### 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.

<table>
<thead>
<tr>
<th>Notice Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DANGER</strong></td>
<td>Indicates that death or severe personal injury will result if proper precautions are not taken.</td>
</tr>
<tr>
<td><strong>WARNING</strong></td>
<td>Indicates that death or severe personal injury may result if proper precautions are not taken.</td>
</tr>
<tr>
<td><strong>CAUTION</strong></td>
<td>With a safety alert symbol, indicates that minor personal injury can result if proper precautions are not taken.</td>
</tr>
<tr>
<td><strong>CAUTION</strong></td>
<td>Without a safety alert symbol, indicates that property damage can result if proper precautions are not taken.</td>
</tr>
<tr>
<td><strong>NOTICE</strong></td>
<td>Indicates that an unintended result or situation can occur if the relevant information is not taken into account.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Notice Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WARNING</strong></td>
<td>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.</td>
</tr>
</tbody>
</table>

#### 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.
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 V4.3:

- SIMOTION SCOUT V4.3 (Engineering System of the SIMOTION product range),

Chapters in this manual

This manual describes the communications possibilities for SIMOTION systems.

- Communications functions and services overview
  General information about the communications possibilities provided by SIMOTION.

- 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 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 a separate list of references. This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises 10 documentation packages.

The following documentation packages are available for SIMOTION V4.3:

- SIMOTION Engineering System
- 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 the following topics:

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

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

Please send any questions about the technical documentation (e.g. suggestions for improvement, corrections) to the following e-mail address:
docu.motioncontrol@siemens.com

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 this 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

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreword</td>
<td>3</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>13</td>
</tr>
<tr>
<td>1.1 The communications subject in the SIMOTION documentation</td>
<td>13</td>
</tr>
<tr>
<td>2 Overview of the communication functions and services</td>
<td>15</td>
</tr>
<tr>
<td>2.1 Network options</td>
<td>15</td>
</tr>
<tr>
<td>2.1.1 Introduction</td>
<td>15</td>
</tr>
<tr>
<td>2.1.2 PROFINET</td>
<td>16</td>
</tr>
<tr>
<td>2.1.3 Industrial Ethernet</td>
<td>16</td>
</tr>
<tr>
<td>2.1.4 PROFIBUS</td>
<td>17</td>
</tr>
<tr>
<td>2.1.5 MPI (Multi-Point Interface)</td>
<td>17</td>
</tr>
<tr>
<td>2.1.6 Point-to-point communication (PtP)</td>
<td>18</td>
</tr>
<tr>
<td>2.2 Communications services (or network functions)</td>
<td>18</td>
</tr>
<tr>
<td>2.2.1 Introduction</td>
<td>18</td>
</tr>
<tr>
<td>2.2.2 PG/OP communication services</td>
<td>19</td>
</tr>
<tr>
<td>2.2.3 S7 communication services</td>
<td>19</td>
</tr>
<tr>
<td>2.2.4 S7 basic communication services</td>
<td>20</td>
</tr>
<tr>
<td>2.2.5 “Global data” communication service</td>
<td>20</td>
</tr>
<tr>
<td>2.2.6 PROFINET communication services</td>
<td>21</td>
</tr>
<tr>
<td>2.2.7 Industrial Ethernet communication services</td>
<td>21</td>
</tr>
<tr>
<td>2.2.8 PROFIBUS communication services</td>
<td>22</td>
</tr>
<tr>
<td>2.3 Additional services for the exchange of information</td>
<td>23</td>
</tr>
<tr>
<td>3 PROFIBUS</td>
<td>25</td>
</tr>
<tr>
<td>3.1 PROFIBUS communication</td>
<td>25</td>
</tr>
<tr>
<td>3.1.1 PROFIBUS communication (overview)</td>
<td>25</td>
</tr>
<tr>
<td>3.2 Communication with SIMATIC S7</td>
<td>25</td>
</tr>
<tr>
<td>3.2.1 Possible communication connections between SIMOTION and SIMATIC</td>
<td>25</td>
</tr>
<tr>
<td>3.2.2 SIMOTION as DP slave on a SIMATIC S7</td>
<td>26</td>
</tr>
<tr>
<td>3.2.2.1 Introduction</td>
<td>26</td>
</tr>
<tr>
<td>3.2.2.2 Connecting SIMOTION to a SIMATIC S7 as DP slave with the aid of a GSD file</td>
<td>27</td>
</tr>
<tr>
<td>3.2.2.3 Connecting SIMOTION as an I-slave to a SIMATIC S7</td>
<td>27</td>
</tr>
<tr>
<td>3.2.3 SIMATIC S7 as DP slave on a SIMOTION</td>
<td>29</td>
</tr>
<tr>
<td>3.2.3.1 Introduction</td>
<td>29</td>
</tr>
<tr>
<td>3.2.3.2 Connecting SIMATIC to a SIMOTION device as DP slave with the aid of a GSD file</td>
<td>30</td>
</tr>
<tr>
<td>3.2.3.3 Connecting a SIMATIC S7 CPU as an I-slave to a SIMOTION device</td>
<td>30</td>
</tr>
<tr>
<td>3.2.4 PROFIBUS master-master connection between SIMATIC and SIMOTION</td>
<td>33</td>
</tr>
<tr>
<td>3.2.4.1 Introduction</td>
<td>33</td>
</tr>
<tr>
<td>3.2.4.2 SIMATIC S7 system functions for a PROFIBUS connection</td>
<td>33</td>
</tr>
</tbody>
</table>
Table of contents

4 PROFINET IO ............................................................................................................................................ 39
  4.1 PROFINET IO overview ........................................................................................................................ 39
  4.1.1 PROFINET IO .................................................................................................................................... 39
  4.1.2 Application model ............................................................................................................................... 39
  4.1.3 IO controller ....................................................................................................................................... 40
  4.1.4 IO device .......................................................................................................................................... 40
  4.1.5 PROFINET IO system ......................................................................................................................... 40
  4.1.6 iDevice ............................................................................................................................................ 41
  4.1.7 RT classes ......................................................................................................................................... 41
  4.1.7.1 RT classes for PROFINET IO ...................................................................................................... 41
  4.1.7.2 Send clock and update time ......................................................................................................... 42
  4.1.7.3 Adjustable send clocks and update times .................................................................................... 43
  4.1.7.4 Setting RT classes ....................................................................................................................... 45
  4.1.7.5 PROFINET IO with RT ................................................................................................................ 46
  4.1.7.6 PROFINET IO with IRT - Overview ......................................................................................... 46
  4.1.7.7 PROFINET IO with IRT (High Performance) ............................................................................ 47
  4.1.8 Sync domain ................................................................................................................................... 49
  4.1.9 Isochronous operation and isochronous mode with PROFINET .................................................... 49
  4.1.10 Addressing of PROFINET IO devices ........................................................................................... 50
  4.1.11 Planning and topology for a PROFINET network ......................................................................... 51
  4.1.12 Isochronous applications with PROFINET .................................................................................. 57
  4.1.13 Acyclic communication via PROFINET ......................................................................................... 59
  4.2 Properties of PROFINET IO with SIMOTION ....................................................................................... 60
  4.2.1 Introduction ....................................................................................................................................... 60
  4.2.2 Cycle clock scaling ......................................................................................................................... 62
  4.2.2.1 Cycle clock scaling with PROFINET IO on SIMOTION devices .................................................. 62
  4.2.2.2 Cycle clock scaling for IO accesses ............................................................................................. 64
  4.2.2.3 Bus cycle clocks that can be adjusted for cycle clock scaling to SIMOTION devices ............... 65
  4.2.3 Task system and time response ....................................................................................................... 67
  4.2.3.1 Overview of SIMOTION task system and system cycle clocks ..................................................... 67
  4.2.3.2 BackgroundTask, MotionTask, and IPOSynchronousTask ......................................................... 68
  4.2.3.3 ServoSynchronousTask ............................................................................................................. 70
  4.2.3.4 Fast I/O processing in the ServoSynchronousTask .................................................................... 72
  4.2.4 Devices with two PROFINET IO interfaces ................................................................................... 72
  4.2.4.1 Two PROFINET IO interfaces .................................................................................................. 72
  4.2.4.2 Applications for devices with two PROFINET IO interfaces ..................................................... 74
  4.2.4.3 Controlled sync master .............................................................................................................. 80
  4.2.4.4 Example configuration for controlled sync master .................................................................... 82
  4.2.5 Connection between sync domain and IO systems ........................................................................ 85
  4.2.6 Redundant sync master ................................................................................................................... 85
  4.2.7 Quantity structures .......................................................................................................................... 87
  4.2.8 Media redundancy (MRP and MRPD) .............................................................................................. 88
  4.2.8.1 Media redundancy for SIMOTION ............................................................................................... 88
  4.2.8.2 Information on PROFINET with MRPD ................................................................................... 92
  4.2.8.3 Sample configurations for MRPD rings ..................................................................................... 94
  4.2.8.4 Use of Safety with MRRT .......................................................................................................... 96
  4.3 Configuring PROFINET IO with SIMOTION ..................................................................................... 98
  4.3.1 New in SIMOTION SCOUT as of V4.3 ............................................................................................. 98
  4.3.2 New with SIMOTION SCOUT V4.2 or higher ............................................................................... 99
  4.3.3 Procedure for configuring PROFINET IO with IRT High Performance .................................... 100
  4.3.4 Inserting and configuring SIMOTION D ......................................................................................... 100
### Table of contents

4.3.4.1 General information on inserting and configuring SIMOTION D .................................................. 100
4.3.4.2 Inserting and configuring SIMOTION D4x5-2/D410 PN .................................................................... 101
4.3.4.3 Insert and configure a SIMOTION D4x5-2 incl. CBE30-2 ............................................................. 104
4.3.4.4 Add and configure a CBE30-2 PROFINET board ........................................................................ 106
4.3.5 Adding and configuring SIMOTION P ................................................................................................ 109
4.3.6 Adding and configuring SIMOTION C ................................................................................................ 112
4.3.7 Creating a sync domain .................................................................................................................... 114
4.3.8 Defining send clock and refresh times .............................................................................................. 117
4.3.9 Servo_fast, scaling down of cycle clocks to the servo at the PROFINET interface.......................... 121
4.3.10 Configuring a topology ................................................................................................................... 123
4.3.10.1 Topology ...................................................................................................................................... 123
4.3.10.2 Topology editor (graphical view) .................................................................................................. 124
4.3.10.3 Interconnecting ports via the topology editor (table view) ............................................................ 126
4.3.11 Creating an IO device ..................................................................................................................... 127
4.3.12 Inserting and configuring the SINAMICS S120 ............................................................................... 129
4.3.13 IP address and communication name .............................................................................................. 133
4.3.14 Assigning device names and IP addresses to IO devices .................................................................. 135
4.3.15 IP address and communication name via user program/DCP (Mini-IP-Config) ............................ 138
4.3.16 Configuring media redundancy (V4.3 and higher) ........................................................................ 143
4.3.16.1 Creating ring topology ............................................................................................................... 143
4.3.16.2 Setting up MRP domain ............................................................................................................ 146
4.3.16.3 Configuring media redundancy .................................................................................................. 147
4.4 Configuring direct data exchange (data exchange broadcast) between IO controllers .................. 150
4.4.1 Introduction ....................................................................................................................................... 150
4.4.2 Configuring the sender ...................................................................................................................... 151
4.4.3 Configuring the receiver .................................................................................................................... 152
4.5 Configuring the iDevice ........................................................................................................................ 153
4.5.1 PROFINET IO and I device .............................................................................................................. 153
4.5.2 I device functionality with SIMOTION SCOUT V4.2 or higher ...................................................... 158
4.5.3 Creating an I device .......................................................................................................................... 160
4.5.4 Exporting the GSD file for the I device ............................................................................................. 163
4.5.5 Creating a substitute I device ............................................................................................................ 163
4.5.6 Inserting a substitute I device in the higher-level IO controller ...................................................... 165
4.5.7 Deleting a substitute I device ........................................................................................................... 169
4.6 Loading the communication configuration .......................................................................................... 169
4.6.1 Loading the PROFINET IO configuration ......................................................................................... 169
4.7 Data exchange between SIMATIC and SIMOTION via PROFINET .................................................. 169
4.7.1 Overview of data exchange between SIMATIC and SIMOTION ..................................................... 169
4.7.2 Data exchange through the use of iDevices ..................................................................................... 170
4.7.3 PN to PN coupler ............................................................................................................................. 171
4.7.4 Communication using standard protocols ......................................................................................... 173
4.8 Diagnostic and alarm behavior ........................................................................................................... 174
4.8.1 PROFINET IO alarm and diagnostic messages to SIMOTION .................................................................. 174
4.8.2 Diagnostics model ............................................................................................................................ 174
4.8.3 Alarms on the IO controller ............................................................................................................. 176
4.8.4 Alarms from the IO device to the IO controller ............................................................................... 177
4.8.5 Alarms for direct data exchange between IO controllers .............................................................. 179
4.8.6 Alarms for SINAMICS S120 drives ................................................................................................. 179
4.8.7 System functions for the diagnostics for PROFINET or PROFIBUS .................................................. 180
4.8.8 PROFINET device diagnosis in STEP 7 ........................................................................................... 182
## Table of contents

4.8.9  PROFINET IO and DS0 diagnostic interrupts ................................................................. 183
4.8.9.1 Diagnostic interrupt PROFINET IO maintenance concept ........................................ 183
4.8.9.2 Device model for IO device ........................................................................................ 184
4.8.9.3 PROFINET IO and DS0 diagnostic interrupts ............................................................... 185
5  Ethernet: General information (TCP and UDP connections) .................................................. 189
   5.1  Ethernet interfaces ........................................................................................................... 189
   5.1.1  Overview of Ethernet ................................................................................................ 189
   5.1.2  Properties of the SIMOTION Ethernet interfaces ........................................................ 189
   5.1.3  Using the Ethernet interface ...................................................................................... 190
   5.2  LCom communications library ...................................................................................... 191
   5.3  TCP communication ....................................................................................................... 192
   5.3.1  Overview of TCP communication .............................................................................. 192
   5.3.2  SIMOTION system functions for TCP communication ................................................ 195
   5.3.2.1  Overview of SIMOTION system functions ............................................................... 195
   5.3.2.2  tcpOpenServer() system function ........................................................................... 196
   5.3.2.3  tcpOpenClient() system function ............................................................................ 197
   5.3.2.4  tcpSend() system function ..................................................................................... 198
   5.3.2.5  tcpReceive() system function ................................................................................ 199
   5.3.2.6  tcpCloseConnection() system function ................................................................. 200
   5.3.2.7  tcpCloseServer() system function ......................................................................... 200
   5.3.3  SIMATIC communication blocks, onboard Ethernet interface .................................. 201
   5.3.3.1  Overview of SIMATIC communication blocks ......................................................... 201
   5.3.3.2  UDT65 structure and parameterization .................................................................... 202
   5.3.3.3  Description of communication blocks ..................................................................... 204
   5.3.4  SIMATIC communication blocks for Ethernet CP ....................................................... 207
   5.3.4.1  Overview of SIMATIC communication blocks ........................................................ 207
   5.3.4.2  Ethernet CP configuration ...................................................................................... 208
   5.3.4.3  Description of communication blocks ..................................................................... 208
   5.4  UDP communication ....................................................................................................... 213
   5.4.1  Overview of UDP communication .............................................................................. 213
   5.4.2  SIMOTION system functions for UDP communication ............................................... 215
   5.4.2.1  Overview of SIMOTION system functions ............................................................... 215
   5.4.2.2  udpSend() system function .................................................................................... 215
   5.4.2.3  udpReceive() system function ................................................................................. 216
   5.4.2.4  udpAddMulticastGroupMembership() system function .................................. 217
   5.4.2.5  udpDropMulticastGroupMembership() system function ................................ 218
   5.4.3  SIMATIC communication blocks, onboard Ethernet interface .................................. 218
   5.4.3.1  Overview of SIMATIC communication blocks ........................................................ 218
   5.4.4  SIMATIC communication blocks for Ethernet CP ....................................................... 219
   5.4.4.1  Overview of S7 communication blocks, UDP .......................................................... 219
   5.4.4.2  Ethernet CP configuration ...................................................................................... 220
6  Routing - communication across network boundaries .......................................................... 223
   6.1  What does routing mean? ............................................................................................... 223
   6.2  Configuration of S7 routing ............................................................................................ 224
   6.3  Routing for SIMOTION .................................................................................................... 224
   6.4  Routing with SIMOTION D (example of D4x5 with CBE30) ........................................ 226
   6.5  Routing with SIMOTION D4x5-2 (example of D455-2 DP/PN) .................................... 229
Table of contents

6.6 Routing for SIMOTION D to the SINAMICS integrated ..................................................... 232
6.7 Routing for SIMOTION P350 .......................................................................................... 233
6.8 Routing for SIMOTION P320 .......................................................................................... 235

7 SIMOTION IT.......................................................................................................................... 237
  7.1 SIMOTION IT - overview .............................................................................................. 237
  7.2 Web access to SIMOTION ............................................................................................ 239
  7.3 SIMOTION IT DIAG ...................................................................................................... 240
  7.4 SIMOTION IT OPC XML DA ....................................................................................... 242
  7.5 FTP data transfer ......................................................................................................... 243

8 PROFIsafe............................................................................................................................ 245
  8.1 Communication relationships for drive-based safety .................................................... 245
  8.2 Message frames and signals in drive-based safety ....................................................... 246
  8.3 SIMOTION F proxy functions ..................................................................................... 248
  8.4 PROFIsafe properties for configuration ...................................................................... 249
  8.5 PROFIsafe via PROFINET ............................................................................................ 252
    8.5.1 Principles of I device failsafe proxy .................................................................... 252
    8.5.2 Supported devices and software requirements for I device failsafe proxy .......... 254
    8.5.3 Detailed description/properties of I device failsafe proxy .................................. 256
    8.5.4 PROFIsafe via PROFINET with an F-CPU .......................................................... 257
    8.5.5 Topology overview s I device F-Proxy ................................................................. 258
    8.5.5.1 Topology for I device failsafe proxy for PROFIBUS drive units .................... 258
    8.5.5.2 Topology for I device failsafe proxy for PROFINET drive units .................... 259
    8.5.5.3 Topology for I device failsafe proxy for PROFIBUS and PROFINET drive units .................................................................................................................................................. 260
    8.5.6 Configuring I device failsafe proxy .................................................................... 261
    8.5.7 Adapting the F address in the existing system with PROFIsafe via PROFINET ... 269
        8.5.7.1 General information on shared device .......................................................... 279
        8.5.7.2 General information on shared device .......................................................... 279
        8.5.7.3 General information on shared device .......................................................... 279
        8.5.7.4 Shared device in a STEP 7 project ................................................................ 280
      8.5.6.1 Basic configuration process for I device failsafe proxy ................................. 261
      8.5.6.2 Configuration example for SIMOTION D435 and SINAMICS S120 via PROFINET .................................................................................................................................................. 263
      8.5.6.3 Adapting the F address in the existing project ............................................. 269
      8.5.6.4 Configuration of D435 with S120 on PROFINET and integrated PROFIBUS .. 272
      8.5.6.5 Upgrading an existing system with PROFIsafe via PROFINET to PROFIsafe via PROFINET .................................................................................................................................................. 276
      8.5.6.6 General information on shared device and signals in drive-based safety .... 278
      8.5.7 Shared device via PROFINET .............................................................................. 279
      8.5.7.1 General information on shared device .......................................................... 279
      8.5.7.2 Shared device in a STEP 7 project ................................................................ 280
    8.6 PROFIsafe via PROFINIBUS ....................................................................................... 285
      8.6.1 General information about PROFIsafe on PROFINIBUS ................................ 285
      8.6.2 Supported devices and software requirements for PROFIsafe on PROFINIBUS .................................................. 285
      8.6.3 I-slave failsafe proxy ......................................................................................... 286
        8.6.3.1 Principles of I-slave failsafe proxy ............................................................... 286
        8.6.3.2 Topology for I-slave failsafe proxy for PROFINIBUS drive units ............. 287
        8.6.3.3 PROFIsafe via PROFINIBUS when SIMOTION D is used ...................... 287
      8.6.4 Failsafe data exchange broadcast ...................................................................... 294
        8.6.4.1 Principles of failsafe data exchange broadcast ........................................... 294
        8.6.4.2 Topology of failsafe data exchange broadcast via PROFINIBUS .......... 294

Communication
System Manual, 02/2012 11
Table of contents

8.6.4.3 PROFIsafe via PROFIBUS with fail-safe internode data exchange taking the example of SIMOTION D.................................................................................................................295
8.7 PROFIsafe configuration - acceptance test and reports........................................300
8.8 Additional information on SIMOTION and PROFIsafe........................................300

9 PROFIdrive.........................................................................................................................301
9.1 Why profiles? ................................................................................................................301
9.2 PROFIdrive overview ................................................................................................302
9.3 PROFIdrive base/parameter model ..........................................................................303
9.4 Segmentation in application classes ........................................................................307
9.5 PROFIdrive-specific data types ................................................................................309
9.6 Acyclic communication (Base Mode Parameter Access) ........................................313
9.6.1 Acyclic communication ........................................................................................313
9.6.2 Reading and writing parameters with Base Mode Parameter Access ................314
9.6.3 Parameter request/response data set ..................................................................316
9.6.4 Specifications for PROFIBUS and PROFINET IO .............................................320
9.6.5 Error assessment ...................................................................................................321
9.6.6 Additional information for the parameters of a PROFIdrive drive .......................324
9.6.7 System commands in SIMOTION ........................................................................325
9.6.7.1 _writeRecord/_readRecord SIMOTION system commands .........................325
9.6.7.2 _writeDrive.../readDrive... SIMOTION system commands .......................326
9.6.7.3 Comparison of the system commands ..............................................................327
9.6.7.4 Deleting _readDrive and _writeDrive jobs ..................................................328
9.6.8 Rules for using _readRecord and _writeRecord ..............................................328
9.6.8.1 Rule 1 - the job has its own job reference.......................................................328
9.6.8.2 Rule 2 - system functions for asynchronous programming............................328
9.6.8.3 Rule 3 - read/write data record per PROFIdrive drive unit............................331
9.6.8.4 Rule 4 - the last call wins for SIMOTION .....................................................331
9.6.8.5 Rule 5 - a maximum of eight concurrent calls is possible in SIMOTION ........333
9.6.8.6 Rule 6 - repeated call of system function for asynchronous programming ........335
9.6.8.7 Rule 7 - multiple concurrent calls per target device .....................................336
9.6.8.8 Rule 8 - release the interlocking after the complete processing of a job ..........338
9.6.8.9 Rule 9 - canceling jobs for an asynchronous call .........................................340
9.6.8.10 Rule 10 - management of sixteen jobs ..........................................................343
9.6.8.11 Rule 11 - parallel jobs for different drive devices ........................................343
9.6.10 Special features ...................................................................................................346
9.6.10.1 Rule 12 - data buffering of up to 64 drive objects .........................................346
9.6.10.2 Rule 13 - a mix of system functions can be used ........................................346
9.6.10.3 Rule 14 - interlocking for the mixed use of commands .................................348
9.6.11 Program examples ...............................................................................................348
9.6.11.1 Programming example ................................................................................348

10 Appendix.........................................................................................................................353
10.1 Standard PROFIBUS/PROFINET data types (only available in English) ...........353
10.2 Profile-specific PROFIBUS/PROFINET data types (only available in English) ....364

Index.....................................................................................................................................371
1.1 The communications subject in the SIMOTION documentation

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

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

This manual contains 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.
1.1 The communications subject in the SIMOTION documentation
Overview of the communication functions and services

2.1 Network options

2.1.1 Introduction

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

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

SIMOTION and SIMATIC networks for all applications

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

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

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.
2.1.2 PROFINET

Overview

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

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

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

- With PROFINET IO, you can connect distributed field devices (e.g. digital or analog signal modules) and drives directly to an Industrial Ethernet subnet.

- PROFINET CBA (Component-Based Automation) supports modular solutions for machine and plant construction. You define your automation system as autonomous components, whereby each component consists of independent, self-contained tasks.

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

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

2.1.3 Industrial Ethernet

Overview

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

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

Overview

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

PROFIBUS provides the following communication services:

- **PROFIBUS DP (Distributed Peripherals)** is a communication protocol that is especially suitable for production automation. PROFIBUS DP provides a fast, cyclic and deterministic exchange of process data between a bus DP master and the assigned DP slave devices. PROFIBUS DP supports isochronous communication. The synchronized execution cycles ensure that the data is transmitted at consistently equidistant time intervals.

- **PROFIBUS PA (Process Automation)** expands PROFIBUS DP to provide intrinsically safe data and power transmission according to the IEC 61158-2 standard.

- **PROFIBUS FMS (Fieldbus Message Specification)** is for communication on the cell level, where the controllers communicate with one another. Automation systems from different manufacturers can communicate with one another by means of PROFIBUS FMS.

- **PROFIBUS FDL (Fieldbus Data Link)** has been optimized for the transmission of medium-sized data volumes to support error-free data transmission on the PROFIBUS subnet.

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

2.1.5 MPI (Multi-Point Interface)

Overview

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

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

The electric transmission medium for MPI uses the RS 485 standard, which is also used by PROFIBUS.
2.1.6 Point-to-point communication (PtP)

Overview

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

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

Point-to-point communication provides the following capabilities:

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

Additional references

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

2.2 Communications services (or network functions)

2.2.1 Introduction

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

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

2.2.2 PG/OP communication services

Overview

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

2.2.3 S7 communication services

Overview

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

Overview

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

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

Note

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

2.2.5 "Global data" communication service

Overview

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

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

Note

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

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

Global data is not available for SIMOTION.
2.2.6 PROFINET communication services

Overview

PROFINET provides the following communication services:

- You can connect I/O devices and drives via a Ethernet physics to the SIMOTION or SIMATIC controller with the communication service PROFINET IO. The user program executed in the controller can process the input and output data of the I/O devices with PROFINET IO. You configure the addressing for PROFINET IO in STEP 7 or SIMOTION SCOUT.

- With PROFINET CBA, you can define your automation system as autonomous subunits or components. These components can be PROFINET IO, PROFIBUS DP or third-party devices or subnets.

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

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

Note

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

2.2.7 Industrial Ethernet communication services

Overview

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

Industrial Ethernet provides the following communication services:

- The ISO transfer provides services for transmitting data via connections that support error-free data transmission. The ISO transfer is only possible with STEP7.

- TCP/IP allows you to exchange contiguous data blocks between the controllers and computers in PROFINET or Industrial Ethernet networks. With TCP/IP, the controller transmits contiguous data blocks.

- ISO-on-TCP (RFC 1006) supports error-free data transmission. For SIMOTION only when going though SCOUT ONLINE. If the communication is performed from the user program, an RFC must be programmed.
Overview of the communication functions and services

2.2 Communications services (or network functions)

- 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.

2.2.8 PROFIBUS communication services

Overview

PROFIBUS provides the following communication services:

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

- PROFIBUS PA (Process Automation) facilitates the direct communication with process automation (PA) instruments. This includes both cyclic access to I/O, typically with a PLC master, as well as acyclic access to the potentially large set of device operating parameters, typically with an engineering tool such as Process Device Manager (PDM). PROFIBUS PA complies with the IEC 61158 standard.

- PROFIBUS FMS (Fieldbus Message Specifications) enables the transmission of structured data (FMS variables). PROFIBUS FMS complies with the IEC 61784 standard.

- PROFIBUS FDL (Fieldbus Data Link) has been optimized for the transmission of medium-sized data volumes to support error-free data transmission on the PROFIBUS subnet. PROFIBUS FDL supports the SDA function (Send Data with Acknowledge).

Note

SIMOTION devices only support the PROFIBUS DP communication service.

