFIenergy commands via the I-device interface to the SIMOTION controller. The SIMOTION controller evaluates these commands in the user program, responds with an appropriate PROFIenergy data set, and activates the relevant sensors and actuators. Please note that the interpretation and answering of the PE data sets must take place in the SIMOTION user program. A PROFIenergy library is available for SIMOTION for this purpose. This is described in more detail in the following chapters.

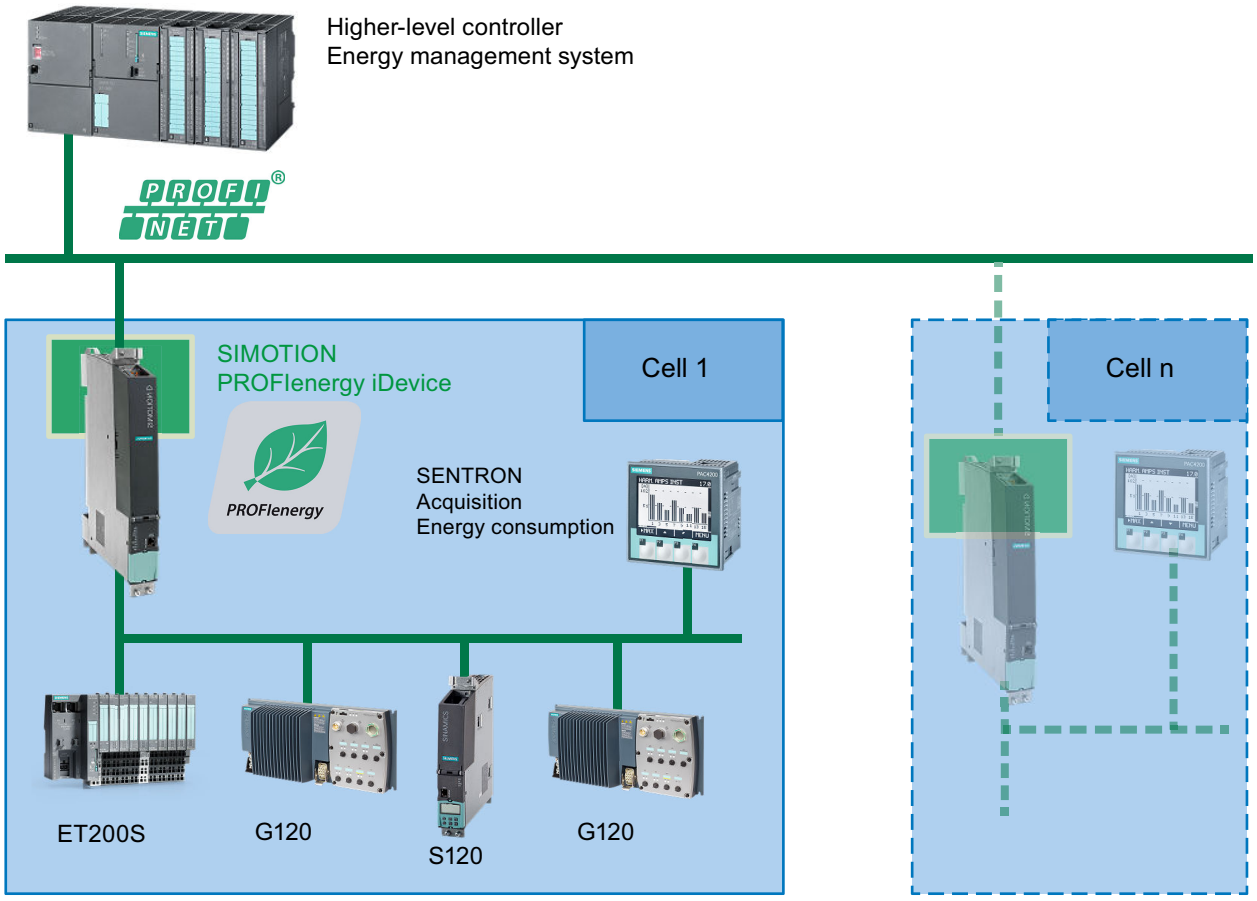


Figure 5-73 Schematic representation of the PROFIenergy cell concept

5.9.3 SIMOTION as PROFIenergy controller and PROFIenergy device

Basic principles of SIMOTION as PROFIenergy controller/device

SIMOTION controllers can control PROFIenergy states and read measured values when used with PROFIenergy-compatible devices (PROFINET devices, e.g. ET200S, S120, G120). SIMOTION controllers can also be operated via the I-device interface as PROFIenergy devices, i.e. the SIMOTION controller can receive PROFIenergy data sets from the higher-level energy management system and respond to them (e.g. by supplying PROFIenergy measured values).

PROFIenergy telegrams are non-cyclic services and are further processed in data sets. SIMOTION uses the I-device interface to provide the necessary non-cyclic services for the use of the PROFIenergy telegrams. In addition to the cyclic data, the higher-level PROFINET controller (e.g. S7 CPU) can also send non-cyclic data to SIMOTION in the form of data sets via the I-device interface. The data set content from the non-cyclic I-device communication is made available for the SIMOTION applications. This data set content can be read by a user program as a receive buffer or written as a send buffer for the response. The processing and evaluation must take place in the user program.

Detailed description

As of SIMOTION V4.4, data sets can be received and sent via the I-device interface, which means that PROFIenergy data sets (commands) can be too. The two new SIMOTION system function blocks _receiveRecord() and _provideRecord() have been made available for this purpose. Data sets can be received by and sent to the higher-level IO controller by calling these system function blocks in the user program.

The interfaces, functionality, and handling of the new SIMOTION system functions are based on the existing SIMOTION system functions for reading/writing data sets, e.g. _readVariableRecord(), and on the corresponding system function blocks for the SIMATIC SFB73 and SFB74.

The system function blocks _receiveRecord() and _provideRecord() are used for non-cyclic communication between two IO controllers, whereby one of the two is functioning as an I-device. This means that non-cyclic data sets can be sent and received from the user program.

A short explanation of the function blocks can be found in the next chapter.

Supported PROFIenergy functions and hardware

PROFIenergy is available as of SIMOTION V4.2 and SINAMICS V4.5.

SINAMICS S120 and SINAMICS G120 support the following PROFIenergy functionalities:

- PAUSE command
- Reading out two or three measured values
  - Active power P (MeasurementId = 34 in W)
  - Power factor cosφ (MeasurementId = 166, for G120 only)
  - Energy absorbed (MeasurementId = 200 in Wh)
- SINAMICS G120D-2 can also shut down encoders and I/Os.

5.9.4 System function blocks _receiveRecord() and _provideRecord() for non-cyclic communication via the iDevice interface

Introduction

A SIMOTION controller uses these system function blocks to send and receive data sets via the iDevice interface. The user program must respond to a non-cyclic request. These system function blocks are used, among other things, to transfer PROFIenergy data sets.

Note

A detailed description of the system functions _provideRecord() and _receiveRecord() can be found in the online help and in the *SIMOTION System Functions/Variables Devices List Manual*.

_provideRecord()

The system function block _provideRecord() includes the following functionality:

- It checks cyclically (when mode=0) whether the iDevice has been sent a request to make a data set available.
- It makes the record data available to the higher-level controller.

_receiveRecord()

The system function block _receiveRecord() includes the following functionality:

- It checks cyclically whether the iDevice has been sent a request to receive a data set.
- It makes the data set available on the output parameters.
- It acknowledges receipt of the data by the higher-level controller.

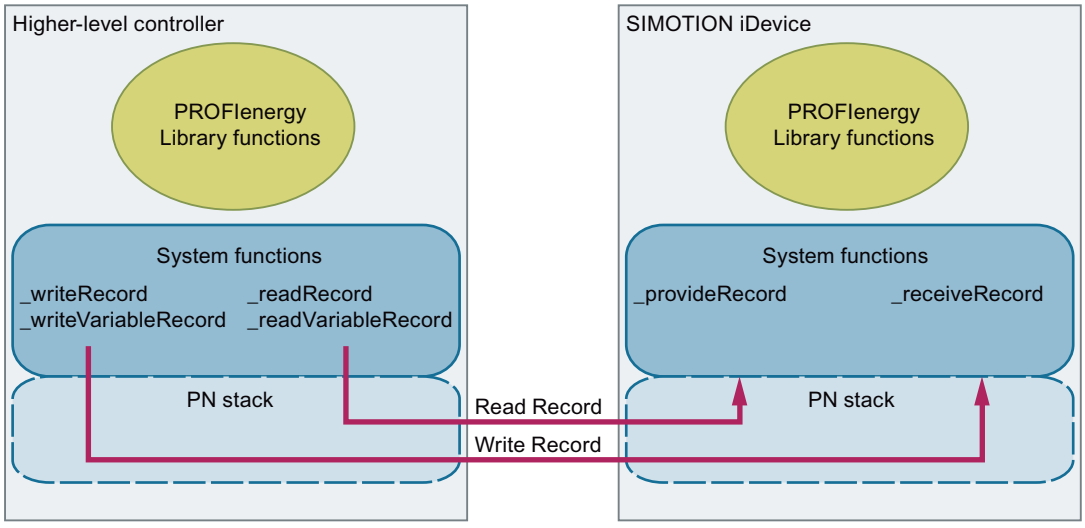


Figure 5-74 Read record and write record as iDevice

5.9.5 Function block for the SIMOTION PROFIenergy iDevice

PROFIenergy iDevice function block

A function block with an example application is available for decoding and coding the PROFIenergy data sets. This function block supports the following modes of operation:

- Forwarding the PROFIenergy commands from the higher-level controller to the lower-level PROFIenergy-compatible devices (e.g. ET200 or SINAMICS). Standard PROFIenergy function blocks that have already been implemented can be used in the user program (e.g. to read out the measured values).
- The PROFIenergy commands can be processed within the function blocks for a complete (production) cell; i.e. the PROFIenergy iDevice FB reflects the complete PROFIenergy functionality of a cell. As a prerequisite, there must be a SIMOTION CPU as the "head" controller. The shut-down and switch-on sequence for the individual PROFIenergy IO devices can be parameterized on the iDevice FB.

Download

The function block for the PROFIenergy iDevice with example application and comprehensive documentation can be found in the Support Portal (http://support.automation.siemens.com/WW/view/en/58386840).

Ethernet: General information (TCP and UDP connections)

6

6.1 Ethernet interfaces

6.1.1 Overview of Ethernet

Overview

Below is a description of how to configure TCP and UDP Ethernet connections between communications partners. TCP and UDP are based on Ethernet and the IP protocol.

6.1.2 Properties of the SIMOTION Ethernet interfaces

Ethernet interfaces

Depending on the device, SIMOTION has one or three onboard Ethernet interfaces, and one or two PROFINET IO interfaces. You can connect an Industrial Ethernet with a transmission rate of 10/100 Mbps or 1,000 Mbps to the RJ45 sockets with D4x5-2 DP/PN.

With several interfaces, TCP/IP timeout parameters can be set once for both interfaces. With several interfaces, the transmission rate/duplex can be set separately for the two interfaces. Utilities via TCP/IP and UDP are supported for both Ethernet interfaces, enabling S7 routing between all interfaces (including PROFIBUS DP). "Utilities via TCP" are not routed from one Ethernet interface to the other.

Alternatively, you can also connect an Industrial Ethernet through the onboard PROFINET interface of a SIMOTION D4x5-2 DP/PN (100 Mbit/s).

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

Note

Default gateway (network transition/router)

In a SIMOTION controller, only one of the existing Ethernet or PROFINET interfaces (onboard PN interfaces, CBE30-2…) can be configured as a default gateway (network transition/router). On the other Ethernet or PROFINET interfaces, the IP address and the subnet mask can be set. If more than one default gateway is configured on the SIMOTION controller, this will be output as an error message on the consistency check as of Step 7 V5.5 SP4. In older Step 7 versions, this check is is only made on the project download.

Converting standard Ethernet interfaces from V4.3 and higher

With SIMOTION V4.3, the standard Ethernet interfaces are marked as PN-IE interfaces:

- Determining neighborhood information via LLDP (Link Layer Discovery Protocol). To do so, use the system function _getPnPortNeighbour in SIMOTION.
- Via the Step7 diagnostic address, you can perform diagnostics from the user program for the Ethernet interface.
- You can assign the device name (NameOfStation) either via DCP or via the user program (see also IP address and device name (Page 149)).
- You can assign the IP address either via DCP or via the user program.
- Provision of topology information via SNMP (Simple Network Management Protocol) (e.g. via the STEP7 Topology Editor, see also Topology editor (graphical view) (Page 139)).
- You can configure a setpoint topology.
- Online diagnostics of the interface and ports via STEP7 device diagnostics.

The interfaces in HW Config are displayed as follows:

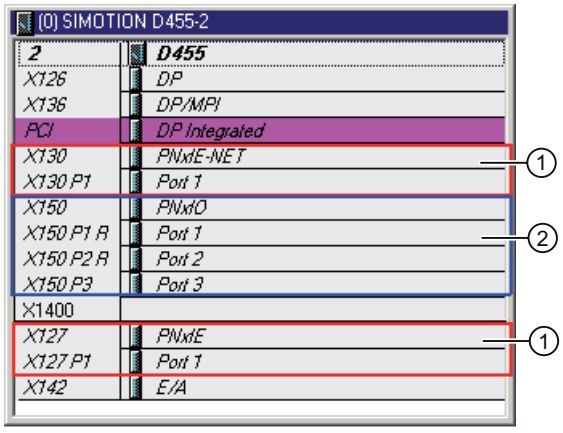


Figure 6-1 Ethernet interfaces in HW Config

①	Standard Ethernet interface with one port
②	PROFINET IO interface with three ports

See also

IP address and device name (Page 149)

Topology editor (graphical view) (Page 139)

6.1.3 Using the Ethernet interface

Using the Ethernet interface

- 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 PCs
- For connecting SIMATIC HMI devices such as MP277, MP370, or PC-based HMIs
- For communication by means of SIMOTION IT Web server and SIMOTION IT OPC XML DA (no license required as of V4.2)
- For communication by means of SIMOTION VM (separate license required)

6.2 LCom communications library

LCom library

The **LCom** library is based on TCP and simplifies the use of both SIMOTION system functions and SIMATIC communication blocks in order to establish communication between multiple machines.

Note

You can find the LCom library on the SIMOTION SCOUT DVD "Documentation, Utilities & Applications". The DVD also contains an example project and comprehensive documentation for the library.

Types of control

The LCom library supports the following control types and combinations:

- SIMOTION ↔ SIMOTION
- SIMATIC ↔ SIMATIC
- SIMOTION ↔ SIMATIC

Functions

- The send and receive data must be of data type BYTE. Outside of the *FBLComMachineCom*, any user structures can be converted into an ARRAY OF BYTES via marshalling.
- Bi-directional operation
  - A logical point-to-point connection is established between two controls.
  - Each control can send and receive via a connection at the same time.
- Alignment of configuration settings between communications partners (e.g. send clock)
  - Assignment of communication parameters to the communications partner
  - Changing the configuration during operation
- Types of data transmission
  - Cyclic transmission (transmission at fixed periods of time)
  - Transmission when data is changed
  - One-off transmission
- Sending and receiving max. 64 kB useful data
- Acknowledgment and monitoring of received data
- Sign-of-life monitoring
- Time-of-day synchronization

6.3 TCP communication

6.3.1 Overview of TCP communication

Communication with TCP (Transfer Control Protocol)

Communication via TCP is connection-oriented, i.e. data can only be transferred once a connection to the communications partner has been established.

The main features of a communication connection are two end points. An end point is a controlled pair consisting of an IP address and a port. The port on the client may be the same as or different from the port on the server. Usually, one end point represents a server and the other end point a client.

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

Principle of TCP communication (see figure)

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

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

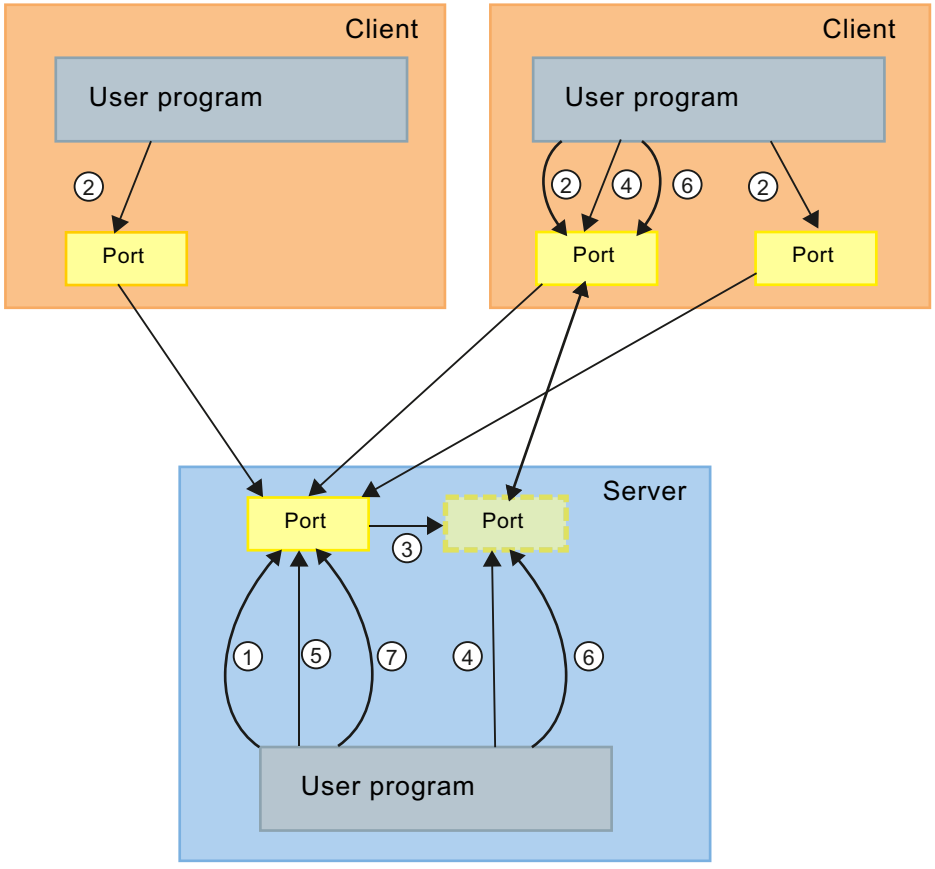


Figure 6-2 Schematic diagram showing TCP communication sequence

Principles of port assignment:

- The port number on the client can be the same as the port on the server.
- The port number on the client can be the different to the port on the server.

Data exchange between user program and TCP stack

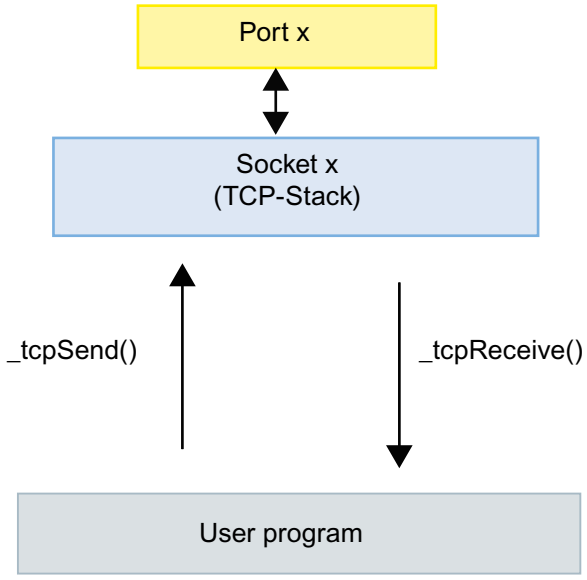


Figure 6-3 Data exchange between user program and TCP stack

Data packets

With TCP communication, the useful data received is buffered in the TCP stack and must be read out from here by the application. The size of the buffered useful data per connection depends on the control involved.

Table 6-1 Useful data and control

Control	Buffer size in bytes
SIMOTION	4096
SIMATIC	8192

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

6.3.2 SIMOTION system functions for TCP communication

6.3.2.1 Overview of SIMOTION system functions

Maximum number of possible TCP connections

The table below contains examples of the number of possible communication connections for a SIMOTION CPU acting as a client. The values relate to a local network without any other external load sources, and a SIMOTION D435 acting as a server.

Table 6-2 Communication connections in relation to a SIMOTION CPU

SIMOTION CPU (client)	Number of communication connections
C2xx	45
D410	45
D410--2	45
D4x5	75
D4x5-2	75
P3x0	40

Function call

The SIMOTION system functions for TCP communication may only be called in the BackgroundTask or in a MotionTask. In the BackgroundTask, it should be noted that the system functions are executed asynchronously (nextCommand=IMMEDIATELY). In the MotionTask, however, the system functions can be executed synchronously (nextCommand=WHEN_COMMAND_DONE).

Note

To ensure reliable cyclic communication, you should use standard mechanisms such as those found in PROFIBUS DP or PROFINET IRT systems.

See also

_tcpOpenServer() system function (Page 227)

_tcpOpenClient() system function (Page 228)

_tcpSend() system function (Page 229)

_tcpReceive() system function (Page 229)

_tcpCloseConnection() system function (Page 230)

_tcpCloseServer() system function (Page 231)

6.3.2.2 _tcpOpenServer() system function

Overview

This system function is used in order to initiate passive establishment of a connection.

Table 6-3 Call example

```
sRetValueTcpOpenServer := _tcpOpenServer(  //StructRetTcpOpenServer
   port        := 3456                     //UINT, 1024-65535
  ,backLog     := 5                        //DINT
  ,nextCommand := IMMEDIATELY              //EnumTcpNextCommandMode
);
```

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 establishing the connection can be queried via the **functionResult** parameter.

The **connectionId** parameter is used as an (input) parameter for the call of the _tcpSend() and _tcpReceive() functions and assigns a unique TCP connection to these functions. This return value is referred to in the above call examples.

The **clientAddress** parameter returns the client IP address from which the connection is activated; this takes the form of an array.

The port number designated as the local port number of the client is specified in the **clientPort** parameter.

The port number is in the range 1024 to 65535.

6.3.2.3 _tcpOpenClient() system function

Overview

The _tcpOpenClient() system function is used to actively establish a connection.

Table 6-4 Call example

```
sRetValueTcpOpenClient :=_TCPOpenClient( // StructRetTcpOpenClient
   port           := 3456               // UINT, 1024-65535
  ,serverAddress  := au8ServerAddress   // ARRAY [0...3] OF USINT
  ,serverPort     := 3456               // UINT, 1024-65535
  ,nextCommand    := IMMEDIATELY        // EnumTcpNextCommandMode
  );
```

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 communications partner, 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 port number is in the range 1024 to 65535.

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

When the _tcpOpenClient() function is called, the structure returned to the user program contains the following parameters.

The status of establishing the connection can be queried via the **functionResult** parameter.

The **connectionId** parameter is used as an (input) parameter for the call of the _tcpSend(), _tcpReceive(), and _tcpCloseConnection() functions and assigns a unique TCP connection to these functions.

6.3.2.4 _tcpSend() system function

Overview

The _tcpSend() system function is used for sending data.

Table 6-5 Call example

```
i32RetValue := _tcpSend(                                    // DINT
   connectionId    := sRetValueTcpOpenClient.ConnectionId  // DINT
  ,nextCommand     := IMMEDIATELY                          // EnumTcpNextCommandMode
  ,dataLength      := 4096                                 // UINT, 0-4096
  ,data            := ab8SendData                          // ARRAY [0..4095] OF BYTE
 );
```

For the **connectionId** parameter, the **connectionId** return value of the _tcpOpenClient() or _tcpOpenServer() function is transferred, depending on whether the communication node that executes the function is a server or client.

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.

The **dataLength** parameter informs the function of the useful data length in bytes that has to be transferred (max. 4,096 bytes per function call).

The **data** parameter specifies the useful data area in which the send data that is to be transferred with the function is located.

The return value of the function to the user program is of data type DINT. The various return values indicate any problems that occurred during execution of the function. There is also a confirmation when the data has been successfully sent. Negative values in **functionResult** indicate a data transmission error. In this case, the connection must be closed by calling _tcpCloseConnection().

6.3.2.5 _tcpReceive() system function

Overview

The _tcpReceive() system function is used for receiving data.

Table 6-6 Call example

```
sRetValueTcpReceive := _tcpReceive(                          // StructRetTcpReceive
   connectionId     := sRetValueTcpOpenClient.connectionId  // DINT
  ,nextCommand      := IMMEDIATELY                          // EnumNextCommandMode
  ,receiveVariable  := ab8ReceiveData                       // ARRAY [0..4095] OF BYTE
  );
```

For the **connectionId** parameter, the **connectionId** return value of the _tcpOpenClient() or _tcpOpenServer() function is transferred, depending on whether the communication node that executes the function is a server or client.

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 immediate and with the second value it is after completion of the command.

For each function call, up to 4,096 bytes of receive data can be read out from the TCP stack. Please note that it will not be possible to predict the sizes of the packets received. On the receiver side, you must take care to ensure that all useful data is present before the evaluation and further processing. For this purpose, the **nextCommand** parameter should be set to IMMEDIATELY.

The **receiveVariable** parameter is used to inform the function of the useful data area in which the receive data is to be stored. The received data is available in the **receiveVariable** parameter in the length **dataLength**, if the return value in the functionresult = 16#0. At the next function call, the **ReceiveVariable** parameter is overwritten with new data in the length **dataLength**.

When the _tcpReceive() function is called, the structure returned to the user program contains the following parameters. The receive status can be queried via the **functionResult** parameter. The **dataLength** parameter signals the number of received useful data bytes once the _tcpReceive() function has been successfully called.

Negative values in **functionResult** indicate a data transmission error. In this case, the connection must be closed by calling _tcpCloseConnection().

6.3.2.6 _tcpCloseConnection() system function

Overview

The _tcpCloseConnection() function is used for closing a connection that has been actively established by the communication node executing the function (client).

Table 6-7 Call example

```
sRetValueTcpCloseConnection := _tcpCloseConnection(      // DINT
   connectionId := sRetValueTcpOpenClient.ConnectionId   // DINT
  );
```

The return value of the _tcpOpenClient() function is transferred at the **connectionId** parameter in order to clearly identify which connection is to be closed.

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

6.3.2.7 _tcpCloseServer() system function

Overview

The _tcpCloseServer() function is used for closing a connection that has been passively established by the communication node executing the function (server).

Table 6-8 Call example

```
sRetValueTcpCloseServer := _tcpCloseServer(port := 3456); //1024 - 65535
```

The server port is transferred at the **port** parameter.

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

6.4 UDP communication

6.4.1 Overview of UDP communication

Communication with UDP (User Datagram Protocol)

UDP offers a method of sending and receiving data over Ethernet from the user program with a minimum use of protocol mechanisms. No information concerning the transferred data is returned in the case of communication via UDP. Communication takes place via ports on both the send and receive sides. Unlike TCP, you do not need to program any connection establishment or closing.

Principle of UDP communication

- 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 and port number of the target system, as well as the port number of the local control.
- 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 transmission if it is not read from the buffer in good time. You must program a secured method of data transmission via your application, e.g. by acknowledging receipt of the data.
- At the very least, the following data is required for sending:
  - IP address of communications partner
  - "Own" port number
  - Port number of communications partner
- UDP is not a secured transfer protocol. You must program feedback concerning the success of data transmission in the user program yourself.

Data exchange between user program and UDP stack

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

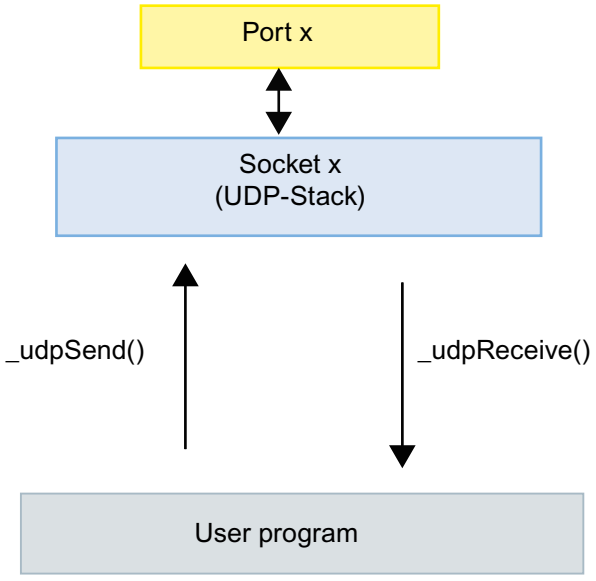


Figure 6-4 Data exchange between user program and UDP stack

With UDP communication, the useful data received is buffered in the UDP stack and must be read out from here by the application. The size of the buffered useful data per connection depends on the control involved. If the receive buffer is not read out on time, the received UDP telegrams will be lost once a new UDP telegram is received.

Table 6-9 Useful data and control

Control	Buffered useful data in bytes
SIMOTION	1,470 (SIMOTION V4.1 SP4 and higher) 1,400 (up to SIMOTION V4.1 SP3)
SIMATIC	Up to 2,048 (depending on CPU and communication via Ethernet CP or integrated Ethernet interface)

6.4.2 SIMOTION system functions for UDP communication

6.4.2.1 Overview of SIMOTION system functions

Function call

The SIMOTION UDP system functions may only be called in the BackgroundTask or in a MotionTask.

Data is sent via _udpSend(). If data is to be received on the SIMOTION side, the _udpReceive() function is used. The sections that follow describe the functions.

Note

To ensure reliable cyclic communication, you should use standard mechanisms such as those found in PROFIBUS DP or PROFINET IRT systems.

See also

_udpSend() system function (Page 234)

_udpReceive() system function (Page 235)

_udpAddMulticastGroupMembership() system function (Page 236)

_udpDropMulticastGroupMembership() system function (Page 236)

6.4.2.2 _udpSend() system function

Overview

The _udpSend() system function is used for sending data.

Table 6-10 Call example

```
i32RetValue :=_udpSend(
   sourcePort         := 3456                   //UINT, 1024 - 65535
  ,destinationAddress := au8DestinationAddress  //ARRAY[0..3] OF USINT
  ,destinationPort    := u16DestinationPort     //UINT, 1024 - 65535
  ,communicationMode  := CLOSE_ON_EXIT          //EnumUdpCommunicationMode
  ,dataLength         := u32DataLength          //UDINT
  ,data               := ab8SendData                //ARRAY[0..1469] OF BYTE
  );
```

When the _udpSend() function is called, the local port is transferred for the **sourcePort** parameter. The **destinationAddress** parameter is the IP address which is transferred in an array. The IP address can be configured and read out in HW Config.

The port of the communications partner is transferred as the **destinationPort**.

The user can use **communicationMode** to specify whether the communication resources are to be released after sending (CLOSE_ON_EXIT) or not released after sending (DO_NOT_CLOSE_ON_EXIT).

The **datalength** and **data** parameters specify the data length to be sent or the area where the send data is stored.

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

Note

Up to SIMOTION 4.1 SP4, the length of the send data (data) was limited to 1,400 bytes.

6.4.2.3 _udpReceive() system function

Overview

The _udpReceive() system function is called for receiving data.

Table 6-11 Call example

```
sRetValueUdpReceive :=_udpReceive(
   port               := 3456                    //UINT, 1024 - 65535
  ,communicationMode  := CLOSE_ON_EXIT           //EnumUdpCommunicationMode
  ,nextCommand        := IMMEDIATELY             //EnumNextCommandMode
  ,receiveVariable    := ab8ReceiveData                 //ARRAY[0..1469] OF BYTE
  );
```

When the function is called, the local port is transferred for the **port** parameter.

In this case too, the user can use **communicationMode** to specify whether the communication resources are to be released after reception (CLOSE_ON_EXIT) or not released after reception (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 immediate 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() system 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. The **sourcePort** parameter of the structure also contains the local port.

The number of received useful data bytes after a successful call of the _udpReceive() system function can be read out in the **dataLength** parameter.

Note

Up to SIMOTION 4.1 SP4, the length of the receive data (receiveVariable) was limited to 1,400 bytes.

6.4.2.4 _udpAddMulticastGroupMembership() system function

Overview

This function is used to join a multicast group on a selected Ethernet interface.

A maximum of three multicast groups can be created on an Ethernet interface. These can occupy the same port or different ports.

```
myRetDINT :=
  _udpAddMulticastGroupMembership(
      multicastIPAddress :=
      ,interfaceIPAddress :=
      ,multicastPort :=
      // ,multicastTTL := 1
      // ,multicastLOOP := 1
      ,nextCommand :=
);
```

Note

In the case of UDP multicast communication, the _multicastPort must correspond to the sourcePort for function _udpSend().

Note

The UDP multicast functions (_updAddMulticastGroupMembership(), updDropMulticastGroupMembership()) can only be used on Ethernet interfaces. PROFINET interfaces are not supported.

6.4.2.5 _udpDropMulticastGroupMembership() system function

Overview

This system function is used to exit a multicast group on a selected Ethernet interface.

```
myRetDINT :=
  _udpDropMulticastGroupMembership(
      multicastIPAddress :=
      ,interfaceIPAddress :=
      ,multicastPort :=
      ,nextCommand :=
);
```

Note

The UDP multicast functions (_updAddMulticastGroupMembership(), updDropMulticastGroupMembership()) can only be used on Ethernet interfaces. PROFINET interfaces are not supported.

6.5 Services used

6.5.1 Communication services and port numbers used

Introduction

SIMOTION supports the protocols listed in the following table. The address parameters, the relevant communication layer as well as the communication role and the communication direction are specified for each protocol. This information allows you to match the security measures for the protection of the automation system to the protocols used.

Documentation of Ethernet services used

Protocol	Port number	Transport level	Role/direction	Function	Description
PROFINET protocols					
DCP Discovery and Configuration Protocol	Not relevant	Ethernet II and IEEE 802.1Q and Ethertype 0x8892 (PROFINET)	Server/incoming Client/outgoing	Accessible nodes, PROFINET discovery and configuration	DCP is used by PROFINET to determine PROFINET devices and to make basic settings.
LLDP Link Layer Discovery Protocol	Not relevant	Ethernet II and IEEE 802.1Q and Ethertype 0x88CC (PROFINET)	Server/incoming Client/outgoing	PROFINET Link Layer Discovery Protocol	LLDP is used by PROFINET to determine and manage neighborhood relationships between PROFINET devices.
MRP Media Redundancy Protocol	Not relevant	Ethernet II and IEEE 802.1Q and Ethertype 0x88E3 (PROFINET)	Server/incoming Client/outgoing	PROFINET media redundancy	MRP enables the control of redundant routes in a ring topology.
PTCP Precision Transparent Clock Protocol	Not relevant	Ethernet II and IEEE 802.1Q and Ethertype 0x8892 (PROFINET)	Server/incoming Client/outgoing	PROFINET send clock and time synchronization, based on IEEE 1588	PTCP enables a time delay measurement between RJ45 ports and, therefore, enables send clock synchronization and time synchronization.

PROFINET IO data	Not relevant	Ethernet II and IEEE 802.1Q and Ethertype 0x8892 (PROFINET)	Server/incoming Client/outgoing	PROFINET cyclic IO data transfer	The PROFINET IO telegrams are used to transfer IO data cyclically between the PROFINET IO controller and IO devices via Ethernet.
PROFINET Context Manager	34964	UDP	IO device/incoming IO Client/ outgoing	PROFINET connectionless RPC	The PROFINET Context Manager provides an endpoint mapper in order to establish an application relationship.
Connection-oriented communication protocols					
FTP	21	TCP	Server/incoming	File Transfer Protocol	FTP is used to transfer files to the flash card. Password protection via Web server user management. The port can be deactivated.
Telnet	23	TCP	Server/incoming		Used for internal debugging. Password protection via Web server user management. The port can be deactivated.
HTTP Hypertext Transfer Protocol	80	TCP	Server/incoming	Hypertext Transfer Protocol	HTTP is used for communication with the internal SIMOTION Web server. Password protection via Web server user management. The port can be deactivated. You can modify the port number if desired.
RFC1006	102	TCP	Server/incoming Client/outgoing	ISO-on-TCP protocol	The protocol extension RFC1006 is used for communication with ES, HMI, OPC server, etc.
SNMP Simple Network Management Protocol	161	UDP	Server/incoming	Simple Network Management Protocol	SNMP enables the reading out of network management data (MIBs) by SNMP clients.

HTTPS Secure Hypertext Transfer Protocol	443	TCP	Server/incoming	Secure Hypertext Transfer Protocol	HTTPS is used to communicate with the internal CU Web server via Secure Socket Layer (SSL). Password protection via Web server user management. The port can be deactivated. You can modify the port number if desired.
Internal protocol	3844	TCP	Server/incoming	System diagnostics	Diagnostic access for traces within the system.
Internal protocol	5188	TCP	Server/incoming	Project download	Communication with SCOUT for downloading project data.
Internal protocol	8412	TCP	Server/incoming Client/outgoing	Trace	Port is used for trace functionality that must be synchronized over several devices. Cannot be switched off.
Internal protocol	44550	UDP	Server/incoming Client/outgoing	Trace	Port is used for trace functionality that must be synchronized over several devices. Cannot be switched off.

Documentation of Ethernet services used

6.5.2 Deactivating PN interface ports in SIMOTION

Introduction

On SIMOTION CPUs as of Version 4.4, PN interface ports can be set to **Disable** in the engineering system. This prevents devices being connected without permission and also increases security with regard to accessing the system. The engineering system and the PN stack ensure that at least one port on each interface is not set to **Disable** to prevent the user locking him/herself out. The default setting is **Automatic settings**.

How to deactivate ports

1. In HW Config, double-click in the station on the port that you want to deactivate.
2. In the dialog box that opens, switch to the **Options** tab.
3. Under **Transmission media/Duplex**, select the item **disable**. The port is deactivated.

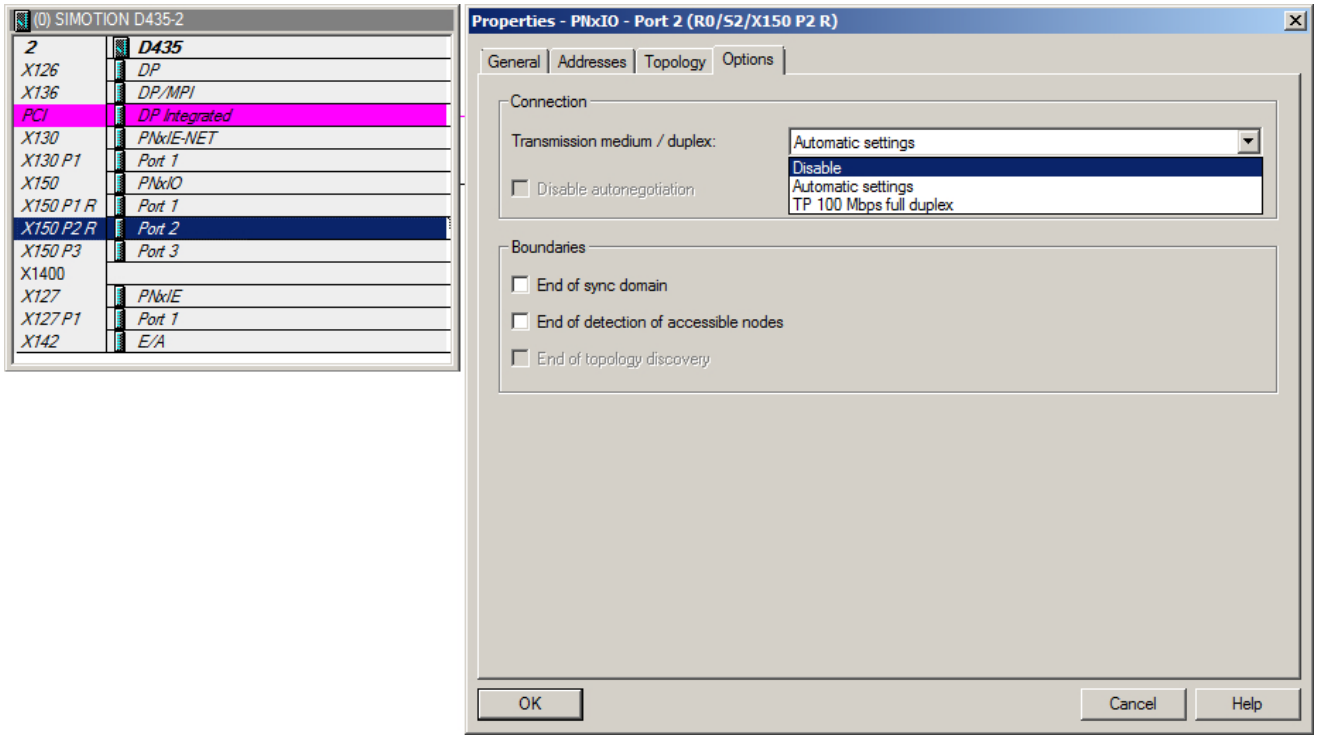


Figure 6-5 Deactivating PN ports on SIMOTION CPUs

Routing - communication across network boundaries 7

7.1 What does routing mean?

Introduction

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.

Note

IP routing is NOT supported by SIMOTION. Only S7 routing is possible between the interfaces of a SIMOTION controller.

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 the so-called 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 (SIMOTION V4.1.2 or higher, including PROFINET; CP343, CPU 315-2 PN/DP...)

CIDR Classless Inter-Domain Routing

Classless Inter-Domain Routing (CIDR) describes a process for a more efficient use of the existing 32-bit IP address space (IPv4). It was introduced to reduce the size of routing tables and utilize the available address areas more effectively. The CIDR (Classless Inter-Domain Routing) function includes subnetting and supernetting.

Subnetting refers to the splitting of an IP subnet into multiple subnetworks (by enlarging the subnet mask).

Supernetting refers to the compilation of multiple subnets into a joint IP subnet (by minimizing the subnet mask).

See also Relationships between subnet masks and IP addresses in relation to subnetting and supernetting (Classless Inter Domain Routing CIDR) (http://support.automation.siemens.com/WW/view/en/2073614).

Note

CIDR is supported by all SIMOTION devices. In the case of SIMATIC devices, please refer to the notes in the manuals.

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).

Note

You will find further notes on Ethernet/PROFINET and the settings required for routing in the *SIMOTION SCOUT Overview of Service and Diagnostics Options* product information as well as in the online help on this topic.

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.

Establishing an online connection to the nodes in the project

Through S7 routing you can go to all the nodes in the project online.

This is how you establish an online connection from your PG/PC to the nodes in the project.

1. Insert a PG/PC station in NetPro.
2. Add a new interface.
3. Assign the new interface of the network card used to the PG/PC.
4. Make the assignment "active" (S7ONLINE access).
5. Compile NetPro and load the data into the routing devices by downloading it.

7.3 Routing for SIMOTION

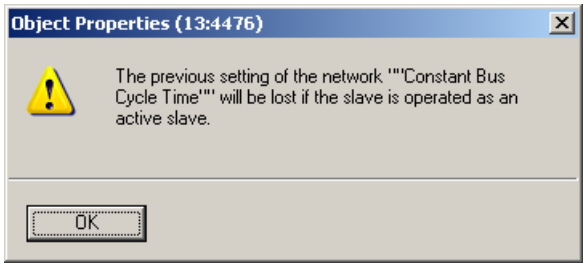
Definition

With routing you can, for example, access devices connected to subnets ONLINE via a PG/PC. S7 routing is supported by SIMOTION, i.e. information (engineering accesses) can be routed by a SIMOTION device from higher-level networks such as Ethernet and MPI to lower-level networks such as PROFIBUS or PROFINET/Ethernet (4.1.2 and higher).

Boundary conditions

The following boundary conditions must be taken into account in the "DP slave" mode when routing information on an isochronously operated PROFIBUS.

- The functions "Equidistant bus cycle" (requirement for isochronous applications) and "Active station" (requirement for routing to a lower-level network segment) mutually exclude each other.
- It is not possible to operate an active I slave on the isochronous bus.



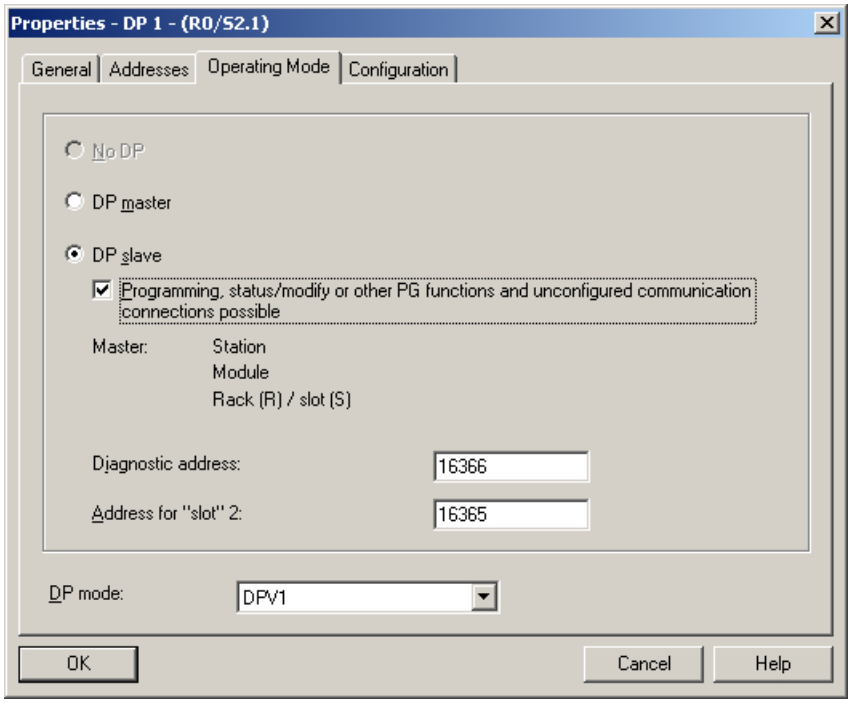


Figure 7-1 DP slave mode: Active station: Testing, commissioning, routing

The "Programming, status/control or other PG functions …" check box 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 routing) SINAMICS drives on the cascaded, lower-level DP master interface of the SIMOTION controller with PG functions (e.g. STARTER).

If the "Programming, Status/Force 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/Force or other PG functions …" check box is not activated, the interface only operates as a server for cyclic data, i.e. S7 routing is not possible.

