rexroth
A Bosch Company

IndraWorks 15VRS

Field Buses

Application Description
R911393284

Edition 05



Title	IndraWorks 15VRS Field Buses
Type of Documentation	Application Description
Document Typecode	DOK-IWORKS-FB******V15-AP05-EN-P
Internal File Reference	RS-49feab6e24cdd0340a347e865dedebd4-5-en-US-3

Change Record	Edition 05, 2021-03 Refer to tab. 1-1 "Change Record" on page 1
Copyright	© Bosch Rexroth AG 2021 All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.
Liability	The specified data is intended for product description purposes only and shall not be deemed to be a guaranteed characteristic unless expressly stipulated in the contract. All rights are reserved with respect to the content of this documentation and the availability of the product.
Editorial Department	Engineering Software and Control Firmware, Control Functions PLC and Field Buses (BAn, HBu, CCl) Technical Writing (MiNi/PiaSt)

Table of Contents

Page

1 About this documentation ... 1
1.1 Validity of the documentation ... 1
1.2 Using safety instructions ... 2
1.2.1 Structure of the safety instructions ... 2
1.2.2 Explaining Signal Words and Safety Alert Symbol ... 2
1.2.3 Symbols Used ... 4
1.2.4 Explaining the signal alert symbol on the device ... 4
1.3 Required and supplementing documentation ... 4

2 General information ... 7
2.1 Terms and abbreviations ... 7
2.2 Transmission modes ... 8

3 Field bus interfaces and characteristics ... 11
3.1 Field buses and field bus communication ... 11
3.1.1 Field bus interfaces at the XM control system ... 11
3.2 Field bus features ... 13
3.2.1 Field bus master ... 13
3.2.2 Field bus slaves ... 14

4 Field bus-comprehensive features ... 15
4.1 Device database ... 15
4.1.1 Device database, overview ... 15
4.1.2 Managing devices ... 15
Device database, dialog ... 15
Adding devices ... 16
Removing devices ... 18
4.2 Mapping of the field bus inputs and outputs ... 19
4.3 Field bus mapping, access from field bus master to variables of the field bus slave ... 23
4.4 Field bus configuration "Maximal Project" ... 26
4.5 Bus cycle task: Copying I/O data between PLC and field bus stack ... 27
4.5.1 Describing the bus cycle task ... 27
4.5.2 Configuring the bus cycle task ... 29
4.5.3 Bus-synchronous PLC processing ... 30
4.5.4 PLC setting "Update variables" ... 31
4.6 IP channel for Profinet I/O & EtherNet/IP (netX) ... 32
4.7 PLC controls drives via field bus connection (field bus axes) ... 35
4.7.1 General information ... 35
4.7.2 Creating field bus axes in the project ... 36
4.7.3 Drive settings for the PLC communication ... 37
General information ... 37
Establishing the communication for drive parameterization ... 38
IndraDrive communication settings ... 38

Page

4.7.4 Including required libraries ... 39
4.7.5 Addressing field bus drives in the PLC project ... 41
AXIS_REF, axis addresses of field bus drives ... 41
AxisData[], axis data ... 43
4.7.6 I/O addressing of the drives in the PLC project ... 43

5 Profibus DP ... 45
5.1 Features of the Profibus DP master ... 45
5.2 Features of the Profibus DP, slave (control as slave) ... 45
5.3 Profibus DP, basics ... 46
5.3.1 Terms and abbreviations ... 46
5.4 Profibus DP, configuration overview ... 46
5.5 Profibus DP, master ... 48
5.5.1 Profibus DP master, configure ... 48
General information ... 48
Tab “General” ... 49
Tab “Profibus DP master parameter” ... 51
Tab “Field bus diagnostics” ... 52
Tab “I/O mapping” ... 55
“Profibus DP master IEC objects” tab ... 55
Tab “Status” ... 56
Tab “Information” ... 56
5.5.2 Profibus DP master, configuring slaves ... 56
Adding slaves ... 56
Enabling/disabling Profibus DP slaves ... 58
Disabling Profibus DP slaves using UserDefs.cfg ... 59
Configuring slaves ... 61
Tab “General” ... 62
Tab 'Profibus DP parameter' ... 64
Tab “Status” ... 64
Tab “Information” ... 64
5.5.3 Profibus DP master, adding modules to slave ... 64
General ... 64
“General” tab ... 65
“I/O mapping” tab ... 66
“Status” tab ... 67
“Information” tab ... 67
5.6 Profibus DP, slave (control as slave) ... 67
5.6.1 Configuring Profibus DP slave ... 67
General ... 67
“General” tab ... 68
“I/O mapping” tab ... 68
“Profibus DP IEC objects” tab ... 68
“Field bus mapping” tab ... 69
“Profibus DP parameters” tab ... 69
“Profibus DP configuration” tab ... 70

Page

“Status” tab ... 70
“Information” tab ... 71
5.6.2 Profibus DP slave, adding coupling modules ... 71
General Information ... 71
“General” tab ... 72
“DP module parameters” tab ... 72
“DP modules I/O mapping” tab ... 72
“Status” tab ... 73
“Information” tab ... 73
5.6.3 Profibus DP slave, master-slave coupling ... 73

6 Profinet I/O ... 75
6.1 Features of the Profinet I/O controller ... 75
6.2 Features of the Profinet I/O device (control as slave) ... 76
6.3 Profinet I/O, basics ... 77
6.3.1 Terms and abbreviations ... 77
6.3.2 Meaning of Device Names in case of Profinet IO ... 78
6.3.3 Shared Device ... 79
6.3.4 Media Redundancy Protocol (MRP) ... 79
6.3.5 Identifying ... 80
6.4 Profinet I/O, configuration overview ... 80
6.5 Profinet I/O, controller ... 84
6.5.1 Profinet I/O controller, configuring ... 84
Device editor ... 84
Tab “General” ... 84
Tab “Overview” ... 87
Tab “Settings” ... 88
Tab “IP channel settings” ... 89
Tab “Profinet I/O controller parameters” ... 89
Tab “Field bus diagnostics” ... 89
Tab “Topology” ... 90
Tab “I/O mapping” ... 92
Tab “Status” ... 92
Tab “Information” ... 93
6.5.2 Profinet I/O controller, adding devices ... 93
General ... 93
Enabling/disabling Profinet I/O devices ... 93
Disabling Profinet I/O devices and Profinet I/O modules using UserDefs.cfg ... 94
Devices to connect I/O modules ... 97
Profinet I/O device - Device editor ... 98
Tab “General” ... 98
Tab “Options” ... 101
Tab “IOxS” ... 103
Tab “PNIO parameters” ... 104
Tab “Status” ... 105
Tab “Information” ... 105

Page

6.5.3 Profinet I/O controller, adding modules to device ... 105
General information ... 105
Enabling/disabling Profinet I/O modules ... 106
Tab “General” ... 107
Tab “PNIO parameters” ... 108
Tab “I/O mapping” ... 108
Tab “Status” ... 109
Tab “Information” ... 109
6.5.4 Profinet I/O, controller, scanning devices ... 109
Scanning devices ... 109
Configuration adjustment ... 113
6.5.5 Profinet I/O controller, topology and device exchange ... 114
Topology and device exchange, general information ... 114
Exchanging an I/O device ... 114
Exchanging the device ... 115
PLC supports topology functions and device exchange functions ... 115
6.6 Profinet I/O, device (control as slave) ... 120
6.6.1 Configuring the Profinet I/O device ... 120
Profinet I/O device - Device editor ... 120
Tab “General” ... 122
Tab “IP channel settings” ... 123
Tab “Field bus mapping” ... 123
Tab “PNIO parameters” ... 123
Tab “I/O mapping” ... 123
Tab “Status” ... 123
Tab “Information” ... 124
6.6.2 Profinet I/O device, adding coupling modules ... 124
General information ... 124
Tab “General” ... 125
Tab “PNIO module parameters” ... 125
Tab “PNIO modules I/O mapping” ... 125
Tab “Status” ... 126
Tab “Information” ... 126
6.6.3 Profinet I/O device, controller-device coupling ... 127
General information ... 127
Tab “General” ... 128
Tab “IOxS” ... 129
Tab 'PNIO parameters' ... 130
Tab “Status” ... 130
Tab “Information” ... 131
6.7 Profinet I/O, Tool Calling Interface (TCI) ... 131
6.7.1 Steps to open a device tool from IndraWorks ... 131
6.7.2 Functional properties of the TCI interface in IndraWorks (engineering system) ... 132
Data transfer from IndraWorks (engineering system) to the device tool ... 132
Basic characteristics of TCI ... 132
Data storage ... 132

Page

Communication from the device tool to the Profinet device ... 133
6.7.3 The most important use cases together with TCI ... 133
Closing/exiting IndraWorks project ... 133
Saving an IndraWorks project ... 133
Saving IndraWorks project as ... 133
Archiving an IndraWorks project ... 134
Restoring an IndraWorks project ... 134
6.8 Profinet I/O, FAQs ... 134
6.8.1 Overview ... 134
6.8.2 Where to find the Profinet device description files (GSDML) for control systems? ... 134
6.8.3 Does the IP routing not work between the engineering port of the control and the Profinet network? ... 135
6.8.4 PLC logger message: "PN_Topology <station-name> IP x.x.x.x multiple No-of-Peers n" ... 135
6.8.5 Profinet I/O controller: Renaming PN device does not work (at runtime)? ... 135
6.8.6 Profinet I/O controller: Is ring topology/redundancy supported? ... 136
6.8.7 Profinet I/O device (control as slave): XMx or VPx is not detected sporadically as PN device ... 136

7 EtherNet/IP ... 137
7.1 Features of the EtherNet/IP scanner ... 137
7.2 Features of the EtherNet/IP adapter ... 137
7.3 Features of the EtherNet/IP adapter (engineering port) ... 140
7.4 EtherNet/IP, basics ... 141
7.4.1 General information ... 141
7.4.2 Terms and abbreviations ... 141
7.5 EtherNet/IP, scanner ... 142
7.5.1 Configuring EtherNet/IP scanner ... 142
Overview ... 142
Adding scanner ... 143
Tab “General” ... 144
Tab “EtherNet/IP channel settings” ... 144
Tab “'EthernetIP scanner parameters” ... 144
Tab “I/O mapping” ... 145
Tab “Status” ... 145
Tab “Information” ... 146
7.5.2 EtherNet/IP scanner, adding adapter ... 146
Adding adapter ... 146
Tab “General” ... 146
Tab “Connections” ... 147
Tab “Assemblies” ... 149
Tab “Status” ... 150
7.5.3 "Generic Device" ... 150
7.6 EtherNet/IP, adapter ... 152
7.6.1 Overview ... 152
7.6.2 Tab “Adapter settings” ... 153
7.6.3 Tab “EtherNet/IP channel settings” ... 155
7.6.4 Tab “I/O mapping” ... 155

Page

7.6.5 Tab “Field bus mapping” 156
7.6.6 Tab “EtherNetIP parameters” 156
7.6.7 Tab “Status” 157
7.6.8 Tab “Information” 157
7.6.9 EtherNet/IP scanner-adapter coupling 157
7.7 EtherNet/IP, configuring adapter (engineering port) 158
7.7.1 Overview 158
7.7.2 Tab “Status” 160
7.7.3 Tab “Information” 160
7.7.4 EtherNet/IP adapter (engineering port), modules 161
Tab “I/O mapping” 161
Tab “Information” 161
7.8 EtherNet/IP, FAQs 161
7.8.1 Frequent problems during EtherNet/IP configuration 161
7.8.2 Notes on the EtherNet/IP connection length (O => T / T => O-size) 163

8 EtherCAT 167
8.1 Features of the EtherCAT master 167
8.2 EtherCAT basics 168
8.2.1 Terms and abbreviations 168
8.2.2 Topology 169
8.2.3 Addressing and identification procedure 170
8.2.4 State machine 172
8.2.5 Acyclic communication (mailbox) 173
8.2.6 Telegram structure 175
8.2.7 WorkingCounter 176
8.2.8 SyncUnits 176
8.2.9 Hot Connect 177
8.2.10 Diagnostics 177
General 177
AL status codes 178
CoE SDO abort codes 181
SoE error codes 182
FoE error codes 184
8.2.11 Synchronization 185
General 185
Free run 185
Synchronizing with SM event 186
Synchronization with DC sync 188
8.2.12 EtherCAT slave controller 193
Structure 193
Commands 194
8.3 EtherCAT configuration overview 196
8.4 Configuring EtherCAT master 199
8.4.1 EtherCAT master, setting device 199
8.4.2 Tab “General” 199

Page

8.4.3 Tab “Sync unit assignment” 200
8.4.4 Tab “Distributed Clocks” 201
8.4.5 Tab “Bus diagnostics” 203
8.4.6 Tab “EoE” 205
8.4.7 Tab “CoE” 206
8.4.8 Tab “Diagnostics” 206
8.4.9 Tab “Information” 207
8.5 Configuring EtherCAT master, slaves 207
8.5.1 Adding slaves and modules 207
8.5.2 Tab “General” 208
8.5.3 Tab “SyncManager” 213
8.5.4 Tab “Expert process data” 214
8.5.5 Tab “Process data” 216
8.5.6 Tab “Startup parameters” 216
8.5.7 Tab “EoE settings” 220
8.5.8 Tab “Online” 221
8.5.9 Tab “ESC register” 221
8.5.10 Tab “CoE” 221
8.5.11 Tab “Diagnostics” 222
8.5.12 Tab “I/O mapping” 224
8.5.13 Tab “Information” 225
8.5.14 SoE parameter editor 225
8.6 EtherCAT master, scanning bus 226
8.7 EtherCAT master, bus-synchronous PLC processing 227
8.8 EtherCAT FAQs 228
8.8.1 Overview 228
8.8.2 IndraDrive: How to connect IndraWorks Ds via EtherCAT? 228
8.8.3 IndraDrive: How to configure SoE using Free Run? 229
8.8.4 S20-EC-BK: Bus coupler does not switch to OP (notes on the AL status codes) 231
8.8.5 S20-EC-BK: How to create a startup parameterization for S20 modules? 232

9 Sercos III I/O 233
9.1 Features of the Sercos III IO master 233
9.2 Terms and abbreviations 233
9.3 Configuring Sercos III 234
9.3.1 Configuring Sercos III, overview 234
9.3.2 Sercos III master, overview on bus addresses 235
9.3.3 Sercos III master, Sercos bus diagnostics 236
9.3.4 Sercos III master, cabling and redundancy (example: MLC system) 243
9.3.5 Sercos III master, adding slave 247
Sercos III master; adding slave, general information 247
Enabling/disabling Sercos slaves 249
Disabling Sercos slaves using UserDefs.cfg 250
Commissioning Sercos IO devices/modules at the Sercos Bus 251
Sercos III master, configuring Sercos III slave 252
Tab “General” 252

Page

Tab “Sercos configuration” 254
Tab “Status” 254
Tab “Information” 254
9.3.6 Sercos III master, adding modules to slave 254
Sercos III master, adding modules to slave, general 254
Enabling/disabling Sercos modules 255
Tab “Sercos module” 256
Tab “Function groups” 256
Tab “General inputs and outputs” 257
Tab “User-defined parameters” 258
Tab “Oversampling” 259
Tab “Sercos III modules configuration” 261
Tab “I/O mapping” 262
Tab “Information” 262

10 Modbus RTU 263
10.1 Modbus bus cycle task 263
10.2 Modbus configurator 264
10.3 Modbus master 265
10.3.1 Tab “Modbus serial port - General” 265
10.3.2 Tab “Modbus master - General” 266
10.3.3 Tab “ModbusTCP master - General“ 266
10.3.4 Tab “Modbus slave - General” 266
10.3.5 Tab “ModbusTCP slave - General“ 267
10.3.6 Tab “Modbus slave - Channel” 267
10.3.7 Tab “Modbus slave init” 269
10.4 Modbus slave device 269
10.4.1 Tab “Modbus serial device - General” 269
10.4.2 Tab "ModbusTCP slave device - General” 270
10.5 Tab “Modbus - Parameters” 270
10.6 Tab “I/O mapping” 271
10.7 Tab “Status” 271
10.8 Tab “Information” 271

11 IO-Link 273
11.1 Introduction and overview 273
11.2 Use cases for the engineering of I/O systems 273
11.2.1 Reading in or outputting values using an IO-Link device 273
11.2.2 Initial commissioning of an IO-Link device 274
11.2.3 Process data and IO Device Description (IODD) of IO-Link devices 274
11.2.4 Serial commissioning of machines with IO-Link devices 274
11.2.5 Parameterizing IO-Link devices from the PLC program 275
11.2.6 Diagnostics of IO-Link devices 275
11.2.7 Exchanging IO-Link devices 275
11.3 Profinet I/O IO-Link gateways 276

Page

11.3.1 Introduction and overview ... 276
11.3.2 Function blocks for Profinet I/O IO-Link gateways ... 277
11.3.3 Exemplary use of a Profinet I/O IO-Link gateway of the company Balluff ... 278
11.3.4 Exemplary use of the Rexroth Profinet I/O IO-Link gateway S67E-PN-IOL8-DI4-M12-6P ... 282
11.4 Sercos IO-Link gateways ... 286
11.4.1 Introduction and overview ... 286
11.4.2 Function blocks for Sercos IO IO-Link gateways ... 286

12 Local S20 I/O ... 287
12.1 Features of the local S20 I/O ... 287
12.2 Configuring S20 I/O ... 288
12.2.1 S20 object and S20 Inline modules, overview ... 288
12.2.2 S20 I/O object ... 288
Tab “Settings” ... 288
Tab “S20 module parameters” ... 290
Tab 'Field bus diagnostics' ... 291
Tab “I/O mapping” ... 293
Tab “Status” ... 293
Tab “Information” ... 295
12.2.3 Adding S20 modules ... 295
12.2.4 Enabling/disabling S20 modules ... 296
12.2.5 Disabling S20 modules using “UserDefs.cfg” ... 297
12.2.6 Configuring S20 modules ... 298
General information ... 298
Tab “User-defined parameters” ... 298
Tab “I/O mapping” ... 299
Tab “Information” ... 299
12.2.7 S20 modules, scanning bus ... 300

13 Building automation ... 301
13.1 General information ... 301
13.2 BACnet ... 303
13.2.1 Features of the BACnet implementation ... 303
13.2.2 BACnet basics ... 307
General information ... 307
Further information ... 307
Terms and abbreviations ... 307
Networks and topologies ... 309
Objects and properties ... 310
Services ... 313
Overview ... 313
Updating (COV Notification) ... 314
BACnet COV notification example ... 314
Job prioritization (outputs) ... 321
Alarm handling ... 322

Page

Data logging (TrendLog) ... 325
Device profiles ... 326
BACnet error codes ... 327
BACnet status codes ... 330
13.2.3 Configuring the BACnet server ... 333
Commissioning the BACnet server ... 333
Creating BACnet servers and objects and starting a BACnet server ... 333
BACnet plugin - Persistence ... 339
BACnet plugin - Schedule ... 340
BACnet plugin - Calendar ... 341
BACnet plugin - Notification Class ... 341
BACnet plugin - Trend Log ... 341
PLC interface to the object properties ... 341
Exporting and importing a BACnet configuration in the EDE format ... 343
13.2.4 Configuring the BACnet/IP client ... 343
13.2.5 Notes on implemented BACnet objects and services ... 345
General information ... 345
Object - Property - Actions ... 346
Services with implementation of standard actions ... 348
Services with internal implementations ... 356
Services without implementation of standard actions ... 357
Examples on how to implement “Service Indication Hooks” ... 357
Implementation example "CreateObject" service hook ... 357
Implementing “Logging” objects (EL/TL/TLM) ... 360

14 Service and support ... 363

Index ... 365

1 About this documentation

Editions of this documentation

Edition	Release date	Note
01	2019-07	First edition for version 15VRS **New**: • Chapter BACnet, see chapter 13.2 "BACnet" on page 303
02	2019-10	**New**: • chapter "BACnet error codes" on page 327 • chapter "BACnet status codes" on page 330 • "**BACnet server settings**" on page 338 **Revision**: • chapter 13.1 "General information" on page 301 • chapter 13.2.1 "Features of the BACnet implementation" on page 303 • S20 module operation modes, see chapter 12.2.2 "S20 I/O object" on page 288
03	2020-06	**Revision**: • chapter 13.2.3 "Configuring the BACnet server" on page 333 • chapter 8.5.6 "Tab “Startup parameters”" on page 216
04	2020-11	**Revision**: • chapter 7.2 "Features of the EtherNet/IP adapter" on page 137 • chapter 13.2.3 "Configuring the BACnet server" on page 333
05	2021-03	**New**: • chapter "BACnet COV notification example" on page 314 **Revision**: • chapter "Further information" on page 307 • chapter 13.2.4 "Configuring the BACnet/IP client" on page 343

Tab. 1-1: Change Record

1.1 Validity of the documentation

Target group

This documentation describes the field bus and local periphery connections supported by the MLC and MTX systems.

The focus of this documentation lies in the configuration, parameterization, commissioning and diagnostics of different periphery connections.

The online help is based on this documentation.

Required qualification: Individual who is able to assess the tasks assigned and to identify possible safety risks owing to qualification in the subject, knowledge and experience. The individual should also be familiar with the standards and regulations.

Application phases

This documentation supports the user during the following phases:

- Configuration of field bus components in IndraWorks
- Field bus application in the PLC user program.

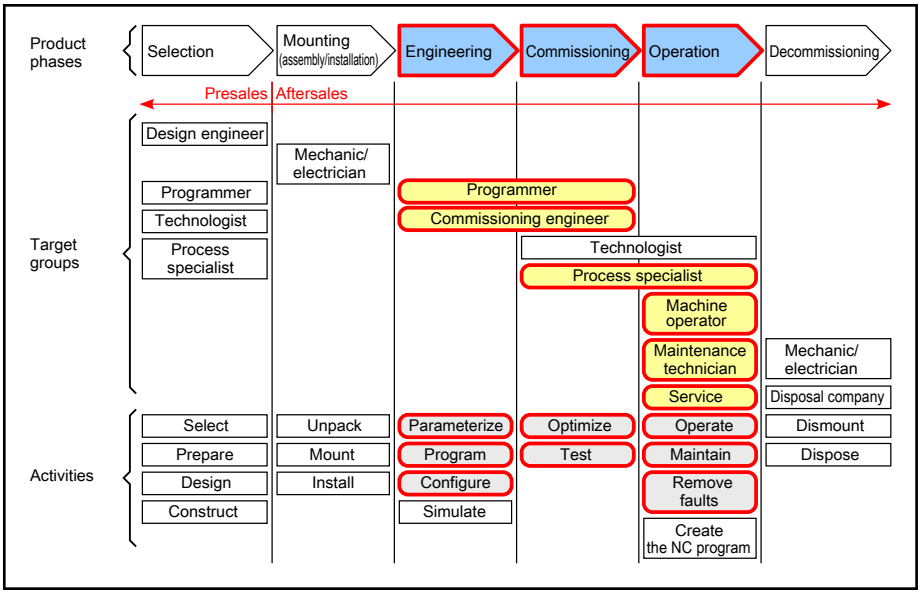


Fig. 1-1: Application phases

1.2 Using safety instructions

1.2.1 Structure of the safety instructions

The safety instructions are structured as follows:

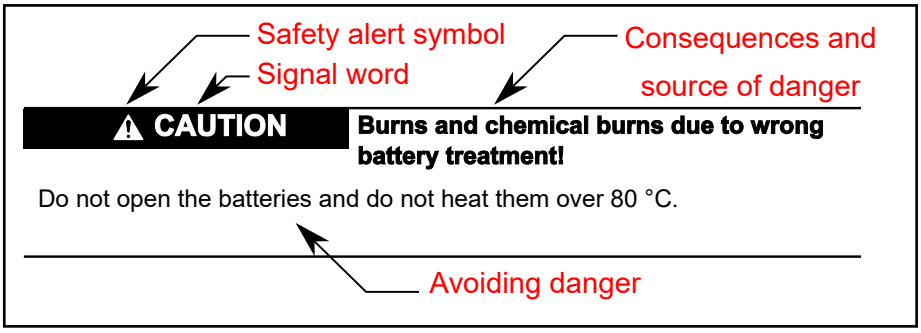


Fig. 1-2: Structure of the safety instructions

1.2.2 Explaining Signal Words and Safety Alert Symbol

The safety instructions in this documentation contain specific signal words (danger, warning, caution, notice) and, if necessary, a safety alert symbol (according to ANSI Z535.6-2006).

The signal word draws attention to the safety instruction and indicates the risk potential.

The safety alert symbol (triangular safety reflector with exclamation marks), preceding the signal words Danger, Warning, Caution indicates hazards for persons.

⚠ DANGER

In case of non-compliance with this safety instruction, death or serious injury **will** occur.

⚠ WARNING

In case of non-compliance with this safety instruction, death or serious injury **can** occur.

⚠ CAUTION

In case of non-compliance with this safety instruction, minor or moderate injury can occur.

NOTICE

In case of non-compliance with this safety instruction, material damage can occur.

1.2.3 Symbols Used

Pointers are displayed as follows:

This is a note.

Tips are displayed as follows:



This is a tip.

1.2.4 Explaining the signal alert symbol on the device



If this symbol is on your device, you have to observe the documentation on the device. The respective documentation informs on the type of hazard as well as the steps required to avoid this hazard.

1.3 Required and supplementing documentation

The documents listed below contain additional information on this subject.

Rexroth IndraWorks 15VRS IndraLogic 2G PLC Programming System DOK-IWORKS-IL2GPRO*V15-APRS-EN-P, R911396137 This documentation describes the PLC programming tool IndraLogic 2G and its use. The documentation includes the basic use, first steps, visualization, menu items and editors.
Rexroth IndraWorks 15VRS - Field Bus Libraries DOK-IWORKS-FB*LIB**V15-LIRS-EN-P, R911393275 This manual describes the field bus libraries: RIL_ProfibusDP_02, RIL_ProfibusDPSlave, RIL_ProfinetIO, RIL_ProfinetIO-Device, RIL_EtherNetIPAdapter, RIL_MappingList, RIL_SERCOSIII, RIL_Inline including their diagnostics and error reactions of the function blocks.
Rexroth IndraWorks 15VRS Basic Libraries IndraLogic 2G DOK-IL*2G*-BASLIB**V15-LIRS-EN-P, R911398633 This documentation describes the system-comprehensive PLC libraries.
Rexroth IndraMotion MLC 15VRS PLCopen Libraries DOK-XLCMLC-FUNLIB**V15-LIRS-EN-P, R911393307 This documentation describes the function blocks, functions and data types of the RIL_CommonTypes, ML_Base and ML_PLCopen libraries. The documentation also includes the error reactions of the function blocks.
Automation Terminals of the Rexroth Inline Product Range DOK-CONTRL-ILSYSINS***-AWRS-EN-P, R911317021 This documentation describes the automation terminals of the Rexroth Inline product family.
Rexroth Inline Bus Coupler for Sercos with Digital Inputs and Outputs R-IL S3 BK DI8 DO4-PAC DOK-CONTRL-ILS3BKDI8DO-KBRS-DE-P, R911170560 The bus coupler is intended to be used in a Sercos® network of the third generation and is a link to the Inline installation system. It is additionally used to detect and output digital signals.

Rexroth Inline Bus Coupler for Profibus DP with Digital Inputs and Outputs R-IL PB BK DI8 DO4-PAC DOK-CONTRL-ILPBBKDI8DO-AWRS-EN-P, R911170476 The bus coupler links the Profibus DP and the Inline installation system. It is additionally used to detect and output digital signals.
Rexroth Inline Bus Coupler for PROFINET with Digital Inputs and Outputs R-IL PN BK DI8 DO4-PAC DOK-CONTRL-ILPNBKDIO**-DARS-EN-P, R911328682 The bus coupler links a PROFINET network and the Inline installation system. It is additionally used to detect and output digital signals.
Rexroth Inline Bus Coupler for Ethernet with Digital Inputs and Outputs R-IL ETH BK DI8 DO4 2TX-PAC DOK-CONTRL-ILETHBKDIO*-DARS-EN-P, R911328680 The bus coupler links a Ethernet network and the Inline installation system. It is additionally used to detect and output digital signals.
Rexroth IndraControl S67 Profibus Coupler 8 Digital Inputs (M8) DOK-CONTRL-S67PBBKDI8*-APRS-EN-P, R911329566
Rexroth IndraControl S67 Ethernet/IP Coupler 8 digital Inputs (M8) DOK-CONTRL-S67ETBKDI8*-APRS-EN-P, R911329564
Rexroth IndraControl S67 Profinet Coupler 8 Digital Inputs (M8) DOK-CONTRL-S67PNBKDI8*-APRS-EN-P, R911329568
Rexroth IndraControl S67 Sercos Coupler 8 Digital Inputs (M8) DOK-CONTRL-S67S3BKDI8*-APRS-EN-P, R911338401
Rexroth IndraControl S20 Bus Coupler for Profinet IO Controller board modular extensions possible using S20 modules DOK-CONTRL-S20*PN*BK**-DARS-EN-P, R911335968
Rexroth IndraControl S20 Bus Coupler for Sercos Controller board modular extensions possible using S20 modules DOK-CONTRL-S20*S3*BK**-DARS-EN-P, R911335970

2 General information

2.1 Terms and abbreviations

The following figure shows the field bus-dependent names of the master or slave interfaces of the field buses:

Terms of the field bus interfaces

Field bus	Master interface	Slave interface
Profibus DP	Master	Slave
Profinet I/O	Controller	Device
EtherNet/IP	Scanner	Adapter
EtherCAT	Master	Slave
Sercos III	Master	Slave

Terms of the data transfer mechanisms

Field bus	Cyclic transfer	Acyclic transfer
Profibus DP	DP/V0	DP/V1
Profinet I/O	RTA (real-time cyclic)	RTA (real-time acyclic)
EtherNet/IP	implicit messaging (polled I/O)	explicit messaging
EtherCAT	Cyclic data/process data	Different mailbox protocols
Sercos III	Cyclic data	Service channel + IP channel

The local I/O bus (Inline or S20) at the control is a special feature. It can be compared with a field bus master with exactly one field bus slave to which I/O modules can be connected.

Device description files

The device description file describes the properties of the field bus device. This file contains all relevant data important for the engineering as well as for the data exchange with the I/O device. The following file formats are specified for the different field buses:

GSD file

This is a device description used by IndraWorks or another engineering system to make the Profibus DP slaves known.

GSDML file

This is a device description used by IndraWorks or another engineering system to make the Profibus I/O devices known.

EDS file

This is the device description with which the EtherNet/IP adapter is made known to IndraWorks or another engineering system (EDS: Electronic Data Sheet).

ESI file

This is a device description used by IndraWorks or another engineering system to make the EtherCAT slaves known (ESI: EtherCAT Slave Information).

SDDML file

This is a device description used by IndraWorks or another engineering system to make the Sercos III slaves known acc. to I/O profile (SDDML: Sercos Device Description Markup Language).

Field bus connection

Bus coupler

Field bus connection (slave, device, adapter),

Compact

- For a compact slave, the module structure is fixedly specified.

  The modules below the "Slave" object node are already present. No more modules are available in the library

Modular

- The module structure of the slave is variable. The modules (terminals) can be arranged individually, but according to the device mounting specifications.

Module, terminal

Smallest I/O unit to assemble a field bus connection and/or the Inline bus.

2.2 Transmission modes

The transmission modes can be divided into the following categories:

- Cyclic transmission channel
- Acyclic transmission channel

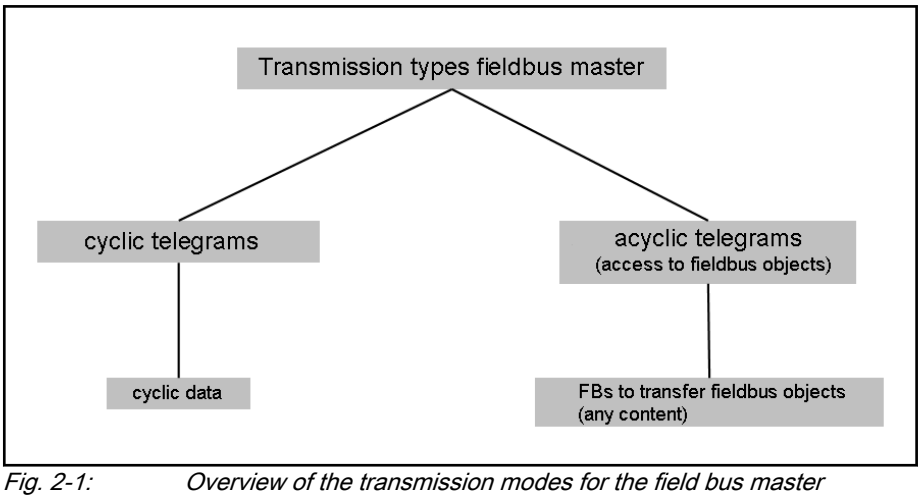


Fig. 2-1: Overview of the transmission modes for the field bus master

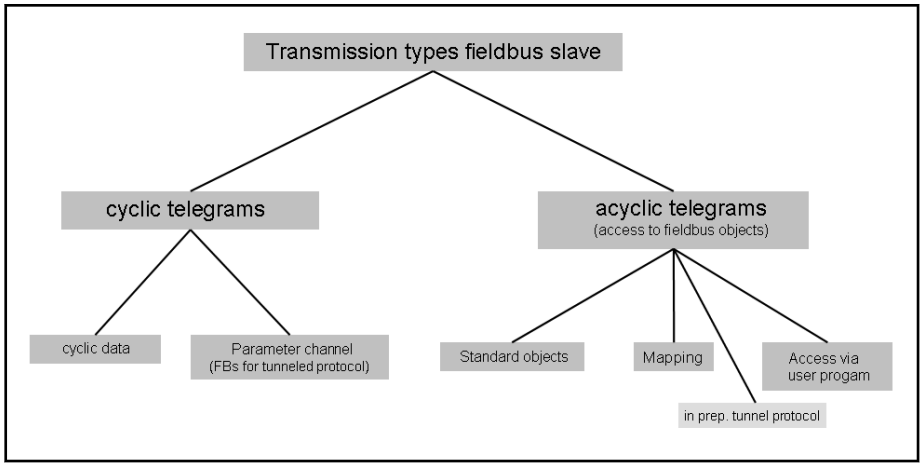


Fig. 2-2: Overview of the transmission modes for the field bus slaves

The field bus objects are described using the following elements:

- Profibus: Slot (8 bits), index (8 bits)
- Profinet IO: Slot (16 bits), subslot (16 bits), index (16 bits)
- EtherNet/IP: Class (8 bits), instance (8/16/32 bits), attribute (16 bits)
- Sercos III: EIDN (32 bits) with parameter number (16 bits), structural element SE (8 bits), structural instance SI (8 bits)

3 Field bus interfaces and characteristics

3.1 Field buses and field bus communication

As platform for the MLC and MTX systems, the controls of the XM series provide different interfaces to connect I/O components and field bus devices. The use of interfaces can be specified via IndraWorks Engineering during the configuration.



When using Safety, use the "First Steps" documentation on SafeLogic and SafeLogic compact.

Rexroth IndraWorks SafeLogic 14VRS First Steps, DOK-IWORKS-SL*STEP*V14-CO02-EN-P, R911341520

Rexroth IndraWorks SafeLogic Compact 14VRS, First Steps, DOK-CONTRL-SLC*F*STEP*-CO02-EN-P, R911346744

3.1.1 Field bus interfaces at the XM control system

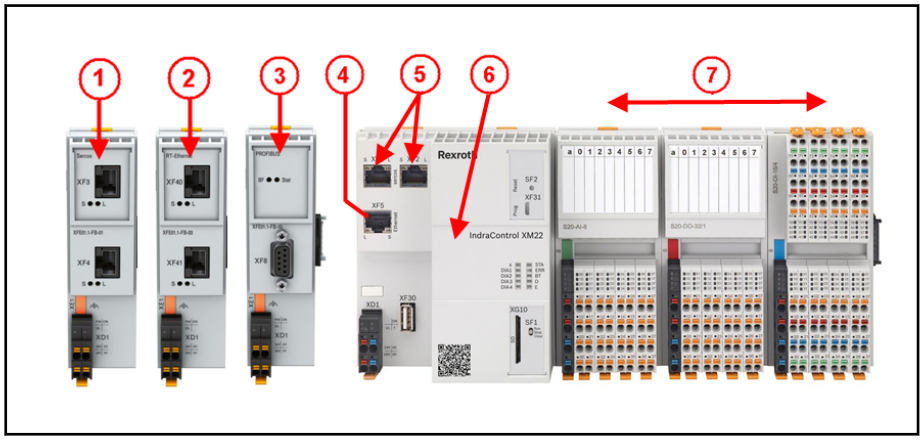


Fig. 3-1: XM2 control with field bus and I/O coupling

No.	Function	Description
1, 2, 3	Extension modules	Up to three extension modules can be connected to the left side of the XM2 control using the function module plug connected there *(1) Extension module Sercos III (XFE01.1-FB01)* • Being prepared *(2) Extension modules RT-Ethernet (XFE01.1-FB03)* • Profinet I/O controller or device, EtherNet/IP scanner or adapter *(3) Extension module Profibus (XFE01.1-FB10)* • Profibus DP master or slave
4	Ethernet interface XF5	*Standard Ethernet interface for a TCP/IP network* • Programming and visualization device interface • BACnet server and client • Ethernet/IP adapter at the engineering port
5	Ethernet interfaces XF1/XF2	Ethernet interfaces to connect the Sercos III drives and I/O These Ethernet interfaces can also be used for EtherCAT

No.	Function	Description
6	XM2	Control with power supply (XD1), diagnostic LEDs and memory card (SD), USB host (XF30), USB device (XF31)
7	Rexroth S20 I/O modules	In the I/O area of the XM2 control, up to 63 S20 modules can be connected via S20 bus base modules. The number of connectable modules depends on the total current consumption of the S20 modules. The XM2 control can provide up to 2 A for the power supply of the S20 modules. For more information on the connection of Rexroth S20 terminals, refer to the documentation "Automation Terminals of the Rexroth S20 Product Family", DOK-CONTRL-S20*SYS*INS-APxx-EN-P, R911335988, xx - Edition

Tab. 3-1: Ports and operating elements of the XM2 control

XM control - Field buses and local bus

- A Profibus DP master can be enabled on an attached extension module (XFE01.1-FB10)
  chapter 5.5.1 "Profibus DP master, configure" on page 48
- A Profibus DP slave can be enabled on an attached extension module (XFE01.1-FB10)
  chapter 5.5.2 "Profibus DP master, configuring slaves" on page 56
- A Profinet I/O controller can be enabled on an attached extension module (XFE01.1-FB03)
  chapter 6.5.1 "Profinet I/O controller, configuring" on page 84
- A Profinet I/O device can be enabled on an attached extension module (XFE01.1-FB03)
  chapter 6.5.2 "Profinet I/O controller, adding devices" on page 93
- A EtherNet/IP scanner can be enabled on an attached extension module (XFE01.1-FB03)
  chapter 7.5 "EtherNet/IP, scanner" on page 142
- A EtherNet/IP adapter can be enabled on an attached extension module (XFE01.1-FB03)
  chapter 7.6 "EtherNet/IP, adapter" on page 152
- A Sercos III master interface for I/O or field devices is the onboard Sercos III interface. A Sercos III extension module is being developed
  chapter 9 "Sercos III I/O " on page 233
- S20 I/O modules can be connected at the right of the XM2 control.
  chapter 12 "Local S20 I/O" on page 287
- The EtherCAT master can also be used on the onboard Ethernet interface XF1/XF2 instead of Sercos
  chapter 8.1 "Features of the EtherCAT master" on page 167

3.2 Field bus features

3.2.1 Field bus master

XM2 control

	Profibus DP master [8]	Profinet I/O controller [9]	EtherNet/IP scanner [9]	EtherCAT master	Sercos III master	S20 local
Max. number of devices	125	64 / 128 [1, 2]	64 [3]	128	256[5]	64 modules
Max. number of cyclic input data of the master	3584 bytes	5652 bytes	4096 bytes	2048 bytes [10]	5600 bytes	1536 bytes[7]
Max. number of cyclic output data of the master	3584 bytes	5700 bytes	4096 bytes	2048 bytes [10]	5600 bytes	1536 bytes[7]
Max. number of cyclic input data per slave	244 bytes	1440 bytes	480 bytes[4]	2048 bytes [10]	5600 bytes	
Max. number of cyclic output data per slave	244 bytes	1440 bytes	480 bytes[4]	2048 bytes [10]	5600 bytes	
Max. telegram data per acyclic telegram	240 bytes	1392 bytes	450 bytes		1500 bytes	

1 For details, refer to chapter 6.1 "Features of the Profinet I/O controller" on page 75

2 The cycle time (best case: 1 ms) depends on the number of devices and the number of I/O data. If there is an overload of the PNIO controller, connections are aborted in the cyclic communication. This can be seen at a rising "ErrorCount" in PNIO controller "Status" dialog. In this case, increase the cycle time ("Send Clock" or "Reduction Ratio") of the devices.

3 If only one connection is set up for each adapter, 64 adapters are possible. Multiple connections can be set up to one adapter. The maximum number of possible adapters reduces accordingly.

4 Per connection. Multiple connections per adapter possible.

5 The driver supports 256 I/O resources. Depending on the control hardware, this number is reduced by individual system interfaces.

7 Input data and output data are located together. The maximum number is 1536 bytes

8 Profibus (XFE01.1-FB10)

9 RT-Ethernet (XFE01.1-FB03)

10 The maximum number of process data of the EtherCAT master is 2048 bytes (= Sum of the cyclic input/output data). There is no restriction with regard to input/output data of a slave.

Tab. 3-2: Features of the field bus master interfaces

3.2.2 Field bus slaves

XM2 control

	Profibus DP slave [1]	Profinet I/O device [2]	EtherNet/IP adapter [2]	EtherNet/IP adapter (Eng. port)
Max. number of cyclic input data of the slave	244 bytes	1024 bytes[3]	480 bytes	128 bytes
Max. number of cyclic output data of the slave	244 bytes	1024 bytes[3]	480 bytes	128 bytes + 4 bytes header
Consistency of cyclic data	128 bytes	128 bytes	128 bytes	128 bytes
Max. telegram data per acyclic telegram	240 bytes	1092 bytes[3]	450 bytes	

1 Profibus (XFE01.1-FB-10 or XFE01.1-FB-11)
2 RT-Ethernet (XFE01.1-FB-03 or XFE01.1-FB-04)
3 For details, refer to chapter 6.2 "Features of the Profinet I/O device (control as slave)" on page 76

Tab. 3-3: Features of the field bus slave interfaces

4 Field bus-comprehensive features

4.1 Device database

4.1.1 Device database, overview

The device database supplements the device library and contains device descriptions installed on the local systems to provide them for projects in IndraWorks. The properties of the respective (e.g. parameters, vendor, device name) are stored.

There are different formats for different devices

- **Profibus DP:** GSD file, *.gs?,
- **Profinet IO:** GSD file, GSDML*.xml,
- **EtherNet/IP:** EDS file, *.xml,
- **EtherCAT:** ESI file, *.xml
- **Sercos III IO:** SDDML file, *.xml.
- **S20 I/O:** SDDML file (corresponds to the Sercos S20 module), *.xml

The device database is installed when installing IndraWorks.

Additional devices can be installed and devices, which are not required anymore, can be uninstalled in the device database.

The device description files are automatically backed up with an IndraWorks project archive.

4.1.2 Managing devices

Device database, dialog

Icon: 

Use the "Device Database" command to install and uninstall device description files. In the main menu, click on **Tools** ▸ **Device Database...** to open the "Device Database" dialog:

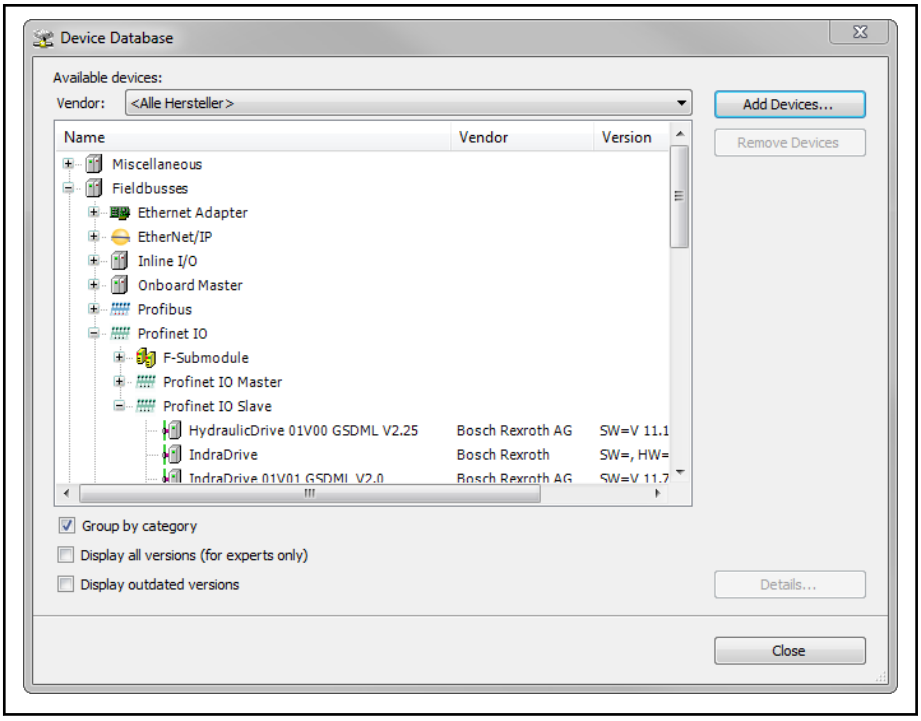


Fig. 4-1: Device database

Available devices

The currently installed devices are listed under "Available devices".

Set the "Vendor" filter to restrict the list.

Group by category: Devices are hierarchically structured, that is controls in the upper part, then function modules below, then integrated interfaces, etc. Otherwise, the individual objects are arranged alphabetically.

Normally, the devices matching the IndraWorks version are displayed. For a complete overview, press **Display all versions (for experts only)**. To display outdated versions, select **Display outdated versions**.

Device can be added or removed under "Available devices".

After selecting a device, more information is available under "Details".

Adding devices

To add devices to IndraWorks, proceed with the following steps:

1. Unzip the complete ZIP archive with the device description files and the respective icons.

   ⚠ WARNING

   If available, unzip all device description files including the iParSDD files!

   If this is not done, the editor “Safe parameterization” is not available anymore in the Safety applications and it is possible that the Safety I/O configurations do not work anymore.

2. In the main menu, click on **Tools ▸ Device Database...** to open the "Device Database" dialog.
3. Click on "Add Devices..." in the "Device Database" dialog.

   The file selection dialog "Install Device Descriptions" opens:

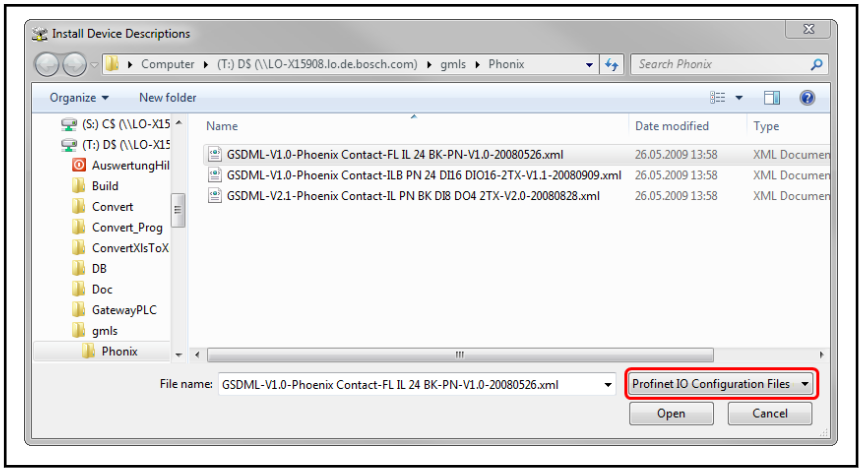


Fig. 4-2: Installing device descriptions

4. Select the device description files to be installed.

☞ Select the correct file type in the drop-down list "File type".
All related files, e.g. device icons are added automatically.

---

5. Click on "Open" to start the installation.
   After the installation, the imported devices are displayed and highlighted under "Available devices":

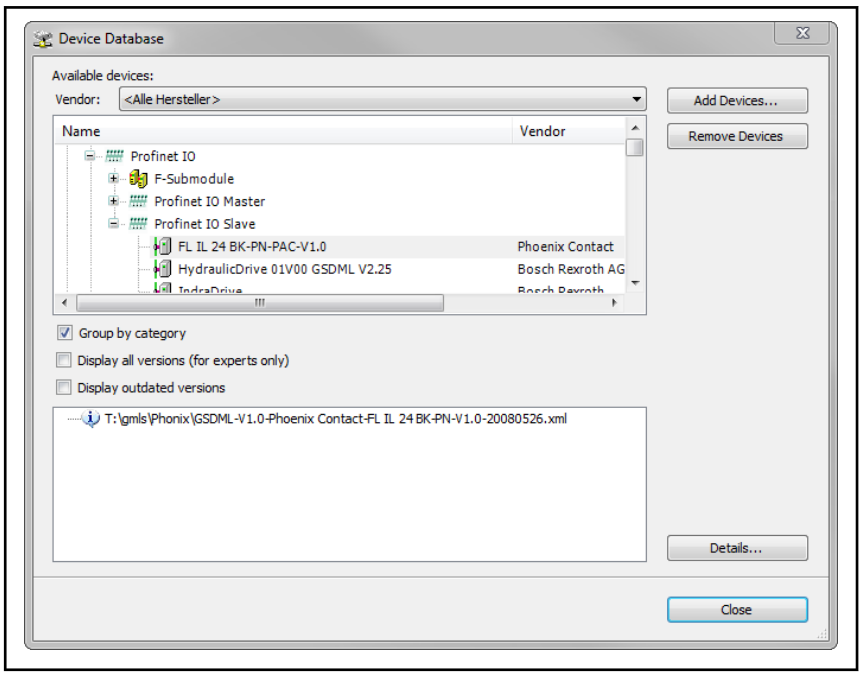


Fig. 4-3: Device descriptions installed

6. Select the device and further information is provided under "Details".

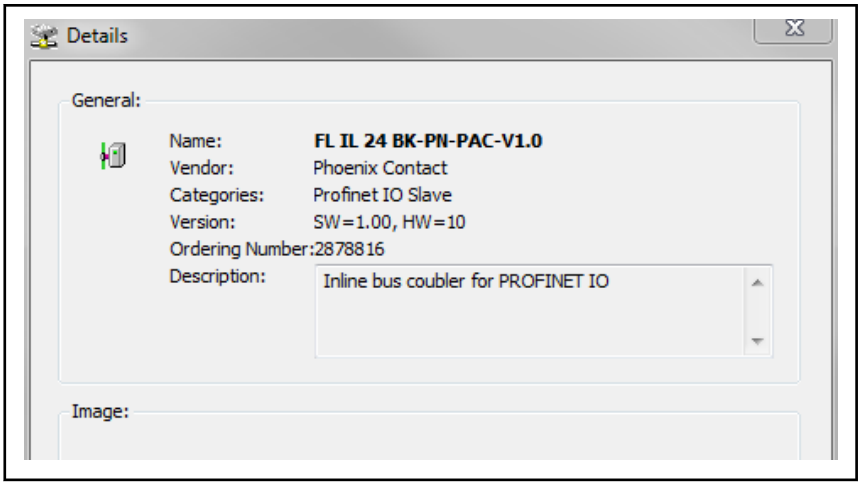


Fig. 4-4: Details on the device description file

7. To close the "Device Database" dialog, press "Close".

   The newly installed devices are also displayed in the library:

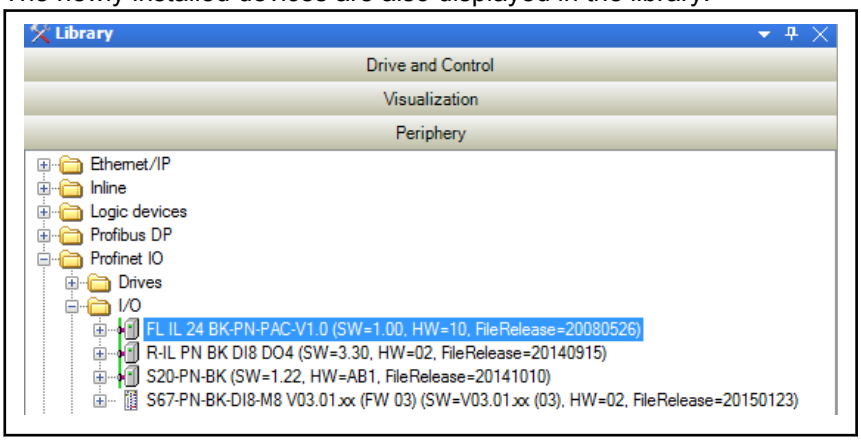


Fig. 4-5: Imported device in the library

Removing devices

To remove the devices from IndraWorks, proceed with the following steps:

1. In the main menu, click on **Tools ▸ Device Database...** to open the "Device Database" dialog.
2. Highlight the devices to be removed in the device tree.
3. Click on "Remove Devices".

   The "Confirm Device Removal" dialog opens

4. Click in "Yes" to remove the selected device.

   The device is removed from the device database and from the library. (will take a while...)

5. To close the "Device Database" dialog, press "Close".

☞ S20 I/O modules are located in the device database in the following directory: ▸Field buses ▸Sercos ▸Module ▸S20*.

Note that the removed modules are not available at the Sercos bus coupler S20-S3-BK+.

4.2 Mapping of the field bus inputs and outputs

These explanations apply equally to the following bus systems:

- Inline (via field bus coupler)
- S20
- Profibus DP
- Profinet I/O
- EtherNet/IP
- Sercos III I/O

The module begins with the address "%IB0" or "%IW0" for the inputs and analog "%QB0" or "%QW0" for the outputs.

The other modules follow accordingly, even if they already belong to the subsequent bus in the Project Explorer.

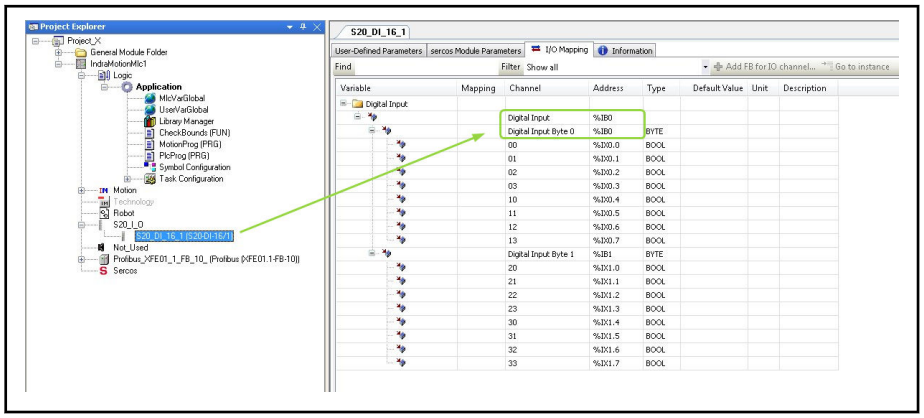


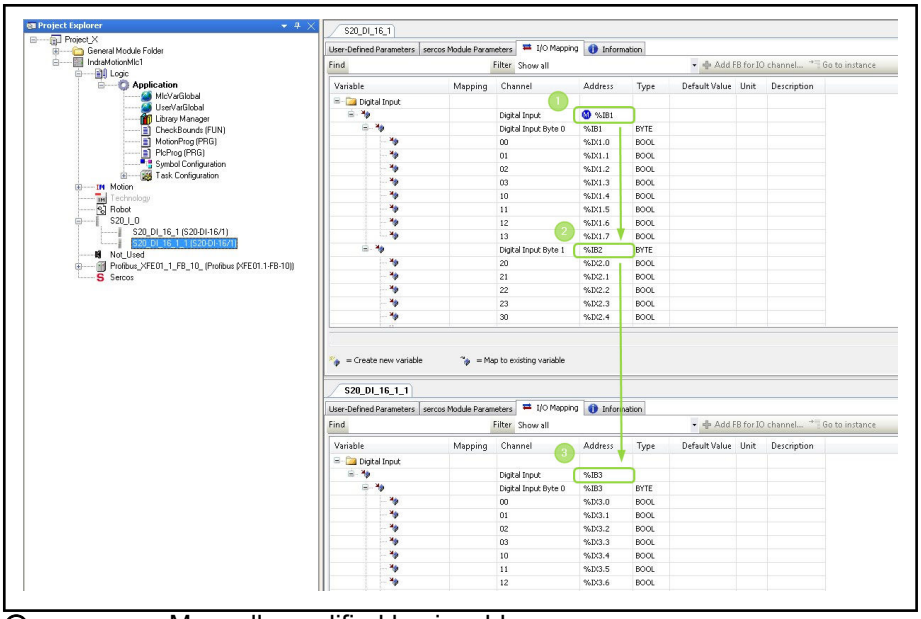
Fig. 4-6: Address allocation

However, users can still assign the addresses as desired.

Clicking in the respective address field allows the address for the current module to be edited. The others follow suit.

An address modified in this way is identified by the white "M" on a blue background.

The other modules follow accordingly, even if they already belong to the subsequent bus in the Project Explorer.



① Manually modified basic address
② Automatically assigned subsequent address
③ Automatically assigned address of the next I/O module

Fig. 4-7: Modified basic address

With multiple mapping, there is a risk that the address ranges could overlap.

This is not detected until the **code is generated**.

The error message contains only the second assigned position....

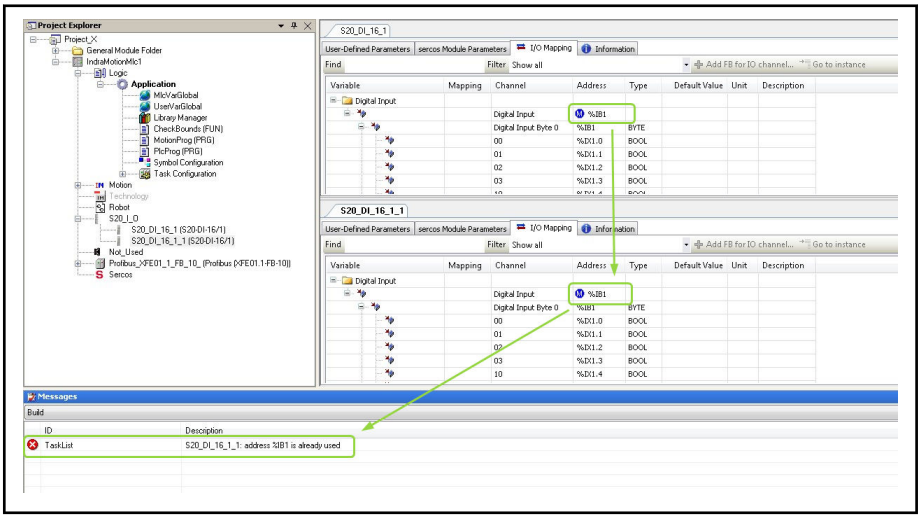


Fig. 4-8: Mapping contains an error, addresses used more than once

I/O modules of the same type on the respective slot, in the same sequence, are the prerequisite for automatic address generation results in the same configuration and then to the connection of the correct inputs and outputs.

Please check this carefully if the control is exchanged.

Mapping local variables:

- **Variable**:

  This column displays the input or output module. The plus and minus symbols allow you to switch between bit and byte display.

  For each absolute address, a symbolic address can be assigned (double-click on the respective field).

  For Map to existing variable the complete path must be entered, i.e.

  `<ApplicationName> <ProgramName> <VariableName>`.

  Example (figure below):

  `Application.MotionProg.x_in_1`,

  `Application.MotionProg.x_in_2`

  or

  `Application.MotionProg.x_out_1`,

  `Application.MotionProg.x_out_2`.

- **Mapping**:

  The  icon is displayed in the Mapping column.

  Subsequently, the address value is displayed with a strike through, i.e. the %Qx.x-/ or %Ix.x address.

- **Type:**

  Byte address are identified with "BYTE" and bit address with "BOOL".

- **Current value:**

  Physical status of the input/output. The status is displayed only in the diagnostic mode for communication between IndraWorks and the control.

- **Unit**:

  Unit for the parameter value, e.g. "ms" for milliseconds.

- **Description**:

  Enter any comment on an address.

Mapping global variables:

- **Variable**:

  This column displays the input or output module. The plus and minus symbols allow you to switch between bit and byte display.

  For each absolute address, a symbolic address can be assigned (double-click on the respective field).

  When mapping, **only the name of the variable** can be entered, i.e.

  `<Variable name>`.

  Example (figure below):

  `x_in_3` or

  `x_out_3`, `x_out_4`.

- **Mapping**:

  The  icon is displayed in the Mapping column.

The variable is entered as VAR_GLOBAL in the "IoConfig_Globals_Mapping" list.

- **Type:**

  Byte address are identified with "BYTE" and bit address with "BOOL".

- **Current value:**

  Physical status of the input/output. The status is displayed only in the diagnostic mode for communication between IndraWorks and the control.

- **Unit**:

  Unit for the parameter value, e.g. "ms" for milliseconds.

- **Description**:

  Enter any comment on an address.

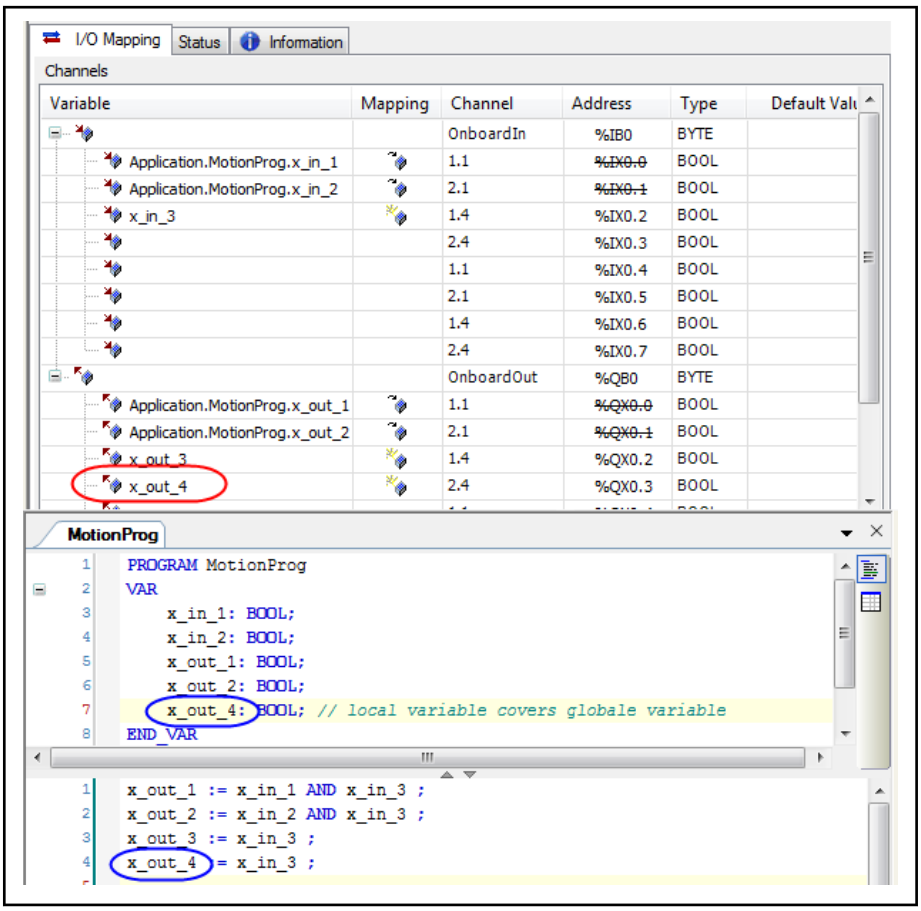


Fig. 4-9: IO mapping, variable declaration and example program (offline)

The declaration section of the POU "MotionProg" contains

- the local POU variables,
- the variables that are to be mapped as inputs and outputs (without AT construction!).

If you declare a local variable of the same name, as with the global list the local variable will be used!

Example: `x_out_4`.

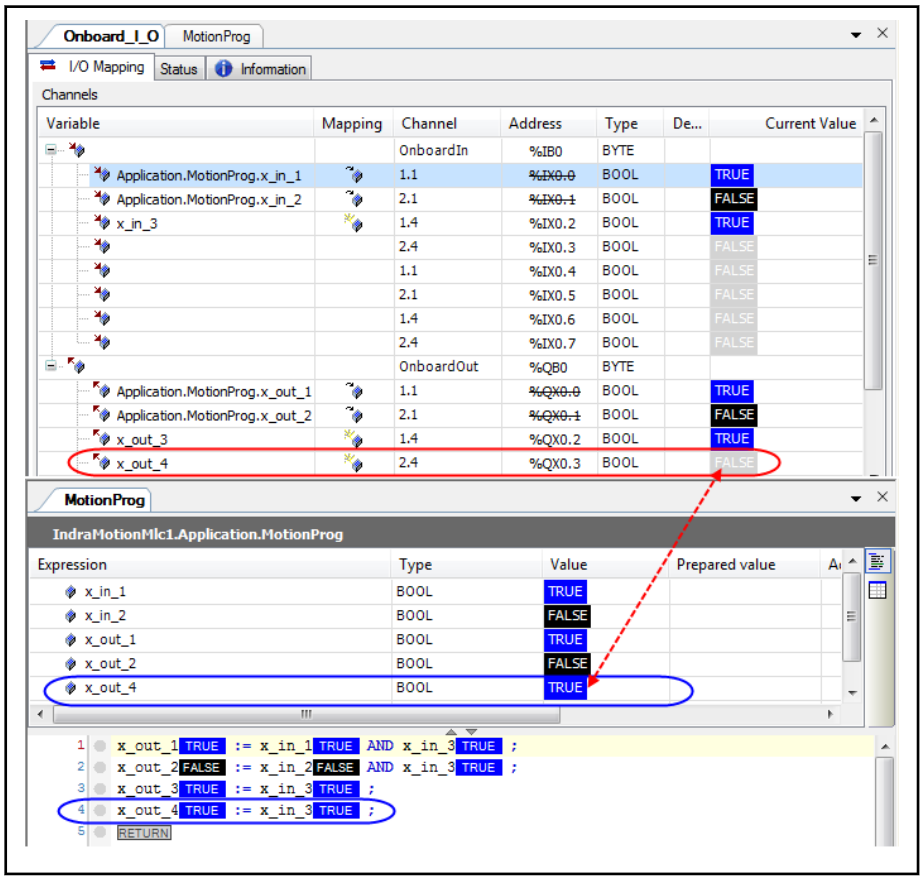


Fig. 4-10: IO mapping, variable declaration and example program (online)

Reset Mapping: Use this button to reset the mapping settings to the default values defined in the device description file.

Update variables: Defines for the module whether I/O variables are updated in the bus cycle task.

The default value is defined in the device description file.

- **Use parent device setting:** Update according to the settings of the superordinate device.
- **Enabled 1 (use bus cycle task if not used in any task):** The I/O variables are updated in the bus cycle task if not used in any other task
- **Enabled 2 (always in the bus cycle task):** All variables are updated in each cycle of the bus cycle task irrespective of whether they are used or mapped on an input or output channel.

4.3 Field bus mapping, access from field bus master to variables of the field bus slave

Basics Use the field bus mapping to access control variables from a parent field bus master.

The following field buses are supported:

- Profibus DP
- Profinet I/O
- EtherNet/IP

To access a control variable from a parent master, assign an address to these variables first (mapping). The parent master can use this address to access the control data using acyclic services (Field bus slaves, page 14).

Variables can be mapped via the "Field bus mapping" tab of the "Field Bus Slave" dialog or via the function blocks of the RIL_MappingList.library. The configurator dialog "Field Bus Mapping" is now described.

To use the field bus mapping, configure the control as field bus slave.

The individual steps are shown using the Profinet I/O. The steps are similar for the other buses.

Configuration

The "Field Bus Mapping" tab is located in the device editor of the respective connection. Addresses can be assigned to individual variables in a table.

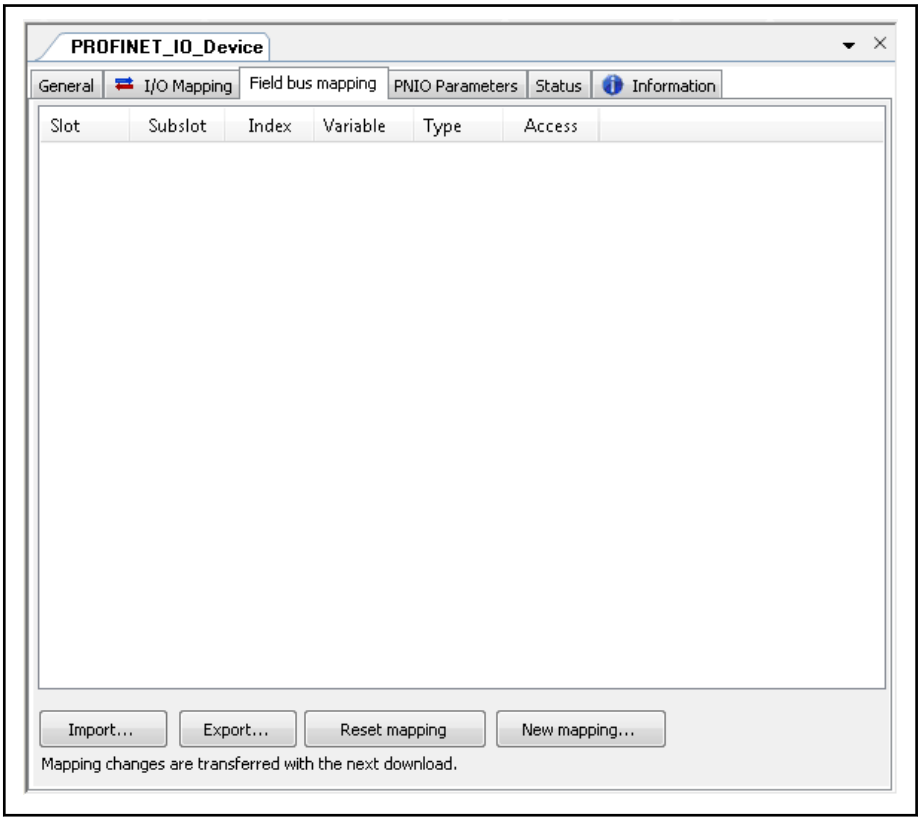


Fig. 4-11: Profinet I/O device: Field bus mapping

Creating new mapping

A new entry can be added via the "New Mapping..." button. A dialog opens to select a variable.

Variables declared in the application can be selected:

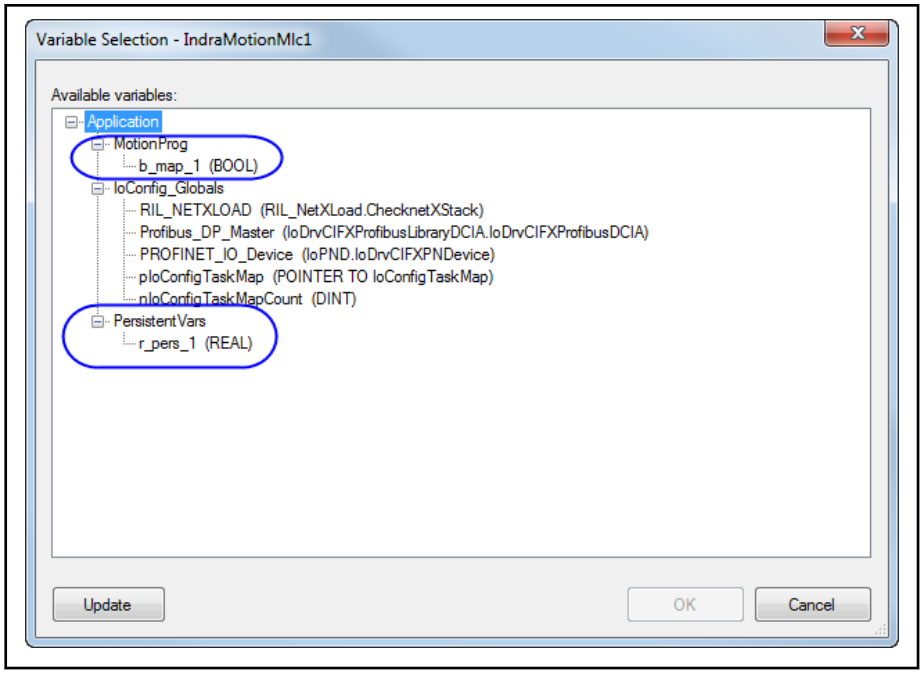


Fig. 4-12: Profinet I/O device: Field bus mapping, selecting variables

If the selection is applied with "OK", a new entry is created in the table for the selected variable.

In addition, the variable is added to the symbol configuration.

The variable selection only contains variables present until the latest compilation. If new variables were added to the PLC program since that time, they are only visible after a new compilation. Therefore, click on the "Update" button.

Address, variable and access can be subsequently changed by clicking on the corresponding table cell. Gray columns cannot be edited.

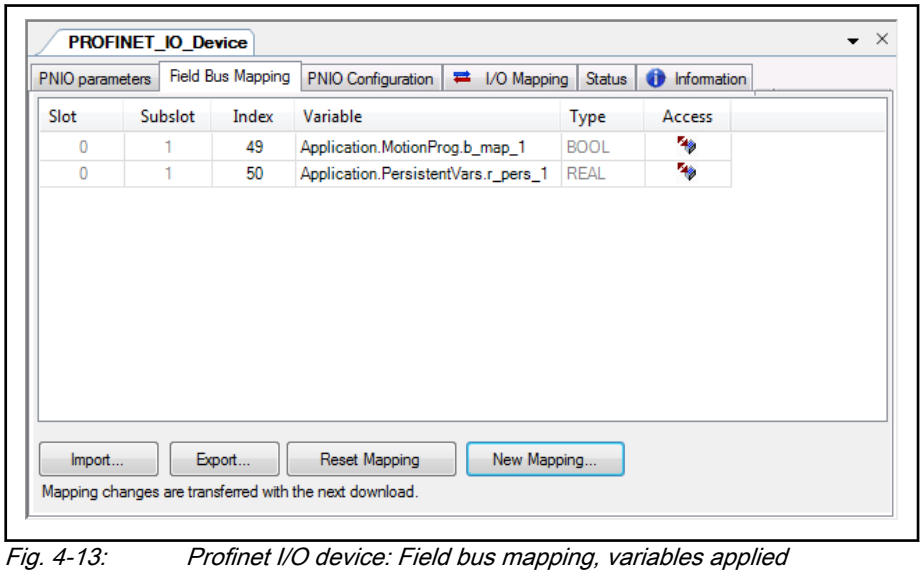


Fig. 4-13: Profinet I/O device: Field bus mapping, variables applied

The address can be specified in the first columns in the table, but the values in the grayed out fields cannot be changed.

- The address of **Profibus DP** consists of **slot** and **index**. The slot is always 0. The index has to be within the range from 49 to 254.

- The address of **Profinet I/O** consists of **slot**, **subslot** and **index**. The slot is always 0 and the subslot is always 1. The index has to be in the range from 49 to 32767.
- For **EtherNet/IP**, the address consists of **class**, **instance** and **attribute**. In this case, the class is always 199; the instance has to be between 2 and 65535 and the attribute is always 1.

The **Variable** column displays the instance path of the variables.

The **Type** column displays the data type of the variables.

The **Access** column displays the access rights for the variables.

Reset mapping The "Reset Mapping" button can be used to delete all table entries.

Import/export To save an existing mapping to a file, use the "Export…" button. It can be loaded later using the "Import…" button.

This file is a simple text file.

Thus, this file can be created manually without a previous export.

The export file contains the name for the field bus "EtherNet IP", "ProfinetIO", "ProfibusDP" (written as in the export file) and a table (see below).

The address is divided into several columns and consists of the following:

- ProfibusDP: Slot, index,
- ProfinetIO: Slot, subslot, index,
- EthernetIP: Class, Instance, Attribute.

This table also includes the variable name, the data type and the access rights of the mapped variables.

```
ProfibusDP
  # Slot    Index Variablename                          Datatyp Access right
     0      49    Application.MotionProg.b_map_01       BOOL    ReadWrite
     0      50    Application.PersistentVars.r_map_01 REAL    ReadWrite
```

Fig. 4-14: Excerpt from the export file Profibus DP, separator TAB:

```
ProfinetIO
# Slot Subslot Index Variablename                          Datatyp Access right
  1    1       49    Application.MotionProg.b_map_01       BOOL    ReadWrite
  1    1       50    Application.PersistentVars.r_map_01 REAL    ReadWrite
```

Fig. 4-15: Excerpt from the export file Profinet I/O, separator TAB:

```
EthernetIP
# Class Inst Attr Variablename                          Datatyp  Access rights
  199   2    1    Application.MotionProg.b_map_01       BOOL     ReadWrite
  199   3    1    Application.PersistentVars.r_map_01 REAL     ReadWrite
```

Fig. 4-16: Excerpt from the export file EtherNet/IP, separator TAB:

4.4 Field bus configuration "Maximal Project"

There are often different variants and designs for machines. For an efficient software engineering, no multiple project configurations have to be maintained. It is only necessary to create **one** IndraWorks project and the components, which are not required, are disabled for a specific machine variant. This procedure is also called "maximal project"

"Disabling" field bus components is reasonable, as no error or diagnostic messages are generated for configured, but not physically existing components.

Using the "UserDefs.cfg" file in the file system of the control, field bus devices and subordinate modules configured in the IndraWorks project are disabled upon control **startup**. The IndraWorks project, including the PLC compilation, is not changed.

Disabling remote devices via "UserDefs.cfg", overview

	Disabling remote slaves	Disabling modules of a remote slave
Sercos I/O	✓	✓
Profinet I/O	✓	✓
Profibus DP	✓	–
EtherNet/IP	– [1)]	–
EtherCAT	–	–
Local bus S20 (XM)	(–)	✓ (as of 14V22)

1) Being prepared

If multiple identical controls are operated below a parent control, these controls have to differ in the slave-interface configuration in case of a field bus communication (at least with regard to addressing, but possibly also in case of configuration modules). The slave-interface configuration can be overwritten using "UserDefs.cfg". The IndraWorks project can thus be identically (unique) created for these subordinate controls:

Overwriting a local field bus slave interface via "UserDefs.cfg", overview

	Overwriting the local field bus slave addressing	Disabling modules of the local field bus slave interface
Sercos I/O	–	–
Profinet I/O	✓	✓
Profibus DP	–	–
EtherNet/IP	–	–
EtherCAT	–	–

For details, refer to the "UserDefs.cfg" creation

- Disabling Sercos slaves using UserDefs.cfg, page 250
- Disabling Profinet I/O devices and Profinet I/O modules using UserDefs.cfg, page 94
- Disabling Profibus DP slaves using UserDefs.cfg, page 59
- Disabling S20 modules using “UserDefs.cfg”, page 297

4.5 Bus cycle task: Copying I/O data between PLC and field bus stack

4.5.1 Describing the bus cycle task

The bus cycle task copies I/O data between the field bus stacks and the PLC I/O buffers.

The I/O buffers are intermediate buffers between field bus stacks and the PLC I/O mapping. The PLC I/O data is updated in ② and ④:

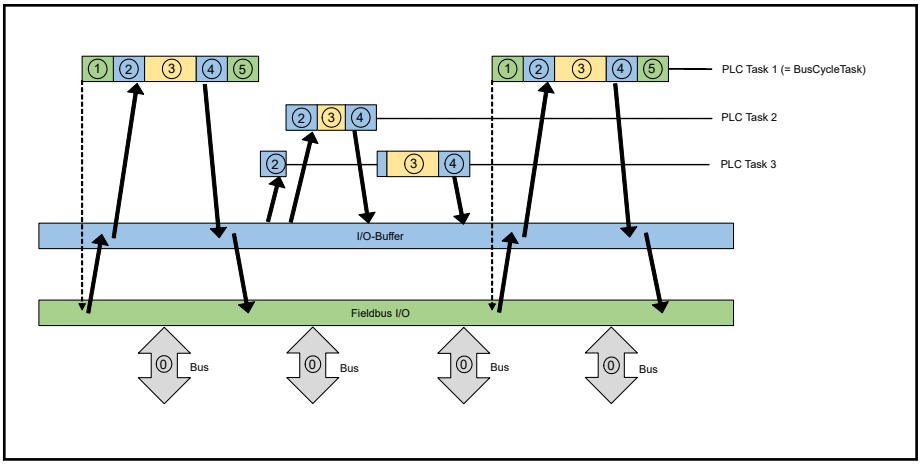


Fig. 4-17: Schematic diagram of the bus cycle task

ⓞ Bus communication cycle

① Copying field bus input data to the PLC I/O buffer (bus cycle task)

② Updating PLC input data used in the respective PLC task (reading I/O buffers)

③ Executing POUs assigned to this PLC task

④ Writing PLC output data used in the respective PLC task (writing to I/O buffers)

⑤ Copying the PLC I/O buffers to the field bus output data (bus cycle task)

☞ For Sercos and EtherCAT, it is copied between field bus stack and PLC I/O buffer within the communication cycle (and not by the bus cycle task).

Data consistency

The copying between field bus stack and PLC I/O buffer cannot be interrupted by a PLC task with a higher priority. Additionally, the bus cycle task can also not interrupt reading and writing PLC I/O data of another PLC task, i.e. data in the PLC I/O buffer is consistent.

☞ When using the same inputs in different tasks, the input data can change while processing a task. This results if this task is interrupted by a task with a higher priority and thus causes the process image to be read again.

Please note: There is no local task “I/O mapping”, the PLC input data is updated globally in ②.

Recommendation: The same inputs and outputs should only be used in one task.

Also refer to chapter 4.5.4 "PLC setting Update variables" on page 31.

Data consistency in the field bus I/O mapping is field bus-specific. Normally, at least the data of a slave and of the communication connection are consistent.

4.5.2 Configuring the bus cycle task

One bus cycle task can be set globally or different bus cycle tasks can be set individually per field bus task.

A global bus cycle task can be selected in the "PLC settings" tab of the device editor of the control (double-click on the control node in the IndraWorks Project Explorer).

The default value "*<Unspecified>*" means that the fastest cyclic PLC task (= task with the smallest cycle time) is used.

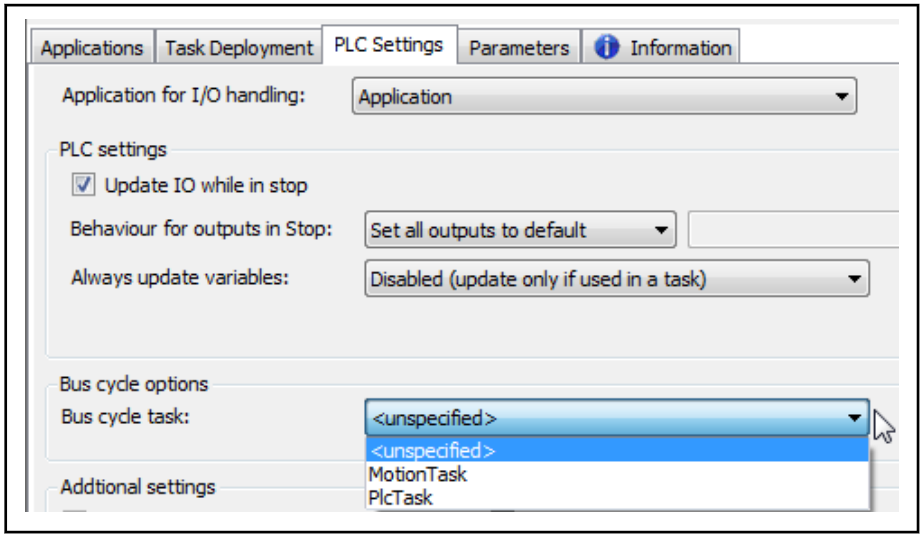


Fig. 4-18: Configuration in the bus cycle task in the "PLC settings" dialog

Notes on the Motion task:

- The Motion task is not a cyclic PLC task, as it is triggered externally.
- It is recommended not to use the Motion task as bus cycle task. For some field buses (e.g. Profinet I/O controller), more functions - apart from copying cyclic data - are executed within the bus cycle task. These functions can affect the Motion task runtime negatively.

For a field bus master or field bus slave (= control is slave), a bus cycle task can be selected for this field bus stack in the "I/O mapping" dialog.

The default setting "*Use parent bus cycle settings*" means that the global bus cycle task is applied from the "PLC settings".

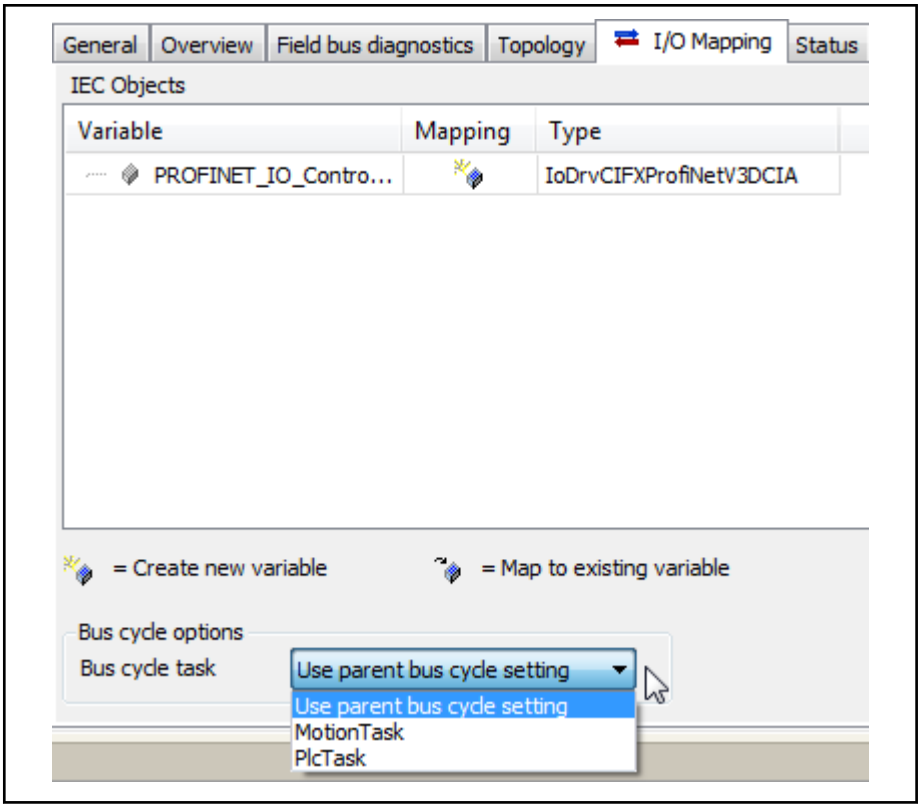


Fig. 4-19: Configuring the bus cycle task in the dialog of a master or slave node

Tip: The "Task deployment" dialog (double-click on the control node) displays in which task I/O data is to be processed.

4.5.3 Bus-synchronous PLC processing

A cyclic PLC task is generally not synchronous to a field bus communication.

For some bus systems (Sercos, EtherCAT, S20 local bus), a externally triggered PLC task can be configured and thus allows a bus-synchronous PLC processing within this task.

Example **S20 local bus**:

- For a bus-synchronous PLC processing, create a PLC task of the type "External" with the event EVENT_OP_MODE_S20_CYCLIC.
- This triggered task has to be set as bus cycle task of the S20 I/O master as well, e.g. on the local bus S20 I/O master in the "I/O mapping" dialog.
- Optionally, the operation mode “Cycle-synchronous to Sercos” can be set additionally at the S20 local bus (refer to the S20 local bus master dialog "Settings").

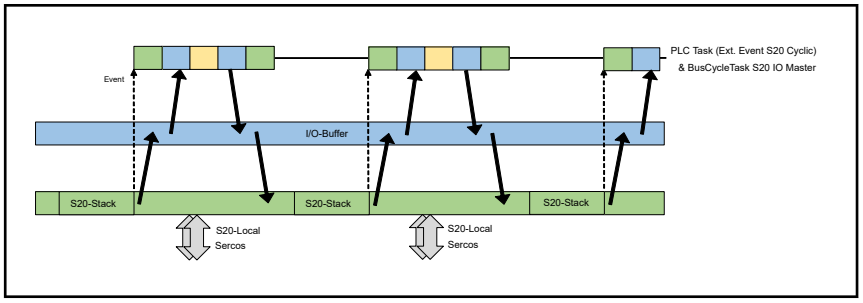


Fig. 4-20: Example of EVENT_OP_MODE_S20_CYCLIC

Example **Sercos**:

- For a bus-synchronous PLC processing, create a PLC task of the type "External" with the event EVENT_OP_MODE_SERCOS_CYCLIC.
- This triggered task is then called synchronously to the bus processing by the Sercos stack. Depending on the application, set a higher priority (e.g. priority 2) for this Sercos event task than for the Motion task (e.g. priority 3) if the Motion task cannot be executed completely within the Sercos cycle.

  Exemplary configuration with 1 ms Sercos cycle time and 2 ms Motion cycle time:

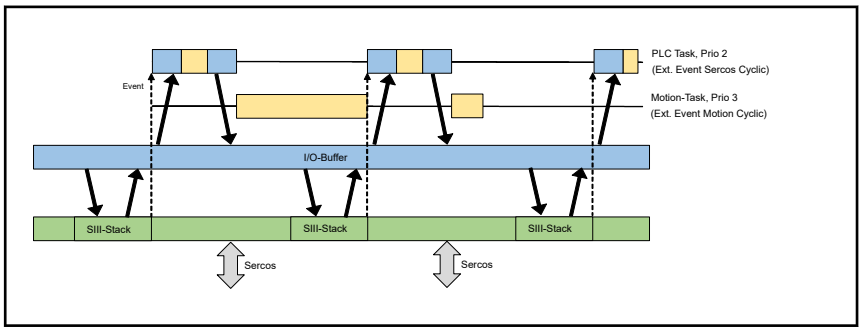


Fig. 4-21: Example of EVENT_OP_MODE_SERCOS_CYCLIC

Example **EtherCAT**:

- Refer to chapter 8.7 "EtherCAT master, bus-synchronous PLC processing" on page 227

4.5.4 PLC setting "Update variables"

For the setting "Update variables",

- go to "PLC settings" (double-click on the control node) (global),
- or go to a field bus slave (or subordinate I/O module) in the "I/O mapping" dialog (individual).

Options:

- **Disabled**: Only I/O data used in a PLC task is updated in the default setting (global) *"Disabled"*.
- **Enabled 1**: To monitor and force field bus I/O data in the logged-in state while commissioning a system for example without the I/O data already being used in a PLC task, this setting has to be set to *"Enabled 1 (use bus cycle task if not used in any task)"*.



The IndraWorks dialogs "I/O mapping" generally show the PLC I/O mapping (not the I/O buffer). The setting "Enabled 1" is used to copy all I/O data not used in any PLC task between I/O buffer and PLC I/O mapping using the bus cycle task.

This setting applies to all bus systems (also Sercos and EtherCAT).

- **Enabled 2**: In the setting *"Enabled 2 (always in bus cycle task)"*, all variables or PLC I/O data are always updated in the bus cycle task and not in the task currently used in[1)] . It is recommended not to use the setting "Enabled 2", but "Enabled 1" during commissioning as well.

4.6 IP channel for Profinet I/O & EtherNet/IP (netX)

General information

An IP channel is an IP communication connection of the Ethernet-based field bus stacks to the operating system of the control. There is an IP routing ("Forwarding") from the field bus interface to the engineering port of the control.

The IP channel was completely revised by changing to netX V3 in IndraWorks 14V22.

The IP channel is supported as follows:

Control as...	Up to 14V20	As of 14V22
Profinet I/O controller	IP channel automatically active	IP channel can be enabled manually
Profinet I/O device	IP channel automatically active	IP channel can be enabled manually
EtherNet/IP scanner	No IP channel	IP channel can be enabled manually
EtherNet/IP adapter	IP channel automatically active	IP channel can be enabled manually

Control system/support

- The IP channel connection is supported by the MLC
- It is currently not supported by the MTX



The IP channel can only be used if the IndraWorks version and the control firmware match.

"IP channel settings" tab

The IP channel can be enabled in the "IP channel settings" (as of 14V22). Another subnet than the field bus communication can be used.

1) *I.e. If "Enabled 2" is selected, it is always copied between PLC I/O data and I/O buffer in the bus cycle task and not in the task using I/O data, see ② and ④ in fig. 4-17 "Schematic diagram of the bus cycle task" on page 28.*

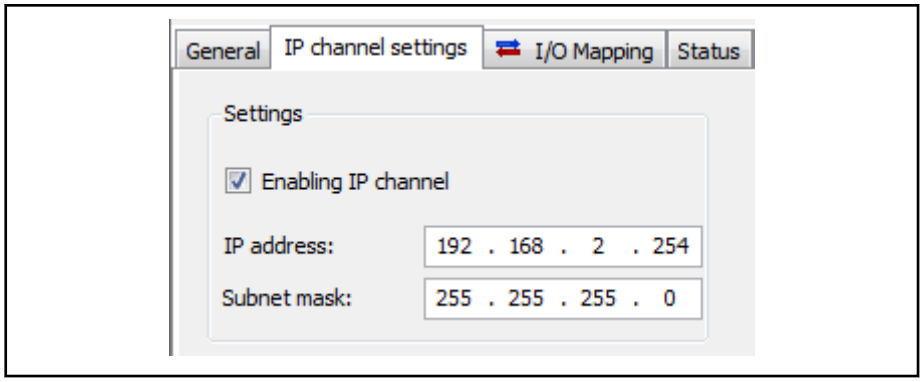


Fig. 4-22: Tab "IP channel settings"

The following boundary conditions are to be considered

- The IP address of the IP channel has to differ from the IP addresses of the field bus devices!

  I.e. Even the IP address of the local field bus interface of the control (PNIO controller/device, ENIP scanner/adapter) may not be identical!
- Also note that the IP channel can only be enabled if no other interface of the control (e.g. engineering port or IP channel of another field bus interface) is provided with the same subnet! → In case of error, there is an entry in the PLC log.
- With regard to the **IP routing** via the engineering port of the control, the gateway address in the field bus devices has to be set to the IP channel address (see use case in the following chapter).

  Note that the gateway address of a field bus device and the IP address have to be in the same subnet. For the use case "IP routing via the engineering port of the control", the IP channel has to be entered as gateway in the field bus slave and has to be in the subnet of the field bus devices.

Use cases:

IP routing via the engineering port of the control

There are the following IP settings in the configuration example:

- Ethernet port of the control: 10.110.245.40 (⇔ network to the engineering PC)
- EtherNet/IP scanner: 192.168.2.1
- IP channel: 192.168.2.254 (⇔ has to differ from the EtherNet/IP scanner or adapters or other devices in the network)
- EtherNet/IP adapter: 192.168.2.2/../21
- As a alternative to the routing via the engineering port, a direct connection can also be established between the engineering PC and the field bus network (in the example shown via a second network interface at the PC, 192.168.2.250)

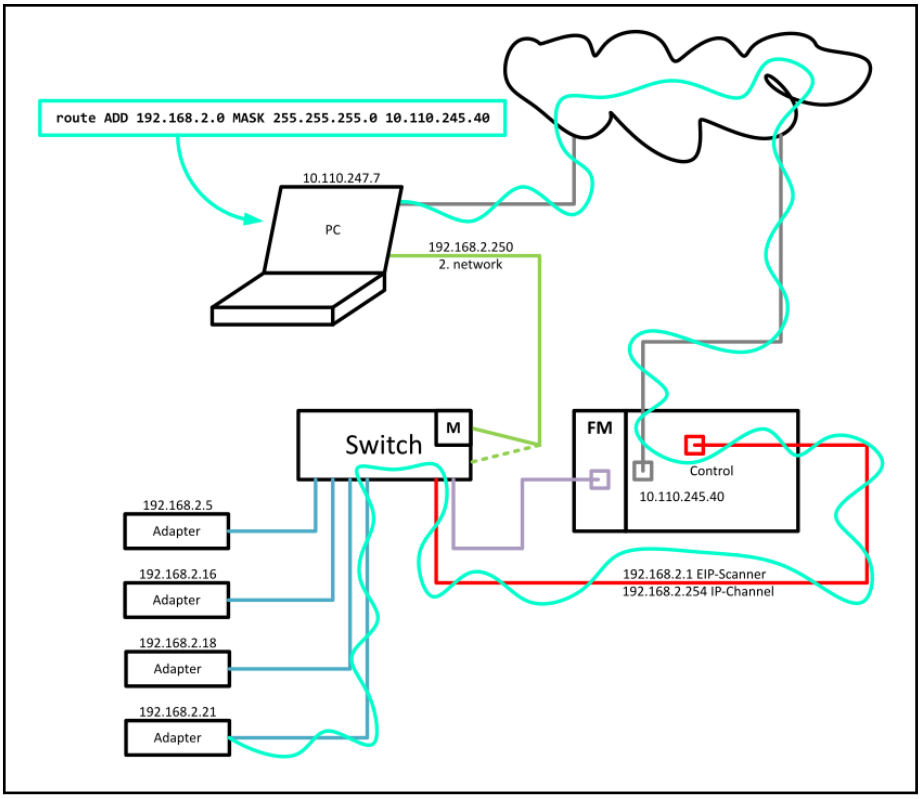


Fig. 4-23: IP channel - Routing examples

Notes on IP routing:

- The IP routing ("Forwarding") routing in the control takes place automatically (if the IP channel is enabled)
- To access the field bus network from the engineering PC via the engineering port of the control using IP communication, an IP routing entry has to be made manually into the routing table of the engineering PC.

  Example of command line command (execute as "administrator"):

  ```
  route ADD 192.168.2.0 MASK 255.255.255.0 10.110.245.40
  ```

  (for more information, enter for example the command line command "route help")
- Return route: The addressed field bus device also requires a gateway address with the IP address of the IP channel of the control. The preceding example shows EtherNet/IP adapters as gateway 192.168.2.254 (= IP channel of the control).

  Configuring gateway address of field bus devices:

  – **EtherNet/IP scanner**: If an ENIP adapter obtains its IP address settings via BootP from the ENIP scanner (from the IndraWorks configuration), the IP channel address of the control is automatically set as gateway. Alternatively, the gateway has to be set manually like all IP settings of the adapter.

  – **Profinet I/O controller**: The gateway addresses of the PN devices have to be set to the IP channel address in the IndraWorks configuration.

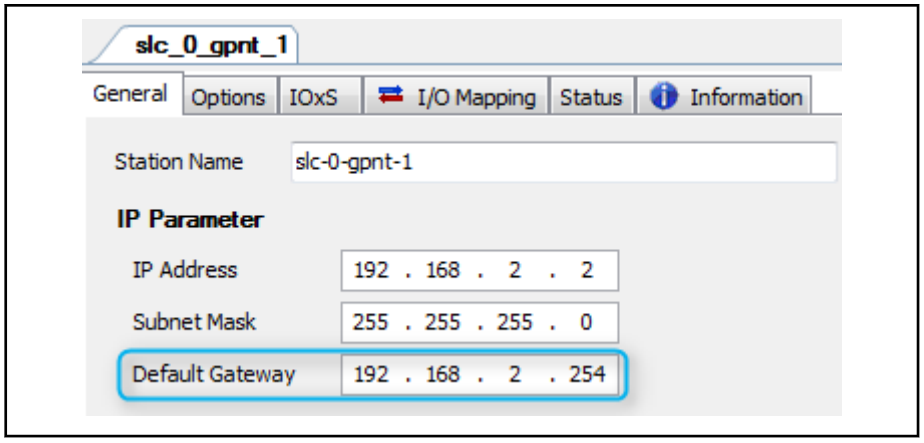


Fig. 4-24: PN device - Default gateway configuration (= IP channel address)

UDP or TCP communication with the PLC

For UDP or TCP communication with the PLC in the field bus network, the respective IP channel has to be enabled. The restrictions described in the preceding chapter apply. Thus, the IP channel address has to differ from the field bus master and slave IP settings.

The function blocks of the RIL_SocketComm.library can be used for example. Use the IP channel address as IP address of the control (to communicate from the field bus network). A value higher than 1024 is recommended as port number.

OPC-UA communication

OPC-UA communication is possible via field bus interface as of IndraWorks 14V22. Use the IP channel address as IP address of the control (to communicate from the field bus network). The restrictions described in the preceding chapter apply. Thus, the IP channel address has to differ from the IP settings of field bus master and slave.

4.7 PLC controls drives via field bus connection (field bus axes)

4.7.1 General information

Drives (IndraDrive from MPx2) can be controlled by an IndraMotion MLC depending on their interfaces via the respective field bus.



The drive coupling is exemplarily described via Profibus DP.
Coupling is similar via Profinet IO.
Coupling via the Sercos III bus is not described here.

The activation is executed from the PLC program using MotionLogic function blocks according to the PLCopen concept.

Before operating such a field bus axis at the Profibus,

- an IndraWorks project has to be created,
- the desired PLC has to be applied to the project from the device library,
- a Profibus interface (onboard or function module) has to be activated as Profibus DP master,
- Introducing field bus axes of the PLC, page 36, and
- Communication-required drive settings, page 37 have to be made.

The PLC programs use MotionLogic function blocks, functions and data types summarized in multiple libraries (Including required libraries, page 39).

These libraries are integrated together with the PLC program(s) into the IndraWorks project.

Additionally to the controlling of the field bus drive via the PLC program, values have to be exchanged with the drive saved in its parameters.

For **IndraDrive**, this can be executed via the PLCopen function blocks (MB_ReadParameter, MB_ReadRealParameter, MB_WriteParameter, MB_WriteRealParameter...) of the **RMB_PLCopenFieldBus.library**.

4.7.2 Creating field bus axes in the project

System and drive settings are required in the IndraWorks project at the PLC to ensure a correct communication between the drives and the PLC.

The field bus axes are ordinary field bus devices at the Profibus.

The description files for possible IndraDrives are included in the IndraWorks device library:

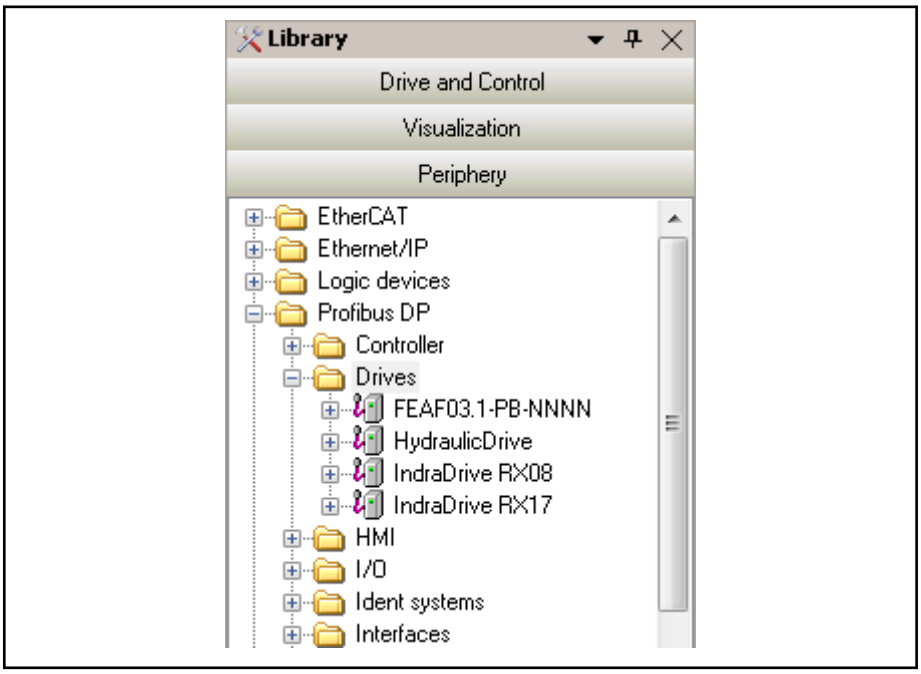


Fig. 4-25: Description files of the drive in the IndraWorks library

Drag the required drives from the library onto the Profibus DP master object.

In the Project Explorer, drives can thus also be inserted between existing drives.

Optionally, slaves can be added in the context menu via **Add** ▸ **Slave** ▸ **Drives** of the Profibus DP master object.

In this case, the new drive is added as last drive below the Profibus DP master object.

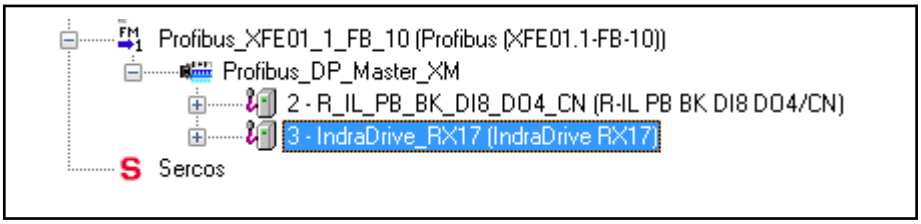


Fig. 4-26: Entering IndraDrive as Profibus device

The drive can be renamed:

Context menu **Rename F2**, e.g. "Drive_X" instead of "IndraDrive_RX05".

The drive can receive a new station address:

Double-click on the object; change the station address for example to "11" instead of "3".

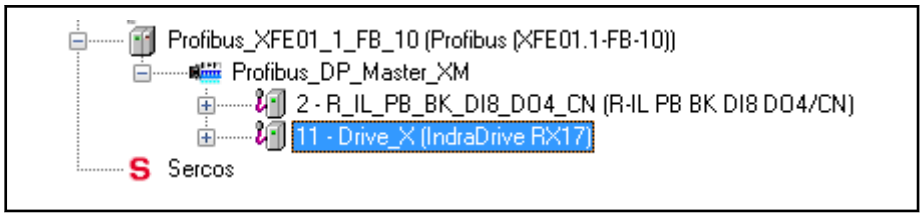


Fig. 4-27: Renaming IndraDrive as Profibus device

The Profibus device is provided with several subdevices whose content is specified by the device description file.

For IndraDrive, the following settings are required:

- ParamCh_5_Words,
- F-Modul_not_used, **(if the module is not used)**
- Input_8_Words,
- Output_10_Words.

The drives (device files) in their original state are not provided with the required settings to communicate together with the PLCopen function blocks from the RMB_PLCopenFieldBus.Library via Profibus.

First, the settings in the note have to be made by "Setting device" of the subdevices.

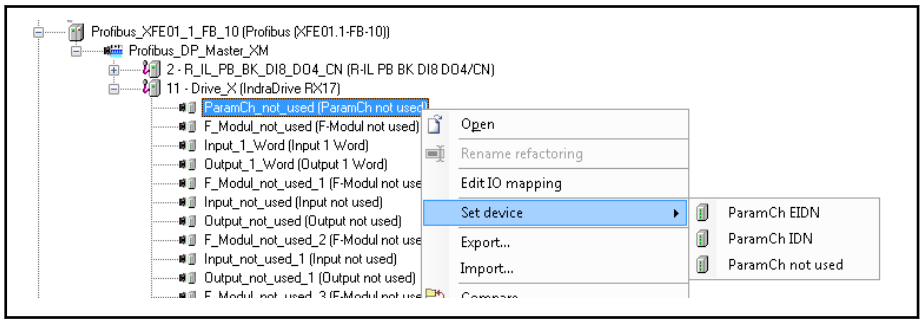


Fig. 4-28: IndraDrive, setting the subdevices

Even the drives are not configured in their original state for such a communication with a PLC via a Profibus.

Therefore, drive settings have to be made for the PLC communication.

4.7.3 Drive settings for the PLC communication

General information

To activate field bus drives with the PLCopen function blocks, some basic settings are required regarding the communication of the PLCopen functions blocks with the field bus drives.

The settings described do only refer to the communication between the drive and the PLC and do NOT describe the general drive commissioning.

The required settings can be made via the "IndraWorks Engineering" desktop or via "IndraWorks DS".

Currently, it **cannot** be parameterized via the Profibus connection of the PLC.

From the PC, on which the IndraWorks Engineering desktop is started, the communication has to be established via RS232 serial or via the existing Profibus DP master for the parameterization.

In the following, the parameterization is described via serial communication.

Establishing the communication for drive parameterization

Using the function **Project ▸ Scan for Devices** in the IndraWorks Engineering Desktop, the parameterization communication to the drive can be established.

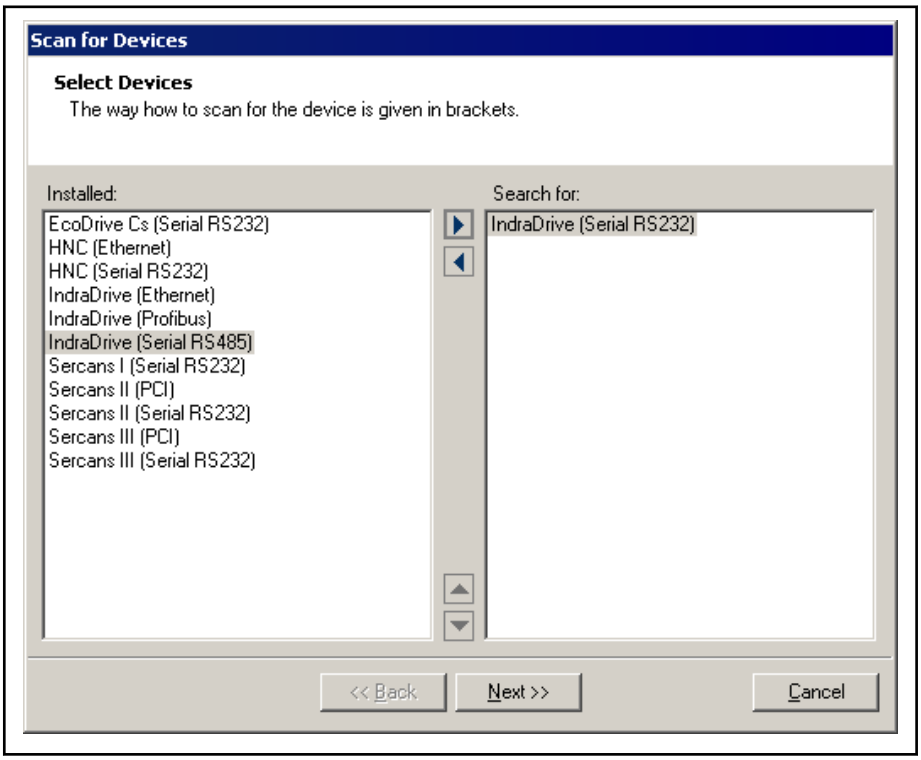


Fig. 4-29: Establishing an IndraWorks parameterization communication to the drive

If the connection to the drive is established, the settings can be made.

IndraDrive communication settings

General information

In the following, the IndraDrive settings required for the communication are described.

Setting bus device address

Set the bus device address of the drive in the node "Master communication with regard to the set station address.

Length of parameter channel

By entering the parameter P-0-4083 directly using the parameter editor, the length of the parameter channel is set to 10 bytes.

A newer IndraDrive firmware (from MPH-03VRS) automatically sets to 5 words (ParamCH_5_Words).

Setting process data and profile type

The profile type "Freely configurable mode" and the parameters of the cyclic data exchange with the PLC are set in the following:

Process data inputs (AT)

- P-0-4078, Field bus status word
- S-0-0051, Actual position value encoder 1
- S-0-0040, Actual velocity value
- S-0-0390, Diagnostic number
- S-0-0144, Signal status word

Process data outputs (MDT)

- P-0-4077, Field bus control word
- S-0-0282, Positioning command value
- S-0-0259, Positioning velocity
- S-0-0260, Positioning velocity
- S-0-0359, Positioning deceleration
- S-0-0145, Signal control word

Signal control word Bit 0: S-0-0346: Positioning control word, bit number 4

Signal status word Bit 0: S-0-0343: Message interpolator stopped, bit number 0

Operation mode settings As primary operation mode, the operating mode "Drive-controlled positioning" has to be set.

"Velocity control" has to be assigned as first secondary operation mode.

The second and following secondary operation modes are not relevant. These modes are not selected by the PLCopen function blocks.



The MB_Home function block only references if the drive is parameterized for referencing.

4.7.4 Including required libraries

The PLCopen concept was implemented for IndraWorks Engineering for two field bus types and is supported by the following libraries:

Sercos II and Sercos III

- ML_PLCopen.library, function blocks of PLCopen for Sercos drives.
- RIL_CommonTypes.library, system-comprehensive data types, addressing, error management.
- ML_Base.library, supplementing data types to the ML* system,
- Supplementing libraries, e.g. RIL_Utilities.library.

Profibus DP

- RMB_PLCopenFieldBus.library, function blocks and data types of the PLCopen for Profibus drives.
- RIL_CommonTypes.library, error management.
- Supplementing libraries, e.g. RIL_Utilities.library.

It was tried to maintain a uniform concept for both field bus types. This also affects identical names for functionally identical function blocks and data types.

Except for the RMB_PLCopenFieldBus.library, all libraries are part of the IndraWorks Engineering standard installation. They are also initialized during the installation and **do not** require namespace specifications.

RMB_PLCopenFieldBus.Library The "RMB_PLCopenFieldBus.library" is not included in the IndraWorks projects by default.

If functions of this library are required, install the library in the project via the library manager.

Proceed as follows for the installation:

1. Open the library manager ① in the Project Explorer and select **Add library...** ②.

   The command opens the “Library” dialog.

2. Open the directory **IndraLogic** ▸ **Base** and select the library "RMB_PLCopenFieldBus" ③.
3. Press "OK" to install the library.

   The library is now installed in the project.

4. Select the library in the library manager ④.

   The library content is displayed in the lower window of the library manager.

5. Open the folder "Datatypes" and select "AXIS_REF" ⑤

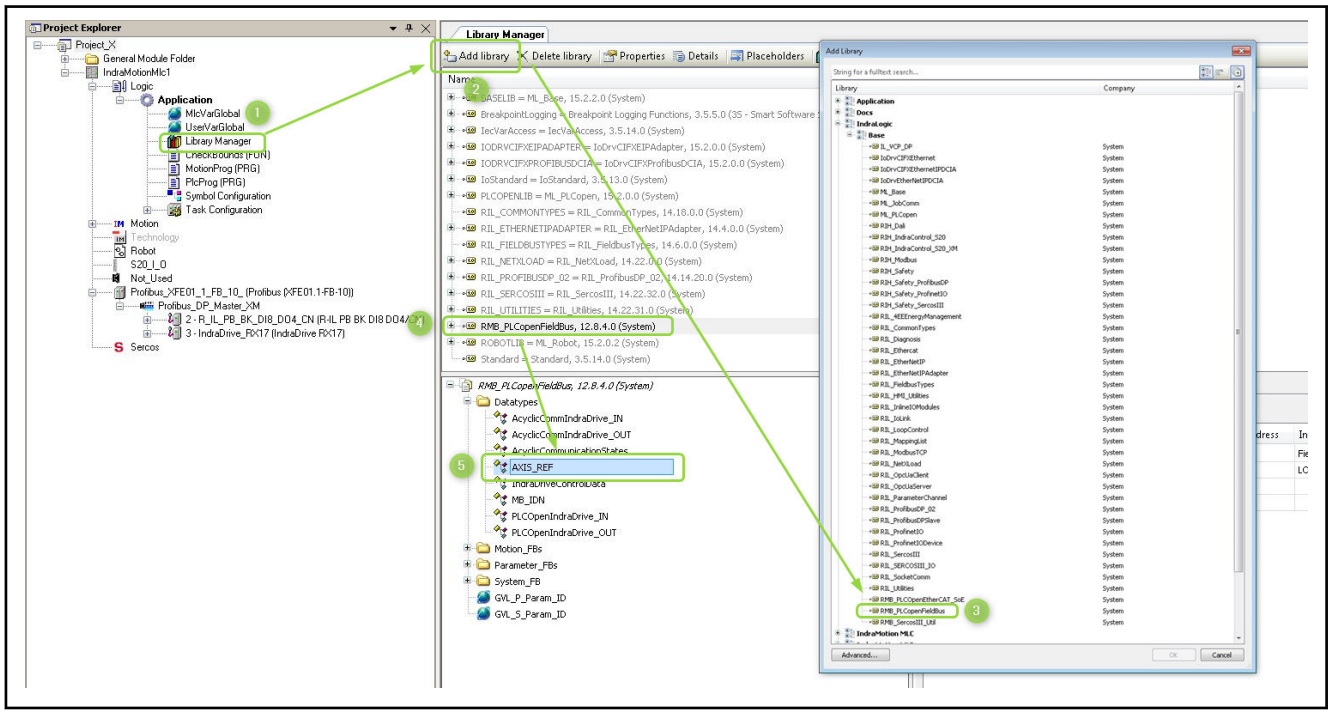


Fig. 4-30: Example: Library Manager

The "RMB_PLCopenFieldBus.library" can run on all IndraWorks target systems.

The elements of the library are always to be used with a specification of the namespace "FieldBusDrives.".

If field bus axes and Sercos axes should be operated together, both libraries (RMB_PLCopenFieldBus.library and ML_PLCopen.library) have to be in the project. With the namespace specification, the function blocks and the axis reference (AXIS_REF) are distinguished.

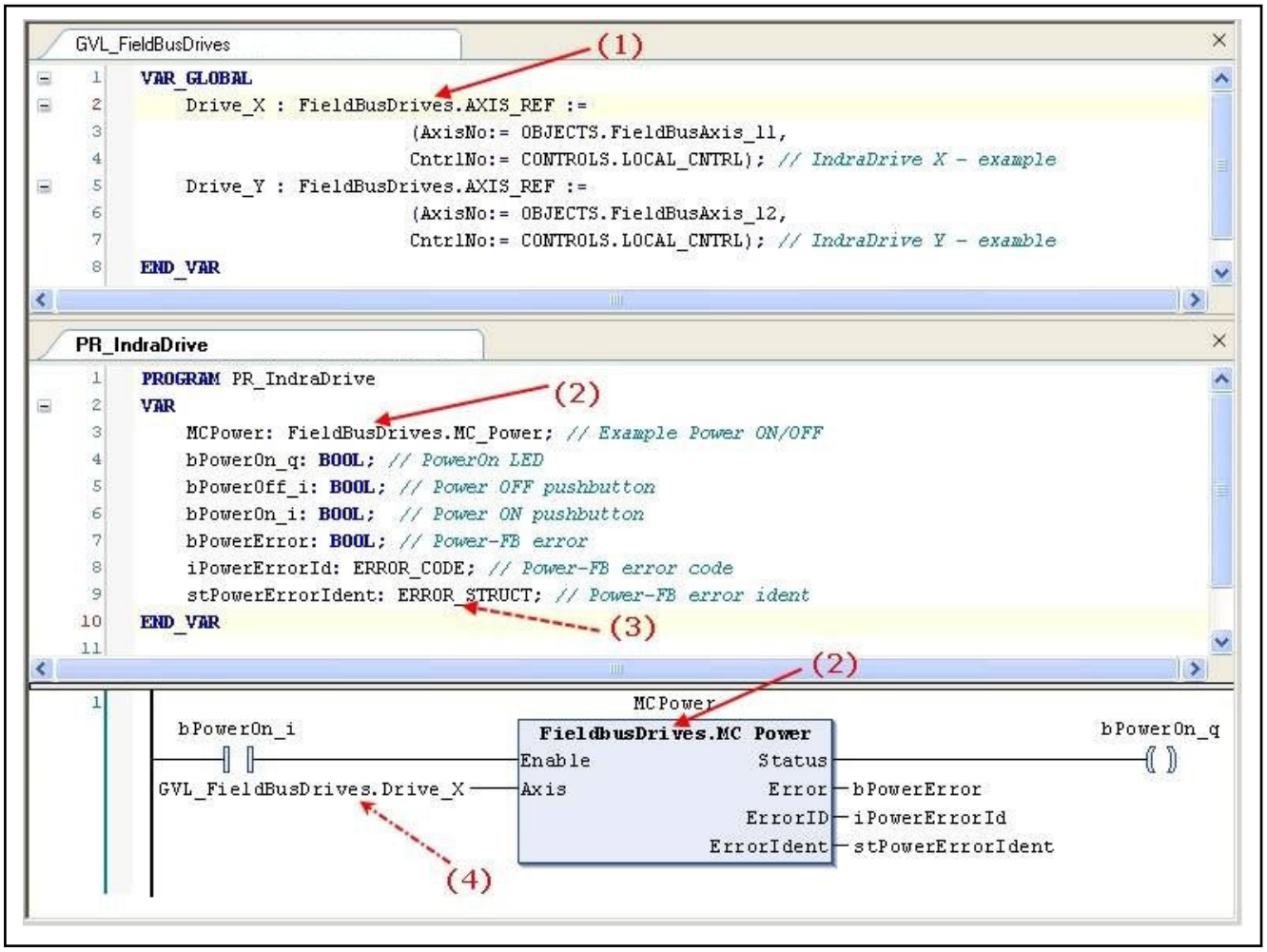
Namespace specification "FieldBusDrives"

The namespace specification "FieldBusDrives." for a field bus drive has to be in front.

The following figure shows the usage of the namespace.

The declaration of an axis reference (AXIS_REF) is always required with namespace specification.

The specification of the namespace during the declaration of a PLCopen function blocks is only required if the library (ML_PLCopen) is available with PLCopen function blocks in the project.



(1) AXIS_REF originates from the RMB_PLCopenFieldBus.library and not from the RIL_CommonTypes.library

(2) MC_Power originates from the RMB_PLCopenFieldBus.library and not from the ML_PLCopen.library (declaration and implementation)

(3) ERROR_STRUCT originates from the RIL_CommonTypes.library

(4) Drive_X originates from the Global Variable List "GVL_FieldBusDrives"

Fig. 4-31: Examples for the namespace specifications

4.7.5 Addressing field bus drives in the PLC project

AXIS_REF, axis addresses of field bus drives

The structure AXIS_REF (RIL_CommonTypes) contains address information on the desired control and on the object controlled by this control (axis number).

In contrast to the AXIS_REF structure (RIL_CommonTypes) supporting the addressing of Sercos-based IndraDrives, the AXIS_REF structure (RMB_PLCopenFieldbus) is extended by two additional elements (MP_Drive and MP_DriveParameter). These two elements are only used by the system.

The axis number can differ from the station number set at the drive.

The Global Variable List "MLCVarGlobal" with all Sercos-based axes is automatically created and maintained for Sercos-based axes.

The global variable list "GVL_FieldBusDrives" with all field bus drives has to be manually created and maintained for the field bus drives.

