



# Communication

For controllers that are  
programmed using SILworX



SAFETY  
NONSTOP



All HIMA products mentioned in this manual are protected by the HIMA trade-mark. Unless noted otherwise, this also applies to other manufacturers and their respective products referred to herein.

HIMax®, HIMatrix®, SILworX®, XMR® and FlexSILon® are registered trademarks of HIMA Paul Hildebrandt GmbH.

All of the instructions and technical specifications in this manual have been written with great care and effective quality assurance measures have been implemented to ensure their validity. For questions, please contact HIMA directly. HIMA appreciates any suggestion on which information should be included in the manual.

Equipment subject to change without notice. HIMA also reserves the right to modify the written material without prior notice.

For further information, refer to the HIMA DVD and our website <http://www.hima.de> and <http://www.hima.com>.

© Copyright 2013, HIMA Paul Hildebrandt GmbH

All rights reserved

## Contact

HIMA contact details:

HIMA Paul Hildebrandt GmbH

P.O. Box 1261

68777 Brühl, Germany

Phone: +49 6202 709-0

Fax: +49 6202 709-107

E-mail: [info@hima.com](mailto:info@hima.com)

| Revision index | Revisions                  | Type of change |           |
|----------------|----------------------------|----------------|-----------|
|                |                            | technical      | editorial |
| 4.01           | Added: HART over IP        | X              | X         |
| 4.02           | Revised: Chapter 4.5.3.2   | X              | X         |
| 6.00           | New edition for SILworX V6 | X              | X         |
| 6.01           | Revised: Table 33          | X              | X         |

## Table of Contents

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Introduction</b>   | <b>14</b> |
| 1.1      | Structure and Use of this Manual                                  | 14        |
| 1.2      | Target Audience   | 15        |
| 1.3      | Formatting Conventions  | 15        |
| 1.3.1    | Safety Notes  | 15        |
| 1.3.2    | Operating Tips  | 16        |
| <b>2</b> | <b>Safety</b>   | <b>17</b> |
| 2.1      | Intended use  | 17        |
| 2.2      | Residual Risk   | 17        |
| 2.3      | Safety Precautions  | 17        |
| 2.4      | Emergency Information   | 17        |
| <b>3</b> | <b>Product Description</b>  | <b>18</b> |
| 3.1      | Safety-Related Protocol (safeethernet)                            | 18        |
| 3.2      | Standard Protocols  | 19        |
| 3.3      | Redundancy  | 21        |
| 3.4      | Protocol Registration and Activation                              | 22        |
| 3.5      | Ethernet interfaces   | 23        |
| 3.5.1    | HIMax Ethernet Interfaces   | 24        |
| 3.5.2    | HIMatrix Ethernet Interfaces                                      | 24        |
| 3.5.3    | Configuring the Ethernet Interfaces                               | 25        |
| 3.5.4    | Network Ports Used for Ethernet Communication                     | 30        |
| 3.6      | Fieldbus interfaces   | 31        |
| 3.6.1    | Installation  | 31        |
| 3.6.1.1  | Part Number Structure   | 31        |
| 3.6.1.2  | HIMax COM Module Part Number                                      | 32        |
| 3.6.1.3  | HIMatrix Controller Part Number                                   | 32        |
| 3.6.2    | Registration and Activation                                       | 33        |
| 3.6.3    | Pin Assignment of the D-Sub Connectors                            | 33        |
| 3.6.3.1  | RS485 Fieldbus Interfaces for Modbus Master, Slave or ComUserTask | 33        |
| 3.6.3.2  | Fieldbus Interface PROFIBUS DP Master or Slave                    | 33        |
| 3.6.3.3  | RS232 Fieldbus Interface for ComUserTask                          | 34        |
| 3.6.3.4  | RS422 Fieldbus Interface for ComUserTask                          | 34        |
| 3.6.3.5  | SSI Fieldbus Interface  | 34        |
| 3.6.3.6  | CAN Fieldbus Interface  | 35        |
| <b>4</b> | <b>safeethernet</b>   | <b>36</b> |
| 4.1      | General information to safeethernet                               | 38        |
| 4.2      | Configuring a Redundant safeethernet Connection                   | 41        |
| 4.2.1    | Connecting Process Variables                                      | 42        |
| 4.2.2    | Verifying safeethernet Communication                              | 43        |
| 4.3      | safeethernet Editor   | 44        |
| 4.4      | Detail View of the safeethernet Editor                            | 46        |
| 4.4.1    | Tab: Peer1<->Peer2  | 46        |
| 4.4.2    | Tab: Peer1 (2)  | 46        |
| 4.4.2.1  | Tab System Variables  | 46        |