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

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

Additional references

You can find a comparison of the SIMATIC S7 and SIMOTION system functions in the 2_FAQ directory on the Utilities & Applications CD.
2.3 Additional services for the exchange of information

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

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

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
- S5-compatible 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.
Overview of the communication functions and services

2.3 Additional services for the exchange of information
3.1 PROFIBUS communication

3.1.1 PROFIBUS communication (overview)

Description

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

![Data protocol on PROFIBUS DP](image)

Figure 3-1 Data protocol on PROFIBUS

3.2 Communication with SIMATIC S7

3.2.1 Possible communication connections between SIMOTION and SIMATIC

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

There are various possibilities:

- A SIMOTION device is connected as DP slave to a DP master system of a SIMATIC S7.
- A SIMATIC S7 device is connected as DP slave to a DP master system of a SIMOTION.
- A master-master communication is used between SIMOTION and SIMATIC S7.
There are two additional variants for the connection as DP slave:

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

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

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

The available I-slaves can be found in the HW catalog of HW Config in the "Preconfigured stations" folder.

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

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

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

---

**Note**

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

---

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

#### 3.2.2.1 Introduction

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

There are two possibilities:

- The SIMOTION device is connected as standard slave to a DP master system by means of a GSD file.
- The SIMOTION device is integrated as what is known as an intelligent DP slave (I-slave) into the DP master system.
3.2.2.2 Connecting SIMOTION to a SIMATIC S7 as DP slave with the aid of a GSD file

Procedure

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

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

Table 3-1 GSD file

<table>
<thead>
<tr>
<th>Device</th>
<th>Name of the GSD file</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIMOTION C</td>
<td>Si0480aa.gsd</td>
</tr>
<tr>
<td>SIMOTION D</td>
<td>Si0280ab.gsd</td>
</tr>
<tr>
<td></td>
<td>(This file can be used for all SIMOTION D)</td>
</tr>
<tr>
<td>SIMOTION P</td>
<td>Si0380fa.gsd</td>
</tr>
</tbody>
</table>

After these GSD files have been imported from the Options - Install GSD file menu into the STEP 7 HW Config, the devices appear in the HW catalog under Additional field devices - PLC - SIMATIC - SIMOTION and can be inserted from there into a DP master system of a S7 station.

Note

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

The name of a GSD file depends on its version, e.g. S10180AA and S10280AA.

Note

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

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

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

3.2.2.3 Connecting SIMOTION as an I-slave to a SIMATIC S7

Requirement

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

If these requirements are fulfilled, SIMOTION can also be connected as what is known as an I-slave to the PROFIBUS DP network of the SIMATIC.
**Procedure**

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

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 DP slave, e.g. SIMOTION C2xx  
   Double-click on the desired interface (e.g. DP2/MPI) in the configuration table and select the DP slave option on the Operating mode tab.

2. Configuring the local I/O addresses  
   You can set the local I/O addresses and the diagnostics address in the **Configuration** tab.

3. Switch to the configured SIMATIC station that is to be DP master for the SIMOTION.

4. Creating an 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 symbol for the DP master system of the SIMATIC station.

5. Specifying the intelligent DP slave  
   Double-click the symbol for the intelligent SIMOTION DP slave and select the **Link** tab. In this tab, assign the station that is to represent the intelligent DP slave. This dialog box displays all the stations that are already available in the project and that are potential link partners.

   ![Figure 3-2 DP slave properties](image)

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 3-3 Properties - configuration](image)

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

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

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.

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

#### 3.2.3.1 Introduction

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

There are two possibilities:

- The SIMATIC station is connected as standard slave to the DP master system of a SIMOTION by means of a GSD file.
- The SIMATIC station is integrated as what is known as an I-slave into the DP master system of a SIMOTION.
3.2.3.2 Connecting SIMATIC to a SIMOTION device as DP slave with the aid of a GSD file

Procedure

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

You will find the corresponding GSD files in Product Support under:

After these GSD files have been imported from the Options - Install GSD file menu into the STEP 7 HW Config, the devices appear in the HW catalog under Additional field devices - PLC - SIMATIC and can be inserted from there into a DP master system of a SIMOTION station.

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

3.2.3.3 Connecting a SIMATIC S7 CPU as an I-slave to a SIMOTION device

Prerequisites

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

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

Procedure

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

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 icon for the intelligent SIMOTION DP slave and select the Link tab. In this tab, assign the station that is to represent the intelligent DP slave. This dialog box displays all the stations that are already available in the project and that are potential link partners.

![Properties - link](image)

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:

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

8. Confirm the settings by clicking OK.

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

3.2.4.1 Introduction

Master-master communication

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

<table>
<thead>
<tr>
<th>Log</th>
<th>SIMATIC device</th>
<th>Function</th>
<th>SIMOTION device</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROFIBUS</td>
<td>S7-300 CPU</td>
<td>SFC65 (XSEND)</td>
<td>SIMOTION C</td>
<td>_Xsend</td>
</tr>
<tr>
<td></td>
<td>S7-400 CPU</td>
<td>SFC66 (XRCV)</td>
<td>SIMOTION D</td>
<td>_Xreceive</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SIMOTION P</td>
<td></td>
</tr>
</tbody>
</table>

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

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

3.2.4.2 SIMATIC S7 system functions for a PROFIBUS connection

Introduction

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

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

SIMOTION functions:

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

Parameterization of the SFC65 X_SEND system function

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

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

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

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

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

SD specifies the area from where the send data originates.

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

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

Call example of the system function SFC66 X_RCV

If data from a SIMOTION device is to be received on a SIMATIC S7 station, the SFC66 X_RCV system function must be called in the S7 program.
The "EN_DT" input of the system function specifies:

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

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

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

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

The RD parameter specifies where the received data is stored.