The two first elements of each drive "CntrlNo" and "AxisNo" are preassigned in this list. The elements "MP_Drive" and "MP_DriveParameter" remain unchanged. Therefore, they do not have to be included in the list.



Address information is transferred as described in the following:

Sercos-based axes

- In IndraWorks 14VRS and higher, information is transferred via **VAR_IN_OUT CONSTANT**.

Field bus axes

- Information is always transferred via **VAR_IN_OUT**.

Global Variable List "GVL_FieldBusDrives", example

```
VAR GLOBAL
Drive_X: FieldBusDrives.AXIS_REF :=
                (CntrlNo:=LOCAL_CNTRL,
                AxisNo:=FieldBusAxis_11); // IndraDrive X - example
Drive_Y: FieldBusDrives.AXIS_REF :=
                (CntrlNo:=LOCAL_CNTRL,
                AxisNo:=FieldBusAxis_12); // IndraDrive Y - example
Drive_Z: FieldBusDrives.AXIS_REF :=
                (CntrlNo:=LOCAL_CNTRL,
                AxisNo:=FieldBusAxis_13); // IndraDrive Z - example
Drive_n: FieldBusDrives.AXIS_REF :=
                (CntrlNo:=LOCAL_CNTRL,
                AxisNo:=FieldBusAxis_14); // IndraDrive n - example
END_VAR
```

Namespace specification FieldBusDrives

The namespace specification "FieldBusDrives." for a field bus drive has to be in front.

Declaration AXIS_REF

```
Drive_A: FieldBusDrives.AXIS_REF: =
```

The data types are in the RMB_PLCopenFieldBus.library, folder "Datatypes", element "AXIS_REF".

Assignment CntrlNo

```
(CntrlNo: = LOCAL_CNTRL
```

The enumeration value LOCAL_CNTRL is contained in RIL_CommonTypes.lib, folder" Data types of logical addresses", CONTROLS.

The field bus function blocks always refer to the local PLC, that is **LOCAL_CNTRL**.

Assignment AxisNo

```
AxisNo: = FieldBusAxis_1);
```

The enumeration value FieldBusAxis_1 is in the RIL_CommonTypes.lib, folder "Data types of logical addresses", OBJECTS.

To avoid ambiguities, there is an individual number range stored for field bus axes

`FieldBusAxis_1` (6001) up to `FieldBusAxis_99` (6099)

.

For field bus drives, specifications on the control and the axis number are used for the error output (ErrorIdent).

A functionality is not combined with the assignment of the axis number.



It is reasonable to compare the Profibus station and axis number.

AxisData[], axis data

The AxisData array, known from the Sercos axes, is not supported for field bus axes.

4.7.6 I/O addressing of the drives in the PLC project

General information

On the Profibus, the field bus drives appear as ordinary I/O modules. An I/O address has to be assigned.

In order to facilitate the I/O address assignment, it is directly executed as variable mapping on the Profibus module of the IndraDrive.

I/O address assignment: IndraDrive

The parameter communication, the F-module (safety technology) and the cyclic communication at IndraDrives are always individual submodules of the device.

The communication "Drive to the PLC" consists of the following:

- Subdevice (ParamCh_IDN), for acyclic parameter communication,
- Subdevice (F-module), optional use; not used here,
- Subdevices of the cyclic data exchange (8 words input or 10 word output)

Addresses are mapped at the respective subdevice channels of the drive.

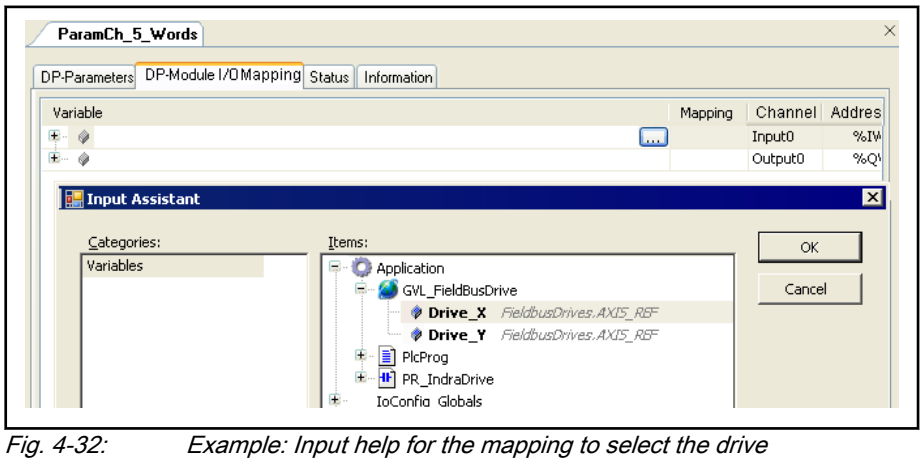


Fig. 4-32: Example: Input help for the mapping to select the drive

In AXIS_REF, the field bus drives are distinguished for the cyclic parameter exchange and "IndraDriveParameter" for the acyclic parameter exchange according to "IndraDrive"

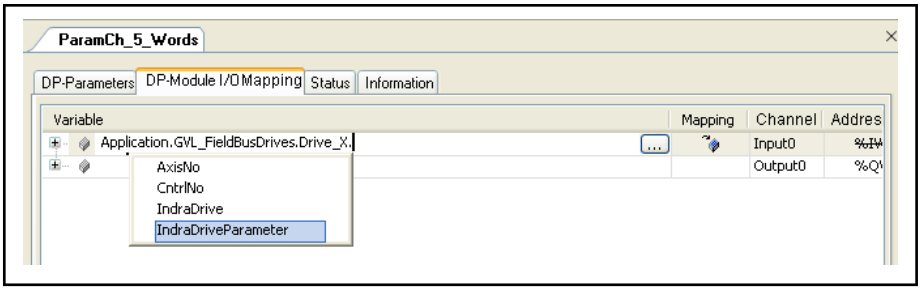


Fig. 4-33: Example: Mapping for the parameter exchange

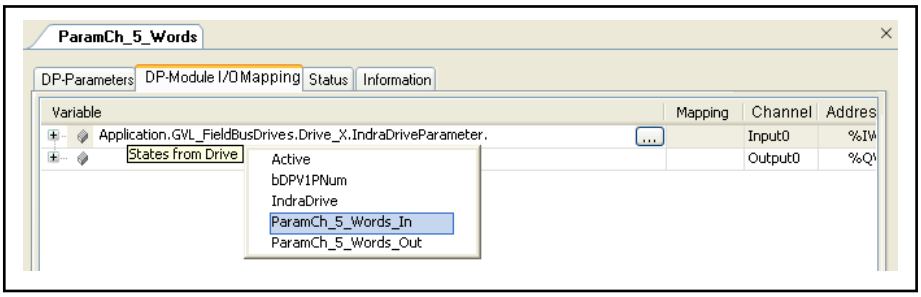


Fig. 4-34: Example: Mapping for the inputs of the parameter exchange

By direct mapping, the initial addresses for the parameter communication, the cyclic data exchange and - if available - an F-module with regard to the drive do not have to be considered.

5 Profibus DP

5.1 Features of the Profibus DP master

The current implementation of the Profibus DP master includes the following functionalities:

- Bus master functionality acc. to DIN EN 50170, part 2
- Cyclic data traffic (DP/V0)
- Acyclic data traffic (DP/V1 class 1) via function blocks
- GSDML file import into the device database
- Profibus slave-related connection status in the interface
- Profibus master diagnostics via function block
- Profibus slave diagnostics via function block
- Sync/Freeze
- Disabling slaves (upon bus startup)

Future extensions are planned for the following functionalities:

- Alarms
- Disabling slaves in the PLC program

Function/characteristic	Value
Max. number of slaves	125
Baud rate	9.6 kBit/s to 12 MBit/s
Max. amount of cyclic input data	3584 bytes
Max. amount of cyclic output data	3584 bytes
Max. number of cyclic input data per slave	244 bytes
Max. number of cyclic output data per slave	244 bytes
Max. length of consistent data blocks	244 bytes
Max. amount of telegram data acyclic per slave/telegram	240 bytes
Number of simultaneous DP/V1 class 1 services per slave	1

Tab. 5-1: Technical data

5.2 Features of the Profibus DP, slave (control as slave)

The current implementation of the Profibus DP slave includes the following functionalities:

- Slave functionality acc. to DIN EN 50170, part 2
- Cyclic data traffic (DP/V0)
- Acyclic data traffic (DP/V1 class 1)
- Disabling slaves when starting the PLC program

Future extensions are planned for the following functionalities:

- Profibus DP slave diagnostics via function block
- Alarms
- Sync/Freeze

Function/characteristics	Value
Max. number of I/O modules	16 (8 in each direction)
Baud rate	9.6 kBit/s to 12 MBit/s
Max. amount of cyclic input data	244 bytes
Max. amount of cyclic output data	244 bytes
Max. length of consistent data blocks	128 bytes
Max. telegram data per acyclic telegram	240 bytes

Tab. 5-2: Technical data

5.3 Profibus DP, basics

5.3.1 Terms and abbreviations

Master The Profibus field bus master is called **master**.

Slave The Profibus field bus slaves are called **slaves**.

Device address The criterion to address a slave is the device address (= Profibus address).

It is between 1 and 125

GSD file This is a device description used by IndraWorks to make the Profibus slaves known.

5.4 Profibus DP, configuration overview

The following controls support Profibus DP:

- **XM2, XM4**

  Connection via the extension module **XFE01.1-FB-10**

  Configurable as Profibus master/Profibus slave.
- **VPx**

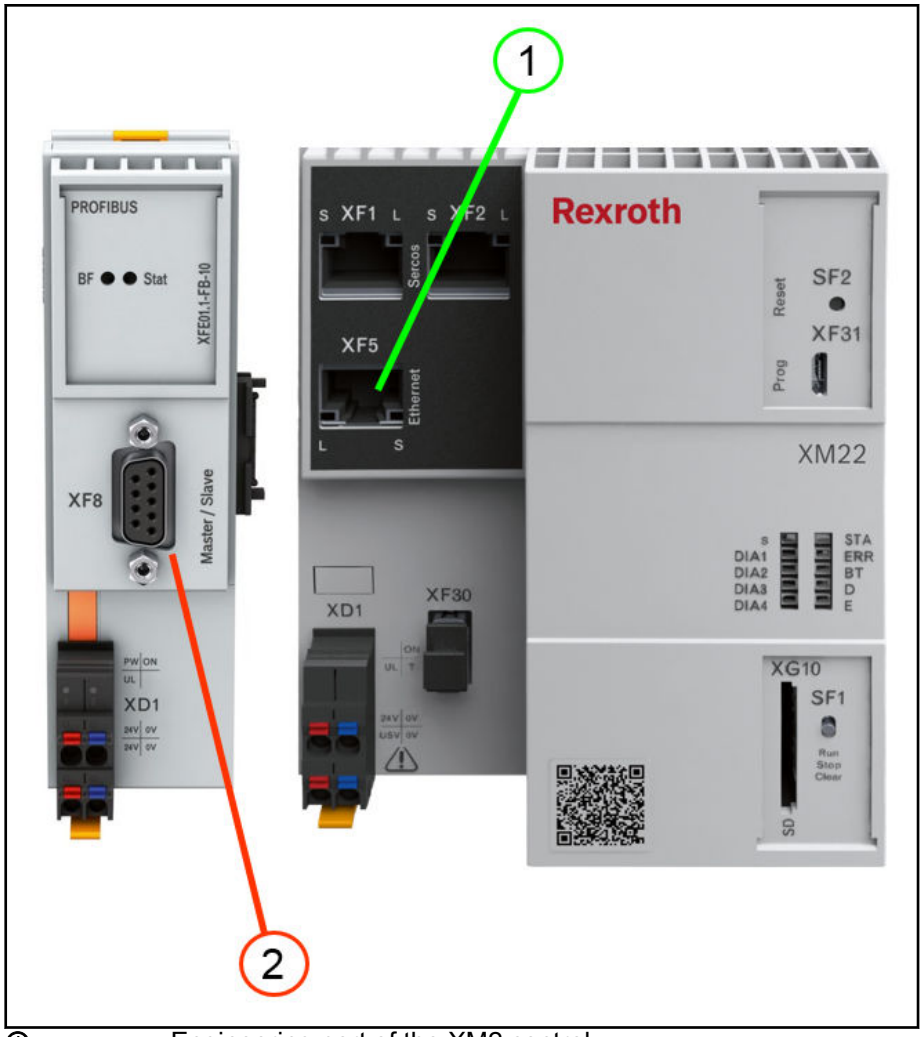
  Connection via the extension module **PFC01.1-FB-13**

  Configurable as Profibus master/Profibus slave.

Installing and adding Profibus DP devices

Extension modules are used to connect the Profibus to the control types XM and VPx:

- For XM controls, the extension modules are connected to the left side of the control.
- For VPx controls, the extension module is plugged into a free slot as plug-in card.



① Engineering port of the XM2 control
② Function module Real Time Ethernet XFE01.1-FB-10; the Profibus interface can either be configured as master or slave.

Fig. 5-1: Profibus connection at the XM2 control

In the IndraWorks project, add the extension module to the control and configure it.

It can either be added upon the first creation of the control or subsequently in the control properties.

After adding the Profibus extension module, configure whether to operate the module as master or slave.

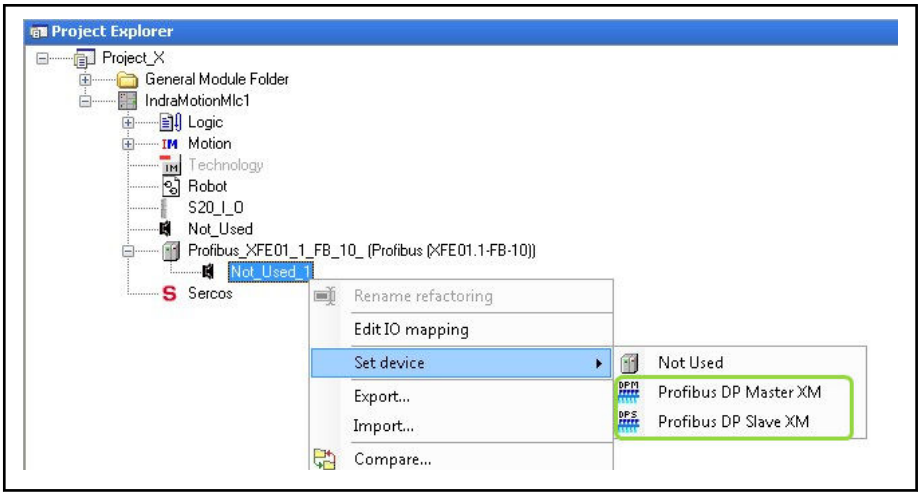


Fig. 5-2: Configuring the Profibus extension module in the Project Explorer

5.5 Profibus DP, master

5.5.1 Profibus DP master, configure

General information

To open the editor window in the Project Explorer, double-click on the Profibus DP master object.

The dialogs provide information on the configuration of the entire Profibus. Modifications can be made.

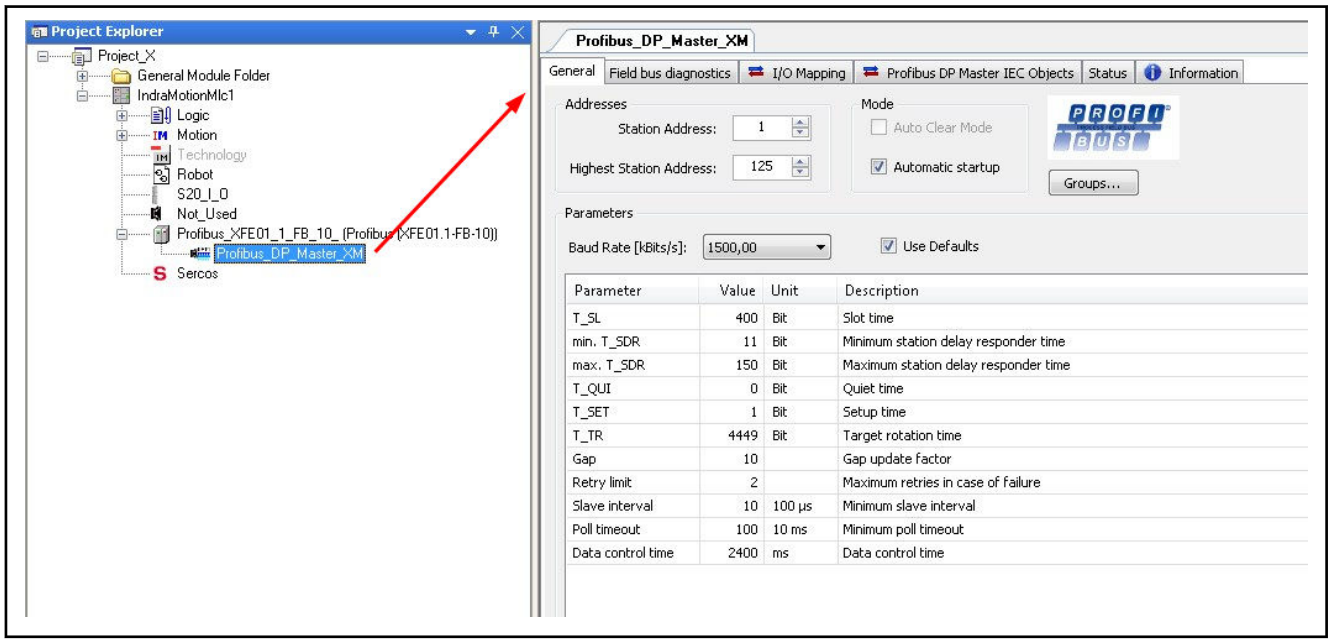


Fig. 5-3: Profibus DP master object

Configuration dialogs for Profibus DP master

- Tab “General” , page 49
- Tab “Profibus DP master parameter”, page 51, this window is used for service purposes and is only visible if the option "Show generic configuration editors" is enabled under **Tools ▸ Options ▸ IndraLogic 2G ▸ Device editor**.

- Tab “Field bus diagnostics”, page 52
- Tab “I/O mapping”, page 55
- Tab “Status”, page 56,
- Tab “Information”, page 56

Tab “General”

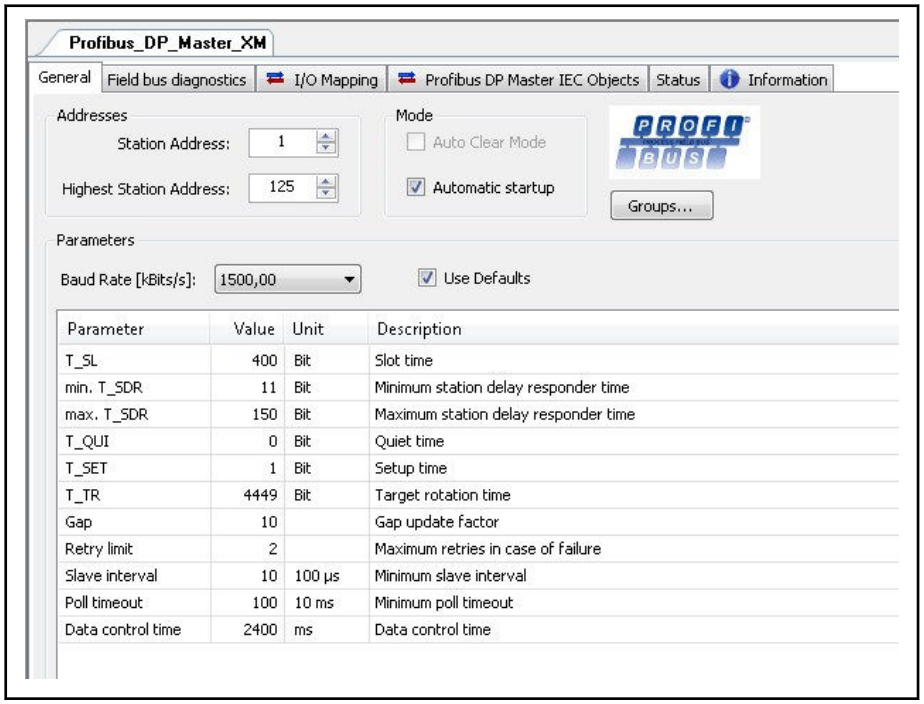


Fig. 5-4: Profibus DP master - "General" tab

Parameters to be configured for each Profibus DP master. In general, they are defined by the device description file, but can be modified in the "General" tab.

Addresses:

- **Station address**: The valid range is from 0 to 125.

  Each new device added to the bus automatically obtains the next higher address

  Manual input is allowed; addresses are checked for duplicates.
- **Highest station address**: The highest station address at the bus is displayed. To keep the address range - used to search for new active devices - small, enter a lower address.



Use the lowest addresses possible. High address values decrease bus performance!

Mode:

- **Auto clear mode:** is currently not supported.
- **Automatic startup**: When this option is enabled, the master starts automatically. Otherwise, it has to be started manually.



It depends on the driver, whether the setting for the "Automatic startup" option is evaluated.

Group properties:

- The "Groups..." button opens the "Group Properties" dialog. The group properties refer to the slaves assigned to the master.

  Up to eight groups can be set up. Set for each group whether it is to be operated in Freeze mode and/or Sync mode.

  The assignment of the slaves (see "Properties of DP Slaves", "Group assignment") to different groups can synchronize the data exchange from the master using a global control command.

  With a **freeze command**, a master causes that a slave or a group “freezes” the inputs in the current status and transfers this data during the next data exchange.

  A **sync command** causes the slaves to switch synchronously through the data received by the master in the following data exchange with the next sync command at the outputs.

  To switch on and off the freeze and sync options for a group, left-click on the corresponding position in the table to place/remove a "v" for the desired option. In addition, group names can be edited.

  Group names can be changed. Thus, select the name from the "Group" column and press the <space bar> to allow the editing of the character string. To enable the freeze and sync option for a group, select the checkbox in the corresponding column. An "x" in the box means that the property is enabled.

Parameter: These parameters describe the time response of the communication on the Profibus.

- **Baud rate [kBits/s]**:

  The selection defined in the device description file is available.

☞ In the baud rate [kBits/s] selection field, only a baud rate supporting all slaves may be selected.

- **Default setting**: If this option is enabled, the values in the parameter table are set to the default values based on the currently set baud rate and **cannot** be modified.

For each parameter, the parameter table displays the unit and a brief description.

To edit the parameter values, double-click on the respective “value” field (option default setting: Disabled).

Parameters [<Unit>]	Description
T_SL [Bit[1)]]	Slot time: Maximum period the master waits for the slave to respond after sending a request.
min. T_SDR [Bit]	Minimum station delay responder time: Minimum response time after which a device on the bus may respond.
max. T_SDR [Bit]	Maximum station delay responder time: Maximum period within a slave has to respond.

1) Bit: Time unit to transfer a bit on the Profibus

Parameters [<Unit>]	Description
T_QUI [Bit]	Quiet time: Quiet time to be considered when converting NRZ signals (Non Return to Zero) to other encodings (switching time for repeater).
T_SET [Bit]	Setup time: Time for setup
T_TR [Bit]	Target rotation time: Token command rotation time; projected time interval during which a master has to receive the token. The result from the sum of the token holding times of all masters at the bus.
Gap	Gap update factor: Number of bus rotations after which another newly added, active station is searched for in the master's gap (address range from its own bus address to the address of the next active device).
Retry limit	Maximum number of times the master tries to call again if it did not receive a valid response from the slave.
Slave interval [µs]	Time between two bus cycles during which a slave can process the request of the master (time basis 100 µs). The value entered has to match the respective specifications in the device description file of the slave.
Poll timeout [ms]	Maximum period after which the response of a master in a master-master communication has to be obtained from the requester (DP_Master class 2, time basis 1 ms).
Data control time [ms]	Time during which the master informs the assigned slaves on its operating state. The master simultaneously monitors whether at least one user data exchange occurred with the slaves within this period and updates the Data_Transfer_List.

Tab. 5-3: Parameter table

Tab “Profibus DP master parameter”

This tab is used for service purposes and is only visible if the option "Show generic configuration editors" was enabled. Opening via the IndraWorks menu **Tools ▸ Options ▸ IndraLogic 2G ▸ Device editor**.

☞ Please clarify any possible modifications of the parameters with the service team.

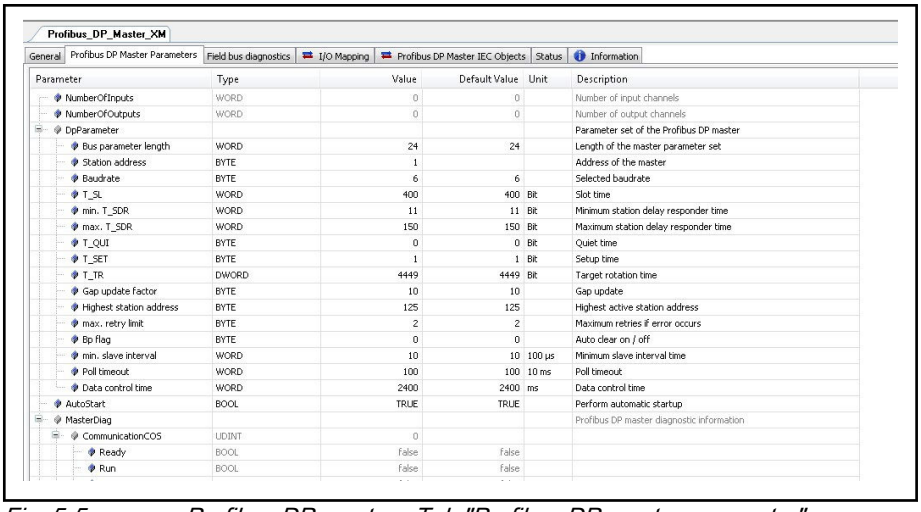


Fig. 5-5: Profibus DP master - Tab "Profibus DP master parameter"

Tab “Field bus diagnostics”

Based on the object node of the master, here: Profibus DP master, a uniform diagnostic concept was developed for the field buses of the MLC and MTX systems.

The "Field bus diagnostics" tab shows the modules applied to the "Field bus" node in online mode.

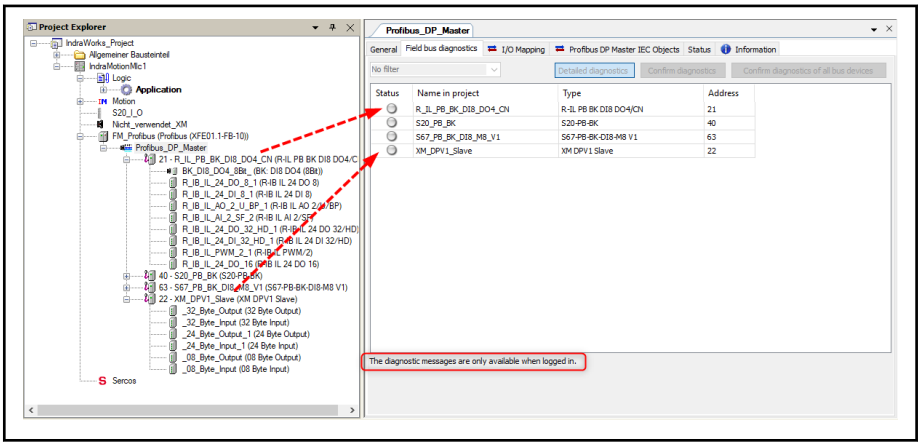


Fig. 5-6: Profibus DP master - Tab "Field bus diagnostics" (offline)

After login, the bus available at the node runs through a diagnostic cycle. The status LEDs change their color in the result of this cycle (here: green), i.e., both modules signal an error-free run.

After selecting a device **(1)**, a detailed diagnostics is available.

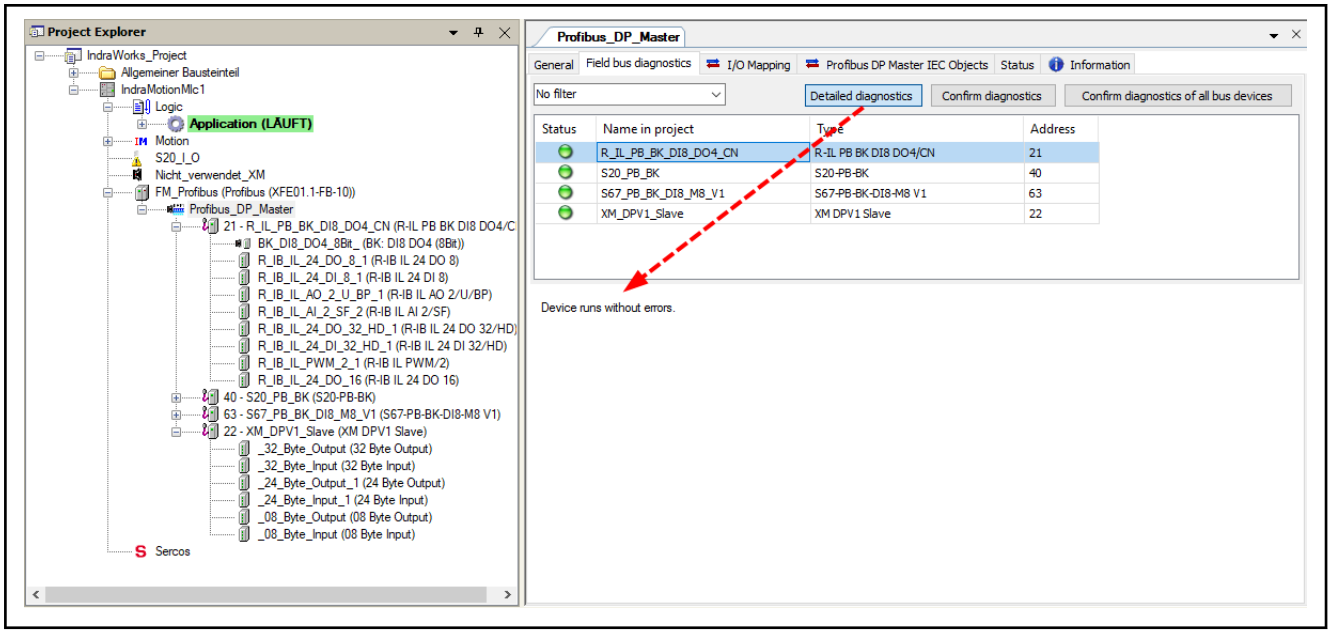


Fig. 5-7: Field bus diagnostics, control online, logged in, error-free

Error diagnostics

Login is repeated with an additional terminal not available at the real control.

A yellow warning triangle is displayed at the respective module indicating that a diagnostic message is present. The status LED of the affected module turns red. Click on "Detailed diagnostics" to provide a "Standard diagnostics" with brief information and "Extended diagnostics" with detailed specification of the error cause.

Project Explorer

- Profibus DP master: Yellow warning triangle → Diagnostics is pending
- Bus coupler (address 21): Yellow warning triangle → Diagnostics is pending
- Module (address 10): Yellow warning triangle → The detailed diagnostics shows that the module is configured in the project, but it does not exist.

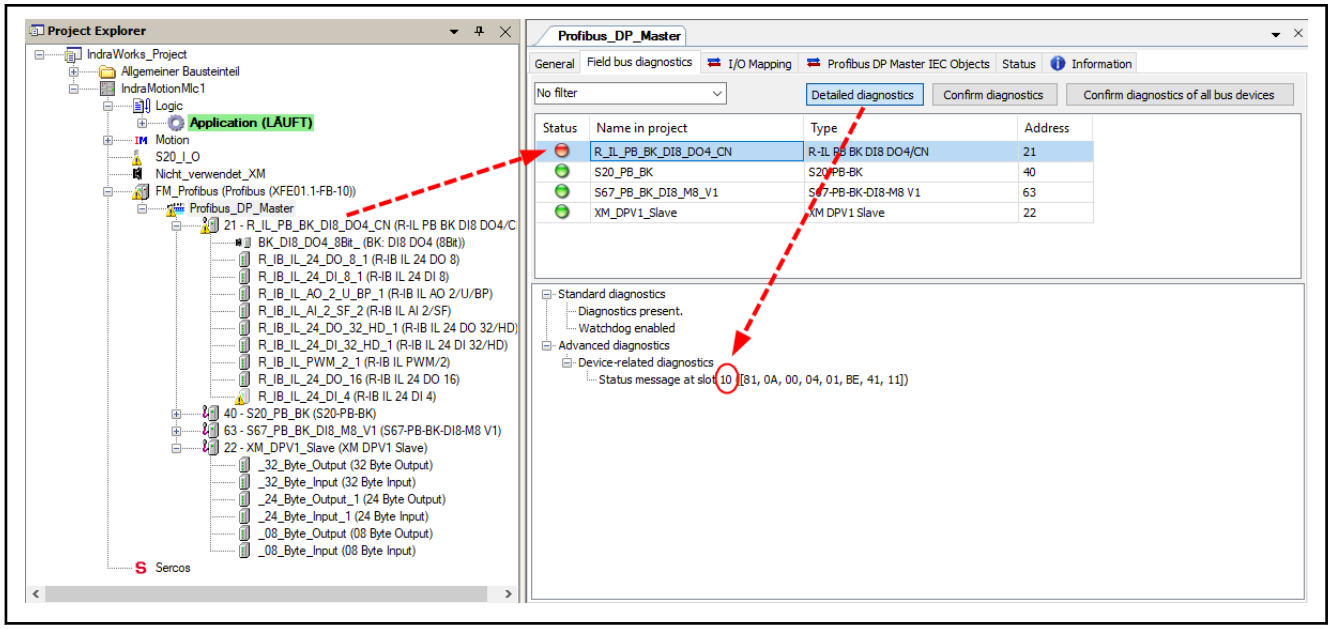


Fig. 5-8: Field bus diagnostics, control online, logged in, error

Also refer to Tab “Status”, page 56.

Confirm Diagnostics: Although there is an error, the warning triangle and the red status LED are disabled for the selected bus device. The "Extended diagnostics" of the "Detailed diagnostics" remains in the text. (The next diagnostic message can be processed...)

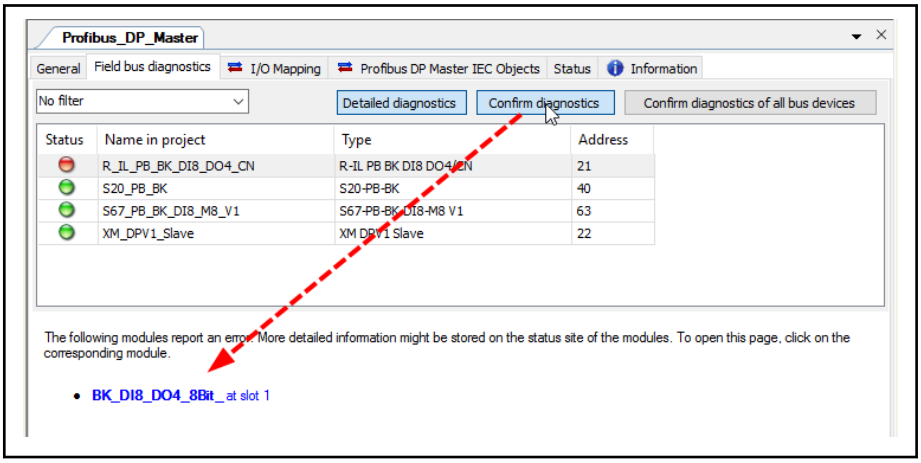


Fig. 5-9: Detailed information after "Confirm Diagnostics"

Confirm Diagnostics of All Bus Devices: Although there are errors, the warning triangle and the red status LED are turned off for all bus devices.

Tab “I/O mapping”

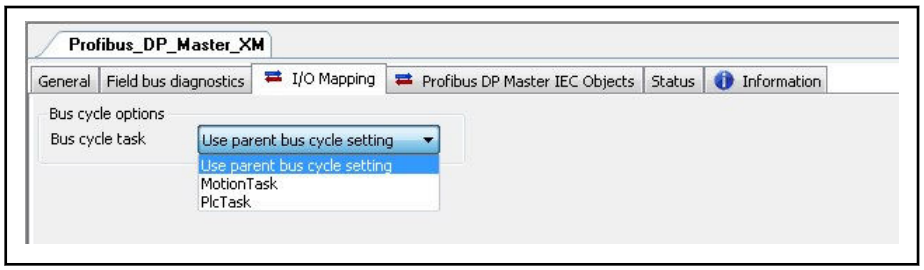


Fig. 5-10: Profibus DP master - tab “I/O mapping”

IEC objects: The data belonging to the actual bus object can be addressed as a (project-global) variable `Profibus_DP_Master_4` (Profibus DP master object with the number 4) of type `IoDrvCIFXProfibusWrapper`.

Bus cycle task: By selecting a bus cycle task, the cycle of the mapping exchange for the Profibus DP master can be connected to a particular task. In this task, it is useful to process I/O data of the master as well.

Default setting: "Use parent bus cycle setting"

With this setting, the task setting made in "Bus cycle options" of the control (double-click on the actual control in the Project Explorer, PLC settings) is applied to the actual bus.

The bus cycle can also be coupled to a PLC or Motion task.

“Profibus DP master IEC objects” tab

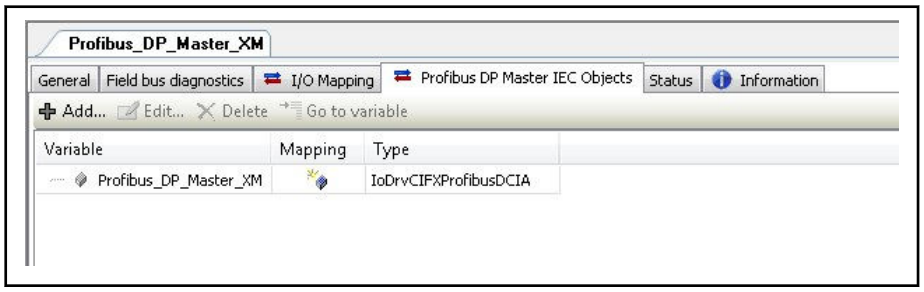


Fig. 5-11: Profibus DP master - Tab “Profibus DP master IEC objects” (offline)

IEC objects: The data belonging to the actual bus object can be addressed as a (project-global) variable `Profibus_DP_Master_4` (Profibus DP master object with the number 4) of type `IoDrvCIFXProfibusWrapper`.

Tab “Status”

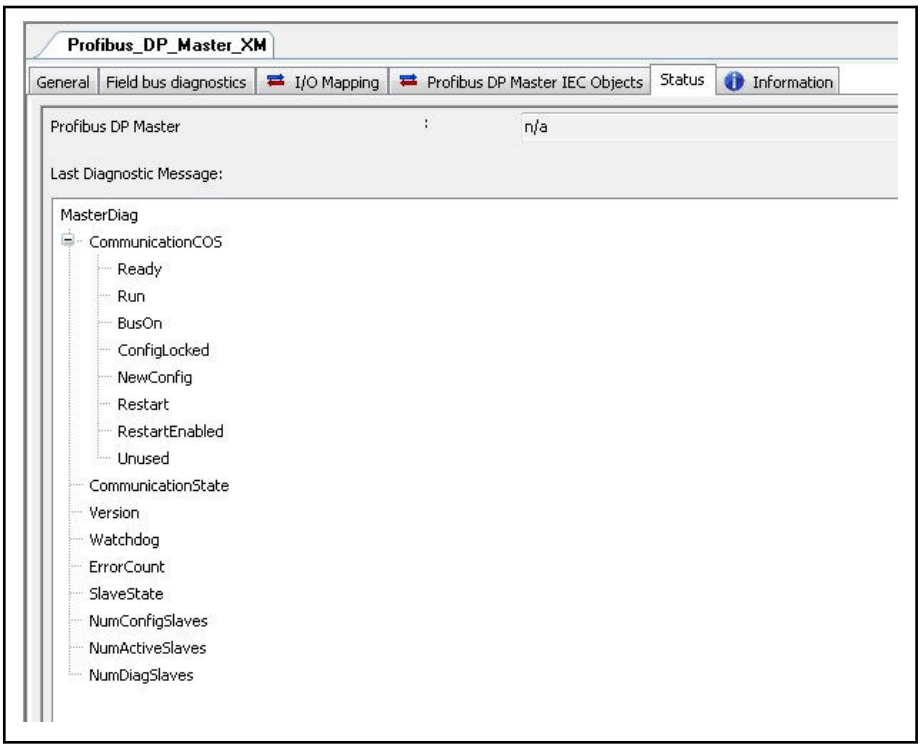


Fig. 5-12: Profibus DP master - Tab “Status” (offline)

The "Status" tab displays status information (e.g. "Running" (bus active), "n/a" (no information available)) and

specific diagnostic messages from the respective device and information on the card used and the internal bus system.

In addition, the "Last diagnostic message" is displayed. Press "Acknowledge" to confirm them.

Tab “Information”

The tab displays general information from the device description file:

Name, Vendor, Categories, Version, Order number, Description and Image if available.

5.5.2 Profibus DP master, configuring slaves

Adding slaves

The slaves are located in the "Periphery" library in the "Profibus DP" folder.

Drag the required slaves from the library to the Profibus DP master object.

In the Project Explorer, slaves can thus also be inserted between existing slaves.

Slaves can also be added in the context menu via **Add ▸ Slave ▸ ...** of the Profibus DP master object.

In this case, the new slave is added as the last slave below the Profibus DP master object.

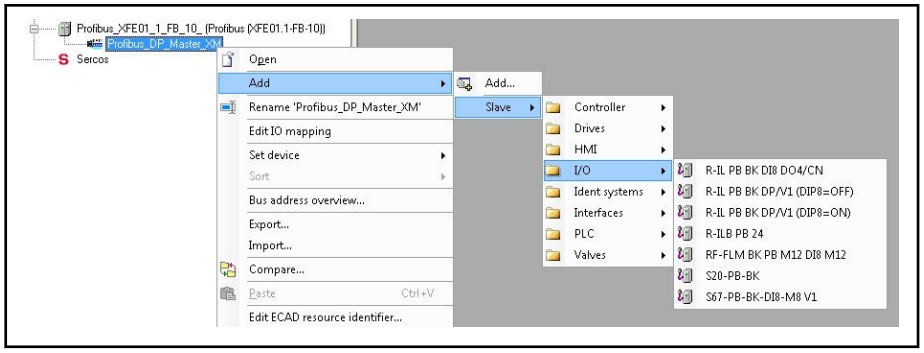


Fig. 5-13: Profibus DP master, adding slave

If a required slave is not present in the library by default, it can be integrated into the library by importing its device description file via the main menu **Tools ▸ Device Database...**.

Also refer to chapter 4.1.2 "Managing devices" on page 15.

Slaves to connect I/O modules

Profibus DP distinguishes two types of slaves to connect I/O modules:

1. **Compact**: For a compact slave, the module structure is fixedly specified.

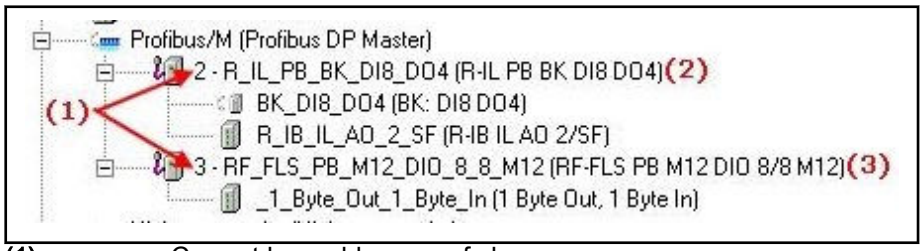
   In case of compact design, the modules below the "Slave" object node are already present after adding a slave to the Project Explorer. No more modules are available in the library.

2. **Modular**: The module structure of the slave is variable.

   The modules (terminals) can be arranged individually, but according to the device mounting specifications.

   Immediately after the slave was added to the Project Explorer, there are no subordinate I/O device levels for the slave.

   The modules for modular slaves have to be assigned manually. For information on how to insert modules, refer to chapter 5.5.3 "Profibus DP master, adding modules to slave" on page 64.

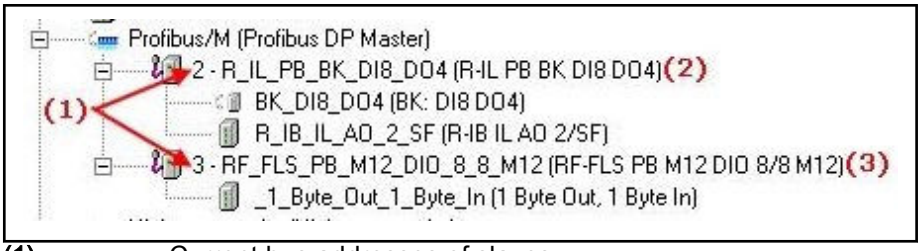


(1) Current bus addresses of slaves
(2) Modular slave with assigned modules
(3) Compact slave

Fig. 5-14: Slaves with I/O modules at the Profibus DP master object

Overview on bus addresses...

To show the complete address assignment of all bus devices, open the context menu item **Bus Address Overview...** of the Profibus DP master object.



(1) Current bus addresses of slaves
(2) Modular slave with assigned modules
(3) Compact slave

Fig. 5-15: Slaves with I/O modules at the Profibus DP master object

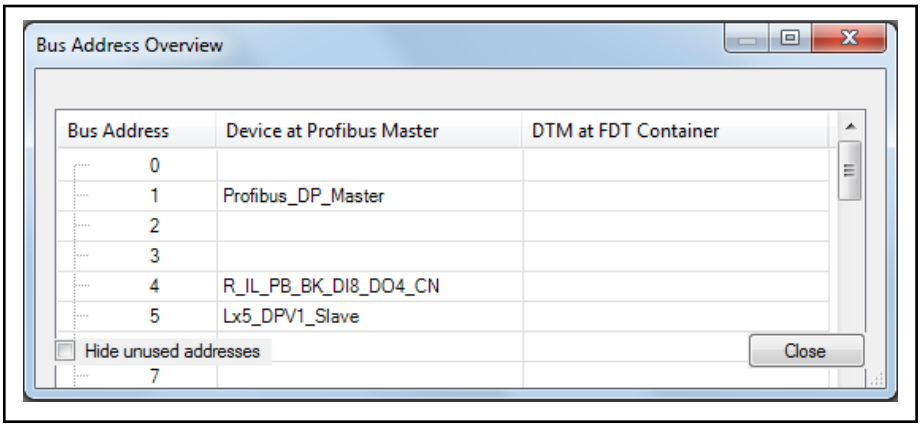


Fig. 5-16: Overview on bus addresses

Sorting field bus devices

If two or more field bus devices are available in one project under one field bus, they can be displayed in any sequence.

The devices can be sorted in ascending and descending order for name, type and address via the context menu of the field bus.

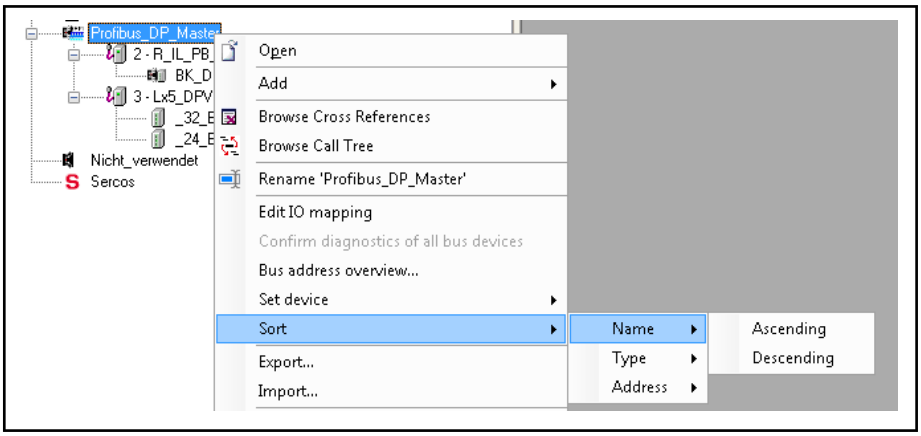


Fig. 5-17: Context menu to sort field bus devices

Enabling/disabling Profibus DP slaves

The Profibus DP slaves configured at a control can be enabled and disabled in the Project Explorer.

Thus, select/deselect the button at the icon of the Profibus DP slave.

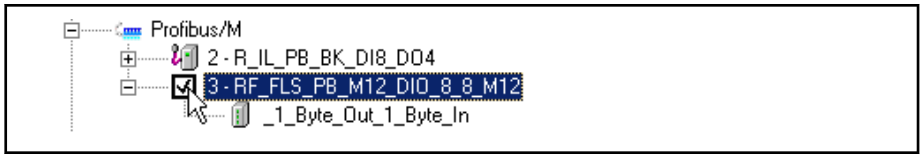


Fig. 5-18: Enabling/disabling Profibus DP slaves

The bus device is enabled if the checkbox is selected. It is considered in the diagnostics of the bus.

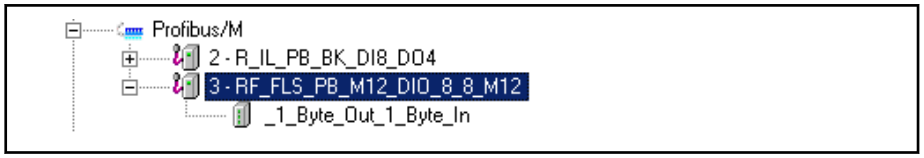


Fig. 5-19: Profibus DP slave as enabled device

The bus device is disabled if the checkbox is deselected. It is not considered in the diagnostics of the bus.

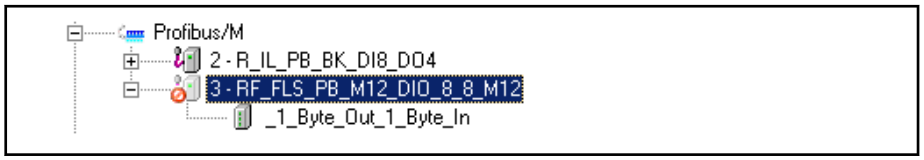


Fig. 5-20: Profibus DP slave as disabled device

If a new Profibus DP slave is created, it is enabled by default.

When loading the configuration, all enabled Profibus DP slaves are taken into account. Disabled Profibus DP slaves are ignored.



An activated, but not present bus device generates an error message in the diagnostics.

A disabled and not present bus device does not generate any error message in the diagnostics.

Also refer to Disabling Profibus DP slaves using UserDefs.cfg on page 59

Disabling Profibus DP slaves using UserDefs.cfg

Use this function to disable Profibus DP slaves configured in IndraWorks. Disable them by the definition in a stored file located on the Compact Flash card of the control.

This impedes diagnostic messages for slaves that are configured, but do not exist.

Enabling the function

The function is enabled if the valid file "UserDefs.cfg" is present in the following partitions on the control Flash Card:

For XM and VPx controls: **\USER**

Another path or file name can be defined by making entries in the "cfg file" of a system.

Example:

```
[IOSpecialDefs]
IODeactFilePath=MyFilePath
IODeactFileName=MyFileName
```



The file "UserDefs.cfg" can contain entries for more buses:

Inline modules, Sercos, Profinet I/O, Profibus DP.

Deactivation point in time

The "UserDefs.cfg" file is analyzed and the entered slaves are disabled at the moment in the context of the internal event "PrepareUpdateConfiguration".

Event takes place in case of the following actions:

- Startup after power on
- Reset (warm)
- Reset (cold)
- Load program after reset (origin)

Structure of the file "UserDefs.cfg"

The file is structured in a single or multiple existing section(s) and the contained entries.

- **Sections**: The name of a section is written in **[** and **]**. A section ends at the beginning of the next section or at the end of the file.

  The currently valid identifiers for a section are:

ProfibusDP.Master.0	First configured Profibus DP master (generally the onboard master)
ProfibusDP.Master.1	Second configured Profibus DP master (if onboard master available, Profibus DP master on function module 1)
ProfibusDP.Master.2	...
ProfibusDP.Master.3	...
ProfibusDP.Master.4	Fifth configured Profibus DP master
ProfibusDP.Master.X	Section is not evaluated

  The digits 0..4 define the Profibus DP master to be used.

  0 corresponds to the first configured Profibus DP master,

  1 corresponds to the second Profibus DP master, etc.

Example:

- The onboard Profibus DP master is enabled and one Profibus DP master function module is attached ⇒
  - "0" corresponds to the onboard master,
  - "1" corresponds to the master on the function module
- The onboard Profibus DP master is disabled and one Profibus DP master function module is attached ⇒
  - "0" corresponds to the master on the function module
- The onboard Profibus DP master is enabled and two Profibus DP master function modules are attached ⇒
  - "0" corresponds to the onboard master,
  - "1" corresponds to the master on the function module attached the closest to the control,
  - "2" corresponds to the master on the Profibus DP master function module that, in top view on the control, is attached the furthest left from the control

---

- **Special function**: Due to the definition of character **X** for the index instead of a digit, this section is ignored in the evaluation.

Several sections can be defined for the master index **0**. All their entries are assigned to the Profibus DP master.

Together with the "x-function" described above, different slave groups can be handled easily.

- **Slave entry**: An entry to disable a Profibus DP slave consists of the keyword with assignment character `SlaveDeactAddr=` and the 1- or 3-digit decimal specification of the slave address.

  Example:

  ```
  SlaveDeactAddr= 28
  ```

  The permitted values range is **1…125**.

  The value **0** is also valid, but not effective.

- **Comments**: A semicolon (**;**) is defined as comment character. The text is not analyzed from the position of the semicolon up to the end of the line.

Size of the "UserDefs.cfg" file

The memory evaluation is allocated according to the specified file size. The permitted size is thus virtually not limited.

Example of the file "UserDefs.cfg"

```
; **************
; Onboard Master
; **************

[ProfibusDP.Master.0]
SlaveDeactAddr = 5
SlaveDeactAddr = 8
SlaveDeactAddr = 25


; ************************
; Master on function module
; ************************

[ProfibusDP.Master.1]
SlaveDeactAddr = 64
SlaveDeactAddr = 30
SlaveDeactAddr = 112
SlaveDeactAddr = 3


; ****************************
; This whole section is ignored
; ****************************

[ProfibusDP.Master.X]
SlaveDeactAddr = 36
SlaveDeactAddr = 7
SlaveDeactAddr = 15
```

Configuring slaves

To open the editor window in the Project Explorer, double-click on the slave.

The dialogs provide information on the configuration of the entire slave. Modifications can be made.

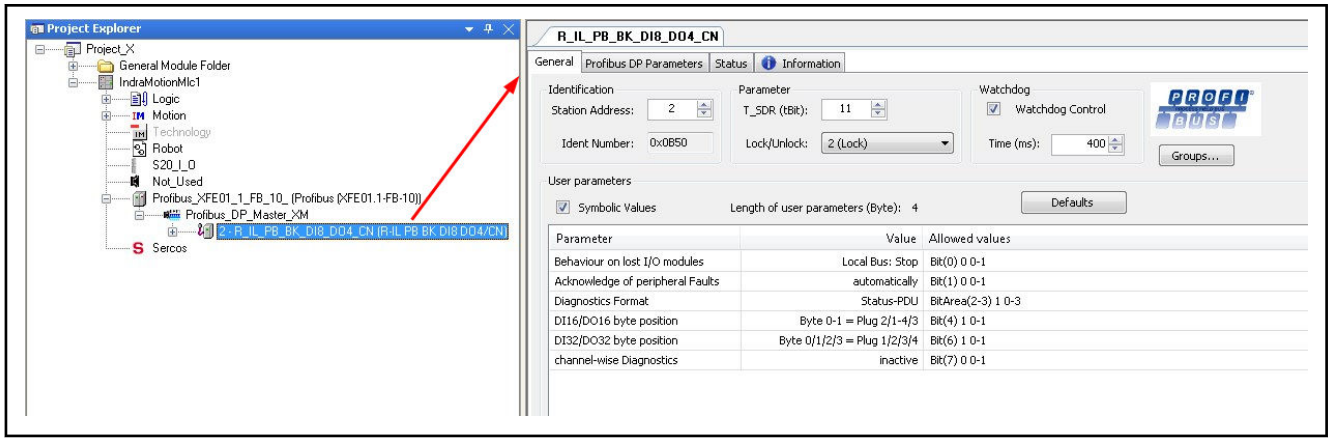


Fig. 5-21: Profibus DP master, configuring slaves

Configuration dialogs for Profibus DP master, configuring slaves

- Tab “General”, page 62
- Tab 'Profibus DP parameter', page 64, this window is used for service purposes and is only visible if the option "Show generic configuration editors" is enabled under **Tools ▸ Options ▸ IndraLogic 2G ▸ Device editor**.
- Tab “Status”, page 64
- Tab “Information”, page 64

Tab “General”

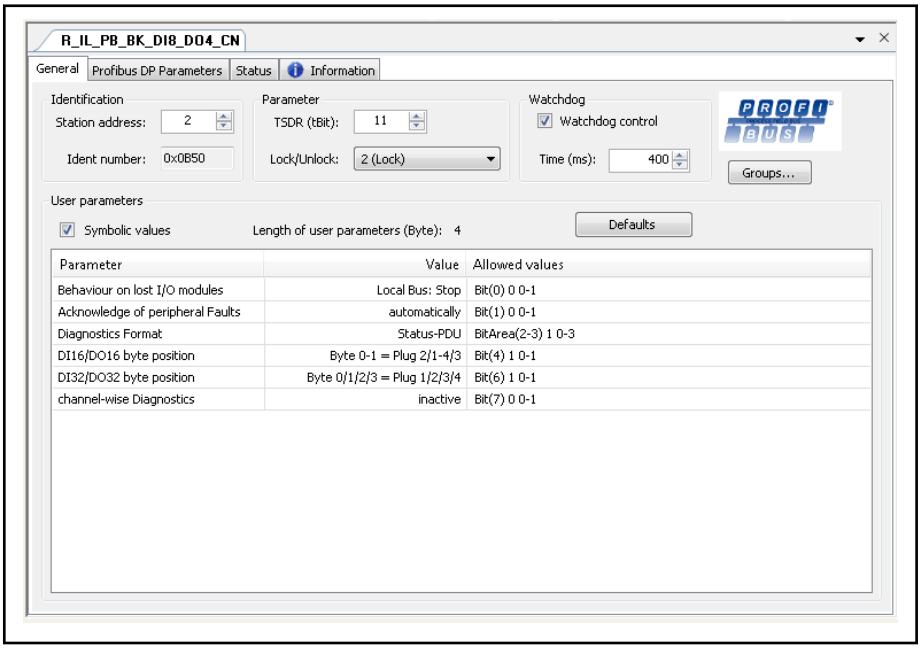


Fig. 5-22: Profibus DP slave - Tab “General”

In contrast to the DP parameters of a DP master, those of a DP slave are not a standard set, but described individually for each device in the device description. They can be modified in the "DP Parameters" dialog.

Identification

- **Station address**: The address is entered automatically. It is to be adjusted according to the actual bus configuration. The address "126" is reserved for service and commissioning and cannot be used.

- **Ident number**: Unique number assigned to this device type by the PNO (Profibus User Organization). This allows a unique reference between the DP Slave and the related device description file.

Parameters

- **T_SDR (tBit)**: Time Station Delay Responder. Response time after which the slave may respond to the master at the earliest.

  "tBit": Time unit to transfer a bit with Profibus;

  Reciprocal value of the transfer rate; e.g., 1 bit at 12 MBaud=1/12.000.000 bit/s=83 ns.

- **Lock/Unlock**: Slave is locked or unlocked for other masters. Select one of the following options from the drop-down list:
  - **0 (T_SDR unlock)**: Min. T_SDR and slave-specific parameters may be overwritten.
  - **1 (Will be unlocked)**: Slave is unlocked for other masters.
  - **2 (Lock)**: Slave is locked for other masters. All parameters are applied.
  - **3 (Unlock)**: Slave is unlocked again for other masters.

Monitoring

- **Watchdog control** If this option is enabled, the monitoring time entered applies. If the slave is not addressed by the master within this period, it returns to the initialization state.
- **Time (ms)**: Monitoring time relevant if the "Watchdog control" option is active.

Groups

- The "Groups..." button opens the "Group Properties" dialog.

  This dialog is used to assign the slave to one or more of the eight possible groups.

  In contrast, the universal group properties (sync mode and/or freeze mode) are defined when configuring the master properties, refer to chapter "Tab “General” " on page 49.

  Also use the "Global Group Properties" button to open this dialog.

  The group(s) to which the slave was assigned is/are selected with a checkmark.

  A slave device can only be assigned to groups whose properties it supports. The related properties of the respective slave (sync mode / freeze mode) are displayed above the table. The modes supported by the device are selected with a checkmark

User parameters:

- **User parameters** are, in addition to the basic DP parameters (see above), individual parameters of a DP slave displayed here if defined in the device description file.

  For each parameter, the parameter table displays the parameter name, the real or the symbolic value (see below the option "Symbolic value") and the permissible values that are also defined in the device description file. Click on the corresponding field to edit the parameter values in the "Value" column.

- **Symbolic values**: If symbolic names for the parameters are also specified in the device description file, this option can be enabled here to display these symbolic values instead of the numeric values in the "Value" column.

- **Length of the user parameters (bytes)**: Sum of the lengths of the user parameters defined in the device description file.
- **Default settings**:

  Use this button to reset the values shown in the table to the standard setting.

Tab 'Profibus DP parameter'

This tab is used for service purposes and is only visible if the option "Show generic configuration editors" was enabled. Opening via the IndraWorks menu **Tools** ▸ **Options** ▸ **IndraLogic 2G** ▸ **Device editor**.



Please clarify any possible modifications of the parameters with the service team.

The dialog displays the module parameters in detail.

Tab “Status”

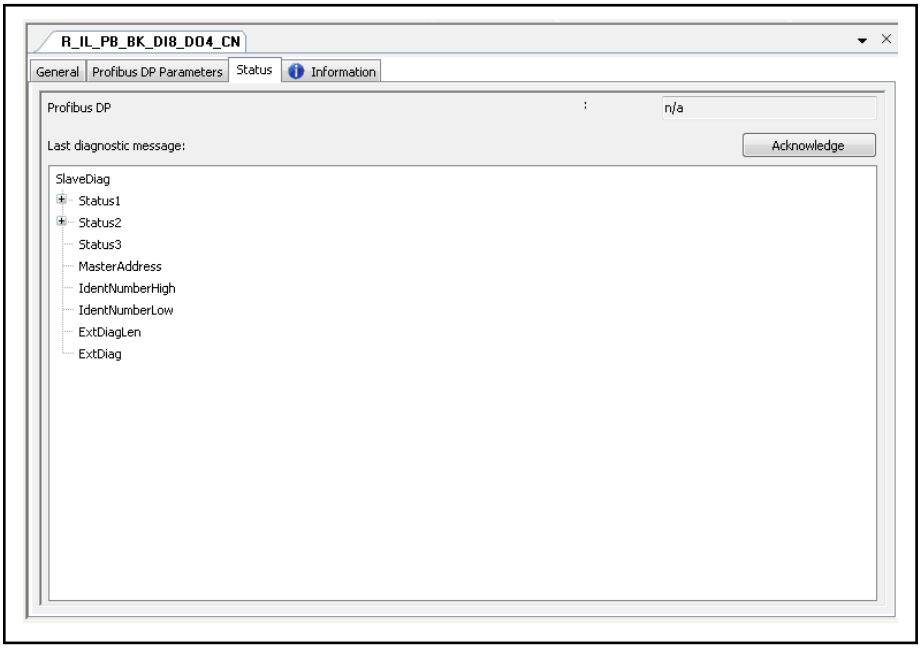


Fig. 5-23: Profibus DP slave - Tab “Status”

The "Status" tab displays status information, e.g., "Running" (bus active), and "n/a" (no information available).

Tab “Information”

The tab displays general information from the device description file:

Name, Vendor, Categories, Version, Order number, Description and Image if available.

5.5.3 Profibus DP master, adding modules to slave

General

The modules matching the respective slave are located in the "Peripherals" library in the "Profibus DP" folder under the respective slave:

- I/O: I/O modules for modular slaves.
- PLC: Declaration of commonly used memory space for data exchange between the participating controls.



I/O modules can only be added to modularly structured slaves, refer to chapter 5.5.2 "Profibus DP master, configuring slaves" on page 56.

Drag the required modules from the library to the "Slave" object.

New modules can also be inserted between existing modules in the Project Explorer.

Modules can also be added in the context menu of the slave via **Add ▸ Slave ▸ ...**.

In this case, the new module is added as last module below the slave.

The dialog to configure and parameterize the respective I/O module (terminal) is displayed via double-click on the corresponding device node in the Project Explorer.

Configuration dialog for modules at the modular Profibus slave

- "General" tab, page 65
- "I/O mapping" tab, page 66
- "Status" tab, page 67
- "Information" tab, page 67.

"General" tab

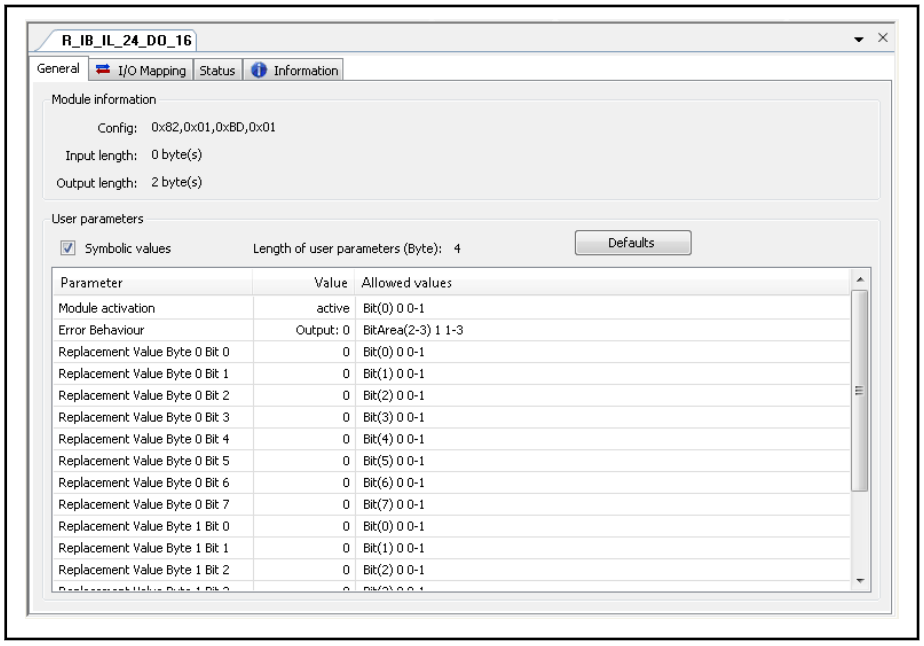


Fig. 5-24: Profibus DP master, I/O module - "General" tab

Module information

- The "Module Information" sections in the "DP Parameters" dialog of an input and output module in a Profibus DP configuration describes the settings for the configuration (module description acc. to Profibus DP standard) and the length of the input and output data (input length and output length) in bytes as defined in the device description file (GSD file).

User parameters

- **User parameters** are, in addition to the basic DP parameters (see above), individual parameters of an I/O module displayed here if defined in the device description file (GSD file) of the bus coupler.

For each parameter, the parameter table displays the parameter name, the real or the symbolic value (see below the option "Symbolic value") and the permissible values that are also defined in the device description file. Click on the corresponding field, to edit the parameter values in the "Value" column.

- **Symbolic values**: If symbolic names are also specified for the parameters in the device description file, enable this option to display these symbolic values instead of the real values in the "Value" column.
- **Length of the user parameters (bytes)**: The length of the user parameters (Length of user parameters) specifies the total length of all user parameters from the device description. Click on the respective field in the "Value" column to edit a parameter value.
- **Default settings**: Use this button to reset the values shown in the table to the standard setting.



For detailed information on the individual parameters, refer to the respective module description.

“I/O mapping” tab

The tab is used to assign inputs and outputs to variables that can be used as local or global variables in the individual POUs.

The current value of the variables is displayed in online mode.

The assignment is described under the mapping of the field bus inputs and outputs (Mapping of the field bus inputs and outputs, page 19).

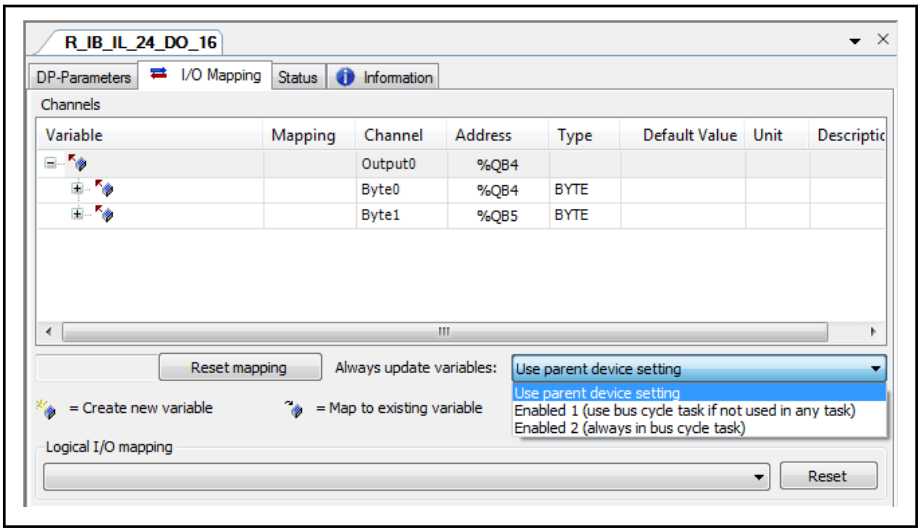


Fig. 5-25: Profibus DP master, tab “I/O mapping”

Reset mapping: Deletes the assignment made in the editor.

Always update variables: If this option is enabled, all variables are updated in each cycle of the bus cycle task irrespective of whether they are used or mapped on an input or output channel.

“Status” tab

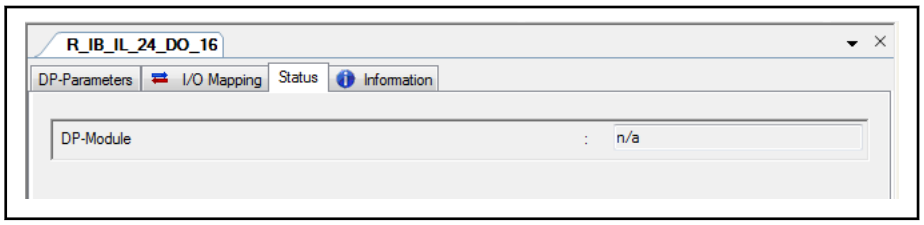


Fig. 5-26: Profibus DP master, slave modules: Status tab

- "DP modules":

  In online mode, the "DP Modules" area displays status information (e.g. "Running", "Not running (n/a)") from the control.

“Information” tab

The tab displays general information from the device description file:

Name, Vendor, Categories, Version, Order number, Description and Image if available.

5.6 Profibus DP, slave (control as slave)

5.6.1 Configuring Profibus DP slave

General

Configure a control as Profibus DP slave either when creating the control or via the context menu item **Set device** using an extension module.

It is coupled between master control and slave via the mapping memory whose content is exchanged, refer to chapter 5.6.3 "Profibus DP slave, master-slave coupling" on page 73.

To open the editor window in the Project Explorer, double-click on the Profibus DP slave object.

The dialogs provides information on the configuration of the entire slave, i.e. the control itself as slave. Modifications can be made.

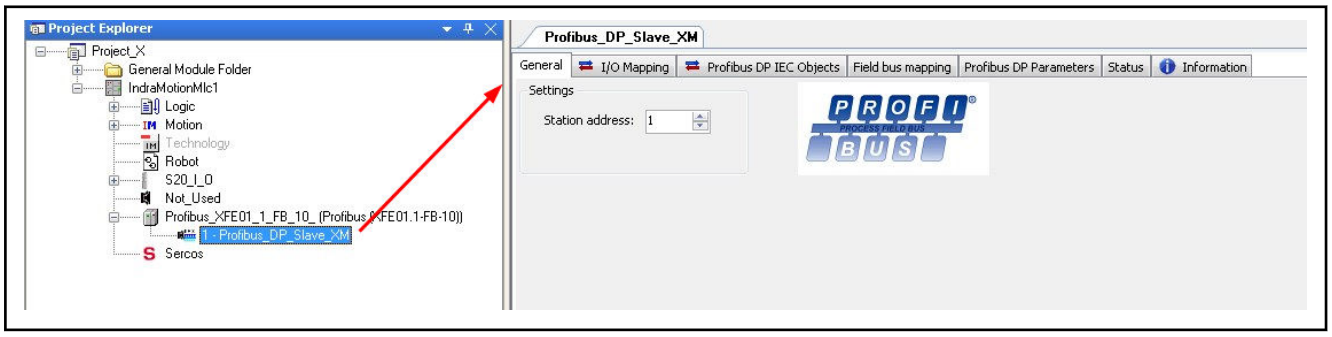


Fig. 5-27: Configuring Profibus DP slave object

Configuration dialog for a Profibus DP slave

- “Profibus DP parameters” tab, page 69
- “I/O mapping” tab, page 68
- “Field bus mapping” tab, page 69
- “Profibus DP configuration” tab, page 70, this window is used for service purposes and is only visible if the option "Show generic configuration

editors" is enabled under **Tools ▸ Options ▸ IndraLogic 2G ▸ Device editor**

- “Status” tab, page 70
- “Information” tab, page 71

“General” tab

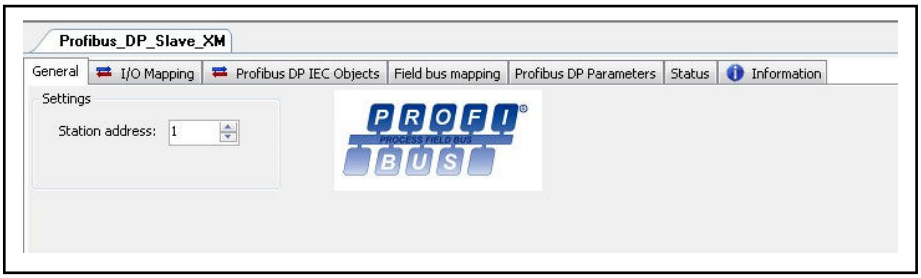


Fig. 5-28: Profibus DP slave object: DP parameters

Settings

- **Station address**: The address is entered automatically. It is to be adjusted according to the actual bus configuration.

  The number selected is to be identical to the number of the slave object to be coupled to the Profibus DP master.

“I/O mapping” tab

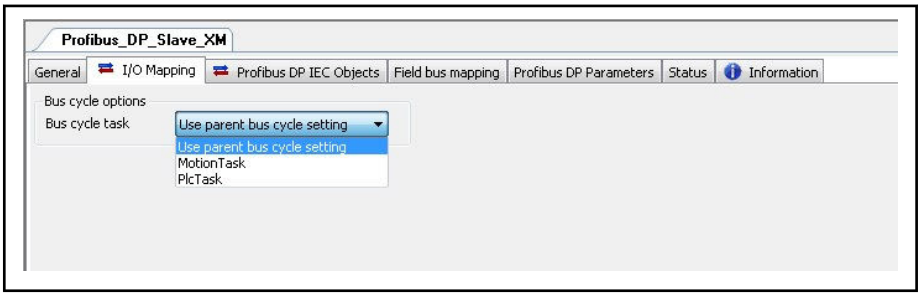


Fig. 5-29: Profibus DP slave object: I/O mapping

Bus cycle task: By selecting a bus cycle task, the cycle of the mapping exchange for the Profibus DP slave can be connected to a particular task. In this task, it is useful to process the I/O data of the slave as well.

Default setting: "Use parent bus cycle setting"

With this setting, the task setting made in "Bus cycle options" of the control (double-click on the actual control in the Project Explorer, PLC settings) is applied to the actual bus.

“Profibus DP IEC objects” tab

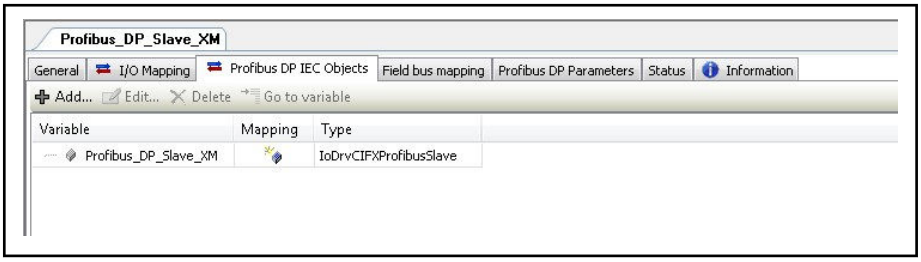


Fig. 5-30: Profibus DP slave object: Profibus DP IEC objects

The "Profibus DP IEC objects" tab displays information on the Profibus master driver that can be opened in the instruction part.

IEC objects: The data belonging to the actual bus object can be addressed as (global project) variable `Profibus_DP_Slave_1` (Profibus DP slave object with number 1) of type `IoDrvCIFXProfibusSlave`.

“Field bus mapping” tab

Use the field bus mapping to access control variables from a parent field bus master.

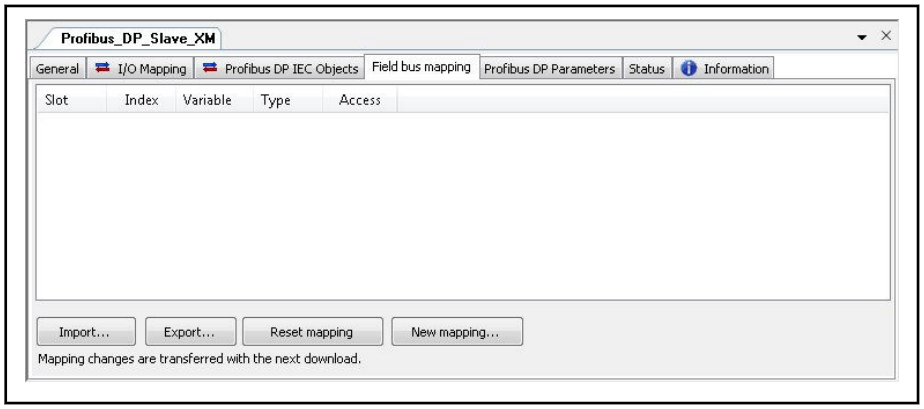


Fig. 5-31: Profibus DP slave object: Field bus mapping

To access a control variable from a parent master, assign an address to these variables first (mapping). The parent master can use this address to access the control data using acyclic services.

To use the field bus mapping, configure the control as corresponding field bus slave.

For details, refer to Field bus mapping, access from field bus master to variables of the field bus slave, page 23.

“Profibus DP parameters” tab

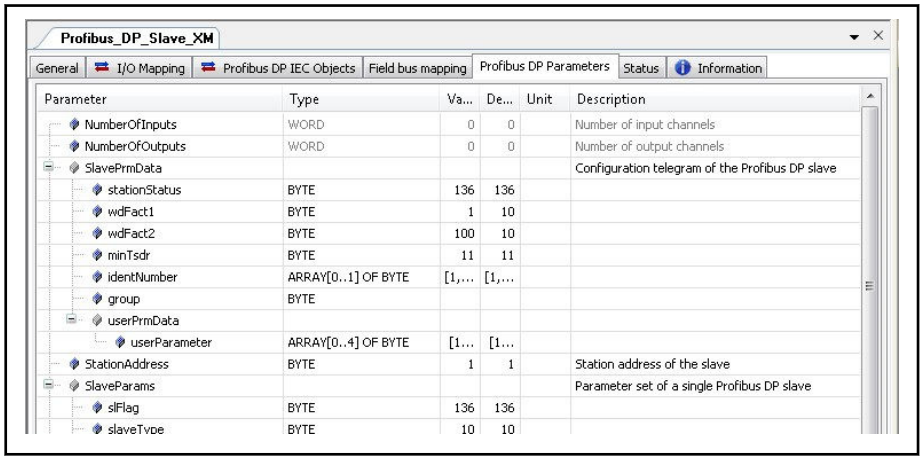


Fig. 5-32: Profibus DP slave object: DP parameters

In contrast to the DP parameters of a DP master, those of a DP slave are not a standard set, but described individually for each device in the device description. They can be modified in the "DP Parameters" dialog.

Parameters

- **T_SDR (Bit)**: Time Station Delay Responder. Response time after which the slave may respond to the master at the earliest.

  Bit: Time unit to transfer a bit with Profibus;

Reciprocal value of the transfer rate; e.g., 1 bit at 12 MBaud=1/12.000.000 bit/s=83 ns.

- **Lock/Unlock**: Slave is locked or unlocked for other masters. Select one of the following options from the drop-down list:
  - **0 (T_SDR unlock)**: Min. T_SDR and slave-specific parameters may be overwritten.
  - **1 (Will be unlocked)**: Slave is unlocked for other masters.
  - **2 (Lock)**: Slave is locked for other masters. All parameters are applied.
  - **3 (Unlock)**: Slave is unlocked again for other masters.

Monitoring

- **Watchdog control**: When this option is enabled, the monitoring time entered applies. If the slave is not addressed by the master within this period, it returns to the initialization state.
- **Time (ms)**: Monitoring time relevant if the "Watchdog control" option is active.

Groups:

Being prepared

User parameters:

- **User parameters** are, in addition to the basic DP parameters (see above), individual parameters of a DP slave displayed here if defined in the device description file.

  For each parameter, the parameter table displays the parameter name, the real or the symbolic value (see below the option "Symbolic value") and the permissible values that are also defined in the device description file. Click on the corresponding field to edit the parameter values in the "Value" column.
- **Symbolic values**: If symbolic names for the parameters are also specified in the device description file, this option can be enabled here to display these symbolic values instead of the numeric values in the "Value" column.
- **Length of the user parameters (bytes)**: Sum of the lengths of the user parameters defined in the device description file.
- **Default settings**: Use this button to reset the values shown in the table to the standard setting.

“Profibus DP configuration” tab

This tab is used for service purposes and is only visible if the option "Show generic configuration editors" was enabled. Opening via the IndraWorks menu **Tools ▸ Options ▸ IndraLogic 2G ▸ Device editor**.

☞ Please clarify any possible modifications of the parameters with the service team.

☞ When the bus is running, click on <Write Parameters> to apply the modified parameters.

“Status” tab

The "Status" tab displays status information [e.g. "Running" (bus active) and "n/a" (no information available)] and specific diagnostic messages from the respective device and on the card used and the internal bus system.

“Information” tab

The tab displays general information from the device description file:

Name, Vendor, Categories, Version, Order number, Description and Image if available.

5.6.2 Profibus DP slave, adding coupling modules

General Information

The interface between a Profibus DP slave and Profibus DP master is implemented as a common memory space. Its size can differ based on requirements.

The coupling area to exchange data with the Profibus DP master is determined with input and output modules at the Profibus DP slave object.

Add the corresponding input and output modules in the context menu of the Profibus DP slave object via **Add** ▸ **Device** ▸ **...**. The modules shown in the figure are available.

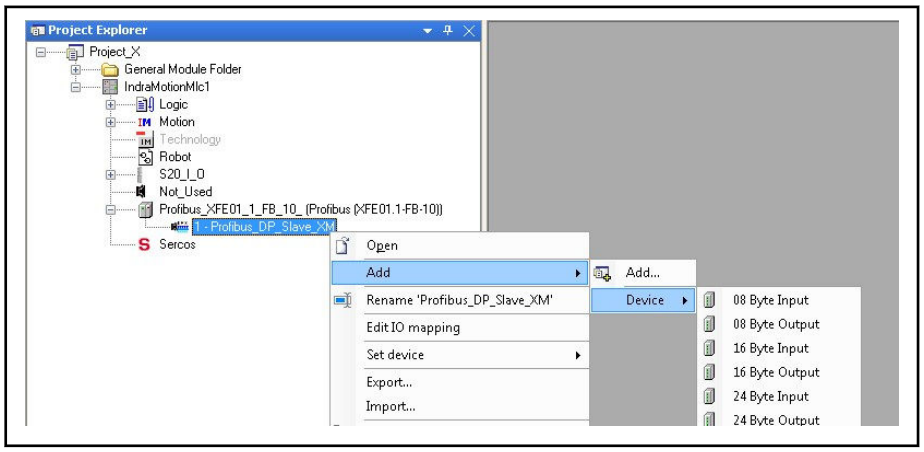


Fig. 5-33: Profibus DP slave object: Adding modules

Inputs on the Profibus DP master side become outputs on the Profibus DP slave side and vice versa.

The station numbers at the coupling position are the same.

Double-click on the module object to open the configuration dialog of the modules inserted below the Profibus DP slave object.

Also open the dialogs via the context menu item **Open** of the module. The following tabs are provided in the configuration dialog of the Profibus DP slave coupling modules:

Register

- “General” tab, page 72
- “DP module parameters” tab , page 72, this window is used for service purposes and is only visible if the option "Show generic configuration editors" is enabled under **Tools** ▸ **Options** ▸ **IndraLogic 2G** ▸ **Device editor**
- “DP modules I/O mapping” tab , page 72
- “Status” tab, page 73
- “Information” tab, page 73

“General” tab

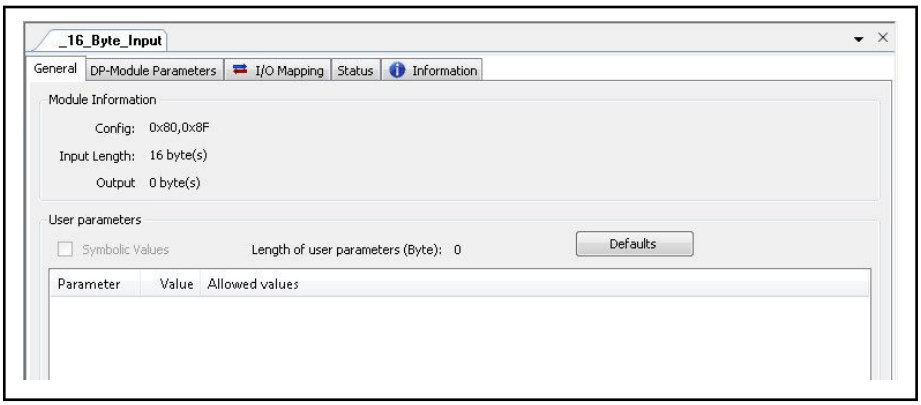


Fig. 5-34: Profibus DP slave, slave modules: General information

Module information

- The "Module Information" sections in the "General" tab of an input and output module in a Profibus DP configuration describes the settings for the configuration (module description acc. to Profibus DP standard) and the length of the input and output data (input length and output length) in bytes as defined in the device description file (GSD file).

User parameters:

→ None

“DP module parameters” tab

This tab is used for service purposes and is only visible if the option "Show generic configuration editors" was enabled. Opening via the IndraWorks menu **Tools ▸ Options ▸ IndraLogic 2G ▸ Device editor**.

Please clarify any possible modifications of the parameters with the service team.

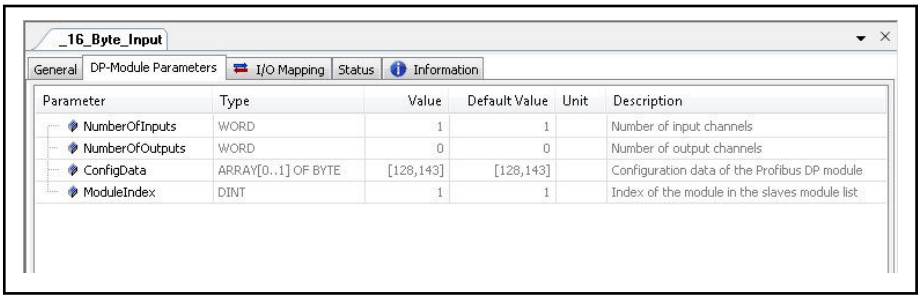


Fig. 5-35: Profibus DP slave, slave modules: Parameters

The dialog provides information on position and size of the parameters.

“DP modules I/O mapping” tab

The tab is used to assign module inputs and outputs to variables that can be used as local or global variables in the individual POUs.

The current value of the variables is displayed in online mode.

The assignment is described under the mapping of the onboard, Inline and field bus inputs and outputs (Mapping of the field bus inputs and outputs, page 19).

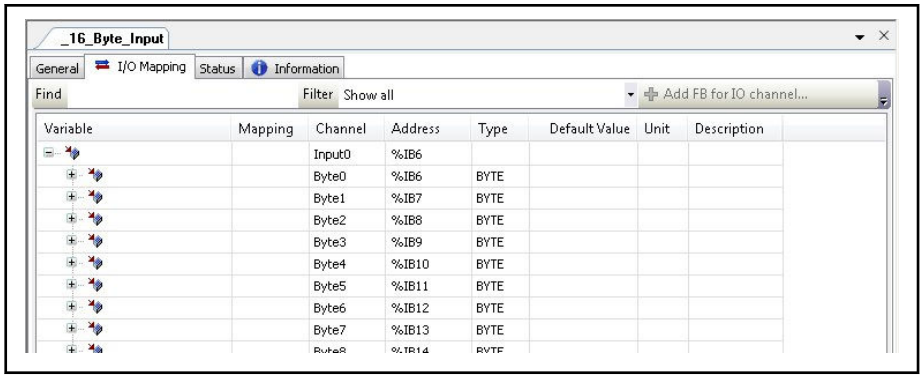


Fig. 5-36: Profibus DP slave, DP modules, I/O mapping

Reset Mapping: Deletes the assignment made in the editor.

Always update variables: If this option is enabled, all variables are updated in each cycle of the bus cycle task irrespective of whether they are used or mapped on an input or output channel.

“Status” tab

"DP modules":

In online mode, the "DP Modules" area displays status information (e.g. "Running", "Not running (n/a)") from the control.

“Information” tab

The tab displays general information from the device description file:

Name, Vendor, Categories, Version, Order number, Description and Image if available.

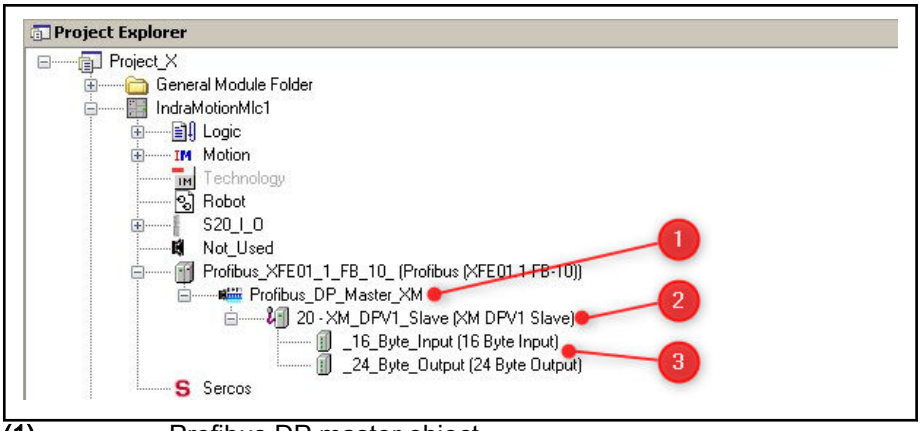
5.6.3 Profibus DP slave, master-slave coupling

To couple a control used as Profibus DP slave, five PLC coupling modules are available under **Library** ▸ **Periphery** ▸ **ProfibusDP** ▸ **PLC** that allow to couple the following controls:

- **L20 DPV1 slave** for the IndraLogic L20 DP control,
- **L40 DPV1 slave** for the IndraLogic L40 DP control and
- **Lx5 DPV1 slave** for the L25/L45/L65/L75 controls.
- **XM DPV1 slave** for the MLC XM2/XM4 controls.
- **VPx DPV1 slave** for the MLC VPx control.

When a PLC coupling modules is added to the Project Explorer, a firmware component is enabled in the Profibus DP master to connect to a Profibus DP slave control via Profibus.

Each of these PLC coupling modules is provided with a selection of modules in the device library that enable input and output variables in the mapping memory of the master to exchange data between the Profibus DP master control and the Profibus DP slave control.



(1) Profibus DP master object
(2) Slave to couple to an XM control
(3) Memory modules to exchange data between controls.

Fig. 5-37: Slave with memory modules to couple a Profibus DP slave control

The figure below shows the coupling between two controls based on Profibus DP.

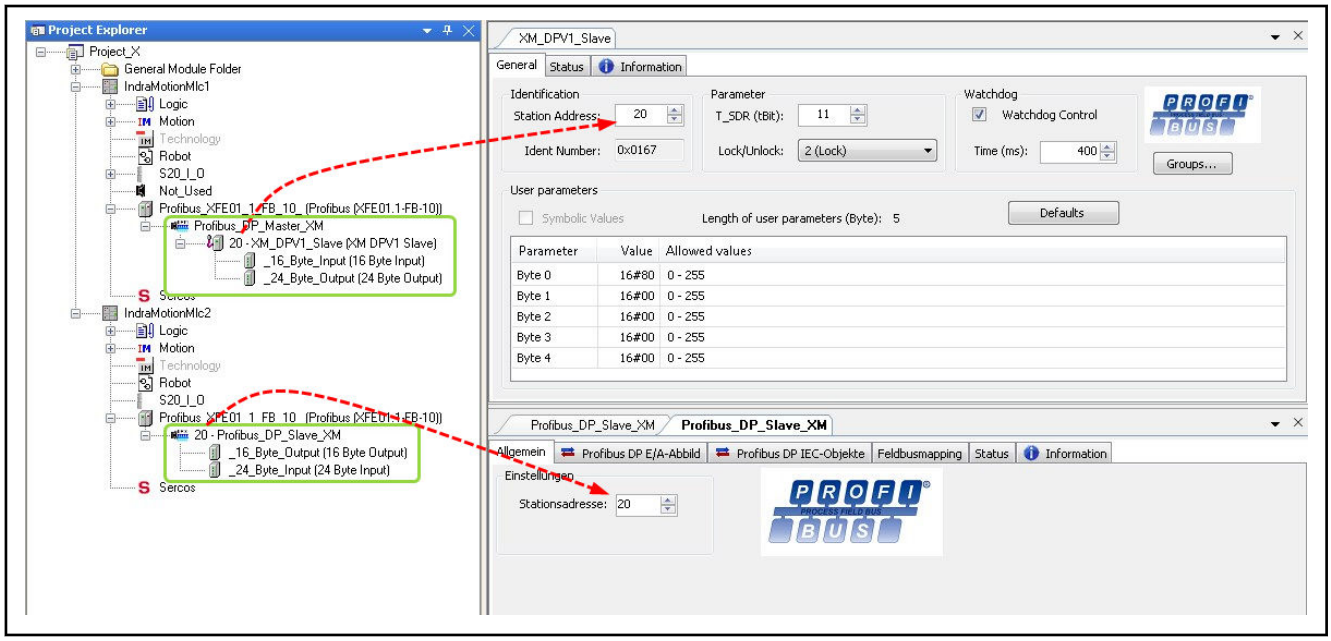


Fig. 5-38: Control coupling via Profibus DP

Inputs to the Profibus DP master side become outputs on the Profibus DP slave side and vice versa. The station numbers at the coupling position are the same.

6 Profinet I/O

6.1 Features of the Profinet I/O controller

The current implementation of the Profinet I/O controller includes the following functionalities:

- Cyclic data traffic
- Acyclic data traffic (function blocks in RIL_ProfinetIO.library)
- GSDML file import into the device database
- Bus scan of Profinet I/O devices
- Profinet I/O device-related connection status in the interface
- Diagnostics of the Profinet I/O controller via function block
- Profinet I/O device-related status via function block
- Shared Device
- Supported protocols:
  - RTC (real-time cyclic) class 1
  - RTA (real-time acyclic)
  - DCP (Discovery and Configuration)
  - CL-RPC (Connectionless Remote Procedure Call)
  - LLDP (Link Layer Discovery Protocol)
  - SNMP v1 (Simply Network Management Protocol)
- IP channel
  - Can be enabled manually with separate IP address, refer to chapter 4.6 "IP channel for Profinet I/O & EtherNet/IP (netX)" on page 32

Function/characteristics	Value
Max. number of devices, a device with Send Clock = 1 ms	64 [2) 3)]
Max. number of devices, all devices with Send Clock > 1 ms	128 [2) 3)]
Max. number of modules per device	255
Max. number of submodules (sum across all devices)	2048
Baud rate	100 MBit/s
Auto negotiation/autocrossing	Yes
Min. cycle time	1 ms [2) 3)]
Max. amount of cyclic input data	5652 bytes [1)]
Max. amount of cyclic output data	5700 bytes [1)]
Max. amount of cyclic input data per device	1440 bytes [1)]
Max. amount of cyclic output data per device	1440 bytes [1)]
Max. amount of telegram data acyclic per device/telegram	1392 bytes
Max. amount of telegram data acyclic per device/request	4096 bytes

Tab. 6-1: Technical data

[1] Including IOxS status bytes (= number of modules "E+A" + 4)

[2] If a Send Clock = 1 ms is set at one single device, up to 64 devices can be operated.



Some devices support only a Send Clock of 1 ms. A greater value cannot be set!



The bus cycle task has to meet the following condition:

Cycle time of bus cycle task ≥ (2 * smallest configured Send Clock).

In the default setting "Unspecified", the bus cycle task is the fastest cyclic PLC task, also refer to chapter 4.5 "Bus cycle task: Copying I/O data between PLC and field bus stack" on page 27. (If no devices are configured for a PNIO controller, there is an internal default send clock of 4 ms that cannot be changed).

Otherwise, a watchdog exception can result during the Profinet data exchange. The compiler warning "Critical bus cycle time ..." is also displayed in IndraWorks.

An external CIFX event is currently not supported for a bus-synchronous PLC processing.

[3] Recommendation to set the cycle times

If the prerequisites are

- maximum of 30% of a 100 Mbit full duplex connection,
- user data of 50 bytes I/O,

the following minimum cycle times are recommended:

Number of devices	Min. cycle time for all devices ("Send Clock x Reduction Ratio")
Up to 32	1 ms
Up to 64	2 ms
Up to 128	4 ms



If the connections are sporadically aborted in the cyclic communication, set a higher cycle time ("Send clock x Reduction ratio"). Amongst others, the reason can be a limited data throughput in switches or in PN devices (e.g. line topologies).

6.2 Features of the Profinet I/O device (control as slave)

General information

The current implementation of the Profinet I/O device includes the following functionalities:

- Cyclic data traffic
  - Real time
  - Isochronous real time, no synchronous application
- Acyclic data traffic
  - Field bus mapping
- Profinet I/O device-related connection status in the interface

- Profinet I/O device diagnostics via function block
- Shared Device
- Supported protocols:
  - RTC (real-time cyclic) class 1
  - RTC (real-time cyclic) class 3
  - RTA (real-time acyclic)
  - DCP (Discovery and Configuration)
  - CL-RPC (Connectionless Remote Procedure Call)
  - LLDP (Link Layer Discovery Protocol)
  - SNMP (Simple Network Management Protocol)
  - MRP client (Media Redundancy Protocol – Client)

Future extensions are planned for the following functionalities:

- Alarms



Each configured Profinet I/O device requires 8.6 kB of the persistent retain data of the IEC program.

For configuration notes, refer to chapter "Profinet I/O device - Device editor" on page 120.

Function/characteristics	Value
Max. number of I/O modules	128 slots (independent of direction)
Max. number of submodules	132 (4 headends + 1 per I/O module)
Max. number of I/O connections	4
Baud rate	100 MBit/s
Auto negotiation/autocrossing	Yes
Min. cycle time (RT / IRT as of 14V22)	RT: 1 ms IRT: 250 µs
Max. amount of cyclic input data	1440 bytes
Max. amount of cyclic output data	1440 bytes
Max. telegram data per acyclic telegram	8192 bytes
Identification & Maintenance	I&M 0-4

Tab. 6-2: Technical data

6.3 Profinet I/O, basics

6.3.1 Terms and abbreviations

Controller The Profinet I/O field bus master is called **controller**.

Device The Profinet I/O field bus slaves are called **devices**.

Station name The criterion to address a device is the device name (= station name).

GSDML file This is a device description used by IndraWorks to make the Profibus IO devices known.

RT / IRT – RT Class1 / RT Class3

Real Time (RT) or RT Class1 is the unsynchronized real-time communication. Each device runs its own cycle.

Isochronous Real Time (IRT), or RT Class3 is the synchronized real-time communication. The cycle time starts for all devices at the same point in time. A time slot is assigned to each device to send cyclic data.

It is not mandatory to route this synchronous cycle up to the application. The application can also run asynchronously even though the bus communication is synchronous.

6.3.2 Meaning of Device Names in case of Profinet IO

Name-based device addressing

For Profinet, a logic, individual and unique device name (device name, station name, IdentStr) is assigned to each device (I/O device).

The Profinet I/O controller uses this device name to assign the configured I/O address to the I/O devices.

In the following, the actual Profinet data traffic takes place via this IP address and thus the MAC address.

When configuring the Profinet IO controller, note that the station name is set as to be used with the I/O devices.

Naming I/O devices - Mainly used procedures

1. The station name can be edited or configured on the device using a configuration tool.

   Example:

   IndraWorks is used to configure if there is a control as I/O device.

   A station name is determined during configuration and loaded to the control together with the application.

2. The device name is set on the hardware of the device, e.g. via Dip switch or jumper.

   Example: S67-PN-BK

3. The device name is written to the device via IndraWorks (and saved in the device).

   Open the "Scan for devices" dialog via the context menu of the Profinet I/O controller in the Project Explorer and name the device.



The station name is used to address the Profinet I/O device.

Missing or deviating IP address settings at the device are ignored.

The station name may consist of the following characters:

- Lower case letters without umlauts "a" to "z".

  Upper case letters "A" to "Z" are allowed, but internally converted into lower case letters.

  The station names are not case sensitive.
- Digits from "0" to "9" (may not be at the beginning of the name).
- Special hyphen character "-" and dot "." (may neither be at the beginning nor at the end of the name).
- The name may not exceed 240 characters. For some devices/vendors, a limit of 127 characters apply.

  The name can be divided into substrings using "." dots. Such a substring may not exceed 63 characters.

  The name may not start with the characters string "port-xyz" (with x,y,z = 0...9).

  The name may not have the sequence "a.b.c.d" (with a,b,c,d = 0...999).

6.3.3 Shared Device

In case of a “Shared Device”, multiple Profinet I/O controllers share the (sub)modules of a Profinet I/O device. Only one connection can be established to each (sub)module at a time. No connection is prioritized. The following principle applies: „First come, first served“.

Example in the following figure:

Two XM controls as Profinet I/O controller when exchanging data with S20-PN-BK+ (below).

The I/O communication relation between the Profinet I/O controller and the Profinet I/O device is shown in colors (blue, red).

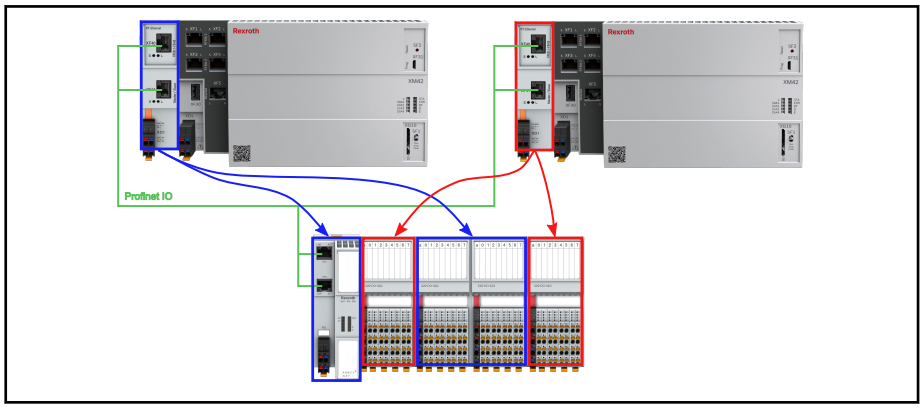


Fig. 6-1: Profinet I/O - Shared device topology

6.3.4 Media Redundancy Protocol (MRP)

The "Media Redundancy Protocol" allows to establish an Ethernet ring redundancy. Each device being part of the ring has to support MRP. There are different roles in one MRP ring:

- The MRP Redundancy Manager (RM) that monitors the ring state, separates it logically and switches from ring to line upon ring break.
- The MRP client configured as ring device by a parent Profinet I/O controller and supporting the switching mechanisms.

Switching times at ring break are typically ~200 ms, depending on ring size, number of devices and Ethernet cable length. The cycle time monitoring has to be selected higher to avoid that the connection is not aborted upon ring break.

6.3.5 Identifying

To assign the device names to a hardware in the Profinet I/O network, there is a Profinet service responsible to flash the devices.

Refer to the Profinet I/O controller dialog "Scan devices", chapter "Scanning devices " on page 109, command "Blink LED".

VPx	During the identification, the text "Identification Profinet device" is shown in the dialog "Diagnostics and settings".
XMx	If the identification was successful, the LED DIAG3 LED flashes three time within three seconds.
XFE01.1-FB-03/04	If the identification was successful, the LED SF-LED flashes three time within three seconds.

Tab. 6-3: Identifying Rexroth controls with a Profinet I/O interface

6.4 Profinet I/O, configuration overview

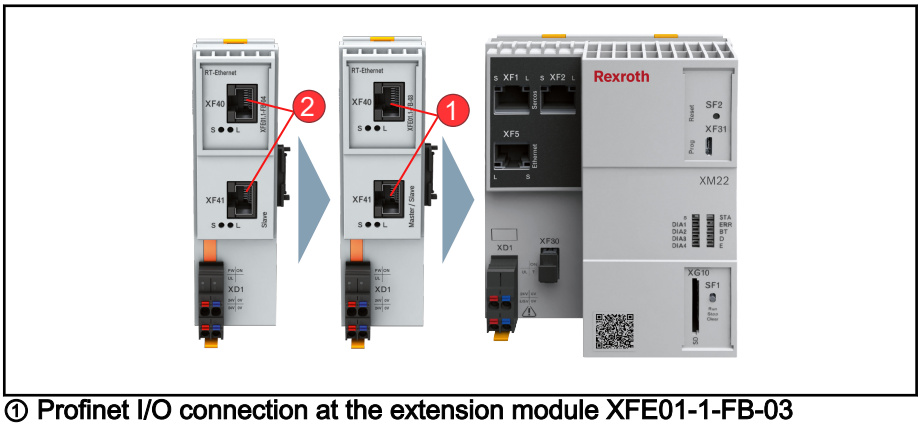
XM and VPx controls support Profinet I/O.

The following chapter provides an overview on the connection options on the respective control type.

Connecting to XM2 controls

For XM2 controls, Profinet I/O is connected via extension modules:

- Up to three extension modules **XFE01-1-FB-03**, each configurable as Profinet I/O controller or a Profinet I/O device
- Up to three extension modules **XFE01-1-FB-04**, configurable as Profinet I/O device



① Profinet I/O connection at the extension module XFE01-1-FB-03
② Profinet I/O connection at the extension module XFE01-1-FB-04

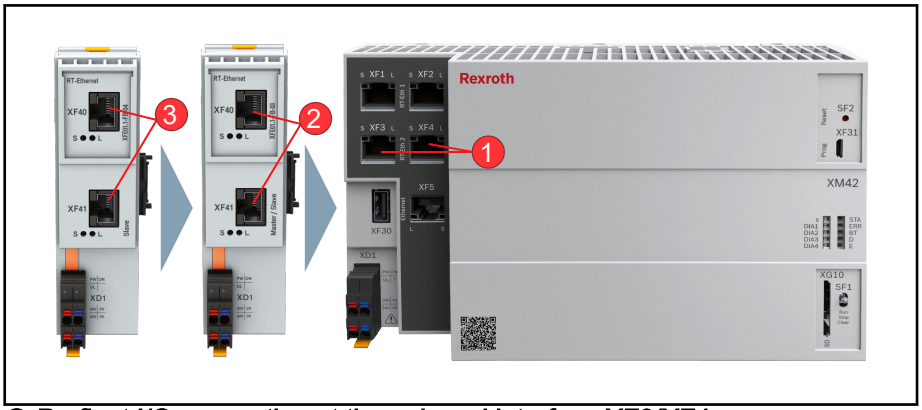
Fig. 6-2: Profinet I/O – Connection to the XM2 control

Both connections at the extension module are equal and can be used to connect to the Profinet I/O devices (no dedicated input and output).

Connecting to XM4 controls

For XM4 controls, Profinet I/O is connected either via the onboard interface of the control or via extension modules:

- Onboard interface **XF3/XF4** configurable as Profinet I/O controller or device
- Up to three extension modules **XFE01-1-FB-03**, each configurable as Profinet I/O controller or a Profinet I/O device
- Up to three extension modules **XFE01-1-FB-04**, configurable as Profinet I/O device



① Profinet I/O connection at the onboard interface XF3/XF4
② Profinet I/O connection at the extension module XFE01-1-FB-03
③ Profinet I/O connection at the extension module XFE01-1-FB-04

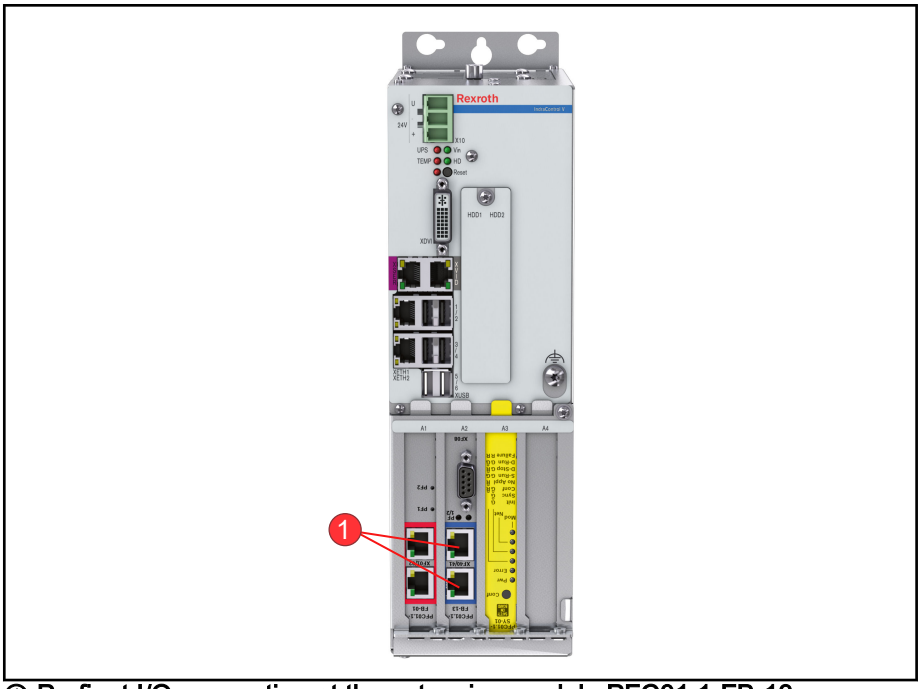
Fig. 6-3: Profinet I/O – Connection to the XM4 control

Both connections at the extension module and at the onboard interface are equal and can be used to connect to Profinet I/O devices (no dedicated input and output).

Connecting to VPx controls

For VPx controls, Profinet I/O is connected via extension modules:

- Up to two extension modules **PFC01.1-FB-13**, each configurable as Profinet I/O controller or device



① Profinet I/O connection at the extension module PFC01.1-FB-13

Fig. 6-4: Profinet I/O – Connection to the VPx control

Both connections at the extension module are equal and can be used to connect to the Profinet I/O devices (no dedicated input and output).

Profinet I/O – Bus topology

A simple topology consists of the control as Profinet I/O controller and the Profinet devices, e.g. I/O blocks and Profinet I/O drives connected to the Profinet I/O controller via Profinet.

The XM and VPx control hardware does not support any Profinet I/O ring (MRP/redundancy class 1)!

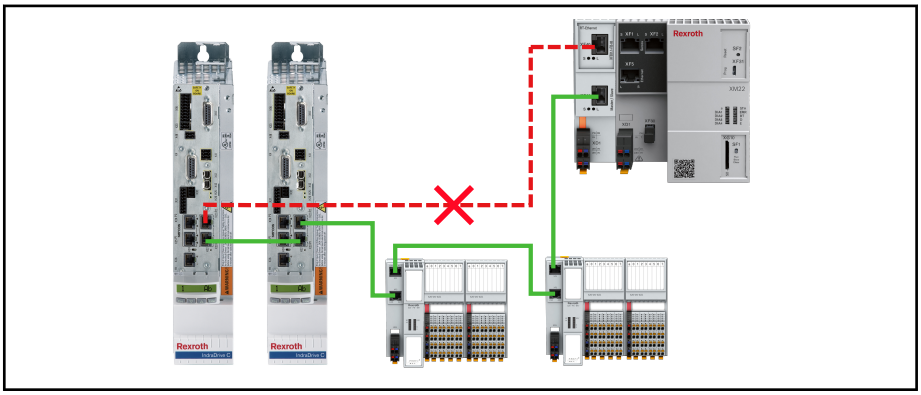


Fig. 6-5: XM2 control with the extension module XFE01-1-FB-03 as Profinet I/O controller with I/O blocks and drives

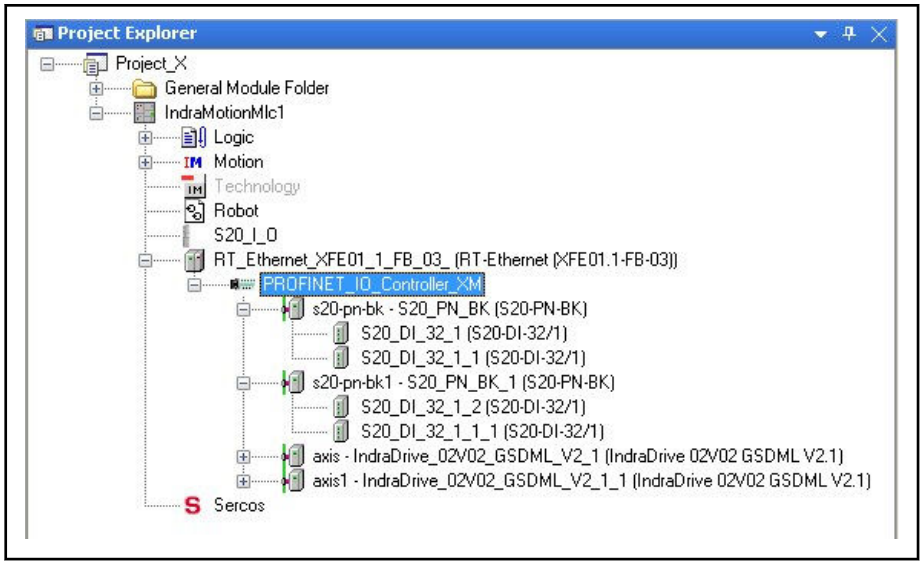


Fig. 6-6: Example: Profinet I/O configuration in the Project Explorer

The following figure shows an example for a more complex topology:

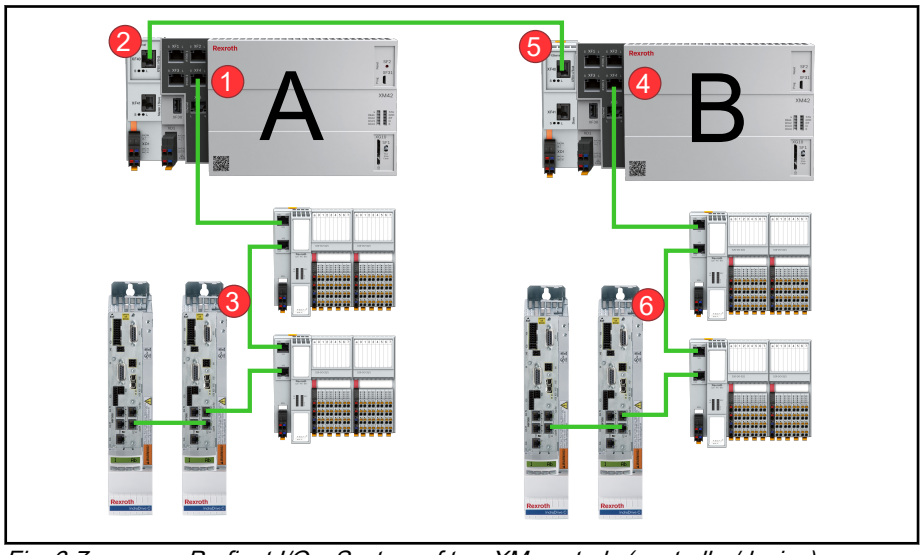


Fig. 6-7: Profinet I/O – System of two XM controls (controller/device)

The topology consists of two XM controls interconnected via Profinet and individually provided with a own, subordinate Profinet system with I/O component groups and drives.

The hierarchy in the complete system (controller/device) is designed as follows:

- Control **"A"** is the primary control (controller).

  The primary control commands the subordinate Profinet devices ③ using the onboard interface ①.

  The primary control additionally commands the secondary control **"B"** (device) using the connected extension module ②.

  The interfaces are configured as follows:

  – The onboard interface XF3/XF4 ① is configured as controller and commands the connected Profinet I/O devices (I/Os & drives).

– The extension module XFE01-1-FB-03 ② is also configured as controller and commands the control **"B"** (device).

- Control **"B"** is the secondary control (device).

  The primary control **"A"** commands the secondary control using the connected extension module ⑤.

  The secondary control commands the subordinate Profinet devices ⑥ using the onboard interface ④.

  The interfaces are configured as follows:

  – The extension module XFE01-1-FB-04 ⑤ operates only as Profinet I/O device and accepts the commands of the primary control **"A"**.

  – The onboard interface XF3/XF4 ④ is configured as controller and commands the connected Profinet I/O devices (I/Os & drives).

6.5 Profinet I/O, controller

6.5.1 Profinet I/O controller, configuring

Device editor

To open the device editor, double-click on the Profinet object in the Project Explorer.

Configure the settings of the Profinet object in the tabs.

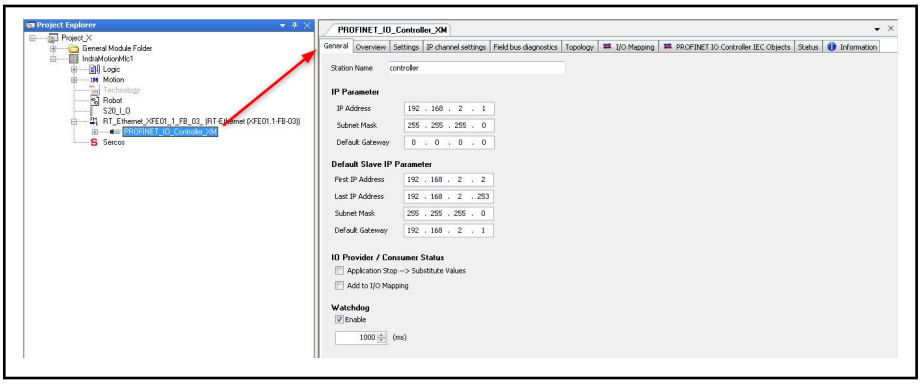


Fig. 6-8: Profinet I/O controller – Generic device editor

Tab “General”

The station name identifies the Profinet I/O controller (like the slaves).

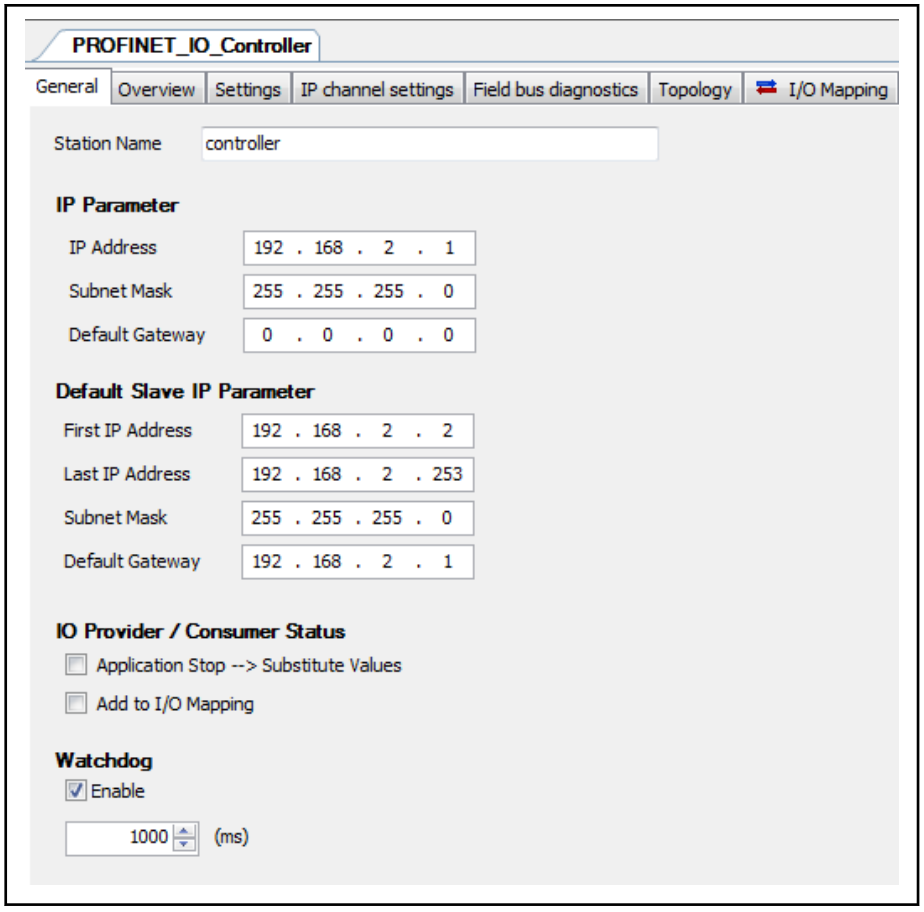


Fig. 6-9: Profinet I/O controller – Tab "General"

Station name

- Station name: Default setting "controller"

☞ *The station name may consist of the following characters:*

- Lower case letters without umlauts "a" to "z".

  Upper case letters "A" to "Z" are allowed, but internally converted into lower case letters.

  The station names are not case sensitive.
- Digits from "0" to "9" (may not be at the beginning of the name).
- Special hyphen character "-" and dot "." (may neither be at the beginning nor at the end of the name).
- The name may not exceed 240 characters. For some devices/vendors, a limit of 127 characters apply.

  The name can be divided into substrings using "." dots. Such a substring may not exceed 63 characters.

  The name may not start with the characters string "port-xyz" (with x,y,z = 0...9).

  The name may not have the sequence "a.b.c.d" (with a,b,c,d = 0...999).

☞ Each device has to have a unique station name in the subnet for certain functionalities in network mode!

IP parameters

The following addresses have to be configured according to the current environment to identify the controller:

- IP address: Default setting "192.168.2.1"



Note the current address settings for devices added to the configuration via scanning when adjusting the IP address to prevent double assignments (Default slave IP parameter, page 86)!