Note

If more than one gateway is used for the PG connection, all configured connection paths must also be physically connected. However, it is not possible to configure which of the configured connection paths is actually being used.

7.4 Routing with SIMOTION D (example of D4x5 with CBE30)

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.

- S7 routing is possible to every interface.
  E. g. you can set up a connection from a PROFINET/Ethernet subnet to a PROFIBUS via S7 routing.
- IP routing from subnet to subnet is not supported. You can use an external IP router for this.

There are three options for connecting a PG/PC or HMI via S7 routing to a SIMOTION D with CBE30.

Note

The Ethernet interfaces X120 and X130 and the CBE30 must be in different subnets for S7 routing.

Engineering system to PROFINET (CBE30)

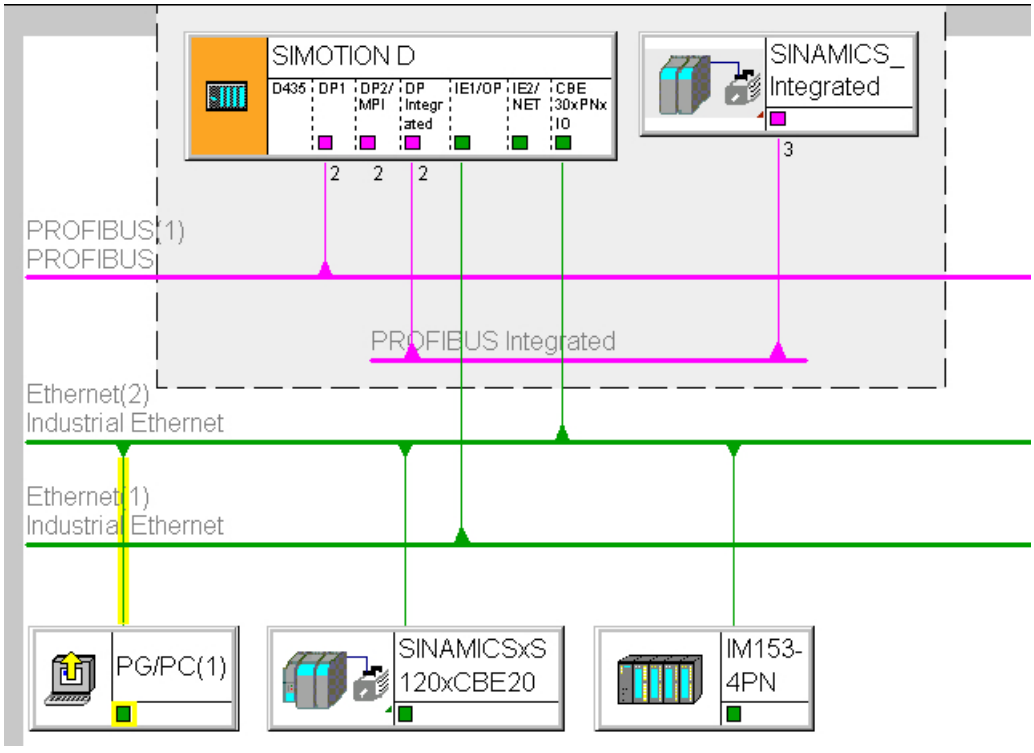


Figure 7-2 Example for PG/PC to CBE30

- S7 routing to the (master) PROFIBUS interfaces (only if configured)
- S7 routing to PROFIBUS Integrated
- S7 routing to the standard Ethernet interfaces ET1/ET2 (X120, X130) (V4.1.2 and higher)
- Access to the components on the same subnet (CBE30) via the switch functionality

Engineering system / HMI to PROFIBUS

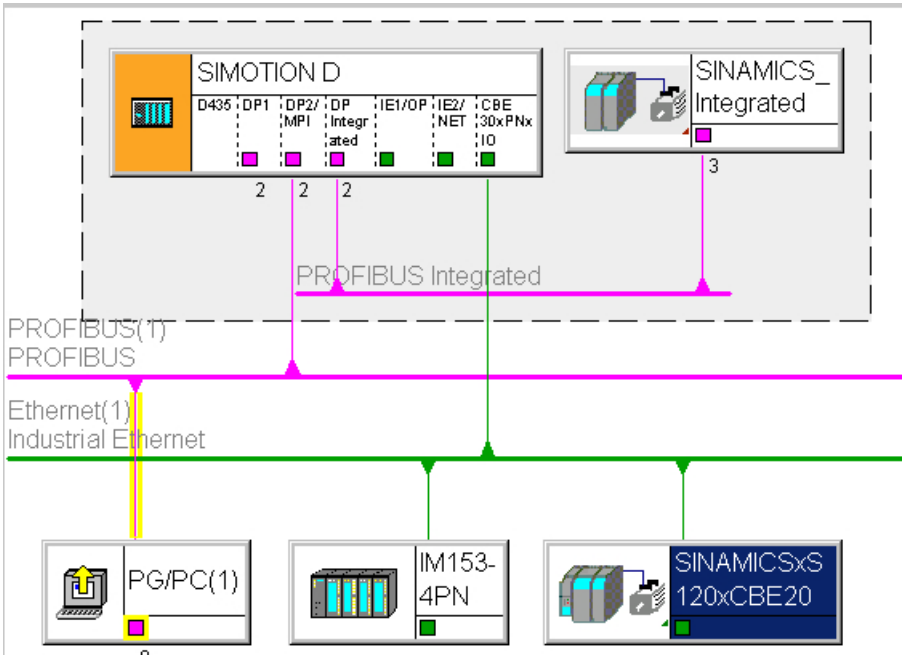


Figure 7-3 Example for PG/PC to PROFIBUS

- S7 routing to the other (master) PROFIBUS interfaces (only if configured)
- S7 routing to PROFIBUS Integrated
- S7 routing to the PROFINET interface (CBE30)
- S7 routing to X1400 on the CBE30
- S7 routing to the standard Ethernet interfaces (X120, X130) (V4.1.2 and higher)
- Access to nodes on the same network, e.g. HMI

Engineering system / HMI to Ethernet

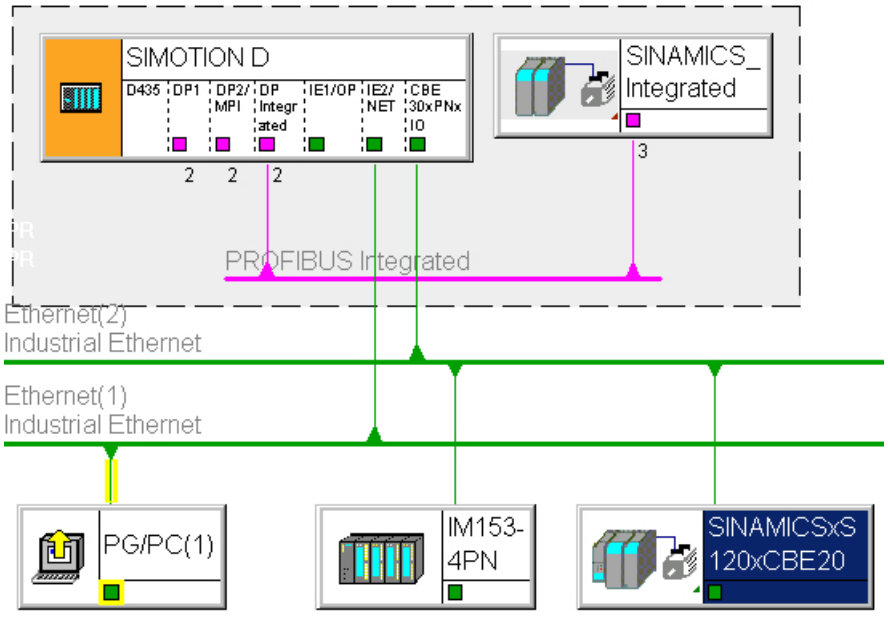


Figure 7-4 Example for PG/PC to Ethernet X120, X130

- S7 routing to the other (master) PROFIBUS interfaces (only if configured)
- S7 routing to PROFIBUS Integrated
- S7 routing to the PROFINET interface (CBE30)
- S7 routing to X1400 on the CBE30
- S7 routing between the Ethernet interfaces
- Access to nodes on the same network, e.g. HMI

7.5 Routing with SIMOTION D4x5-2 (example of D455-2 DP/PN)

Routing between the different interfaces

The two Ethernet interfaces of the D4x5-2 DP/PN (X127 P1 or X130 P1) each form a separate subnet.

The D4x5-2 DP/PN onboard PROFINET IO interface (X150, P1-P3) also forms a separate subnet. All ports of a PROFINET IO interface always belong to the same subnet.

- S7 routing is possible to every interface.
  E. g. you can set up a connection from a PROFINET/Ethernet subnet to a PROFIBUS via S7 routing.
- IP routing from subnet to subnet is not supported. You can use an external IP router for this.

There are therefore the following options for connecting a PG/PC or HMI device to a SIMOTION D using S7 routing.

Engineering system / HMI to PROFINET

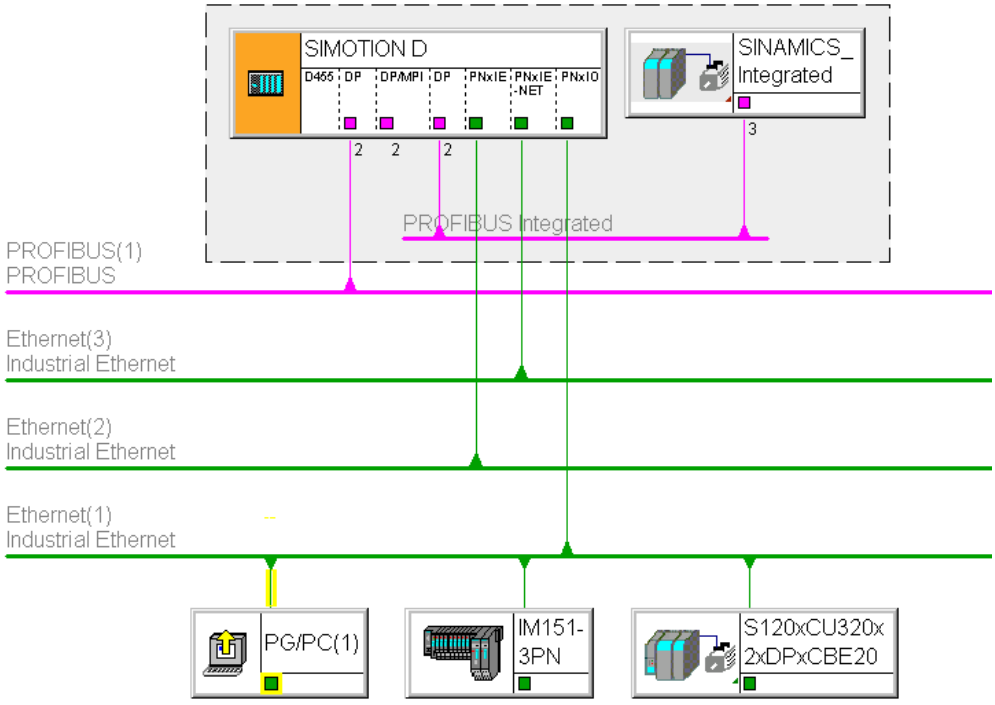


Figure 7-5 Example of PG/PC to PROFINET interface (PNxIO, X150)

- S7 routing to the (master) PROFIBUS interfaces (only if configured)
- S7 routing to the PROFIBUS Integrated

- S7 routing to the Ethernet interfaces PN/IE (X127 P1) and PN/IE-NET (X130 P1)
- Access to the components on the same subnet via the switch functionality of the PROFINET IO interface

Engineering system / HMI to PROFIBUS

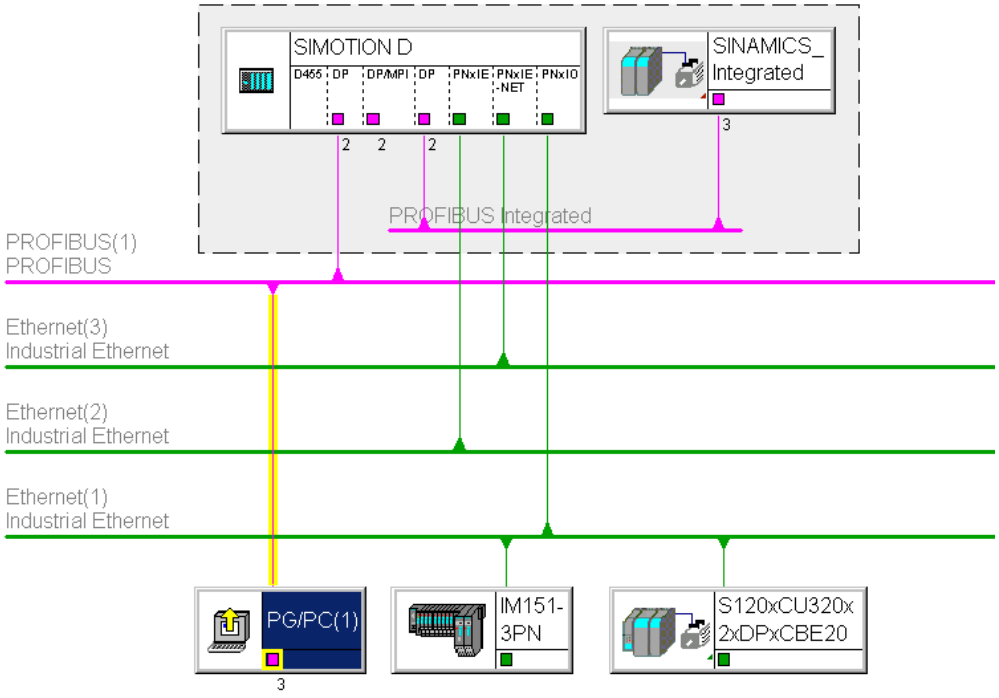


Figure 7-6 Example of PG/PC to PROFIBUS interface (DP, X126)

- S7 routing to the other (master) PROFIBUS interfaces (only if configured)
- S7 routing to the PROFIBUS Integrated
- S7 routing to the PROFINET interface (CBE30)
- S7 routing to the onboard PROFINET IO interface (X150, P1-P3)
- S7 routing to the Ethernet interfaces PN/IE (X127 P1) and PN/IE-NET (X130 P1)

Engineering system / HMI to Ethernet

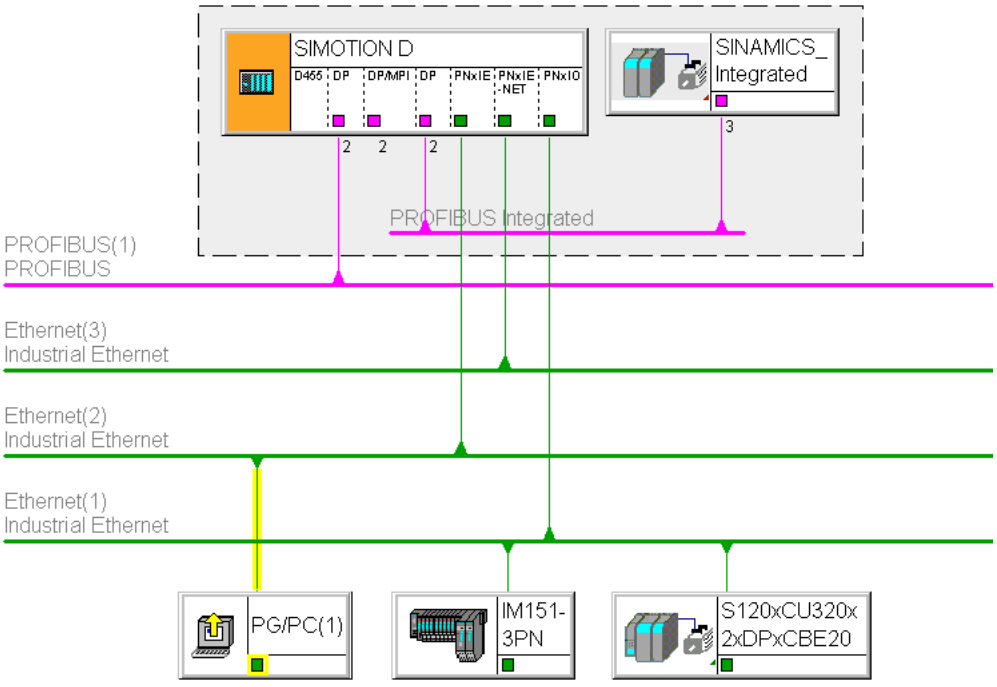


Figure 7-7 Example of PG/PC to Ethernet interface (PNxIE, X127)

- S7 routing to the other (master) PROFIBUS interfaces (only if configured)
- S7 routing to the PROFIBUS Integrated
- S7 routing to the PROFINET interface (CBE30)
- S7 routing to the onboard PROFINET IO interface (X150, P1-P3)
- S7 routing between the Ethernet interfaces

7.6 Routing for SIMOTION D to the SINAMICS integrated

All SIMOTION D controllers have a SINAMICS drive integrated: the so-called SINAMICS Integrated. As this is connected internally to the SIMOTION controller via PROFIBUS DP, S7 routing is imperative here to be able to access it. Here the telegrams have to be routed from the external interfaces of the SIMOTION controller to the internal PROFIBUS DP. Here, the internal PROFIBUS DP forms a separate subnet. This must be especially taken into account for the communication to several routing nodes.

7.7 Routing for SIMOTION P350

Description

S7 routing is possible:

- From PROFIBUS (IsoPROFIBUS board) on PROFINET subnet to MCI-PN board
- From PROFINET subnet to MCI-PN board on PROFIBUS (IsoPROFIBUS board)
- From SCOUT on SIMOTION P via softbus through the runtime on PN devices to the MCI-PN board and IsoPROFIBUS board
- From onboard Ethernet interfaces on PROFIBUS (IsoPROFIBUS board) and on PROFINET
- S7 routing between the Ethernet interfaces

IP routing is **not** possible via the P350 Ethernet interfaces.

Routing from PROFIBUS to PROFINET

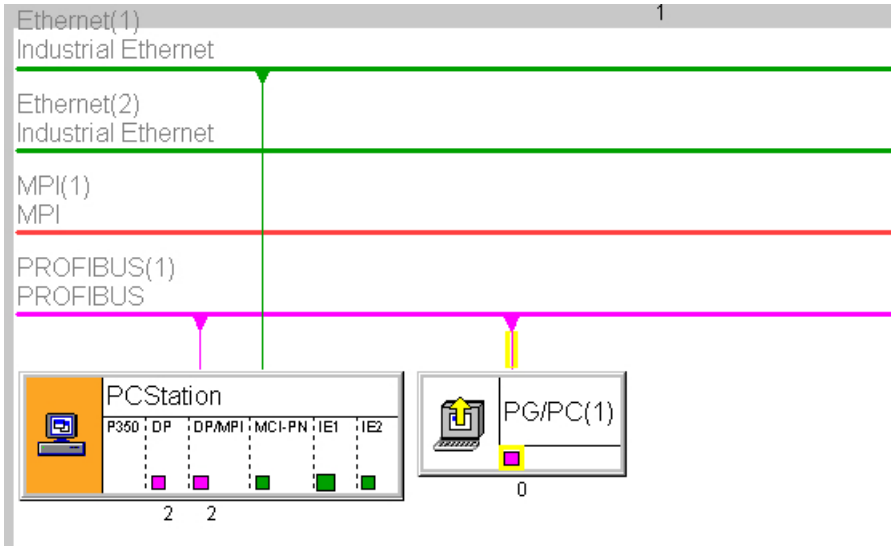


Figure 7-8 Example for P350 routing from PROFIBUS to PROFINET

Routing from PROFINET on PROFIBUS

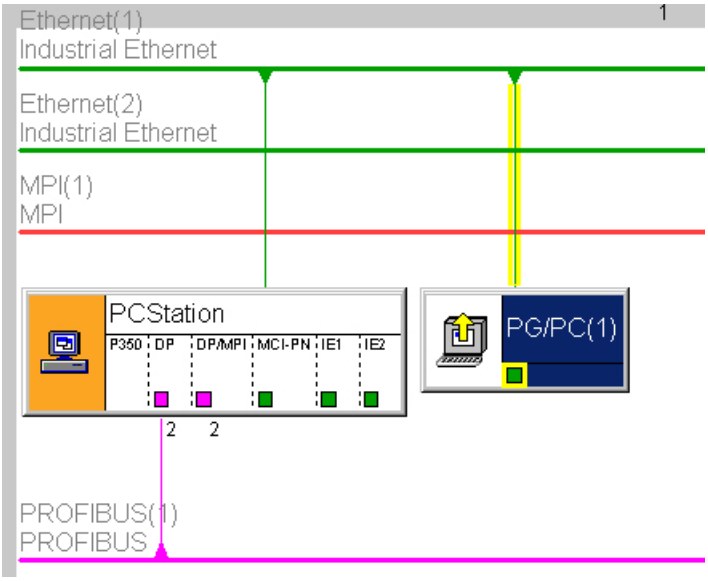


Figure 7-9 Example for P350 routing from PROFINET to PROFIBUS

7.8 Routing for SIMOTION P320

Description

S7 routing is possible:

- From the onboard Ethernet interface to the PROFINET subnet and the drive units or SIMOTION devices on the PROFINET subnet

Routing from Ethernet to PROFINET

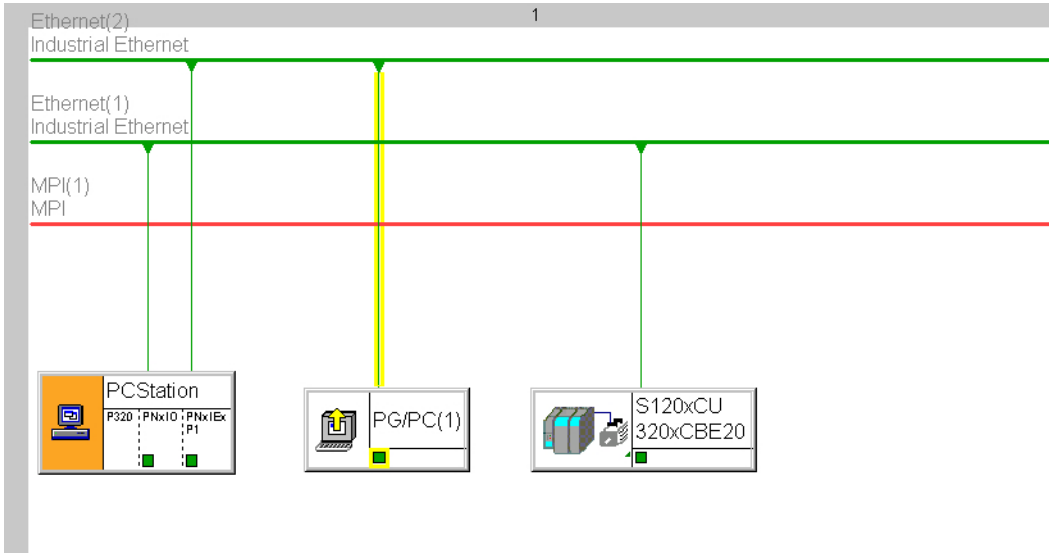


Figure 7-10 Routing for SIMOTION P320

SIMOTION IT 8

8.1 SIMOTION IT - overview

Description

SIMOTION IT provides the option of accessing this via standard web services (HTTP) and the integrated web server of the SIMOTION controllers.

Note

Appropriate protective measures (among other things, IT security, e.g. network segmentation) are to be taken in order to ensure safe operation of the system. You can find more information on Industrial Security on the Internet at:

www.siemens.de/industrialsecurity

This provides the following advantages.

- Location-independent open diagnosis / process monitoring
- Client device independent of the operating system (Windows, Linux, ...)
- Standardized communication interface for manufacturer-specific tools
- Independent of engineering system

- No version conflict between client application and SIMOTION RT (runtime)
- Series commissioning without engineering system

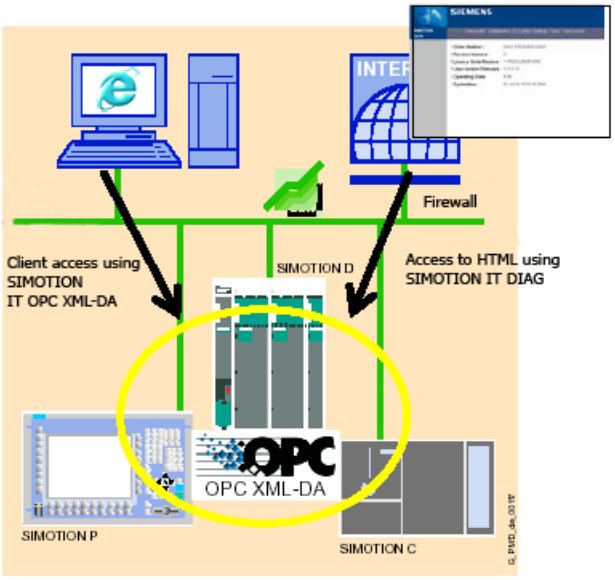


Figure 8-1 SIMOTION IT overview

SIMOTION IT provides the following service and diagnostics options via Internet technology:

- SIMOTION IT web server
- SIMOTION IT OPC XML DA
- Trace via SOAP
- File download using FTP (File Transfer Protocol)
- SIMOTION IT Virtual Machine

Further references

A detailed description of the SIMOTION IT products can be found in the *SIMOTION IT Ethernet-based HMI and Diagnostic Functions* product information on the SIMOTION SCOUT Documentation DVD.

Further information is available in the following manuals:

- *SIMOTION IT Diagnosis and Configuration, Diagnostics Manual*
- *SIMOTION – Jamaica*, Diagnostics Manual (Information on SIMOTION VM)

See also

Web access to SIMOTION (Page 261)

SIMOTION IT web server (Page 262)

SIMOTION IT OPC XML DA (Page 265)

8.2 Web access to SIMOTION

Description

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

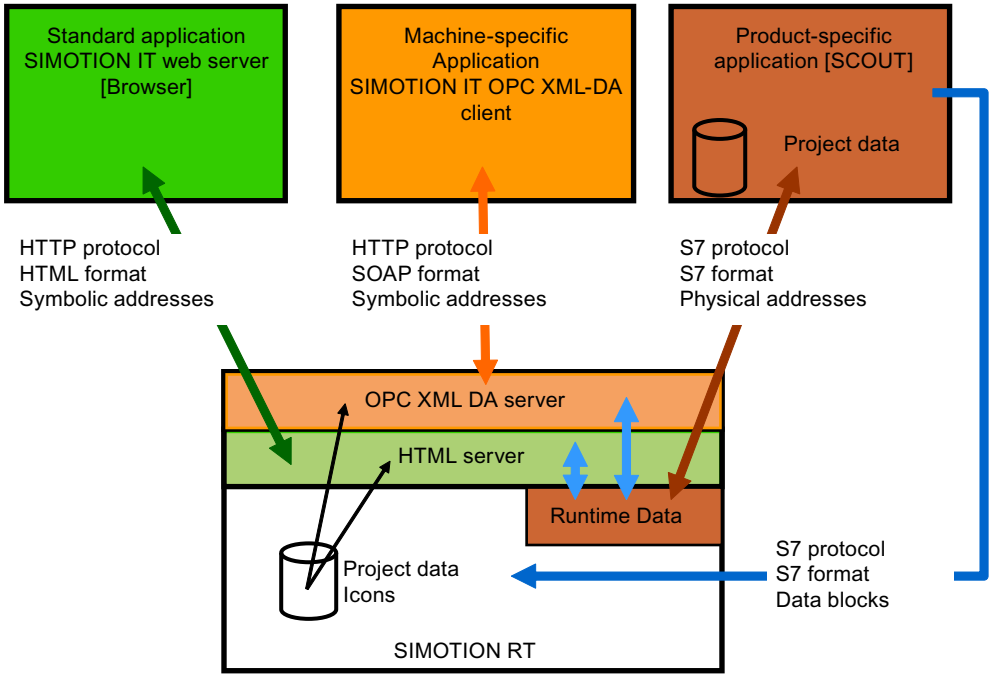


Figure 8-2 Access to SIMOTION

See also

SIMOTION IT web server (Page 262)

SIMOTION IT OPC XML DA (Page 265)

8.3 SIMOTION IT web server

Description

The SIMOTION IT web server allows a PC to use any Internet browser to access the the web pages of the integrated web server of the SIMOTION controllers.

You will find a detailed description in the *SIMOTION IT Diagnostics and Configuration* Diagnostics Manual.

Standard diagnostic pages

SIMOTION provides the following standard diagnostic pages:

- Start page
- Device Info/IP-Config
  (information about the firmware, devices, device components, technology packages, and data of the SIMOTION device Ethernet interface)
- Diagnostics
  (CPU utilization, memory use, operating mode, display of task runtimes, trace for devices and system, service overview)
- Message&Logs
  (diagnostics buffer, SIMOTION alarms and Drives, syslog and userlog)
- Machine Overview
  (modules and topology of a machine as well as hardware configuration information)
- Manage Config
  (loading SIMOTION IT web server configurations, device updates, saving device data)
- Settings
  (setting the time zone, switching operating modes, changing the display of user-defined pages)
- Files
  (accessing the SIMOTION file system, uploading and downloading files, creating folders, and storing additional data, e.g. documentation)

Simplified standard pages

To enable the best possible display of HTML pages on devices such as cell phones or PDAs, a set of special pages is provided for version 4.1.3 and higher. These contain a simplified representation of information from the standard pages. The start page of the simplified standard pages can be reached at the address http://<IPAddr>/BASIC.

Configuration via WebCfg.xml

There are two files which can be used to configure SIMOTION IT web server:

- WebCfg.xml
- WebCfgFrame.xml

The WebCfg.xml configuration file is used to configure user-relevant settings in the web server. The WebCfgFrame.xml file contains the manufacturer's settings.

User-defined pages

The SIMOTION IT web server offers the option of integrating individually designed web pages. Two mechanisms are available for accessing SIMOTION variables:

- JavaScript libraries opcxml.js and appl.js
- MiniWeb Server Language (MWSL)

User-defined pages can be displayed in the "User's Area". For this purpose, they must be stored in the FILES folder of the SIMOTION file system.

Trace via SOAP

The WebTrace is almost identical to the SIMOTION SCOUT trace. The only difference is in output of the data format and in the number of signals that can be parameterized.

The "Trace via SOAP" function package enables SIMOTION variables to be written to a buffer. The values are packed in a file and can be retrieved asynchronously via an HTTP request. This interface can only be used by client applications.

The trace can be parameterized via a web interface and the recording viewed immediately using a TraceViewer.

Variable access

The variable access for the SIMOTION IT applications is implemented using a variable provider. The following variable providers currently exist.

- MiniWeb
- SIMOTION
- SIMOTION diagnostics
- UserConfig
- IT Diag

These variable providers allow access to the following variables:

- Device system variables
- TO system variables
- Global unit variables from the interface section
- TO configuration data
- Also global device variables and I/O variables as of 4.2
- Drive parameters
- Setting of the operating mode, execute RamToRom, execute ActiveToRom
- Technological alarms
- Diagnostics buffer

Secure HTTPS connection

The Secure Socket Layer protocol (SSL) enables encrypted data transmission between a client and the SIMOTION device. The Secure Socket Layer protocol forms the basis for HTTPS access. Encrypted access can take place via both SIMOTION IT OPC XML DA and the SIMOTION IT web server.

8.4 SIMOTION IT OPC XML DA

Description

The SIMOTION IT OPC XML DA server enables access via Ethernet to data and operating modes of the SIMOTION controller. As of V4.2, a separate license is no longer required for SIMOTION IT OPC XML DA.

OPC stands for open connectivity and refers to standardized software interfaces which enables the exchange of data between applications from different manufacturers in automation technology. With OPC XML DA, it is possible to communicate with a control using Ethernet-based standard message frames. Commands are transmitted via the SOAP (Simple Object Access Protocol) communication protocol.

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:

- Open communication using HTTP, SOAP, OPC XML between client and SIMOTION device
- Uses the OPC XML DA 1.0 specification of the OPC Foundation
- Access to SIMOTION variables
  - Reading/writing
  - Cyclical reading of so-called "subscriptions"
  - Browsing
- Trace 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 the application that you want to access on the SIMOTION OPC XML DA server yourself.
- Access protection with user groups, user ID, and password

The following figure is a schematic representation of access to the OPC XML DA server.

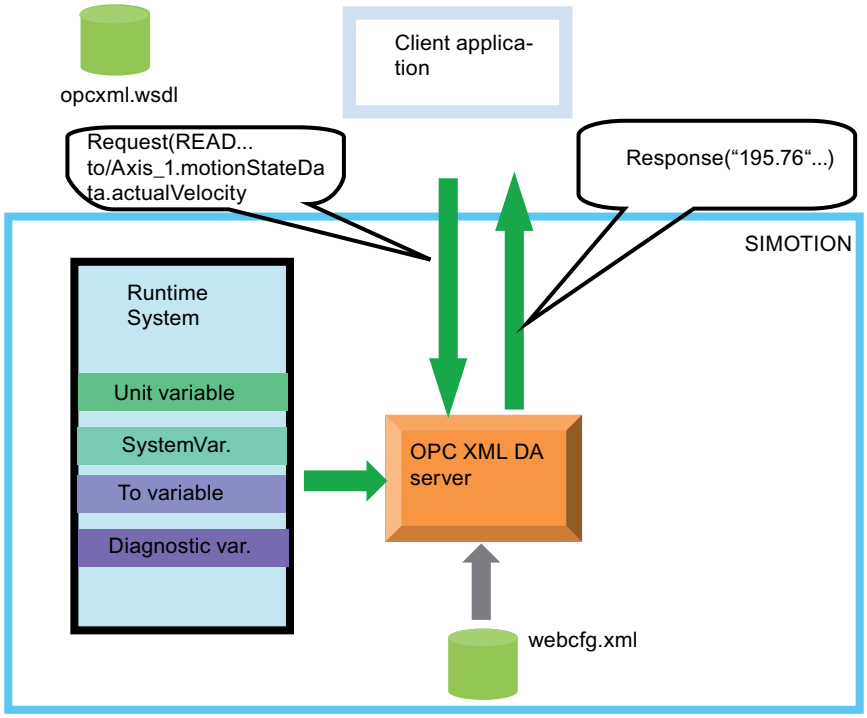


Figure 8-3 Access to the OPC XML DA server

PROFIsafe

9

9.1 Communication relationships for drive-based safety

Description

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

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

Signal flow for selecting and deselecting the drive-based safety functions and their signaling to the drive control process

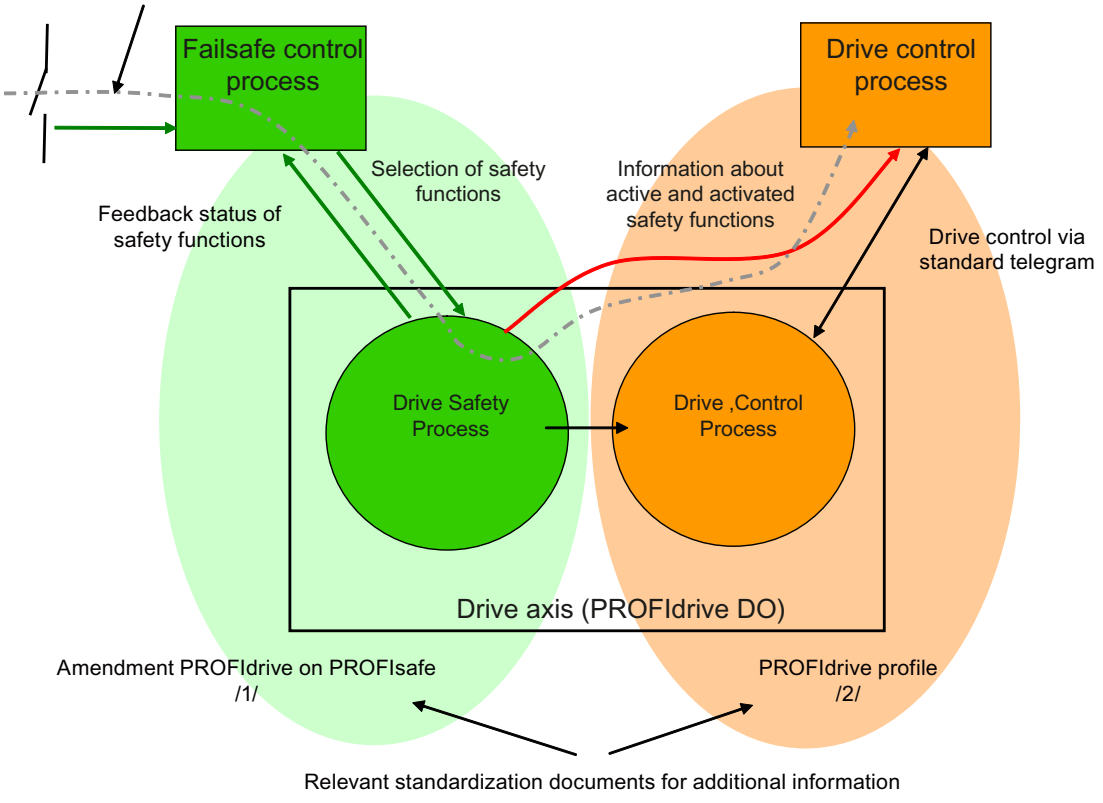


Figure 9-1 Communication relationships for drive-based safety

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

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

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

9.2 Message frames and signals in drive-based safety

Description

Since SIMOTION does not have a safe logic function, it cannot attend to the F control process. Instead, this is carried out using a second controller featuring F functionality (usually a SIMATIC F-CPU). The image below shows how this setup works, using the example of a SINAMICS axis. A PROFIsafe-secured communication channel runs between the DO drive and the SIMATIC F-CPU. Standard telegram 30, consisting of a safety control word and status word (among other things), is normally available for this communication channel.

As of V4.3, the following PROFIsafe telegrams are available:

- Telegram 30
- Telegram 31
- Telegram 901
- Telegram 902 (from V4.4; only in connection with the TIA portal)

With the user program on the F-CPU, the safety control word is used to select or deselect the configured drive-based safety functions in the drive (drive safety process). The feedback from the active safety functions is sent in the input data to the F-CPU, via the safety status word. A safe process in the drive is used to send the feedback from the active safety functions via a secured communication channel; therefore, the feedback may be used for activating protection zones and doors via the F-CPU.

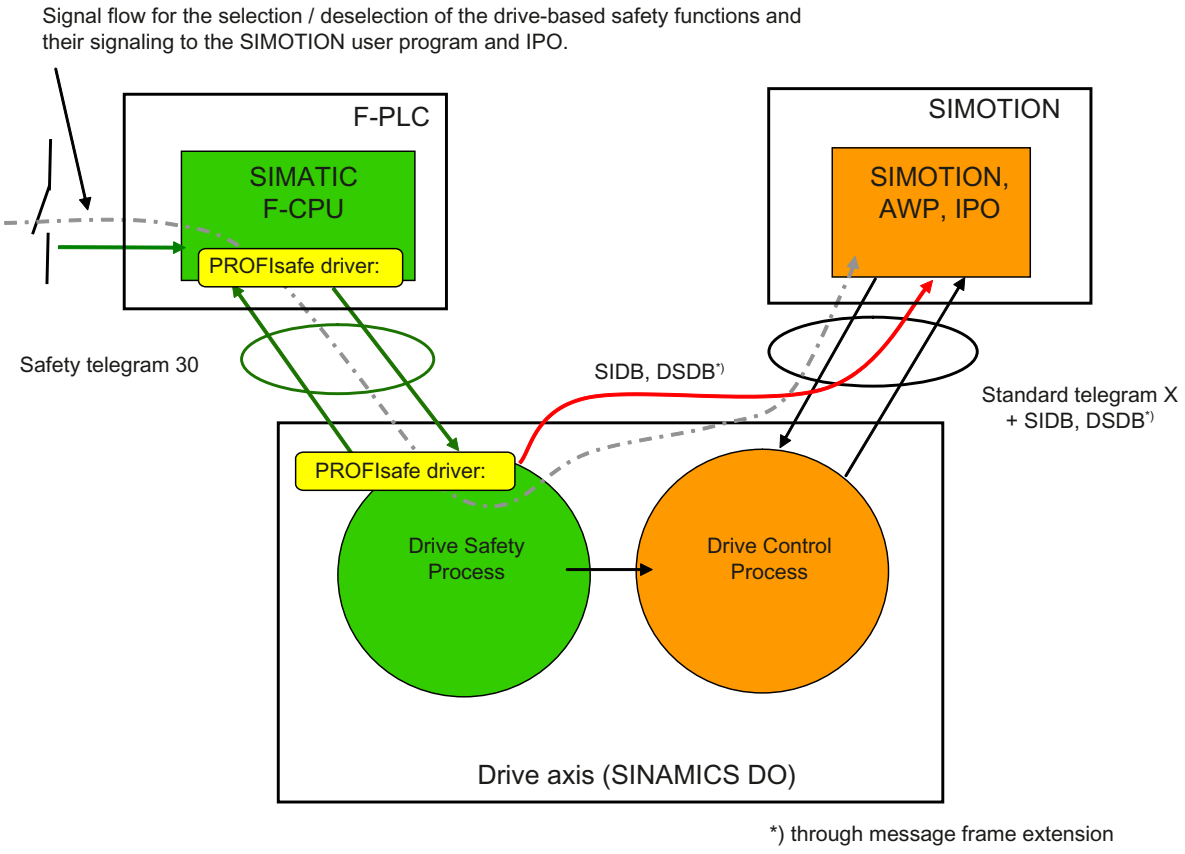


Figure 9-2 Messages and signals in drive-based safety

Safety information channel SIDB (up to V4.4) and DSDB (from SIMOTION V4.4)

In addition to controlling safety functions in the drive via the PROFIsafe channel, you have the option of transmitting statuses and activation states of the drive-based safety functions from the drive to SIMOTION via a safety information channel (SIDB Safety Data Block). This has been expanded through the new standard DSDB (Drive Safety Data Blocks) from SIMOTION V4.4. The previous SIDB telegram extension block is replaced by two telegram extension blocks, DSDB1 and DSDB2. Instead of the previous single bico interconnections to bits in SINAMICS, interconnections are made to complete signals (bit blocks) with the new SC types. As a result, status information also becomes independent of the activated safety function (basic or extended functions). With DSDB the entire safety-oriented functionality configured on the drive is available on the TO. Configuration is automatic and uses the DSDB channel (Drive Safety Data Blocks).

With DSDB the interface to the user program has been expanded and new interfaces for the safe brake test (SBT) and test stop functionality have been implemented.

Note

The Drive Safety Data Block is an extension of the PROFIdrive axis telegrams. Transmission is according to PROFIdrive and is therefore not a secure safety telegram.

The purpose of the safety information channel is to provide the option of integrating the drive motion control (IPO, SERVO) and the entire user program (UP) into the higher-priority execution of both drive-based safety functions and the F-control user program.

Typical SIMOTION responses are, for example:

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

See also

New in SIMOTION SCOUT as of V4.3 (Page 112)

9.3 SIMOTION F proxy functions

Description

SIMOTION features integrated F-Proxy functionality for the purpose of PROFIsafe connection of integrated SIMOTION D drives, as well as SINAMICS drives that are controlled by SIMOTION but are in a different communication domain from the F-CPU, for example. The F proxy functionality enables transparent routing of safety message frames from the SIMOTION I slave or I device interface to the respective SIMOTION master or controller interface on which the drive is configured. Despite the SIMOTION routing function, PROFIsafe communication between the F-CPU and drive is secured, as the PROFIsafe drivers in the end points (F-CPU, drive) monitor communication securely.

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

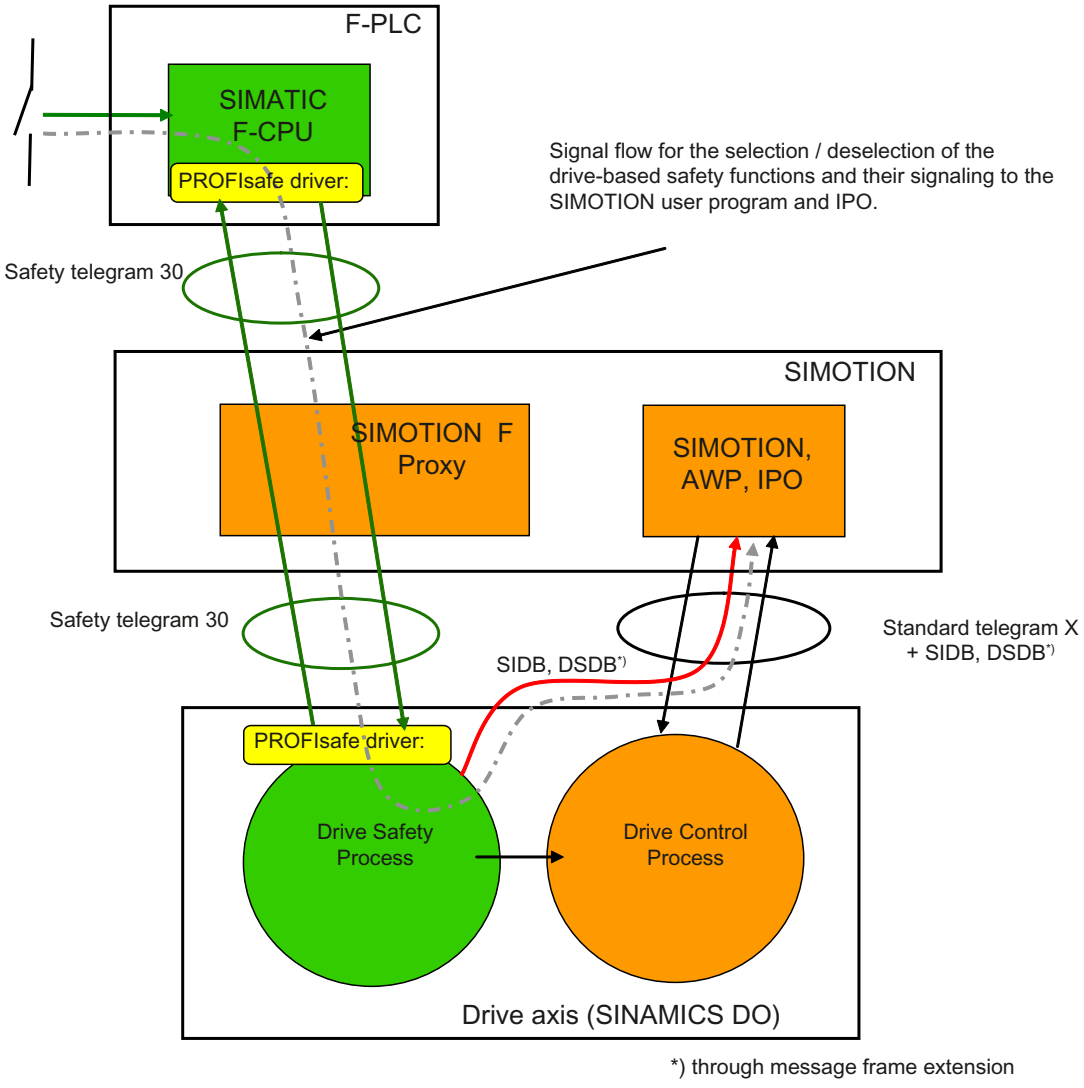


Figure 9-3 Routing the PROFIsafe channel with F proxy functionality

9.4 PROFIsafe properties for configuration

Mechanisms with PROFIsafe

Basically the PROFIsafe can communicate both via PROFINET and PROFIBUS. The I-device F-Proxy, the I-slave F-Proxy, the PROFINET shared device, and the PROFIBUS data exchange broadcast are available as mechanisms for PROFINET and PROFIBUS.