**SIMOTION functions**

```plaintext
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 for calling the SIMOTION _xsend function

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

**Note**

It is only possible to send data (_xsend()) to a SIMOTION DP slave if a check mark has been placed in *Programming, status/force, or other PG functions and non-configured communication connections possible*.

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

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

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

The deviceid parameter is used for the respective SIMOTION hardware. The physical connection point is specified with the parameter. The value 1 is entered for interface X8 for a SIMOTION C2. 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.
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.
4.1 PROFINET IO overview

4.1.1 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 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.

4.1.2 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.

4.1.3 IO controller

The PROFINET IO controller has the same functions as the PROFIBUS DP master. The IO controller of, for example, a D 4x5-2 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.

![Diagram of PROFINET IO system]

Figure 4-1 Examples of IO controllers and IO devices

4.1.4 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 127)

4.1.5 PROFINET IO system

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

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

4.1.7 RT classes

4.1.7.1 RT classes for PROFINET IO

Description

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

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

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

RT

RT communication uses the option of prioritizing telegrams (as described in the Ethernet standard). This mechanism is also used for Voice over IP, for example. For more detailed information, see PROFINET IO with RT (Page 46).

IRT

A time slot procedure is superimposed on the Ethernet for PROFINET IO with IRT. This means there are 2 slots. IRT telegrams are transferred in the first and RT and IP telegrams in the second. Under this arrangement, a transmission bandwidth guaranteed to cope with any load/overload situation is reserved for the IRT data. IRT requires devices to be synchronized so that all devices involved know when the time slot begins.

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.

As well as synchronizing the IRT transmission network, IRT High Performance enables the application (e.g. SIMOTION position controller and interpolator) to be synchronized in the devices (isochronous application). This mirrors the behavior of the isochronous PROFIBUS.
This is an essential requirement for closing control loops across the network and isochronous switching of inputs and outputs in the network.

**Note**

Only IRT High Performance is used for SIMOTION. Whenever "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 47).

### Comparing RT and IRT

<table>
<thead>
<tr>
<th>Property</th>
<th>RT</th>
<th>IRT (High Performance)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real-time class</td>
<td>Real-time class 1</td>
<td>Real-time class 3</td>
</tr>
<tr>
<td>Transfer mode</td>
<td>Prioritization of cyclic RT data using Ethernet-Prio (VLAN tag)</td>
<td>Bandwidth reservation optimized by the engineering system on the basis of topology information</td>
</tr>
<tr>
<td>Determinism</td>
<td>Variance of the transmission duration for cyclic RT data using TCP/IP telegrams</td>
<td>Transmission and receiving times for cyclic IRT data are precisely defined and guaranteed for all kinds of topologies.</td>
</tr>
<tr>
<td>Isochronous application</td>
<td>Not supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Hardware support using special Ethernet controller</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

#### 4.1.7.2 Send clock and update time

**Description**

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

**Send clock**

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

**Update time**

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

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

4.1.7.3 Adjustable send clocks and update times

Description

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

Note

The following versions relate to the use of the "normal" servo cycle clock (1st servo) with servo or IPO task. When using the Servo_Fast (2nd servo) 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 4-2 Adjustable send clocks and update times when using first servo

<table>
<thead>
<tr>
<th>Send clock</th>
<th>Down-scaling (update time = factor * send clock)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RT</td>
</tr>
<tr>
<td>&quot;Even&quot; range</td>
<td>1,2,4,8,16,32,64,128,256,512</td>
</tr>
<tr>
<td>250, 500, 1,000 µs</td>
<td>1,2,4,8,16,32,64,128,256</td>
</tr>
<tr>
<td>2,000 µs</td>
<td>1,2,4,8,16,32,64,128,256</td>
</tr>
<tr>
<td>4,000 µs</td>
<td>1,2,4,8,16,32,64,128,256</td>
</tr>
<tr>
<td>&quot;Odd&quot; range Note 1</td>
<td>Not supported</td>
</tr>
<tr>
<td>375, 625, 750, 875</td>
<td>1</td>
</tr>
<tr>
<td>975, 1,125, 1,250 µs</td>
<td>1</td>
</tr>
<tr>
<td>(increment 125 µs)</td>
<td></td>
</tr>
<tr>
<td>3,875 µs</td>
<td></td>
</tr>
</tbody>
</table>

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

Note 2) Down-scaling (factor) and isochronous application
Some IO devices support the factors 2, 4, 8, 16 in addition to factor 1 with IRT High Performance. Where IO devices (e.g. ET200S IM151-3 PN HS, SINAMICS S) are operated with an isochronous application, it is usually only possible to set 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 to always match the send clock.
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 controllers assigned to the sync domain accept this value. Update times can be set independently for each IO device.

**Send clock mode for update time**

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

---

**Note**

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

---

![Figure 4-2 Update time and factor](image-url)
4.1.7.4 Setting RT classes

RT classes

The IO controller determines which RT class its IO system supports, by setting the real time class at its controller interface. RT devices can always be operated, even if IRT classes are set.

Setting the RT class

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

1. In HW Config, double-click the PROFINET interface in the module.

   The Properties dialog box is called.

   ![Properties Dialog Box]

   2. Select the realtime class for RT class in the Synchronization tab. With SIMOTION, IRT High Performance is automatically set for IRT.

   3. Click OK to confirm.

Note

Only IRT High Performance is used for motion control applications with SIMOTION and SINAMICS.
4.1.7.5 PROFINET IO with RT

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

Not isochronous

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

Data exchange

Communication via PROFINET IO with RT and IRT is based on the Ethernet frame and the MAC address. This means that cross-network communication via a router using RT and IRT is not possible. PROFINET IO telegrams have priority over IT telegrams 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.

4.1.7.6 PROFINET IO with IRT - Overview

Overview

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

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

4.1 PROFINET IO overview

See also IRT High Performance (Page 47) 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 49).

4.1.7.7 PROFINET IO with IRT (High Performance)

The performance capability of motion control applications is significantly increased with PROFINET IO IRT (High Performance). If field buses such as PROFIBUS are used, devices are connected in parallel to the bus. This has a number of consequences. Firstly, only 1 device can transmit at any one time. Secondly, the parallel connection of all devices means that the physical limit is reached at approximately 12 Mbps.

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

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

IRT (High Performance) is particularly suitable for:

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

**Send clock**

The send clock can be set between 250 µs and 4 ms for PROFINET IRT High Performance. In mixed operation involving RT and IRT High Performance, only the values 250 µs, 500 µs, 1.0 ms, 2.0 ms or 4.0 ms can be set.
The actual send clock used depends on various factors:

- The process; communication should be no faster than required. This reduces the bus and CPU loads.
- The bus load (number of devices and the amount of IO data per device)
- Computing power available in the controller
- Supported send clocks in the participating PROFINET devices of a sync domain

A typical send clock is, for example, 1 ms. However, it can be set in a 125 µs grid within the limits of 250 µs to 4 ms. See also Adjustable send clocks and update times (Page 43).

The supported send clocks can be found in the corresponding manuals of the respective SIMOTION devices. A minimum cycle time of 250 µs is only supported by selected components (SIMOTION P320-3, P350-3, D4x5-2 DP/PN and ET 200S HS modules).

**Isochronous application**

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

In contrast to standard Ethernet and PROFINET IO with RT, the transmission of message frames for PROFINET IO with IRT High Performance is scheduled.

**Time-scheduled data transmission**

Scheduling is the specification of the communication paths and the exact transmission times for the data to be transferred. The bandwidth can be optimally utilized through communication scheduling and therefore the best possible performance achieved. This requires the network topology to be configured, with the engineering system automatically calculating the communication schedule from the configured topology (see also Topology). 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 2 or more IO systems which can all be synchronized with one another. Within the IO system, IRT High Performance is then used.
4.1.8 Sync domain

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

See also

Creating a sync domain (Page 114)

4.1.9 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.
4.1 PROFINET IO overview

Compatibility

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

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 57)

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

Alternatively, the device can be initialized automatically by the controller on the basis of topology information. This is only possible if topology information (who is wired to whom) is stored in the engineering system. During ramp-up, the controller identifies the connected devices using the device names, before assigning the IP address defined in the engineering system to the 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, communication name (devices identified by the controller during ramp-up)

See also

Assigning device names and IP addresses to IO devices (Page 135)

4.1.11 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
PROFINET IO

4.1 PROFINET IO overview

- 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.

<table>
<thead>
<tr>
<th>Topology</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Star</td>
<td>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.</td>
</tr>
<tr>
<td>Tree</td>
<td>If you interconnect several star-shaped structures, you obtain a tree network topology.</td>
</tr>
<tr>
<td>Line</td>
<td>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.</td>
</tr>
<tr>
<td>Ring</td>
<td>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.</td>
</tr>
</tbody>
</table>

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 HMI, 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.
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.
4.1.12 Isochronous applications with PROFINET

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

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

Configuration model for isochronous mode, V4.2 and higher

The configuration model for isochronous mode has changed as of V4.2. The Application tab for the properties of the IO device has been replaced with settings on the controller and on the IO cycle tab of the IO device. You can find a more detailed outline of the procedure for making settings in isochronous applications in the section titled Inserting and configuring the SINAMICS S120 (Page 129)
4.1 PROFINET IO overview

Procedure

When configuring isochronous applications, proceed as follows:

1. On controllers and devices, choose the setting "High Performance"; with SIMOTION (controller) V4.2 or higher, this setting is made automatically.

![Image](image.png)

Figure 4-6 Setting IO device synchronization to Sync slave and setting IRT High Performance.

2. On the devices, set the **Update time** mode to **Fixed factor** and select the cycle clock under **Assign IO device isochronously**. This is normally the servo cycle clock.
4.1.13 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 313).
4.2 Properties of PROFINET IO with SIMOTION

4.2.1 Introduction

Requirement

To work with SIMOTION using PROFINET IO, you must have PN slots available. These can be found directly on the controller or added by inserting Option Boards.

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 P320-3
- SIMOTION D410 PN/SIMOTION D410-2 DP/PN
- SIMOTION C240 PN

![System topology with PROFINET](image)
PROFINET devices 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 IO device
  - Data exchange between controllers via IRT High Performance (e.g. distributed synchronous operation)
- RT - realtime Ethernet
  - Operation of RT - peripherals (e.g. ET 200S, ET 200pro)
  - ASi link via IE/AS interface link PN IO for the PROFINET IO gateway to AS interface
  - SINAMICS as PROFINET IO with an RT device
- TCP/IP, UDP, HTTP, … standard Ethernet services

**Note**

For mixed operation of IRT High Performance and RT, it must be ensured that the IRT High Performance-compatible devices are directly connected to one another. In other words, there must not be any devices that are not IRT-compatible between the IRT devices.

**Note**

With SIMOTION SCOUT, it is possible to access a maximum of 9 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**

A PROFINET device connected to a SIMOTION CPU may have a maximum submodule size of 254 bytes.

This limit must be observed in particular if a SIMATIC CPU is configured as an iDevice for a SIMOTION CPU, since submodules > 254 bytes can be configured for the SIMATIC iDevice.
PROFINET V2.2

SIMOTION SCOUT supports PROFINET V2.2. Older versions are no longer supported as standard or must be changed over in the hardware.

When inserting SIMOTION devices 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.

Note

All IRT nodes must comply with the PROFINET V2.2 standard. Mixed configurations involving older versions are not permissible.

The hardware supplied as standard is PROFINET V2.2. If you are upgrading existing systems or downgrading new hardware, please get in touch with Siemens Support.

Overview of configuration for PROFINET V2.2 with STEP7 5.4 SP4 and SCOUT V4.1.2

<table>
<thead>
<tr>
<th>Configuration step</th>
<th>IRT according to PN V2.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration of the RT class in HW Config</td>
<td>IRT High Performance</td>
</tr>
<tr>
<td>Configuration of SINAMICS S120 via GSD</td>
<td>GSD V2.2 and higher</td>
</tr>
<tr>
<td>Configuration of SINAMICS S120 via DeviceOM</td>
<td>Supported</td>
</tr>
<tr>
<td>SIMOTION</td>
<td>4.1.2 and higher</td>
</tr>
<tr>
<td>SINAMICS</td>
<td>2.5.1.10 and higher</td>
</tr>
<tr>
<td>SCALANCE X200 IRT switch</td>
<td>V4.1 and higher</td>
</tr>
</tbody>
</table>

4.2.2 Cycle clock scaling

4.2.2.1 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 the Expert > Set system cycle clocks context menu.

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 129).
V4.2 and higher, two servo cycle clocks (servo, Servo_fast)

With V4.2 and higher, Servo_fast is introduced as a second servo task 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 cycle clock of the servo can then be set again as a multiple of 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 72).

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:
  - Isochronous mode is configured for an IO device.
  - Isochronous mode is configured for an iDevice.
  - Controller-controller communication is configured.

**Note**

Only the SIMOTION device acting as the Sync-Master synchronizes itself automatically with the higher-level bus. If the SIMOTION device has no PROFIBUS DP interface configured as an isochronous DP master, the Sync-Slaves synchronize themselves automatically too.

If a PROFIBUS DP interface is configured as an isochronous DP master for lower-level drives, the device will have to be synchronized by means of the application. This application is programmed in the StartupTask via the _enableDPIInterfaceSynchronizationMode command.

**Combinations of cycle clocks and cycle clock sources for PROFIBUS and PROFINET IO**

- Servo can be scaled in integral multiples (1, 2, ...n) of the send clock.
- If a Servo_fast is configured, it is permanently set at a ratio of 1:1 to the send clock. As before, the CACF can be adjusted as the ratio of the send clock/Servo_fast to servo.
- Cycle clocks for SINAMICS Integrated and isochronous DP master interfaces must run simultaneously with the servo cycle clock.

**4.2.2.2 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).

● Even if PROFINET is being used and the PROFINET cycle clock is down-scaled to the servo 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 cycle cycle, since the servo priority class must always be finished with the first bus cycle 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.

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

**Overview of the possible bus cycle clocks**

<table>
<thead>
<tr>
<th></th>
<th>PROFIBUS&lt;sup&gt;2)&lt;/sup&gt;</th>
<th>PROFINET IRT High Performance</th>
<th>PROFINET IRT High Performance</th>
<th>Servo Servo_fast</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Minimum</td>
<td>Minimum</td>
<td>Maximum</td>
<td>Minimum</td>
</tr>
<tr>
<td>SINAMICS S120 CU320</td>
<td>1 ms</td>
<td>0.5 ms</td>
<td>4.0 ms</td>
<td>0.5 ms</td>
</tr>
<tr>
<td>SINAMICS S120 CU310</td>
<td>1 ms</td>
<td>0.5 ms</td>
<td>4.0 ms</td>
<td>0.5 ms</td>
</tr>
<tr>
<td>SINAMICS S120 CU310-2</td>
<td>1 ms</td>
<td>0.25 ms</td>
<td>4.0 ms</td>
<td>0.25 ms</td>
</tr>
<tr>
<td>SINAMICS S120 CU320-2</td>
<td>1 ms</td>
<td>0.25 ms</td>
<td>4.0 ms</td>
<td>0.25 ms</td>
</tr>
</tbody>
</table>
## PROFINET IO

### 4.2 Properties of PROFINET IO with SIMOTION

<table>
<thead>
<tr>
<th></th>
<th>PROFIBUS</th>
<th>PROFINET IRT High Performance</th>
<th>PROFINET IRT High Performance</th>
<th>Servo Servo_fast</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Minimum</td>
<td>Minimum</td>
<td>Maximum</td>
<td>Minimum</td>
</tr>
<tr>
<td>SINAMICS S110 CU305</td>
<td>1 ms</td>
<td>1 ms</td>
<td>4 ms</td>
<td>1 ms</td>
</tr>
<tr>
<td>C230-2</td>
<td>1.5 ms</td>
<td>-</td>
<td>-</td>
<td>1.5 ms</td>
</tr>
<tr>
<td>C240 PN</td>
<td>1 ms</td>
<td>0.5 ms</td>
<td>4.0 ms</td>
<td>0.5 ms</td>
</tr>
<tr>
<td>C240</td>
<td>1 ms</td>
<td>-</td>
<td>-</td>
<td>0.5 ms</td>
</tr>
<tr>
<td>D410 PN</td>
<td>-</td>
<td>0.5 ms</td>
<td>4.0 ms</td>
<td>2.0 ms</td>
</tr>
<tr>
<td>D410 DP</td>
<td>2 ms</td>
<td>-</td>
<td>-</td>
<td>2.0 ms</td>
</tr>
<tr>
<td>D410-2 DP</td>
<td>1 ms</td>
<td>-</td>
<td>-</td>
<td>0.5 ms</td>
</tr>
<tr>
<td>D410-2 DP/PN</td>
<td>1 ms</td>
<td>0.5 ms</td>
<td>4.0 ms</td>
<td>0.5 ms</td>
</tr>
<tr>
<td>D425</td>
<td>2 ms</td>
<td>0.5 ms</td>
<td>4.0 ms</td>
<td>2.0 ms</td>
</tr>
<tr>
<td>D425-2 DP</td>
<td>1 ms</td>
<td>0.25 ms</td>
<td>-</td>
<td>0.5 ms</td>
</tr>
<tr>
<td>D425-2 DP/PN</td>
<td>1 ms</td>
<td>0.5 ms</td>
<td>4.0 ms</td>
<td>0.5 ms</td>
</tr>
<tr>
<td>D435</td>
<td>1 ms</td>
<td>0.5 ms</td>
<td>4.0 ms</td>
<td>1.0 ms</td>
</tr>
<tr>
<td>D435-2 DP</td>
<td>1 ms</td>
<td>-</td>
<td>-</td>
<td>0.5 ms</td>
</tr>
<tr>
<td>D435-2 DP/PN</td>
<td>1 ms</td>
<td>0.5 ms</td>
<td>4.0 ms</td>
<td>0.25 ms&lt;sup&gt;1)&lt;/sup&gt;</td>
</tr>
<tr>
<td>D445/D445-1</td>
<td>1 ms</td>
<td>0.5 ms</td>
<td>4.0 ms</td>
<td>0.5 ms</td>
</tr>
<tr>
<td>D445-2 DP/PN</td>
<td>1 ms</td>
<td>0.25 ms</td>
<td>4.0 ms</td>
<td>0.25 ms&lt;sup&gt;1)&lt;/sup&gt;</td>
</tr>
<tr>
<td>D455-2 DP/PN</td>
<td>1 ms</td>
<td>0.25 ms</td>
<td>4.0 ms</td>
<td>0.25 ms&lt;sup&gt;1)&lt;/sup&gt;</td>
</tr>
<tr>
<td>P350-3</td>
<td>1 ms</td>
<td>0.25 ms</td>
<td>4.0 ms</td>
<td>0.25 ms&lt;sup&gt;1)&lt;/sup&gt;</td>
</tr>
<tr>
<td>P320-3</td>
<td>-</td>
<td>0.25 ms</td>
<td>4.0 ms</td>
<td>0.25 ms</td>
</tr>
<tr>
<td>ET200S HS</td>
<td>-</td>
<td>0.25 ms</td>
<td>4.0 ms</td>
<td>0.25 ms</td>
</tr>
</tbody>
</table>

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 for fast I/O processing or high-performance hydraulic applications. The sensors and actuators are connected via high-speed PROFINET IO I/O modules

2) Explanation:
- The specifications apply to "external" PROFIBUS interfaces and not to PROFIBUS Integrated for SIMOTION D.
4.2 Properties of PROFINET IO with SIMOTION

4.2.3 Task system and time response

4.2.3.1 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.
### 4.2.3.2 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.

![Data transmitted in IRT time slot](image)

**Figure 4-8 Logic evaluation for an axis in the BackgroundTask or MotionTask**

#### Boundary conditions

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

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

Assigning the variables to a process image has an impact on the response time as well. The process images are made available to other tasks or to the communication interface at the end of the respective task, rather than after the variables are updated.
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 4-9  Logic evaluation for an axis in the IPOSynchronousTask
4.2 Properties of PROFINET IO with SIMOTION

### 4.2.3.3 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.

![Servo-synchronous evaluation of input](image1)

**Figure 4-10** 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.
If the option of two servo cycle clocks is configured with Servo_fast, the axis can also be assigned to the Servo_fast processing cycle clock if the axis 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.

![Diagram of position control time response with ServoSynchronousTask](image-url)
4.2.3.4 Fast I/O processing in the ServoSynchronousTask

Fast I/Os (e.g. ET200S HS) are evaluated in the ServoSynchronousTask, resulting in a system response time of one cycle. Due to the terminal-terminal response, there is a delay of Ti + servo cycle clock + To.

![Diagram of Fast I/O processing in the ServoSynchronousTask]

If the option of two servo cycle clocks is configured with Servo_fast, fast I/O processing in the ServoFastSynchronousTask can also take place in the Servo_fast cycle clock. This requires the I/Os to be configured on the faster bus.

4.2.4 Devices with two PROFINET IO interfaces

4.2.4.1 Two PROFINET IO interfaces

SIMOTION devices with two PROFINET interfaces

With the SIMOTION D4x5-2 DP/PN control units, optionally a second PROFINET 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.
Applications

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 devices
  - to increase the available I/O address space
- Devices are to be operated with different send cycle clocks and assigned different isochronous tasks (Servo and Servo_fast).
- 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 controller are to be operated isochronously at the same time.

If one PROFINET interface is operated simultaneously as an iDevice and controller, either only the iDevice or the controller can be operated isosynchronously (i.e. with IRT). With two PROFINET interfaces, it is possible to operate a PROFINET IO controller and an iDevice isosynchronously on a D4x5-2 DP/PN at the same time.

Rules for two PROFINET interfaces

The following rules must be taken into account when using two PROFINET interfaces if both interfaces are to be operated isosynchronously (an isochronous device or an isochronous iDevice or slave-to-slave communication is configured at the interface).

- Both interfaces can be configured as a PROFINET IO iDevice and/or as a controller.
- A higher-level controller must always be connected isosynchronously via the CBE30-2.
  This applies both to controller-controller communication and to iDevice communication with a higher-level controller. Lower-level isochronous drives/I/Os should be connected via the onboard PROFINET interface.
- The onboard PROFINET interface must be configured as SYNC master.
- The CBE30-2 can be configured as SYNC master or SYNC slave.
- An F-CPU can only be connected to one of the two PROFINET interfaces, because fail-safe I/O transfer areas can only be configured either on the I device on the onboard PROFINET interface or on the CBE30-2.

The PROFIsafe message frames are routed to the following drives:

- Drives on the SINAMICS Integrated and CX32-2
- 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.
If Servo_fast is not used, isochronous devices on both PROFINET interfaces can be assigned to the Servo.

No redundant sync master is permitted in the SYNC domain of the onboard PROFINET interface if both PROFINET interfaces are operated isosynchronously and the CBE30-2 is configured as a SYNC slave or IRT iDevice.

### Rules for Servo_fast

The following rules apply for the onboard PROFINET interface (X150) assigned to the Servo_fast:

- The interfaces are permanently assigned:
  - Servo/IPO/IPO_2 to the CBE30-2 (or DP and DP Integrated)
  - Servo_fast and IPO_fast to the onboard PROFINET interface
- If the Servo_fast is used, only one CACF (Controller Application Cycle Factor) = 1 is permitted, that is, the following must be set:
  - Servo cycle clock = send cycle clock PROFINET interface X1400 (CBE30-2)
  - Servo_fast cycle clock = send cycle clock PROFINET interface X150 (onboard interface)
- The Servo_fast must have a reduction ratio to the Servo of 2, 4, 8, or 16.

### See also

- Example configuration for controlled sync master (Page 82)
- Cycle clock scaling with PROFINET IO on SIMOTION devices (Page 62)
- Use of Safety with MRRT (Page 96)

### 4.2.4.2 Applications for devices with two PROFINET IO interfaces

#### Applications for two PN interfaces

#### Overview

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 devices on PROFINET (Page 75).
2. Fast and slow clock cycle for data to isochronous devices, see Application 2 - fast IO processing and slower electrical drives on one module (Page 76).
3. Distributed synchronous operation independent of isochronous devices, see Application 3 - Hierarchical IRT networks with IRT distribution bus for synchronous operation and lower-level local IRT network (Page 77).
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 - Independent IP address and NameOfStation for the iDevice and controller (Page 78).

6. Two independent iDevices, see Application 5 - iDevice on both interfaces (Page 79).

Application 1 - increase in the maximum possible number of 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 4-14 Application 1 for two PROFINET IO interfaces

<table>
<thead>
<tr>
<th>PN1</th>
<th>Integrated PROFINET interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>PN2</td>
<td>CBE30-2</td>
</tr>
</tbody>
</table>

See also

Applications for two PN interfaces (Page 74)
Application 2 - fast IO 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 device.
- Slower SINAMICS drives are controlled with a cycle of e.g. 1 ms servo cycle clock via PN2.
- Fast I/Os (e.g. ET200 HS or TMC), for example, with 250 µs clock cycle via Servo_fast via PN1

Figure 4-15 Application 2 for two PROFINET IO interfaces

| PN1 | Integrated PROFINET interface |
| PN2 | CBE30-2                       |

See also

Applications for two PN interfaces (Page 74)
Application 3 - Hierarchical IRT networks with IRT distribution bus for synchronous operation and lower-level local IRT network

Description:

The following functions are illustrated in the topology:

- Distributed synchronous operation via controller-controller communication between the SIMOTION devices 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

![Diagram showing application 3 with two PROFINET IO interfaces](image)

Figure 4-16 Application 3 for two PROFINET IO interfaces

| PN1 | Integrated PROFINET interface |
| PN2 | CBE30-2                        |

See also

Applications for two PN interfaces (Page 74)
Application 4 - Independent IP address and NameOfStation for the iDevice and controller

**Description**

The following functions are illustrated in the topology:

- 2 PN interfaces in the SIMOTION D devices disconnect the subnet of the basic machine of the machine modules.
- IP address and NameofStation on PN2 as iDevice can be assigned as controllers independently of PN1.
- IP addresses and NameofStation in different machine modules can be identical.
- The project of the basic machine includes proxies of the modules (GSD), D4x5-2 as a proxy with iDevice.
- IRT I device on the CBE30-2 and an IRT controller on the integr. PN interface also isochronous.
- Required distributed synchronous operation of basic machine with the axes of the modules:
  - Isochronous coupling of the subnets, if PN axes
  - IRT communication on iDevice and controller interface

![Application 4 for two PROFINET IO interfaces](image-url)
See also

Applications for two PN interfaces (Page 74)

**Application 5 - iDevice on both interfaces**

**Description**

The following functions are illustrated in the topology:

- Distributed synchronous operation of the master controller to lower-level SIMOTION via IRT iDevice on PN2
- RT communication between another controller and the lower-level SIMOTION as iDevice on PN1 for PROFIsafe, diagnostics, process value recording (IRT communication not possible)

![Diagram of Application 5 for two PROFINET IO interfaces]

**See also**

Applications for two PN interfaces (Page 74)
4.2 Properties of PROFINET IO with SIMOTION

4.2.4.3 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 iDevice with controller local I/O)

The following figure shows if a PN CBE30-2 is used as iDevice and the integrated PROFINET interface as controller for the local I/O:

- PN CBE30-2 is used as IRT iDevice for a higher-level controller.
- PN Integrated (X150) is used as an IRT I device for a lower-level controller.

Requirements for synchronization:

- PN CBE30-2: Sync slave (synchronizes with the higher-level controller).
- PN Integrated (X150): iDevice operates as controlled sync master and synchronizes the lower-level controller.
- 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.
- An example configuration can be found at Example configuration for controlled sync master (Page 82).
Figure 4-19  Controlled sync master
Synchronization example of controller slave-to-slave communication - controller local I/O

The following figure shows the synchronization if a controller is used for the local I/O and a controller is used for slave-to-slave communication:

- PN Integrated is used as the controller for the local I/O.
- PN CBE30-2 is used for slave-to-slave communication.

Requirements for synchronization:

- PN Integrated: Controlled sync master.
- PN CBE30-2: Sync master or sync slave with regard to slave-to-slave communication.
- 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.

![Diagram of controller slave-to-slave communication with controlled sync master](image-url)

**Figure 4-20** Controller slave-to-slave communication with controlled sync master

### 4.2.4.4 Example configuration for controlled sync master

**Example for an application with two PROFINET IO subnets synchronized via controlled sync masters**

The topology consists of a SIMOTION D4x5-2 module (sync master), a SCALANCE Switch (redundancy manager, redundant sync master) and two SIMOTION modules, each connected to an IO device (SINAMICS S120).
Distributed synchronous operation via two PROFINET interfaces

The example is designed to show how SIMOTION modules with two PROFINET interfaces can be used. PROFINET 1 (sync domain 1) is designed as an MRPD ring for distributed synchronous operation. The sync master specifies the bus cycle clock as well as servo and IPO cycle clock. The synchronous axes establish the PROFINET subnet 2 (Sync Domain 2) via the second PROFINET interface. As controlled sync masters, they forward the bus cycle clock to the lower-level IO devices. See also Controlled sync master (Page 80) and Two PROFINET IO interfaces (Page 72).

<table>
<thead>
<tr>
<th>PROFINET 1 (sync domain 1)</th>
<th>Functions</th>
<th>Cycle clocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus cycle clock = 1 ms</td>
<td>Virtual axis</td>
<td>DP/PN = 1 ms</td>
</tr>
<tr>
<td></td>
<td>Distributed synchronous</td>
<td>Servo = 1 ms</td>
</tr>
<tr>
<td></td>
<td>operation</td>
<td>IPO1 = 4 ms</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PROFINET 2 (sync domain 2) or PROFINET 3 (sync domain 3)</th>
<th>Functions</th>
<th>Cycle clocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus cycle clock = 2 ms</td>
<td>Following axis</td>
<td>Servo = 1 ms</td>
</tr>
<tr>
<td></td>
<td>SINAMICS IPO</td>
<td>IPO2 = 4 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Servo = 1 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IPO1 = 1 ms</td>
</tr>
</tbody>
</table>
PROFINET IO

4.2 Properties of PROFINET IO with SIMOTION

- The PROFINET subnet 1 (sync domain 1) is established via the interfaces of the CBE30-2 (X1400).
- The PROFINET subnet 2 (sync domain 2) is established in each case via the integrated interface (X150) of the SIMOTION module.
- The PROFINET subnet 3 (sync domain 3) is established in each case via the integrated interface (X150) of the SIMOTION module.
- The two SIMOTION modules automatically become controlled sync masters that handle synchronization between PROFINET 1 and PROFINET 2.

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

- either all sync domains have different names
- or at the ports forming the end of the sync domain, the setting End of sync domain is activated on the Options tab. To do so, call the Properties PNIO - <XXX> dialog and click on the Options tab in the foreground.
4.2.5 Connection between sync domain and IO systems

The devices of several IO systems can be synchronized by a single sync master, provided they are connected to the same Ethernet subnet and belong to a sync domain. Conversely, an IO system may only belong to a single sync domain.

Devices with two PROFINET IO interfaces

The two PROFINET IO interfaces must belong to different (disjunctive) sync domains. You synchronize separate sync domains by using two PROFINET IO interfaces. Synchronization then takes place via a controlled sync master, see also Controlled sync master (Page 80).

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 114).

4.2.6 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 has one sync. master, the other component would not be capable of functioning independently. The "redundant sync master" function was developed for this very reason. Under this arrangement, a sync master is defined for every component. One of these is defined as the "sync master", and the other as the "sync master (redundant)". Provided the system components are combined during operation, the "sync master (redundant)" will synchronize itself with the sync master.

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 88).

- 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 94) and Information on PROFINET with MRPD (Page 92). For this, an additional SCALANCE switch must be configured in the ring as a redundant sync master.
4.2 Properties of PROFINET IO with SIMOTION

Limitations of use

- There can be up to two devices in the line between the two sync masters. If the transmission link between the sync master and sync master (redundant) fails, leaving 2 subnets with one sync master each, both subnets remain synchronized with the remaining sync master in each case. This results in two independent synchronized subnets that drift apart due, among other things, to the temperature drift of the quartzes. Once the transmission link has been reestablished, there can be no smooth synchronization of the sync master (redundant) with the sync master, i.e. the drives assigned to the sync master (redundant) would lose synchronization and fail for a short time. For synchronization, one section of the machine must be switched off and on again.

- If the plant section with the sync master fails as the result of a fault or shutdown, the other plant section with the sync master (redundant) will continue operating independently. The sync slaves assigned to the sync master will lose their synchronization and cease to operate.

Configuring the second sync master

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

![Figure 4-22 Configuring the second sync master](image)

Comunication

86

System Manual, 02/2012
4.2.7 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.
- Up to 64 IRT High Performance devices
- Up to 64 controller-controller data exchange broadcast relationships may be set up between IO controllers.
- If a SIMOTION has been configured as an I device, it counts as a device; in other words, only another 63 devices can be connected.
- A SIMOTION device can receive data from up to 64 other SIMOTION devices in the context of data exchange broadcast between controllers and can send data to any number of SIMOTION devices.
- The amount of data involved must be taken into consideration where the data exchange broadcast is between controllers. A data exchange broadcast only counts as a connection if the amount of data involved does not exceed a certain level. Where a second telegram is required, the 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 communication.

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:
IRT High Performance

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

Note

In a data exchange broadcast relationship, it is not the number of configured slots (lines in the lug receiver - see Configuring the receiver (Page 152)) that is intended for IRT High Performance data exchange broadcast configuration, but rather the number of Ethernet frames received for the data exchange broadcast. An Ethernet frame can contain up to 768 bytes of exchange broadcast useful data.

One slot has up to 254 bytes and 3,072 bytes of useful data can be exchanged during data exchange broadcast (divided into 4 frames of 768 bytes each). The value will depend on the sizes of the slot chosen. This means a transmitter can receive more than one slot, although these must be transmitted within a single message frame.

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

Where data exchange broadcast is taking place between a controller and a device, the frame contains 1,440 bytes of useful data.

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

Address space

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

4.2.8 Media redundancy (MRP and MRPD)

4.2.8.1 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.

MRP (Media Redundancy Protocol, 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:

![Ring topology for media redundancy without fault](image)

Figure 4-23 Ring topology for media redundancy without fault

After the occurrence of a fault, the Redundancy Manager reconfigures the communication of the ring.
**4.2 Properties of PROFINET IO with SIMOTION**

**Ring ports**

A SIMOTION/SINAMICS device may only be inserted in 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 devices (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 IRT frames, as of SIMOTION V4.3)

MRPD (Media Redundancy for Planned Duplication) is a method for smooth media redundancy in PROFINET IO with 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 smooth operation by the provider sending the cyclic data in both directions of the ring which the consumers then receive twice. 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:

![Ring communication in MRPD](image)

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.
4.2 Properties of PROFINET IO with SIMOTION

Devices that support MRPD

The following devices can be found in the IRT sync domain for MRPD:

- SIMOTION devices V4.3 and higher (D4xx-2, P320, C240 PN)
- SINAMICS V4.5 and higher (CU320-2 and CU310-2)
- SCALANCE X200 IRT series V5.0 or higher

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 121)
- Redundant sync master (Page 85)

4.2.8.2 Information on PROFINET with MRPD

MRPD in SIMOTION

Similarly to MRP, MPRD guarantees media redundancy in the event of a fault in the ring (cable or device defect).

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.

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.

Configuration instructions

You must comply with the following rules to operate media redundancy with MRPD:

- The rules for configuring MRP and 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, this IRT property has the devices described in Media redundancy for SIMOTION (Page 88).

See also

Media redundancy for SIMOTION (Page 88)
4.2.8.3 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 controllers can fail or be deactivated with their 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.

![Ring configuration with SCALANCE switches](image)

Figure 4-26 Ring configuration with SCALANCE switches
MRPD configuration with a ring from SIMOTION controllers and a SCALANCE switch

The ring contains a SCALANCE switch as redundancy manager and SIMOTION controllers as redundancy clients. SIMOTION controllers are attached to stubs with their I/O devices.

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

Figure 4-27  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 I/O devices. An I/O device can fail or be deactivated. The rest of the system continues to run smoothly.

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.

See also

Redundant sync master (Page 85)

4.2.8.4 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. In order to ensure this, the SIMOTION on the second PROFINET interface which is not part of the MRRT ring must be used for the communication to the F-CPU of the F proxy. The Safety RT data of the F-CPU are then transferred internally from the SIMOTION CPU into the IRT data of the MRRT ring.

The use of the shared device (F-CPU communicates directly with SINAMICS) is not permitted for MRRT as the Safety data are then transferred in RT mode.
Safety using F-proxy

When using the F-proxy, the Safety data from the F Proxy are transferred to the drives in the MRRT-Ring via IRT and you therefore benefit from the smooth media redundancy with MRRT.

To enable smooth transfer of the Safety data, the communication link from the SIMOTION F-proxy to the F-CPU must not be via the MRRT ring, for example as shown in the figure, connect the F-CPU directly to a free port of the interface of the SIMOTION I-Device F-proxy or use the second PN-IO interface for the I-Device F-proxy. For an exact description of how to configure an I-Device F-proxy see Principles of I device failsafe proxy (Page 252).

How you can configure Safety via PROFINET with a 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).

For a description of the SIMOTION PROFINET interfaces, see Two PROFINET IO interfaces (Page 72).

See also

PROFIsafe via PROFINET with an F-CPU (Page 257)
4.3 Configuring PROFINET IO with SIMOTION

4.3.1 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 88).

Second PROFINET I/O Interface

With SIMOTION D4x5-2 DP/PN, it is possible to have two PROFINET I/O interfaces (integrated interface and CBE30-2) on one device, see Two PROFINET IO interfaces (Page 72). 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 121).

Standard Ethernet interfaces

The standard Ethernet interfaces now include basic PROFINET services, Properties of the SIMOTION Ethernet interfaces (Page 189).

More message frames for PROFIsafe

Additional message frames are provided for PROFIsafe, see also Message frames and signals in drive-based safety (Page 246).

See also

Media redundancy for SIMOTION (Page 88)
Two PROFINET IO interfaces (Page 72)
Servo_fast, scaling down of cycle clocks to the servo at the PROFINET interface (Page 121)
Properties of the SIMOTION Ethernet interfaces (Page 189)
Message frames and signals in drive-based safety (Page 246)
4.3.2 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.

I device
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 (SINAMICS_Integrated with Servo and IPO) and a fast part (IO devices on the Profinet interface with Servo_fast and IPO_fast).
4.3 Configuring PROFINET IO with SIMOTION

4.3.3 Procedure for configuring PROFINET IO with IRT High Performance

Procedure

The following steps need to be performed when configuring PROFINET IO with IRT High Performance V2.2:

1. Insert the SIMOTION module.
   - Select the Device followed by the Device version. Only PROFINET V2.2 variants are possible. 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. V4.3.
   - 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 V4.3, 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 from SIMOTION Drive Based > SIMOTION D4xx > 6AUxx > V4.3 - PN-V2.2xx.

2. Insert IO devices:
   - Insert IO devices from the hardware catalog in HW Config into the I/O system. The IO devices can be found in the hardware catalog under PROFINET IO.

3. Configure sync domain and specify send clock:
   - 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 IO devices are interconnected with one another. The topology only needs to be configured for PROFINET IO with IRT High Performance. If topology-based initialization is to be used, the topology will need to be configured for all PROFINET devices.

5. Controller-controller data exchange broadcast:
   - Specify which address areas are to be used for sending and receiving.

4.3.4 Inserting and configuring SIMOTION D

4.3.4.1 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, 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 in the dialog or insert it at a later point via HW Config by selecting it from the hardware catalog under SIMOTION Drive-based > SIMOTION D4x5 > 6AUxx > V4.2 - PN-V2.2xx.

Implementing the PROFINET interface:
- Already integrated in the case of D410 PN 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.

4.3.4.2 Inserting and configuring SIMOTION D4x5-2/D410 PN

General information
You have created a project and want to configure a SIMOTION D4x5-2 DP/PN or SIMOTION D410 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.
2. Select SIMOTION D under Device and click on a Device version e.g. D455-2 DP/PN.
3. Select **SIMOTION Version V4.2** and **SINAMICS S120 Integrated**.

![Insert SIMOTION device](image)

**Figure 4-30** 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 **IP address** and **subnet mask**. Click **OK** to confirm.

![Figure 4-31 Creating a new Ethernet subnet](image)

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.
4.3 Configuring PROFINET IO with SIMOTION

Requirement
You have already created a project and now want to insert a SIMOTION D4x5-2 with option module CBE30-2-2 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. D435-2.
3. Select **SIMOTION Version** V4.3 and CBE30-2 as the **Option module**.
   You can also insert the option module CBE30-2 in HW Config at a later point (see Add and configure a CBE30-2 PROFINET board (Page 106)).

![Image](image.png)

**Figure 4-33** Create a SIMOTION D435-2 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 IP address and subnet mask. Click 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 4-34 SIMOTION D435-2 in HW Config

4.3.4.4 Add and configure a CBE30-2 PROFINET board

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 D435). 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 > V4.2 - PN-V2.2xx. Note the CPU type and version.

3. Click PROFINET module CBE30-2-PN. As soon as the appropriate CBE30-PN is selected, 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).

7. Select the subnet.

8. Assign the required IP address.

9. Accept the settings by clicking **OK**.
4.3.5 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 P350.

Procedure
1. Click on Insert SIMOTION device to open the device selection dialog box.
2. Under Device, select SIMOTION P, then click on a Device version (e.g. P350 PN), and select SIMOTION Version V4.2.
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 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.
4.3.6 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 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 SIMOTION Version V4.2.

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 IP address and the subnet mask here. Click OK to confirm.

![Properties - Ethernet interface PNIO (R0/S2.0)](image)

Figure 4-36 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.
4.3 Configuring PROFINET IO with SIMOTION

4.3.7 Creating a sync domain

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

Note

All devices that exchange data via IRT must belong to a single Sync-Domain and be directly connected to one another; i.e. the connection must not be interrupted by devices that do not support IRT, as this will prevent the synchronization information from being passed on.
Procedure

1. In HW Config, open the station with the PROFINET devices to be involved in IRT communication and select, for example, the 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.

4. Set the synchronization type to Sync-Master. IRT High Performance is the standard setting on SIMOTION controllers.

5. Confirm the settings with OK.

6. Then, select all devices which are to be configured as sync slaves (keep the Ctrl key depressed and select the devices one after the other).

7. Then, click on the Properties device button.

8. Set the synchronization type to Sync-Slave in the dialog box.

9. Confirm the settings with OK.

Note

Any devices for which not synchronized is selected will not be involved in IRT communication, but will automatically take part in RT communication.
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/I/O system. It is then entered in the list of stations/devices.

### 4.3.8 Defining send clock and refresh times

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

**Note**

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.

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

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

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

**Note**
Communication should be no faster than required, irrespective of what the maximum communication speed may be. This will reduce the bandwidth requirement and the relieve the load that the devices need to support.
Defining update times for PROFINET IO using PROFINET devices with RT

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

1. Click the path for the PROFINET IO system in HW Config and select Object properties from the context menu. The dialog is displayed.

2. Switch to the Update time tab and select the device from the overview containing all the IO devices.

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

4. Click OK to confirm.

Send clock/DP cycle ratio for SINAMICS_Integrated (SIMOTION D) with IRT High Performance

If a SINAMICS drive is being operated on a SIMOTION D via PROFINET IO IRT High Performance, the DP cycle for the integrated PROFIBUS must be the same as the position control cycle. The DP cycle is set to 3 ms by default. In most cases, this will not be the same as the 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 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 \( T_{dp} \). 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 PROFINET is operated isochronously, the position control cycle clock must always correspond to the PROFIBUS cycle clock. The position control cycle clock and the PROFIBUS cycle clock can be scaled to the PROFINET cycle clock.

**Example:**

PROFINET send clock = 0.5 ms

PROFIBUS cycle clock = position control cycle clock = 1 ms

The PROFIBUS cycle clock can operated relative to the PROFINET cycle clock at a ratio of 1:1 to 1:16.
4.3.9 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.

Options available for isochronous use

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

- The MACF (master application cycle) is set for the DP slave. Only MACF=1 is supported if you are also using PROFINET IO. All PROFIBUS devices run in the slow position control cycle clock.
- 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 have been configured and used on both Servo_fast devices, only CACF=1 is permitted as the setting. During scaling down of the cycle clocks, Servo_fast must be scaled to the servo at a factor of 2, 4, 8, or 16.

The following settings are possible:

<table>
<thead>
<tr>
<th>Product version</th>
<th>Property</th>
<th>Application cycle of the devices on PROFINET IO</th>
<th>Application cycle of the devices on PROFINET IO (CBE30-2)</th>
<th>Application cycle of the devices on PROFIBUS DP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before V4.2</td>
<td>1 position control cycle clock</td>
<td>--</td>
<td>--</td>
<td>Servo</td>
</tr>
<tr>
<td>Additionally for V4.2</td>
<td>2 position control cycle clocks</td>
<td>Servo_fast</td>
<td>--</td>
<td>Servo</td>
</tr>
<tr>
<td>or higher</td>
<td></td>
<td></td>
<td>Servo</td>
<td></td>
</tr>
<tr>
<td>Additionally for V4.3</td>
<td>2 position control cycle clocks</td>
<td>Servo_fast</td>
<td>Servo</td>
<td>Servo</td>
</tr>
<tr>
<td>and higher</td>
<td></td>
<td></td>
<td>Servo</td>
<td></td>
</tr>
</tbody>
</table>

Communication
System Manual, 02/2012
121
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 (integrated 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.

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, e.g. System cycle clock PROFINET=1ms and PROFIBUS DP=4ms.
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. The values for the Servo and Servo_fast are displayed here; in the example, the cycle clock ratio is 1:4.

![System Cycle Clocks](image)

**Figure 4-41** Servo and Servo_fast system cycle clocks

---

**Note**

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

---

### 4.3.10 Configuring a topology

#### 4.3.10.1 Topology

**Introduction**

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

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

- Using the topology editor (Page 126)
- Using the object properties
4.3.10.2 Topology editor (graphical view)

Procedure

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

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

The topology editor offers the user 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.
4. Click **Graphical view** to bring the tab into the foreground.

![Topology editor (graphical view)](image)

Figure 4-42  Topology editor (graphical view)

5. Establish connections between ports by pressing and holding down the left mouse button and drawing a line between the two ports to be connected. The **Interconnection Properties** window opens.
6. The port interconnection is displayed: You can configure the cable data: A cable length of < 100 m is set by default (recommended length). Alternatively, you can configure the signal propagation delay:

![Interconnection Properties](image)

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.

**4.3.10.3 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).
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.

**4.3.11 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/D410 PN (Page 101)).
4.3 Configuring PROFINET IO with SIMOTION

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.

4.3.12 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 message frame in SIMOTION SCOUT V4.2 or higher
For V4.2 and higher, the message frames are assigned automatically in SIMOTION SCOUT by means of symbolic assignment. For isochronous communication, a message frame must be configured (e.g. message frame 105) which enables the drive to be synchronized with PROFINET. The message frame 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 message frame 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 message frame will automatically be created (symbolic assignment).
5. SelectProject > 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 itemProject > Set up addresses.

6. Once the axis configuration process is complete, clickCommunication > Message frame configuration under the drive unit in the project navigator. Tab IF1: PROFIdrive PZD message frames now lists the message frame for the drive (e.g. SIEMENS message frame 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 message frame interconnection. For this purpose, make sure that a check mark is applied forProject > Use symbolic assignment in the menu.

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

Settings for isochronous mode (V4.2 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 chooseEdit > Object properties in the menu.
2. Click theDetails... button on the Isochronous tasks tab.
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 4-44 Configuring an isochronous task on the SIMOTION device.](image)

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.

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 is slower than the IRT cycle clock, for example, although the servo cycle clock must already have been taken into account in the IRT cycle 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

During the next steps, you must include the drive in the sync domain (see Creating a sync domain), load the IP address to the drive (see Assigning device names and IP addresses to IO devices (Page 135)), and interconnect the ports (see Interconnecting ports via the topology editor (table view) (Page 126)).
See also

Creating a sync domain (Page 114)

4.3.13 IP address and communication name

Introduction

Devices and controllers have a fixed MAC address, a configurable IP address, and a communication name. The IP address, subnet mask, and communication name are defined within the Ethernet interface properties in the engineering software. This ensures the devices within the project have their own unique assignment.

You can find the communication name (device name) in the Properties - Interface dialog by double-clicking on the device or the controller. Click on Properties in the dialog to define the IP address. See also Add and configure a CBE30-2 PROFINET board (Page 106).

Communication name guidelines

The communication name for the device is the same as the device name (e.g. Drive1 for a drive). The communication name for a controller is the name of the PROFINET interface (e.g. CBE30xPNxI/O for a CBE30 on SIMOTION D).

Communication names are subject to certain syntax rules, i.e. they must always be DHCP-compliant. Additional boundary conditions also apply to SIMOTION and SIMATIC.

- Letters a-z and numbers 0-9 may be used.
- Special characters are not permitted: these include ! “# $ % & / ( ) = ? * ' _ : ; > < , # + | ~ { } ] [ { }
- The name may consist of more than one part.
  - Label.Label.Label.Label
  - Periods are used as separators.
  - The label must start and end with a letter.
  - Labels must not exceed 63 characters.
  - Labels must not begin with "xn-".
- The total maximum length for a name is 63 characters
- Reserved names "port-xyz" or "port-xyz-abcd"E
- No distinction is made between upper and lower case. However, since all names are shown in lower case in the engineering system, you should use lower-case letters only.
- In V4.3 or higher, SIMOTION SCOUT also processes names that do not comply with the ST naming convention. No more characters are replaced by "x". In places where it must be clearly recognized that a device name exists such as in the programming, names are shown in " ".
Initialization of the controller and devices in online mode

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

Controller initialization

- Download the application
- Engineering software
  - HW Config, NetPro, SCOUT
  - Primary Setup Tool (PST)
- Via the application (the _setNameofStation system function for SIMOTION)

Note

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

Device initialization

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

Topology-based initialization for devices

Devices can also be initialized without an MMC or CF card. This method is known as topology-based initialization and is only supported as of certain software versions.

- SIMATIC S7-300 FW ≥ V 2.7
- SIMATIC S7-400 FW ≥ V 5.2
- SIMOTION FW ≥ V4.1.2 with PN V2.2
- SINAMICS FW ≥ V2.5.1.10 with PN V2.2
- ET200S FW ≥ 6.0
- ET200S HS FW ≥ 2.0
4.3.14 Assigning device names and IP addresses to IO devices

Introduction

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

You can specify the IP address in the Properties - Ethernet interface...dialog (double-click 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 communication names and then assigns the IP address defined in Engineering to them. We recommend leaving this function activated.

![Properties - SINAMICS S120 CU310 C0020](image)

Figure 4-46 Properties SINAMICS S120
IO device initialization

Note

If you are connecting the programming device/PC directly to the device’s PROFINET interface, you can use a patch or crossover cable.

During commissioning, we recommend that the device to be initialized is connected directly to the programming device/PC.

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

2. Click the **Browse** button.
3. The Browse Network dialog box opens. The connected nodes are displayed.

![Browse Network dialog box]

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

You can also perform the node initialization in SCOUT.

- In SCOUT, execute **Reachable nodes** and, in the dialog box displayed, right-click the device that you want to edit.

- Execute **Edit Ethernet nodes**. The corresponding dialog box is displayed.

![Edit Ethernet nodes](image)

- 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.

### 4.3.15 IP address and communication name via user program/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 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.

**Note**

From V4.3 and higher the IP addresses and device names of standard Ethernet interfaces are also given in this way.
This mechanism allows the IP addresses and station names to be adjusted without making changes to the project. The IP settings can be changed locally, especially for series machines.

Applications

- You configure the IP address and the device name in the project

A project is specific to the plant or machine and specifies the IP address and device name. For this reason, the checkboxes "Use different method to obtain device name" and "Use different method to obtain IP address" are not set in the project. The following thus applies:

  - The device name and the IP address from the project are leading. Following power off/on, the IP address and device name from the project are again valid.
  - You can nevertheless set an "emergency IP" via DCP (e.g. "Edit Ethernet node"), e.g. to load a project with corrected IP address if access is no longer possible via the IP address configured in the project on the CPU.

    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, since SIMOTION boots in certain cases during downloading, and the IP address from the project becomes effective after booting.

- You configure the IP address and the device name on-site

This application is of significance for a series machine, for example, in which the IP address and device name are first assigned on-site, even if the project is not available. For this reason, the checkboxes "Use different method to obtain device name" and "Use different method to obtain IP address" are set in the project and no IP address or device name is configured. The following thus applies:

  - If the device name and/or IP address have been set with STEP 7 or a primary setup tool, these set values remain valid even after power off/on.
  - SIMOTION currently offers no support for spare parts, that is, after replacement of the CPU, the IP address and NameOfStation must be set again via DCP or the user program.
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 Obtain IP address using a different method checkbox and confirm by clicking OK.

![Figure 4-48 Obtaining the IP address using a different method]
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** checkbox on the General tab and confirm by clicking **OK**.

![Figure 4-49 Obtaining the device name using a different method](image)

General information on modifying the IP address

- The IP address can be modified at any time.
- You can power up without a valid IP address.
- It is also possible to change the IP suite (IP address, subnet, router address, device name), which includes the transition to an invalid IP address, in RUN mode and during actual communication. During this process, a brief interruption in the application can be tolerated.
- While the IP suite is being changed, the failure and restoration of the PROFINET IO Controller can be tolerated.
- Where there is a fixed application relation (AR) to a higher-level IO controller, no negative acknowledgement is issued when the IP address is modified via DCP. This behavior also applies during use as an I device.
- When the IP address is modified via the user program, the AR to the higher-level IO controller is aborted locally and the change is accepted. This results in a CPU stop on the higher-level IO controller.
- Changing the IP address for PN/IO interfaces via the UP is not possible (see below).
- If the IP address has been changed externally via DCP, it is retained after power off/on.
General information on changing the device name

- The device name can be changed at any time.
- You can power up without a valid device name.
- It is also possible to change the IP suite (IP address, subnet, router address, device name), which includes the transition to an invalid IP address, in RUN mode and during actual communication. During this process, a brief interruption in the application can be tolerated.
- Changing the device name, which includes the transition to an "empty" device name, is possible in RUN mode and during actual communication. During this process, a brief interruption in the application can be tolerated.
- Changing the device name via DCP with a fixed application relation causes the application relation to be aborted. This also applies during use as an I device.
- With an IO controller, all existing application relations to lower-level IO devices are aborted and reestablished when the device name is changed.
- The unique device name is not checked; duplication can only be detected on the higher-level IO controller or in the LifeList.

System functions for "Use different method to obtain IP address"

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

<table>
<thead>
<tr>
<th>PN/IE interfaces (PN Basic interfaces)</th>
<th>Assigning NameOfStation via the UP (_setNameOfStation, etc.)</th>
<th>Assignment of IP Suite via UP (_setIPConfig, _getIPConfig)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

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.
4.3.16 Configuring media redundancy (V4.3 and higher)

4.3.16.1 Creating ring topology

Setting up ring topology

In order to use MRP, you must establish a ring topology. The following modules can be nodes in the ring:

- SIMOTION modules V4.3 or higher
- Switches (SCALANCE X2 IRT, V5.0 and higher)
- SINAMICS S120 (CU320-2/CU310-2), SINAMICS V4.5 and higher
- SIMATIC modules

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 master 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 required SIMOTION modules. SIMOTION modules assume the role of redundancy clients.

Figure 4-50  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 4-51  Ring topology in the Topology Editor
4.3.16.2 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 master system in the working area of HW Config.

2. Execute Edit>PROFINET IO>Domain Management.

The Domain Management dialog opens.

3. Click the MRP domain tab in the foreground.

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.

   ![Edit Media Redundancy](image)
   
   **Figure 4-53  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.

**4.3.16.3 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:

1. Select the device in the work area of HW Config.
2. Double-click on the interface lines in the detailed view of the device, for example, on X1 in a SCALANCE switch.

The Properties - PN (xx) dialog opens.

3. Click the Media redundancy tab in the foreground.
4. Select the MRP domain under Domain. If you are using only one domain, this is preselected.
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 Diagnostics alarms option if you want to see a ring communication fault 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 detail view of the device, double-click on the interface line, e.g. on X1400 for a SIMOTION device.

The Properties - Interface (xx) dialog opens.

3. Click the Media redundancy tab in the foreground.
4. Select the MRP domain under Domain. If you are using only one domain, this is preselected.
5. Activate the Diagnostics alarms option if you want to see a ring communication fault in the diagnostics buffer and in the device diagnostics.
4.4 Configuring direct data exchange (data exchange broadcast) between IO controllers

4.4.1 Introduction

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

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

Note
This function is not available for SIMATIC CPUs.

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

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

Recommendation

We recommend, initially configure the send areas for all PROFINET devices and then the receive areas. Adopting this procedure will enable you to assign the previously defined send areas when defining the receive areas. This prevents invalid inputs.
4.4 Configuring direct data exchange (data exchange broadcast) between IO controllers

Data volume

Around 3 KB of data can be transmitted. 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 $5 \times 24$ bytes. The remaining data is available for application-specific data exchange broadcast.

Note

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

This FAQ section deals with the subjects of distributed gearing and controller-controller data exchange broadcast.

4.4.2 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 of the sender, the start address from the I/O area and the length of the address area to be used for sending. Comment the data area so that you will be able to identify the data transmitted via this area later on. A variable may not exceed 254 bytes in size.

5. Confirm the settings with **OK**.

6. Repeat steps 3 to 5 for further send areas.

7. Change the preset diagnostics address for the send areas, if required.

8. Confirm your entries with **OK**.

A single diagnostics address must be assigned for the communication relationship in which a PROFINET interface is the transmitter for direct data exchange.

### 4.4.3 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. In the Properties dialog box of the receiver, enter the start address of the address area via which the reception is to be implemented. The length of the address area must not be changed as it is automatically adapted to the length of the send area. The configuration can only be compiled if the send and receive areas have identical lengths!

8. Repeat steps 3 to 7 for further receive areas.

9. A diagnostics address is reserved for each assigned sender via which the receiver can detect a failure of the sender.

10. Click the Diagnostics addresses button if you want to edit these addresses.

11. Confirm your entries with OK.

4.5 Configuring the iDevice

4.5.1 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).

An I device is a controller which also assumes the function of an IO device. The term "I device" is used to refer to an intelligent IO device. Intelligent I devices are characterized by the fact that their input/output data is pre-processed in the I device, rather than being made directly available to the higher-level IO controller by actual inputs/outputs.

When operating as an I device, a SIMOTION device can be used for data exchange 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 iDevice involves providing distributed synchronous operation beyond the limits of a project (see the *Motion Control Technology Objects Synchronous Operation, Cam Function Manual*).

**Note**

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

### Properties of an I device

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

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

The following boundary condition applies in terms of the options available for combining functions:

<table>
<thead>
<tr>
<th>SIMOTION function</th>
<th>RT IDevice</th>
<th>RT controller</th>
<th>IRT I device</th>
<th>IRT controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>RT IDevice</td>
<td></td>
<td><strong>X</strong></td>
<td>-</td>
<td><strong>X</strong></td>
</tr>
<tr>
<td>RT controller</td>
<td><strong>X</strong></td>
<td>-</td>
<td><strong>X</strong></td>
<td><strong>X</strong></td>
</tr>
<tr>
<td>IRT I device</td>
<td>-</td>
<td><strong>X</strong></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IRT controller</td>
<td><strong>X</strong></td>
<td><strong>X</strong></td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

*Either an IRT I device or an IRT controller

As with any other IO device, an I device's PROFINET interface requires parameter assignment data in order to operate. With IO devices, this data is usually loaded via the associated IO controller in the form of parameter assignment data sets. Two options are available for I devices. An I device's interface and PROFINET interface ports can either be parameterized by the higher-level IO controller or by the I device itself on a local level. The preferred option can be selected as part of the I device's configuration.

If parameters are being assigned locally, the required data is loaded to the I device while the Engineering System is being downloaded. Parameter assignment data for the PROFINET interface is contained in the device download data. The higher-level IO controller does not need to be used for parameterizing the PROFINET interface of the I device. This option should be used if the IDevice is to be operated with RT, and an IRT controller is configured on the same interface.

If the higher-level IO controller is being used for parameter assignment, the parameter assignment data for the I device's PROFINET interface must be loaded by the IO controller together with the remaining parameter assignment data. For this purpose, the IO controller loads parameter assignment data sets for the PROFINET interface to the I device. If the I device is to be operated with IRT, the parameter assignment data will need to be loaded by the IO controller.
If the I device is being operated with IRT, the I device's send clock must be set to match the send clock for the sync domain of the higher-level IO controller's PROFINET IO system. If the I device is being operated with RT, the I device's update time must be either set to match the send clock for the sync domain of the higher-level IO controller's PROFINET IO system, or down-scaled by a multiple of this send clock.

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

Table 4-5 Send clocks/update times of an I device

<table>
<thead>
<tr>
<th>Higher-level IO controller and I device with IRT, no local PROFINET IO system or local PROFINET IO system with IO devices with RT</th>
</tr>
</thead>
<tbody>
<tr>
<td>• I device send clock:</td>
</tr>
<tr>
<td>- Must be the same as the send clock for the higher-level IO controller</td>
</tr>
<tr>
<td>- To be set on the I device in &lt;PROFINET Interface&gt; properties using the &quot;Sending cycle&quot; drop-down list box on the PROFINET tab</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Higher-level IO controller with IRT and I device with RT, local PROFINET IO system with IRT</th>
</tr>
</thead>
<tbody>
<tr>
<td>• I device update time:</td>
</tr>
<tr>
<td>- 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</td>
</tr>
<tr>
<td>- To be set on the substitute I device in &lt;PROFINET Interface&gt; properties, under the update time on the &quot;IO Cycle&quot; tab</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Higher-level IO controller and I device with RT, no local PROFINET IO system</th>
</tr>
</thead>
<tbody>
<tr>
<td>• I device update time:</td>
</tr>
<tr>
<td>- Any of the possible update times for the I device may be set.</td>
</tr>
<tr>
<td>- To be set on the substitute I device in &lt;PROFINET Interface&gt; properties, under the update time on the &quot;IO Cycle&quot; tab</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Higher-level IO controller and I device with RT, local PROFINET IO system with IRT:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• I device update time:</td>
</tr>
<tr>
<td>- Must be less than or equal to the send clock for the IO controller in the I device</td>
</tr>
<tr>
<td>- To be set on the substitute I device in &lt;PROFINET Interface&gt; properties, under the update time on the &quot;IO Cycle&quot; tab</td>
</tr>
</tbody>
</table>

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.
**Configuration procedure**

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

Since a manual process in the hardware catalog is required to create a substitute I device, there is no automatic alignment between the project with the I device and the corresponding substitute I device in the GSD file. As a result, no subsequent amendments can be made to the I device configuration. If an amendment is made, however, a new GSD file will have to be created and installed. Where numerous amendments have subsequently been made to the configuration of a given I device, with multiple GSD files created and installed by the I device, the version shown under "Preconfigured Stations" in the hardware catalog will always be the most recent one. The version will only be updated, however, if the identifier used for the substitute I device when creating and installing the GSD file remains the same. Only the input and output addresses for data exchange may be amended in the project for the higher-level IO controller.
Since I devices connected to their higher-level IO controller and IO devices connected on the PROFINET IO system of a single I device are connected via one and the same PROFINET interface, they will also be located in one and the same Ethernet subnet. This means that the device names and IP addresses of all these devices must be different from each other, and the subnet masks must be identical. It is particularly important to bear this in mind if the higher-level IO controller and the I device are in different projects, as HW Config cannot check that device names, IP addresses, and subnet masks are consistent across different projects.

Device name (NameOfStation) for the I device

As with all IO devices on PROFINET IO, a device name also has to be defined for the I device in the configuration. As the device name (NameOfStation) for the I device is set in the properties for its PROFINET interface, it is identical to the device name of the IO controller in the I device. The name set is written to the GSD file for the substitute I device when this file is created and installed. When the substitute I device is inserted into the higher-level IO controller's PROFINET IO system, the device name previously assigned in the GSD file will be accepted into the configuration. In all cases, it is important to ensure that the device name in the configuration of the higher-level IO controller is the same as the device name defined for the I device. As a result, device names must not be amended after the higher-level IO controller has been added to the PROFINET IO system.

If the device names are different, the higher-level IO controller will be unable to power up the relevant I device and, consequently, be unable to commence cyclic exchange of input/output data.

The following scenarios result in different device names and must therefore be avoided:

- If you use a SIMOTION device as the higher-level IO controller, the device name (NameOfStation) of the I device may not contain any "-" marks, as these will be changed into "x" when you insert the I device into the PROFINET IO system.

- Since a device name must not be used twice within an Ethernet subnet, any device name which already exists will be amended when a substitute I device is inserted into the PROFINET IO system of its higher-level IO controller. In view of this, it is important to ensure that the device name previously assigned in the GSD file is not used at this stage.

- If more than one substitute I device from the same "Preconfigured Stations" entry is inserted into the higher-level IO controller's PROFINET IO system, the device name previously assigned in the GSD file will be amended. With this in mind, a substitute I device must also be created for every I device to be used in a PROFINET IO system.

See also

Creating an I device (Page 160)
4.5.2  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 I device must therefore be re-exported/reinstalled and integrated into the project of the newly designated higher-level 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 containing a GSD device on a computer where the GSD is not installed and then open with SCOUT, the GSD file is installed automatically. 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 controller, as this is the minimum version required to import the GSD V2.25 file.

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

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 I device 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.

4.5.3 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 153).

Procedure

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

Figure 4-57  I device Properties dialog box
5. You have to activate different checkboxes depending on whether I device communication is to take place on the higher-level controller using RT or IRT.

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

   - **I device using IRT:**
     If the I device is to be operated on the higher-level IO controller using IRT, you also have to activate the **Parameterization of PN interface and its ports on higher-level controller** and **Operate total I device (all submodules) isochronously** 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 option, cyclic communication between the higher-level IO controller and the I devices can only take place via RT. On the substitute I device, this selection causes the IO cycle tab to appear in the Properties dialog box for the substitute I device's PROFINET interface. It will then be possible to select the **Servo** entry under **Assign I device isochronously** 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.

**Note**

Please note that if the I device is to work in IRT mode, the device name and IP address of the controller must be assigned. You therefore have to set this option on the I device. To do this, refer to chapter IP address and communication name via user program/DCP (Mini-IP-Config) (Page 138) and Inserting a substitute I device in the higher-level IO controller (Page 165).

6. If the I device is being operated with IRT, its send clock will have to be set. To do this, select the **PROFINET** tab and set the send clock as appropriate.

7. 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. Do not perform any further settings in the **Sender** and **Receiver** tabs.

8. Click **OK** to accept these settings and save the project.

9. Continue by creating the substitute I device.
4.5.4 Exporting the GSD file for the I device

The GSD file always needs to be exported so that an I device in a project can be used on another PC.

**Requirement**

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

1. First save the project.
2. Select **Options > Create GSD file for I device** .... The "Create GSD file for I device" dialog opens.
3. Select the I device and enter a name for the substitute I device. The substitute I device will appear under this name in "Preconfigured Stations" in the HW catalog.

   ![Create GSD File for I-Device dialog](image)

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**.

4.5.5 Creating a substitute I device

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

**Requirement**

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

**Procedure 1:**

1. First save the project.
2. Create an I device as described in Chapter Exporting the GSD file for the I device (Page 163). There is no need to export the file; instead, it is installed immediately.
3. Click **Create** and then **Install**.

![Create GSD File for I-Device window](image1)

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

**Procedure 2:**

1. Select **Options > GSD files**... The "Install GSD files" dialog box opens.

2. Click **Browse...**. The Find folder dialog opens.

3. Select the path in which the GSD files of the substitute I device are stored and click on **OK**.

![Install GSD Files window](image2)
4. Select the required GSD files and click **Install**.

5. Click **Close**. The substitute I devices will now be available under "Preconfigured Stations".

![Image of hardware catalog entry](image)

**Figure 4-58** I device entry in the hardware catalog

### 4.5.6 Inserting a substitute I device in the higher-level IO controller

**Requirement**

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

1. Open the hardware catalog.
2. Drag the relevant substitute I device from the hardware catalog (PROFINET IO > Preconfigured stations) to the PROFINET IO system. The substitute I device is displayed as a normal IO device on the PROFINET IO system. Whether or not ports are displayed depends on whether Parameter assignment of the PN interface and its ports on higher-level IO controller is selected.

![Image of PROFINET IO system]

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

Figure 4-59 I device on the IO controller
Assigning the IP address for the substitute I device

1. Double-click the I device to display the Properties dialog.
2. Deactivate the Assign IP address via IO controller option.

The IP address should not be assigned by the higher-level IO controller, since it is already allocated in the Step7 project of the I device. This option is only available if you have previously activated Obtain device name using a different method and Obtain IP address using a different method under Properties (see also Mini-IP-Config (Page 138)).
Setting the synchronization type and isochronous mode for an I device with IRT

1. Double-click the **Interface entry** in the rack to display the **Interface properties** dialog box.
2. In the **Synchronization** tab, select **Sync-Slave** and **IRT** as synchronization type and RT class respectively.
3. On the **IO cycle** tab, select automatic **Mode** under update time and select **Assign IO device isochronously** Servo under isochronous mode.

Setting the update time and send clock

I device with RT

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

I device with IRT

- The update time needs to be set for I devices with IRT. The setting made for the send clock in the I device's project must be the same as for the send clock in the higher level IO controller's project. Set the send clock for the higher-level project in HW Config using **Edit > PROFINET IO > Domain Management**.
4.5.7 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 I devices displayed under "Preconfigured Stations" are only updated when a new GSD file is created and installed.

4.6 Loading the communication configuration

4.6.1 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.

4.7 Data exchange between SIMATIC and SIMOTION via PROFINET

4.7.1 Overview of data exchange between SIMATIC and SIMOTION

Description

SIMATIC and SIMOTION can be linked using the following functions:

- TCP/UDP user communication
- PROFINET IO/RT, via S7-300 CP as a device
4.7 Data exchange between SIMATIC and SIMOTION via PROFINET

- PROFINET IO/RT, via PN/PN coupler
- PROFINET IO/RT, via I device

See also

Data exchange through the use of iDevices (Page 170)
Communication using standard protocols (Page 173)

4.7.2 Data exchange through the use of iDevices

Description

With PROFINET IO/RT, data can be exchanged via an iDevice.

Note

A PROFINET device connected to a SIMOTION CPU may have a maximum submodule size of 254 bytes.

This limit must be observed in particular if a SIMATIC CPU is configured as an iDevice for a SIMOTION CPU, since submodules > 254 bytes can be configured for the SIMATIC iDevice.
Note
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 Chapter Configuring the iDevice (Page 153).

Please also refer to the FAQ on configuration I device FAQ (http://support.automation.siemens.com/WW/view/en/29578823).

See also
Overview of data exchange between SIMATIC and SIMOTION (Page 169)

4.7.3 PN to PN coupler

Description
The PN/PN coupler is used to link two PROFINET IO system with one another and to exchange data between them. The maximum size of the data which can be transferred is 256-byte input data and 256-byte output data.

As a device, the PN/PN coupler has two PROFINET interfaces, each of which is linked to another subnet.

During configuration, two IO devices are derived from one PN/PN coupler which means that there is one IO device for each controller with its own subnet. The other part of the PN/PN coupler is called the coupling partner. Once configuring is complete, the two parts are joined.
The two machines in the figure are isolated via the PN/PN coupler. If it is also possible, for example, to use the programming device of machine B to go online on machine A, a jumper can be inserted between ports 2 and 3 in the PN/PN coupler. This removes the isolation.

**Note**
Detailed information about the PN/PN coupler is contained in the appropriate device documentation.

**Configuring PN/PN couplers**
Two PROFINET devices are created for the purpose of configuring the PN/PN coupler: one on the left (X1) and one on the right (X2). You use HW Config to configure the PN/PN coupler. Once both subnets in a project have been configured, you can use STEP 7 to configure the PN/PN coupler for both subnets. Once the subnets in various projects have been configured, you must configure the coupler in each project.

**Note**
The PN/PN coupler is used for data exchange between SIMOTION and SIMATIC. It is also, however, the preferred solution for fast exchange of F signals between SIMATIC CPUs.
4.7.4 Communication using standard protocols

Description

TCP and UDP can be used to exchange data between SIMOTION, SIMATIC, other controls, and third-party systems. The PROFINET interface is a standard Ethernet interface and supports these protocols. The user program must handle the management of the connection. You can use this connection, for example, to exchange data between a SIMATIC CPU and SIMOTION controller via PROFINET.

UDP is a connection-less protocol; this means transmission is not followed by feedback indicating whether the receiver actually received the message. TCP is a connection-based protocol; this means a logical connection is established at the outset. Transmission only begins once this connection is present. The connection (i.e. message receipt) is monitored.

Data exchange via TCP:

- Establishing a connection
- Data management
- Connection monitoring
- Connection termination

The use of the system commands is described in detail in the Introduction to Ethernet (TCP/IP and UDP connections) section.

See also

Overview of SIMOTION system functions (Page 195)
Overview of SIMOTION system functions (Page 215)
Overview of data exchange between SIMATIC and SIMOTION (Page 169)
4.8 Diagnostic and alarm behavior

4.8.1 PROFINET IO alarm and diagnostic messages to SIMOTION

Description

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

PROFINET device diagnosis
- In three levels
- Device slot
- Channel for network components: address
- Fault location

Open network diagnosis
- SNMP
- Web

Figure 4-64 Diagnostics overview using the example of ET200

Device diagnostics

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

4.8.2 Diagnostics model

PROFINET IO uses completely standardized diagnostics mechanisms. This is especially helpful for manufacturer-wide device and system diagnostics.

Because of the large quantity structures, it is not possible to keep the status information of all stations in the IO controller. Therefore only the current diagnostic events are transferred to the IO controller via the standardized alarms.
The use of an acknowledged service enables the transfer of the diagnostic events in causal sequence. A station saves its status, which can be fetched directly by a diagnostics unit at any time using standardized data sets, see corresponding STEP7 documentation and documentation on the _readRecord() system function.

**Access to the alarm and diagnostic data**

For PROFINET IO, a differentiation is made between the following alarm and diagnostic messages:

- Alarms sent from IO devices to the IO controller
- Alarms that occur in the IO controller

The following figure shows the access possibilities to the diagnostic data:

1. **Diagnostics on PG**
   The PG reads the diagnostics directly from the IO device. Visualization takes place in the PG.

2. **Diagnostics on controller**
   The IO device sends the diagnostics to the IO controller, the response to the fault takes place in the controller.

![Diagram of access to diagnostic data](image-url)
4.8.3 Alarms on the IO controller

Description

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

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

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

<table>
<thead>
<tr>
<th>Alarm (TSl#interruptId)</th>
<th>TSl#eventClass</th>
<th>TSl#faultId</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Station failure (_SC_STATION_DISCONNECTED ( = 202))</td>
<td>16#39</td>
<td>16#CA</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16#CB</td>
<td>Station failure of an IO device</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16#CC</td>
<td>IO device fault present. Channel diagnostics or manufacturer-specific diagnostics pending.</td>
</tr>
<tr>
<td>Station reconnection (_SC_STATION_RECONNECTED ( = 203))</td>
<td>16#38</td>
<td>16#CB</td>
<td>An IO device has been reconnected without errors</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16#CC</td>
<td>IO device error corrected</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16#CD</td>
<td>An IO device has been reconnected, but with an error: set configuration &lt;&gt; actual configuration</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16#CE</td>
<td>An IO device has been reconnected, but error during module parameterization</td>
</tr>
</tbody>
</table>

Use of the TaskStartInfo

Information concerning the TaskStartInfo for the PeripheralFaultTask is contained in the Base Functions manual.
4.8.4 Alarms from the IO device to the IO controller

Description

The alarms are transferred using the PROFINET alarm mechanism from the IO device to its associated IO controller. The alarms are entered in the diagnostic buffer and can be evaluated using the PeripheralFaultTask. The following table shows how alarms are represented as PeripheralFaultTask.

<table>
<thead>
<tr>
<th>Alarm (TSI#InterruptId)</th>
<th>TSI#event Class</th>
<th>TSI#faultId</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diagnosis (incoming)</td>
<td>16#39</td>
<td>16#42</td>
<td>Incoming diagnostic interrupt</td>
</tr>
<tr>
<td>Diagnosis disappears (outgoing)</td>
<td>16#38</td>
<td>16#42</td>
<td>Outgoing diagnostic interrupt</td>
</tr>
<tr>
<td>Multicast Communication Mismatch</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Port Data Change Notification</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sync Data Changed Notification</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Isochronous Mode Problem Notification</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network component problem notification (_SC_DIAGNOSTIC_INTERRUPT (=201))</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process interrupt (_SC_PROCESS_INTERRUPT (= 200))</td>
<td>16#11</td>
<td>16#41</td>
<td>Process interrupt</td>
</tr>
<tr>
<td>Pull Alarm</td>
<td>16#39</td>
<td>16#51</td>
<td>PROFINET IO module has been removed or cannot be addressed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16#54</td>
<td>PROFINET IO submodule has been removed or cannot be addressed.</td>
</tr>
<tr>
<td>Plug Alarm</td>
<td>16#38</td>
<td>16#54</td>
<td>PROFINET IO module or submodule has been inserted, module type OK (actual configuration = set configuration)</td>
</tr>
<tr>
<td>Plug Wrong Submodule Alarm</td>
<td></td>
<td>16#55</td>
<td>PROFINET IO module or submodule has been inserted, but wrong module type (actual configuration &lt;&gt; preset configuration)</td>
</tr>
<tr>
<td>Return of Submodule Alarm (_SC_PULL_PLUG_INTERRUPT (=216))</td>
<td></td>
<td>16#56</td>
<td>PROFINET IO module or submodule has been inserted, but error during module parameterization</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16#58</td>
<td>IO status of a module has changed from BAD to GOOD</td>
</tr>
<tr>
<td>Status</td>
<td></td>
<td></td>
<td>Not Supported</td>
</tr>
<tr>
<td>Update</td>
<td></td>
<td></td>
<td>Not supported</td>
</tr>
<tr>
<td>Time data changed notification</td>
<td></td>
<td></td>
<td>Not supported</td>
</tr>
<tr>
<td>Upload and storage notification</td>
<td></td>
<td></td>
<td>Not supported</td>
</tr>
<tr>
<td>Pull module</td>
<td></td>
<td></td>
<td>Not supported</td>
</tr>
<tr>
<td>Manufacturer-specific</td>
<td></td>
<td></td>
<td>Not supported</td>
</tr>
<tr>
<td>Profile-specific</td>
<td></td>
<td></td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Alarm types indicated as "not supported" are acknowledged by the SIMOTION controller with "not supported" and not entered in the diagnostic buffer.
Use of the TaskStartInfo

Information 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.

Example of program for reading out TSI# information of the PeripheralFaultTask