- Subnet mask: Default setting "255.255.255.0"
- Default gateway: Default gateway, default setting "0.0.0.0" (gateway address optional)

Example: IP addresses after scanning

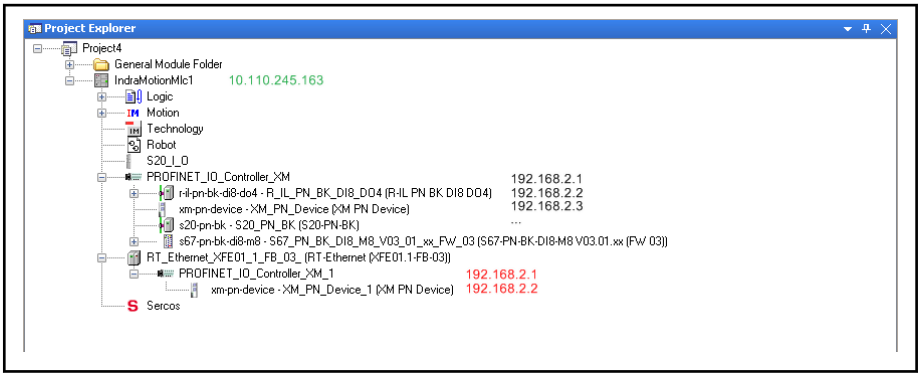


Fig. 6-10: IP addresses after scanning the controller

The control is located in three independent networks:

- Green (top): Enterprise network (engineering interface)
- Black (middle): PNIO controller on the onboard interface
- Red (bottom): PNIO controller via the extension module XFE01-1-FB-03



Although the networks of the two controllers are independent of each other, use a different subnet, e.g., 192.168.**3**.1. on the second controller to avoid possible problems.

Default slave IP parameter

The address settings for devices are only used if devices found while scanning the hardware (**Project ▸ Scan devices...**) are mapped to the Project Explorer and no address definitions were selected in the "Scan" dialog.

The values defined in this dialog are automatically applied to the devices added. For the IP address, the next free address from the defined range is set.

The default settings are taken from the device description file. To edit them, set the cursor to the respective input field:



To prevent double assignments when adjusting these addresses, note the current IP address of the controller!

- First IP address: Default setting "192.168.2.2"
- Last IP address: Default setting "192.168.2.254"
- Subnet mask: Default setting "255.255.255.0"
- Default gateway: Default gateway; default value "0.0.0.0"

I/O provider/consumer status

- **Application stop → Substitute values**

   : If the user stops the application, the provider status is set to "BAD".

The slaves then set the inputs/outputs to predefined substitute values.

- **Add to I/O mapping**

   : The incoming status information is added to the I/O mapping for all modules (provider state to input data and consumer state to output data).

Watchdog

- If the watchdog is "enabled", the monitoring time entered applies. Thus, upon timeout without the controller being active, a device-specific response results, e.g. an error message.
- Time (ms): Monitoring time, default setting 1000 ms. Possible values: 0 to 65535.

The defaults for the watchdog monitoring are taken from the device description.

Tab “Overview”

The tab provides an overview on all devices including their communication settings.

Station name, IP address and subnet mask can be changed directly in the list.

The dialog "Communication settings" - that can also be opened as context menu of the selected line - opens for the other parameters.



Multiple devices can be selected at a time and the settings can also be set simultaneously in the selected devices.

- **Default IP** command:

  Sets the IP address of the selected devices to a value within the default address range.

  The address range is defined in the controller settings.

- **I/O performance** command:

  Opens the "Communication settings" dialog.

  – "**Communication settings**" dialog:

    In the communication settings, the transmission rate of I/O data can be configured for all selected devices.

    A value range valid for all selected devices can be selected for the individual parameters (RT-Class, Send Clock, Reduction Ratio).

    Refer to chapter "Tab “General” " on page 98.

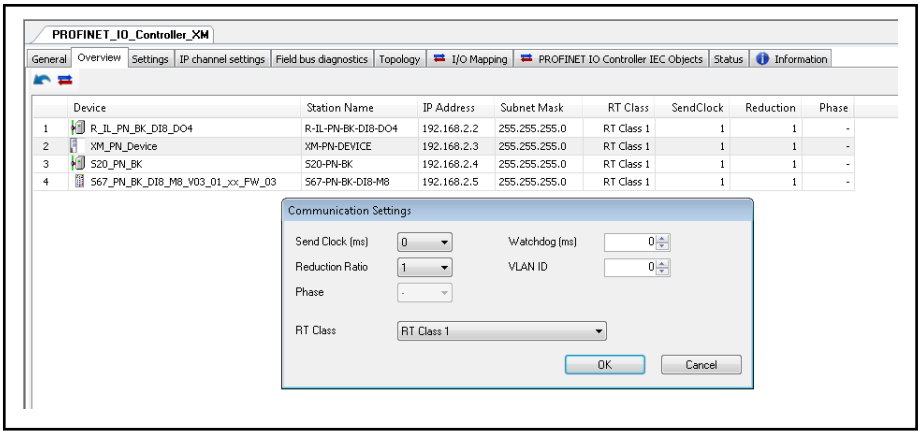


Fig. 6-11: Profinet I/O controller – "Overview" tab

Tab “Settings”

The tab provides additional settings for the PNIO controller.

Disabled I/O modules

I/O modules of PNIO devices can be disabled:

- In the Project Explorer, refer to chapter "Enabling/disabling Profinet I/O modules" on page 106
- Or by using a **UserDefs.cfg** file on the control, refer to chapter "Disabling Profinet I/O devices and Profinet I/O modules using UserDefs.cfg" on page 94

The following options can be selected to handle the slot configuration of the devices (each PNIO controller instance):

- **Slot compression** (default):

  The slot configuration of the enabled modules is "compressed" ensuring a continuous slot numbering without gaps of the enabled modules.

  ⇒ *Use case:* A "Maximal project" configuration is created and only a subset of the configured I/O modules is available for a specific system, i.e. the modules that do physically not exist are disabled in the configuration using the previously mentioned methods.

  For PNIO device with slot numbers corresponding to the slot number at the local bus (e.g. IL PN BK DI8 DO4; S20-PN-BK/+; Siemens ET200S) and thus a continuous numbering without gaps for the physically existing modules, a slot configuration without gaps has to be created respectively upon the enabled module configuration (= "slot compression").

- **No slot compression for shared devices**:

  For PNIO devices that can be operated as "Shared Device", there is no slot compression (= if "Shared device" is selected in the slave dialog "Options", irrespective of whether this option is selected).

  ⇒ *Use case:* In case of a shared device application, a PNIO controller typically operates only a subset of physically existing modules of a PNIO device. Modules not operated by the PNIO controller have to be configured, but have to be disabled without a slot compression taking place.

- **No slot compression**:

  In all PNIO devices, no slots of this PNIO controller are compressed.

  ⇒ *Use case:* This option can be used for “Shared Device” applications (alternative to the setting above) or for PNIO devices with specified or configurable slot numbers. Thus, there is no continuous slot numbering

without gaps. (Note: this setting applies to all PNIO devices of the PNIO controller).

Note: For diagnostic purposes, the "active" slot number in a configuration in the generic parameter dialog of a PNIO I/O module can be viewed, in logged-in state, refer to the dialog *"PNIO module parameters"* → *"Module parameters"* → "*Slot number*". (The dialog has to be enabled via IndraWorks → Tools → Options → IndraLogic 2G → Device editor → Show generic configuration editors).

Tab “IP channel settings”

Refer to ""“IP channel settings” tab" on page 32

Tab “Profinet I/O controller parameters”

This tab is used for service purposes and is only visible if the option "Show generic configuration editors" was enabled. Opening via the IndraWorks menu **Tools ▸ Options ▸ IndraLogic 2G ▸ Device editor**.

Please clarify any possible modifications of the parameters with the service team.

Tab “Field bus diagnostics”

Based on the object node of the master (here: Profinet I/O controller), a uniform diagnostic concept was designed for the field buses of the MLC and MTX systems.

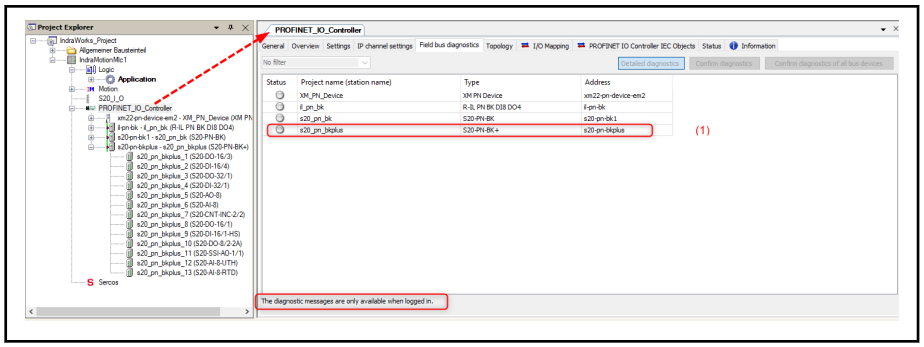


Fig. 6-12: Field bus diagnostics, control online, not logged in

Bus coupler (1) is not present at the bus system. It would generate an error. Thus, this bus coupler is disabled and excluded from the diagnostics via **Properties ▸ Compile ▸ Exclude from compilation**

After login, the bus available at the node runs through a diagnostic cycle. The status LEDs change their color as a result of this cycle (here: green), i.e., the module signals an error-free run.

After selecting a module (click on the line), a detailed diagnostics is available.

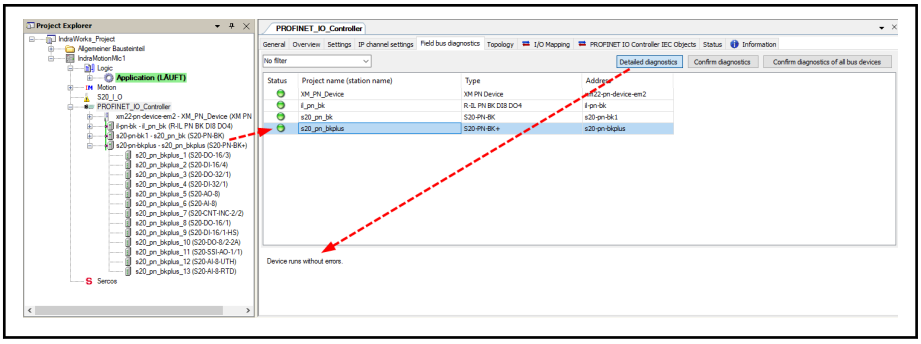


Fig. 6-13: Field bus diagnostics, control online, logged in, error-free

The login is repeated with the additional coupler not available at the real control. There is no compilation.

A yellow warning triangle is displayed at the respective module (and at the controller) indicating that a diagnostic message is present. The status LED of the affected module turns red. Click on <Detailed diagnostics> to provide a "Standard diagnostics" with brief information and a "Detailed diagnostics" with detailed error cause.

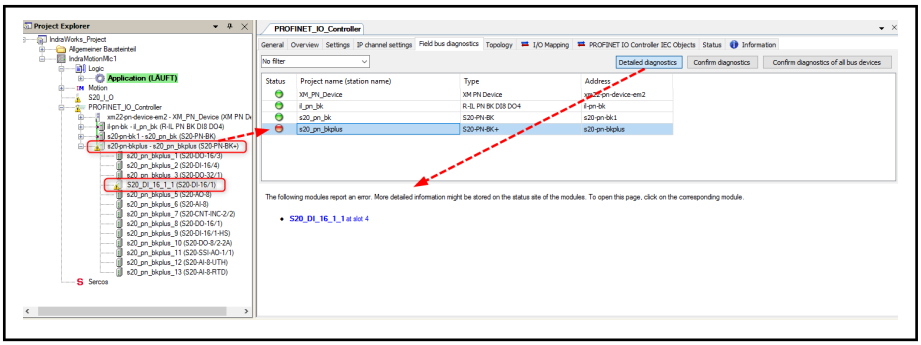


Fig. 6-14: Field bus diagnostics, control online, logged in, error

For temporary errors, it is only displayed as long as the error is present.

Confirm Diagnostics:

Although there is an error, the warning triangle and the red status LED are turned off for the selected module.

The "Extended diagnostics" of the "Detailed diagnostics" remains in the text (the next diagnostics can be edited...).

Confirm Diagnostics of All Bus Devices:

Although there are errors, the warning triangle and the red status LED are turned off for all bus devices.

Tab “Topology”

This tab provides an overview on the topology of the Profinet I/O network and on the neighboring/peer information for each Profinet I/O device.

☞ Only available for devices with LLDP support (example: Conformance Class B) and all later devices acc. to the Profinet V2.3 standard.

To retrieve topology information, set a valid IP address before in the PNIO devices. Also refer to the command Auto IP.

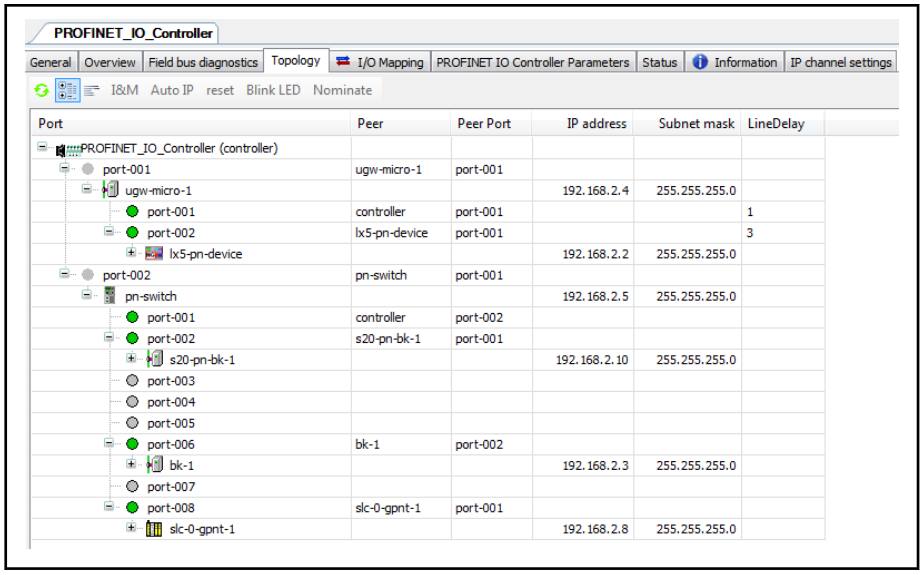


Fig. 6-15: Profinet I/O controller - Tab “Topology”

Port	Number of the local port, beginning with 1. Possible statuses: : Unknown or not connected : OK : Error
Neighbor	Device name and port of the neighboring device.
Neighbor-port	
IP address	Set IP address of the slave
Subnet mask	Set subnet mask of the slave
Line Delay	Signal propagation time in nanoseconds (recommended value: 6 ns for 1 meter of cable)

Commands:

Update	Updates displayed data (bus scan)
Tree/list view	Toggles between the tree and list structure representation
I&M	Retrieves the I&M data (identification and maintenance) of the device and display in a dialog
Set IP	Automatic setting of a valid IP address within the network (network settings of the master) if the address scanned by the device is invalid
Reset	Resets the device settings of the selected device to default values
Blink LED	To identify a listed device by its hardware, select the device and click on "Blink LED". The device should respond with a flashing signal.
Set name and IP	It is tried to set the station name and the IP address specified for the selected slave.

Tab “I/O mapping”

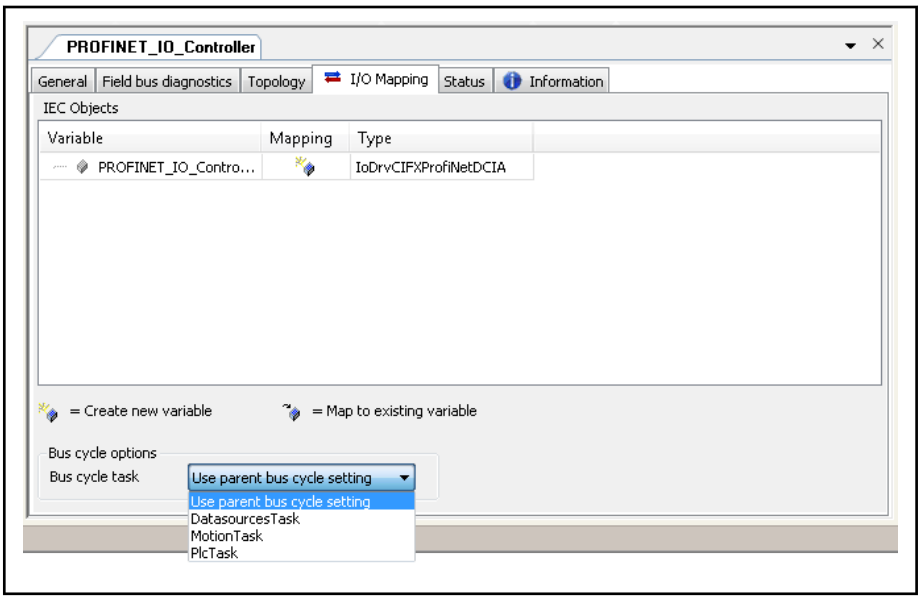


Fig. 6-16: Profinet I/O controller - Tab "I/O mapping"

IEC objects: When the Profinet I/O controller is defined, "IoDrvCIFXProfiNetDCIA.library" or RIL_ProfinetIO.library are automatically used. Thus, the memory space to implement the Profinet I/O controller can be defined.

Bus cycle options

Bus cycle task: Select a bus cycle task, to connect the cycle of the map exchange for the Profinet I/O controller to a particular task. In this task, it is useful to process I/O data of the master as well.

Default setting: Use parent bus cycle setting

With this setting, the task setting made in "Bus cycle options" of the control (double-click on the actual control in the Project Explorer, PLC settings) is applied to the actual bus.

Tab “Status”

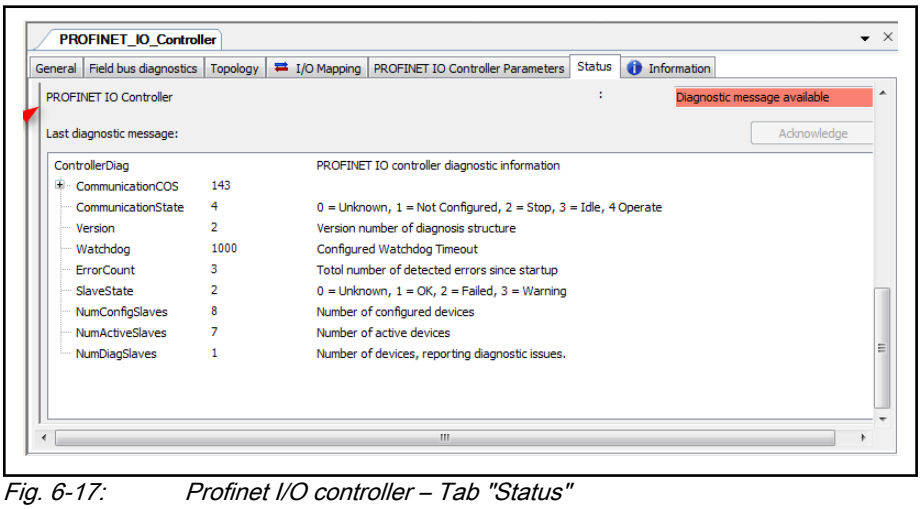


Fig. 6-17: Profinet I/O controller – Tab "Status"

The "Status" tab in the "Profinet I/O device editor" displays status information (e.g., "Running", "Stopped") and specific diagnostic messages from the Profinet device.

In addition, the diagnostic messages in the status flags are displayed in "Diag".

The output is a hexadecimal value determined by the status flags set (refer to the document "Protocol Interface Manual" for Profinet I/O).

Tab “Information”

The tab displays general information from the device description file:

Name, Vendor, Categories, Version, Order number, Description and Image if available.

6.5.2 Profinet I/O controller, adding devices

General

The available Profinet I/O devices are located in the "Periphery" library in the "Profinet I/O" folder.

Drag the required devices from the library to the Profinet object.

In the Project Explorer, devices can thus also be inserted between existing devices.

Optionally, add devices in the context menu via **Add ▸ ...** of the Profinet object.

In this case, the new device is added as last module below the Profinet object.

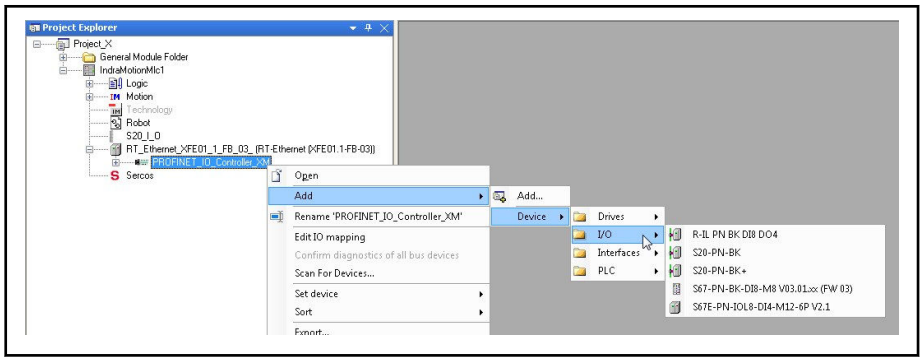


Fig. 6-18: Profinet I/O controller, adding devices

If a required Profinet I/O device is not in the library by default, it can be added by importing its GSDML file via the main menu **Tools ▸ Device Database ▸ Add devices**. Also refer to Device database on page 15.

Enabling/disabling Profinet I/O devices

The Profinet I/O devices configured at a control can be enabled and disabled in the Project Explorer.

Thus, select/deselect the button at the icon of the Profinet I/O device.

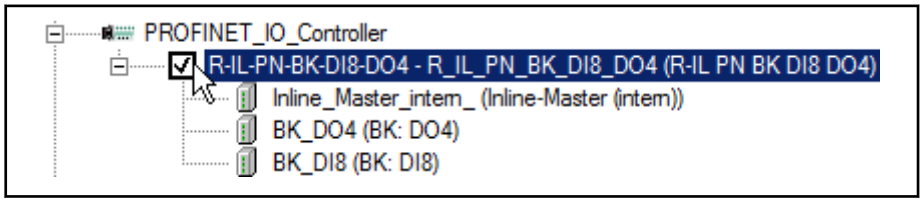


Fig. 6-19: Enabling/disabling Profinet I/O devices

The bus device is enabled if the checkbox is selected. It is considered in the diagnostics of the bus.

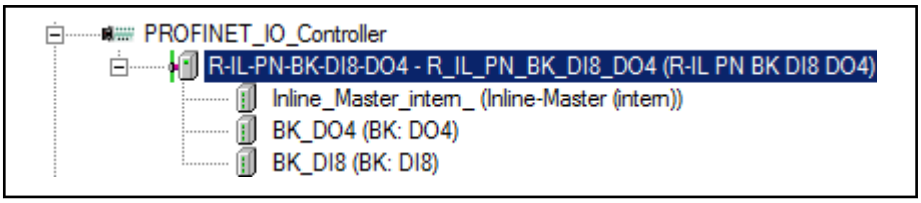


Fig. 6-20: Profinet I/O device as enabled device

The bus device is disabled if the checkbox is deselected. It is not considered in the diagnostics of the bus.

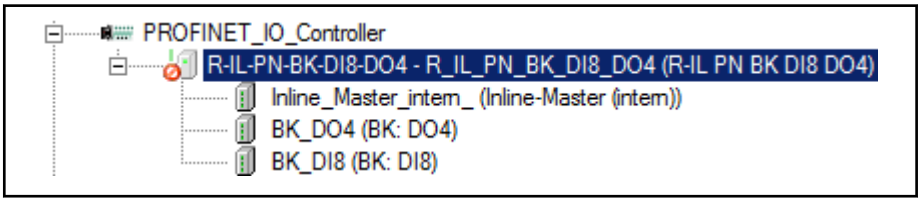


Fig. 6-21: Profinet I/O device as disabled device

If a new Profinet I/O device is created, it is enabled by default.

When loading the configuration, all enabled Profinet I/O devices are taken into account. Disabled Profinet I/O devices are ignored.

An activated, but not present bus device generates an error message in the diagnostics.

A disabled and not present bus device does not generate any error message in the diagnostics.

Also refer to Disabling Profinet I/O devices and Profinet I/O modules using UserDefs.cfg, page 94

Disabling Profinet I/O devices and Profinet I/O modules using UserDefs.cfg

Use this function to disable Profinet IO devices / Profinet IO modules configured in IndraWorks. Disable them by the definition in a stored file located on the Compact Flash card of the control.

This impedes diagnostic messages for Profinet I/O devices / Profinet I/O modules that are configured, but do not exist.

Enabling the function

The function is enabled if the valid file "UserDefs.cfg" is present in the following partitions on the control Flash Card:

For XM and VPx controls: **\USER**

Another path or file name can be defined by making entries in the "cfg file" of a system.

Example:

```
[IOSpecialDefs]
IODeactFilePath=MyFilePath
IODeactFileName=MyFileName
```

The file "UserDefs.cfg" can contain entries for more buses:

Inline modules, Sercos, Profinet I/O, Profibus DP.

Deactivation point in time

The "UserDefs.cfg" file is analyzed and the entered Profinet I/O devices / Profinet I/O modules are disabled at the moment and in the context of the internal event "PrepareUpdateConfiguration".

Event takes place in case of the following actions:

- Startup after power on
- Reset (warm)
- Reset (cold)
- Load program after reset (origin)

Structure of the file "UserDefs.cfg"

The file is structured in a single or multiple existing section(s) and the contained entries.

- **Sections**: The name of a section is written in **[** and **]**. A section ends at the beginning of the next section or at the end of the file.

  Each section identifier also contains an instance number `[x,0..4]` of the respective object. It labels the possible Profinet field bus instances of the same type in ascending order.

- **Special function**: Instance number **x** is used to ignore the section during the evaluation.

  Several sections can be defined for **one** instance number. All their entries are assigned to the specified instance number.

  Together with the "x-function" described above, different device groups can be handled easily.

- **Comments**: A semicolon (**;**) is defined as comment character. The text is not analyzed from the position of the semicolon up to the end of the line.

Profinet

Profinet uses device names to address the devices (slaves) also called "station name" in the configuration.

☞ *The station name may consist of the following characters:*

- Lower case letters without umlauts "a" to "z".

  Upper case letters "A" to "Z" are allowed, but internally converted into lower case letters.

  The station names are not case sensitive.

- Digits from "0" to "9" (may not be at the beginning of the name).
- Special hyphen character "-" and dot "." (may neither be at the beginning nor at the end of the name).
- The name may not exceed 240 characters. For some devices/vendors, a limit of 127 characters apply.

  The name can be divided into substrings using "." dots. Such a substring may not exceed 63 characters.

  The name may not start with the characters string "port-xyz" (with x,y,z = 0...9).

  The name may not have the sequence "a.b.c.d" (with a,b,c,d = 0...999).

Profinet I/O controller

- The name `PROFINETIO.Controller.` and one instance number describe a control as I/O controller.

  Example:

  ```
  [PROFINETIO.Controller.2]
  ```

- **Complete Profinet device**: An entry to disable a complete Profinet device below a controller consists of the keyword `DeviceDeactName=` and the specification of the device name as text.

  Example:

  ```
  DeviceDeactName = myDeviceName1
  ```

- **Module entry**: To disable optional modules below a device, specify

  the keyword `DeviceName=` and the device name as well as

  the keyword `DeactModule=` and - separated by comma - the module numbers to the disabled from **1** to **n**.

  Module number **0** is the bus connection of the device and may not be specified here.

  Example:

  ```
  DeviceName = myDeviceName3, DeactModule=3,4,7
  ```

- **Last module entries**: An entry to disable the **n** latest configured modules below a device consists of

  the keyword `DeviceName=` and the device name as well as

  the keyword `DeactLastModules=` and the number of modules to be disabled.

  This disables the last (lowest) modules in the configuration tree.

  Example:

  ```
  DeviceName = myDeviceName2, DeactLastModules=2
  ```

Profinet I/O device

- The name `PROFINETIO.Device.` and one instance number describe a control operated as I/O device.

  Example:

  ```
  [PROFINETIO.Device.0]
  ```

- **Overwrite device name**: With `StationNameSet=` and a device name, the configured station name of the device is overwritten.

  Example:

  ```
  StationNameSet= overloadDeviceName
  ```

- **Module entry**: To disable optional modules below a device, specify

  the keyword `DeactModule=` and - separated by comma - the module numbers to the disabled from **1** to **n**.

  Module number **0** is the bus connection of the device and may not be specified here.

  Example:

  ```
  DeactModule=3,4,7
  ```

- **Last module entries**: An entry to disable the **n** latest configured modules below a device consists of

  the keyword `DeactLastModules=` and the number of modules to be disabled.

  This disables the last (lowest) modules in the configuration tree.

  Example:

  ```
  DeactLastModules=2
  ```

Size of the "UserDefs.cfg" file

The memory evaluation is allocated according to the specified file size. The permitted size is thus virtually not limited.

Example of the file "UserDefs.cfg"

Profinet I/O Controller.0

```
;************************************************
;---------- PROFINET IO Controller.0 ----------
;************************************************
[PROFINETIO.Controller.0]

DeviceDeactName = myDeviceName1
DeviceName = myDeviceName2, DeactLastModules=2
DeviceDeactName = myDeviceName3
```

Profinet I/O Controller.2

```
;************************************************
;---------- PROFINET IO Controller.2 ----------
;************************************************
[PROFINETIO.Controller.2]

DeviceDeactName = myDeviceName12
DeviceName = myDeviceName14, DeactModule=2,3
```

Profinet I/O Device.0

```
;************************************************
;---------- PROFINET IO Device.0 ----------
;************************************************
[PROFINETIO.Device.0]
StationNameSet= newDeviceName
```

Profinet I/O Device.1

```
;************************************************
;---------- PROFINET IO Device.1 ----------
;************************************************
[PROFINETIO.Device.1]
DeactLastModules= 3
```

Devices to connect I/O modules

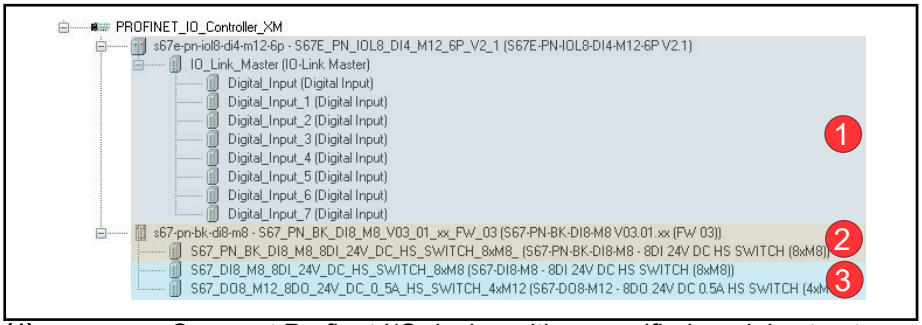
Profinet I/O distinguishes two types of I/O devices:

1. **Compact**: For a compact device, the module structure is fixedly specified.

   After adding a compact Profinet I/O device to the Project Explorer, the modules below the bus coupler already exist. The modules are not visible in the device library.

2. **Modular**: The module structure is variable.

   Based on a bus coupler (with its own existing I/Os), modules can be arranged individually - but according to the device placement specifications.



(1) Compact Profinet I/O device with a specified module structure
(2) Modular Profinet I/O device with inputs and outputs belonging to the bus terminal
(3) Terminal assigned individually

Fig. 6-22: I/O modules at the Profinet I/O device

Profinet I/O device - Device editor

Open the device editor to configure and parameterize the Profinet I/O device object. To open the device editor, double-click on the Profinet device in the Project Explorer. Alternatively, open the device editor via the context menu of the Profinet I/O device.

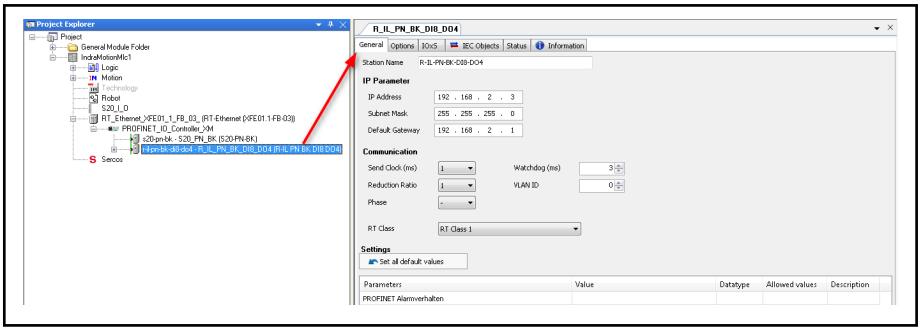


Fig. 6-23: Profinet I/O device - Tab “General”

Configuration dialogs for Profinet I/O devices

- Tab “General” , page 98
- Tab “IOxS” , page 103,
- Tab “PNIO parameters” , page 104

  This window is used for service purposes and is only visible if the option "Show generic configuration editors" is enabled under **Tools ▸ Options ▸ IndraLogic 2G ▸ Device editor**.
- Tab “Status”, page 105,
- Tab “Information”, page 105,

Tab “General”

For all settings in this tab applies that it depends on the device description whether the settings can be edited and which values are specified and permitted.

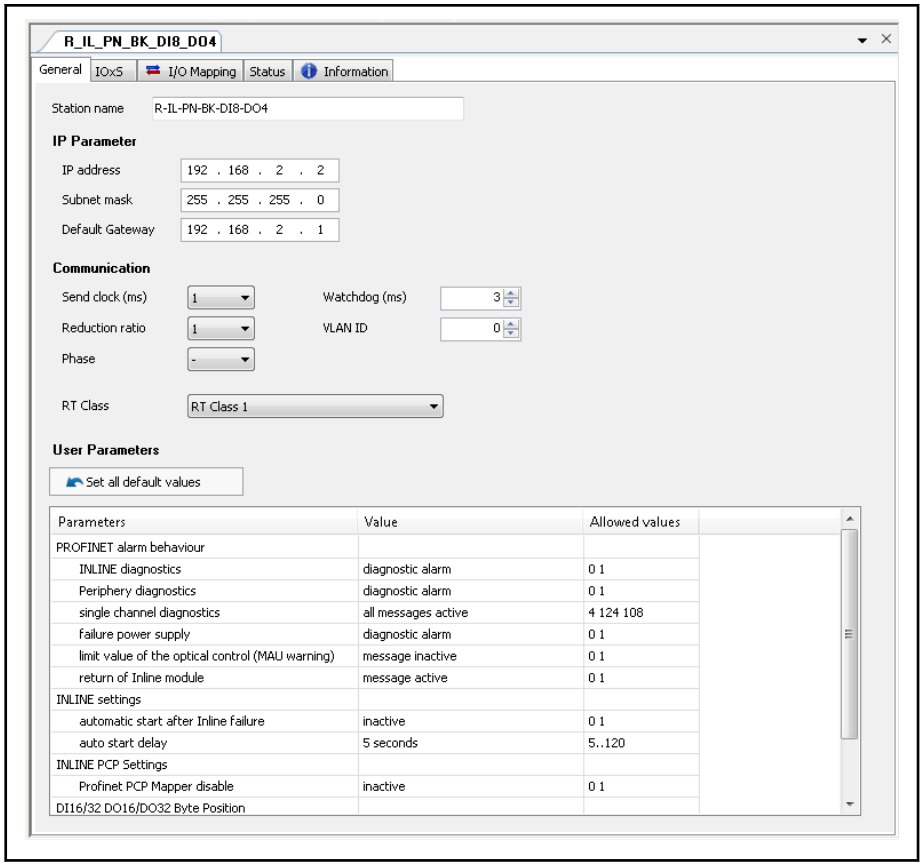


Fig. 6-24: Profinet I/O device - Tab "General"

Station name

The station name is used within Profinet to address I/O devices. Thus, it has to be unique in the Profinet subnet.

The field "Station name" has to be adapted in most of the cases as well.

When creating the I/O device, the default name from the device description file is specified.



The station name may consist of the following characters:

- Lower case letters without umlauts "a" to "z".

  Upper case letters "A" to "Z" are allowed, but internally converted into lower case letters.

  The station names are not case sensitive.
- Digits from "0" to "9" (may not be at the beginning of the name).
- Special hyphen character "-" and dot "." (may neither be at the beginning nor at the end of the name).
- The name may not exceed 240 characters. For some devices/vendors, a limit of 127 characters apply.

  The name can be divided into substrings using "." dots. Such a substring may not exceed 63 characters.

  The name may not start with the characters string "port-xyz" (with x,y,z = 0...9).

  The name may not have the sequence "a.b.c.d" (with a,b,c,d = 0...999).



The station name is used to address the Profinet I/O device. Missing or deviating IP address settings at the device are ignored.

IP parameters

The fields "IP address", "Subnet mask" and "Default gateway" contain default entries derived from the settings of the I/O controller.

These values can be adapted with network knowledge. However, it is recommended not to change these values.

Communication Parameters

This area describes the time response of the communication on the Profinet. They are set by the Profinet I/O controller while the connection is established.

From the two parameters

`Send clock` and

`Reduction Ratio`

the actual cycle time results

`(t = Reduction Ratio * Send clock ),`

acc. to sending device data:

- **Send clock (ms)**: Transmission period in milliseconds.
- **Reduction ratio**: Factor to calculate the cycle time from the transmission period.
- **Watchdog**: Monitoring time (ms), possible values: 0 to 65535. If the I/O device does not obtain any more signals from the I/O controller within this period, a device-specific response is output, e.g. an error message.
- **Phase**: The phase is internally set in the controller.
- **VLAN ID:** VLAN ID: Enter a number between 0–4095 for VLAN of type 802.1Q or between 0–32767 for VLAN of type ISL (availability depends on the device).
- **RT Class:** The I/O controller currently supports only real-time (RT) class 1. This field is intended for future extensions.

User parameters

"User parameters" button

- **Set all default values**: Press this button to replace all changed settings with the default settings from the GSDML file.

Parameter table

In the parameter table, double-click on the respective value to edit it. Depending on the parameter, this action is performed via a drop-down list or by directly entering a value.

- **Parameter:** Parameter name or parameter category (without value assignment)
- **Value:** Current parameter value
- **Value range:** Specifies what may be entered into the Value column.

  Syntax:

  ```
  <Parameter_Type>(<BitsUsed>)
                  <BasicValue> <ValueRange>.
  ```

  Example:

  "BitArea (4-5) 0 0-2" means:

  This is a bit combination stored in the bits 4 and 5 of the configuration byte.

  The basic value is 0. The value can be between 0 and 2.

Tab “Options”

This tab provides additional configuration options (as of IndraWorks 14V22).

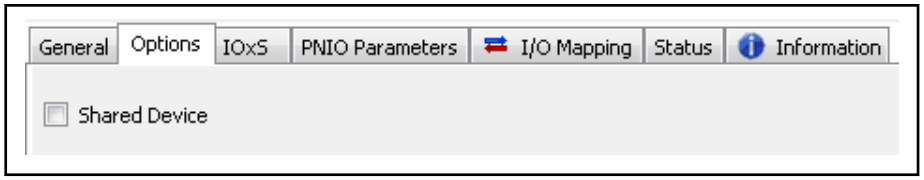


Fig. 6-25: Profinet I/O device - Tab “Options”

Shared Device

In case of a “Shared Device”, multiple Profinet I/O controllers share the (sub)modules of a Profinet I/O device. Only one connection can be established to each (sub)module at a time (exception "Shared Input"). No connection is prioritized. The following principle applies “First come, first served”.

The "Shared device" option is only provided if the I/O device supports this feature.

If the option is selected, no connection is established to the headend of the I/O devices. Only a connection to the modules is established instead. Only one Profinet I/O controller may be connected to the headend, i.e. for all other I/O controllers that also try to access to this device, this option has to be set.

To use the "Shared device" feature, the same I/O device has to be configured for multiple I/O controllers. Ensure the following:

- The controllers have to have different names as well as different IP addresses in the same subnet.
- The device has to have the same station name, IP address, subnet mask and default gateway in both the configurations.
- In one controller, "Shared device" may only not be set in the “Device” options.
- The modules of the devices have to be configured in all controllers, but have to be changed alternatingly to ensure that the modules are only active at one controller (exception "Shared Input").

☞ When using a "shared device", the slots of the disabled modules **may not be compressed**.

By default, the slot compression is active for disabled I/O modules, i.e. it has to be disabled for a “Shared Device” application, refer to chapter "Tab “Settings”" on page 88.

Example: Shared Device configuration

The following applies to the subsequent figure: XM4 (top left) and XM4 (top right) are Profinet I/O controllers during data exchange with S20-PN-BK+ (down). The I/O communication relation between the Profinet I/O controller and the Profinet I/O device is shown in colors (blue, orange).

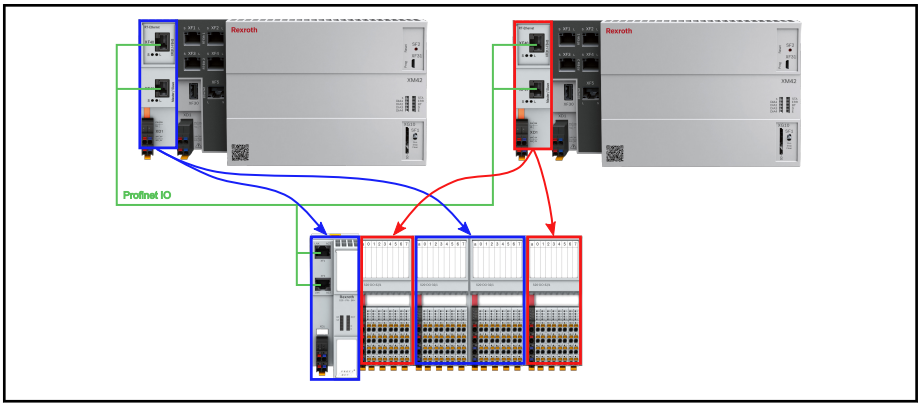


Fig. 6-26: "Shared Device" example topology

The following example configuration is used:

- XM42 (blue marking)
  - Station name: **extension.xm42.1**
  - IP: **192.168.3.1**
  - Subnet: **255.255.255.0**
- XM42 (red marking)
  - Station name: **extension.xm42.2**
  - IP: **192.168.3.2**
  - Subnet: **255.255.255.0**
- S20-PN-BK+
  - Station name: **s20-pn-bk**
  - IP: **192.168.3.3**
  - Subnet: **255.255.255.0**
  - The complete module configuration is available on the two I/O controllers. However, the modules are disabled alternatingly.

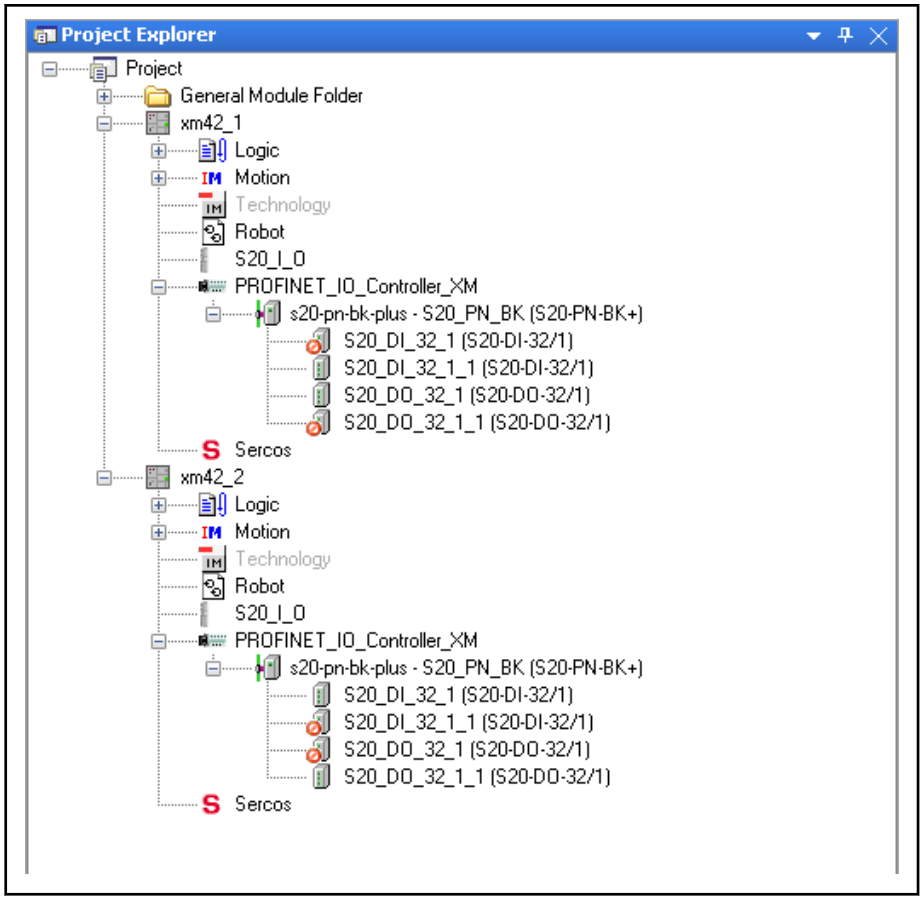


Fig. 6-27: "Shared Device" example configuration

Tab “IOxS”

With a corresponding entry in the device description, I/O provider and I/O consumer data can be accessed using this tab.



Submodules without own I/Os also provide a "provider state" for inputs.

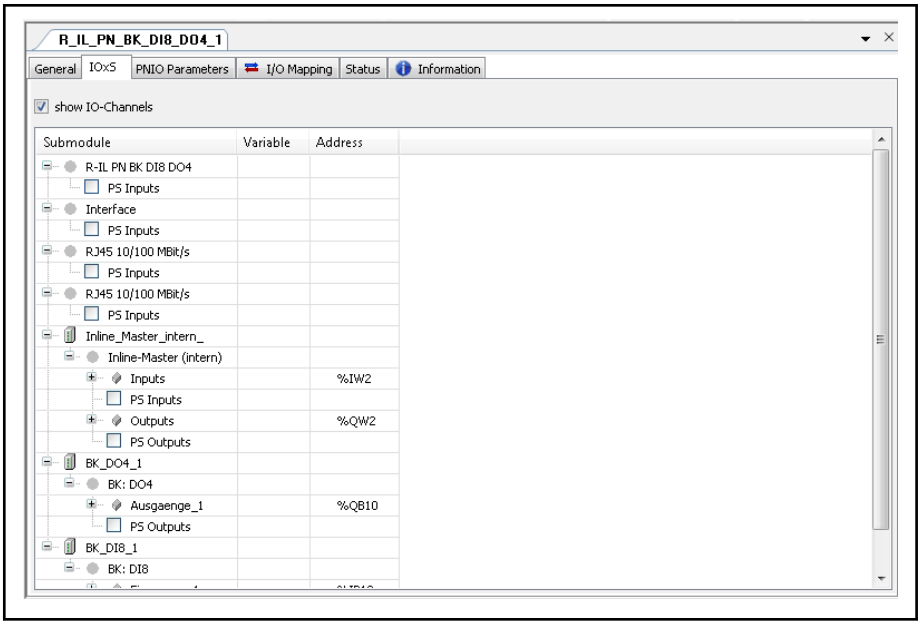


Fig. 6-28: Profinet I/O device - Tab "IOxS"

If the checkbox is selected, the corresponding module provides additional I/Os with the provider state in the "I/O mapping" dialog.

Double-click on the address field to open the corresponding module.

The status of the value hex 80 (bit 7) of the provider “state byte” indicates "OK". For detailed information on this byte definition, refer to the technical Profinet I/O specification.

☞ If an I/O provider state is mapped for output data, the PS output has to be set explicitly to "GOOD" (=0x80), as device data might not be applied and substitute values are used instead.

☞ If the application is stopped, the unmapped I/O provider states for outputs are set to "bad". Mapped I/O provider states are set to their configured default values.

Tab “PNIO parameters”

This tab is used for service purposes and is only visible if the option "Show generic configuration editors" was enabled. Opening via the IndraWorks menu **Tools ▸ Options ▸ IndraLogic 2G ▸ Device editor**.

☞ Please clarify any possible modifications of the parameters with the service team.

☞ When the bus is running, modified parameters can be applied using the <Write parameters> button (invisible if offline).

The "MAC address" is only used for information purposes and is only set if

- the I/O device was applied from the "Scan devices" menu
- the address was applied from the "Scan devices" menu

Tab “Status”

The "Status" tab in the "Profinet I/O device editor" displays status information (e.g. "Running", "Stopped") and specific diagnostic messages from the device.

In addition, the diagnostic messages in the status flags are displayed in "Diag". The output is a hexadecimal value determined by the set status flags (refer to the document "Protocol Interface Manual" for Profinet I/O).

Tab “Information”

The tab displays general information from the device description file:

Name, Vendor, Categories, Version, Order number, Description and Image if available.

6.5.3 Profinet I/O controller, adding modules to device

General information

This section contains basic information on the configuration and parameterization of I/O modules (I/O terminals) at the modular Profinet I/O device.

Add Profinet I/O modules to be created either via the device library or the context menu of the Profinet object in the Project Explorer.

The device library contains modules for the respective Profinet I/O device. Go to the folder "Profinet I/O" below the Profinet I/O device in the "Periphery" library. Drag the module from the library to the Profinet I/O device in the Project Explorer. This allows to insert I/O modules between already existing modules in the Project Explorer.

Alternatively, open the context menu of the Profinet I/O object to add the newly created Profinet module. Open the context menu of the Profinet I/O object in the Project Explorer and select the context menu item: **Add ▸ Module ▸ ...**. The new module is inserted as last module below the Profinet I/O device.

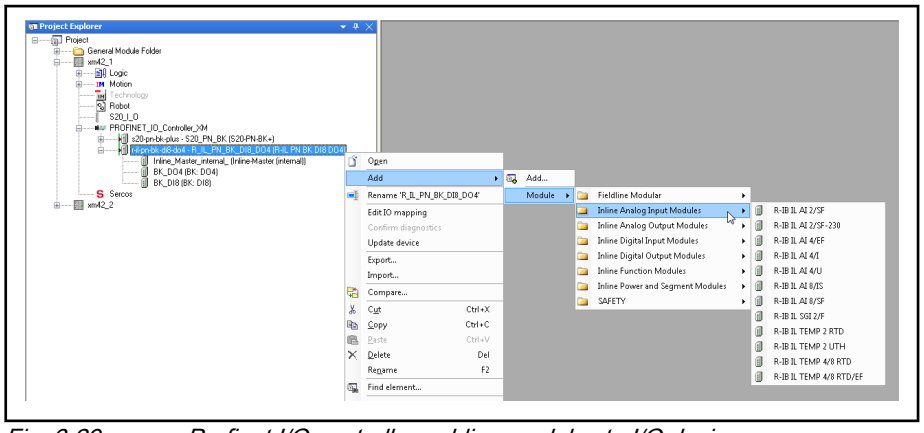


Fig. 6-29: Profinet I/O controller, adding modules to I/O device

To configure the new modules, double-click on the module to be configured in the Project Explorer to open the device editor.

Dialogs for modules at the Profinet I/O device

- Enabling/disabling Profinet I/O modules, page 106
- Disabling Profinet I/O devices and Profinet I/O modules using UserDefs.cfg, page 94
- Tab “General” , page 107

- Tab “PNIO parameters” , page 108, this window is used for service purposes and is only visible if the option "Show generic configuration editors" is enabled under **Tools** ▸ **Options** ▸ **IndraLogic 2G** ▸ **Device editor**.
- Tab “I/O mapping”, page 108
- Tab “Status”, page 109
- Tab “Information”, page 109

Enabling/disabling Profinet I/O modules

The Profinet I/O modules configured at a Profinet I/O device can be enabled and disabled in the Project Explorer.

Thus, select/deselect the button at the icon of the Profinet I/O module.

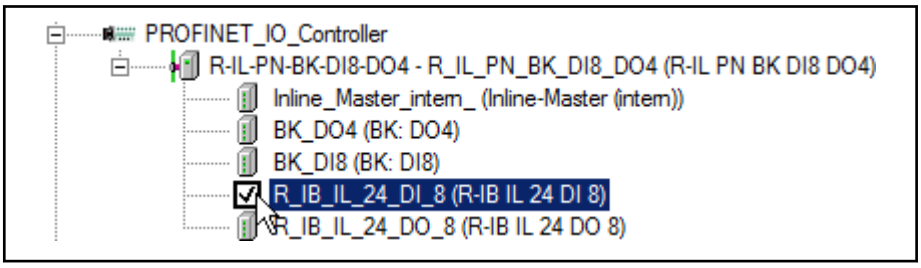


Fig. 6-30: Enabling/disabling Profinet I/O modules

The bus device is enabled if the checkbox is selected. It is considered in the diagnostics of the bus.

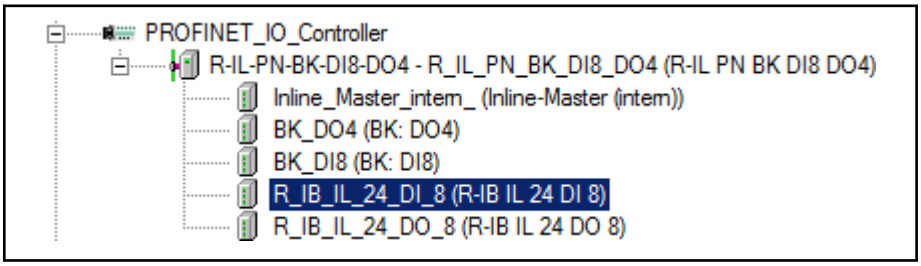


Fig. 6-31: Profinet I/O module as enabled device

The bus device is disabled if the checkbox is deselected. It is not considered in the diagnostics of the bus.

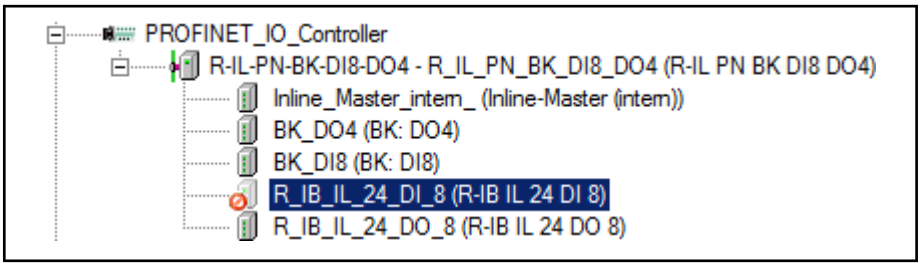


Fig. 6-32: Profinet I/O module as enabled device

If a new Profinet I/O module is created, it is enabled by default.

When loading the configuration, all enabled Profinet I/O modules are taken into account. Disabled Profinet I/O modules are ignored.



An activated, but not present bus device generates an error message in the diagnostics.

A disabled and not present bus device does not generate any error message in the diagnostics.

Also refer to Disabling Profinet I/O devices and Profinet I/O modules using UserDefs.cfg, page 94

Tab “General”

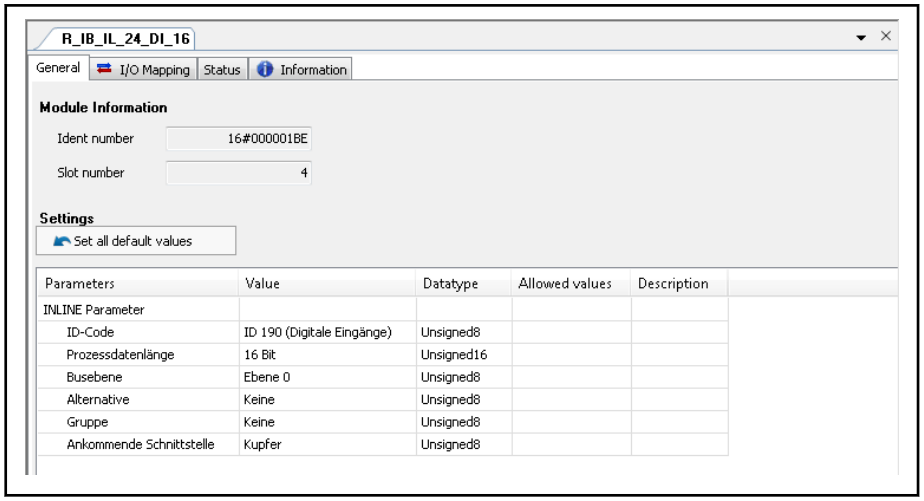


Fig. 6-33: Profinet I/O module - Tab “General”

Module information

- **ID number**: Module ID from device description.
- **Slot number:** Module position below the device, starts with "1" for the first module and increases for each other module. This results from the current structure in the device tree.

User parameters

Depending on the device, there are either no or multiple setting options for the user-specific parameters:

- **Length of user parameters**: The total length of all user parameters is displayed in bytes.

In the parameter table, double-click on the values to edit them. Depending on the parameter, this action is performed via a drop-down list or by directly entering a value.

- **Parameter**: Parameter name or parameter category (without value assignment).
- **Value**: Current parameter value
- **Value range:**
  - **Basic value**: Default parameter value.
  - **Permitted values**: Specifies what may be entered into the "Value" column.

    Syntax:

    `<ParameterType> (<BitsUsed>) <BasicValue> <PermittedValueRange>.`

    Example:

    "BitArea (4-5) 0 0-2" means: This is a bit combination stored in the bits 4 and 5 of the configuration byte. The basic value is 0. The value can be between 0 and 2.
- *Buttons "User parameters" (only active online)*
  - **Set all default values**: Press this button to replace all changed settings with the default settings from the GSDML file.

Tab “PNIO parameters”

This tab is used for service purposes and is only visible if the option "Show generic configuration editors" was enabled. Opening via the IndraWorks menu **Tools** ▸ **Options** ▸ **IndraLogic 2G** ▸ **Device editor**.

Please clarify any possible modifications of the parameters with the service team.

"Write parameters" is only available in online mode.

The dialog provides information on position and size of the parameters.

Submodules: Some later Profinet modules are provided with submodules defined in the GSDML file. The import of the GSDML file creates these submodules.

Tab “I/O mapping”

The tab is used to assign module inputs and outputs to variables that can be used as local or global variables in the individual POUs.

The current value of the variables is displayed in online mode.

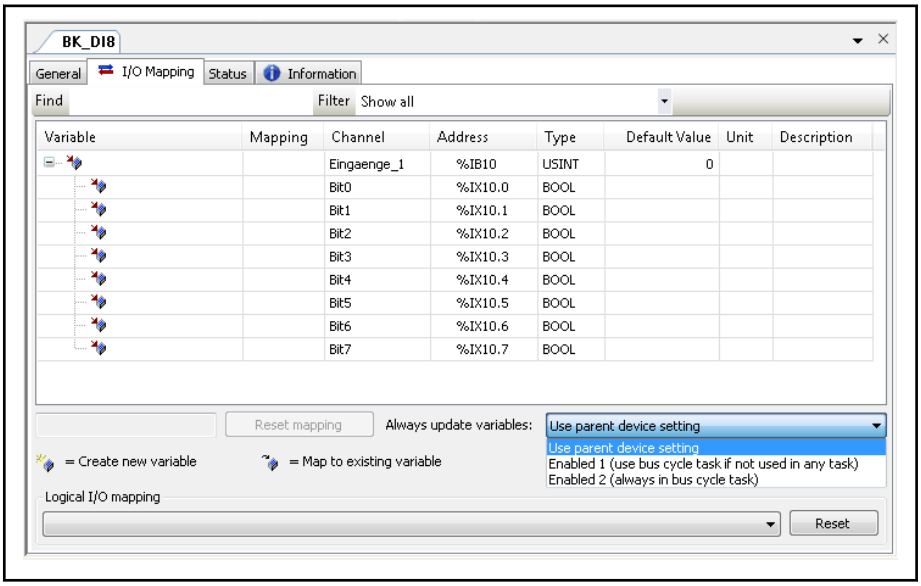


Fig. 6-34: Profinet I/O module - Tab “I/O mapping”, here digital output bus terminal

Channels: For this PNIO device, the "Channel" area contains the IOPS inputs.

This assignment is described in Mapping of the field bus inputs and outputs, page 19,.

Reset Mapping: Use this button to reset the mapping settings to the default values defined in the device description file.

Update variables: Defines for the module whether I/O variables are updated in the bus cycle task.

The default value is defined in the device description file.

- **Use parent device setting:** Update according to the settings of the superordinate device.
- **Enabled 1 (use bus cycle task if not used in any task):** The I/O variables are updated in the bus cycle task if not used in any other task
- **Enabled 2 (always in the bus cycle task):** All variables are updated in each cycle of the bus cycle task irrespective of whether they are used or mapped on an input or output channel.

Tab “Status”

The "Status" tab in the "Profinet I/O device editor" displays status information (e.g. "Running", "Stopped") and specific diagnostic messages from the device.

In addition, the diagnostic messages in the status flags are displayed in "Diag". The output is a hexadecimal value determined by the set status flags (refer to the document "Protocol Interface Manual" for Profinet I/O).

Tab “Information”

The tab displays general information from the device description file:

Name, Vendor, Categories, Version, Order number, Description and Image if available.

6.5.4 Profinet I/O, controller, scanning devices

Scanning devices

"Scan devices" automatically determines hardware components connected to the control.

Prerequisites

Before executing the command, the Profinet I/O driver for the respective interface has to be loaded to the control.

Log in the control initially with a PLC project whose corresponding field bus interface is configured as Profinet I/O controller.

The device description files of the connected devices have to be in the IndraWorks device database.

If the device description file is missing for a specific device in the device database, the search works only partially! In this case, the device description file has to be added to the device database and scanning has to be repeated (Reload device description file, page 15).

For the search, proceed as follows:

- Switch the control online
- The application has to be in the operating state "Running"
- Select the Profinet I/O controller object in the Project Explorer
- Open the context menu of the object and select "Scan devices...".

Context menu command "Scan devices"

The "Scan device" command is used to detect the currently connected hardware, i.e. to determine its structure, to display it in a dialog and to provide it for the application to the Project Explorer.

The scan functionality is operated differently for different network types:

The call can be implemented fixedly in the control (in a dialog of the wizard to create the control) or it can be provided as a field bus menu item (onboard interface or at the extension module).

When executing the "Scan devices" command, a connection is automatically established to the control and then canceled again.

To start the search, select the Profinet object in the Project Explorer, open its context menu and select the command **Scan devices...**.

The devices found in the current search and the related modules are displayed in the "Scan devices – Scanned devices" dialog.

Functions for Profinet I/O devices:

Devices as well as modules can be scanned in a Profinet I/O system.

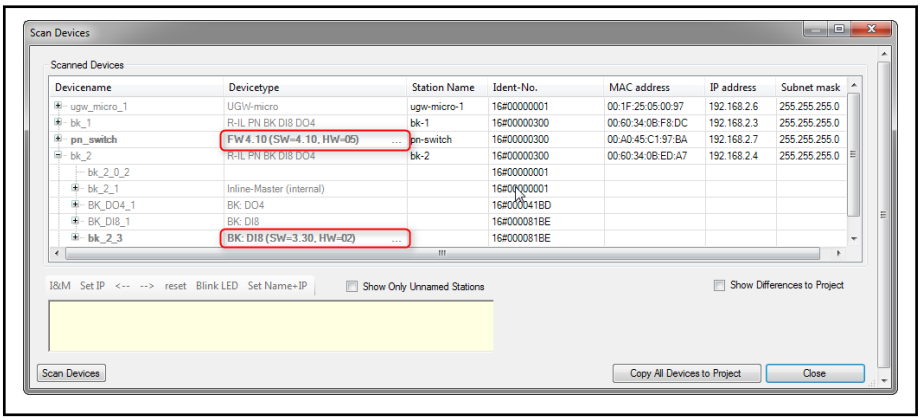


Fig. 6-35: Dialog "Scan devices" for Profinet I/O, bus terminal with DI8 module found

There are different types (drop-down list) for the bus terminal selected in the figure above.

The user has to device which type to use.

Configuration procedure

- In the Project Explorer, execute the command "Scan devices" in the context menu of the Profinet I/O controller.

  The dialog shows all devices found.
- Enter the name for the respective device into the "**Station name**" field (addressing at Profinet I/O) and press the button "**Set Name+IP**" to apply it to the device (IP address can be 0.0.0.0).

  Further devices have to be edited individually, as the function "Set Name+IP" affects only the selected device.
- Press the button "**Auto IP**" to assign a free IP address from the IP range of the PNIO controller configuration to the address.
- If the device is provided with a station name and a valid IP address, the configuration can be applied to IndraWorks ("Copy to project" button). Multiple devices can also be selected at the same time.

More functions

- For each device, the **station name**, **MAC address**, **IP address** and **subnet mask** are displayed in other columns in the "Found devices" window.

Each device must have a **station name**, as it is required for certain functionalities in network operation!

- If the filter **Show only unnamed stations** is selected, the only devices listed are those that do not yet have a station name.

- The **station name** can be changed in this dialog.

  Double-click and the field "Station Name" changes from display to editing mode.

  Use the <Set Name+IP> button when the device entry is selected to inform the bus system on the new name.



If values were changed in the list of available devices (e.g. IP address change), the corresponding value is displayed in italics.

This new value is only changed in the editor, but not in the device.

After loading the value to the device, it is displayed normally.

Values displaying differences between the project and the devices found are displayed in orange.



After changing the previously scanned data and after executing the command <Set Name+IP>, press "Scan devices" again.

The current device data is then shown in the display.

- The entries for **Device name**, **IP address** and **Subnet mask** can also be edited in the dialog before the device settings are applied to the project.

  If the device is accepted with any address defined, the default setting defined in the "PNIO Controller Parameters" dialog of the device editor defined for such cases are used.

  Refer to chapter "Tab “General” " on page 84

  The final address for a device can be defined in its configuration dialog "PNIO Parameters".

  Refer to chapter "Tab “General” " on page 98.

- The **module ID number** is only displayed in the list if
  1. it is provided by the device,
  2. if device and controller are part of the same logic network and
  3. if IP address and network mask of the device are set correctly.

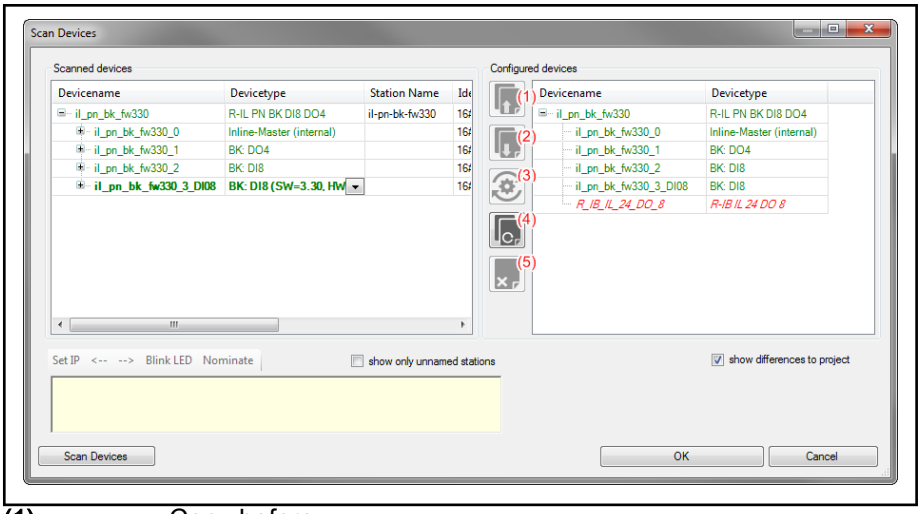
  If the ID number is not available, a device cannot be "copied" to the project if it cannot be identified by other available information.

Auto-IP is used to set selected devices without a valid IP configuration so that they match the controller settings and that there will be no conflicts with other devices.

The command **I&M** is used to display I&M data ("Identification and Maintenance") of the device, e.g. serial number, hardware revision and software revision.

Show differences to project

This option opens a dialog in which the found devices and modules are compared to the configured devices.



(1) Copy before
(2) Copy after
(3) Change to
(4) Copy all
(5) Delete

Fig. 6-36: "Show Differences to Project" dialog: R_IB_IL_24_DO_8 only in the Project Explorer

The devices highlighted in green are identical, whereas the devices highlighted in red are only listed in the overview of found or configured devices.

- Use the commands "Copy (before)" and "Copy (after)", to copy the highlighted devices to the site of the "Configured devices". Copying depends on the command before or after the selected device.
- A configured device cannot be replaced by a scanned device with the command "Replace by". For this action, the concerned devices are highlighted.

Auto IP: This command automatically sets a valid IP address if the address scanned by the (highlighted) device is invalid.

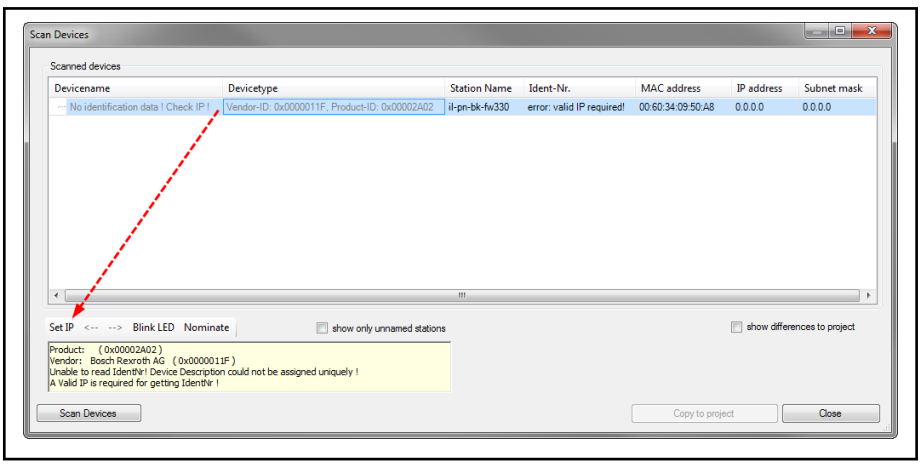


Fig. 6-37: Dialog "Scan devices", device without any identification data found

Press <Set IP> after highlighting the line. Missing data is replaced as defined in the default setting (for the result, refer to fig. 6-35 "Dialog Scan devices for Profinet I/O, bus terminal with DI8 module found" on page 110).

<-- Assign online device: The settings of the configured device are assigned to the scanned device using this command.

--> Assign configured device: The settings of the configured device are assigned to the scanned device using this command.

Blink LED: A listed device can be identified at the hardware using this command.

The device should react with a flashing signal (often "Link ok").

Name: Applies the changed **station name**.

Configuration adjustment

Note that this functionality is only available for Profinet I/O bus systems!

By using this command, the configuration settings of a Profinet I/O device defined in the project can be easily synchronized with the settings of a device (same device and manufacturer ID) in the connected hardware.

Consequently, the local settings are overwritten with the hardware settings.

The "Scan devices" makes a preselection and displays it.

Thus, the user has to compare the preselection and the physically available devices and to adjust the selection in the dialog.

Perform the following steps:

- Make the communication settings accordingly.
- Select the device object in the device tree in offline mode and the command "Configuration adjustment" in the context menu.
- The "Scan devices" dialog opens and it is searched for the matching devices in the hardware. The devices found are listed. The devices already used in the project (identified by means of the station name) can be hidden with "Hide used devices".
- It can be searched again via the "Scan devices" button.

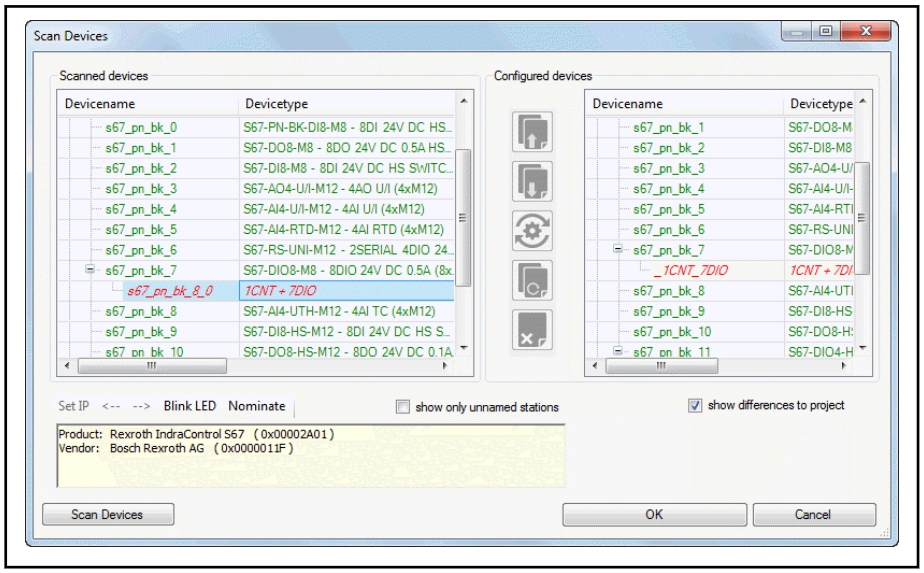


Fig. 6-38: Dialog "Scan devices" for offline configuration adjustment, example

<-- Assign online device: The settings of the configured device are assigned to the scanned device using this command.

--> Assign configured device: The settings of the configured device are assigned to the scanned device using this command.

Subdevice objects are not removed.

6.5.5 Profinet I/O controller, topology and device exchange

Topology and device exchange, general information

Overview: As for other field buses, "slaves" (I/O devices) can also be exchanged with Profinet without accessing IndraWorks. Thus, the “Profinet topology detection” is used to determine and store the network and device structure of a Profinet device setup. Based on this topology detection data, exchanged I/O devices bus can be operated again at the Profinet field bus.

Name-based device addressing: For Profinet, a logic, individual and unique device name (device name, station name, IdentStr) is assigned to each device (I/O device). The Profinet I/O controller uses this device name to assign the configured I/O address to the I/O devices. In the following, the actual Profinet data traffic is performed via this IP address and thus via the resolved MAC address.

IP addresses and/or device names are accessed with the DCP protocol (DCP=Discovery and Configuration Protokoll) available with every Profinet device by default.

Neighboring detection: Neighboring devices can exchange information using LLDP (Link Layer Discovery Protokoll). Each device sends information on itself in intervals. The respective neighbor stores this information for each of its network ports.

Topology detection: The LLDP information of all configured devices is retrieved by the Profinet I/O controller, summarized in a topology list, backed up in XML format and stored on the control.

Exchanging an I/O device

To replace an I/O device in a system, only the device name of the I/O device has to be transferred to the substitute device.

Due to this device name, the new I/O device is detected by the Profinet I/O controller, obtains a matching IP address and is thus included again in the product data traffic.

Options to exchange an I/O device:

To transfer the device name to the substitute device, there are different procedures:

1. The device name is stored on the storage medium. The storage medium is transferred from the original device to the substitute device. In this case, no special Profinet activity is required.
2. The device name can be set on the device. The settings on the substitute device are the same. In this case, no special Profinet activity is required.

   Example: S67-PN-BK
3. The device name is manually transferred (named) from the engineering system to the substitute device. This is for example possible in the IndraWorks Engineering in the **Profinet** dialog **Scan devices** ▸ **Name**.
4. The Profinet I/O controller determines and stores its network topology.

   If the device fails and a new compatible device appears, the device name is automatically transferred. This is now explained in detail.
5. An HMI application simulates the procedure of the engineering system. Not yet planned.

Requirements for the device exchange:

- To determine the complete device topology, all Profinet devices have to support LLDP. Switches used also have to be Profinet I/O devices at the same time and have to be configured in the engineering system.
- There may be no errors in the Profinet configuration to allow the Profinet I/O controller to study the topology completely.
- *Requirements on the substitute device:*
  - Structurally identical to the original device
  - Installation at the same position in the network
  - Use of the same port
  - A possibly still existing device name may not occur in the existing network.

    Ideally, the device is in the "Reset to factory default" state

Exchanging the device

Generating topology data:

New topology information is automatically created after each project download. The Profibus driver waits until the field bus is complete and started without errors.

If all configured devices are not provided with any diagnostics, Livelist data and LLDP data of the I/O devices are used to generate the topology information on the configured devices which is then stored in an XML file.

Specify name and storage location of the XML file via the configuration file. The Profinet driver supplements the name by the instance number of the Profinet I/O controller.

(Default is `/ata0b/PNTOPO_n.xml`; n – Instance number Profinet)

After switching on and off the control, The XML file is loaded again.

When downloading a PLC project, the XML file is discarded.

Topology data can also be created again in the PLC at any time using the function block `IL_PNIOStoreTopology`.

Exchanging a device:

At runtime, it is continuously monitored whether all configured I/O devices can be reached.

If an I/O device cannot be reached anymore, the neighboring devices of the missing device are checked (if available) to find out whether they detect another I/O device while trying to connect to the missing I/O device.

If a new I/O device is connected, it is checked for compatibility. It is checked whether the new device corresponds to the original device (using the VendorID and the DeviceID).

If compatibility is ensured, the new device obtains the device name of the original device. Subsequently, the Profinet I/O controller detects that there is a new I/O device with a matching name registered at the bus. It includes the I/O device in the production data traffic.

It is the same procedure for other non-reachable I/O devices.

PLC supports topology functions and device exchange functions

To support the topology functions and device exchange functions, the RIL_ProfinetIO.library includes several function blocks:

IL_PNIOAssignName

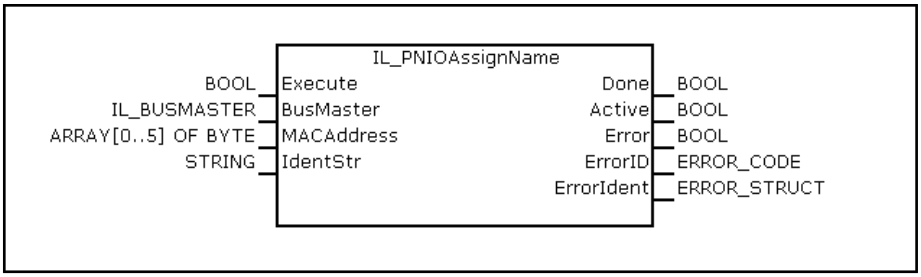


Fig. 6-39: IL_PNIOAssignName

The function block assigns a device name (ident string) to a Profinet I/O device.

This is also called "Name device".

The I/O device is addressed by its MAC address.

The MAC address is in the IL_PNIOGetLiveList function block.

IL_PNIOCheckTopology

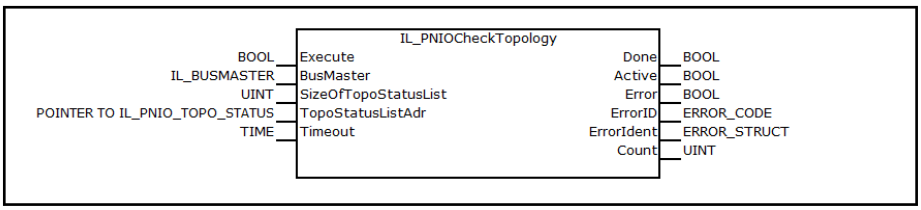


Fig. 6-40: IL_PNIOCheckTopology

The function block checks the saved topology of the Profinet device replacement function using the currently available Profinet topology. This is executed by reading out the port information from the Profinet I/O devices.

IL_PNIODeviceReplacement

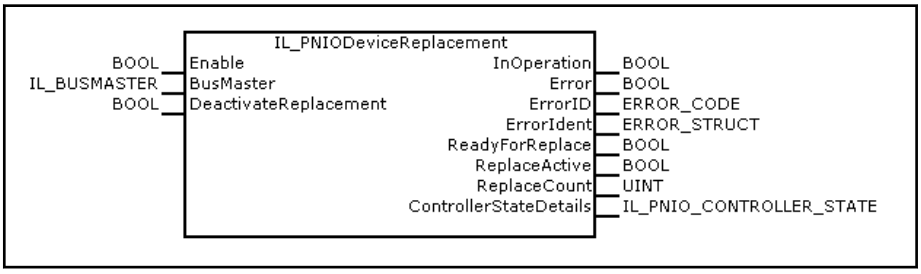


Fig. 6-41: IL_PNIODeviceReplacement

The function block provides a diagnostics via the status of the automatic device replacement function. The device replacement function can also be disabled.

Using the IL_PNIODeviceReplacement function block

- Call the IL_PNIODeviceReplacement function block cyclically to monitor the number of PN devices in the cyclic communication and report a diagnostics if required (refer to the "ControllerStateDetails" output).
- The "ReadyForReplace" output indicates that there is valid topology data (after the data was determined after a download/reset or successfully loaded after a control boot).

  The "ReadyForReplace" state can only be reached if the device exchange function is not disabled (input "DeactivateReplacement" in IL_PNIODeviceReplacement).

- Use the IL_PNIORestoreTopology function block to restore the topology (e.g. after a cross-exchange of devices) or to generally check the topology.

  In case of an error "topology error", the function block IL_PNIOCheckTopology can retrieve further details. It is necessary for both function blocks that "ReadyForReplace" is present and it is recommended to disable the device function (input "DeactivateReplacement" in IL_PNIODeviceReplacement) when executing "Restore" and "Check", as the internally used resources can block each other.

IL_PNIODeviceResetToDefault

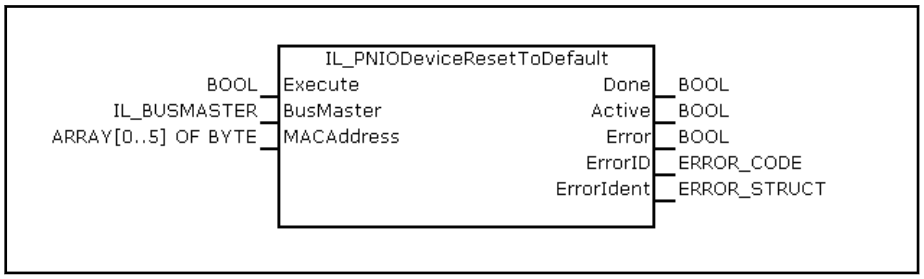


Fig. 6-42: IL_PNIODeviceResetToDefault

The function block sets the Profinet I/O device to its delivery state.

Background information: With this call, an already used I/O device is reset to its delivery state. Depending on the vendor and the I/O device, this can also be performed using other methods.

IL_PNIODeviceSetIP

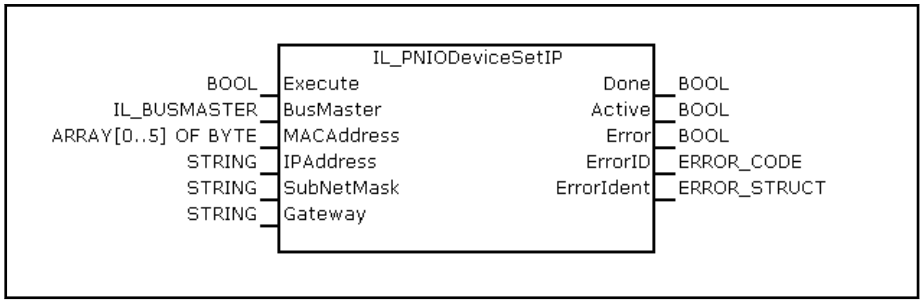


Fig. 6-43: IL_PNIODeviceSetIP

The function block sets the IP settings of a Profinet I/O device. The I/O device is addressed via its MAC address. The MAC address is in the IL_PNIOGetLiveList function block.

IL_PNIOGetLiveList

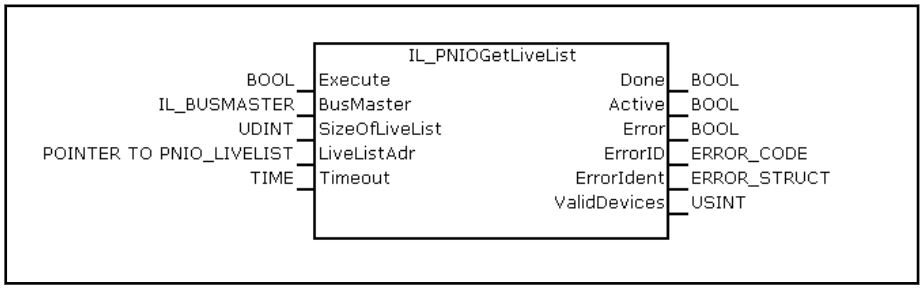


Fig. 6-44: IL_PNIOGetLiveList

The function block provides a list of the currently reachable Profinet devices.

The MAC address is used to address the I/O device. The MAC ADDRESS is in the IL_PNIOGetLiveList function block.

IL_PNIOGetLogMsg

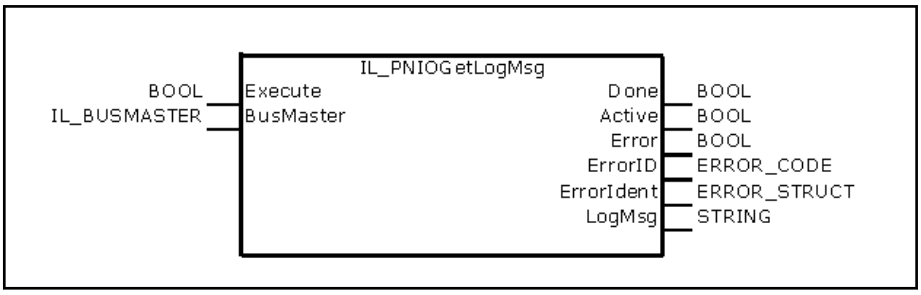


Fig. 6-45: IL_PNIOGetLogMsg

The function block retrieves a logger message to exchange the device from the Profinet I/O controller.

The oldest message is always retrieved. Every message can only be retrieved once.

Structure of the logger messages:

These logger messages can be present in the engineering system as well as in the function block IL_PNIOGetLogMsg.

Each logger message consists of three parts:

1. the term "PN_Topology"
2. The instance number of the controller 0..5
3. the actual message text

Message text	Explanation
new topology file stored	A new XML file with the topology information was created
new topology list generated	New topology information calculated
topology file deleted	Topology information was deleted, e.g. when deleting the project on the control
read topology file error	No file with topology information could be read.
fragmentary; DEVNAME not configured	The device called DEVNAME is available in the system hierarchy, but not in the configuration of the engineering system. Depending on the network structure and the device to be replaced, it might be possible that the substitute device cannot be "named".
device DEVNAME not in XML	The configured device DEVNAME is missing in the topology
missing device in config DEVNAME	The device DEVNAME from the topology file is missing in the configuration.
check device for DEVNAME	Another device was found for the missing device DEVNAME. It is now checked for compatibility.

Message text	Explanation
not fitting device type : DEVICETYPE	The compatibility resulted in a DeviceType string that does not match
not fitting VendorID : XXX-YYY	The compatibility resulted in a vendor ID XXX that does not match. ID YYY is expected
not fitting DeviceID : XXX-YYY	The compatibility resulted in a Device ID XXX that does not match. ID YYY is expected
Success naming DEVNAME	A device was successfully named with the name DEVNAME
Error naming DEVNAME	Naming the device DEVNAME was not successful

Tab. 6-4: Logger messages

Example: PN_Topology 0: new topology file stored

IL_PNIOStoreTopology

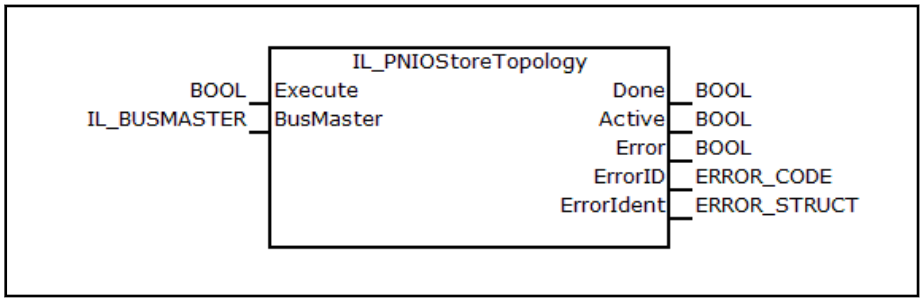


Fig. 6-46: IL_PNIOStoreTopology

The function block starts generating the current network topology basis if all configured I/O devices are available diagnostic-free.

The topology is stored in an XML file.

IL_PNIORestoreTopology

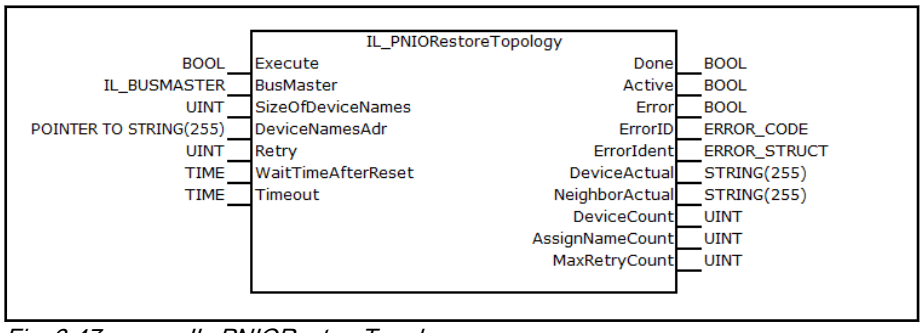


Fig. 6-47: IL_PNIORestoreTopology

The function block tries to restore the saved topology of the PNIO controller (e.g. use case "cross-exchange" of PN devices). The function block can also be used to check the topology information, i.e., whether the "neighboring information" of all configured PN devices result in a completely connected network.

Restrictions:

- Changes made at the network cabling are not automatically detected. To rebuild in an existing network, either the project has to be subse-

quently loaded again or the topology has to be determined again using the RIL_ProfinetIO.library, IL_PNIOStoreTopology.

- The DAP (DeviceAccessPoint) module ident number in a device (DeviceID) is currently not checked. This value of the substitute device can only be determined after complete startup of the device at the Profinet I/O.
- The I/O extension of modular devices is not part of the topology data. These values can only be determined after complete startup of the device at the Profinet.
- If a device is missing, it is scanned for **a** reachable neighbor of this device. If this neighbor has a matching device in its LLDP information, it is named. Thus, this device can be “incorrectly” connected to its other port, but it can still be named.

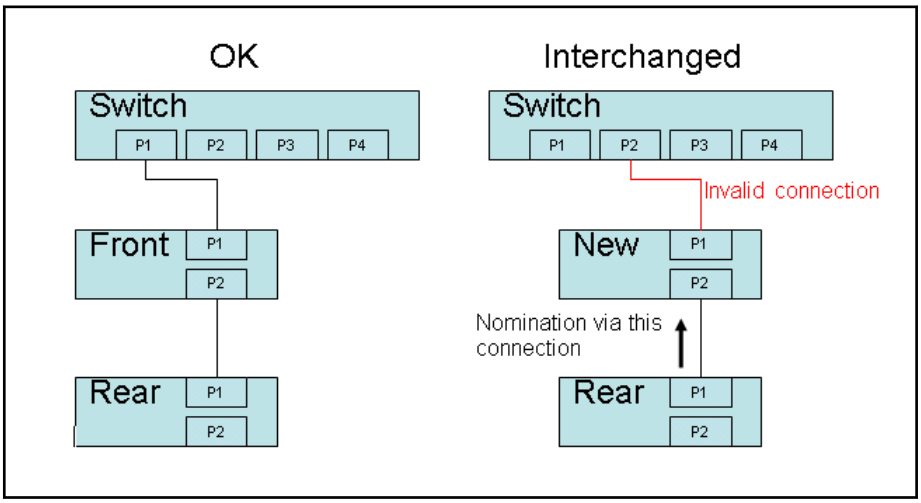


Fig. 6-48: Possible problems when naming a device

6.6 Profinet I/O, device (control as slave)

6.6.1 Configuring the Profinet I/O device

Profinet I/O device - Device editor

For XM2 and VPx controls, a Profinet I/O device can be configured at an RT-Ethernet extension module.

In the Project Explorer, open the node of the RT Ethernet extension module and then the context menu of the child node.

Select "Set device" from the context menu of the node and then "Profinet IO device XM" from the box.

The extension module is now configured

☞ For XM4 controls, the onboard interface can also be set as Profinet I/O device.

Select the node of the onboard interface in the Project Explorer and then "Set device".

Select "Profinet IO device XM" from the box. The onboard interface is now configured.

It is coupled between the controller control and the device via the image memory whose content is exchanged (see Profinet I/O device, controller-device coupling, page 127).

To open the editor window in the Project Explorer, double-click on the Profinet I/O device object.

The dialogs provide information on the device configuration of the Profinet I/O bus. Modifications can be made.

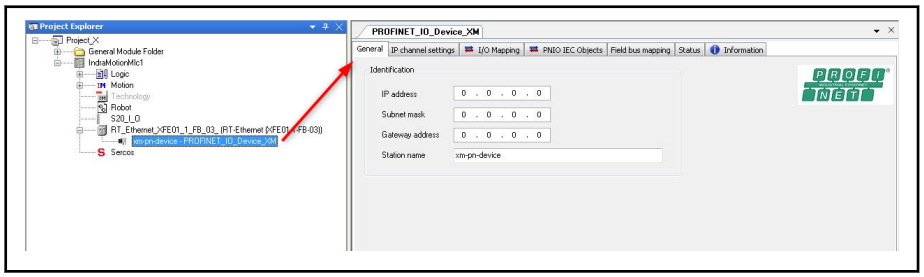


Fig. 6-49: Profinet I/O device - Tab “General”

Each configured Profinet I/O device requires 8..6 kB of the persistent retain data of the IEC program (as of the compatibility mode 14V22).

Proceed as follows to create persistent retain data:

1. Add "Global Persistent Variable List..." to the application, e.g. (default name "PersistentVars").
2. Select "Build" from (menu call: **Create** ▸ **Build** or press **Shift** + **F11**)
3. Selects the just added variable list in the Project Explorer.

   The persistent variable list is displayed in the IndraWorks menu.

4. Open the menu of the variable list and select the command "Add all instance paths".

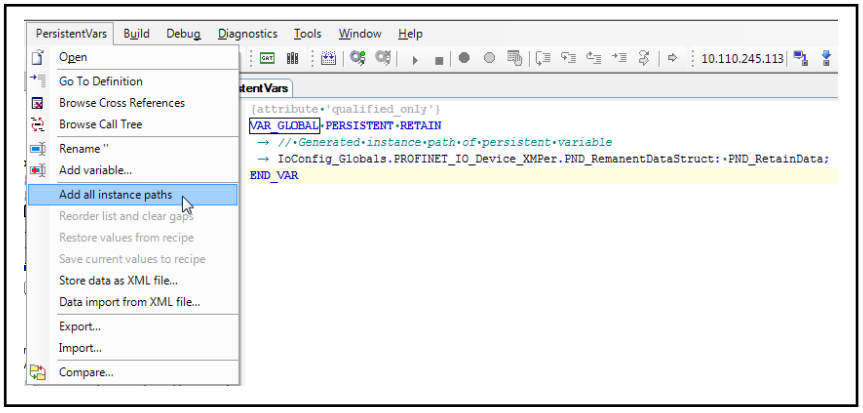


Fig. 6-50: Control as Profinet I/O device - Persistent retain data

An entry of type `PND_RetainData` is created for each configured Profinet I/O device.

See:

Configuration dialogs for a control as Profinet I/O device

- Tab “General” , page 122
- Tab “Field bus mapping”, page 123

- Tab “PNIO parameters” , page 123, this window is used for service purposes and is only visible if the option "Show generic configuration editors" is enabled under **Tools** ▸ **Options** ▸ **IndraLogic 2G** ▸ **Device editor**.
- Tab “I/O mapping”, page 123
- Tab “Status”, page 123
- Tab “Information”, page 124

Tab “General”

All settings in the dialog depend on the device description. Thus, whether they can be edited here and which values are specified or possible.

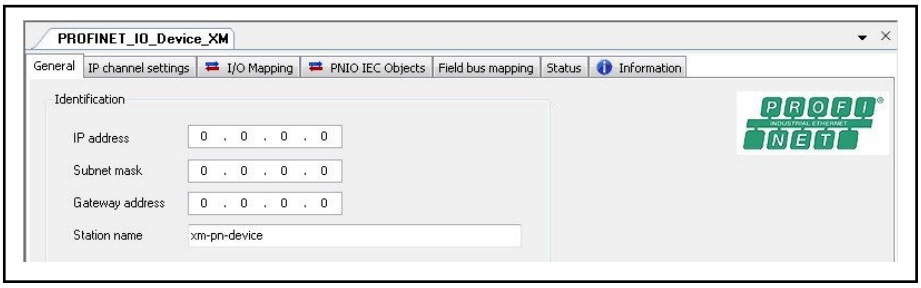


Fig. 6-51: Profinet I/O device - Tab “General”

Station name

Station name: The field contains the default entry.

The device is identified by a subnet-comprehensive unique device name.

IP parameters

The fields on "IP address", "Subnet mask" and "Default gateway" have to remain empty.

The entries into these fields have to be synchronized manually with the associated module to couple a Profinet I/O device of the Profinet I/O controller as shown in the figure below.

The station name is the decisive factor.

☞ *The station name may consist of the following characters:*

- Lower case letters without umlauts "a" to "z".

  Upper case letters "A" to "Z" are allowed, but internally converted into lower case letters.

  The station names are not case sensitive.
- Digits from "0" to "9" (may not be at the beginning of the name).
- Special hyphen character "-" and dot "." (may neither be at the beginning nor at the end of the name).
- The name may not exceed 240 characters. For some devices/vendors, a limit of 127 characters apply.

  The name can be divided into substrings using "." dots. Such a substring may not exceed 63 characters.

  The name may not start with the characters string "port-xyz" (with x,y,z = 0...9).

  The name may not have the sequence "a.b.c.d" (with a,b,c,d = 0...999).

☞ Specifying IP address parameters can cause connection problems. Set it only if it cannot be specified via Profinet I/O controllers!

Tab “IP channel settings”

Refer to chapter 4.6 "IP channel for Profinet I/O & EtherNet/IP (netX)" on page 32.

Tab “Field bus mapping”

Use the field bus mapping to access control variables from a parent field bus master.

To access a control variable from a parent master, assign an address to these variables first (mapping). The parent master can use this address to access the control data using acyclic services.

To use the field bus mapping, configure the control as corresponding field bus slave.

For details, refer to Field bus mapping, access from field bus master to variables of the field bus slave, page 23.

Tab “PNIO parameters”

This tab is used for service purposes and is only visible if the option "Show generic configuration editors" was enabled. Opening via the IndraWorks menu **Tools** ▸ **Options** ▸ **IndraLogic 2G** ▸ **Device editor**.

☞ Please clarify any possible modifications of the parameters with the service team.

Tab “I/O mapping”

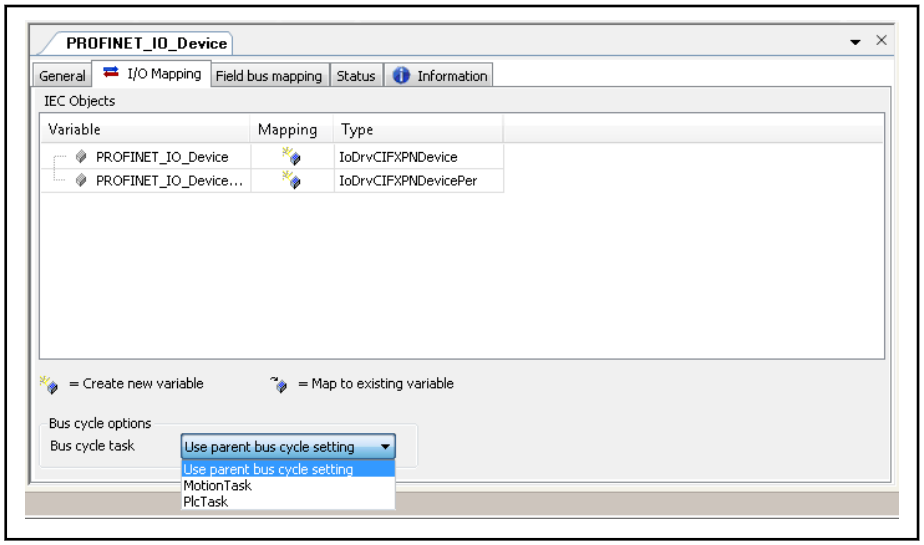


Fig. 6-52: Profinet I/O device - Tab “I/O mapping”

IEC objects: When the Profinet I/O controller is defined, the "IoDrvCIFXPNDevice.library" or RIL_ProfinetIODevice.library are applied automatically. The memory space required to implement the Profinet I/O device can now be defined.

Bus cycle options: Select a bus cycle task, to connect the cycle of the map exchange for the Profinet I/O controller to a particular task. In this task, it is useful to process the I/O data of the controller as well.

Tab “Status”

The "Status" tab in the "Profinet I/O device editor" displays status information (e.g. "Running", "Stopped") and specific diagnostic messages from the de-

vice. In addition, the diagnostic messages in the status flags are displayed in "Diag".

The output is a hexadecimal value determined by the set status flags (refer to the documentation "Protocol Interface Manual" for the Profinet I/O).

Tab “Information”

The tab displays general information from the device description file:

Name, Vendor, Categories, Version, Order number, Description and Image if available.

6.6.2 Profinet I/O device, adding coupling modules

General information

The interface from a Profinet I/O device to the Profinet I/O controller is implemented as common memory space. Its size can differ based on requirements.

The coupling area to exchange data with the Profinet I/O controller is determined with input and output modules at the Profinet I/O device object.

Add the corresponding input and output devices in the context menu of the Profinet I/O device object via **Add** ▸ **Device** ▸ **...**.

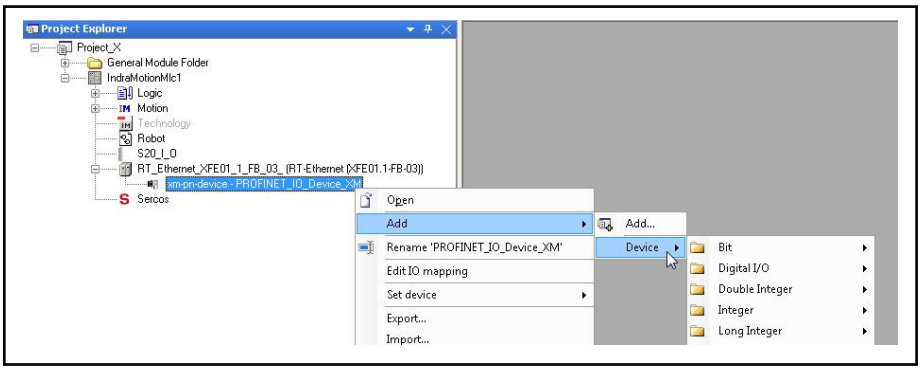
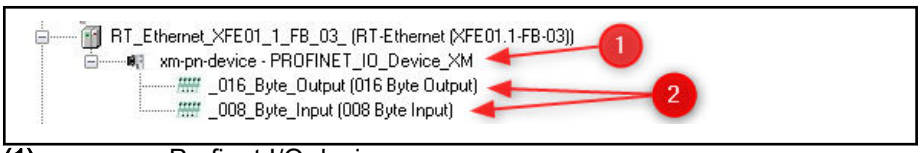


Fig. 6-53: Profinet I/O device: Add device

Data between the controller and the device is exchanged using common storage areas. The outputs of one are the inputs of the other and vice versa.

A check is required and if necessary, the device has to be manually adjusted to the modules of the controller.



(1) Profinet I/O device
(2) Coupling modules for data exchange between the controls.

Fig. 6-54: Profinet I/O device with memory modules to couple a Profinet I/O controller via a function module

Double-click on the module object to open the configuration dialog of the coupling modules inserted below the Profinet I/O device object.

Also open the dialogs via the context menu item **Open** of the module.

The following tabs are provided in the configuration dialog of the Profinet I/O device coupling modules:

Configuration dialogs for coupling modules of a Profinet I/O device to couple a Profinet I/O controller

- Tab “General” , page 125
- Tab “PNIO module parameters” , page 125, this window is used for service purposes and is only visible if the option "Show generic configuration editors" is enabled under **Tools ▸ Options ▸ IndraLogic 2G ▸ Device editor**.
- Tab “PNIO modules I/O mapping” , page 125
- Tab “Status”, page 126
- Tab “Information”, page 126.

Tab “General”

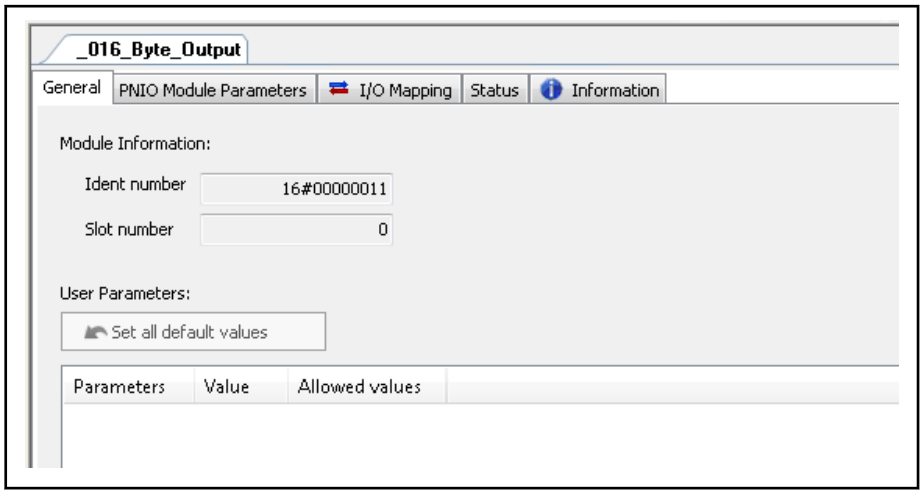


Fig. 6-55: Profinet I/O coupling modules - Tab “General”

Module information

- **Ident number**: Module ID from device description
- **Slot number:** 0 for memory mapping exchange

User parameters

None.

Tab “PNIO module parameters”

This tab is used for service purposes and is only visible if the option "Show generic configuration editors" was enabled. Opening via the IndraWorks menu **Tools ▸ Options ▸ IndraLogic 2G ▸ Device editor**.

Please clarify any possible modifications of the parameters with the service team.

The dialog provides information on position and size of the parameters.

Tab “PNIO modules I/O mapping”

The tab is used to assign module inputs and outputs to variables that can be used as local or global variables in the individual POUs.

The current value of the variables is displayed in online mode.

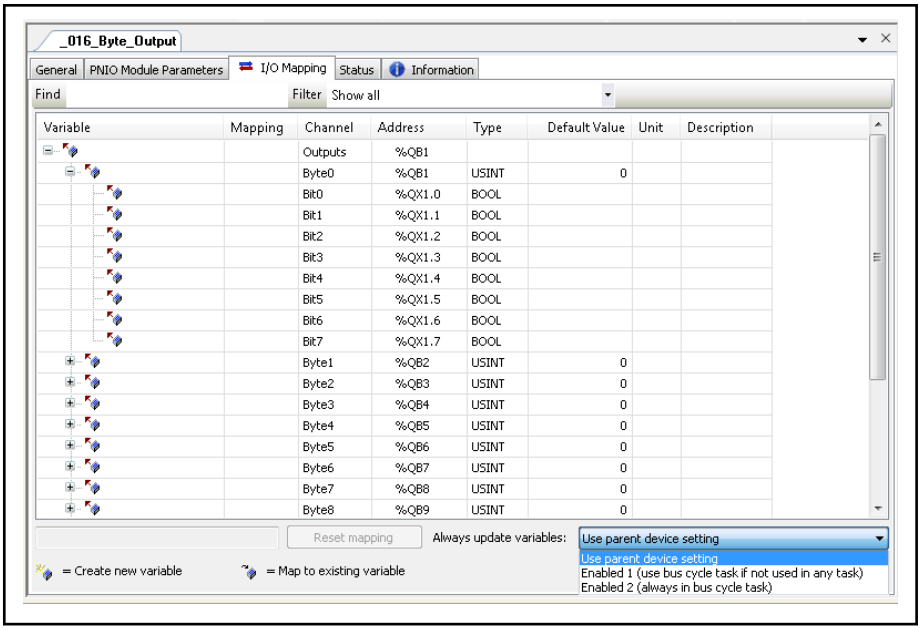


Fig. 6-56: Profinet I/O coupling module - Tab 'PNIO modules I/O mapping"

Channels: "Channel" contains variables of the I/O coupling module; the outputs in this case.

This assignment is described in Mapping of the field bus inputs and outputs, page 19,.

Reset Mapping: Use this button to reset the mapping settings to the default values defined in the device description file.

Update variables: Defines for the module whether I/O variables are updated in the bus cycle task.

The default value is defined in the device description file.

- **Use parent device setting:** Update according to the settings of the superordinate device.
- **Enabled 1 (use bus cycle task if not used in any task):** The I/O variables are updated in the bus cycle task if not used in any other task
- **Enabled 2 (always in the bus cycle task):** All variables are updated in each cycle of the bus cycle task irrespective of whether they are used or mapped on an input or output channel.

Tab “Status”

The "Status" tab in the "Profinet I/O device editor" displays status information (e.g. "Running", "Stopped") and specific diagnostic messages from the device. In addition, the diagnostic messages in the status flags are displayed in "Diag".

The output is a hexadecimal value resulting from the set status flags (see "Protocol Interface Manual" for the Profinet I/O).

Tab “Information”

The tab displays general information from the device description file:

Name, Vendor, Categories, Version, Order number, Description and Image if available.

6.6.3 Profinet I/O device, controller-device coupling

General information

To couple a control used as Profinet I/O device, a device is available under **Library ▸ Periphery ▸ Profinet /IO ▸ PLC** that allows to couple the following controls:

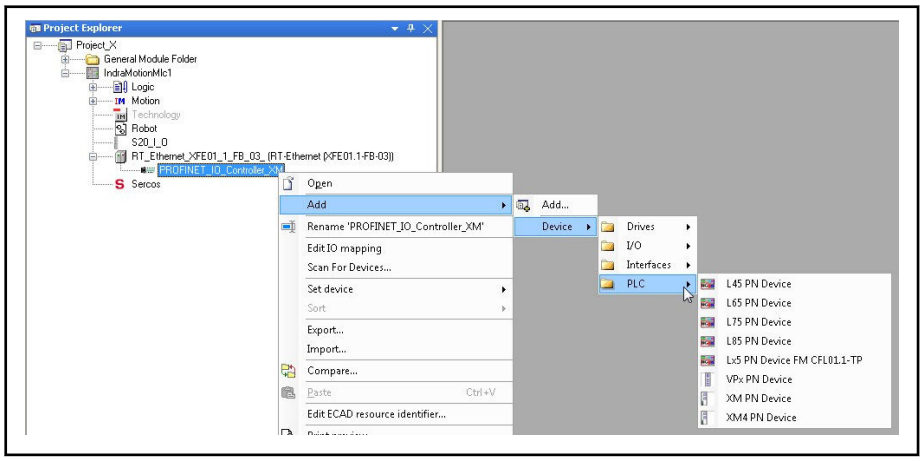
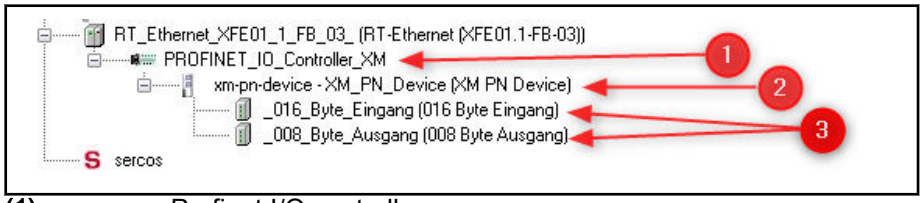


Fig. 6-57: Profinet I/O controller: Selecting the module to couple a Profinet I/O device control

- **XM PN Device** for the XM2 control via its extension module "RT_Ethernet_XFE01.1-FB-03"

When adding the module to couple a Profinet I/O device control in the Project Explorer, a firmware component in the controller is activated to connect to a Profinet I/O device via Profinet.

Each of these modules has a selection of (sub)modules in the device library that activates input and output variables in the image memory of the controller to exchange data between the Profinet I/O controller and the Profinet I/O device control.



(1) Profinet I/O controller
(2) Module for coupling
(3) Image memory modules to exchange data between the controls

Fig. 6-58: Profinet I/O controller with module to couple a Profinet I/O device via a function module

Configuration dialog Profinet I/O, controller-device coupling

- Tab “General”, page 128
- Tab “IOxS”, page 129
- Tab 'PNIO parameters', page 130, this window is used for service purposes and is only visible if the option "Show generic configuration editors" is enabled under **Tools ▸ Options ▸ IndraLogic 2G ▸ Device editor**.
- Tab “Status”, page 130
- Tab “Information”, page 131.



The dialogs of the image memory correspond to those on the device (Profinet I/O device, adding coupling modules, page 124).

Tab “General”

To open the editor window in the Project Explorer, double-click on the Profinet I/O module to couple a Profinet I/O device.

All settings in the dialog depend on the device description. Thus, whether they can be edited here and which values are specified or possible.

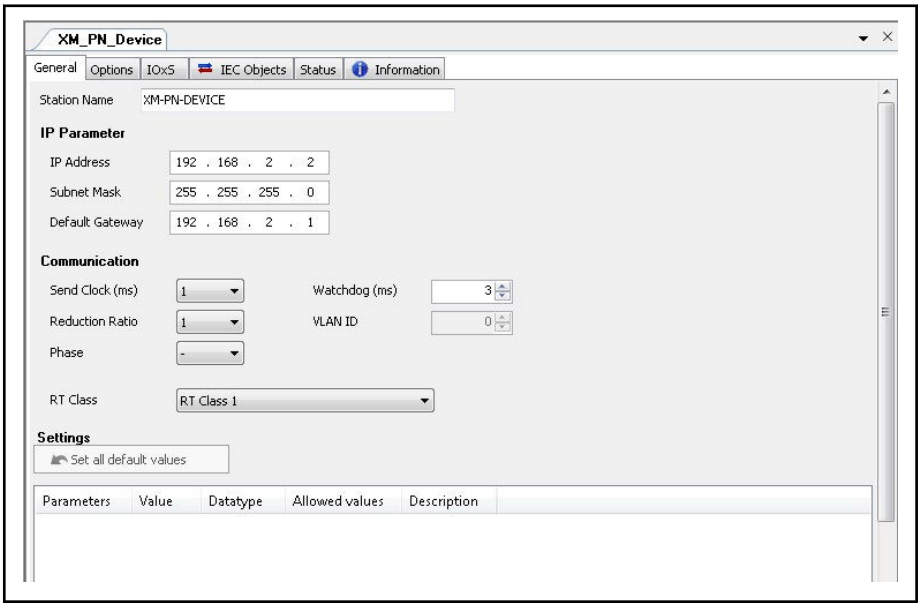


Fig. 6-59: Profinet I/O, controller-device coupling - Tab “General”

Station name

- Station name: Default setting "XM-PN-DEVICE"



The station name may consist of the following characters:

- Lower case letters without umlauts "a" to "z".

  Upper case letters "A" to "Z" are allowed, but internally converted into lower case letters.

  The station names are not case sensitive.
- Digits from "0" to "9" (may not be at the beginning of the name).
- Special hyphen character "-" and dot "." (may neither be at the beginning nor at the end of the name).
- The name may not exceed 240 characters. For some devices/vendors, a limit of 127 characters apply.

  The name can be divided into substrings using "." dots. Such a substring may not exceed 63 characters.

  The name may not start with the characters string "port-xyz" (with x,y,z = 0...9).

  The name may not have the sequence "a.b.c.d" (with a,b,c,d = 0...999).



Each device has to have a unique station name in the subnet, as it is required for certain functionalities in network mode!

IP parameters

The following addresses have to be configured according to the current environment to identify the controller.

- IP Address: Default setting "192.168.2.2", derives from the controller address and the adjacent devices/bus terminals.



Note the current address settings for devices added to the configuration via scanning when adjusting the IP address to prevent double assignments!

Refer to "Default slave IP parameter" on page 86.

- Subnet mask: Default setting "255.255.255.0"
- Default gateway: Default gateway, default setting "192.168.2.1" (gateway address optional)

Communication

These parameters describe the time response of the communication on the Profinet.

The product of the two parameters

`Reduction Ratio` and

`Send clock`, the actual cycle time results:

`(t = Reduction Ratio * Send clock ),`

acc. to sending device data:

- **Send clock (ms):** Transmission period in milliseconds.
- **Reduction ratio:** Factor to calculate the cycle time from the transmission period.
- **RT Class:** Currently, RT_Class_1 is supported.
- **VLAN ID**: Enter a value between 0 and 4095 for the VLAN type 802.1Q.

  For VLAN type ISL, enter a value between 0 and 32767 if available.
- **Watchdog**: Time (ms): Monitoring time, default setting 1000 ms. Possible values: 0 to 65535.

  If the Profinet I/O device does not receive any signal from the controller within this period, a device-specific response occurs, e.g., an error message.

User parameters

- **Set all default values**: This command resets all selected settings to the default settings (see "Default value" column) of the GSDML file.

Currently, no user parameters are available.

Tab “IOxS”

I/O provider and I/O consumer data can be added with this tab.



Submodules without own I/Os also provide a provider state for inputs

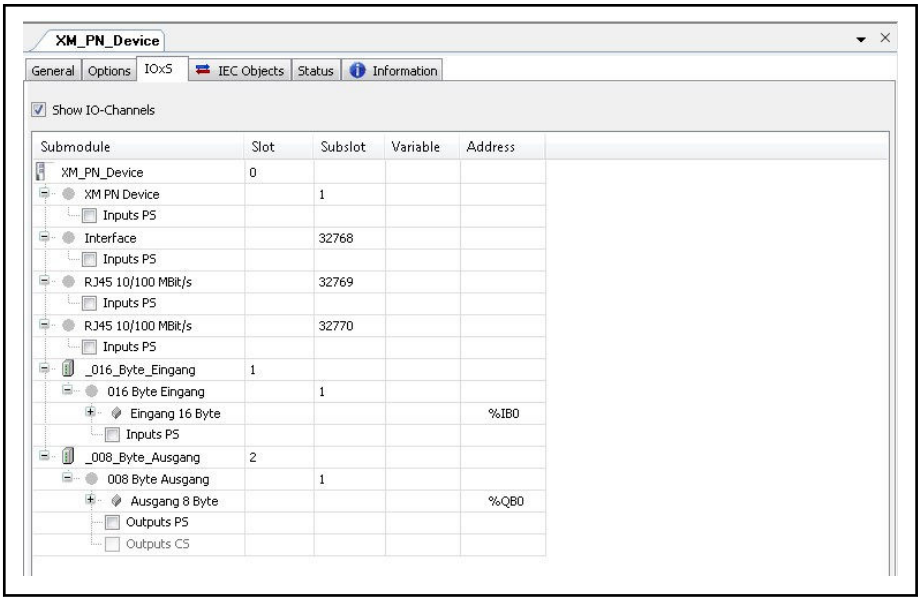


Fig. 6-60: Profinet I/O, controller-device coupling - Tab “IOxs”

If the checkbox is selected, the corresponding module provides additional I/Os with the provider state in the "I/O mapping" dialog. Double-click on the address field to open the corresponding module.

The status of the value hex 80 (bit 7) of the provider “state byte” indicates "OK". For detailed information on this byte definition, refer to the technical Profinet I/O specification.

☞ If an I/O provider state is mapped for output data, the PS output has to be set explicitly to "GOOD" (=0x80), as device data might not be applied and substitute values are used instead.

☞ If the application is stopped, the unmapped I/O provider states for outputs are set to "bad". Mapped I/O provider states are set to their configured default values.

Tab 'PNIO parameters'

This tab is used for service purposes and is only visible if the option "Show generic configuration editors" was enabled. Opening via the IndraWorks menu **Tools ▸ Options ▸ IndraLogic 2G ▸ Device editor**.

☞ Please clarify any possible modifications of the parameters with the service team.

☞ When the bus is running, modified parameters can be applied via the "Write parameter" button.

Tab “Status”

The "Status" tab in the "Profinet I/O device editor" displays status information (e.g. "Running", "Stopped") and specific diagnostic messages from the device. In addition, the diagnostic messages in the status flags are displayed in "Diag".

The output is a hexadecimal value resulting from the set status flags (see "Protocol Interface Manual" for the Profinet I/O).

Tab “Information”

The tab displays general information from the device description file:

Name, Vendor, Categories, Version, Order number, Description and Image if available.

6.7 Profinet I/O, Tool Calling Interface (TCI)

TCI (Tool Calling Interface) is an interface to call device tools (Windows programs) from an engineering system (e.g. IndraWorks). The device tools are used to parameterize field bus devices and they are provided by the device manufacturers. A use case for device tools is the proprietary computation of the checksum (iParCRC) using the individual device parameters (iParameter) of PROFIsafe devices entered into the engineering system. The device parameters are sent to the device tool via TCI. TCI was specified by "PROFIBUS International / PNO".