Note

Mixed operation is **not** supported for the PROFIsafe configuration. If, for example, in the case of a SINAMICS drive, the safety configuration for one axis is carried out via F-Proxy and those of the other axes are configured via shared device, this configuration will be rejected with a consistency error during compilation. The F-Proxy variants are recommended for the PROFIsafe configuration.

For all the mechanisms, the F data is managed by the F-CPU and the Motion Control data by the SIMOTION controller. The PROFIsafe communication properties required are set in the HW Config for the fail-safe parameters.

Note

For SIMOTION projects (RT version) < V4.2, secured PROFIsafe communication is only possible via PROFIBUS. Therefore, with SIMOTION devices < V4.2 with PROFIBUS and PROFINET interfaces, you can only implement PROFIsafe via the PROFIBUS interface and the Motion Control tasks (for example) via the PROFINET interface. Mixed operation (Motion Control via PROFINET and PROFIsafe via PROFIBUS) is, however, only possible with the integrated SIMOTION D drives. It is not permissible if you are using external drives (e.g.S120 CU320-2) on a SIMOTION controller.

Overview of failsafe parameters for PROFIsafe during configuration

To access the PROFIsafe dialog box, double-click the **PROFIsafe** entry in the drive rack in HW Config.

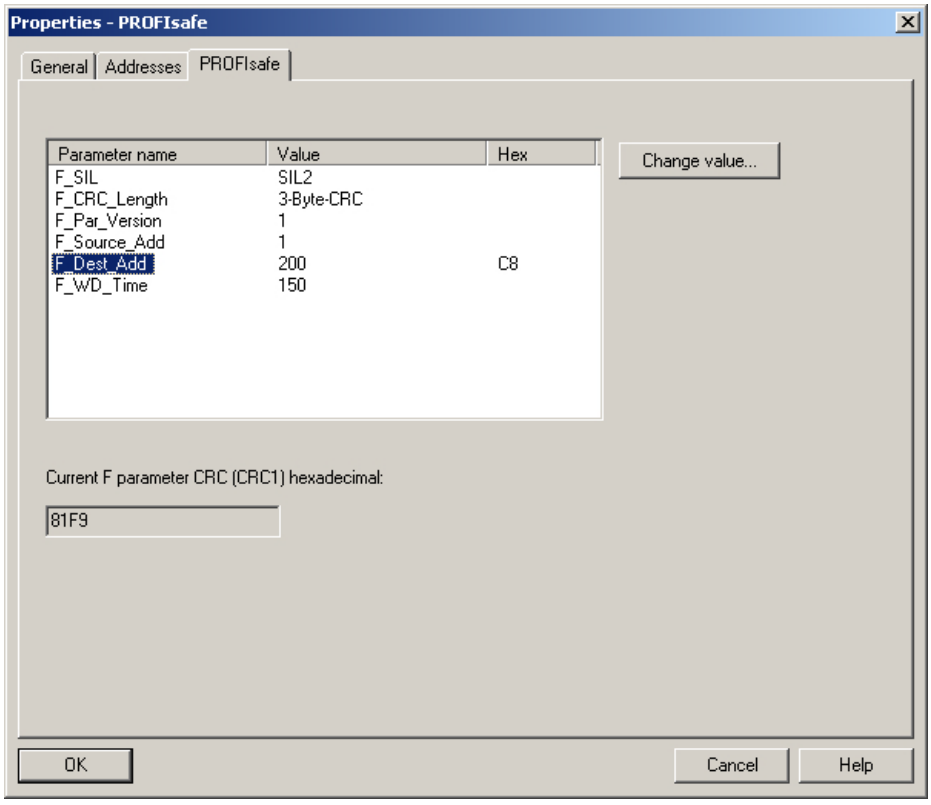


Figure 9-4 Properties of PROFIsafe using the example of I-device F-proxy

F_CRC_Length and F_Par_Version

Identifies not only the length of the failsafe useful data but also the PROFIsafe MODE.

- PROFIsafe V1-MODE
  F_CRC_Length = 2 to useful data length of 12 bytes
  F_CRC_Length = 3 from useful data length of 13 bytes
  F_Par_Version = 0
- PROFIsafe V2-MODE
  F_CRC_Length = 3 to useful data length of 12 bytes
  F_CRC_Length = 4 from useful data length of 13 bytes
  F_Par_Version = 1

Note

PROFIsafe V2 mode is not supported by all devices and/or firmware versions. Before configuration, find out which mode is available as of which version.

SINAMICS G120 CU240D DP F and SINAMICS G120 CU240S DP F only support V2 mode as of firmware version V3.2.

F_Dest_Add: 1-65534
F_Dest_Add determines the PROFIsafe destination address of the drive object.
Any value within the range is allowed, although it must be manually entered again in the Safety configuration of the drive in the SINAMICS drive unit and/or entered when configuring Safety Integrated (drive parameters p9610/p9810).

F_WD_Time: 10- 65535
A valid current safety telegram must be received from the F-CPU within the monitoring time. The drive will otherwise switch to the safe state.
The monitoring time should be of sufficient length to ensure not only that the communication functions tolerate telegram time delays, but also that the fault response is triggered as quickly as possible if a fault occurs (e.g. interruption of the communication connection).

For additional information on F Parameters, refer to the online help of the **PROFIsafe** dialog box.

9.5 PROFIsafe via PROFINET

9.5.1 Principles of I device failsafe proxy

Brief description

Using the I device F-Proxy you can produce a PROFIsafe configuration with an F host (F-CPU SIMATIC) on PROFINET with SIMOTION devices (SIMOTION D, SIMOTION P, SIMOTION C) for the lower-level drives. The routing of cyclic PROFIsafe data to SINAMICS Integrated and SINAMICS drives on external PROFIBUS or PROFINET is therefore possible.

You can also use the functionality of a shared iDevice for the iDevice F Proxy from SIMOTION V4.4 and higher. The iDevice interface can be shared between an F CPU and a SIMOTION controller. This means that an iDevice can communicate with two higher-level controllers as an IO device. At the same time, higher-level controllers can access certain modules, e.g. an F-CPU can access the F-Proxy submodules, using the shared device. You can find a configuration example in the Shared iDevice section and PROFIsafe (Page 310).

Note

F-Proxy modules can be set up either on the CBE30-2 or on the integrated PROFINET interface.

A failsafe host communicates with the drives via the I device interface and an F-Proxy of a SIMOTION CPU. These drives may be located on PROFIBUS DP external, PROFIBUS DP integrated and the PROFINET IO system of the SIMOTION CPU. The SIMOTION CPU's communication segments feature SINAMICS S120/S110, incl. SINAMICS Integrated/CX32/ CX32-2 and G120.

The higher-level project with the F-CPUs is the master project. There may be several F-CPUs in the master project. The lower-level projects with SIMOTION CPUs are designated as subprojects. All F slaves of all segments (PROFIBUS/PROFINET) which originate from SIMOTION are mapped by the F-Proxy as n F submodules in an F-Proxy module.

During configuration, a failsafe I device is produced as a substitute of a SIMOTION device with drives and imported into the F-CPU master project.

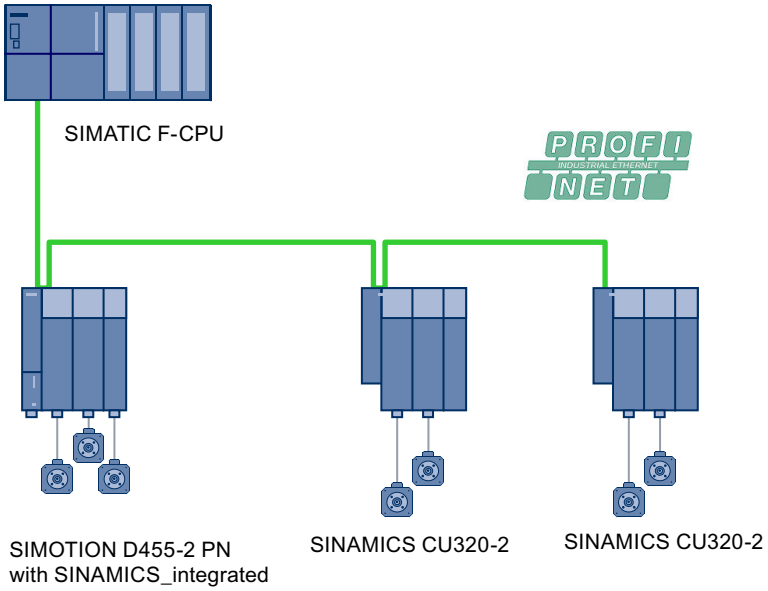


Figure 9-5 Example topology for F-CPU master with SIMOTION D and lower-level drives

Module/submodule structure

The submodules of the drive with the safety message frame are mapped to submodules of the PROFINET I device interface, regardless of whether the drive is connected to SIMOTION via PROFIBUS, integrated PROFIBUS, or PROFINET.

With an I device failsafe proxy, all failsafe proxy submodules of a SIMOTION device are depicted in one single module (module 2).

The diagram below shows an F-CPU as the master, to which a SIMOTION with integrated and lower-level drives is subordinate.

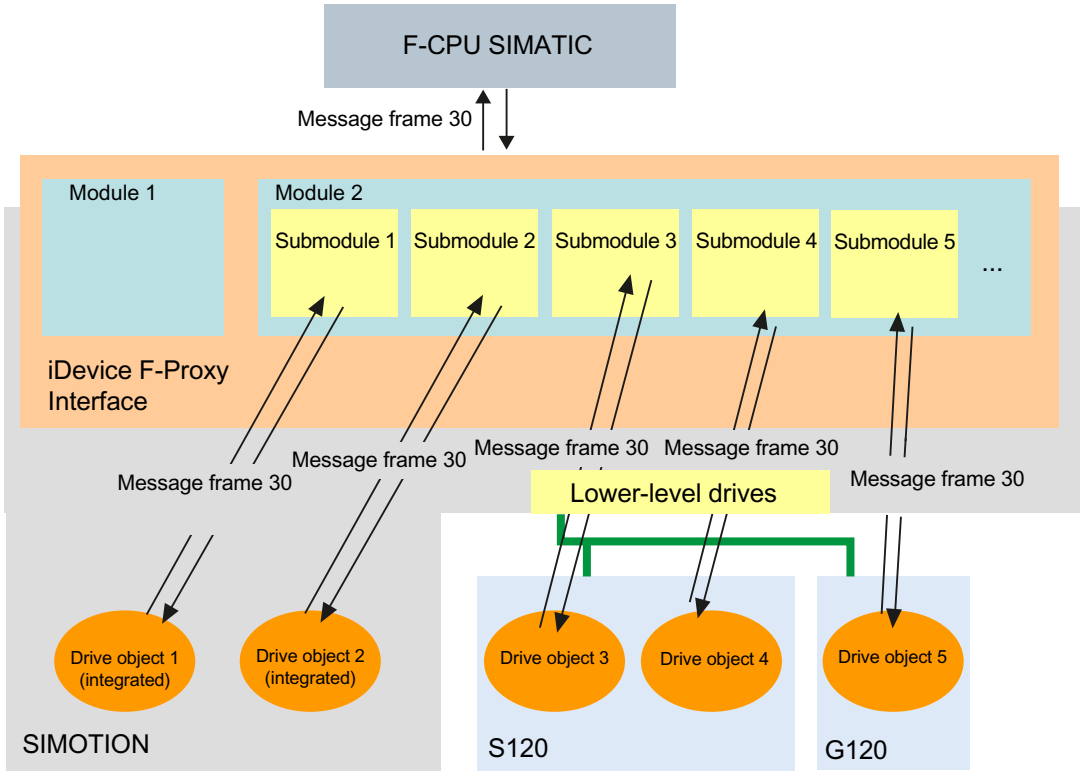


Figure 9-6 Master project with a sub-project as the I device failsafe proxy

See also

Use of Safety with MRRT (Page 105)

Shared device via PROFINET (Page 303)

Configuring I device failsafe proxy (Page 285)

9.5.2 Supported devices and software requirements for I device failsafe proxy

Software requirement

- Step7 V5.5 or higher
- SINAMICS S120, V2.6 or higher
- SINAMICS G120, V3.2 or higher
- SIMOTION, V4.2 or higher
- S7 F configuration pack, V5.5 SP8 or higher (if using Safety/PROFIsafe)
- Valid from PROFIsafe V2

Supported devices

Drive units

- SINAMICS Integrated
- SINAMICS Controller Extension (CX32, CX32-2)
- SINAMICS devices on lower-level PROFIBUS
- SINAMICS devices on lower-level PROFINET

Devices with I-device F-proxy functionality

- SIMOTION D4xx, D4x5-2
- SIMOTION C240 PN
- SIMOTION P350-3, P320-3, P320-4
- F-CPU (List of supported F-CPUs (http://support.automation.siemens.com/WW/view/en/44383955))

Table 9-1 The table shows which SIMOTION devices support the I-device F-Proxy and which SINAMICS devices can be accessed via the I-device F-Proxy.

Overview of devices capable of I-device F-proxy functions	
SIMOTION IO controller	
Controller Based	C240 PN
PC Based	P320-3 P350-3 P320-4
Drive Based (blocksize)	D410 PN D410-2 DP/PN
Drive Based (booksize)	D4x5 with CBE30 D445-1 with CBE30 D4x5-2 DP/PN D4x5-2 DP/PN with CBE30-2
SINAMICS IO devices and DP slaves	
S120 CX32	CX32 CX32-2
S120	CU320 DP CU320 CBE20 CU320-2 DP CU320-2 DP CBE20 CU310-2 DP CU310 DP CU310 PN CU310-2 PN CU320-2 PN

Overview of devices capable of I-device F-proxy functions	
S110	CU305 CU305 PN
G120	CU240S PN F CU240D PN F CU240E-2 CU240D-2 CU250D-2

9.5.3 Detailed description/properties of I device failsafe proxy

Specific properties of I device failsafe proxy

Acyclic services on I device failsafe proxy

The current I device interface does not support acyclic data transfer. The I device failsafe proxy submodules do not have parameter access (Parameter Access Point PAP) and cannot convey alarms.

If an alarm channel is to be used, the drive must be operated on PROFINET and directly incorporated in SIMOTION and the F-CPU using the Shared Device function. Here the F data of the F-CPU and the Motion Control data are managed by SIMOTION.

Supported telegrams

The message frames 30, 31, 901, and 902 are supported (TIA portal only).

Isochronous mode

The F-Proxy submodules on the SIMOTION CPU are RT and can be operated non-isochronously.

Supported lower-level bus systems

Drives on PROFIBUS integrated, PROFIBUS external and PROFINET external are supported. All these bus interfaces can also be routed via the I device failsafe proxy at the same time.

Supported I device interfaces

All PROFINET-capable SIMOTION devices are supported.

Supported number of failsafe proxy submodules

A maximum of 128 I device submodules are supported. Of these, up to 64 submodules can be used for Safety. The other 64 are available for standard IOs.

Reaction times - Transmission time for I device F-Proxy

If data is transmitted from the F-CPU via the I device F-Proxy, this extends the runtime for the SINAMICS drives via the I device F-Proxy by a maximum of 2 servo cycle clocks per transmit direction. Use the servo cycle clock of the lower-level system as the servo cycle clock.

9.5.4 PROFIsafe via PROFINET with an F-CPU

Description

A detailed description of how you can configure PROFIsafe with PROFINET in conjunction with a SIMATIC F-CPU can be found described in an FAQ.

See Actuating internal drive safety functions via SIMOTION and PROFINET with PROFIsafe (http://support.automation.siemens.com/WW/view/en/50207350).

See also

Use of Safety with MRRT (Page 105)

9.5.5 Topology overviews I device F-Proxy

9.5.5.1 Topology for I device failsafe proxy for PROFIBUS drive units

Example of topology

The diagram below shows a topology in diagrammatic form in which the Safety drives are connected to the SIMOTION CPU via PROFIBUS DP.

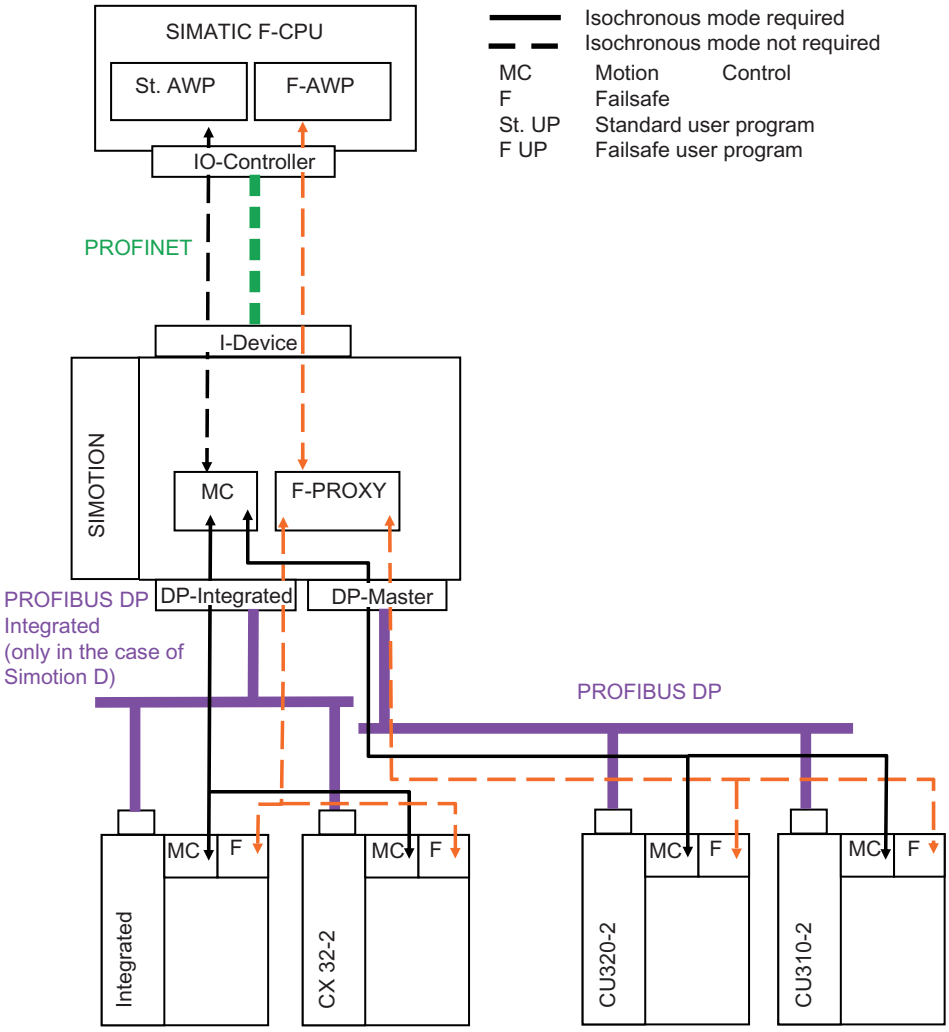


Figure 9-7 Topology for I device failsafe proxy for PROFIBUS drive units

9.5.5.2 Topology for I device failsafe proxy for PROFINET drive units

Example of topology

The diagram below shows a topology in schematic form in which the Safety drives are connected to the SIMOTION CPU via PROFINET and/or internally via PROFIBUS DP Integrated.

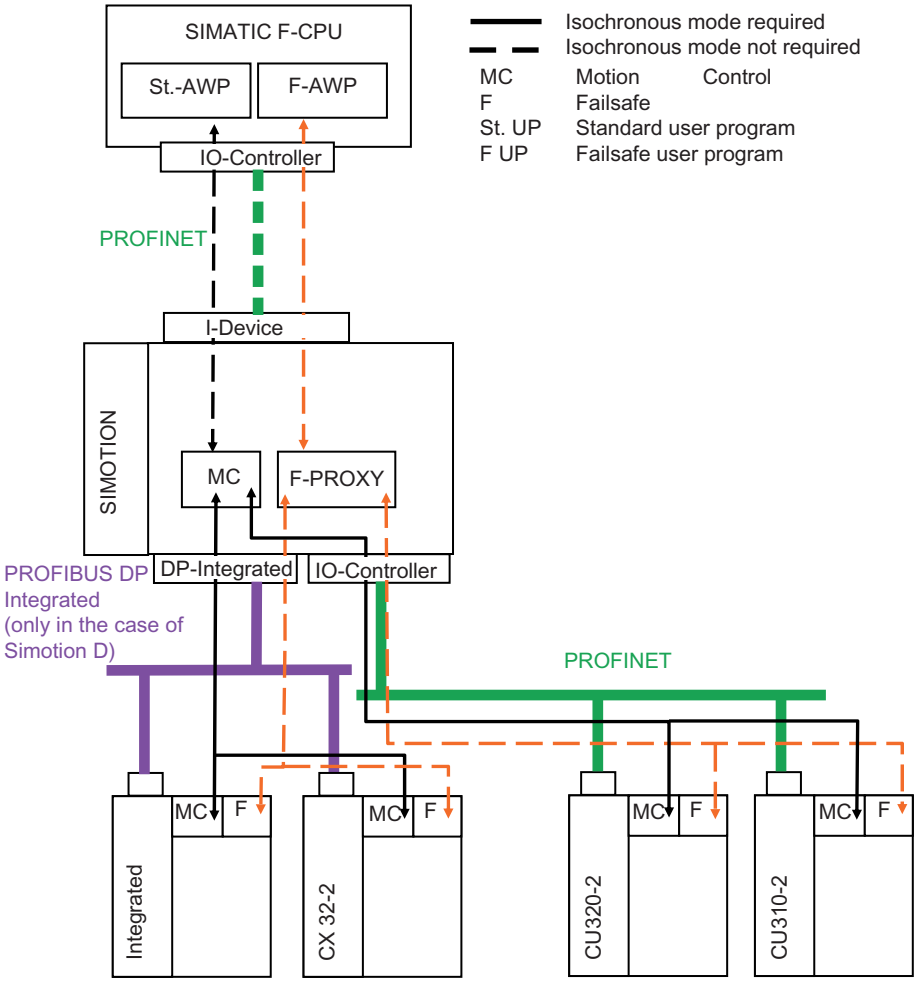


Figure 9-8 Topology for I device failsafe proxy for PROFINET drive units

9.5.5.3 Topology for I device failsafe proxy for PROFIBUS and PROFINET drive units

Example of topology

The diagram below shows a topology in schematic form in which the Safety drives are connected to the SIMOTION CPU via PROFINET and PROFIBUS DP.

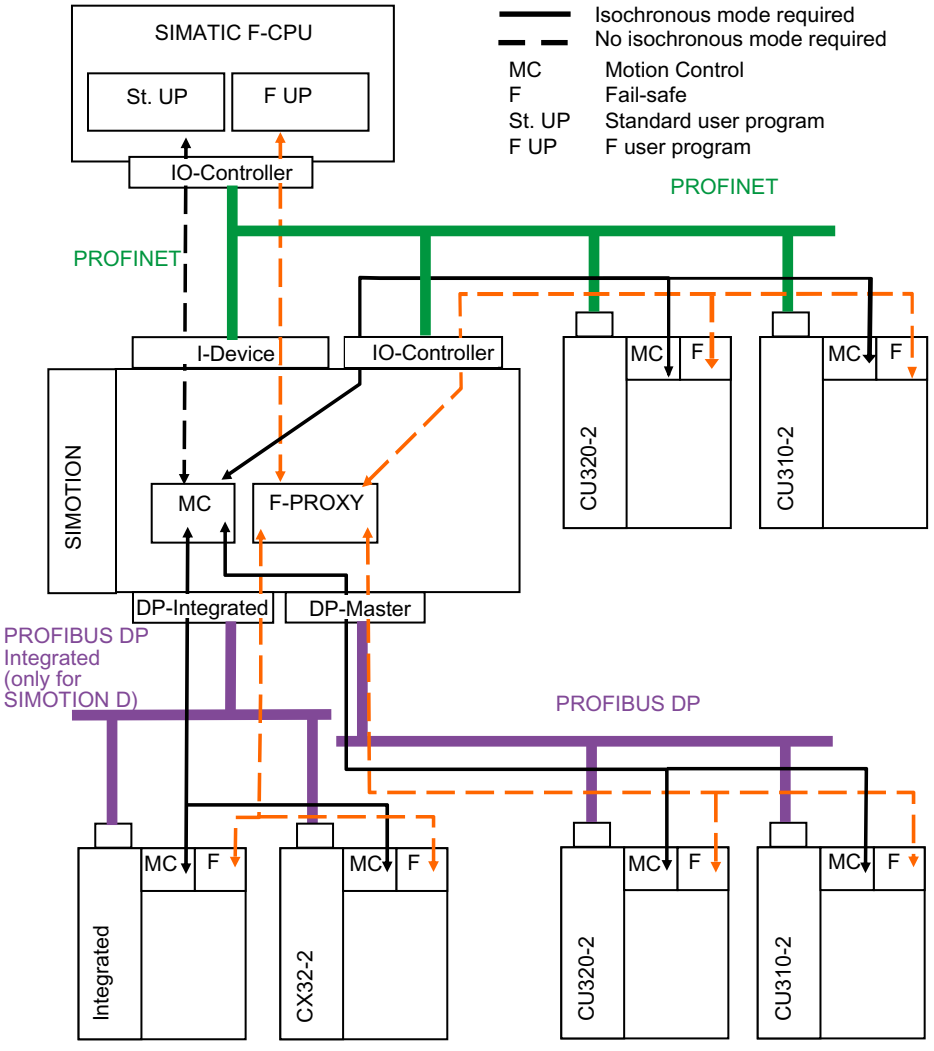


Figure 9-9 Topology for I device failsafe proxy for PROFIBUS and PROFINET drive units

9.5.6 Configuring I device failsafe proxy

9.5.6.1 Basic configuration process for I device failsafe proxy

Configuration requirements

A failsafe host communicates with the drives via the I device interface and an F-Proxy of a SIMOTION CPU. These drives may be located on PROFIBUS DP external, PROFIBUS DP integrated and the PROFINET IO system of the SIMOTION CPU.

Basic configuration process

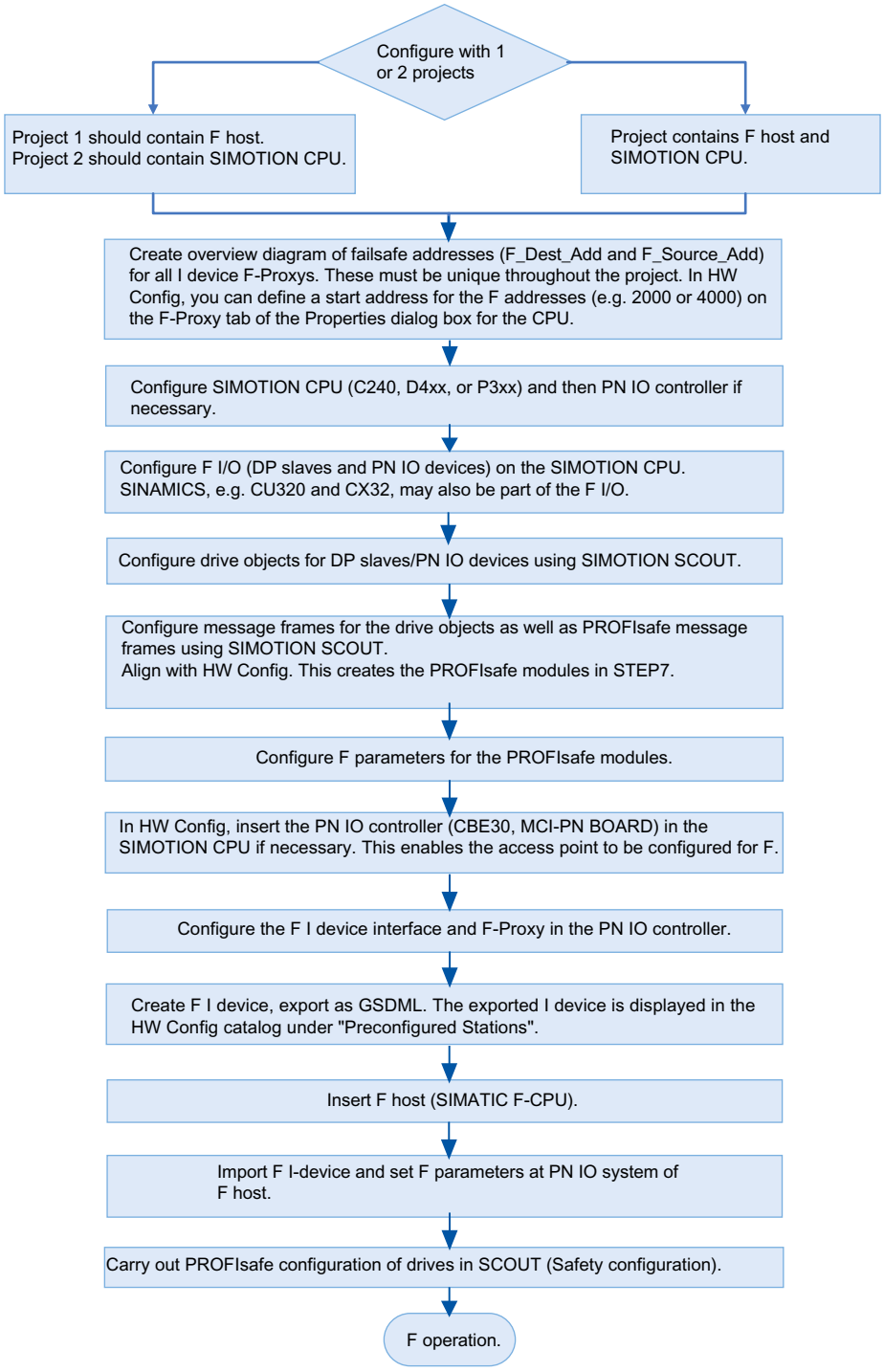


Figure 9-10 Basic configuration process for I device failsafe proxy

9.5.6.2 Configuration example for SIMOTION D435 and SINAMICS S120 via PROFINET

Configuration example for I device failsafe proxy

In the example below, you will configure a SIMATIC F-CPU 317F-2PN/DP V3.2. In addition, you will use a SIMOTION D435-2 DP/PN with a SINAMICS S120 CU320-2 PN, which is connected to the SIMOTION via the PN interface (CBE30-2). The exported I device F-Proxy of the lower-level SIMOTION CPU is then imported into the higher-level F-CPU. In the example, only one project is used in the configuration represented.

Configuring an I device F-Proxy

1. Create an overview diagram with the F-CPU, SIMOTION CPU and the drives intended to support PROFIsafe. In the example, there is just one SIMOTION CPU and a SINAMICS S120. Enter the start addresses of the CPU and the required drives into this overview diagram. The specified addresses are to be used later during configuration. The overview diagram is only required for larger projects.
2. Create a new project in SIMOTION SCOUT.
   You can find further information on configuration with PROFINET in the chapter Configuring PROFINET IO with SIMOTION (Page 111).
3. Add a SIMOTION D435-2 DP/PN. The check box **Open HW Config** must be activated. Click **OK** to confirm. HW Config opens.
4. In HW Config, enter a start address for the F addresses of the F-Proxy under SIMOTION CPU. All F_Dest_Add for the lower-level drives then use this start address, making them easier to manage in the case of more extensive projects. If you use 4000 as a start address, for example, the first F_Dest_Add of the drive is allocated as 400, etc. Start address 2000 is used as standard.
5. Insert a CBE30 if necessary and configure the PROFINET network.
6. Add a SINAMICS S120 CU320-2 PN (e.g. V4.7) to HW Config and configure the interface.
7. Configure the drive unit in SIMOTION SCOUT with the help of the wizard. You should already be thinking here about selecting the appropriate safety functions in the drive; activate **Expanded Safety Functions via PROFIsafe**, for example.
8. Then insert a new TO axis and run through the axis wizard. In the wizard, you link the axis with the corresponding drive object of the S120, and a corresponding telegram (use symbolic assignment) is created automatically.
9. Save and compile the project.
10. After configuration, you have to select the PROFIsafe telegram. In the SIMOTION SCOUT project navigator, under the drive unit double-click on **<"Drive unit_xx"> - Communication > Telegram configuration**. The telegrams appear in the working area.

11. Mark the appropriate drive in the tab **IF1: PROFIdrive PZD telegram** of the telegram overview and in the bottom part of the window under **Adapt telegram configuration** select the entry **Add PROFIsafe**. The PROFIsafe telegram is added. Dependent on the drive, you can select from several telegrams.

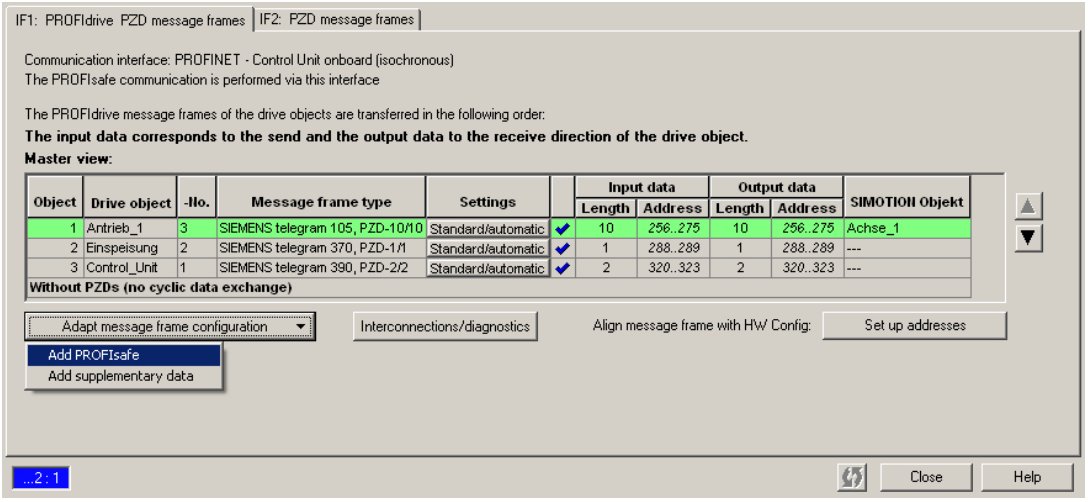


Figure 9-11 Adding PROFIsafe telegram

12. Save and compile the project.
13. Click on **Set up address** to run alignment between SIMOTION and HW Config. Configuration in SIMOTION SCOUT is completed by saving.
14. Move to HW Config and configure SYNC master and slave and an isochronous application.
15. Double-click in the SIMOTION D435 station overview on X1400 P1 and select the drive unit assignment in the **Topology** tab under **Partner port**. Click **OK** to confirm.

16. Double-click in the station on the PN IO interface to activate iDevice mode in the properties. In the **I device** tab, activate the check box **I device mode**. Click on **New...** and select the **Fail-safe I/O** entry in the dialog which opens under **Transfer area type**.

Note

In the **properties** dialog of the PN IO interface, you interconnect all the configured fail-safe I/Os directly (from STEP 7 V5.5 SP4). To do this, press the **Interconnect fail-safe I/O** button. The automatically generated transfer areas are displayed. You can then go straight on to Point 18.

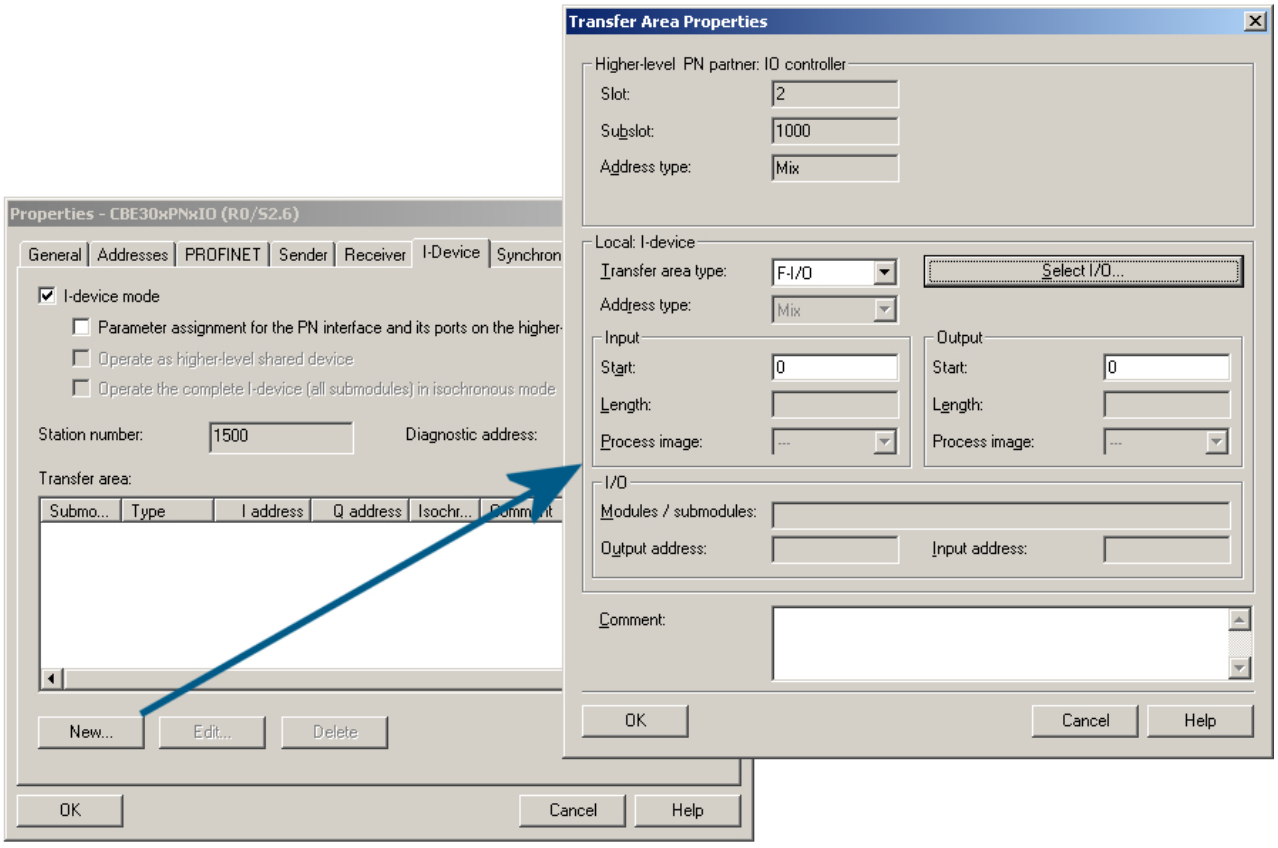


Figure 9-12 Properties of I device transfer area

17. Click on the **Select I/O** button and select the corresponding PROFIsafe channel in the dialog which opens. Confirm both dialogs with **OK**.
    This completes configuration of the lower-level drive.

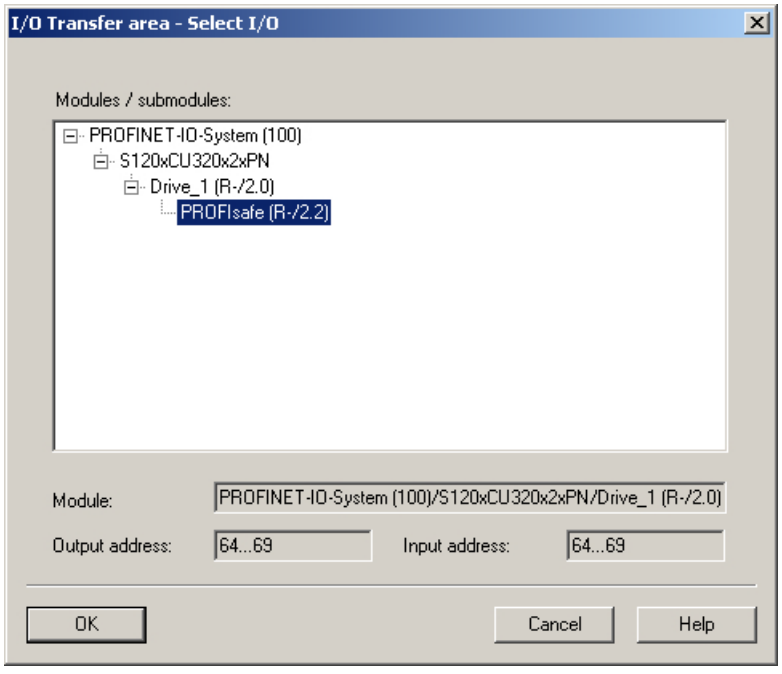


Figure 9-13 Selecting PROFIsafe channel of I/O

18. The transfer area must always contain two areas. You can either have two fail-safe I/O areas or you must define an application area for one fail-safe I/O area only. You should also refer to the iDevice chapter.

19. You can view the PROFIsafe settings for the drive unit. The detail view for the rack contains the entry **PROFIsafe**. Double-click on this to display the properties.
The **PROFIsafe** tab contains the F destination address under **F_Dest_Add**. This address must be unique within the entire project. If you are using several failsafe proxies, you must ensure that this address is only issued once. Change this value as required. The failsafe address is displayed in HW Config in the detail view in the station window under comments. This allows for clear assignment if using several participants.
The parameters F_CRC_Length=3-Byte-CRC and F_Par_Version=1 indicate PROFIsafe V2 mode. Please note these values because an I device F-Proxy configuration is only possible with this version and higher.
For more information about the fail-safe parameters, see PROFIsafe properties for configuration (Page 273)

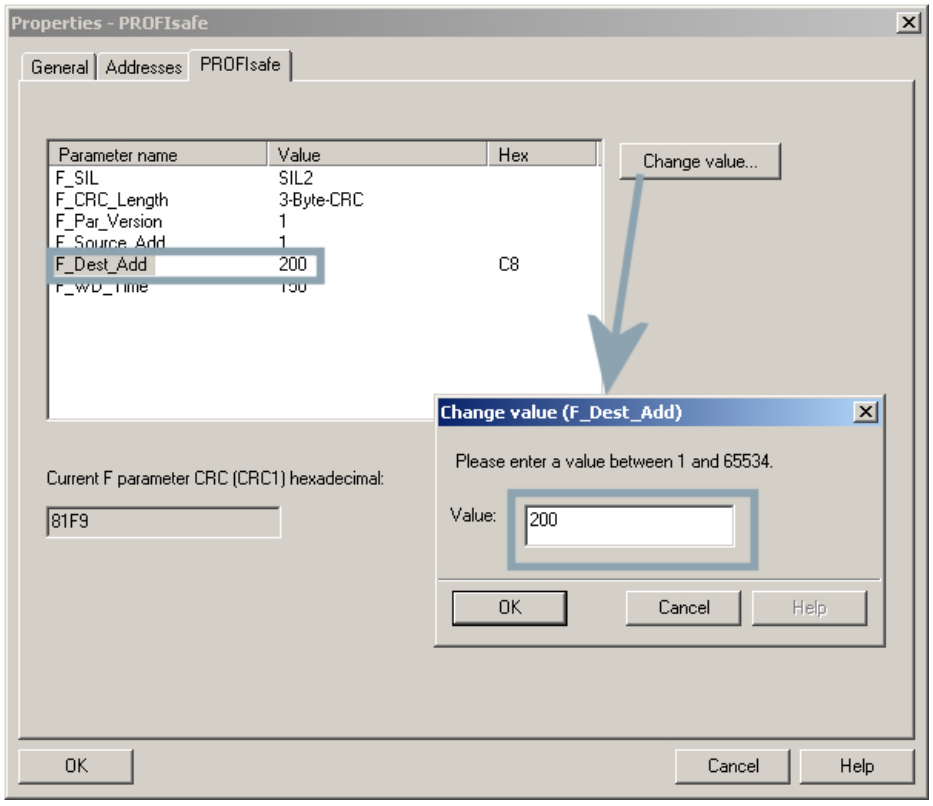


Figure 9-14 Setting the F destination address (F_Dest_Add)

20. The F destination address (F_Dest_Add) must match the PROFIsafe address during Safety configuration of the drive in SIMOTION SCOUT. In the example, this is address 200 dec or C8H. You enter the address in the Configuration window when configuring Safety Integrated. The value is stored in drive parameters p9610/p9810.