```plaintext
; ....
; Variable declaration
    PERIPHERIE_Alarminfo  : STRUCT
        ALH_internalID   : UDINT;
        ALH_EingangsAdresse : DINT;
        ALH_AusgangsAdresse : DINT;
        ALH_DiagnoseAdresse : DINT;
        ALH_Details        : DWORD;
        ALH_Starttime      : DATE_AND_TIME;
        ALH_EventClass     : UINT;
        ALH_FaultID        : UINT;
    ; ....
; Reading out the I/O fault task information:
    P_ALH_Info[p_alhcount].ALH_internalID := TSI#interruptID;
    P_ALH_Info[p_alhcount].ALH_EingangsAdresse := TSI#logbaseadrin;
    P_ALH_Info[p_alhcount].ALH_AusgangsAdresse := TSI#logbaseadrout;
    P_ALH_Info[p_alhcount].ALH_DiagnoseAdresse := TSI#logdiagadr;
    P_ALH_Info[p_alhcount].ALH_Details := TSI#details;
    P_ALH_Info[p_alhcount].ALH_Starttime := TSI#starttime;
    P_ALH_Info[p_alhcount].ALH_EventClass := TSI#eventClass;
    P_ALH_Info[p_alhcount].ALH_FaultID := TSI#faultID;
; ....
```
4.8.5 Alarms for direct data exchange between IO controllers

Description

For PROFINET IO with IRT, communication monitoring takes place between IO controllers. If this establishes that IRT data is no longer being received (either there is no data arriving, or it is arriving too late) a station failure alarm is generated. If communication is re-established, a station reconnection alarm is generated. If IRT data arrives late on three occasions, a station failure alarm is reported.

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:

<table>
<thead>
<tr>
<th>Alarm (TSI#interruptId)</th>
<th>TSI#eventIdClass</th>
<th>TSI#faultId</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Station failure (_SC_STATION_DISCONNECTED (= 202))</td>
<td>16#39</td>
<td>16#F3</td>
<td>The receiver in the direct data exchange is no longer receiving data.</td>
</tr>
<tr>
<td>Station reconnection (_SC_STATION_RECONNECTED (= 203))</td>
<td>16#38</td>
<td>16#F0</td>
<td>The transmitter in the direct data exchange has started up and is able to transmit.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16#F1</td>
<td>The receiver in the direct data exchange has started up without errors and the receiver is receiving data again (all receiving areas are available).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16#F2</td>
<td>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).</td>
</tr>
</tbody>
</table>

4.8.6 Alarms for SINAMICS S120 drives

Description

In SINAMICS firmware Version 4.5 and higher, SINAMICS drives (CU320-2) 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.
4.8 Diagnostic and alarm behavior

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 176)

4.8.7 System functions for the diagnostics for PROFINET or PROFIBUS

Overview of system and diagnostics functions

The following table provides an overview of the various system and diagnostics functions for PROFINET IO. Differences with PROFIBUS DP are also indicated.

You will find detailed information about each of the functions in the reference list for functions Parameter Manual, SIMOTION system functions/variables for devices.

<table>
<thead>
<tr>
<th>Function</th>
<th>Note</th>
<th>PROFIBUS</th>
<th>PROFINET</th>
</tr>
</thead>
<tbody>
<tr>
<td>_getStateOfSingleDpSlave()</td>
<td>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).</td>
<td>Logical diagnostic address of the DP slave</td>
<td>Logical diagnostics address of the IO device</td>
</tr>
<tr>
<td>_getStateOfDpSlave()</td>
<td>_getStateOfDpSlave provides information concerning whether the PROFIBUS DP slave or the PROFINET IO device is activated or deactivated.</td>
<td>Logical diagnostic address DP slave</td>
<td>Logical diagnostics address of the IO device</td>
</tr>
<tr>
<td>_getStateOfAllDpStations()</td>
<td>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.</td>
<td>Logical diagnostics address of the interfaces</td>
<td>Logical diagnostics address of the interfaces</td>
</tr>
</tbody>
</table>

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.
### 4.8 Diagnostic and alarm behavior

<table>
<thead>
<tr>
<th>Function</th>
<th>Note</th>
<th>PROFIBUS</th>
<th>PROFINET</th>
</tr>
</thead>
</table>
| `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. |                                                                                             |                                                                                            |
| `getNextLogAddress()`    | All configured logical addresses of a segment can be determined using this function.                                                                                                                                                                                                                                                | Logical diagnostics address                                                                 | Logical diagnostics address                                                                 |
| `readDiagnosticData()`   | 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 device, of the module/slot or IO address of the channel/subslot |
### 4.8 Diagnostic and alarm behavior

<table>
<thead>
<tr>
<th>Function</th>
<th>Note</th>
<th>PROFIBUS</th>
<th>PROFINET</th>
</tr>
</thead>
<tbody>
<tr>
<td>_readVariableDiagnosticData ()</td>
<td>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.</td>
<td>-</td>
<td>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.</td>
</tr>
<tr>
<td>_readDriveFaults()</td>
<td>SINAMICS only (ET200FC, S120, etc.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This function is used to read the current fault buffer entry in the drive.</td>
<td></td>
<td>Logical start address of drive (slot).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Each valid logical I/O address of the subslot concerned or diagnostic address of the PAP (for modules without cyclic data)</td>
</tr>
</tbody>
</table>

#### 4.8.8 PROFINET device diagnosis in STEP 7

**Device diagnosis in STEP 7**

In SCOUT, HW Config can be used to perform an online device diagnosis via PROFINET. The diagnosis supplies not only the slot and the channel number, but also the error type. The diagnosis operates similar to that for PROFIBUS.

**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.
4.8.9 PROFINET IO and DS0 diagnostic interrupts

4.8.9.1 Diagnostic interrupt PROFINET IO maintenance concept

Description

A device or module of an automation system can essentially adopt 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.

![Maintenance state model](image)

**Figure 4-66** 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
4.8.9.2 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 4-67 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.

4.8.9.3 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 4-6 Table: DS0 byte 0**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Module Fault/OK</td>
<td>Group failure (diagnostics) for a submodule:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: No group failure (diagnostics) pending</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Group failure (diagnostics) pending</td>
</tr>
<tr>
<td>1</td>
<td>Internal error</td>
<td>Always 0</td>
</tr>
<tr>
<td>2</td>
<td>External error existent</td>
<td>Has the same content as bit 0</td>
</tr>
<tr>
<td>3</td>
<td>Channel error existent</td>
<td>Has the same content as bit 0</td>
</tr>
<tr>
<td>4</td>
<td>External auxiliary power missing</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Front connector missing</td>
<td>Always 0</td>
</tr>
<tr>
<td>6</td>
<td>Module not parameterized</td>
<td>Always 0</td>
</tr>
<tr>
<td>7</td>
<td>Wrong parameters in module</td>
<td>Always 0</td>
</tr>
</tbody>
</table>

**Table 4-7 Table: DS0 byte 1**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-3</td>
<td>Type class of module</td>
<td>3: Type class 3 is to be understood as a distributed I/O and as such also includes PROFINET IO.</td>
</tr>
<tr>
<td>4</td>
<td>Channel information existent</td>
<td>0: No readable channel information present</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>5</td>
<td>User information existent</td>
<td>0: No channel diagnostics or manufacturer-specific diagnostics available</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: At least one set of channel diagnostics and/or manufacturer-specific diagnostics is available</td>
</tr>
</tbody>
</table>
### 4.8 Diagnostic and alarm behavior

#### Table 4-8 Table: DS0 byte 2

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>Always 0</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>Always 0</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>Always 0</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>Always 0</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>Always 0</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>Always 0</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>Always 0</td>
</tr>
<tr>
<td>7</td>
<td>Maintenance Demanded</td>
<td>Group maintenance demand for a submodule:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: No group maintenance demand pending</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Group maintenance demand pending</td>
</tr>
</tbody>
</table>

#### Table 4-9 Table: DS0 byte 3

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>Always 0</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>Always 0</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>Always 0</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>Always 0</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>Always 0</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>Always 0</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>Always 0</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
<td>Always 0</td>
</tr>
</tbody>
</table>

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.
PROFINET IO

4.8 Diagnostic and alarm behavior
Ethernet: General information (TCP and UDP connections)

5.1 Ethernet interfaces

5.1.1 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.

5.1.2 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 8-pin RJ45 sockets with D4x5-2 DP/PN.

The SIMOTION devices do not have IP router functionality; they do not forward the message frames from one subnet to another.

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" 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 PROFINET modules, such as CBE30-2 of a SIMOTION D4x5-2 (100 Mbit/s).

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

Converting standard Ethernet interfaces from V4.3 and higher

With SIMOTION V4.3, the standard Ethernet interfaces are designed as PN-IE interfaces:

- Determining neighborhood information via LLDP (Link Layer Discovery Protocol). To do so, use the system function _getPnPrtNeighbour in SIMOTION.
- If a Step7 diagnostics address is available, you can perform diagnostics for the Ethernet interface from the user program.
- You can assign the device name (NameOfStation) either via DCP or via the user program (see also IP address and communication name (Page 133)).
You can assign the IP address either via DCP or via the user program. This prevents project details from being unintentionally overwritten.

Provision of topology information via SNMP (Simple Network Management Protocol) (e.g. via the STEP7 Topology Editor, see also Topology editor (graphical view) (Page 124)).

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 5-1 Ethernet interfaces in HW Config](image)

<table>
<thead>
<tr>
<th></th>
<th>Standard Ethernet interface with one port</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>PROFINET IO interface with three ports</td>
</tr>
</tbody>
</table>

See also

IP address and communication name (Page 133)
Topology editor (graphical view) (Page 124)

### 5.1.3 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 DIAG and SIMOTION IT OPC XML DA (no license required as of V4.2)

For communication by means of SIMOTION VM (separate license required)

5.2 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
5.3 TCP communication

5.3.1 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. There must be at least one client and one server when establishing a TCP connection.

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)
5.3 TCP communication

- 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)

![Diagram of TCP communication](image)

**Figure 5-2** Schematic diagram showing TCP communication sequence

**Principles of port assignment:**

- The port on the client can be the same as the port on the server.
- The port on the client can be different from the port on the server.
Data exchange between user program and TCP stack

![Diagram of data exchange](image)

**Data packets**

With TCP communication, the send data from the TCP stack is divided into subpackets of any size (up to 1,456 bytes per packet). This means that data packets of different sizes may arrive at the receiver side.

**The following scenarios are possible on the receiver side:**

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

The order of the data is maintained. Users must ensure that these subpackets are restored to the length of the sent data packet in their user program.

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 5-1 Useful data and control**

<table>
<thead>
<tr>
<th>Control</th>
<th>Buffer size in bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIMOTION</td>
<td>4.096</td>
</tr>
<tr>
<td>SIMATIC</td>
<td>8.192</td>
</tr>
</tbody>
</table>

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.
5.3.2 SIMOTION system functions for TCP communication

5.3.2.1 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>
<thead>
<tr>
<th>SIMOTION CPU (client)</th>
<th>Number of communication connections</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2xx</td>
<td>45</td>
</tr>
<tr>
<td>D410</td>
<td>45</td>
</tr>
<tr>
<td>D410-2</td>
<td>45</td>
</tr>
<tr>
<td>D4x5</td>
<td>75</td>
</tr>
<tr>
<td>D4x5-2</td>
<td>75</td>
</tr>
<tr>
<td>P3x0</td>
<td>40</td>
</tr>
</tbody>
</table>

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 196)
_tcpOpenClient() system function (Page 197)
_tcpSend() system function (Page 198)
_tcpReceive() system function (Page 199)
_tcpCloseConnection() system function (Page 200)
_tcpCloseServer() system function (Page 200)

### 5.3.2.2 _tcpOpenServer() system function

**Overview**

This system function is used in order to initiate passive establishment of a connection.

#### Table 5- 3 Call example