IndraWorks (engineering system) supports the TCI interface in the version TCI 1.1 for Profinet devices in the Conformance Class C2:

- To open the device tool, go to a context menu item at the device/module node in the Project Explorer
- IndraWorks sends data (TPF file) to the device tool while calling
- TCI is supported by the control families MLC and MTX

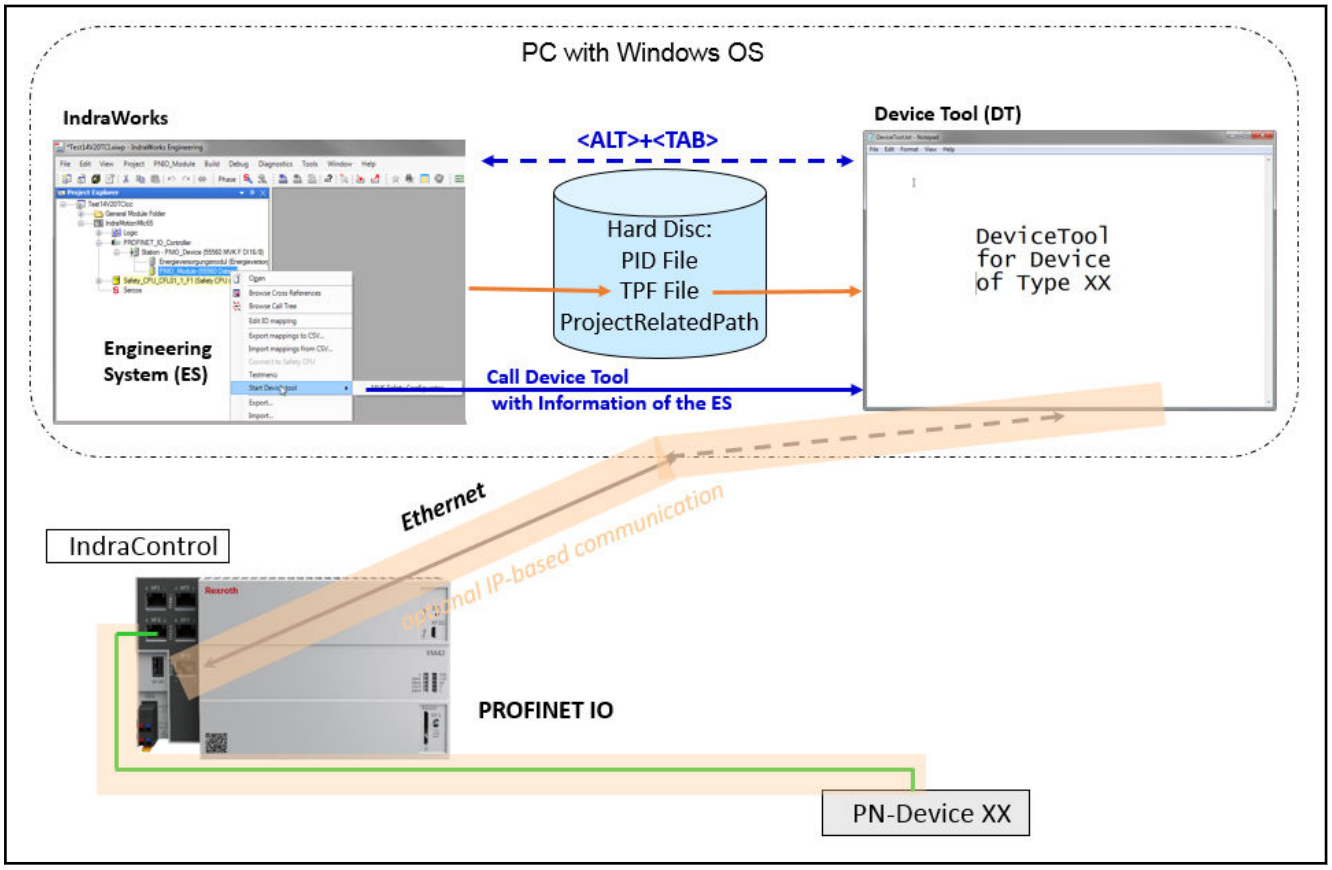


Fig. 6-61: Overview: TCI in IndraWorks

6.7.1 Steps to open a device tool from IndraWorks

1. Install the required device tool (TCI-capable device tools enter themselves into the Windows Registry)

2. Start IndraWorks Engineering
3. Add the GSDML file for the Profinet devices to the IndraWorks device library
4. Create a project. Add a control and enable a PNIO controller
5. Add the Profinet device with modules, etc. and parameterize them using the "PNIO parameters" editor
6. For the Profinet device/module, open the context menu in the Project Explorer. If a device tool is available, the context menu item "Start device tool" is displayed (IndraWorks automatically scans the Windows Registry whether a device tool is installed for the Profinet device)
7. Normally, another context menu item is displayed with the name of the device tool. Click on this context menu item to open the device tool. While opening, IndraWorks provides data to the device tools using the TPF file

6.7.2 Functional properties of the TCI interface in IndraWorks (engineering system)

Data transfer from IndraWorks (engineering system) to the device tool

IndraWorks transfers data via the TPF file when the device tool is called.

For the generated TPF file (TPF[n].xml), go to the subfolder "TCI\TPF" in the path "_LinkToCommonAppData" (e.g. "C:\ProgramData\Rexroth\IndraWorks \A"). "n" is an incremented number.

Basic characteristics of TCI

☞ TCI does not allow any return of data from the device tool to IndraWorks.

☞ The device tools do not depend on IndraWorks. Open device tools are not closed when closing the IndraWorks project or when exiting IndraWorks for example!

☞ The context menu item "Start device tool" is also provided for Profinet devices/modules disabled in IndraWorks. The information on Profinet device/module being disabled, is not included in the TPF file. Thus, an (offline) engineering of these devices/modules is also possible using device tools.

☞ The context menu item "Start device tool" is also provided for Profinet modules shared in another Profinet controller or operated by it ("shared device"). However, the information that the Profinet module is a "shared device" is not included in the TPF file. Profinet modules operated by another Profinet controller are disabled in IndraWorks. Use the respective naming convention of these Profinet modules to identify "shared devices" in the device tools.

Data storage

IndraWorks provides a directory for each IndraWorks project to store device tool data. The directory is transferred to the device tool with the "ProjectRelatedPath" entry in the TDF file. Each device tool can create a subfolder and

store its data there. The directory is also archived when creating an IndraWorks archive.

By default, the path for ProjectRelatedData is the subdirectory "TCI" in the directory "Public Documents"

A folder is created for each project. The device tool can store its data in that folder. The name of the folder and the xiwp file of the IndraWorks project are the same.

Example: If a new project is created with the name "Project189", the name of the xiwp file is "Project189.xiwp". The "ProjectRelatedPath" is "C:\Users\Public\Documents\TCI\Project189".

Ensure that the name of the xiwp file is unambiguous.

The ProjectRelatedPath remains and is NOT deleted if the IndraWorks project is deleted.

In the TPF file, the path "C:\Users\Public\Documents\TCI" is specified as "ProjectRelatedPath". In the Windows Explorer, navigate to this directory as follows: **Libraries** ▸ **Documents** ▸ **Public Documents** ▸ **TCI**.

Communication from the device tool to the Profinet device

IndraWorks (engineering system) supports the TCI interface in the version TCI 1.1 for Profinet devices in the Conformance Class C2, i.e. the TCI communication server is not supported.

TCP/IP can be used to communicate between the device tool and the device. It is required that the Profinet device supports TCP/IP. Proceed with the following steps:

- Enter the IP address of the engineering interface of the control into the routing table of the engineering PC
- Set the IP address of the PNIO controller of the control in the Profinet devices as gateway address

6.7.3 The most important use cases together with TCI

Closing/exiting IndraWorks project

Open device tools are not closed when closing the IndraWorks project or when exiting IndraWorks!

The ProjectRelatedPath remains and is NOT deleted. The ProjectRelatedPath exists on its own and the device tools can still be saved under this path.

Saving an IndraWorks project

The content of the ProjectRelatedPath is not included into the xiwp file of IndraWorks. It an own path.

Saving IndraWorks project as

A new ProjectRelatedPath is generated and the respective path is created. The content (device tool data) of the present ProjectRelatedPath is copied to the new ProjectRelatedPath. The content of the ProjectRelatedPath is not included into the xiwp file of IndraWorks. It an own path.

☞ Ensure that the name of the xiwp file is unambiguous.

If "Save as" is used on an already existing project, the content of a possible already existing ProjectRelatedPath is overwritten with device tool data.

Archiving an IndraWorks project

The content of ProjectRelatedPath (device tool data) is included in the archive file. IndraWorks project data and tool device data is included in the IndraWorks archive file.

☞ Ensure that all data is saved from all open device tools before archiving. An option is to close all device tools before archiving.

Restoring an IndraWorks project

The content of the ProjectRelatedPath (device tool data) from the archive file is unzipped on the ProjectRelatedPath. The device tool data in the archive is now available under ProjectRelatedPath. IndraWorks project data is included in the xiwp file.

☞ It is recommended to close all device tools before the restoration in order not to cause access problems to files in the ProjectRelatedPath.

☞ Ensure that the name of the xiwp file is unambiguous.

If the "restoration" is performed on an already existing project, the content of a possible already existing ProjectRelatedPath is overwritten with device tool data from the IndraWorks archive file.

6.8 Profinet I/O, FAQs

6.8.1 Overview

There are notes on the following FAQs in the following chapters:

- Where to find the Profinet device description files (GSDML) for control systems?, page 134
- Does the IP routing not work between the engineering port of the control and the Profinet network?, page 135
- PLC logger message: PN_Topology <station-name> IP x.x.x.x multiple No-of-Peers n, page 135
- Profinet I/O controller: Renaming PN device does not work (at runtime)?, page 135
- Profinet I/O controller: Is ring topology/redundancy supported?, page 136
- Profinet I/O device (control as slave): XMx or VPx is not detected sporadically as PN device, page 136

6.8.2 Where to find the Profinet device description files (GSDML) for control systems?

See www.boschrexroth.com → Products → Electric drives and control →, select the control system, e.g. Embedded Controls → XM → XM22 → Downloads.

6.8.3 Does the IP routing not work between the engineering port of the control and the Profinet network?



The IP channel connection is supported by the MLC.

Currently not supported for the MTX system.

The IP channel can only function if no other interface of the control (e.g. engineering port or IP channel of another field bus interface) is provided with the same subnet as the IP channel configuration of the Profinet interface.

If there is an error, an entry is generated in the PLC logger, e.g. "netX IP-chan not started - subnet already used" (as ERROR or as INFO message).

Enable the display of the PLC logger if required:

Go to IndraWorks → Tools → Options → IndraLogic 2G → "Enable PLC logger" and double-click on the control in the project tree → and go to the "Log" tab.

In 14V22 and above, the IP channel has to be enabled manually. For more details and sample configurations, refer to chapter 4.6 "IP channel for Profinet I/O & EtherNet/IP (netX)" on page 32.

6.8.4 PLC logger message: "PN_Topology <station-name> IP x.x.x.x multiple No-of-Peers n"

Problem: The Profinet I/O controller exchange function detected that the Profinet topology cannot be identified clearly.

Cause: Normally, the reason is a standard switch that does not support the LLDP protocol. I.e. a unique neighboring identification of the Profinet device is not possible and thus, the device exchange function of the Profinet I/O controller does not work.

Solution:

- If the device exchange function is not required, ignore this message.

  The device exchange function can be disabled via the function block IL_PNIODeviceReplacement (from the RIL_ProfinetIO library), e.g. directly after the PLC program start.
- To use the device exchange function, all devices in the Profinet network have to support the LLDP protocol. If possible, implement and configure switch devices as Profinet devices.

For more details on the Profinet I/O controller device exchange function refer to chapter 6.5.5 "Profinet I/O controller, topology and device exchange" on page 114.

6.8.5 Profinet I/O controller: Renaming PN device does not work (at runtime)?

Initial situation: One Profinet configuration runs without errors and on Profinet is to be renamed (i.e. change station name).

Problem: If a Profinet device is renamed using the "Scan devices" dialog, renaming is not successful or the Profinet I/O controller renames the device automatically using the old station name.

Description: If a Profinet configuration runs without errors, The Profinet I/O controller saves the topology automatically (= device exchange function). Renaming a Profinet device is then analog to a "device exchange", i.e. the Profinet I/O controller renames the device again as per the topology information saved.

Solution: Proceed as follows to rename a Profinet device:

1. Change the station name in the IndraWorks configuration and load the project to the control via a PLC download.

   The "old" topology information of the previous project configuration is deleted automatically.

2. Rename the device via the "Scan devices" dialog.

   If the changed configuration runs properly, the device exchange function saves the new topology information.

To monitor the status of the device exchange function, use the function block IL_PNIODeviceReplacement (from the RIL_ProfinetIO library).

For more details on the Profinet I/O controller device exchange function refer to chapter 6.5.5 "Profinet I/O controller, topology and device exchange" on page 114.

6.8.6 Profinet I/O controller: Is ring topology/redundancy supported?

No, The Profinet I/O controller neither supports the ring topology nor the redundancy. The Profinet cabling always has to be in a line. The two Ethernet ports of the Profinet interface at the control system can use two lines if required.

6.8.7 Profinet I/O device (control as slave): XMx or VPx is not detected sporadically as PN device

If a control, e.g. XMx is configured as Profinet I/O device (onboard or extension module), a Profinet station name and the IP configuration for these Profinet interfaces can be set in the IndraWorks dialogs.

⇒ It is recommended to completely set this Profinet IP configuration (IP address, subnet and gateway) of the Profinet I/O device interface to 0.0.0.0.

A Profinet I/O device is only addressed via the Profinet station name. The IP configuration for a Profinet I/O device is configured in the Profinet I/O controller and written to the Profinet I/O device upon connection setup. This does not depend on whether the Profinet I/O device already had an IP configuration.

7 EtherNet/IP

7.1 Features of the EtherNet/IP scanner

The current implementation of the scanner includes the following functionalities:

- Cyclic data traffic ("implicit messaging")
- Acyclic data communication ("explicit messaging")
- EDS file import to the device database (note: modular EDS files are not supported)
- Adapter-related status via cyclic data
- Adapter-related diagnostics via function blocks
- Detailed diagnostics in the interface for all adapters
- BootP or DHCP mechanism for IP address assignment

Function/characteristics	Value
Maximum number of connections ("implicit messaging")	64
Baud rate	100 MBit/s
Auto negotiation/autocrossing	Yes
Min. cycle time	2 ms
Max. amount of cyclic input data	4096 bytes
Max. amount of cyclic output data	4096 bytes
Max. amount of cyclic input data per adapter device; adjusted with adapter devices	480 bytes
Max. amount of cyclic output data per adapter device; adjusted with adapter devices	480 bytes

Tab. 7-1: Technical data

7.2 Features of the EtherNet/IP adapter

The current implementation of the EtherNet/IP adapter (at the real-time Ethernet interfaces) includes the following functionalities:

- Cyclic data traffic ("implicit messaging")
- Diagnostics of the adapter
- Acyclic data traffic acc. to the mapping concept

Function/characteristic	Value
Max. number of input data; adjusted with scanner devices	480 bytes [1])
Max. number of output data; adjusted with scanner devices	480 bytes [1])
I/O connection	1 "exclusive owner", up to 2 "listeners"
I/O connection type	Cyclic, min. 2 ms

Function/characteristic	Value
Explicit messages	*CIP standard services:* • Set_Attribute_Single • Get_Attribute_Single *Rockwell-specific services:* • ReadDataTable • WriteDataTable
UCMM	Supported
Max. number of user-specific objects	20
Max. number of connections	8 explicit and implicit connections
DCHP	Supported
BOOTP	Supported
Baud rate	10 and 100 Mbit/s
Max. telegram data per acyclic telegram	450 bytes

[1]) Per connection

Tab. 7-2: Technical data

Supported CIP classes

CIP classes are part of the CIP specification of the ODVA.

They describe the properties of the objects irrespective of the physical interface, e.g., Ethernet, CAN (volume 1).

The physical interface is described in another specification.

This is volume 2 for EtherNet/IP describing the EtherNet/IP adaption to CIP.

The classes 1, 2, 4 described in volume 1 ("Common Industrial Protocol") are supported.

The classes 16#F5 and 16#F6 of volume 2 ("EtherNet/IP adaptation of CIP") are supported.

A "Vendor Specific Object" class 16#C7 is defined for acyclic data traffic.

The classes listed in the following table are described in detail.

Class	Name
16#01	Identity Object
16#02	Message Router Object
16#04	Assembly Object
16#F5	TCP/IP Interface Object
16#F6	EtherNet Link Object
16#C7	Vendor Specific Object

Tab. 7-3: Supported CIP classes

Identity Object (16#01)

The "Identity" class provides general adapter information uniquely identifying this adapter.

Instance	Name	Attribute	Name	Supported Services	
				Get Attribute Single	Get Attribute All
0	Class	1	Revision	Yes	Yes
		2	Max. Instance		
		6	Max. Class Attrib.		
		7	Max. Instance Attrib.		
1	Instance Attributes	1	Vendor ID	Yes	Yes
		2	Device Type		
		3	Product Code		
		4	Major Revision		
			Minor Revision		
		5	Status		
		6	Serial Number		
		7	Product Number		
		8	State		
		9	Conf. Consist. Value		

Fig. 7-1: Class "Identity Object": Class attribute (instance = 0) and instance attribute (instance = 1)

Message Router Object (16#02)

The "Message Router Object" provides connection points as classes or instances, which a client can use to address services (read, write). These messages can be sent from the client to the adapter both with connection (connected) and without connection (unconnected).

No services are supported for the Message Router Object (Predefined Standard Object 16#02).

A "Vendor Specific Object" 16#C7 is defined on the message router for acyclic data traffic. This object supports the Get/Set_Attribute_Single services. Various “Vendor-Specific Services" can be defined for this object (e.g., Read/Write DataTable, Read/Write Fragmented Data services).

Instance	Name	Attribute	Name	Supported Services	
				Get Attribute Single	Set Attribute Single
1	Instance Attributes	1	Einstiegspunkt Tunnelprotokoll	N/A	N/A
2			Variable 1	Yes	Yes
3			Variable 2 (read only)	Yes	No
4			Variable 3	Yes	Yes
...			...	...	...
0xFFFF			Variable 0xFFFF	Yes	Yes

Fig. 7-2: Class "Vendor Specific Object (16#C7)”: Only instance attribute (instance = 1, 2, 3, …, 16#FFFF)

Assembly Object (16#04)

The Assembly class can be used to combine several objects, even if they are different. Such objects can for example be input and output data. The manufacturer-specific instances are used to provide these objects in various arrangements. This results in an efficient process data exchange.

Instance	Name	Attribute	Name	Supported Services	
				Get Attribute Single	Set Attribute Single
0	Class	1	Revision	Yes	No
		2	Max. Instance		No
1-x	Instance Attributes	3	Data	Yes	Yes
		4	Size		No

Fig. 7-3: Class "Assembly Object": Class attribute (instance = 0) and instance attribute (instance = 1)

TCP/IP Interface Object (16#F5)

The "TCP/IP Interface Object" provides the setup to configure the TCP/IP network interface of an adapter.

Examples of configurable objects are IP address, network mask and gateway address of the adapter.

Instance	Name	Attribute	Name	Supported Services		
				Get Attribute Single	Get Attribute All	Set Attribute Single
0	Class	1	Revision	Yes	Yes	No
		2	Max. Instance			No
1	Instance Attributes	1	Status	Yes	Yes	Yes
		2	Configuration Capability			No
		3	Configuration Control			Yes
		4	Physical Link Object			No
		5	Interface Configuration			No
		6	Host Name			Yes

Fig. 7-4: Class "TCP/IP Interface Object": Class attribute (instance = 0) and instance attribute (instance = 1)

Ethernet Link Object (16#F6)

The “Ethernet Link Object" contains link-specific counter and status information for a communication interface of type Ethernet.

Instance	Name	Attribute	Name	Supported Services		
				Get Attribute Single	Get Attribute All	Set Attribute Single
0	Class	1	Revision	Yes	No	No
		2	Max. Instance	Yes	No	No
1	Instance Attributes	1	Interface Speed	Yes	Yes	Yes
		2	Interface Flags	Yes		No
		3	Physical Address	Yes		No
		4	Interface Counters (not yet implemented)	No		No
		5	Media Counters (not yet implemented)	No		No
		6	Interface Control	Yes		Yes

Fig. 7-5: Class “Ethernet Link Object": Class attribute (instance = 0) and instance attribute (instance = 1)

7.3 Features of the EtherNet/IP adapter (engineering port)

The current implementation of the EtherNet/IP adapter (standard Ethernet interface/engineering port) includes the following functionalities:

- Cyclic data traffic ("implicit messaging")
- Diagnostics of the adapter

Function/characteristic	Value
Max. amount of input data	128 bytes[1)]
Max. number of output data	128 bytes
I/O connection	1 exclusive owner, 1 listener
I/O connection type	Cyclic, min. 5 ms
Explicit messages	Get_Attribute Set_Attribute
UCMM	Supported
Max. number of user-specific objects	-
Max. number of connections	1 explicit and 1 implicit connection
DCHP	-
BOOTP	-
Baud rate	10 and 100 Mbit/s

Tab. 7-4: Technical data

7.4 EtherNet/IP, basics

7.4.1 General information

The field bus EtherNet/IP is based on Ethernet. The application protocol "Common Industrial Protocol (CIP)" on Ethernet was implemented and provided as open industry standard with the help of the Control-Net International (CI), the Open DeviceNet Vendor Association (ODVA) as well as the Industrial Ethernet Association (IEA).

Based on the standard TCP and UDP, EtherNet/IP allows the communication between the office network and the system to be controlled.

Most of the EtherNet/IP end devices support DHCP and BootP when specifying the IP address. As each standard-compliant Ethernet, EtherNet/IP is also not suitable for high level real-time applications (<1ms) such as the control of servo motors. The typical cycle time of an EtherNet/IP network is around 10 ms.

7.4.2 Terms and abbreviations

Term	Explanation
EtherNet/IP	"Ethernet Industrial Protocol": Application protocol "Common Industrial Protocol" (CIP) used on standard Ethernet (TCP/IP, UDP/IP). EtherNet/IP and CIP are managed by the ODVA ("Open DeviceNet Vendor Association", www.odva.org)
ENIP, EIP	Abbreviations for EtherNet/IP
Scanner	The EtherNet/IP field bus master is called **scanner**
Adapter	The EtherNet/IP field bus slaves are called **adapters**

1) *The data width of the coupling area is preset to 8 bytes of input data and 8 bytes of output data plus a 4-byte run header. Select the "Set device" context menu item to change it.*

Term	Explanation
Originator	The device establishing the connection is called **originator**
Target	The device establishing the connection is called **target**
O→T	Data direction from originator to target
T→O	Data direction from target to originator
RPI	"Requested Packet Interval": Cycle time for the cyclic I/O data telegrams
Assembly	Data is defined as assembly objects (e.g. cyclic input/output data or configuration data)
EDS file	"Electronic Data Sheet": Device description file for EtherNet/IP adapter used by IndraWorks to introduce the adapter devices
BootP	"Bootstrap Protocol": Predecessor of DHCP ("Dynamic Host Configuration Protocol") used to assign an IP address to a device

Tab. 7-5: Terms and abbreviations, EtherNet/IP field bus

7.5 EtherNet/IP, scanner

7.5.1 Configuring EtherNet/IP scanner

Overview

The EtherNet/IP configuration editor provides dialogs to configure EtherNet/IP scanner and adapter devices displayed in the device editor in addition to the standard dialogs.

To insert EtherNet/IP devices as objects into the device tree and to configure them, install the corresponding device description files (*eds files of the adapter devices) in the device database.

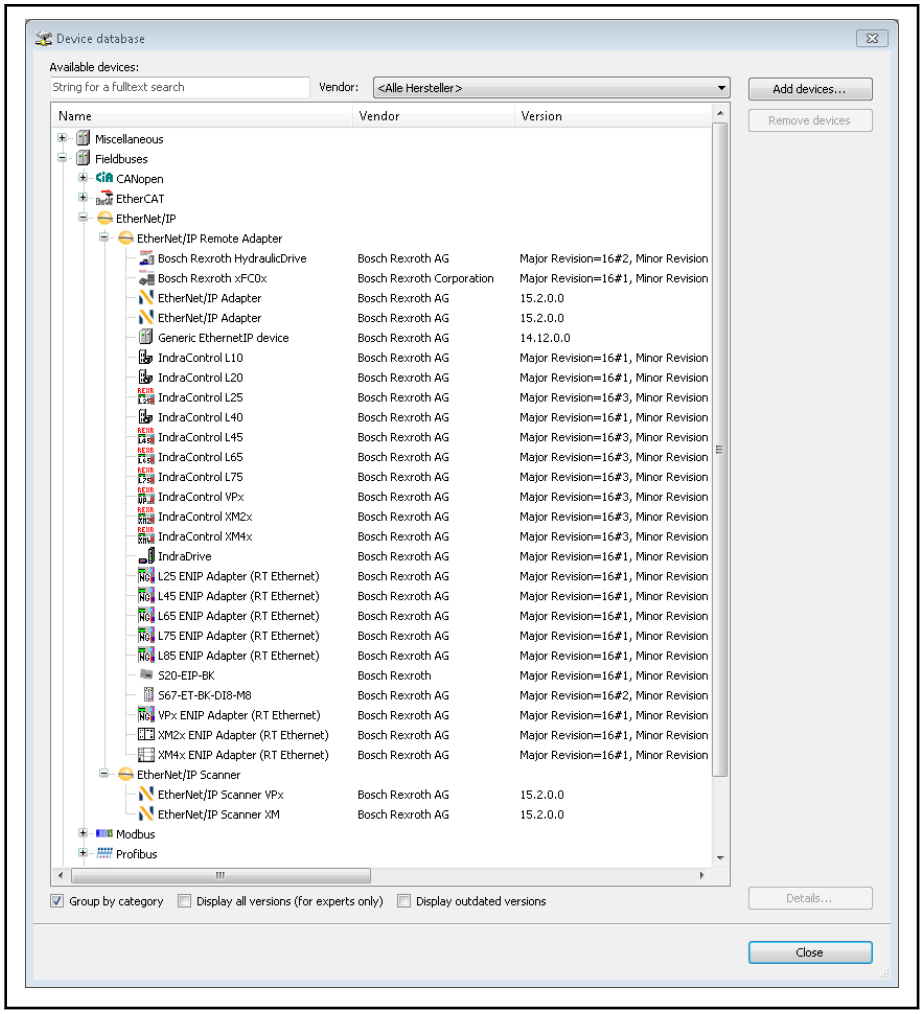


Fig. 7-6: Device database

Adding scanner

An EtherNet/IP scanner can be applied to the Project Explorer with "Set Device". Inserting an EtherNet/IP scanner into the device tree automatically integrates the corresponding device-specific library into the library manager.

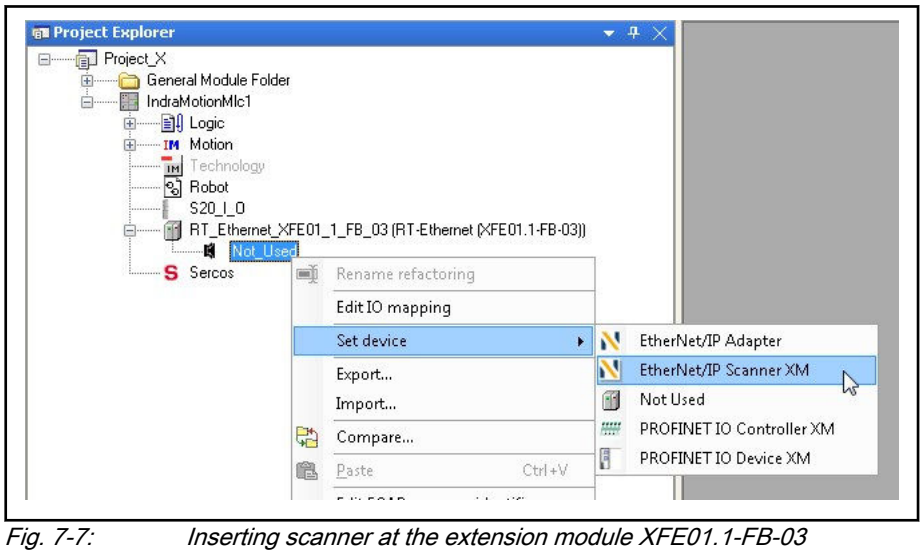


Fig. 7-7: Inserting scanner at the extension module XFE01.1-FB-03

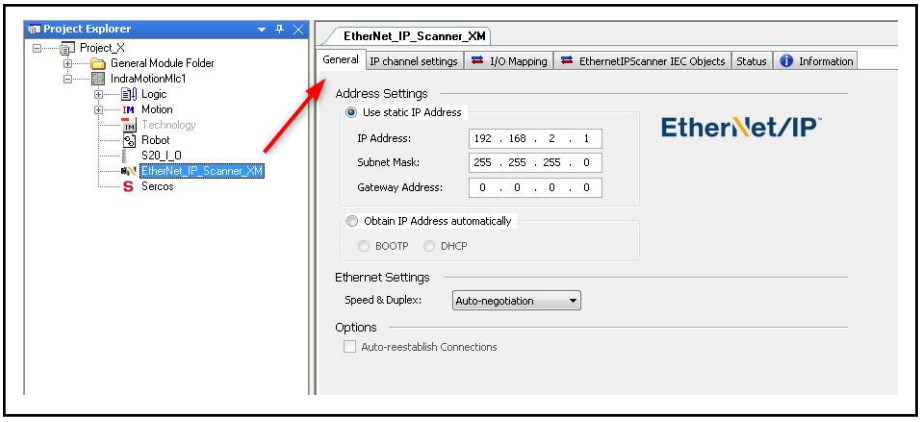


Fig. 7-8: EtherNet/IP scanner on the OnBoard interface of an XM4 control

Tab “General”

IP address, subnet mask and gateway address can be set for the EtherNet/IP scanner.

The IP address of the scanner can also be retrieved by a parent control via BootP or DHCP. Therefore, use the respective button.

Additionally, the network settings for direction and velocity can also be specified.

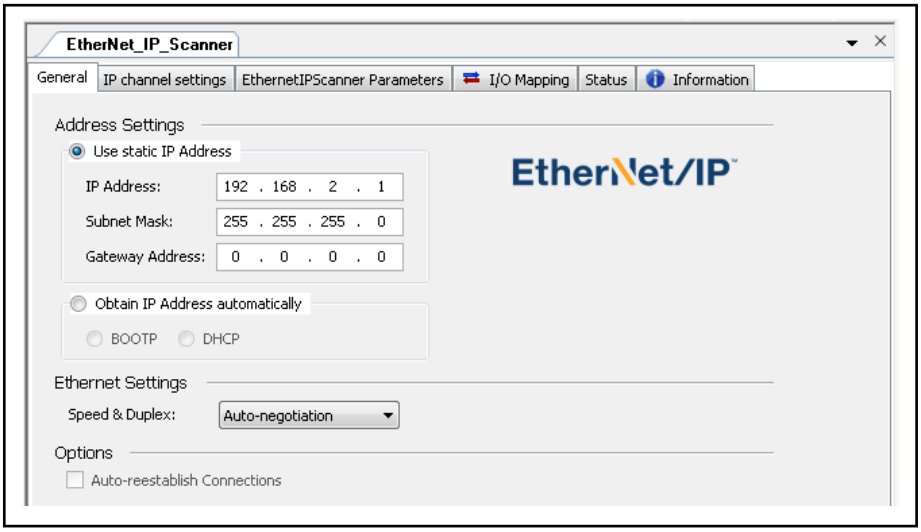


Fig. 7-9: EtherNet/IP scanner, "General" tab

Set IP address, subnet mask and gateway address in this window. If the button "Obtain IP address automatically:" is enabled, the scanner receives its address from a parent control or a BootP/DHCP server.

The MAC address of the scanner is on the device.

Tab “EtherNet/IP channel settings”

Refer to "“IP channel settings” tab" on page 32.

Tab “"EthernetIP scanner parameters”

The "EthernetIP scanner parameters" tab displays the communication parameters of the EtherNet/IP scanner.

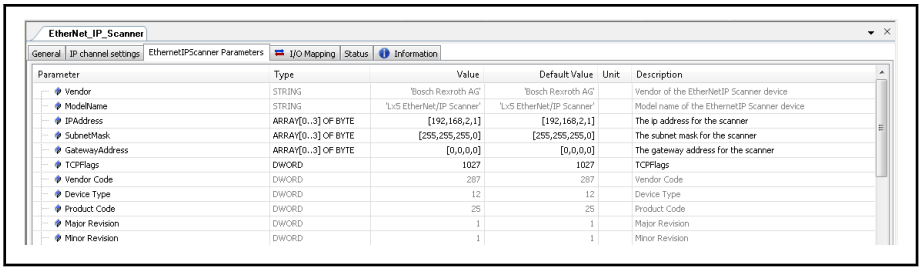


Fig. 7-10: EtherNet/IP scanner, "EthernetIP scanner parameters" tab

Some parameters (in black) can be changed at this location.

This window is used for service purposes and is only visible if the option "Show generic configuration editors" was enabled.

Call: IndraWorks menu **Tools** ▸ **Options** ▸ **IndraLogic 2G** ▸ **Device editor**

Please contact the service team before making any modifications in the editable parameters.

Tab “I/O mapping”

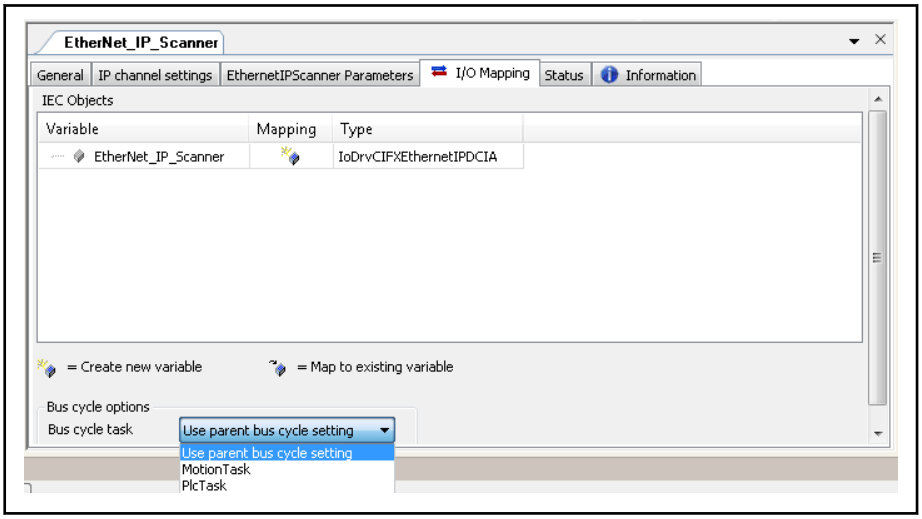


Fig. 7-11: EtherNet/IP scanner, "I/O mapping" tab

IEC objects: When the EtherNet/IP scanner is defined, "oDrvCIFXPROFINETDCIA.library" or RIL_EtherNetIP.library is automatically used. The memory space required to implement the EtherNet/IP scanner can be thus defined.

Bus cycle options

Bus cycle task: By selecting a bus cycle task, the cycle of the mapping exchange for the EtherNet/IP scanner can be connected to a particular task. In this task, it is useful to process I/O data of the master as well.

Default setting: Use parent bus cycle setting

With this setting, the task setting made in "Bus cycle options" of the control (double-click on the actual control in the Project Explorer, PLC settings) is applied to the actual bus.

Tab “Status”

The "Status" tab in the "EtherNet/IP IO Editor" displays status information (e.g., "Running", "Stopped") and specific diagnostic messages from the de-

vice. In addition, the diagnostic messages in the status flags are displayed in "Diag".

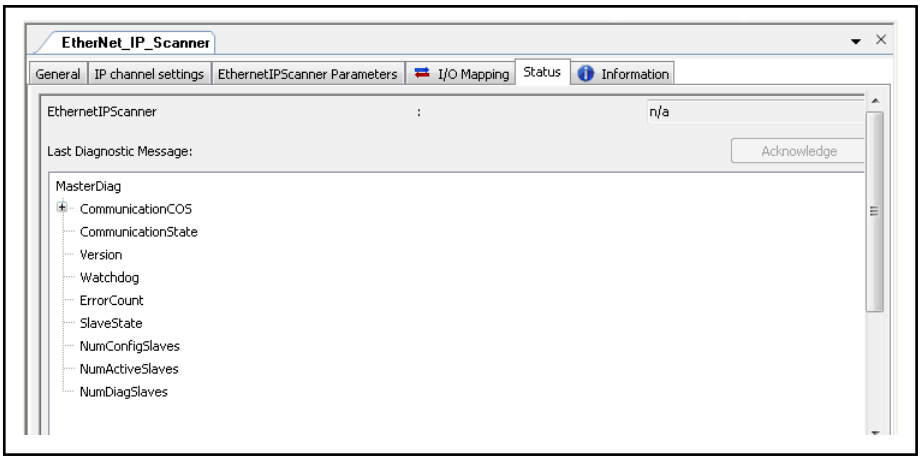


Fig. 7-12: EtherNet/IP scanner, "Status" tab (no diagnostic message present)

Tab “Information”

The tab displays general information from the device description file:

Name, Vendor, Categories, Version, Order number, Description and Image if available.

7.5.2 EtherNet/IP scanner, adding adapter

Adding adapter

One or multiple EtherNet/IP adapters can be added to the scanner. Drag&drop them from the device library via the menu item **Ethernet_IP_Scanner ▸ Add ▸ ...** or via "Add" in the context menu of the scanner.

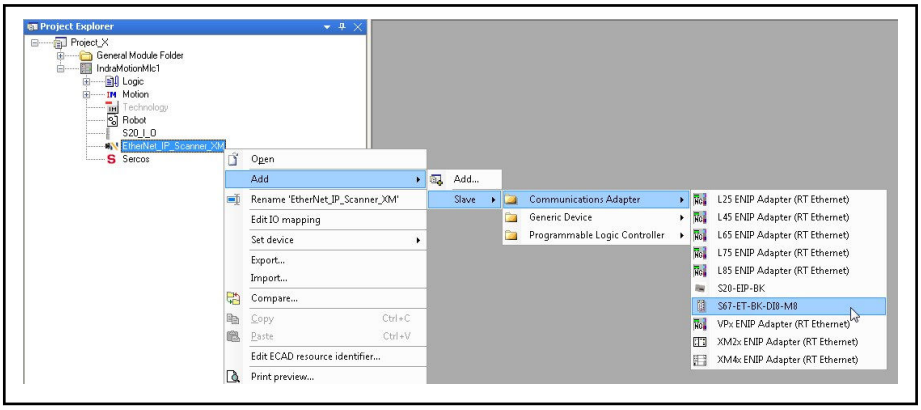


Fig. 7-13: Adding adapter

Tab “General”

When configuring an adapter, enter the target system settings.

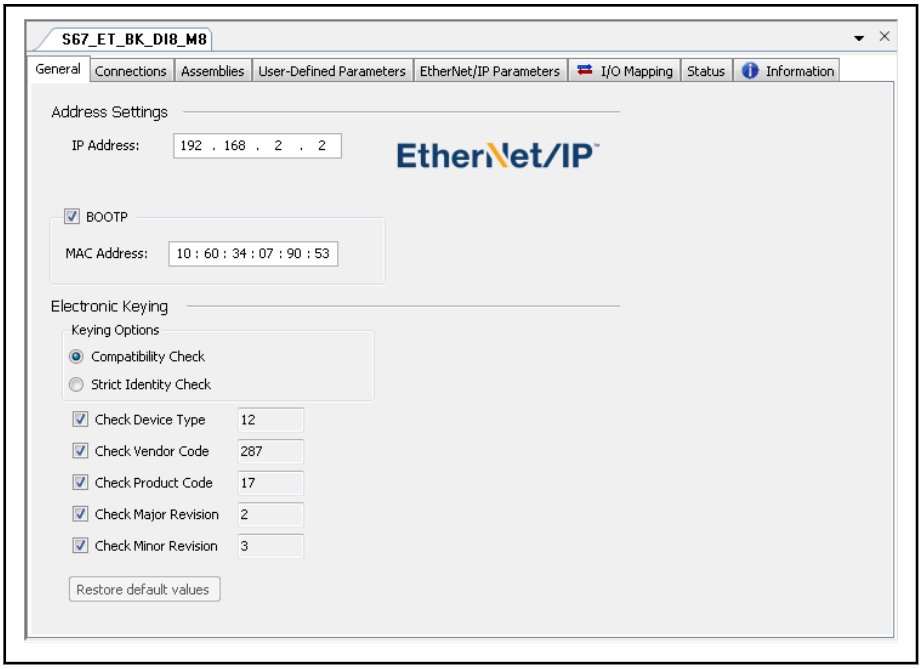


Fig. 7-14: EtherNet/IP scanner, adapter - "General" tab

Address setting	An adapter is addressed via its IP address. Enter this address here
BOOTP	If the adapter is not provided with an IP address, assign the configured IP address to the adapter via "BOOTP". Enter the MAC address of the adapter. **Note**: When exchanging the device and using "BOOTP", update the MAC address of the new adapter in the IndraWorks project.
Electronic keying	An EtherNet/IP device is uniquely identified with "Electronic Key". These values are applied from the device description (EDS file) of the adapter. *Keying options (from 14V08)* • **Compatibility Check (recommended)**: The electronic key in the configuration does not have to be identical to the one in the adapter. The adapter checks whether there is another version and whether it is compatible. EtherNet/IP communication is then permitted. If the compatibility bit is set, vendor code, product code and major provision may not be “0” or disabled! • **Strict Identity Check**: If this option is enabled when setting up the connection at the EtherNet/IP, all attributes of the electronic key "enabled" in the checkboxes in this configuration dialog, are checked. A disabled value is not checked. The minor revision is normally ignored.

Tab “Connections”

Enter the connections after the target system settings. Click on <Add Connection>.

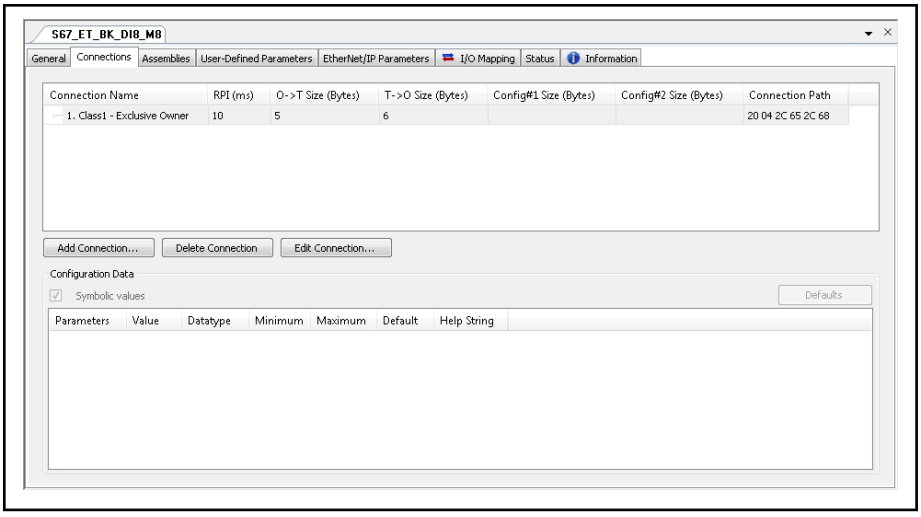


Fig. 7-15: EtherNet/IP scanner, adapter - "Connections" tab (entry was created using the following help dialog)

The connections are configured in this window. The example shows an established "Class1 Exclusive Owner" connection with 6-byte scanner output data (O->T) and 8-byte scanner input data (T->O).

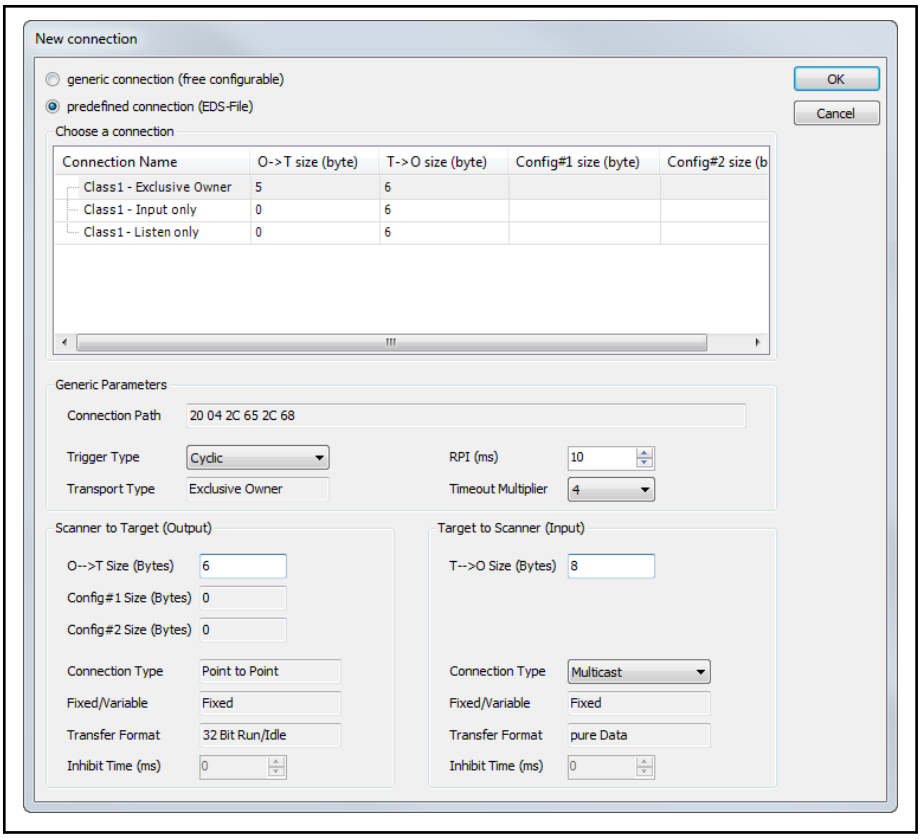


Fig. 7-16: Help dialog "New connection"

A generic connection can be created. All connection parameters are freely configurable (only for experts).

It is recommended to use the predefined connections from the device description (EDS file). Multiple connection properties are already specified (and cannot be edited) and editing the parameter is limited to the permitted value range.

☞ In the cyclic communication, some adapter devices send manufacturer-specific "status data" to the scanner. The input data length of the scanner "T→O Size" has to be increased by the length of the sent "status data".

For more information, refer to Notes on the EtherNet/IP connection length (O => T / T => O-size), page 163

☞ The connection parameters "Trigger Type", "RPI" and "Timeout Multiplier" are mostly provided with "reasonable" default values. For "Mode" (connection type), "Point-to-Point" is recommended.

Press <Delete Connection> to delete a highlighted connection (after prompt).

Press <Edit Connection> to postedit a highlighted connection.

Confirm whether the connection should really be deleted.

Describing connection parameters:

Trigger type	The trigger is only "cyclically" supported in the current version. Adapter devices can also support the triggers "Status change" and "Application"
Transport type	"Exclusive Owner" (input and output data) or "Input Only" (only scanner input data) or "Listen Only" (only scanner input data)
RPI	"Requested Packet Interval", cycle time of the cyclic I/O communication
Timeout multiplier	Number of telegrams that may be lost
O→T size	Data length from originator to target (scanner output data)
T→O size	Data length from target to originator (scanner input data)
Config#1 size	Amount of configuration data to target
Config#2 size	Amount of configuration data to target
Mode	"Connection type": Point-to-point, multicast or null
Type	"Length format": Fix or variable data length
Transfer format	Only data, with run/idle header or heartbeat
Inhibit time	Delay period to be at least present between two cyclic I/O telegrams (only relevant for trigger "Status Change" or "Application"; no function if "cyclic")

Tab “Assemblies”

In the "Assemblies" tab, the data structures of input and output data are displayed and can be edited.

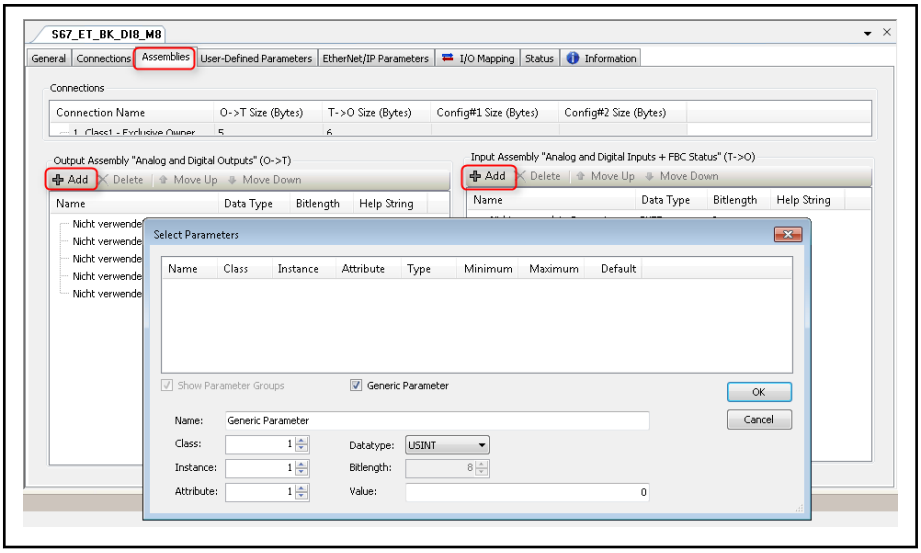


Fig. 7-17: EtherNet/IP scanner, adapter - "Assemblies" tab

Data structures and names in the device description (EDS file) are defined for some EtherNet/IP adapters. They are applied to the I/O mapping and can be adjusted if necessary.

If there are no data structures in the device description, the I/O mapping is shown in its original state as array of a simple data type (BYTE/UINT).

Individual data structures can be created and positioned accordingly ("Add", "Delete", "Move up/down").



When adding or deleting a data type, the I/O length (O=>T / T=>O size) is adjusted accordingly.

The I/O length of the connection has to match the corresponding data length in the adapter to establish an EtherNet/IP communication.

Tab “Status”

The "Status" tab in the "EtherNet/IP scanner, adapter" displays status information (e.g. "Running", "Stopped") and specific diagnostic messages from the device.

In the EtherNet/IP "Status" dialog, no function is assigned to the "Confirm" button.

7.5.3 "Generic Device"

General information

If no EDS file is available for an adapter device, a "Generic Device" can be created in the configurator of the scanner (already available in the device library). However, all required configuration parameters have to be entered manually, i.e. apart from the IP address and the MAC address if required, the values of the "Electronic Key" have to be entered.

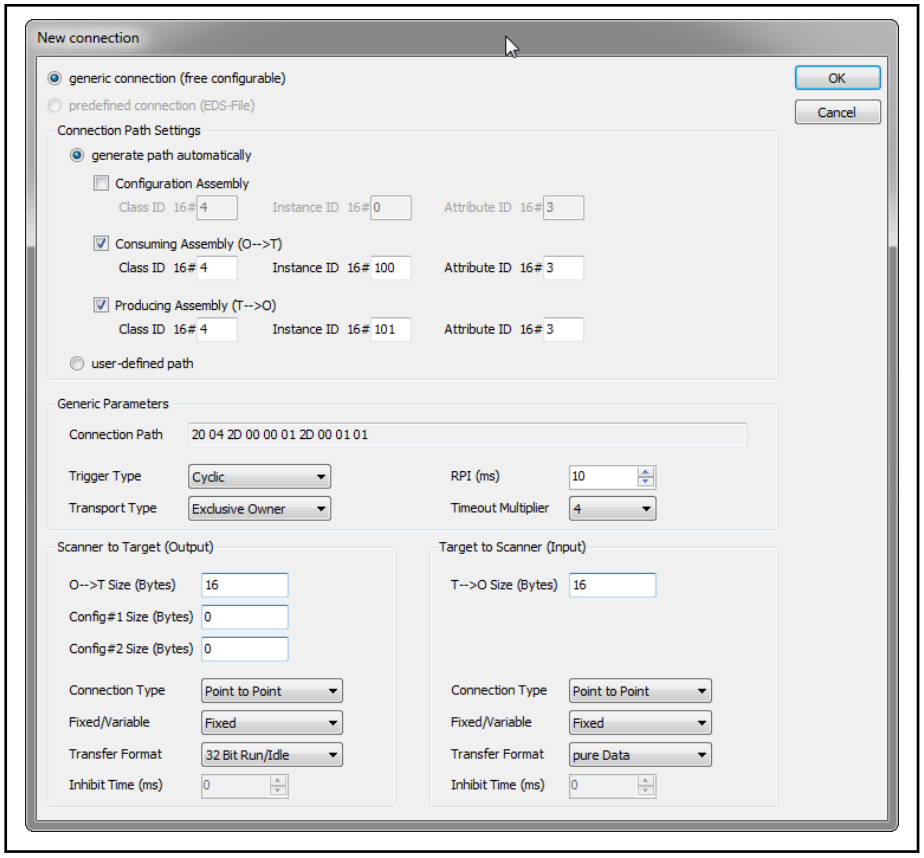


Fig. 7-18: "New connection" dialog

All data required to set up the connection can be entered into this window.

The "Generic Device" is not connected by default. A new connection has to be set up manually. Specific knowledge is required for the adapter device (e.g. assembly instances for the connection path). It is possible that an adapter supports only a limited value range for the connection parameters.

Tips and recommendations when configuring a "generic connection" (freely configurable)

Connection path settings

Configuration Assembly	It is rarely used and thus remains on "Not enabled" or "empty".
Consuming Assembly (O→T)	Also called "Output Assembly". Output data of the scanner received by the adapter. Enter the respective output assembly instance into the "Instance ID" field.
Producing Assembly (T→O)	Also called "Input Assembly". Input data in the scanner generated by the adapter. Enter the respective input assembly instance to the "Instance ID" field.

Generic parameters

Trigger type	Remains normally on "Cyclic".
Transport type	Remains normally on "Exclusive Owner".

RPI (ms)	Do not set the cycle time shorter than required. Cycles times ≥ 10 ms are mostly sufficient.
Timeout multiplier	The default value 4 is recommended.

More connection settings

O→T / T→O size	The connection length is entered here. The number in bytes for the "Output Assembly" (O→T) and for the "Input Assembly" (T→O).
Config #1 / Config #2 size	Only required if there is a "Configuration Assembly" instance (only for experts). Otherwise, the values remain on "0".
Connection type	The "Point-to-point" setting is recommended. It is also called "Unicast". Use multicast only if required.
Fixed/Variable	Use the setting "Fixed".
Transfer format	For "Scanner to target" (output), the format "32 Bit Run/ Idle" is normally used. For "Target to scanner" (input), it depends on the adapter device: • if it is a general I/O device, the format is normally "Pure data" • if the adapter is a control, it can also be "32 Bit Run/ Idle"
Inhibit Time (ms):	It is not used for the trigger type "Cyclic".

☞ Modular devices that require a "slot" configuration are not supported by the Rexroth EtherNet/IP scanner.

7.6 EtherNet/IP, adapter

7.6.1 Overview

The following control types support the EtherNet/IP adapter:

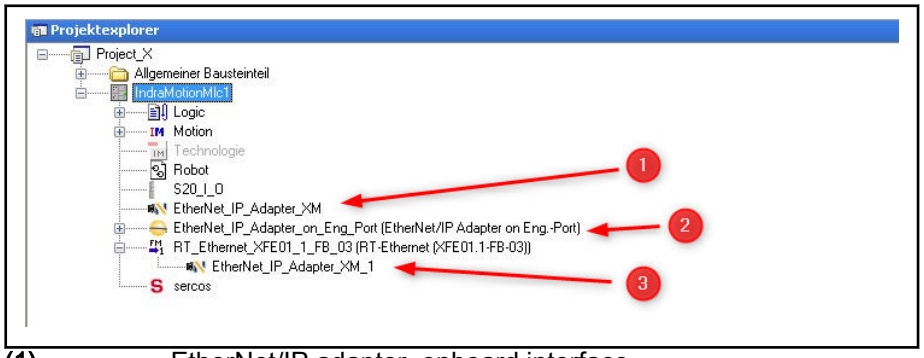
- **XM2**
  Connecting options:
  1. Via the engineering port
  2. Via RT-Ethernet extension modules:
     XFE01.1-FB-03 (master)
     XFE01.1-FB-04 (slave)
- **XM4**
  Connecting options:
  1. Via the engineering port
  2. Via the onboard interface
  3. Via RT-Ethernet extension modules:
     XFE01.1-FB-03 (master)
     XFE01.1-FB-04 (slave)
- **VPx**

Connecting options:

1. Via the engineering port
2. Via the RT-Ethernet extension module PFC01.1-FB-13

The engineering software available on the control is shown as an EtherNet/IP adapter object in the Project Explorer. The object can either be configured as EtherNet/IP adapter when creating the control or via the context menu item **Set device**.

For the position of the interfaces, refer to Field buses and field bus communication on page 11.



(1) EtherNet/IP adapter, onboard interface
(2) EtherNet/IP adapter, engineering interface, refer to EtherNet/IP, configuring adapter (engineering port) on page 158
(3) EtherNet/IP adapter, extension module

Fig. 7-19: EtherNet/IP adapter objects, not yet activated

EtherNet/IP adapter

- Tab “Adapter settings”, page 153
- Tab “I/O mapping”, page 155
- Tab “Field bus mapping”, page 156
- Tab “EtherNetIP parameters”, page 156, this window is used for service purposes and is only visible if the option "Show generic configuration editors" is enabled under **Tools ▸ Options ▸ IndraLogic 2G ▸ Device editor**.
- Tab “Status”, page 157
- Tab “Information”, page 157

7.6.2 Tab “Adapter settings”

In the "Adapter settings" tab, the target system settings of the EtherNet/IP adapters are made.

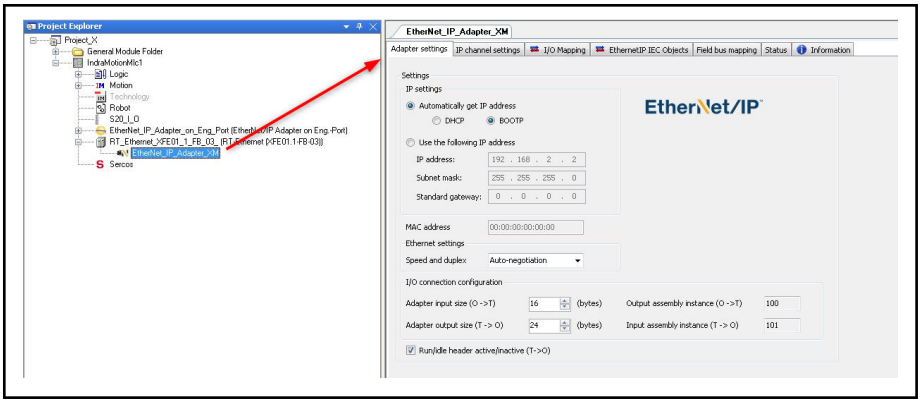


Fig. 7-20: Setting the EtherNet/IP adapter

Settings

IP settings	
Obtaining IP address automatically	The IP-address of the EtherNet/IP adapters can be automatically obtained: • **DHCP** - Obtaining IP configuration of a DHCP server automatically. • **BOOTP** - Obtaining IP configuration of a BOOTP server automatically.
Using following IP address	Manual IP address specification by specifying: IP address, subnet mask and gateway address.
MAC address	The MAC address of the EtherNet/IP adapter is automatically read and only displayed.
Ethernet settings	The velocity and the duplex of the Ethernet interface can be set under "Speed and Duplex". Possible values are: • autonegotiation • 10 MBits/half duplex • 10 MBits/full duplex • 100 MBits/half duplex • 100 MBits/full duplex
MAC address of netX interface	Initial "00:00:00:00:00:00", the real MAC address is only displayed if logged in. The MAC address is empty again after the user logged out.

I/O connection configuration	In the EtherNet/IP, the input/output assembly instances have to be defined as follows seen from the scanner (originator): • **Adapter input size (O -> T)** - Length of input data in bytes • **Adapter output size (T -> O)** - Length of output data in bytes • **Output assembly instance (O -> T)** - Display of output assembly instance • **Input assembly instance (T -> O)** - Display of input assembly instance **Note**: The specified length of the input and output data affects the setting options in the tab "I/O Mapping" (Tab “I/O mapping”, page 155).
Run/idle header active/ inactive (T->O)	Run/idle header (4 bytes) from adapter to scanner. **Note**: This option has to be selected (run/idle header active) if the scanner configuration is performed via the EDS file.

7.6.3 Tab “EtherNet/IP channel settings”

Refer to chapter 4.6 "IP channel for Profinet I/O & EtherNet/IP (netX)" on page 32.

7.6.4 Tab “I/O mapping”

The “I/O mapping” tab allows to map variables to exchange data with the scanner.

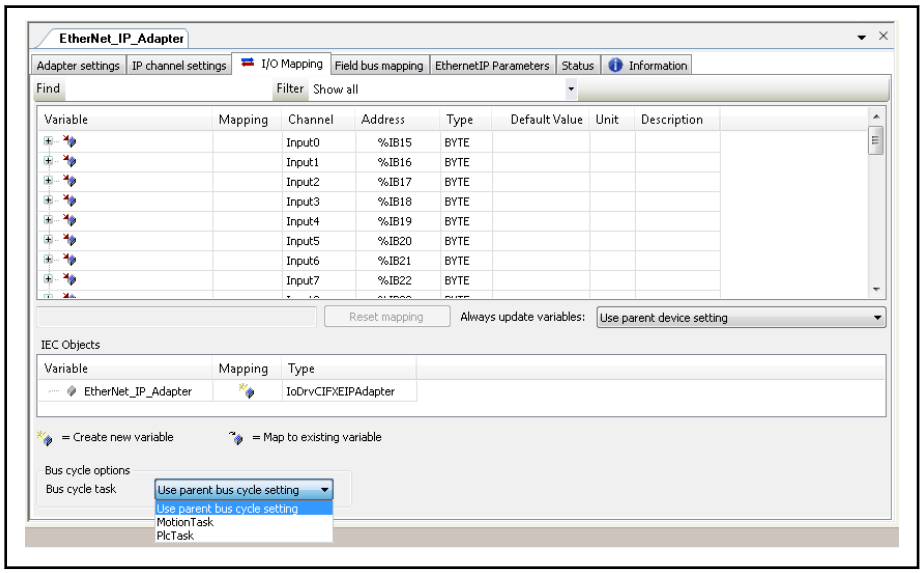


Fig. 7-21: EtherNet/IP adapter: I/O mapping, subsequent to establishing the connection

Channels: The upper part of the dialog provides the inputs and outputs for communication between scanner and adapter specified in the configuration of the connection.

The window is used to assign module inputs and outputs to variables that can be used as (local or) global variables in the individual POUs.

The current value of the variables is displayed in online mode.

This assignment is described in Mapping of the field bus inputs and outputs, page 19,.

Reset Mapping: Use this button to reset the mapping settings to the default values defined in the device description file.

Update variables: Defines for the module whether I/O variables are updated in the bus cycle task.

The default value is defined in the device description file.

- **Use parent device setting:** Update according to the settings of the superordinate device.
- **Enabled 1 (use bus cycle task if not used in any task):** The I/O variables are updated in the bus cycle task if not used in any other task
- **Enabled 2 (always in the bus cycle task):** All variables are updated in each cycle of the bus cycle task irrespective of whether they are used or mapped on an input or output channel.

IEC objects: When the EtherNet/IP adapter is defined, the libraries "IoDrvCIFXEIPAdapter.library" and/or RIL_EtherNetIPAdapter.library are automatically applied. The required memory space to implement the EtherNet/IP adapter can thus be defined.

Bus cycle options: By selecting a bus cycle task, the cycle of the mapping exchange for the EtherNet/IP adapter can be connected to a particular task. In this task, it is useful to process the I/O data of the adapter as well.

7.6.5 Tab “Field bus mapping”

Use the "Field bus mapping" tab to access control variables from a parent field bus master.

To access a control variable from a parent master, assign an address to these variables first (mapping). The parent master can use this address to access the control data using acyclic services.

To use the field bus mapping, configure the control as corresponding field bus slave.

For details, refer to Field bus mapping, access from field bus master to variables of the field bus slave, page 23.

7.6.6 Tab “EtherNetIP parameters”

The "EthernetIP parameters" tab displays the communication parameters of the EtherNet/IP adapter.

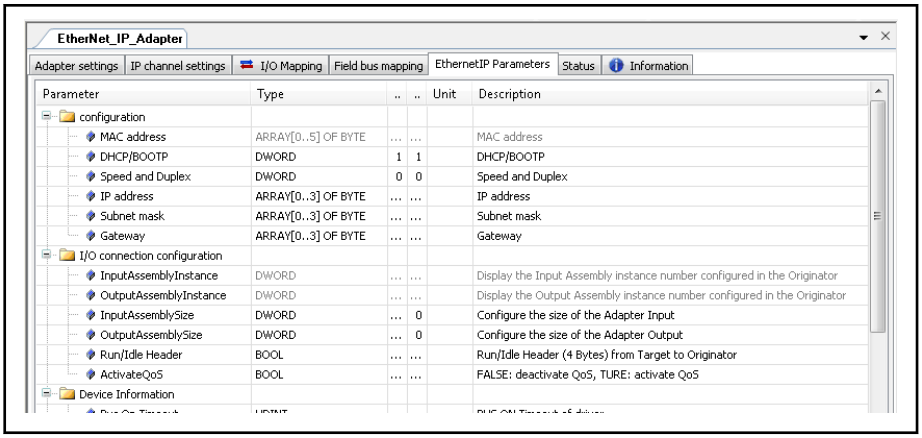


Fig. 7-22: EtherNet/IP adapter: EtherNetIP parameter

Some parameters (in black) can be changed at this location.

This window is used for service purposes and is only visible if the option "Show generic configuration editors" was enabled. Opening via the IndraWorks menu **Tools ▸ Options ▸ IndraLogic 2G ▸ Device editor**

Please contact the service team before making any modifications in the editable parameters.

7.6.7 Tab “Status”

The "Status" tab in the "EtherNet/IP adapter device editor" displays status information (e.g. "Running", "Stopped") and specific diagnostic messages from the device. In addition, the diagnostic messages in the status flags are displayed in "Diag".

7.6.8 Tab “Information”

The tab displays general information from the device description file:

Name, Vendor, Categories, Version, Order number, Description and Image if available.

7.6.9 EtherNet/IP scanner-adapter coupling

MLC controls can be configured as EtherNet/IP scanner as well as adapter. The coupling of two controls based on EtherNet/IP is shown in the following.

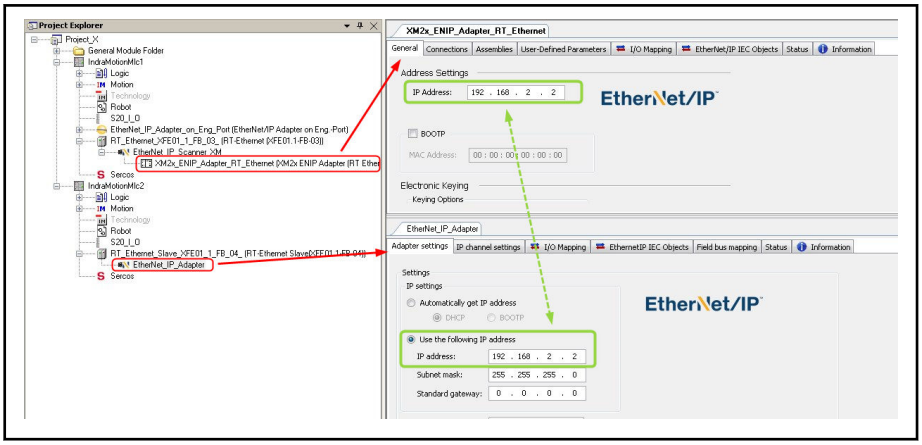


Fig. 7-23: EtherNet/IP coupling IP address

The control configured as EtherNet/IP adapter is added to the EtherNet/IP scanner as adapter device ("XM2" control in this example). When configuring

the adapter device below the scanner , set the same IP configuration as for the control with adapter functionality.

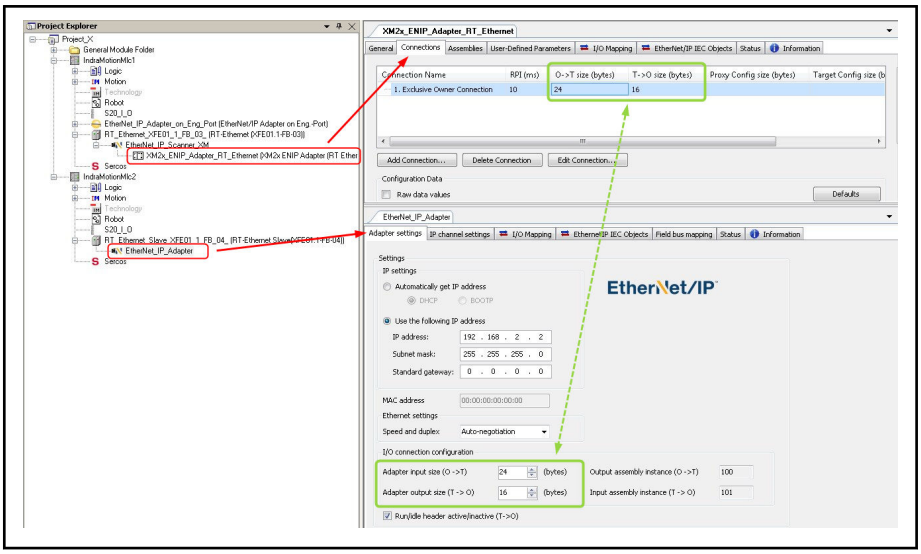


Fig. 7-24: EtherNet/IP coupling O <-> T size

If a control is used as EtherNet/IP adapter, the size of the input and output data can be variably adjusted.

When configuring the connection, identical sizes have to be set in the EtherNet/IP scanner. Thus, the adapter output data is the associated scanner input data (T -> O-size) and the adapter input data is the scanner output data (O -> T-size).

7.7 EtherNet/IP, configuring adapter (engineering port)

7.7.1 Overview

The EtherNet/IP adapter at the engineering port can be used for the following controls:

- MLC system: XM2, XM4, VPx
- MTX system: XM4

If the control is created in the Project Explorer, the respective Ethernet engineering interface available at the control is configured such that it assumes the double function of engineering software and EtherNet/IP adapter interface. The IP address/gateway setting is applied. The required port configuration is made without the user.

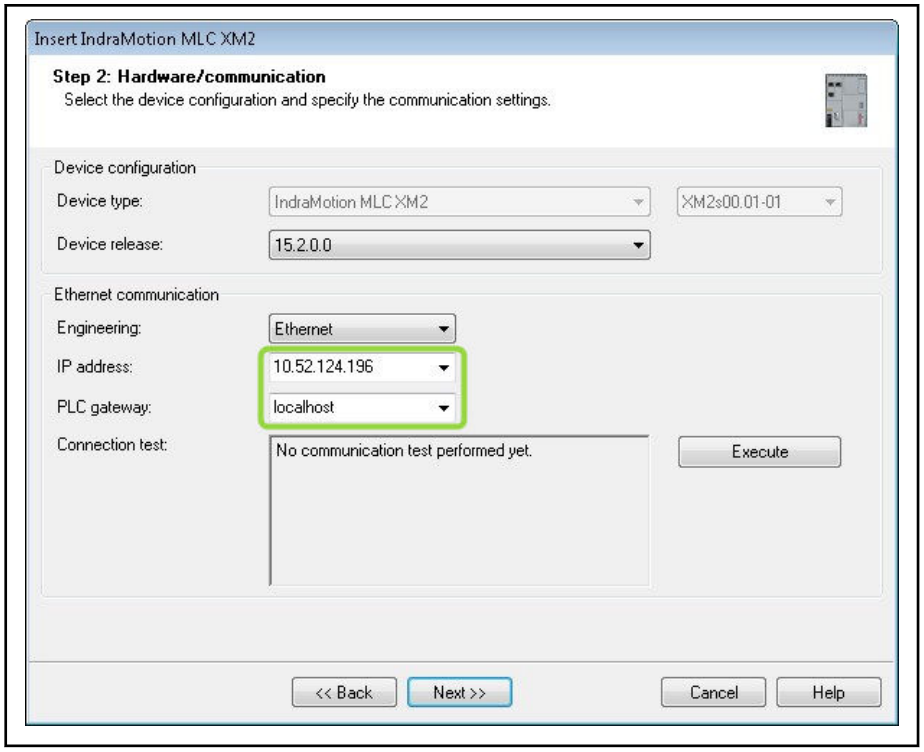


Fig. 7-25: Dialog: Configuration of the control, IP address/PLC gateway

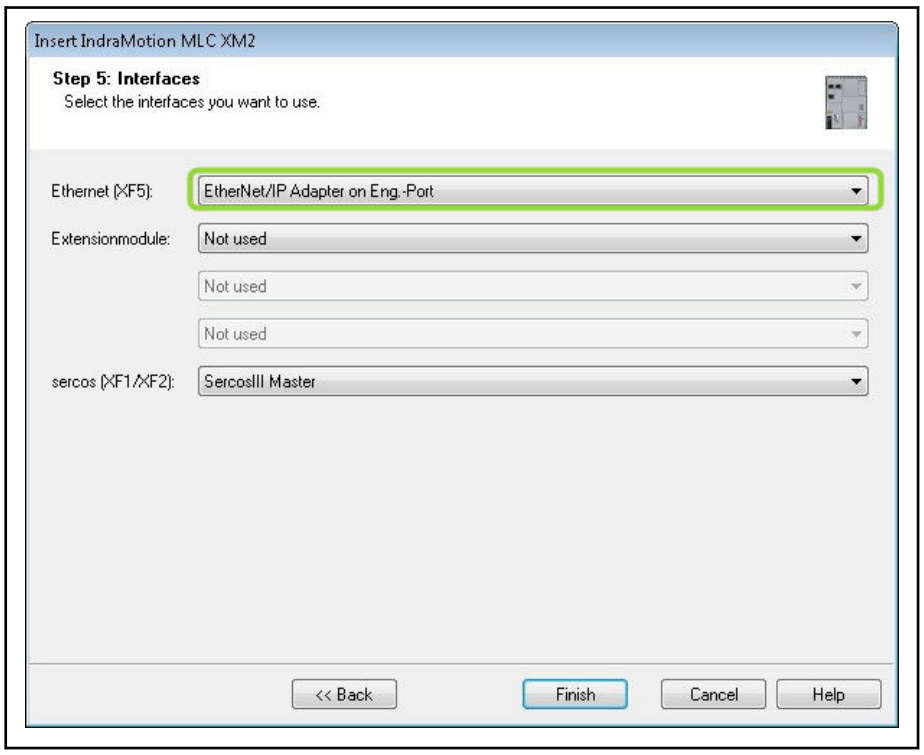


Fig. 7-26: Dialog: Configuration of the control, enabling of the EtherNet/IP adapter (engineering port) functionality

After successfully completing the control configuration, the preconfigured folder is displayed in the Project Explorer.

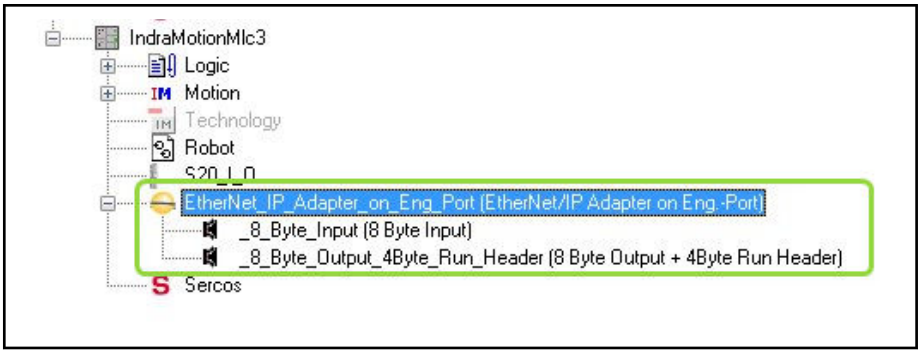


Fig. 7-27: EtherNet/IP adapter (engineering)

The interface between an EtherNet/IP adapter (engineering port) and the EtherNet/IP scanner is executed as common memory space. This space is preconfigured with inputs of 8 bytes and outputs of 8 bytes plus a 4-byte header.

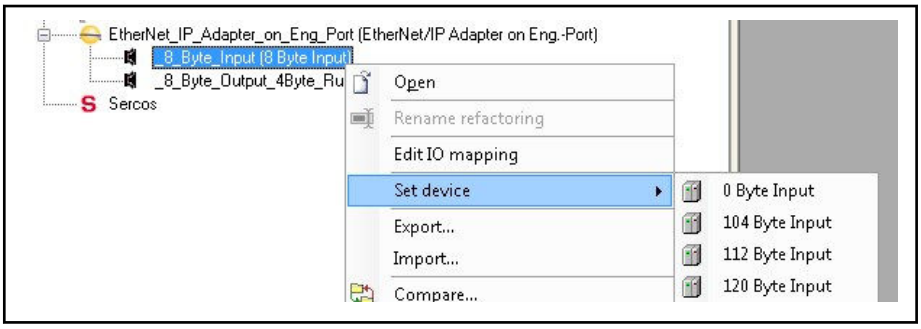


Fig. 7-28: EtherNet/IP adapter (engineering port) , changing coupling area size

1. Inputs on the EtherNet/IP scanner side become outputs on the EtherNet/IP adapter and vice versa.
2. If the input and output areas are set to 0 bytes, cyclic communication is not possible.

EtherNet/IP adapter (engineering port)

- *EtherNet/IP adapter (engineering port), object*
  - Tab “Status”, page 160,
  - Tab “Information”, page 160.
- *EtherNet/IP adapter (engineering port), modules*
  - Tab “I/O mapping”, page 161,
  - Tab “Information”, page 161.

7.7.2 Tab “Status”

The "Status" tab in the "EtherNet/IP adapter device editor" displays status information (e.g. "Running", "Stopped") and specific diagnostic messages from the device. In addition, the diagnostic messages in the status flags are displayed in "Diag".

7.7.3 Tab “Information”

The tab displays general information from the device description file:

Name, Vendor, Categories, Version, Order number, Description and Image if available.

7.7.4 EtherNet/IP adapter (engineering port), modules

Tab “I/O mapping”

The tab is used to assign module inputs and outputs to variables that can be used as local or global variables in the individual POUs.

The current value of the variables is displayed in online mode.

This assignment is described in Mapping of the field bus inputs and outputs, page 19,.

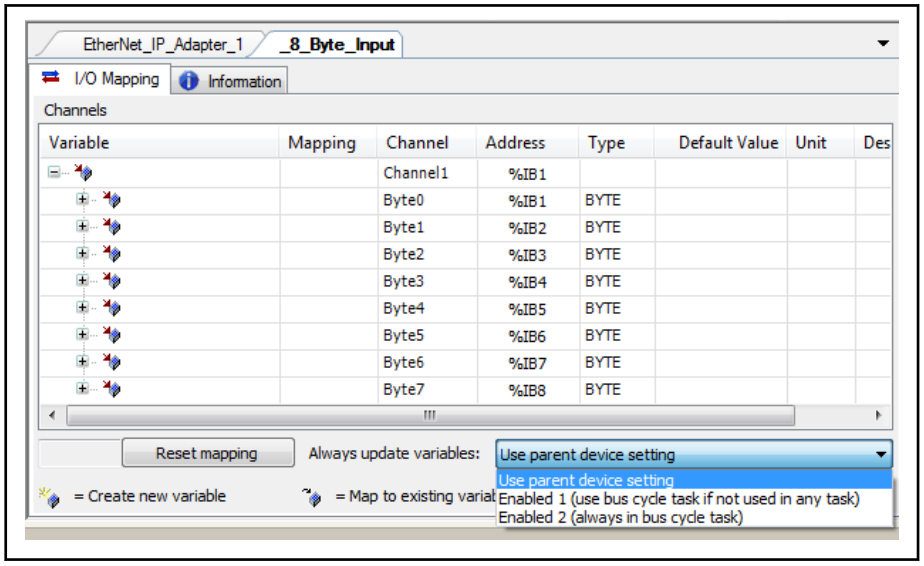


Fig. 7-29: EtherNet/IP adapter (engineering port): I/O mapping

Reset Mapping: Use this button to reset the mapping settings to the default values defined in the device description file.

Update variables: Defines for the module whether I/O variables are updated in the bus cycle task.

The default value is defined in the device description file.

- **Use parent device setting:** Update according to the settings of the superordinate device.
- **Enabled 1 (use bus cycle task if not used in any task):** The I/O variables are updated in the bus cycle task if not used in any other task
- **Enabled 2 (always in the bus cycle task):** All variables are updated in each cycle of the bus cycle task irrespective of whether they are used or mapped on an input or output channel.

Tab “Information”

The tab displays general information from the device description file:

Name, Vendor, Categories, Version, Order number, Description and Image if available.

7.8 EtherNet/IP, FAQs

7.8.1 Frequent problems during EtherNet/IP configuration

If the EtherNet/IP is faulty, an error message is issued in the generic device editor of the EtherNet/IP scanner.

Call:

Double-click on the EtherNet/IP scanner in the Project Explorer and go to the "Status" tab to open the generic device editor.

The following table provides information on possible error causes and troubleshooting measures:

Extended Status Code 0x0114 / 0x0115 / 0x0116 "Vendor ID / Product Code mismatch" "Device Type mismatch" "Revision mismatch"	The device description file (EDS) used to configure the adapter does not match the existing device. For the matching EDS, contact the vendor of the adapter device. The attributes of the existing device can be retrieved for example using the function block IL_ENIPGetAttributeSingle (see library RIL_EtherNetIP): Class = 01 ("Identity Object"), Instance = (generally) 01, attributes: • 1 = Vendor ID • 2 = Device Type • 3 = Product Code • 4 = Major u. Minor Revision
Extended Status Code 0x0127 / 0x0128 "invalid connection size" (O=>T / T=>O settings)	See Notes on the EtherNet/IP connection length (O => T / T => O-size), page 163.
Extended Status Code 0x0204 "unconnected request timed out" or "connection open/close timed out"	1. Check the IP address of the target system settings in the adapter configuration: • Is the IP address entered correctly? • Is the MAC address entered correctly when using BOOTP? 2. Check the IP address in the adapter device (if specified separately or in an own tool for example): • For the Rexroth IndraDrive as EtherNet/IP adapter, the "Engineering over IP" address has to be different than the configured EtherNet/IP IP address for example. • If the IP address was changed in the adapter device, a voltage reset is often required to apply the new IP address. • To test the IP setting in the adapter, a direct Ethernet connection to the PC can be established and a simple "ping" can be executed via the command line. 3. Check the IP addresses in your network: • Check that IP addresses were not assigned twice in the network or do not exist twice. • If switches are used in the network, try a direct connection between scanner and adapter devices. 4. Check the IP address in the scanner settings: • If BOOTP or DHCP is enabled for the scanner IP address, but no BOOTP/DHCP server is used in the network, the static IP address setting has to be used. • If, after loading or beginning with a change in the scanner settings, the error "update configuration failed" is issued in the diagnostic log, proceed as follows: 1. A control "clean-up" 2. Reload the project to the control

Extended Status Code 0x0315 "invalid segment in connection path"	This error can for example occur if O => T / T => O-size is incorrect. Refer to Notes on the EtherNet/IP connection length (O => T / T => O-size), page 163.
Extended Status Code 0x0320 to 0x07FF	These status codes are manufacturer-specific. Thus, for more information, refer to the documentation of the adapter device.
"Driver not found" or **"Module not found"**	An internal error occurred. If "cleaning" and "reloading" the PLC project does not eliminate the error, proceed as follows: 1. A control "clean-up" 2. Reload the project to the control
Diagnostic log "EtherNet/IP scanner diagnostic available"	This logbook entry is generated when an error occurs in the EtherNet/IP connection. Possible causes: • When starting the system, the scanner is faster ready than the adapter. Thus, establishing a connection fails at first (timeout). If all devices are ready for communication, the connections are automatically established. • If an error occurred while establishing a connection, the adapter responded with a CIP error code. Details are displayed in the "Status" dialog of the adapter. • Errors occurred during the cyclic communication. Refer to the following section "I/O performance".
I/O performance and **Error in the cyclic data communication**	With regard to I/O performance and network load, the following settings are recommended: • Generally, use point-2-point (unicast) connections. Multicast only if required. • Set the "Requested Packet Interval" (RPI) only as small (fast) as required. Between 10 and 20 ms is sufficient in most of the cases. • Operate the EtherNet/IP network separately from the other industrial and office networks (use routers if required), as malfunctions and network load can reduce the availability of the EtherNet/IP communication. If errors occur in the cyclic communication, check the following measures: • Refer to recommendations on changing connections to point-2-point and an own or separate EtherNet/IP network. • To reduce the network load, set a higher value for the RPI values of the connections. With regard to cyclic connection aborts, the timeout multiplier can also be increased to improve availability. • Network topology: If multiple adapters are connected in series, adapter devices might not be able to forward all telegrams due to network load. In this case, a star topology using switches is recommended.

Tab. 7-6: Frequent problems during EtherNet/IP configuration

7.8.2 Notes on the EtherNet/IP connection length (O => T / T => O-size)

The currently available EtherNet/IP adapters (bus coupler) can vary strongly in their response and configuration.

There are adapter devices additionally sending status information of different lengths in the data to the scanner (scanner input data). This control data can be either at the beginning or at the end of the scanner input data.

This status data is to be taken into account when configuring a connection, that is T->O-size (scanner input data length) has to be increased by the length of this status data. A connection with invalid data length causes an error while establishing the connection (no connection possible).

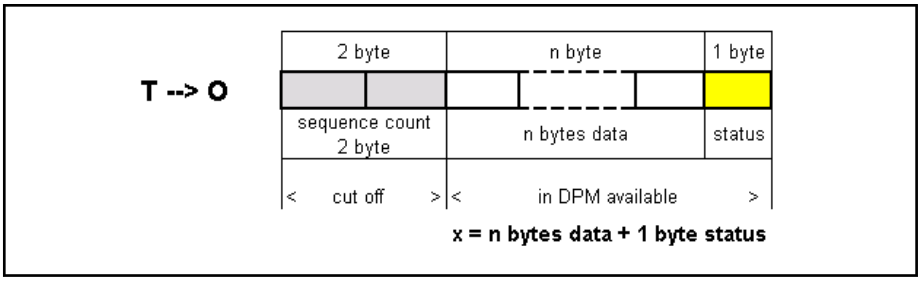


Fig. 7-30: Input data, variant 1

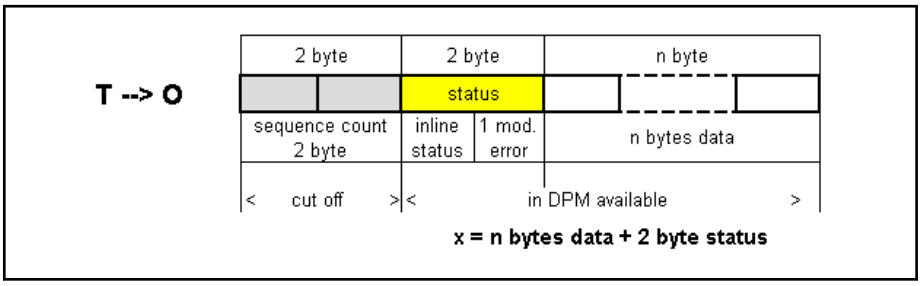


Fig. 7-31: Input data, variant 2

Some adapters force an even number (2, 4, 6, …) of input and/or output data. Otherwise, no connection is accepted.

For modular bus couplers that can be extended using plug-in modules, the individual modules are added to achieve whole bytes.

It is not possible to trigger a 2-bit output module with one byte for example.

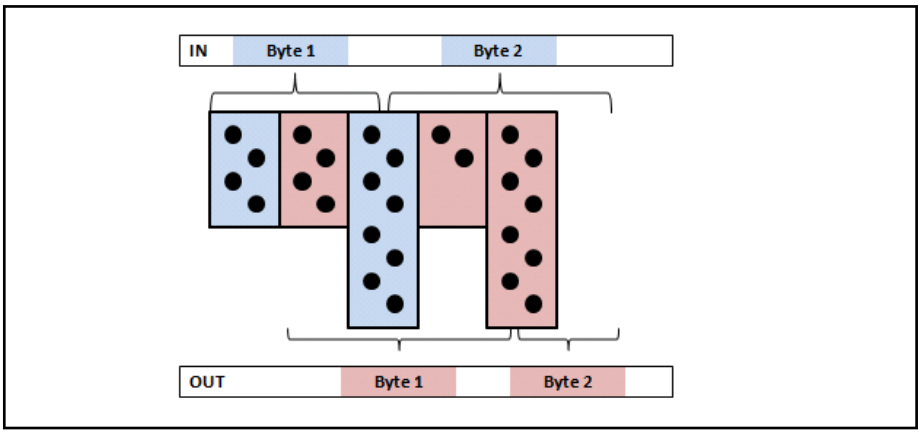


Fig. 7-32: Byte limits for modular bus couplers

If the length for a connection or for the assembly in a connection is now known, this length can also be retrieved. The connection path 20 04 24 03 2C **64** 2C **65** addresses the two assemblies with the instances 0x64 and 0x65 for example.

The size of each assembly can be retrieved via the function block IL_ENIPGetAttributeSingle (refer to the library RIL_EtherNetIP): Class = 4 (assembly object), instance = instance from the connection path, attribute = 4 (size). This retrieved length is then to be entered as O => T or as T => O-size in the connection configuration.

For "older" bus couplers, it might be necessary to describe the connection path "in detail" as the common abbreviated form was not yet supported here, e.g.

20 04 **24 03** 2C 64 2C 65 compared to 20 04 2C 66 2C 69 … .



If required, use the specification or the user documentation of the adapter or contact the manufacturer of the adapter device.

8 EtherCAT

8.1 Features of the EtherCAT master

Function package “License”

The functionality "EtherCAT master" is licensed as **function package**, i.e. a respective license has to be purchased.

Functional scope

The current implementation of the EtherCAT master includes the following functionality:

- Cyclic communication "Free Run"
- Synchronization mode ("Distributed Clocks")
- Import of the device description file (ESI) in the IndraWorks device database
- Offline configuration of the slaves
- Manual configuration of SyncUnits
- Online diagnostics in the IndraWorks interface
- Bus scan and automatic configuration in the IndraWorks project tree
- PLC function blocks (RIL_Ethercat library)
  - Function blocks for state machine
  - Function blocks for acyclic communication
  - Function blocks for diagnostics (e.g. master status, slave status, configuration/online info)

Being prepared:

- Identification procedures "Station Alias", "Explicit Device Identification" and "Data Word"

Target systems

The EtherCAT master can only be implemented with the MLC system:

Control hardware

The following control hardware is supported:

- XM21 and XM22

  (as of IndraWorks 14V20, on the onboard Ethernet interface XF1 alternatively to Sercos, XF2 not used)
- XM42

  (as of IndraWorks 14V22 P1, on the onboard Ethernet interface XF1 alternatively to Sercos, XF2 not used)
- VPx

  (as of IndraWorks 14V22 P1, on the Ethernet interface XF01 alternatively to Sercos, XF02 not used)

Also refer to "Cabling" on page 196.

Technical data

Property/functionality	Value
Max. number of devices (slaves)	128
Max. number of cyclic input data of the master	2048 bytes [1.)]
Max. number of cyclic output data of the master	2048 bytes [1.)]
Max. number of cyclic input data of a slave	2048 bytes [1.)]
Max. number of cyclic output data of a slave	2048 bytes [1.)]

Max. number of data for the acyclic communication	CoE, SoE: 65535 bytes
Max. number of parallel acyclic tasks using function blocks	CoE, SoE: 128
Baud rate	100 MBit/s
Auto negotiation/autocrossing	Yes
Min. cycle time	1 ms
Support of multiple different cycle times	No
Synchronization mode (Distributed Clocks)	Bus Shift (as of 14V22 P1)
Hot Connect groups (flexible topologies)	No
Mailbox CoE (CAN application layer over EtherCAT)	Yes, (as of 14V20)
Mailbox SoE (Sercos drive profile over EtherCAT)	Yes, (as of 14V20)
Mailbox EoE (Ethernet over EtherCAT)	Yes (as of 14V22)
Mailbox FoE (File access over EtherCAT)	No
Mailbox VoE (Vendor specific protocol over EtherCAT)	No
Mailbox AoE (ADS over EtherCAT)	No
Slave-to-slave communication	No
Redundancy	No
Field bus gateways versus third-party vendors with proprietary configuration **(e.g.. EtherCAT slave terminals EL6xxx by Beckhoff)**	No
Disabling slaves or modules **("Maximal project")**	No

1.) The maximum number of process data of the EtherCAT master is 2048 bytes (= Sum of the cyclic input/output data). There is no restriction with regard to input/output data of a slave.

8.2 EtherCAT basics

8.2.1 Terms and abbreviations

EtherCAT	Ethernet for Control Automation Technology. EtherCAT® is a registered trademark and patented technology licensed by Beckhoff Automation GmbH, Germany. EtherCAT is an open, internationally standardized standard and developed further by the EtherCAT Technology Group (ETG).
EC/ECAT **ECM**	Abbreviations for EtherCAT (EC/ECAT) or EtherCAT master (ECM)
ENI	EtherCAT Network Information: Description of the bus configuration as XML file
ESI	EtherCAT Slave Information: Device description file as XML file
ESC	EtherCAT Slave Controller: Communication interface integrated into hardware (ASIC, FPGA) or as software solution
ETG	EtherCAT Technology Group: User organization

EAP	EtherCAT Automation Protocol: Protocol for networking on control level
DC	Distributed Clocks The procedure "Distributed Clocks" is used to synchronize bus devices. The following procedures are distinguished: • Master Shift: The first DC slave at the bus is used as reference clock and as re-adjustment of the timer of the EtherCAT master. • Bus Shift: The EtherCAT master sets the system time and writes it to the first DC slave at the bus (shifting the bus time). • External reference clock (IEEE 1588)
FSoE	FailSafe over EtherCAT: Safety protocol for EtherCAT
HC **HC group**	Hot Connect: Connecting and disconnecting slaves at runtime Hot Connect group: Slaves can be combined to one group. The group can be declared as optional (flexible topologies).
Mailbox **CoE** **SoE** **FoE** **EoE** **VoE** **AoE**	Acyclic communication over EtherCAT: CoE: CAN application protocol over EtherCAT SoE: Servo drive profile over EtherCAT (adapted Sercos II profile for servo drives) FoE: File access over EtherCAT EoE: Ethernet over EtherCAT (e.g. TCP/IP) VoE: Vendor-specific protocol over EtherCAT (vendor-specific mailbox protocols) AoE: Automation Device Specification (ADS) over EtherCAT (device- and field bus-independent interface by Beckhoff).
WC/WKC **WcState**	Abbreviations for WorkingCounter (WC/WKC) or WorkingCounter status (WcState): Validity of command executions or data (WcState value 0 = valid, 1 = invalid). Also refer to "Process data status" in Tab “I/O mapping”, page 224.

8.2.2 Topology

An EtherCAT network consists of a master and of up to 65535 slaves.

Topologies can be implemented arbitrarily, e.g. by using junction terminals.

When cabling, note that slave devices are provided with exactly one defined input port. This is generally the port 0, normally labeled with "IN". Most of the slaves are provided with two ports, i.e. an “IN” port and an “OUT” output port (port 1). In case of a bus coupler with E-BUS as backplane bus, the Ethernet telegrams are routed via this backplane bus (E-BUS), i.e. each module (terminal) is a slave with IN port and OUT port. The E-BUS bus coupler or junction terminals are thus provided with three or four ports (= multiple output ports).

The following figure shows an exemplary topology with different device types:

- Slave **(A)** is a modular bus coupler after the "Modular Device Profile" (e.g. Rexroth S20-EC-BK): The extension modules are connected to a device-specific backplane bus with the bus coupler. The bus coupler is a slave device in the network and contains all I/O data of the modules.
- Slave **(B)** is a modular system with E-BUS (e.g. Beckhoff EK1100): Each module, the bus coupler and possible junction terminals and extension terminals each represent an own slave device at the EtherCAT network.
- Slave **(C)** is a drive. Note: Multi-axis devices generally represent only one slave.

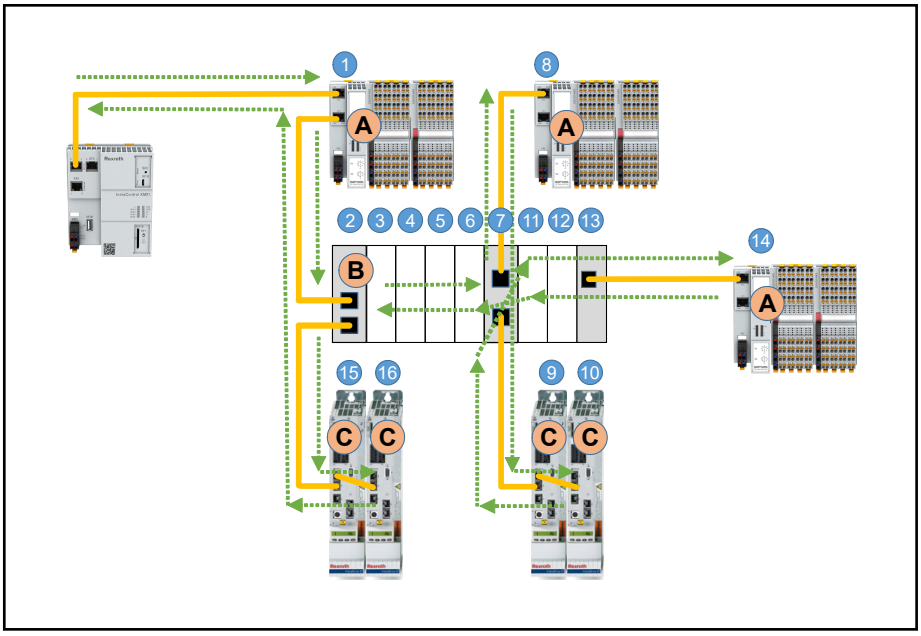


Fig. 8-1: Exemplary topology with different slave device types (A/B/C)

The EtherCAT telegram is sent by the master and runs through all slave devices. Input/output data is only processed in "forward" direction in the EtherCAT slave controllers (ESC). The last slave device sends the telegram back to the master. The telegram in the slaves in then only routed, not processed. From a logical point of view, EtherCAT thus represents a ring - irrespective of the topology. The previous figure shows the numbers from (1) to (16) in the logical sequence of the EtherCAT telegram running through the slave devices. For the sequence of the EtherCAT telegram running through the ports in case of junction terminals or devices with more than two ports, refer to the product documentation of the device.

Cable redundancy

Cable redundancy can only be achieved with a ring topology. The last slave device is connected to the master at its second Ethernet port.

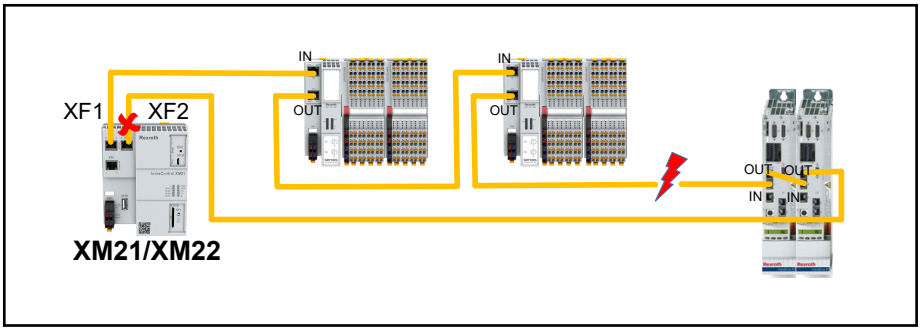


Fig. 8-2: Ring topology for cable redundancy

8.2.3 Addressing and identification procedure

The are different procedures to address and identify a slave device in an EtherCAT network.

Auto increment address

The auto increment address is a topological address, i.e. the slave is addressed using a physical position in the network.

The master mainly uses the auto increment address when initializing the bus communication. The "EtherCAT address" ("fixed physical address") is used for communication purposes as of the state "Pre-Operational".

The auto increment address is "0" for the first slave after the master. The value is decremented for each further slave, i.e. the second slave has the address "-1" (0xFFFF) and the third slave has the address "-2" (0xFFFE).

For the communication, each slave increments the auto increment address in the EtherCAT telegram. The slave reading the value "0" in the telegram is addressed correspondingly.

EtherCAT address

The "EtherCAT address" is also called "Fixed Physical Address" in the ETG specification. Synonyms of other vendors are for example "Slave Address" or "Station Address".

Upon the bus initialization, the EtherCAT address is not saved in the slave devices, but it is assigned to each slave by the master. The assignment which EtherCAT address obtains a certain slave device takes place in the EtherCAT configuration either via the auto increment address or - for Hot Connect groups (flexible topologies) via an identification value.

The EtherCAT address "0" is reserved for the master. Slaves can have an address from "1" to "65535" (16 bits). EtherCAT addresses for slaves are often used starting from the value "1001".

Identification value

For Hot Connect groups (flexible topologies) or to exclude wiring errors, it is required to clearly identify slave devices irrespective of their topological position.

To always assign a specified EtherCAT address to a certain slave device, there are three different identification procedures. Which identification procedures are supported by a slave device (no support, one or more procedures are supported) is described in its device description or product documentation. The identification value is for example configured via the DIP switch or rotary switch at the slave device.

Identification procedure:

- (Configured) **Station alias**: The station alias is also often called "Second Slave Address" (SSA) or "Second Station Address". The value can be retrieved from the ESC register 0x0012. It is generally configured via the EtherCAT configuration tool to which the identification value is written to the slave EEPROM. The EtherCAT slave controller then applies this value to its register upon restart.
- **Explicit Device Identification**: The slave informs on its identification value in the AL status code register (0x0134) upon master request. In the ETG.1020, this procedure is also called "Requesting ID".
- **Data Word (/ Input Word)**: The identification value is located in a specified memory range of the EtherCAT slave (e.g. process data area from ADO 0x1000). For the address offset (ADO), refer to the device description (attribute "IdentificationAdo") or set it if required. In the ETG.1020, this procedure is also called "Direct ID".

Example

In the following example, the first two slaves (EtherCAT addresses 1001 and 1005) do not have any identification value, i.e. the EtherCAT address is only specified using the topology position (AutoInc address). If the physical order of these two devices is exchanged (identical device type), e.g. wiring error or intended device change, the EtherCAT address is still assigned to the respective position

In case of the following two slaves (EtherCAT addresses 1042 and 1043), an identification procedure is configured, i.e. only if the identification values (here 2 and 3) match the configured topology position (AutoInc address), the slave are accepted in the cyclic communication by the master. Thus, there will be no wiring error between these two slaves. For an intended cross exchange or device change (spare device), the identification value has to be set accordingly.

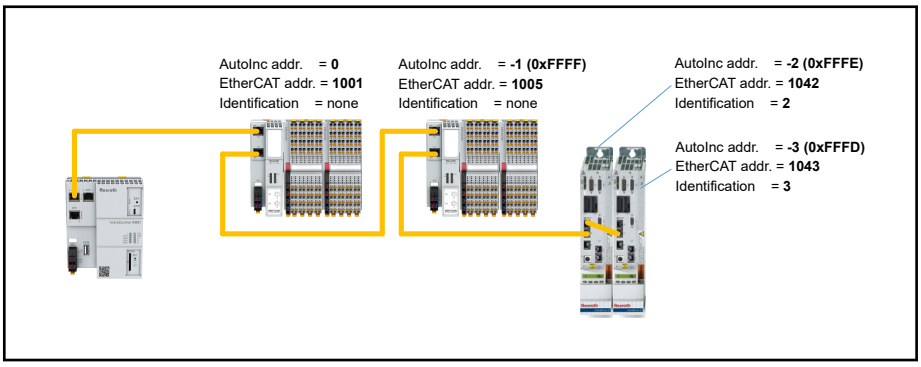


Fig. 8-3: Example of an EtherCAT addressing

8.2.4 State machine

There are the following communication states for the EtherCAT:

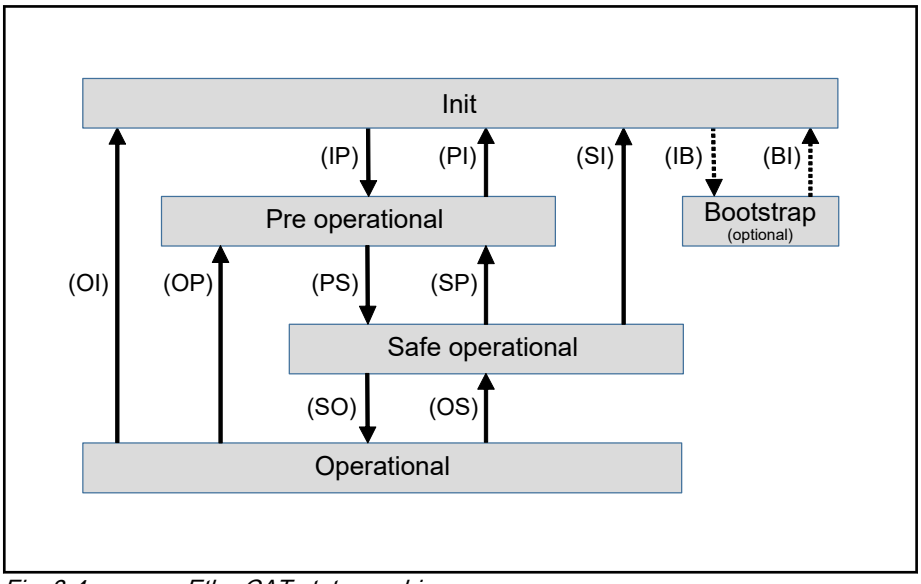


Fig. 8-4: EtherCAT state machine

State	Description
Init (I)	• No communication with application layer • Transition to “PreOP” (IP): Master configures slave register, e.g. address register and Sync manager for mailbox communication
Pre-Operational (P)	• The mailbox communication on the application layer is possible as of “PreOP” • Not yet any process data communication • Transition to “SafeOP” (PS): Master also configures process data mapping (FMMU) and Sync manager for process data communication

Safe-Operational (S)	• Process data communication. Only input data is analyzed. Output data remains in the "safe" state • DC synchronization is adjusted
Operational (O)	• Process data communication: Input and output data is valid • DC synchronization
Bootstrap (B)	• Optional state in EtherCAT slave devices (recommended for firmware update) • The state “Bootstrap” can only be reached from “Init” • No process data communication • Mailbox communication via FoE (FoE only possible in “Bootstrap” => device-specific)

Additional information:

- An EtherCAT slave device can be switched to a "lower" state than the master bus state, e.g. if the EtherCAT is in the bus state “OP”, a slave can be switched individually to “PreOP”.
- Only a master can initiate a change in state. Otherwise, a slave can only “downshift” its state in case of error.

8.2.5 Acyclic communication (mailbox)

For EtherCAT, acyclic communication is also called mailbox communication and only possible starting from the "Pre-Operational" state. There are different protocols that can be tunneled via EtherCAT. For information on which protocols are supported by a slave, refer to the documentation (or device description) of the slave device.

Overview on mailbox protocols

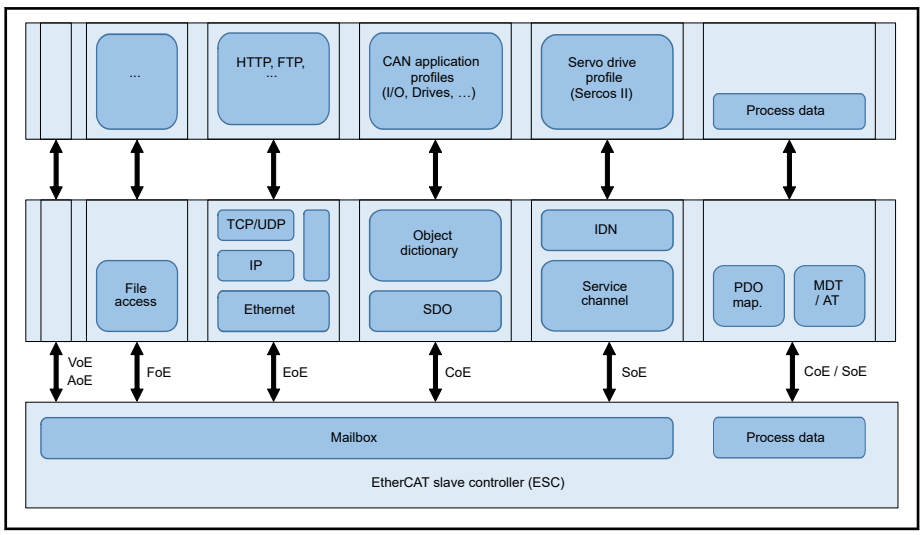


Fig. 8-5: Overview on mailbox protocols

CoE: CAN application protocol over EtherCAT

CoE objects are also called "Process Data Objects" (PDO) or "Service Data Objects" (SDO). An object is addressed with "Index" and "Subindex", e.g. 0x1018:01. If an object has more than one subindex, the number of subindices is written to the subindex "0".

The object directory of a slave as well as the object descriptions and element descriptions of the subindices can be retrieved using the optional CoE service "SDO Information".

Also refer to chapter "CoE SDO abort codes" on page 181.

SoE: Servo drive profile over EtherCAT

SoE is an adapted Sercos II profile for Servo drives, i.e. some Sercos II parameters do not have any or not the same meaning for SoE. One parameter is addressed with a 16-bit identification number (IDN: Parameter type S/P, parameter set, data block number).

In contrast to Sercos, multi-axis devices represent only one slave device for SoE. The subordinate drives differ by a drive number ("DriveNo": 0 / 1 / ... / 7), alternatively also called channel ("Channel": A / B / ... / H).

Also refer to chapter "SoE error codes" on page 182.

The following IDNs are irrelevant for SoE:	
S-0-0003	Minimum AT transmit starting time (T1min)
S-0-0004	Transmit/receive transition time (TATMT)
S-0-0005	Minimum feedback acquisition time (T4min)
S-0-0009	Position of data record in MDT
S-0-0010	Length of master data telegram
S-0-0088	Receive to receive recovery time (TMTSY)
S-0-0090	Command value proceeding time (TMTSG)
S-0-0127	C0100 Communication phase 3 transition check It is performed at the EtherCAT state transition PreOP => SafeOP. If there is an error, the cause can be determined using S-0-0021.
S-0-0128	C5200 Communication phase 4 transition check It is performed at the EtherCAT state transition SafeOP => OP. If there is an error, the cause can be determined using S-0-0022.

The following IDNs have an adjusted meaning at SoE:	
S-0-0006	AT Transmission starting time (T1) T1 specifies the time shift from the EtherCAT Sync signal to the point in time when the application should provide new AT data in the ESC memory.
S-0-0014	Interface status This parameter is intended to display the DL status, AL status and the AL status code of the EtherCAT.
S-0-0028	MST error counter Counter for missing datagrams of cyclic data and RX error counter and lost link counter of the EtherCAT.
S-0-0089	MDT Transmit starting time (T2) T2 specifies the time shift from the EtherCAT Sync signal to the point in time when new data is available to the master in the ESC memory.

FoE: File access over EtherCAT

FoE is a protocol, analog to TFTP, to transfer files. Normally, a firmware update is used for FoE. Thus, FoE is only possible in the EtherCAT state “Bootstrap” (device-specific).

Also refer to chapter "FoE error codes" on page 184.

EoE: Ethernet over EtherCAT

EoE can be used to tunnel the standard Ethernet configuration (e.g. TCP/IP) via EtherCAT. A "virtual Ethernet switch" is required in the EtherCAT master.

An IP-capable device (e.g. laptop) cannot be connected to any location in the EtherCAT network. A "switch port" is required (e.g. special terminal with "switch port" ⇔ is not an EtherCAT port).

Alternatively, an "IP port" can also be configured for an EtherCAT slave device (if the device supports EoE/IP port, e.g. integrated WebServer).

In both cases, the EtherCAT slave requires a "virtual MAC address" specified in the EtherCAT network configuration. The EoE settings are then written from the master to the slave devices upon bus startup (transition Init => PreOP). EoE is only possible starting from the PreOP state as in case of all mailbox protocols.

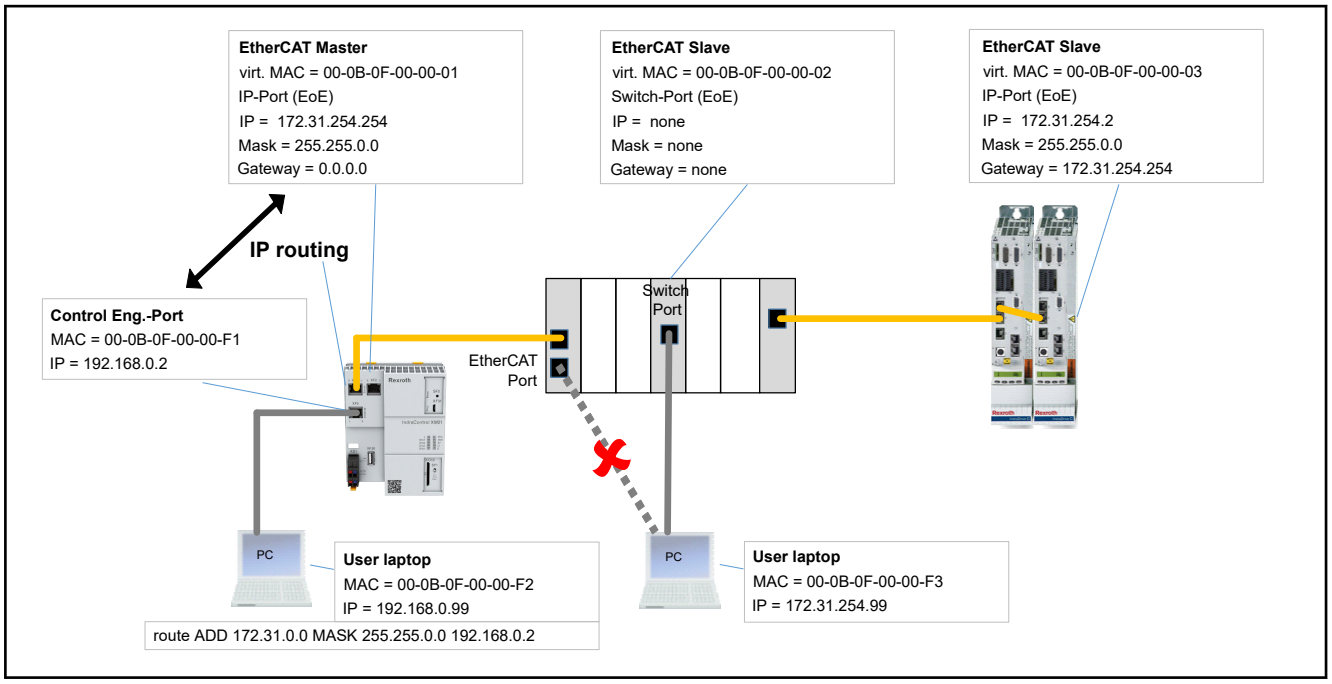


Fig. 8-6: EoE configuration example

VoE: Vendor-specific protocol over EtherCAT

VoE is used to transmit vendor-specific protocols.

AoE: ADS over EtherCAT

AoE is a device- and field bus-independent interface by Beckhoff (ADS: Automation Device Specification).

Slaves without mailbox communication

EtherCAT slave device do not have to support a mailbox protocol. Simple digital input/output terminals do for example not support any mailbox protocol. For these slave devices, only the registers supported by EtherCAT slave can be read and written. This has to be supported by every EtherCAT slave controller (ESC).

8.2.6 Telegram structure

EtherCAT uses standard Ethernet frames (IEEE 802.3) with an own EtherType 0x88A4. Ethernet telegrams of another type are not supported by an EtherCAT slave controller and have to be tunneled by EtherCAT (e.g. EoE).

The EtherCAT frame consists of multiple subtelegrams or datagrams with different commands. Each SyncUnit (see chapter 8.2.8 "SyncUnits" on page 176) is for example an own datagram and an acyclic communication (mailbox) or other commands are own datagrams as well.

Up to 15 datagrams can be combined into one EtherCAT frame. Multiple frames are required if there are more than 15 datagrams.

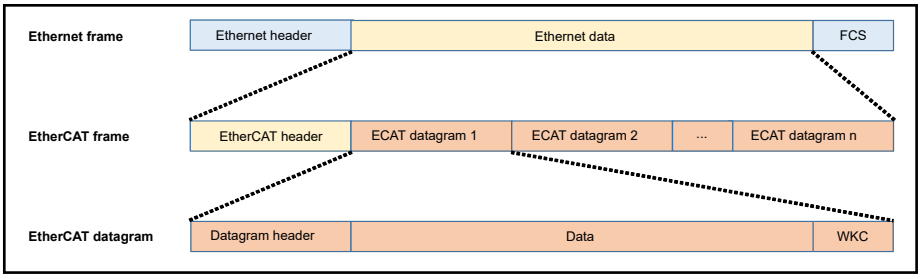


Fig. 8-7: EtherCAT telegram structure

8.2.7 WorkingCounter

The checksum of the Ethernet frame (FCS : Frame Check Sequence) ensures the correct data transfer to the bus.

To check the correct command execution in the slave devices, each datagram has a **WorkingCounter** (WKC). The WKC is sent by the master with 0. Each addressed slave increments the WKC with regard from the command. After the frame ran through all slaves and reached the master again, the master can compare the expected value (command value) of the WKC to the actual value and thus check the correct command processing.

For the validity of cyclic process data, the status of the WorkingCounter check is considered (refer to "PdStatus", chapter 8.5.12 "Tab “I/O mapping”" on page 224).

8.2.8 SyncUnits

SyncUnit

A SyncUnit describes a logical grouping of cyclic data that should be exchanged consistently and synchronously. In the easiest case, all EtherCAT slaves are combined to a SyncUnit, but there can also be individual SyncUnits for each slave.

Generally, optional slaves (e.g. slaves at runtime that "may fail") or machine modules may be grouped in an own SyncUnit. Thus, these devices or this SyncUnit is provided with an own datagram and thus with an own WorkingCounter. A missing device for example results only in an invalid WorkingCounter of this SyncUnit. The data status of all other SyncUnits is still valid.

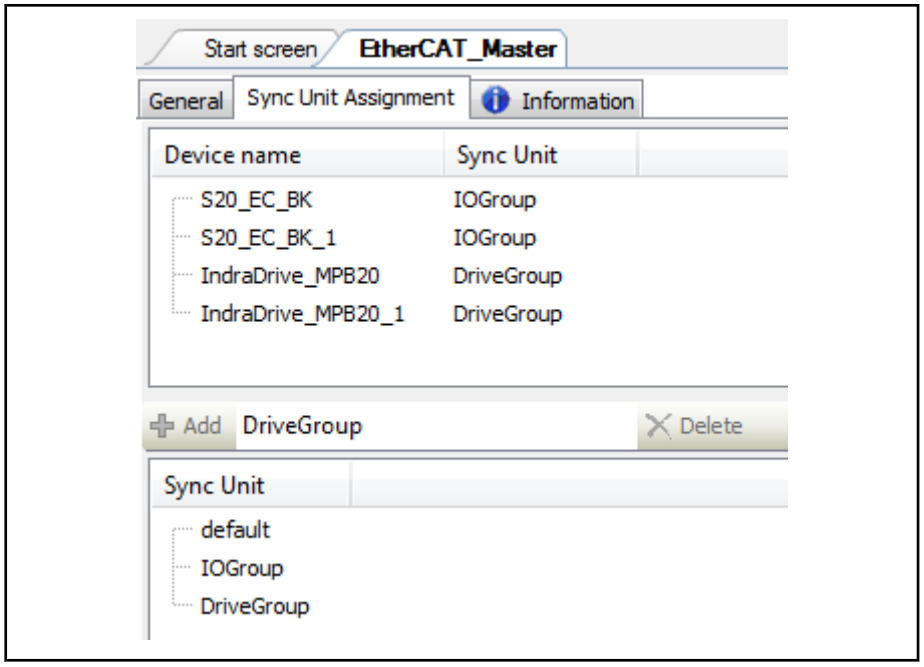


Fig. 8-8: Example of a SyncUnit configuration

Different EtherCAT cycle times

Different EtherCAT cycle times can also be executed with SyncUnits. The SyncUnits (= datagrams) are then sent in own frames with different cycle times. Fast I/O components and drive groups are for example grouped into individual SyncUnits with different cycle times.

☞ The Rexroth EtherCAT master does not support different EtherCAT cycle times!

8.2.9 Hot Connect

Hot Connect

Hot Connect (HC) generally means that slaves can be removed and connected at runtime (bus state "Operational"). This also includes use cases with multiple missing devices or machine modules or uses cases that already allow the commissioning of a part of the system.

HC group

The configuration of HC groups automatically groups the slaves into an own SyncUnit. An HC group can be declared as "optional". Thus, the EtherCAT master does not detect a missing HC group as error (flexible topologies). The first slave of an HC group is detected by an identification procedure (refer to chapter 8.2.3 "Addressing and identification procedure" on page 170). It can also be specified for an HC group whether this group may only be connected at a specific topology position (port of the predecessor EtherCAT device) or at any position.

☞ The Rexroth EtherCAT master does not support the configuration of HC groups!

8.2.10 Diagnostics

General

EtherCAT diagnostics are available on different communication layers.

A cycle-accurate diagnostics is generated in the master. It comprises the (master) bus state and the analysis of the WorkingCounter (WKC). By analyzing the WKC, there is a cycle-accurate detection whether a datagram (e.g.

cyclic process data) was processed correctly and whether the data status is valid. All slaves in a datagram (e.g. SyncUnit) addressed with a faulty WKC do thus have an invalid data status (refer to "PdStatus", chapter 8.5.12 "Tab “I/O mapping”" on page 224).

A detailed diagnostics whether a slave has an EtherCAT state error for example or whether it is not connected anymore or whether there is a link problem (port status) is then acyclically determined by the master.

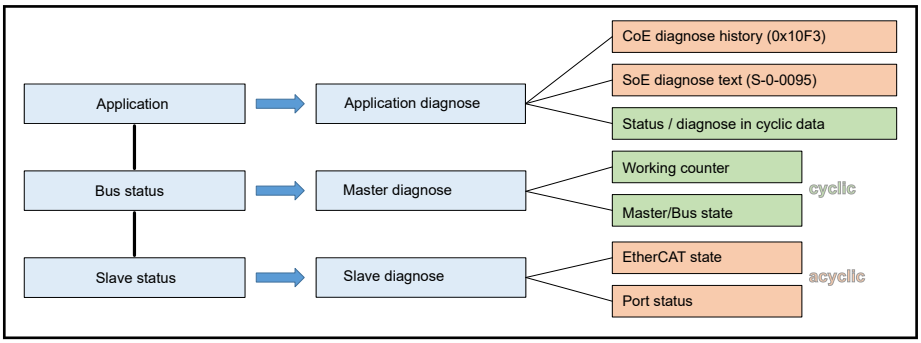


Fig. 8-9: Overview on EtherCAT diagnostics

On application level, many slaves are already provided with status messages or simple diagnostic data in the cyclic data. Detailed diagnostics can also be retrieved acyclically, e.g. there is a diagnostic history object (0x10F3) for CoE slaves or a diagnostic text (S-0-0095) for SoE slaves.

To analyze the diagnostics in the PLC program, the Rexroth EtherCAT master provides different function blocks in the RIL_Ethercat library.

Some CoE slaves also support the CoE emergency messages. These emergency messages mostly contain only device-specific information. Acknowledging them is not required. Thus, only these are generally entered into the control log.

AL status codes

The "AL status code" (AL: Application Layer) describes the error code of a slave device if a requested change in state cannot be executed or if a slave detects an internal error.

In case of error, the Rexroth EtherCAT master automatically retrieves the AL status code (ESC register 0x0134:0x0135) and acknowledges the error indication in the slave. Upon the next change in slave state, the read AL status code is reset in the master.



If there is an unspecified error or a configuration is detected as invalid, refer to the product documentation of the slave device or contact the vendor of the slave device.