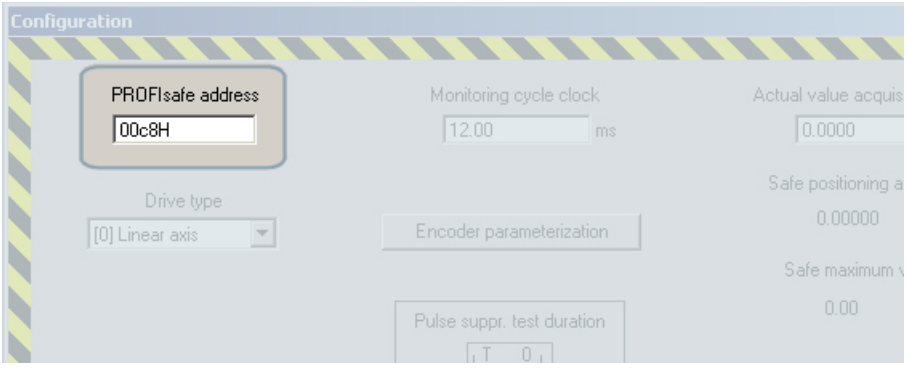


Figure 9-15 Entering F_Dest_Add as PROFIsafe address

21. Now produce the GSD file for the I device F-Proxy. In the menu, select **Options > Create GSD file for iDevice.** In the dialog which opens, click on **Create** and then **Install**. The I device is displayed under **Preconfigured Stations** in the hardware catalog.
22. You can now create a new project with F-CPU or open and use the existing project in the SIMATIC Manager. In our example, open the existing project in the SIMATIC Manager.
23. In the menu, select e.g. **Insert > Station > SIMATIC 300 Station.** Double-click on the station and then the entry **Hardware**. HW Config opens.
24. From the hardware catalog, insert e.g. an S7 300 rack if you want to select an F-CPU from the S7 300 series.

25. Insert the F-CPU, e.g. CPU317-2 PN/DP. This must at least be version V3.2.
26. Use drag&drop to move the I device previously created under **Preconfigured Stations** to the PROFINET IO network. Once saved and compiled, configuration is complete. Please observe the rules regarding device names in the chapter PROFINET IO and I device (Page 176).

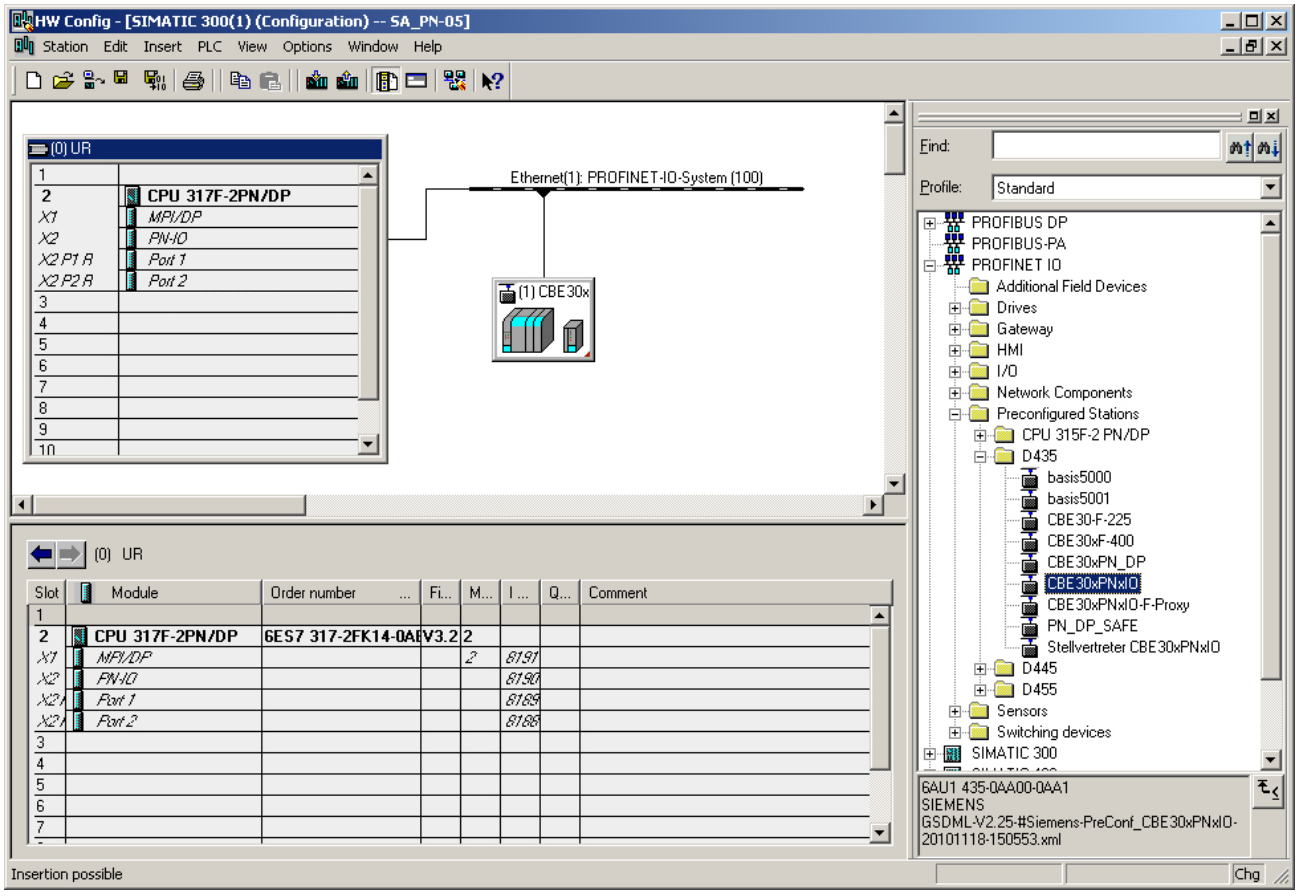


Figure 9-16 Master project with an I device failsafe proxy submodule

9.5.6.3 Adapting the F destination address (F_Dest_Add) in the existing project

Adapting the F destination address (F_Dest_Add) for the entire project

In an existing project, you can check whether the F destination address (F_Dest_Add) for the iDevice F-Proxy has been set correctly at a later point. The F destination address (F_Dest_Add) must be the same at the following locations:

- PROFIsafe slot of the drive on the SIMOTION CPU (HW Config)
- PROFIsafe slot of the I device for the SIMOTION CPU (HW Config)
- Safety configuration for drive in SIMOTION (SIMOTION SCOUT)

If you want to change the F destination address (F_Dest_Add) at a later point without having to create and install the iDevice again, you must make the change at the three locations referred to above.

In the example below, you set the F destination address (F_Dest_Add) to the value 300 (12CHex) for a SIMOTION CPU with a drive on PROFINET and an F-CPU in the same project.

How to check whether the F destination address (F_Dest_Add) is identical

1. Open the SIMOTION CPU project in HW Config.
2. In the detail view of the drive unit, double-click **PROFIsafe**. In the window that opens, switch to the **PROFIsafe** tab. The value 300 must be present next to **F_Dest_Add**. To change **F_Dest_Add**, click the **Change Value** button and enter 300 in the dialog box that appears.

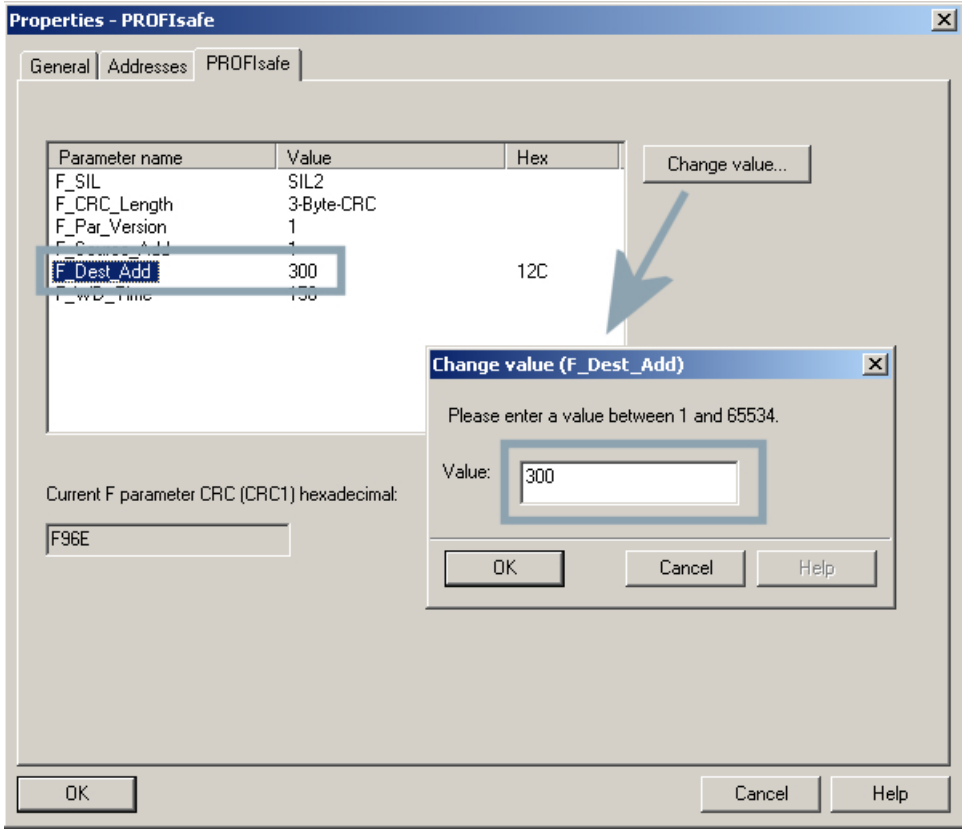


Figure 9-17 F destination address (F_Dest_Add) for PROFIsafe slot on drive unit

3. Confirm by selecting **OK** and save and compile the project.
4. Open the project with the F-CPU in HW Config.

5. In the detail view of the I device, double-click the **PROFIsafe I/O**, e.g. B. 6I/6O F-Periphery. In the window that opens, switch to the **PROFIsafe** tab. The value 300 must be present next to **F_Dest_Add**. To change **F_Dest_Add**, click the **Change Value** button and enter 300 in the dialog box that appears.

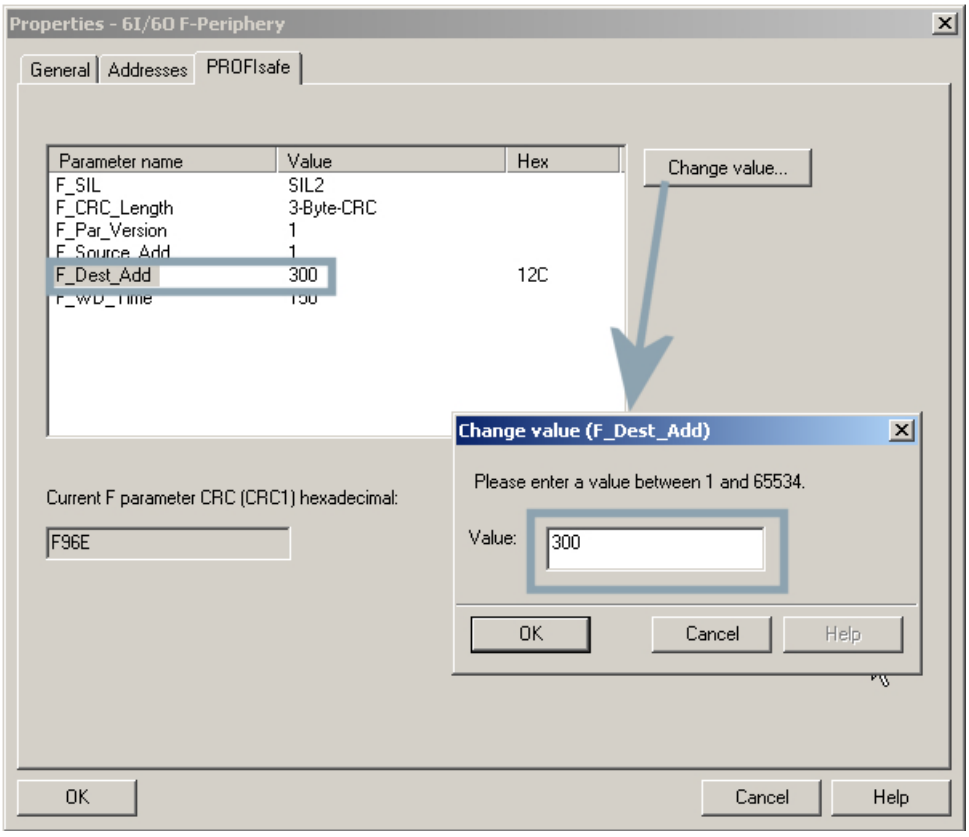


Figure 9-18 F destination address (F_Dest_Add) for iDevice on F-CPU

6. Confirm by selecting **OK** and save and compile the project.
7. Open the SIMOTION project in SIMOTION SCOUT.
8. In the project navigator, navigate to the drive (e.g. D435 > S120xCU320xCBE20 > Drives > Drive_1).
9. Under Functions in the project navigator, double-click **Safety Integrated**.
10. Click the **Configuration** button. The Configuration window appears.

11. Under **PROFIsafe address**, check the value 12CHex (300) and change it if necessary.

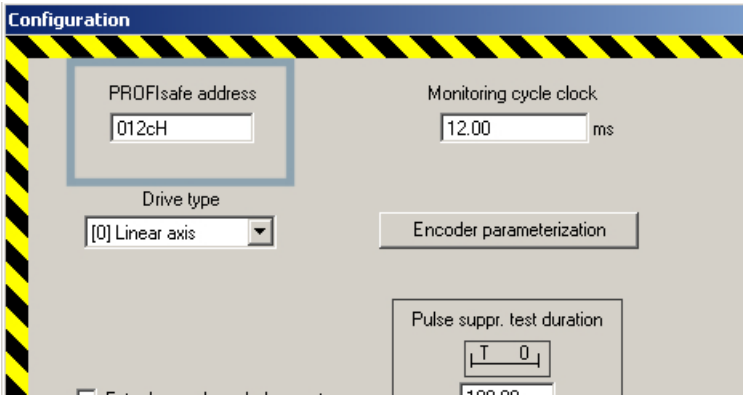


Figure 9-19 F destination address (F_Dest_Add) for configuring Safety Integrated

12. Confirm by selecting **Close** and save and compile the project.

9.5.6.4 Configuration of D435 with S120 on PROFINET and integrated PROFIBUS

Integrated drive on D435 for PROFIsafe

In the previous example you configured an S120 on D435 with PROFINET and imported it as an I-device. A SINAMICS_Integrated is now added to the project using the internal PROFIBUS DP.

This is how you configure a SINAMICS_Integrated

You have configured a project with D435 and SINAMICS S120 and imported it as an I-device to an F-CPU.

1. In SIMOTION SCOUT, configure the drive unit on SINAMICS_Integrated and insert a PROFIsafe telegram (see points 7-11 in the example Configuration example for SIMOTION D435 and SINAMICS S120 via PROFINET (Page 287)).
2. In HW Config highlight SINAMICS_Integrated and double-click on the PROFIsafe module in the rack's detail view.
3. In the **Configuration** tab of the dialog which opens, highlight **PROFIsafe telegram** and click on **PROFIsafe...** . In the **PROFIsafe properties** dialog box, you can see and change the F Parameters.
   (If the **PROFIsafe...** button is not shown, you first have to click on the **Activate** button to make changes.)

4. Double-click on the PN IO interface of the SIMOTION CPU. In the **I-Device** tab in the dialog box which opens, select the **I-device mode** check box. Click on **New...** and select the **Fail-safe I/O** entry in the dialog which opens under **Transfer area type**. Click on the **Select I/O** button and select the corresponding PROFIsafe channel in the dialog which opens.

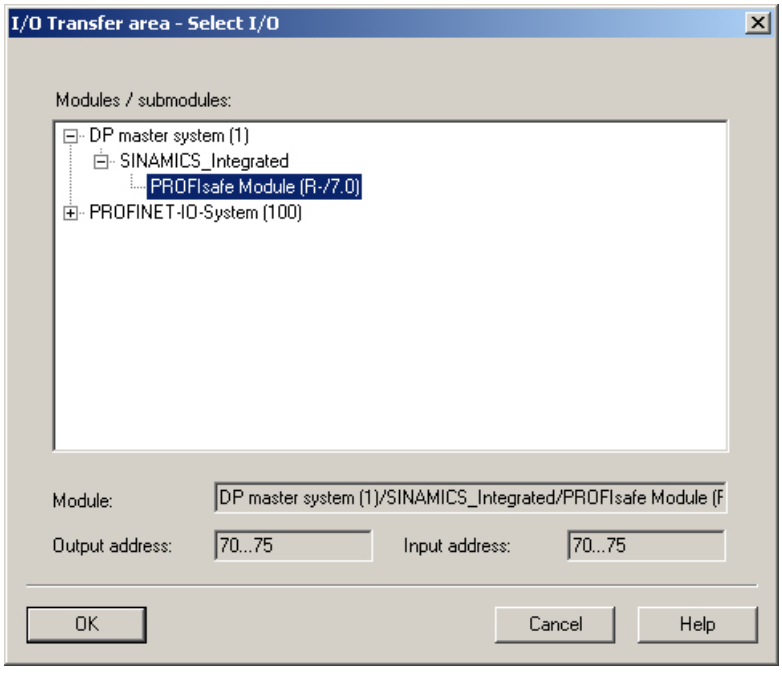


Figure 9-20 Releasing PROFIsafe module for SINAMICS_Integrated for transfer area

5. Click **OK** to confirm the dialog box. In the **Transfer Area Properties** dialog box, the inputs/outputs are assigned and an automatically generated comment is displayed. This comment includes, among other things, the subslot, the SIMOTION device name, the connection, and the device name of the drive. The F_Dest_Add is at the end. You can change the comment if necessary.

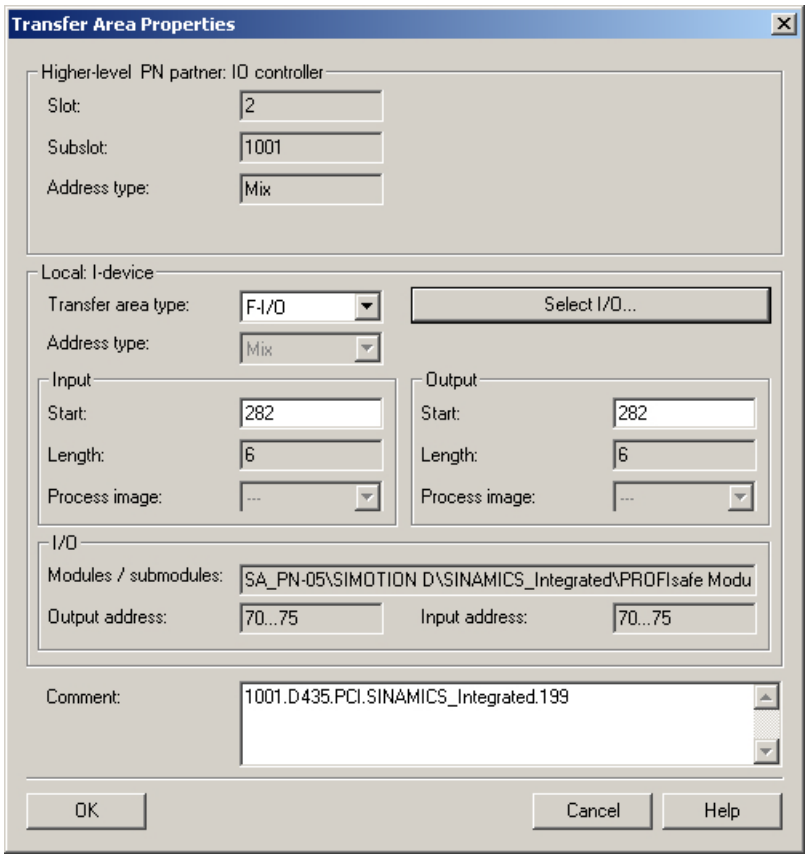


Figure 9-21 Properties of I-device transfer area, comment

6. Click **OK** to confirm this dialog box. The failsafe data for the two drives is displayed in the transfer area.
   This concludes configuration of the lower-level drive. All you need to do now is save and compile the station.

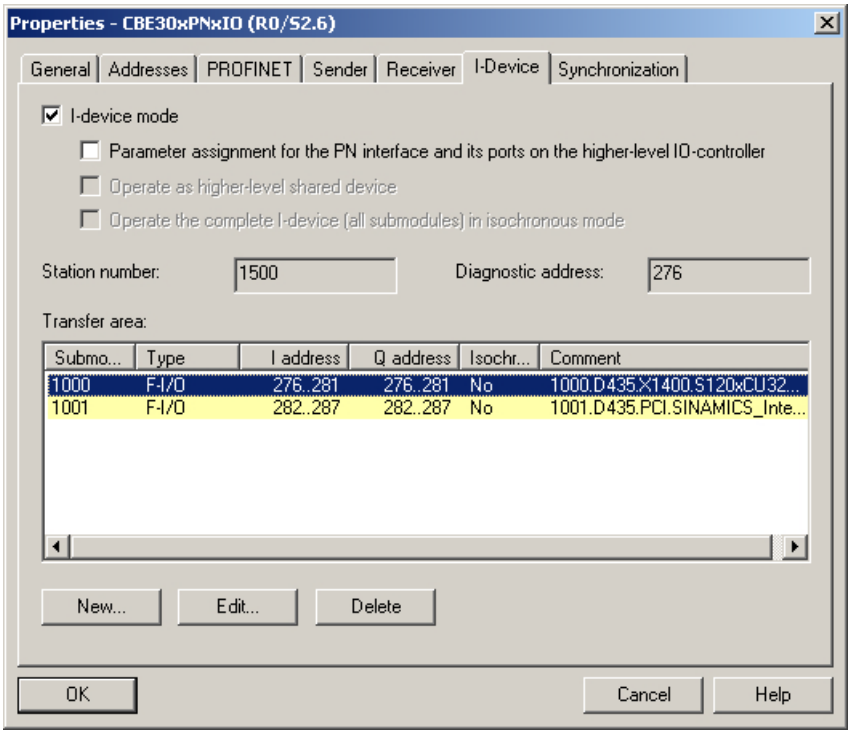


Figure 9-22 I-device transfer area

7. Now produce the GSD file for the I-device F-Proxy. In the menu, select **Options > Create GSD file for I-device**. In the dialog which opens, click on **Create** and then **Install**. The I-device is displayed under Preconfigured stations in the hardware catalog.
8. As in the previous example, create an F-CPU in the project and add the I-device of the SIMOTION module.
   The diagram shows the project with F-CPU and I-device F-Proxy with one drive on PROFINET and one drive on SINAMICS_Integrated on a D435.

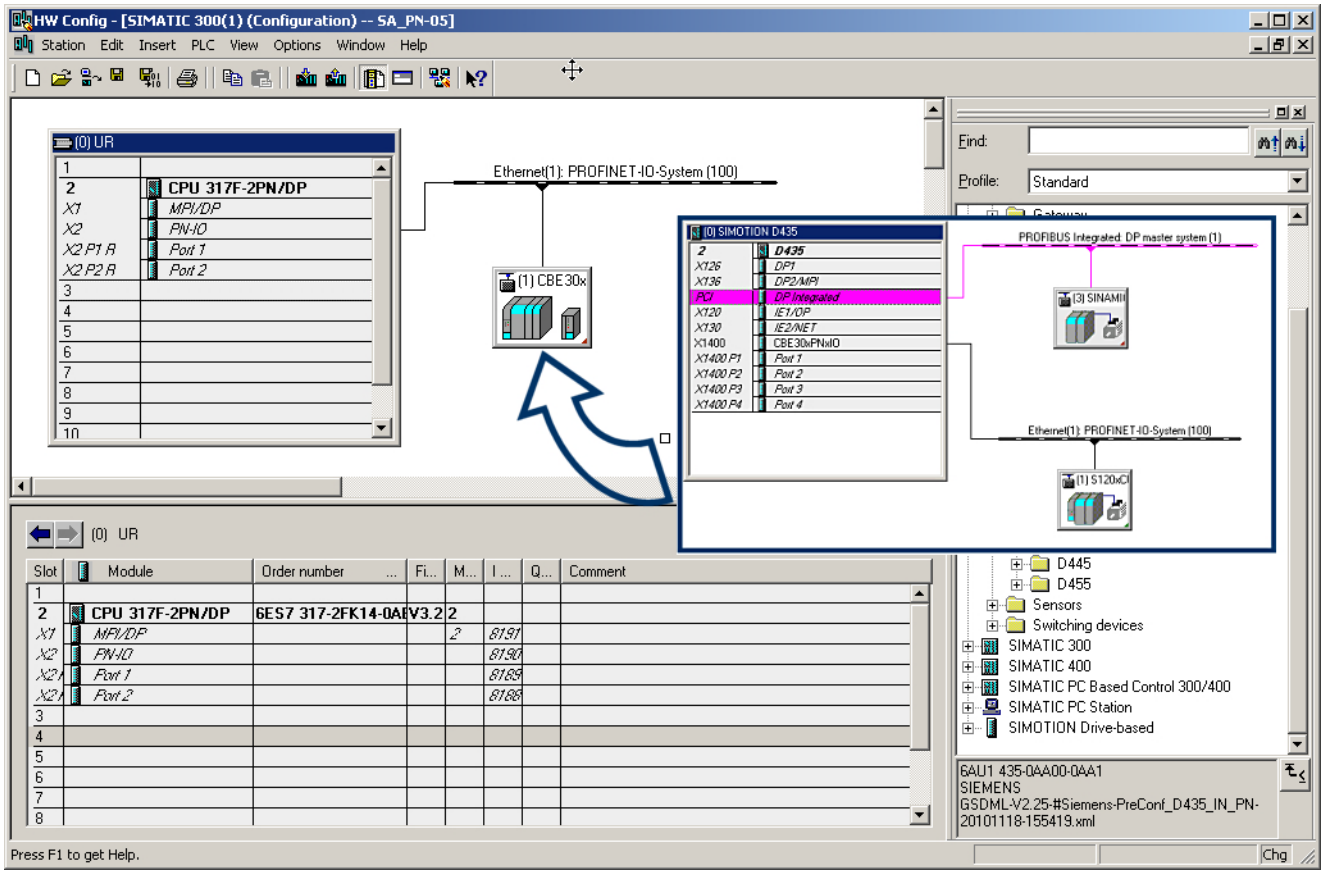


Figure 9-23 F-CPU with I-device F-proxy on PROFINET and integrated PROFIBUS

9.5.6.5 Upgrading an existing system with PROFIsafe via PROFIBUS to PROFIsafe via PROFINET

PROFIBUS to PROFINET

If PROFIsafe communication on an existing system has been operated via PROFIBUS up until now and you want to switch it to PROFINET, then you will need to upgrade the system.

How to carry out the upgrade

1. Delete the old I slave link.
2. If necessary, switch the DP interface from DP slave to DP master (SIMOTION CPU).

3. The PROFIsafe message frame configuration settings for the drives can remain unchanged.
4. If necessary, create the F parameters **F_Par_Version = 1** and **F_CRC_Length = 3-Byte-CRC** in order to use the PROFIsafe V2 standard. This will mean that a link cannot be established in the I device. V2 is automatically selected when the new PROFIsafe slots are created.

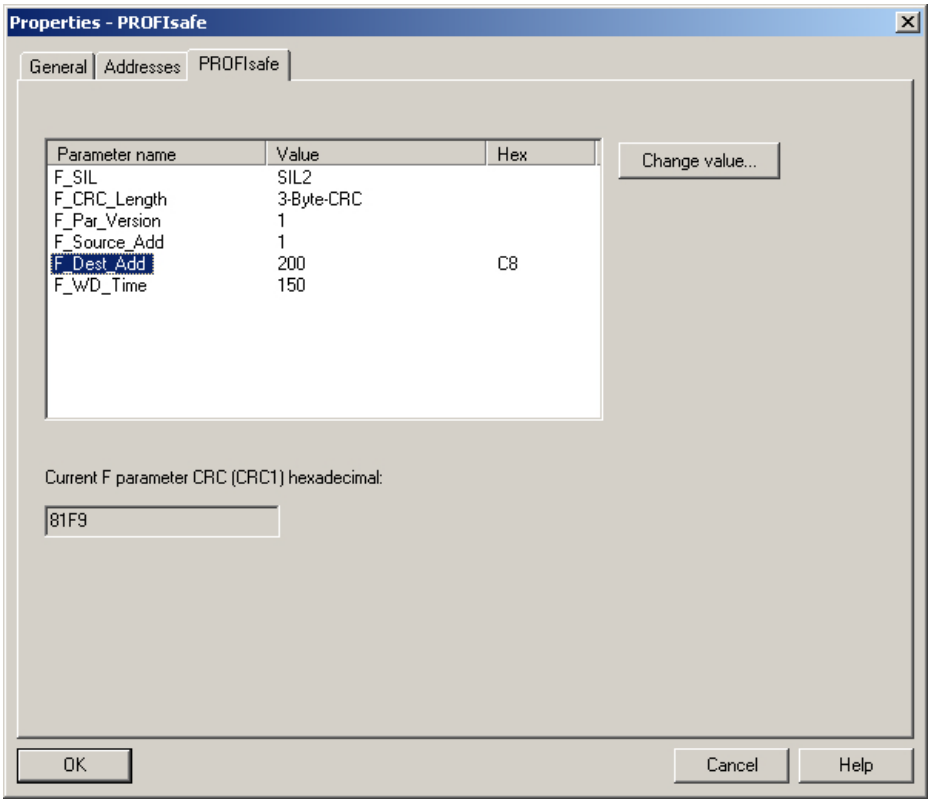


Figure 9-24 Properties of PROFIsafe taking the example of I device failsafe proxy

5. Select I device mode in HW Config on the CBE30.
6. Select **New** , followed by **failsafe periphery** under transfer area type to select the I/O and create a submodule.
7. Then create a new GSD via **Options > Create GSD file for I device...** and install it.
8. Link the GSD file to the S7 F-CPU.

9.5.6.6 General information on F destination addresses (F_Dest_Add) with iDevice F-Proxy

Communication addresses for F source (F_Source_Add) or F destination address (F_Dest_Add)

Create new F hosts, F modules/submodules in HW Config. HW Config then suggests the F source/destination address (F_Dest_Add) as a default setting. You can change or overwrite this default setting. The default failsafe addressing is based on the failsafe start address parameter on the F-CPU and also on the SIMOTION CPU if applicable.

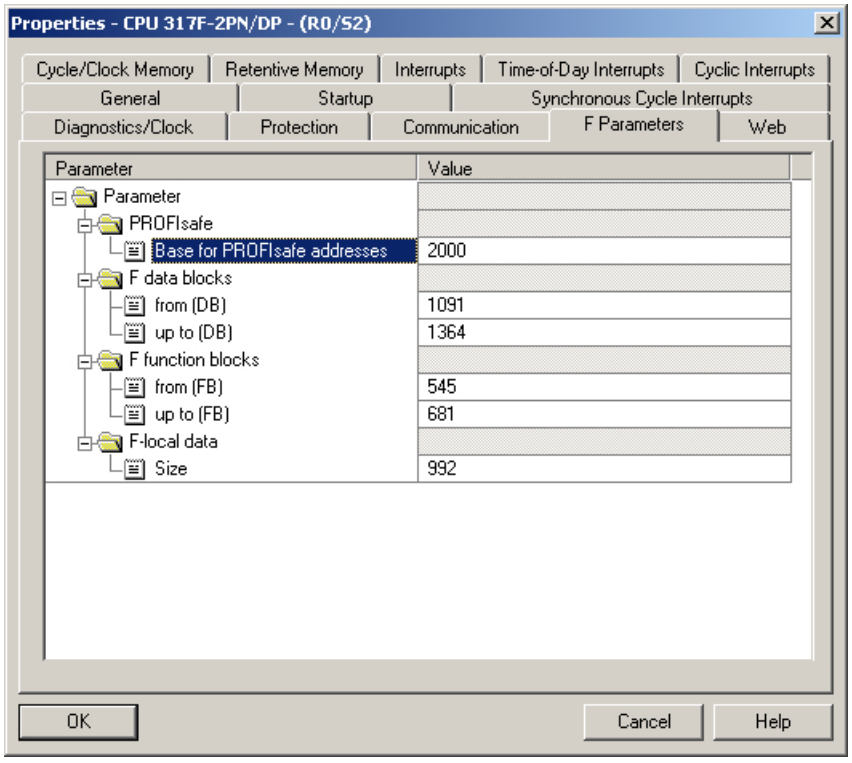


Figure 9-25 PROFIsafe start address

The F source address (F_Source_Add) is assigned in the same way as for Siemens F modules with OM:

- PROFIsafe start address of CPU + number of DP master system for PROFIBUS
- PROFIsafe start address of CPU for PROFINET
- 1, if the CPU does not have a PROFIsafe start address parameter (standard CPU and F-H-CPU)

Guidelines for addressing

- The full range of values between 1...65534 is used for addressing.
- The F destination address (F_Dest_Add) is assigned automatically.
- Automatic issuing takes place when plugging in as with the other failsafe modules in Step7: starting with an start value and working upwards, the system looks for the next free address.

- The start value is the PROFIsafe start address of the CPU/10 (or 1022, if there is no PROFIsafe start address or if it is greater than 10000).
- If the F start address is changed (CPU parameter), then F source and F destination addresses (F_Dest_Add) that have already been issued are not updated (remain in place). The change only affects the default address of newly created submodules.
- Several F-Source addresses may be issued in an F-CPU. The F source address and F destination address (F_Dest_Add) are included in the PROFIsafe CRC total.
- If addresses from an issued range of addresses are no longer used (when failsafe submodules are subsequently deleted), the gaps are filled first when new addresses are issued.

Note

If you want to use Safety to work beyond the project limits/proxies, you must ensure that the F source and F destination addresses are unique within the entire project. Before starting configuration, you should have/produce a failsafe addresses plan where unique ranges of failsafe addresses are assigned to particular sub-projects.

9.5.7 Shared device via PROFINET

9.5.7.1 General information on shared device

Description

With the shared device functionality, you can configure access to an IO device with several IO controllers using PROFINET. This enables channels/modules to be flexibly assigned to different IO controllers. This option is available for inputs and outputs. You can use this mechanism to access the failsafe data of a drive configured below a SIMOTION CPU via the F-CPU, for example.

Note

When configuring PROFIsafe, it is recommended that you use the I device F-Proxy instead of the Shared Device function.

Schematic diagram of Shared Device

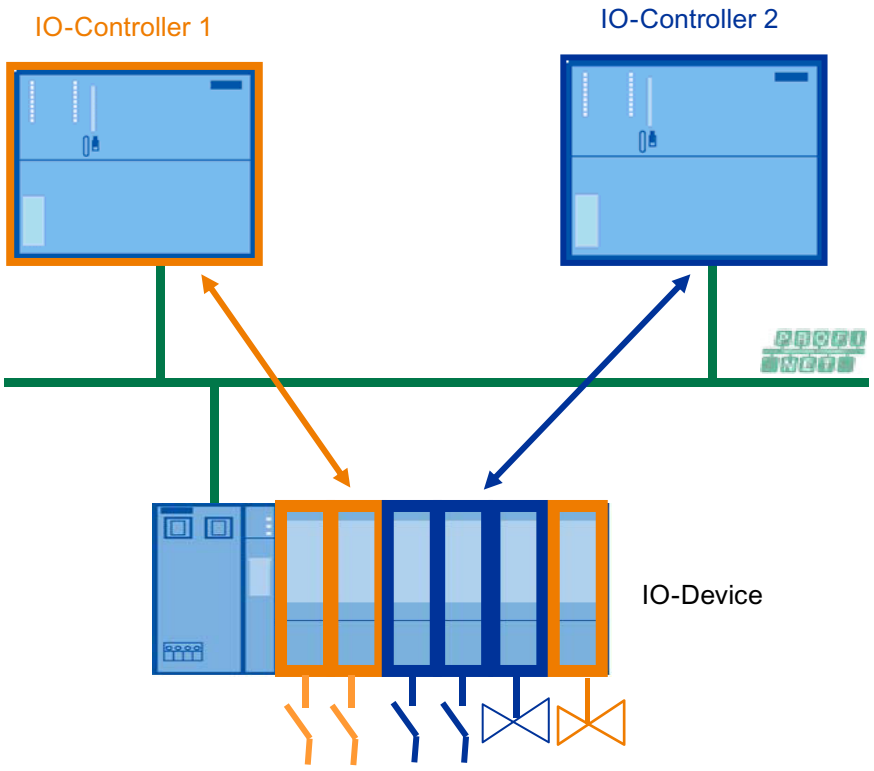


Figure 9-26 Schematic diagram of Shared Device

Software requirements

- SIMATIC Step 7 V5.5 or higher
- SINAMICS firmware and Support Packages V4.3.2 or higher
- SIMOTION SCOUT V4.2 or higher
- S7 F Configuration Pack, V5.5 SP7 or higher (if using Safety/PROFIsafe)
- GSDML file V2.25 or higher

Configuration principles

- I/O addresses can be assigned as usual for the submodules (that are assigned to the controller).
- A shared device must have the same IP parameters and the same device name in each station. There are two types of configuration case:
  - Shared device in the same project: STEP 7 takes important consistency check functions away from the user. STEP 7 checks that the IP parameters have been assigned correctly and monitors the IO controller access to the individual submodules to ensure that it is correct.
  - Shared device in different projects: The stations with the IO controllers that use the shared device are created in different projects. It is important to ensure in each project that the shared device is configured in exactly the same way in each station. Only one IO controller may have full access to a submodule in any case (see below). The IP parameters and device names must be identical. Inconsistencies in the configuration will cause the shared device to fail.

See also

Shared device (Page 63)

9.5.7.2 Shared device in a STEP 7 project

Introduction

In the following example, the simplest configuration of a shared device is described: Two IO controllers (SIMOTION D445-2 DP/PN and CPU 317F-2 PN/DP) share the submodules of an IO device (ET200S HF). The two IO controllers are in the same STEP 7 project with the advantage that the consistency check is made automatically.

Procedure

To be able to use the shared device function, you need to take certain configuration steps in SIMOTION SCOUT, SIMATIC Manager, and HW Config.

Preparatory steps for SIMOTION CPU

1. In SIMOTION SCOUT, create a project called **Shared device project**.
2. Insert a SIMOTION D445-2 DP/PN and configure it.
3. Open the SIMOTION CPU in HW Config and configure the PROFINET interface.

4. Configure a PROFINET IO device ET 200S (IM151-3PN HF) with several submodules as shown in the figure below.

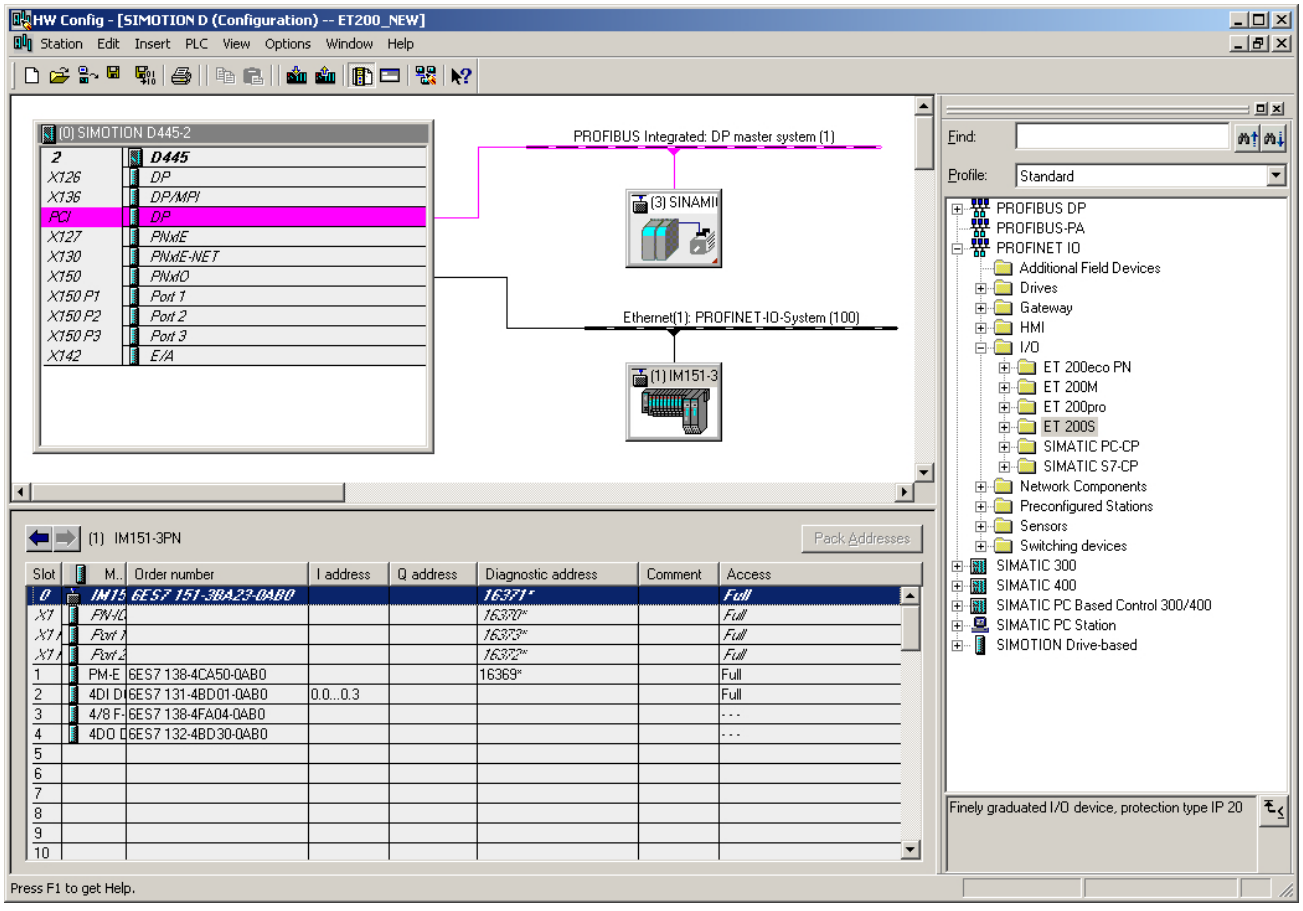


Figure 9-27 SIMOTION D445-2 DP/PN with IO device ET200S

5. **Save and compile** in HW Config.

Preparatory steps for SIMATIC CPU

1. Open the project you created in the SIMATIC Manager.
2. Insert a SIMATIC 300 station and open it in HW Config.
3. Insert a CPU 317F-2 PN/DP, for example, and configure the PROFINET interface.
4. **Save and compile** in HW Config.

Creating the shared device

1. Open one of the SIMOTION CPUs you created in HW Config.
2. **Copy** the IO device ET200S you created using the context menu (right mouse button).
3. **Save** the hardware configuration and close the configured station.
4. Open the station you created previously with the SIMATIC F-CPU in HW Config.

5. In order to add the IO device as a shared device, right click on the PROFINET IO system. Select the context menu command **Paste shared**.

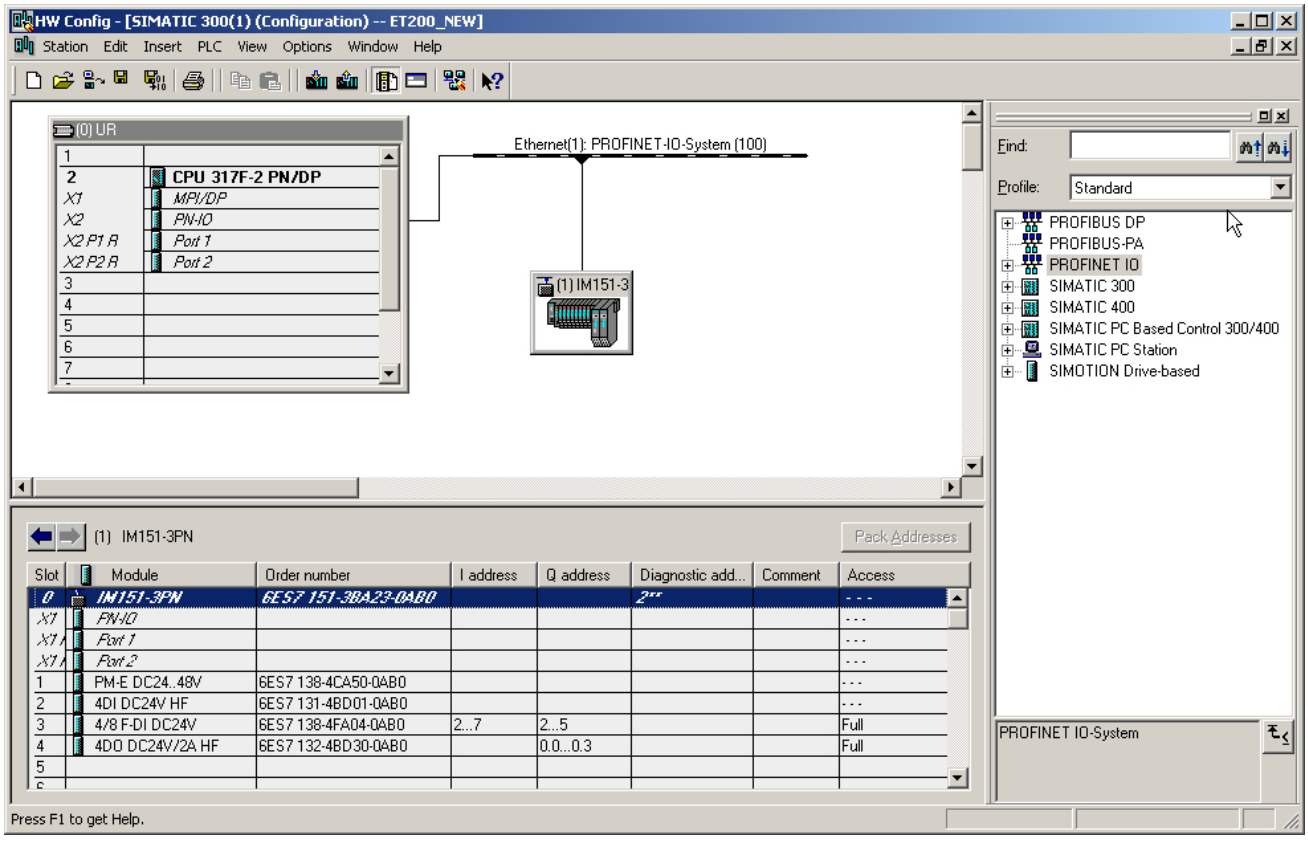


Figure 9-28 SIMATIC CPU 317F-2 PN/DP with shared device ET200S

6. Save the hardware configuration and close the configured station.

You have successfully created the shared device, now set the assignments of the submodules to the configured stations.

Assigning submodules

The submodules must be assigned separately for each station. Remember that changes to a station will also impact the other station(s)! A submodule can only ever be assigned to one station!