| <b>1 Introduction</b> |   | <b>Communication</b> |
|-----------------------|---|----------------------|
| 4.4.2.2               | Tab: Fragment Definitions   | 50                   |
| <b>4.5</b>            | <b>Possible safeethernet Connections</b>  | <b>51</b>            |
| 4.5.1                 | Mono safeethernet Connection (Channel 1)  | 51                   |
| 4.5.2                 | Redundant safeethernet Connection (Channel 1 and Channel 2)                           | 51                   |
| 4.5.3                 | Permitted Combinations  | 51                   |
| 4.5.3.1               | Redundant safeethernet with 2 Separated Transmission Pathsn                           | 52                   |
| 4.5.3.2               | Redundant safeethernet with HIMA Ring Master  | 53                   |
| 4.5.3.3               | Redundant safeethernet with one Transmission Path                                     | 54                   |
| 4.5.3.4               | Separating Switch Ports via VLAN  | 55                   |
| <b>4.6</b>            | <b>safeethernet Parameters</b>  | <b>56</b>            |
| 4.6.1                 | Maximum Cycle Time (Minimum Watchdog Time) of the HIMax Controller                    | 56                   |
| 4.6.2                 | Maximum Cycle Time of the HIMatrix Controller   | 57                   |
| 4.6.3                 | Receive Timeout   | 57                   |
| 4.6.4                 | Response Time   | 57                   |
| 4.6.5                 | Sync/Async  | 58                   |
| 4.6.6                 | Resend Timeout  | 58                   |
| 4.6.7                 | Acknowledge Timeout   | 59                   |
| 4.6.8                 | Production Rate   | 59                   |
| 4.6.9                 | Queue   | 59                   |
| <b>4.7</b>            | <b>Worst Case Reaction Time for safeethernet</b>                                      | <b>60</b>            |
| 4.7.1                 | Calculating the Worst Case Reaction Time of Two HIMax controllers                     | 61                   |
| 4.7.2                 | Calculating the Worst Case Reaction Time in Connection with One HIMatrix controller   | 61                   |
| 4.7.3                 | Calculating the Worst Case Reaction Time with two HIMatrix Controllers or Remote I/Os | 62                   |
| 4.7.4                 | Calculating the Worst Case Reaction Time with two HIMax and one HIMatrix Controller   | 62                   |
| 4.7.5                 | Calculating the Worst Case Reaction Time of Two HIMatrix Controllers                  | 63                   |
| 4.7.6                 | Calculating the Worst Case Reaction Time with two Remote I/Os                         | 64                   |
| 4.7.7                 | safeethernet Profile  | 65                   |
| 4.7.7.1               | safeethernet in a Noisy Network   | 65                   |
| 4.7.8                 | Profile I (Fast & Cleanroom)  | 67                   |
| 4.7.9                 | Profile II (Fast & Noisy)   | 67                   |
| 4.7.10                | Profile III (Medium & Cleanroom)  | 68                   |
| 4.7.11                | Profile IV (Medium & Noisy)   | 68                   |
| 4.7.12                | Profile V (Slow & Cleanroom)  | 69                   |
| 4.7.13                | Profile VI (Slow & Noisy)   | 69                   |
| <b>4.8</b>            | <b>Cross-Project Communication</b>  | <b>70</b>            |
| <b>4.9</b>            | <b>Cross-project communication SILworX&lt;-&gt;SILworX</b>                            | <b>71</b>            |
| 4.9.1                 | Configuration B in Project A  | 71                   |
| 4.9.1.1               | Creating the Proxy Resource B in Project A  | 71                   |
| 4.9.1.2               | Create and archive global variables for the safeethernet Connection                   | 72                   |
| 4.9.1.3               | Create a connection between the Resource A and the Proxy Resource B.                  | 73                   |
| 4.9.1.4               | Connecting Process Variables  | 73                   |
| 4.9.1.5               | Archive safeethernet connection in Project A  | 74                   |
| 4.9.2                 | Configuration A in Project B  | 74                   |
| 4.9.2.1               | Creating the Proxy Resource A in Project B  | 74                   |
| 4.9.2.2               | Create and archive global variables for the safeethernet Connection                   | 75                   |
| 4.9.2.3               | Restore safeethernet connection in Project B  | 76                   |
| <b>4.10</b>           | <b>Cross-Project Communication SILworX&lt;-&gt;ELOP II Factory</b>                    | <b>77</b>            |
| 4.10.1                | Configuration B in SILworX Project A  | 77                   |