The following abbreviations are used in the following table:

- I = State "Init"
- B = State "Bootstrap"
- P = State "Pre-Operational"
- S = State "Safe-Operational"
- O = State "Operational"

AL status code	Description	Current state or change in state	Resulting state
0x0000	No error	All	Current state
0x0001	Unspecified error	All	All
0x0002	No memory	All	All
0x0003	Invalid device configuration	P → S	P
0x0004	Invalid revision	P → S	P
0x0005	Reserved due to compatibility reasons	-	-
0x0006	SII/EEPROM data does not match the firmware	I → P	I
0x0007	Firmware update not successful	All	All
0x000E	License errors	All	I
0x0011	Invalid change in state requested See chapter 8.2.4 "State machine" on page 172	I → S, I → O, P → O, P → B, S → B, O → B	Current state
0x0012	Unknown state requested	All	Current state
0x0013	“Bootstrap” state not supported	I → B	I
0x0014	No valid firmware	I → P	I
0x0015	Invalid mailbox configuration (Bootstrap)	I → B	I
0x0016	Invalid mailbox configuration (PreOP)	I → P	I
0x0017	Invalid Sync manger configuration	P → S, S → O	Current state
0x0018	No valid inputs available	O, S → O	S
0x0019	No valid outputs possible	O, S → O	S
0x001A	Synchronization error (different causes)	O, S → O	S
0x001B	Sync manager watchdog	O, S	S
0x001C	Invalid Sync manger types	O, S, P → S	S
0x001D	Invalid Sync manger configuration (outputs)	O, S, P → S	S
0x001E	Invalid Sync manger configuration (inputs)	O, S, P → S	P
0x001F	Invalid watchdog configuration	O, S, P → S	P
0x0020	Slave requires cold start	All	Current state
0x0021	Slave requires the state "Init"	B, P, S. O	Current state
0x0022	Slave requires the state "Pre-operational"	S, O	S, O
0x0023	Slave requires the state "Safe-operational"	O	O
0x0024	Invalid mapping input data	P → S	P
0x0025	Invalid mapping output data	P → S	P
0x0026	Inconsistent settings (general)	P → S	P
0x0027	“Free Run” not supported	P → S	P
0x0028	Sync mode not supported	P → S	P

AL status code	Description	Current state or change in state	Resulting state
0x0029	“Free Run” requires “Sync Manager” in 3 buffer mode	P → S	P
0x002A	Background watchdog	S, O	P
0x002B	No valid input and output data	O, S → O	S
0x002C	Fatal sync error Sync0 or Sync1 signal not received anymore.	O	S
0x002D	No sync signal Slave waits for Sync0/Sync1 signal (S → O timeout) in "Safe-Operational".	S → O	S
0x002E	Cycle time too small The set EtherCAT cycle time is not supported by the slave.	S → O	S
0x0030	Invalid DC sync configuration	O, S → O, P → S	S, P
0x0031	Invalid DC latch configuration	O, S → O, P → S	S, P
0x0032	PLL errors, Master is not synchronized, but at least one DC event is received.	O, S → O	S
0x0033	DC sync I/O error Multiple synchronization errors; master and slave are not synchronized anymore.	O, S → O	S
0x0034	DC sync timeout error Multiple synchronization error, too many SM events omitted	O, S → O	S
0x0035	Invalid DC cycle time	P → S	P
0x0036	DC Sync0 cycle time does not run with application	P → S	P
0x0037	DC Sync1 cycle time does not run with application	P → S	P
0x0041	Mailbox AoE	B, P, S, O	Current state
0x0042	Mailbox EoE	B, P, S, O	Current state
0x0043	Mailbox CoE	B, P, S, O	Current state
0x0044	Mailbox FoE	B, P, S, O	Current state
0x0045	Mailbox SoE	B, P, S, O	Current state
0x004F	Mailbox VoE	B, P, S, O	Current state
0x0050	EEPROM no access	All	All
0x0051	EEPROM errors	All	All
0x0052	External hardware not ready for operation	All	All
0x0060	Slave executed restart	All	I

AL status code	Description	Current state or change in state	Resulting state
0x0061	Device identification value updated	P	P
0x0070	Invalid module configuration For a slave with modular device profile, the configured modules (0xF030) do not match the physical modules (0xF050).	P → S	P
0x00F0	Application control available	I	I
< 0x8000	Reserved	-	-
0x8000 - 0xFFFF	Vendor-specific	-	-

Also refer to:

- chapter 8.8.4 "S20-EC-BK: Bus coupler does not switch to OP (notes on the AL status codes)" on page 231

CoE SDO abort codes

For an acyclic CoE mailbox communication, the following standardized SDO abort codes can result. These are mapped on the error codes of the control (→ ECM error code).

SDO abort code	Description	ECM error code
-	Ecat CoE: Protocol not supported	F0ED0032
0x0503 0000	Ecat SDO: Toggle bit not alternated	F0ED0040
0x0504 0000	Ecat SDO: SDO protocol timeout	F0ED0041
0x0504 0001	Ecat SDO: Client/server command specifier not valid	F0ED0042
0x0504 0002	Ecat SDO: Invalid block size (block mode only)	F0ED0043
0x0504 0003	Ecat SDO: Invalid sequence number (block mode only)	F0ED0044
0x0504 0004	Ecat SDO: CRC error (block mode only)	F0ED0045
0x0504 0005	Ecat SDO: Out of memory	F0ED0046
0x0601 0000	Ecat SDO: Unsupported access to an object	F0ED0047
0x0601 0001	Ecat SDO: Attempt to read a write only object	F0ED0048
0x0601 0002	Ecat SDO: Attempt to write a read only object	F0ED0049
0x0601 0003	Ecat SDO: Subindex cannot be written, SI0 must be 0	F0EE0004
0x0601 0004	Ecat SDO: Complete access not supported	F0EE0005
0x0601 0005	Ecat SDO: Object length exceeds mailbox size	F0EE0006
0x0601 0006	Ecat SDO: Object mapped to RxPDO, SDO Download blocked	F0EE0007
0x0602 0000	Ecat SDO: Object does not exist in object dictionary	F0ED004A
0x0604 0041	Ecat SDO: Object cannot be mapped to PDO	F0ED004B
0x0604 0042	Ecat SDO: PDO length exceeded	F0ED004C
0x0604 0043	Ecat SDO: General incompatibility (parameter)	F0ED004D

SDO abort code	Description	ECM error code
0x0604 0047	Ecat SDO: General incompatibility (device internal)	F0ED004E
0x0606 0000	Ecat SDO: Access failed (hardware error)	F0ED004F
0x0607 0010	Ecat SDO: Data type and length code do not match	F0ED0050
0x0607 0012	Ecat SDO: Wrong data type (parameter too long)	F0ED0051
0x0607 0013	Ecat SDO: Wrong data type (parameter too short)	F0ED0052
0x0609 0011	Ecat SDO: Subindex does not exist	F0ED0053
0x0609 0030	Ecat SDO: Parameter value out of range	F0ED0054
0x0609 0031	Ecat SDO: Parameter value out of high limit	F0ED0055
0x0609 0032	Ecat SDO: Parameter value out of low limit	F0ED0056
0x0609 0033	Ecat SDO: Module Ident List mismatch Invalid module configuration for a slave with modular device profile (MDP), the configured modules (0xF030) do not match the physical modules (0xF050).	F0ED005E
0x0609 0036	Ecat SDO: Maximum value less than minimum value	F0ED0057
0x0800 0000	Ecat SDO: General error	F0ED0058
0x0800 0020	Ecat SDO: Data cannot be transferred or stored	F0ED0059
0x0800 0021	Ecat SDO: Data cannot be transferred (local control)	F0ED005A
0x0800 0022	Ecat SDO: Data cannot be transferred (device state)	F0ED005B
0x0800 0023	Ecat SDO: Object dictionary does not exist	F0ED005C
-	Ecat SDO: Unknown error code	F0ED005D

SoE error codes

For acyclic SoE mailbox communication, the following standardized SoE error codes can result. These are mapped on the error codes of the control (→ ECM error code).

Sercos error code	Description	ECM error code
-	Ecat SoE: Protocol not supported	F0ED0035
-	Ecat SoE: Invalid access to element 0	F0ED0078
0x1001	Ecat SoE: Does not exist	F0ED0079
0x1009	Ecat SoE: Invalid access to element 1	F0ED007A
0x2001	Ecat SoE: No name	F0ED007B
0x2002	Ecat SoE: Name transmission too short	F0ED007C
0x2003	Ecat SoE: Name transmission too long	F0ED007D
0x2004	Ecat SoE: Name cannot be changed (read only)	F0ED007E
0x2005	Ecat SoE: Name is write-protected at this time	F0ED007F
0x3002	Ecat SoE: Attribute transmission too short	F0ED0080
0x3003	Ecat SoE: Attribute transmission too long Transmission error, attribute too long	F0ED0081

Sercos error code	Description	ECM error code
0x3004	Attribute cannot be changed (read-only)	F0ED0082
0x3005	Attribute is currently write-protected	F0ED0083
0x4001	No unit	F0ED0084
0x4002	Transmission error, unit too short	F0ED0085
0x4003	Transmission error, unit too long	F0ED0086
0x4004	Unit cannot be changed (read-only)	F0ED0087
0x4005	Unit is currently write-protected	F0ED0088
0x5001	No minimum value	F0ED0089
0x5002	Transmission error, minimum value too short	F0ED008A
0x5003	Transmission error, minimum value too long	F0ED008B
0x5004	Minimum value cannot be changed (read-only)	F0ED008C
0x5005	Minimum value is currently write-protected	F0ED008D
0x6001	No maximum value	F0ED008E
0x6002	Transmission error, maximum value too short	F0ED008F
0x6003	Transmission error, maximum value too long	F0ED0090
0x6004	Maximum value cannot be changed (read-only)	F0ED0091
0x6005	Maximum value is currently write-protected	F0ED0092
0x7001	No operating data	F0ED0093
0x7002	Transmission error, operating data too short	F0ED0094
0x7003	Transmission error, operating data too long	F0ED0095
0x7004	Operating data cannot be changed (read-only)	F0ED0096
0x7005	Operating data is currently write-protected	F0ED0097
0x7006	Operating data is smaller than minimum value	F0ED0098
0x7007	Operating data is greater than maximum value	F0ED0099
0x7008	Invalid operating data	F0ED009A
0x7009	Operation data is write-protected (password)	F0ED009B
0x700A	Operating data is write-protected (cycl. conf.)	F0ED009C
0x700B	Invalid addressing (data container, list)	F0ED009D
0x700C	Operating data is write-protected (other)	F0ED009E
0x7010	Command sequence already active	F0ED009F
0x7011	Command sequence cannot be interrupted	F0ED0100
0x7012	Command sequence can currently not be executed	F0ED0101
0x7013	Command sequence cannot be executed, parameter invalid	F0ED0102
-	Wrong drive number in response	F0ED0103
-	Wrong IDN in response	F0ED0104
-	Fragment lost	F0ED0105

Sercos error code	Description	ECM error code
-	Receiving buffer full	F0ED0106
-	No data status	F0ED0107
-	No default value	F0ED0108
-	Transmission error, default value too long	F0ED0109
-	Default value cannot be changed	F0ED010A
-	Invalid drive number	F0ED010B
-	General error	F0ED010C
-	No element addressed	F0ED010D

FoE error codes

For acyclic FoE mailbox communication, the following standardized FoE error codes can result. These are mapped on the error codes of the control (→ ECM error code).

FoE error code	Description	ECM error code
-	Mailbox FoE not supported	F0ED0034
-	Mailbox FoE not supported in boot strap	F0ED010F
0x8000	Manufacturer-specific FoE error (not defined)	F0ED0060
0x8001	Not found	F0ED0061
0x8002	Access denied	F0ED0062
0x8003	Disk full	F0ED0063
0x8004	Not permitted	F0ED0064
0x8005	Wrong package number	F0ED0065
0x8006	Already available	F0ED0066
0x8007	User does not exist	F0ED0067
0x8008	Only in bootstrap	F0ED0068
0x8009	File name invalid in boot strap	F0ED0069
0x800A	No access rights	F0ED006A
0x800B	Program error	F0ED006B
0x800C	Wrong checksum	F0ED006C
0x800D	Firmware does not match hardware	F0ED006D
0x800E	Reserved	-
0x800F	File not found	F0ED006F
0x8010	File header missing	F0EE0001
0x8011	Flash memory error	F0EE0002
0x8012	File incompatible	F0EE0003
-	Data is greater than permitted	F0ED017A

8.2.11 Synchronization

General

An individual synchronization mode can be set for each EtherCAT slave. For the modes supported by a slave device, refer to the documentation or device description of the slave device. The following synchronization modes are distinguished:

- Free Run (no synchronization)
- Synchronization with SM event (SM: SyncManager [1])
- Synchronization with DC sync (DC: "Distributed Clocks")

The following example shows the DC configuration with the Rexroth IndraDrive MPx20 SoE. The configurator shows the predefined configurations from the device description (ESI). Apart from the "Free Run" mode, three DC modes are provided with different "Sync0Shift" (time shift in the slave to the Sync0 event).

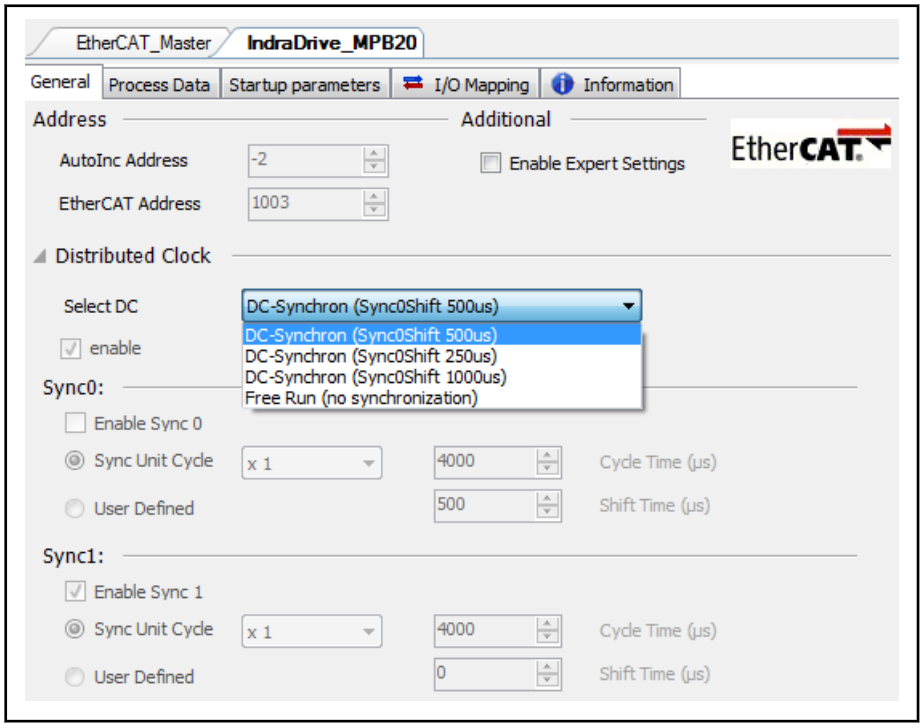


Fig. 8-10: DC configuration with Rexroth IndraDrive MPx20 SoE

Free run

The EtherCAT communication cycle and the input/output data processing run independently in the slave device. I.e. the local application cycle in the slave device can be slower or faster than the EtherCAT bus cycle.

☞ The following figures show the copying and processing times in the slave devices using the "In"/"Out" bars.

[1] *The main task of the SyncManager (SM) is to ensure data consistency in the DPRAM of the ESC. There can be up to 16 independent SyncManager channels. Four are used in the standard configuration: SM0 = Mailbox Out, SM1 = Mailbox In, SM2 = Process data Out, SM3 = Process data In. Additionally, an SM event can be used for synchronization after receiving a telegram in the slave.*

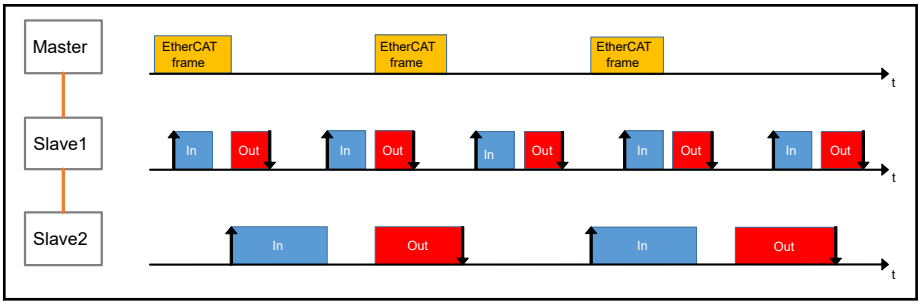


Fig. 8-11: "Free Run" mode

For CoE, the following parameters of the SyncManager object 0x1C3m are relevant (with the standard configuration "SM2/3". The following applies to process data: 0x1C32 for outputs and 0x1C33 for inputs):

SI	Description	Access	Usage	Explanation/default value
1	Synchronization type	R/RW	M	Synchronization type: 0x00: Free Run
2	Cycle time	R/RW	O	Local cycle time of the application controller in ns
4	Synchronization types Supported	R	M	Supported synchronization types: Bit 0 = 1: Free Run supported
5	Minimum cycle time	R	C	Minimum cycle time: Maximum runtime of the local application cycle (required if SI=2 can be written)

SI = Subindex, R = Read-only, RW = Read/Write, M = Mandatory , O = Optional, C = Conditional

Synchronizing with SM event

This synchronization is often called "SM2/3" event, as the SyncManagers normally use 0/1 for mailbox out/in and the SyncManagers use 2/3 for process data out/in.

The output data processing in the slave is triggered by receiving the EtherCAT telegram by the SyncManager. When output data becomes valid depends on the jitter at the sending point in time of the master, on the duration of the telegram during transfer and on the processing times in the slaves. To apply input data, an "Input Shift" can optionally be configured (per device) to have the most "current" input data possible upon the next SM event (next telegram run).

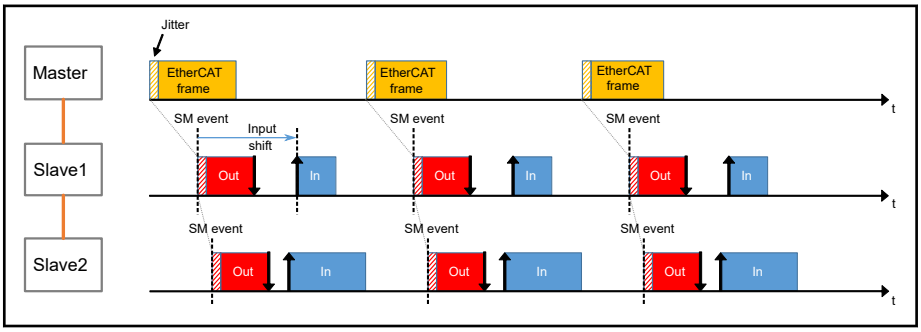


Fig. 8-12: Mode "Synchronization with SM event"

For CoE, the following parameters of the SyncManager object 0x1C3m are relevant (with the standard configuration "SM2/3". The following applies to process data: 0x1C32 for outputs and 0x1C33 for inputs):

SI	Description	Access	Usage	Explanation/default value
1	Synchronization type	RW	M	Synchronization type: Outputs (0x1C32): 0x01: Synchronous with SM2 event Inputs (0x1C33): 0x01: Synchronous with SM3 event (if no outputs are available) 0x22: Synchronous with SM2 event (if outputs are available)
2	Cycle time	R/RW	O	EtherCAT communication cycle in ns (Time between two SM events)
3	Shift time	RW	-/M	Time shift, input data application (Shift Input Latch) (only inputs / 0x1C33) in ns
4	Synchronization types Supported	R	M	Supported synchronization types: Bit 1 = 1: Synchronous SM supported
5	Minimum cycle time	R	M	Minimum cycle time supported by the slave (Maximum time of the local application cycle in ns)
8	Get cycle time	RW	C	Used in the SM and DC mode for a variable cycle time: Bit 0: Local time measurement started Bit 1: Reset of error counter
11	SM-event missed	R	O	This error counter is incremented if the SM event is not received within the expected time (consequently, data cannot be copied anymore)
12	Cycle time too small	R	M	This error counter is incremented if the cycle time is too short to completely execute the local application cycle (input data cannot be prepared until next SM event)
14	RxPDO toggle failed	R	O	This error counter is incremented if the slave supports the RxPDO toggle and if no change in status was received anymore by the master
32	Sync error	R	C	Synchronization error: Should be supported if SI = 11 is available. Can be mapped in TxPDO. Value 0: No Sync error or Sync not supported Value 1: Sync error

SI = Subindex, R = Read-only, RW = Read/Write, M = Mandatory , O = Optional, C = Conditional

Synchronization with DC sync

For the synchronization procedure "Distributed Clocks" (DC), there are multiple options to specify the reference clock:

- Master Shift: The first DC slave at the bus is used as reference clock and as re-adjustment of the timer of the EtherCAT master.
- Bus Shift: The EtherCAT master sets the system time and writes it to the first DC slave at the bus (shifting the bus time).
- External reference clock (IEEE 1588)

Based on the reference clock, a global system time is adjusted in all devices. Use this system time to generate corresponding "DC Sync" events in the EtherCAT slave controllers to process input/output data.

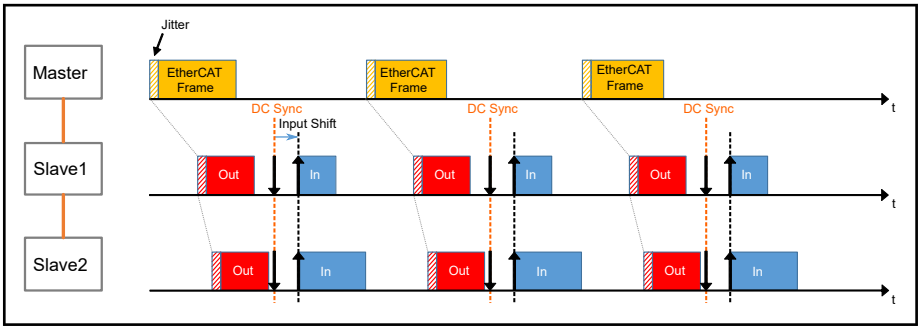


Fig. 8-13: Mode "Synchronization with DC sync" (simplified representation)

In addition to the Sync event "Sync0", another subordinate Sync event "Sync1" is generally supported (functionality of a standard EtherCAT slave controller (ESC)). Which Sync modes are supported by a slave device should be included in its device description file (ESI) or in the product documentation. The following DC modes are distinguished:

1. Synchronous with Sync0 event
2. Synchronous with Sync0 event, input shift with Sync1 event
3. Synchronous with Sync0 event, output shift with Sync1 event
4. Synchronous with Sync1 event
5. DC with subordinate cycles (Sync1 event only every nth Sync0 cycle)

The point in time of the Sync event is globally set in the EtherCAT master as "Sync Shift Time" (or “Sync Offset“) related to the master application cycle. One Sync0 and Sync1 cycle time can be set individually for each slave. This Sync cycle time is a multiple of the EtherCAT communication cycle. Additionally, a shift ("Shift Time") related to the global Sync point in time is possible (refer to fig. 8-14 "Slave configuration of the DC settings" on page 189).



The DC settings are normally predefined configurations from the device description (ESI) of a slave and should only be changed by experts!

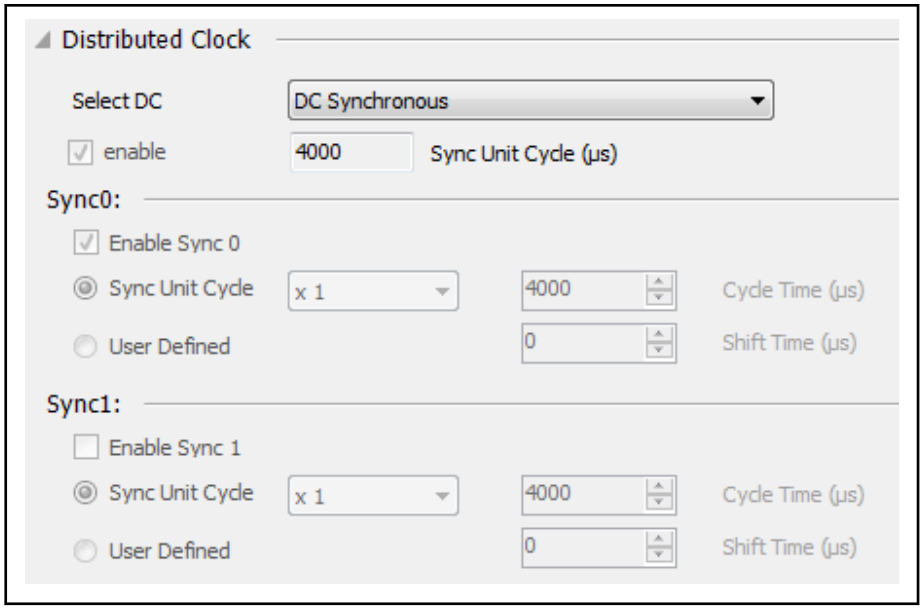


Fig. 8-14: Slave configuration of the DC settings

For CoE, the following parameters of the SyncManager object 0x1C3m are relevant (with the standard configuration "SM2/3". The following applies to process data: 0x1C32 for outputs and 0x1C33 for inputs):

SI	Description	Access	Usage	Explanation/default value
1	Synchronization type	RW	M	Synchronization type: 0x02: DC synchronous with Sync0 event 0x03: DC Synchronous with Sync1 event
2	Cycle time	R	O	Cycle time Sync0 event, time between two Sync0 events in ns (register 0x09A3:0x09A0) For subordinate cycles: Time between two Sync1 events in ns (register 0x09A7:0x09A4) The cycle times Sync0/1 and, where applicable, the shift time of Sync0/1 are specified in the configurator.
3	Shift time	RW	M	Time shift in ns: Outputs: Time between Sync0 event and "Outputs valid" (initialized with 0x1C32:09) Inputs: Time between Sync0 event and "Input Latch" (initialized with 0x1C33:05 – 0x1C33:06)

4	Synchronization types Supported	R	M	Supported synchronization types: Bit 4-2: Supported DC modes 001 = DC Sync0 010 = DC Sync1 100 = Subordinate cycle Bits 5-6: Shift settings 00 = Output shift not supported 01 = Input/output shift with local timer 10 = Input/output shift with Sync1
5	Minimum cycle time	R	M	Minimum cycle time supported by the slave (Maximum time of the local application cycle in ns)
6	Calc and copy time	R	M	Calculation and copying time in ns: Minimum time between SM2 event and Sync0 event. For Sync1: Minimum time between Sync0 and setting of outputs
8	Get cycle time	RW	C	Is used in the SM and DC mode for a variable cycle time Bit 0: Local time measurement started Bit 1: Reset of error counter
9	Delay time	R	M	Hardware delay time in ns: Time between trigger (Sync0 or Sync1 event) and validity of data in the process (e.g. electric signals)
11	SM-event missed	R	O	This error counter is incremented if the SM event is not received within the expected time (consequently, data cannot be copied anymore)
12	Cycle time too small	R	M	This error counter is incremented if the cycle time is too short to completely execute the local application cycle (input data cannot be prepared until next SM event)
13	Shift time too short	R	O	This error counter is incremented if the SM event was received too late so that the Sync0 event is received before "Calc+Copy Time" is completed (master telegram too late; configuration problem)
14	RxPDO toggle failed	R	O	This error counter is incremented if the slave supports the RxPDO toggle and if no change in status was received anymore by the master.
15	Minimum cycle distance	R	O	Minimum distance between two SM events in ns (to monitor jitter)
16	Maximum cycle distance	R	O	Maximum distance between two SM events in ns (to monitor jitter)

17	Minimum SM Sync distance	R	O	Minimum distance between SM event and Sync0 event in ns (to monitor jitter)
18	Maximum SM Sync distance	R	O	Maximum distance between SM event and Sync0 event in ns (to monitor jitter)
32	Sync error	R	C	Synchronization error: Should be supported if SI = 11 or SI = 13 is available. Can be mapped in TxPDO. Value 0: No Sync error or Sync not supported Value 1: Sync error

SI = Subindex, R = Read-only, RW = Read/Write, M = Mandatory , O = Optional, C = Conditional

The following boundary conditions apply to the DC mode:

(1) Synchronous with Sync0 event

Optional time shifts of "Outputs valid" and "Input Latch" by Shift Time (0x1C3m:03).

Synchronization type:

- Outputs (0x1C32:01) = 2 (Sync0 event)
- Inputs (0x1C33:01) = 2 (Sync0 event)

Cycle time:

- Sync0 cycle time (0x1C3m:02) **>** Min. cycle time (0x1C3m:05)

Shift time:

- Output Shift Time (0x1C32:03) **>** Output Delay Time (0x1C32:09)
- Input Shift Time (0x1C33:03) **>** [Min Cycle time (0x1C32:05) – Input Calc and Copy Time (0x1C33:06)]
- Input Shift Time (0x1C33:03) **<** [Sync0 Cycle Time (0x1C33:02) – Input Calc and Copy Time (0x1C33:06)]

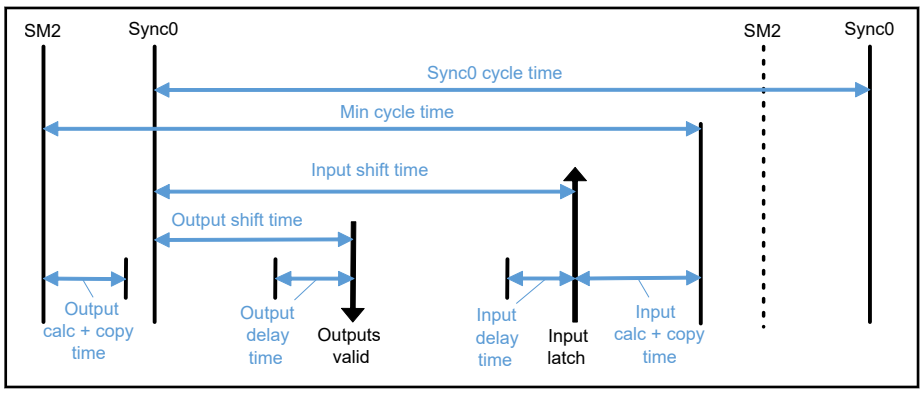


Fig. 8-15: Boundary conditions (1) Synchronous with Sync0 event

(2) Synchronous with Sync0 event, Input Shift with Sync1 event

Time shift from "Input Latch" with Sync1 event (in contrast to "Shift Time", the "Start Latch" point in time is described, i.e. without "Input Delay Time").

Synchronization type:

- Outputs (0x1C32:01) = 2 (Sync0 event)
- Inputs (0x1C33:01) = 3 (Sync1 event)

Cycle time:

- Sync0 cycle time (0x1C3m:02) **>** Min. cycle time (0x1C3m:05)

Shift time:

- Shift Sync0 <> Sync1 (config.) >= [Min Cycle Time (0x1C3m:05) – Output Calc and Copy Time (0x1C32:06) – Input Calc and Copy Time (0x1C33:06) – Input Delay Time (0x1C33:09)]

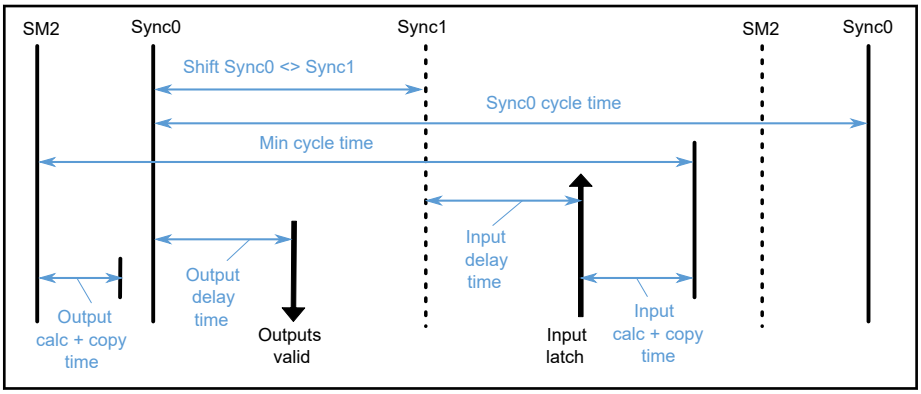


Fig. 8-16: Boundary conditions (2) Synchronous with Sync0 event, Input Shift with Sync1 event

(3) Synchronous with Sync0 event, output shift with Sync1 event

Time shift from "Outputs valid" with Sync1 event (in contrast to "Shift Time", the "Start Outputs" point in time is described, i.e. without "Output Delay Time").

Synchronization type:

- Outputs (0x1C32:01) = 3 (Sync1 event)
- Inputs (0x1C33:01) = 2 (Sync0 event)

For more detailed information on the configuration "Output shift with Sync1 event", contact the ETG.

(4) Synchronous with Sync1 event

Time shifts of "Outputs valid" and "Input Latch" by Sync1 event.

Synchronization type:

- Outputs (0x1C32:01) = 3 (Sync1 event)
- Inputs (0x1C33:01) = 3 (Sync1 event)

Cycle time:

- Sync0 cycle time (0x1C3m:02) > Min. cycle time (0x1C3m:05)

Shift time:

- Shift Sync0 <> Sync1 (Config.) >= Output Calc and Copy Time (0x1C32:06)

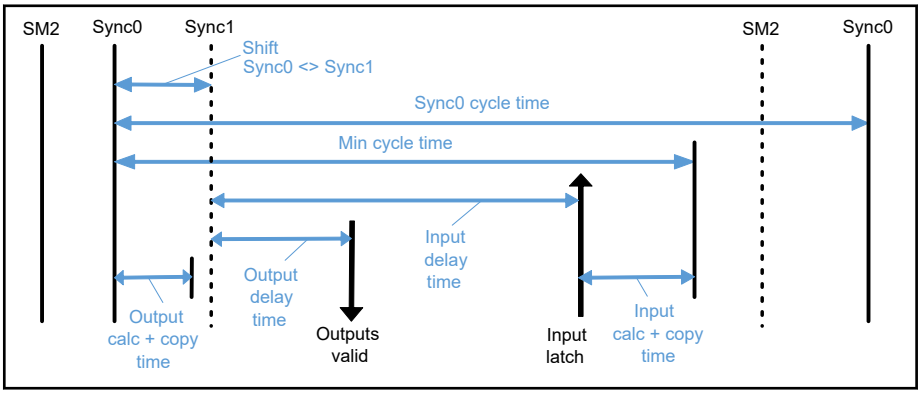


Fig. 8-17: Boundary conditions (4) Synchronous with Sync1 event

(5) DC with subordinate cycles

The Sync0 event is used as trigger for the local application cycle (e.g. for fast controls) while the Sync1 event is used for a (slower) communication cycle. I.e. the Sync1 cycle time is a multiple of the Sync0 cycle time (config.).

Synchronization type:

- Outputs (0x1C32:01) = 3 (Sync1 event)
- Inputs (0x1C33:01) = 3 (Sync1 event)

Cycle time:

- Cycle time (0x1C3m:02) is Sync1 cycle time

Shift time:

- The time shifts of "Outputs valid" and "Input Latch" can only be specified by "Input / Output Shift Time" (0x1C3m:03).

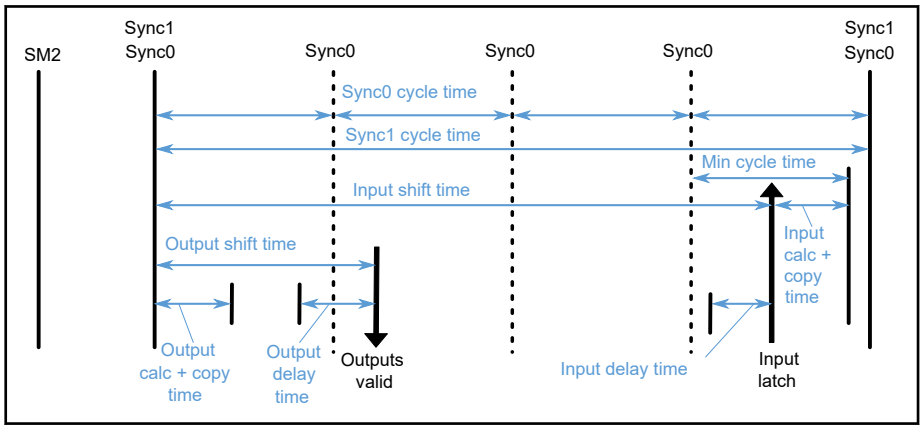


Fig. 8-18: Boundary conditions (5) DC with subordinate cycles

8.2.12 EtherCAT slave controller

Structure

The following figure shows a schematic diagram of an EtherCAT slave controller (ESC).

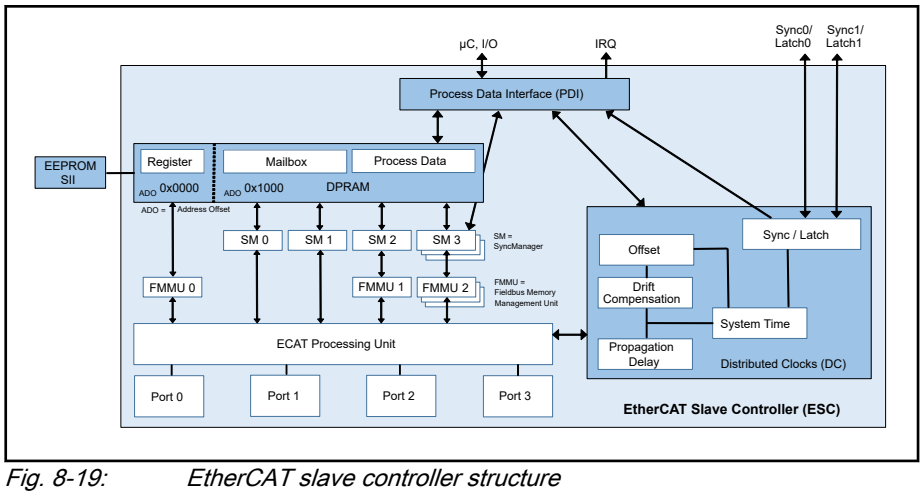


Fig. 8-19: EtherCAT slave controller structure

Register

The registers are in the local address range from 0x0000 to 0x0FFF of the ESC and are also used

- to configure the ESC, e.g. EtherCAT address (“fixed physical address“), SyncManager, FMMU, DC,
- to access the EEPROM/SII (Slave information Interface). It also includes the device identification and the default values for the ESC configuration,

- for control and status data of the EtherCAT state machine.

SyncManager (SM)

The SyncManagers ensure data consistency in the DPRAM of the ESC, i.e. simultaneous access to the memory ranges is inhibited, e.g. if the ECAT processing unit (telegram processing unit) and the local application try to access the memory range in the DPRAM at the same time.

SM configuration from the register ADO 0x0800.

There can be up to 16 independent SyncManager channels:

- Normally, a slave without mailbox has two SyncManager:
  - SM0 = Process data outputs (or inputs if there are no outputs),
  - SM1 = Process data inputs.
- Normally, a slave with mailbox has four SyncManager:
  - SM0 = Mailbox-Out,
  - SM1 = Mailbox-In,
  - SM2 = Process data outputs (or inputs if there are no outputs),
  - SM3 = Process data inputs.
- Process data of one direction (inputs or outputs) can also be divided into multiple SyncManagers; e.g. if the data consistency of individual memory range is to be monitored separately. This is also called "SyncUnits" in the slave.

It is distinguished between different types:

- Mailbox (acyclic communication): 1 buffer system with handshake mechanism.
- Process data: 1- or 3-buffer system. With three buffers, there can be independent read and write accesses, e.g. for the Free Run communication if the local application and the ECAT telegram processing run independently from each other.

Fieldbus Memory Management Unit (FMMU)

Process data is assigned to a logic address space on the EtherCAT bus (up to 4 GB). The address space can be split on any number of slaves. The FMMU is responsible to map a global logical address to the local physical address of the ESC.

The FMMU configuration takes place from the register ADO 0x0600.

There can be up to 16 independent FMMU channels, e.g. separate FMMUs for data direction and memory ranges.

Distributed Clocks (DC)

The procedure "Distributed Clocks" with a specified global reference clock is used to synchronize EtherCAT devices.

The system time starting at 1/1/2000 at 00:00 o'clock (unit 1 ns/64 bits) is controlled in all devices, i.e. the drift and the offset of the local ESC clock is respectively compensated to the reference clock (accuracy < 100ns).

The DC configuration takes place from the register ADO 0x0900.

The DC unit generates respective events (Sync/Latch) for input/output data processing. The second Sync signal (Sync1) can be configured using a defined offset towards the Sync0 event. The latch signals are for example used for the "Time Stamping" of the system time.

Commands

There are different EtherCAT commands for all addressing procedures. Apart from separate read and write commands, there are also combined com-

mands executing read and write accesses with a datagram. These combined accesses are more efficient, but not supported by all slave devices.

Value	Code	Name	Description
0	NOP	No Operation	Slave ignores command.
1	APRD	Auto Increment Read	Slave addressing with auto increment address. The slave receiving address 0 processes data. Each slave increments its address by 1.
2	APWR	Auto Increment Write	
3	APRW	Auto Increment Read Write	
4	FPRD	Fixed Physical Read	Slave addressing with configured EtherCAT address (“fixed physical address”).
5	FPWR	Fixed Physical Write	
6	FPRW	Fixed Physical Read Write	
7	BRD	Broadcast Read	All slaves execute an OR operation of the receiving datagram data with the addressed memory range and write the result to the same location in the datagram.
8	BWR	Broadcast Write	Data is written to all slaves.
9	BRW	Broadcast Read Write	All slaves execute an OR operation of the receiving datagram data with the addressed memory range and write the result to the same location in the datagram and to the memory. (This command is normally not used).
10	LRD	Logical Memory Read	Slave addressing with logic memory address, i.e. all slaves whose FMMUs are configured with the address of the logic memory mapping from the datagram, processes data.
11	LWR	Logical Memory Write	
12	LRW	Logical Memory Read Write	
13	ARMW	Auto Increment Read Multiple Write	Slave addressing with auto increment address. The slave receiving address 0 adds input data to the datagram (“read”), all other slaves extract output data at the same location of the datagram (“Multiple Write”). Each slave increments its address by 1.
14	FRMW	Fixed Physical Read Multiple Write	The slave with the configured EtherCAT address (fixed physical address) adds input data to the datagram (“read”), all other slaves extract output data at the same location of the datagram (“Multiple Write”).
15-255	-	-	Reserved.

Depending on the command type, the WorkingCounter of the datagram is incremented:

Command type	Result	Incremention
Read	Reading not successful (or slave not addressed)	No
	Reading successful	+ 1

Command type	Result	Incremention
Write	Writing not successful (or slave not addressed)	No
	Writing successful	+ 1
Read + write	Reading and writing not successful (or slave not addressed)	No
	Reading successful	+ 1
	Writing successful	+ 2
	Reading and writing successful	+ 3

8.3 EtherCAT configuration overview

Cabling

The Rexroth EtherCAT master does not support cable redundancy. Only one Ethernet port can be used for the EtherCAT cabling at the EtherCAT master:

Only the port XF1 can be used for the XM2 controls (port XF2 is not possible).



Fig. 8-20: EtherCAT cabling at an XM2 control

Only the port XF1 can be used for the XM4 control (port XF2 is not possible).

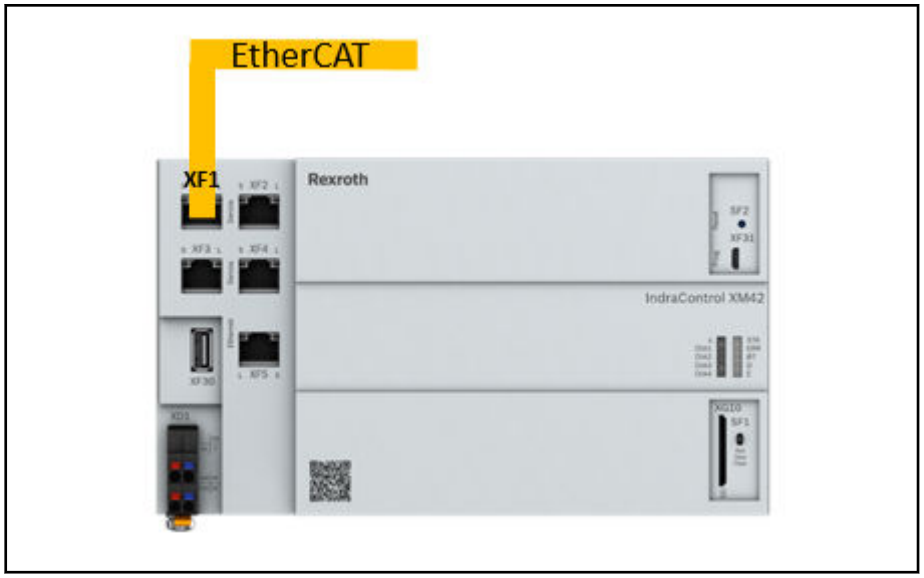


Fig. 8-21: EtherCAT cabling at an XM42 control

Only the port XF01 can be used for the VPx control (port XF02 is not possible).

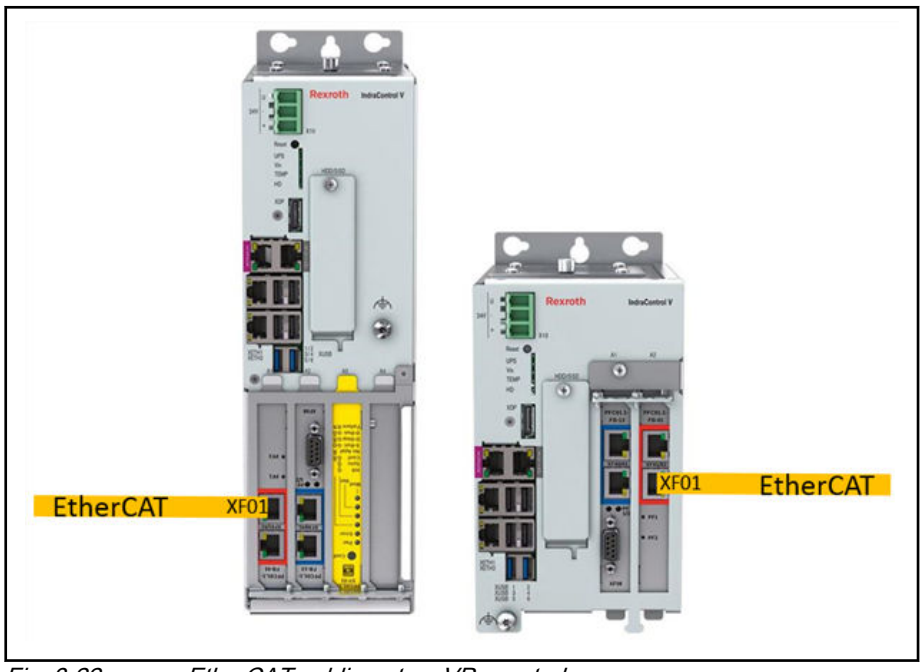


Fig. 8-22: EtherCAT cabling at an VPx control

Function package EtherCAT master

The functionality "EtherCAT master" is licensed as function package, i.e. a respective license has to be purchased.

As for other function packages, the function package "EtherCAT master" can be enabled when creating a control or subsequently via the "properties" of the control.

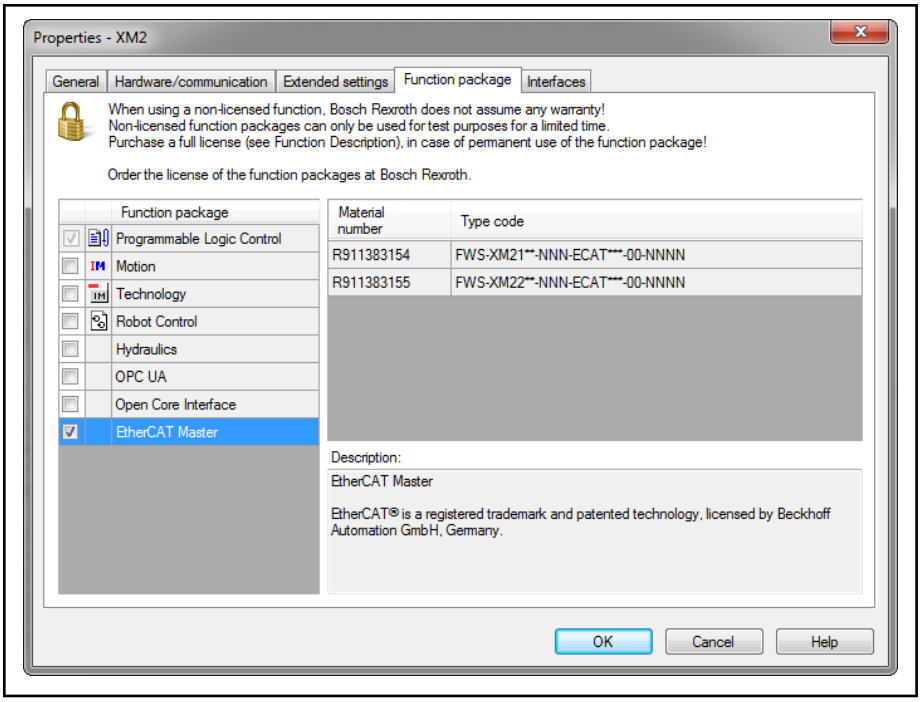


Fig. 8-23: Function package of the EtherCAT master

Configuration download

The EtherCAT bus configuration is transferred to the control by the PLC login and the EtherCAT master is started automatically.

The target state of the EtherCAT master after download (or after control start-up) can be set in the master Tab “General”, page 199.

If the function package “Motion” is used in the control project, the Motion configuration has to be loaded once to the control after setting the EtherCAT master in the control project. The Sercos interface is reconfigured for EtherCAT. Otherwise, the EtherCAT error "Invalid interface" results upon the PLC download.

Overview on configuration dialogs

The following chapters describes the configuration of the EtherCAT master and the slave devices in the IndraWorks project. Double-click on the master or slave objects in the IndraWorks project tree to open the configuration dialogs.

- chapter 8.4 "Configuring EtherCAT master" on page 199
  - chapter 8.4.1 "EtherCAT master, setting device" on page 199
  - chapter 8.4.2 "Tab “General”" on page 199
  - chapter 8.4.3 "Tab “Sync unit assignment”" on page 200
  - chapter 8.4.4 "Tab “Distributed Clocks”" on page 201
  - chapter 8.4.5 "Tab “Bus diagnostics”" on page 203
  - chapter 8.4.6 "Tab “EoE”" on page 205
  - chapter 8.4.7 "Tab “CoE”" on page 206
  - chapter 8.4.8 "Tab “Diagnostics”" on page 206
  - chapter 8.4.9 "Tab “Information”" on page 207
- chapter 8.5 "Configuring EtherCAT master, slaves" on page 207
  - chapter 8.5.1 "Adding slaves and modules" on page 207
  - chapter 8.5.2 "Tab “General”" on page 208
  - chapter 8.5.3 "Tab “SyncManager”" on page 213

– chapter 8.5.4 "Tab “Expert process data”" on page 214
– chapter 8.5.5 "Tab “Process data”" on page 216
– chapter 8.5.6 "Tab “Startup parameters”" on page 216
– chapter 8.5.7 "Tab “EoE settings”" on page 220
– chapter 8.5.8 "Tab “Online”" on page 221
– chapter 8.5.9 "Tab “ESC register”" on page 221
– chapter 8.5.10 "Tab “CoE”" on page 221
– chapter 8.5.11 "Tab “Diagnostics”" on page 222
– chapter 8.5.12 "Tab “I/O mapping”" on page 224
– chapter 8.5.13 "Tab “Information”" on page 225
– chapter 8.5.14 "SoE parameter editor" on page 225

- chapter 8.6 "EtherCAT master, scanning bus" on page 226
- chapter 8.7 "EtherCAT master, bus-synchronous PLC processing" on page 227
- chapter 8.8 "EtherCAT FAQs" on page 228

8.4 Configuring EtherCAT master

8.4.1 EtherCAT master, setting device

The EtherCAT master can be selected instead of the Sercos interface.

This can either be set when selecting the interfaces while creating a control, or if a control already exists in the IndraWorks project, it can be set in the context menu of the “Sercos” node via "Set device".

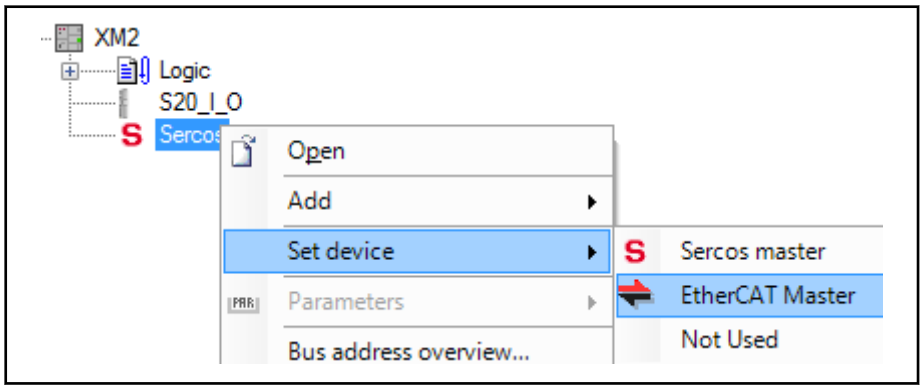


Fig. 8-24: Context menu item "Set device" EtherCAT master

The command "Set device" is not supported for the control type "VPx".

Select the EtherCAT master when selecting the control interfaces, e.g. go to the context menu of the control node and select **Properties ▸ Interfaces**.

8.4.2 Tab “General”

The general settings of the EtherCAT master can be made in this tab.

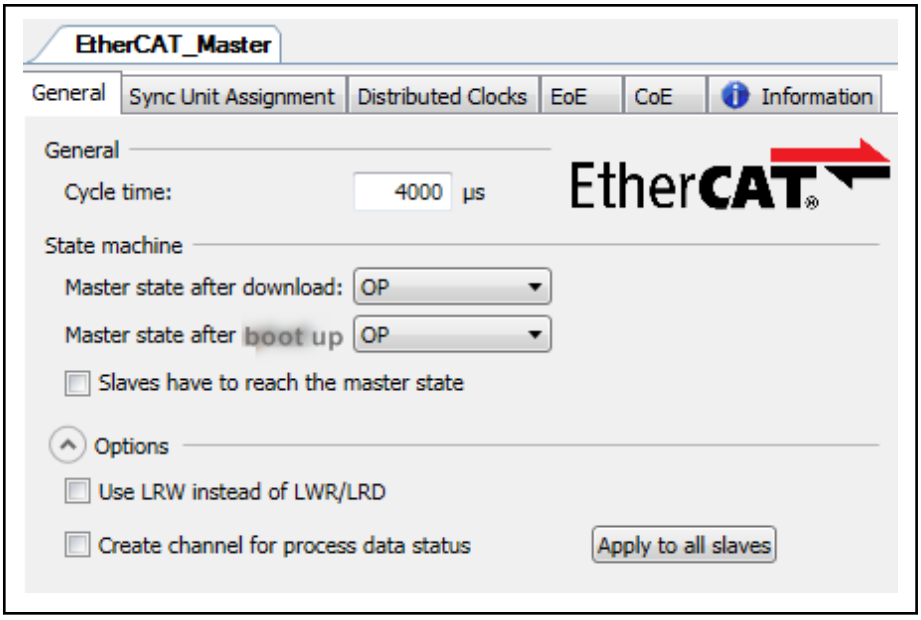


Fig. 8-25: EtherCAT master - "General" tab

General information

Cycle time	Bus cycle time of the EtherCAT communication.

State machine

Master state after download	Master target state after configuration download.
Master state after start-up	Target master state after control startup.
Slaves have to reach the master state	If this option is enabled, **all** states have to reach the master state. If this option is disabled, the master can execute a bus startup (e.g. target state OP) if not all slaves have reached the target state or if some slave are missing at the EtherCAT bus.

Options

Use LRW instead of LWR/LRD	If the option is enabled instead of separate read (LRD) and write (LWR) commands, combined read/write commands (LRW) are used. **Attention**: • **Not all slaves support LRW** • **LRW cannot be used in case of cable redundancy**
Create channel for process data status	If the option is enabled and when a slave is added, the input channel "PdStatus" is created in its I/O mapping. Also refer to chapter 8.5.12 "Tab “I/O mapping”" on page 224. Use the command "Apply to all slaves" to apply the setting to all existing slaves.

8.4.3 Tab “Sync unit assignment”

In this tab, SyncUnits can be created manually and assigned to slaves.

Also refer to the basics chapter 8.2.8 "SyncUnits" on page 176.

The preset SyncUnit "Standard" is not always present and cannot be deleted. At first, all slaves are assigned to the SyncUnit "Standard". Further SyncUnits can then be created and assigned to slaves.

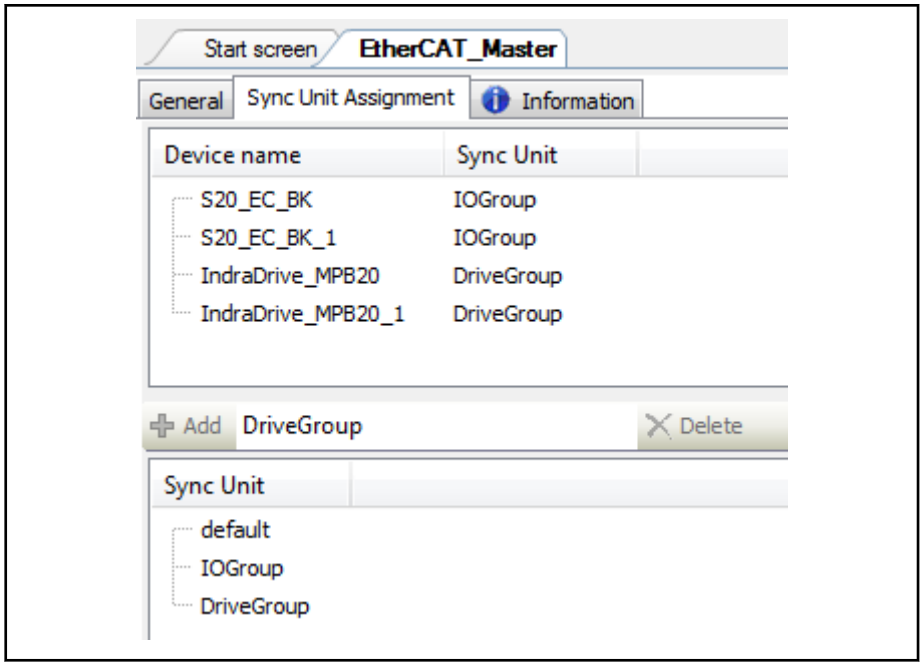


Fig. 8-26: EtherCAT master - "Sync unit assignment" tab

Device name	Name of the slave from the project tree
Sync unit	Slave assignment to a SyncUnit. Double-click on the SyncUnit field of a slave (on the right of the device name) to select one of the configured SyncUnits.
Add	If a name is entered into the text field on the right of "Add", a SyncUnit with this name is created via “Add”.
Delete	Deletes the SyncUnit selected in the lower window. The preset SyncUnit "Standard" cannot be deleted. If SyncUnit unit is deleted, but this SyncUnit is still assigned to slaves, these slaves are again assigned to the SyncUnit "Standard".

8.4.4 Tab “Distributed Clocks”

Make the general settings of the "Distributed Clocks" in this tab.

Additionally, diagnostics are displayed for the Distributed Clocks if logged in.

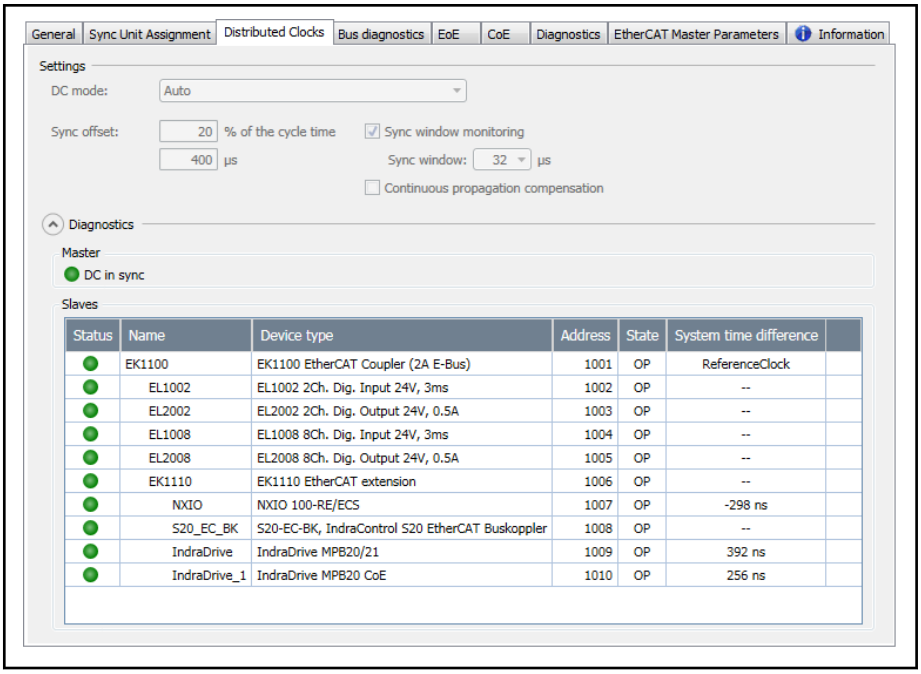


Fig. 8-27: "Bus diagnostics"

Settings

DC mode	• **Automatic**: If at least enabled for one slave DC, "Bus Shift" is set automatically. • **Free Run**: Cyclic communication without clock synchronization (i.e. the "Distributed Clocks" are not controlled). • **Bus Shift**: Synchronous mode of the slaves with enabled DC. Bus Shift: The EtherCAT master sets the system time and writes it to the first DC slave at the bus (shifting the bus time).
Sync Offset	The "Sync Offset" is the temporal offset of the Sync0 pulse in the DC slaves when sending the cyclic data telegram in the master. It is set as percentages of the cycle time. A typical setting is 20%.
Sync Window Monitoring	If this option is enabled, the synchronization of the Distributed Clocks is monitored in all slaves. Each slave cyclically reads the "System time difference" (reg. 0x092C). The "Sync Window" input specifies the maximum system time deviation allowed (reference clock). If - for all slaves - the system time difference is smaller than the monitoring window, the synchronization is considered to be adjusted ("DC in Sync").
Continuous propagation compensation	"Continuous Propagation Compensation": If the option is enabled, the propagation delay of the EtherCAT telegram is cyclically measured at the bus. This allows to adapt the propagation compensation continuously. For EtherCAT bus configurations without a topology change in the operating mode, do not disable this option.

Diagnostics

The DC diagnostics is only displayed in logged-in state.

Master: "DC in Sync" applies if - for all slaves - the system time difference is smaller than the monitoring window (also refer to "Sync Window Monitoring" for DC settings).

Slaves: More values can be shown in the table header next to the predefined columns using the context menu (right click).

Status	"Green": OK, "Red": Error present. Go to the "Bus diagnostics" tab.
Name	Configured device name of the slave
Device type	Device type description of the slave (from ESI)
Address	EtherCAT address
State	EtherCAT state of the slave
System time difference	System time difference in ns (ESC register 0x092C) (this value is only valid if DC is configured) The first slave at the bus with DC enabled sets the reference clock.
SlaveDelay	Time delay “after” the slave in ns (this value is only valid if DC is configured)
PropagationDelay	Transmission delay in ns (ESC Register 0x0928) (this value is only valid if DC is configured)
DC	Indicates, whether the EtherCAT slave controller supports DC (32/64 bits). Note: The display that DC is supported does not mean that DC can be activated for the slave in the configuration (specified in the ESI).
Vendor/ID **Product code, version** **Serial number**	Identity information (EEPROM offset 0x0008)

8.4.5 Tab “Bus diagnostics”

The tab is only displayed in logged-in state.

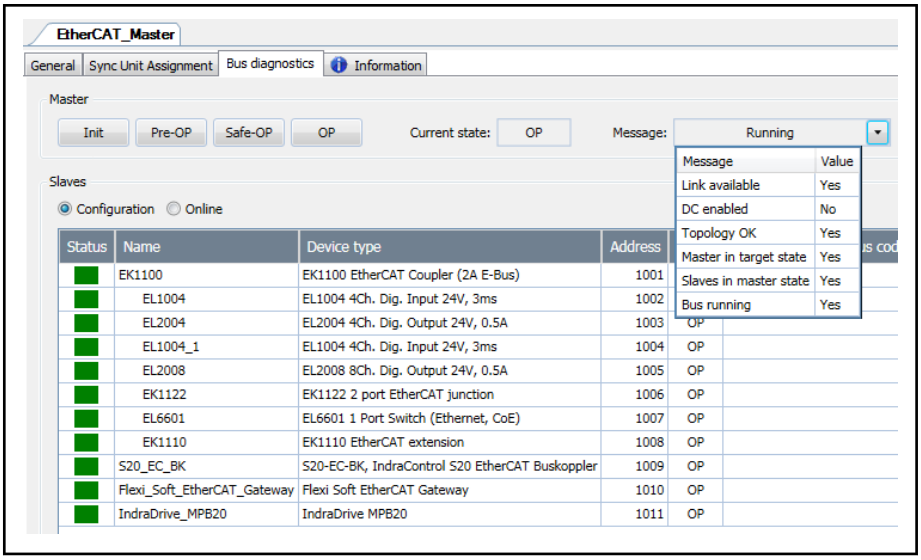


Fig. 8-28: EtherCAT master - "Bus diagnostics" tab

Use the buttons "Init" / "Pre-OP" / "Safe-OP" / "OP" to toggle the bus state at the EtherCAT master. The current bus state is also displayed.

The following diagnostics can be displayed in the "Message" field:

Text	Description
Link not available	No link at the EtherCAT master.
DC not in sync	The synchronization of the distributed clocks (DC) is not within the configured monitoring window (only if DC is enabled).
Topology not OK	The configured slaves do not match the slaves at the bus, e.g. slaves are missing or the configuration of the topology does not match the real topology at the bus. Also refer to the following table of the EtherCAT state "not connected". This diagnostics is also set if there are more slaves at the bus than configured.
Master not in target state	The requested EtherCAT master state could not be reached.
Slaves not in master state	One or multiple slaves are not in the master state.
Slaves in error state	One or multiple slaves report(s) an error (e.g. AL status code).
Master not in OP	The EtherCAT master is not in the state OP.
Bus running	The EtherCAT master and all slaves are in a valid cyclic communication (state OP).

Two alternative views can be toggled under "Slaves".

- All configured devices are displayed if "Configuration" is selected.
- All currently present slave devices at the bus are displayed if "Online" is selected. This view is especially useful if a "topology error" is present, i.e. the configured topology in the IndraWorks project tree does not match the cabling of the slaves at the bus.

The columns of the table have the following meaning:

Status	"Green": OK, "Red": An error is present. The slave is not connected or refer to the "Diagnostics" column
Name	Configured device name of the slave. An unconfigured slave is shown with the name "Slave_x" in the "Online" view.
Device type	Device type description of the slave (from ESI). A slave, whose ESI is not installed in the IndraWorks device database, is shown as "Unknown device" in the "Online" view.
Address	EtherCAT address
State	EtherCAT state of the slave Note: The state "not connected" means that the configured slave could not be assigned to any device at the bus, i.e. the device is either physically not connected to the bus or the topology position of the device at the bus does not match the configuration. Tip: In case of a topology error • Compare the configuration of the topology in the IndraWorks project tree with to "online" view of the slaves • Also refer to "Configuring the topology" on page 208
Diagnostics	Diagnostics, e.g. EtherCAT state error (refer to chapter "AL status codes" on page 178)

Changing slave state

Right-click (context menu) on a slave to individually change this slave state.

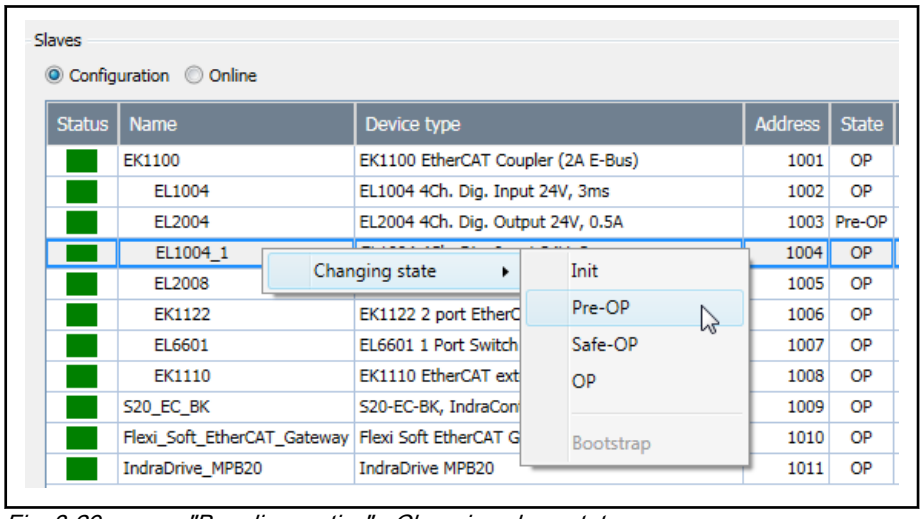


Fig. 8-29: "Bus diagnostics" - Changing slave state

8.4.6 Tab “EoE”

In this tab, a virtual Ethernet port can be enabled in the EtherCAT master.

If the IP address is set for the EtherCAT master, the IP routing to the engineering port of the control is enabled at the same time.

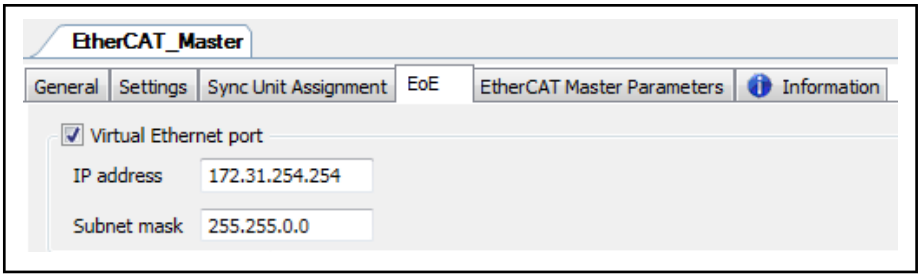


Fig. 8-30: EtherCAT master - "EoE" tab

Valid IP settings have to be specified with regard to the network configuration.

Also refer to

- Basics: "EoE: Ethernet over EtherCAT" on page 174
- Slave configuration: chapter 8.5.7 "Tab “EoE settings”" on page 220

8.4.7 Tab “CoE”

The tab is only displayed in logged-in state.

CoE objects of the EtherCAT master are displayed for diagnostic purposes.

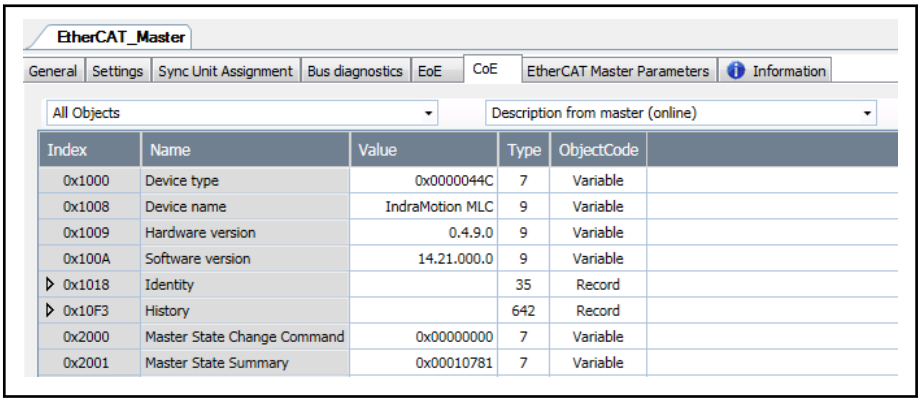


Fig. 8-31: EtherCAT master - "CoE" tab

8.4.8 Tab “Diagnostics”

The tab is only displayed in logged-in state.

The CoE object "Diagnosis History" (0x10F3) of the EtherCAT master is displayed.

For a detailed description of the dialog, refer to chapter 8.5.11 "Tab “Diagnostics”" on page 222.

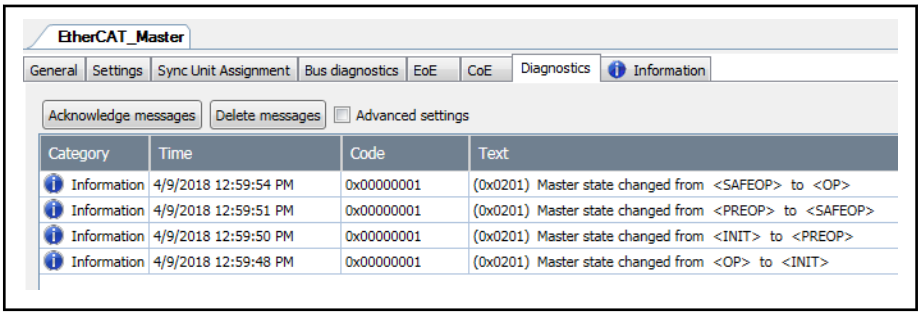


Fig. 8-32: EtherCAT master - "Diagnostics" tab

8.4.9 Tab “Information”

This tab displays some general information from the device description file: Name, vendor, category, version, description, etc.

8.5 Configuring EtherCAT master, slaves

8.5.1 Adding slaves and modules

Adding slaves

Go to the context menu item "Add" on the EtherCAT master object to add slave devices to the master.

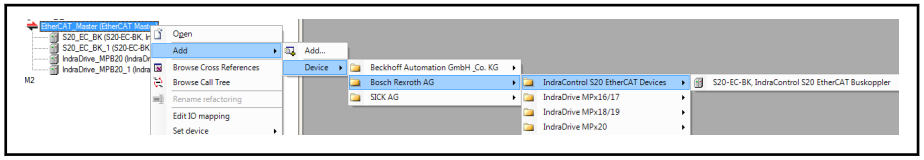


Fig. 8-33: EtherCAT master context menu item "Add slave"

Alternatively, drag devices from the IndraWorks device library and drop them to the EtherCAT master to add them.

Adding modules

Modules at a slave with modular device profile (e.g. Rexroth S20-EC-BK) are analogously added to the bus coupler via the context menu item "Add" on the slave object.

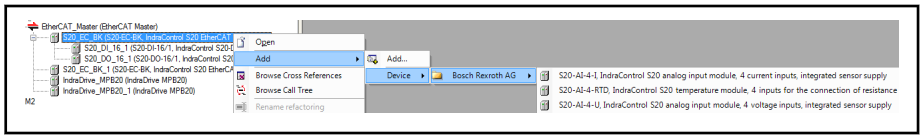


Fig. 8-34: EtherCAT master context menu item "Add module"

Adding slaves to the E-BUS

Modules (terminals) at a bus coupler with E-BUS, e.g. Beckhoff EK1100, are also attached to the bus coupler via the context menu item "Add". These E-BUS modules are shown as subnodes to the bus coupler in the project tree. However, they are an own slave device with regard to the EtherCAT bus configuration and are thus provided with all dialogs of a slave.

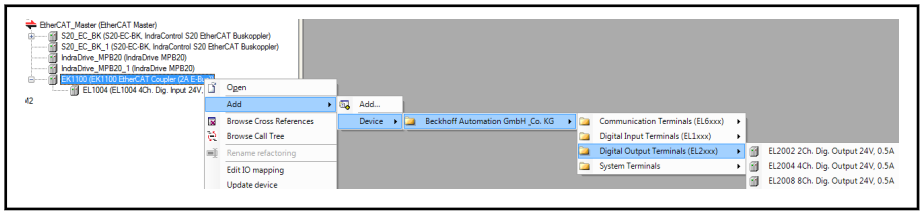


Fig. 8-35: EtherCAT_BusCoupler context menu item "Add module"

Tip: Input dialog "Add"

The "Add" input dialog helps when adding slaves and modules.

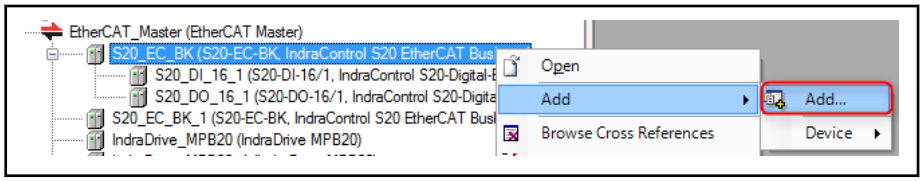


Fig. 8-36: Input dialog "Add", call

This dialog shows all devices that can be added to this location. The devices shown depends on the highlighted position in the project tree.

Additionally, a filter can be used to easily search for specific slaves or modules.

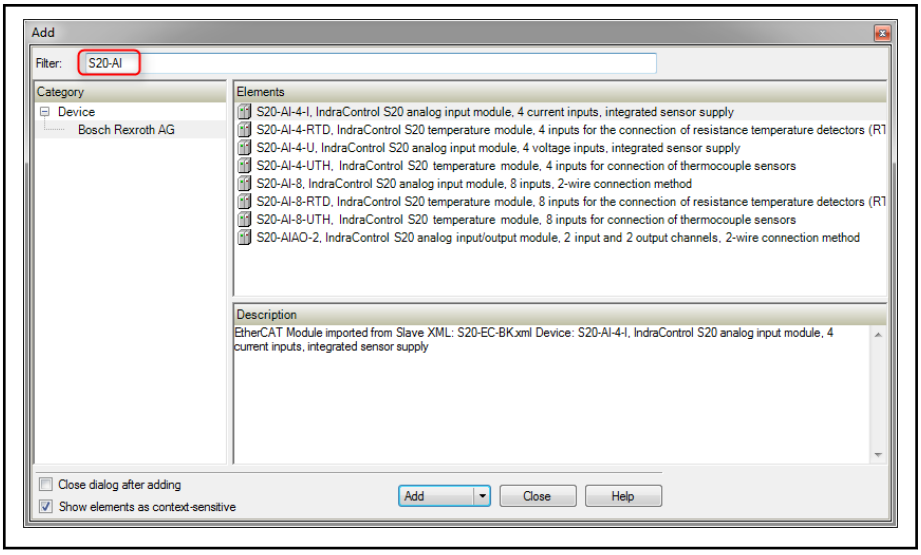


Fig. 8-37: Input dialog "Add", view

Configuring the topology

When adding a slave, note that the topology of the EtherCAT network has to be uniquely mapped in the configuration. The physical order of the slaves at the bus has to match the order of the slaves in the IndraWorks project tree (auto increment address). The auto increment address is automatically configured by the configurator "from up to down".

The "preceding port" configuration has to match as well. I.e. Note especially for junction terminals (e.g. Beckhoff EK1122) or extension terminals (e.g. Beckhoff EK1110) that the slaves connected to the terminals also have to be added to the respective subordinate node in the project tree.

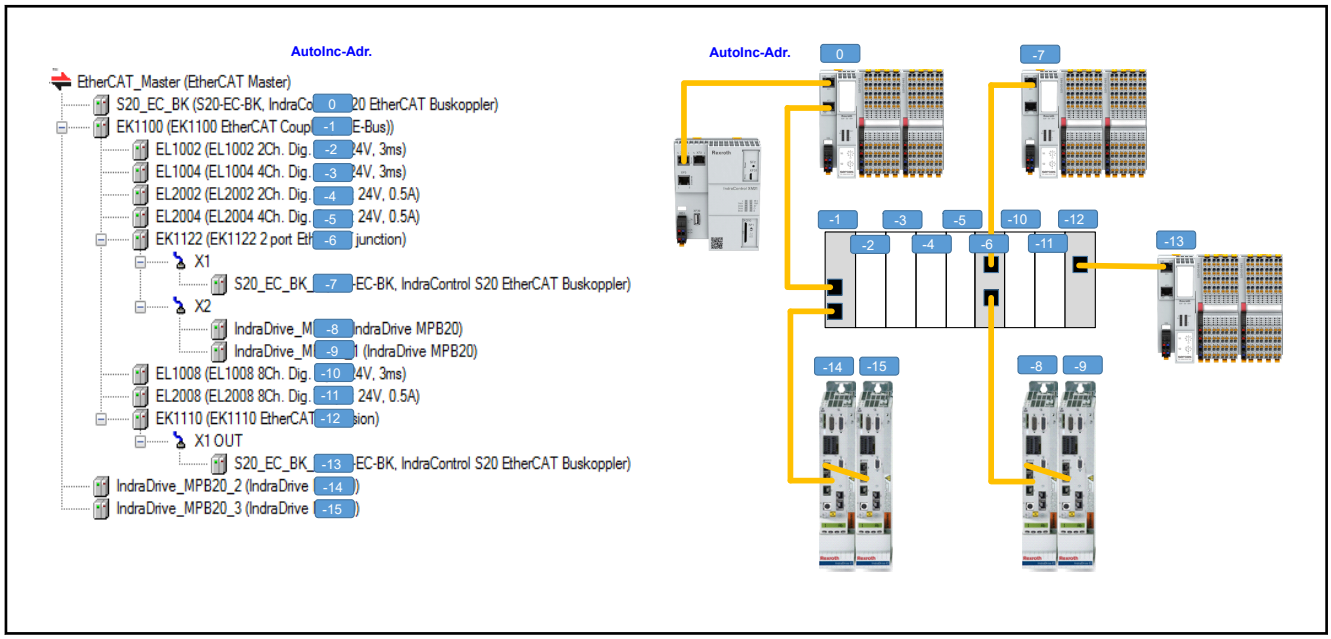


Fig. 8-38: Configuring the EtherCAT topology in the IndraWorks project tree

The slave sequence in the IndraWorks project tree can also be changed via drag&drop.

8.5.2 Tab “General”

The basic settings for the EtherCat slave are configured in this tab.

The default basic settings result from the device description file (ESI) of the slave.

Dialog "Address and Additional"

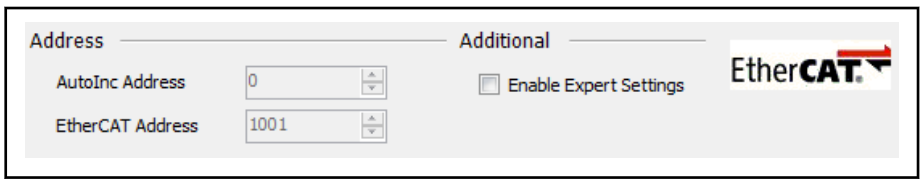


Fig. 8-39: Dialog "Address and Additional"

AutoInc address	Auto increment address: Topological address of the slave as negative auto increment (16 bits), for example: 0, -1 (0xFFFF), -2 (0xFFFE) and -3 (0xFFFD). This AutoInc address is automatically specified by the configurator in the sequence in which the slaves were created in the project tree (from top to bottom).
EtherCAT address	EtherCAT address of the slave (also called “Fixed Physical Address“). The address is not saved in the slave, but the master assigns this address to the slave upon bus startup. Also refer to the basics chapter 8.2.3 "Addressing and identification procedure" on page 170 **Note**: The EtherCAT address can currently not be changed. The configurator specifies it automatically.
Expert settings	☑ Additional configuration settings can be made, e.g. startup checks, timeouts, watchdog and identification (refer to the following description). The “Expert process data” tab is enabled.

Dialog "Diagnostics"

The diagnostic field "Current status" is not logged-in in displayed state and it is currently not supported.

Dialog "Distributed Clocks"

Also refer to the basics chapter 8.2.11 "Synchronization" on page 185

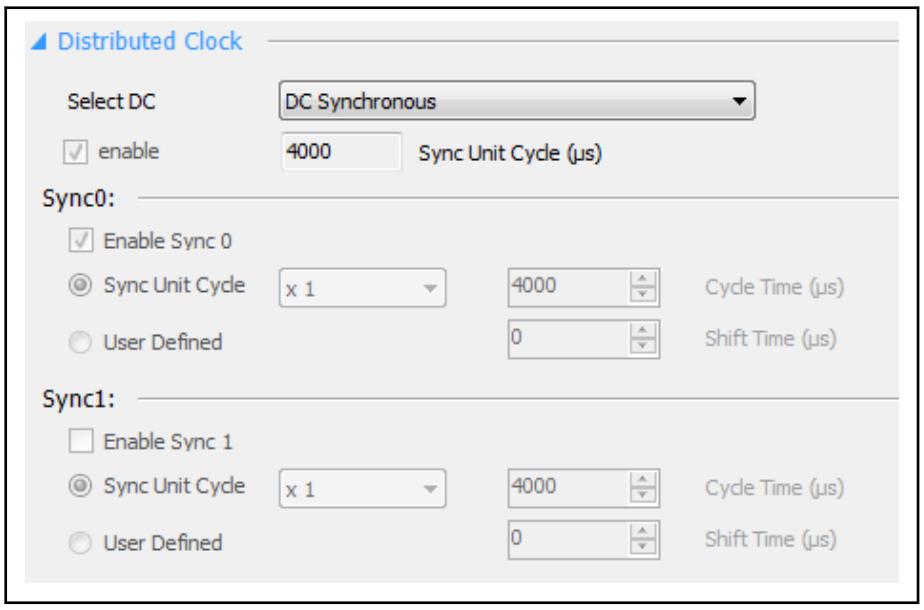


Fig. 8-40: Dialog "Distributed Clocks"

Select DC	Dropdown list with predefined settings for Distributed Clocks from the device description file (ESI) of the slave
Sync Unit Cycle (µs)	Display field of the EtherCAT communication cycle time (it is the basis for the DC cycle time)
Enable	☑ If the "Expert settings" option is selected, a user-defined DC setting can be configured.
Enable Sync 0/ **Enable Sync 1**	☑ Use Sync0 event in EtherCAT slave/ ☑ Use Sync1 event in EtherCAT slave
Sync Unit Cycle (Sync0/Sync1)	The dropdown list can be used to set a factor for multiplying the EtherCAT communication cycle time. This corresponds to the synchronization cycle time or the DC cycle time of the Sync event in the slave (=> display field "Cycle time (µs)")
User-defined	The (Sync) "cycle time (µs)" can be set manually.
Shift time	Shifting the Sync event in the slave

Dialog "Startup checking and timeouts"

These dialogs are only displayed if the "Expert settings" are enabled (see above).

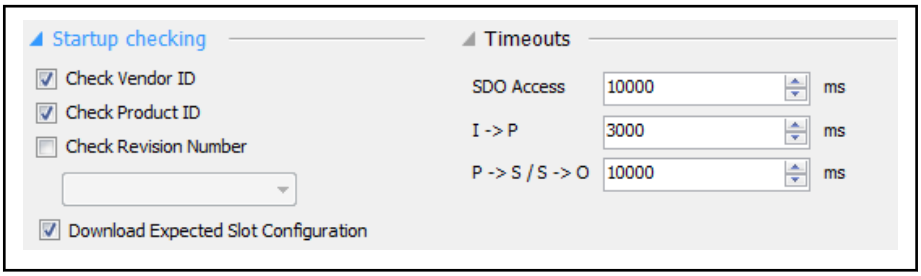


Fig. 8-41: Dialog "Startup checking and timeouts"

Startup check **The startup checks are used to see whether the configured slave matches the physical device. Thus, there is no startup of a faulty configuration.**	
Check vendor ID	☑ Upon bus startup, it is checked whether the vendor ID of the configured slave matches the physical device.
Check product ID	☑ Upon bus startup, it is checked whether the product ID of the configured slave matches the physical device.
Check revision number	☑ Upon bus startup, the revision number of the configured is checked with regard to the value of the physical device according to the selection of the drop-down list. Partially different checks can be selected for low word (LW / bits 0-15) and high word (HW / bits 16-31) of the revision number. Note: The Rexroth EtherCAT master supports only the "==" check. A ">=" check is not yet standardized in ETG. 2100 (ENI).

Download expected slot configuration	☑ for a slave with "modular device profile" (e.g. Rexroth S20-EC-BK), expected or configured slot/module configuration is loaded to the device to be able to compare them to the actual module configuration in the device.
Timeouts **Timeout monitoring for acyclic services and accesses. The default settings result generally from the device description file (ESI) of the slave.**	
SDO access	Timeout for SDO accesses upon bus startup
I -> P	Timeout for state transition Init -> PreOperational
P -> S / S -> O	Timeout for state transition PreOperational -> SafeOperational and SafeOperational -> Operational

Dialog "Watchdog"

The watchdog monitors that the outputs can be switched to the safe state again after a defined period if an error occurred in the process data communication. The EtherCAT slave controller (ESC) is provided with watchdogs for the Sync manager (SM) and for the process data interface (PDI).

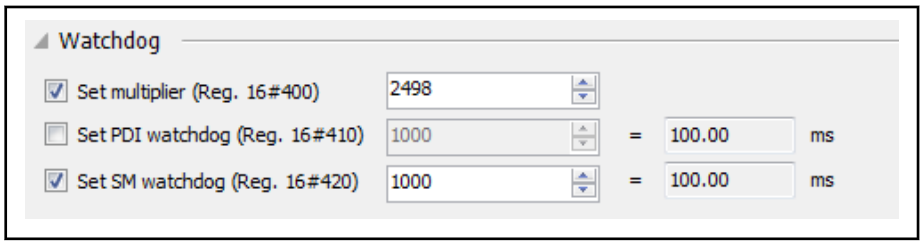


Fig. 8-42: "Watchdog" dialog

Set multiplier	The multiplier (register 16#400) divides the local cycle to specify the trigger for the watchdogs: (1 / 25 MHz) * (multiplier + 2) With the multiplier standard setting 2498, the interval 100 µs results (computation: (1 / 25 MHz) * (multiplier + 2) = 40 ns * 2500 = 100 µs) Example of PDI watchdog: With the factor 1,000 results a watchdog timeout of 100 ms. A manual adaptation of the multiplier is for example required for very slow EtherCAT communication cycle times.
Set PDI watchdog	The PDI watchdog (register 16#410) monitors the process data communication from the ESC to the application (e.g. local micro controller) of the slave.
Set SM watchdog	The SM watchdog (register (16#420) monitors the cyclic EtherCAT communication. The watchdog is reset after each successful EtherCAT process data communication. If the watchdog is enabled and longer than the monitoring time set, no EtherCAT process data communication takes place and the outputs are set to the safe state.

Dialog "Identification"

An identification procedure can be used to uniquely identify a slave irrespective of its topological position (e.g. to ensure that there is no wiring error).

Normally, rotary or DIP switches are used to set an identification value at the slave devices.

Also refer to the basics chapter 8.2.3 "Addressing and identification procedure" on page 170.

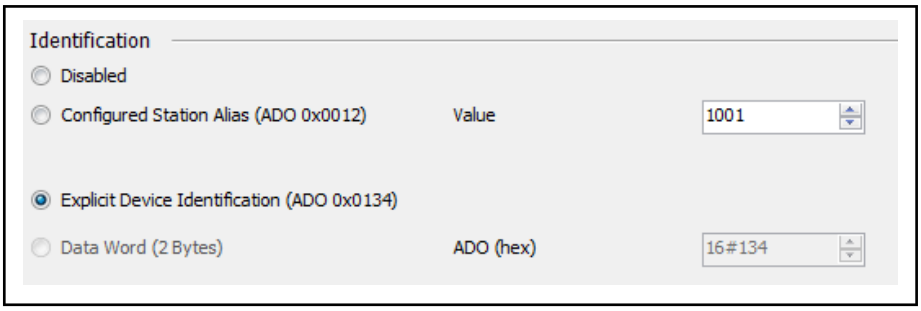


Fig. 8-43: "Identification" dialog

Disabled	There is no identification.
Configured station alias	Identification via "(configured) station alias": Normally, the value of the station alias is written to the slave EEPROM via the remote assignment. The slave then loads this value to its register ADO 0x0012 when booting (ADO = Address Offset). Example: This procedure is for example used for the Rexroth IndraDrive for which the station alias can also be set via the display.
Explicit device identification	Identification via "Explicit Device Identification": Upon request, the slave informs the master on its identification value in the AL status code register (ADO 0x0134). Example: For the Rexroth S20-EC-BK, the identification values for "Explicit Device Identification" is for example set using the rotary switch at the device.
Data word	Identification via "Data Word" (also called "Input Word"): The identification value is expected at a specific register address offset (ADO) specified from the device description file (ESI) of the slave and arbitrarily set if required (refer to the product description of the slave device). Example: For the Beckhoff EK1101, the identification value for "Data Word" is for example set using the rotary switch at the device.

Value	This is the "command value" specified for the selected procedure. This value has to match the value set in the slave device.
"Write EEPROM" and actual address	In logged-in state, the "Write EEPROM" and the field "Actual address" are displayed. **Note**: This online functions are not supported by the Rexroth EtherCAT master.

8.5.3 Tab “SyncManager”

This tab is only displayed if the "Expert settings" are enabled in the "General" dialog.

The dialog displays the SyncManager configuration for this slave and allows a manual change → Recommended only for experts!

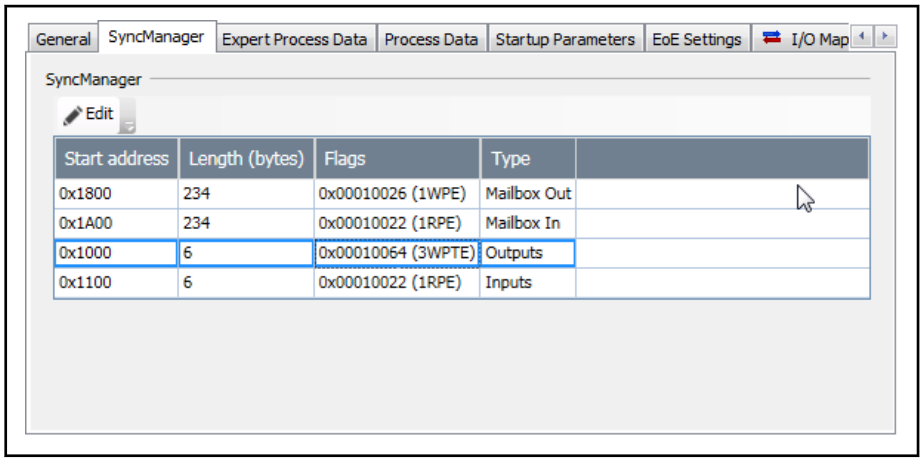


Fig. 8-44: "SyncManager" tab

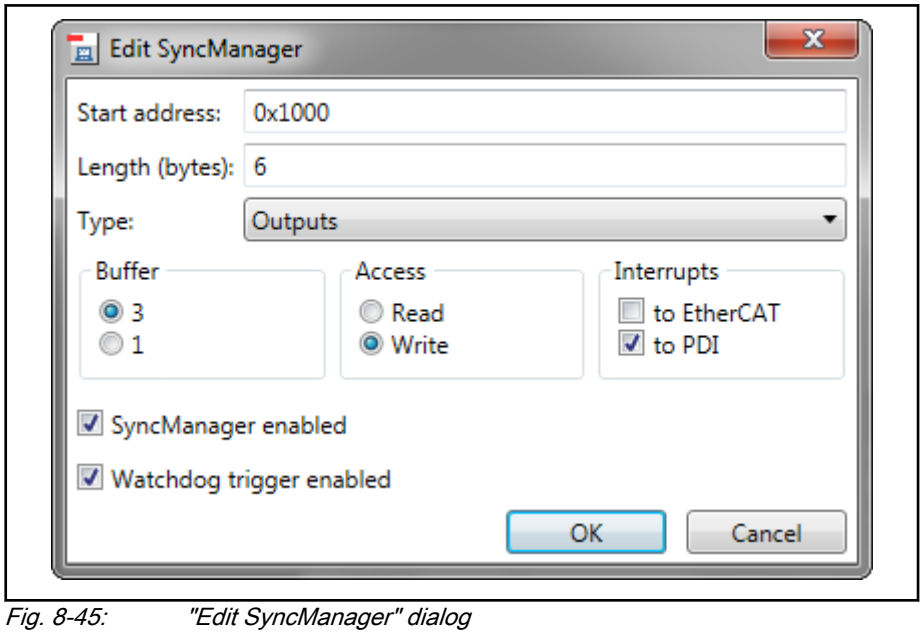


Fig. 8-45: "Edit SyncManager" dialog

Example:

For SoE devices, e.g. Rexroth IndraDrive, the SyncManagers are set to "1 buffer" by default (acc. to the ETG specification). This can cause “Working Counter” errors in the “Free Run” mode.

Thus, the SyncManager setting "3 buffers" is recommended for cyclic data (inputs/outputs) in the ”Free run” mode.

8.5.4 Tab “Expert process data”

This tab provides a detailed view and expert configuration for process data shown in the "Process data" dialog. The download of the PDO assignment and the PDO configuration is enabled additionally.

Prerequisite: This dialog is only displayed if the "Expert settings" are enabled in the "General" dialog.

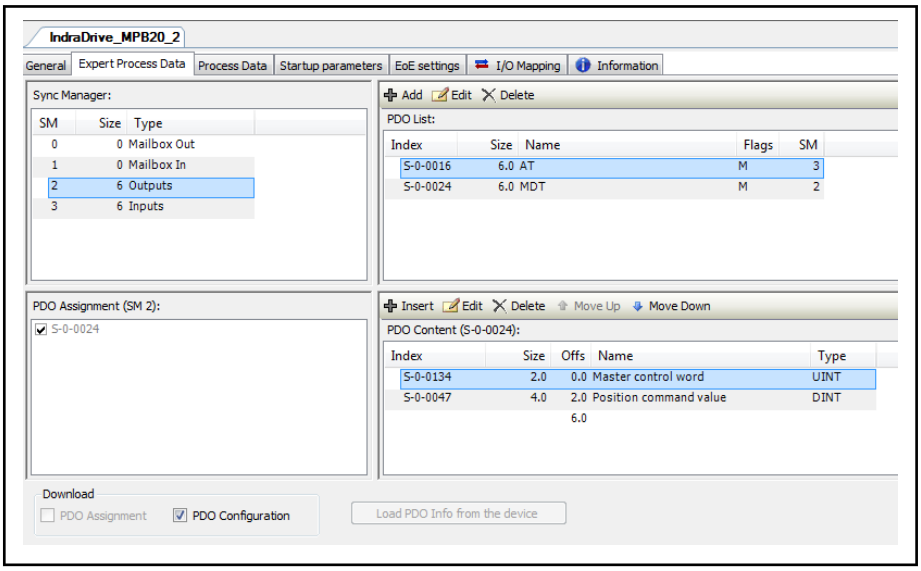


Fig. 8-46: Tab "Expert process data"

Sync manager List of Sync managers (SM) with data size and PDO type (PDO: Process data object).

PDO assignment (SM x) List of the PDOs assigned to the selected Sync manager (in the field "Sync Manager").

If a checkbox is selected, the PDOs are selected and the I/O channels are generated (as for the object selection in the "Process data" dialog)

PDO list List of the PDOs assigned to the selected Sync manager (in the field "Sync Manager").

To add new entries or edit or delete existing PDOs via the respective commands (Add, Delete, Edit) in the command bar or via the context menu (=> see "Edit PDO list").

Edit PDO list This dialog opens when "adding" or "editing" PDOs in the "PDO list".

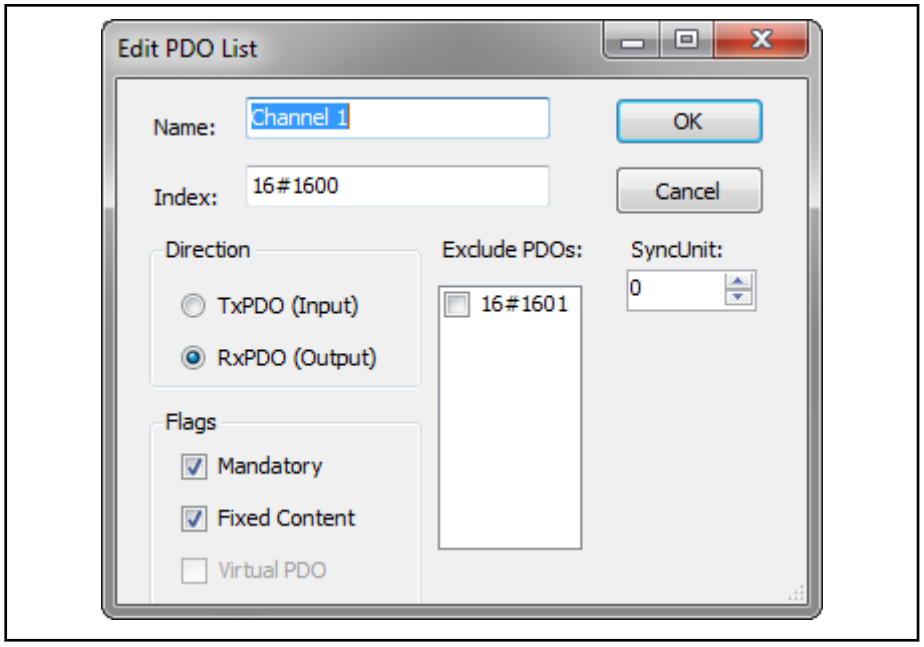


Fig. 8-47: "Edit PDO list" dialog

Name	PDO name
Index	PDO index (or IDN for SoE)
Direction	• TxPDO (input in the PLC): PDO is transferred from slave to master • RxPDO (output in der PLC): PDO is transferred from master to slave
Flags	• Mandatory: The PDO is required and cannot be disabled in the PDO assignment. • Fixed content: The PDO content is specified and cannot be changed. It is then not possible to add entries to the PDO content. • Virtual PDO: Reserved for future use
Exclude PDOs	It is possible to specify an exclusion list. If a PDO is enabled in the PDO assignment, the others are inactive and cannot be enabled.
SyncUnit	ID of the SyncUnit in the slave (Sync manager) to assign the PDO to.

PDO content

Indicates the PDO content selected in the "PDO list".

To add new entries or edit or delete existing entries via the respective commands (Add, Delete, Edit) use the command bar or the context menu.

Change the PDO order with "Move up" and "Move down".

Download

PDO assignment	☑ Specific CoE commands are generated to initialize the objects (0x1Cxx) and written to the slave.
PDO configuration	☑ CoE commands are generated for the PDO mapping (0x16xx and 0x1Axx) and loaded to the slave. Normally, the default values originate from the ESI file and the device has to support this. For example, if a device has a fixed configuration, these commands are considered to be faulty.

Load PDO info from device

The current PDO configuration is retrieved from the slave and entered into the configuration. The lists in the upper and lower right are then completely deleted and filled with the read data. This is always reasonable if the ESI file is incomplete or if the configuration is only present in the slave.

☞ "Load PDO info from device" is not supported by the Rexroth EtherCAT master!

8.5.5 Tab “Process data”

This tab shows process data for the inputs and outputs of the slave.

The data originates from the device description file (ESI).

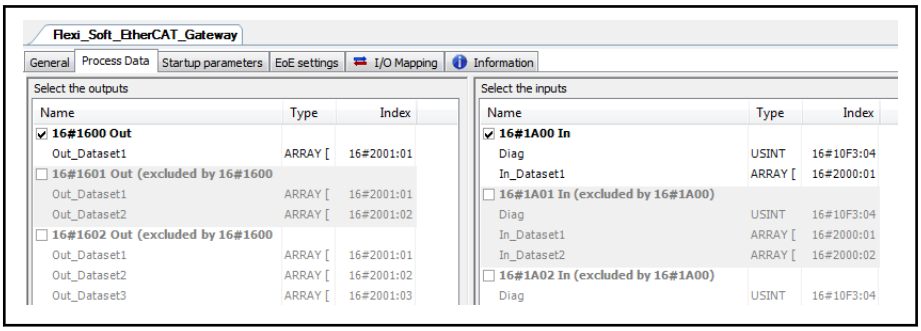


Fig. 8-48: Tab "Process data"

Select the outputs

The table shows the outputs of the slave defined by starting address, type and index.

If the outputs of the slave are enabled here (☑), these outputs can be used in the "I/O mapping" dialog. Output objects can be excluded by other (activated) output objects (these dependencies are specified in the device description file of the slave).

Select the inputs

The table shows the inputs of the slave defined by name, type and index.

If the inputs of the slave are enabled here (☑), these inputs can be used in the "I/O mapping" dialog. Input objects can be excluded by other (activated) input objects (these dependencies are specified in the device description file of the slave).

8.5.6 Tab “Startup parameters”

In this tab, parameters are defined that are written to the slave device upon bus startup (during the transition from PreOP to SafeOP).

It is required that the slave device supports "CAN application layer over EtherCAT" or "Servo drive profile over EtherCAT" (SoE). Thus, only SDOs (Service Data Objects) or IDNs (identification numbers) can be entered.

Some modules inserted below a slave are provided with individual startup parameters. These parameters are then also displayed in this list, but cannot be edited. The parameter is changed in the editor of the respective module.

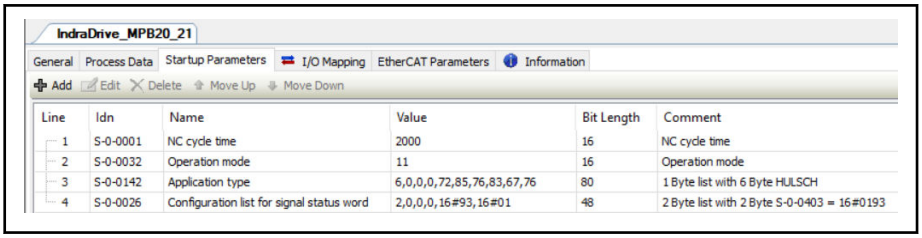


Fig. 8-49: Tab "Startup parameters"

The order (from top to down) in the table specifies the sequence of the transfer of SDOs and IDNs to the slave device.

Line	Line number
Idn	Identification number (SoE)
Index: Subindex	Index and subindex of the object (CoE)
Name	Parameter name
Value	Parameter value
Bit length	Bit length of the parameter
Comment	Input field for comment
Move up	Moves up the selected line by 1
Move down	Moves down the selected line by 1
Add	Opens the input dialog (SDO or IDN) to add parameters to the startup parameter table
Edit	Opens the input dialog (SDO or IDN) to edit the selected parameter
Delete	Removes the currently selected entry

Adding IDN/editing (SoE)

An entry can be selected from the object directory (prerequisite is that the parameters are described in the device description file (ESI) of the slave device) or it can be defined manually via the input fields.

Columns	
Idn	Identification number
Basic value	Basic value of the IDN, editable (double-click to open)
Input fields	
IDN	Parameter type: S- or P-parameters
PSet	Parameter set
Offset	Data block number
Bit length	Bit length

Value	Value
Is List	 : If the option is selected "as list", Sercos lists can be specified. The option is in the dialog to select the object directories, refer to the example "Example of 1 byte list" on page 218. Lists are specified in the following format as byte arrays: `<ActualLength (LowByte)>,<ActualLength (HighByte)>,0,0,<Byte1>,<Byte2>,<Byte3>...` → The byte array format has to be in "Little Endian".

Adding SDO/editing (CoE)

An entry can be selected from the object directory (prerequisite is that the objects are described in the device description file (ESI) of the slave device) or an SDO can be defined manually via the input fields.

Columns	
Flags	Display of the access flag: RW ("read write" = read and write), RO ("read only" = read-only), WO ("write only" = write-only)
Basic value	Basic value of the object, editable (double-click to open)
Input fields	
Name	Object name
Index	Index of the object (16#)
SubIndex	Subindex of the object (16#)
Bit length	Bit length
Value	Value
Byte array	The object is a list.

Example of "1 byte list"

The string “Hulsch” (1-byte list) is for example written to the "Application type" parameter.

The "Hulsch" string corresponds to the following ASCII characters:

72(H),**85**(u), **76**(l),**83**(s),67(c),**72**(h)

The "Hulsch" string has six characters.

Thus, the "actual length" is 6 bytes and the following array results for the list: "6,0,0,0,72,85,76,83,67,72"

Enter the array into the "Value" field and select the option "As list" as shown in the following screen:

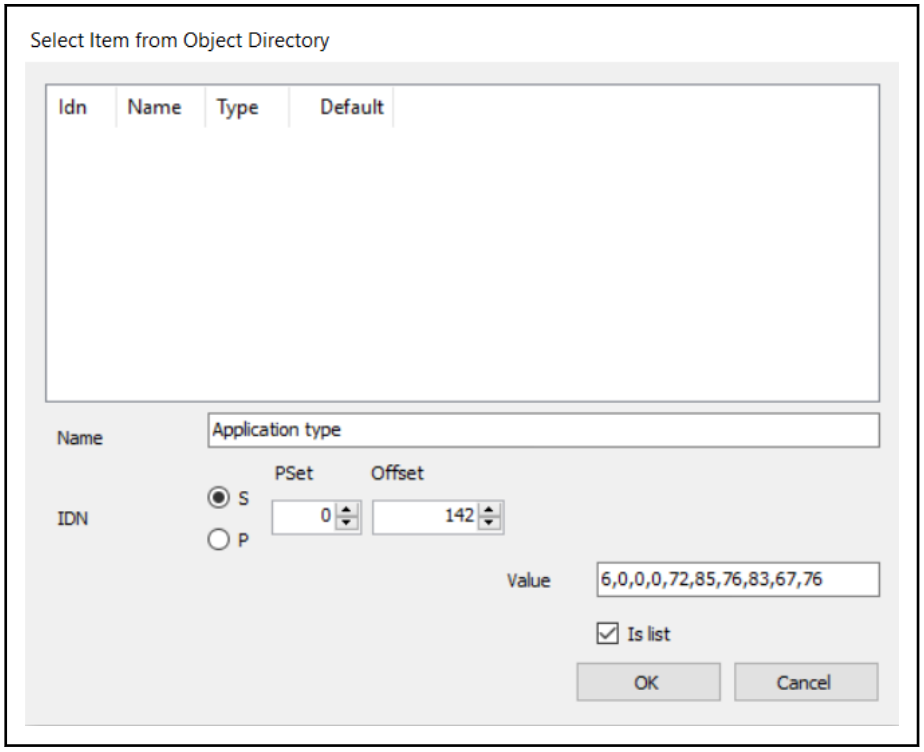


Fig. 8-50:

Example of a "2 byte list"

The IDN S-0-403 is written as first list element into the parameter list S26 "Configuration list for signal status word".

S-0-403 corresponds to `0x0193`

This results in an "actual length" of 2 bytes = `0x0002`

Enter the following into the "Value" field: 2,0,0,0,16#93,16#01

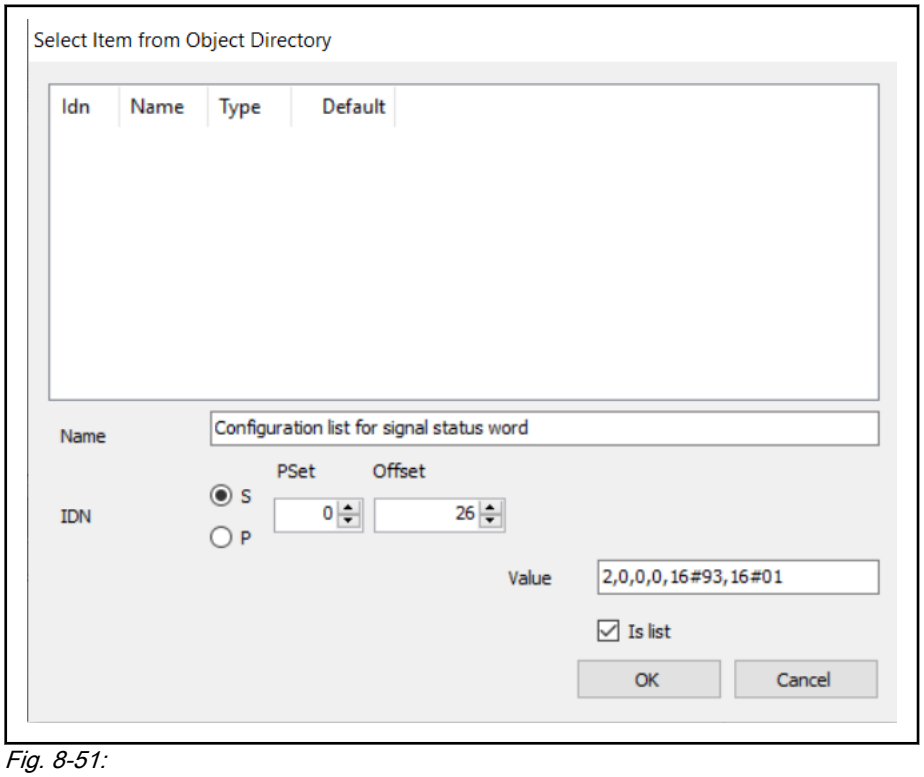


Fig. 8-51:

8.5.7 Tab “EoE settings”

In this tab, the communication settings are made for the Ethernet over EtherCAT (EoE). The dialog is only displayed if the slave device supports EoE and if "Expert settings" are enabled in the "General" dialog.

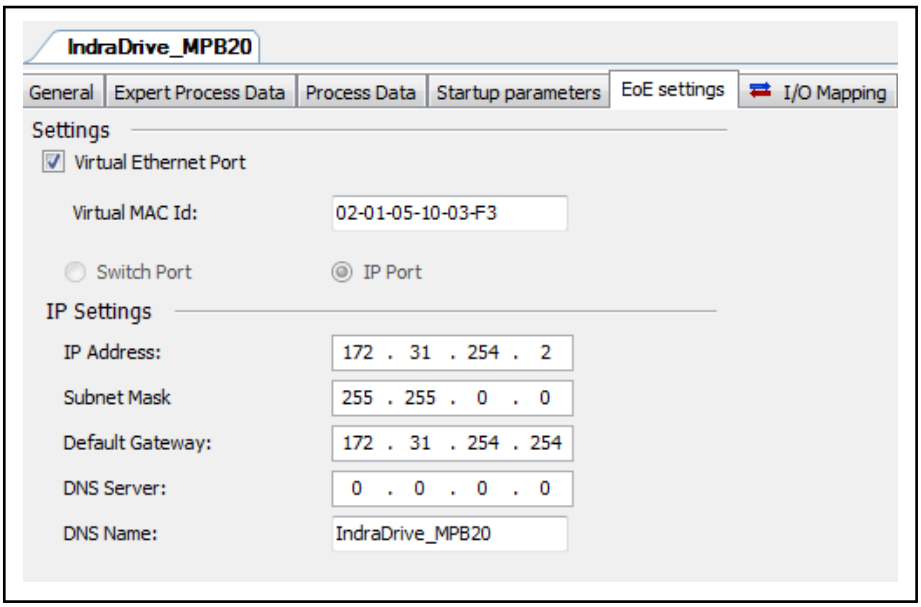


Fig. 8-52: Tab "EoE settings"

Tab “EoE settings”

Settings	
Virtual Ethernet Port	☑: Enables the EoE functionality of the slave. A unique virtual MAC ID has to be defined
Virtual MAC ID	Input field for the virtual MAC ID
Switch port	◉ Device operates as switch. No further network settings required
IP port	◉ Device operates as IP port. IP settings have to be made.
IP settings **Valid IP settings have to be specified with regard to the network configuration.**	
IP address	IP address of the slave in the network Example: If the IP address of the EtherCAT master is 172.31.254.254 and if the subnet mask is 255.255.0.0, the IP address of the slave has to be between 172.31.0.0 and 172.31.254.253. Also refer to • Basics: "EoE: Ethernet over EtherCAT" on page 174 • Master configuration: chapter 8.4.6 "Tab “EoE”" on page 205
Subnet mask	Subnet mask of the slave
Standard gateway	Standard gateway of the slave (e.g. IP address of the master)

DNS server	IP address of the DNS server
DNS name	DNS server name

8.5.8 Tab “Online”

The tab is only displayed in logged-in state.

This dialog can be used to switch the EtherCAT state of the slave device.

The slave state may not be "higher" than the master state, refer to chapter 8.4.5 "Tab “Bus diagnostics”" on page 203.

8.5.9 Tab “ESC register”

The tab is only displayed in logged-in state.

The dialog displays the register values of the EtherCAT slave controller (ESC).

The values are retrieved when opening the dialog and via the "Refresh" button. Selecting ("click") an address can additionally select a register for writing. Refer to the lower section of the dialog.

⚠ CAUTION

By reading and writing the registers, messages from the slave to the master can be acknowledged which are then not visible anymore for the master stack.

Thus, only experts should read/write registers.

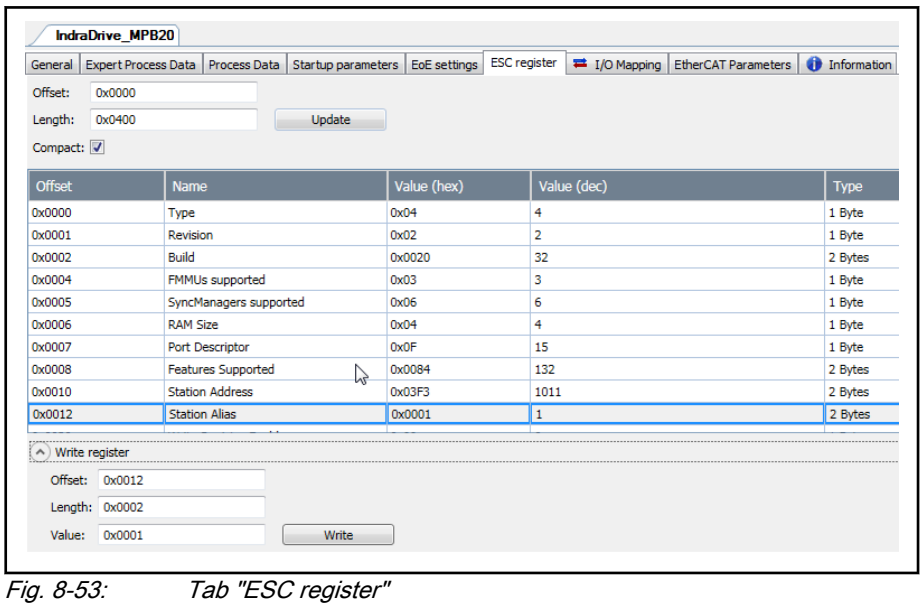


Fig. 8-53: Tab "ESC register"

8.5.10 Tab “CoE”

This tab displays information on the CoE objects of an EtherCAT slave with CoE profile.

The COE object descriptions (offline from ESI or online from device) are optional services and thus not supported by all CoE slaves.

Refer to the product description of the slave device.

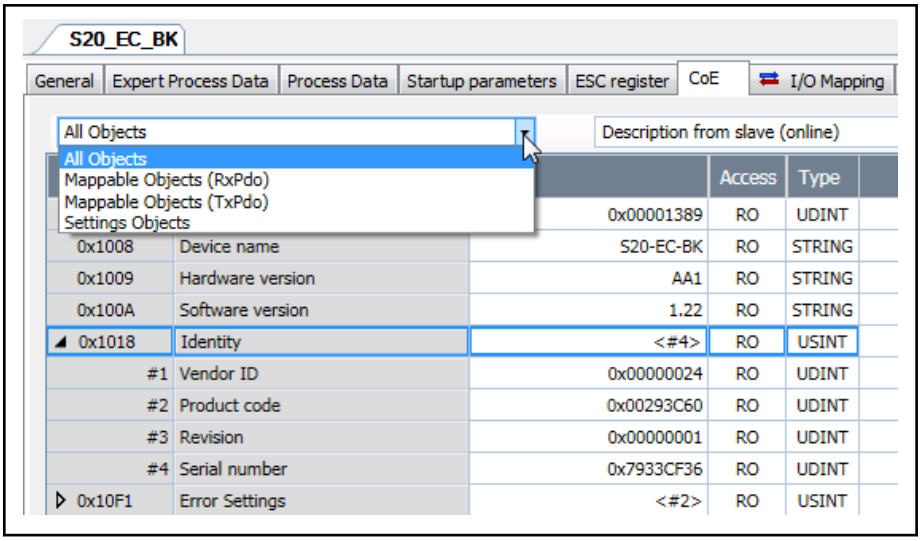


Fig. 8-54: Tab "CoE"

In the right upper drop-down list, select:

- whether the description from the device description (ESI)[2] of the slave is to be displayed or
- or whether the description is to be loaded from the device (online).

For a device with modular device profile, e.g. Rexroth S20-EC-BK, information on the physically existing modules is only available online.

It is required that the device supports the information retrieval (SDO information services).

Select the objects to be displayed in the left upper drop-down list [3] :

All Objects	(RxPDO): Objects that can be configured as process data (output data)
Mappable Objects	(RxPDO): Objects that can be configured as process data (output data)
Backup Objects	Objects saved in the slave for a device exchange
Settings Objects	Objects that can be configured as start parameters upon startup

In logged-in state, the current values of the slave device are displayed.

To **write** the objects, edit the respective field in the "Value" column. The selected object has to have "RW" access (RO: "read only" = read-only; RW: "read write" = read and write).

8.5.11 Tab “Diagnostics”

The tab is only displayed in logged-in state.

2) As an alternative to the ESI device description, a CANopen EDS file can also be selected. The button on the right of the drop-down list can be used to add an EDS file for this slave.

3) Only object groups provided with a description from the slave are displayed (e.g. ESI/online information can differ).

The CoE object "Diagnosis History" (0x10F3) of the slave device is displayed. This diagnostic object is an optional functionality and it is not supported by all CoE slaves.

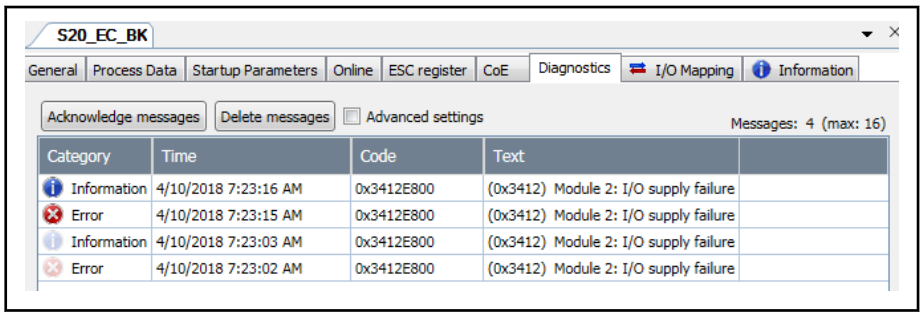


Fig. 8-55: "Diagnostics" tab

Category	Info/warning/error There is no default setting on how "incoming" and "outgoing" errors are reported. Refer to the product description of the slave device. Acknowledged messages are shown by a greyed out icon.
Time	The time stamp is reported by the slave: • it is "0" (= date 01.01.2000 00:00:00) if no time stamp is supported by the slave or • it is a local time stamp of the slave device (time converted to the PLC time is displayed) or • it corresponds to the "Distributed Clocks" if DC 64 bits is supported (e.g. corresponds to the time set on the control).
Code	Diagnostic code
Text	The diagnostic texts are obtained from the device description (ESI) of the slave device, i.e. the supported languages also depend on the slave device (e.g. mainly supported only in English).