1. Open the Properties dialog box of the PROFINET IO device for the SIMOTION CPU.
2. Click the **Access** tab.

3. Configure the access to the individual submodules. To do this, select the type of access from the drop-down list in the **Value** column. You can select from the following:
   - No access to the submodule: "- - -"
   - Full access to the submodule: "full"

   Note that the setting "Full" automatically leads to the setting "- - -" in the other station(s).

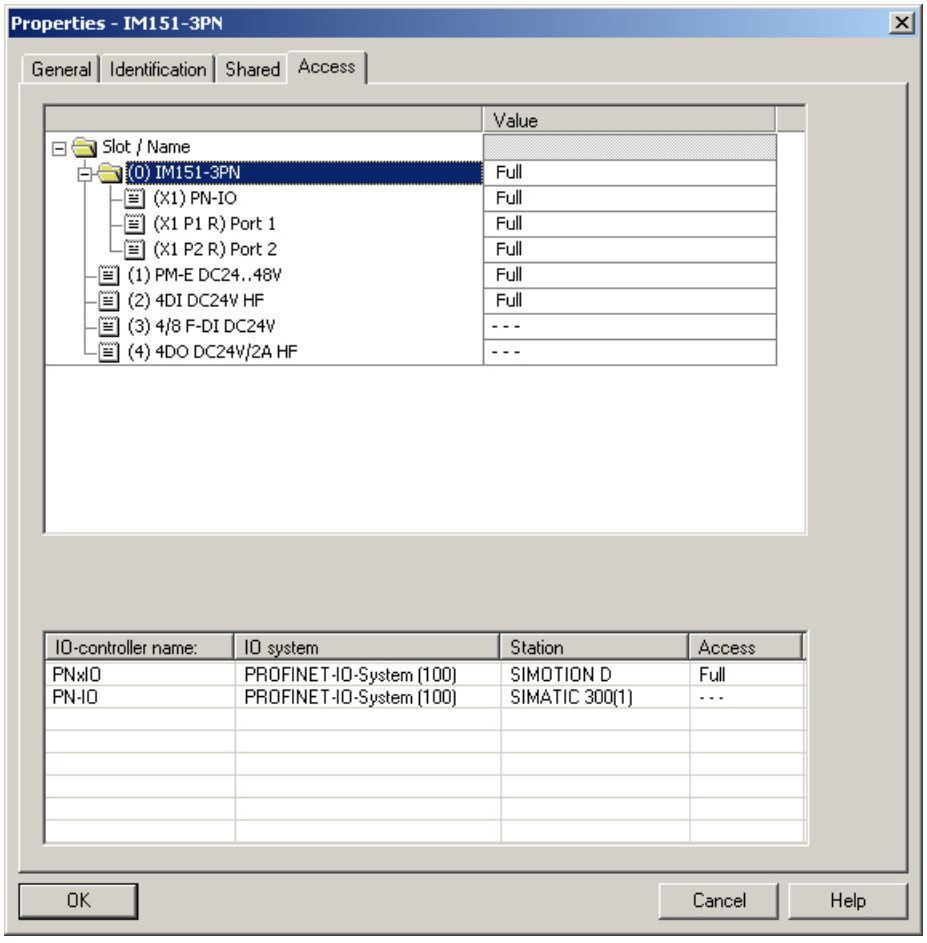


Figure 9-29 SIMOTION D445-2 DP/PN access to ET200S

4. Save and compile the station and close it.

5. Repeat steps 1 to 4 for the shared device on the SIMATIC F-CPU.

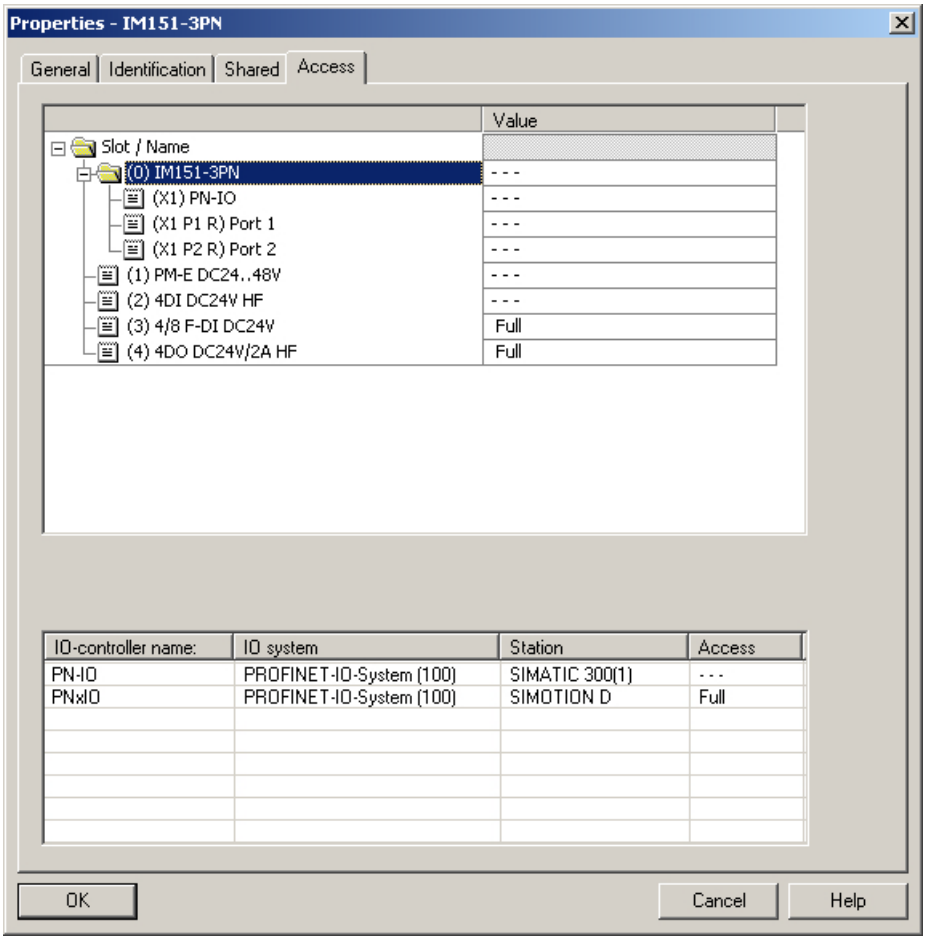


Figure 9-30 SIMATIC CPU 317F-2 PN/DP access to shared device ET200S

6. Then download the configuration to the stations.

Shared device in the user program

The shared device has no special role in the user program. The submodules assigned in the station are addressed as usual through your addresses, the other submodules do not receive addresses.

9.5.8 Shared iDevice and PROFIsafe

9.5.8.1 Using a shared iDevice with PROFIsafe

Introduction

An iDevice can be operated on two higher-level IO controllers as an IO device. This functionality is known as the shared iDevice. It provides the following application areas:

- One of the higher-level controllers works as as the "automation CPU" and uses the SIMOTION iDevice as an isochronous iDevice in the IRT operating mode.
- The other higher-level IO controller works as the "safety CPU" and uses the submodules configured as the F-Proxy in the RT operating mode.

This means that an iDevice can communicate with two higher-level controllers as an IO device. At the same time, higher-level controllers can access certain modules, e.g. an F-CPU can access the F-Proxy submodules, using the shared device.

The schematic representation below shows two IO controllers which share the submodules on the iDevice of a third IO controller.

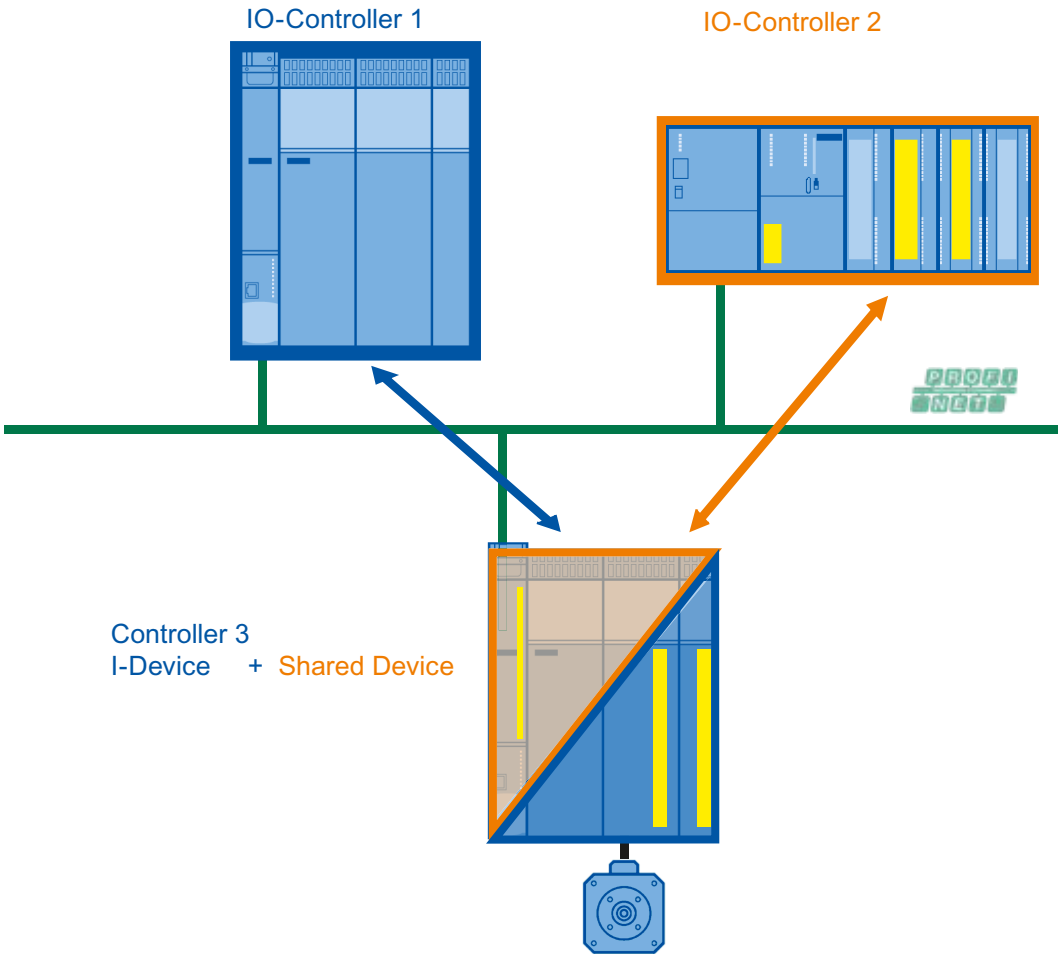


Figure 9-31 iDevice as a shared device

Note

In this configuration, only the submodules that are used by the "automation CPU" are configured as isochronous (IRT).

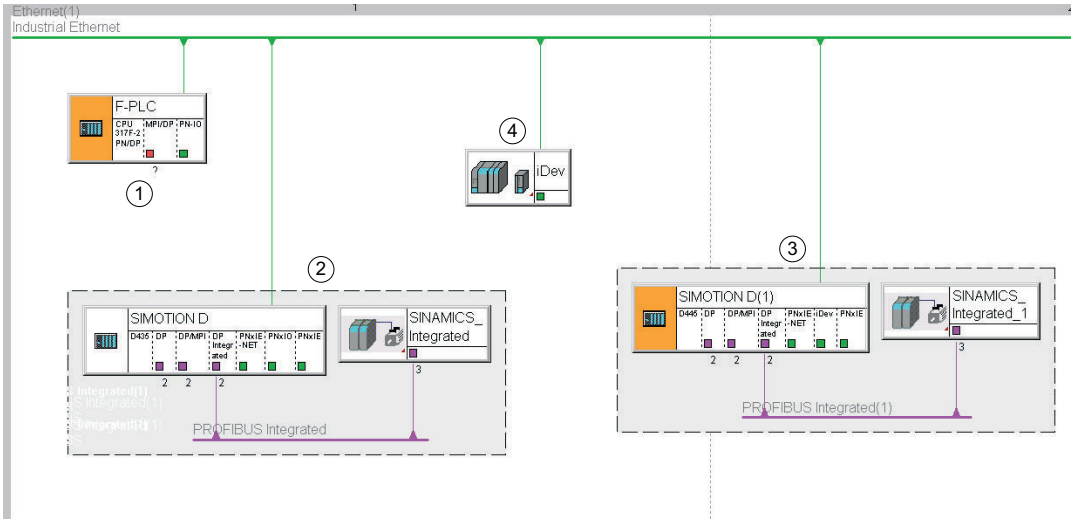
Basic information on this subject can be found in the chapter Shared iDevice (Page 194).

9.5.8.2 Configuration example for shared iDevice and PROFIsafe

Configuration example

In the following example, a SIMOTION D445-2 DP/PN with two Safety drives on SINAMICS_Integrated is operated as a shared iDevice on a SIMOTION D435-2 DP/PN and a SIMATIC CPU317F-2. The SIMOTION D435-2 takes on the motion control tasks, while the SIMATIC CPU317F-2 is responsible for safety monitoring.

The overview diagram below shows the network view. The IO controllers are in the same subnet and are located in a project.



1 SIMATIC CPU 317F-2 as higher-level IO controller in RT mode (F-CPU)

2 SIMOTION D435-2 DP/PN as higher-level IO controller in IRT mode

3 SIMOTION D445-2 DP/PN with two Safety drives on SINAMICS_Integrated (PROFIBUS)

4 iDevice of SIMOTION D445-2 DP/PN which is connected to both higher-level IO controllers (shared iDevice).

Note:
In this example, the SIMOTION D445-2 DP/PN is shown twice in Netpro: Once as an iDevice proxy and once as a complete SIMOTION D device.

Figure 9-32 Configuration example for PROFIsafe via shared iDevice, representation in Netpro

Basic process for the example configuration

The following configuration steps must be observed for the example configuration:

1. Create a new project in SIMOTION SCOUT and configure a SIMOTION D435-2 DP/PN and a SIMOTION D445-2 DP/PN. These must be configured on the same subnet.
2. Configure two drives (power unit, supply, Safety functions, F destination address F_Dest_Add) on the SINAMICS_Integrated of the SIMOTION D445-2 DP/PN. (See also Configuring the iDevice F-Proxy (Page 287))
3. Create two axes and connect each of them to one of the two drives in the axis configuration.

4. Insert the PROFIsafe telegrams in the telegram configuration for the drive unit and align the telegrams with HW Config.
   The diagram below shows an example project with the PROFIsafe telegram configuration open.

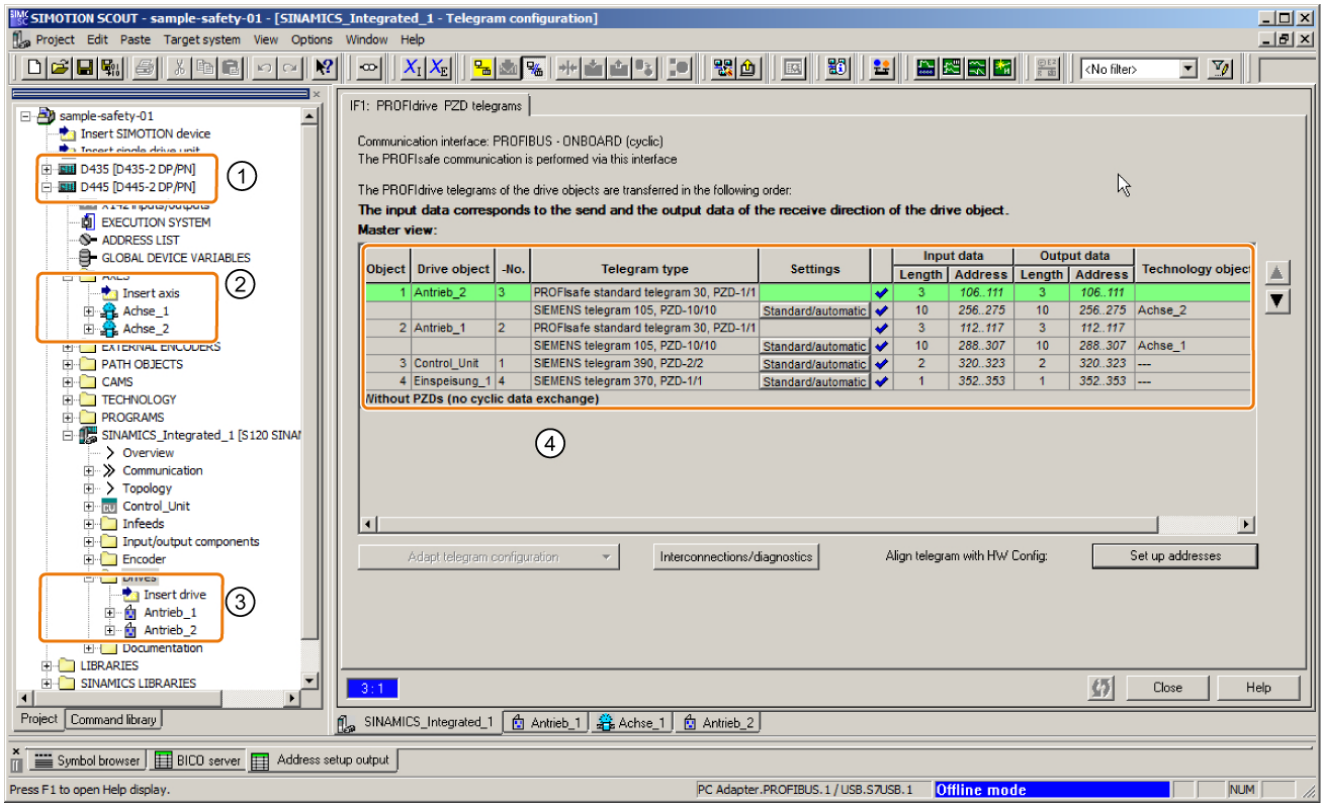


Figure 9-33 Example project for shared iDevice with two SIMOTION D

5. In the project navigator, double-click on a SIMOTION CPU, e.g. D445-2 DP/PN, to open HW Config. Insert a new SIMATIC 300 station with a SIMATIC CPU317F-2 in the project. The F-CPU is located in the same subnet as the SIMOTION CPUs.
6. In HW Config for the SIMOTION D445-2 DP/PN, configure the iDevice (i.e. the iDevice mode and transfer areas) and the isochronous mode. Export and install the iDevice (see also Configuring SIMOTION D for shared iDevice (Page 314)).
7. When HW Config for the SIMATIC CPU317F-2 is open, add the iDevice to its PROFINET IO system. Copy the iDevice and use "Shared paste" to insert it into the PROFINET IO system of the SIMOTION D435-2 DP/PN. Configure the access to the submodules of the iDevice and the synchronization roles (see also Allocating an iDevice to two IO controllers as a shared device (Page 316)).
8. **Save and compile** the stations/project.

More details about the configuration process can be found in the following chapters.

9.5.8.3 Configuring SIMOTION D for shared iDevice

Requirement

A SIMOTION D445-2 DP/PN with two drives on SINAMICS_Integrated and two axes with safety is configured in the example project. The SIMOTION D435-2 DP/PN and the SIMATIC CPU317F-2 are configured in the project as described in the previous step. In the next step, the SIMOTION D445-2 DP/PN will be configured as an I-device.

Configuring and creating a shared I-device

How to configure the SIMOTION D445-2 DP/PN for the creation of the I-device:

1. In the SIMOTION SCOUT project navigator, double-click on the SIMOTION D445-2 DP/PN to open HW Config.
2. Double-click on the **PN interface** (X150) and assign a new **device name** in the Properties dialog box in the **General** tab, e.g. I-Device-Shared.
3. Switch to the **I-Device** tab. Activate the **I-device** option and the additional options **Parameter assignment for the PN interface and its ports on the higher-level IO controller** and **Operate as higher-level shared device.**

4. In this step, you configure the transfer areas (submodules) which are to be shared with the higher-level controllers. Two fail-safe I/Os and two application areas are used in each case.
   - Click **Interconnect fail-safe I/O**. The configured transfer areas are automatically set up for the fail-safe I/O. In the example, two PROFIsafe drives are used. These are **not** operated **isochronously**.
   - Click **New...** and configure the transfer areas for the application area. Two application areas are configured which are operated **isochronously**. The addresses for the transfer areas are suggested by STEP 7. You can also define your own address areas.

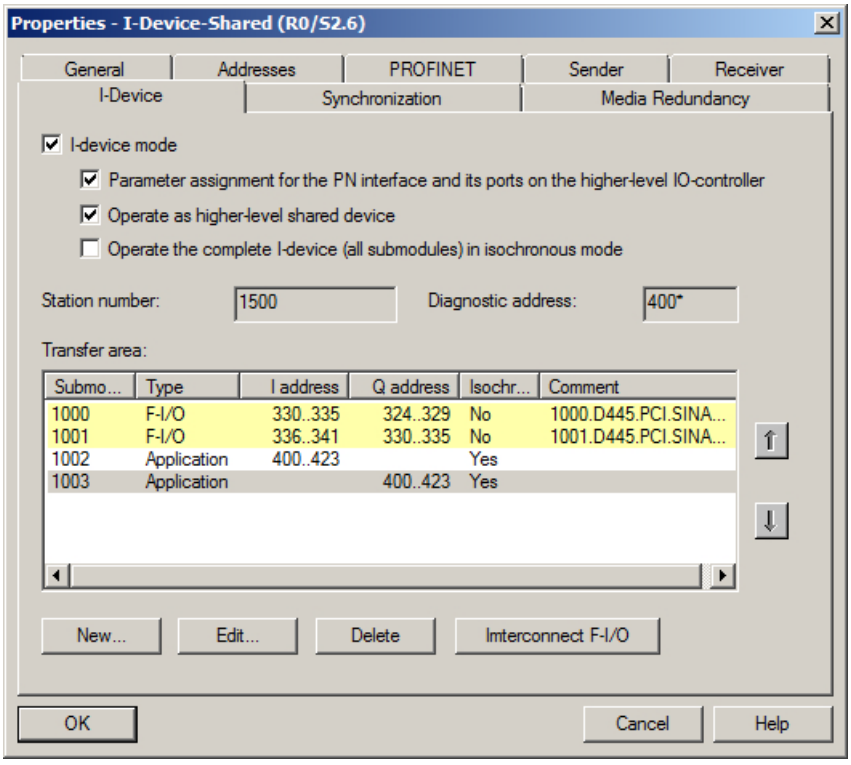


Figure 9-34 Shared I-device transfer areas

5. Activate the option **Use I/O data isochronously to the Servo** for the PN interface in the **Isochronous tasks** tab in the **Properties** of the SIMOTION D445-2 DP/PN and click **OK** to confirm. The I-device is used as a reference clock.
6. In the **PROFINET IO isochronous operation** of the SIMOTION D445-2 DP/PN, select servo as the **Isochronous application**.
7. Select the menu command **Station > Save and Compile**.
8. In the menu, select **Options > Create GSD file for I-device**.
9. Enter a **name** and **comment** for the I-device in the dialog box that appears.
10. Click **Create** and then **Install**. The I-device is displayed under **PROFINET IO > Preconfigured Stations > D445** in the hardware catalog.

In the next step, you will add the I-device to the higher-level IO controllers and configure the access to the transfer areas.

9.5.8.4 Allocating an iDevice to two IO controllers as a shared device

Requirement

The example project has been completely configured and the GSD file has been created for the iDevice. In the next step, the shared iDevice is configured on the two IO controllers, SIMOTION D435-2 DP/PN and SIMATIC CPU317F-2.

Inserting and configuring a shared iDevice

To configure the shared iDevice, proceed as follows:

1. Open the configured station SIMATIC CPU317F-2 in HW Config.
2. Drag&drop the created iDevice from the hardware catalog (**PROFINET IO > Preconfigured Stations > ...)** to the PN IO system of the SIMATIC CPU. The iDevice is inserted.
3. Double-click on the inserted iDevice. The Properties window is displayed.

Note

The first time you insert a device, you must set up the dialog for the safety password. Enter a password and confirm the dialog. This password will be requested whenever safety-relevant functions are changed.

4. In the **General** tab, enter the **device name** that you assigned earlier on the PN interface of the D445-2 DP/PN before exporting the iDevice (in the example, **iDevice shared**). The device names must be identical in order for communication to be possible.

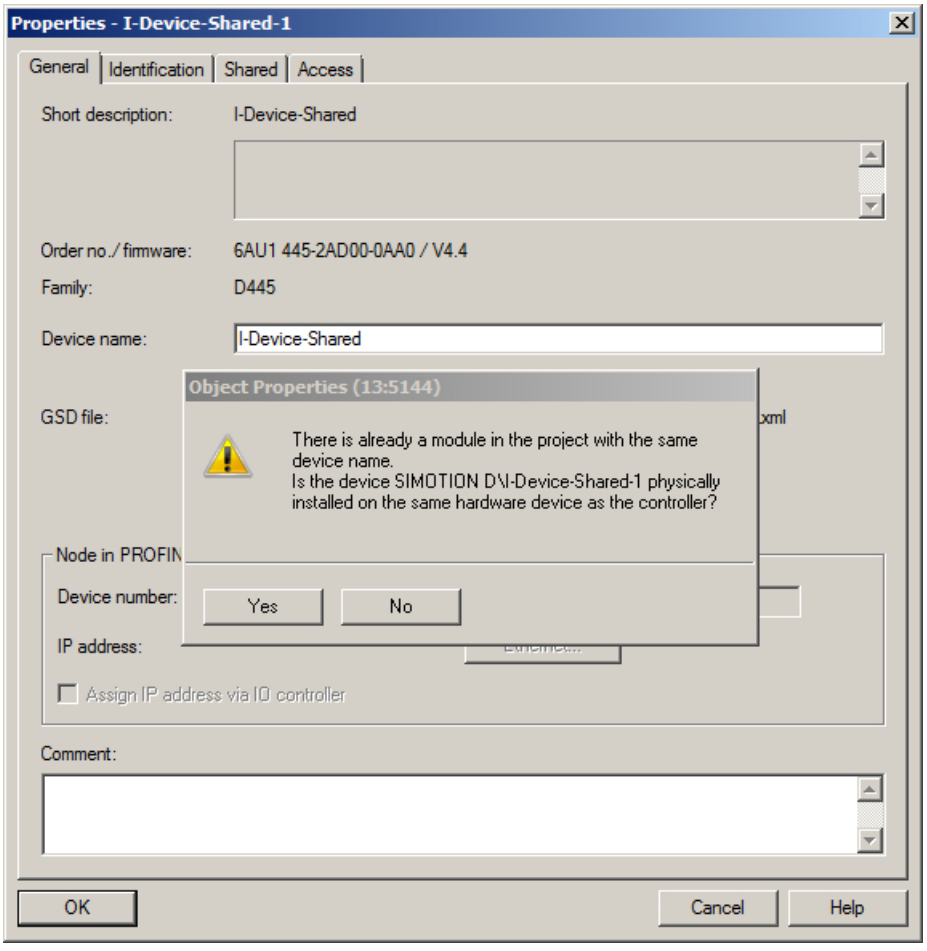


Figure 9-35 Entering a device name for the shared iDevice

Note

A notification informs you that there is already a module in the project with the same device name (IO controller of the iDevice). Click **Yes** to confirm this dialog box. This dialog may appear several times during configuration.

5. In the **Access** tab, enter the access to the submodules of the iDevice. On the F-CPU, the **fail-safe I/O** subslots are configured as **full**. Confirm by clicking **OK** and save and compile the station.

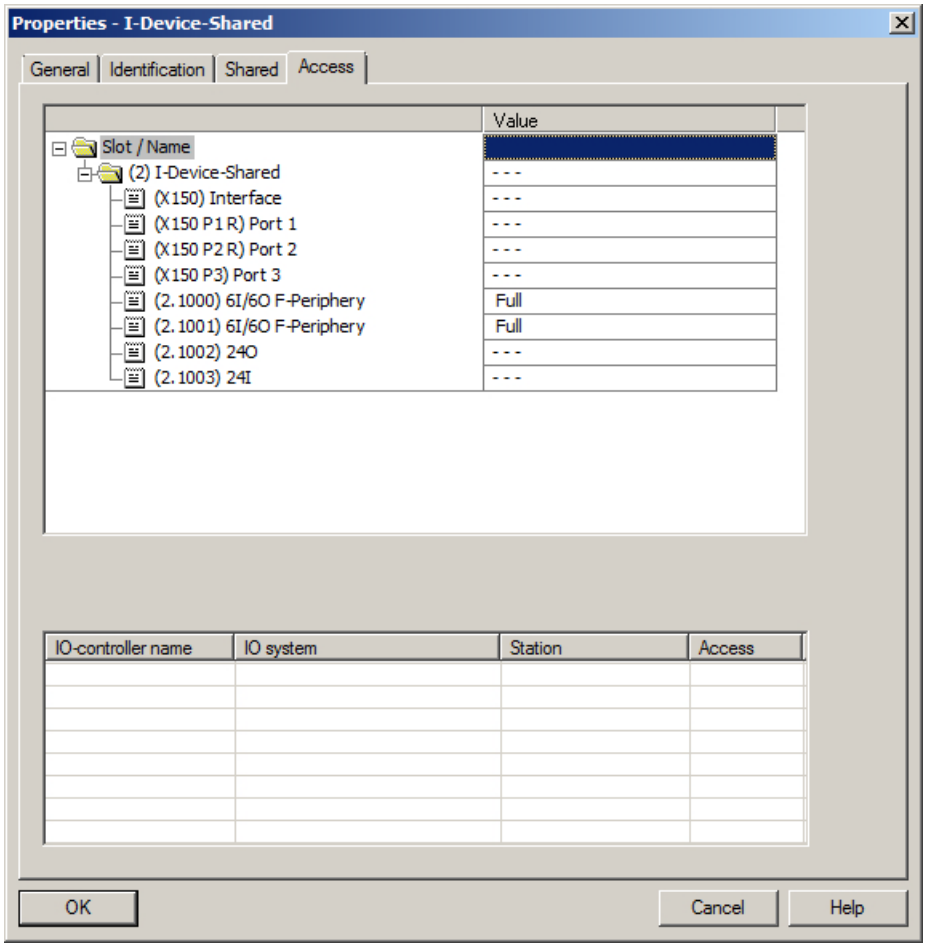


Figure 9-36 Setting access to the submodules for the F-CPU

6. Select the iDevice inserted in HW Config and select **Copy** in the context menu.

7. Switch to HW Config for the SIMOTION D435-2, select the PN IO system, and select **Shared paste** in the context menu. The iDevice is inserted.

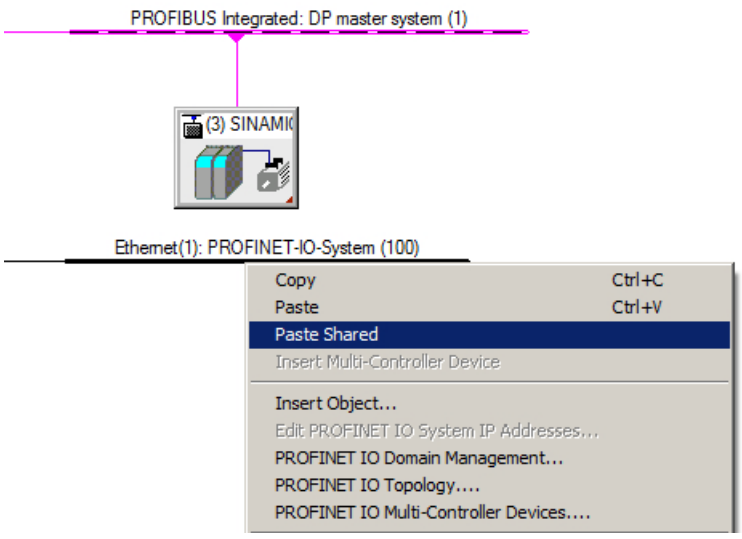


Figure 9-37 Shared paste for iDevice via context menu

8. In the **Access** tab, check the access to the submodules of the iDevice. These are automatically aligned with the HW Config of the F-CPU when the shared iDevice is inserted. The SIMOTION CPU must have full access to all subslots with the exception of the fail-safe I/O. The consistency of the subslots is checked automatically when you select "Save and compile", if the IO controllers are located in the same project.

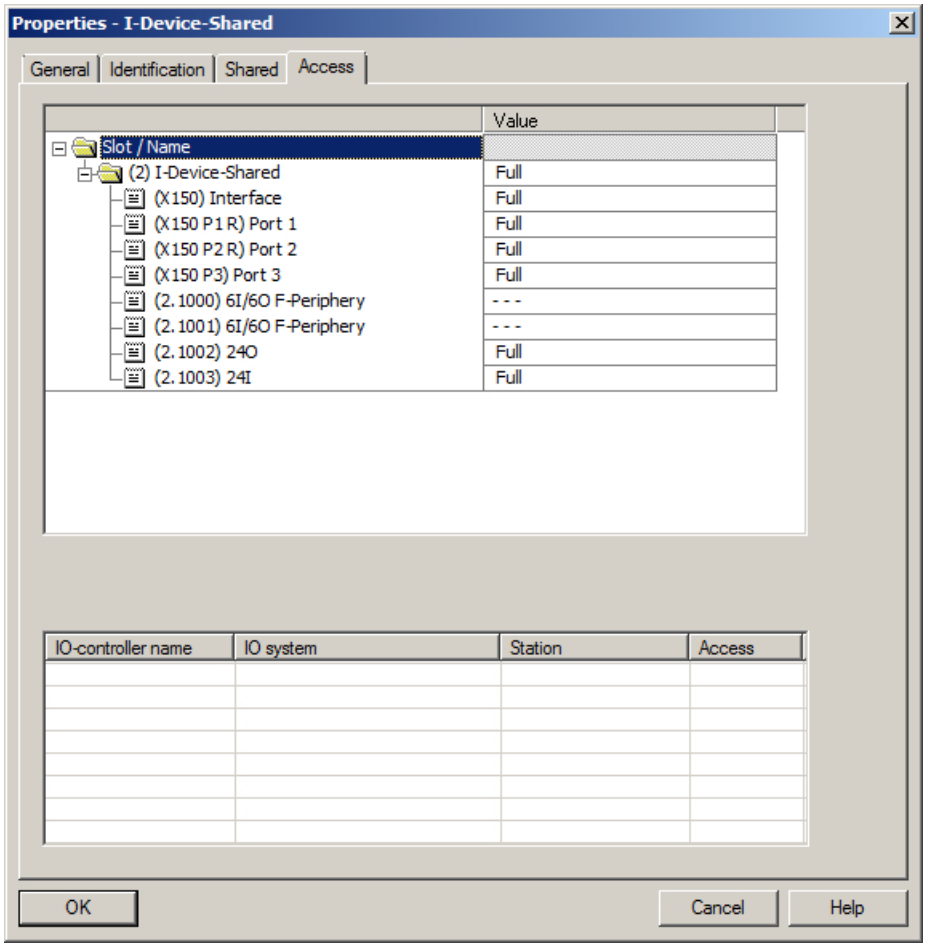


Figure 9-38 Setting access to the submodules for the SIMOTION D

9. On the SIMOTION D435-2, configure the **PN interface** (X150) as the **sync master** and as isochronous to the **servo**. (You should also refer to the chapter Isochronous application (Page 58)).

10. In the SIMOTION D435-2 station, double-click on **Port 1 (X150 P1)** to interconnect the topology. In the dialog that appears, select a **partner port** of the shared iDevice, e.g. Port 1.

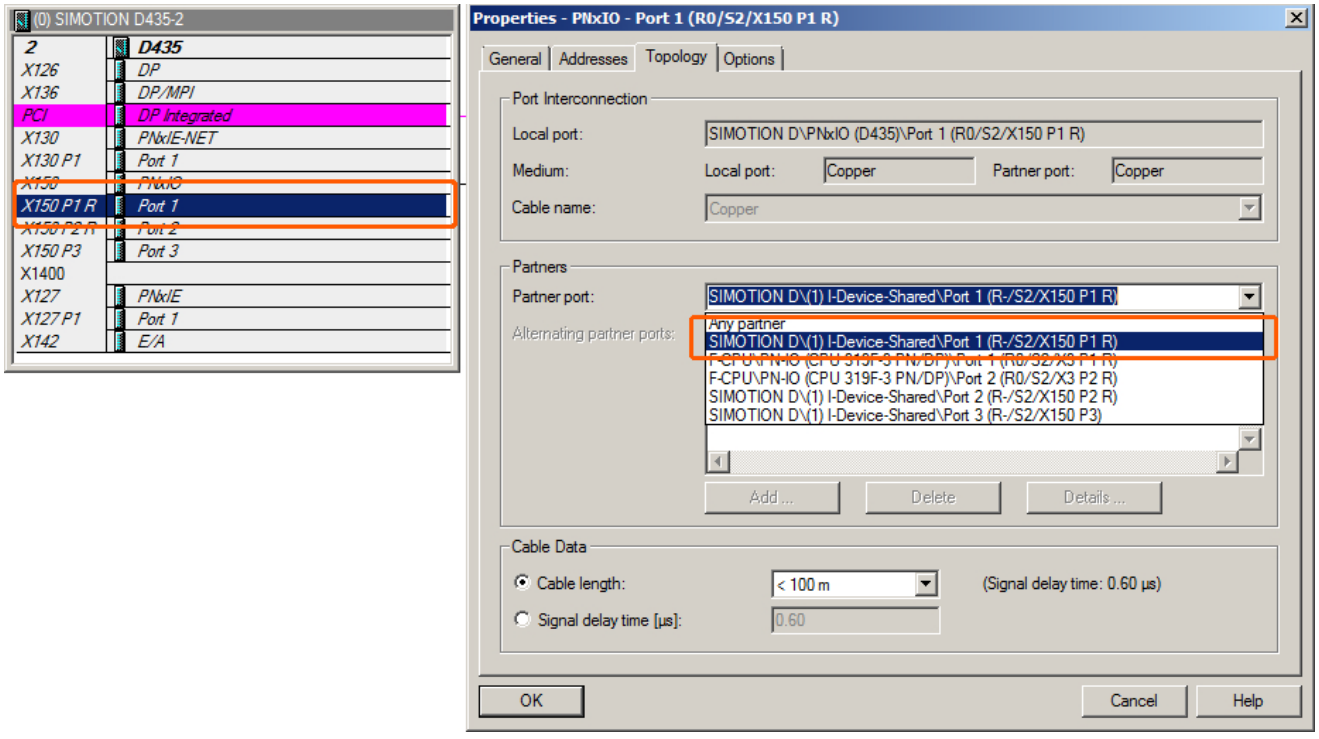


Figure 9-39 Port interconnection on the iDevice

Note

The ports of the SIMOTION D455-2 DP/PN are not displayed in the topology editor, as there is a shared iDevice with the same name in the project. For this reason, the ports must be interconnected using the Properties dialog.

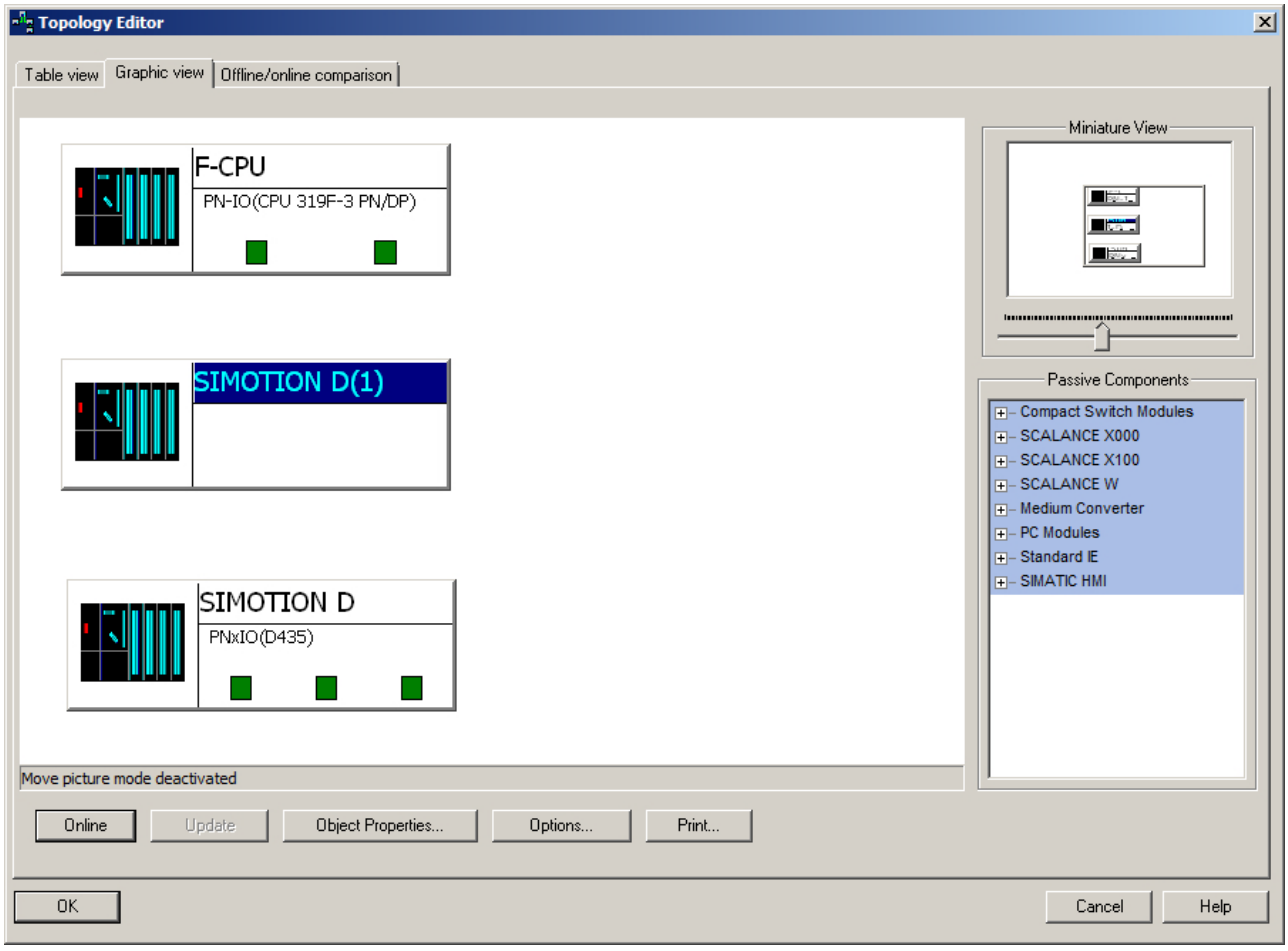


Figure 9-40 Shared iDevice in the topology editor

11. Select the iDevice in the PROFINET IO system of the D435-2 DP/PN and double-click on the PN interface of the iDevice (X150 interface). In the **Properties dialog** that appears, set sync slave and IRT in the **Synchronization** tab. On the **IO Cycle** tab, select **Servo** for the isochronous mode under **Assign IO device isochronously.**

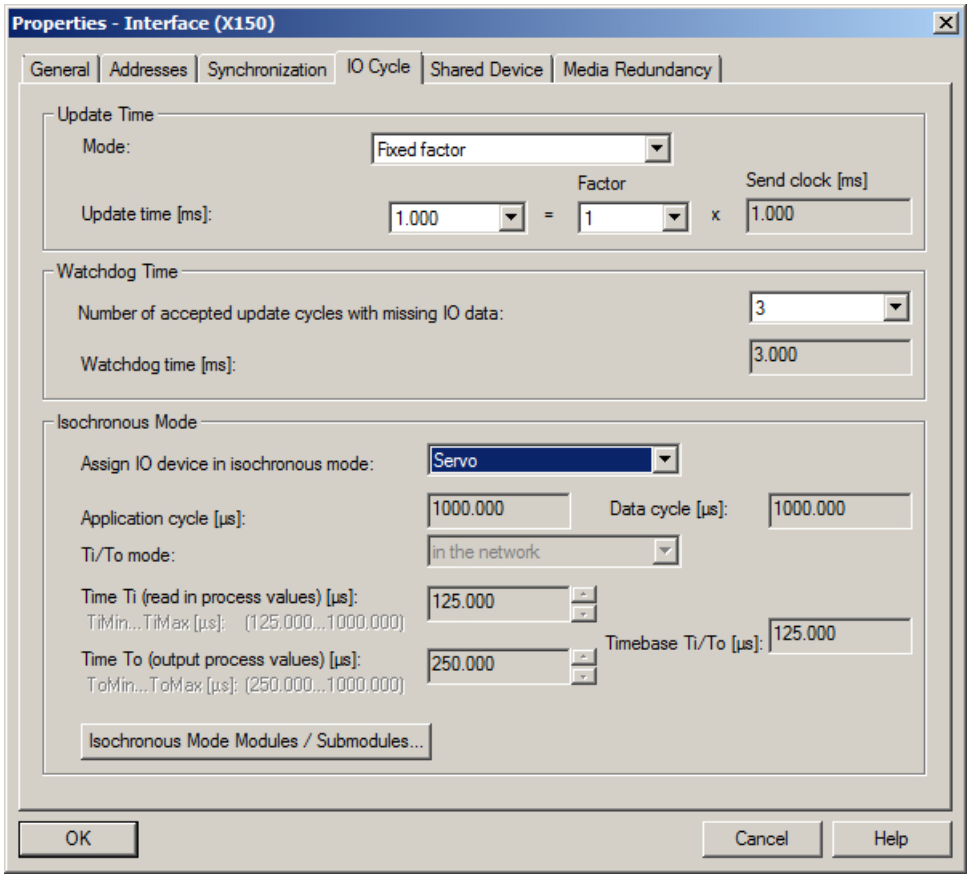


Figure 9-41 Configuring the iDevice as a slave IO device

12. **Save and compile** all stations. Once the stations have been compiled without any errors, configuration is complete.

9.6 PROFIsafe via PROFIBUS

9.6.1 General information about PROFIsafe on PROFIBUS

Two possibilities for PROFIsafe

There are basically two possible forms of PROFIsafe communication on PROFIBUS:

- **I-slave F-Proxy**
  F-CPU is the DP master in the project and monitors the drives on the lower-level SIMOTION I-slave
- **Failsafe data exchange broadcast**
  SIMOTION is the DP master in the project and the F-CPU monitors the drives as the I-slave.

The procedure for configuring PROFIsafe communication is virtually the same in both cases. The sections below each contain a brief example.