```c
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.
5.3.2.3 _tcpOpenClient() system function

Overview

The _tcpOpenClient() system function is used to actively establish a connection.

Table 5- 4 Call example

```c
sRetValTcpOpenClient := _TCPOpenClient( // StructRetTcpOpenClient
    port := 3456 // UINT, 1024-65535
    ,serverAddress := au8ServerAddress // ARRAY [0...3] OF UINT
    ,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.
_tcpSend() system function

Overview

The _tcpSend() system function is used for sending data.

Table 5- 5 Call example

```c
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().
5.3.2.5 _tcpReceive() system function

Overview

The _tcpReceive() system function is used for receiving data.

Table 5-6 Call example

```plaintext
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().
5.3.2.6 _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 5-7 Call example

```plaintext
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.

5.3.2.7 _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 5-8 Call example

```plaintext
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.
5.3.3 SIMATIC communication blocks, onboard Ethernet interface

5.3.3.1 Overview of SIMATIC communication blocks

Overview

The following function blocks are available for TCP communication with a SIMATIC and its onboard Ethernet interfaces.

- TSEND (FB63)
- TRCV (FB64)
- TCON (FB65)
- TDISCON (FB66)

To establish the connection, FB65 TCON requires the data structure UDT65 TCON_PAR. By way of a return value, the data structure receives a connection ID which must be transferred when the communication blocks are called. The FB66 TDISCON function block is used to close the connection. Useful data is sent and received using the FB63 TSEND and FB64 TRCV blocks.

Maximum number of connections

Table 5-9 Control types and max. number of connections

<table>
<thead>
<tr>
<th>SIMATIC</th>
<th>Max. number of connections</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU 315(F)-2 PN/DP</td>
<td>8</td>
</tr>
<tr>
<td>CPU 317(F)-2 PN/DP</td>
<td>8</td>
</tr>
<tr>
<td>CPU 319(F)-2 PN/DP</td>
<td>32</td>
</tr>
<tr>
<td>CPU 414-3 PN/DP</td>
<td>30</td>
</tr>
<tr>
<td>CPU 416(F)-3 PN/DP</td>
<td>62</td>
</tr>
<tr>
<td>IM151-8(F) PN/DP CPU</td>
<td>8</td>
</tr>
</tbody>
</table>

See also

Overview of SIMATIC communication blocks (Page 218)
5.3 TCP communication

5.3.3.2 UDT65 structure and parameterization

Overview

In the case of communication with a SIMATIC control and integrated Ethernet interface, the connection is not configured in NetPro. Instead, it is configured using the data structure UDT65 TCON_PAR in the user program (see figure below).

![Declaration view of UDT65 TCON_PAR in the LAD/STL/FBD editor](image-url)

The structure and parameterization of the UDT65 is described in detail below.

In principle, there are two different scenarios when it comes to parameterizing the UDT65 or the data block derived from this:

1. The connection is established actively from the S7 station (client).
2. The S7 station passively waits for the connection to be established from the communications partner (server).

You can find the most important parameters in the tabular overview (distinction made between client and server). Parameters that are not referred to are usually left at their default values or 0. For more information, please refer to the online help for the UDT65 data block.
### Table 5-10  Data block UDT65, parameterization with TCP/UDP

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value for TCP</th>
<th>Value for UDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>block_length</td>
<td>Length of a parameterization block</td>
<td>64 hex</td>
<td>64 hex</td>
</tr>
<tr>
<td>id</td>
<td>A separate number must be assigned for each connection via the id parameter (value range 1-0FFF hex). This reference is required for parameterizing the TCON, TSEND, TRCV, and TDISCON function blocks.</td>
<td>In the relevant block, you must specify the value of this parameter for ID.</td>
<td>In the relevant block, you must specify the value of this parameter for ID.</td>
</tr>
<tr>
<td>connection_type</td>
<td>Connection type</td>
<td>11 hex</td>
<td>13 hex</td>
</tr>
<tr>
<td>local_device_id</td>
<td>The local_device_id parameter is dependent upon the CPU type. You can obtain the values for it from the STEP 7 online help.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>local_tsap_id_len</td>
<td>Used length of the local_tsap_id parameter</td>
<td>2 hex</td>
<td>2 hex</td>
</tr>
<tr>
<td>rem_staddr_len</td>
<td>Length of the address for the remote connection end point</td>
<td>4 hex</td>
<td>0, not used</td>
</tr>
<tr>
<td>active_est</td>
<td>1: Active connection establishment</td>
<td>TRUE</td>
<td>FALSE</td>
</tr>
<tr>
<td></td>
<td>2: Passive connection establishment</td>
<td>FALSE</td>
<td>FALSE</td>
</tr>
</tbody>
</table>
### Ethernet: General information (TCP and UDP connections)

#### 5.3 TCP communication

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value for TCP</th>
<th>Value for UDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>rem_staddr</td>
<td>For 1 and 2, IP address of the communications partner, e.g. 192.168.0.1 (SIMOTION device).</td>
<td>rem_staddr[1] = B#16#C0 (192), rem_staddr[2] = B#16#A8 (168), rem_staddr[3] = B#16#00 (0), rem_staddr[4] = B#16#01 (1), rem_staddr[5-6] = B#16#00 (reserved)</td>
<td>0, not used</td>
</tr>
<tr>
<td>rem_tsap_id</td>
<td>For 1, remote port no. values apply to connection_type11hex.</td>
<td>B#16#11: rem_tsap_id[1] = high byte for port no. in hexadecimal representation, rem_tsap_id[2] = low byte for port no. in hexadecimal representation, rem_tsap_id[3-16] = B#16#00</td>
<td>0, not used</td>
</tr>
</tbody>
</table>

For 2

0, not used 0, not used

The next section contains descriptions of communication blocks.

**See also**

Overview of SIMATIC communication blocks (Page 218)

**5.3.3.3 Description of communication blocks**

**Function blocks for establishing a connection**

Below is a description of the function blocks that can be used to program the establishment of a connection.

**TCON (FB65)**

**Table 5-11 Call example**

```
CALL "TCON", DB66
REQ :=M1.0
ID :=W#16#1
DONE :=M2.0
```
If data is to be received or sent on an S7 station with integrated Ethernet interface, a connection must first be established between the S7 station and the communications partner via the TCON (FB65) function block.

Establishment of the connection is controlled via the REQ parameter. If the parameter is set to 1 and, therefore, an edge is created, the data (connection description) from the area specified under CONNECT is transferred to the function block in order to establish the connection.

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

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

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

### TSEND (FB63)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REQ</td>
<td>M5.0</td>
</tr>
<tr>
<td>ID</td>
<td>W#16#1</td>
</tr>
<tr>
<td>LEN</td>
<td>10</td>
</tr>
<tr>
<td>DONE</td>
<td>M6.0</td>
</tr>
<tr>
<td>BUSY</td>
<td>M7.0</td>
</tr>
<tr>
<td>ERROR</td>
<td>M8.0</td>
</tr>
<tr>
<td>STATUS</td>
<td>MW200</td>
</tr>
<tr>
<td>DATA</td>
<td>DB10.DB00</td>
</tr>
</tbody>
</table>

If a communication connection is established, this data can be sent. This is performed by calling the TSEND (FB63) function block.

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

The communication connection with which the data is to be sent is referenced via the ID parameter. The LEN parameter specifies the length of the data to be sent in bytes.
The parameters **DONE**, **BUSY**, and **ERROR** also specify the processing status of the function block in this case. In addition to the information that an error has occurred (ERROR = 1), the user also receives detailed information about the type of error via the **STATUS** parameter.

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

**TRCV (FB64)**

Table 5-13 Call example

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

Data can also be received using the TRCV (FB64) function block via an established connection.

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

The **ID** is used to select a specific communication connection to be used for receiving the data.

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

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

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

The meaning of the **RCVD_LEN** parameter has already been explained above. If the **LEN** parameter has been assigned the value 0, the number of data items contained in the data block received last is specified in the **RCVD_LEN** parameter. If a value other than 0 has been assigned in the **LEN** parameter, then the same value is present in **RCVD_LEN**.
The DATA parameter contains the address and length of the send area. The received data can be taken from here for further processing.

**FDISCON (FB66)**

Table 5-14  Call example

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

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

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

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

### 5.3.4 SIMATIC communication blocks for Ethernet CP

#### 5.3.4.1 Overview of SIMATIC communication blocks

**Overview**

The following communication blocks are available:

- AG_SEND (FC5)
- AG_RECV (FC6)
- AG_LSEND (FC50)
- AG_LRECV (FC60)
- AG_SSEND (FC53)
- AG_SRECV (FC63)
The section below only describes SIMATIC communication blocks AG_SEND and AG_RECV.

For more information, please refer to the *SIMATIC NET functions (FC) and function blocks (FB) for SIMATIC NET S7 CPs Programming Manual*.

**Note**

For the **CP 300**, the maximum send and receive length per function call (AG_SEND and AG_RECV) is 8,192 bytes.

The AG_SEND and AG_RECV functions can also be used for the **CP 400**. However, the transferable data length in this case is generally limited to <= 240 bytes per job.

### 5.3.4.2 Ethernet CP configuration

**Procedure**

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

![Figure 5-5 Selected S7 CPU and the associated connection table](image)

2. Double-clicking an empty line in the connection table opens the *Insert New Connection* dialog box for inserting a new communication connection. The dialog box is the same for TCP and UDP connections. With the CPU selected, the dialog box can also be opened via the context menu **Insert > New connection**... or by clicking the button in the menu bar.
3. In the **Insert New Connection** dialog box, select the connection partner.

**Note**

In the case of communication between an S7 CP and a SIMOTION device, select the setting "(unspecified)".

![Figure 5-6 "Insert New Connection" dialog box with selected TCP/IP connection](image)

4. The desired connection type (TCP connection) is selected in the Connection field.

**Note**

In the case of UDP communication, select UDP connection here.
5. If you exit the Insert New Connection dialog box with OK, a prompt appears to inform you that connections are also possible via subnets and that the router addresses may have to be checked. Once you have acknowledged this prompt, in the case of a TCP connection the Properties dialog box will open.

![Figure 5-7 “Properties - TCP Connection” dialog box - “Addresses” tab](image)

6. The IP address and the port for the local communications partner are already pre-assigned in the Addresses tab. The settings still have to be made for the remote communications partner. The IP address of the communications partner must be entered in the IP (DEZ) field. A port, specified by the user for the communications partner for this connection, must be entered in the Port (DEZ) field. Boundary conditions must be met for the port on the S7 side, i.e. a port between 2000 and 5000 must be selected on the S7 side.

**Note**

SIMOTION supports port 1024 to 65535.
7. In the **Block parameters** field on the **General** tab, the **ID** and **LADDR** parameters for the S7 communication blocks can be used. **ID** assigns the communication connection a unique reference. Additionally, the address of the CP is specified as **LADDR**. The **Active connection buildup** checkbox can be used to specify whether the connection is to be established from the S7 station. If active connection establishment is selected on the S7 side, the communications partner must be configured as a server. If active connection establishment is not selected on the S7 side, however, the communications partner must be configured as a client.

![Figure 5-8 "Properties - TCP Connection" dialog box - "General" tab](image)

8. If the Properties dialog box is now exited with **OK**, the **Insert New Connection** dialog box must also be closed (by clicking the **Close** button) to finish the connection configuration. If you wish to configure additional connections, you can do this by selecting the required connection type and then clicking the **Accept** button. Once you have finished configuring the communication connection, the parameters for establishing the connection are permanently defined with a communications partner.
5.3 TCP communication

5.3.4.3 Description of communication blocks

Overview

Below is a description of the communication blocks that can be used to program the establishment of a connection.

AG_SEND (FC5)

Table 5-15 Call example

CALL "AG_Send"
Act :=M0.0
ID :=1
LADDR :=#16#3FFC
SEND :=P#DB100.DBX0.0 BYTE 1000
LEN :=1000
DONE :=M0.1
ERROR :=M0.2
STATUS :=MW10

The ID and LADDR parameters are displayed when creating the connection in NetPro and must be transferred for this connection when the functions are called. Sending is triggered on a positive edge at the ACT input. The SEND parameter defines the send data and the length to be sent is specified via LEN. The parameters DONE, ERROR, and STATUS are used for diagnostics or to return the status of the send job.

AG_RECV (FC6)

Table 5-16 Call example

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

The parameters ID and LADDR are specified by NetPro when creating the connection in NetPro and must be transferred for this connection when the functions are called.
At the RECV parameter, a pointer is transferred; this indicates the data area in which the received data is stored. TRUE at the NDR output informs the user that new data has been received. The LEN parameter specifies how many bytes have been read out. The parameters ERROR and STATUS return a diagnosis or the status of the receive call.

Note
In the case of TCP communication, the NDR output is only set once the data area for the received data is full.

5.4 UDP communication

5.4.1 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.

![UDP Communication Model](image)

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 in good time, the UDP message frames received will be lost.

Table 5-17 Useful data and control

<table>
<thead>
<tr>
<th>Control</th>
<th>Buffered useful data in bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIMOTION</td>
<td>1,470 (SIMOTION V4.1 SP4 and higher)</td>
</tr>
<tr>
<td></td>
<td>1,400 (up to SIMOTION V4.1 SP4)</td>
</tr>
<tr>
<td>SIMATIC</td>
<td>Up to 2,048 (depending on CPU and communication via Ethernet CP or integrated Ethernet interface)</td>
</tr>
</tbody>
</table>
5.4.2 SIMOTION system functions for UDP communication

5.4.2.1 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 215)
_udpReceive() system function (Page 216)
_udpAddMulticastGroupMembership() system function (Page 217)
_udpDropMulticastGroupMembership() system function (Page 218)

5.4.2.2 _udpSend() system function

Overview

The _udpSend() system function is used for sending data.

Table 5-18  Call example

```plaintext
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.

### 5.4.2.3 _udpReceive() system function

**Overview**

The _udpReceive() system function is called for receiving data.

<table>
<thead>
<tr>
<th>Table 5- 19 Call example</th>
</tr>
</thead>
</table>

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.

5.4.2.4 _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 (_udpAddMulticastGroupMembership(), updDropMulticastGroupMembership()) can only be used on Ethernet interfaces. PROFINET interfaces are not supported.
5.4 UDP communication

5.4.2.5 _udpDropMulticastGroupMembership() system function

Overview

This system function is used to exit a multicast group on a selected Ethernet interface.

```plaintext
myRetDINT :=
    _udpDropMulticastGroupMembership{
        multicastIPAddress := ,
        interfaceIPAddress := ,
        multicastPort := ,
        nextCommand :=
    };
```

Note

The UDP multicast functions (_udpAddMulticastGroupMembership(), _udpDropMulticastGroupMembership()) can only be used on Ethernet interfaces. PROFINET interfaces are not supported.