Tab. 8-1: Column description of the "Diagnosis History"

Acknowledging/deleting messages

The diagnostic log is operated as ring buffer and can save up to 250 messages (device-specifically it might be less).

Messages can be acknowledged and deleted.

Extended settings

There are different optional settings that can be set in the slave. If a specific option is not supported by the slave, a writing error is reported while disabling/enabling (parameter value beyond the valid range).

☑ **Enable info, warning, error**: It can be disabled/enabled which diagnostic types are to be entered into the log.

☑ **Enable emergency**: It can be disabled/enabled whether the slave should sent new diagnostics to the master as CoE emergency messages.

☑ **Enable AcknowledgeMode**: It is distinguished between "OverwriteMode" and "AcknowledgeMode". In the "OverwriteMode", if the message buffer is full, a new message overwrites the oldest message in the buffer irrespective of whether the messages were acknowledged. Unacknowledged messages are not overwritten in the "AcknowledgeMode", i.e. if the message buffer is full, the latest messages are not saved and thus lost.

8.5.12 Tab “I/O mapping”

This tab assigns inputs and outputs of the EtherCAT slave to variables that can be used as local or global variables in the PLC.

The current value of the variables is displayed in logged-in state.

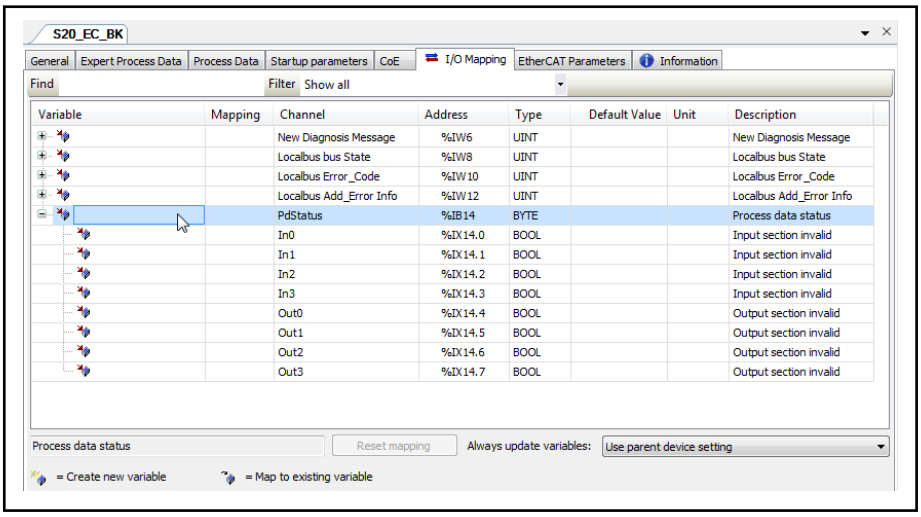


Fig. 8-56: Tab "EtherCAT I/O mapping"

Tab “EtherCAT I/O mapping”

Reset mapping	Use this button to reset the mapping settings to the default values (generally an empty mapping) defined in the device description file
Dialog Update variables	Specification for the device, whether the I/O variables are updated in the bus cycle task: • **Use parent device setting**: Update according to the settings of the parent device (refer to the PLC settings). • **Enabled 1 (use bus cycle task if not used in any task)**: The I/O variables are updated in the bus cycle task if not used in any other task. • **Enabled 2 (always in the bus cycle task)**: All variables are updated in each cycle of the bus cycle task irrespective of whether they are used or mapped on an input or output channel.

Process data status

The process data status "PdStatus" can be enabled in the master Tab “General”, page 199.

The “PdStatus” input shows the validity of the cyclic process data communication. The “PdStatus” can become invalid if

- there is not cyclic data exchange,
- a cyclic data telegram is not received,
- a cyclic telegram is received with invalid data (CRC error),
- the working counter of the cyclic command does not correspond to the expected value (also refer to basics chapter 8.2.8 "SyncUnits" on page 176).

The EtherCAT master supports slaves with up to four cyclic input sync managers and four cyclic output sync managers. Thus, there can be four input

data ranges and four output data ranges per slave. These are mapped by the four 4 In (In0..In3) or 4 Out (Out0..Out3) bits of the “PdStatus” respectively.

Currently, IndraWorks creates only configurations if the four possible input or output data ranges of a slave are grouped to one input data status bit In0 or output data status bit Out0. Thus, on In0 is applied to input data range and Out0 to the output data range.

If the process data exchange of the input data becomes invalid, the respective bit In0 becomes TRUE (TRUE = Process data status is invalid). Bit Out0 = TRUE signals that output data is not applied by the EtherCAT slave.

The “PdStatus” can be mapped to PLC variables like other input variables or it can be queried directly via the %IB address. In the PLC program, the process data exchange status can be checked as follows irrespective of the input and output data ranges.

```
IF PdStatus = 0 THEN
; //valid cyclic data exchange
ELSE
; // invalid cyclic data exchange
END_IF
```

Fig. 8-57: Querying the process data status

8.5.13 Tab “Information”

This tab displays some general information from the device description file (ESI) like name, vendor, category, version, description.

8.5.14 SoE parameter editor

In logged-in state, the parameter editor is provided in the context menu of a slave if it supports the mailbox protocol SoE, e.g. IndraDrive with SoE profile.

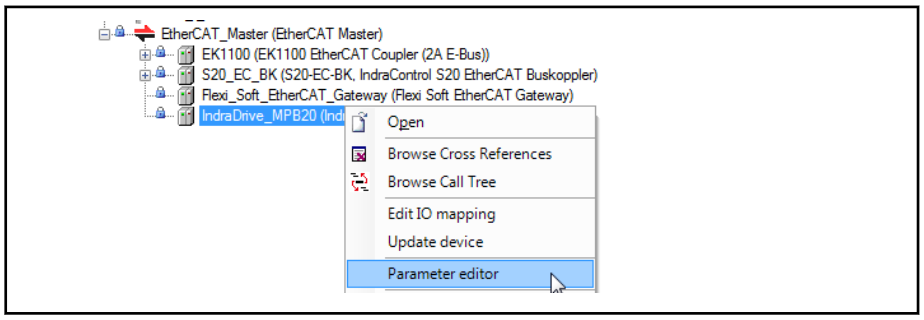


Fig. 8-58: SoE parameter editor call

The parameter editor allows an individual SoE parameter to be displayed and changed, i.e. only up to 16-bit S-/P-parameters can be edited (32-bit Sercos III parameter are not possible for SoE).

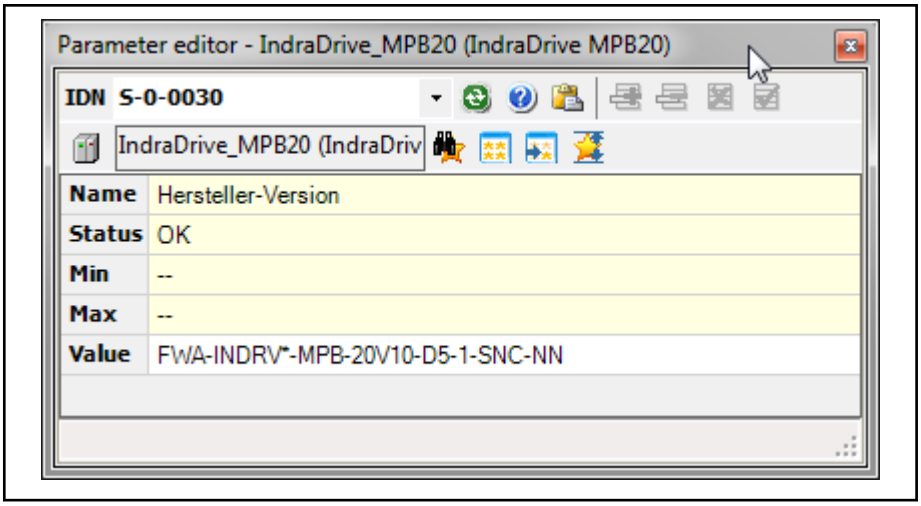


Fig. 8-59: SoE Parameter editor input

The functions "Find parameters" and "Parameter group" are not yet supported in the SoE parameter editor.

8.6 EtherCAT master, scanning bus

Function The "Scanning bus" dialog displays already configured device on the left and the devices currently found at the bus on the right.

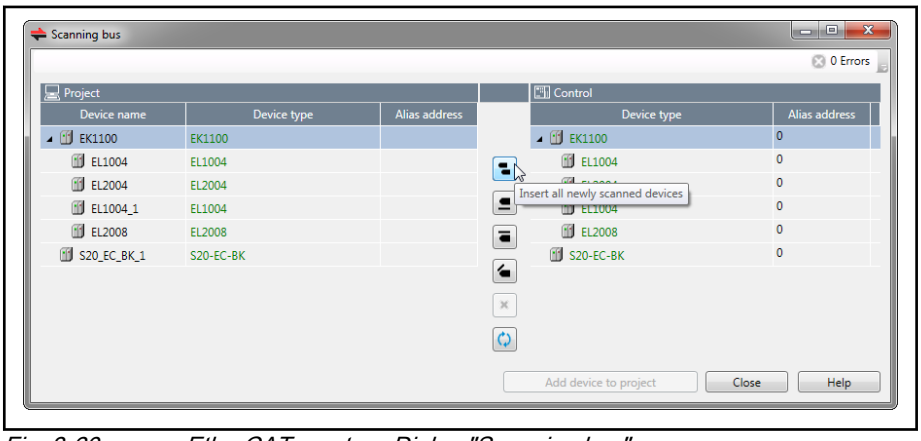


Fig. 8-60: EtherCAT master - Dialog "Scanning bus"

The device found (right) can be applied to the project configuration (left) using the button in the middle of the dialog. Either all new devices can be added or individual devices can be highlighted and edited (refer to the tooltip support of the buttons).

Call In the Project Explorer via the context menu of the EtherCAT master

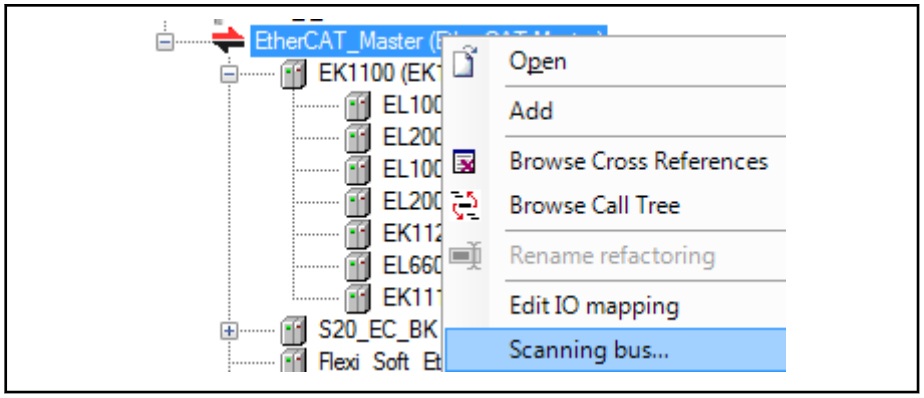


Fig. 8-61: EtherCAT master - "Scanning bus" call

8.7 EtherCAT master, bus-synchronous PLC processing

For an EtherCAT bus-synchronous PLC processing, a task with the external event "EVENT_OP_MODE_ETHERCAT_CYCLIC" has to be created in the task configuration of the PLC.

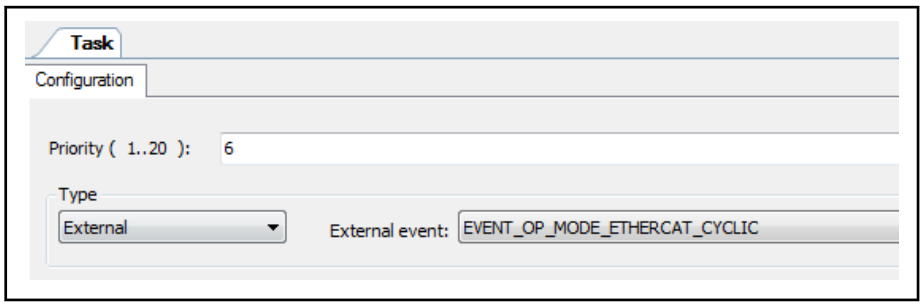


Fig. 8-62: EtherCAT task configuration

The POUs linked to this task are then called synchronously to the EtherCAT communication cycle. The following sequence applies:

- The EtherCAT stack is triggered by the system timer within the EtherCAT cycle time set.
- Cyclic input and output data is processed (receiving and sending of the EtherCAT telegram)
- The EtherCAT stack triggers the PLC task.

☞ It is currently not monitored whether the linked PLC task execution is completed upon the next EtherCAT event. The standard watchdog of the PLC task can be used to monitor the PLC task execution.

- Jitters in the PLC task execution do not affect the EtherCAT telegram transfer.

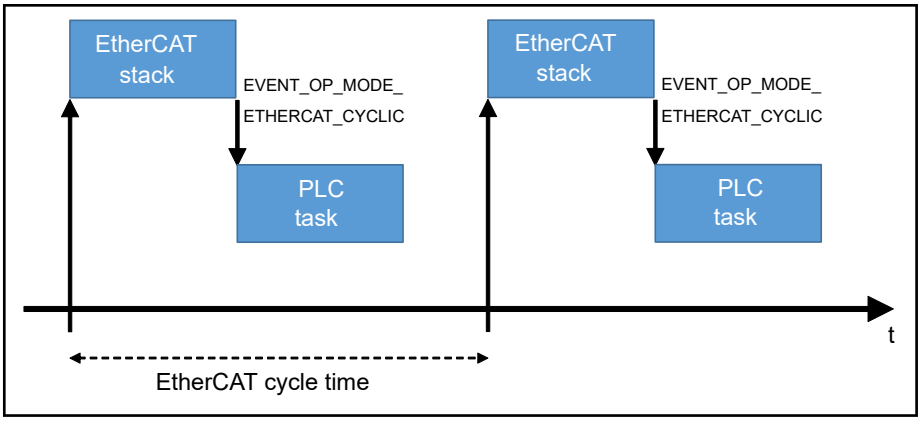


Fig. 8-63: EtherCAT task call

8.8 EtherCAT FAQs

8.8.1 Overview

There are notes on the following FAQs in the following chapters:

- IndraDrive: How to connect IndraWorks Ds via EtherCAT?, page 228
- IndraDrive: How to configure SoE using Free Run?, page 229
- S20-EC-BK: Bus coupler does not switch to OP (notes on the AL status codes), page 231
- S20-EC-BK: How to create a startup parameterization for S20 modules?, page 232

8.8.2 IndraDrive: How to connect IndraWorks Ds via EtherCAT?

IndraWorks Ds can set up a connection to IndraDrive at the Rexroth EtherCAT master either via MLPI (with SoE) or via EoE.

Variant A.) "IndraWorks Ds here ..." on slave nodes in IndraWorks

Opening "IndraWorks Ds here ..." in the context menu of the IndraDrive slave node below an EtherCAT master is only provided for IndraDrive with SoE profile.

An MLPI connection to the MLC is used. It communicates acyclically to the drives via the mailbox SoE.

For IndraDrive with CoE profile, the following variants C.) and D.) are recommended with EoE.

Variant B.) IndraWorks Ds with MLPI to the EtherCAT master

In IndraWorks Ds, select "MLC (EtherCAT)" as type in the “Selecting the connection” dialog in the “Third-party control unit” tab.

Enter the engineering port of the MLC control as IP address. IndraWorks Ds uses the connection to the MLC. It communicates acyclically with the drives via the mailbox SoE.



IndraWorks Ds with MLPI connection can only be used for drives with SoE profile. An EoE configuration or IP routing settings on the engineering PC are not required.

For IndraDrive with CoE profile, the following variants C.) and D.) are recommended with EoE.

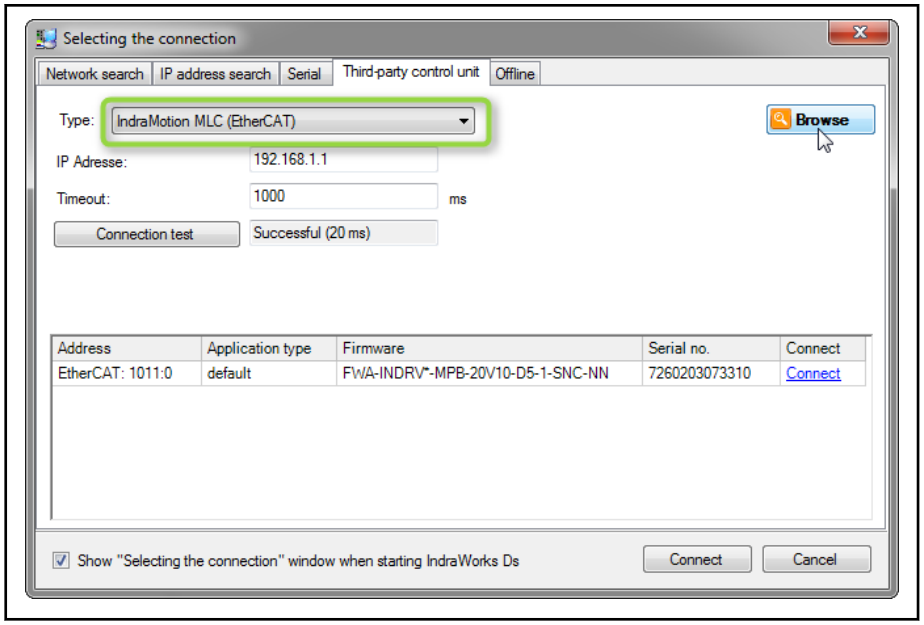


Fig. 8-64: Connecting IndraWorks Ds to the MLC control (EtherCAT)

Variant C.) IndraWorks Ds with EoE and switch port

With a switch port, e.g. Beckhoff EL6601, any Ethernet device can be decentrally connected to an EtherCAT network. An IP communication is thus possible with the EtherCAT slave.

For IndraDrive, an EoE configuration has to be set. In this configuration, the IP settings have to be in the same network as the those of the engineering PC, also refer to EoE configuration example, page 175.

In IndraWorks Ds, a connection can be established via network search or IP address search. An IP routing is not required on the engineering PC.

Variant D.) IndraWorks Ds with EoE and IP routing in the EtherCAT master

The engineering PC with IndraWorks Ds is connected to the engineering port of the control and the control executes an IP routing on EoE to the EtherCAT network.

For a valid IP configuration to be established in the EtherCAT network, set an EoE configuration for IndraDrive and EtherCAT master, refer to EoE configuration example, page 175.

Before establishing a connection in IndraWorks Ds via network search or IP address search, add a respective routing entry manually into the IP routing table of the engineering PC (e.g. refer to the Windows command line, command "route help").

8.8.3 IndraDrive: How to configure SoE using Free Run?

1. **Add IndraDrive as slave device**

   Create IndraDrive with a respective version MpxVyy below the EtherCAT master in the Project Explorer, for example MPB20:

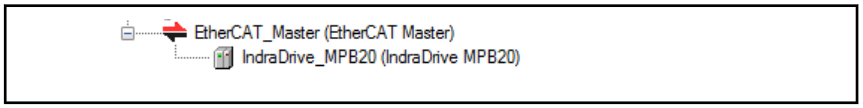


Fig. 8-65: IndraDrive as slave device

2. **Settings in the slave dialog "General"**

For the Free Run operation, select "Free Run" in the "Distributed Clock" setting. This is a setting value from the device description file of the slave (here: from IndraDrive MPB20).

Additionally, set the checkmark in front of "Enable expert settings" in this dialog to show the additional dialog "Expert process data".

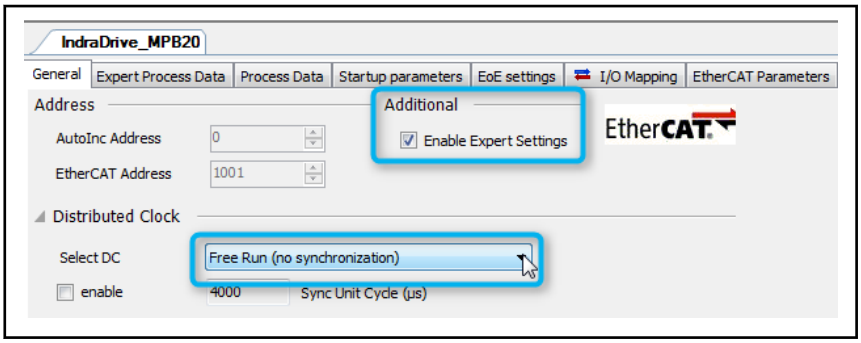


Fig. 8-66: IndraDrive configuration "General"

3. **Settings in the slave dialog "Expert process data"**

The requested process data for the drive can be configured manually in this dialog.

Select either S-0-0016 (AT) or S-0-0024 (MDT) in the field "PDO list" on the upper right. Then, the currently configured cyclic parameters are shown in the lower right "PDO content". Use the functions “Add/Edit/ Delete” in the bottom dialog on the right to manually configure the parameters for AT (drive telegram) or MDT (master data telegram).

If there are no predefined parameters in the device description of the slave (as shown in the following screenshot), name, parameter number (IDN) and data type have to be entered correctly and manually.

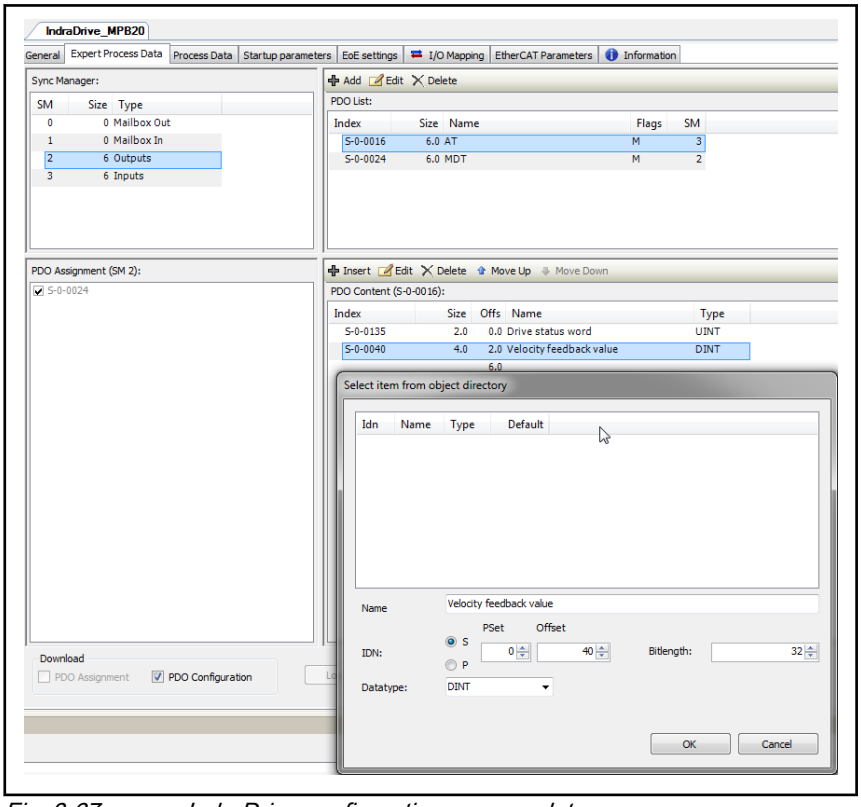


Fig. 8-67: IndraDrive configuration process data

4. **Settings in the slave dialog "Startup parameters"**

   For the Free Run operation, create the start parameter S-0-0002 ("Sercos cycle time") with the same value as S-0-0001 ("NC cycle time"). While S-0-0001 has already been predefined as start parameter via the device description, S-0-0002 has to be set manually as 16-bit parameter.

   The value of S-0-0001 corresponds to the EtherCAT cycle time and it is automatically set by the configurator. Set the same value for S-0-0002.

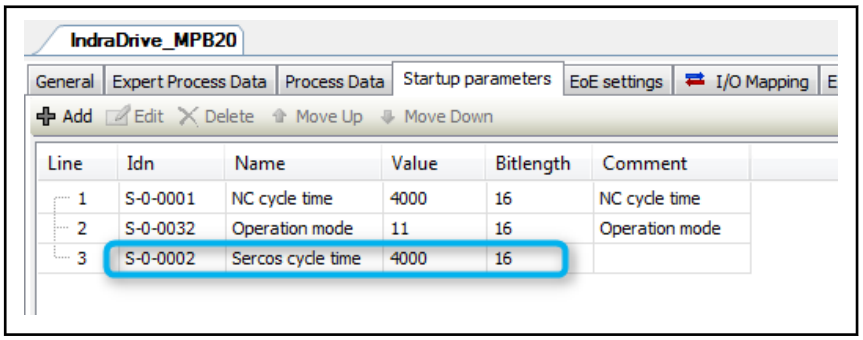


Fig. 8-68: IndraDrive configuration start parameters

5. **Expert settings "SyncManager"**

   The SyncManager configuration of SoE is set to "1 buffer" system by default. For the “Free Run” mode, the settings "3 buffers" is recommended. Refer to chapter 8.5.3 "Tab “SyncManager”" on page 213.

8.8.4 S20-EC-BK: Bus coupler does not switch to OP (notes on the AL status codes)

If the EtherCAT bus is switched to OP and if the S20-EC-BK does not reach the OP state, the bus coupler normally reports an AL status code. This AL status code is for example displayed in the EtherCAT master "Tab “Bus diagnostics”, page 203" and can have different causes:

AL status code	Description
0x001D / **0x001E**	Invalid Sync manger configuration (outputs)/ Invalid Sync manger configuration (inputs). Solution: These errors can for example be caused if the configured modules do not match the physical modules. For further information, refer to the AL status code 0x0070.
0x0028	Sync mode not supported The "DC Synchronous" mode is only supported if at least one S20 module is mounted at the bus coupler and if this module supports the local bus synchronization, e.g. S20-DI-16/1-HS. Refer to the data sheets of the S20 modules.

AL status code	Description
0x002B	No valid input and output data. Solution: Check the plug-in connections of the S20 bus base modules at the bus coupler and at the modules. Then, restart the bus coupler.
0x0070	Invalid module configuration. The configured modules (CoE object 0xF030) do not match the physical modules (CoE object 0xF050). Solution: 1. Check the module configuration in the IndraWorks configurator with the physical structure. 2. In the slave Tab “CoE”, page 221, the configured module numbers in the object 0xF030 can be compared to the physically found modules in the object 0xF050. More information on the existing modules can be retrieved from the objects 0x9yyy. 3. If no information is displayed on a physical module, check the plug-in connection of the S20 bus base modules. 4. If the physical structure has been modified or the plug-in connections were disconnected, the bus coupler has to be restarted.

8.8.5 S20-EC-BK: How to create a startup parameterization for S20 modules?

For a parameterization (incl. startup parameterization) of the S20 modules, refer to the following application description for S20-EC-BK:

http://www.boschrexroth.com/various/utilities/mediadirectory/index.jsp?publication=NET&search_query=**DOK-CONTRL-S20*EC*BK**-AP**

9 Sercos III I/O

9.1 Features of the Sercos III IO master

The current implementation of the Sercos III IO master includes the following functionalities:

- Cyclic data traffic
- SDDML file import into the device database
- Bus scan of Sercos III slaves
- Remote address allocation
- Automatic configuration of the devices in the device tree (offline/online adjustment of the devices in the project)
- Sercos III slave-related connection status in the interface
- Diagnostics for the Sercos III IO master and slaves via function block
- Acyclic data transfer (service channel or IP channel) via function block

Function/characteristics	Value
Maximum number of I/O slave, driver side[1)]	256
Baud rate	100 MBit/s
Auto negotiation/autocrossing	Yes
Min. cycle time	VEP: 1 ms
Max. amount of cyclic input data	5600 bytes
Max. amount of cyclic output data	5600 bytes
Max. number of cyclic input data per slave	5600 bytes
Max. number of cyclic output data per slave	5600 bytes
Max. amount of modules per slave	61
Max. acyclic telegram data per slave/telegram (MTU)	1500 bytes

Tab. 9-1: Technical data

9.2 Terms and abbreviations

Master The Sercos III field bus master is called **master**.

Slave The Sercos III field bus slaves are called **slaves**.

Sercos address The criterion for addressing a slave is the Sercos address.

SDDML file IndraWorks uses this device description to introduce Sercos III slaves (typically with I/O profile).

1) 256 I/O resources are supported by the driver. This number is reduced by individual system interfaces with regard to the control hardware.

9.3 Configuring Sercos III

9.3.1 Configuring Sercos III, overview

Sercos III is an IEC-compliant, open universal bus for Ethernet-based real-time communication. As universal bus, Sercos III is provided with communication channels and device profiles for all established automation applications.

Sercos III devices can be used with all controls supporting the Sercos master functionality. In IndraWorks, these controls are provided with the Sercos object (on board) node (function of a Sercos master).

- The master can be extended with slaves and modules. These can be configured later with regard to the device description file.
- Go to the Sercos object and click on the context menu item Scan Bus Configuration to identify all devices running at the Sercos bus and to apply them to the IndraWorks project.

Sercos III master

- Sercos III master, overview on bus addresses, page 235
- Sercos III master, cabling and redundancy (example: MLC system), page 243
- Tab “Settings”, page 245

Sercos III master; adding slave

- Sercos III master; adding slave, general information, page 247
- Enabling/disabling Sercos slaves, page 249
- Disabling Sercos slaves using UserDefs.cfg, page 250 only MTX system
- Tab “General”, page 252
- Tab “Sercos configuration”, page 254, this window is used for service purposes and is only visible if the option "Show generic configuration editors" is enabled under **Tools ▸ Options ▸ IndraLogic 2G ▸ Device editor**
- Tab “Status”, page 254
- Tab “Information”, page 254

Sercos III master; adding modules to slave

- Sercos III master, adding modules to slave, general, page 254
- Enabling/disabling Sercos modules, page 255
- Tab “Sercos module”, page 256
- Tab “Function groups”, page 256
- Tab “General inputs and outputs”, page 257
- Special tab Tab “User-defined parameters”, page 258
- Tab “Oversampling”, page 259
- Tab “Sercos III modules configuration”, page 261, this window is used for service purposes and is only visible if the option "Show generic configuration editors" is enabled under **Tools ▸ Options ▸ IndraLogic 2G ▸ Device editor**
- Tab “I/O mapping”, page 262
- Tab “Information”, page 262

9.3.2 Sercos III master, overview on bus addresses

The "Overview on bus addresses" dialog is used for a clear representation and editing of the addresses of the Sercos devices. It can be opened via the command "Overview on Bus Addresses" in the context menu of the Sercos node in the Project Explorer.

In the dialog, all Sercos devices are listed with their important properties in a table. Addresses can be edited after selecting the cell. Cells, which cannot be edited, are grayed out. The currently selected cell is highlighted in color.

Data in the table is checked for validity. Errors and warnings can result indicated by a respective icon in the first column. The tooltip on the icon shows the error and warnings. If at least one error is present, changes cannot be applied with **Apply** and the dialog cannot be closed with **OK**.

All changes are applied with **Apply** or **OK**.

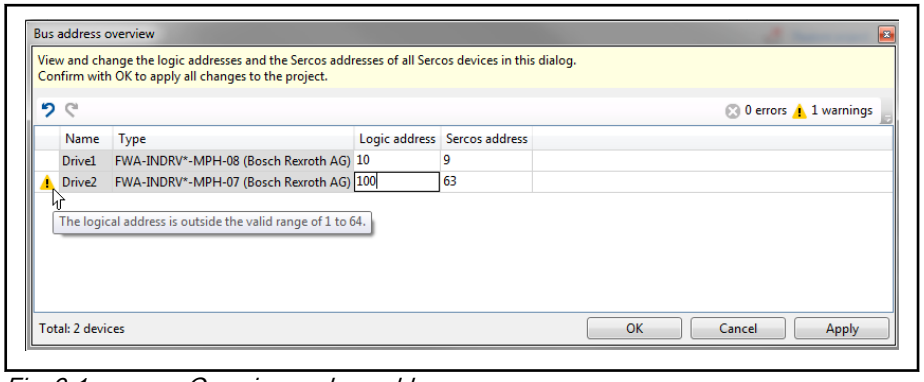


Fig. 9-1: Overview on bus addresses

There is a toolbar above the table. It provides the following functions:

Icon	Name	Hotkey	Meaning
	Undo	Ctrl+Z	If changes were made in the table, use this function to undo them step by step. This function is disabled after applying the changes with **Apply**
	Repeat	Ctrl+Y	If the changes were undone, they can be restored step by step using this function. This function is disabled after applying the changes with **Apply**
	Error		Data in the table is checked for validity. The number of errors found is displayed. The Sercos devices with errors are labeled by the same icon in the first column of the table. If at least one error is present, changes cannot be applied and the dialog cannot be closed with **OK**
	Warnings		Data in the table is checked for validity. The number of warnings found is displayed. The Sercos devices with warnings are labeled by the same icon in the first column of the table

Tab. 9-2: Toolbar functions

9.3.3 Sercos III master, Sercos bus diagnostics

Double-click on the Sercos master node in the Project Explorer to open the diagnostic window for the Sercos bus.

With regard to this diagnostics, all Sercos devices, IndraDrive as well as Sercos I/Os show a similar behavior:

For diagnostic purposes, Sercos devices are provided with:

- S-0-0390, Diagnostic message number
- S-0-0095, Diagnostic message, as text
- Sercos I/O devices have an additional I/O-specific diagnostic message (S-0-1500.0.32, I/O diagnostic message)

Class 1 errors can be acknowledged by executing the command S-0-0099, C0500 Reset class 1 diagnostics).

It shows detailed diagnostics from the reporting Sercos I/O device. For more information, refer to the device documentation.

For details, refer to The Sercos I/O status code can be found in: , page 238.

The display of the Sercos III diagnostic window is optimized with regard to the use case of the respective Bosch Rexroth control system. Thus, there are differences in details due to the respective control system used (MLC, MTX...).

In the following, the essential diagnostic steps are exemplarily shown using an MLC system.

1. In IndraWorks Engineering, open the dialog/tab "I/O Diagnostics" at the Sercos node in the respective project (double-click on the Sercos icon in the IndraWorks project tree).
2. Log in with the IndraLogic application (PLC logged in is a prerequisite for the next steps).
3. Select error-reporting Sercos I/O device
4. Click on **Extended device status**, **Device identification** or **Device communication**.

   Selection help

   4.1 **Extended device status**: Displays all general diagnostics and provides a good starting point

   4.2 **Device Identification**: Displays diagnostics of device/hardware design. This dialog includes important information on the initial commissioning or the remedy of module failures

   4.3 **Device communication**: Displays all Sercos connection-relevant diagnostics
5. **Extended device status ▸ Diagnostic number**

   Error type, error class and status code can be displayed by extending/ expanding elements.

   The status code is interpreted as text and is a good indicator.

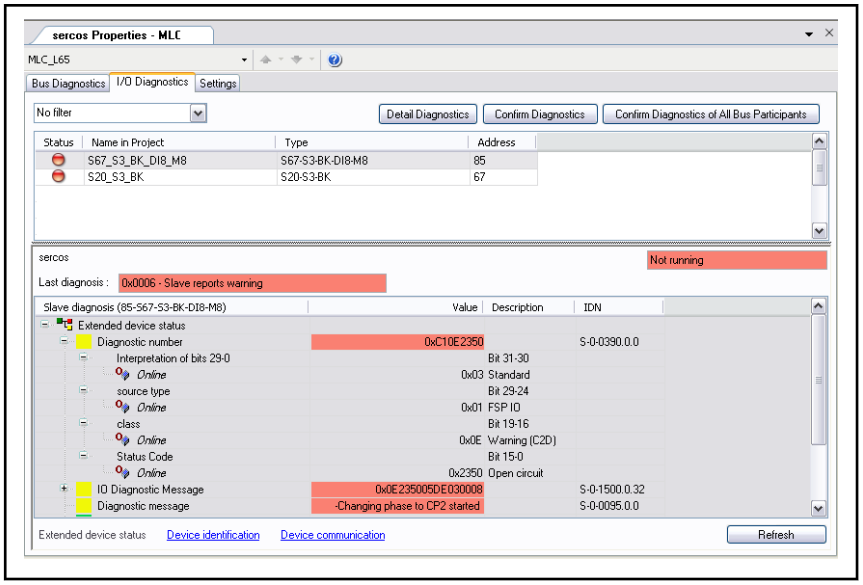


Fig. 9-2: Extended device status, status code

6. **Extended Device Status ▸ IO Diagnostic Message** supplements the **Diagnostic Number**. Information on the diagnostics relevant for the I/O device such as slot and channel number are provided. This identifies the module and its channel at a modular Sercos device.

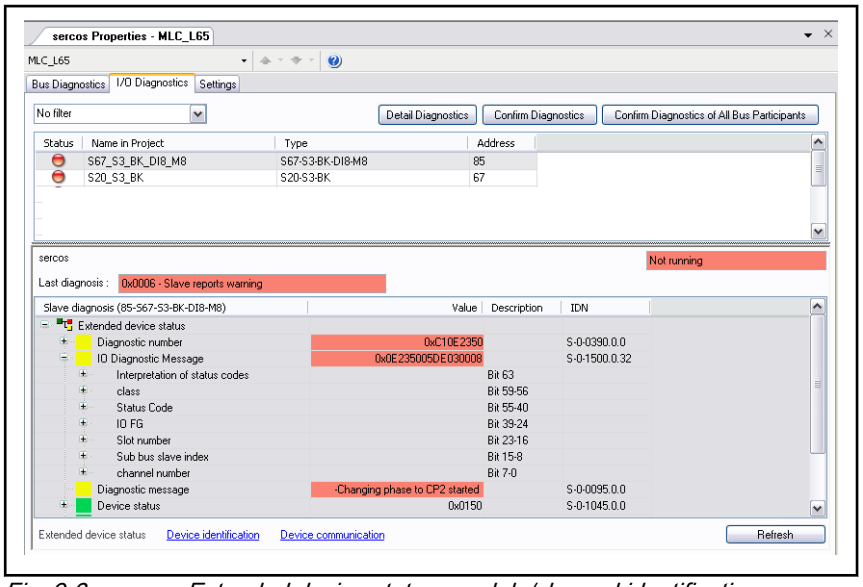


Fig. 9-3: Extended device status, module/channel identification

7. **Device identification** is important if the modules are installed or plugged in incorrectly or if they are defective.

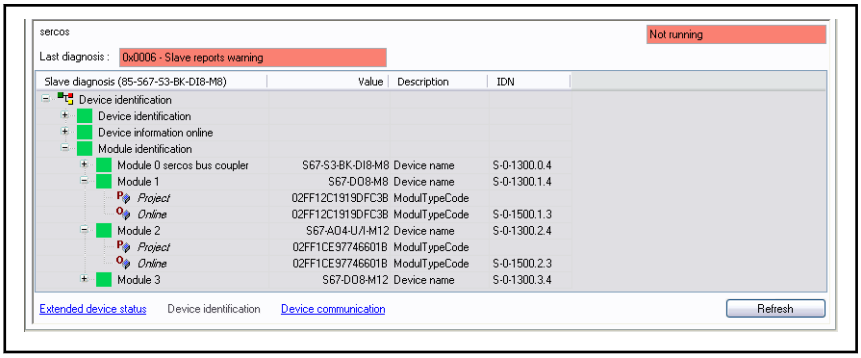


Fig. 9-4: Device identification; Module identification

The Sercos I/O status code can be found in:

- S-0-0390, Diagnostic message number

  Source type (bits 29-24) | class (bits 19-16) | status code (bits 15-0)

  e.g. 16#C10E2350 ⇒ C1 | 0E | 2350
- S-0-1500.0.32 I/O diagnostic message

  Class (bits 59-56) | status code (bits 55-40) | function group (bits 39-24) | slot number (bits 23-16) | sub-bus slave (bits 15-8) | channel number (bits 7-0)

  e.g. 16#0E235005DE010008 ⇒ 0E | 2350 | 05DE | 01 | 00 | 08

Value	Description
Status codes with the diagnostic classes Operating state (0xC10A), Warning (0xC10E), Error (0xC10F)	
16#0	No error
16#1000	General error
16#2000	Current
16#2100	Current on the side of the device input
16#2110	Short circuit/earth fault
16#2120	Earth fault
16#2130	Short circuit
16#2136	Short circuit at VCC
16#2137	Short circuit to earth
16#2200	Current within the device
16#2213	Overcurrent upon startup
16#2214	Overcurrent at runtime
16#2220	Permanent overcurrent
16#2230	Short circuit/earth fault
16#2240	Earth fault
16#2250	Short circuit
16#2300	Current on the side of the device output
16#2310	Permanent overcurrent
16#2320	Short circuit/earth fault

Value	Description
Status codes with the diagnostic classes Operating state (0xC10A), Warning (0xC10E), Error (0xC10F)	
16#2330	Earth fault
16#2340	Short circuit
16#2344	Output overloaded
16#2345	Sensor supply overload
16#2346	Short circuit at VCC
16#2347	Short circuit to earth
16#2350	Wire break
16#2360	Wire break
16#3000	Voltage
16#3100	Mains voltage (voltage at the side of the device input - voltage supply)
16#3110	Mains overvoltage
16#3120	Mains undervoltage
16#3130	Phase error
16#3134	Phase sequence
16#3140	Mains frequency
16#3141	Mains frequency too high
16#3142	Mains frequency too low
16#3200	Voltage within the device
16#3210	Overvoltage within the device
16#3220	Undervoltage within the device
16#3230	Loading error
16#3300	Output voltage (voltage at the side of the device output, e.g. supply and operating voltage for external components)
16#3310	Output overvoltage
16#3320	Output undervoltage
16#3400	Supply voltage
16#3410	Sensor supply
16#3411	Sensor supply, undervoltage
16#3412	Sensor supply missing
16#3420	Actuator supply
16#3421	Actuator supply, undervoltage
16#3422	Actuator supply missing
16#4000	Temperature
16#4100	Ambient temperature
16#4110	Ambient overtemperature

Value	Description
Status codes with the diagnostic classes Operating state (0xC10A), Warning (0xC10E), Error (0xC10F)	
16#4120	Ambient undertemperature
16#4130	Temperature intake air
16#4140	Temperature suction air
16#4200	Device temperature
16#4210	Device overtemperature
16#4220	Device undertemperature
16#4300	External temperature (e.g. of the drive)
16#4310	External overtemperature (e.g. of the drive)
16#4320	External undertemperature (e.g. of the drive)
16#4400	Connection temperature
16#4410	Connection overtemperature
16#4420	Connection undertemperature
16#5000	Device hardware (only in the device)
16#5010	Component defect
16#5100	Supply
16#5110	Low-voltage supply (for all internal voltages 24 V, 5 V, 3.3 V, 3.0 V, 2.5 V, 1.8 V)
16#5120	Air supply
16#5150	Initiator supply (fault in supplying an external sensor supply from within the device)
16#5151	Short circuit
16#5160	Supply of the peripheral devices (supply fault for external devices supplied from within the device)
16#5200	Control unit/measuring circuit
16#5210	Measuring circuit
16#5220	Computer circuitries
16#5230	Communication (within the device)
16#5300	Operating unit
16#5400	Power section
16#5410	Output states
16#5420	Chopper (interrupter)
16#5430	Input stages
16#5440	Contactor
16#5450	Fuses
16#5500	Communication with extension modules
16#6000	Device software
16#6010	Software reset (watchdog)
16#6100	Internal software (firmware)

Value	Description
Status codes with the diagnostic classes Operating state (0xC10A), Warning (0xC10E), Error (0xC10F)	
16#6200	Application software
16#6300	Data set incorrect
16#6310	Missing parameter
16#6320	Parameter error
16#6330	Parameter not yet initialized
16#6800	Configuration
16#7000	Extension module(s) defective
16#7100	Performance
16#7200	Measuring circuit
16#7300	Sensor
16#7400	Computer circuitry
16#7500	Communication
16#7600	Data memory
16#7610	RAM
16#7620	EPROM
16#7630	EEPROM
16#7700	Line break/cable fault
16#7710	Wire break between cable and sensor
16#8000	Monitoring
16#8100	Communication
16#8110	Process data monitoring
16#8120	Host monitoring
16#8200	Closed-loop control
16#8210	High system deviation (command > actual, manufacturer-specific), (actual deviation more than permitted)
16#8211	Maximum control value reached
16#8220	System deviation (command > actual, manufacturer-specific), (actual deviation more than permitted)
16#8221	Maximum control value reached
16#8900	Sensors
16#8910	Measured value above the value range
16#8920	Measured value below the value range
16#8A00	Actuator
16#8B00	Preventative maintenance required (state monitoring)
16#9000	External error
16#A000	Subbus slave

Value	Description
Status codes with the diagnostic classes Operating state (0xC10A), Warning (0xC10E), Error (0xC10F)	
16#A001	Subbus slave missing
16#A002	Wrong subbus slave present
16#A003	Compatible substitute (expected subbus slave replaced by compatible one)
16#A004	Error in the number of subbus devices (too many devices)
16#A010	Device error
16#A012	Application on subbus slave not ready
16#A013	Subbus device reset
16#A020	Communication error
16#A021	Subbus error - Timeout (break)
16#A022	Multiple transmission error on subbus
16#A023	Subbus I/O data communication error
16#A024	Data communication fault in subbus management
16#A030	Subbus configuration error
16#A031	I/O data configuration mapped twice
16#A040	General error
16#A041	Subbus hardware error
16#A042	Subbus firmware error
16#A043	Subbus runs asynchronously (with regard to the Sercos bus cycle)
16#B000	Local bus (system bus of the modular station)
16#B001	Local bus slave missing
16#B002	Wrong local bus connected
16#B003	Compatible substitute (expected local bus slave replaced by compatible one)
16#B004	Error in the number of local bus devices (too many devices)
16#B010	Device error
16#B012	Application on local bus slave not ready
16#B013	Local bus device reset
16#B020	Communication error
16#B021	Local bus error - Timeout (interruption)
16#B022	Multiple transmission error on local bus
16#B023	Local bus I/O data communication error
16#B024	Data communication error in local bus management
16#B030	Configuration error on the local bus
16#B031	I/O data configuration mapped twice
16#B040	General error
16#B041	Hardware error on the local bus

Value	Description
Status codes with the diagnostic classes Operating state (0xC10A), Warning (0xC10E), Error (0xC10F)	
16#B042	Firmware error on the local bus
16#B043	Local bus runs asynchronously (with regard to the Sercos bus cycle)
16#F000	Additional functions

Value	Description
Status codes with diagnosis class "procedure command specific state" (0xC10C)	
16#A100	S-0-1508.x.67 Sub-bus Tunnel Command
16#A101	S-0-1508.x.67 Sub-bus Tunnel Command failed. Unsupported Sub-bus Tunnel PDU version
16#A102	S-0-1508.x.67 Sub-bus Tunnel Command failed. Unsupported Sub-bus SDU Type
16#A103	S-0-1508.x.67 Sub-bus Tunnel Command failed. Unsupported Sub-bus Tunnel SDU version
16#A104	S-0-1508.x.67 Sub-bus Tunnel Command failed. Sub-bus slave index wrong
16#A105	S-0-1508.x.67 Sub-bus Tunnel Command failed. Wrong length of SDU
16#A106	S-0-1508.x.67 Sub-bus Tunnel Command failed. Wrong length of Command SDU
16#A107	S-0-1508.x.67 Sub-bus Tunnel Command failed. Answer PDU too large
16#A108	S-0-1508.x.67 Sub-bus Tunnel Command failed. Buffer overflow
16#BC00	S-0-1500.x.12 Rearrangement of IO resource
16#BC01	Forbidden insertion/replacement - An attempt was made to insert/replace a generic IO function group with an invalid/forbidden slot number
16#BC02	Forbidden insertion/replacement - An attempt was made to insert/replace the function group IO bus coupler (S-0-1500.x.0)
16#BC03	Forbidden insertion/replacement - An attempt was made to insert/replace an undefined generic IO function group
16#BC04	Forbidden insertion/replacement - The list of inserted/replaced function groups (S-0-1500.x.4/11) contains duplicate entries
16#BC05	Forbidden insertion/replacement - The list of inserted/replaced function groups (S-0-1500.x.4/11) is not sorted in an increasing order (slot number, data block number)
16#BC0F	Forbidden insertion/replacement - Manufacturer-specific reason

9.3.4 Sercos III master, cabling and redundancy (example: MLC system)

To increase the fault tolerance with regard to EMC tolerance and defective cables, use a Sercos ring topology. This creates a redundant Sercos communication with all Sercos devices. If the ring is interrupted, connection data directly exchanged between the individual Sercos devices is copied via the master. Connection data from the master to the Sercos devices and vice versa are not affected.

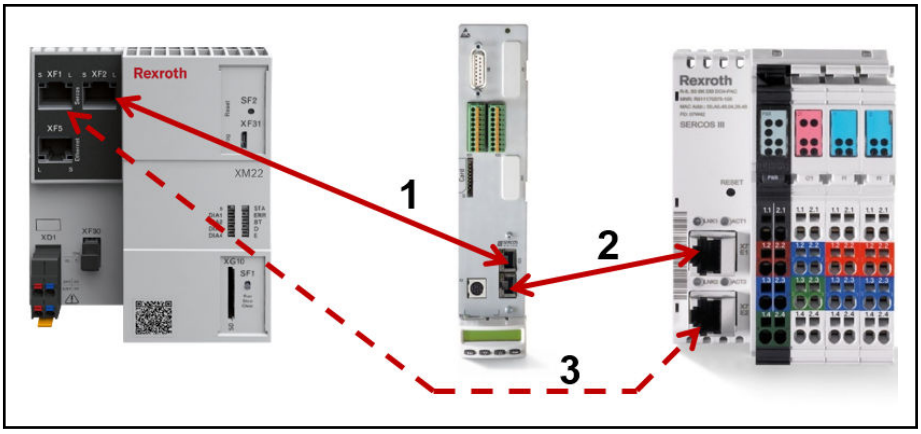


Fig. 9-5: Wiring ring topology

- **Ring topology:** Wiring the cables 1 and 2 and the optional cable 3 creates a redundant ring topology.
- **Line topology:** Only the cables 1 and 2 are installed. No redundancy present.
- **Double line topology:** Only the cables 1 and 3 are installed. No redundancy present as well.

Bus diagnostics, ring break

In the initial state, the MLC Sercos master expects a ring topology. If the ring topology is not detected in the Sercos phase startup or interrupted at runtime due to malfunctions, the warning "E00B0179, sercos III: redundancy loss" is output. The ring interruption is displayed in the "Sercos bus diagnostics" dialog.

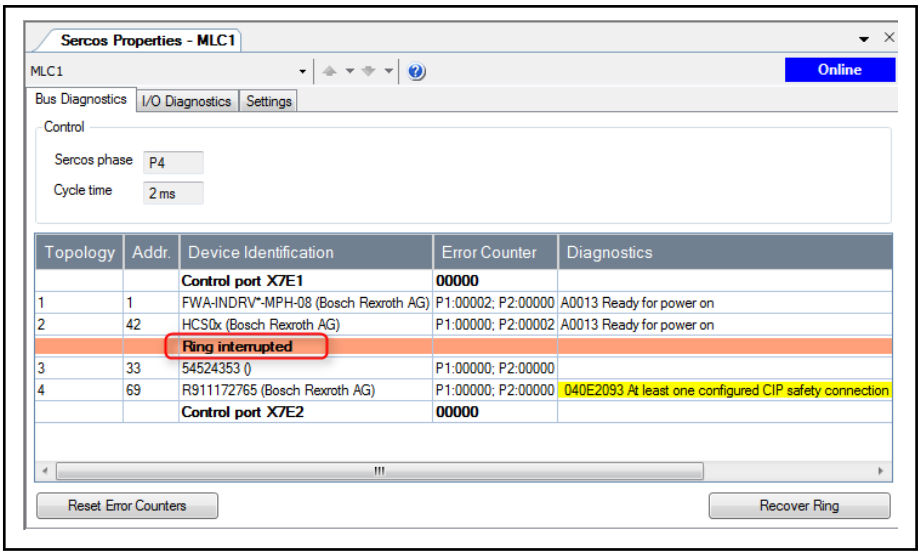


Fig. 9-6: Bus diagnostics for double line topology (figure above: Cable 3 missing)

The <Reset Error Counters> button deletes the displayed error counters of all devices.

The "Error Counter" column displays the parameter S-0-1035.0.0 "Error counter port 1 and port 2" for Sercos devices. Error counter Port2 is on the left and error counter Port1 on the right of the semicolon. The low word of the parameter C-0-0535, sercos master, error counter for Port1 and Port2 is displayed in the **Control port X7E1** line and the high word of C-0-0535 in the **Control port X7E2** line.

To logically close the ring after the interruption, press the <Recover Ring> button.



If the ring interruption was caused by a failing device, the ring cannot be closed. Only when switching phases from CP0 to CP2, the failed device is included again in the communication. The ring is closed again upon phase switching.

If a line or double-line topology is to be used, the warning can be disabled in the "Settings" dialog. The "Settings" dialog is only enabled if switched to online.

Tab “Settings”

The dialog allows the specification

- of a permissible number of telegram failures
- as well as the Sercos cycle time for the complete Sercos object

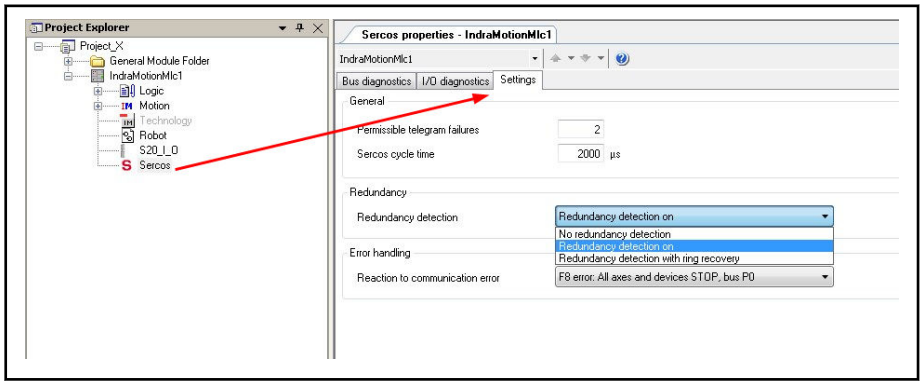


Fig. 9-7: Sercos settings

Sercos redundancy settings

- **No redundancy detection:** The warning "E00B0179, sercos III: redundancy loss" is not output. The status information is still generated and the ring can be recovered by the user.
- **Redundancy detection on (default):** The warning "E00B0179, sercos III: redundancy loss" is not output at ring interruption. The user can recover the ring.
- **Redundancy detection with ring recovery:** The warning "E00B0179, sercos III: redundancy loss" is not output at ring interruption. It is automatically tried to close the ring again (Restoring ring, page 246).

Sercos settings for the error reaction to communication errors

- **F8 error: All axes and devices STOP, bus P0** (default setting. Compatible to the previous versions and Sercos 2). If there are communication errors (e.g. invalid AT data), an F8 error is output (e.g. F80B0016, Invalid AT data) and the Sercos bus is thus stopped with all connected devices
- **F5 error: All axes are devices STOP**. An F5 error (e.g. F50B017A, Invalid AT data) is output if there are communication errors. All motion axes and the respective device are stopped with the F5 error. The Sercos bus remains in operating or parameterization mode. PLC devices at the Sercos bus are still ready. An error reaction on a device failure might have to be implemented in the PLC program if required.



After deleting the F5 error, the axes - that are not involved - can be traversed again. Especially in case of coupled axes, implement an error reaction on a device failure in the PLC program.

- **F2 error: Affected device STOP**. An F2 error (e.g. F20B017D, Invalid AT data) is output if there are communication errors. This F2 error only stops the corresponding device. The Sercos bus remains in operating or parameterization mode. Motion axes and PLC devices remain ready at the Sercos bus. An error reaction on a device failure might have to be implemented in the PLC program if required.

  If a real axis, which is consumed by another axis, fails, an error also occurs at the consuming axis (invalid master axis data).

Restoring ring

If line end(s) were physically correctly connected to the free master port, the Sercos topology still has to be closed logically to ensure that the communication remains established when there is an interruption at another location.

Ring recovery can be started via the "Sercos Bus Diagnostics" dialog, the **Recover Ring** button or

with a rising edge at

`ControlData[LOCAL_CNTRL].arSercos[ML_LOCAL_SERCOS_IFC].REBUILD_RING`

from the PLC.

The ring is successfully recovered if

`ControlData[LOCAL_CNTRL].arSercos[ML_LOCAL_SERCOS_IFC].ERROR_RED` becomes 0 again.

The error message "F00B0173, Ring topology could not be established" is output if ring recovery is impossible.

Cause	Solution:
Cables between the device reported in the parameter "C-0-0517, sercos master, addresses ring break" is either not available or incorrect	Check cabling
A port of the devices reported in "C-0-0517, sercos master, addresses ring break" is defective	Exchange device
The communication phase is lower than CP2	Switch at least to CP2
At least one device did not apply the specified topology	Check diagnostics of devices
New ring delay could not be applied to all Sercos devices	Check diagnostics of devices
The last slaves of the lines reported an incorrect topology. There is either no telegram or a wrong telegram at the disabled port	Check diagnostics of devices

PLC interface

- PLC input `ControlData[LOCAL_CNTRL].arSercos[ML_LOCAL_SERCOS_IFC].ERROR_PORT1`: Malfunction at X7E1 of the control.
- PLC input `ControlData[LOCAL_CNTRL].arSercos[ML_LOCAL_SERCOS_IFC].ERROR_PORT2`: Malfunction at X7E2 of the control.

- PLC input `ControlData[LOCAL_CNTRL].arSercos[ML_LOCAL_SERCOS_IFC].ERROR_RED`: No redundancy.
- PLC output `ControlData[LOCAL_CNTRL].arSercos[ML_LOCAL_SERCOS_IFC].REBUILD_RING`: Trigger ring recovery at rising edge.
- PLC output `ControlData[LOCAL_CNTRL].arSercos[ML_LOCAL_SERCOS_IFC].ACTUAL_PHASE_BIT0`: Sercos phase CP0 is active.
- PLC output `ControlData[LOCAL_CNTRL].arSercos[ML_LOCAL_SERCOS_IFC].ACTUAL_PHASE_BIT1`: Sercos phase CP1 is active.
- PLC output `ControlData[LOCAL_CNTRL].arSercos[ML_LOCAL_SERCOS_IFC].ACTUAL_PHASE_BIT2`: Sercos phase CP2 is active.
- PLC output `ControlData[LOCAL_CNTRL].arSercos[ML_LOCAL_SERCOS_IFC].ACTUAL_PHASE_BIT3`: Sercos phase CP3 is active.
- PLC output `ControlData[LOCAL_CNTRL].arSercos[ML_LOCAL_SERCOS_IFC].ACTUAL_PHASE_BIT4`: Sercos phase CP4 is active.
- PLC output `ControlData[LOCAL_CNTRL].arSercos[ML_LOCAL_SERCOS_IFC].PHASE_SWITCH_ACTIVE`: Sercos phase switching running.

Parameters involved

- C-0-0506, sercos Master, sercos III, configuration of function modes (bit 7 and bit 8)
- C-0-0507, sercos master, allowed telegram losses
- C-0-0511, sercos master, binary inputs, command values (bit 4)
- C-0-0514, sercos master, status (bits 1 to 3)
- C-0-0517, sercos master, addresses ring break
- C-0-0518, sercos master, position ring break

9.3.5 Sercos III master, adding slave

Sercos III master; adding slave, general information

The slaves are located in the "Periphery" library in the "Sercos" folder.

As slaves, compact I/O modules and bus terminals with fixedly specified I/O channels can be used if required.

Drag the required slaves from the library to the Sercos object.

In the Project Explorer, slaves can thus also be inserted between existing slaves.

Slaves can also be added in the context menu via **Add ▸ ...** of the Sercos object.

In this case, the new device is added as last module below the Sercos object.

It is also possible to click on the context menu item Scan bus configuration to identify all devices running at the Sercos bus and to apply them to the IndraWorks project.



If a required slave is not in the library by default, it can be integrated into the library by importing its device description file via the main menu item **Tools ▸ Device database...**.

Sercos III differentiates two types of slaves to connect I/O modules

1. **Compact**:

   For a compact slave, the module structure is fixedly specified.

   In case of compact design, the modules below the "Slave" object node are already present after adding a slave to the Project Explorer. The modules are not visible in the library.

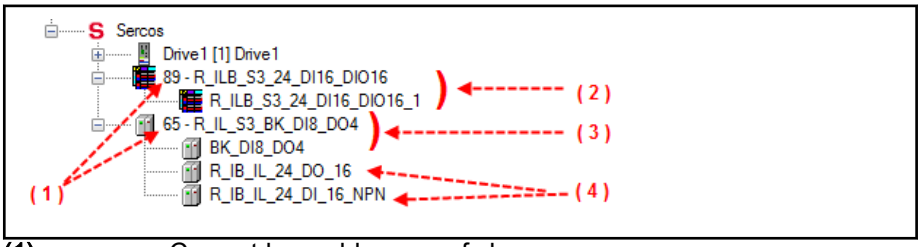
2. **Modular**:

   The module structure of the slave is variable.

   In addition to a fixed part - in the figure below "BK_DI8_DO4_1" with 8 digital inputs and 4 digital outputs - further modules can be added.

   The modules can be freely arranged, but taking the mounting specification of the device into account.

   For information on how to insert modules, refer to Sercos III master, adding modules to slave, page 254.



(1) Current bus addresses of slaves
(2) Compact slave with fixedly assigned inputs and outputs
(3) Modular slave with fixedly assigned inputs and outputs
(4) Output and input terminals attached to the bus coupler

Fig. 9-8: Slaves at the Sercos III master object

With Sercos III, a drive can be added as PLC device:

In contrast to an axis, a drive as PLC device has no motion functionality and no A-parameters.

The drive is PLC-commanded and freely programmable.

For Sercos, an IndraDrive single-axis device for MPx18VRS and higher is provided as HCS0x in the peripherals "Sercos" (IndraDrive double-axis devices as HMD01).

For a detailed description of the "drive as PLC device", refer to:

Rexroth IndraLogic MLC 15VRS Functional Description

DOK-MLC***-FUNC****V15-APRS-EN-P, R911393305

This documentation describes wizards, context menus, dialogs, control commissioning, device configuration and functionalities of the MLC system.

Overview on Sercos addresses...

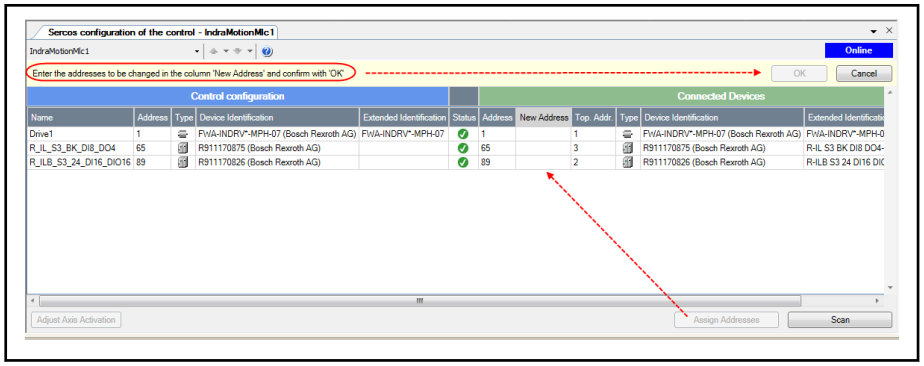


Fig. 9-9: Overview on Sercos addresses

The Sercos bus can only be scanned in P2, that is P4 -> P0 -> P2 -> P4.

Enabling/disabling Sercos slaves

The Sercos slaves configured at a control can be enabled and disabled in the Project Explorer.

Thus, select/deselect the button at the icon of the Sercos slave.

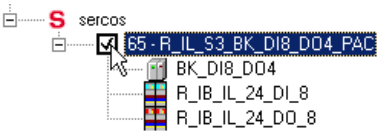


Fig. 9-10: Enabling/disabling Sercos slaves

The bus device is enabled if the checkbox is selected. It is considered in the diagnostics of the bus.

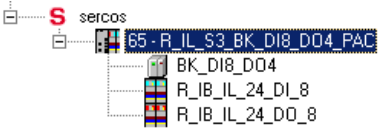


Fig. 9-11: Sercos slave as enabled device

The bus device is disabled if the checkbox is deselected. It is not considered in the diagnostics of the bus.

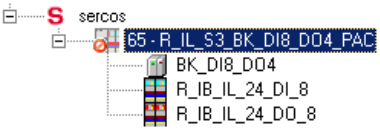


Fig. 9-12: Sercos slave as disabled device

If a new Sercos slave is created, it is enabled by default.

When loading the configuration, all enabled Sercos slaves are taken into account. Disabled Sercos slaves are ignored.



An activated, but non-existing bus device inhibits the bus switching to P4. The device configuration is compared.

Also refer to Sercos III master, overview on bus addresses, page 235.

A disabled and not present bus device does not generate any error message in the diagnostics.

Also refer to Disabling Sercos slaves using UserDefs.cfg, page 250

Disabling Sercos slaves using UserDefs.cfg

Use the function to disable Sercos slaves or Sercos modules configured in IndraWorks. Disable them by the definition in a stored file located on the Compact Flash card of the control.

This inhibits diagnostic messages for Sercos slaves or Sercos modules that are configured, but do not exist.

Enabling the function

The function is enabled if the valid file "UserDefs.cfg" is present in the following partitions on the control Flash Card:

For XM and VPx controls: **\USER**

Another path or file name can be defined by making entries in the "cfg file" of a system.

Example:

```
[IOSpecialDefs]
IODeactFilePath=MyFilePath
IODeactFileName=MyFileName
```



The file "UserDefs.cfg" can contain entries for more buses:

Inline modules, Sercos, Profinet I/O, Profibus DP.

Deactivation point in time

The "UserDefs.cfg" file is analyzed and the entered Sercos slaves / Sercos modules are disabled at the moment and in the context of the internal event "PrepareUpdateConfiguration".

Event takes place in case of the following actions:

- Startup after power on
- Reset (warm)
- Reset (cold)
- Load program after reset (origin)

Structure of the file "UserDefs.cfg"

The file is structured in a single or multiple existing section(s) and the contained entries.

- **Sections**: The name of a section is written in **[** and **]**. A section ends at the beginning of the next section or at the end of the file.

  The name consists of a symbol **SercosIO.Master.** and the constant index **0** (compatible to the Profibus slave deactivation).

  Example:

  ```
  [SercosIO.Master.0]
  ```

- **Special function**: Due to the definition of character **X** for the index instead of a digit, this section is ignored in the evaluation.

  Several sections can be defined for the master index **0**. All their entries are assigned to the Sercos master.

Together with the "x-function" described above, different slave groups can be handled easily.

- **Slave entry**: An entry to disable a Sercos slave consists of the keyword with assignment character `SlaveDeactAddr=` and the 1- or 2-digit decimal specification of the slave address.

  Example:

  ```
  SlaveDeactAddr = 28
  ```

  The permitted values range is **1…255**.

  The value **0** is also valid, but not effective.

- **Module entry**: To disable modules below a slave,

  the keyword `SlaveAddr=` and the Sercos address as well as

  the keyword `DeactModule=` and - separated by comma - the module numbers to the disabled from **1** to **n** are specified.

  Module number **0** is the bus connection of the slave and may not be specified here (it is also not reasonable to disable removable modules).

  Example:

  ```
  SlaveAddr = 3, DeactModule=3,4,7
  ```

- **Last module entries**: An entry to disable the **n** latest configured modules below a slave consists of

  the keyword `SlaveAddr=` and the slave address as well as

  the keyword `DeactLastModules=` and the number of modules to be disabled.

  This disables the last (lowest) modules in the configuration tree.

  Example:

  ```
  SlaveAddr = 3, DeactLastModules=2
  ```

- **Comments**: A semicolon (;) is defined as comment character. The text is not analyzed from the position of the semicolon up to the end of the line.

Size of the "UserDefs.cfg" file

The memory evaluation is allocated according to the specified file size. The permitted size is thus virtually not limited.

Example of the file "UserDefs.cfg"

```
;------------ Master.0 -----------
[SercosIO.Master.0]
SlaveDeactAddr= 20
SlaveDeactAddr= 32
SlaveDeactAddr= 35
SlaveDeactAddr= 56
;---------------------------------
SlaveAddr= 40, DeactModule= 2, 4
SlaveAddr= 55, DeactLastModules= 3
;---------------------------------
```

Commissioning Sercos IO devices/modules at the Sercos Bus

Reparameterizing Sercos IO devices/modules is only partially possible in the Sercos phase P2 and thus a recompilation and reloading of the PLC.

Changing or adapting to Sercos IO devices/module requires the following steps (recommendation):

- Switch Sercos bus to phase P2 (always via P0!)
- Go offline with the control
- Log out with PLC
- Make changes/adaptations at the Sercos IO device/module

- Go offline with the control (configuration adjustment if required)
- Compile and log in with PLC
- Switch Sercos to phase P4

Sercos III master, configuring Sercos III slave

To open the editor window in the Project Explorer, double-click on the Sercos III object.

The tabs of the generic device editor provide information on the configuration of the entire slave. Modifications can be made.



Change the address of a Sercos III slave via the context menu item **Configuration Sercos devices** of the Sercos III node (Overview on Sercos addresses..., page 249).

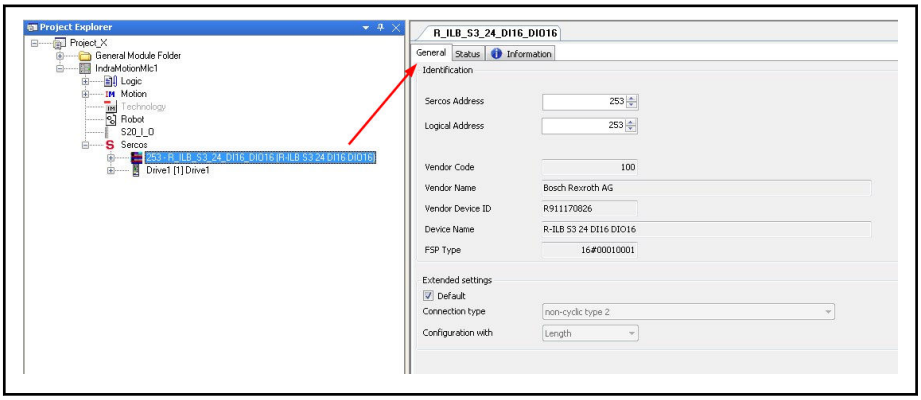


Fig. 9-13: Configuring a Sercos III slave object

Sercos III master; adding slave

- Enabling/disabling Sercos slaves, page 249
- Disabling Sercos slaves using UserDefs.cfg, page 250
- Tab “General”, page 252
- Tab “Sercos configuration”, page 254, this window is used for service purposes and is only visible if the option "Show generic configuration editors" is enabled under **Tools ▸ Options ▸ IndraLogic 2G ▸ Device editor**.
- Tab “Status”, page 254
- Tab “Information”, page 254

Tab “General”

The "General" tab includes settings and values that depend on the device description of the device.

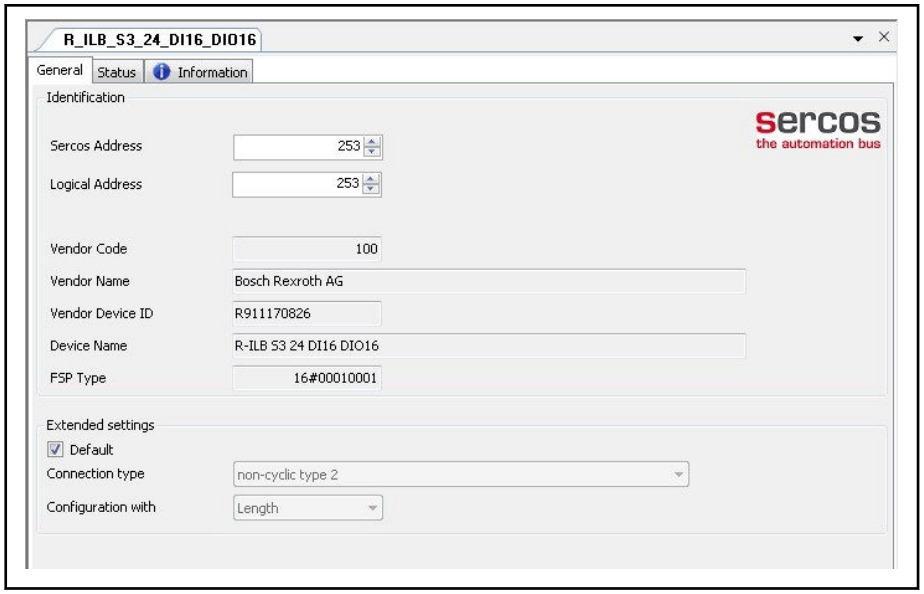


Fig. 9-14: Sercos III slave: "General" tab

Identification

In the "Identification" section, there is specific information on the Sercos III slave which identifies the slave unambiguously.

Sercos address:	Bus-specific address of the Sercos III slave
Logical address:	Logical address of the Sercos III slave
Vendor code	Vendor ID of the Sercos III slave
Vendor	Vendor name of the Sercos III slave, e.g. Rexroth
Vendor device ID	Identification number for the Sercos III slave specified by the vendor
Device name:	Name of the Sercos III slave by the vendor
FSP type:	This number defines the device-specific functions of the Sercos III slave, e.g., device properties

Extended settings

The "Extended settings" provides setting options affecting the configuration or the behavior of the slaves.

These are settings which should only be made by Sercos experts.

The settings can be changed when the checkbox (default) is disabled.

- **Connection type:** To set the connection type. Depending on the options of the respective slave, the following connection types can be selected:
  - **non-cyclic type 2**

    The connection is acyclic and without time monitoring. This is the default settings
  - **non-cyclic type 1 (with time monitoring)**

    The connection is acyclic and with time monitoring
  - **cycle-synchronous**

    The connection is synchronous to the master cycle
- **Configuration with:** Select whether the Sercos connection of a slave is achieved via **Length** or via **IDN**.

  This option is only available for devices supporting the I/O profile and a variable connection configuration.

Tab “Sercos configuration”

This tab is used for service purposes and is only visible if the option "Show generic configuration editors" is enabled under **Tools** ▸ **Options** ▸ **IndraLogic 2G** ▸ **Device editor**.

The <Write Parameters> button is only visible in online mode.

Tab “Status”

The "Status" tab displays status information [e.g. "Running" (bus active) and "n/a" (no information available)] and specific diagnostic messages from the respective device and on the card used and the internal bus system.

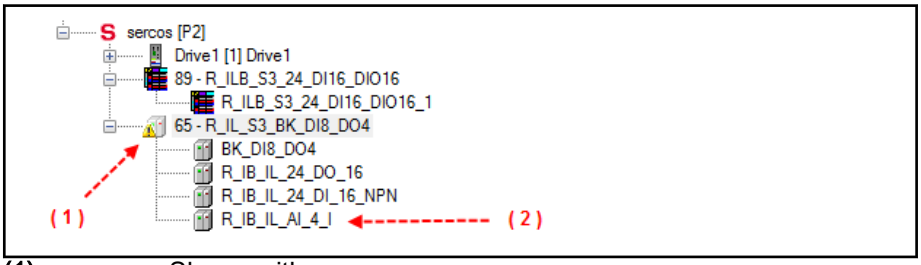
Modifications in the bus only offline

Prerequisites for the Sercos III bus to go online:

1. Configuration of the Sercos devices in the project and control match with respect to type, sequence and Sercos address.
2. All modules run without errors.

   This can be determined for the actual slave in the "Status" window by clicking on "Device identification" and "Advanced device identification".
3. Sercos state: P4

Slaves with errors (starting with the first) are displayed in the project tree with a yellow warning triangle and an error tool tip.



(1) Slaves with errors
(2) Terminal is only in the project, not on the control

Fig. 9-15: Sercos bus, error message in the Project Explorer

Tab “Information”

The tab displays general information from the device description file:

Name, Vendor, Categories, Version, Order number, Description and Image if available.

9.3.6 Sercos III master, adding modules to slave

Sercos III master, adding modules to slave, general

The modules (terminals) matching the respective slave (bus terminal) are located in the "Peripherals" library in the "Sercos III" folder under the respective slave:

I/O modules can only be inserted to modularly structured slaves (Sercos III differentiates two types of slaves to connect I/O modules, page 248).

Drag the required modules from the library to the "Slave" object.

New modules can also be inserted between existing modules in the Project Explorer.

Modules can also be added in the context menu of the slave via **Add ▸ Module ▸ ...**.

In this case, the new module is added as last module below the slave.

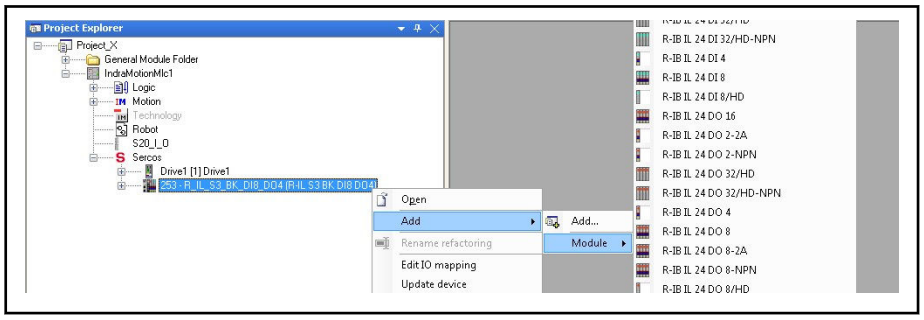


Fig. 9-16: Sercos III, adding modules to slave

To open the editor window in the Project Explorer, double-click on the module.

The dialogs provide information on the configuration of the module Modifications can be made.

The settings in all dialog tabs are specified by the device description file of the module. This file defines the preset value and whether it can be edited.

Sercos III master, adding modules to slave

- Tab “Sercos module”, page 256
- Tab “Function groups”, page 256
- Special tab Tab “User-defined parameters”, page 258,
- Tab “Sercos III modules configuration”, page 261, this window is used for service purposes and is only visible if the option "Show generic configuration editors" is enabled under **Tools ▸ Options ▸ IndraLogic 2G ▸ Device editor**
- Tab “I/O mapping”, page 262
- Tab “Information”, page 262

Enabling/disabling Sercos modules

The Sercos modules configured at a Sercos slave can be enabled and disabled in the Project Explorer.

Thus, select/deselect the button at the icon of the Sercos module.

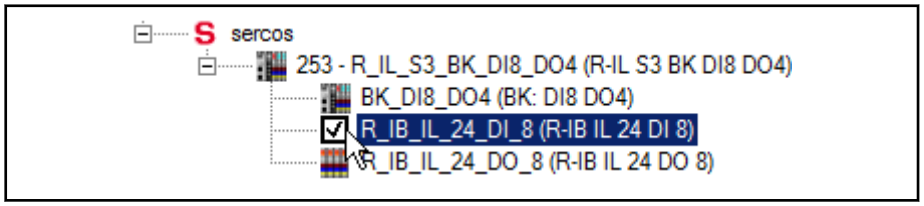


Fig. 9-17: Enabling/disabling Sercos modules

The module is enabled if the checkbox is selected. It is considered in the diagnostics of the bus.

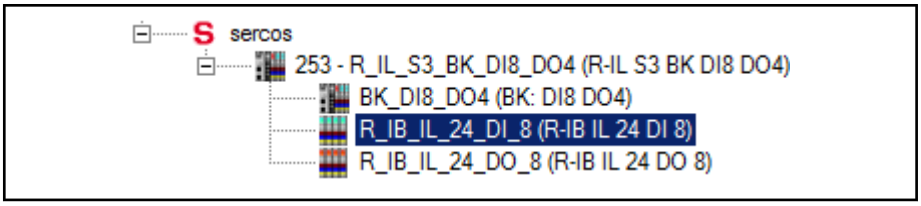


Fig. 9-18: Sercos module as enabled device

The module is disabled if the checkbox is deselected. It is not considered in the diagnostics of the bus.

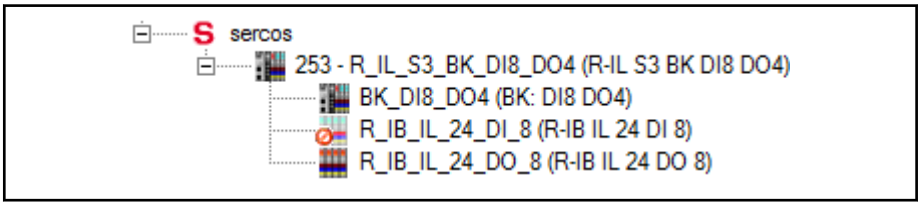


Fig. 9-19: Sercos module as disabled device

If a new Sercos module is created, it is enabled by default.

When loading the configuration, all enabled Sercos modules are taken into account. Disabled Sercos modules are ignored.

☞ An activated, but non-existing module inhibits the bus switching to P4. The device configuration is compared.

Also refer to Sercos III master, overview on bus addresses, page 235.

A module that is disabled and not present does not generate any error message in the diagnostics.

Also refer to Disabling Sercos slaves using UserDefs.cfg, page 250

Tab “Sercos module”

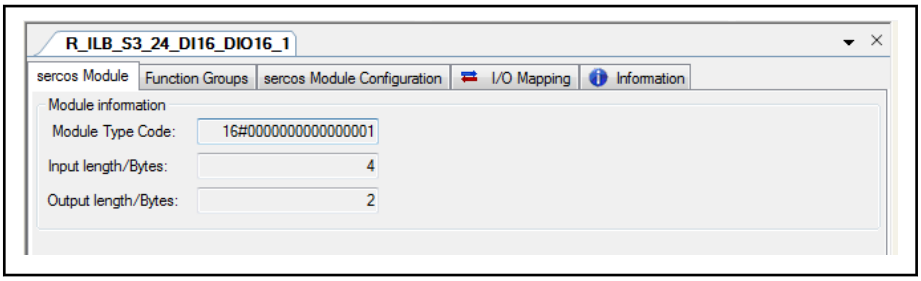


Fig. 9-20: Sercos III module: Sercos module

Module information

Module type code:	Each module has a unique module type ID from the respective vendor. Depending on the vendor, this number can be a hexadecimal number or an ordering number
Input length/bytes:	Specifies the input length of the module in bytes
Output length/bytes:	Specifies the output length of the module in bytes

Tab “Function groups”

The "Function groups" tab provides information on the inputs and outputs of the function group. None of the settings in this dialog can be edited.

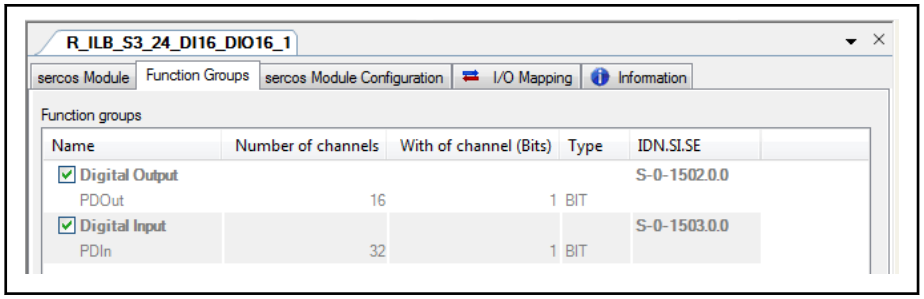


Fig. 9-21: Sercos III module: Function groups

Name:	Channel name, cannot be edited
Number of channels:	Number of supported channels for the module.
Channel width (bits):	Bit size of the individual channels, cannot be edited
Type:	Number of structural elements (SE) of the I/O function groups. Depends on the I/O functions
IDN.SI.SE:	
• IDN	Identification number, e.g. 15xx
• SI (Structure Instance)	The SI number is identical to the slide-in number of the modules Module 1 has SI number 1 For fixed modules, module 1 has SI number 0
• SE (Structure Element)	Number of structural elements (SE) of the I/O function groups. Depends on the I/O functions

Tab “General inputs and outputs”

This tab is displayed instead of the "Function group" tab if the module is either a drive, an encoder or a supplier. It is used to configure input and output connections of the module.

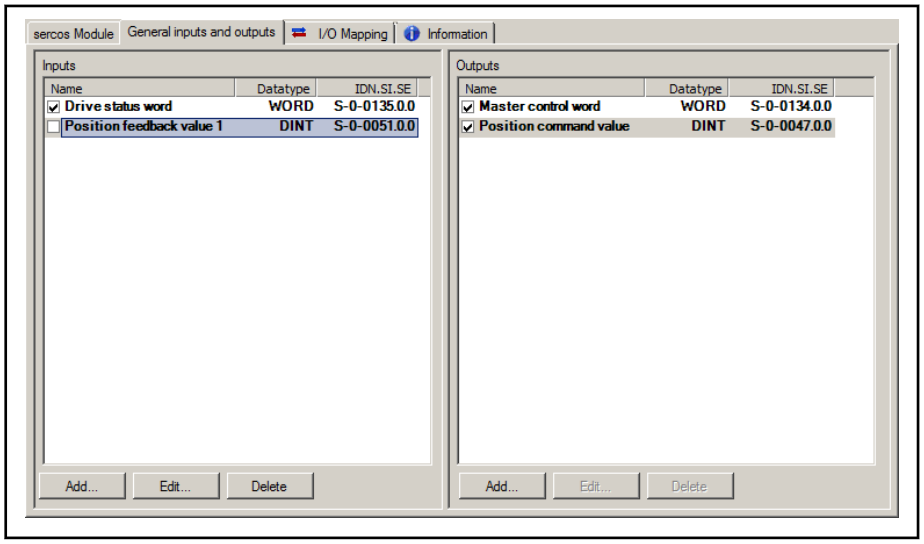


Fig. 9-22: Sercos III module: General inputs and outputs

To enable or disable the currently defined input and output connections, click on the checkbox. If a connection is enabled or disabled, the respective channels are added or removed in the I/O mapping.

Click on "Add" to add a new input or output or click on "Edit" to edit the currently selected connection. Either the "Input configuration" or the "Output configuration" dialog opens.

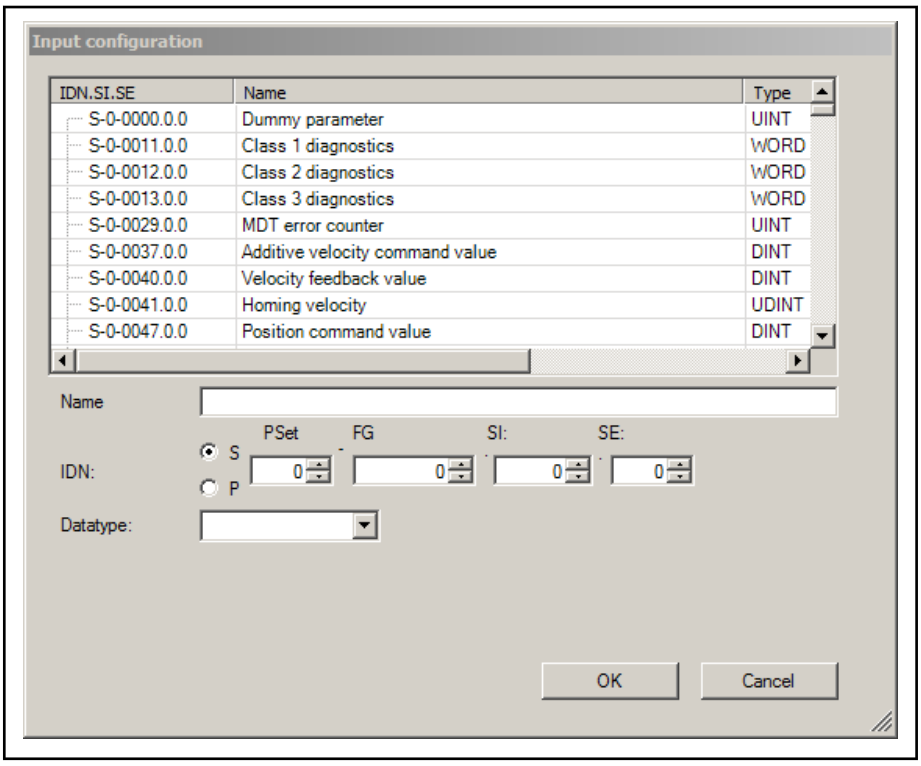


Fig. 9-23: Dialog “Input configuration”

This dialog displays a list of input and output objects defined in the device description of the module. Add new inputs and outputs by selecting one or multiple entries from the list and confirming with "OK". It can also be freely entered.

To remove a connection from the list of the inputs or outputs, select the entry and press "Delete".

Tab “User-defined parameters”

This (special) tab is used to preset parameters of the module and is only displayed for those modules which require or allow this default setting.

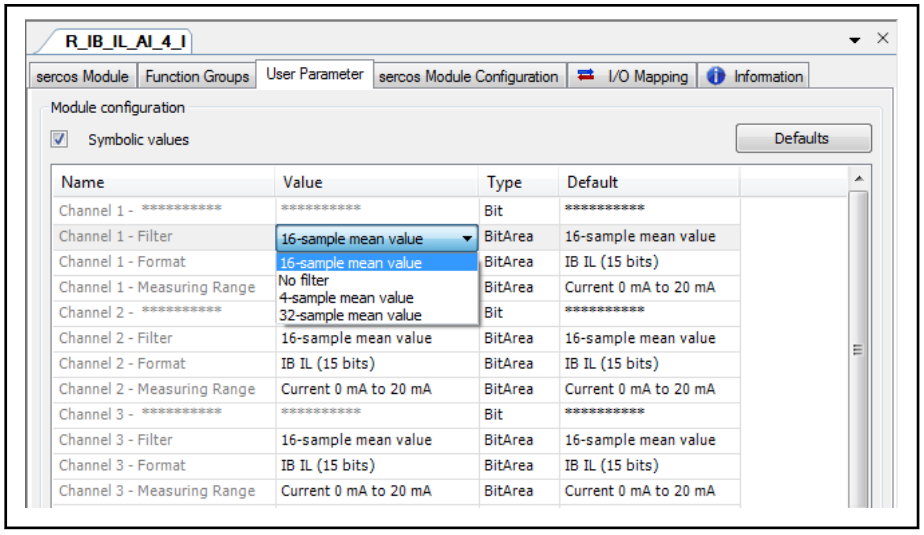


Fig. 9-24: Sercos III module: User-defined parameters

Module configuration

This example shows a module with four analog inputs.

The filter, format and measuring range of the expected input information is selected for the analog input AI4I.

A drop-down menu based on the enumeration data type is provided for each "Value".

Symbolic Values: Defines whether the value is displayed in plain text or as integer (16).

Default: Resets all parameters to their initial value.

Tab “Oversampling”

The "Oversampling" tab is used to configure the oversampling functionality. It is only visible for modules supporting this functionality.

Using the oversampling function, specific input signals can be loaded faster than allowed by the bus cycle time. Analog inputs can for example be read in 125µs, even though the bus cycle time is 1ms.

In this setting, eight data channels are available in the I/O mapping. The signals are transferred in addition to the normal signals. Thus, the normal input and the oversampled signals can be used at the same time.

The oversampling configuration is split in two areas. The available oversampling modules are listed in the upper area. They can be configured there:

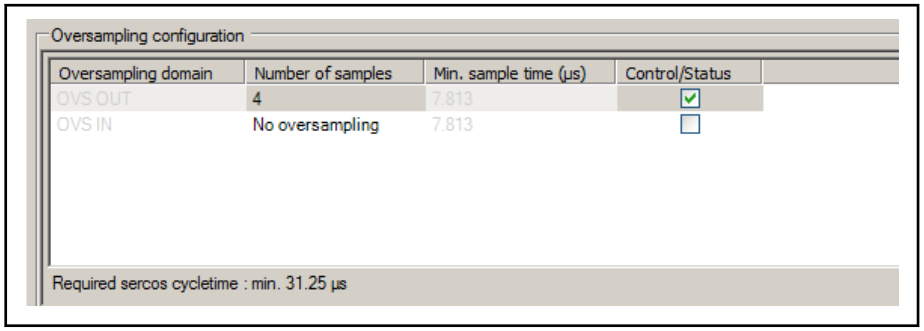


Fig. 9-25: Configuring the oversampling domains

- **Oversampling domain:** Name of the oversampling unit. Each unit can be configured separately.

- **Number of sampled:** Set the number of samples in one bus cycle here. The values to be set depend on the module. There can be modules allowing only certain specified ratios (e.g. 1, 2, 4, 8).
- **Min. sample time (µs):** This is the smallest sample interval supported by the oversampling domain.
- **Control/Status:** If the checkbox is selected, the control and status word are also transferred. The corresponding I/O channels are added to the dialog "I/O Mapping" (see ① in fig. 9-27 "E/A image with channels for oversampling" on page 261).
- **Required sercos cycle time:** Indicates the Sercos cycle time, which is at least required to ensure that it is not fallen below the minimum sample time.

  Example:

  If the minimum sample time of the device is 7.81 µs and the number of samples is set to 16, the Sercos cycle time has to be at least 125 µs.

If an oversampling domain is selected in the upper area, the channels of this oversampling domain are displayed in the lower area.

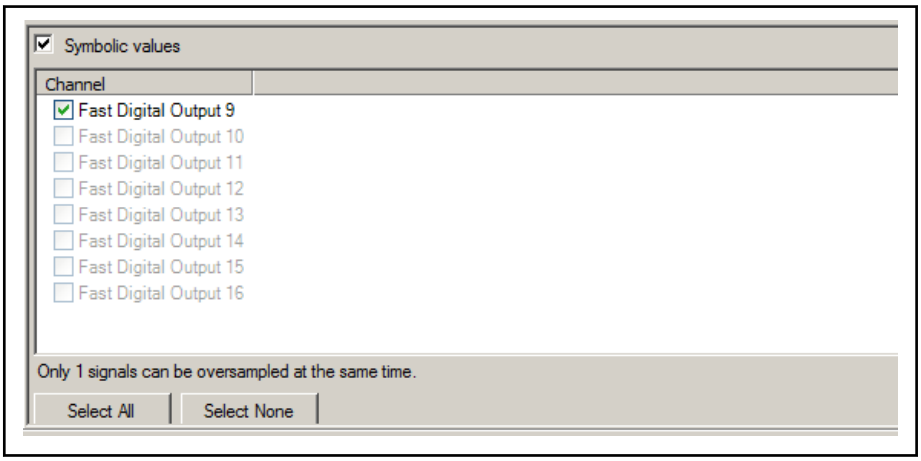


Fig. 9-26: Selecting the channels to be sampled

Select the checkbox to enable oversampling for one channel. Depending on the number of samples set, a corresponding number of channels are added to the I/O image (see ② in the following figure).

The original normal signal remains and is labeled with "oversampled" in the channel name (see ③ in the following figure).

The number of channels that can be sampled at the same time depends on the module.

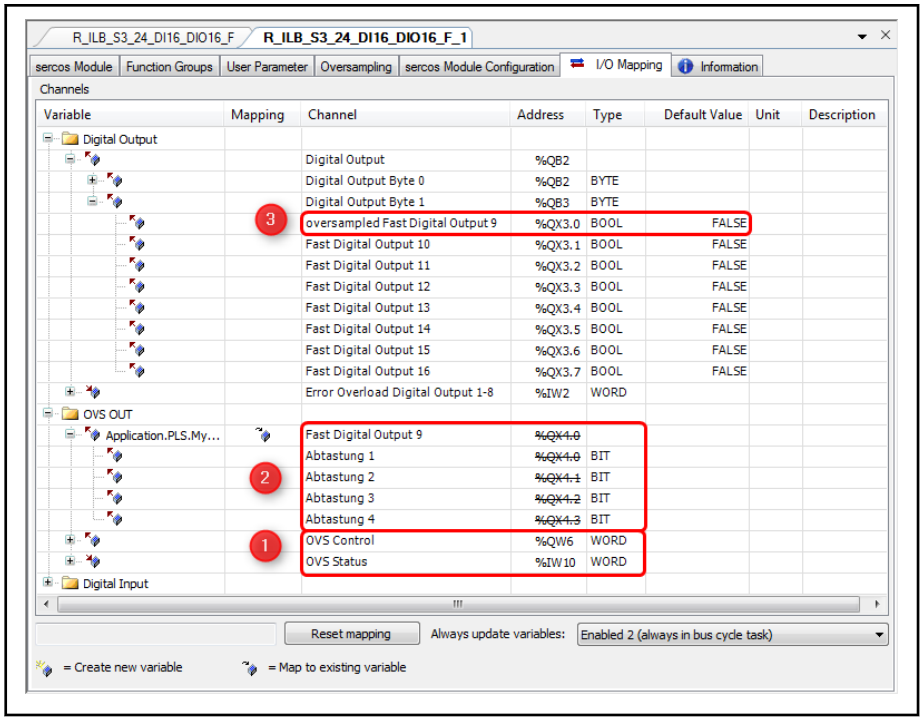


Fig. 9-27: E/A image with channels for oversampling

Tab “Sercos III modules configuration”

This tab is used for service purposes and is only visible if the option "Show generic configuration editors" was enabled. Opening via the IndraWorks menu **Tools ▸ Options ▸ IndraLogic 2G ▸ Device editor**.

Please clarify any possible modifications of the parameters with the service team.

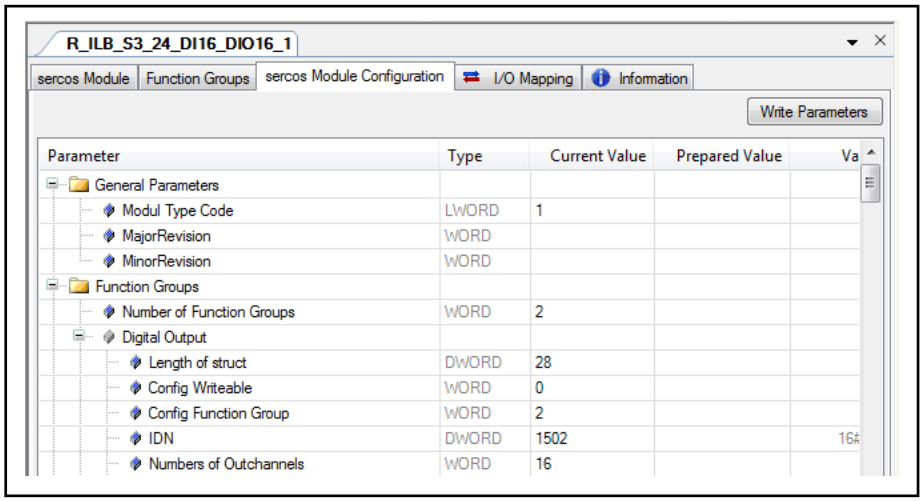


Fig. 9-28: Sercos III module: Sercos module configuration

The <Write Parameters> button is only visible in online mode.

Tab “I/O mapping”

The tab is used to assign Sercos III module inputs and outputs to variables that can be used as local or global variables in the individual POUs.

The current value of the variables is displayed in online mode.

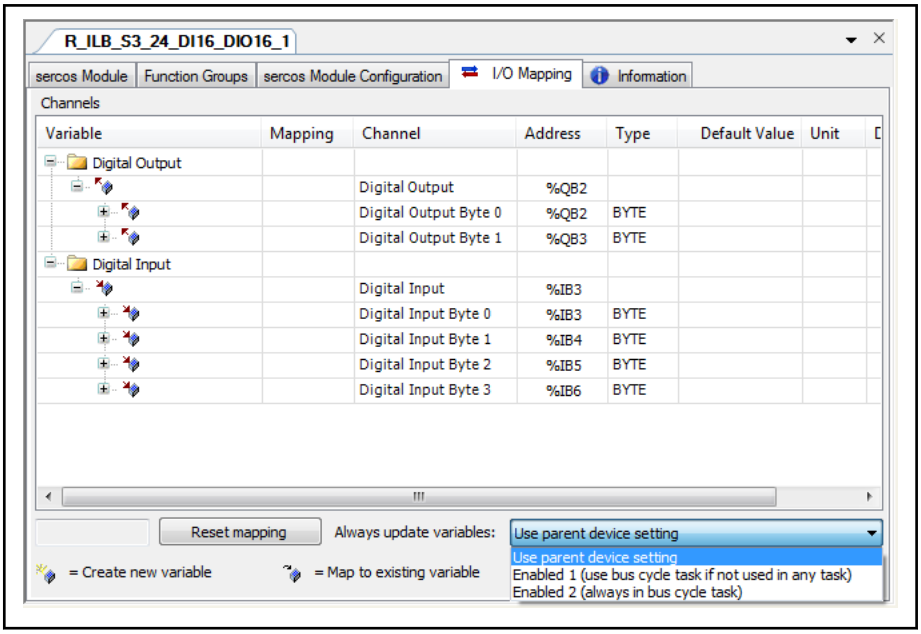


Fig. 9-29: Sercos III module: I/O mapping

Reset Mapping: Use this button to reset the mapping settings to the default values defined in the device description file.

Update variables: Defines for the module whether I/O variables are updated in the bus cycle task.

The default value is defined in the device description file.

- **Use parent device setting:** Update according to the settings of the superordinate device.
- **Enabled 1 (use bus cycle task if not used in any task):** The I/O variables are updated in the bus cycle task if not used in any other task
- **Enabled 2 (always in the bus cycle task):** All variables are updated in each cycle of the bus cycle task irrespective of whether they are used or mapped on an input or output channel.

This assignment is described in Mapping of the field bus inputs and outputs, page 19,.

Tab “Information”

The tab displays general information from the device description file:

Name, Vendor, Categories, Version, Order number, Description and Image if available.

10 Modbus RTU

10.1 Modbus bus cycle task

General information

As a rule, for each IEC task, the used input data is read at the start of each task (1) and the written output data is transferred to the I/O driver at the end of the task (3). The implementation in the I/O driver is the decisive factor for a further I/O data transfer. The implementation is therefore responsible for the time grid and the specific time when the actual transmission occurs on the respective bus system.

The bus cycle task of the PLC can be specified globally for all field buses in the PLC settings. For some field buses, however, you can change them irrespective of the global setting. The task with the shortest cycle time is automatically used as bus cycle task (setting: unspecified in the PLC settings). In this task, the messages are normally sent to the bus.

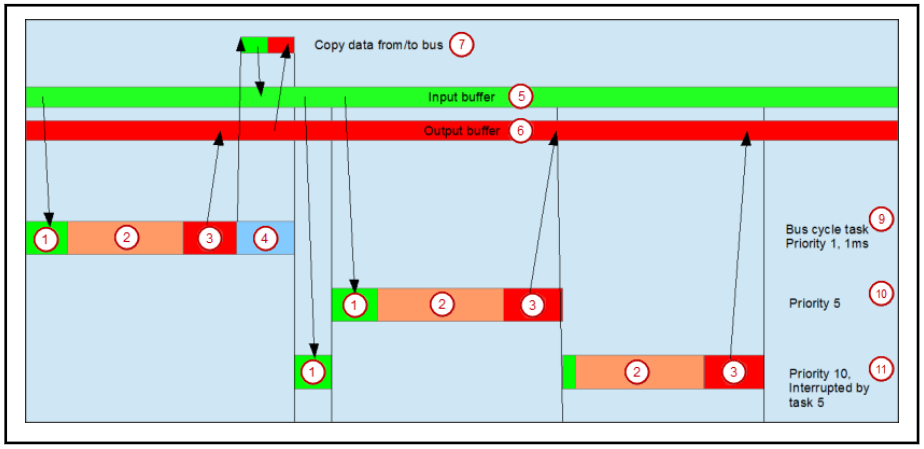


Fig. 10-1: Modbus - Bus cycle task

1	Reading the inputs of the input buffer
2	IEC task
3	Writing the outputs to the output buffer
4	Bus cycle
5	Input buffer
6	Output buffer
7	Copying data from/to the bus
9	Bus cycle task, priority 1, 1 ms
10	Bus cycle task, priority 5
11	Bus cycle task, priority 10, interrupted by task 5

Task use

The “Task deployment” tab provides an overview on the I/O channels used, the set bus cycle task and the usage of the channels.

⚠ WARNING

If an output is written in various tasks, the status is undefined, as this can be overwritten in each case.

When using the same inputs in different tasks, the input can change while processing a task. This results when the task is interrupted by a task with a higher priority and causes the process image to be read again. Solution: At the beginning of the IEC task, copy the input variables to variables and then work only with the local variables in the rest of the code.

Result: Using the same inputs and outputs in several tasks is not reasonable and can cause unexpected reactions.

Also refer to:

- chapter 10.2 "Modbus configurator" on page 264
- For further documentation on the IndraLogic programming system:
  DOK-IWORKS-IL2GPRO*V14-APRS-EN-P
  Chapter:
  – Tab “Task deployment”
  – "PLC settings” tab

10.2 Modbus configurator

A Modbus network consists of a Modbus master and one or multiple Modbus slaves. Up to 32 slaves can be below one Modbus master.

Modbus devices can be integrated into IndraLogic via the serial interface using the device “Modbus COM port”. The IndraLogic runtime can be the Modbus master (1) or the Modbus slave. This Modbus slave is called "Modbus device" in the following. The operation mode “Modbus RTU” is supported for Modbus serial.

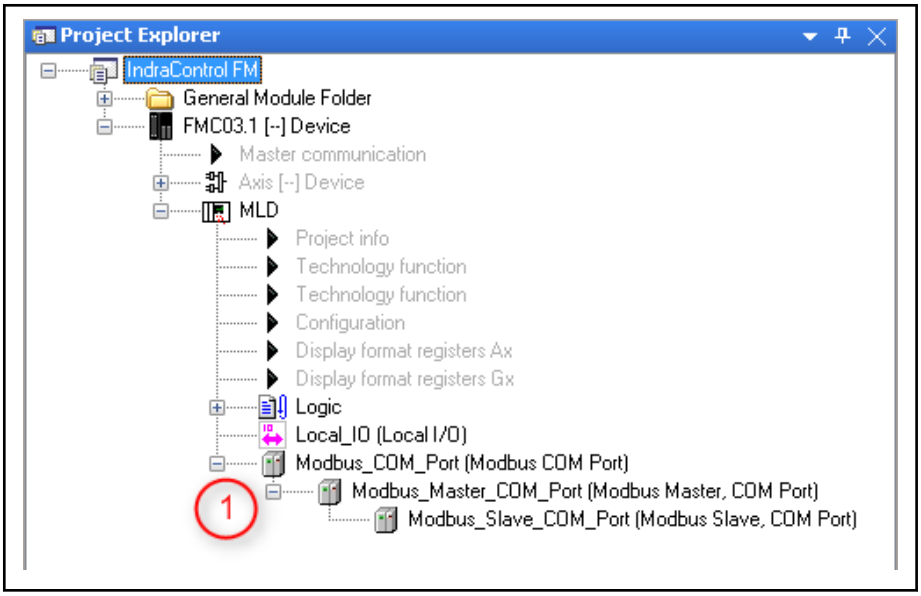


Fig. 10-2: IndraWorks - Project tree with Modbus devices

It can be configured via the Modbus configuration sites on the communication parameter and the so-called Modbus channels can be created. One individual Modbus command (read/write data) as well as the respective I/O channels are behind one Modbus channel.



For the general settings on the device editors, refer to the “Device editor” options.

Also refer to:

- Device editor options in the IndraLogic application description:
  DOK-IWORKS-IL2GPRO*V14-APRS-EN-P
- chapter 10.3 "Modbus master" on page 265
- chapter 10.4 "Modbus slave device" on page 269

Supported function codes

The master generally supports the following function codes if not restricted by the special type of the control:

- FC01 Read Coils
- FC02 Read Discrete Inputs
- FC03 Read Holding Registers
- FC04 Read Input Registers
- FC05 Write Single Coil
- FC15 Write Multiple Coils
- FC06 Write Single Register
- FC16 Write Multiple Registers
- FC23 Read/Write Multiple Registers

Accessing Modbus devices from the application

If a Modbus device is included in the device tree, an instance of the respective function block is created automatically. The variable name of the instance corresponds to the device name in the device tree. This function block can be used to access the functions of the Modbus devices from the application. The outputs of this function block can for example query the status of the last Modbus command (whether successful or with error).

For the variable name, go to the I/O mapping tab of the respective device editor under IEC objects.



For Modbus master (RTU/TCP)

In addition/alternatively to the Modbus commands (and the implicitly created I/O mapping) predefined in the configurator, Modbus requests can be programmatically executed via the ModbusRequest function block.

10.3 Modbus master

10.3.1 Tab “Modbus serial port - General”

The tab in the configurator of the Modbus serial port includes the basic settings for the communication via the serial interface.

COM port	Number of the serial interface
Baud rate	Parameter of the serial interface
Parity	
Data bits	
Stop bits	

Also refer to: chapter 10.2 "Modbus configurator" on page 264

10.3.2 Tab “Modbus master - General”

The tab includes the basic setting for the Modbus communication.

Modbus RTU/ASCII

Transmission mode	• RTU: Binary transmission • ASCII: Transmission in ASCII code (it is currently not supported by all drivers)
Response Timeout [ms]	Time interval during which the master waits for the response of a slave node. If the node does not respond within this interval, an error is present at the implicit slave function block. The value entered for this interval is also the default values for each node. However, a specifically adjusted value can be set for each node within the slave configuration.
Time between the frames [ms]	The time between the frames is the intermission of the master between the last response (e.g. timeout) and the next request. Use this parameter to ensure a less faulty communication.
Automatic communication restart	☑: After a communication error, IndraLogic automatically acknowledges the error and tries to continue with the Modbus command execution. ☐: The error has to be confirmed explicitly via the slave function block.

Also refer to:

- chapter 10.3.4 "Tab “Modbus slave - General”" on page 266
- chapter 10.2 "Modbus configurator" on page 264

10.3.3 Tab “ModbusTCP master - General“

The tab includes the basic settings for the communication in the network.

Modbus TCP

Response Timeout	Time interval during which the master waits for the response of a slave node. If the node does not respond within this interval, an error is present at the implicit slave function block. The value entered for this interval is also the default values for each node. However, a specifically adjusted value can be set for each node within its Modbus TCD slave configuration.
Socket Timeout	Maximum waiting period for the incoming TCP/IP packages. For this period, the bus cycle task can be blocked (e.g. if a Modbus TCP slave is removed)

Also refer to:

- chapter 10.3.4 "Tab “Modbus slave - General”" on page 266
- chapter 10.2 "Modbus configurator" on page 264

10.3.4 Tab “Modbus slave - General”

The tab includes the basic settings for the communication via the serial interface.

Modbus RTU/ASCII

Slave address [1..247]	Address of a serial Modbus device. Value between 1 and 247.
Response Timeout [ms]	Time interval during which the master waits for the response of a slave node. It is especially adjusted to this slave node and overwrites the general response timeout setting of the respective master.

Also refer to:

- chapter 10.3.2 "Tab “Modbus master - General”" on page 266
- chapter 10.2 "Modbus configurator" on page 264

10.3.5 Tab “ModbusTCP slave - General“

The tab includes the basic settings for the communication in the network.

Modbus TCP

Slave IP address:	
Unit ID [1..247]	No significance for Modbus TCP devices. Used to address serial Modbus devices addressed via a Modbus TCP/serial gateway.
Response Timeout	Time interval during which the master waits for the response of this slave. It is especially adjusted to this slave node and overwrites the general response timeout setting of the respective master.
Port	Port number (TCP/IP) of the slave. 502 is standard for Modbus.



The communication settings can be changed at runtime using the method `UpdateCommunicationSettings`. This method is available for the implicitly generated function block of this Modbus slave.

Also refer to:

- chapter 10.3.2 "Tab “Modbus master - General”" on page 266
- chapter 10.2 "Modbus configurator" on page 264

10.3.6 Tab “Modbus slave - Channel”

This tab is used to define slave channels.

Each channel represents an individual Modbus request.

Click on the "Add channel" button and a dialog is displayed to create new channels for this Modbus slave. It is possible to predefine descriptions on individual Modbus registers or to complete channels in the device description file of the Modbus slave.

Add channel	Opens the "Modbus channel" dialog.

“Modbus channel” dialog

Name	A string containing the channel name
Access type	• Read Coils (function code 03) • Read Discrete Inputs (function code 04) • Read Holding Registers (function code 03) • Read Input Registers (function code 04) • Read Single Coil (function code 05) • Write Single Register (function code 06) • Write Multiple Coils (function code 15) • Write Multiple Registers (function code 16) • Read/Write Multiple Registers (function code 23)
Trigger	• "CYCLIC": The request is periodic. • "RISING_EDGE": The query results from a rising edge of the Boolean trigger variables. The trigger variable is defined in the “I/O mapping” tab.
Cycle time (ms)	For trigger = CYCLIC: Query interval Note: The query interval should either be equal to or a multiple of the cycle time of the application.
Comment	Channel description

READ register

Offset	Starting address used to start reading from (value range 0-65535)
Length	Number of registers to be read (for word access) or number of Discrete Inputs to be read (for bit access).
Troubleshooting	Defines the procedure of further data processing if there is a communication error. • Setting to ZERO • Retaining last value

WRITE Register

Offset	Number of the register to be written to (value range 0-65535)
Length	Number of registers to be written (= Words). Value range of the parameter depends on the function code:

Example:

The following example defines the first line of a combined read-write operation (function code 23). It retrieves a word from the “Holding Register" with distance `'16#0001'` and writes two words to the register with distance `'16#0003'`. The operation is executed as soon as a rising edge is detected for the defined trigger variable on the “I/O mapping” tab.

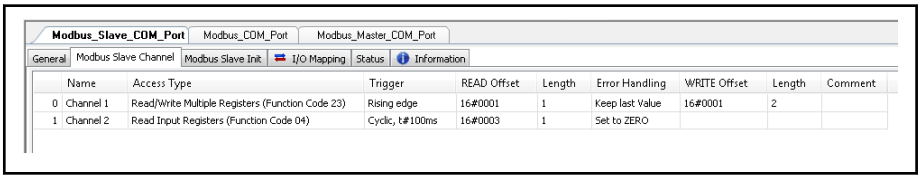


Fig. 10-3: “Modbus slave channel” dialog

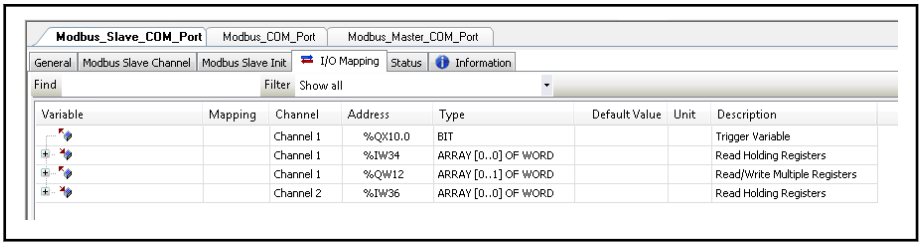


Fig. 10-4: "Modbus slave I/O mapping" dialog

Also refer to:

- chapter 10.3.4 "Tab "Modbus slave - General"" on page 266
- chapter 10.3.5 "Tab "ModbusTCP slave - General"" on page 267

10.3.7 Tab "Modbus slave init"

This tab is used to define initialization commands.

Initialization commands are executed once when starting the bus or enabling the slave (set the "Enabled" flag of the slave instance).

When setting and editing a slave initialization value, the following parameters are provided in the respective dialogs:

"Move up" "Down"	The channel sequence also specifies the sequence of the variable initialization.
"New"	Opens the "Initialization value" dialog. It specifies the initialization commands.

"Initialization value" dialog

"Access type"	• Write Single Coil (function code 05) • Write Single Register (function code 06) • Write Multiple Coils (function code 15) • Write Multiple Registers (function code 16)
"Register Offset "	Number of the register to be written to (value range 0-65535).
"Length"	Number of registers to be written (= Words) Value range of the parameter depends on the function code:
"Initialization value"	Initialization value for register
"Comment"	Option to briefly describe data

Also refer to: chapter 10.3.4 "Tab "Modbus slave - General"" on page 266

10.4 Modbus slave device

10.4.1 Tab "Modbus serial device - General"

The tab includes the basic setting for the communication in the network.

Unit ID	Unit ID of the slave
Timeout	☑: Enabling the time monitoring function. The timeout span is specified in milliseconds. Values are set in steps of 500 ms. If no write command was received for the time specified, the outputs are set to 0.

Holding Registers (%IW)	Number of holding registers: Possible values: 1-4096 (Maximum number can be limited in the device description).
Input Registers (%QW)	Number of input registers Possible values: 1-4096 (Maximum number can be limited in the device description).

Also refer to chapter 10.2 "Modbus configurator" on page 264

10.4.2 Tab "ModbusTCP slave device - General”

The tab includes the basic setting for the communication in the network.

Configured parameters

"Timeout "	☑: Enabling the time monitoring function. The timeout span is specified in milliseconds. Values are set in steps of 500 ms. If no write command was received for the time specified, the outputs are set to 0.
"Slave port "	Port number of the slave.
"Unit ID "	Optional: Unit ID of the slave. Only for TCP/RTU gateway.
"Holding Registers (%IW) "	Number of holding registers: Possible values: 1-4096 (Maximum number can be limited in the device description).
"Input Registers (%QW)"	Number of input registers Possible values: 1-4096 (Maximum number can be limited in the device description).
"IP master address"	IP address of the master, example: "192 . 168 . 0 . 1" Whether the option is displayed in the dialog depends on the device description.
"IO mapping active"	☑: Enabling the I/O scanner. Whether the option is displayed in the dialog depends on the device description.

☞ Changes in the input/output assembly size do immediately affect the number of configurable parameters in the I/O mapping.

Data model

"Coils"	Definition of the starting address of the respective Modbus data area.
"Discrete Inputs"	
"Holding Register"	
"Input Register"	
"Superimposed Holding and Input Register data area"	☑: Superimposition of the process image of the Holding Register and of the Input Register. This is for example required when writing to the Holding Register via the slave application. For details on the Modbus data model, refer to the Modbus protocol specification, chapter 4.3 and 4.4 (www.modbus.org)

Also refer to: chapter 10.2 "Modbus configurator" on page 264

10.5 Tab “Modbus - Parameters”

The tab corresponds to the description of the general configuration dialog. The dialog displays the device-specific parameters in a list.

The tab is available if the option "Show generic configuration editors" is selected.



This dialog is for test purposes only. Only experts should change values.

Also refer to:

- Chapter: Dialog “Options” - “Device editor” in the IndraLogic application description:
  DOK-IWORKS-IL2GPRO*V14-APRS
- chapter 10.2 "Modbus configurator" on page 264

10.6 Tab “I/O mapping”

The description corresponds to the general description of the "I/O mapping" tab.

Also refer to:

- Chapter: **“<Device> I/O mapping” tab** in the IndraLogic application description:
  DOK-IWORKS-IL2GPRO*V14-APRS
- chapter 10.2 "Modbus configurator" on page 264

10.7 Tab “Status”

The description corresponds to the general description of the "Status" tab.

See:

- Chapter: **“Status” tab** in the IndraLogic application description
  DOK-IWORKS-IL2GPRO*V14-APRS
- chapter 10.2 "Modbus configurator" on page 264

10.8 Tab “Information”

The description corresponds to the general description of the "Information" tab.

See:

- Chapter: **“Information” tab** in the IndraLogic application description:
  DOK-IWORKS-IL2GPRO*V14-APRS
- chapter 10.2 "Modbus configurator" on page 264

11 IO-Link

11.1 Introduction and overview

IO-Link is a communication interface to connect intelligent sensors and actuators to automation systems. An IO-Link system always consists of one IO-Link master and one or multiple IO-Link devices. IO-Link master and IO-Link device are connected via a point-to-point connection. Thus, when using a master with multiple devices, a star topology always results. The IO-Link master is normally a gateway to a superordinate field bus. However, USB IO-Link gateways are also used for the initial commissioning.

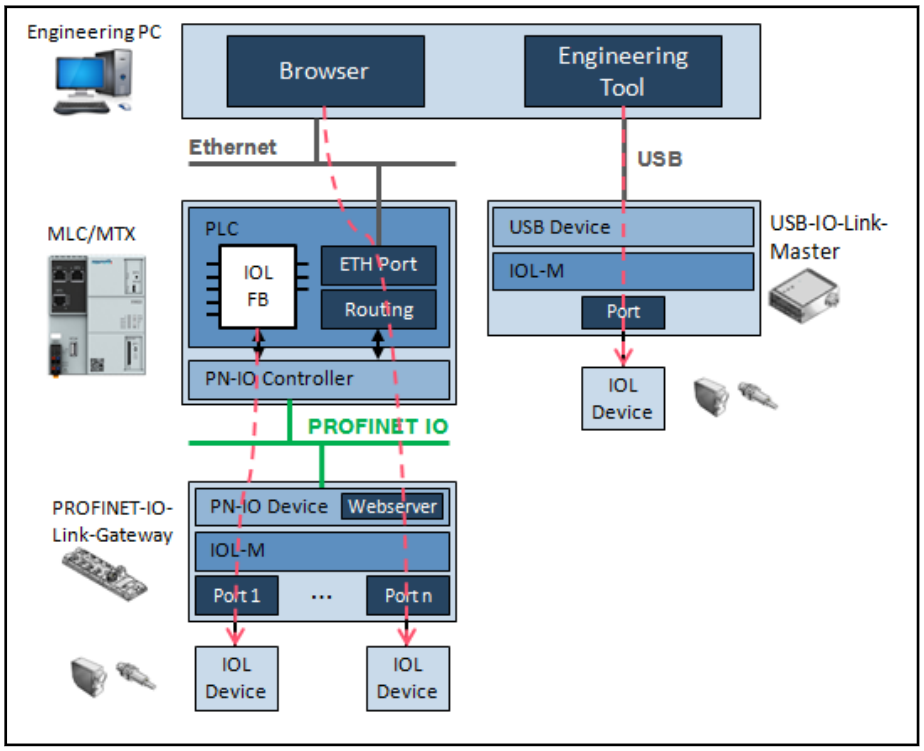


Fig. 11-1: Overview on the architecture when using an IO-Link

11.2 Use cases for the engineering of I/O systems

11.2.1 Reading in or outputting values using an IO-Link device

The onboard ports of the IO-Link master gateway can process either only IO-Link values or digital values.

To process analog values, adapters are required to convert from analog to IO-Link.