9.6.2 Supported devices and software requirements for PROFIsafe on PROFIBUS

Software packages to be installed on the programming device:

- SIMATIC Manager STEP7 version 5.4 SP2 or higher
- S7 F Configuration Pack Version 5.5 SP3 or higher
- S7 Distributed Safety Programming Version 5.4 SP3 or higher
- SIMOTION SCOUT Version 4.1.1 HF6 or higher
- SINAMICS firmware Version 2.5 or higher

Note

As of SIMOTION firmware 4.1.1 HF10 and SINAMICS firmware 2.5 SP1 HF10, five drives can be configured with a CX32. With earlier firmware versions, a maximum of 4 drives can be configured.

You will find the components suited to PROFIsafe in the *S120 Safety Integrated Function Manual*.

Supported devices

Table 9-2 Device overview

SIMOTION CPU	
Controller Based	C240 C240 PN
PC Based	P350-3 P320-4
Drive-based (blocksize)	D410 DP D410-2 DP D410-2 DP/PN
Drive-based (booksize)	D4x5 D445-1 D4x5-2 DP D4x5-2 DP/PN
SINAMICS drive units	
S120 CX32	CX32 CX32-2
S120	CU320 DP CU320-2 DP CU310 DP CU310-2 DP
S110	CU305 DP

Number of drive axes supported

With PROFIBUS, only 16 drive axes can be used per PROFIBUS interface.

9.6.3 I-slave failsafe proxy

9.6.3.1 Principles of I-slave failsafe proxy

Short description

Using the I slave F-Proxy you can produce a PROFIsafe configuration with an F host (F-CPU SIMATIC) on PROFIBUS with SIMOTION devices (SIMOTION D, SIMOTION P350, SIMOTION C) for the lower-level drives. Cyclic PROFIsafe data can then be routed to SINAMICS drives on SINAMICS_Integrated / PROFIBUS DP.

A failsafe host communicates with the drives via the I-slave interface and a failsafe proxy of a SIMOTION CPU. The drives may be located on the PROFIBUS DP of the SIMOTION CPU. The SIMOTION CPU's communication segments feature SINAMICS S120/S110 and SINAMICS Integrated/CX32/CX32-2.

9.6.3.2 Topology for I-slave failsafe proxy for PROFIBUS drive units

Example of topology for I-slave failsafe proxy

The diagram below shows a topology in diagrammatic form in which the Safety drives are connected to the SIMOTION CPU via PROFIBUS DP and this is connected to the F-CPU via PROFIBUS DP.

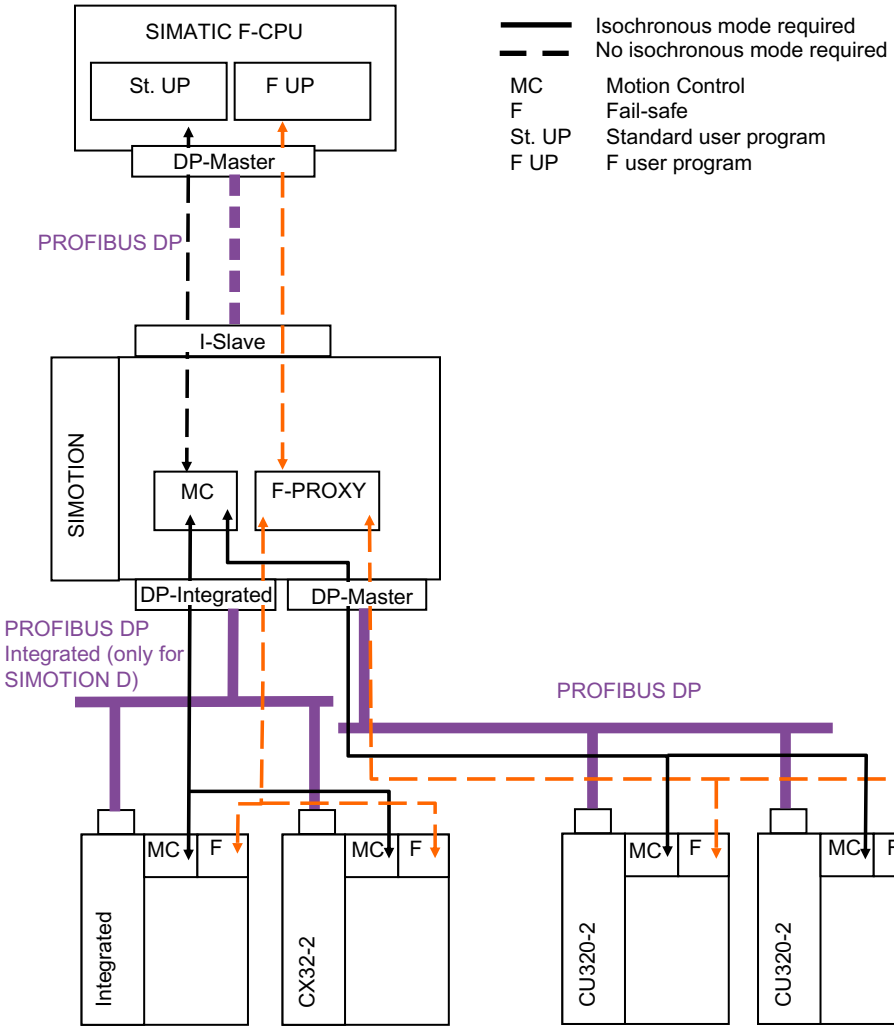


Figure 9-42 Topology for I-slave failsafe proxy for PROFIBUS drive units

9.6.3.3 PROFIsafe via PROFIBUS when SIMOTION D is used

The next sections deal with the configuration of PROFIsafe communication via PROFIBUS between the integrated drive unit SINAMICS S120 of a SIMOTION D or CX32 and a higher-level SIMATIC F-CPU.

Topology (network view of the project)

The basic topology of the components involved in PROFIsafe communication via PROFIBUS (SIMATIC F-CPU and D4x5 integrated with SINAMICS S120 or CU320) can be found in the previous section.

The drive unit (SINAMICS) and the SIMATIC F-CPU are located on different PROFIBUS subnets. In this case, a PROFIsafe router to SIMOTION D is configured so that the necessary data is copied from one network to the other.

Configuring PROFIsafe communication

The next sections describe the configuration of PROFIsafe communication between a SIMATIC F-CPU and a drive object of an integrated SINAMICS drive unit of a SIMOTION D. The procedure for configuring PROFIsafe communication between a drive unit of a CU320 and a SIMATIC F-CPU is basically the same and is not covered separately.

1. Create an F-CPU (e.g. CPU 317F-2) and a SIMOTION D4x5 controller (with integrated SINAMICS S120) in accordance with the hardware installed.
2. Define a SIMOTION CPU for operation as DP slave and the F-CPU as associated DP master.
3. Configure the SINAMICS drive unit in SIMOTION SCOUT in accordance with your hardware configuration.
4. Then insert a new TO axis and run through the axis wizard. In the wizard, interconnect the axis to the corresponding drive object of the S120 and a corresponding telegram will automatically be created (symbolic assignment).
5. Save and compile the project.

6. Create a PROFIsafe slot in the configuration of the SINAMICS drive unit.
   For this purpose, select in tab **IF1 the following: PROFIdrive PZD message frames** - the drive object which is to communicate with the SIMATIC F-CPU via PROFIsafe. Click on the **Adapt message frame configuration** button and select **Add PROFIsafe**.

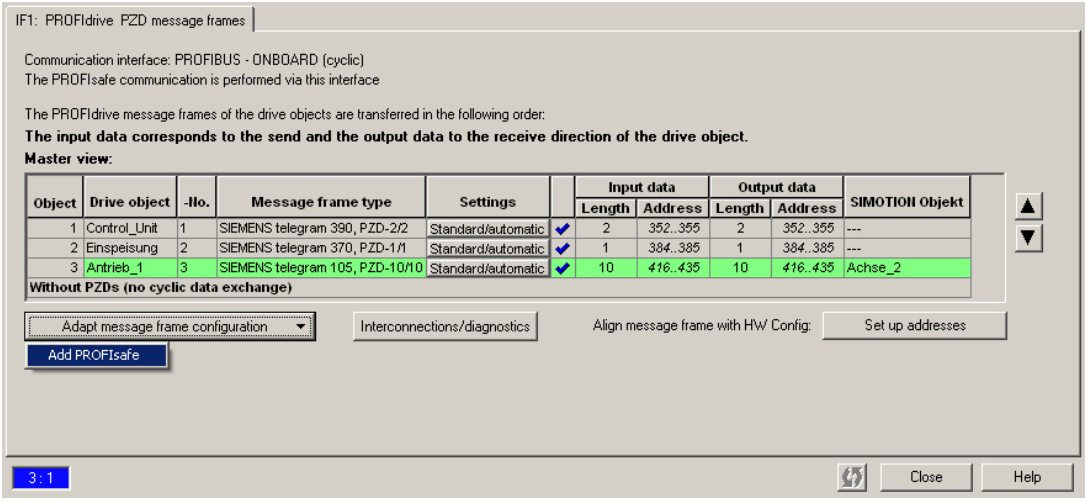


Figure 9-43 Inserting a PROFIsafe slot

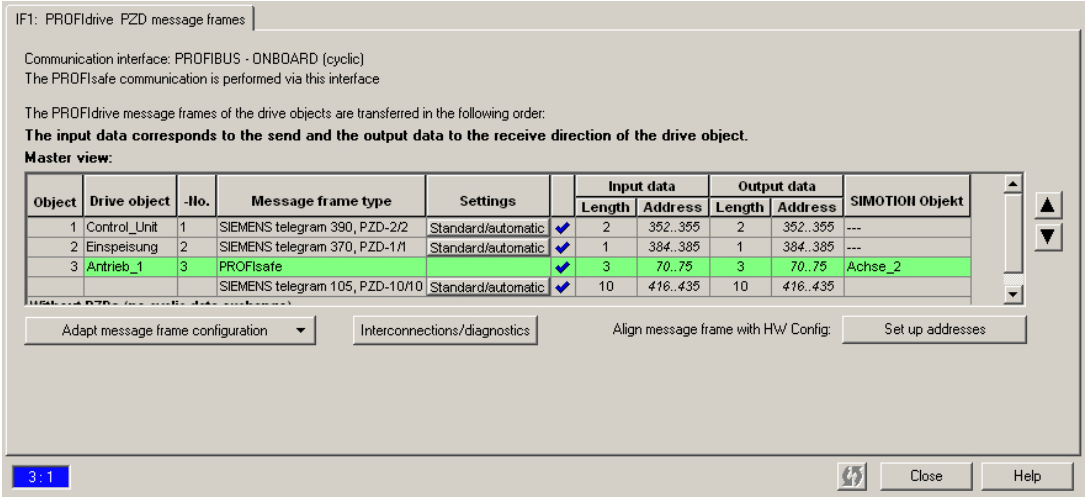


Figure 9-44 PROFIBUS message frame

7. Save and compile the project.
8. Transfer the new PROFIsafe slot to HW Config by clicking on the **Set up address** button.

9. In HW Config for the F-CPU, connect the preconfigured SIMOTION station to the F-CPU (hardware catalog: **PROFIBUS DP > Preconfigured stations** ...).

Note

To configure the SINAMICS Safety Integrated extended functions by means of SIMOTION, the message frames must be extended. To do so, a safety data block is appended to the PROFIdrive actual value message frame. Configuration and parameter assignment of this Safety data block are described in the Function Manual *SIMOTION Motion Control TO Axis electric/hydraulic, external encoder*.

10. You can display the F-communication parameters via the DP slave properties (double-click on SIMOTION I-slave). To do this, go to the **F configuration** tab and click on **New**.
Mode: Displays the communication relationship. F-MS module represents safety-related master-slave communication with SIMOTION.
DP partner (F I/O): SINAMICS drive properties.
Here you can select the relevant PROFIsafe drive via **DP address** or **Address**.
local: Properties of the SIMOTION CPU.
Enter the logical start address for F-communication of the SIMOTION CPU in the "Address" row.
The address space for sending and receiving the safety frames depends on the message frame used and must be located outside the process image of the SIMOTION CPU (>= 64).
Master (safety program): SIMATIC F-CPU properties.
The logical start address for F communication of the SIMATIC F-CPU must be entered here under "Address" (LADDR).
The address space for sending and receiving safety message frames depends on the message frame used and must lie within the process image of the SIMATIC F CPU.
In the SIMATIC F-CPU safety program, this address space can be used to access the PROFIsafe control or status words.
An overview of which message frames are possible for the individual drives can be found in the SIMOTION FAQs.

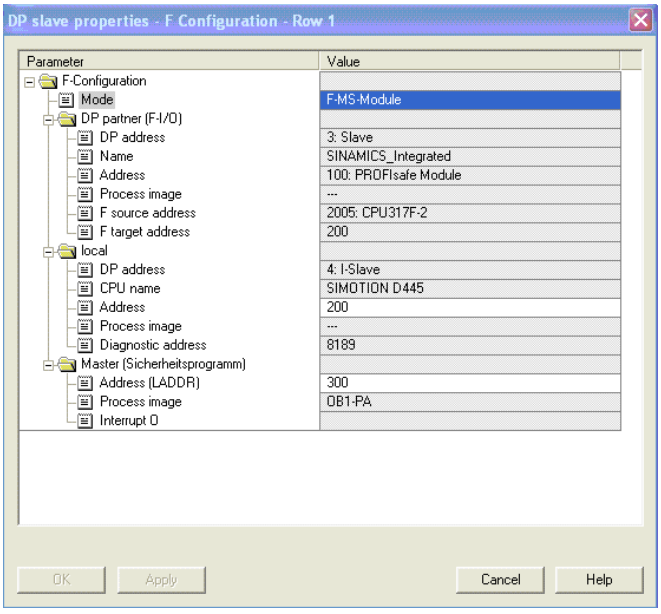


Figure 9-45 Master-slave coupling in PROFIsafe

11. Open the SIMOTION CPU project in HW Config.

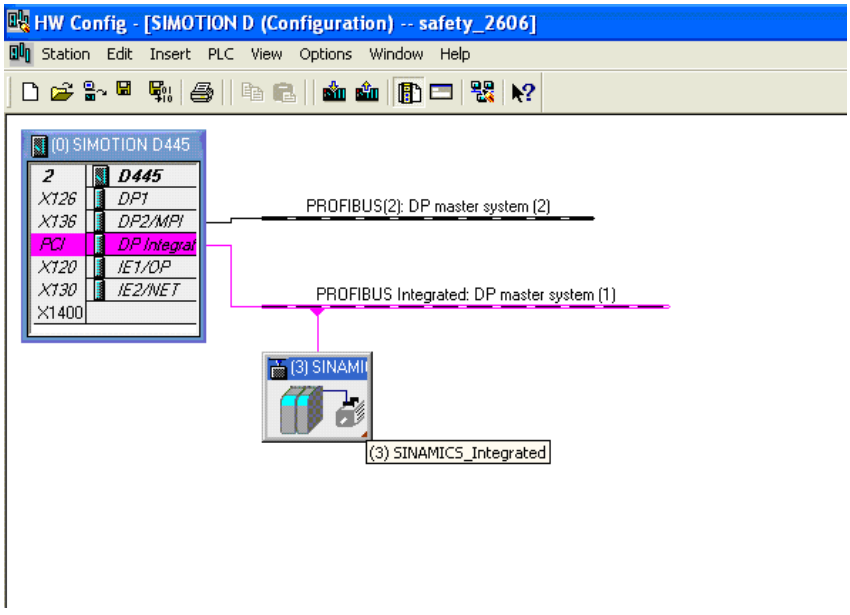


Figure 9-46 SIMOTION D configuration

12. Double-click on the icon of the SINAMICS drive unit and select the **Details** tab in the **Configuration** tab.

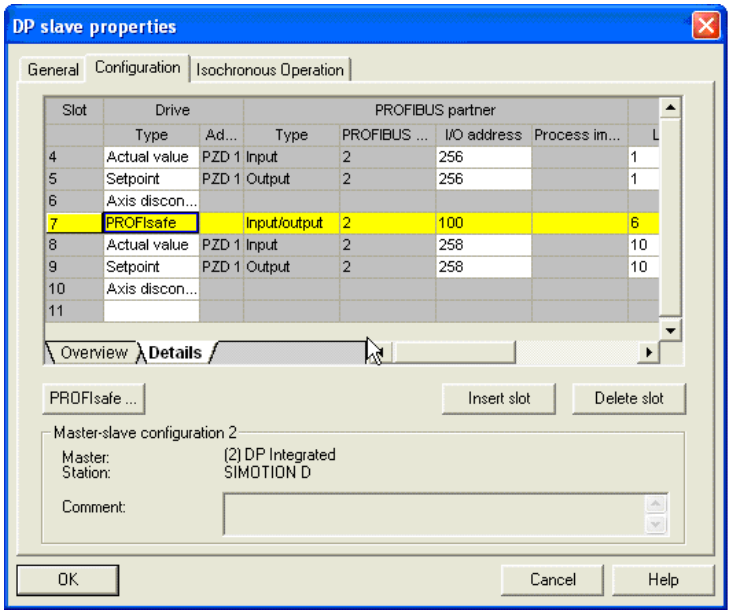


Figure 9-47 PROFIsafe configuration for SINAMICS drive unit

13. Click on the **PROFIsafe...** button and then define the F parameters which are important to F communication. As of STEP7 V5.5, PROFIsafe V2 is used by default.
(If the **PROFIsafe...** button cannot be used, you need to activate it using the **Activate...** button.)
For more information about the fail-safe parameters, see PROFIsafe properties for configuration (Page 273)

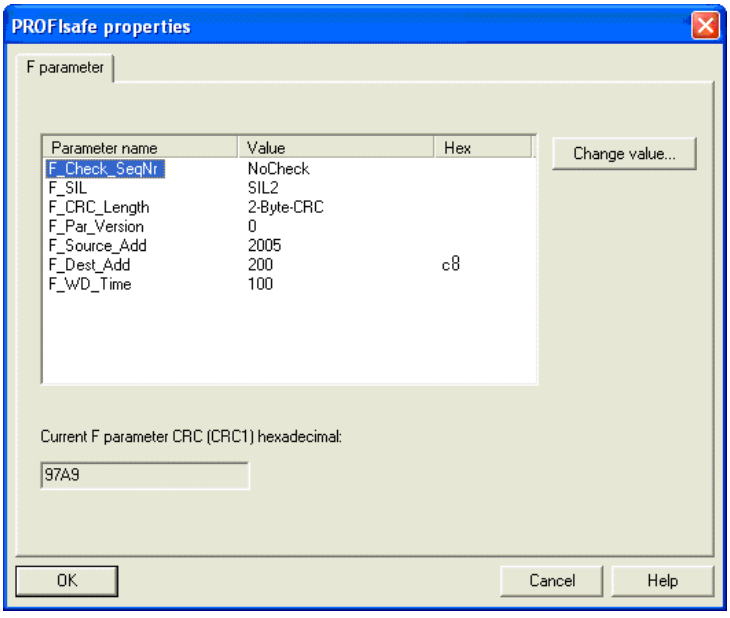


Figure 9-48 PROFIsafe properties (F-parameters)

14. Compile HW Config of the SIMOTION CPU. Compile the F-CPU configuration data in HW Config.

Note

For information about creating a safety program and accessing PROFIsafe useful data (e.g STW and ZSW) within the safety program, refer to the *SIMATIC, S7 Distributed Safety - Configuring and Programming* Programming and Operating Manual.

Safety configuration (online) in the SINAMICS drive

1. Call the configuration for Safety Integrated by selecting "Functions" at the SINAMICS drive entry in the tree structure.
2. Configure Safety Integrated and set to hex representation the F_Dest_Add parameter already defined under the drive's **PROFIsafe address** (p9610/p9810).
3. Finally, perform a POWER ON. The safety configuration is now active in the drive.

Note

For further information on safety configuration, see the SINAMICS S120 Safety Integrated Function Manual.

9.6.4 Failsafe data exchange broadcast

9.6.4.1 Principles of failsafe data exchange broadcast

Method of operation

SIMOTION CPU is the DP master for failsafe data exchange broadcast. The SIMATIC F-CPU is the DP slave on PROFIBUS DP and controls failsafe communication, e.g. with a CU320 of the SINAMICS S120.

Note

Control for the Safety Integrated functions cannot be routed to the SINAMICS Integrated of the SIMOTION D, Controller Extension CX32/CX32-2, or any other DP network in this constellation.

9.6.4.2 Topology of failsafe data exchange broadcast via PROFIBUS

Example of topology for failsafe data exchange broadcast for PROFIBUS

The diagram below shows a topology in diagrammatic form in which the Safety drives are connected to the SIMOTION CPU via PROFIBUS DP and the SIMATIC F-CPU is the PROFIBUS I-slave for failsafe communication.

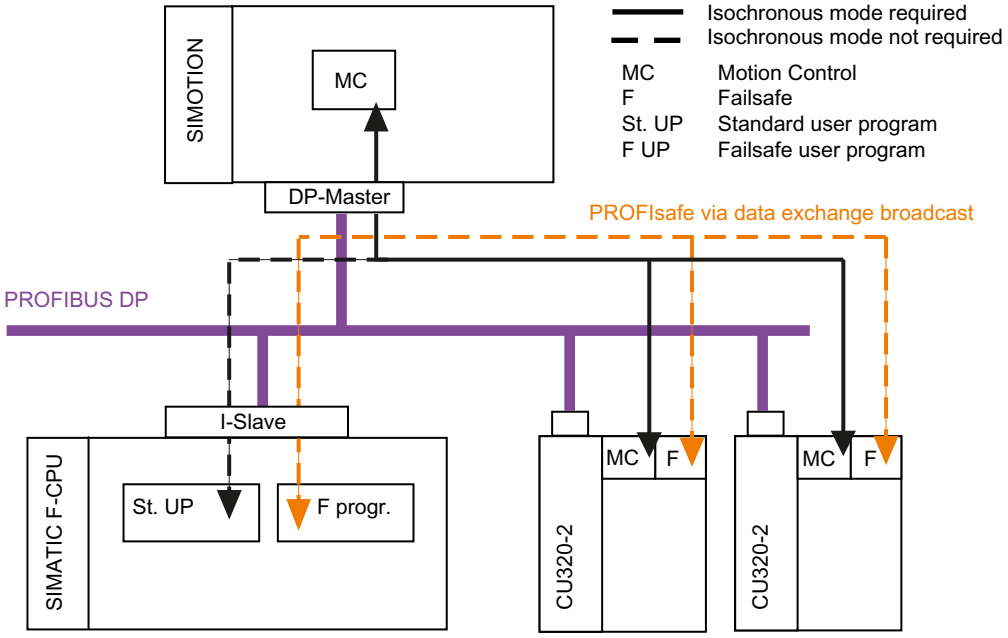


Figure 9-49 Topology of failsafe data exchange broadcast for PROFIBUS drive units

9.6.4.3 PROFIsafe via PROFIBUS with fail-safe internode data exchange taking the example of SIMOTION D

PROFIsafe communication is to be configured between a SINAMICS S120, SIMOTION D as DP master and SIMATIC F-CPU as I-slave via PROFIBUS.

Topology (network view of the project)

The basic topology of the components involved in PROFIsafe communication via PROFIBUS (SIMATIC F-CPU and D4x5 integrated with SINAMICS S120 or CX32) can be found in the following section:

The drive unit (SINAMICS) and the SIMATIC F-CPU are located in the same PROFIBUS subnet.

Configuring PROFIsafe communication via failsafe data exchange broadcast

1. In HW Config, create an F-CPU (e.g. CPU 317F-2), a SIMOTION D4x5 control and a SINAMICS S120 CU320 in accordance with the hardware installed.
2. Define the desired SIMOTION CPU as the DP master and the connected F-CPU as the associated DP slave.
3. Configure the SINAMICS drive unit in SIMOTION SCOUT in accordance with your hardware configuration.
4. Then insert a new TO axis and run through the axis wizard. In the wizard, interconnect the axis to the corresponding drive object of the S120 and a corresponding message frame will automatically be created (symbolic assignment).
5. Save and compile the project.
6. Create a PROFIsafe slot in the configuration of the SINAMICS drive unit.
   For this purpose, select in tab IF1 the following: PROFIdrive PZD message frames - the drive object which is to communicate with the SIMATIC F-CPU via PROFIsafe. Click on the **Adapt message frame configuration** button and select **Add PROFIsafe**.

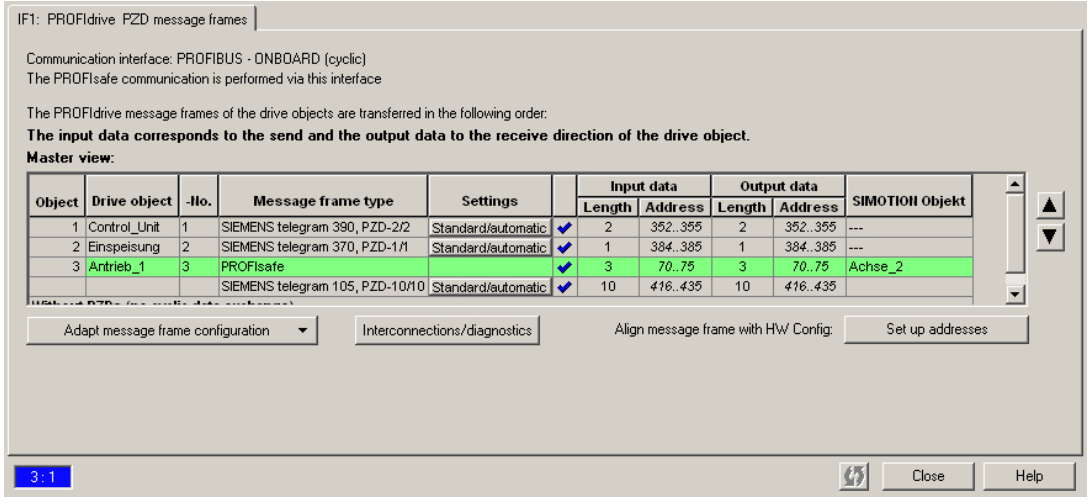


Figure 9-50 PROFIBUS message frame

7. Save and compile the project.
8. Transfer the new PROFIsafe slot to HW Config by clicking on the **Set up address** button.
9. In HW Config for the SIMOTION station, connect the preconfigured F-CPU to the SIMOTION station.(HW catalog: **PROFIBUS DP > Preconfigured stations...).**

10. The F-communication parameters are displayed in the DP slave (F-CPU) properties, tab **F Configuration**.
Mode: Displays the communication relationship. F-DX modules must be selected for data exchange broadcast. This represents a safety-related I-slave-slave relationship.
DP partner (F I/O): SINAMICS drive properties.
Here you can select the relevant PROFIsafe drive via DP address or Address.
local (safety program): SIMATIC F-CPU properties.
The logical start address for F communication of the SIMATIC F-CPU must be entered here under **Address (LADDR)**. The address space for sending and receiving safety message frames depends on the message frame used and must lie within the process image of the SIMATIC F CPU.
In the SIMATIC F-CPU safety program, this address space can be used to access the PROFIsafe control or status words.
Master address: Properties of the SIMOTION CPU.
Enter the logical start address for F communication of the SIMOTION CPU under **Input address** .
The address area for sending and receiving safety messages depends on the message frame used and must be located outside the process image of the process image of the SIMOTION CPU (>=64).
An overview of which message frames are possible for the individual drives can be found in the SIMOTION FAQs.

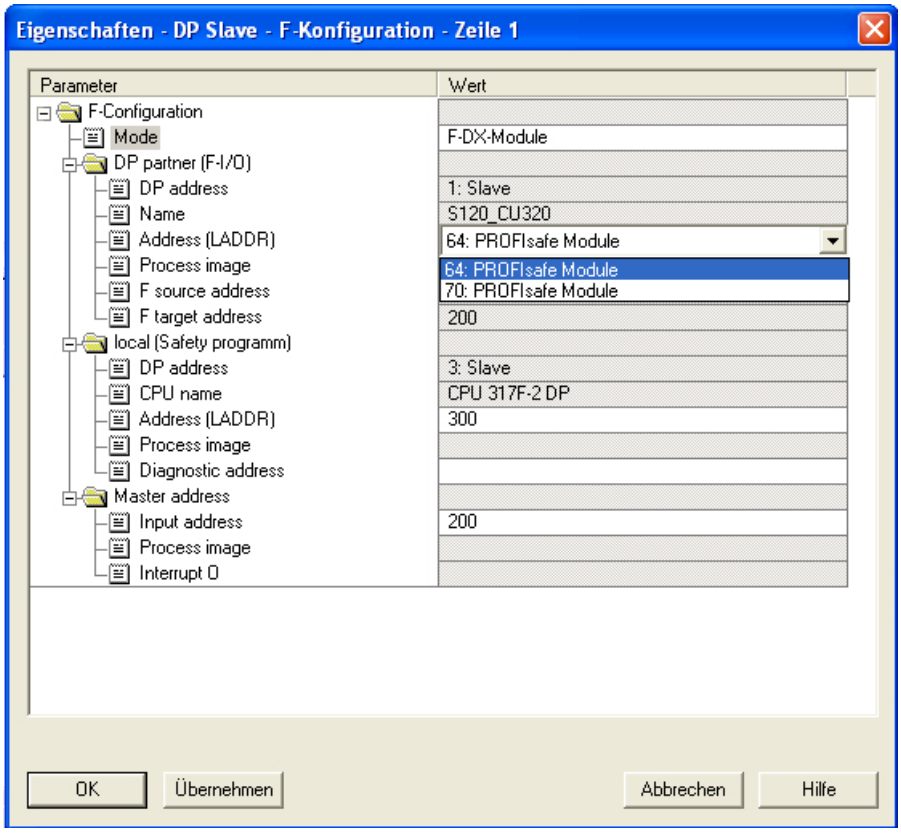


Figure 9-51 Properties of failsafe configuration for data exchange broadcast

11. Open the SIMOTION CPU project in HW Config.

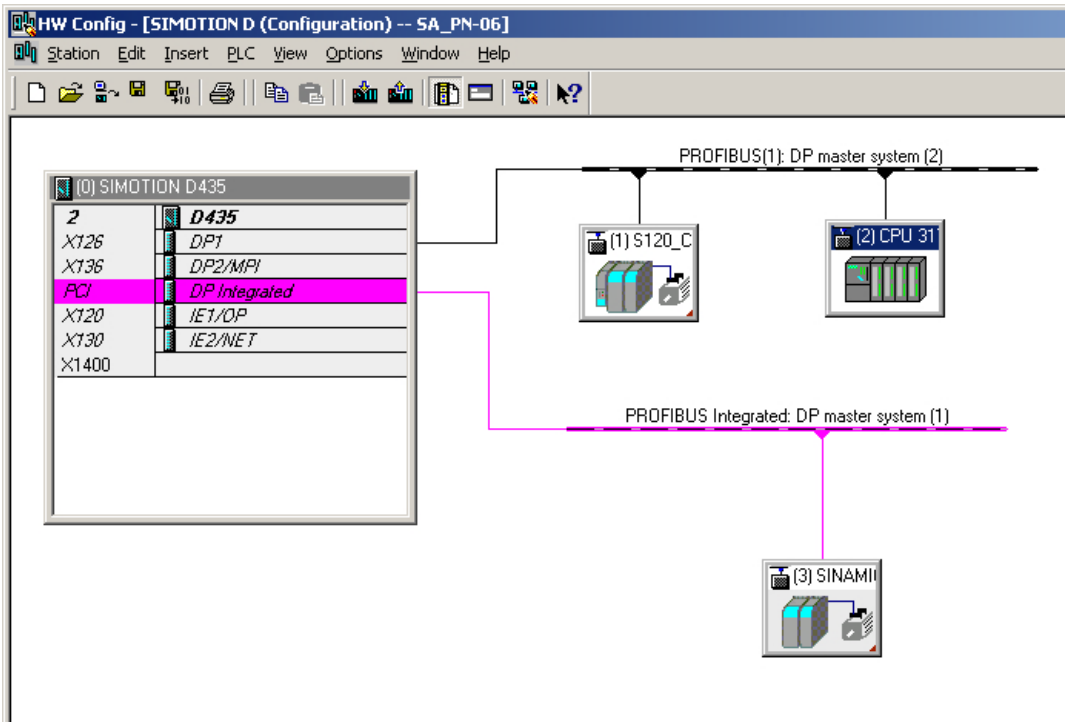


Figure 9-52 HW Config failsafe data exchange broadcast

12. Double-click on the icon of the SINAMICS drive unit and select the **Details** tab in the **Configuration** tab. Click the **PROFIsafe…** button to specify the relevant F Parameters for failsafe communication.
(If the **PROFIsafe…** button cannot be used, you need to activate it using the **Activate...** button.)
For more information about the failsafe parameters, see PROFIsafe properties for configuration (Page 273)

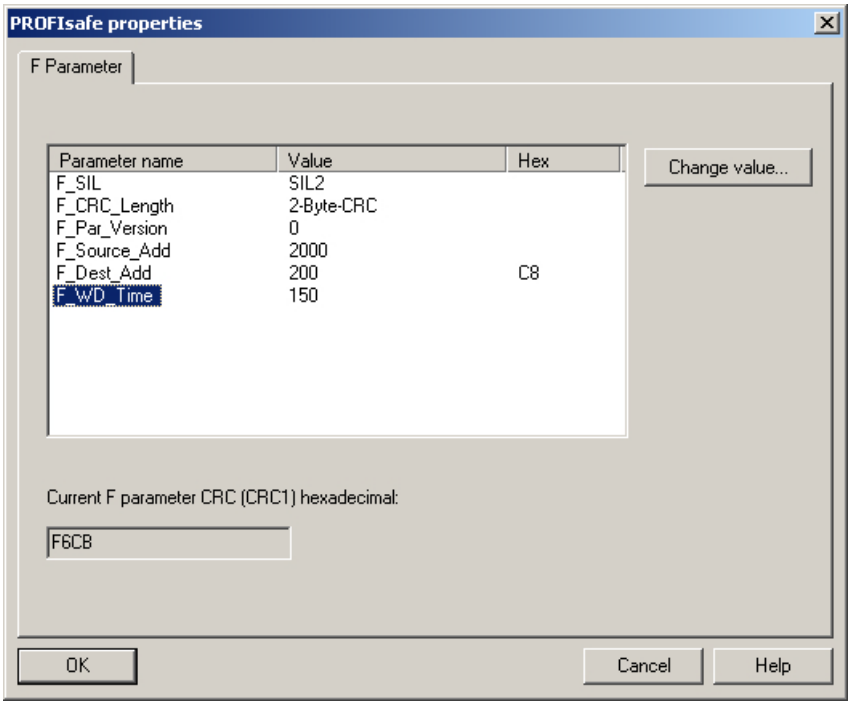


Figure 9-53 PROFIsafe properties failsafe data exchange broadcast mode V1

13. Compile HW Config of the SIMOTION CPU. Compile the F-CPU configuration data in HW Config.

Safety configuration (online) in the SINAMICS drive

1. Call the configuration for Safety Integrated by selecting the **Functions** entry in the structure tree where it says SINAMICS drive.
2. Configure Safety Integrated and set to hex representation the F_Dest_Add parameter already defined under the drive's PROFIsafe address (p9610/p9810).
3. Finally, perform a POWER ON. The safety configuration is now active in the drive.

Note

For further information on Safety configuration, see the *SINAMICS S120 Safety Integrated* Function Manual.

9.7 PROFIsafe configuration - acceptance test and reports

Acceptance tests and acceptance reports

Once configuring and commissioning has been successfully completed, an acceptance test of the drive safety functions must be carried out. This involves checking correct parameter assignment of the safety functions. The tests carried out are documented in reports.

Note

When carrying out the acceptance test for PROFIsafe communication, please note the information provided in the *SIMATIC, S7 Distributed Safety - Configuring and Programming* Programming and Operating Manual and the *SINAMICS S120 Safety Integrated* Function Manual of the drive used.

9.8 Additional information on SIMOTION and PROFIsafe

Description

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

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

9.9 Exporting/importing drive objects (DO)

Description

You can replace drive objects (DO) in an existing SIMOTION project. This means that you can replace deleted drive objects with new imported drive objects. You might need to do this, for example, if you want to replace motors with others. Whether replacement is successful or not depends on the configuration (for example, configured PROFIsafe, telegrams).

Requirement

Both the deleted drive object and the new imported drive object must meet the following conditions:

- The drive objects are the same version
- The number of deleted and the number of imported drive objects is identical
- The length of the telegrams is identical. This means, the activated technology data must be identical (TO axis) if they result in a telegram extension.
- F addresses are identical if PROFIsafe (F-Proxy) is used.

This is how you replace drive objects

1. Open the export list of the Control Unit whose drive you want to replace and navigate to parameter P978. This is where the sequence of drive objects is stored.
2. Make a note of the settings of parameter P978 or export the parameter.

   Note

   Alternatively, you can open the telegram configuration and make a mental or written note of the sequence of drive objects.

3. Delete the drive objects via the shortcut menu in the project navigator.
4. Import the new drive objects via the shortcut menu in the project navigator. The new drive objects are inserted automatically and are assigned new address ranges and positions in the telegram. You will have to change these in the next step.

NOTICE
Do not compile or align telegrams After completing this step, **do not** compile (store and compile) the project and **do not** align the telegrams.

5. Open the expert list of the CU and navigate to parameter P978.
6. Enter the sequence of drive objects you previously noted or import the previous values. Alternatively, you can open the telegram configuration and place the imported drive objects in the correct sequence using the arrow buttons on the right.

7. If symbolic assignment is active, go into the address list and open the shortcut menu in the top left field (next to the name) and run the "assignment setup" function.
8. Save and compile the project.

Note about replacing drive objects

If the above requirements for deleting and importing drive objects are not met, the project cannot be made consistent merely by adapting the sequence of drive objects. In this case, you will have to adapt the telegrams and addresses manually. If an F-Proxy is configured, the relevant drive, if it has a red background, must be deleted and created new. It may be necessary to adapt the PROFIsafe configuration on the drive.

PROFIdrive

10

10.1 Why profiles?

Profiles used in automation technology define certain characteristics, behaviors, and interfaces 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, behaviors, and interfaces of devices and systems. They aim to ensure a certain degree of interoperability of devices and systems on a bus which are part of the same product family due to "profile-compliant" development.

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

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

PROFIdrive

PROFIdrive is a device-specific (Drives) application profile. PROFIdrive essentially describes a "standard interface" for drives to PROFIBUS and PROFINET. It contains a detailed description of how the communication functions "data exchange broadcast", "equidistance" and "isochronous operation" are used 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 telegrams, the access to drive parameters, the drive diagnostics, etc.

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

The basic philosophy: – Keep it simple –

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

10.2 PROFIdrive overview

The PROFIdrive Profile

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

PROFIdrive is split into a general part and a bus-specific part. The following properties are defined in the general part.

- Base model
- Parameter model
- Application model

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

- PROFIdrive to PROFIBUS
- PROFIdrive to PROFINET

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

Literature note

PROFIdrive profile

Profile Drive Technology – PROFIdrive Profile
Version V4.1, May 2006, Order Number 3.172
PROFIBUS User Organization e. V.
Haid-und-Neu-Straße 7, D-76131 Karlsruhe
http://www.profibus.com

Standards

Standard IEC 61800
IEC 61800-7 Part 203, Part 303

10.3 PROFIdrive base/parameter model

Description

The PROFIdrive base model describes an automation system in terms of a number of devices and their interrelationships (application interfaces, parameter access).

Table 10-1 The base model distinguishes between the following device classes:

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

PROFIdrive device classes

In the picture below, the possible communication relationships of the PROFIdrive device classes are presented.

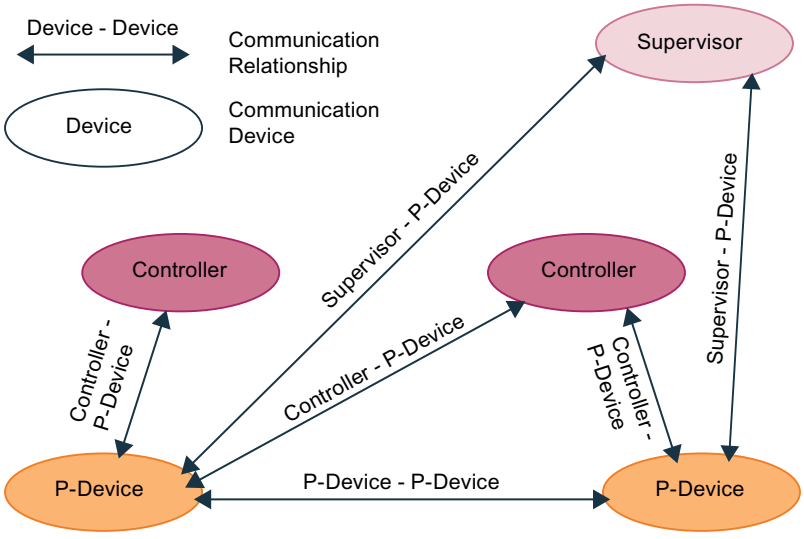


Figure 10-1 Communication between device classes according to PROFIdrive

Example of a PROFIdrive automation concept

The graphic below shows a typical automation concept.

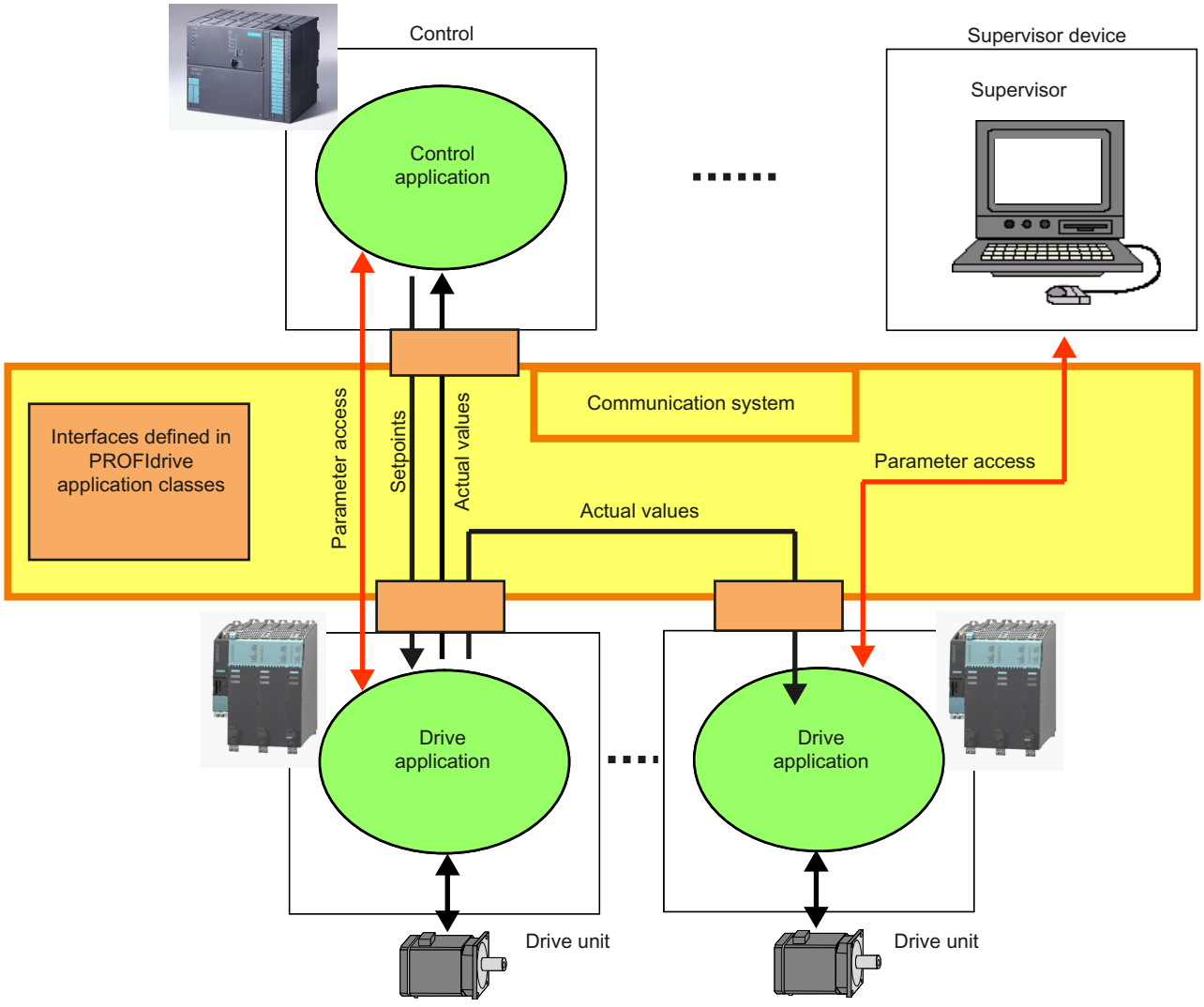


Figure 10-2 Automation concept

Communication services

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

- Cyclic data exchange via a cyclic data channel
  Motion control systems need cyclically updated data during operation for open- and closed-loop control purposes. This data must be sent to the drive units in the form of setpoints or transmitted from the drive units in the form of actual values, via the communications system. Transmission of this data is usually time-critical.
- Acyclic data exchange via an acyclic data channel
  In addition to cyclic data exchange, there is an acyclic parameter channel for exchanging parameters between the control/supervisor and drive units. Access to this data is not time-critical.