5.4.3 SIMATIC communication blocks, onboard Ethernet interface

5.4.3.1 Overview of SIMATIC communication blocks

Overview

For UDP communication with a SIMATIC and its onboard Ethernet interfaces, the same communication blocks are available as for TCP communication. You can find an overview in the section titled Overview of SIMATIC communication blocks (Page 201).

The structure and parameterization of the UDT65 is described in detail below.

You can find a detailed explanation of parameterizing the UDT65 and the data block derived from this in the section titled UDT65 structure and parameterization (Page 202). The Value for UDP column contains the required settings for UDP configuration. The only feature that does not exist with UDP is the distinction between active and passive connection establishment. Therefore, the active_est parameter is set to FALSE.
See also

Description of communication blocks (Page 204)

5.4.4 SIMATIC communication blocks for Ethernet CP

5.4.4.1 Overview of S7 communication blocks, UDP

Overview

The functions used for this application are the same as the S7 functions that have already been described (see Overview of SIMATIC communication blocks (Page 207)).

The sequence of configuration steps for UDP communication is similar to the one for TCP communication. UDP-specific parameters only need to be selected when configuring a connection in NetPro. Additionally, there is no need to define active or passive connection establishment.

Note

For the CP 300, the maximum send and receive length per function call (AG_SEND and AG_RECV) is 2,048 bytes.

The AG_SEND and AG_RECV functions can also be used for the CP 400. However, the transferable data length in this case is generally limited to <= 240 bytes per job.
5.4.4.2 Ethernet CP configuration

Specific configuration settings for UDP communication

The configuration process is similar to the one for TCP. The following section only describes the differences between the two configuration processes; therefore, please make sure you take note of the one for TCP too (see Ethernet CP configuration (Page 208)).

1. The desired connection type **UDP connection** is selected in the Connection field.

![Insert New Connection dialog box with selected UDP connection](image)

2. In the **Block parameters** field on the **General** tab, the ID and LADDR parameters for the S7 communication blocks can be used. ID assigns the communication connection a unique reference. Additionally, the address of the CP is specified as "LADDR".
Figure 5-11  "Properties - UDP Connection" dialog box - "General" tab
5.4 UDP communication
Routing - communication across network boundaries

6.1 What does routing mean?

Routing is the transfer of information from Network x to Network y.

There is a fundamental difference between intelligent, self-learning routing (e.g. IP routing in the Internet) and routing according to previously specified routing tables (e.g. S7 routing).

IP routing

IP routing is a self-learning routing procedure (which can also be performed manually), used exclusively in Ethernet communication networks which operate with the IP protocol, such as the Internet.

The function is performed by special routers that pass on the information to adjacent networks based on the IP address, when the IP address is not detected in the own network.

Note

IP routing is NOT supported by SIMOTION. It is possible to have S7 routing between the Ethernet interfaces.

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...
6.2 Configuration of S7 routing

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.

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.

6.2 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.

6.3 Routing for SIMOTION

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.

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 routes) SINAMICS drives on the cascaded, lower-level DP master interface of the SIMOTION 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:
6.4 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.

- Routing from subnet to subnet (IP routing) is not supported. You can use an external IP router for this
- The S7 routing from a PROFINET/Ethernet subnet to a PROFIBUS is possible.

There are three options for connecting a PG/PC or HMI via S7 routing to a SIMOTION D with CBE30.

Note
The Ethernet interfaces X120 and X130 and the CBE30 must be in different subnets for S7 routing.
Engineering system to PROFINET (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

Figure 6-2  Example for PG/PC to CBE30
Engineering system / HMI to PROFIBUS

- S7 routing to the other (master) PROFIBUS interfaces (only if configured)
- S7 routing to PROFIBUS Integrated
- S7 routing to X1400 on the CBE30
- S7 routing to the standard Ethernet interfaces (X120, X130) (V4.1.2 and higher)
- Access to nodes on the same network, e.g. HMI
Engineering system / HMI to Ethernet

![Diagram of SIMOTION D and HMI connected to Ethernet]

- S7 routing to the other (master) PROFIBUS interfaces (only if configured)
- S7 routing to PROFIBUS Integrated
- S7 routing to X1400 on the CBE30
- S7 routing between the Ethernet interfaces
- Access to nodes on the same network, e.g. HMI

6.5 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.

- Routing from subnet to subnet (IP routing) is not supported. You can use an external IP router for this
- S7 routing from a PROFINET/Ethernet subnet to PROFIBUS is possible.

There are therefore the following options for connecting a PG/PC or HMI device to a SIMOTION D using S7 routing.
Routing - communication across network boundaries

6.5 Routing with SIMOTION D4x5-2 (example of D455-2 DP/PN)

Engineering system / HMI to PROFINET

- 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

Figure 6-5 Example of PG/PC to PROFINET interface (PNxIO, X150)
Routing - communication across network boundaries

6.5 Routing with SIMOTION D4x5-2 (example of D455-2 DP/PN)

Engineering system / HMI to PROFIBUS

- S7 routing to the other (master) PROFIBUS interfaces (only if configured)
- S7 routing to the PROFIBUS Integrated
- 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 6-7 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 onboard PROFINET IO interface (X150, P1-P3)
- S7 routing between the Ethernet interfaces

6.6 Routing for SIMOTION D to the SINAMICS integrated

S7 routing to the internal PROFIBUS on SINAMICS Integrated

All SIMOTION D have an integrated SINAMICS drive control. In order to be able to access drive parameters, the telegrams must be routed from the external SIMOTION D interfaces to the internal PROFIBUS DP. S7 routing can be used to access the integrated PROFIBUS. Here, the internal PROFIBUS DP forms a separate subnet. This must be especially taken into account for the communication to several routing nodes.
6.7 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

![Diagram of routing from PROFIBUS to PROFINET]

Figure 6-8 Example for P350 routing from PROFIBUS to PROFINET
Routing from PROFINET on PROFIBUS

<table>
<thead>
<tr>
<th>Ethernet(1)</th>
<th>Industrial Ethernet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethernet(2)</td>
<td>Industrial Ethernet</td>
</tr>
<tr>
<td>MPI(1)</td>
<td>MPI</td>
</tr>
<tr>
<td>PROFIBUS(1)</td>
<td>PROFIBUS</td>
</tr>
</tbody>
</table>

Figure 6-9  Example for P350 routing from PROFINET to PROFIBUS
6.8 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

![Routing Diagram](Figure 6-10 Routing for SIMOTION P320)
Routing - communication across network boundaries

6.8 Routing for SIMOTION P320
**SIMOTION IT**

**7.1 SIMOTION IT - overview**

**Description**

SIMOTION IT offers the option of accessing the SIMOTION control via standard web services (HTTP).

**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
SIMOTION IT
7.1 SIMOTION IT - overview

- No version conflict between client application and SIMOTION RT (runtime)
- Series commissioning without engineering system

![SIMOTION IT overview diagram](image)

SIMOTION IT consists of the following function packages:
- SIMOTION IT DIAG
- 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.

You will find further information on SIMOTION VM in the *SIMOTION – Jamaica Diagnostics Manual*.

See also
Web access to SIMOTION (Page 239)
SIMOTION IT DIAG (Page 240)
SIMOTION IT OPC XML DA (Page 242)
7.2 Web access to SIMOTION

Description

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

![Access to SIMOTION Diagram]

Figure 7-2 Access to SIMOTION

See also

SIMOTION IT DIAG (Page 240)
SIMOTION IT OPC XML DA (Page 242)
7.3 SIMOTION IT DIAG

Description

SIMOTION IT DIAG allows a PC to use any Internet browser to access the HTML pages in SIMOTION.

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 IT DIAG 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 IT DIAG pages on devices such as cell phones or PDAs, a set of special pages is provided for version 4.1.3 and higher. These contain a simplified representation of information from the standard pages. 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 IT DIAG:

- **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 IT DIAG settings.
User-defined pages

SIMOTION IT DIAG 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. There are currently four variable providers.

- MiniWeb
- SIMOTION
- SIMOTION diagnostics
- UserConfig

This makes it possible to access 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 SIMOTION IT DIAG.

7.4 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 device. As of V4.2, a separate license is no longer required for SIMOTION IT OPC XML DA.

OPC is the abbreviation for "OLE for Process Control" and describes a standard interface for communication in automation systems. 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
  - Cyclic reading 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.

![Diagram of OPC XML DA server access]

Figure 7-3 Access to the OPC XML DA server

### 7.5 FTP data transfer

#### File access using FTP

You can use the FTP server of the SIMOTION control to access the SIMOTION file system specifically. FTP is protected through access protection.

You can use FTP, for example, to perform firmware updates or load user-defined pages.

The FTP service does not require its own license.
7.5 FTP data transfer
8.1 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.

Figure 8-1 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.

8.2 Message frames and signals in drive-based safety

Description

Since SIMOTION does not have a safe logic function, it cannot attend to the F control process. Instead, this is carried out using a second controller featuring F functionality (usually a SIMATIC F-CPU). The image below shows how this setup works, using the example of a SINAMICS axis. A PROFIsafe-secured communication channel runs between the DO drive and the SIMATIC F-CPU. Standard message frame 30, consisting of a safety control word and status word (among other things), is normally available for this communication channel. With V4.3 and higher, there are a total of three PROFIsafe message frames available:

- Message frame 30
- Message frame 31
- Message frame 901

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.
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 information channel is not secured and is implemented in practice by extending the standard message frame to include the SIDB data block. 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 98)
8.3 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 8-3  Routing the PROFIsafe channel with F proxy functionality
8.4 PROFIsafe properties for configuration

Mechanisms with PROFIsafe

Basically the PROFIsafe can communicate both via PROFINET and PROFIBUS. F-Proxy mechanisms and/or a data exchange broadcast variant are available for this purpose.

For all the mechanisms, the F data is managed by the F-CPU and the Motion Control data by the SIMOTION CPU. The PROFIsafe properties required are set in the HW Config for the failsafe 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. However, mixed operation (Motion Control via PROFINET and PROFIsafe via PROFIBUS) is only possible with integrated SIMOTION D drives. It is not permissible if you are using external drives (e.g. S120 CU320-2) on a SIMOTION CPU.
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.

![Properties of PROFIsafe taking the example of I device failsafe proxy](image)

Figure 8-4  Properties of PROFIsafe taking the example of I device failsafe 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 message 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.
8.5 PROFIsafe via PROFINET

8.5.1 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.

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.
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.
8.5.2 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

---

Figure 8-6 Master project with a sub-project as the I device failsafe proxy

See also

Use of Safety with MRRT (Page 96)
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 failsafe proxy functionality
- SIMOTION D4xx, D4x5-2
- SIMOTION C240 PN
- SIMOTION P350-3, P320-3
- F-CPU V3.2 or higher

Table 8-1 The table shows which SIMOTION devices support the iDevice F-Proxy and which SINAMICS devices can be accessed via the iDevice F-Proxy.

<table>
<thead>
<tr>
<th>Overview of devices capable of I device failsafe proxy functions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SIMOTION IO Controller V4.2</strong></td>
</tr>
<tr>
<td>Controller Based</td>
</tr>
<tr>
<td>PC Based</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Drive-based (blocksize)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Drive-based (booksize)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SINAMICS IO devices and DP slaves</th>
</tr>
</thead>
<tbody>
<tr>
<td>S120 CX32</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>S120</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>S110</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>G120</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
8.5.3 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 and 901 are supported.

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.
8.5.4 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 96)
8.5.5 Topology overviews I device F-Proxy

8.5.5.1 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.

![Diagram of topology]

Figure 8-7 Topology for I device failsafe proxy for PROFIBUS drive units
8.5.5.2 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.

![Topology Diagram](image-url)

Figure 8-8 Topology for I device failsafe proxy for PROFINET drive units
8.5.5.3 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.

![Topology diagram](image)

Figure 8-9 Topology for I device failsafe proxy for PROFIBUS and PROFINET drive units
8.5.6 Configuring I device failsafe proxy

8.5.6.1 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
Configure with 1 or 2 projects

Project 1 should contain F host. Project 2 should contain SIMOTION CPU.

Project contains F host and SIMOTION CPU.

Create overview diagram of failsafe addresses (F_Dest_Add and F_Source_Add) for all I device F-Prosxs. These must be unique throughout the project. In HW Config, you can define a start address for the F addresses (e.g. 2000 or 4000) on the F-Proxy tab of the Properties dialog box for the CPU.

Configure SIMOTION CPU (C240, D4xx, or P3xx) and then PN IO controller if necessary.

Configure F I/O (DP slaves and PN IO devices) on the SIMOTION CPU. SINAMICS, e.g. CU320 and CX32, may also be part of the F I/O.

Configure drive objects for DP slaves/PN IO devices using SIMOTION SCOUT.

Configure message frames for the drive objects as well as PROFIsafe message frames using SIMOTION SCOUT. Align with HW Config. This creates the PROFIsafe modules in STEP7.

Configure F parameters for the PROFIsafe modules.

In HW Config, insert the PN IO controller (CBE30, MCI-PN BOARD) in the SIMOTION CPU if necessary. This enables the access point to be configured for F.

Configure the F I device interface and F-Proxy in the PN IO controller.

Create F I device, export as GSDML. The exported I device is displayed in the HW Config catalog under "Preconfigured Stations".

Insert F host (SIMATIC F-CPU).

Import F I-device and set F parameters at PN IO system of F host.

Carry out PROFIsafe configuration of drives in SCOUT (Safety configuration).

F operation.

Figure 8-10 Basic configuration process for I device failsafe proxy
8.5.6.2 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 with a SINAMICS S120 CU320 PN, which is connected to the SIMOTION via the PN interface (CBE30). 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 Section Configuring PROFINET IO with SIMOTION (Page 98).

3. Add a SIMOTION D435 V4.2. 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 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 PN (V2.6 or higher) 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 message frame (use symbolic assignment) is created automatically.

9. Save and compile the project.

10. After configuration, you have to select the PROFIsafe message frame. In the SIMOTION SCOUT project navigator, under the drive unit double-click on "Drive unit_xx" - Communication > Message frame configuration. The message frames appear in the working area.
11. Mark the appropriate drive in the tab **IF1: PROFIdrive PZD message frame** of the message frame overview and in the bottom part of the window under **Adapt message frame configuration** select the entry **Add PROFIsafe**. The PROFIsafe message frame is added. Dependent on the drive, you can select from several message frames.

![Figure 8-11 Adding PROFIsafe message frame](image)

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 I device 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.

![Figure 8-12 Properties of I device transfer area](image)
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. All you need do now is save and compile the station.

![Figure 8-13 Selecting PROFIsafe channel of I/O](image)

Figure 8-13 Selecting PROFIsafe channel of I/O
18. 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 failsafe 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 failsafe parameters, see PROFIsafe properties for configuration (Page 249)

---

![Figure 8-14 Setting failsafe address (F_dest_Add)](image)
19. The fail-safe address 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 8-15 Entering F_Dest_Add as PROFIsafe address](image)

20. 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.

21. 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.

22. In the menu, select e.g. **Insert > Station > SIMATIC 300 Station**. Double-click on the station and then the entry **Hardware**. HW Config opens.

23. From the hardware catalog, insert e.g. an S7 300 rack if you want to select an F-CPU from the S7 300 series.
24. Insert the F-CPU, e.g. CPU317-2 PN/DP. This must at least be version V3.2.

25. 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.

Figure 8-16 Master project with an I device failsafe proxy submodule

### 8.5.6.3 Adapting the F address in the existing project

Adapting the F address (F_Dest_Add) for the entire project

In an existing project, you can check whether the F address for the I device F-Proxy has been set correctly at a later point. The F address 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 address at a later point without having to create and install the I device again, you must make the change at the three locations referred to above.
In the example below, you set the F address 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 address 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.

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 8-18  F address for I device on F-CPU](image)

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 8-19 F address for Safety Integrated configuration](image)

12. Confirm by selecting **Close** and save and compile the project.

### 8.5.6.4 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 message frame (see points 7-11 in the example Configuration example for SIMOTION D435 and SINAMICS S120 via PROFINET (Page 263)).

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 message frame** 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 which opens, 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. Click on the Select I/O button and select the corresponding PROFIsafe channel in the dialog which opens.

Figure 8-20  Releasing PROFIsafe module for SINAMICS_Integrated for transfer area
5. Click **OK** to confirm the dialog box. In the Properties transfer area 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 8-21 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 do now is save and compile the station.

![Device transfer area](image)

**Figure 8-22** 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 SIMOTION module's I device. The diagram shows the project with F-CPU and I device failsafe proxy with one drive on PROFINET and one drive on SINAMICS_Integrated on a D435.

---

**8.5.6.5 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.

![Properties of PROFIsafe taking the example of I device failsafe proxy](image.png)

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.
8.5.6.6 General information on failsafe addresses with 1 device F-Proxy

Communication addresses for failsafe source and/or F_Destination-Address

Create new failsafe hosts, failsafe modules/submodules in HW Config. HW Config then suggests the default failsafe source/destination address. 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 8-25 PROFIsafe start address

The F_Source_Address is assigned using the same process as for Siemens failsafe 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 is issued 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.
8.5 PROFIsafe via PROFINET

8.5.7 Shared device via PROFINET

8.5.7.1 General information on shared device

Description

With the new 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.
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

8.5.7.2 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.
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.

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).

4. Save and compile the station and close it.
5. Repeat steps 1 to 4 for the shared device on the SIMATIC F-CPU.

![Properties - IM151-3PN](image)

Figure 8-30 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.
8.6 PROFIsafe via PROFINET

8.6.1 General information about PROFIsafe on PROFINET

Two possibilities for PROFIsafe

There are basically two possible forms of PROFIsafe communication on PROFINET:

- **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.

8.6.2 Supported devices and software requirements for PROFIsafe on PROFINET

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 8-2  Device overview**

<table>
<thead>
<tr>
<th>SIMOTION CPU</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller Based</td>
<td>C240</td>
</tr>
<tr>
<td></td>
<td>C240 PN</td>
</tr>
<tr>
<td>PC Based</td>
<td>P350-3</td>
</tr>
<tr>
<td>Drive-based (blocksize)</td>
<td>D410 DP</td>
</tr>
<tr>
<td></td>
<td>D410-2 DP</td>
</tr>
<tr>
<td></td>
<td>D410-2 DP/PN</td>
</tr>
<tr>
<td>Drive-based (booksize)</td>
<td>D4x5</td>
</tr>
<tr>
<td></td>
<td>D4x5-1</td>
</tr>
<tr>
<td></td>
<td>D4x5-2 DP</td>
</tr>
<tr>
<td></td>
<td>D4x5-2 DP/PN</td>
</tr>
<tr>
<td>SINAMICS drive units</td>
<td></td>
</tr>
<tr>
<td>S120 CX32</td>
<td>CX32</td>
</tr>
<tr>
<td></td>
<td>CX32-2</td>
</tr>
<tr>
<td>S120</td>
<td>CU320 DP</td>
</tr>
<tr>
<td></td>
<td>CU320-2 DP</td>
</tr>
<tr>
<td></td>
<td>CU310 DP</td>
</tr>
<tr>
<td></td>
<td>CU310-2 DP</td>
</tr>
<tr>
<td>S110</td>
<td>CU305 DP</td>
</tr>
</tbody>
</table>

**Number of drive axes supported**

With PROFIBUS, only 16 drive axes can be used per PROFIBUS interface.

### 8.6.3  I-slave failsafe proxy

#### 8.6.3.1  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.
8.6.3.2 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.

![Diagram](image)

Figure 8-31 Topology for I-slave failsafe proxy for PROFIBUS drive units

8.6.3.3 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 message frame will automatically be created (symbolic assignment).
5. Save and compile the project.
6. Create a PROFlsafe 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 PROFlsafe. Click on the Adapt message frame configuration button and select Add PROFlsafe.

7. Save and compile the project.

8. Transfer the new PROFlsafe 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 8-34 Master-slave coupling in PROFIsafe](image-url)
11. Open the SIMOTION CPU project in HW Config.

12. Double-click on the icon of the SINAMICS drive unit and select the Details tab in the Configuration tab.
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 failsafe parameters, see PROFIsafe properties for configuration (Page 249)

![PROFIsafe properties](image)

**Figure 8-37** 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.
8.6.4 Failsafe data exchange broadcast

8.6.4.1 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.

8.6.4.2 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.

![Topology of failsafe data exchange broadcast for PROFIBUS drive units](image-url)
8.6.4.3 **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 fail-safe 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**.

![Image](image.png)

**Figure 8-39** 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 8-40 Properties of failsafe configuration for data exchange broadcast
11. Open the SIMOTION CPU project in HW Config.

Figure 8-41 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 249).

![Figure 8-42 PROFIsafe properties failsafe data exchange broadcast mode V1](image)

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.
8.7 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.

8.8 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*
PROFIdrive

9.1 Why profiles?

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

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

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

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

PROFIdrive

The PROFIdrive profile is a specific application profile. It contains a detailed description of
how the communication functions "data exchange broadcast", "equidistance" and
"isochronous operation" are used appropriately in drive applications. In addition, it specifies
all device characteristics which influence interfaces connected to a controller over
PROFIBUS or PROFINET. This also includes the State machine (sequential control), the
encoder interface, the normalization of values, the definition of standard 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 master have no or very little
effect on the drive interface.
9.2 PROFIdrive overview

The PROFIdrive Profile

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

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

- Base model
- Parameter model
- Application model

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

- PROFIdrive to PROFIBUS
- PROFIdrive to PROFINET

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

Literature note

PROFIdrive profile

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

Standards

IEC 61800 standard
9.3 PROFIdrive base/parameter model

Description

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

<table>
<thead>
<tr>
<th>PROFIdrive</th>
<th>PROFIBUS DP</th>
<th>PROFINET IO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller (higher-level control or host of the automation system)</td>
<td>Class 1 DP master</td>
<td>IO controller</td>
</tr>
<tr>
<td>Peripheral device (P device)</td>
<td>DP slave (I slaves)</td>
<td>IO device</td>
</tr>
<tr>
<td>Supervisor (engineering station)</td>
<td>Class 2 DP master</td>
<td>IO Supervisor</td>
</tr>
</tbody>
</table>

PROFIdrive device classes
Example of a PROFIdrive automation concept

The graphic below shows a typical automation concept.

Figure 9-1  Automation concept
Communication services

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

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

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

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

The graphic below shows the data model and data flow in the P device.
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 PROFINET function block makes a distinction between profile parameters and manufacturer-specific parameters:

- Profile parameters are defined for objects derived from the device model of the PROFINET function block. These may include general functions such as drive identification, fault buffer, or drive control, for example. These parameters are the same for all drives.
- All other parameters are manufacturer-specific. The parameters are defined by the interface for the application process, rather than by the profile.
Access to a parameter's elements (values, parameter descriptions, text elements) essentially works on an acyclic basis (with the exception of G120 drives, which can exchange data cyclically with PIV). An independent request/response data structure is defined for this purpose.

9.4 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>
<thead>
<tr>
<th>Category</th>
<th>Drive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category 1</td>
<td>Standard drives (such as pumps, fans, agitators, etc.); implemented in SIMOTION and SINAMICS</td>
</tr>
<tr>
<td>Category 2</td>
<td>Standard drives with technology functions</td>
</tr>
<tr>
<td>Category 3</td>
<td>Positioning drives; implemented in SIMOTION and SINAMICS</td>
</tr>
<tr>
<td>Category 4</td>
<td>Motion control drives with central, higher-level motion control intelligence and the &quot;Dynamic Servo Control&quot; position control concept; implemented in SIMOTION and SINAMICS</td>
</tr>
<tr>
<td>Category 5</td>
<td>Motion control drives with central, higher-level motion control intelligence and position setpoint interface</td>
</tr>
<tr>
<td>Category 6</td>
<td>Motion control drives with distributed, motion control intelligence integrated in the drives</td>
</tr>
</tbody>
</table>

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

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

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

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

Note

Jobs involving Base Mode Parameter Access are coded in a way you may already know from data set 47 (DPV1 communication from PROFIBUS). For any differences, please refer to Specifications for PROFIBUS and PROFINET IO (Page 320).
PROFIdrive uses DP V0, DP V1, and the DP V2 expansions for PROFIBUS, and the slave data exchange broadcast and isochronous operation functions contained within them as the communication protocol.

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

**Utilization categories**

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

The DSC (Dynamic Servo Control) function significantly improves the dynamic response and stiffness of the position control loop. With SIMOTION, this improvement usually relates to the dead times which occur for speed setpoint interfaces (transmission time, computing time for the controller and device), which are minimized by an additional, relatively simple feedback network in the drive. The position controller is pre-controlled in the drive by the SIMOTION controller using precontrol and position deviation, which enables very fast position control cycle clocks (e.g. 125 μs for servo in SINAMICS S), thereby restricting dead times to the control behavior alone.
9.5 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 S7 communication or communication with SINAMICS, you will have to use the AnyType_to_BigByteArray or BigByteArray_to_AnyType system functions to perform a type conversion for different data types (normalized value N2, N4; normalized value X2, X4; fixed-point value E2 and time constants T2 and T4).

PROFIdrive profile-specific data types

<table>
<thead>
<tr>
<th>Data types used in the PROFIdrive profile</th>
<th>Definition</th>
<th>Coding (dec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Boolean (IEC 61158-5)</td>
<td>1</td>
</tr>
<tr>
<td>Integer8</td>
<td>Integer8 (IEC 61158-5)</td>
<td>2</td>
</tr>
<tr>
<td>Integer16</td>
<td>Integer16 (IEC 61158-5)</td>
<td>3</td>
</tr>
<tr>
<td>Integer32</td>
<td>Integer32 (IEC 61158-5)</td>
<td>4</td>
</tr>
<tr>
<td>Unsigned8</td>
<td>Unsigned8 (IEC 61158-5)</td>
<td>5</td>
</tr>
<tr>
<td>Unsigned16</td>
<td>Unsigned16 (IEC 61158-5)</td>
<td>6</td>
</tr>
<tr>
<td>Unsigned32</td>
<td>Unsigned32 (IEC 61158-5)</td>
<td>7</td>
</tr>
<tr>
<td>FloatingPoint32</td>
<td>Float32 (IEC 61158-5)</td>
<td>8</td>
</tr>
<tr>
<td>FloatingPoint64</td>
<td>Float64 (IEC 61158-5)</td>
<td>15</td>
</tr>
<tr>
<td>VisibleString</td>
<td>VisibleString (IEC 61158-5)</td>
<td>9</td>
</tr>
<tr>
<td>OctetString</td>
<td>OctetString (IEC 61158-5)</td>
<td>10</td>
</tr>
<tr>
<td>TimeOfDay (with date indication)</td>
<td>TimeOfDay (IEC 61158-5)</td>
<td>11</td>
</tr>
<tr>
<td>TimeDifference</td>
<td>TimeDifference (IEC 61158-5)</td>
<td>12</td>
</tr>
<tr>
<td>Date</td>
<td>Date (IEC 61158-5)</td>
<td>13</td>
</tr>
<tr>
<td>TimeOfDay (without data indication)</td>
<td>TimeOfDay (IEC 61158-5)</td>
<td>52</td>
</tr>
<tr>
<td>TimeDifference (with data indication)</td>
<td>TimeDifference (IEC 61158-5)</td>
<td>53</td>
</tr>
<tr>
<td>TimeDifference (without data indication)</td>
<td>TimeDifference (IEC 61158-5)</td>
<td>54</td>
</tr>
</tbody>
</table>

Specific data types

See below for description
## 9.5 PROFIdrive-specific data types

### Data types used in the PROFIdrive profile

<table>
<thead>
<tr>
<th>Data type</th>
<th>Definition</th>
<th>Coding (dec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>N2 (normalized value (16-bit))</td>
<td></td>
<td>113</td>
</tr>
<tr>
<td>N4 (normalized value (32-bit))</td>
<td></td>
<td>114</td>
</tr>
<tr>
<td>V2 bit sequence</td>
<td></td>
<td>115</td>
</tr>
<tr>
<td>L2 nibble</td>
<td></td>
<td>116</td>
</tr>
<tr>
<td>R2 reciprocal time constant</td>
<td></td>
<td>117</td>
</tr>
<tr>
<td>T2 time constant (16-bit)</td>
<td></td>
<td>118</td>
</tr>
<tr>
<td>T4 time constant (32-bit)</td>
<td></td>
<td>119</td>
</tr>
<tr>
<td>D2 time constant</td>
<td></td>
<td>120</td>
</tr>
<tr>
<td>E2 fixed-point value (16-bit)</td>
<td></td>
<td>121</td>
</tr>
<tr>
<td>C4 fixed-point value (32-bit)</td>
<td></td>
<td>122</td>
</tr>
<tr>
<td>X2 normalized value, variable (16-bit)</td>
<td></td>
<td>123</td>
</tr>
<tr>
<td>X4 normalized value, variable (32-bit)</td>
<td></td>
<td>124</td>
</tr>
</tbody>
</table>

### Normalized value N2, N4

Linear normalized value, 0% corresponds to 0 (0x0), 100% corresponds to 2\(^{12}\) (0x4000) for N2, or 2\(^{28}\) (0x40,000,000) for N4. The length is 2 or 4 octets.