Available IO-Link interface modules:

Type	Part number	Function
AXL E IOL AI1 I M12 R	R911175300	Reading in analog current sensors
AXL E IOL AO1 I M12 R	R911175301	Analog current output
AXL E IOL AI1 U M12 R	R911175302	Reading in analog voltage sensors
AXL E IOL AO1 U M12 R	R911175303	Analog voltage output

Type	Part number	Function
AXL E IOL RTD1 M12 R	R911175304	Reading in temperature resistor sensors
AXL E IOL DI8 M12 6P	R911175266	Reading in 8 digital inputs
AXL E IOL DI16 M12 6P	R911175298	Reading in 16 digital inputs
AXL E IOL DO8 M12 6P	R911175299	Outputting 8 digital outputs

11.2.2 Initial commissioning of an IO-Link device

The easiest way for the initial commissioning of an IO-Link device is to use a USB link master allowing the connection of IO-Link devices at the PC. USB IO-Link masters are provided by different vendors with the corresponding engineering tools and are used to set and test the functions of the IO-Link device. After adding the device description file of the IO-Link device (IODD file), cyclic process data as well as the parameters with the names stored in the device description file can be displayed and edited.

Procedure:

- Connect IO-Link device at USB IO-Link master
- Open engineering tool of the USB master (import IODD file of the IO-Link device if required)
- Parameterize IO-Link device
- Connect the IO-Link device to the IO-Link gateway and operate it with the PLC

11.2.3 Process data and IO Device Description (IODD) of IO-Link devices

☞ IndraWorks does currently not support any IO Device Description (IODD)!

Process data (byte order) is interpreted using the data sheet of an IO-Link device.

According to the IO-Link specification, process data is to be transferred as `BYTE`.

Process data is interpreted in the application, e.g. whether a `WORD` analog value has to be swapped. The interpretation can be different between the different IO-Link devices.

11.2.4 Serial commissioning of machines with IO-Link devices

For the serial commissioning of the ProfiNet, the following IO-Link-Backup/ Restore function blocks can be used to transfer configuration parameters to I/O devices:

- IL_PNIOIoLinkBackupDevice

  The function block creates a backup file on the control to save the parameterization of an IO-Link device.

- IL_PNIOIoLinkRestoreDevice

  The function block reads a backup file on the control and writes the parameterization from the file to the respective IO-Link device.

To commission a machine of the same design, the backup file can be loaded to the new control, e.g. using the FTP client or the device data backup in IndraWorks.

The parameterization in the backup file can be written to the respective IO-Link device by executing the IO-Link Restore function block.



In case of Sercos, data is backed up and restored using the Sercos parameter backup, i.e. the data storage is included in the Sercos parameter backup.

Procedure:

- Parameterize IO-Link device at the machine for initial commissioning
- Create backup file on the control of the machine for initial commissioning
- Copy backup file to the control of the structurally identical machine
- Restore the parameterization of the IO-Link device using the copied backup file

11.2.5 Parameterizing IO-Link devices from the PLC program

To change the parameters of an IO-Link device from the PLC application, the following IO-Link call function blocks can be used:

- Profinet: IL_PNIOIoLinkCall
- Sercos: IL_IoLinkCall

The functions blocks are also used to read and write individual parameters of an IO-Link device which also allows to change parameters at runtime for a product-specific parameterization of the IO-Link device for example.

11.2.6 Diagnostics of IO-Link devices

There are different use cases to diagnose IO-Link devices. The easiest way to check the parameterization and function of an IO-Link device is to use a USB IO-Link master. Depending on the gateway, it can also be provided with a web server. If a routing entry is set to the field bus via the control at the Engineering PC, the web sites of the gateway can be displayed in the browser and the gateways and the IO-Link device can be diagnosed and parameterized without removing the IO-Link device from the machine (Overview on the architecture when using an IO-Link, page 273).

The following function blocks provide more diagnostic options:

IO-Link call function blocks to retrieve individual parameters:

- Profinet: IL_PNIOIoLinkCall
- Sercos: IL_IoLinkCall

IO-Link RemoteDeviceState function block to retrieve the device status, error counter and currently pending messages:

- Profinet: IL_PNIOIoLinkRemoteDeviceState

11.2.7 Exchanging IO-Link devices

For the device exchange of the IO-Link devices, a parameter server is available in the IO-Link master. If the parameter server function is enabled and the upload or download of parameter server data is also enabled, the parameterization of the IO-Link device is uploaded to the parameter server after each valid parameter change. A defective IO-Link can be replaced by another IO-Link device either at runtime or if it is disabled. This causes an interruption in the communication with the IO-Link master that outputs a diagnostics via the field bus. The IO-Link master automatically establishes the connection to the new I//O link device again and checks the CRC checksum of the parameterization. If it does not match the parameter set on the parameter server, the

IO-Link master downloads the parameter set. Thus, the data of the defective IO-Link device backed up on the parameter server is applied to the new IO-Link device.



To ensure this automatic device exchange, the following prerequisites have to be met in the IO-Link device:

- IO-Link device acc. to the IO-Link specification 1.1
- IO-Link index 0x0003 (see IO-Link specification) is available and contains all parameters to be backed up

If these prerequisites are not met, execute the parameter backup for the device exchange in the PLC program, e.g. using the following backup/restore function blocks:

- IL_PNIOIoLinkBackupDevice
- IL_PNIOIoLinkRestoreDevice
- IL_PNIOIoLinkCall
- IL_IoLinkCall

11.3 Profinet I/O IO-Link gateways

11.3.1 Introduction and overview

Structures and function of the Profinet I/O IO-Link gateway are described in the specification "IO-Link Integration Part 1 – Technical Specification for Profibus and Profinet". There are two structural variants of a Profinet I/O IO-Link gateway. Variant A is an “IO-Link master module“ connected to a Profinet device via a local bus. Variant B is an “IO-Link compact master”.

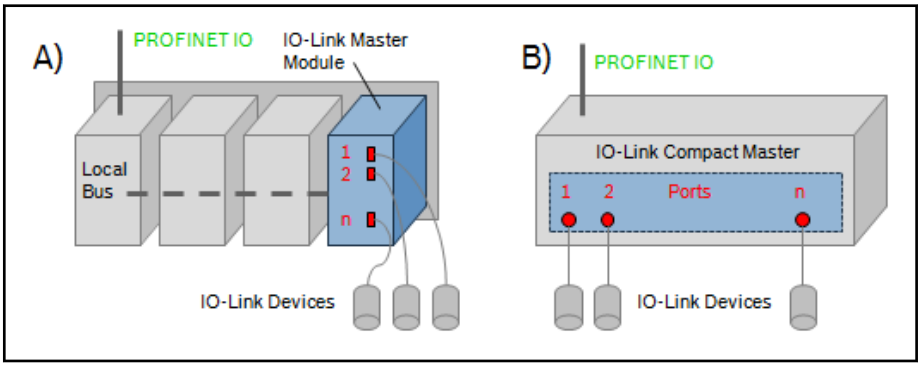


Fig. 11-2: Structuring Profinet I/O IO-Link gateways

The IO-Link master is addressed via the respective slot. In case of a compact device, the IO-Link master can normally be reached via slot 1. There are two variants to assign the subslots: Single subslot mapping and multi subslot mapping. For the single subslot mapping, all devices are accessed via subslot 1. The devices are selected via the port (starting with 1). For the multi subslot mapping, a subslot is available for each device. The cyclic input and output data as well as the diagnostics and alarms are accessed via the subslot. Acyclic writing and reading on the parameters is also executed via subslot 1 of the IOL master as well as the port specification.

Thus, when using the function blocks, subslot 1 is always used when writing and reading parameters via acyclic services. The devices are addressed via the port. In case of single subslot mapping, the cyclic data of all IO-Link devices is assigned to subslot 1. In case of multi subslot mapping, the cyclic data of the IO-Link devices is configured at the subslots.

The type of mapping variant implemented in a gateway depends on device and vendor. The existing function blocks at the IO-Link operate with both variants.

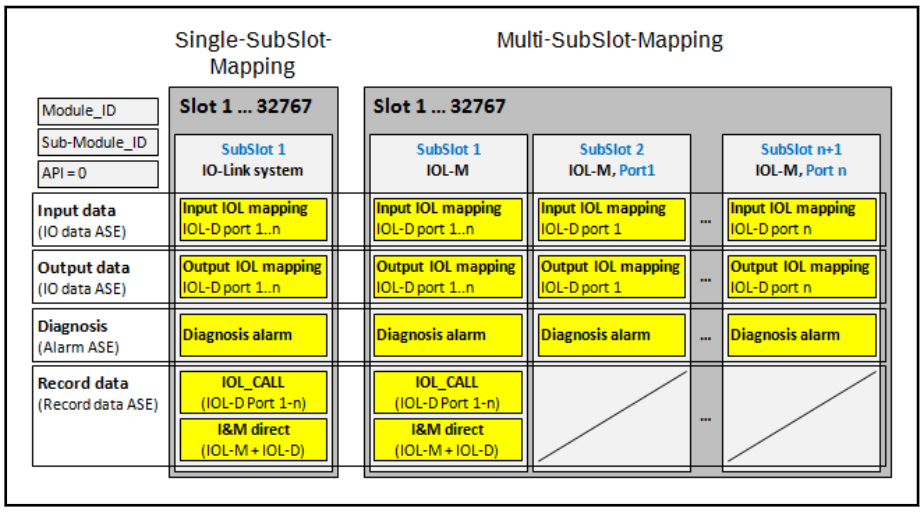


Fig. 11-3: Single subslot mapping and multi subslot mapping for Profinet I/O IO-Link gateways

11.3.2 Function blocks for Profinet I/O IO-Link gateways

The following function blocks to communicate with IO-Link devices at the Profinet I/O IO-Link gateways are provided in the RIL_ProfinetIO library.

Parameter access

IL_PNIOIoLinkCall: The function block is used to read and write parameters of the IO-Link device. It is accessed via index and subindex of the respective parameter.

Identification

IL_PNIOIoLinkIdent: The function block is used to identify the IO-Link device. The device type of the IO-Link device can be uniquely identified via VendorID and DeviceID. This is for example required for backup/restore functions.

IL_PNIOIoLinkIdentDetails: The function block additionally reads multiple character strings such as vendor name or IO-Link device name, which can, for example, be displayed on the machine HMI.

Status

IL_PNIOIoLinkRemoteDeviceState: The function block retrieves the diagnostic data of the IO-Link device. These are the error counter, the current device state as well as the currently pending error messages and the warning.

System command

IL_PNIOIoLinkSystemCommand: The function block is used to initiate commands of the IO-Link device, such as restoration the factory settings.

Access locks

IL_PNIOIoLinkSetDeviceAccessLock: The function block is used to set and reset access locks in the IO-Link device. In the IO-Link, four access locks are specified.

- If the write lock is set for parameters (ParameterAccessLock), no parameters - except for the parameter "Device Access Locks" (0x000C) - can be written anymore via the IO-Link communication. Writing parameters causes an error message at the function block **IL_PNIOIoLinkCall**. Reading of the parameters is not affected. The parameter server function of the IO-Link master is also not blocked.
- If the lock of the parameter server (DataStorageLock) is set, the download of the parameter server of the IO-Link master is locked. Thus, data of an IO-Link device backed up in the parameter server is not written to the connected IO-Link device if a device exchange is detected. The parameter server upload is not affected.

- The lock of the local parameterization (LocalParamLock) is important for IO-Link devices with an operating interface at the device. If it is set, no parameters can be changed at the IO-Link device via local operation.
- The lock of the local operation (LocalUserInterfaceLock) is also important for IO-Link devices with a local operation at the device. If it is set, the local operation of the IO-Link device is locked.

IL_PNIOIoLinkGetDeviceAccessLock: The access lock states in the IO-Link device can be retrieved via the function block.

Backup/restore

IL_PNIOIoLinkBackupDevice: The function block is used to back up the parameterization of an IO-Link device to a file on the control.

IL_PNIOIoLinkRestoreDevice: The function block is used to load the parameterization from a file on the control to an IO-Link device.

These allow to back up the parameterization of an IO-Link device to the control and to restore the parameterization in a structurally identical machine or after exchanging a device of the IO-Link device.

11.3.3 Exemplary use of a Profinet I/O IO-Link gateway of the company Balluff

An exemplary configuration of an IO-Link gateway using a device of the company Balluff is described in the following. Gateways of other vendors are similarly configured. Differences are caused by the figure of the gateway in the device description file (see for example above in single slot or multi slot mapping) and are not explained further.



The example is based on the firmware of the device "Balluff v2.0.0.0 / GSDML-V2.2-Balluf-BNI-PNT-508-105-20120807.xml"

The following firmware version is recommended: v2.6.0.0 / GSDML-V2.31-Balluff-BNI-PNT-508-105-Z015-20160901.xml

Adding the Profinet I/O IO-Link gateway to the project

First, add the device description file (GSDML file) of the Profinet I/O IO-Link gateway via **Tools** ▸ **Device Database** ▸ **Add Devices…** . The Profinet I/O controller is either configured when creating the control or via the context menu item **Set Device**. The Profinet I/O IO-Link gateway can be added either manually, via drag&drop from the device library or via the context menu **Add** or - if already available - online via **Scan Devices**

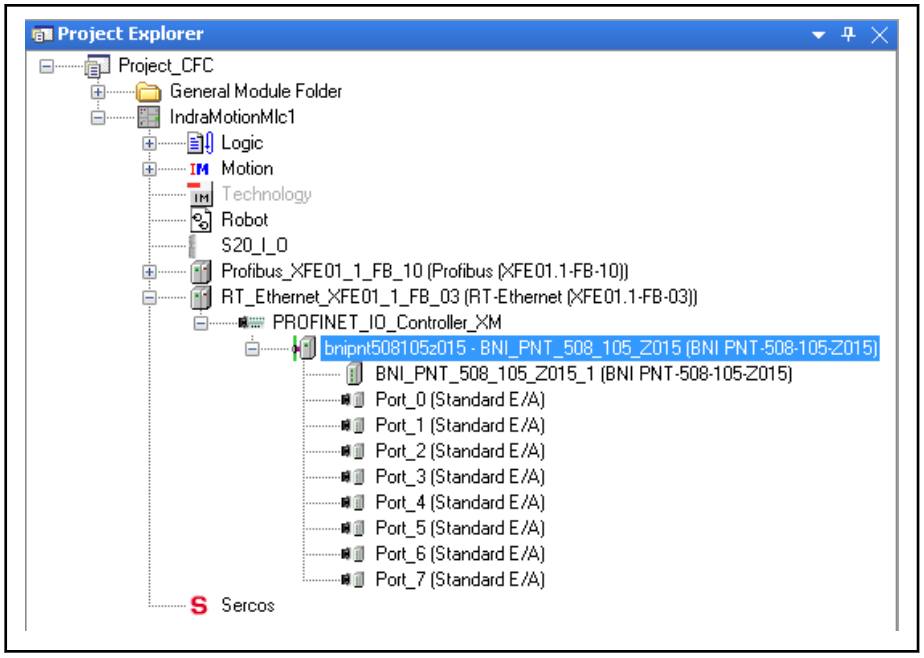


Fig. 11-4: Profinet I/O IO-Link gateway in the Project Explorer

Configuring the Profinet IO-Link gateway

The function of the port can be configured in the IO-Link master (slot number 1). For ports used to connect IO-Link devices, select the IO-Link function.



Current firmware versions, e.g. v2.6.0.0 and GSDML-V2.31-Balluff-BNI-PNT-508-105-Z015-20160901.xml, only support NO/NC contacts and outputs.

IO-Link is automatically applied by the device if an IO-Link module is selected under “Set device”.

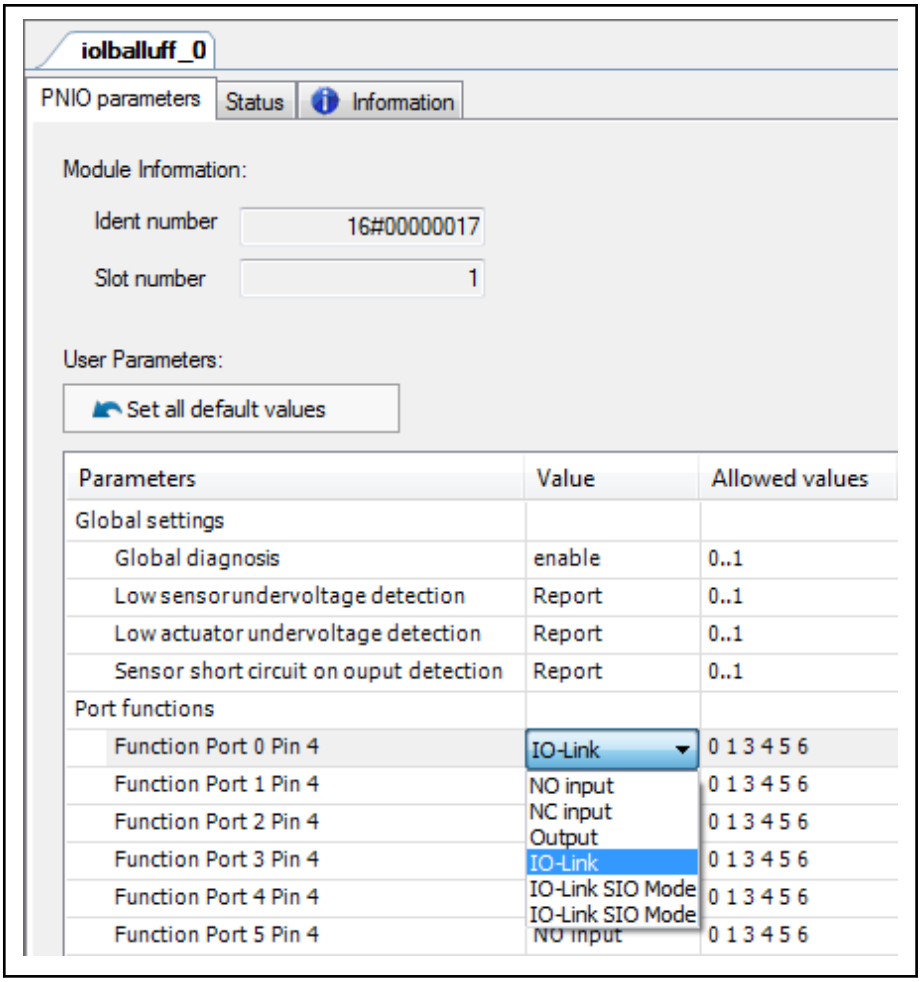


Fig. 11-5: Selecting the function of the port as IO-Link

Configuring IO-Link device

The IO-Link device with the respective input and output data (seen from the control) has to be set for the port. Four bytes are cyclically sent to the IO-Link device in the example given in fig. 11-6 "Setting IO-Link device" on page 281. For input and output data, refer to the documentation of the IO-Link device.

If more cyclic data than of the IO-Link device is configured, no error is generated upon startup. The additionally configured data is only relevant for the performance. If less cyclic data than of the IO-Link device is configured, an error is generated upon startup.

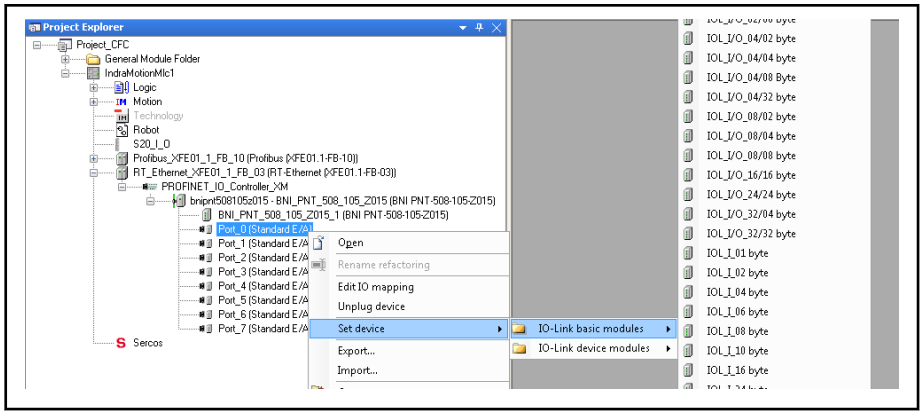


Fig. 11-6: Setting IO-Link device

For the IO-Link device, a cycle time consisting of cycle time basis and multiplier has to be set. For the minimum cycle time of each IO-Link device, refer to the documentation of the device. Other important settings refer to the configuration of the parameter server. The use of the parameter server can be completely enabled/disabled or the parameter server data can be deleted. The upload of the parameterization to the gateway and the download of the parameterization to the IO-Link device can be enabled/disabled separately. Complete archiving of the parameter server function (incl. Upload/download) is recommended, as thus a device exchange without user interaction can be ensured.



With the current firmware versions, e.g. v2.6.0.0 and GSDML-V2.31-Balluff-BNI-PNT-508-105-Z015-20160901.xml, configuring the cycle time is not mandatory.

The cycle setting can remain on default.

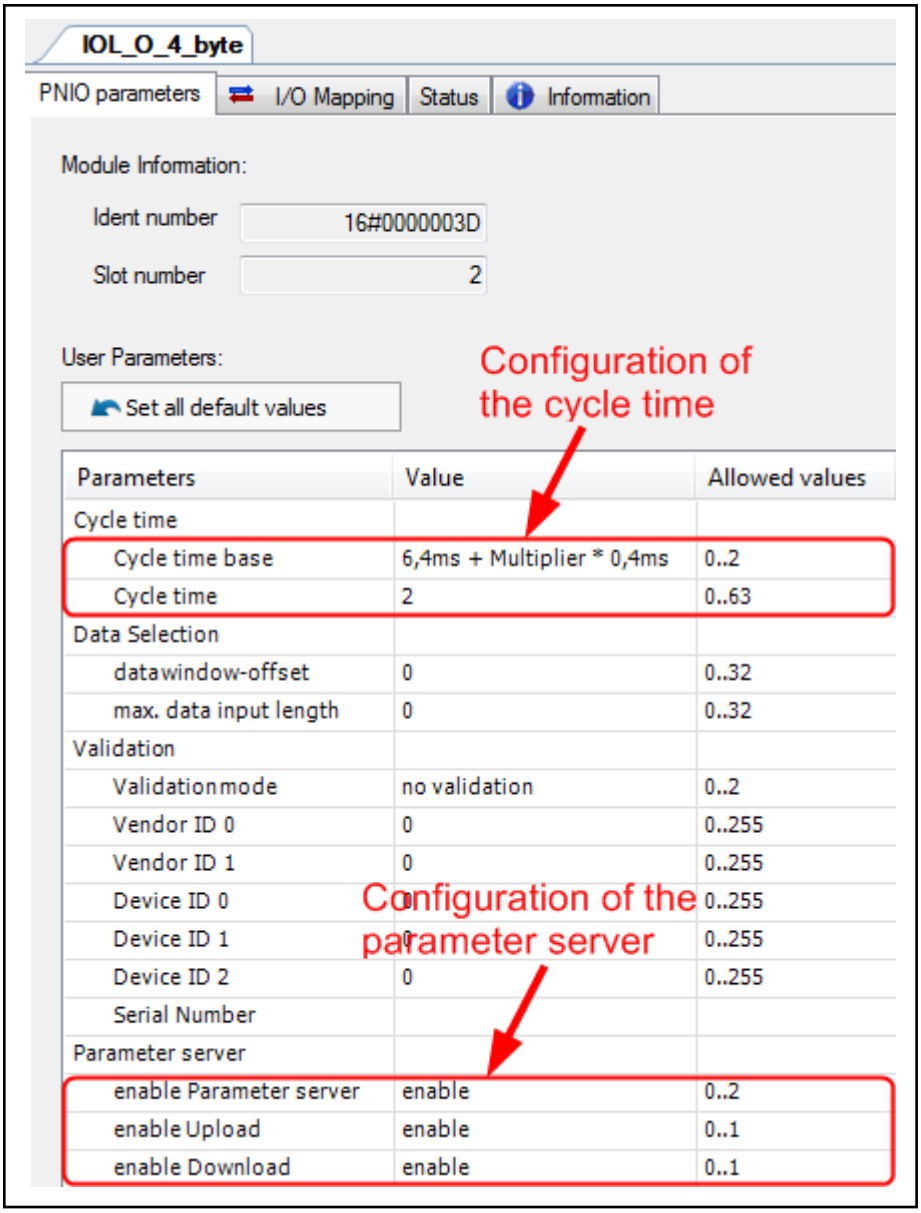


Fig. 11-7: Configuring IO-Link device

11.3.4 Exemplary use of the Rexroth Profinet I/O IO-Link gateway S67E-PN-IOL8-DI4-M12-6P

This chapter exemplarily describes the configuration of an IO-Link gateway using the device S67E-PN-IOL8-DI4-M12-6P.

Third-party gateways are configured similarly. However, individual module mapping in the GSD file[1)] can cause differences in the module configuration.

Refer to chapter 11.3.1 "Introduction and overview" on page 276.

Prerequisites to use the I/O link gateway:

- The I/O-Link gateway has to be in the IndraWorks device database.

1) *Device description file*

If this is not the case, add the device description file of the IO-Link gateway to the device database, refer to chapter 4.1.2 "Managing devices" on page 15

- The control has to be configured with a real-time Ethernet function module. In this example, an XM2 control with the function module XFE01_1_FB_03.

  The function module can be configured when creating the control or subsequently with the context menu item **Add** of the control

- The real-time Ethernet function module has to be configured with a Profinet I/O controller

  The Profinet I/O controller is configured via the context menu of the child node of the function module → **Set device**:

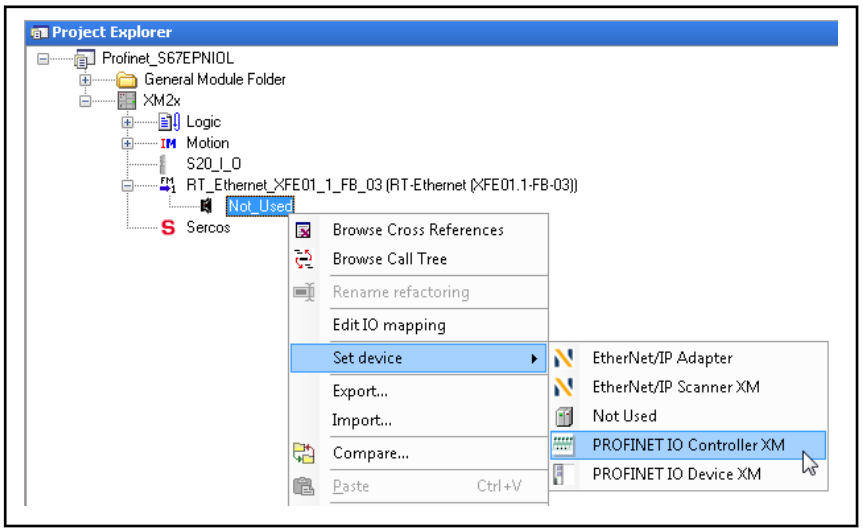


Fig. 11-8:

Adding the Profinet I/O IO-Link gateway to the project

The Profinet I/O-Link gateway can be added to the project as follows:

1. Via drag&drop from the device library in the category **Periphery ▸ Profinet I/O ▸ I/O**
2. In the Project Explorer via the context menu of the Profinet I/O controller**Add ▸ Device ▸ I/O**
3. In the context menu of the Profinet I/O controller via the command **Scan devices** if the Profinet I/O IO-Link gateway is already connected to the bus (online), refer to chapter 6.5.4 "Profinet I/O, controller, scanning devices" on page 109

For the S67E-PN-IOL8-DI4-M12-6P applies that the device is configured with eight digital input modules when it is created.

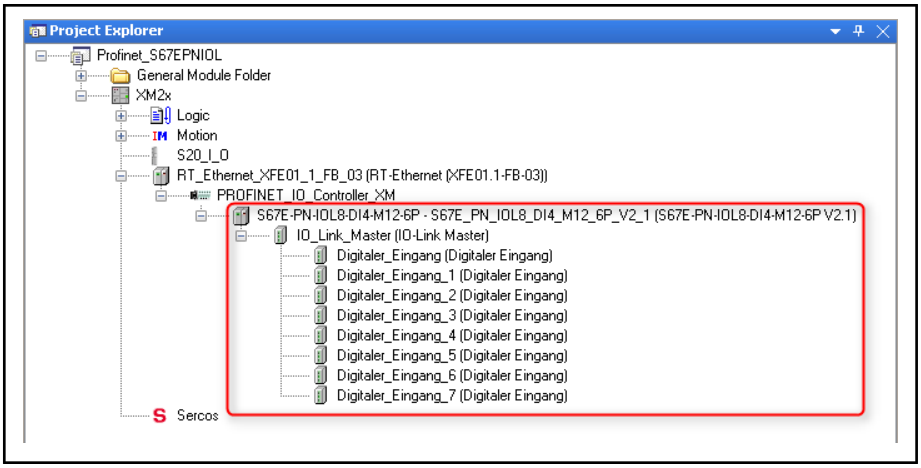


Fig. 11-9: Project Explorer - Profinet I/O IO-Link gateway

No further modules can be added in this configuration, as the device does not support more than eight modules.

To configure another module type, an existing module has to be deleted first.

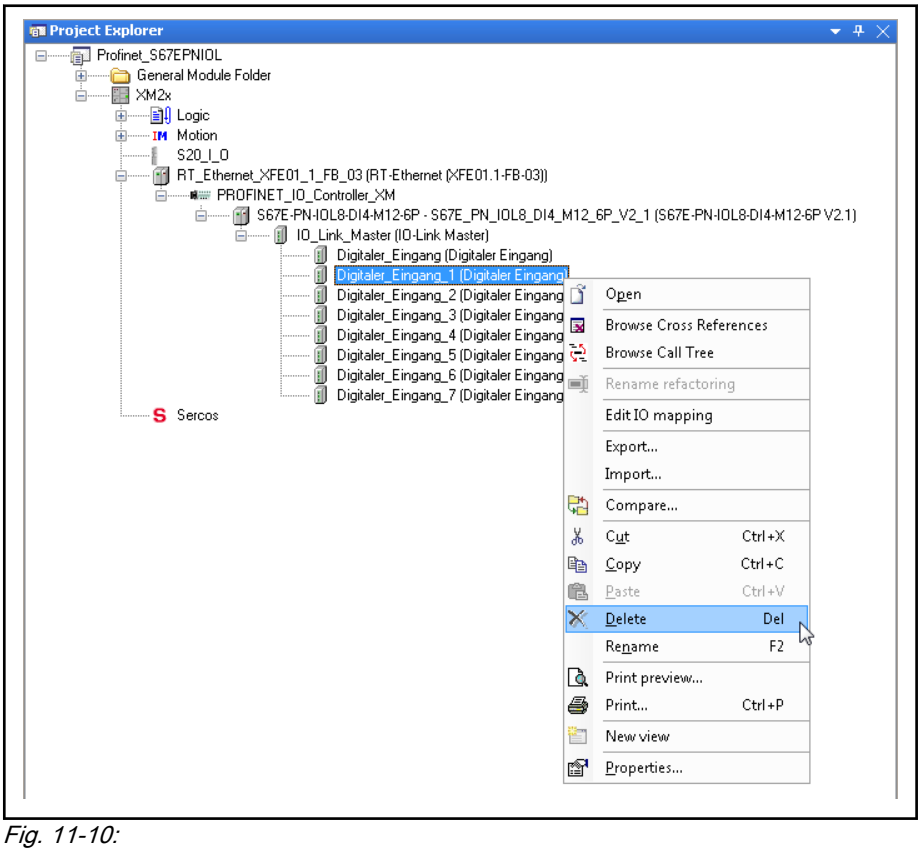


Fig. 11-10:

After the module has been deleted, another module can be added via the context menu of the IO-Link master. Go to **Add ▸ Module**

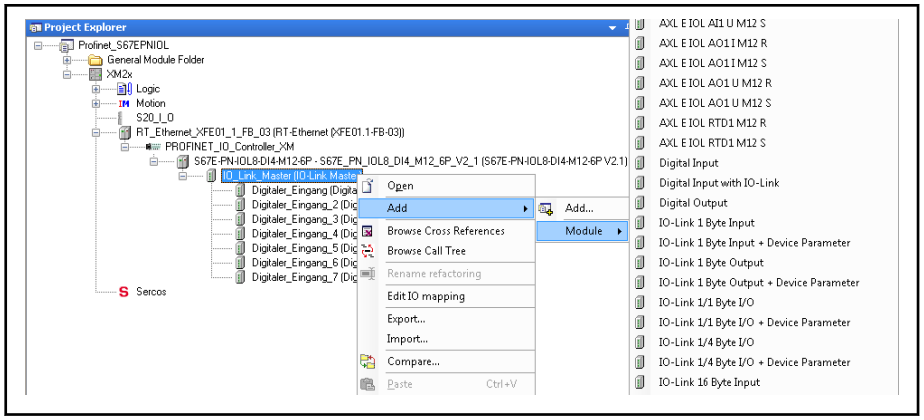


Fig. 11-11: Project Explorer, adding IO-Link module

The new module is added below the existing modules in the Project Explorer.

If the new module should be in a specific module position, go to the context menu items → **Add ▸ Add...**

The "Add" dialog opens.

The position can be specified via the setting "**Insert before**" in the shown module list.

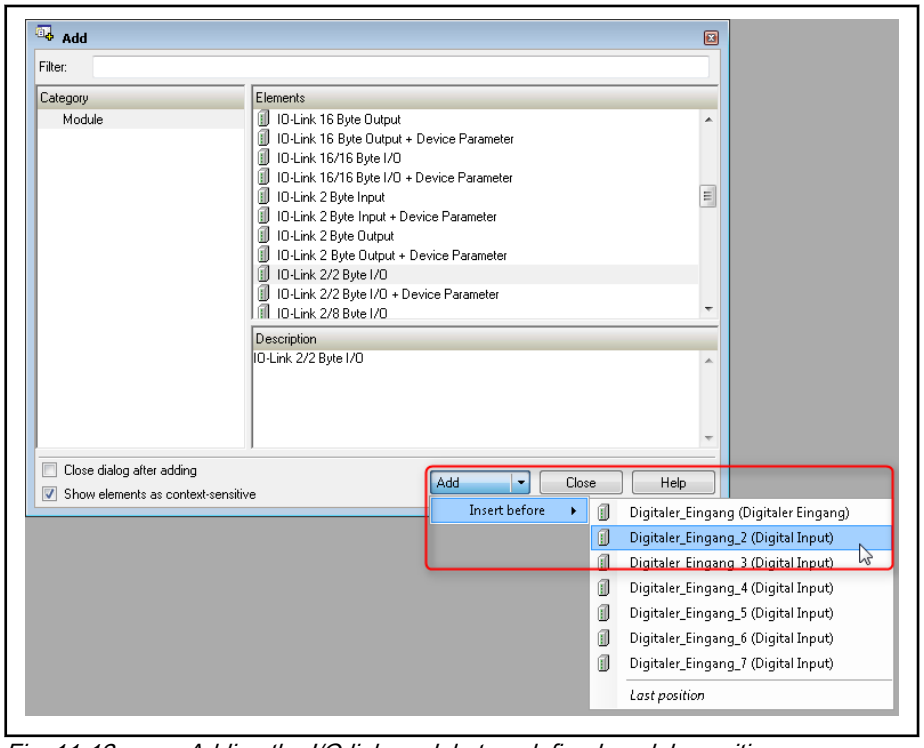


Fig. 11-12: Adding the I/O link module to a defined module position

Configuring IO-Link module

The process data width of the connected IO-Link device is one criterion when configuring the IO-Link module (refer to the device documentation). Ideally, the configuration of the process data width of the IO-Link module corresponds to the connected IO-Link device.

- If the process data width is too low for the IO-Link module, an error is issued upon startup.
- If the IO-Link module is configured with a band width higher than the band width of the device, no error is issued, but the process data width is not optimally used (relevant for performance).

Configuration examples

IO-Link module	IO-Link device	
Process data width		Operation
24 bits	24 bits	OK
32 bits	24 bits	OK (data band width not used best)
24 bits	**32 bits**	**Error**

For each port, the cycle time is set automatically and matching to the I/O-Link device (fastest cycle time).

11.4 Sercos IO-Link gateways

11.4.1 Introduction and overview

The Sercos IO-Link commissioning differs from the Profinet I/O-Link commissioning. For Sercos, it is for example not required to create slots.

Required configurations in the generic device editor of the I/O-Link gateway:

- The IO-Link ports have to be configured in the "User-defined parameters".
- The lengths have to be configured as the sum of all ports in the "Function groups" tab.

For more details on the configuration, refer to the documentation **"DOK-CONTRL-S67E*S3*IOL-AP"**, chapter "Settings at the IO-Link ports" → "Network-specific properties of the IO-Link master for Sercos S67E-S3-IOL8-DI4-M12-6P".

The procedure is identical for S20-IO and S67E-S3-IO.

11.4.2 Function blocks for Sercos IO IO-Link gateways

To communicate with IO-Link devices at Sercos IO-Link gateways, use the following function blocks from the RIL_IoLink library.

Parameter access **IL_IoLinkCall:** The function block is used to read and write parameters of the IO-Link device. It is accessed via index and subindex of the respective parameter.

Identification **IL_IoLinkIdent:** The function block is used to identify the IO-Link device. The device type of the IO-Link device can be uniquely identified via VendorID and DeviceID.

Backup/restore Standard Sercos Backup and Restore function blocks.

These function blocks allow to back up the parameterization of an IO-Link device to the control and to restore the parameterization in a structurally identical machine or after exchanging a device of the IO-Link device.

12 Local S20 I/O

12.1 Features of the local S20 I/O

For S20 I/O modules, these are I/O terminals directly connected to XM and VPx controls and to modular S20 bus couplers S20-yy-BK[1] .

The corresponding bus couplers are available for Sercos III and Profinet I/O.

The following I/O types are available:

- Digital input modules
- Digital output modules
- Analog input modules
- Analog output modules
- Infeed module
- Technology modules
  - Counter modules
  - Modules with incremental encoder inputs
  - Modules with SSI encoder inputs
- Modules for temperature measurement
- Safety I/O modules
  - Digital Safety I/O modules

Title	Identification
Automation Terminals of the Rexroth S20 Product Family	DOK-CONTRL-S20*SYS*INS-APxx-EN-P[2] R911335988
Rexroth S20 Bus Coupler for Sercos III	DOK-CONTRL-S20*S3*BK**-DAxx-EN-P R911335970
Rexroth S20 Bus Coupler for Profibus DP	DOK-CONTRL-S20*PB*BK**-DAxx-EN-P R911342760
Rexroth S20 Bus Coupler for Profinet I/O	DOK-CONTRL-S20*PN*BK**-DAxx-EN-P R911335968
Rexroth S20 Bus Coupler for Safety Applications (Sercos III, Profibus DP, Profinet I/O), chapter "Safe bus systems"	Rexroth MTX SafeLogic System Overview DOK-MTX***-SL**SYS*V14-PRxx-EN-P R911341698
	Rexroth MLC SafeLogic System Overview DOK-XLCMLC-SL**SYS*V14-PRxx-EN-P R911341696

Tab. 12-1: Related documentation

1) *yy - Respective bus system*

2) *xx - Edition*

For the characteristics and the mode of operation of the different S20 I/O modules, refer to the respective module documentations.

For information on the use of the S20 Safety I/O modules, refer to the corresponding Safety documentations.

12.2 Configuring S20 I/O

12.2.1 S20 object and S20 Inline modules, overview

XM controls allow to locally extend available I/O units by connecting S20 modules in series at the right side of the control.

The S20 I/O object is to be extended in the Project Explorer by the desired S20 modules.

S20 I/O object

- Tab “Settings”, page 288
- Tab “S20 module parameters”, page 290
- Tab 'Field bus diagnostics', page 291
- Tab “I/O mapping”, page 293
- Tab “Status”, page 293
- Tab “Information”, page 295

S20 I/O modules added to the "S20 I/O object" (example)

- Adding S20 modules, page 295
- Enabling/disabling S20 modules, page 296
- Tab “User-defined parameters”, page 298
- Tab “I/O mapping”, page 299
- Tab “Information”, page 299

12.2.2 S20 I/O object

Tab “Settings”

The S20 bus master on an XM control supports the following operation modes to trigger the connected S20 modules:

1. **Free running:**

   In the "Free running" mode, the S20 local bus runs without temporal reference to the PLC.

   A bus cycle task definition is not required in this operation mode (except for the following note). The bus cycle task setting can remain on default **<unspecified>**.

   With each cycle of the task using S20 inputs/outputs, the outputs are output with regard to them and the inputs are loaded.

   This applies to the default settings of the option "Always update variables" in the PLC settings of the control and in the “I/O mapping” tabs of the assigned S20 modules:

   - Default value:

     Deactivated (update only if used in a task)



If the setting "**Enabled 2 (always in bus cycle task)**" is set in the setting "**Always update variables**" of the S20 bus master (control), all S20 I/Os are assigned to the bus cycle task and the S20 inputs/outputs are only replaced with regard to this task.

See:

- "Configuring event-controlled PLC task as bus cycle task for the S20 local bus" on page 290
- chapter 4.5.1 "Describing the bus cycle task" on page 27

The S20 master updates the S20 modules the fastest possible, depending on the S20 module equipment.

The external event "**EVENT_OP_MODE_S20_CYCLIC**" is not supported in the "Free running" mode. However, if the event is used, the S20 modules are not updated!

⇒ Inputs and outputs are updated the fastest possible at the S20 module.

2. **Cyclic:**

   The cycle of the cycle time is specified for the S20 local bus operation. It is "engaged" to the Sercos cycle of the system.

   The S20 cycle can be a fraction or a multiple of or equal to the Sercos cycle.

   S20 local bus runs irrespective of the Sercos phases.

   Together with an externally event-controlled PLC task, it is recommended to use the external event "EVENT_OP_MODE_S20_CYCLIC" for synchronization purposes.



If multiple PLC tasks are used, configure the event-controlled PLC task as bus cycle task of the S20 master, refer to "Configuring event-controlled PLC task as bus cycle task for the S20 local bus" on page 290.

The S20 master initializes S20 modules supporting the "Fast response", i.e. the outputs are output the earliest possible and inputs are loaded the latest possible).

⇒ Inputs and outputs are updated in the fixed grid at the S20 module in the "Fast Response" mode (little to no jitter).

3. **Cycle-synchronous to Sercos:**

   The S20 local bus cycle is equal to the Sercos cycle.

   S20 local bus runs with respect to the Sercos phases (CP4/BB - I/O data exchange; NRT, CP0-CP3 - No I/O data exchange).

   Together with an externally event-controlled PLC task, it is recommended to use either the external event "EVENT_OP_MODE_SERCOS_CYCLIC" or "EVENT_OP_MODE_S20_CYCLIC" for synchronization purposes.



If multiple PLC tasks are used, configure the event-controlled PLC task as bus cycle task of the S20 master, refer to "Configuring event-controlled PLC task as bus cycle task for the S20 local bus" on page 290.

The input signal detection and the output of the outputs at the S20 modules take place at T4 (as for Sercos).

Synchronous-capable S20 modules are required for a cycle-synchronous detection and output of the inputs and outputs. A mixed operation with asynchronous S20 modules is possible.

S20 modules at the S20 local bus and Sercos devices supporting the synchronous operation are synchronous among each other at T4. The external event "EVENT_OP_MODE_SERCOS_CYCLIC" has to be configured for the PLC task.

⇒ Inputs and outputs are updated in the fixed grid at the S20 module at T4; in case of "EVENT_OP_MODE_SERCOS_CYCLIC" synchronous to Sercos (little to no jitter).



The exact points in time in the operation modes 2 and 3 of the bus timing can only be met by S20 modules enabled for the synchronous operation. Mixed operation with synchronous and asynchronous S20 modules is permitted.

Configuring event-controlled PLC task as bus cycle task for the S20 local bus

To configure an event-controlled PLC task as bus cycle task for the S20 local bus, proceed as follows:

1. Double-click on the control node in the Project Explorer and go to the "**PLC settings**" tab to open the generic device editor of the control.
2. Change the configuration in the field "Always update variables" to the following setting: "**Activated 2 (always in bus cycle task)**"
3. Define the task in the field "Bus cycle options Bus cycle task" in the example illustration "**PlcTask**".

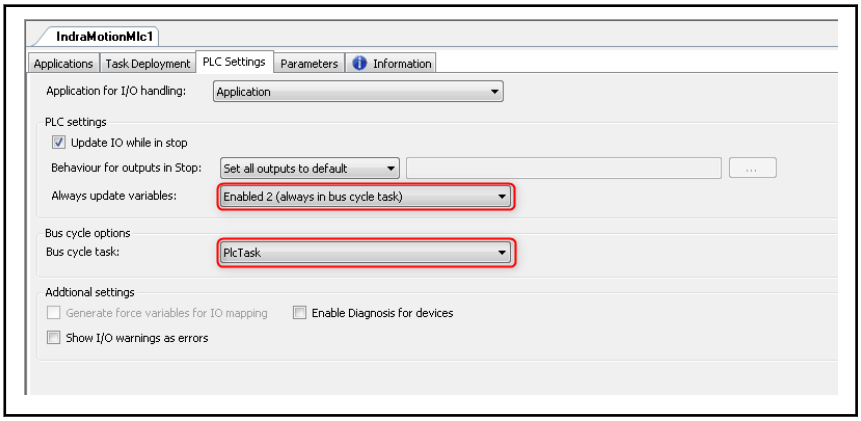


Fig. 12-1:

The S20 inputs/outputs are updated with regard to the defined task.



The definition of a specific task in the bus cycle options is only applied if the configuration is enabled in the field "Always update variables".

Tab “S20 module parameters”



The tab "S20 module parameters" is not displayed anymore by default.

To display this tab, enable the display "Generic device editor" in the “Options”.

The window contains information on the configuration and diagnostics of the S20 bus.

Window structure

- **Diagnostics**: Parameter name from the device description file. It cannot be edited.
  - Configuration error
  - Master error
  - BusStatus
  - BusInfo
  - BusSyncInfo
  - Module errors
- **S20 operating mode**: Operation mode S20 local bus, not editable.
- **S20 cycle time**: S20 local bus cycle time, not editable.
- **Diag Provide Counter**
- **Diag Provide Access**
- **Diag Provide CommonData**
- **Diag Provide DataField 0**
- **Diag Provide DataField 1**
- **Diag Provide DataField 2**
- **Diag Provide DataField 3**

If the parameter can be edited (this depends on the device description. Parameters that cannot be edited are displayed in light gray), double-click on the table element to open an input field or a selection list (or press the <space bar> in a previously selected field) where the value can be changed.

Values are applied with <Write parameter>.

If the value is a file specification, the standard dialog to select a file opens.

Tab 'Field bus diagnostics'

In online mode, the "Field bus diagnostics" tab shows the modules at the S20 node together with their respective diagnostic data.

To display the respective diagnostic message, select the module. The upper section shows all modules available in the project tree.

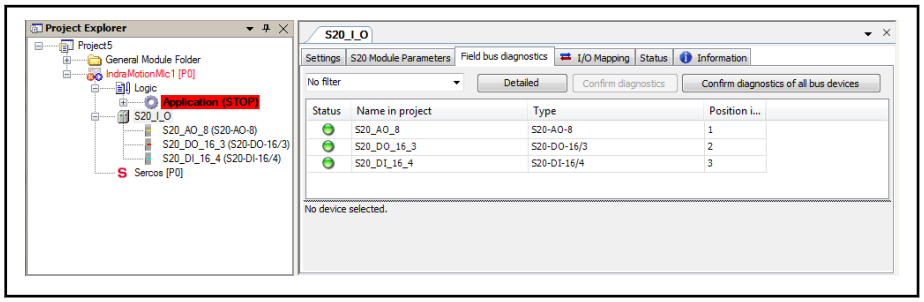


Fig. 12-2: S20 field bus diagnostics, control online, logged in

The lower section displays the message on the currently selected module. The control has to be online. If this is not the case, a reference text is displayed in the section showing the diagnostic messages.

In the upper section of the tab, position numbers are displayed in addition to the name and the type of the modules. This position number indicates the current location of the respective module in the project tree. This subsequent

numbering is automatically executed by the system. If the sequence or the number of the modules change, the numbering is adjusted automatically.

Additionally, the status is displayed with LEDs in the first line changing colors correspondingly.

Green indicates that there is no error.

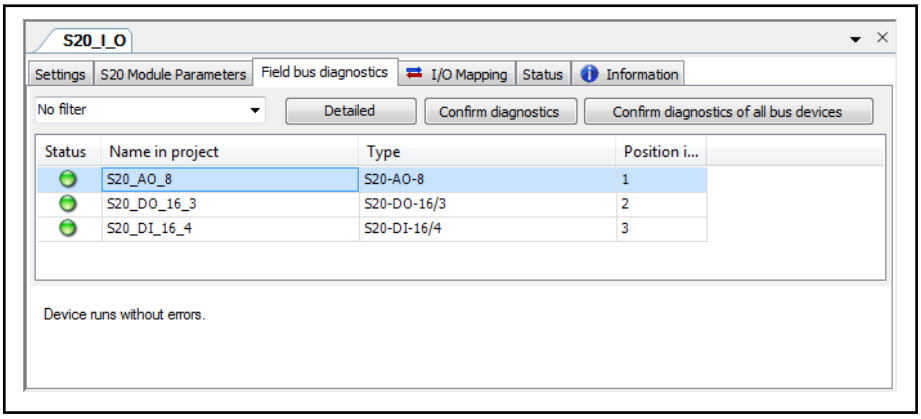


Fig. 12-3: S20 field bus diagnostics, control online, logged in, error-free

Example of a diagnostics in case of error

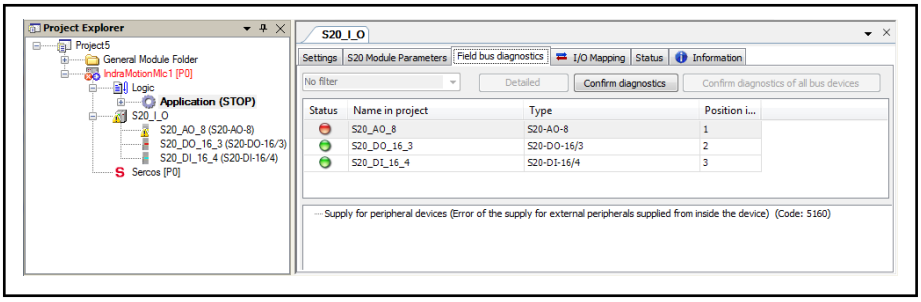


Fig. 12-4: S20 field bus diagnostics, control online, logged in, error case

The voltage supply is interrupted to one of the modules. Thus, the module is not detected by the control. In this case, a yellow warning triangle is displayed at the respective module indicating that a diagnostics is present. The status LED of the affected module turns red. If the module is selected, the lower section displays text with specifications on the error cause. The diagnostic-specific error code is also provided. Refer to chapter 9.3.3 "Sercos III master, Sercos bus diagnostics" on page 236

Grey is another possible color of the LED. The LED becomes grey if either the control is not logged in or a module was disabled.

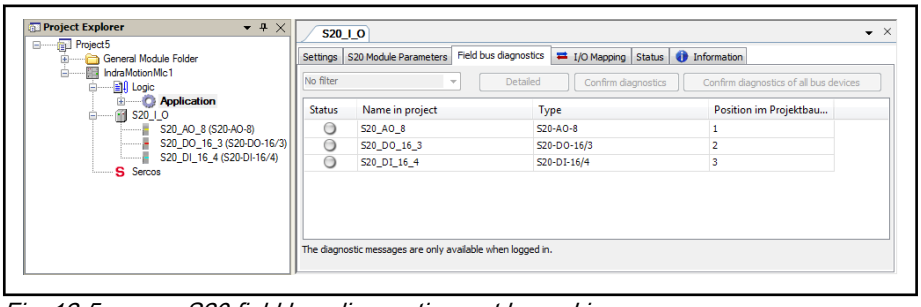


Fig. 12-5: S20 field bus diagnostics, not logged in

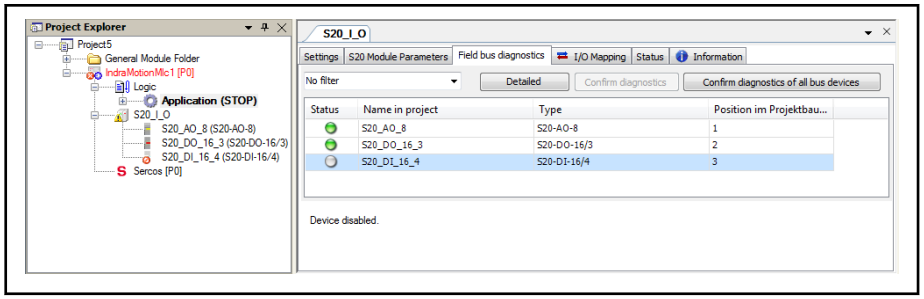
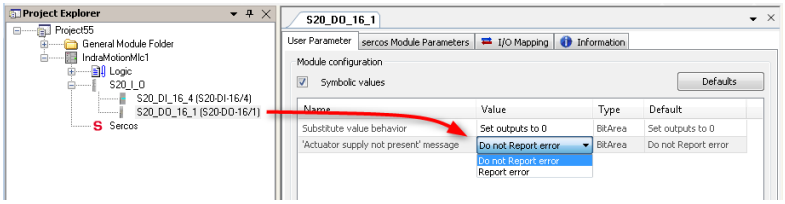


Fig. 12-6: S20 field bus diagnostics, control online, logged in, one module disabled

Removing the supply voltage at an S20 module does not mandatorily cause a diagnostics.

For some modules, the response on the missing supply voltage is assigned to an own parameter.

Example: The "S20_DO_16_1" module.



The respective parameter (if available) is "Do not Report error" by default.

Thus, no message is generated.

If a diagnostic message is to be output, set the parameter to "Report error".

Tab “I/O mapping”

Bus cycle options

- **Bus cycle task:** Specifies the PLC task used for the I/O mapping
  - **Default:** Use parent bus cycle settings
  - User task

Refer to chapter 4.5 "Bus cycle task: Copying I/O data between PLC and field bus stack" on page 27

Tab “Status”

The tab displays the status of the entire S20 bus.

Offline: n/a

Online: "Running", "Not running (n/a)"

In addition, the "Last diagnostic message" is displayed. Press "Acknowledge" to confirm them.

Error diagnostics

In case of error, a detailed diagnostics is transmitted. The third module is incorrectly equipped at the real control.

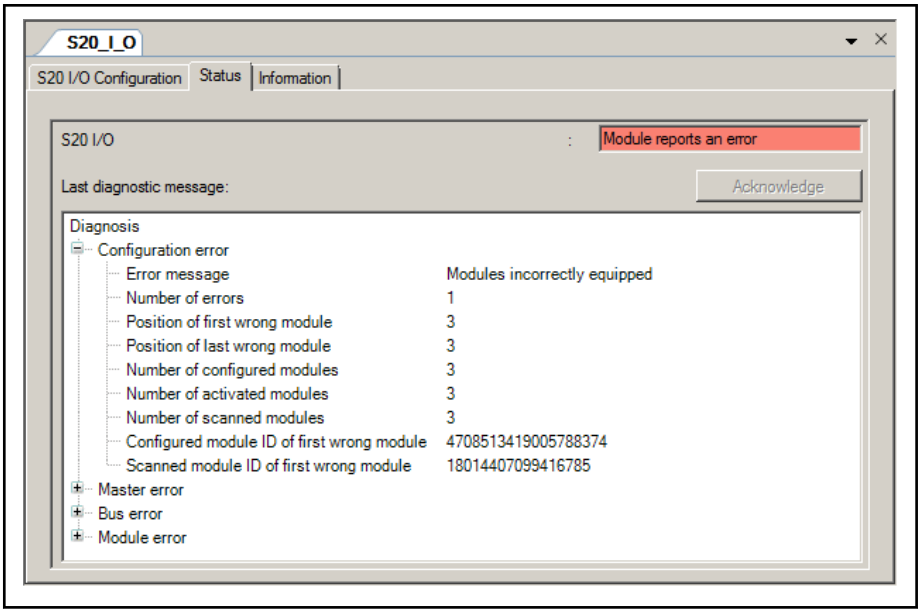


Fig. 12-7: Detailed error message in the S20 I/O bus

1. **Configuration error**: In case of such an error, the configuration of the modules in the project does not match the modules physically present at the bus.

   Possible messages are:

   - Configuration without errors
   - Too many modules equipped

     There are more modules at the bus than configured in the project.
   - Too few modules equipped

     There are less modules at the bus than configured in the project. The missing modules are provided with a yellow warning triangle in the project tree.
   - Modules incorrectly equipped

     There are other modules at the bus than those configured in the project. The incorrectly configured modules are provided with a yellow warning triangle in the project tree.
   - Other configuration errors

   Additionally, further information is provided:

   - Number of errors
   - Position of the first and last incorrectly configured module
   - Number of configured modules in the project
   - Number of enabled modules in the project
   - Number of modules present at the bus
   - For the first incorrectly configured module:
     - Module ID in the project
     - Module ID at the bus
2. **Master error**: Internal error of the S20 master.

   Possible messages are:

   - Master without errors
   - General master error

An error number is provided as additional information.

3. **BusStatus**

   Possible messages are:

   - Bus without errors
   - General bus error: In case of such an error, the bus is not running, e.g. since a module failed.

4. **BusInfo**

   Possible messages are:

   - My message is available
   - Additional module detected
   - Additional module removed

5. **BusSyncInfo**

   Possible messages are:

   - Synchronization OK
   - Synchronization lost
   - Synchronization undefined

6. **Module error**: At least one module reports an error (e.g. short circuit). The bus continues to run. The latest detected diagnostics is displayed. Incorrect modules are also provided with a yellow warning triangle outside the project tree.

   - **Error message:**
     - Module without errors
     - Module-specific error
   - **Module position:** Slot, position of the S20 module
   - **Error code:** 4-digit diagnostic code. Refer to the documentation of the S20 module or to Sercos bus diagnostics, as the error codes are identical. Refer to chapter 9.3.3 "Sercos III master, Sercos bus diagnostics" on page 236
   - **Error location:** Indicates the channel. If the value is 255, the complete S20 module is affected

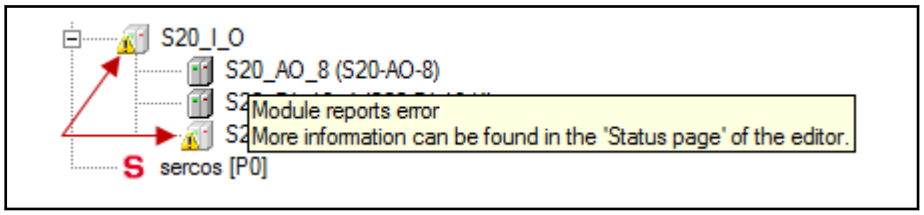


Fig. 12-8: S20 I/O bus, error message in the Project Explorer

Tab “Information”

The tab displays general information from the device description file:

Name, Vendor, Categories, Version, Order number, Description and Image if available.

12.2.3 Adding S20 modules

All S20 modules available for the respective control are located in the "S20" folder of the "Periphery" library.

Drag the required S20 modules from the library to the respective "S20 I/O" object. New S20 modules can also be inserted between existing S20 modules in the Project Explorer.

The S20 I/O modules can also be added via the context menu items **Add ▸ Module ▸ <S20 module>** of the "S20 I/O" object. In this case, the new module is added as last module under "S20-IO".

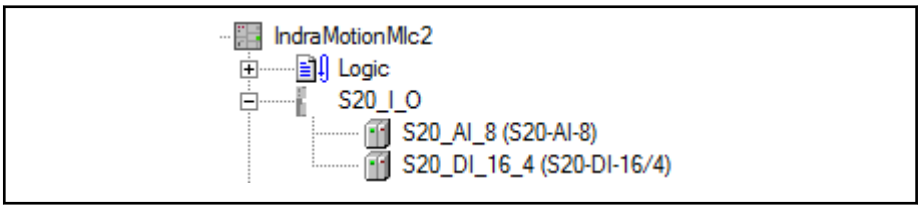


Fig. 12-9: S20 object with two modules (example)

If a required S20 module is not in the IndraWorks device library by default, add it by importing an GSDML file to the IndraWorks device database.

Select the menu command:

Tools ▸ Device database ▸ Add devices.

Also refer to chapter 4.1 "Device database" on page 15.

12.2.4 Enabling/disabling S20 modules

The S20 modules configured at a control can be enabled and disabled in the Project Explorer.

Thus, select/deselect the button at the icon of the S20 module.

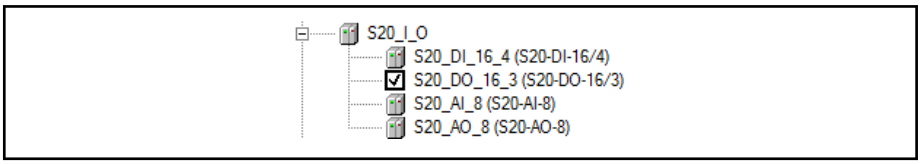


Fig. 12-10: Enabling/disabling S20 modules

The module is enabled if the checkbox is selected. It is considered in the diagnostics of the bus.

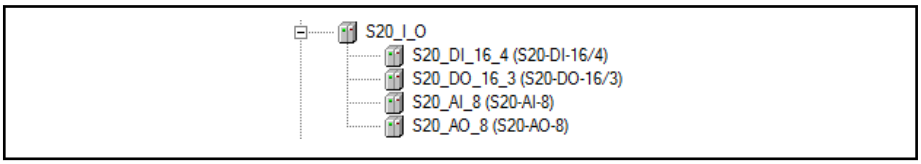


Fig. 12-11: S20 bus with enabled modules

The module is disabled if the checkbox is deselected. It is not considered in the diagnostics of the bus.

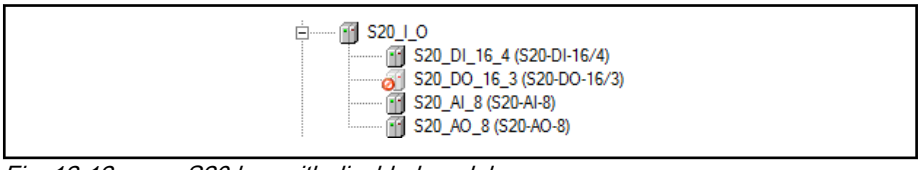


Fig. 12-12: S20 bus with disabled module

If a new S20 module is created, it is enabled by default.

When loading the configuration, all enabled S20 modules are taken into account. Disabled S20 modules are ignored.



A module that is enabled but not present generates an error message in the diagnostics.

A module that is disabled and not present does not generate any error message in the diagnostics.

12.2.5 Disabling S20 modules using “UserDefs.cfg”

As of IndraWorks 14V22, S20 I/O modules at the local bus of the control system XMx configured in IndraWorks can be disabled. Disable them by the definition in a stored file located on the Compact Flash card of the control.

This impedes diagnostic messages for modules that are configured, but do not exist.

Enabling the function

The function is enabled if the valid file "UserDefs.cfg" is present in the following partitions on the control Flash Card:

For XM and VPx controls: **\USER**

Another path or file name can be defined by making entries in the "cfg file" of a system.

Example:

```
[IOSpecialDefs]
IODeactFilePath=MyFilePath
IODeactFileName=MyFileName
```



The file "UserDefs.cfg" can contain entries for more buses:

Inline modules, Sercos, Profinet I/O, Profibus DP.

Point in time of module deactivation

The "UserDefs.cfg" file is analyzed and the entered modules are disabled at this point in time and in the context of the internal event "PrepareUpdateConfiguration".

Event takes place in case of the following actions:

- Startup after power on
- Reset (warm)
- Reset (cold)
- Load program after reset (origin)

Structure of the file "UserDefs.cfg"

The file is structured in a single or multiple existing section(s) and the contained entries.

- **Sections**: The name of a section is written in **[** and **]**. A section ends at the beginning of the next section or at the end of the file.

  The name consists of a symbol **S20IO.Master.** and the constant index **0** (compatible to the Profibus slave deactivation).

  Example:

  ```
  [S20IO.Master.0]
  ```

- **Special function**: Due to the definition of character **X** for the index instead of a digit, this section is ignored in the evaluation.

  Several sections can be defined for the master index **0**. All the included entries are assigned to the S20 master.

  Together with the "x function" described above, different module groups can be handled easily.

- **Module entry**: To disable modules below a slave,

the keyword `ModuleDeactPosi=` and the 1- or 2-digit specification of the module position are given in decimals.

Example:

```
ModuleDeactPosi= 3
```

The permitted values range is **1...64**.

0 is also permitted, but has no effect.

- **Comments**: A semicolon (;) is defined as comment character. The text is not analyzed from the position of the semicolon up to the end of the line.

Size of the "UserDefs.cfg" file

The memory evaluation is allocated according to the specified file size. The permitted size is thus virtually not limited.

Example of the file "UserDefs.cfg"

```
;---------- Master.0 ----------
[S20IO.Master.0]
ModuleDeactPosi = 3
ModuleDeactPosi = 7
ModuleDeactPosi = 15
ModuleDeactPosi = 16
;------------------------------
```

12.2.6 Configuring S20 modules

General information

In the Project Explorer, double-click on the S20 module to be configured.

Click to open the three tabs in the S20 module editor.

Tab:

- Tab “User-defined parameters”, page 298
- Tab “I/O mapping”, page 299
- Tab “Information”, page 299

Tab “User-defined parameters”

This (special) tab is used to preset module parameters and it is only displayed for modules which require or allow this default setting.

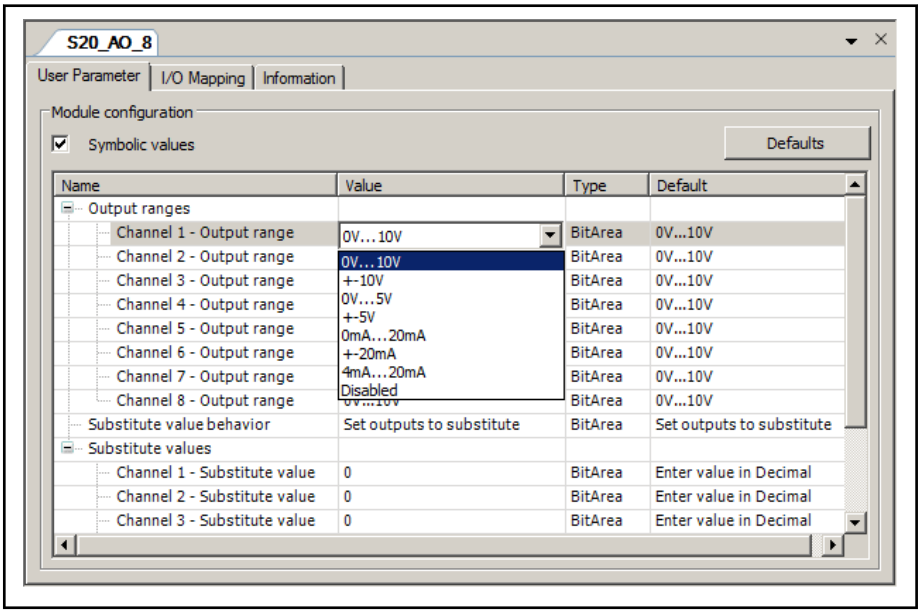


Fig. 12-13: S20 module: User-defined parameters

Module configuration

This example shows a module with eight analog outputs.

In the screen, the value range of the expected output information is selected for analog output 1.

A drop-down menu based on the enumeration data type is provided for each "Value".

Symbolic values: Defines whether the value is displayed in plain text ("0V... 10V") or as integer ("0").

Default setting: Resets all parameters to their initial value.

Tab “I/O mapping”

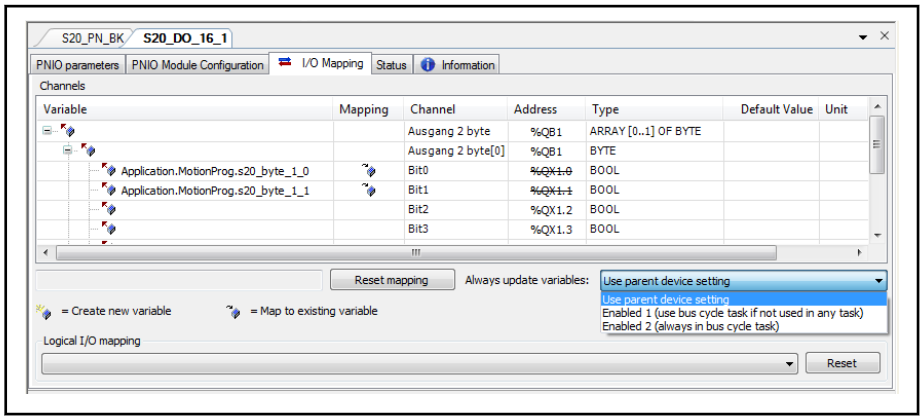


Fig. 12-14: S20 module: “I/O mapping” tab

The tab is used to assign S20 module inputs and outputs to variables that can be used as local or global variables in the individual POUs.

The current value of the variables is displayed in online mode.

Reset Mapping: Use this button to reset the mapping settings to the default values defined in the device description file.

Update variables: Defines for the module whether I/O variables are updated in the bus cycle task.

The default value is defined in the device description file.

- **Use parent device setting:** Update according to the settings of the superordinate device.
- **Enabled 1 (use bus cycle task if not used in any task):** The I/O variables are updated in the bus cycle task if not used in any other task
- **Enabled 2 (always in the bus cycle task):** All variables are updated in each cycle of the bus cycle task irrespective of whether they are used or mapped on an input or output channel.

This assignment is described in Mapping of the field bus inputs and outputs, page 19,.

Tab “Information”

The tab displays general information from the device description file:

- Name
- Vendor
- Categories
- Version
- Ordering number
- Description

- Device image (if available)

12.2.7 S20 modules, scanning bus

Prerequisite "Scanning bus..." is only available in the context menu if the PLC application has already been loaded.

Function The "Scanning bus" dialog displays already configured devices in the left window and the devices found at the bus after scanning in the right window.

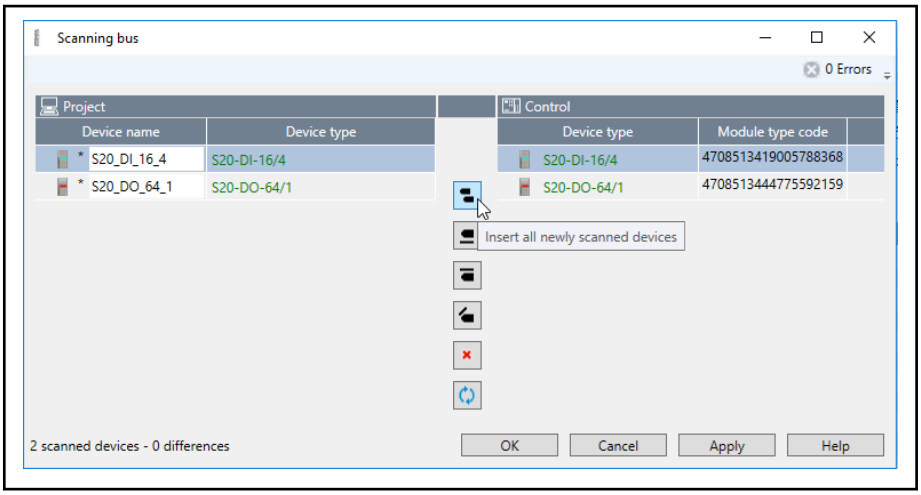


Fig. 12-15: S20IO - Dialog "Scanning bus"

The device found (right) can be applied to the project configuration (left) using the command icons in the middle of the dialog. Either all new devices can be added or individual devices can be highlighted and edited (refer to the tooltip support of the buttons).

Call In the Project Explorer via the context menu of the S20IO master

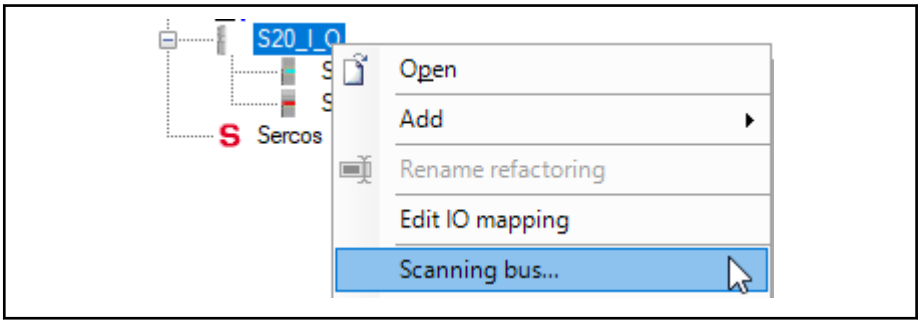


Fig. 12-16: S20IO - Starting "Scanning bus" from the context menu of the Project Explorer.

13 Building automation

13.1 General information

Introduction and overview

Bosch Rexroth provides different services and interfaces for building automation applications.

The following chapter provides an overview on the supported systems.

System overview

- **BACnet** (from IndraWorks 15VRS)

  BACnet is a network protocol to manufacturer-comprehensively connect to a master level or to set up a communication between the BACnet devices and the BACnet controllers.

  Another function is the cross communication between different BACnet systems to exchange data.

- **KNX**

  KNX is a standard for the building automation on field level.

  KNX collects and manages data or switching states of pushbuttons, switches, presence sensors, brightness sensors, temperature sensors or air quality sensors.

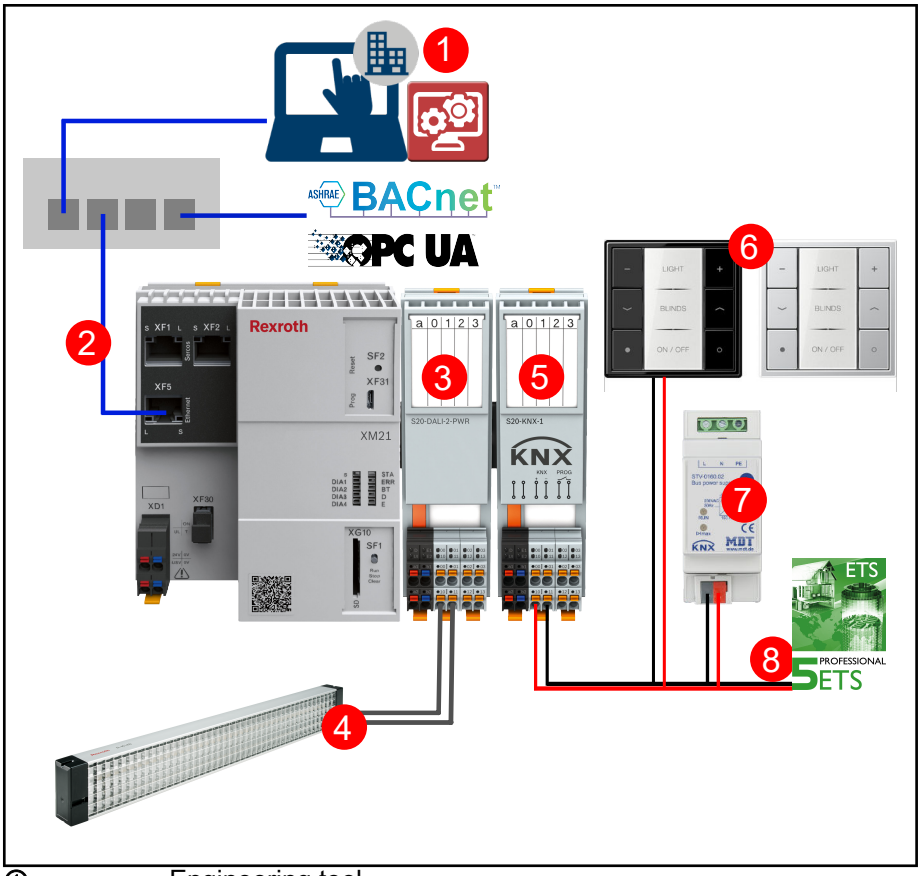
  The KNX application can be used to control multiple activities on an IndraControl XM control. These activities are for example:

  – Controlling actuators
  – Controlling infeed axes
  – Commanding shutters
  – Detecting state detectors (e.g. state LED at the pushbutton)

  Use the XM control to route information to other BACnet controllers or to other systems, e.g. to DALI lamps.

- **DALI**

  "Digital Addressable Lighting Interface" to control ballasts or dimmers.



① Engineering tool
② Configurable Ethernet connection (engineering port, BACnet, OPC-UA etc.)
③ S20 DALI-PWR-2 module (DALI power supply unit)
④ DALI devices
⑤ S20 KNX-1 module (contains the ETS/KNX configuration)
⑥ KNX devices, e.g. pushbuttons, presence sensors, shutters, window contacts
⑦ KNX power supply unit; supplies the KNX devices with bus voltage (required)
⑧ KNX programming tool, e.g. ETS 5 (to configure the S20 KNX-1 module)

Fig. 13-1: Overview on building automation components

Rexroth components to connect to building automation systems

System	Components	Function
Dali	S20-DALI-2-PWR	2-channel DALI master For the documentation, go to the Rexroth media directory DOK-CONTRL-S20*DALI*2*-DA
	RIH_DALI	PLC library with function blocks to communicate with DALI devices using the S20-DALI-2-PWR.

KNX	S20-KNX-1	1-channel KNX master For the documentation, go to the Rexroth media directory DOK-CONTRL-S20*KNX*1**-DA
	RIH_IndraControl_S20_KNX	PLC library with function blocks to communicate with KNX devices using the S20 KNX-1 module
BACnet	Package for the engineering tool, configurator-supported configuration	For the documentation, go to the Rexroth media directory: DOK-IWORKS-FB******V15-APRS-EN-P R911393284
OPC-UA	Package for the control	For the OPC-UA documentation, go to the Rexroth media directory: DOK-IWORKS-OPC*UA*****-APRS-EN-P R911379309

Tab. 13-1: Overview - Available components for the building automation

13.2 BACnet

13.2.1 Features of the BACnet implementation

The current BACnet implementation includes the following functionality:

Functional scope BACnet/IP server and client

Supported BACnet objects:

Name - object type	Object type ID number	Objekt type identifier	Comment
OBJ_DEVICE	8	DEV	BACnet device properties and information, unique for each BACnet device
OBJ_ANALOG_INPUT	0	AI	Analog input, measure
OBJ_ANALOG_OUTPUT	1	AO	Analog output, set
OBJ_ANALOG_VALUE	2	AV	Analog value, virtual analog value, e.g.. calculation result
OBJ_AVERAGING	18	AVG	Average value
OBJ_BINARY_INPUT	3	BI	Binary input, message
OBJ_BINARY_OUTPUT	4	BO	Binary output, set
OBJ_BINARY_VALUE	5	BV	Binary value, virtual binary value e.g. logic result
OBJ_CALENDAR	6	CAL	Calendar
OBJ_COMMAND	7	CMD	Command, run pre-defined action
OBJ_EVENT_ENROLLMENT	9	EE	Event enrollment, event category, reaction to events
OBJ_FILE	10	FIL	File, to an external program or for backup
OBJ_GROUP	11	GRP	Group, group of different objects for trend or logging of one device
OBJ_GLOBAL_GROUP	26	GGRP	Global group in a BACnet network

Name - object type	Object type ID number	Objekt type identi-fier	Comment
OBJ_LOOP	12	LP	Loop control PID
OBJ_MULTISTATE_INPUT	13	MI	Multistate input
OBJ_MULTISTATE_OUTPUT	14	MO	Multistate output
OBJ_MULTISTATE_VALUE	19	MV	Multistate value
OBJ_NOTIFICATION_CLASS	15	NC	Notification class, message distribution list
OBJ_PROGRAM	16	PR	Program
OBJ_SCHEDULE	17	SCHED	Schedule
OBJ_TRENDLOG	20	TLOG	Simple trendlog
OBJ_TRENDLOG_MULTIPLE	27	TLOGM	Multiple trendlog
OBJ_LIFE_SAFETY_POINT	21	LSP	Safety point
OBJ_LIFE_SAFETY_ZONE	22	LSZ	Safety zone, collection of safety points
OBJ_ACCUMULATOR	23	ACC	Counter, accumulator
OBJ_PULSE_CONVERTER	24	PC	Pulse converter
OBJ_EVENTLOG	25	ELOG	Event log
OBJ_LOAD_CONTROL	28	LC	Load control
OBJ_STRUCTURED_VIEW	29	SV	Structured view
OBJ_ACCESS_DOOR	30	AD	Door access
OBJ_LIGHTING_OUTPUT	31	LO	Lighting output, light control
OBJ_ACCESS_CREDENTIAL	32	AC	Access credentials
OBJ_ACCESS_POINT	33	AP	Access point
OBJ_ACCESS_RIGHTS	34	AR	Access rights, condition, time range
OBJ_ACCESS_USER	35	AU	Access user, with it's rights
OBJ_ACCESS_ZONE	36	AZ	Access zone
OBJ_CREDENTIAL_DATA_INPUT	37	CD	Credential data input
OBJ_NETWORK_SECURITY	38		Network security
OBJ_BITSTRING_VALUE	39	BSV	Bit string value
OBJ_CHARACTERSTRING_VALUE	40	CSV	Character string value
OBJ_DATE_PATTERN_VALUE	41	DPV	Date pattern value
OBJ_DATE_VALUE	42	DV	Date value
OBJ_DATETIME_PATTERN_VALUE	43	DTPV	Date and time pattern value
OBJ_DATETIME_VALUE	44	DTV	Date and time value
OBJ_INTEGER_VALUE	45	IV	Integer value
OBJ_LARGE_ANALOG_VALUE	46	LAV	Float value
OBJ_OCTETSTRING_VALUE	47	OSV	Octet string
OBJ_POSITIVE_INTEGER_VALUE	48	PIV	Integer value positive
OBJ_TIME_PATTERN_VALUE	49	TPV	Time pattern value

Name - object type	Object type ID number	Objekt type identifier	Comment
OBJ_TIME_VALUE	50	TV	Time
_OBJ_MAX_SIZE	16#7FFFFFFF		

Tab. 13-2:

Example in a PLC application for addressing in connection with a client function block for the **objType** input : `BACnet.CmpBACnet.IEC_BACNET_OBJECT_TYPE.OBJ_ANALOG_INPUT`

In case of a BACnet server, the object, including its properties, is mapped as an element under the BACnet server node. Refer to chapter "Creating BACnet servers and objects and starting a BACnet server" on page 333.

☞ Customized objects or properties cannot be created.

Functional scope BACnet/IP client

Supported BACnet objects:

Function / function block	Comment
BACnetClientAcknowledgeAlarm	alarm / operator acknowledge
BACnetClientCreateObject	create object
BACnetClientDeleteObject	delete object
BACnetClientDeviceCommControl	activate / deactivate / control the communication of a device
BACnetClientGetAlarmSummary	list of actual alarms
BACnetClientGetEnrollmentSummary	list of objects which are configured for alarms
BACnetClientEventInfo	list of actual alarms and events
BACnetClientReadStreamFile	file read stream based
BACnetClientWriteStreamFile	file write stream based
BACnetClientReintializeDevice	restart / reinitialize a device
BACnetClientTimeSynchronization	time synchronization
BACnetClientUTCTimeSynchronization	UTC time synchronization
BACnetClientAddListElement	add object to a list
BACnetClientRemoveListElement	remove object from list
BACnetClientReadProperty	read object property or present value
BACnetClientReadPropertyMultiple	read multiple object property or present value
BACnetClientReadRange	read a range of an object e.g. trend log
BACnetClientWriteProperty	write object property or present value
BACnetClientWritePropertyMultiple	write multiple object property or present value
BACnetClientSubscribeCOV	register to COV of a `PresentValue` / `StatusFlags` of an object
BACnetClientSubscribeCOVProperty	register to COV of a Property of an object

Tab. 13-3:

☞ For details on the supported client functions and code examples, go to the programming examples in the BACnet library.

BACnet standard

The BACnet/IP integration implements the ANSI/ASHRAE standard 135-2012 protocol revision 14 and takes the standards AMEV AS-A and AS-B into account.

The functional scope corresponds to the device profile B-ASC (in preparation: B-BC).

Target systems

The BACnet/IP implementation can be implemented with the following systems:

- MLC (from 15V02)

Control hardware

The following control hardware is supported via the Engineering port interface:

- XM2
- XM4

Technical data

Property	Value
BACnet addressing	• IP address of the engineering port of the control. • UDP port: 47808 *) (default). • Network number: 1 (default).
Max. number of BACnet objects	Memory and runtime limit the number of objects. A BACnet object requires approximately 12 KB RAM. The PLC program also occupies the available memory (incl. other field bus configurations).
Update rate of the BACnet stack	The BACnet stack is called by an internal, free-running system task (i.e. the bus cycle task is insignificant). The PLC interfaces to read/write the BACnet objects from the PLC can be called individually.
PLC online change	**>> Not possible! <<** Possible online change effects (e.g. moving pointers) are not supported in the PLC interface to the BACnet stack. Thus, do not execute a PLC online change, as inconsistent data can be generated.

*) Each BACnet/IP device has to support the UDP port 47808 (= 0xBAC0).

⚠ CAUTION **PLC online change not possible!**

Download PLC program upon a code change.

13.2.2 BACnet basics

General information

BACnet is a standardized data communication protocol for building automation and building control networks.

BACnet was developed by the ASHRAE (American Society of Heating, Refrigerating and Air-Conditioning Engineers), and standardized in the ANSI/ASHRAE standard 135 in 1995.

The BACnet ISO standard 16484-5 is valid since 2003.

BACnet uses an object-oriented approach, i.e. each element is an object. Object properties describe the objects. Vendor-specific objects, properties or services can be defined.

BACnet is based on the client/server model. The server provides data. The client is the user, accessing data. Depending on the service, the communication is either specific (point-to-point) or to all devices (broadcast).

Further information

BACnet PICS ANNEX A – PROTOCOL IMPLEMENTATION CONFORMANCE STATEMENT(PICS)	R911401885 Reference book

Terms and abbreviations

ASHRAE	American Society of Heating, Refrigerating and Air-Conditioning Engineers
B/IP	Abbreviation for BACnet/IP
BACnet	Building Automation and Control networks (BACnet® is an ASHRAE trademark)
BBMD	BACnet/IP Broadcast Management Device If there are two or multiple IP subnets, there has to be one BBMD device in every subnet. The BBMD device is responsible to route the BACnet broadcasts to the other subnets. A Broadcast Distribution Table (BDT) is used for the configuration in the BBMD. There is an asymmetric BBMD if the BBMD in a subnet knows only the BBMD of a master control station, but the BBMD in the master control station knows all BBMDs in all subnets.
BIBB	BACnet Interoperability Building Block BIBBs are a collection of one or multiple BACnet functionalities and services. Two devices interoperate if one device supports the A-side of the BIBB (client) and the other device supports the B-side of the BIBB (server).
BTL	BACnet Testing Laboratories Test laboratory for BACnet conformity tests

Client / Server	• The **Server** provides data. Each BACnet device is a server, as at least the "Device" object has to be supported. A device that supports only the “Device” object is called client. • The **Client** is the user accessing data. Software tools (clients) are also called BACnet browsers. Client tools also have a “Device” object and a respective "Device ID" in the BACnet network.
COV	Change Of Value
COV Notification	Automatic notification upon change in value
Device ID	The device ID is the instance number of the "Device" object. All BACnet devices in the BACnet network require a unique device ID. The configuration is manufacturer-specific, e.g. with a DIP switch at the device, WebServer or proprietary tools. A remote specification via a client is an optional function and is rarely used.
EDE-Datei	Engineering Data Exchange Actual device configuration (configured objects) specified in an Excel table (EDE format), e.g. to exchange project configurations (export/import)
FD	Foreign Device A Foreign Device is a BACnet device (generally a client) that is not part of a BACnet IP network, but that can register at a BBMD device in the BACnet IP network to become a device of this BACnet IP network (e.g. for remote access).
Device profile	A device has to support a certain amount of BIBBs to comply with the specifications of a BACnet device profile. Example of a profile: B-BC (BACnet Building Controller)
GLT	Building control system
PICS	Protocol Implementation Conformance Statement The Protocol Implementation Conformance Statement is the self-declaration of the manufacturer describing the implemented BACnet functions of a device (e.g. the supported BIBBs, object types, properties)
API	**Application Programming Interface**: Interface to program applications
PDU	**Protocol Data Unit**: Complete data set provided as unit between peer entities of a network and which can include control information, address information or data.
APDU	**Application Protocol Data Unit**: Combined command/data block of the communication protocol

Tab. 13-4: BACnet terms and abbreviations

Networks and topologies

The BACnet standard is based on the ISO/OSI layer model for communication networks. BACnet data can be exchanged with different access methods using multiple media.

The following alternatives are defined for the layers 1 and 2:

- BACnet/IP (UDP/IP)
- BACnet MS/TP (master-slave/token-passing, RS-485)
- BACnet PTP (point-to-point, RS-232)
- BACnet LonTalk
- BACnet ARCNET
- BACnet Ethernet

For a network example, refer to the following figure:

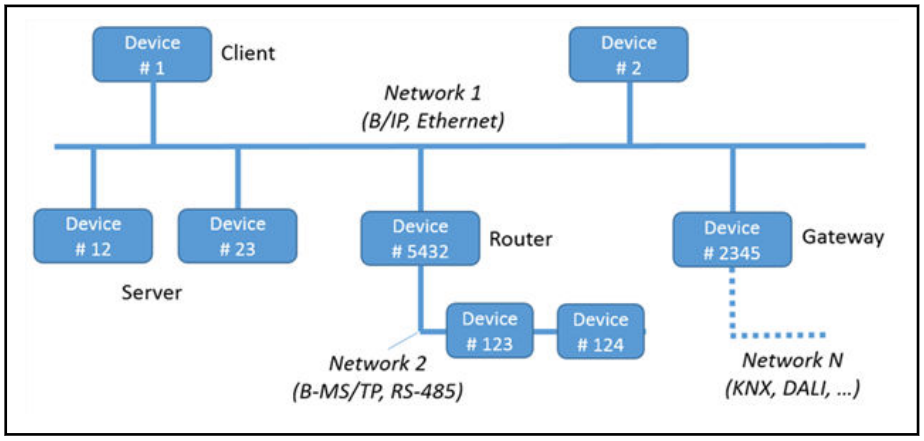


Fig. 13-2: BACnet network example

General notes on BACnet networks:

- A BACnet router connects BACnet subnets consisting of different media.
- A BACnet client can also access BACnet devices in the BACnet subnets (as router, e.g. B/IP ⇔ B-MS/TP, also routing broadcasts).
- When interconnecting multiple BACnet subnetworks with different media, each subnetwork requires a unique BACnet network number (possible with up to 65,535 subnetworks).
- A BACnet gateway connects a BACnet network to other network technologies or devices to other protocols. Normally, a BACnet gateway is a BACnet server with objects mapping device data of the seconds network interface.
- Each BACnet device requires a unique device ID (= instance number of the “Device” object of the device). The configuration is manufacturer-specific, e.g. with a DIP switch at the device, WebServer or proprietary tools.

Notes on BACnet/IP networks:

- BACnet/IP networks consist of one or multiple IP subnets provided with the same BACnet network number.

- A "MAC" address is used for addressing. This address does not correspond to the Ethernet MAC address, but consists of the IP address of the device and the port used.

  Example: B/IP address = 192.168.1.1:47808.

  The port 47808 (= 0xBAC0) is set by default and has to be supported by all B/IP devices.
- Also refer to the notes on "BACnet/IP Broadcast Management Device" (BBMD) and "Foreign Device" (FD) in Terms and abbreviations, page 307.

Objects and properties

The BACnet specification defines more than 50 standard objects. Manufacturer-specific objects and properties can also be defined, but not all devices support these definitions.

Object identification

Each BACnet object is provided with three identification properties:

- ObjectIdentifier

  The "ObjectIdentifier" (32 bits) consists of the "ObjectType" (10 bits) and the object instance number (22 bits):

  – There can be up to 1.024 (= 2^{10}) object types.

  – There can be up to 4.194.304 (= 2^{22} - 1) instances per object type.

  – The "ObjectIdentifier" is for example represented as: "AI 1"

  (i.e. "AI" is the abbreviation for the object type “AnalogInput” and "1" is the object instance number).

  – The "ObjectIdentifier" has to be unique in a device.
- ObjectType

  (as separate property, even though the "ObjectType" is also included in the "ObjectIdentifier")
- ObjectName

  The "ObjectName" has to be unique in the device and it is used to determine the "ObjectIdentifier" (Who-Has, I-Have).

Device object

The “Device” object is a specific BACnet object. One BACnet device may only have one “Device” object.

The instance number of the “Device” object is also called “Device ID” and has to be unique in the BACnet network.

Properties

The properties of BACnet objects are different ("Properties").

The type of the BACnet object specifies its properties.

The response of a property never changes (irrespective of the object it is used in).

Property		AI	AO	AV	BI	BO	BV	MI	MO	MV
Identification	ObjectIdentifier	R	R	R	R	R	R	R	R	R
	ObjectName	R	R	R	R	R	R	R	R	R
	ObjectType	R	R	R	R	R	R	R	R	R
	Description	O	O	O	O	O	O	O	O	O
	DeviceType	O	O	-	O	O	-	O	O	-

Property		AI	AO	AV	BI	BO	BV	MI	MO	MV
Status	PresentValue	R	W	R	R	W	R	R	W	R
	StatusFlags	R	R	R	R	R	R	R	R	R
	EventState	R	R	R	R	R	R	R	R	R
	Reliability	O	O	O	O	O	O	O	O	O
	OutOfService	R	R	R	R	R	R	R	R	R
Analog Properties	UpdateInterval	O	-	-	-	-	-	-	-	-
	Units	R	R	R	-	-	-	-	-	-
	MinPresValue	O	O	-	-	-	-	-	-	-
	MaxPresValue	O	O	-	-	-	-	-	-	-
	Resolution	O	O	-	-	-	-	-	-	-
	COVIncrement	O	O	O	-	-	-	-	-	-
Binary Properties	Polarity	-	-	-	R	R	-	-	-	-
	InactiveText	-	-	-	O	O	O	-	-	-
	ActiveText	-	-	-	O	O	O	-	-	-
	ChangeOfStateTime	-	-	-	O	O	O	-	-	-
	ChangeOfStateCount	-	-	-	O	O	O	-	-	-
	TimeOfStateCountReset	-	-	-	O	O	O	-	-	-
	ElapsedActiveTime	-	-	-	O	O	O	-	-	-
	TimeOfActiveTimeReset	-	-	-	O	O	O	-	-	-
	MinimumOffTime	-	-	-	-	O	O	-	-	-
	MinimumOnTime	-	-	-	-	O	O	-	-	-
Multistate Properties	NumberOfStates	-	-	-	-	-	-	R	R	R
	StateText	-	-	-	-	-	-	O	O	O
Commandable	PriorityArray	-	R	O	-	R	O	-	R	O
	RelinquishDefault	-	R	O	-	R	O	-	R	O

Property		AI	AO	AV	BI	BO	BV	MI	MO	MV
Alarm Properties	TimeDelay	O	O	O	O	O	O	O	O	O
	Notification-Class	O	O	O	O	O	O	O	O	O
	HighLimit	O	O	O	-	-	-	-	-	-
	LowLimit	O	O	O	-	-	-	-	-	-
	Deadband	O	O	O	-	-	-	-	-	-
	LimitEnable	O	O	O	-	-	-	-	-	-
	AlarmValue	-	-	-	O	-	O	-	-	-
	FeedbackVal-ue	-	-	-	-	O	-	-	O	-
	AlarmValues	-	-	-	-	-	-	O	-	O
	FaultValues	-	-	-	-	-	-	O	-	O
	EventEnable	O	O	O	O	O	O	O	O	O
	AckedTransi-tions	O	O	O	O	O	O	O	O	O
	NotifyType	O	O	O	O	O	O	O	O	O
	EventTimeS-tamps	O	O	O	O	O	O	O	O	O
	EventMessa-geTexts	O	O	O	O	O	O	O	O	O

Tab. 13-5: Overview on the properties of the important objects

R = Required (has to be available and readable).

W = Writeable (has to be available, readable and writable).

O = Optional (can be available).

Property	Data type	Description
AckedTransitions	`STRING`	Acknowledged changes in state, “1” = Event not yet occurred • `TO_OFFNORMAL` • `TO_FAULT` • `TO_NORMAL`
EventEnable	`STRING`	Event messages active, “1” = Enabled • `TO_OFFNORMAL` • `TO_FAULT` • `TO_NORMAL`
EventMessage-Texts	`ARRAY [0..2] OF WSTRING(255)`	Event message texts • `TO_OFFNORMAL` • `TO_FAULT` • `TO_NORMAL`

Property	Data type	Description
LimitEnable	STRING	Limit values active, “1” = Active • LowLimit • HighLimit
StatusFlags	STRING	State specifications, “1” = Active • IN_ALARM • FAULT • OVERRIDDEN • OUT_OF_SERVICE
TimeDelay	IEC_BACNET_UNSIGNED	Message delay (in seconds)

Tab. 13-6: Description of selected properties (with the data type in the interface to the PLC)

EventState Many objects support the EventState property. It describes the general object state, e.g. whether there is an alarm ("OffNormal") or whether the reliability of the object value displays a fault ("Fault").

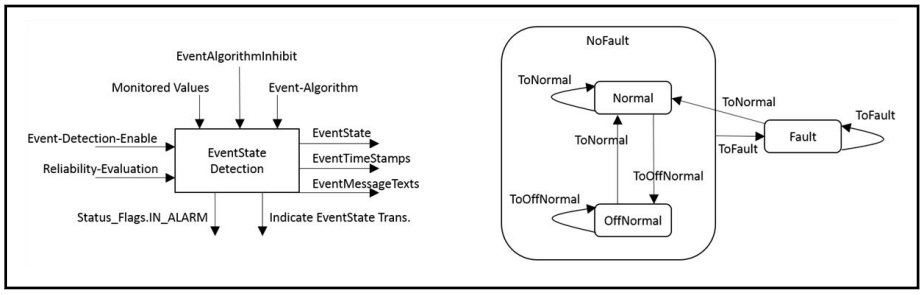


Fig. 13-3: EventState dependencies (left) and state machine (right)

Services

Overview

The BACnet standard defines more than 38 services describing the data exchange between the BACnet devices.

There are five categories of services:

Object access	• Read/write properties • Create/delete objects • List processing
File transfer	• Read/write files
Alarm and event functions	• Subscription, automatic notification upon changes in value (COV) • Alarm and event messages • Security systems (Life-Safety)

Remote Device Management	• Clock synchronization • Device restart • Dynamic Device Binding (DDB): Determining devices via broadcast (Who-Is, I-Am) • Dynamic Object Binding (DOB): Determining objects dynamically (Who-Has, I-Have)
Virtual Terminal	• Opening/closing/data transferring to a virtual interface of the device

Tab. 13-7: BACnet communication services (examples)

Updating (COV Notification)

The "Change Of Value"(COV) method is an event-controlled notification, e.g. the server notifies the client on value changes.

It is distinguished between:

- COV: Change in value of the "Present_Value" (current object value)
- COVproperty: Change in value of another property

Procedure:

- Clients can subscribe to “COV” messages (SubscribeCOV / UnsubscribeCOV).
- The notifications can include a "Time-To-Live". If "Time-To-Live" is set to indefinite, actively undo the subscription (Resubscribe).
- The server notifies the clients on changes in values (COVnotification).

BACnet COV notification example

The example shows a simple program to evaluate a COV notification. An IndraControl XM control with a BACnet object "Analog value" is used as BACnet server as well as another IndraControl XM control which is used as BACnet client to subscribe to and to get the COV notification. The individual program steps in the code example below can be extended or modified, if required.

The process to initialize the BACnet clients is identical to initialize the basic BACnet server.

The `COV` program of the BACnet client contains `CASE` structures, responsible for the initialization and processing of the COV notification. In the `COV` program, the `FB_SendCOVnotification` function block is processed cyclically outside the CASE statement.

Possible troubleshooting is contained in the commend as code and has to be implemented individually.

The actual troubleshooting takes place in the following order in the CASE step "run":

1. The COV notification is subscribed to
2. Subsequently, the callback is executed.

   In the example explained in the following, the callback is executed for a certain time (60 seconds).
3. After the COV notification has sent a value or after the specified time has lapsed, cancel the subscription and stop the callback.

During the subscription process, the subscription period is specified. By specifying the value "0" at the `lifetime` function block input, it is possible to configure the subscription time to an infinite runtime. If an infinite runtime is prac-

tical, depends on the application. For example, a check is carried out cyclically if the BACnet object still exists. In the example explained in the following, the subscription lapses after 3600 seconds. The subscription has to be renewed by the latest after this time has lapsed.

In the "`finish`" CASE step, select whether to execute the process again or whether to exit the process. The repetition of the process is controlled using the "`bAgain`" variable. Important: the program parts always have to run through outside the CASE loops. Outside the CASE loops, the incoming messages are received and acknowledged. Processing can be monitored using the "`FB_SendCOVnotification.iState`" variable.

Prerequisite:

In case of a BACnet server with a BACnet object `BACnet_Analog_Value`*, returning a value used for the BACnet client for subscribing to the COV notification.*

1. Create a new project, containing a BACnet server which is used as BACnet client.

   Refer to chapter 13.2.4 "Configuring the BACnet/IP client" on page 343.

2. Create the enumeration `TestState`, as follows:

Program:

```
TYPE TestState :
(
    once := 0,    // is passed through only once
    init,           // initialization
    start,          // start settings
    run,            // Test run
    finish,         // Conclusion settings
    end,            // End
    error,          // by error
    stop            // stopped
);
END_TYPE
```

3. For the COV notification, create the `FB_CB_CONF_COV_NOTIFICATION` function block:

Program:

```
FUNCTION_BLOCK FB_CB_CONF_COV_NOTIFICATION IMPLEMENTS BACnet.IBACnetEventConsumer
VAR_INPUT
  ResetOutputs        : BOOL := FALSE;
END_VAR
VAR_OUTPUT
  EvtCnt              : UINT := 0; // call count
  LastEvtValid        : BOOL := FALSE;
  LastCovNotifyInfo   : BACnet.CmpBACnet.IEC_BACNET_COV_NOTIF_INFO;
  LastPropertyValue   : BACnet.CmpBACnet.IEC_BACNET_PROPERTY_VALUE;
  LastRealValue       : REAL := 0;
END_VAR
VAR
  ResetOutputs_old    : BOOL := FALSE;
  prEvtConfCovNotify  : POINTER TO BACnet.CmpBACnet.EVT_BACNET_CONFCOVNOTIFICATION;
END_VAR
```

Program:

```
(* Reset Event-Count*)
 IF(TRUE = ResetOutputs  AND  FALSE = ResetOutputs_old) THEN
   EvtCnt := 0;
   LastEvtValid := FALSE;
 ELSE
```

```
  ResetOutputs_old := ResetOutputs;
END_IF
```

4. Add the callback method `BACnetEventCallback` to the `FB_CB_CONF_COV_NOTIFICATION` function block:

Program:

```
(* This method gets called, whenever a event occurs for which the implementing
  FB is registered via |BACnetServer.RegisterHook| or
 |BACnetServer.RegisterCallback|. Since several FBs implementing IBACnetEventConsumer
 can be registered for one event or callback the
 following rules are used for the returned BOOL:
 If the method returns FALSE, all other registered IBACnetEventConsumer objects for
 the specified Hook/Callback are further called.
 If the method return TRUE, the other registeres IBACnetEventConsumer objects are not
 called anymore. The calling order is as
 the objects were registered. All calls have to treat on their own, whether the
 EventIds are correct (see CmpBACnet.EVENTIDS) and/or
 BACnet-Object-IDs or other additional data fits the needs of the registered FB
 (i.e. the specific BACnet-Object-FB).*)

METHOD BACnetEventCallback : BOOL
VAR_INPUT
    // If only one event is possibly called, the following Ids can be ignored. If there are more
 // at least the EventId should be checked, to get the right type of pParameter.

 // ID of the occured Event. Contains the class and the event (see CmpBACnet.EVENTIDS)
    EventId    : DWORD;

    // Id of the parameter structure (see pParameter and CmpBACnet.EVENTIDS)
    usParamId    : WORD;

    // Version of the parameter structure (see pParameter and CmpBACnet.EVENTIDS)
    usVersion    : WORD;

    // Pointer to the event specific parameter, that is specified by Id
 // (see CmpBACnet.EVENTIDS and CmpBACnet, i.e. EVT_BACNET_ACKALARM)
    pParameter    : POINTER TO BYTE;
END_VAR
```

Program:

```
// METHOD BACnetEventCallback
// NOTE: no breakpoint possible, because this event callback is called from the
// BACstack task, which is not an IEC task!

  (* Check Event *)
  IF( EventId   = BACnet.CmpBACnet.EVT_ConfCOVNotification AND
      usParamId = BACnet.CmpBACnet.EVTPARAMID_CmpBACnet_ConfCOVNotification
    ) THEN

    LastEvtValid := TRUE;
    EvtCnt := EvtCnt + 1;

    (* set event parameter *)
    prEvtConfCovNotify   := pParameter;
    LastCovNotifyInfo    := prEvtConfCovNotify^.pServiceInfo^;
    LastPropertyValue    := LastCovNotifyInfo.listOfValues^;

// Note: the buffer in 'LastPropertyContents' is stack internal,
// that means you have to copy the value here in the callback!

    (* analog object COV notificatin *)
    IF( LastCovNotifyInfo.monitoredObjectID.typ = BACnet.CmpBACnet.OBJ_ANALOG_INPUT OR
        LastCovNotifyInfo.monitoredObjectID.typ = BACnet.CmpBACnet.OBJ_ANALOG_OUTPUT OR
        LastCovNotifyInfo.monitoredObjectID.typ = BACnet.CmpBACnet.OBJ_ANALOG_VALUE
      ) THEN

      LastRealValue := BACnet.GetRealFromContents(LastPropertyValue.value);

      (* Event found Meldung *)

    ELSE
      LastRealValue := 0.0;
    END_IF
```

```
    (* event handled *)
    BACnetEventCallback := TRUE; // TRUE = no calls of other registered callbacks of this event

  ELSE // unexpected EventId
    (* not my event *)
    BACnetEventCallback := FALSE; // FALSE = call the other registered callbacks of this event
  END_IF

(* Note:  prEvtConfCovNotify^.pCBStatus^ := BACnet.CmpBACnet.IEC_BACNET_CB_STATUS.CB_STATUS_DEFAULT
    => default: the stack handles the default actions (and not the hook / callback)

   Note:  prEvtConfCovNotify^.pCBStatus^ := BACnet.CmpBACnet.IEC_BACNET_CB_STATUS.CB_STATUS_OK
    => the hook or application generates the reply, see the ((Response-)-'CbCompletion' functions,
       which are mentioned in the corresponding event structures,
       e.g. see documentation of EVT_BACNET_CONFCOVNOTIFICATION.
*)
```

5. Create COV program COV:

Program:

```
PROGRAM COV
VAR

    (* Test_00 - Example COV Notification *)
    Test_00 : TestState := once;


  (* COV Notification *)
  StartCOV : BOOL := FALSE;
  FB_SendCOVnotification : BACnet.BACnetClientSubscribeCOV;

  COVTargetDeviceNumber : DWORD := 0; // DeviceId of the device (BACnet-Server) at the BACnet network
  COVObjectType : DINT := 0;  // AI = 0, AO = 1, AV = 2, BI = 3, BO = 4, BV = 5
  COVObjectInstance : DWORD := 0;
  COVSubscribe : BOOL := TRUE;  // Subscribe or Un-Subscribe
  CovValue : REAL := 0.0; // for analog values, with GetRealFromContent()
  COVState : UINT := 0; // at the busy state

  (* COV Callback *)
  StartCallback : BOOL := FALSE;
  FB_CB_Conf_COV_Notification : FB_CB_CONF_COV_NOTIFICATION; // Instance of the Calback-FBs
  BACnetStatus : BACnet.CmpBACnet.IEC_BACNET_STATUS;
  EventCount : UINT := 0;

  (* Timeout Timer *)
  Timeout : TON;
  tTimeout: TIME := T#1M;
    waitTon: Ton;
    eventTon:    Ton;
    Run_case: INT;
    bAgain: BOOL := TRUE;            //if bAgain = false then reading of the value is stopp
    wfinish: WORD;
END_VAR
```

Program:

```
IF TRUE = UserVarGlobal.BACnetReady THEN

    CASE Test_00 OF

        once: // run through only once at the beginning
        //--------------------------------------
            (* Registering the FB with your own BACnet server *)
            FB_SendCOVnotification.RegisterToServer(BACnet_Server);

            (* Registering the Callbacks with your own BACnet-Server *)
      BACnetStatus := BACnet_Server.RegisterHook(
                    BACnet.CmpBACnet.IEC_BACNET_CB_TYPE.CB_CONF_COV_NOTIFICATION,
                    FB_CB_Conf_COV_Notification );
            IF BACnetStatus = BACnet.CmpBACnet.IEC_BACNET_STATUS.BACNET_STATUS_OK THEN
                (* success message *)
                (* ... *)
                (* set Event-Count *)
                EventCount := FB_CB_Conf_COV_Notification.EvtCnt;
            ELSE
```

```
        (* error message *)
        (* ... *)
    END_IF


    Test_00 := init;

init: // init
//---------------------------------------
    FB_SendCOVnotification(xExecute:=FALSE);
    COVState := 0; // if busy

    COVTargetDeviceNumber := 21; // TargetDeviceNumber of the BACnet-Server of the BACnet Partner
    COVObjectInstance := 1; // Instance
    COVObjectType := 2; // 0 = AI, 1 = AO, 2 = AV, 3 = BI, 4 = BO, 5 = BV

    Run_case := 0;
    Test_00 := start;

start: // start COV
//---------------------------------------

    Test_00 := run;


run: // run
//---------------------------------------

    StartCallback := TRUE;
    waitTon(IN:= NOT (WaitTon.q), PT:= T#1S, Q=> , ET=> );

    //Test COV
    CASE Run_case OF
        0:

            IF waitTon.Q = TRUE THEN
                Run_case := Run_case + 1;
            END_IF

        1:                                      //send COV subcribe
            StartCOV := TRUE;  // FB start
            COVSubscribe := TRUE; // Subscribe
            Run_case := Run_case + 1;

        2:                                      // wait until FB done
       // see below section


        3:                                      // wait of events
            (* Start evaluation *)
            StartCallback := TRUE;
            eventTon(IN:= NOT (eventTon.Q), PT:= T#60s, Q=> , ET=> );
            IF eventTon.Q = TRUE THEN
                Run_case := Run_case + 1;
            END_IF

        4:                                      // send COV unsubcribe
            StartCOV := TRUE;                    // FB start
            COVSubscribe := FALSE;               // unsubscribe
            Run_case := Run_case + 1;

        5:                                      // wait until FB done
       // see below section


        6:                                      // Test end
            (* Stopp evaluation *)
            StartCallback := FALSE;
            IF waitTon.Q = TRUE THEN
                Test_00 := finish;
                Run_case := 0;
            END_IF

    END_CASE

finish: // finish
//---------------------------------------
wfinish := wfinish + 1;

(* Stopp evaluation *)
StartCallback := FALSE;
```

```
        IF bAgain THEN
            Test_00 := init; // Neustart
        ELSE
            Test_00 := end;
        END_IF

        end: // exit
        //---------------------------------------
            FB_SendCOVnotification(xExecute:=FALSE);    //Exit FB cleanly

            (* Un-registering the FB at the own BACnet server *)
            FB_SendCOVnotification.UnregisterFromServer();

            (* remove the callback *)
            BACnetStatus := BACnet_Server.UnregisterHook(
                     BACnet.CmpBACnet.IEC_BACNET_CB_TYPE.CB_CONF_COV_NOTIFICATION,
                     FB_CB_Conf_COV_Notification );

            IF BACnetStatus = BACnet.CmpBACnet.IEC_BACNET_STATUS.BACNET_STATUS_OK THEN
                (* success message *)
            ELSE
                (* error message *)
            END_IF

            Test_00 := stop;


        error: // error
        //---------------------------------------

            Test_00 := finish;
            bAgain := FALSE;

        stop: // exit
        //---------------------------------------

    END_CASE
    (* ################ ENDE USER ################ *)

    (* ********************************************************************** *)
      IF StartCOV THEN

        (* call Function Block *)
        FB_SendCOVnotification(
          xExecute  := TRUE,  // BOOL : Rising edge: starts the FB execution
          xAbort    := FALSE,  // BOOL : TRUE: Abort of the FB-Operation
          dwTargetDeviceNumber := COVTargetDeviceNumber,
    // DWORD : DeviceId of the addressed device in the BACnet network

    //pTargetDeviceAddress := 0,
    // [Optional:] POINTER TO IEC_BACNET_ADDRESS : Device address of the
    // addressed device in the BACnet network.

    // If an address is set here, the dwTargetDeviceNumber is ignored.
          xSubscribe := COVSubscribe,  // BOOL : TRUE: registers or renews the COV subscription.
                                       //       FALSE: delete the COV Subscription.
          xIssueConfirmedNotifications := TRUE,  //  (TRUE) "confirmed" *
                                                                          //   (FALSE) "unconfirmed" COV n
          lifetime := 3600, // IEC_BACNET_UNSIGNED: Lifetime of COV notifications in seconds
                      //                     (0 = infinite)

 // Object type to be observed
          //monitoredObjType := BACnet.CmpBACnet.IEC_BACNET_OBJECT_TYPE.OBJ_ANALOG_INPUT,
          //monitoredObjType := BACnet.CmpBACnet.IEC_BACNET_OBJECT_TYPE.OBJ_ANALOG_OUTPUT,
          //monitoredObjType := BACnet.CmpBACnet.IEC_BACNET_OBJECT_TYPE.OBJ_BINARY_INPUT,
          //monitoredObjType := BACnet.CmpBACnet.IEC_BACNET_OBJECT_TYPE.OBJ_BINARY_OUTPUT,

          monitoredObjType := COVObjectType, // or ... 0 = AI, 1 = AO, 2 = AV, 3 = BI, 4 = BO, 5 = BV
          monitoredObjInst := COVObjectInstance
    // IEC_BACNET_DWORD :  Object instance of the object to be observed

(* at SendCOV Property *)
          //monitoredPropID := 85,  // IEC_BACNET_PROPERTY_ID : PROP_PRESENT_VALUE = 85
          //monitoredPropIndex := 0,  // IEC_BACNET_SIGNED : Property-Index, which is to be observed
          //covIncrement := 0.5  // IEC_BACNET_REAL : COV increment for the COV notifications
        );

        (* Evaluation done *)
        IF (TRUE = FB_SendCOVnotification.xDone) THEN
          (* End execution of the FB *)
          FB_SendCOVnotification(xExecute := FALSE);
          (* message "success" *)
```

```
      (* finish *)
      // ? COVState := 0;  // reset busy State ?

      StartCOV := FALSE;  // finish  ?
      Run_case := Run_case + 1;
    END_IF

    (* Evaluation busy *)
    IF (TRUE = FB_SendCOVnotification.xBusy) THEN
      CASE FB_SendCOVnotification.iState OF
        0: // no evaluation, Reset-state
          COVState := COVState + 1;
        1: // Searching for the addressed device
          COVState := COVState + 10;
        2: // Sending the BACnet request
          COVState := COVState + 100;
        3: // Waiting for response to the request
          COVState := COVState + 1000;
        4: // Processing the response and setting the outputs
          COVState := COVState + 10000;
      END_CASE
    END_IF

    (* evaluation error *)
    IF (TRUE = FB_SendCOVnotification.xError) THEN
      (* End execution of the FB *)
      FB_SendCOVnotification(xExecute:= FALSE);
      (* error message *)
      (* Error evaluation *)
      // ? COVState := 0;  // busy State reset ?
      StartCOV := FALSE;  // finish execution ?
      Test_00 := error;
    END_IF

  END_IF // StartCOV


(* ********************************************************************************************** *)
  (* for the Callback *)
  (* real evaluation ... *)
  IF StartCallback THEN
    // either the value changes ... CovValue = old value
    // or the EventCount increases ... EventCount = old value

    (* get value from object *)
    CovValue := FB_CB_Conf_COV_Notification.LastRealValue; // PresentValue Analog Objects
    //CovValue := FB_CB_Conf_COV_Notification.LastBinaryValue; // PresentValue Binary Objects
    //CovValue := FB_CB_Conf_COV_Notification.LastPropertyValue.value; // Object-Property


    (* CovValue change *)
    IF FB_CB_Conf_COV_Notification.LastRealValue <> CovValue THEN

      (* save CovValue  *)
      CovValue := FB_CB_Conf_COV_Notification.LastRealValue;

      (* stopp timer *)
      Timeout(IN:=FALSE);

      (* next step *)
    END_IF

    (* Event Count increase *)
    IF FB_CB_Conf_COV_Notification.EvtCnt > EventCount THEN

      (* save EventCount *)
      EventCount := FB_CB_Conf_COV_Notification.EvtCnt;

      (* stopp Timer *)
      Timeout(IN:=FALSE);

      (* next step *)
    END_IF

    (* Verification Timeout *)
    Timeout(IN:=TRUE, PT:=tTimeout);
    IF (TRUE = Timeout.Q) THEN
      Timeout(IN:=FALSE);
      (* Timeout message *)
    END_IF

    (* Reset Callback *)
    //FB_CB_Conf_COV_Notification(ResetOutputs := TRUE);  // resets the values
```

```
        //FB_CB_Conf_COV_Notification(ResetOutputs := FALSE);  // resets the flag (can be done better)
      END_IF // StartCallback
END_IF // BACnetReady
```

6. Insert the `COV` program in the desired PLC program part. Additionally, the COV notification result can be edited.
   - `COV();`
   - `Temp := cov.CovValue;`

Job prioritization (outputs)

There is a problem with objects with output values (e.g. AO, BO, MO). Different clients can specify different values, e.g. a manual intervention switches "off" an output and the parent control switches "on" the same output.

To avoid conflicts, commanding BACnet objects have the prioritization property "PriorityArray".

With the prioritization property, a writing command is not directly executed on the "PresentValue", but the value is entered internally into a field of the priority list.

Properties of the priority list

- The priority list has 16 fields. Field 1 has the highest priority.
- To be applied, one client has to be assigned to one field. The value "`null`" deletes the write access of a client, i.e. a field has to be set explicitly to "`null`" to remove this priority.
- The client with the highest priority provides its value to "PresentValue".
- If the priority list is empty, i.e. no fields are assigned or they are assigned with "`null`", the "RelinquishDefault"property is applied as "PresentValue".
- This prioritization is always active and cannot be disabled.

In BACnet, the following priorities are reserved for special purposes:

Priority	Application
1	Safety controls, manual (Manual-Life Safety)
2	Safety controls, automatic (Automatic-Life Safety)
3	Free
4	Free
5	Parent system control (Critical Equipment Control)
6	On/off time limit (Minimum On/Off)
7	Free
8	Manual intervention/hand (Manual Operator)
9	Free
10	Free
11	Free
12	Free
13	Free

Priority	Application
14	Free
15	Free
16	Free

Tab. 13-8: Reserved priorities in BACnet

Alarm handling

General information

BACnet classifies two methods for alarm handling:

- **Intrinsic Reporting**
  - A BACnet object (e.g. AI, BI, ...) triggers an alarm.
  - The triggering object also determines the trigger.
  - Sending the alarm.
- **Algorithmic Reporting**
  - A BACnet object (e.g AI, BI, ...) triggers an alarm.
  - Another BACnet object (EventEnrollment) determines the trigger.
  - Sending the alarm.

Intrinsic Reporting

The conditions to generate the alarm result from the properties of the triggering object.

The triggering object (e.g. an AnalogInput (AI) in the following figure) has a reference to the reporting channel (NotificationClass (NC)) distributing the alarm message.

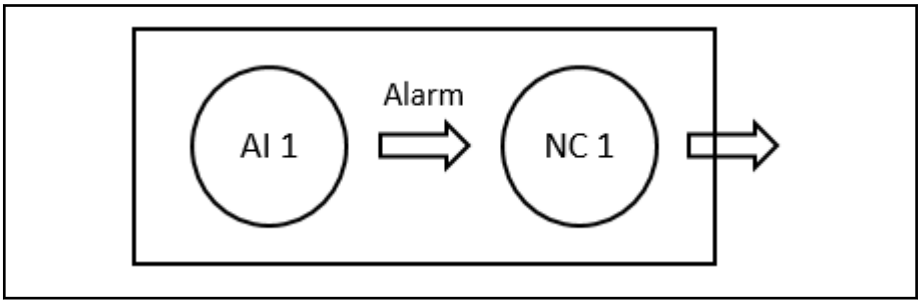


Fig. 13-4: Example of Intrinsic Reporting

The following algorithms are frequently used for Intrinsic Reporting:

Algorithm	Object
OUT_OF_RANGE	AnalogInput, AnalogOutput, AnalogValue, PulseConverter
CHANGE_OF_STATE	BinaryInput, BinaryValue, MultistateInput, MultistateValue
COMMAND_FAILURE	BinaryOutput, MultistateOutput
BUFFER_READY	EventLog, TrendLog, TrendLogMultiple
FLOATING_LIMIT	Loop

Tab. 13-9: Exemplary algorithms for intrinsic reporting

Algorithm OUT_OF_RANGE

The algorithm OUT_OF_RANGE is normally a limit value monitoring of an analog value (AI, AO, AV).

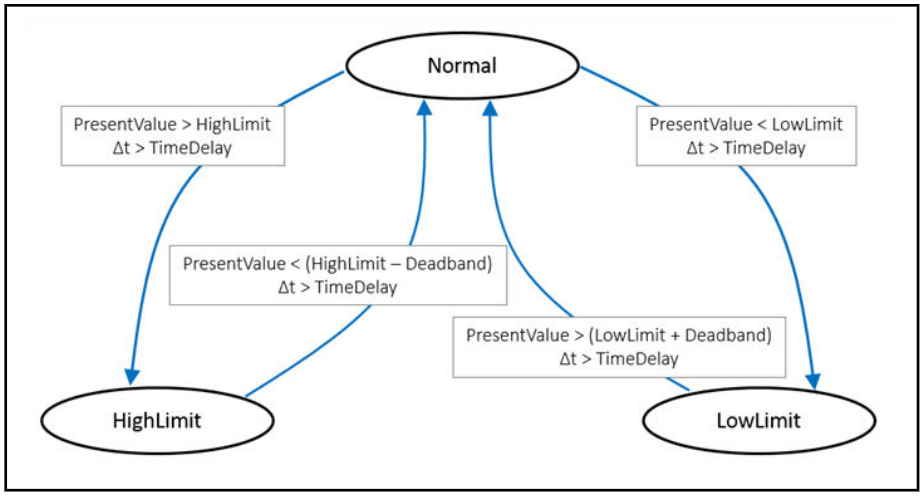


Fig. 13-5: State machine OUT_OF_RANGE

Algorithm CHANGE_OF_STATE

The algorithm CHANGE_OF_STATE is a state monitoring of a binary or multi-state input or value (BI, BV, MI, MV).