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

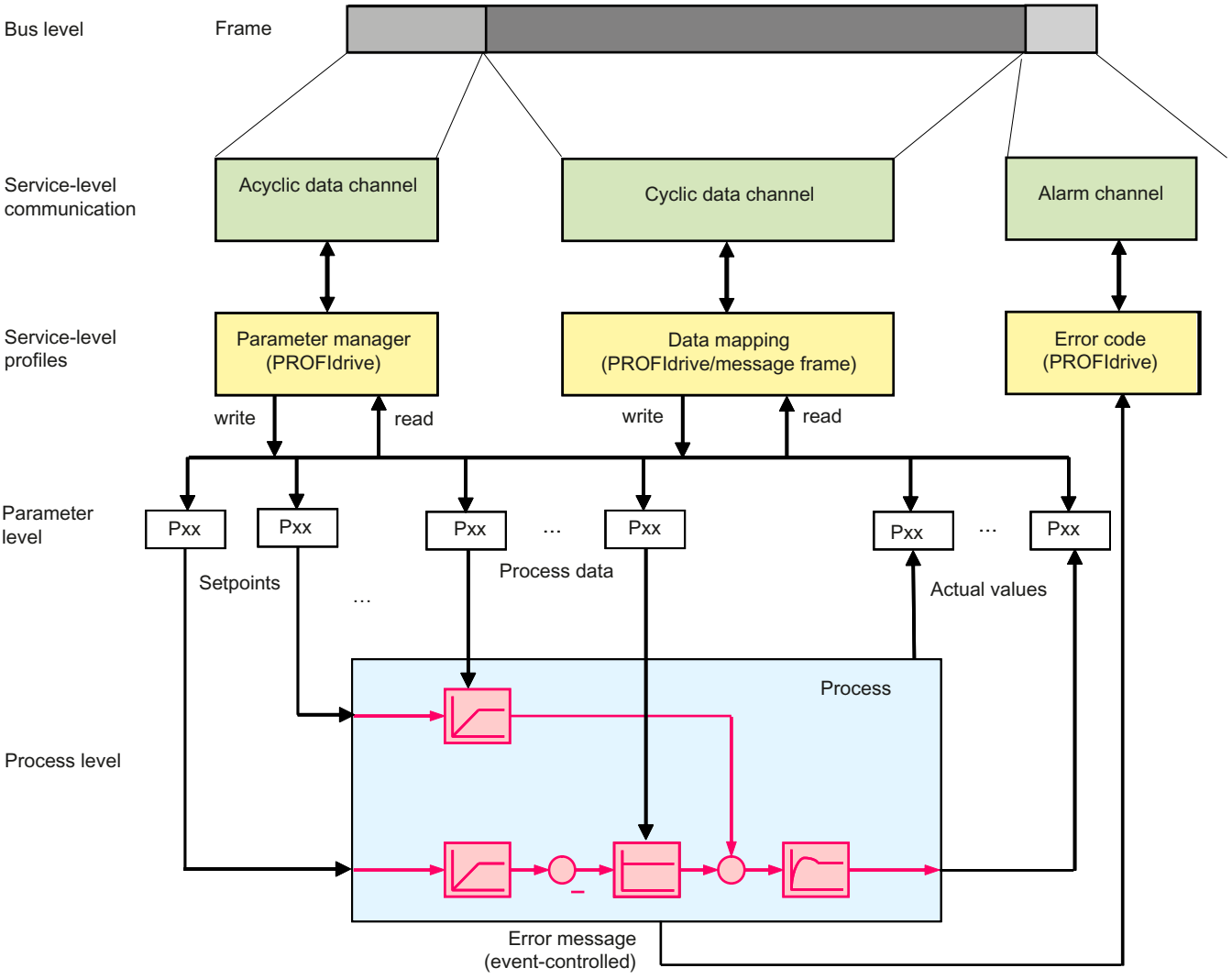


Figure 10-3 Data model and data flow in the P device

Alarms and error messages

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

Parameter model

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

- Profile parameters are defined for objects derived from the device model of the PROFIdrive profile. These may include general functions such as drive identification, fault buffer, or drive control, for example. These parameters are the same for all drives. The profile-specific parameters are in the range 900–999 and 60,000–65,535.
- All other parameters are manufacturer-specific. The parameters are defined by the interface for the application process, rather than by the profile. The manufacturer-specific parameters are in the range 1–699 and 1,000–59,999.

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

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

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

PROFIdrive defines the access mechanism to the parameter (PROFIdrive parameter channel) and approx. 70 profile-specific parameters; e.g. for fault buffer, drive control, and device identification.

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

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

PROFIdrive for PROFINET contains the functions PROFINET RT, PROFINET (isochronous mode) and IO controller–IO device communication.

See also

Specifications for PROFIBUS and PROFINET IO (Page 363)

10.4 Segmentation in application classes

Integration of drives in automation solutions

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

Table 10-2 Application classes

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

Application classes

Application 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. The position control loop is closed via the bus. The synchronization of the position control cycles in the control and in the speed controllers in the drive requires a clock synchronization (PROFIBUS DP-V2 or PROFINET with IRT).

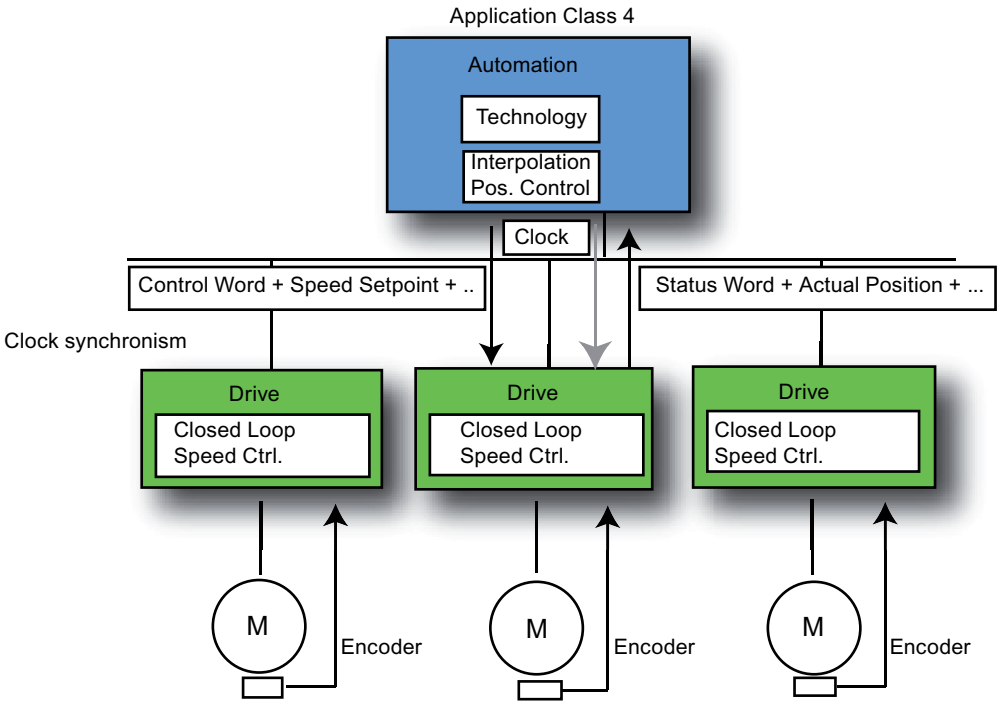


Figure 10-4 Application classes

Using the DSC function (Dynamic Servo Control), the dynamic interference immunity of the position control loop is increased considerably for mechanically rigid systems. This is achieved using an additional minimum dead time position feedback in the drive. As a result, short position control cycle clocks (e.g. for servo 125 µs in the SINAMICS S) are enabled and transmission dead time on the bus then only has an effect on the control behavior of the axis.

See also

Specifications for PROFIBUS and PROFINET IO (Page 363)

10.5 PROFIdrive-specific data types

Description

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

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

These standards contain detailed descriptions of the data types. The most important data types are listed below. Data types are provided by the _readDriveParameterDescription function, for example.

Note

For the communication with SINAMICS, you will have to use the AnyType_to_BigByteArray() or BigByteArray_to_AnyType() SIMOTION system functions to perform a type conversion for different data types (normalized value N2, N4; normalized value X2, X4; fixed-point value E2, C4 and time constants T2 and T4).

PROFIdrive profile-specific and standard data types

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

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

Normalized value N2, N4

Linear normalized value, 0% corresponds to 0 (0x0), 100% corresponds to 2^{14} (0x4,000) for N2, or 2^{30} (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% ≤ i ≤ (200-2^{-14})%	2^{-14} = 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% ≤ i ≤ (200-2^{30})%	$2^{-30,}$ = 9,3 * 10^{-8}%	114	3	2^{-15}	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}	2^{-21}	2^{-22}
			4	2^{-23}	2^{-24}	2^{-25}	2^{-26}	2^{-27}	2^{-28}	2^{-29}	2^{-30}

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

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

Coding

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

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

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

Fixed-point value E2

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

Coding

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

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

Range of values E2	Resolution	Cod. (dec.)	Octet	Bit							
8	7	6	5	4	3	2	1				
-256+2^{-7} ≤ i ≤ 256-2^{-7}	2^{-7} = 0,0078125	121	1	SN	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}
			2	2^{0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}

Fixed-point value C4

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

Coding

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

Range of values	Resolution	Coding (dec.)	Length
-214,748.3648 ≤ i ≤ 214,748.3648	10^{-4} = 0,0001	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

Overview of time constants T2, T4, D2, and R2

Note

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.

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 ≤ x ≤ 32767.
  When interpreted, internal values that fall outside this range of values are set to 0.
- T4: As with Unsigned32

Range of values	Resolution	Coding (dec.)	Length
0 ≤ i ≤ 32,767 * T_a	T_a	118	2 octets
0 ≤ i ≤ 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/16384

Coding

- T2: As with Unsigned16, with a restricted range of values of 0 ≤ x ≤ 32767.
  When interpreted, internal values that fall outside this range of values are set to 0.

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

Time constant R2

Time data as a reciprocal multiple of the constant sampling time T_a. Interpreted value = 16384 * T_a/internal value

Coding

- T2: As with Unsigned16, with a restricted range of values of 1 ≤ x ≤ 16,384.
  When interpreted, internal values that fall outside this range of values are set to 16,384.

Range of values	Resolution	Coding (dec.)	Length
$1 * Ta \leq i \leq 16{,}384 * T_a$	T_a	117	2 octets

Note

Further data types:

Standard PROFIBUS/PROFINET data types (only available in English) (Page 393)

Profile-specific PROFIBUS/PROFINET data types (only available in English) (Page 404)

See also

Parameter request/response data set (Page 359)

10.6 Acyclic communication (Base Mode Parameter Access)

10.6.1 Acyclic communication

Description

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

These signals are transferred cyclically between the controller and the drive.

In addition, PROFIdrive drive devices recognize parameters in which additional, necessary data are held; such as error codes, warnings, controller parameters, motor data, etc. This data is not usually transmitted cyclically, but "acyclically" only when required. Commands for the drive can also be transferred using parameter accesses.

The reading/writing of parameters from PROFIdrive units is always performed acyclically, using what is known as "Base Mode Parameter Access". "Base Mode Parameter Access" can be used with both PROFIBUS and PROFINET. For differences between PROFIBUS and PROFINET, please refer to Specifications for PROFIBUS and PROFINET IO (Page 363).

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

Note

The DPV1lib significantly simplifies configuration of acyclic communication. You can find it on the SIMOTION DVD U&A under **Applications > Cross-Sector Applications > Drive Communication**.

10.6.2 Reading and writing parameters with Base Mode Parameter Access

Description

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

Under this arrangement, parameter access always consists of:

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

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

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

The following figure shows the method of operation of the parameter channel on reading/writing the data set.

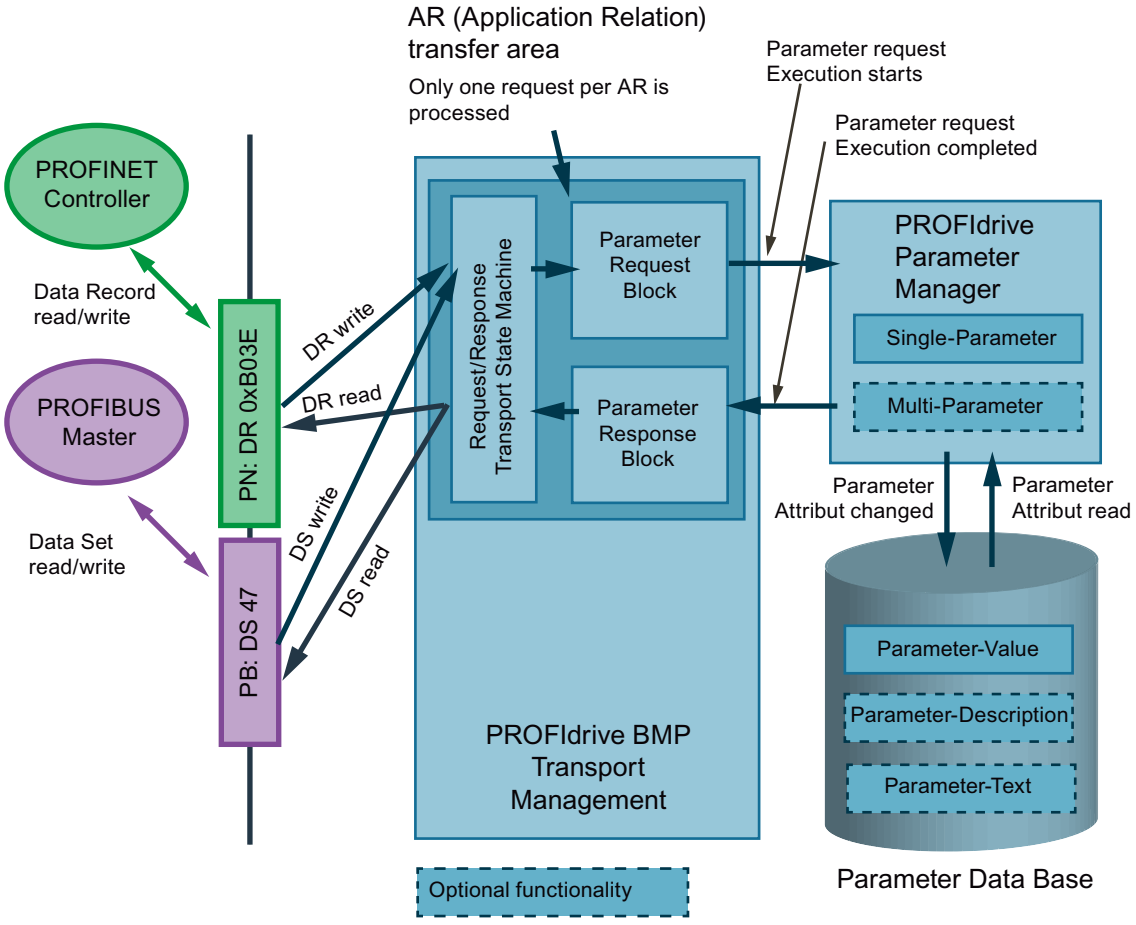


Figure 10-5 Function of the PROFIdrive parameter channel

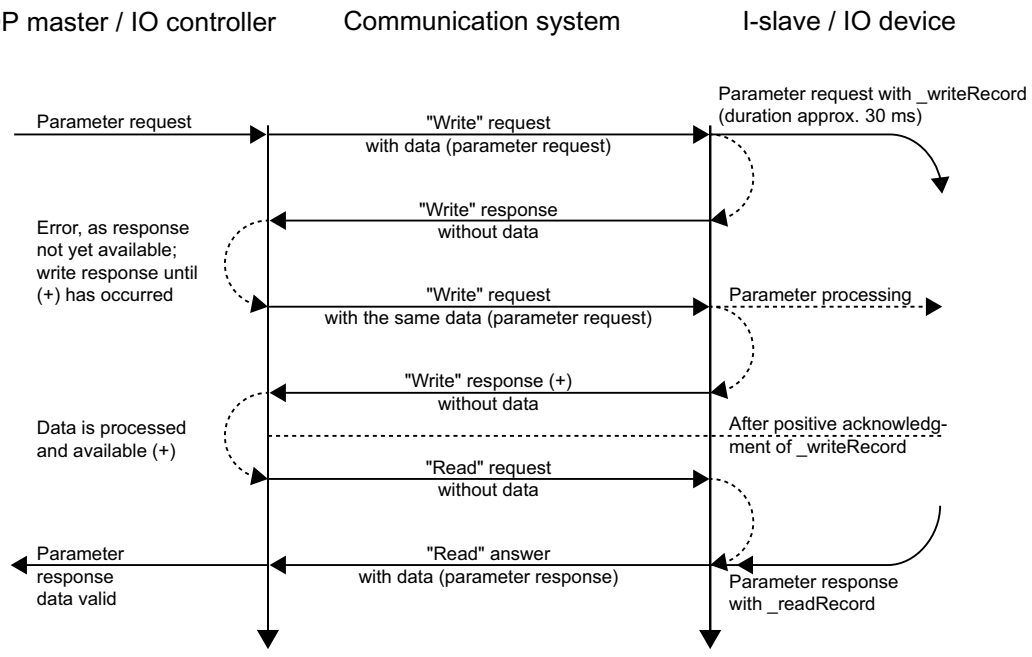


Figure 10-6 Reading and writing acyclically

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

- Request
- Response

Writing parameter records

Initially, data (P request/response data set) is transmitted to the job structure for the purpose of writing (one or more) parameter values. The data is subsequently transmitted with "Write data set" using _writeRecord. Repeated instances of "Write data set" (_writeRecord without data) can be used to monitor the status until a positive acknowledgment is given. With a "Read data set" request (_readRecord), execution is requested in the target device.

Note

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

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

Reading parameter records

For the purpose of reading parameter values, the data block for determining which parameter(s) is/are to be read is created first. This data record is transferred to the drive as a request using "Write data record" (_writeRecord). A subsequent "Read data set" (_readRecord) as a request then returns the requested values one time.

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

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

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

WORD and DWORD representation in Big Endian format

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

A conversion may be required for SIMOTION, see Program example (Page 387).

See also

Rule 5 - a maximum of eight concurrent calls is possible in SIMOTION (Page 375)

Parameter request/response data set (Page 359)

10.6.3 Parameter request/response data set

Structure of the P request/response data set

This always consists of:

- Header (job identifier (request/change), job reference (job ID), number of parameters in the job, target axis / drive object ID)
- Parameter details (number of elements / indices, attribute, parameter number, subindex)
- Parameter values (only with job identifier "Change")

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

	Job parameters	Byte n+1	Byte	Offset
Header	Request Header	Request Reference	RequestID	0
		Axis	Number of parameters	2

	Job parameters	Byte n+1	Byte	Offset
Details	1. parameter	Attribute	Number of elements	4
		Parameter number		6
		Subindex		8
	...			
	nth parameter	Attribute	Number of elements	
		Parameter number		
		Subindex		
Only write values for parameter (change)	1. parameter value(s)	Format	Number of values	
		Values		
	...			
	nth parameter value(s)	Format	Number of values	
		Values		
		...		

Structure after Base Mode Parameter Access - Parameter Request

The following structure is defined for the subsequent Parameter Response. This is received after a request with _readRecord (request).

	Parameter response	Byte n+1	Byte	Offset
	Request Header	Request Reference mirrored	RequestID mirrored or error	0
		Axis no./DO ID mirrored	Number of parameters	2
Only read values for parameter (request) / Error values for negative response only	1. parameter value(s)	Format	Number of values	4
		Values or error codes		6
		...		
	...			
Only read values for parameter (request) / Error values for negative response only	nth parameter value(s)	Format	Number of values	
		Values or error codes		
		...		

Structure after Base Mode Parameter Access - Parameter Response

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

Request Reference

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

Description of fields in parameter job and response

Field	Data type	Values	Comments
Job reference	Unsigned8	0x01 to 0xFF	
	Unique identification of the job/response pair for the master. The master changes the job reference with each new job. The slave mirrors the job reference in its response.		
Job identifier	Unsigned8	0x01 0x02	Read job Write job
	Specifies the type of job. In the case of a write job, the changes are made in a volatile memory (RAM). A save operation is needed in order to transfer the modified data to the non-volatile memory (p0971, p0977).		
Response ID	Unsigned8	0x01 0x02 0x81 0x82	Read job(+) Write job(+) Read job(-) Write job(-)
	Mirrors the job identifier and specifies whether job execution was positive or negative. Negative means: Cannot execute part or all of job. The error values are transferred instead of the values for each subresponse.		
Drive object number	Unsigned8	0x01 to 0xFE	Number
	Setting for the drive object number on a drive unit with more than one drive object. Different drive objects with separate parameter number ranges can be accessed over the same DPV1 connection.		
Number of parameters	Unsigned8	0x01 to 0x27	No. 1 to 39 Limited by DPV1 telegram length
	Defines the number of adjoining areas for the parameter address and/or parameter value for multi-parameter jobs. The number of parameters = 1 for single jobs.		
Attribute	Unsigned8	0x10 0x20 0x30	Value Description Text (not implemented in the case of SINAMICS)
	Type of parameter element accessed		
Number of elements	Unsigned8	0x00 0x01 ... 0xEA	Special function No. 1 to 234 Limited by DPV1 telegram length
	Number of array elements accessed		

Field	Data type	Values	Comments
Parameter number	Unsigned16	0x0001 to 0xFFFF	No. 1 to 65535
	Addresses the parameter accessed		
Subindex	Unsigned16	0x0000 to 0xFFFE	No. 0 to 65534
	Addresses the first array element of the parameter to be accessed		
Format	Unsigned8	0x02 0x03 0x04 0x05 0x06 0x07 0x08	Data type Integer8 Data type Integer16 Data type Integer32 Data type Unsigned8 Data type Unsigned16 Data type Unsigned32 Data type FloatingPoint
		Other values:	See PROFIdrive PROFILE v3.1
		0x40	Zero (without values as a positive subresponse of a write request)
		0x41 0x42 0x43 0x44	Byte Word Double word Error
	The format and number specify the adjoining space containing values in the telegram. Data types in conformity with PROFIdrive Profile shall be preferred for write access. Bytes, words, and double words are also possible as a substitute.		
Number of values	Unsigned8	0x00 to 0xEA	No. 0 to 234 Limited by DPV1 telegram length
	Specifies the number of subsequent values.		
Error values	Unsigned16	0x0000 to 0x00FF	Meaning of error values --> see table 4-29
	The error values in the event of a negative response. If the values make up an odd number of bytes, a zero byte is appended. This ensures the integrity of the word structure of the telegram.		
Values	Unsigned16	0x0000 to 0x00FF	
	The values of the parameter for read or write access. If the values make up an odd number of bytes, a zero byte is appended. This ensures the integrity of the word structure of the telegram.		

For more information on coding PROFIdrive data types, see PROFIdrive-specific data types (Page 351).

See also

Programming example (Page 387)

Reading and writing parameters with Base Mode Parameter Access (Page 356)

10.6.4 Specifications for PROFIBUS and PROFINET IO

Global and local parameters

PROFIdrive makes a distinction between two parameter ranges:

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

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

- Base Mode Parameter Access - local (BMPL)
- Base Mode Parameter Access - global (BMPG)

Note

The PROFIdrive standard states that "pipelining" of jobs on PROFIdrive drives (PROFINET and PROFIBUS) is not supported. For each drive unit/DO, only one "Write/read data set" per AR (Application Relationship = communication relationship = controller) is possible.

If, however, more than one PROFIdrive drive unit is connected to a controller via PROFIBUS or PROFINET, one job can be processed in parallel for each of these drive units. The maximum number then depends on the controller. The data for SIMOTION is specified in Rule 5 - a maximum of eight concurrent calls is possible in SIMOTION (Page 375).

Specific properties of acyclic communication with PROFIBUS

For communication via PROFIBUS, data set 47 (0x002F) is used to access parameters in PROFIdrive drives. Data set 47 is the parameter access point (PAP). Only the global (BMPG) access option is available on PROFIBUS devices. The desired DOs are addressed by supplying the DO ID in the parameter request job structure.

- Base Mode Parameter Access – global; the drive unit's parameters (all DOs, global and local parameters) can be addressed via all the drive unit's PAPs. The SINAMICS devices have a PAP on each PROFIBUS module. Strictly speaking, the PAP is a data record in the PROFIBUS module.

Specific properties of acyclic communication with PROFINET IO

When using PROFINET nothing changes in the fundamental processes for parameter access; however, there are two PAPs available per DO / module.

Note

The MAP submodule is the DO proxy and hosts the alarm channel along with one or more PAPs (Parameter Access Point). The PAP is a data record in the MAP submodule. MAP is displayed in the GSD file.

In principle, with both BMPL (data set number 0xB02E) and BMPG (data set number 0xB02F) it is possible to access global parameters of a drive via any parameter access point (PAP/MAP) of the drive.

With **BMPL**, when accessing local parameters (DO), addressing automatically takes place at the local parameters of the DO to which the PAP/MAP is assigned. Therefore, there is no need to specify the DO ID in the parameter request structure, or this will be ignored by the drive.

Use case:
Each axis (DO) has its own parameter access point (this is necessarily the case in single-axis drives). The axis is selected using the logical address; the user does not need to worry about managing the DO IDs.
We recommend this type of access for new projects.

With **BMPG**, it is possible to access not only local parameters of a specific DO via the PAP/MAP of the DO, but also local parameters of other DOs. The required DO is addressed via the DO ID in the parameter request structure. This means that in the case of BMPG, a valid DO ID must also be transferred in all cases.

Use case:
If access to the parameters of a multi-axis drive is only to take place via a PAP/MAP (logical address), or if parameters are to be read to DOs that do not have a dedicated MAP/PAP (particularly PROFIBUS devices). In the case of BMPG, the user is responsible for managing the DO IDs for this purpose.
With this type of access, you remain compatible with old or existing user programs which previously ran on PROFIBUS.

10.6.5 Error assessment

Description

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

- Error in the communication (transfer of data)
  For example, the addressed device may not exist and is not switched on. This type of error is indicated with the return values of the system functions and is defined in the description of the system functions in the SIMOTION reference lists.
- Error during the processing of the jobs themselves
  For example, an attempt is made to write to a read-only parameter.
  Error codes for this second type of error are defined for PROFIdrive-compliant drives in the PROFIdrive standard and listed below.
  The ID 0x81 (hex) or 0x82 (hex) response indicates an error for the parameter access. Error codes are returned in the drive unit's response in the P response data block (see table below). The "Format" field in the parameter response can be used to distinguish whether the queried parameter represents an error code or a "true" value. See the Structure after Base Mode Parameter Access - parameter response (Page 359) table, offset 4, "Format". This table also contains the coding for the "Format" field. Code 0x44 (hex) indicates an error code in the "Values" field. Other "Format" values specify the number format (e.g. Bool, Byte, Integer8, etc.) with which the value in the "Values" field was returned.

Note

The error codes up to 0x20 correspond to the PROFIdrive profile. The error codes from 0x65 onwards are manufacturer-specific and may, therefore, vary from drive to drive.

Error codes in Base Mode Parameter Access responses

Error code	Meaning	Comments	Additional info
0x00	Illegal parameter number	Access to a parameter which does not exist	–
0x01	Parameter value cannot be changed.	Modification access to a parameter value which cannot be changed	Subindex
0x02	Lower or upper value limit exceeded	Modification access with value outside value limits	Subindex
0x03	Invalid subindex	Access to a subindex which does not exist	Subindex
0x04	No array	Access with subindex to non-indexed parameter.	–
0x05	Wrong data type	Modification access with a value which does not match the data type of the parameter	–
0x06	Setting not allowed (only reset allowed)	Modification access with a value not equal to 0 in a case where this is not allowed	Subindex
0x07	Description element cannot be changed.	Modification access to a description element which cannot be changed	Subindex
0x09	No description data	Access to a description which does not exist (the parameter value exists)	–
0x0B	No operating priority	Modification access with no operating priority	–

Error code	Meaning	Comments	Additional info
0x0F	No text array exists	Access to a text array which does not exist (the parameter value exists)	–
0x11	Job cannot be executed due to operating mode.	Access is not possible temporarily for unspecified reasons.	–
0x14	Illegal value	Modification access with a value which is within the limits but which is illegal for other permanent reasons (parameter with defined individual values)	Subindex
0x15	Response too long	The length of the present response exceeds the maximum length that can be transferred.	–
0x16	Illegal parameter address	Impermissible or unsupported value for attribute, number of elements, parameter number, subindex, or a combination of these	–
0x17	Illegal format	Write job: illegal or unsupported parameter data format	–
0x18	No. of values inconsistent	Write job: a mismatch exists between the number of values in the parameter data and the number of elements in the parameter address.	–
0x19	Drive object does not exist.	You have attempted to access a drive object that does not exist.	–
0x65	Presently deactivated	You have tried to access a parameter that, although available, is currently inactive (e.g. n control set and access to parameter from V/f control).	–
0x6B	Parameter %s [%s]: no write access with enabled controller	–	–
0x6C	Parameter %s [%s]: unit unknown	–	–
0x6D	Parameter %s [%s]: Write access only in the commissioning state, encoder (p0010 = 4).	–	–
0x6E	Parameter %s [%s]: Write access only in the commissioning state, motor (p0010 = 3)	–	–
0x6F	Parameter %s [%s]: Write access only in the commissioning state, power unit (p0010 = 2)	–	–
0x70	Parameter %s [%s]: Write access only in the quick commissioning mode (p0010 = 1)	–	–
0x71	Parameter %s [%s]: Write access only in the ready state (p0010 = 0)	–	–
0x72	Parameter %s [%s]: Write access only in the commissioning state, parameter reset (p0010 = 30)	–	–
0x73	Parameter %s [%s]: Write access only in the commissioning state, safety (p0010 = 95)	–	–
0x74	Parameter %s [%s]: Write access only in the commissioning state, tech. application/units (p0010 = 5)	–	–

Error code	Meaning	Comments	Additional info
0x75	Parameter %s [%s]: Write access only in the commissioning state (p0010 not equal to 0)	–	–
0x76	Parameter %s [%s]: Write access only in the commissioning state, download (p0010 = 29)	–	–
0x77	Parameter %s [%s] may not be written in download.	–	–
0x78	Parameter %s [%s]: Write access only in the commissioning state, drive configuration (device: p0009 = 3)	–	–
0x79	Parameter %s [%s]: Write access only in the commissioning state, define drive type (device: p0009 = 2)	–	–
0x7A	Parameter %s [%s]: Write access only in the commissioning state, data set basis configuration (device: p0009 = 4)	–	–
0x7B	Parameter %s [%s]: Write access only in the commissioning state, device configuration (device: p0009 = 1)	–	–
0x7C	Parameter %s [%s]: Write access only in the commissioning state, device download (device: p0009 = 29)	–	–
0x7D	Parameter %s [%s]: Write access only in the commissioning state, device parameter reset (device: p0009 = 30)	–	–
0x7E	Parameter %s [%s]: Write access only in the commissioning state, device ready (device: p0009 = 0)	–	–
0x7F	Parameter %s [%s]: Write access only in the commissioning state, device (device: p0009 not equal to 0)	–	–
0x81	Parameter %s [%s] may not be written in download.	–	–
0x82	Transfer of the control authority (master) is inhibited by BI: p0806.	–	–
0x83	Parameter %s [%s]: requested BICO interconnection not possible	BICO output does not supply float values. The BICO input, however, requires a float value.	–
0x84	Parameter %s [%s]: parameter change inhibited (refer to p0300, p0400, p0922)	–	–
0x85	Parameter %s [%s]: access method not defined.	–	–
0xC8	Below the valid values	Modification job for a value that, although within "absolute" limits, is below the currently valid lower limit	–

Error code	Meaning	Comments	Additional info
0xC9	Above the valid values	Modification job for a value that, although within "absolute" limits, is above the currently valid upper limit (e.g. governed by the current inverter rating)	–
0xCC	Write access not permitted	Write access is not permitted because an access key is not available.	–

10.6.6 Additional information for the parameters of a PROFIdrive drive

Description

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

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

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

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

Note

Normally, parameter descriptions are read-only.

10.6.7 System commands in SIMOTION

10.6.7.1 _writeRecord/_readRecord SIMOTION system commands

Description

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

The description of the system functions, their input parameters and return values can be found in the SIMOTION system documentation:

- C2xx reference list
- D4xx/D4xx-2 reference list
- P3xx 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: Access to the logical address of a telegram module or diagnostics address of the device (module 0)
- PROFINET: Data set 0xB02E for BMPG is accessed via the diagnostic address of the MAP submodule of the DOs . The MAP submodule is the DO proxy and hosts the alarm channel along with one or more PAPs (Parameter Access Point). The PAP is a data record in the MAP submodule. In the GSD file, the MAP is stated with submodule ID 0xFFFF and the submodule name "MAP."

Furthermore, the DO ID is only relevant for data set 47 (0x002F) and Global Access (PROFINET 0xB02F). Use data set 47 for compatibility with PROFIBUS user programs.

The diagnostics address of the corresponding PAP is relevant for Local Access (PROFINET IO 0xB02E), the DO ID is not analyzed. Use this type of access for new projects.

As a result, for example in connection with PROFIdrive units, the telegram start address of the PROFIdrive telegram exchanged cyclically with the device is required. Use of the diagnostic address of the MAP submodule is recommended. In SINAMICS drives, the log addresses of the telegram modules can also be used. In this way, an F-controller can also access PROFIsafe submodules.

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

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

- P0918: PROFIBUS address
- P0964: Device identification (manufacturer, version, number of axes, etc.)
- P0965: Profile number (the implemented PROFIdrive version)

- P0978: List of the DO Ids (the set "Axis-No." or "DO-ID")
- P61000: Name of the station (PROFINET devices only)

10.6.7.2 _writeDrive.../_readDrive... SIMOTION system commands

Description

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

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

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

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

See also

Scope for the rules (Page 378)

10.6.7.3 Comparison of the system commands

Description

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

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

Properties of the system commands

The use of the drive-specific _write/_readDrive... system commands 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 387).

Up to SIMOTION V4.1, the use of the _write/_readDrive... commands is restricted to those cases for which there is cyclic data traffic to the associated drive object, because this is required as an input parameter. From this version up, it is possible to transfer the DO ID or axis no. via the **doId** parameter of the system functions. This means that it is possible to communicate with every DO, even if acyclic data traffic is involved.

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 telegram traffic for control) or they are not generally known for "generic programming".

10.6.7.4 Deleting _readDrive and _writeDrive jobs

Description

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

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

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

10.6.8 Rules for using _readRecord and _writeRecord

10.6.8.1 Rule 1 - the job has its own job reference

Each job has its own job reference

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

10.6.8.2 Rule 2 - system functions for asynchronous programming

Description

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

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

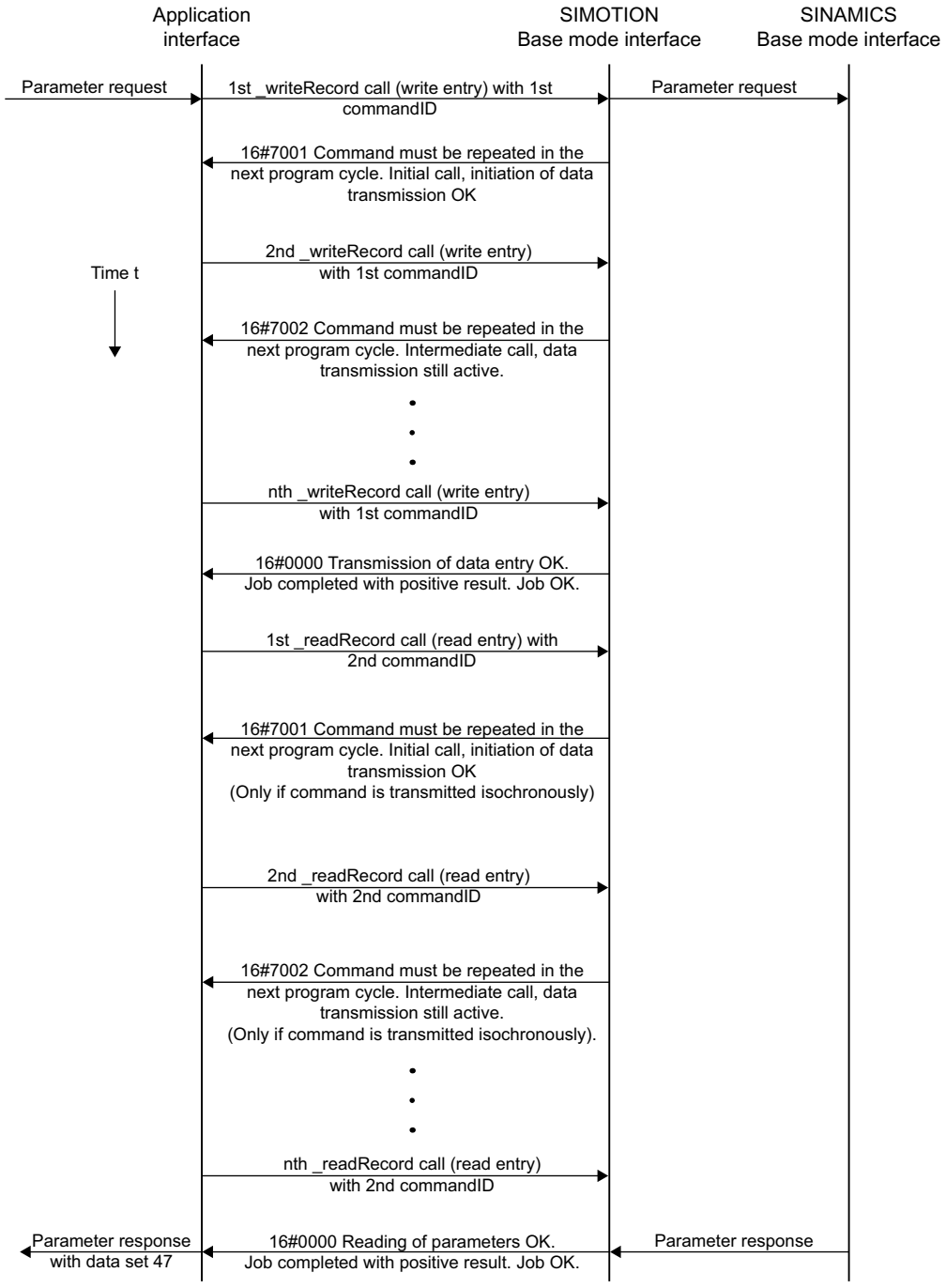


Figure 10-7 Correct processing with the _readRecord and _writeRecord system functions

10.6.8.3 Rule 3 - read/write data record per PROFIdrive drive unit

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 379).

10.6.8.4 Rule 4 - the last call wins for SIMOTION

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

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

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

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

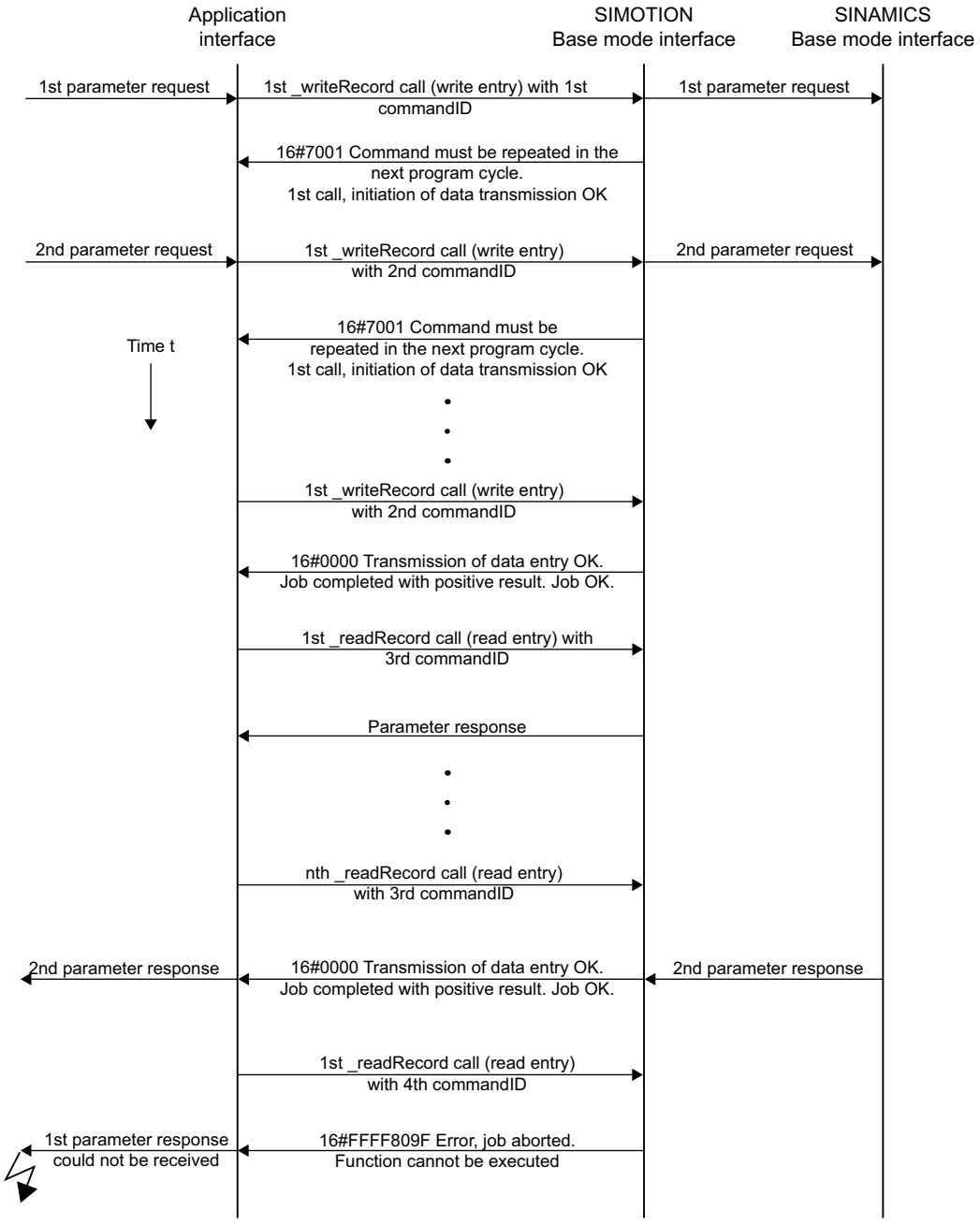


Figure 10-8 The second _writeRecord call wins in case of doubt

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

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

Although according to rule 3 (see Rule 3 (Page 374)) 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.

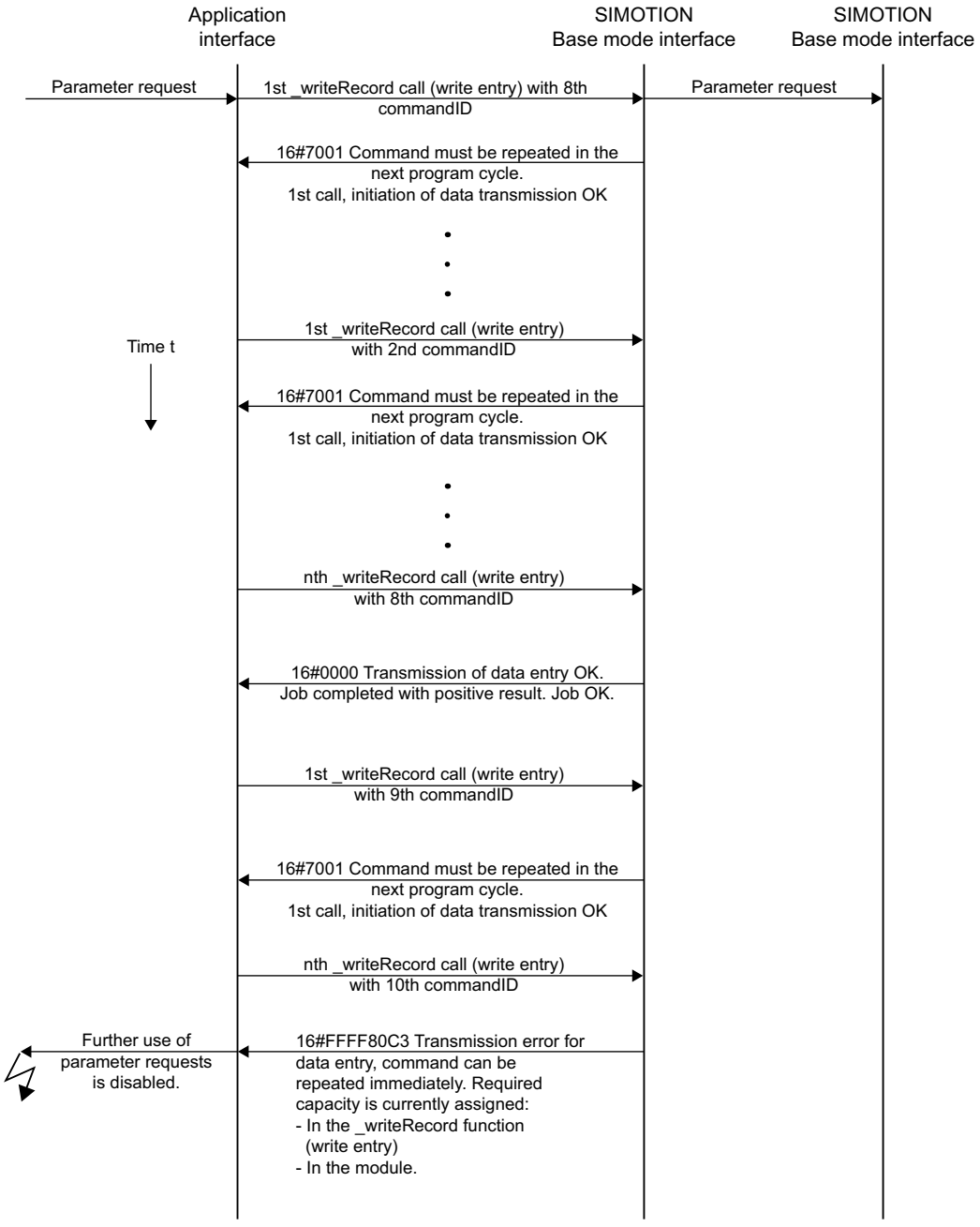


Figure 10-9 Managing 8 jobs simultaneously

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.