**Coding**

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

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

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

Linear normalized value, 0% corresponds to 0 (0x0), 100% corresponds to 2\(^{4}\). 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
### Fixed-point value E2

Linear fixed-point value with four places after the decimal point. 0 corresponds to 0 (0x0), 128 corresponds to 2\(^4\) (0x4000). 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

<table>
<thead>
<tr>
<th>Range of values E2</th>
<th>Resolution</th>
<th>Cod. (dec.)</th>
<th>Octet</th>
<th>Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>-256 + 2(^7) ≤ i ≤ 256 - 2(^7)</td>
<td>2(^7) = 0.0078125</td>
<td>121</td>
<td>6 5 4 3 2 1</td>
<td>7 6 5 4 3 2 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SN</th>
<th>2(^7)</th>
<th>2(^6)</th>
<th>2(^5)</th>
<th>2(^4)</th>
<th>2(^3)</th>
<th>2(^2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.007</td>
<td>0.007</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1.000</td>
<td>0.007</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 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\(^4\) (0x0000 0001).

**Coding**

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

<table>
<thead>
<tr>
<th>Range of values</th>
<th>Resolution</th>
<th>Coding (dec.)</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>-214,748.3648 ≤ i ≤ 214,748.3648</td>
<td>10(^{-4}) = 00001</td>
<td>122</td>
<td>4 octets</td>
</tr>
</tbody>
</table>

### Bit sequence V2

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

<table>
<thead>
<tr>
<th>Range of values</th>
<th>Resolution</th>
<th>Cod. (dec.)</th>
<th>Octet</th>
<th>Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>115</td>
<td>6 5 4 3 2 1</td>
<td>8 7 6 5 4 3 2 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>15 14 13 12 11 10 9 8</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
</tbody>
</table>
PROFIdrive

9.5 PROFIdrive-specific data types

Nibble (half-byte) L2

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

**Coding**

<table>
<thead>
<tr>
<th>Range of values</th>
<th>Resolution</th>
<th>Cod. (dec.)</th>
<th>Octet</th>
<th>Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>116</td>
<td>1</td>
<td>Nibble 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>Nibble 1</td>
</tr>
</tbody>
</table>

**Time constants T2 and T4**

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

**Coding**

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

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

<table>
<thead>
<tr>
<th>Range of values</th>
<th>Resolution</th>
<th>Coding (dec.)</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0 \leq i \leq 32,767 * T_a$</td>
<td>$T_a$</td>
<td>118</td>
<td>2 octets</td>
</tr>
<tr>
<td>$0 \leq i \leq 4,294,967,295 * T_a$</td>
<td>$T_a$</td>
<td>119</td>
<td>4 octets</td>
</tr>
</tbody>
</table>

**Time constant D2**

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

**Coding**

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

<table>
<thead>
<tr>
<th>Range of values</th>
<th>Resolution</th>
<th>Coding (dec.)</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0 \leq i \leq (2-2-14) * T_a$</td>
<td>$T_a$</td>
<td>120</td>
<td>2 octets</td>
</tr>
</tbody>
</table>

**Time constant R2**

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

**Coding**

- T2: As with Unsigned16, with a restricted range of values of $0 \leq x \leq 16,384$. When interpreted, internal values that fall outside this range of values are set to 16,384.
9.6 Acyclic communication (Base Mode Parameter Access)

### 9.6.1 Acyclic communication

**Description**

PROFIdrive drive devices are supplied with control signals and setpoints by the controller and return status signals and actual values. These signals are normally transferred cyclically (i.e. continuously) between the controller and the drive.

In addition, PROFIdrive drive units recognize parameters that contain other required data, such as error codes, warnings, controller parameters, motor data. This data is normally not transferred cyclically (namely, continuous), but "acyclically" 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 320).

---

<table>
<thead>
<tr>
<th>Range of values</th>
<th>Resolution</th>
<th>Coding (dec.)</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1 \cdot T_a \leq i \leq 16,384 \cdot T_a$</td>
<td>$T_a$</td>
<td>117</td>
<td>2 octets</td>
</tr>
</tbody>
</table>

---

**Note**

Further data types:

- Standard PROFIBUS/PROFINET data types (only available in English) (Page 353)
- Profile-specific PROFIBUS/PROFINET data types (only available in English) (Page 364)

---

**See also**

- Parameter request/response data set (Page 316)
This service is defined and provided by PROFIdrive, and it can be used in parallel to the cyclic communication on the relevant bus. The PROFIdrive profile specifies precisely how this basic mechanism is used for write access to parameters of a PROFIdrive-compliant drive.

**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.

### 9.6.2 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 two elements:

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

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

A "Write data record" is used to transfer the parameter job (for example, read parameter x). A "Read data record" is used to fetch the response for this parameter job (value of parameter x).
The figure **Reading and writing acyclically** shows that both "Write data set" and "Read data set" consist of the following elements:

- Request
- Response

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

### Writing parameter records

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

**Note**

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

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

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

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

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

<table>
<thead>
<tr>
<th>WORD</th>
<th>High Byte (Byte 1)</th>
<th>Low Byte (Byte 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOUBLE WORD</td>
<td>High Word</td>
<td>High Byte (Byte 1)</td>
</tr>
<tr>
<td></td>
<td>Low Word</td>
<td>High Byte (Byte 3)</td>
</tr>
</tbody>
</table>

WORD and DWORD representation in Big Endian format

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

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

See also

Rule 5 - a maximum of eight concurrent calls is possible in SIMOTION (Page 333)
Parameter request/response data set (Page 316)

9.6.3 Parameter request/response data set

Structure of the P request/response data set

This always consists of:

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

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

<table>
<thead>
<tr>
<th>Job parameters</th>
<th>Byte n+1</th>
<th>Byte</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>Request Header</td>
<td>RequestID</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Request Reference</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Axis</td>
<td>Number of parameters</td>
<td>2</td>
</tr>
</tbody>
</table>
### 9.6 Acyclic communication (Base Mode Parameter Access)

<table>
<thead>
<tr>
<th>Job parameters</th>
<th>Byte n+1</th>
<th>Byte</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Details</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. parameter</td>
<td>Attribute</td>
<td>Number of elements</td>
<td>4</td>
</tr>
<tr>
<td>Parameter number</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subindex</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nth parameter</td>
<td>Attribute</td>
<td>Number of elements</td>
<td></td>
</tr>
<tr>
<td>Parameter number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subindex</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Values only for write access</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. parameter value(s)</td>
<td>Format</td>
<td>Number of values</td>
<td></td>
</tr>
<tr>
<td>Values</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nth parameter value(s)</td>
<td>Format</td>
<td>Number of values</td>
<td></td>
</tr>
<tr>
<td>Values</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Structure after Base Mode Parameter Access - Parameter Request

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

<table>
<thead>
<tr>
<th>Parameter response</th>
<th>Byte n+1</th>
<th>Byte</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request Header</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Request Reference mirrored</td>
<td>RequestID mirrored or error</td>
<td>RequestID mirrored or error</td>
<td>0</td>
</tr>
<tr>
<td>Axis no./DO ID mirrored</td>
<td>Number of parameters</td>
<td>Number of parameters</td>
<td>2</td>
</tr>
<tr>
<td>Values only for read access</td>
<td>1. parameter value(s)</td>
<td>Format</td>
<td>Number of values</td>
</tr>
<tr>
<td>Values or error codes</td>
<td></td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error values for negative response only</td>
<td>nth parameter value(s)</td>
<td>Format</td>
<td>Number of values</td>
</tr>
<tr>
<td>Values or error codes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Structure after Base Mode Parameter Access - Parameter Response

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

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

Description of fields in parameter job and response

<table>
<thead>
<tr>
<th>Field</th>
<th>Data type</th>
<th>Values</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job reference</td>
<td>Unsigned8</td>
<td>0x01 to 0xFF</td>
<td>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.</td>
</tr>
<tr>
<td>Job identifier</td>
<td>Unsigned8</td>
<td>0x01, 0x02</td>
<td>Read job, Write job</td>
</tr>
<tr>
<td>Response ID</td>
<td>Unsigned8</td>
<td>0x01, 0x02, 0x81, 0x82</td>
<td>Read job (+), Write job (+), Read job (-), Write job (-)</td>
</tr>
<tr>
<td>Drive object number</td>
<td>Unsigned8</td>
<td>0x01 to 0xFE</td>
<td>Number</td>
</tr>
<tr>
<td>Number of parameters</td>
<td>Unsigned8</td>
<td>0x01 to 0x27</td>
<td>No. 1 to 39, Limited by DPV1 message-frame length</td>
</tr>
<tr>
<td>Attribute</td>
<td>Unsigned8</td>
<td>0x10, 0x20, 0x30</td>
<td>Value, Description, Text (not implemented in the case of SINAMICS)</td>
</tr>
<tr>
<td>Number of elements</td>
<td>Unsigned8</td>
<td>0x00, 0x01 to 0x75</td>
<td>Special function, No. 1 to 117, Limited by DPV1 message-frame length</td>
</tr>
<tr>
<td>Parameter number</td>
<td>Unsigned16</td>
<td>0x0001 to 0xFFFF</td>
<td>No. 1 to 65535, Addresses the parameter accessed</td>
</tr>
</tbody>
</table>
### 9.6 Acyclic communication (Base Mode Parameter Access)

<table>
<thead>
<tr>
<th>Field</th>
<th>Data type</th>
<th>Values</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subindex</td>
<td>Unsigned16</td>
<td>0x0000 to 0xFFFF</td>
<td>No. 0 to 65534</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Addresses the first array element of the parameter to be accessed</td>
</tr>
<tr>
<td>Format</td>
<td>Unsigned8</td>
<td></td>
<td>Data type integer8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x02 to 0x04</td>
<td>Data type integer16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x05</td>
<td>Data type integer32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x06</td>
<td>Data type unsigned8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x07</td>
<td>Data type unsigned16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x08</td>
<td>Data type unsigned32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Other values</td>
<td>See PROFIdrive profile V3.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x40</td>
<td>Zero (without values as a positive subresponse to a write job)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x41</td>
<td>Byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x42</td>
<td>Word</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x43</td>
<td>Double word</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x44</td>
<td>Error</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Number of values</th>
<th>Data type</th>
<th>Values</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Unsigned8</td>
<td>0x00 to 0xEA</td>
<td>No. 0 to 234</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Limited by DPV1 message-frame length</td>
</tr>
</tbody>
</table>

Specifies the number of subsequent values.

<table>
<thead>
<tr>
<th>Error values</th>
<th>Data type</th>
<th>Values</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Unsigned16</td>
<td>0x0000 to 0x0FF</td>
<td>Meaning of error values --&gt; see table 4-29</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Values</th>
<th>Data type</th>
<th>Values</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Unsigned16</td>
<td>0x0000 to 0x0FF</td>
<td>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.</td>
</tr>
</tbody>
</table>

For more information on coding PROFIdrive data types, see PROFIdrive-specific data types (Page 309).

**See also**

- Programming example (Page 348)
- Reading and writing parameters with Base Mode Parameter Access (Page 314)
9.6.4 Specifications for PROFIBUS and PROFINET IO

Global and local parameters

PROFIdrive makes a distinction between two parameter ranges:

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

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

- **Base Mode Parameter Access - local (BMPL)**
- **Base Mode Parameter Access - global (BMPG)**

Specific properties of acyclic communication with PROFIBUS

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

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

**Note**

The PROFIdrive standard specifies that in PROFIdrive drives no pipelining of jobs is supported, namely, only one "Read/write data record" is possible concurrently for a single drive device. If, however, more than one PROFIdrive drive unit is connected to a control via PROFIBUS, a job can be processed in parallel for each of these drive units. The maximum number then depends on the controller. The data for SIMOTION is specified in Rule 5 - a maximum of eight concurrent calls is possible in SIMOTION (Page 333).

Specific properties of acyclic communication with PROFINET IO

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

In principle, with both BMPL and BMPG 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.
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.

### 9.6.5 Error assessment

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

- **Error in the communication (transfer of data)**
  
  For example, the addressed device may not exist and is not switched on. This type of error is indicated with the return values of the system functions and is defined in the description of the system functions in the SIMOTION reference lists.

- **Error during the processing of the jobs themselves**
  
  For example, an attempt is made to write to a read-only parameter.

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

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

Error codes are returned in the drive unit's response in the P response/request data block (see table below). The "Format" field in the parameter response can be used to distinguish whether the queried parameter represents an error code or a "true" value. See the Structure after Base Mode Parameter Access - parameter response (Page 316) table, offset 4, "Format".

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

**Note**

The error codes up to 0x19 correspond to the PROFIdrive profile. The error codes from 0x65 onwards are manufacturer-specific and may, therefore, vary from drive to drive.
### Error codes in Base Mode Parameter Access responses

<table>
<thead>
<tr>
<th>Error code</th>
<th>Meaning</th>
<th>Comments</th>
<th>Additional info</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>Illegal parameter number</td>
<td>Access to a parameter which does not exist</td>
<td></td>
</tr>
<tr>
<td>0x01</td>
<td>Parameter value cannot be changed.</td>
<td>Modification access to a parameter value which cannot be changed</td>
<td>Subindex</td>
</tr>
<tr>
<td>0x02</td>
<td>Lower or upper value limit exceeded</td>
<td>Modification access with value outside value limits</td>
<td>Subindex</td>
</tr>
<tr>
<td>0x03</td>
<td>Invalid subindex</td>
<td>Access to a subindex which does not exist</td>
<td>Subindex</td>
</tr>
<tr>
<td>0x04</td>
<td>No array</td>
<td>Access with subindex to non-indexed parameter.</td>
<td></td>
</tr>
<tr>
<td>0x05</td>
<td>Wrong data type</td>
<td>Modification access with a value which does not match the data type of the parameter</td>
<td></td>
</tr>
<tr>
<td>0x06</td>
<td>Setting not allowed (only reset allowed)</td>
<td>Modification access with a value not equal to 0 in a case where this is not allowed</td>
<td>Subindex</td>
</tr>
<tr>
<td>0x07</td>
<td>Description element cannot be changed.</td>
<td>Modification access to a description element which cannot be changed</td>
<td>Subindex</td>
</tr>
<tr>
<td>0x09</td>
<td>No description data</td>
<td>Access to a description which does not exist (the parameter value exists)</td>
<td></td>
</tr>
<tr>
<td>0x0B</td>
<td>No operating priority</td>
<td>Modification access with no operating priority</td>
<td></td>
</tr>
<tr>
<td>0x0F</td>
<td>No text array exists</td>
<td>Access to a text array which does not exist (the parameter value exists)</td>
<td></td>
</tr>
<tr>
<td>0x11</td>
<td>Job cannot be executed due to operating mode.</td>
<td>Access is not possible temporarily for unspecified reasons.</td>
<td></td>
</tr>
<tr>
<td>0x14</td>
<td>Illegal value</td>
<td>Modification access with a value which is within the limits but which is illegal for other permanent reasons (parameter with defined individual values)</td>
<td>Subindex</td>
</tr>
<tr>
<td>0x15</td>
<td>Response too long</td>
<td>The length of the present response exceeds the maximum length that can be transferred.</td>
<td></td>
</tr>
<tr>
<td>0x16</td>
<td>Illegal parameter address</td>
<td>Impermissible or unsupported value for attribute, number of elements, parameter number, subindex, or a combination of these</td>
<td></td>
</tr>
<tr>
<td>0x17</td>
<td>Illegal format</td>
<td>Write job: illegal or unsupported parameter data format</td>
<td></td>
</tr>
<tr>
<td>0x18</td>
<td>No. of values inconsistent</td>
<td>Write job: a mismatch exists between the number of values in the parameter data and the number of elements in the parameter address.</td>
<td></td>
</tr>
<tr>
<td>0x19</td>
<td>Drive object does not exist</td>
<td>You have attempted to access a drive object that does not exist.</td>
<td></td>
</tr>
<tr>
<td>0x65</td>
<td>Presently deactivated</td>
<td>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).</td>
<td></td>
</tr>
<tr>
<td>0x6B</td>
<td>Parameter %s [%s]: no write access with enabled controller</td>
<td>変</td>
<td></td>
</tr>
<tr>
<td>0x6C</td>
<td>Parameter %s [%s]: unit unknown</td>
<td>変</td>
<td></td>
</tr>
<tr>
<td>0x6D</td>
<td>Parameter %s [%s]: Write access only in the commissioning state, encoder (p0010 = 4).</td>
<td>変</td>
<td></td>
</tr>
</tbody>
</table>
## 9.6 Acyclic communication (Base Mode Parameter Access)

<table>
<thead>
<tr>
<th>Error code</th>
<th>Meaning</th>
<th>Comments</th>
<th>Additional info</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x6E</td>
<td>Parameter %s [%s]: Write access only in the commissioning state, motor (p0010 = 3)</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>0x6F</td>
<td>Parameter %s [%s]: Write access only in the commissioning state, power unit (p0010 = 2)</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>0x70</td>
<td>Parameter %s [%s]: Write access only in the quick commissioning mode (p0010 = 1)</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>0x71</td>
<td>Parameter %s [%s]: Write access only in the ready state (p0010 = 0)</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>0x72</td>
<td>Parameter %s [%s]: Write access only in the commissioning state, parameter reset (p0010 = 30)</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>0x73</td>
<td>Parameter %s [%s]: Write access only in the commissioning state, safety (p0010 = 95)</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>0x74</td>
<td>Parameter %s [%s]: Write access only in the commissioning state, tech. application/units (p0010 = 5)</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>0x75</td>
<td>Parameter %s [%s]: Write access only in the commissioning state (p0010 not equal to 0)</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>0x76</td>
<td>Parameter %s [%s]: Write access only in the commissioning state, download (p0010 = 29)</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>0x77</td>
<td>Parameter %s [%s] may not be written in download.</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>0x78</td>
<td>Parameter %s [%s]: Write access only in the commissioning state, drive configuration (device: p0009 = 3)</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>0x79</td>
<td>Parameter %s [%s]: Write access only in the commissioning state, define drive type (device: p0009 = 2)</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>0x7A</td>
<td>Parameter %s [%s]: Write access only in the commissioning state, data set basis configuration (device: p0009 = 4)</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>0x7B</td>
<td>Parameter %s [%s]: Write access only in the commissioning state, device configuration (device: p0009 = 1)</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>0x7C</td>
<td>Parameter %s [%s]: Write access only in the commissioning state, device download (device: p0009 = 29)</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>
### 9.6.6 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":

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x10</td>
<td>Value</td>
</tr>
<tr>
<td>0x20</td>
<td>&quot;Parameter Description&quot; parameter description</td>
</tr>
<tr>
<td>0x30</td>
<td>Parameter Text</td>
</tr>
</tbody>
</table>

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

9.6.7 System commands in SIMOTION

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

Furthermore, the DO ID is only relevant for data set 47 (0x002f) and Global Access (PROFINET 0xb02f). The diagnostics address of the corresponding PAP is relevant for Local Access (PROFINET IO 0xb02e), the DO ID is not analyzed.
As a result, for example in connection with PROFIdrive units, the message frame start address of the PROFIdrive message frame exchanged cyclically with the device is required.

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

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

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

9.6.7.2 _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 335)
9.6.7.3 Comparison of the system commands

Description

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

<table>
<thead>
<tr>
<th>Command group</th>
<th>Advantage</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>_readRecord</td>
<td>• Generally usable, not just for DP V1 services for drives</td>
<td>• The user must create the data record</td>
</tr>
<tr>
<td>_writeRecord</td>
<td>• Assumes only the knowledge of some I/O address on the drive device</td>
<td>• The user must program two calls for parameter accesses in a PROFIdrive drive</td>
</tr>
<tr>
<td></td>
<td>• “DO-ID” or “Axis-No” on the drive device</td>
<td>• Users may need to perform the required data conversions themselves</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• “DO-ID” or “Axis-No” must be known</td>
</tr>
<tr>
<td>_readDrive...</td>
<td>• Tailored for the typical communication with PROFIdrive drives</td>
<td>• Assumes the presence or knowledge of an I/O address of the associated drive object</td>
</tr>
<tr>
<td>_writeDrive...</td>
<td>• The user does not need to know the structure of data set 47</td>
<td>• 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</td>
</tr>
<tr>
<td></td>
<td>• Reduced programming effort for the user for communication to drives</td>
<td>• The user must make any required data conversions</td>
</tr>
</tbody>
</table>

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 348).

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".

9.6.7.4 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.

9.6.8 Rules for using _readRecord and _writeRecord

9.6.8.1 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.

9.6.8.2 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 9-5 Correct processing with the _readRecord and _writeRecord system functions
9.6.8.3 Rule 3 - read/write data record per PROFldrive drive unit

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

The PROFldrive profile specifies that PROFldrive 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 PROFldrive 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 PROFldrive 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 PROFldrive drive device, also refer to section Interlocking of several calls (Page 336).

9.6.8.4 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 PROFldrive 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 9-6 The second _writeRecord call wins in case of doubt
9.6.8.5 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 331)) 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.
9.6 Acyclic communication (Base Mode Parameter Access)

Figure 9-7 Managing 8 jobs simultaneously
Note
If the error 16#80C3 occurs, you must set the CPU to STOP and then back to RUN. This deletes the job buffer. In order to prevent the error, you should end the job with an abort command, if you are unable to end the job.

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

9.6.9.1 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.

9.6.9.2 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.
9.6 Acyclic communication (Base Mode Parameter Access)

9.6.9.3 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.

Figure 9-8 Correct processing with the _writeDriveParameter and _readDriveParameter system functions
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 338).

Figure 9-9 Interlocking of several _readDriveParameter jobs on a target device
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 9-10** Complete processing of a _readDriveParameter required to release the interlock
9.6.9.5 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 328).

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.
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.
**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.
9.6.9.6 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 333).

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.

9.6.9.7 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 controller uses the SINAMICS Integrated of a D445 (for example) as first PROFIdrive drive unit and the CX32 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)) are issued with the _readDriveParameter system function.
• The first read job for the SINAMICS Integrated is intentionally called just once so that the interlock acts.
• The second read job is then issued to the second PROFIdrive drive device (CX32). This job is processed successfully.
• The third read job is addressed again to the first drive device (SINAMICS Integrated) and can no longer be executed successfully because the first job is still running.
Figure 9-13 Parallel processing of _readDriveParameter jobs to different drive devices of a controller
9.6.10 Special features

9.6.10.1 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.

9.6.10.2 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... is essential.

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 9-14 Mixed use of _readDrive... and _read/_writeRecord
9.6.10.3 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
- `_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.

9.6.11 Program examples

9.6.11.1 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.