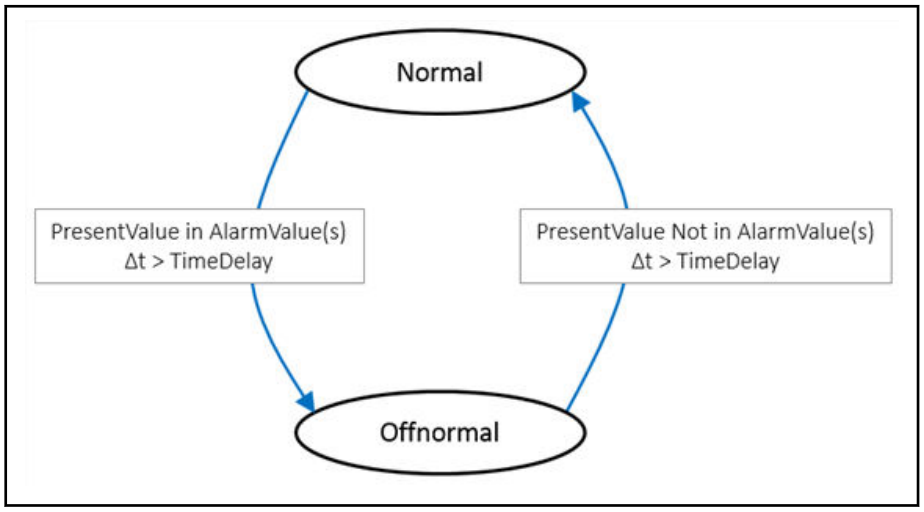


Fig. 13-6: State machine CHANGE_OF_STATE

Algorithm COMMAND_FAILURE

The algorithm COMMAND_FAILURE is a command execution check of a binary or multi-state output (BO, MO).

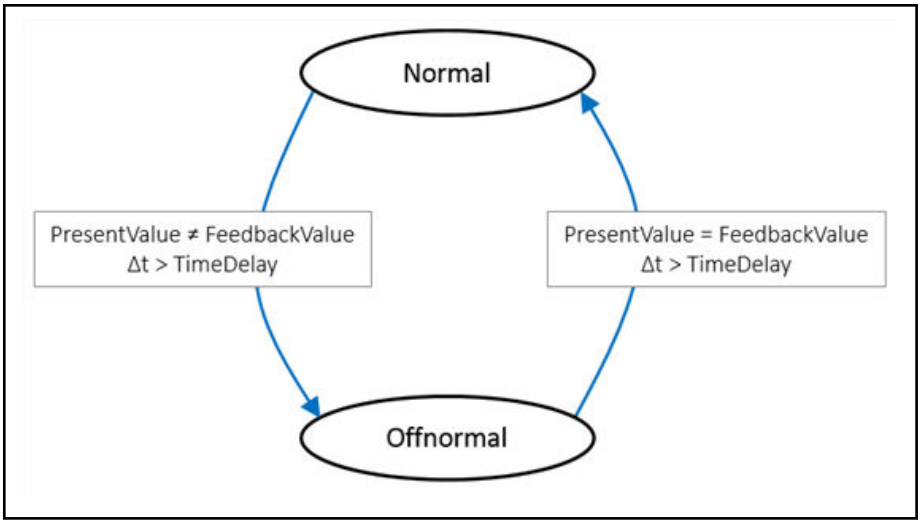


Fig. 13-7: State machine COMMAND_FAILURE

Algorithmic Reporting

The conditions to generate the alarm are not configured in the triggering object, but in another object (EventEnrollment (EE)).

The EventEnrollment object has a reference to the reporting channel (NotificationClass (NC))

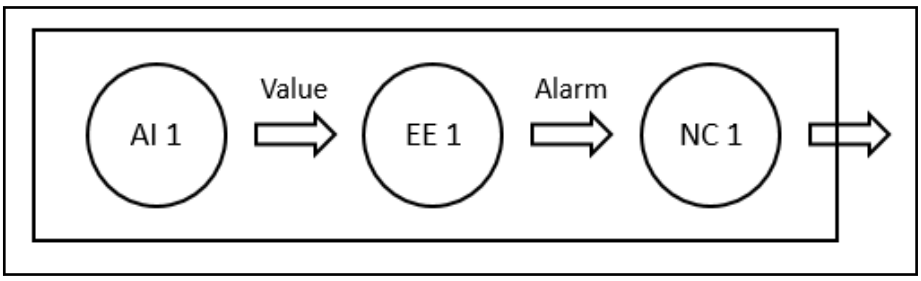


Fig. 13-8: Example of Algorithmic Reporting

Algorithmic Reporting can be used to implement complex alarm functions, as the algorithm can be freely selected (as it is not integrated in the triggering object, but implemented using an additional object).

Alarm distribution

The BACnet Objekt NotificationClass (NC) is used as reporting channel of the alarm generation method.

- The relevant transmission properties are set in the NC object.
- The NC object is provided with a list of (Recipients) informing on the alarms.
- At least one NC object has to be in the device (if to use alarms).
- The NC object does not depend on the alarm generation (only used as reporting channel).
- Multiple NC object are reasonable to distinguish between warnings and critical alarms.

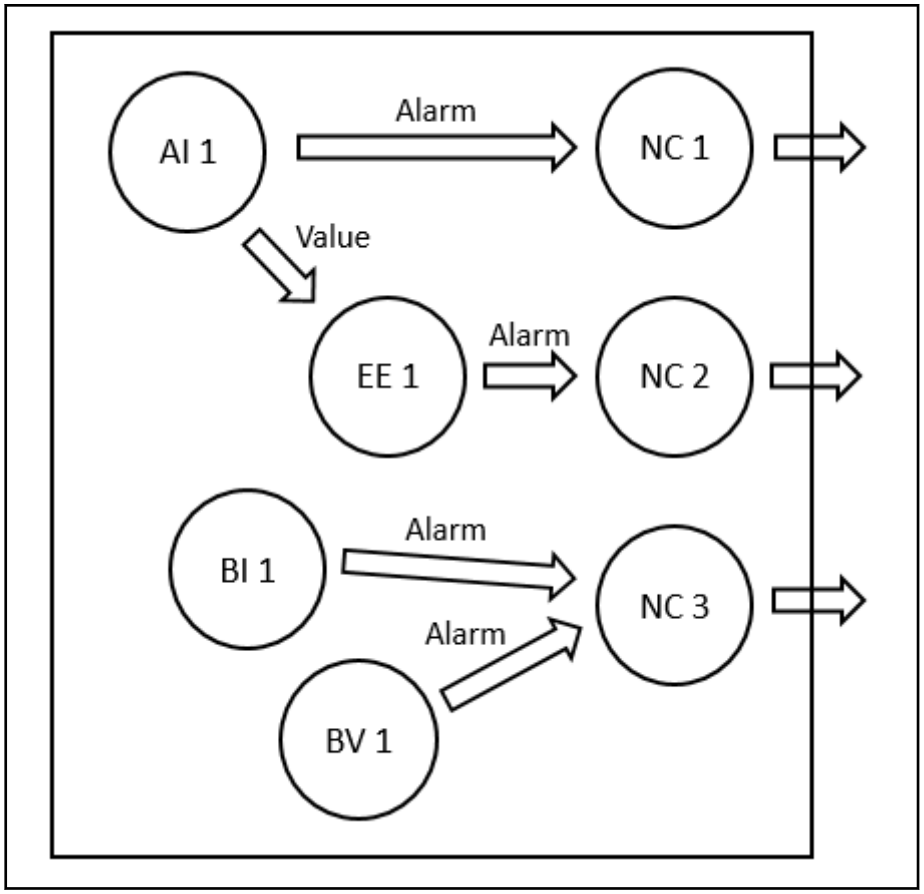


Fig. 13-9: Alarm distribution example

Data logging (TrendLog)

- The TrendLog (TL) object collects data pairs (value + time stamp).
- Data is logged either regularly or event-controlled, e.g. due to changes in value.
- The referenced object can be in the own devices ("internal trending") or in another device ("external trending").
- The properties of the TL object control the logging.
- The load of the LogBuffers is monitored (Intrinsic Reporting). The algorithm BUFFER_READY is used.

  There is a notification to obtain the values if `EventState == NORMAL` and `RecordsSinceNotification >= NotificationThreshold`.

  Logged data can be retrieved using the ReadRange service.

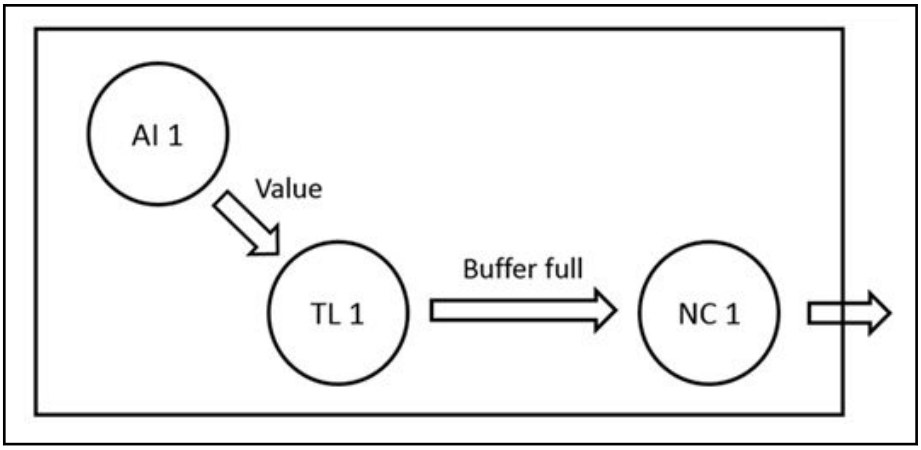


Fig. 13-10: Example of TrendLog

Device profiles

The BACnet specification describes standardized profiles for BACnet devices. Each profile describes a functional scope. A BACnet device has to support this functional scope to ensure that the device meets the device profile. The most important device profiles are:

B-AWS	BACnet-Advanced Workstation	Management and operating station with a performance scope higher than B-OWS
B-OWS	BACnet-Operator Workstation	Management and operating station
B-OD	BACnet-Operator Display	Operating device (on site)
B-BC	BACnet-Building Controller	Freely-programmable automation device
B-AAC	BACnet-Advanced Application Controller	Automation device with program selection
B-ASC	BACnet-Application Specific Controller	Automation device with specific program
B-SA	BACnet-Smart Actuator	Controlling or regulating unit
B-SS	BACnet-Smart Sensor	Sensor

Tab. 13-10: BACnet device profile

BACnet functionalities and services are described as BIBB (BACnet Interoperability Building Block). The following categories are distinguished:

- Data Sharing (DS)
- Alarm / Event Management (AE)
- Scheduling (SCH)
- Trending (TR)
- Device and Network Management (DN)

The functionalities are classified either with "A" (client) or "B" (server).

Examples of BIBB:

- DS-RP-A (/-B): Data Sharing - ReadProperty - Client (/-Server)

- DS-WP-A (/-B): Data Sharing - WriteProperty - Client (/-Server)

BACnet error codes

Error Code	Name
16#7FFFFFFF	_ERR_CODE_MAX_SIZE
0	ERR_CODE_OTHER
1	ERR_CODE_AUTHENTICATION_FAILED
2	ERR_CODE_CONFIG_IN_PROGRESS
3	ERR_CODE_DEVICE_BUSY
4	ERR_CODE_DYN_CREATION_NOT_SUPP
5	ERR_CODE_FILE_ACCESS_DENIED
6	ERR_CODE_WRONG_SECURITY_LEVELS
7	ERR_CODE_INCONSISTENT_PARAMS
8	ERR_CODE_INVALID_PARAMETERS
9	ERR_CODE_INVALID_DATA_TYPE
10	ERR_CODE_INVALID_FILE_ACCESS
11	ERR_CODE_INVALID_FILE_START_POS
12	ERR_CODE_INVALID_OPERATOR_NAME
13	ERR_CODE_INVALID_PARAMETER_TYPE
14	ERR_CODE_INVALID_TIME_STAMP
15	ERR_CODE_KEY_GENERATION_ERROR
16	ERR_CODE_MISSING_PARAMETER
17	ERR_CODE_NO_OBJS_OF_SPEC_TYPE
18	ERR_CODE_NO_SPACE_FOR_OBJECT
19	ERR_CODE_NO_SPACE_FOR_ELEMENT
20	ERR_CODE_NO_SPACE_TO_WRITE_PROP
21	ERR_CODE_NO_VT_SESSIONS_AVAIL
22	ERR_CODE_PROP_IS_NOT_A_LIST
23	ERR_CODE_OBJ_DELETION_DENIED
24	ERR_CODE_OBJ_ID_ALREADY_EXISTS
25	ERR_CODE_OPERATIONAL_PROBLEM
26	ERR_CODE_PASSWORD_FAILURE
27	ERR_CODE_READ_ACCESS_DENIED
28	ERR_CODE_SECURITY_NOT_SUPPORTED
29	ERR_CODE_SERVICE_REQUEST_DENIED
30	ERR_CODE_TIMEOUT
31	ERR_CODE_UNKNOWN_OBJECT
32	ERR_CODE_UNKNOWN_PROPERTY
34	ERR_CODE_UNKNOWN_VT_CLASS

Error Code	Name
35	ERR_CODE_UNKNOWN_VT_SESSION
36	ERR_CODE_UNSUPPORTED_OBJ_TYPE
37	ERR_CODE_VALUE_OUT_OF_RANGE
38	ERR_CODE_VT_SESSION_CLOSED
39	ERR_CODE_VT_SESSION_TERM_FAILED
40	ERR_CODE_WRITE_ACCESS_DENIED
41	ERR_CODE_CHAR_SET_NOT_SUPPORTED
42	ERR_CODE_INVALID_ARRAY_INDEX
43	ERR_CODE_COV_SUBS_FAILED
44	ERR_CODE_NOT_COV_PROPERTY
45	ERR_CODE_OPTIONAL_FUNC_NOT_SUPP
46	ERR_CODE_INVALID_CONFIG_DATA
47	ERR_CODE_DATATYPE_NOT_SUPPORTED
48	ERR_CODE_DUPLICATE_NAME
49	ERR_CODE_DUPLICATE_OBJECT_ID
50	ERR_CODE_PROP_IS_NOT_AN_ARRAY
51	ERR_CODE_ABORT_BUFFER_OVERFLOW
52	ERR_CODE_ABORT_INVALID_APDU
53	ERR_CODE_ABORT_PREEMPTED
54	ERR_CODE_ABORT_SEGMENT_NOT_SUPP
55	ERR_CODE_ABORT_PROPRIETARY
56	ERR_CODE_ABORT_OTHER
57	ERR_CODE_INVALID_TAG
58	ERR_CODE_NETWORK_DOWN
59	ERR_CODE_REJECT_BUFFER_OVERFLOW
60	ERR_CODE_REJECT_INCONSISTENT_PARAMS
61	ERR_CODE_REJECT_INVALID_DATA_TYPE
62	ERR_CODE_REJECT_INVALID_TAG
63	ERR_CODE_REJECT_MISSING_REQUIRED_PARAM
64	ERR_CODE_REJECT_PARAMETER_OUT_OF_RANGE
65	ERR_CODE_REJECT_TOO_MANY_ARGUMENTS
66	ERR_CODE_REJECT_UNDEFINED_ENUMERATION
67	ERR_CODE_REJECT_UNRECOGNIZED_SERVICE
68	ERR_CODE_REJECT_PROPRIETARY
69	ERR_CODE_REJECT_OTHER
70	ERR_CODE_UNKNOWN_SERVICE
71	ERR_CODE_UNKNOWN_ROUTER
72	ERR_CODE_VALUE_NOT_INITIALIZED

Error Code	Name
73	ERR_CODE_INVALID_EVENT_STATE
74	ERR_CODE_NO_ALARM_CONFIGURED
75	ERR_CODE_LOG_BUFFER_FULL
76	ERR_CODE_LOGGED_VALUE_PURGED
77	ERR_CODE_NO_PROPERTY_SPECIFIED
78	ERR_CODE_NOT_CONFIGURED_FOR_TRIGGERED_LOGGING
79	ERR_CODE_UNKNOWN_SUBSCRIPTION
80	ERR_CODE_PARAMETER_OUT_OF_RANGE
81	ERR_CODE_LIST_ELEMENT_NOT_FOUND
82	ERR_CODE_BUSY
83	ERR_CODE_COMMUNICATION_DISABLED
84	ERR_CODE_SUCCESS
85	ERR_CODE_ACCESS_DENIED
86	ERR_CODE_BAD_DESTINATION_ADDRESS
87	ERR_CODE_BAD_DESTINATION_DEVICE_ID
88	ERR_CODE_BAD_SIGNATURE
89	ERR_CODE_BAD_SOURCE_ADDRESS
90	ERR_CODE_BAD_TIMESTAMP
91	ERR_CODE_CANNOT_USE_KEY
92	ERR_CODE_CANNOT_VERIFY_MESSAGE_ID
93	ERR_CODE_CORRECT_KEY_REVISION
94	ERR_CODE_DESTINATION_DEVICE_ID_REQUIRED
95	ERR_CODE_DUPLICATE_MESSAGE
96	ERR_CODE_ENCRYPTION_NOT_CONFIGURED
97	ERR_CODE_ENCRYPTION_REQUIRED
98	ERR_CODE_INCORRECT_KEY
99	ERR_CODE_INVALID_KEY_DATA
100	ERR_CODE_KEY_UPDATE_IN_PROGRESS
101	ERR_CODE_MALFORMED_MESSAGE
102	ERR_CODE_NOT_KEY_SERVER
103	ERR_CODE_SECURITY_NOT_CONFIGURED
104	ERR_CODE_SOURCE_SECURITY_REQUIRED
105	ERR_CODE_TOO_MANY_KEYS
106	ERR_CODE_UNKNOWN_AUTHENTICATION_TYPE
107	ERR_CODE_UNKNOWN_KEY
108	ERR_CODE_UNKNOWN_KEY_REVISION
109	ERR_CODE_UNKNOWN_SOURCE_MESSAGE
110	ERR_CODE_NOT_ROUTER_TO_DNET

Error Code	Name
111	ERR_CODE_ROUTER_BUSY
112	ERR_CODE_UNKNOWN_NETWORK_MESSAGE
113	ERR_CODE_MESSAGE_TOO_LONG
114	ERR_CODE_SECURITY_ERROR
115	ERR_CODE_ADDRESSING_ERROR
116	ERR_CODE_WRITE_BDT_FAILED
117	ERR_CODE_READ_BDT_FAILED
118	ERR_CODE_REGISTER_FOREIGN_DEVICE_FAILED
119	ERR_CODE_READ_FDT_FAILED
120	ERR_CODE_DELETE_FDT_ENTRY_FAILED
121	ERR_CODE_DISTRIBUTE_BROADCAST_FAILED
122	ERR_CODE_UNKNOWN_FILE_SIZE
123	ERR_CODE_ABORT_APDU_TOO_LONG
124	ERR_CODE_ABORT_APPLICATION_EXCEEDED_REPLY_TIME
125	ERR_CODE_ABORT_OUT_OF_RESOURCES
126	ERR_CODE_ABORT_TSM_TIMEOUT
127	ERR_CODE_ABORT_WINDOW_SIZE_OUT_OF_RANGE
128	ERR_CODE_FILE_FULL
129	ERR_CODE_INCONSISTENT_CONFIGURATION
130	ERR_CODE_INCONSISTENT_OBJECT_TYPE
131	ERR_CODE_INTERNAL_ERROR
132	ERR_CODE_NOT_CONFIGURED
133	ERR_CODE_OUT_OF_MEMORY
134	ERR_CODE_VALUE_TOO_LONG
135	ERR_CODE_ABORT_INSUFFICIENT_SECURITY
136	ERR_CODE_ABORT_SECURITY_ERROR
65535	ERR_CODE_MAX

Tab. 13-11: TYPE IEC_BACNET_ERROR_CODE

BACnet status codes

Name	Initial	Comment
_BACNET_STATUS_MAX_SIZE	16#7FFFFFFF	
BACNET_STATUS_OK	16#0	Request was successful.
BACNET_STATUS_UNKNOWN_ERROR	16#1	Some undefined error has been detected.
BACNET_STATUS_SERVICE_NOT_FOUND	16#2	Application could not connect to the datalink layer.
BACNET_STATUS_NOT_INITIALIZED	16#3	Some request should be sent but the API is not initialized correctly.
BACNET_STATUS_NOT_SUPPORTED	16#4	Some request should be executed which is not supported.

Name	Initial	Comment
BACNET_STATUS_REJECT	16#5	Some request was make which was rejected.
BACNET_STATUS_DCC_DISABLED	16#6	Some request should be sent but the device communication control is disabled.
BACNET_STATUS_CHARSET_NOT_SUP-PORTED	16#7	Some character set given which is not supported.
BACNET_STATUS_FUNCTION_NOT_SUP-PORTED	16#8	Some optional object or property function is not supported.
BACNET_STATUS_INVALID_PARAM	16#10	Some request should be sent but some parameters had invalid values.
BACNET_STATUS_BUSY	16#11	Some request should be sent but datalink or device is busy with other tasks.
BACNET_STATUS_ALREADY_EXISTS	16#12	Construction of object failed because of duplicate name or instance.
BACNET_STATUS_BAD_CONFIG	16#13	BACnet stack datalink layer has some bad configuration detected.
BACNET_STATUS_OBJECT_NOT_FOUND	16#14	The requested object was not found.
BACNET_STATUS_PROPERTY_NOT_FOUND	16#15	The requested object property was not found.
BACNET_STATUS_MISSING_RE-QUIRED_PARAM	16#16	Some parameter was not found.
BACNET_STATUS_SECURITY_EN-TRY_NOT_FOUND	16#17	The requested security entry was not found.
BACNET_STATUS_OUT_OF_MEMORY	16#20	Application has run out of memory.
BACNET_STATUS_OUT_OF_RESOURCES	16#21	Application has run out of system resources (sockets/ timers...).
BACNET_STATUS_XMT_OVERFLOW	16#22	Application transmitter queue is missing data.
BACNET_STATUS_RCV_OVERFLOW	16#23	Application receiver queue is missing data.
BACNET_STATUS_TIMEOUT	16#30	Request sent had no response received. If the functions BACnetReadProperty, BACnetReadPropertyMultiple, BAC-NET_READ_PROP_COMPLETE_CB return this as status the request could be sent but no response has been received within the configured APDU timeout.
BACNET_STATUS_REQUEST_TOO_LONG	16#31	Request is too long to be sent to datalink.
BACNET_STATUS_ILLEGAL_RESPONSE	16#32	Application tried to send an invalid response for some received request.
BACNET_STATUS_RESPONSE_TOO_LONG	16#33	Response is too long to be sent to datalink.
BACNET_STATUS_INVALID_DATA_TYPE	16#34	Application specified an invalid datatype.
BACNET_STATUS_VAL_OUT_OF_RANGE	16#35	Application specified an value which is out of range.
BACNET_STATUS_VAL_OUT_OF_SPACE	16#36	Application specified an value which is too long.
BACNET_STATUS_SEG_NOT_SUPPORTED	16#37	Application tried to send an segmented request but segmentation is not supported or disabled.
BACNET_STATUS_TRANSACTION_ABOR-TED	16#38	An BACnet transaction has been cancelled.

Name	Initial	Comment
BACNET_STATUS_BACNET_ERROR	16#40	An sent request has been answered with an BACnetError response. If the functions BACnetReadProperty, BACnetReadPropertyMultiple, BACNET_READ_PROP_COMPLETE_CB return this as status the provided BACNET_ERROR structure will contain additional error information.
BACNET_STATUS_BACNET_REJECT	16#41	An sent request has been answered with an BACnetReject response. If the functions BACnetReadProperty, BACnetReadPropertyMultiple, BACNET_READ_PROP_COMPLETE_CB return this as status the provided BACNET_ERROR structure will contain additional error information.
BACNET_STATUS_BACNET_ABORT	16#42	An sent request has been answered with an BACnetAbort response. If the functions BACnetReadProperty, BACnetReadPropertyMultiple, BACNET_READ_PROP_COMPLETE_CB return this as status the provided BACNET_ERROR structure will contain additional error information.
BACNET_STATUS_INVALID_PACKET	16#43	Application received some invalid packet (obsolete code).
BACNET_STATUS_SEG_ERROR	16#44	Application received some invalid segmented packet (obsolete code).
BACNET_STATUS_INCONSISTENT_TAGS	16#45	Application received some packet with inconsistent tags (obsolete code).
BACNET_STATUS_CANNOT_BIND_TO_PORT	16#49	Application tried to bind to an datalink port which doesn't exist.
BACNET_STATUS_DRV_NOT_FOUND	16#50	Application tried to bind to an datalink (obsolete code).
BACNET_STATUS_DRV_NOT_SUPPORTED	16#51	Configuration has an unknown or unsupported datlink configured (obsolete code).
BACNET_STATUS_DRV_CANT_INIT	16#52	Some datalink could not be initialized correctly (obsolete code).
BACNET_STATUS_DRV_CANT_SEND	16#53	Application tried to send some request but corresponding datalink queues are full.
BACNET_STATUS_WRITEACCESS_DENIED	16#60	Property which is not writeable has been accessed (obsolete code).
BACNET_STATUS_NO_ROUTE_FOUND	16#61	Frame could not be sent over router (obsolete code).
BACNET_STATUS_SEC_DIS_NON_TRUST	16#70	Security disabled non trusted routing.
BACNET_STATUS_SEC_INVAL_KEY_DATA	16#71	Invalid key data specified.
BACNET_STATUS_SEC_INVAL_DEST_ID	16#72	Invalid destination device id specified.
BACNET_STATUS_SEC_INVAL_SRC_ID	16#73	Invalid source device id specified.
BACNET_STATUS_SEC_BAD_VALIDATION	16#74	Unable to validate message.
BACNET_STATUS_SEC_DEC_NO_DECRYPT	16#75	Unable to decrypt and verify the message (possible wrong key).
BACNET_STATUS_RAW_VALUE	16#F0	Some proprietary value has been decoded (only for internal use).
BACNET_STATUS_PRIMITIVE_VALUE	16#F1	Some proprietary primitive value has been decoded (only for internal use).

Name	Initial	Comment
BACNET_STATUS_STANDARD_VALUE	16#F2	Some standard value has been decoded (only for internal use).
BACNET_STATUS_INVALID_VALUE	16#F3	Some invalid value has been decoded (only for internal use).

Tab. 13-12: TYPE IEC_BACNET_STATUS

13.2.3 Configuring the BACnet server

Commissioning the BACnet server

To start the BACnet server properly

1. A separate PLC task (slow 100 ms, without watchdog) is required for the start, as some initializations require more time.
2. A "BACnetStartPRG" program is required to execute the steps and their sequence mentioned in the following. More initializations can be made in the steps before the status is set to "BACnetReady"

   Required steps:

   1. Set the time zone
   2. Set the BACnet server communication settings with IP address
   3. Register plugins and objects that have to be registered before executing the `StartBACnetStack()` function
   4. Start the BACnet server with the `StartBACnetStack()` function and set a BACnet Ready flag
   5. Register plugins and objects that have to be registered after executing the `StartBACnetStack()` function
3. A "BACnetStartPRG" program is required to execute regular triggering of the time and to refresh the BACnet objects, see"Creating a BACnet project" on page 333 step 4.



It is recommended to initialize all plugins used before executing the function `StartBACnetStack()`.

Plugins with a mandatory initialization have a respective note in the description.

The following plugins have to be initialized before executing the function `StartBACnetStack()`:

- TrendLog
- ObjectPersistence
- TimeSync und UtcTimeSync
- deviceDateTime

Creating BACnet servers and objects and starting a BACnet server

Creating a BACnet project

Only one BACnet server can be added to one control. This server represents the device as device in the BACnet network.

This procedure describes how to create and start a BACnet server.

To configure a BACnet project:

1. Create a control in the IndraWorks Project Explorer (e.g. XM2x).
2. Add a "BACnet server" to the control.

   Open the context menu of the control and select the context menu item **Add ▸ BAC server**.

3. Add the object "BACnet IP Datalink" to the "BACnet server". This is required to configure the BACnet communication settings in the PLC. The "BACnet IP Datalink" object is for example used to configure the network number (NetworkNumber, default = 1), the IP address and the BACnet port. The file bacstac.ini is generated automatically in the system.

   The PLC program is used in the next step to assign the IP address.

4. Create a PLC program "`BACnetSartPRG`" to initialize the BACnet server.

5. Assign a new PLC task (e.g. BACnetTask) and add the program "`BACnetStartPRG`" to the task call list.

   The task "BACnetTask" initializes the BACnet server and can run in a slow interval (recommended task cycle time is 100 ms).

   → No watchdog may be used!

6. Initialize the time zone information to synchronize time and date of the BACnet server. Takes place before starting the BACnet server. Set the time zone information using the SysTimeRTC library to correctly display the local time of the BACnet and the UTC. Set that values of the structure "**TimezoneInformation**" matching your country and take the points in time of the time shift of your country into account. The two global variables "`g_ulDaylightDate : UDINT := TO_UDINT(DT#2020-03-29-2:0:0);`" and "`g_ulStandardDate : UDINT := TO_UDINT(DT#2020-10-25-3:0:0);`" are used to declare them. Global variables can be used to yearly adjust the point in time of the time shift.

   The "TimeSync" plugins are registered in the example.

7. The BACnet server is started and stopped using the PLC program "BACnetStartPRG" as shown in the following.

   The following functions in the program are always used:

   - To start the BACnet server: `StartBACnetStack`
   - To stop the BACnet server: `StopBACnetStack`
   - To assign the IP address: `BACnet_IP_datalink.IPaddress := [AAA.BBB.CCC.DDD];`

Program:

```
PROGRAM BACnetStartPRG
VAR
(* BACnet Init Case *)
    BACnetInit : INT := 0; // init case counter
    InitError : UINT := 0; // error counter

(* to set Timezone ... *)
    TZI : TimezoneInformation;  //form SysTimeRTC.library

(* get IP from PLC *)
    FB_GetIP : ML_GetIpAddress; // from ML_BASE.library
    (* IP als String *)
    IPString : STRING;
    (* IP als DWORD *)
    dwIPAdr : DWORD;
    (* IP als array of Byte *)
    IPAdr : ARRAY [0..3] OF BYTE := [192,168,1,1]; // default

// TimeSync --> from  BACnetDefaultImpl.library
    timeSync : BACnetDefaultimpl.TimeSync_SvcAppHook;
    utcTimeSync : BACnetDefaultimpl.UTCTimeSync_SvcAppHook;
```

```
    deviceDateTime : BACnetDefaultImpl.DeviceDateTime;

// trigger time sync of BACnet
    TimeTON : TON;
    TimeWait: TIME := T#1S;

END_VAR
```

Program:

```
IF UserVarGlobal.BACnetReady = FALSE THEN
// ...
// mandatory steps to initialize BACnet
// ...
    CASE BACnetInit OF
    0: (* set TimeZoneInformation of your local location in this example Germany *)
        // iBias - unit mintues - distance to UTC
        TZI.iBias := -60;
        // iDaylightBias - unit minutes - distance daylight saving time UTC if exist in your country
        TZI.iDaylightBias := -60;
        TZI.szDaylightName := 'CEST'; // Timzone shortcut
        TZI.szStandardName := 'CET'; // Timzone shortcut
        TZI.ulDaylightDate := UserVarGlobal.g_ulDaylightDate; // last Sunday in Mar start daylight
        TZI.ulStandardDate := UserVarGlobal.g_ulStandardDate; // last Sunday in Oct end daylight
        (*  *)
        SysTimeRtcSetTimezone(TZI); // set Timzone in runtime
        (* ... *)
        BACnetInit := BACnetInit + 1;

    1: (* get/set IP-Adress and start BACnet-Server *)
        // get IPAdress of control ...
        FB_GetIP(Execute := TRUE);
        (* run FB *)
        IF FB_GetIP.Done THEN
            FB_GetIP(Execute := FALSE);
            (* get IP-String *)
            IPString := FB_GetIP.IpAddress; // as string
            (* convert string to array of byte *)
            dwIPAdr := IL_IPV4Decode(IPString);
            (* convert DWORD to array of byte *)
            IPAdr[3] := DWORD_TO_BYTE( SHR( dwIPAdr,  0));
            IPAdr[2] := DWORD_TO_BYTE( SHR( dwIPAdr,  8));
            IPAdr[1] := DWORD_TO_BYTE( SHR( dwIPAdr, 16));
            IPAdr[0] := DWORD_TO_BYTE( SHR( dwIPAdr, 24));
            (* set datalink *)
            BACnet_IP_datalink.IPaddress := IPAdr;
            (* ... *)
            BACnetInit := BACnetInit + 1;
        END_IF // done
        (* *)
        IF FB_GetIP.Error THEN
            FB_GetIP(Execute := FALSE);
            (* ... *)
            InitError := InitError + 1;
            (* set default Adress ... *)
            BACnet_IP_datalink.IPaddress := IPAdr;
            (* ... *)
            BACnetInit := BACnetInit + 1;
        END_IF // error

    2: (* prepare BACnet services for time synchronization see BACnetDefaultImpl *)
        timeSync.Logging := FALSE;
        timeSync.Init();
        BACnet_Server.RegisterPlugin(timeSync); // register Service Request Time
        (* ... *)
        utcTimeSync.Logging := FALSE;
        utcTimeSync.Init();
        BACnet_Server.RegisterPlugin(utcTimeSync); // register Service Request UTCTime
        (* ... *)
        BACnet_Server.RegisterPlugin(deviceDateTime); // register Service DateTime
        (* ... *)
        BACnetInit := BACnetInit + 1;

    3:    // ...
        // Register other plugins, schedule, notification and/or calendar,
  // if component must be registered before start of BACnet server
        // ...
        BACnetInit := BACnetInit + 1;
```

```
    4: (* Start BACnet Server *)
        (* ... *)
        BACnet_Server.StartBACnetStack();
        (* ... *)
        BACnetInit := BACnetInit + 1;

    5:    // ...
        // Register other plugins, schedule, notification and/or calendar,
  // if component must be registered after start of BACnet server
        // ...
        BACnetInit := BACnetInit + 1;

    6: (* finish *)
        IF InitError = 0 THEN
            UserVarGlobal.BACnetReady := TRUE;
        ELSE
            ;
        END_IF
        (* *)

        BACnetInit := 0;

    END_CASE // BACnetInit

END_IF // UserVarGlobal.BACnetReady

IF UserVarGlobal.BACnetReady THEN

    (* BACnet Server - set Time and Date *)
    (* must be called cyclic that the BACnet time is correct *)
    (* currently its once a second, can be adjusted for application need *)
    TimeTON(IN:=TRUE, PT:=TimeWait);
    IF (TRUE = TimeTON.Q) THEN
        TimeTON(IN:=FALSE);
        // need to call deviceDateTime, because of datetime calculation
        deviceDateTime();

    END_IF

    (* ... *)

    (* ... *)

    (* ... *)


END_IF // BACnetReady
```

8. Declare the global variable "`BACnetReady`" in a global variable list.

   The PLC program "`BACnetStartPRG`" uses the variable and indicates that the BACnet server has been started.

9. To synchronize the time of the BACnet server, call the "**deviceDateTime()**" function cyclically, in the example each second in the PLC program "BACnetStartPRG". This is required to switch the Schedule object at the expected time for example.

10. Add the BACnet objects using the context menu of the "BACnet server".

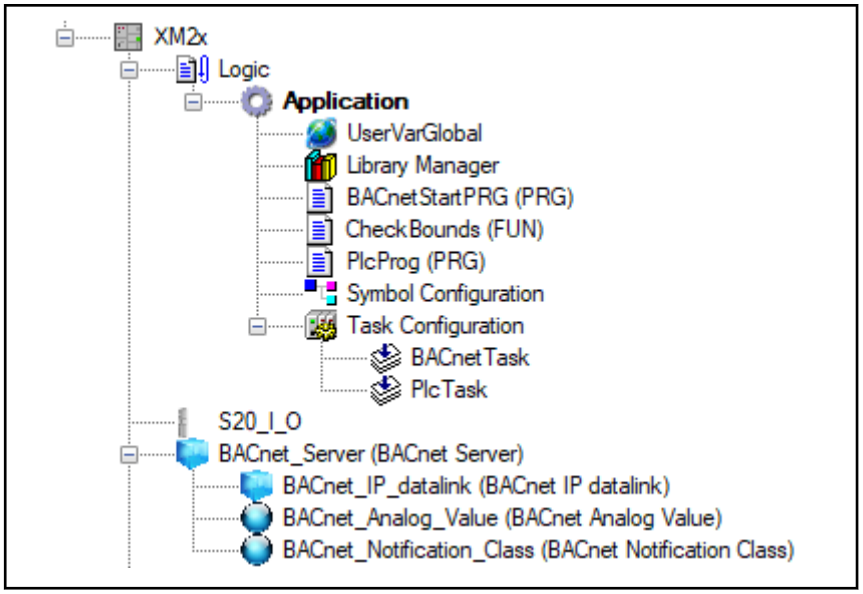
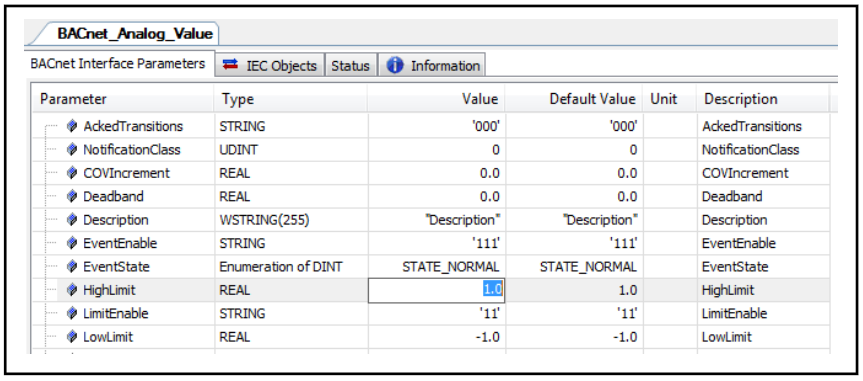


Fig. 13-11: BACnet server configuration example

11. To set the device properties, double-click into the Project Explorer to open the generic device editor of the object.

    Edit the properties of the object in the "Value" column of the "BACnet interface parameter" tab.



Parameter	Type	Value	Default Value	Unit	Description
AckedTransitions	STRING	'000'	'000'		AckedTransitions
NotificationClass	UDINT	0	0		NotificationClass
COVIncrement	REAL	0.0	0.0		COVIncrement
Deadband	REAL	0.0	0.0		Deadband
Description	WSTRING(255)	"Description"	"Description"		Description
EventEnable	STRING	'111'	'111'		EventEnable
EventState	Enumeration of DINT	STATE_NORMAL	STATE_NORMAL		EventState
HighLimit	REAL	1.0	1.0		HighLimit
LimitEnable	STRING	'11'	'11'		LimitEnable
LowLimit	REAL	-1.0	-1.0		LowLimit

Fig. 13-12: BACnet device editor – Configuration of object properties



Not all **properties** can be configured via the device editor.

Some settings (e.g. properties with array data types/BACnet lists) can only be set in the PLC program using the interfaces of the automatically instantiated function blocks of the objects.

Refer to chapter "PLC interface to the object properties" on page 341.



An **online display** of the current values of the object properties is possible in the tab "BACnet interface I/O mapping" within the function block instance variable if logged in.

However, the tab "BACnet interface parameter" shows even in the logged-in state only the configured values and not the current values of the object properties.

BACnet server settings

InstanceNumber (in "Device properties")	The BACnet server represents the "Device" object and the device in the BACnet network. Each BACnet device in the network requires a unique device ID (= instance number of the device object). Thus, set the instance number of the server device as specified in the network configuration (default value is "1").
AutoStart	• "AutoStart" = FALSE: (default, recommended) The BACnet server is not started automatically. The BACnet server can be started and stopped using the BACnet server function block methods `StartBACnetStack` and `StopBACnetStack`. ☞ Note that all BACnet objects are initialized with their properties when called for the first time. This initialization might take some time. Thus, use the `StartBACnetStack` in a separate task without watchdog. • "AutoStart" = TRUE: The BACnet server is started automatically if the PLC application is loaded. This application includes the BACnet server. ☞ The "BACstac.ini" file is required in the user partition of the control. The file includes the IP address, network number, etc. of the BACnet server.

Tab. 13-13: BACnet server settings

Overview on tabs

BACnet interface parameter	Configuration of object properties
BACnet interface I/O mapping	Mapping of the function block instance variable. It is also used as online display
Status	Indicates whether the server or the object is "running"
Information	General device description of the object

Tab. 13-14: BACnet - Overview on tabs

Example configuration

The following example shows an exemplary configuration of two BACnet objects to write an analog value to the PLC at runtime and to trigger an alarm using the BACnet "NotificationClass" when the current BACnet value is beyond a specific range.

To configure the example, proceed as follows:

1. Use the context menu to add a BACnet server to the control
2. Add the objects BACnet "AnalogValue" and BACnet "NotificationClass" to the BACnet server using its context menu

3. Double-click into the Project Explorer to open the device editor of the object "BACnet_Analog_Value".

   Open the "BACnet interface parameters" tab in the device editor.

   Set the following properties in the table column "Value":

   - AckedTransitions: `'111'`
   - NotificationClass: `1`
   - HighLimit: `10`
   - LimitEnable: `'11'`
   - LowLimit: `0`

   This enables the limit values `[0, 10]` for the analog value ("PresentValue") and the "NotificationClass" (instance 1) generates alarms when the value range is exceeded.

4. Declare an `lrValue` variable of the type `LREAL` in the PLC program and insert the following program code into the implementation part:

Program:

```
BACnet_Analog_Value_1.PresentValue := BACnet_Analog_Value_1.PresentValue + 0.0001;
lrVal := BACnet_Analog_Value_1.PresentValue;
```

5. Load the application to the control and start it.

   Result:
   The "`PresentValue`" of the object "`BACnet_Analog_Value`" is incremented.

6. Start an external BACnet client, e.g. "Innea BACnet Explorer" or "MBS BACeye"

   Result:
   There is the following behavior when starting the external client:
   The server device generates a "`NotificationClass`" alarm when the "`PresentValue`" is beyond the upper limit value `10`

This example shows how to access the properties of a BACnet object in the PLC program at runtime.

If no explicit "Property" of a BACnet object function block is available, the respective function block methods or auxiliary functions of the "BACnet" library can be used.

Refer to chapter "PLC interface to the object properties" on page 341.

BACnet plugin - Persistence

☞ Execute the plugin "Persistence" before the BACnet start command `BACnet_Server.StartBACnetStack();`.

The plugin cannot be used if the BACnet server AutoStart is configured

The plugin "Persistence" saves selected properties of objects to a binary data file per object and has an index file for saving.

Naming convention for the object persistence file: `pdata_{deviceId}_{objectType}_{objectInstanceNumber}.dat`.

Comply with the naming conventions to ensure that object information for a BACnet object that match the tuple {deviceId},{objectType},{objectInstance-Number} can be identified and restored.

The "Persistence" plugin provides an optional function to save and restore objects created by the client service `CreateObject-Requests` (this option is enabled by default).

The "Persistence" plugin has to be registered before starting the BACnet server to ensure the restoration of objects from the persistence during the stack initialization using the command `StartBACnetStack()`.

Program:

```
(* Persistenz *)
            objectPersistence.Logging := TRUE;
            objectPersistence.HardShutdown := TRUE;
            objectPersistence.PersistentPath := '/USER/BACnet/bacnetfiles/persistence';
            objectPersistence.RestoreClientCreatedObjects := FALSE;

            // BACnet_PersistentServer.RegisterPlugin(objectPersistence);

            IF BS_Otto.RegisterPlugin(objectPersistence) <> CmpErrors.Errors.ERR_OK THEN
                // Log an Error that it was not successful
                InitError := InitError + 1;
            END_IF

            //BACnet_PersistentServer.RegisterPlugin(logPropertyAttributes);
            IF BS_Otto.RegisterPlugin(logPropertyAttributes) <> CmpErrors.Errors.ERR_OK THEN
                // Log an Error that it was not successful
                InitError := InitError + 1;
            END_IF

            (* ! ! ! Important ! ! ! *)
            // ObjectPersistence.DeleteAllData(); -> Delete only the latest files not the temp files
            (*
                The function ObjectPersistence.DeleteAllData();
                must be called, after the PLC application is changed
          e.g. Download all, new BACnet objects added
                Otherwise a corrupt BACnet configuration is the result ! ! !
            *)
            (* ... *)
(* use DumpDataBase to save property attributes, which are marked to be persistent,
   cyclic executed in PLC program e.g. once a day
   need to be called after ObjectPersistence.DeleteAllData(); , too
*)

   // ObjectPersistence.DumpDataBase(); *)

            (* ... *)
            (* ... *)
```

⚠ CAUTION **If basic changes are made at the BACnet objects, persistence data can become corrupt.**

Delete the old persistence data in the respective path used, e.g. with an FTP client.

In this code example, the path is: `/USER/BACnet/bacnetfiles/persistence`

BACnet plugin - Schedule

☞ Execute the plugin "Schedule" after the BACnet start command `BACnet_Server.StartBACnetStack();`.

Plugin function:

The plugin allows to create and manage schedules to execute repetitive activities and to define single exceptions at a certain point in time.

BACnet plugin - Calendar

☞ Execute the plugin "Calendar" after the BACnet start command `BACnet_Server.StartBACnetStack();`.

Plugin function:

The plugin allows to create and manage absentee, public holiday and vacation lists to change or pause scheduler events.

The configurable periods are without time reference, i.e. only for full days.

BACnet plugin - Notification Class

☞ Execute the plugin "Notification Class" after the BACnet start command `BACnet_Server.StartBACnetStack();`.

Plugin function:

Use the plugin to create and manage mailing lists for event messages.

Individual BACnet objects are summarized in/linked to a distributor list.

BACnet plugin - Trend Log

☞ Execute the plugin "Trend Log" after the BACnet start command `BACnet_Server.StartBACnetStack();`.

Plugin function:

The plugin provides a trend recording of BACnet object values over a defined period and allows to visually display or apply them to a database.

PLC interface to the object properties

A function block instance (from the "BACnet" library) is created automatically for the BACnet server and the BACnet objects. The name of this IEC instance is identical to the names of the objects in the Project Explorer. The IEC instance represents the interface between the BACnet objects and the PLC. The mapping of the function block instance variables is also displayed in the “I/O mapping” tab of the respective device editor.

Object function blocks provide function block properties for the most frequently used object properties. These properties allow a direct access to read and write the properties. Additionally, there are also specific object methods or general access functions.

The specific object function blocks derive from the basic function block "`BACnetStaticObjectBase`" and thus from "`BACnetObjectBase`". The basic function blocks provide methods available to all object function blocks.

SetPropertyInstance

The `SetPropertyInstance` method for example allows to change any BACnet property:

- Data of the BACnet property to be set is transferred in the `IEC_BACNET_PROPERTY_CONTENTS` structure.
- Apart from an enumeration describing the data type, this structure also includes a buffer with a `POINTER TO BYTE` to actual data.
- Thus, any BACnet property data structures can be transferred within the `IEC_BACNET_PROPERTY_CONTENTS` structure. The specified data type defines to which structures or data the buffer point to. The PLC data

type IEC_BACNET_X corresponds to the BACnet data type DATA_TYPE_X.

Basic function blocks provide auxiliary functions, e.g. SetBACnetRealProperty, SetBACnetBooleanProperty, for some frequently used data types.

The following example shows the use of IEC_BACNET_PROPERTY_CONTENTS how a BACnet property of type REAL is set to 0.0.

Program:

```
METHOD SetBACnetRealProperty
VAR_INPUT
 propID    : CmpBACnet.IEC_BACNET_PROPERTY_ID;
 rVal      : REAL;
END_VAR
VAR
 BACreal   : CmpBACnet.IEC_BACNET_REAL             := 0.0;
 conts     : CmpBACnet.IEC_BACNET_PROPERTY_CONTENTS := ( tag := CmbBACnet.IEC_BACNET_DATA_TYPE.DATA_TYPE_REAL,
 nElements := 1,
 buffer    := ( nBufferSize := SIZEOF(LREAL), pBuffer := ADR(BACreal) ),
 rawBuffer := ( nBufferSize := 0, pBuffer :=0) );
END_VAR

BACreal  := rVal;
SetPropertyInstance(propID, -1, conts);
```

GetPropertyInstance

To retrieve an object property, use the method GetPropertyInstance:

- Use the method to transfer a buffer within the IEC_BACNET_PROPERTY_CONTENTS structure which is pre-initialized with the correct data type and then filled.
- Furthermore, the method can be used to transfer a buffer with the size "0" which is then automatically initialized.



The buffer has to be released after use!

Use the function FreeStackAllocatedMemory.

Irrespective of this general case, auxiliary functions (GetBACnet-x property) are provided for the most important data types.

Data types

The object function blocks of the "BACnet" library generally use BACnet data types in the function block interfaces (method arguments, method return values, function block properties).

If other data types, e.g. IEC basic data types, are used at the interfaces, respective methods and properties are used to implicitly convert from/to the internally used BACnet data types.

IEC BOOL ⇔ CmpBACnet.IEC_BACNET_BOOLEAN	Using BOOL (TRUE/FALSE) instead of CmpBACnet.IEC_BACNET_BOOLEAN improves the readability of the IEC code.
IEC WSTRING ⇔ CmpBACnet.IEC_BACNET_STRING	Using WSTRING is required, as there is a conversion from/to the internally used BACnet data types CmpBACnet.IEC_BACNET_STRING_TYPE.BACNET_STRING_UTF_8 and CmpBACnet.IEC_BACNET_STRING_TYPE.BACNET_STRING_ISO_8859_1.
IEC DATE / TIME / BacnetDateTime ⇔ CmpBACnet.IEC_BACNET_DATE / CmpBACnet.IEC_BACNET_TIME	Using IEC DATE/TIME is more common for the IEC programmer and thus preferred at the interface.

Tab. 13-15: Examples of data type conversion

Exporting and importing a BACnet configuration in the EDE format

Probably available in 15V10 and above.

The current BACnet configuration in the Project Explorer of the project can be exported to a *.csv file in EDE format (Engineering Data Exchange).

- The command **BACnet ▸ Export to EDE file** opens the “EDE export” dialog to enter project name, file version and author of the last change. The dialog opens to save a file in the file system.
- Load a BACnet configuration from an EDE file to a project via **BACnet ▸ Import from EDE file** . The command opens the standard dialog to select the respective *.csv in the file system. The “EDE import” dialog is now displayed containing information on the selected EDE file.

  If the selected EDE file includes the BACnet configurations of multiple devices, select the device (EDE file from the BACnet device). If your project contains several controls, select the device to import the BACnet configuration to.

13.2.4 Configuring the BACnet/IP client

In addition to BACnet objects, client function blocks can also be added as "Devices" under a BACnet server.

The BACnet client functionality assumes a configured BACnet server, refer to chapter 13.2.3 "Configuring the BACnet server" on page 333

As for the BACnet server and the BACnet objects, an IEC instance is automatically created with the same name for the BACnet client function blocks.

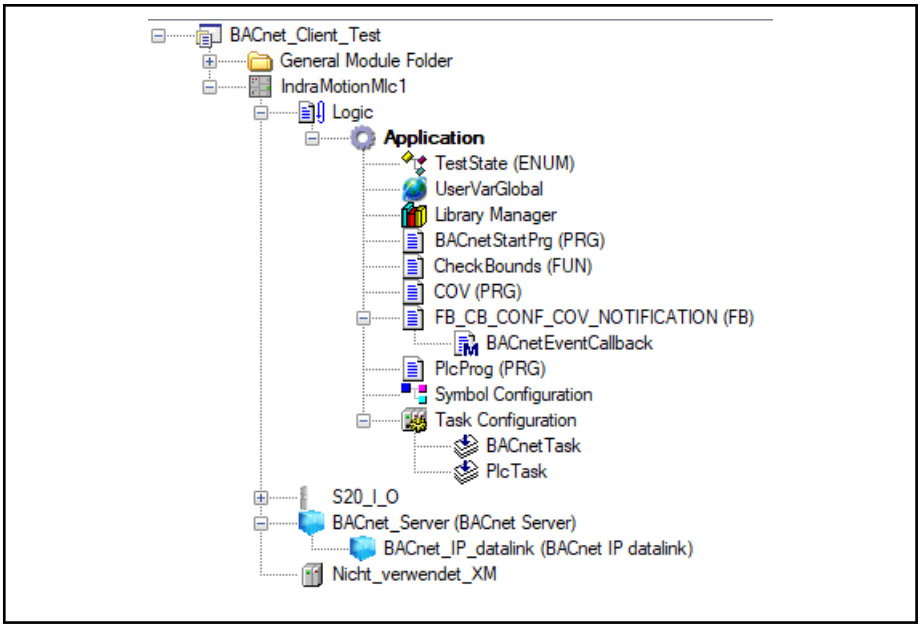


Fig. 13-13: BACnet client configuration

As the client functionality does not require any static configuration, creating the client "Devices" in the project tree is not mandatory. Function blocks can also be instantiated manually in the PLC program.

If the client function blocks are instantiated manually, create a shortcut to the server. Use the client function block method `RegisterToServer(BACnet_Server)` for a shortcut in the PLC program.

This takes place automatically in the Project Explorer when configuring the client function blocks.

☞ Even if a control is only to be used as BACnet client, the BACnet server has to be configured, see chapter 13.2.3 "Configuring the BACnet server" on page 333.

The server device in the project tree instantiates the "Device object" required by each BACnet device.

The BACnet client function blocks follow the "Common Behaviour Model". Thus, they are provided with inputs and outputs for the sequence control:

- Inputs `xExecute` and `xAbort`
- Outputs `xBusy`, `xDone`, `xError` and `xAborted`
- Inputs an outputs representing either the client request parameters or results.

PLC program example

The following code sends a "`Client-Read-Property-Request`" to a specific BACnet device in the BACnet network using the function block "`BACnetClientReadProperty`" from the BACnet.library.

To retrieve the output variables `readProp.result` of the type `IEC_BACNET_PROPERTY_CONTENTS`, the auxiliary function "`GetRealFromContents()`" is used.

Program:

```
PROGRAM PlcProg
VAR
    initDone          : BOOL := FALSE;
    readProp          : BACnet.BACnetClientReadProperty;
    readPropCnt       : INT := 0;
    readPropErrorCnt  : INT := 0;
    readPropCancelCnt : INT := 0;
    readPropVal       : REAL;
END_VAR

IF NOT initDone THEN
    readProp.RegisterToServer(BACnet_Server);
    readProp(
        dwTargetDeviceNumber := 43,
        objType:= BACnet.CmpBACnet.IEC_BACNET_OBJECT_TYPE.OBJ_ANALOG_VALUE,
        objInst:= 1,
        propID:= BACnet.CmpBACnet.IEC_BACNET_PROPERTY_ID.PROP_PRESENT_VALUE,
        nIndex:= -1);
END_IF

IF readProp.xExecute AND readProp.xBusy AND readProp.iState = 3 THEN
    IF readPropCnt MOD 2 = 0 THEN // cancel every second request
        readProp.xAbort := TRUE;
        readPropCancelCnt := readPropCancelCnt + 1;
    END_IF
END_IF

IF NOT readProp.xExecute AND NOT readProp.xAborted AND NOT readProp.xDone THEN
    readProp.xExecute := TRUE;
    readPropCnt := readPropCnt + 1;
END_IF

IF readProp.xExecute THEN
    IF readProp.xDone THEN
        readProp.xExecute := FALSE;
    END_IF
    IF readProp.xError THEN
        readProp.xExecute := FALSE;
        readPropErrorCnt := readPropErrorCnt + 1;
    END_IF
    IF readProp.xAborted THEN
        readProp.xExecute := FALSE;
        readProp.xAbort := FALSE;
    END_IF
END_IF

readProp();

IF readProp.xDone OR readProp.xError THEN
    IF readProp.xError THEN
        ; // ToDo handle error
    ELSE
        readPropVal := BACnet.GetRealFromContents(readProp.result);
    END_IF
END_IF
```

13.2.5 Notes on implemented BACnet objects and services

General information

For the described implementation of a BACnet API/stack, the major part of the object response specified in the BACnet standard, has already been implemented.

There are still specific object actions that have to be implemented into the application to ensure that the actions completely comply with the BACnet standard (e.g. to obtain the BTL certification).

The following description lists all object actions and the object response implemented or not implemented into the current BACnet API/stack.

Additionally, there are descriptions and an example of how to implement missing object responses.

Object - Property - Actions

The general object property actions described in the following are implemented for all objects containing the listed properties.

Property StatusFlags

The `bit string` value of the `StatusFlags` property is 4 bits:

- `IN_ALARM`
- `FAULT`
- `OVERRIDDEN`
- `OUT_OF_SERVICE`

The values of these three bits refer to the values of other properties in the same object.

The relation is defined in the BACnet standard and it is implemented as action in the described API:

Logical behavior:

- The bit `IN_ALARM` is logic `FALSE` if the `EventState` property value is `NORMAL`.

  Otherwise, `IN_ALARM` is logically `TRUE`.
- The bit `FAULT` is logic `TRUE` if the `Reliability` property exists and if its value is not `NO_FAULT_DETECTED`.

  Otherwise, the `FAULT` bit is logically `FALSE`.
- If the `OVERRIDDEN` bit is `TRUE`, it indicates that the `PresentValue` property and the `Reliability` property do not map the physical input anymore, but that they were overwritten by another mechanism.

  Otherwise, the `OVERRIDDEN` bit is logically `FALSE`.
- The bit `OUT_OF_SERVICE` is logic `TRUE` if the `OutOfService` property exists and if its value is not `TRUE`.

  Otherwise, the `OUT_OF_SERVICE` bit is logically `FALSE`.

Property OutOfService

The `OutOfService` property (`BOOL`) indicates the operating state of the physical point that represents the BACnet object:

- `TRUE` = Operating
- `FALSE` = Not operating

If the physical point is not being operated, it has to be possible to write on other properties.

It has to be possible to write on the properties `PresentValue` and `Reliability` to simulate different events and states of an object that do not result during the normal operation for example.

If the `OutOfService` property is set to `TRUE`, it can be written on the event and state properties even if the user application did not define any access rights for the properties.

Properties relating to intrinsic reporting

For objects defining intrinsic reporting procedures in the standard and that are also supported by the present API, their properties are automatically updated with regard to intrinsic reporting by internal actions in the API.

Examples:

- The property `EventState` is set in the respective enumeration value for the intrinsic status of the object.
- The `EventTimeStamps` property is updated when the event confirmation is sent.

- Bits in the AckedTransitions property are also changed correspondingly.

Properties relating to change in value Reporting (COV)

For objects with COV reporting procedures defined in the standard and that are also supported by the API, their properties are automatically updated with regard to COV reporting by internal actions in the API. The API sends notifications to changes in values to subscribed devices for objects or object properties.

Properties relating to command prioritization (outputs)

For BACnet objects, commands are prioritized based on a specified number of properties assigned to command generating units. A prioritized command for a commandable property of an object is executed using either a Write-Property service request or a WritePropertyMultiple service request. The basic request includes a conditional Priority parameter between 1 and 16. Each commandable priority of an object is provided with a prioritization table represented by the Priority_Array property. The Priority_Array consists of an array of commended values in descending priority. The first value in the array corresponds to priority 1 (highest), the second value corresponds to priority 2, etc. up to the 16th value that corresponds to priority 16 (lowest). If the API receives such a prioritized command, all actions described in section 19 in the BACnet standard are executed, as internal actions handle the prioritized command correctly.

If the application subscribed Callbacks for the properties Present_Value and Priority_Array, it is informed on the current command status in the object.

Object Accumulator

The object type Accumulator defines a standardized object. Its properties represent the externally visible characteristics of a device.

The object indicates measurements executed by pulse counting.

There are two options to enter the pulse at the object:

1. Adding the Prop_Pulse_Value_Source property with a reference to a BINARY or BOOLEAN value source
2. By calling the function BACnet.BACnetServer. UpdateAccumulatorDataSourceValue available for each inserted BACnet server in the Project Explorer.

In each case, toggling values of the object is counted and the API performs all property actions defined in the BACnet standard. The intrinsic reporting procedure for this object is also performed by internal actions in the API.

Object Analog-Input

The object type Analog-Input defines a standardized object. Its properties represent the externally visible characteristics of a physical analog input. The properties Min_Pres_Val and Max_Pres_Val showing the lowest and the highest number in Engineering units that can be reliably reached for the Present_Value property of this object.

Min_Pres_Val and Max_Pres_Val generate value out of range errors if Present_Value is beyond this range.

The values of the properties High_Limit, Low_Limit and Deadband are also monitored. It is monitored whether they are beyond this range.

The optional property Resolution is the smallest detectable change in Present_Value specified in engineering units that directly affects the setting of the Present_Value property and adjusts its value automatically.

The Change-Of-Value reporting procedure and the intrinsic reporting procedure are completely implemented for this object type by internal API actions.

Object Analog-Output

The object type `Analog-Output` defines a standardized object. Its properties represent the externally visible characteristics of a physical analog output.

All actions implemented for the `Analog-Input` object are also implemented for the object type `Analog-Output`.

Object Analog-Value

The object type `Analog-Value` defines a standardized object. Its properties represent the externally visible characteristics of an analog value.

The object is a monitoring system parameter in the memory of the BACnet device and does not have a physical device.

All actions implemented for the `Analog-Input` object are also implemented for the object type `Analog-Output`.

Object Averaging

The object type `Averaging` defines a standardized object. Its properties represent the externally visible characteristics of a value that is collected periodically over a certain period. The `Averaging` object shows the minimum, maximum and average value over a period and visibly provides them as properties of the `Averaging` object.

The logged value can be the value of a property of the following type:

- `BOOLEAN`
- `INTEGER`
- `UNSIGNED`
- `ENUMERATED`
- `REAL`

The value of the object referenced via `Object_Property_Reference` is determined and all dependent property values (e.g. `Minimum_Value`, `Maximum_Value` or `Average_Value`) are updated by internal actions in the API.

Objects without implementation of standard actions

No actions to a property interrelated to the following object types are internally implemented into the current API version:

- Object type `Notification-Class`
- Object type `Program`
- Object type `Structured-View`

Services with implementation of standard actions

The Service-Indication service hooks listed in the following are already implemented into standard actions and do normally not have to be implemented into the application:

AcknowledgeAlarm Service Hook

After the validity of the request has been verified, the API tries to find the specified object.

The following actions are executed if the object is available and if the `Time Stamp` parameter matches the point in time when the event was confirmed last:

- The bit in the `Acked_Transitions` property of the object matching the value of the `Event State Acknowledged` parameter is set to `1`.
- A basic result `result(+)` is created
- An event message is output with a `NotifyType` parameter that is "equal to `ACK_NOTIFICATION`"

If no object is found during the search, the basic result `result(-)` is output.

A confirmation message uses the same service type (confirmed or unconfirmed).

The confirmation message is for the same recipients to which the originally confirmed or unconfirmed event message were sent to.

The `Time Stamp` transferred in the confirmation message is not derived from the `Time Stamp` of the original event message, but includes the time when the confirmation message is generated.

GetAlarmSummary service hook

After the validity of the request has been verified, the API searches for all event-triggering objects with the following properties:

- An `Event_State` property unequal to `NORMAL`
- An a `Notify_Type` property with the value `ALARM`

If objects were found during the search, a positive response is created that contains the `Alarm Summaries` for the objects found.

If no objects were found that match the criteria, a list with a length of `0` is returned.

GetEnrollmentSummary service hook

After the validity of the request has been verified, the API searches for all event-triggering objects using search criteria specified in the basic request.

Search criteria are the logical operations of all explicitly set filters omitted in the basic request.

If objects were found during the search, a positive response is created that contains the `Enrollment Summaries` for the objects found.

If no objects were found that match these criteria, a list with a length of `0` is returned.

SubscribeCOV service hook

If there are neither the parameters `Lifetime` nor `Issue Confirmed Notifications`, the request is considered as canceled.

Each COV content that already exists for the same BACnet address and which is included in the PDU that routed the `SubscribeCOV` request and is provided with the same `Subscriber Process Identifier` and `Monitored Object Identifier` is disabled and `SimpleAck` is returned.

If no matching COV content is found for the created cancelation, it is still canceled as if there would be a content.

In this case, the cancelation returns the result `SimpleAck`.

If there are no `Lifetime` parameters, but an `Issue Confirmed Notifications` parameter, null (unlimited lifetime) is assumed as value for `lifetime`.

If the `Issue Confirmed Notifications` parameter exists, but the object to be monitored does not support the COV reporting, an error is issued.

If the object to be monitored supports the COV reporting, the following condition is checked:

- There is a COV context for the same BACnet address
  - included in the PDU
    - containing the `SubscribeCOV` request **and** the same `Subscriber Process Identifier` **and** `Monitored Object Identifier`.

  If an existing COV context is found, the request is handled as if a new subscription would be created.

If no COV context is found that matches the request, a new COV context is created containing the BACnet address of the PDU which contains the `SubscribeCOV` request and is provided with the same `Identifiers`:

- `Subscriber Process Identifier` **and** `Monitored Object Identifier`

If no context can be created, an error is returned.

If a new context is created or a new subscription is received, the COV context is initialized. Its lifetime is determined by the `Lifetime` parameter (if present).

If the `Lifetime` parameter does not exist, the value of the lifetime is `0`.

The subscription is cancelled automatically if no re-subscription has been received.

☞ If the service life is `0`, the subscription is unlimited and it is not canceled automatically.

`SimpleAck` is always returned in any case.

After a subscription or resubscription request, either the status `ConfirmedCOVNotification` or the status `UnconfirmedCOVNotification` is output in the `Issue Confirmed Notifications` parameter.

AddListElement service hook

After the validity of the request has been verified, the API tries to modify the object identified in the `Object Identifier` parameter.

If the identified object exists and if it has the `property` specified in the `Property Identifier` parameter, it is tried to add all elements listed in the `List of Elements` parameter to the specified `property`.

If adding was successful, a basic `SimpleAck` is generated.

If one or multiple of the elements are already in the list, they are ignored and not added to the list anymore.

Ignoring an already existing element does not cause the service to fail.

The service fails and generates a basic message in the following cases:

- If the specified object does not exist
- If the specified `Property` does not exist
- If the specified `Property` is not a list

If one or multiple elements cannot be added to the list and are not yet in the list, a basic error message is generated and no elements are added to the list.

The service either adds all specified elements of the list which are not yet included or it does not add any elements to the list at all.

RemoveListElement service hook

After the validity of the request has been verified, the API tries to modify the object identified in the `Object Identifier` parameter.

If the identified object exists and if it contains the `property` specified in the `Property Identifier` parameter, it is tried to remove all elements listed in the `List of Elements` parameter from the specified `property`.

If one or multiple elements do not exist or cannot be removed due to an insufficient authorization or other reasons, none of the elements is removed and a basic error message is generated.

ReadProperty service hook

After the validity of the request has been verified, the API tries to access the specified `property` of the specified object.

The API handles the `Object Identifier` as if it would correspond to the local device object if the following conditions are met:

- The object-type in the Object Identifier parameter contains the value Device Object **and** the instance in the Object Identifier parameter contains the value 4194303

This allows to determine a device instance of a device which does not generate I-Am messages.

Messages/error response:

A basic ComplexAck is generated after a successful access. The basic “ComplexAck” returns the read value.

If the access fails, a basic error message is displayed showing the cause of failure.

ReadPropertyMultiple service hook

After the validity of the request has been verified, the API tries to access the specified property of the specified object.

The List of Read Access Results is created. Its sequence is specified in the request.

If the List of Property References part of the List of Read Access Specifications parameter contains either the property identifier ALL, REQUIRED or OPTIONAL, the List of Read Access Results is designed as if each returned property would have been referenced explicitly.

The API ensures that all read operations take place in the shortest time possible. Preferential processing only takes place for processings with a higher priority.

The request is processed until it is tried to access all specified properties.

The API handles the Object Identifier as if it would correspond to the local device object if the following conditions are met:

- The object-type in the Object Identifier part of the Read Access Specification parameter contains the value Device Object **and** the instance in the Object Identifier parameter contains the value 4194303.

This allows to determine a device instance of a device which does not generate I-Am messages.

Messages/error response:

If none of the specified objects is found or if none of the specified parameters of the specified object can be accessed, a basic error or a basic ComplexAck is generated returning error codes for all properties.

If at least one of the specified properties for the specified objects can be accessed, a basic ComplexAck is generated returning the read values.

An error code is output for all properties that could not be accessed.

☞ **Note for the BACnet server user applications that cannot use the internal database and thus need to implement a support for the ReadPropertyMultiple service:**

Implement the ReadProperty service hook, as the ReadPropertyMultiple requests are split internally into individual ReadProperty requests and the completions are collected to create the response.

WriteProperty service hook

After the validity of the request has been verified, the API tries to modify the specified property in the specified object using the value provided in the Property Value parameter.

Messages/error response:

If the change attempt is successful, a basic `SimpleAck` is generated.

If the change attempt fails, a basic error message is generated displaying the error cause.

WritePropertyMultiple service hook

For each `Write Access Specification` in the `List of Write Access Specifications`, the value of each specified property is replaced with the property value specified in the `Write Access Specification`.

Subsequently, a basis `SimpleAck` is generated showing the successful processing of the complete service request.

The intended properties are processed in the sequence specified in the `List of Write Access Specifications`.

If a property is edited that cannot be changed during processing, the PI returns a basic error message indicating the cause of failure.

The service result is:

- Either all specified properties are changed **or** only the properties up to the property that is not specified in the `First Failed Write Attempt` parameter (exclusive the property specified)

Messages/error response:

If no write action was executed successfully and the service request was rejected completely, `BACnet-Reject-PDU` is displayed.

If one of the write operations from the `List of Write Access Specifications` could not be executed successfully, an error message is issued displaying the cause of failure.

ReadRange service hook

At first, the API verifies the validity of the parameters:

- `Object Identifier`
- `Property Identifier`
- `Property Array Index`

If the `Range` parameter does not exist, the API reads all available elements from the list or the array and returns the retrieved elements.

If the `Range` parameter exists and if it specifies the `By Position` parameters, the API reads all specified elements and returns the retrieved elements.

The specified elements include the element at the index position defined by the `Reference Index` plus those elements following up to `Count` - 1 if `Count` is positive or the elements preceding up to `Count` -1 if `Count` is negative.

The first element of a list is linked to index 1.

If the `Range` parameter exists and if it specifies the `By Time` parameter or the `By Sequence Number` parameter, the API rejects the request by returning an error message with the error class and error code.

The returned response transfers the number of read and returned elements using the `Item Count` parameter.

The current elements are returned in the item `Item Data` parameter.

If the returned response includes the first position index, the `FIRST_ITEM` flag is set to `TRUE` in the `Result Flags` parameter. Otherwise, it is set to `FALSE`.

If the returned response includes the last position index, the `LAST_ITEM` flag is set to `TRUE` in `Result Flags`. Otherwise, it is set to `FALSE`.

If the list does not include any elements matching the `Range` parameter criteria, a `ComplexAck` is returned with an `Item Count` of `0` and no `First Sequence Number` parameter.

The standard implementation of this service does NOT support access to the property `log-buffer` of the trend-logging and Event-logging objects.

Messages/error response:

If required, the API returns an error message with the respective error class and error code.

Potential error causes:

- Either the object or the property is unknown
- The data referenced is not a list
- The data referenced is an array
- No access possible

☞ If the application has to support the “Logging” objects, the saving and accessing of data have to be implemented into the application.

SubscribeCOVProperty service hook

The request is considered as being canceled if neither the `Lifetime` parameter nor the `Issue Confirmed Notifications` parameter exists.

Each COV content that already exists for the same BACnet address in the PDU that routes the `SubscribeCOVproperty` request and that is provided with the same `Subscriber Process Identifier`, `Monitored Object Identifier` and `Monitored Property Identifier` is disabled and `SimpleAck` is returned.

Cancelations without a matching COV context are handled successfully, as if there was a context and a `SimpleAck` is returned.

If a COV context is found, it is removed from the `Active_COV_Subscriptions` property in the Device object.

If the `Issue Confirmed Notifications` parameter exists, but the property to be monitored does not support COV reporting, an error message is issued.

If the property to be monitored supports “COV Reporting”, it is checked whether there is a COV context for the same BACnet address in the PDU that routes the `SubscribeCOVproperty` request and that is provided with the same `Subscriber Process Identifier`, `Monitored Object Identifier` and `Monitored Property Identifier`.

If an existing COV context is found, the request is interpreted as re-subscription and handled successfully as if it would be a newly generated subscription.

If no COV context is found that matches the request, a new COV request is generated.

The new COV context includes the BACnet address from the PDU that routes the `SubscribeCOVProperty` request and the same `Subscriber Process Identifier`, `Monitored Object Identifier` and `Monitored Property Identifier`.

The new context is included in the `Active_COV_Subscriptions` property of the “Device” object.

If no context can be generated, an error message is returned.

If either a new context was generated or a re-subscription was received, the COV context is initialized and it obtains a lifetime as defined in the `Lifetime` parameter.

The subscription is cancelled automatically if no re-subscription has been received.

A `SimpleAck` is returned and a `ConfirmedCOVNotification` or an `UnconfirmedCOVNotification` is created after a successful subscription or resubscription request as specified in the `Issue Confirmed Notifications` parameter.

GetEventInformation service hook

After the validity of the request has been verified, the API searches for all event-triggering objects matching the conditions listed below.

The search starts with the object behind the object defined with the `Last Received Object Identifier` parameter (in ascending internal order of object instances).

Conditions:

- The object either contains an `Event_State` property with a value unequal to `NORMAL` **or** an `Acked_Transitions` property with at least one of the following bits set to `FALSE`:
  - `TO-OFFNORMAL`
  - `TO-FAULT`
  - `TO-NORMAL`

A positive response is created. It contains the "Event Summaries" for objects found during this search.

If no objects were found that match the criteria, a list with a length of zero is returned.

The number of objects returned is equal to the number of objects that can be returned within the `APDU`.

If multiple objects match the criteria, but cannot be returned to the `APDU`, the `More Events` parameter is set to `TRUE`. Otherwise, it is set to `FALSE`.

I-Am service hook

After the validity of the request has been verified, the API adds the received information to the device object `Address_Binding` property.

It resolves device instance numbers in BACnet MAC addresses required to communicate to the device.

I-Have service hook

After the validity of the request has been verified, the API adds the received information to an internally sorted list.

It is used to resolve object names in the `Object_Identifier` required to access the objects.

Who-Has service hook

The API sends the unconfirmed `Who-Has` request using a broadcast address.

If the parameters `Device Instance Range Low Limit` and `Device Instance Range High Limit` are present, the API that is qualified to respond is the one with the `Object_Identifier Instance Number` being in the following range:

`Device Instance Range Low Limit` ≥ `Object_Identifier Instance Number` ≤ `Device Instance Range High Limit`.

The API

If the parameterObject Name is present, the API sends an I-Have service request with an object with an Object_Name property value corresponding to the Object Name parameter.

If the parameter Object Identifier is present, the API sends an I-Have service request with an object with an Object_Identifier property value corresponding to the Object identifier parameter.

Who-Is service hook

The API sends the unconfirmed Who-Is request using a broadcast address.

If the parameters Device Instance Range Low Limit and Device Instance Range High Limit are missing, the API returns their Device-Object_Identifier in individual responses using the I-Am service hook.

If the parameters Device Instance Range Low Limit and Device Instance Range High Limit are present, the API that is qualified to return their Device-Object_Identifier via the I-Am service hook is the API whose Device Object_Identifier Instance Number is in the following range:

Device Instance Range Low Limit ≥ Device Object_Identifier Instance Number ≤ Device Instance Range High Limit

AtomicReadFile service hook

At first, the API verifies the validity of the Object Identifier.

If the object is unknown, the API returns an error message result(-) with the respective error class and error code.

The AtomicReadFile service hook only supports files accessed via STREAM.

The validity of the local file access is then checked on the local drive.

If the result of the validity check is that the file start position is invalid or that the file cannot be accessed, a response result(-) is returned with the respective error class and error code.

If the validity check returns a positive result, the specified number of Request Record Count octets is retrieved from the local file.

If the actual number of read octets is smaller than the number specified, the Returned Record Count value of the returned result(+) response shows the number of octets read.

If the end of file was reached, the End Of File parameter of the returned result(+) response is set to TRUE.

This service hook is implemented using the SysFile interface and supports only the STREAM access.

Thus, the application specifically intends to change this service hook for the file access.

AtomicWriteFile service hook

At first, the API verifies the validity of the Object Identifier.

If the object is unknown, the API returns an error message result(-) with the respective error class and error code.

The service hook only supports files accessed via STREAM.

If the start position is invalid in the file or it cannot be accessed, the validity of the local file access on the local drive is checked and a response result(-) is returned with the respective error class and error code.

If the File Start Position parameter of the service request exceeds the local file size, the file to write the incoming octets is extended.

If the value of the File Start Position parameter is -1, the incoming octets are attached to the end of file.

If writing of the local file failed, a result(-) response is returned.

If the local file was written successfully, the response result(+) is returned containing the File Start Position.

This service hook is implemented using the SysFile interface and supports only the STREAM access.

Thus, the application specifically tries to change this service hook for the file access.

CreateObject service hook

The API verifies the validity of the request. It checks whether the Object Specifier of the request specifies a creatable object type (as per the internal rule: BACnet.IsBACnetObjectAMEVCreatable).

If no generatable object type can be specified, the API returns the error message result(-).

If specified in the Object Specifier, the validity of the object instance number is checked and an error message result(-) is issued if the instance number does not exist anymore.

If no List of Initial Values exists, all properties of the generated object are created using the standard values.

If the object generation fails, the API returns an error message result(-) and the First Failed Element Number value of the response is set to 0.

If the generation was successful, the API returns a result(+) response containing the Object Identifier of the generated objects.

DeleteObject service hook

At first, the API verifies the validity of the Object Identifier.

If the object is unknown, the API returns an error message result(-) with the respective error class and error code.

If the specified object can be deleted, the API returns a result(+) response.

If the deletion fails, the API returns a result(+) response, e.g. if the object cannot be deleted dynamically as per the internal rule BACnet.IsBACnetObjectAMEVCreatable.

Services with internal implementations

The following services do not contain any standard actions, but the Advanced Client API uses this set of functions to obtain data from removed devices:

- ConfirmedCOVNotification service hook
- ConfirmedEventNotification service hook
- UnconfirmedCOVNotification service hook
- UnconfirmedEventNotification service hook

These service Indication Hooks can be used to receive information returned after a SubscribeCOVProperty request.

A certain property of the object of a device of the COVNotification service hook is for example called after a subscription to inform on a property change.

Services without implementation of standard actions

The following services do not have an implemented standard action. The application has to implement own routines for the request handling if a service is to be supported:

- `DeviceCommunicationControl` service hook
- `ReinitializeDevice` service hook
- `TimeSynchronization` service hook
- `UtcTimeSynchronization` service hook

It might be required that the user application has to implement the listed service hooks (depends on the user application and its specifications acc. to the BACnet standard).

Examples on how to implement “Service Indication Hooks”

Implementation example "CreateObject" service hook

The following example shows how to implement a `CreateObject` service hook.

First, enter the service hook anywhere into the program code.

Declare the function block `CallbackFB` that implements the interface `IBACnetEventConsumer` with the `RegisterHook` function of the BACnetServer function block.

Entering the function block inserted into the Project Explorer for the local BACnet server device:

```
BACnetServer.RegisterHook(CmpBACnet.IEC_BACNET_CB_TYPE.CB_CREATE_OBJECT, CallbackFB);
```

When the BACnet server receives a CreateObject service, the function of the "BACnetEventCallback" function block is started (defined by the `IBACnetEventConsumer`interface).

The following function shows the declaration of the `BACnetEventCallback` function of the `CallbackFB` function block.

Declaration part:

Program:

```
METHOD BACnetEventCallback : BOOL
VAR_INPUT
   (*If only one event is possibly called, the following Ids can be ignored.
   If there are more at least the EventId should be checked, to get the right type of pParameter.
      ID of the occured Event. Contains the class and the event.*)
   EventId    : DWORD;
   (*Id of the parameter structure (see pParameter)*)
   usParamId  : WORD;
   (*Version of the parameter structure (see pParameter)*)
   usVersion  : WORD;
   (*Pointer to the event specific parameter, that is specified by Id (see GVL EVENTID in CmpBACnet)*)
   pParameter : POINTER TO BYTE;
END_VAR
VAR
   pCreateObj : POINTER TO CmpBACnet.EVT_BACNET_CREATEOBJECT;
   destInstNumber : CmpBACnet.IEC_BACNET_DWORD;

   objSpec    : CmpBACnet.IEC_BACNET_OBJECT_SPECIFIER;
   objID      : CmpBACnet.IEC_BACNET_OBJECT_ID;

   i          : UDINT;
   bacObj     : BACnetObjectBase;
   runObjID   : CmpBACnet.IEC_BACNET_OBJECT_ID;
   objCount   : UDINT := 0;

   strBuffer  : STRING;
   errCode    : CmpBACnet.IEC_BACNET_ERROR_CODE := 0;
```

```
   bacResult  : CmpBACnet.IEC_BACNET_STATUS;
END_VAR
```

Note the `pCreateObj` variable that contains the basic information of the `CreateObject` request.

The following function shows the implementation of the `BACnetEventCallback` function of the `CallbackFB` function block.

Implementation part:

Program:

```
BACnetEventCallback := FALSE;

//Check if the issued service event is the one we are interested in
//meaning the create object event
IF EventId = CmpBACnet.EVT_CreateObject AND
   usParamId = CmpBACnet.EVTPARAMID_CmpBACnet_CreateObject AND
   usVersion = CmpBACnet.EVTVERSION_CmpBACnet_CreateObject THEN

   //Get the service information using the proper
   //event structure CmpBACnet.EVT_BACNET_CREATEOBJECT
   pCreateObj := pParameter;

   //The pCBStatus property of the event structure allows to define
   //the result of this service hook. Choosing the default action
   //will advice the API to use its default implementations if available
   //(see also documentation of CB_STATUS_DEFAULT)
   pCreateObj^.pCBStatus^ := CmpBACnet.IEC_BACNET_CB_STATUS.CB_STATUS_DEFAULT;

   //Get the device instance number for the service request and check if it is
   //our device that the request is for (only important for the case of multiple
   //devices in one PLC)
   bacResult := CmpBACnet.BACnetDeviceAddressToInstNumber(pCreateObj^.destinationAddress,
                        ADR(destInstNumber));

   IF bacResult = CmpBACnet.IEC_BACNET_STATUS.BACNET_STATUS_OK
                        AND destInstNumber = m_Device.DeviceInstNumber THEN

      //Get the object specifier defining what to create
      objSpec := pCreateObj^.pServiceInfo^.objectSpecifier;
      //Check if the object type specified is creatable
      IF IsBACnetObjectAMEVCreatable(objSpec.objectSpec.typ) THEN

         //If no object ID is given find a free one
         IF objSpec.tag = CmpBACnet.IEC_BACNET_CREATE_OBJECT_TYPE.CREATE_OBJECT_TYPE THEN
            objID.typ := objSpec.objectSpec.typ;
            objID.instNumber := 1;
            objCount := GetObjectCount();
            i := 1;
            WHILE i <= objCount DO
               GetObject(i, bacObj);
               runObjID := bacObj.ObjectID;
               IF runObjID.typ = objID.typ AND
                                    runObjID.instNumber = objID.instNumber THEN
                  objID.instNumber := objID.instNumber + 1;
                  i := 1;
               END_IF
               i := i + 1;
            END_WHILE
            BACnetEventCallback := TRUE;

         //If the object ID is given check if it is a valid/free object ID
         ELSIF objSpec.tag = CmpBACnet.IEC_BACNET_CREATE_OBJECT_TYPE.CREATE_OBJECT_ID THEN
            objID := objSpec.objectSpec.objectID;
            objCount := GetObjectCount();
            i := 1;
            BACnetEventCallback := TRUE;
            WHILE i <= objCount DO
               GetObject(i, bacObj);
               runObjID := bacObj.ObjectID;
               IF runObjID.typ = objID.typ AND
                                    runObjID.instNumber = objID.instNumber THEN
                  errCode := ERR_CODE_OBJ_ID_ALREADY_EXISTS;
                  BACnetEventCallback := FALSE;
                  EXIT;
               END_IF
               i := i + 1;
            END_WHILE
```

```
         ELSE
            //No object specification given which can be interpretated, so
            //issue a standard error
            BACnetEventCallback := FALSE;
         END_IF

      ELSE
         //If the object type is not supported to be created issue the
         //appropriate error
         errCode := ERR_CODE_DYN_CREATION_NOT_SUPP;
         BACnetEventCallback := FALSE;
      END_IF

      //If everything is all right till now (BACnetEventCallback is TRUE) try to
      //create the object
      IF BACnetEventCallback THEN
         IF pCreateObj^.pServiceInfo^.fValuesPresent > 0 THEN
            //If property values are given by the service request use them
            bacResult := CmpBACnet.BACnetConstructObject(destInstNumber,
                              ADR(objID), pCreateObj^.pServiceInfo^.pValues,
                              pCreateObj^.pServiceInfo^.nValuesCount,
                              m_SupportUnsolicitedCOV, 1, m_HideToOutside);
         ELSE
            //If no property values are given by the service request just
            //create the object using the default implementation
            bacResult := ConstructDefaultObject(objID.typ, objID.instNumber,
                              STRING_TO_WSTRING(CONCAT('CreatedObject', DWORD_TO_STRING(objID.instNumber))),
                              bacObj, strBuffer);
         END_IF
      END_IF

      //Check if everything went well
      IF BACnetEventCallback AND bacResult = BACNET_STATUS_OK THEN
         //If everything succeeded issue an result(+) response containing the
         //object ID
         CmpBACnet.BACnetCreateObjectResponseCbCompletion(pCreateObj^.phTransaction,
                        ADR(objID));
         //We treated the service request, so no default API actions are
         //necessary anymore
         pCreateObj^.pCBStatus^ := CmpBACnet.IEC_BACNET_CB_STATUS.CB_STATUS_OK;
      ELSE
         //If something went wrong issue an error
         BACnetEventCallback := FALSE;
         //If no explicit error code was given till now, issue an internal error
         IF errCode = 0 THEN
            errCode := ERR_CODE_INTERNAL_ERROR;
         END_IF
         //Issue the result(-) response with the appropriate error code
         CmpBACnet.BACnetCreateObjectErrorCbCompletion(pCreateObj^.phTransaction,
         ERR_CLASS_OBJECT, errCode, 0);
         //We treated the service request, so no default API actions are
         //necessary anymore
         pCreateObj^.pCBStatus^ := CmpBACnet.IEC_BACNET_CB_STATUS.CB_STATUS_OK;
      END_IF
   END_IF
END_IF
```

The main task of this code example is to receive the service request information to access information in the `pParameter`.

The `pParameter` is accessed by casting `pCreateObj (EVT_BACNET_CREATEOBJECT)` to the `pParameter`.

The `CmpBACnet` library includes the hook request structure `pCreateObj` that can include the variable `pCBStatus` and that is the return value to the API.

The return value `pCBStatus` is the decisive factor whether the API continues to process service requests (`Default`) or whether it does not (`OK`).

Another function of the code example is the property check in the service request and the object generation. Resulting errors are saved in the `errCode` variable.

Finally, `result(+)` or `result(-)` responses are generated using the following functions:

- `CmpBACnet.BACnetCreateObjectResponseCbCompletion`, **or** `CmpBACnet.BACnetCreateObjectErrorCbCompletion`

The code example shows a typical implementation example of a service hook. All event mechanisms of the BACnet stack/IP are used.

☞ To get a call after a service request has been received, enter an Event/Callback.

To provide a compliant service, the service hook has to be implemented as provided in the specifications of the BACnet standard.

Implementing “Logging” objects (EL/TL/TLM)

For the following object type, the BACnet stack/API does not provide the complete implementation of the functionalities, as saving logged data is hardware-specific and thus implemented by the respective application:

- `Event-Log` (EL)
- `Trend-Log` (TL)
- `Trend-Log-Multiple` (TLM)

The possible response of the “Logging” objects is already implemented in the API. Thus, only applying and reading trend data has to be implemented into the user application.

Implementing data logging for “Logging” objects

Each “Logging” object uses its `Log_Buffer` property to save logged data.

The `Log_Buffer` property is accessed via the `ReadRange` service hook. Thus, an internal representation is not required in the application.

The BACnet stack/API handles the `Log_Buffer` as if it were only one logging long and writes the incoming data to this logging whenever required (e.g. if the logged object generates new data).

To implement a logging of the logged data, it is sufficient to detect the write events on the `LogBuffer` property and to save the incoming individual loggings.

Use the `Write-Callback` of the BACnet stack/API for detection purposes.

This `Callback` is always called when an object property is written to an object and if the `Callback` of this object is enabled using the following function: `BACnet.BACnetObjectBase.SetPropertyCallbackAttachment`

Entering a `Write Callback` is similar to an entering of a service hook. Only the entering function used is different.

Use the `RegisterCallback` function of the BACnetServer function block to enter up to two `CallbackFB` function blocks implementing the `IBACnetEventConsumer` interface:

```
BACnetServer.RegisterCallback(CmpBACnet.IEC_BACNET_CALLBACK_TYPE.BACNET_CALLBACK_WRITE, CallbackFB, 0);
```

To back up the loggings returned with the `EVT_BACNET_WRITEPROPERTY-CALLBACK` data upon each call, the `BACnetEventCallback` can be implemented into the function block `CallbackFB`.

When implementing, ensure that it is filtered for the correct object ID and the correct property ID for the callback data and that only logging data is written to the `Log_Buffer` of a logging object.

Implementing data reading for “Logging” objects

The `Log_Buffer` property of a “Logging” object can only be retrieved using the `ReadRange` service hook.

Thus, it is sufficient to implement the `ReadRange` service hook to implement the data retrieval function for “Logging” objects.

To implement the `ReadRange` service hook, enter a `CallbackFB` to the respective register hook as shown in the following example: chapter "Implementation example CreateObject service hook" on page 357

It can be filtered for the correct object ID and property ID within the `BACnetEventCallback` function of this `CallbackFB` and correct service data can thus be retrieved (EVT_BACNET_READRANGE).

Data previously logged in the `Write Callback` (see above) has exactly to be returned within the “ReadRange” hooks (depends on the specified object ID).

The `ReadRange` service hook is responsible to sort out data depending on the requested logging numbers and logging time stamps.

Additional notes

The procedure described before applies to all available “Logging” objects:

- `Event-Log` object
- `Trend-Log` object
- `Trend-Log-Multiple` object



The prerequisite to save data is the implementation of the `Write-Callback` service hook.

The prerequisite to retrieve data is the implementation of the `ReadRange` service hook.

Otherwise, no object actions are required with regard to the properties.

14 Service and support

Our worldwide service network provides an optimized and efficient support. Our experts offer you advice and assistance should you have any queries. You can contact us **24/7**.

Service Germany

Our technology-oriented Competence Center in Lohr, Germany, is responsible for all your service-related queries for electric drive and controls.

Contact the **Service Hotline** and **Service Helpdesk** under:

Phone:	**+49 9352 40 5060**
Fax:	**+49 9352 18 4941**
E-mail:	service.svc@boschrexroth.de
Internet:	http://www.boschrexroth.com

Additional information on service, repair (e.g. delivery addresses) and training can be found on our internet sites.

Service worldwide

Outside Germany, please contact your local service office first. For hotline numbers, refer to the sales office addresses on the internet.

Preparing information

To be able to help you more quickly and efficiently, please have the following information ready:

- Detailed description of malfunction and circumstances
- Type plate specifications of the affected products, in particular type codes and serial numbers
- Your contact data (phone and fax number as well as your e-mail address)

Index

0 … 9

0000**** - Sercos I/O status code (for S20 local)..... 236

A

About this documentation..... 1
  Validity of the documentation..... 1
Adding Sercos III modules to slave
  General..... 254
ANSI Z535.6-2006..... 2
AXIS_REF
  Application of field bus drive..... 41
AxisData[],
  Application of the field bus drive..... 43

B

BACnet
  BACnet server settings..... 338
  Basics..... 307
  Configure BACnet server..... 333
  Configure the BACnet/IP client..... 343
  Example configuration..... 338
  Features of the BACnet implementation..... 303
  Functional scope BACnet/IP client..... 305
  Functional scope BACnet/IP server and client..... 303
  Notes, implementation notes..... 345
  Overview on tabs..... 338
BACnet basics
  Device profiles..... 326
  Error codes..... 327
  General information..... 307
  Networks and topologies..... 309
  Objects and properties..... 310
  Overview..... 313
  Services..... 313
  Services - Alarm handling..... 322
  Services - Data logging(TrendLog)..... 325
  Services - Job prioritization (outputs)..... 321
  Services - Updating (COV)..... 314
  Status codes..... 330
  Terms and abbreviations..... 307
BACnet error codes..... 327
BACnet implementation notes
  Examples on how to implement “Service Indication Hooks”..... 357
  General information..... 345
  Implement “Logging” objects (EL/TL/TLM)..... 360
  Implementation example CreateObject service hook..... 357
  Implementing data reading for “Logging” objects..... 360
  Object - Property - Actions..... 346
  Object Accumulator..... 347
  Object Analag-Value..... 348
  Object Analog-Output..... 348
  Object AnalogInput..... 347
  Object Averaging..... 348
  Objects without implementation of standard actions..... 348
  Properties relating to change in value Reporting (COV)..... 347
  Properties relating to intrinsic reporting..... 346
  Property OutOfService..... 346
  Property StatusFlags..... 346
  Services with implementation of standard actions..... 348
  Services with internal implementations..... 356
  Services without implementation of standard actions..... 357
BACnet PICS
  IMPLEMENTATION CONFORMANCE STATEMENT..... 307
BACnet server
  BACnet server commissioning..... 333
  Calendar..... 341
  COV notification example..... 314
  Create BACnet servers and objects..... 333
  EDE import/export..... 343
  Notification Class..... 341
  Persistence..... 339
  PLC interface to the object properties..... 341
  Schedule..... 340
  Trend Log..... 341
BACnet services with the implementations of standard actions
  AcknowledgeAlarm Service Hook..... 348
  AddListElement service hook..... 350
  AtomicReadFile service hook..... 355
  AtomicWriteFile service hook..... 355
  CreateObject service hook..... 356
  DeleteObject service hook..... 356
  GetAlarmSummary service hook..... 349
  GetEnrollmentSummary service hook..... 349
  GetEventInformation service hook..... 354
  I-Am service hook..... 354
  I-Have service hook..... 354
  ReadProperty service hook..... 350
  ReadPropertyMultiple service hook..... 351
  ReadRange service hook..... 352
  RemoveListElement service hook..... 350
  SubscribeCOV service hook..... 349
  SubscribeCOVProperty service hook..... 353
  Who-Has service hook..... 354
  Who-Is service hook..... 355
  WriteProperty service hook..... 351
  WritePropertyMultiple service hook..... 352
BACnet status codes..... 330
Building automation

General information..... 301
Introduction and overview..... 301
System overview..... 301
Bus coupler..... 8
Bus cycle task
Bus-synchronous PLC processing..... 30
Configure bus cycle task..... 29
Description of bus cycle task..... 27
Setting “Update variables"..... 31
Bus diagnostics, ring break..... 244

C
C10A**** - Sercos I/O status code..... 236
C10C**** - Sercos I/O status code..... 236
C10E**** - Sercos I/O status code..... 236
C10F**** - Sercos I/O status code..... 236
Cabling and redundancy
Sercos III master..... 243
Change Sercos address..... 249, 252
Change Sercos slave address..... 249, 252
Command
Scan bus (EtherCAT master)..... 226
Scan bus (S20IO)..... 300
Communication and field buses
BACnet..... 303
Bus cycle task..... 27
Configure EtherNet/IP scanner..... 142
EtherCAT..... 167
EtherNet/IP..... 137
EtherNet/IP adapter (engineering port)..... 158
EtherNet/IP basics..... 141
EtherNet/IP scanner..... 142
Features of the EtherNet/IP adapter..... 137
Features of the EtherNet/IP scanner..... 137
Features of the field buses..... 13
Field bus interfaces and properties..... 11
Field bus master..... 13
Field bus slaves..... 14
Field bus-comprehensive features..... 15
General information..... 7
IP channel..... 32
Local S20 I/O..... 287
Profibus DP..... 45
Profibus DP, basics..... 46
Profinet I/O..... 75
Sercos III I/O..... 233
Terms and abbreviations..... 7
Configuration
Profinet I/O controller..... 84
Configuration adjustment
Profinet I/O..... 113
Configure control as adapter
Overview..... 152
Configuring EtherCAT master
EtherCAT Master - Tab Bus diagnostics..... 203
EtherCAT master, set device..... 199
Tab “CoE”..... 206
Tab “Diagnostics”..... 206
Tab “Distributed Clocks”..... 201
Tab “EoE”..... 205
Tab “General”..... 199
Tab “Information”..... 207
Tab “Sync unit assignment”..... 200
Configuring EtherCAT master, slaves
Add modules..... 207
Add slaves to E-BUS..... 207
Configure slaves and modules..... 207
Configure the topology..... 208
Dialog - Diagnostics..... 209
Dialog - EoE settings/IP settings..... 220
Dialog - Expert process data/download..... 216
Dialog - Expert process data/Load PDO info from device..... 216
Dialog - Expert process data/PDO content..... 215
Dialog - Expert process data/PDO list..... 214
Dialog - Expert process data/Sync manager..... 214
Dialog - Identification..... 212
Dialog - Process data expert mode/PDO assignment (SM x)..... 214
Dialog - Process data/Select the inputs..... 216
Dialog - Process data/Select the outputs..... 216
Dialog - Startup checking and timeouts..... 210
Dialog - Startup parameter/IDN..... 217
Dialog - Startup parameter/SDO..... 218
Dialog - Watchdog..... 211
Dialog -Expert process data/edit PDO list..... 214
SoE parameter editor..... 225
Tab “CoE”..... 221
Tab “Diagnostics”..... 222
Tab “EoE settings”..... 220
Tab “EoE settings”/Virtual Ethernet port..... 220
Tab “ESC register”..... 221
Tab “Expert process data”..... 214
Tab “General”..... 208
Tab “I/O mapping”..... 224
Tab “Information”..... 225
Tab “Online”..... 221
Tab “Process data”..... 216
Tab “Startup parameters”..... 216
Tab “SyncManager”..... 213
Configuring EtherNet/IP Scanner
Overview..... 142
Configuring S20 modules..... 298
General information..... 298
Configuring Sercos III
Overview..... 234

D
Device database..... 15
add devices..... 16
Device database dialog..... 15
Manage devices database..... 15

Overview..... 15
remove devices..... 18
Device editor
Profibus DP master - “Profibus DP master IEC objects”..... 55
Profibus DP master - Tab “Field bus diagnostics”..... 52
Profibus DP master - Tab “I/O mapping”..... 55
Profibus DP master - Tab “Information”..... 56
Profibus DP master - Tab “Profibus DP master parameter”..... 51
Profibus DP master - Tab “Status”..... 56
Profibus DP slave - Tab 'Profibus DP parameter'..... 64
Profibus DP slave - Tab “General”..... 62
Profibus DP slave - Tab “Information”..... 64
Profibus DP slave - Tab “Status”..... 64
Profinet I/O controller - Tab “Field bus diagnostics”..... 89
Profinet I/O controller - Tab “General”..... 84
Profinet I/O controller - Tab “I/O mapping”..... 92
Profinet I/O controller - Tab “Information”..... 93
Profinet I/O controller - Tab “IP channel settings”..... 89
Profinet I/O controller - Tab “Profinet I/O controller parameters”..... 89
Profinet I/O controller - Tab “Status”..... 92
Profinet I/O controller - Tab “Topology”..... 90
Profinet I/O device..... 98
Profinet I/O device - Tab “Field bus mapping”..... 123
Profinet I/O device - Tab “General”..... 98, 122
Profinet I/O device - Tab “I/O mapping”..... 123
Profinet I/O device - Tab “Information”..... 105, 124
Profinet I/O device - Tab “IOxS”..... 103
Profinet I/O device - Tab “Options”..... 101
Profinet I/O device - Tab “PNIO parameters”..... 104, 123
Profinet I/O device - Tab “Status”..... 105, 123
Profinet I/O device coupling modules - Tab “General”..... 125
Profinet I/O device coupling modules - Tab “Information”..... 126
Profinet I/O device coupling modules - Tab “PNIO module parameters”..... 125
Profinet I/O device coupling modules - Tab “PNIO modules I/O mapping”..... 125
Profinet I/O device coupling modules - Tab “Status”..... 126
Profinet I/O modules - Tab “I/O mapping”..... 108
Profinet I/O modules - Tab “Information”..... 109
Profinet I/O modules - Tab “PNIO parameters”..... 108
Profinet I/O modules - Tab “Status”..... 109
Profinet I/O modules- Tab “General”..... 107
Profinet I/O, controller-device coupling - Tab “General”..... 128
Profinet I/O, controller-device coupling - Tab “Information”..... 131
Profinet I/O, controller-device coupling - Tab “IOxS”..... 129
Profinet I/O, controller-device coupling - Tab “PNIO parameters”..... 130
Profinet I/O, controller-device coupling - Tab “Status”..... 130
Profinet IO controller - Tab “Overview”..... 87
Profinet IO controller - Tab “Settings”..... 88
Diagnostics
Sercos III..... 254
Disable Profinet I/O devices
UserDefs.cfg..... 94
Disable Profinet I/O modules
UserDefs.cfg..... 94
Disable S20 modules
UserDefs.cfg..... 297
Disable Sercos modules
UserDefs.cfg..... 250
Disable Sercos slaves
UserDefs.cfg..... 250
Disabled Profibus DP slave
UserDefs.cfg..... 59
Documentation
Change record..... 1

E

Enable/disable
Profibus DP slaves..... 58
Profinet I/O devices..... 93
Profinet I/O modules..... 106
S20 modules..... 296
Sercos modules..... 255
Sercos slaves..... 249
Engineering port..... 11
EtherCAT
Configuration overview..... 196
Configure EtherCAT master..... 199
EtherCAT FAQs..... 228
EtherCAT master, configure slaves..... 207
EtherCAT master, event for externally triggered PLC task..... 227
EtherCAT master, scan bus..... 226
Features of the EtherCAT master..... 167
EtherCAT basics..... 168
Acyclic communication (mailbox)..... 173
Addressing and identification procedure..... 170
Diagnostic AL status codes..... 178
Diagnostics..... 177
Diagnostics CoE SDO abort codes..... 181
Diagnostics, FoE error codes..... 184
Diagnostics, general..... 177
Diagnostics, SoE error codes..... 182
EtherCAT slave controller..... 193
EtherCAT Slave Controller/commands..... 194
EtherCAT Slave Controller/structure..... 193

Hot Connect... 177
State machine... 172
Synchronization... 185
Synchronization with DC sync... 188
Synchronization/free run... 185
Synchronization/general... 185
Synchronize with SM event... 186
SyncUnits... 176
Telegram structure... 175
Terms and abbreviations... 168
Topology... 169
Topology/cable redundancy... 170
WorkingCounter... 176
EtherCAT configuration overview
Cabling... 196
Configuration download... 198
Function package EtherCAT master... 197
Overview on configuration dialogs... 198
EtherCAT FAQs
Bus coupler does not switch to OP (notes on the AL status codes)... 231
How to configure SoE using Free Run?... 229
How to connect IndraWorks Ds via EtherCAT?... 228
How to create a startup parameterization for S20 modules?... 232
Overview... 228
EtherCAT master, configuring slaves
Dialog - I/O mapping/Reset mapping... 224
Dialog - I/O mapping/Update variables... 224
EtherNet/IP
Add scanner... 143
FAQs... 161
General information... 141
Generic Device... 150
Scanner-adapter coupling... 157
Terms and abbreviations... 141
EtherNet/IP - FAQs
Notes on the EtherNet/IP connection length (O => T / T => O-size)... 163
EtherNet/IP adapter
“Field bus mapping” tab... 156
“I/O mapping” tab... 155
Tab “Adapter settings”... 153
Tab “EtherNetIP parameters”... 156
Tab “Information”... 157, 161
Tab “IP channel settings”... 155
Tab “Status”... 157
EtherNet/IP adapter
Configure control as adapter... 152
EtherNet/IP adapter (engineering port)
Adapter modules... 161
Adapter modules - Tab “I/O mapping”... 161
Overview... 158
Tab “Information”... 160
Tab “Status”... 160
EtherNet/IP adapter (engineering)
Features... 140
EtherNet/IP adapter modules
Mapping... 161
Modify basic address... 161
EtherNet/IP configuration - FAQs
Frequent problems during EtherNet/IP configuration... 161
EtherNet/IP scanner
Adapter - Assemblies... 149
Adapter - Connections... 147
Adapter - Set target system... 146
Add adapter... 146
Configure scanner... 144
Features... 137
Scanner settings... 144
Tab “EthernetIP scanner parameters”... 144
Tab “General”... 144
Tab “I/O mapping”... 145
Tab “Information”... 146
Tab “Status”... 145
EtherNet/IP scanner, adapter
Tab “Assemblies”... 149
Tab “Connections”... 147
Tab “General”... 146
Tab “Status”... 150
EtherNet/IP. scanner
Tab “IP channel settings”... 144

F

FAQs
EtherNet/IP... 161
Features of the EtherNet/IP adapter
Features... 137
Field bus configuration
UserDefs.cfg... 26
Field bus diagnostics
Sercos III... 236
Field bus drive
Acyclic parameter communication... 43
AXIS_REF... 41
AxisData[]... 43
Bus device address... 38
Communication-required settings... 37
Create in project... 36
Cyclic data exchange... 43
Drive in original state... 37
I/O addressing... 43
Include libraries... 39
IndraDrive communication settings... 38
Length of parameter channel... 38
Operation mode settings... 39
PLC program, required declaration... 41
Process data and profile type... 38
Reference drive... 39
Signal control word... 39
Signal status word... 39
Field bus mapping
EtherNet/IP adapter... 23

Profibus DP slave..... 23
Profinet IO device..... 23
Field buses and field bus communication..... 11
Freeze mode
Profibus DP master..... 49
Profibus DP slave..... 68, 69

G
Group properties
Profibus DP master..... 49
Profibus DP slave..... 68, 69
GVL_FieldBusDrives..... 42

H
Hazard information..... 2
Helpdesk..... 363
Hotline..... 363

I
I/O address assignment of field bus drives..... 43
I/O components of the XM control..... 11
I/O mapping..... 19
Global variables..... 21
Local variables..... 21
Sercos III modules..... 262
I/O Mapping
Address blocks..... 19
IndraDrive
Bus device address..... 38
Length of parameter channel..... 38
Setting process data and profile type..... 38
Signal control word..... 39
Signal status word..... 39
IndraWorks DS
Settings of field bus drives..... 37
IndraWorks Engineering desktop
Settings of field bus drives..... 37
IO-Link..... 273
Introduction and overview..... 273
IO-Link devices
Analog values..... 273
Device exchange..... 275
Diagnostics..... 275
Initial commissioning..... 274
IO Device Description..... 274
Parameterize from PLC..... 275
Process data..... 274
Serial commissioning..... 274
IP channel
Configuration..... 32
Configuration/boundary conditions..... 33
Control system/support..... 32
General information..... 32
Use cases..... 33
IP channel use cases
IP routing via the engineering port of the control..... 33
OPC-UA communication..... 35
UDP or TCP communication with the PLC..... 35

L
Length of parameter channel, IndraDrive..... 38
Local S20 I/O
Features..... 287

M
Mapping..... 19
EtherNet/IP adapter modules..... 161
Sercos III modules..... 262
Modbus..... 263, 265
“Information”..... 271
Modbus configurator..... 264
Modbus master..... 265
Modbus slave device..... 269
Tab “I/O mapping”..... 271
Tab “Modbus - Parameters”..... 270
Tab “Modbus master - General”..... 266
Tab “Modbus serial device - General”..... 269
Tab “Modbus serial port - General”..... 265
Tab “Modbus slave - Channel”..... 267
Tab “Modbus slave - General”..... 266
Tab “Modbus slave init”..... 269
Tab “ModbusTCP master - General”..... 266
Tab “ModbusTCP slave - General”..... 267
tab “ModbusTCP slave device - General”..... 270
Tab “Status”..... 271
Modbus bus cycle task..... 263
Modbus configurator..... 264
Modbus RTU..... 263
Modbus slave device..... 269
Modify basic address
S20 modules..... 299
Modifying basic address
Sercos III modules..... 262
Module..... 8

N
Namespace..... 40

P
PLC controls drives via field bus connection..... 35
Profibus DP..... 45
Add modules to slave..... 64
Basics..... 46
Configuration overview..... 46
Features..... 45
Master..... 48
Slave (control as slave)..... 67
Terms and abbreviations..... 46
Profibus DP configuration
Profibus DP device editor..... 46
Profibus DP master
Add slaves..... 56
Adding modules to slave..... 64

Configure master..... 48
Configure slaves..... 56, 61
Features..... 45
Tab “Field bus diagnostics”..... 52
Tab “I/O mapping”..... 55
Tab “Information”..... 56
Tab “Profibus DP master IEC objects”..... 55
Tab “Profibus DP master parameter”..... 51
Tab “Status”..... 56
Profibus DP master object
DP parameter..... 49
Profibus DP master, add modules to slave
General..... 64
Profibus DP master, adding modules to slave
Tab “General”..... 65
Tab “I/O mapping”..... 66
Tab “Information”..... 67
Tab “Status”..... 67
Profibus DP slave
Add coupling modules..... 71
Add coupling modules - General..... 71
Configure..... 67
General..... 67
Master-slave coupling..... 73
Slave (control as slave)..... 45
Tab 'Profibus DP parameter'..... 64
Tab “General”..... 62
Tab “Information”..... 64
Tab “Status”..... 64
Profibus DP slave object
Profibus DP parameters..... 69
Profibus DP slave, adding modules
Tab “DP module parameters”..... 72
Tab “DP modules I/O mapping”..... 72
Tab “General”..... 72
Tab “Information”..... 73
Tab “Status”..... 73
Profibus DP slave, configure
Tab “Field bus mapping”..... 69
Tab “General”..... 68
Tab “I/O mapping”..... 68
Tab “Information”..... 71
Tab “Profibus DP configuration”..... 70
Tab “Profibus DP IEC objects”..... 68
Tab “Profibus DP parameters”..... 69
Tab “Status”..... 70
Profibus DP slaves
Enable/disable..... 58
Profinet I/O
Configuration adjustment..... 113
Configuration overview..... 80
Configure controller..... 84
Connecting to VPx controls..... 81
Connecting to XM2 controls..... 80
Connecting to XM4 controls..... 80
Controller-device coupling..... 127
Device description file..... 80
Device editor..... 84
Device exchange..... 114
Exchange device..... 115
Exchange I/O device..... 114
FAQs..... 134
GSDML file..... 80
Identify..... 80
IO-Link gateway..... 276
Media Redundancy Protocol (MRP)..... 79
Name-based device addressing..... 78
Naming I/O devices - Mainly used procedures..... 78
PLC supports topology functions and device exchange functions..... 115
Profinet I/O device editor..... 80
Scan devices..... 109
Shared Device..... 79
Terms and abbreviations..... 77
Topology..... 82
Topology and device exchange..... 114
Topology and device exchange - General information..... 114
Profinet I/O controller..... 84
Add devices..... 93
Add devices - General..... 93
Add module to device..... 105
Add modules to device - General information..... 105
Features..... 75
Scan devices..... 109
Tab “Field bus diagnostics”..... 89
Tab “General”..... 84
Tab “I/O mapping”..... 92
Tab “Information”..... 93
Tab “IP channel settings”..... 89
Tab “Overview”..... 87
Tab “Profinet I/O controller parameters”..... 89
Tab “Settings”..... 88
Tab “Status”..... 92
Tab “Topology”..... 90
Profinet I/O controller-device coupling
Tab “General”'..... 128
Tab “Information”..... 131
Tab “IOxS”..... 129
Tab “PNIO parameters”..... 130
Tab “Status”..... 130
Profinet I/O device
Add coupling modules..... 124
Configure..... 120
Control as slave..... 120
Device editor..... 98, 120
Exchange..... 114
Features..... 76
General information..... 124
Tab “Field bus mapping”..... 123
Tab “General”..... 98, 122
Tab “I/O mapping”..... 123
Tab “Information”..... 105, 124
Tab “IOxS”..... 103

Tab “IP channel settings”..... 123
Tab “Options”..... 101
Tab “PNIO parameters”..... 104, 123
Tab “Status”..... 105, 123
Profinet I/O device (control as slave)..... 120
Profinet I/O device coupling modules
Tab “General”..... 125
Tab “Information”..... 126
Tab “PNIO module parameters”..... 125
Tab “PNIO modules I/O mapping”..... 125
Tab “Status”..... 126
Profinet I/O devices
Enable/disable..... 93
Profinet I/O IO-Link gateways
Exemplary use of a Profinet I/O IO-Link gateway of the company Balluff..... 278
Exemplary use of the Rexroth Profinet I/O IO-Link gateway S67E-PN-IOL8-DI4-M12-6P..... 282
Function blocks for Profinet I/O IO-Link gateways..... 277
Introduction and overview..... 276
Profinet I/O modules
Enable/disable..... 106
Tab “General”..... 107
Tab “I/O mapping”..... 108
Tab “Information”..... 109
Tab “PNIO parameters”..... 108
Tab “Status”..... 109
Profinet I/O,
Basics..... 77
Profinet I/O, controller-device coupling
general information..... 127
Profinet I/O, FAQs
Does the IP routing not work between the engineering port of the control and the Profinet network?..... 135
Overview..... 134
PLC logger message: PN_Topology <station-name> IP x.x.x.x multiple No-of-Peers n..... 135
PNIO controller: Renaming PN device does not work (at runtime)..... 135
Profinet I/O controller: Is ring topology/ redundancy supported?..... 136
Profinet I/O device (control as slave): XMx or VPx is not detected sporadically as PN device..... 136
Where to find the Profinet device description files (GSDML) for control systems?..... 134
Profinet IO
Device names..... 78
Profinet IO controller
Devices to connect I/O modules..... 97

R

Redundancy
Sercos III master..... 243
Restore ring, Sercos..... 246
Ring break, bus diagnostics 244
RMB_PLCopenFieldBus.library
Application..... 39

S

S20 bus cycle task
Configure event-controlled PLC task as bus cycle task for the S20 local bus..... 290
S20 I/O object..... 288
S20 module parameters..... 288
Tab 'Field bus diagnostics'..... 291
Tab “I/O mapping”..... 293
Tab “Information”..... 295
Tab “S20 module parameters”..... 290
Tab “Status”..... 293
S20 module
Configure I/O..... 288
S20 modules..... 12
Add S20 modules..... 295
Enable/disable..... 296
S20 modules, scan bus..... 300
Tab “I/O mapping”..... 299
Tab “Information”..... 299
Tab “User-defined parameters”..... 298
S20 object and S20 modules
Overview..... 288
Safety instructions..... 2
Scan devices
Profinet I/O..... 109
Sercos
“Settings” tab..... 245
IO-Link gateways..... 286
Sercos I/O IO-Link gateways
Introduction and overview..... 286
Sercos I/O status code..... 238
Sercos III
Add modules to slave..... 254
Add slave..... 247
Change slave address..... 247
Commission Sercos IO devices/modules..... 251
Configure..... 234
Sercos III IO master
Features..... 233
Terms..... 233
Sercos III master
Cabling and redundancy..... 243
Field bus diagnostics..... 236
Overview on bus addresses..... 235
Sercos III module
Tab “Function groups”..... 256
Tab “General inputs and outputs”..... 257
Tab “I/O mapping”..... 262
Tab “Information”..... 262
Tab “Oversampling”..... 259
Tab “Sercos module”..... 256

Tab “Sercos modules configuration”..... 261
Tab “User-defined parameters”..... 258
Sercos III slave
Tab “General”..... 252
Tab “Information”..... 254
Tab “Sercos configuration”..... 254
Tab “Status”..... 254
Sercos IO-Link gateways
Function blocks for Sercos IO-Link gateways..... 286
Sercos modules
Enable/disable..... 255
Sercos slaves
Enable/disable..... 249
Sercos, restore ring 246
Service hotline..... 363
Signal alert symbol..... 2
Signal words..... 2
Support..... 363
Symbol configuration
Field bus mapping..... 23
Symbols..... 4
Sync mode
Profibus DP master..... 49
Profibus DP slave..... 68, 69

T

Tab
EtherNet/IP adapter - “Adapter settings”..... 153
EtherNet/IP adapter - “Information”..... 160
EtherNet/IP adapter - “Status”..... 160
EtherNet/IP adapter - EtherNetIP parameters..... 156
EtherNet/IP adapter - Field bus mapping..... 156
EtherNet/IP adapter - Information..... 157
EtherNet/IP adapter - Status..... 157
EtherNet/IP scanner - EthernetIP scanner parameters..... 144
EtherNet/IP scanner - I/O mapping..... 145
EtherNet/IP scanner - Information..... 146
EtherNet/IP scanner - IP channel settings..... 144, 155
EtherNet/IP scanner - Status..... 145
EtherNet/IP scanner, adapter - Assemblies..... 149
EtherNet/IP scanner, adapter - Connections..... 147
EtherNet/IP scanner, adapter - General..... 146
EtherNet/IP scanner, adapter - Status..... 150
IP channel settings..... 32
Profibus DP master - “Field bus diagnostics”..... 52
Profibus DP master - “I/O mapping”..... 55
Profibus DP master - “Information”..... 56
Profibus DP master - “Profibus DP master parameter”..... 51
Profibus DP master - “Status”..... 56
Profibus DP master - Tab “Profibus DP master IEC objects”..... 55
Profibus DP slave - 'Profibus DP parameter'..... 64
Profibus DP slave - “General”..... 62
Profibus DP slave - “Information”..... 64
Profibus DP slave - “Status”..... 64
Profibus DP slave, configure - Field bus mapping..... 69
Profibus DP slave, configure - General..... 68
Profibus DP slave, configure - I/O mapping..... 68
Profibus DP slave, configure - IEC objects..... 68
Profibus DP slave, configure - Information..... 71
Profibus DP slave, configure - Profibus DP configuration..... 70
Profibus DP slave, configure - Profibus DP parameters..... 69
Profibus DP slave, configure - Status..... 70
Profinet I/O controller - “Field bus diagnostics”..... 89
Profinet I/O controller - “General”..... 84
Profinet I/O controller - “Overview”..... 87
Profinet I/O controller - “Settings”..... 88
Profinet I/O controller - I/O mapping..... 92
Profinet I/O controller - Information..... 93
Profinet I/O controller - IP channel settings..... 89
Profinet I/O controller - Profinet I/O controller parameters..... 89
Profinet I/O controller - Status..... 92
Profinet I/O controller - Topology..... 90
Profinet I/O device - “General”..... 98, 122
Profinet I/O device - “IOxS”..... 103
Profinet I/O device - “PNIO parameters”..... 104, 123
Profinet I/O device - Field bus mapping..... 123
Profinet I/O device - I/O mapping..... 123
Profinet I/O device - Information..... 105, 124
Profinet I/O device - Options..... 101
Profinet I/O device - Status..... 105, 123
Profinet I/O device coupling modules - “General”..... 125
Profinet I/O device coupling modules - “PNIO module parameters”..... 125
Profinet I/O device coupling modules - “PNIO modules I/O mapping”..... 125
Profinet I/O device coupling modules - Information..... 126
Profinet I/O device coupling modules - Status..... 126
Profinet I/O modules - “PNIO parameters”..... 108
Profinet I/O modules - I/O mapping..... 108
Profinet I/O modules - Information..... 109
Profinet I/O modules - Status..... 109
Profinet I/O modules- “General”..... 107
Profinet I/O, controller-device coupling - General..... 128

Profinet I/O, controller-device coupling - IOxS..... 129
Profinet I/O, controller-device coupling - PNIO parameters..... 130
Profinet I/O, controller-device coupling - Status..... 130
Profinet I/O, controller-device coupling - Tab “Information”..... 131
S20 I/O object - I/O mapping..... 293
S20 I/O object - Information..... 295
S20 I/O object - S20 module parameters..... 290
S20 I/O object - Settings..... 288
S20 I/O object - Status..... 293
S20 I/O object - Tab “Field bus diagnostics”..... 291
S20 modules - I/O mapping..... 299
S20 modules - Information..... 299
S20 modules - User-defined parameters..... 298
Sercos III module - Function groups..... 256
Sercos III module - General inputs and outputs..... 257
Sercos III module - I/O mapping..... 262
Sercos III module - Oversampling..... 259
Sercos III module - Sercos module..... 256
Sercos III module - Sercos modules configuration..... 261
Sercos III module - User-defined parameters..... 258
Sercos III modules - Information..... 262
Sercos III slave - General..... 252
Sercos III slave - Information..... 254
Sercos III slave - Sercos configuration..... 254
Sercos III slave - Status..... 254
Tab “I/O mapping”..... 271
Tab “Information”..... 271
Tab “Modbus - Parameters”..... 270
Tab “Modbus master - General”..... 266
Tab “Modbus serial device - General”..... 269
Tab “Modbus serial port - General”..... 265
Tab “Modbus slave - Channel”..... 267
Tab “Modbus slave - General”..... 266
Tab “Modbus slave init”..... 269
Tab “ModbusTCP master - General”..... 266
Tab “ModbusTCP slave - General”..... 267
Tab “ModbusTCP slave device - General”..... 270
Tab “Status”..... 271
Task
EtherNet/IP adapter - I/O mapping..... 155
Terminal..... 8
Transmission modes
Field bus master..... 8
Field bus slaves..... 8

U

UserDefs.cfg
Disable Profibus DP slave..... 59
Disable Profinet I/O devices / Profinet I/O modules..... 94
Disable S20 modules..... 297
Disable Sercos slaves / Sercos modules..... 250
Field bus configuration..... 26

W

Warning information..... 2

Notes

Notes

rexroth
A Bosch Company

Bosch Rexroth AG
P.O. Box 13 57
97803 Lohr a.Main, Deutschland
Bgm.-Dr.-Nebel-Str. 2
97816 Lohr a.Main, Deutschland
Phone +49 9352 18 0
Fax +49 9352 18 8400
www.boschrexroth.com/electrics



R911393284

DOK-IWORKS-FB******V15-AP05-EN-P