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

10.6.9.1 Scope for the rules

Description

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

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

Description

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

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

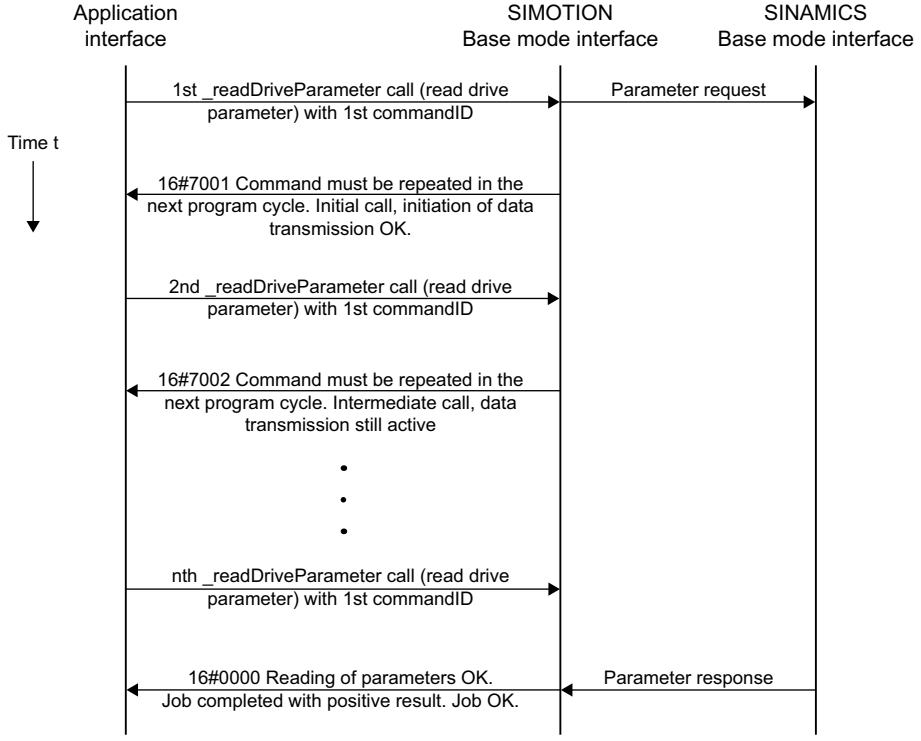


Figure 10-10 Correct processing with the _writeDriveParameter and _readDriveParameter system functions

10.6.9.3 Rule 7 - multiple concurrent calls per target device

Description

The PROFIdrive standard specifies that PROFIdrive units 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 units, 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 380).

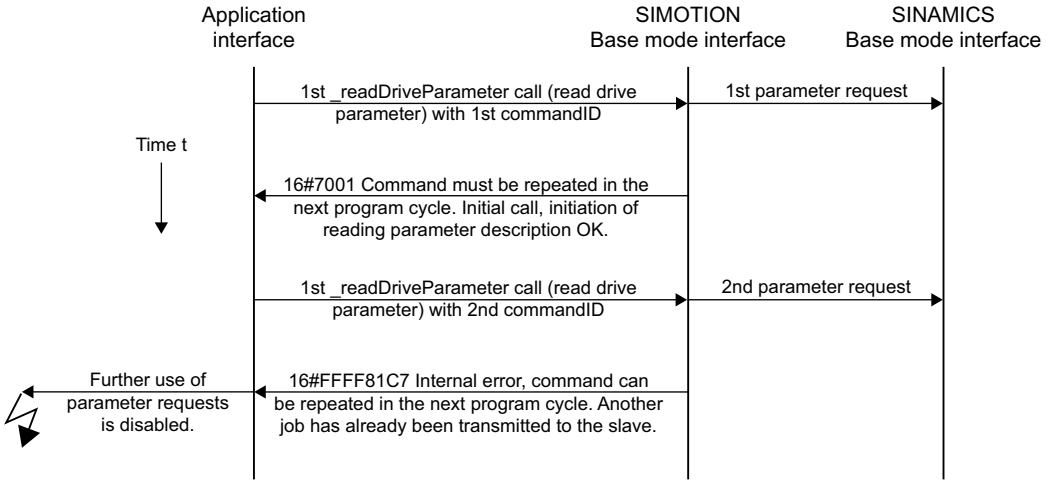


Figure 10-11 Interlocking of several _readDriveParameter jobs on a target device

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

Enable 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.

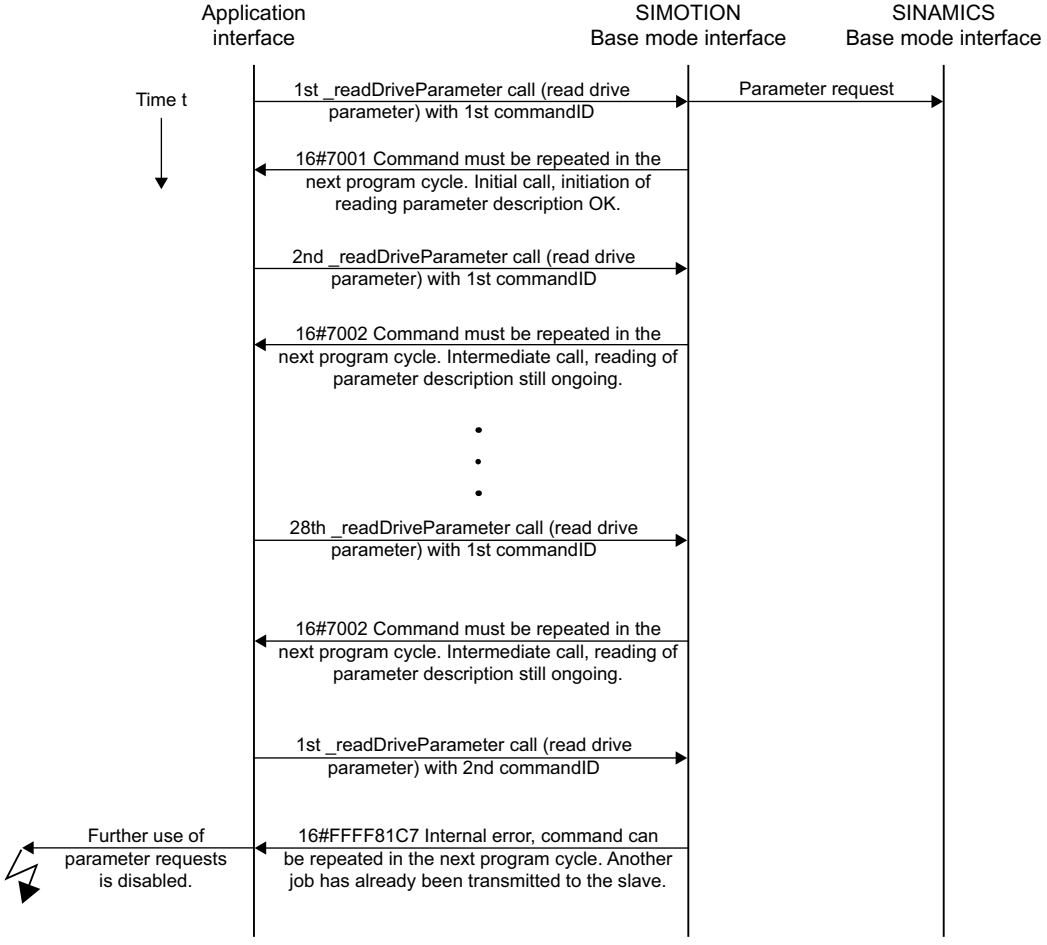


Figure 10-12 Complete processing of a _readDriveParameter required to release the interlock

10.6.9.5 Rule 9 - canceling jobs for an asynchronous call

CommandID is needed to cancel jobs for an asynchronous call

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

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

Note

From V4.1 and up, it is possible to cancel without knowing the commandID, see Deleting _readDrive and _writeDrive jobs (Page 372) .

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.

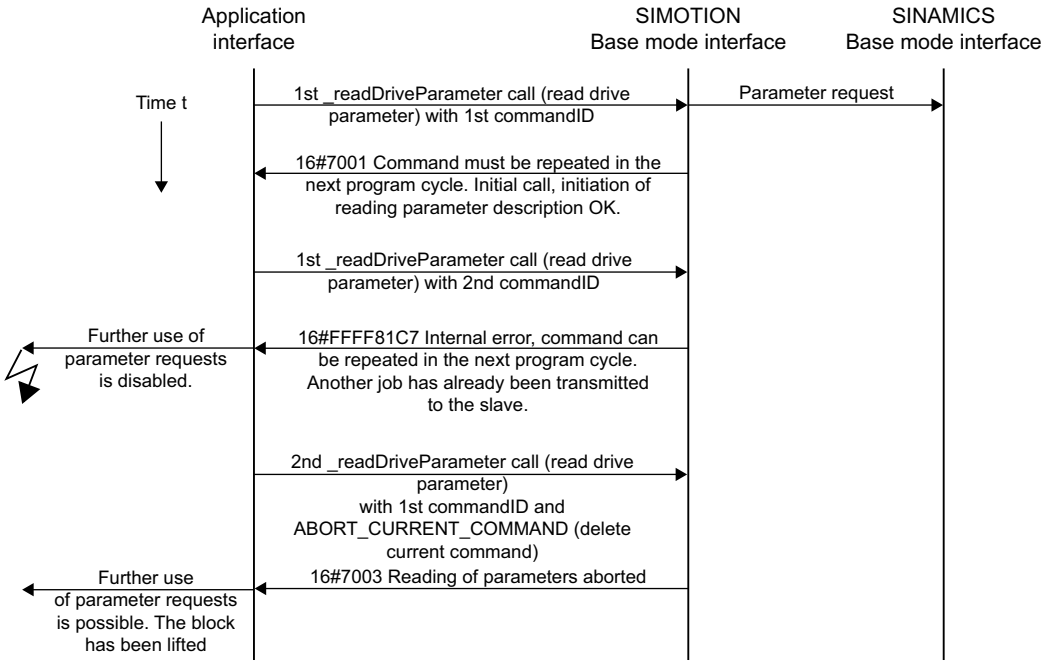


Figure 10-13 Canceling a _readDriveParameter job with known commandID

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.

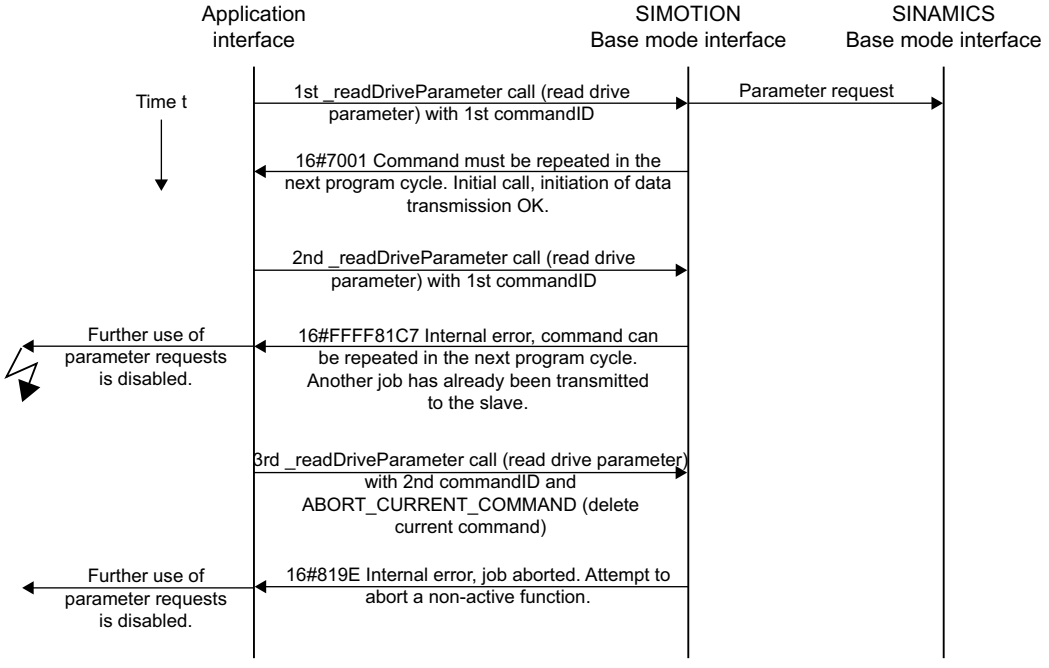


Figure 10-14 No cancelation of a _readDriveParameter job with new CommandID

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.

10.6.9.6 Rule 10 - management of sixteen jobs

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

The controller has limited resources (memory space) available for storing the management data for _write/_readDrive... system function calls. If too many calls are issued in parallel, an error message will be issued, similar to the limit for _read/_writeRecord in Section Maximum Number of Calls (Page 375).

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.

10.6.9.7 Rule 11 - parallel jobs for different drive devices

Parallel jobs to different drive units 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 D445-2 controller uses the SINAMICS Integrated of a D445-2 (for example) as first PROFIdrive drive unit and the CX32-2 expansion module as second PROFIdrive drive unit.

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 (CX32-2)) 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 (CX32-2). 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.

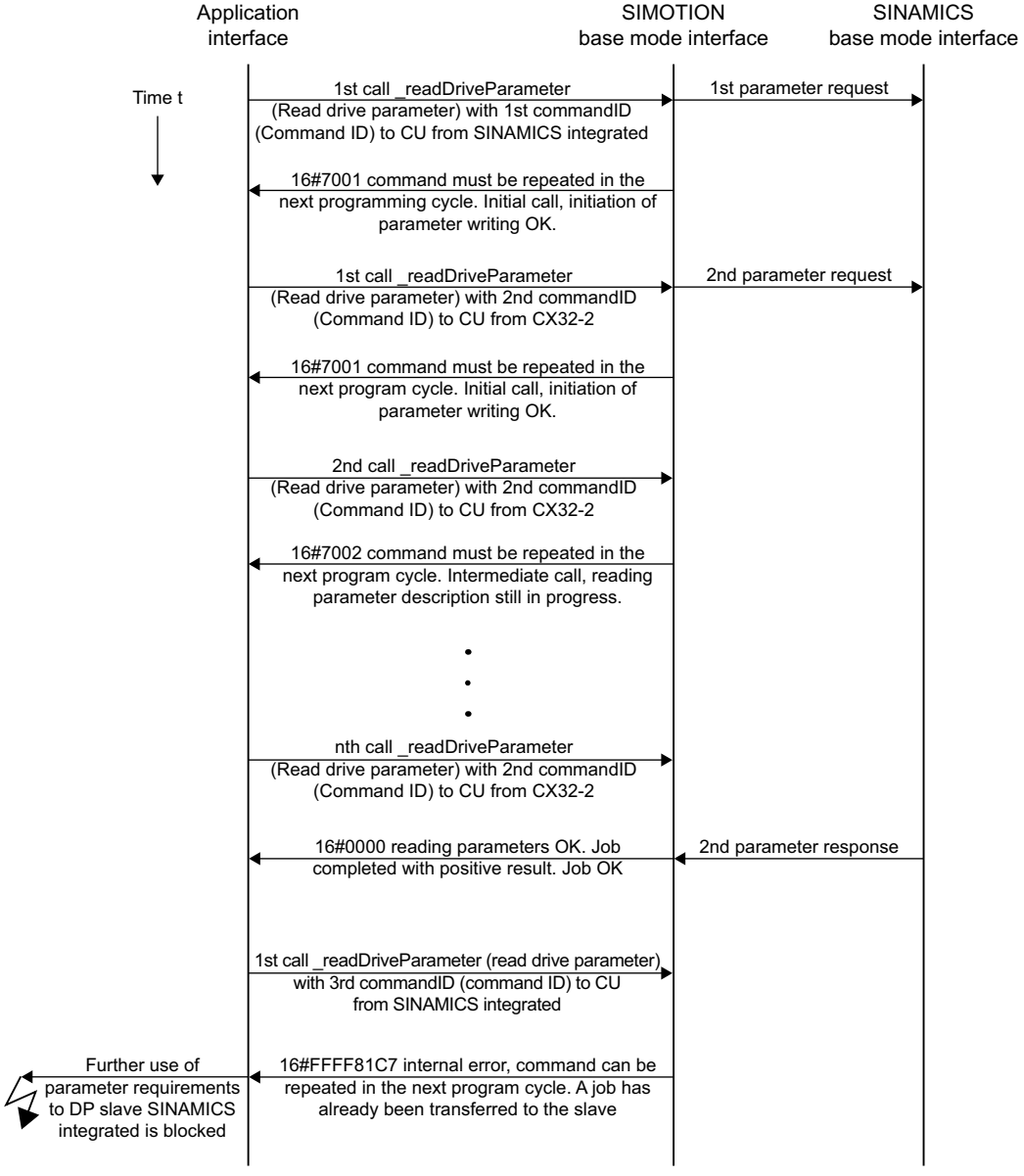


Figure 10-15 Parallel processing of _readDriveParameter jobs to different drive devices of a controller

10.6.10 Special features

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

SIMOTION buffers the data of up to 64 drive objects

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

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

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

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

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

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

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

Note

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

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

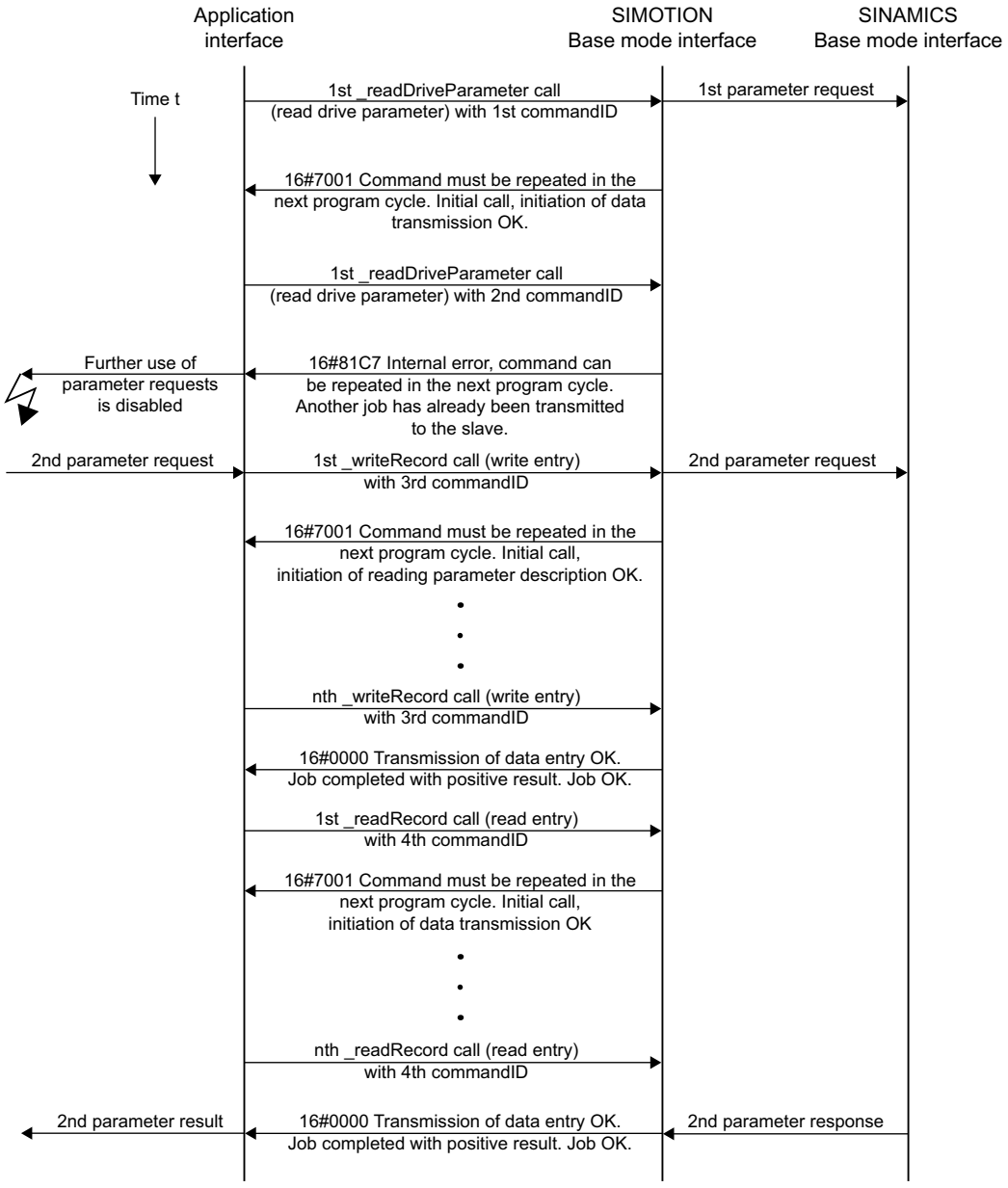


Figure 10-16 Mixed use of _readDrive... and _read/_writeRecord

10.6.10.3 Rule 14 - interlocking for the mixed use of commands

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

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

- _writeRecord/_readRecord SIMOTION system commands

and

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

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

10.6.11 Program examples

10.6.11.1 Programming example

Description

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

Example

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

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

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

 // declare struct of request
 Request : STRUCT
    Header : Header_Type_Request;
    ParameterAddress : Parameter_Address_Request;
 END_STRUCT;
```

```
// declare struct of header response
 Header_Type_Response : STRUCT
 Response_Reference : USINT;
 Response_Id : USINT;
 Axis : USINT;
 Number_Of_Parameter : USINT;
 END_STRUCT;

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

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

Implementation

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

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

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

 // next step
 program_step := 1;

// execute _writeRecord -----------------------------------------------
```

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

Appendix

11

11.1 Standard PROFIBUS/PROFINET data types (only available in English)

Subset of IEC 61158-5 standard data types for use in PROFIBUS/PROFINET profiles

Coding of most of the data types in these profile guidelines is defined in IEC 61158-6:2003, clause 6.2. However, the usage in practice is different in some cases. Thus it is highly recommended to follow the definitions and hints within this annex. The next edition of IEC 61158 is supposed to comply with the content of this annex.

Boolean

A Boolean is representing a data type that only can have two different values i.e. TRUE and FALSE. Hint: for efficiency reasons this data type is not used in application profiles.

Numeric Identifier	Data type name	Value range	Resolution	Length
1	Boolean	True/false	-	1 Octet

Bits	7	6	5	4	3	2	1	0	
True	x	x	x	x	x	x	x	x	0x01 to 0xFF
False	0	0	0	0	0	0	0	0	0x00

Integer16

An Integer16 is representing a signed number depicted by 16 bits.

Numeric Identifier	Data type name	Value range	Resolution	Length
3	Integer16	$-32768 \leq i \leq 32767$	1	2 Octets

In two's complement; the most significant bit (MSB) is the bit after the sign (SN) in the first Byte.

SN = 0: positive numbers and zero

SN = 1: negative numbers

Bits	7	6	5	4	3	2	1	0
Octets 1	SN	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Octets 2	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Integer32

An Integer32 is representing a signed number depicted by 32 bits.

Numeric Identifier	Data type name	Value range	Resolution	Length
4	Integer32	$-2^{31} \leq i \leq 2^{31}-1$	1	4 Octets

In two's complement; the most significant bit (MSB) is the bit after the sign (SN) in the first Byte.

SN = 0: positive numbers and zero

SN = 1: negative numbers

Bits	7	6	5	4	3	2	1	0
Octets 1	SN	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
Octets 2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
Octets 3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Octets 4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Integer64

An Integer64 is representing a signed number depicted by 64 bits.

Numeric Identifier	Data type name	Value range	Resolution	Length
55	Integer64	$-2^{62} \leq i \leq 2^{62}-1$	1	8 Octets

In two's complement; the most significant bit (MSB) is the bit after the sign (SN) in the first Byte.

SN = 0: positive numbers and zero

SN = 1: negative numbers

Bits	7	6	5	4	3	2	1	0
Octets 1	SN	2^{62}	2^{61}	2^{60}	2^{59}	2^{58}	2^{57}	2^{56}
Octets 1	2^{55}	2^{54}	2^{53}	2^{52}	2^{51}	2^{50}	2^{49}	2^{48}
Octets 1	2^{47}	2^{46}	2^{45}	2^{44}	2^{43}	2^{52}	2^{41}	2^{40}
Octets 1	2^{39}	2^{38}	2^{37}	2^{36}	2^{35}	2^{34}	2^{33}	2^{32}
Octets 1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
Octets 2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
Octets 3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Octets 4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Unsigned8

An Unsigned8 is representing an unsigned number depicted by 8 bits ("enumerated"). Some application profiles (e.g. PROFIsafe) are using this data type for the coding of individual single bits.

Numeric Identifier	Data type name	Value range	Resolution	Length
5	Unsigned8	$0 \leq i \leq 255$	1	1 Octet

Table 11-1 Enumerated:

Bits	7	6	5	4	3	2	1	0
Octets 1	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Table 11-2 Single bits:

Bits	7	6	5	4	3	2	1	0
Octets 1	7	6	5	4	3	2	1	0

Unsigned16

An Unsigned16 is representing an unsigned number depicted by 16 bits ("enumerated"). Some application profiles (e.g. PROFIsafe) are using this data type for the coding of individual single bits.

Numeric Identifier	Data type name	Value range	Resolution	Length
6	Unsigned16	0 ≤ i ≤ 65535	1	2 Octets

Table 11-3 Enumerated:

Bits	7	6	5	4	3	2	1	0
Octets 1	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8
Octets 2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Table 11-4 Single bits:

Bits	7	6	5	4	3	2	1	0
Octets 1	15	14	13	12	11	10	9	8
Octets2	7	6	5	4	3	2	1	0

Unsigned32

An Unsigned32 is representing an unsigned number depicted by 32 bits ("enumerated"). Some application profiles (e.g. PROFIsafe) are using this data type for the coding of individual single bits.

Numeric Identifier	Data type name	Value range	Resolution	Length
7	Unsigned32	0 ≤ i ≤ 4294967295	1	4 Octets

Table 11-5 Enumerated:

Bits	7	6	5	4	3	2	1	0
Octets 1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
Octets 2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}

Bits	7	6	5	4	3	2	1	0
Octets 3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Octets 4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Table 11-6 Single bits:

Bits	7	6	5	4	3	2	1	0
Octets 1	31	30	29	28	27	26	25	24
Octets 2	23	22	21	20	19	18	17	16
Octets 3	15	14	13	12	11	10	9	8
Octets 4	7	6	5	4	3	2	1	0

Unsigned64

An Unsigned64 is representing a signed number depicted by 64 bits.

Numeric Identifier	Data type name	Value range	Resolution	Length
56	Unsigned64	$0 \le i \le 2^{64}-1$	1	8 Octets

In two's complement; the most significant bit (MSB) is the bit after the sign (SN) in the first Byte.

SN = 0: positive numbers and zero

SN = 1: negative numbers

Bits	7	6	5	4	3	2	1	0
Octets 1	2^{63}	2^{62}	2^{61}	2^{60}	2^{59}	2^{58}	2^{57}	2^{56}
Octets 2	2^{55}	2^{54}	2^{53}	2^{52}	2^{51}	2^{50}	2^{49}	2^{48}
Octets 3	2^{47}	2^{46}	2^{45}	2^{44}	2^{43}	2^{52}	2^{41}	2^{40}
Octets 4	2^{39}	2^{38}	2^{37}	2^{36}	2^{35}	2^{34}	2^{33}	2^{32}
Octets 5	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
Octets 6	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
Octets 7	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Octets 8	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Float32

A Float32 is representing a number defined by ANSI/IEEE 754 as single precision.

Numeric Identifier	Data type name	Value range	Resolution	Length
8	Float32	refer to ANSI/IEEE 754	refer to ANSI/IEEE 754	4 Octets

SN: sign 0 = positive, 1 = negative.

Bits	7	6	5	4	3	2	1	0
Octets 1	Exponent (E)							
	SN	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}
Octets 2	(E)	Fraction (F)						
	2^{0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}

Bits	7	6	5	4	3	2	1	0
Octets 3	Fraction (F)							
	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}
Octets 4	Fraction (F)							
	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}	2^{-21}	2^{-22}	2^{-23}

Float64

A Float64 is representing a number defined by ANSI/IEEE 754 as single precision.

Numeric Identifier	Data type name	Value range	Resolution	Length
15	Float64	refer to ANSI/IEEE 754	refer to ANSI/IEEE 754	8 Octets

SN: sign 0 = positive, 1 = negative.

Bits	7	6	5	4	3	2	1	0
Octets 1	Exponent (E)							
	SN	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}
Octets 2	Exponent (E)				Fraction (F)			
	2^{3}	2^{2}	2^{1}	2^{0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}
Octets 3	Fraction (F)							
	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}
Octets 4	Fraction (F)							
	2^{-13}	2^{-14}	2^{-15}	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}
Octets 5	Fraction (F)							
	2^{-21}	2^{-22}	2^{-23}	2^{-24}	2^{-25}	2^{-26}	2^{-27}	2^{-28}
Octets 6	Fraction (F)							
	2^{-29}	2^{-30}	2^{-31}	2^{-32}	2^{-33}	2^{-34}	2^{-35}	2^{-36}
Octets 7	Fraction (F)							
	2^{-37}	2^{-38}	2^{-39}	2^{-40}	2^{-41}	2^{-42}	2^{-43}	2^{-44}
Octets 8	Fraction (F)							
	2^{-45}	2^{-46}	2^{-47}	2^{-48}	2^{-49}	2^{-50}	2^{-51}	2^{-52}

Visible String

This data type is defined as the ISO 646 string type. Characters are based on 8 Bit ASCII.

Numeric Identifier	Data type name	Value range	Resolution	Length
9	VisibleString	refer to ISO 646	-	variable

Bits	7	6	5	4	3	2	1	0
Octets 1	1. Character							
Octets 2	2. Character							
...	...							
Octets n	n. Character							

OctetString

An OctetString is an ordered sequence of Bytes, numbered from 1 to n.

Numeric Identifier	Data type name	Value range	Resolution	Length
10	OctetString	-	-	variable

Bits	7	6	5	4	3	2	1	0
Octet 1	1. Character							
Octet 2	2. Character							
...	...							
Octet n	n. Character							

BYTE

A Byte is 1 octet.

Numeric Identifier	Data type name	Value range	Resolution	Length
22	Byte	-	-	1 Octet

Bits	7	6	5	4	3	2	1	0
Octet 1	1. Byte							

WORD

A Word is an ordered sequence of 2 octets.

Numeric Identifier	Data type name	Value range	Resolution	Length
23	Word	-	-	2 Octets

Bits	7	6	5	4	3	2	1	0
Octet 1	1. Byte							
Octet 2	2. Byte							

DWORD

A DWord is an ordered sequence of 4 octets.

Numeric Identifier	Data type name	Value range	Resolution	Length
24	DWord	-	-	4 Octets

Bits	7	6	5	4	3	2	1	0
Octet 1	1. Byte							
Octet 2	2. Byte							

Bits	7	6	5	4	3	2	1	0
Octet 3	3. Byte							
Octet 4	4. Byte							

LWORD

A LWord is an ordered sequence of 8 octets.

Numeric Identifier	Data type name	Value range	Resolution	Length
25	LWord	-	-	8 Octets

Bits	7	6	5	4	3	2	1	0
Octet 1	1. Byte							
Octet 2	2. Byte							
Octet 3	3. Byte							
Octet 4	4. Byte							
Octet 5	5. Byte							
Octet 6	6. Byte							
Octet 7	7. Byte							
Octet 8	8. Byte							

TimeOfDay

This data type is composed of two elements of unsigned values representing the time of day and the date. The first element is an Unsigned32 data type and contains the number of milliseconds since midnight, where midnight = 0.

The second element is of type Unsigned 16 containing the number of completed days since January 1, 1984.

Numeric Identifier	Data type name	Value range	Resolution	Length
12	TimeOfDay	$0 \le i \le (2^{28}-1)$ ms $0 \le i \le (2^{16}-1)$ days	-	6 Octets

The time is interpreted as 32 bit value. The first 4 (MSB) bits have the value zero. The date indication is coded as 16 bit value.

Bits	7	6	5	4	3	2	1	0	
Octets 1	0	0	0	0	2^{27}	2^{26}	2^{25}	2^{24}	Number of millisec-onds since midnight
Octets 2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
Octets 3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
Octets 4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
Octets 5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	days since January 1st, 1984
Octets 6	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	

TimeDifference

This data type is composed of two elements of unsigned values and expresses the difference in time. The first element is an Unsigned32 data type that contains the fractional portion of one day in milliseconds. The optional second element is an Unsigned16 data type that contains the difference in days.

Numeric Identifier	Data type name	Value range	Resolution	Length
13	TimeDifference	$0 \le i \le (2^{32}-1)$ ms $0 \le i \le (2^{16}-1)$ days	-	4 or 6 Octets

The time is interpreted as 32 bit value. The first 4 (MSB) bits have the value zero. The date indication is coded as 16 bit value.

Bits	7	6	5	4	3	2	1	0	
Octets 1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	Number of milliseconds (of one day)
Octets 2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
Octets 3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
Octets 4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
Octets 5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	Number of days (optional)
Octets 6	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	

Date

This data type is composed of six elements of unsigned values and expresses calendar date and time. The first element is an Unsigned16 data type and gives the fraction of a minute in milliseconds. The second element is an Unsigned8 data type and gives the fraction of an hour in minutes. The third element is an Unsigned8 data type and gives the fraction of a day in hours with the most significant bit indicating Standard Time or Daylight Saving Time. The fourth element is an Unsigned8 data type. Its upper three (3) bits give the day of the week and its lower five (5) bits give the day of the month. The fifth element is an Unsigned8 data type and gives the month. The last element is Unsigned8 data type and gives the year. The values 0 ... 50 correspond to the years 2000 to 2050; the values 51 ... 99 correspond to the years 1951 to 1999.

Numeric Identifier	Data type name	Value range	Resolution	Length
50	Date	0 ms ≤ i ≤ 99 years	-	7 Octets (Unsigned16 + 5 x Unsigned8)

Bits	7	6	5	4	3	2	1	0	
Octet 1	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	0...59999 milliseconds
Octet 2	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
Octet 3	res	res	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	0...59 minutes
Octet 4	SU	res	res	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	0...23 hours
Octet 5	day of week			day of month					1...7 day of week 1...31 day of month
	2^{2}	2^{1}	2^{0}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	

Bits	7	6	5	4	3	2	1	0	
Octet 6	res	res	2^5	2^4	2^3	2^2	2^1	2^0	1...12 month
Octet 7	res	2^6	2^5	2^4	2^3	2^2	2^1	2^0	0...99 years

SU: Standard Time = 0; Daylight Saving Time = 1

day of week: 0 Monday = 1; Tuesday =2.....Sunday = 7

res = reserved

TimeOfDay without date indication

This data type consists of one element of an unsigned value and expresses the time of day without date indication. The element is an Unsigned32 data type and contains the time after midnight in milliseconds.

Numeric Identifier	Data type name	Value range	Resolution	Length
52	TimeOfDay without date indication	$0 \le i \le (2^{28}-1)$	ms	4 Octets (Unsigned32)

The time is interpreted as 32 bit value.

Bits	7	6	5	4	3	2	1	0	
Octet 1	0	0	0	0	2^{27}	2^{26}	2^{25}	2^{24}	Number of millisec-onds since midnight
Octet 2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
Octet 3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	
Octet 4	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	

TimeDifference with Date indication

This data type is composed of two elements of unsigned values that express the difference in time. The first element is an Unsigned32 data type that provides the fractional portion of one day in milliseconds. The second element is an Unsigned16 data type that provides the difference in number of days.

Numeric Identifier	Data type name	Value range	Resolution	Length
53	TimeDifference with date indication	$0 \le i \le (2^{32}-1)$ ms $0 \le i \le (2^{16}-1)$ days	ms, days	6 Octets (Unsigned32 + Unsigned16)

The time is interpreted as 32 bit value. The date indication is coded as 16 bit value.

Bits	7	6	5	4	3	2	1	0	
Octet 1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	Number of millisec-onds (of one day)
Octet 2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
Octet 3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	
Octet 4	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
Octet 5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	Number of days
Octet 6	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	

TimeDifference without Date indication

This data type consists of one element of an unsigned value that expresses the difference in time. he element is an Unsigned32 data type providing the fractional portion of one day in milliseconds.

Numeric Identifier	Data type name	Value range	Resolution	Length
54	TimeDifference without date indication	$0 \le i \le (2^{32}-1)$ ms	ms, days	4 Octets (Unsigned32)

The time is interpreted as 32 bit value.

Bits	7	6	5	4	3	2	1	0	
Octet 1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	Number of millisec-onds (of one day)
Octet 2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
Octet 3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
Octet 4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	

NetworkTime

This data type is based on the RFC 1305 standard and composed of two unsigned values that express the network time related to a particular date. Its semantic has changed in IEC 61158-6:2003.

The first element is an Unsigned32 data type that provides the network time in seconds since 1.1.1900 0.00,00(UTC) or since 7.2.2036 6.28,16(UTC) for time values less than 0x9DFF4400, which represents the 1.1.1984 0:00,00(UTC). The second element is an Unsigned32 data type that provides the fractional portion of seconds in $1/2^{32}$ s. Rollovers after 136 years are not automatically detectable and are to be maintained by the application.

Numeric Identifier	Data type name	Value range	Resolution	Length
58	NetworkTime	Byte 1 to 4: $0 \le i \le (232-1)$ Byte 5 to 8: $0 \le i \le (232-1)$	S $(1/2^{32})$ s	8 Octets (Unsigned32 + Un-signed32)

Bits	7	6	5	4	3	2	1	0	
Octet 1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	Number of seconds since 1.1.1900. Rollover af-ter 136 years. Thus, next would be 7.2.2036.
Octet 2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
Octet 3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
Octet 4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
Octet 5	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	Fractional portion of seconds: $1/2^{32}$ s
Octet 6	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
Octet 7	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
Octet 8	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	

The Time Stamp mechanism in PROFIBUS DP is using this data type. However, the semantic is slightly different:

- The least significant Bit of the fractional portion (2^0) is device internally used to indicate a synchronized or unsynchronized state of the clock time.

NetworkTimeDifference

This data type is composed of an integer value and of an unsigned value that express the difference in network time. The first element is an Integer32 data type that provides the network time difference in seconds. The second element is an Unsigned32 data type that provides the fractional portion of seconds in $1/2^{32}$ s.

Numeric Identifier	Data type name	Value range	Resolution	Length
59	NetworkTimeDifference	Byte 1 to 4: $-2^{31} \le i \le (2^{31}-1)$ Byte 5 to 8: $0 \le i \le (2^{32}-1)$	S $(1/2^{32})$ s	8 Octets (Integer32 + Unsigned32)

Bits	7	6	5	4	3	2	1	0	
Octet 1	SN	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	Signed number of seconds
Octet 2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
Octet 3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	
Octet 4	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
Octet 5	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	Fractional portion of seconds: $1/2^{32}$ s
Octet 6	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
Octet 7	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	
Octet 8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	

See also

PROFIdrive-specific data types (Page 351)

11.2 Profile-specific PROFIBUS/PROFINET data types (only available in English)

Existing PROFIBUS/PROFINET profile specific data types

This topic contains the profile specific data types of PROFIBUS/PROFINET.

Float32+Unsigned8 (former "DS33")

This data structure consists of the value and the status of a Float32 parameter. The parameter can be an input or output..

Numeric Identifier	Data type name	Value range	Resolution	Length
101	Float32 +Unsigned8	See Float32 and Unsigned8	-	5 Octets

SN: sign 0 = positive, 1 = negative

Bits	7	6	5	4	3	2	1	0
Octets 1	Exponent (E)							
	SN	2^7	2^6	2^5	2^4	2^3	2^2	2^1
Octets 2	(E)	Fraction (F)						
	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
Octets 3	Fraction (F)							
	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}
Octets 4	Fraction (F)							
	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}	2^{-21}	2^{-22}	2^{-23}
Octet 5	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Unsigned8+Unsigned8 (former "DS34")

This data structure consists of the value and the status of the Unsigned8 parameter. The parameter can be an input or output.

Numeric Identifier	Data type name	Value range	Resolution	Length
102	Unsigned8 + Unsigned8	See Unsigned8	-	2 Octets

In two’s complement; the most significant bit (MSB) is the bit after the sign (SN) in the first Byte.

Bits	7	6	5	4	3	2	1	0
Octet 1	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Octet 2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

OctetString2+Unsigned8 (former "DS35")

This data structure consists of the value and the status of the OctetString parameter. The parameter can be input or output.

Numeric Identifier	Data type name	Value range	Resolution	Length
103	OctetString2 + Unsigned8	see OctetString2 and Unsigned8	-	3 Octets

Bits	7	6	5	4	3	2	1	0
Octet 1	1. Byte							
Octet 2	2. Byte							
Octet 3	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Unsigned16_S

This data structure consists of the value and the status embedded in an unsigned 16 data type. The parameter using this data type can be input or output.

Numeric Identifier	Data type name	Value range	Resolution	Length
104	Unsigned16_S	0 to 2^{13} + 0 to 3	1	2 Octets

Bits	7	6	5	4	3	2	1	0
Octets 1	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6
Octets 2	2^5	2^4	2^3	2^2	2^1	2^0	St1	St0

st1 (bit1)	st0 (bit0)	Meaning
0	0	input channel: bad (value is fail-safe value) output channel: reserved
0	1	input channel: simulation output channel: reserved
1	0	input channel: uncertain output channel: reserved
1	1	input channel: good output channel: reserved

Integer16_S

This data structure consists of the value and the status embedded in an integer 16 data type. The parameter using this data type can be input or output.

Numeric Identifier	Data type name	Value range	Resolution	Length
105	Integer16_S	- 2^{12} to 2^{12} –1 + 0 to 3	1	2 Octets

SN: sign 0 = positive, 1 = negative

Bits	7	6	5	4	3	2	1	0
Octets 1	SN	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}
Octets 2	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	St1	St0

st1 (bit1)	st0 (bit0)	Meaning
0	0	input channel: bad (value is fail-safe value) output channel: reserved
0	1	input channel: simulation output channel: reserved
1	0	input channel: uncertain output channel: reserved
1	1	input channel: good output channel: reserved

Unsigned8_S

This data structure consists of the value and the status embedded in an Unsigned8 data type. The parameter using this data type can be an input or output.

Numeric Identifier	Data type name	Value range	Resolution	Length
106	Unsigned8_S	0 to 2^{5} + 0 to 3	1	1 Octet

Bits	7	6	5	4	3	2	1	0
Octets 1	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	St1	St0

st1 (bit1)	st0 (bit0)	Meaning
0	0	input channel: bad (value is fail-safe value) output channel: reserved
0	1	input channel: simulation output channel: reserved
1	0	input channel: uncertain output channel: reserved
1	1	input channel: good output channel: reserved

OctetString_S

This data structure consists of the value and the status embedded in an octet string data type. The parameter using this data type can be an input or output.

In the scope of this profile, this is the preferred data type for digital values. It contains the value of a binary channel as a bit in a value field (ch(x)), and a status information in a status field

(st1(x) and st0(x)) coded as shown below. If a channel is unused, its position in the value field and in the status field has to be set to 0.

The value field and the status field are packed into an OctetString with the bit coding as shown below. This data type is defined as OctetString_S.

Numeric Identifier	Data type name	Value range	Resolution	Length
107	OctetString_S	See OctetString and below	-	n Octets

Bits	bit7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Octet 1	ch(8)	ch(7)	ch(6)	ch(5)	ch(4)	ch(3)	ch(2)	ch(1)
***	***	***	***	***	***	***	***	***
Octet m	ch(n)	ch(n-1)	ch(n-2)	ch(n-3)	ch(n-4)	ch(n-5)	ch(n-6)	ch(n-7)
Octet m+1	st1(4)	st0(4)	st1(3)	st0(3)	st1(2)	st0(2)	st1(1)	st0(1)
***	***	***	***	***	***	***	***	***
Octet 3*m	st1(n)	st0(n)	st1(n-1)	st0(n-1)	st1(n-2)	st0(n-2)	st1(n-3)	st0(n-3)

ch(x) value for channel x; st(x) status information for channel x; 1<x≤n

st1 (bit1)	st0 (bit0)	Meaning
0	0	input channel: bad (value is fail-safe value) output channel: reserved
0	1	input channel: simulation output channel: reserved
1	0	input channel: uncertain output channel: reserved
1	1	input channel: good output channel: reserved

F message trailer with 4 octets

This data structure consists of the status/control byte, consecutive number, and 2 byte CRC parameters in PROFIsafe's V1-mode or status/control byte and 3 byte CRC parameters in PROFIsafe's V2- mode. This data type can be associated with input or output data up to 12 byte. So far the data type "OctetString" with numeric identifier "10" has been used for the coding of the F message trailer. For new product developments the new data type specification shall be considered.

Numeric Identifier	Data type name	Value range	Resolution	Length
110	F message trailer with 4 octets	PROFIsafe: V1-mode and V2-mode	-	4 Octets

Bits	7	6	5	4	3	2	1	0
Octet 1	Status/Control octet							
Octet 2	Consecutive number (V1-mode) or High-octet CRC(V2-mode) *)							

Bits	7	6	5	4	3	2	1	0
Octet 3	CRC *)							
Octet 4	Low-byte CRC (least significant octet) *)							

* Byte ordering according to IEC 61158-5 Type 3

F message trailer with 5 octets

This data structure consists of the status/control byte and 4 byte CRC parameters in PROFIsafe's V2-mode. This data type can be associated with input or output data up to 122 byte.

Numeric Identifier	Data type name	Value range	Resolution	Length
111	F message trailer with 5 octets	PROFIsafe: V2-mode	-	5 Octets

Bits	7	6	5	4	3	2	1	0
Octet 1	Status/Control octet							
Octet 2	1. byte CRC (most significant octet) *)							
Octet 3	2. byte CRC *)							
Octet 4	3. byte CRC *)							
Octet 5	4. byte CRC (least significant octet) *)							

* Byte ordering according to IEC 61158-5 Type 3

F message trailer with 6 octets

This data structure consists of the status/control byte, consecutive number and 4 byte CRC parameters in PROFIsafe's V1-mode. This data type can be associated with input or output data up to 122 byte. So far the data type "Octet String" with numeric identifier "10" has been used for the coding of the F message trailer. For new product developments the new data type specification shall be considered.

Numeric Identifier	Data type name	Value range	Resolution	Length
111	F message trailer with 6 octets	PROFIsafe: V1-mode	-	6 Octets

Bits	7	6	5	4	3	2	1	0
Octet 1	Status/Control octet							
Octet 2	Consecutive number							
Octet 3	1. byte CRC (most significant octet) *)							
Octet 4	2. byte CRC *)							
Octet 5	3. byte CRC *)							
Octet 6	4. byte CRC (least significant octet) *)							

* Byte ordering according to IEC 61158-5 Type 3

See also

PROFIdrive-specific data types (Page 351)

Index

I

L

M

P

R

S

T

U