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

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

// declare struct of request
Request : STRUCT
  Header : Header_Type_Request;
  ParameterAddress : Parameter_Address_Request;
END_STRUCT;

// declare struct of header response
Header_Type_Response : STRUCT
  Response_Reference : USINT;
  Response_Id : USINT;
  Axis : USINT;
  Number_Of_Parameter : USINT;
END_STRUCT;

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

// declare struct of response
Response : STRUCT
  Header : Header_Type_Response;
  ParameterAddress : Parameter_Address_Response;
END_STRUCT;
END_TYPE

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

Implementation

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

// parameter address from the request
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
  bytarray := ANYTYPE_TO_BIGBYTEARRAY(
    myRetstructretreadrecord.data, 0);
  myResponse := BIGBYTEARRAY_TO_ANYTYPE(bytarray, 0);
  // received data can now be read from myResponse...
END_IF;
END_CASE;
END_PROGRAM
END_IMPLEMENTATION
9.6 Acyclic communication (Base Mode Parameter Access)
10.1 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.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Boolean</td>
<td>True/false</td>
<td></td>
<td>1 Octet</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>0x01 to 0xFF</td>
</tr>
<tr>
<td>False</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0x00</td>
</tr>
</tbody>
</table>

Integer16

An Integer16 is representing a signed number depicted by 16 bits.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Integer16</td>
<td>-32768 ≤ i ≤ 32767</td>
<td>1</td>
<td>2 Octets</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octets 1</td>
<td>SN</td>
<td>2^14</td>
<td>2^13</td>
<td>2^12</td>
<td>2^11</td>
<td>2^10</td>
<td>2^9</td>
<td>2^8</td>
</tr>
<tr>
<td>Octets 2</td>
<td>2^7</td>
<td>2^6</td>
<td>2^5</td>
<td>2^4</td>
<td>2^3</td>
<td>2^2</td>
<td>2^1</td>
<td>2^0</td>
</tr>
</tbody>
</table>
Appendix

10.1 Standard PROFIBUS/PROFINET data types (only available in English)

**Integer32**

An Integer32 is representing a signed number depicted by 32 bits.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Integer32</td>
<td>-2³¹ ≤ i ≤ 2³¹-1</td>
<td>1</td>
<td>4 Octets</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octets 1</td>
<td>SN</td>
<td>2³⁰</td>
<td>2²⁹</td>
<td>2²⁸</td>
<td>2²⁷</td>
<td>2²⁶</td>
<td>2²⁵</td>
<td>2²⁴</td>
</tr>
<tr>
<td>Octets 2</td>
<td>2²³</td>
<td>2²²</td>
<td>2²¹</td>
<td>2²⁰</td>
<td>2²⁹</td>
<td>2²⁸</td>
<td>2²⁷</td>
<td>2²⁶</td>
</tr>
<tr>
<td>Octets 3</td>
<td>2¹⁶</td>
<td>2¹⁵</td>
<td>2¹⁴</td>
<td>2¹³</td>
<td>2¹²</td>
<td>2¹¹</td>
<td>2¹⁰</td>
<td>2⁹</td>
</tr>
<tr>
<td>Octets 4</td>
<td>2⁷</td>
<td>2⁶</td>
<td>2⁵</td>
<td>2⁴</td>
<td>2³</td>
<td>2²</td>
<td>2¹</td>
<td>2⁰</td>
</tr>
</tbody>
</table>

**Integer64**

An Integer64 is representing a signed number depicted by 64 bits.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>55</td>
<td>Integer64</td>
<td>-2⁶² ≤ i ≤ 2⁶²-1</td>
<td>1</td>
<td>8 Octets</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octets 1</td>
<td>SN</td>
<td>2⁶²</td>
<td>2⁶¹</td>
<td>2⁶⁰</td>
<td>2⁵⁹</td>
<td>2⁵⁸</td>
<td>2⁵⁷</td>
<td>2⁵⁶</td>
</tr>
<tr>
<td>Octets 1</td>
<td>2⁵⁵</td>
<td>2⁵⁴</td>
<td>2⁵³</td>
<td>2⁵²</td>
<td>2⁵¹</td>
<td>2⁵⁰</td>
<td>2⁴⁹</td>
<td>2⁴⁸</td>
</tr>
<tr>
<td>Octets 1</td>
<td>2⁴⁷</td>
<td>2⁴⁶</td>
<td>2⁴⁵</td>
<td>2⁴⁴</td>
<td>2⁴³</td>
<td>2⁴²</td>
<td>2⁴¹</td>
<td>2⁴⁰</td>
</tr>
<tr>
<td>Octets 1</td>
<td>2³⁰</td>
<td>2²⁹</td>
<td>2²⁸</td>
<td>2²⁷</td>
<td>2²⁶</td>
<td>2²⁵</td>
<td>2²⁴</td>
<td>2²³</td>
</tr>
<tr>
<td>Octets 1</td>
<td>2²³</td>
<td>2²²</td>
<td>2²¹</td>
<td>2²⁰</td>
<td>2²⁹</td>
<td>2²⁸</td>
<td>2²⁷</td>
<td>2²⁶</td>
</tr>
<tr>
<td>Octets 3</td>
<td>2¹⁵</td>
<td>2¹⁴</td>
<td>2¹³</td>
<td>2¹²</td>
<td>2¹¹</td>
<td>2¹⁰</td>
<td>2⁹</td>
<td>2⁸</td>
</tr>
<tr>
<td>Octets 4</td>
<td>2⁷</td>
<td>2⁶</td>
<td>2⁵</td>
<td>2⁴</td>
<td>2³</td>
<td>2²</td>
<td>2¹</td>
<td>2⁰</td>
</tr>
</tbody>
</table>
10.1 Standard PROFIBUS/PROFINET data types (only available in English)

**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.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Unsigned8</td>
<td>0 ≤ i ≤ 255</td>
<td>1</td>
<td>1 Octet</td>
</tr>
</tbody>
</table>

Table 10-1 Enumerated:

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octets 1</td>
<td>2^7</td>
<td>2^6</td>
<td>2^5</td>
<td>2^4</td>
<td>2^3</td>
<td>2^2</td>
<td>2^1</td>
<td>2^0</td>
</tr>
</tbody>
</table>

Table 10-2 Single bits:

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octets 1</td>
<td>1</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Unsigned16</td>
<td>0 ≤ i ≤ 65535</td>
<td>1</td>
<td>2 Octets</td>
</tr>
</tbody>
</table>

Table 10-3 Enumerated:

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octets 1</td>
<td>2^15</td>
<td>2^14</td>
<td>2^13</td>
<td>2^12</td>
<td>2^11</td>
<td>2^10</td>
<td>2^9</td>
<td>2^8</td>
</tr>
<tr>
<td>Octets 2</td>
<td>2^7</td>
<td>2^6</td>
<td>2^5</td>
<td>2^4</td>
<td>2^3</td>
<td>2^2</td>
<td>2^1</td>
<td>2^0</td>
</tr>
</tbody>
</table>

Table 10-4 Single bits:

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octets 1</td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>Octets 2</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
## 10.1 Standard PROFIBUS/PROFINET data types (only available in English)

### 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.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Unsigned32</td>
<td>0 ≤ i ≤ 4294967295</td>
<td>1</td>
<td>4 Octets</td>
</tr>
</tbody>
</table>

#### Table 10-5  Enumerated:

<table>
<thead>
<tr>
<th>Bits</th>
<th>Octets 1</th>
<th>Octets 2</th>
<th>Octets 3</th>
<th>Octets 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>2^31</td>
<td>2^30</td>
<td>2^29</td>
<td>2^28</td>
</tr>
<tr>
<td>6</td>
<td>2^27</td>
<td>2^26</td>
<td>2^25</td>
<td>2^24</td>
</tr>
<tr>
<td>5</td>
<td>2^23</td>
<td>2^22</td>
<td>2^21</td>
<td>2^20</td>
</tr>
<tr>
<td>4</td>
<td>2^19</td>
<td>2^18</td>
<td>2^17</td>
<td>2^16</td>
</tr>
<tr>
<td>3</td>
<td>2^15</td>
<td>2^14</td>
<td>2^13</td>
<td>2^12</td>
</tr>
<tr>
<td>2</td>
<td>2^11</td>
<td>2^10</td>
<td>2^9</td>
<td>2^8</td>
</tr>
<tr>
<td>1</td>
<td>2^7</td>
<td>2^6</td>
<td>2^5</td>
<td>2^4</td>
</tr>
<tr>
<td>0</td>
<td>2^0</td>
<td>2^1</td>
<td>2^2</td>
<td>2^3</td>
</tr>
</tbody>
</table>

#### Table 10-6  Single bits:

<table>
<thead>
<tr>
<th>Bits</th>
<th>Octets 1</th>
<th>Octets 2</th>
<th>Octets 3</th>
<th>Octets 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>31</td>
<td>23</td>
<td>15</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>30</td>
<td>22</td>
<td>14</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>29</td>
<td>21</td>
<td>13</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>28</td>
<td>20</td>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>27</td>
<td>19</td>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>26</td>
<td>18</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>25</td>
<td>17</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>24</td>
<td>16</td>
<td>8</td>
<td>0</td>
</tr>
</tbody>
</table>

### Unsigned64

An Unsigned64 is representing a signed number depicted by 64 bits.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>56</td>
<td>Unsigned64</td>
<td>0 ≤ i ≤ 2^64-1</td>
<td>1</td>
<td>8 Octets</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Bits</th>
<th>Octets 1</th>
<th>Octets 2</th>
<th>Octets 3</th>
<th>Octets 4</th>
<th>Octets 5</th>
<th>Octets 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>2^63</td>
<td>2^62</td>
<td>2^61</td>
<td>2^60</td>
<td>2^59</td>
<td>2^58</td>
</tr>
<tr>
<td>6</td>
<td>2^58</td>
<td>2^57</td>
<td>2^56</td>
<td>2^55</td>
<td>2^54</td>
<td>2^53</td>
</tr>
<tr>
<td>5</td>
<td>2^52</td>
<td>2^51</td>
<td>2^50</td>
<td>2^49</td>
<td>2^48</td>
<td>2^47</td>
</tr>
<tr>
<td>4</td>
<td>2^46</td>
<td>2^45</td>
<td>2^44</td>
<td>2^43</td>
<td>2^42</td>
<td>2^41</td>
</tr>
<tr>
<td>3</td>
<td>2^40</td>
<td>2^39</td>
<td>2^38</td>
<td>2^37</td>
<td>2^36</td>
<td>2^35</td>
</tr>
<tr>
<td>2</td>
<td>2^34</td>
<td>2^33</td>
<td>2^32</td>
<td>2^31</td>
<td>2^30</td>
<td>2^29</td>
</tr>
<tr>
<td>1</td>
<td>2^28</td>
<td>2^27</td>
<td>2^26</td>
<td>2^25</td>
<td>2^24</td>
<td>2^23</td>
</tr>
<tr>
<td>0</td>
<td>2^22</td>
<td>2^21</td>
<td>2^20</td>
<td>2^19</td>
<td>2^18</td>
<td>2^17</td>
</tr>
</tbody>
</table>
## Appendix

### 10.1 Standard PROFIBUS/PROFINET data types (only available in English)

#### Float32

A Float32 is representing a number defined by ANSI/IEEE 754 as single precision.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Float32</td>
<td>refer to ANSI/IEEE 754</td>
<td>refer to ANSI/IEEE 754</td>
<td>4 Octets</td>
</tr>
</tbody>
</table>

SN: sign 0 = positive, 1 = negative.

#### Float64

A Float64 is representing a number defined by ANSI/IEEE 754 as single precision.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Float64</td>
<td>refer to ANSI/IEEE 754</td>
<td>refer to ANSI/IEEE 754</td>
<td>8 Octets</td>
</tr>
</tbody>
</table>

SN: sign 0 = positive, 1 = negative.
Appendix

10.1 Standard PROFIBUS/PROFINET data types (only available in English)

Visible String
This data type is defined as the ISO 646 string type. Characters are based on 8 Bit ASCII.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>VisibleString</td>
<td>refer to ISO 646</td>
<td>-</td>
<td>variable</td>
</tr>
</tbody>
</table>

OctetString
An OctetString is an ordered sequence of Bytes, numbered from 1 to n.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>OctetString</td>
<td>-</td>
<td>-</td>
<td>variable</td>
</tr>
</tbody>
</table>

BYTE
A Byte is 1 octet.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>Byte</td>
<td>-</td>
<td>-</td>
<td>1 Octet</td>
</tr>
</tbody>
</table>
### WORD

A Word is an ordered sequence of 2 octets.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>Word</td>
<td>-</td>
<td>-</td>
<td>2 Octets</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octet 1</td>
<td>1. Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Octet 2</td>
<td>2. Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### DWORD

A DWord is an ordered sequence of 4 octets.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>DWord</td>
<td>-</td>
<td>-</td>
<td>4 Octets</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octet 1</td>
<td>1. Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Octet 2</td>
<td>2. Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Octet 3</td>
<td>3. Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Octet 4</td>
<td>4. Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### LWORD

A LWord is an ordered sequence of 8 octets.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>LWord</td>
<td>-</td>
<td>-</td>
<td>8 Octets</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octet 1</td>
<td>1. Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Octet 2</td>
<td>2. Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Octet 3</td>
<td>3. Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Octet 4</td>
<td>4. Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Octet 5</td>
<td>5. Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Octet 6</td>
<td>6. Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Octet 7</td>
<td>7. Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Octet 8</td>
<td>8. Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix
10.1 Standard PROFIBUS/PROFINET data types (only available in English)

**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.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
</table>
| 12                 | TimeOfDay      | 0 ≤ i ≤ (2^32-1) ms  
                    |                | 0 ≤ i ≤ (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.

**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.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
</table>
| 13                 | TimeDifference | 0 ≤ i ≤ (2^32-1) ms  
                    |                | 0 ≤ i ≤ (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.

---

Communication
System Manual, 02/2012
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.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>Date</td>
<td>0 ms ≤ i ≤ 99 years</td>
<td></td>
<td>7 Octets (Unsigned16 + 5 x Unsigned8)</td>
</tr>
</tbody>
</table>

| 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 | 1...12 month | 0...99 years |
| Octet 6 | res | res | 2^5  | 2^4  | 2^3  | 2^2  | 2^1  | 2^0  |                          |
| Octet 7 | res | 2^4  | 2^3  | 2^2  | 2^1  | 2^0  |      |      |                          |

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.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>52</td>
<td>TimeOfDay without date indication</td>
<td>0 ≤ i ≤ (2^32-1)</td>
<td>ms</td>
<td>4 Octets (Unsigned32)</td>
</tr>
</tbody>
</table>

The time is interpreted as 32 bit value.
### 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.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>53</td>
<td>TimeDifference with date indication</td>
<td>0 ≤ i ≤ (2^{32}-1) ms</td>
<td>ms, days</td>
<td>8 Octets (Unsigned32 + Unsigned16)</td>
</tr>
</tbody>
</table>

The time is interpreted as 32 bit value. The date indication is coded as 16 bit value.

### TimeDifference without Date indication

This data type consists of one element of an unsigned value that expresses the difference in time. The element is an Unsigned32 data type providing the fractional portion of one day in milliseconds.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>54</td>
<td>TimeDifference without date indication</td>
<td>0 ≤ i ≤ (2^{32}-1) ms</td>
<td>ms, days</td>
<td>4 Octets (Unsigned32)</td>
</tr>
</tbody>
</table>

The time is interpreted as 32 bit value.
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.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>58</td>
<td>NetworkTime</td>
<td>Byte 1 to 4: 0 ≤ i ≤ (232-1)</td>
<td>S</td>
<td>8 Octets</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Byte 5 to 8: 0 ≤ i ≤ (232-1)</td>
<td>(1/2^32) s</td>
<td>(Unsigned32 +</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unsigned32)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>Octet 1</th>
<th>Octet 2</th>
<th>Octet 3</th>
<th>Octet 4</th>
<th>Octet 5</th>
<th>Octet 6</th>
<th>Octet 7</th>
<th>Octet 8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2^31</td>
<td>2^30</td>
<td>2^29</td>
<td>2^28</td>
<td>2^27</td>
<td>2^26</td>
<td>2^25</td>
<td>2^24</td>
</tr>
<tr>
<td></td>
<td>2^23</td>
<td>2^22</td>
<td>2^21</td>
<td>2^20</td>
<td>2^19</td>
<td>2^18</td>
<td>2^17</td>
<td>2^16</td>
</tr>
<tr>
<td></td>
<td>2^15</td>
<td>2^14</td>
<td>2^13</td>
<td>2^12</td>
<td>2^11</td>
<td>2^10</td>
<td>2^9</td>
<td>2^8</td>
</tr>
<tr>
<td></td>
<td>2^7</td>
<td>2^6</td>
<td>2^5</td>
<td>2^4</td>
<td>2^3</td>
<td>2^2</td>
<td>2^1</td>
<td>2^0</td>
</tr>
</tbody>
</table>

The least significant Bit of the fractional portion (2^5) 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.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>59</td>
<td>NetworkTimeDifference</td>
<td>Byte 1 to 4: -2^31 ≤ i ≤ (2^31-1)</td>
<td>S</td>
<td>8 Octets</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Byte 5 to 8: 0 ≤ i ≤ (2^32-1)</td>
<td>(1/2^32) s</td>
<td>(Integer32 +</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unsigned32)</td>
</tr>
</tbody>
</table>
10.2 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.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Float32 +Unsigned8</td>
<td>See Float32 and Unsigned8</td>
<td>-</td>
<td>5 Octets</td>
</tr>
</tbody>
</table>

SN: sign 0 = positive, 1 = negative

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octet 1</td>
<td>SN</td>
<td>2^30</td>
<td>2^29</td>
<td>2^28</td>
<td>2^27</td>
<td>2^26</td>
<td>2^25</td>
<td>2^24</td>
</tr>
<tr>
<td>Octet 2</td>
<td>2^23</td>
<td>2^22</td>
<td>2^21</td>
<td>2^20</td>
<td>2^19</td>
<td>2^18</td>
<td>2^17</td>
<td>2^16</td>
</tr>
<tr>
<td>Octet 3</td>
<td>2^15</td>
<td>2^14</td>
<td>2^13</td>
<td>2^12</td>
<td>2^11</td>
<td>2^10</td>
<td>2^9</td>
<td>2^8</td>
</tr>
<tr>
<td>Octet 4</td>
<td>2^7</td>
<td>2^6</td>
<td>2^5</td>
<td>2^4</td>
<td>2^3</td>
<td>2^2</td>
<td>2^1</td>
<td>2^0</td>
</tr>
<tr>
<td>Octet 5</td>
<td>2^31</td>
<td>2^30</td>
<td>2^29</td>
<td>2^28</td>
<td>2^27</td>
<td>2^26</td>
<td>2^25</td>
<td>2^24</td>
</tr>
<tr>
<td>Octet 6</td>
<td>2^23</td>
<td>2^22</td>
<td>2^21</td>
<td>2^20</td>
<td>2^19</td>
<td>2^18</td>
<td>2^17</td>
<td>2^16</td>
</tr>
<tr>
<td>Octet 7</td>
<td>2^15</td>
<td>2^14</td>
<td>2^13</td>
<td>2^12</td>
<td>2^11</td>
<td>2^10</td>
<td>2^9</td>
<td>2^8</td>
</tr>
<tr>
<td>Octet 8</td>
<td>2^7</td>
<td>2^6</td>
<td>2^5</td>
<td>2^4</td>
<td>2^3</td>
<td>2^2</td>
<td>2^1</td>
<td>2^0</td>
</tr>
</tbody>
</table>

Signed number of seconds

Fractional portion of seconds: 1/2^32 s

See also

PROFIdrive-specific data types (Page 309)
**10.2 Profile-specific PROFIBUS/PROFINET data types (only available in English)**

**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.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>102</td>
<td>Unsigned8 + Unsigned8</td>
<td>See Unsigned8</td>
<td>-</td>
<td>2 Octets</td>
</tr>
</tbody>
</table>

In two’s complement; the most significant bit (MSB) is the bit after the sign (SN) in the first Byte.

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octet 1</td>
<td>$2^7$</td>
<td>$2^6$</td>
<td>$2^5$</td>
<td>$2^4$</td>
<td>$2^3$</td>
<td>$2^2$</td>
<td>$2^1$</td>
<td>$2^0$</td>
</tr>
<tr>
<td>Octet 2</td>
<td>$2^7$</td>
<td>$2^6$</td>
<td>$2^5$</td>
<td>$2^4$</td>
<td>$2^3$</td>
<td>$2^2$</td>
<td>$2^1$</td>
<td>$2^0$</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>103</td>
<td>OctetString2 + Unsigned8</td>
<td>see OctetString2 and Unsigned8</td>
<td>-</td>
<td>3 Octets</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octet 1</td>
<td>1. Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Octet 2</td>
<td>2. Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Octet 3</td>
<td>$2^7$</td>
<td>$2^6$</td>
<td>$2^5$</td>
<td>$2^4$</td>
<td>$2^3$</td>
<td>$2^2$</td>
<td>$2^1$</td>
<td>$2^0$</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>104</td>
<td>Unsigned16_S</td>
<td>0 to $2^{15}$ + 0 to 3</td>
<td>1</td>
<td>2 Octets</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octets 1</td>
<td>$2^{13}$</td>
<td>$2^{12}$</td>
<td>$2^{11}$</td>
<td>$2^{10}$</td>
<td>$2^9$</td>
<td>$2^8$</td>
<td>$2^7$</td>
<td>$2^6$</td>
</tr>
<tr>
<td>Octets 2</td>
<td>$2^7$</td>
<td>$2^6$</td>
<td>$2^5$</td>
<td>$2^4$</td>
<td>$2^3$</td>
<td>$2^2$</td>
<td>$2^1$</td>
<td>$2^0$</td>
</tr>
</tbody>
</table>
### 10.2 Profile-specific PROFIBUS/PROFINET data types (only available in English)

#### 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.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>105</td>
<td>Integer16_S</td>
<td>-2^{12} to 2^{12} - 1 + 0 to 3</td>
<td>1</td>
<td>2 Octets</td>
</tr>
</tbody>
</table>

SN: sign 0 = positive, 1 = negative

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octets 1</td>
<td>SN</td>
<td>2^{12}</td>
<td>2^{11}</td>
<td>2^{10}</td>
<td>2^9</td>
<td>2^8</td>
<td>2^7</td>
<td>2^6</td>
</tr>
<tr>
<td>Octets 2</td>
<td>2^5</td>
<td>2^4</td>
<td>2^3</td>
<td>2^2</td>
<td>2^1</td>
<td>2^0</td>
<td>St1</td>
<td>St0</td>
</tr>
</tbody>
</table>

#### 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.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>106</td>
<td>Unsigned8_S</td>
<td>0 to 2^8 + 0 to 3</td>
<td>1</td>
<td>1 Octet</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>st1 (bit1)</th>
<th>st0 (bit0)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>input channel: bad (value is fail-safe value)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>output channel: reserved</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>input channel: simulation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>output channel: reserved</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>input channel: uncertain</td>
</tr>
<tr>
<td></td>
<td></td>
<td>output channel: reserved</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>input channel: good</td>
</tr>
<tr>
<td></td>
<td></td>
<td>output channel: reserved</td>
</tr>
</tbody>
</table>
10.2 Profile-specific PROFIBUS/PROFINET data types (only available in English)

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octets 1</td>
<td>$2^7$</td>
<td>$2^6$</td>
<td>$2^5$</td>
<td>$2^4$</td>
<td>$2^3$</td>
<td>$2^2$</td>
<td>$2^1$</td>
<td>$2^0$</td>
</tr>
<tr>
<td>st1</td>
<td>st0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0</td>
<td>0 0</td>
<td>input channel: bad (value is fail-safe value) output channel: reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1</td>
<td>0 1</td>
<td>input channel: simulation output channel: reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0</td>
<td>0 0</td>
<td>input channel: uncertain output channel: reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1</td>
<td>0 0</td>
<td>input channel: good output channel: reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>107</td>
<td>OctetString_S</td>
<td>See OctetString and below</td>
<td>-</td>
<td>n Octets</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>bit 7</th>
<th>bit 6</th>
<th>bit 5</th>
<th>bit 4</th>
<th>bit 3</th>
<th>bit 2</th>
<th>bit 1</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octet 1</td>
<td>ch(8)</td>
<td>ch(7)</td>
<td>ch(6)</td>
<td>ch(5)</td>
<td>ch(4)</td>
<td>ch(3)</td>
<td>ch(2)</td>
<td>ch(1)</td>
</tr>
<tr>
<td>Octet m</td>
<td>ch(n)</td>
<td>ch(n-1)</td>
<td>ch(n-2)</td>
<td>ch(n-3)</td>
<td>ch(n-4)</td>
<td>ch(n-5)</td>
<td>ch(n-6)</td>
<td>ch(n-7)</td>
</tr>
<tr>
<td>Octet m+1</td>
<td>st1(4)</td>
<td>st0(4)</td>
<td>st1(3)</td>
<td>st0(3)</td>
<td>st1(2)</td>
<td>st0(2)</td>
<td>st1(1)</td>
<td>st0(1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>107</td>
<td>OctetString_S</td>
<td>See OctetString and below</td>
<td>-</td>
<td>n Octets</td>
</tr>
</tbody>
</table>

ch(x) value for channel x; st(x) status information for channel x; 1<x≤n
10.2 Profile-specific PROFIBUS/PROFINET data types (only available in English)

### st1 (bit1) st0 (bit0) Meaning

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>
| 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.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>110</td>
<td>F message trailer with 4 octets</td>
<td>PROFIsafe: V1-mode and V2-mode</td>
<td>-</td>
<td>4 Octets</td>
</tr>
</tbody>
</table>

**Bits**

<table>
<thead>
<tr>
<th>Octet 1</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status/Control octet</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Octet 2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consecutive number (V1-mode) or High-octet CRC(V2-mode) *)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Octet 3</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRC *)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Octet 4</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low-byte CRC (least significant octet) *)</td>
<td></td>
</tr>
</tbody>
</table>

* 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.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>F message trailer with 5 octets</td>
<td>PROFIsafe: V2-mode</td>
<td>-</td>
<td>5 Octets</td>
</tr>
</tbody>
</table>
10.2 Profile-specific PROFIBUS/PROFINET data types (only available in English)

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octet 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Status/Control octet</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Octet 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. byte CRC (most significant octet) *</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Octet 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. byte CRC *</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Octet 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. byte CRC *</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Octet 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. byte CRC (least significant octet) *</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* 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.

<table>
<thead>
<tr>
<th>Numeric Identifier</th>
<th>Data type name</th>
<th>Value range</th>
<th>Resolution</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>F message trailer with 6 octets</td>
<td>PROFIsafe: V1-mode</td>
<td>-</td>
<td>6 Octets</td>
</tr>
</tbody>
</table>

* Byte ordering according to IEC 61158-5 Type 3

**See also**

PROFIdrive-specific data types (Page 309)
Appendix

10.2 Profile-specific PROFIBUS/PROFINET data types (only available in English)
Index

- _readRecord
  Application, 328
- _tcpCloseConnection
  TCP communication, 200
- _tcpCloseConnection system function, 200
- _tcpCloseServer
  TCP communication, 200
- _tcpCloseServer system function, 200
- _tcpOpenClient
  TCP communication, 197
- _tcpOpenClient system function, 197
- _tcpOpenServer
  TCP communication, 196
- _tcpOpenServer system function, 196
- _tcpReceive
  TCP communication, 199
- _tcpReceive system function, 199
- _tcpSend
  TCP communication, 198
- _tcpSend system function, 198
- _udpAddMulticastGroupMembership, 217
- _udpAddMulticastGroupMembership system function, 217
- _udpDropMulticastGroupMembership, 218
- _udpReceive, 216
- _udpReceive system function, 216
- _udpSend, 215
- _udpSend system function, 215
- _writeRecord
  Application, 328
  _xreceive, 37
  _xsend, 35

A

Acyclic communication, 313
AG_LRECV (FC60), 207
AG_LSEND (FC50), 207
AG_RECV (FC6), 207
AG_SEND (FC5), 207
AG_SRECV (FC63), 207
AG_SSEND (FC53), 207

B

Base Mode Parameter Access, 313

C

CIDR
  Classless Inter-Domain Routing, 224
Commissioning
  PROFIsafe with Scout, 287, 295
Communication
  Between SIMOTION and SIMATIC, 25
  SIMATIC as an I-slave, 30
  SIMATIC as DP slave, 30
  SIMOTION as an I-slave, 27
  SIMOTION as DP slave, 27
Configuring
  Shared Device, 280
Configuring the sender, 151
Controller Application Cycle Factor, 62
Cycle clock scaling, 62

D

Data block (PAP), 316
Diagnostics model, 174
DP cycle
  PROFINET IO, 119
DP slave
  SIMOTION, 27
DP V1 communication
  Program example, 348
Drive Based Safety
  Functions, 245
  Message frames, 246

E

Ethernet
  LCom library, 191
  Properties of the subnets, 189
  Using interface, 190
Ethernet CP communication
  Configuring, 208, 220
Index

F
F-Proxy
  iDevice F-Proxy, 252
  PROFIsafe, 248
FTP file transfer, 243

I
I device F-Proxy
  Configuring procedure, 261
  Failsafe address, 278
  PROFIBUS integrated, 260
  PROFIBUS topology, 258
  PROFINET, 259
  Properties, 256
  Runtime, 256
iDevice F-Proxy
  Basic information, 252
  Configuring, 263, 272
  Requirement, 254
Insert PROFINET board, 106
IRT High Performance, 47
I-slave
  SIMATIC, 30
  SIMOTION, 27
I-slave-F-Proxy, 286

L
LCom library, 191

M
Media redundancy
  Media redundancy for IRT frames, 88
  Media Redundancy Protocol, 88
  Ring ports, 90
MRPD
  Configuring, 147

P
PN/PN coupler, 171
PROFIBUS
  Acyclic communication, 313
  Cyclic services, 25
  DPV1 communication, 313
PROFIBUS master-master
  S7 system functions, 34
  SIMOTION functions, 35
PROFIBUS message frame, 289, 295
PROFIdrive
  Application classes, 307
  Profile, 301
PROFINET
  Inserting CBE30, 106
PROFINET IO
  2 interfaces, 72
  Controlled SYNC master, 80
  IO controller, 40
  IO device, 40
  RT, 46
  SIMATIC - SIMOTION data exchange, 173
PROFIsafe
  Adapting F addresses for the entire project, 269
  Basic information, 245
  Configuring SIMOTION D, 287
  Failsafe data exchange broadcast, 285, 295
  F-Proxy, 248
  I-slave F-Proxy, 285
  Master-slave coupling, 291
  PROFIBUS I slave F-Proxy, 287
  PROFIBUS, requirement, 285
  Upgrading PROFIBUS to PROFINET, 276
PROFIsafe communication, 287, 295
PROFIsafe slot, 289, 295

R
Recipients
  Configuring, 152
References, 4
Ring ports, 90
Routing
  S7 routing, 224

S
S7 communication blocks, TCP
  Ethernet CP, 207, 212
  Onboard Ethernet interface, 201
  UDT65, 202, 204
S7 communication blocks, UDP
  Ethernet CP, 219
  Onboard Ethernet interface, 218
Send clock
  DP cycle, 119
Shared Device
  Basic information, 279
  Configuring, 280
  SIMATIC F-CPU, 287, 295
SIMOTION IT
  IT-DIAG, 240
  OPC XML DA, 242
  Variable access, 241
SINAMICS drive unit, 289, 295
SP slave
  SIMATIC, 30
Speed controller
  Automatic controller setting, 245
Sync domain, 49
  creating, 114
Sync master
  Redundant, 85

T
TCP communication, 192
  LCom library, 191
  SIMOTION system functions, 195
Topology
  Configuring, 123

U
UDP communication, 213
  System functions, 215
UDT65, 202, 204

X
  X_RCV, 34
  X_SEND, 34