|             |  |            |
|-------------|--|------------|
| 4.10.1.1    | Create ELOP II Factory Proxy Resource B in Project A.                | 77         |
| 4.10.1.2    | Create a connection between the Resource A and the Proxy Resource B. | 78         |
| 4.10.1.3    | Connecting Process Variables   | 79         |
| 4.10.1.4    | Exporting the Configuration File from SILworX                        | 80         |
| 4.10.2      | Configuring a HIMatrix in an ELOP II Factory Project                 | 81         |
| 4.10.2.1    | Assigning ELOP II Factory Process Signals                            | 81         |
| <b>4.11</b> | <b>Control Panel (safeethernet)</b>                                  | <b>83</b>  |
| 4.11.1      | View Box (safeethernet Connection)                                   | 83         |
| <b>4.12</b> | <b>safeethernet Reload</b>   | <b>85</b>  |
| 4.12.1      | Requirements   | 85         |
| 4.12.2      | Technical Concept  | 85         |
| 4.12.3      | Changes to the safeethernet configuration                            | 86         |
| 4.12.4      | Procedure to be observed   | 87         |
| 4.12.4.1    | Align signatures N and N+1   | 88         |
| 4.12.5      | Integrated Protective Mechanisms                                     | 89         |
| 4.12.5.1    | Automatic Test during Code Generation                                | 89         |
| 4.12.5.2    | Automatic Test during the Controller's Reload                        | 89         |
| 4.12.6      | safeethernet Version State   | 90         |
| 4.12.7      | Maximum Number of safeethernet Connections during Reload             | 90         |
| 4.12.8      | safeethernet Connection via the Communication Module                 | 90         |
| <b>5</b>    | <b>PROFINET IO</b>   | <b>91</b>  |
| <b>5.1</b>  | <b>PROFINET IO Function Blocks</b>                                   | <b>91</b>  |
| <b>5.2</b>  | <b>Controlling the Consumer/Provider Status (IOxS)</b>               | <b>92</b>  |
| 5.2.1       | Control Variables in the HIMA Controller                             | 92         |
| 5.2.2       | Control Variables in the HIMA DO Device                              | 92         |
| 5.2.3       | Control Variables in the HIMA DI Device                              | 92         |
| <b>5.3</b>  | <b>PROFIsafe</b>   | <b>93</b>  |
| 5.3.1       | PROFIsafe Control Byte and Status Byte                               | 94         |
| 5.3.2       | F_WD_Time (PROFIsafe watchdog time)                                  | 94         |
| 5.3.2.1     | Remarks about F_WD_Time (PROFIsafe watchdog time)                    | 95         |
| 5.3.3       | SFRT (Safety Function Response Time)                                 | 96         |
| 5.3.3.1     | Calculation of the SFRT between an F-Device and a HIMA F-Host        | 96         |
| 5.3.3.2     | Calculation of the SFRT using an F-Device and a HIMA F-Host          | 96         |
| 5.3.3.3     | Remark to the SFRT Calculations                                      | 96         |
| <b>5.4</b>  | <b>Requirements for Safely Operating PROFIsafe</b>                   | <b>98</b>  |
| 5.4.1       | Addressing   | 98         |
| 5.4.2       | Network Aspects  | 98         |
| <b>5.5</b>  | <b>HIMA PROFINET IO Controller and PROFIsafe F-Host</b>              | <b>100</b> |
| <b>5.6</b>  | <b>PROFINET IO/PROFIsafe Example</b>                                 | <b>101</b> |
| 5.6.1       | Creating a HIMA PROFINET IO Controller in SILworX                    | 101        |
| 5.6.1.1     | Configuring the Device in the HIMA PROFINET IO Controller            | 101        |
| 5.6.1.2     | Configuring the Device Access Point (DAP) Module                     | 103        |
| 5.6.1.3     | Configuring the PROFINET IO Device Modules                           | 103        |
| 5.6.1.4     | Configuring the PROFIsafe IO Device Modules                          | 104        |
| 5.6.1.5     | Identifying the PROFINET IO Devices within the Network               | 105        |
| <b>5.7</b>  | <b>Menu Functions for the PROFINET IO Controller</b>                 | <b>106</b> |
| 5.7.1       | Example of Structure Tree for the PROFINET IO Controller             | 106        |
| 5.7.2       | PROFINET IO Controller   | 106        |

|          |  |            |
|----------|--|------------|
| 5.7.2.1  | Tab: PROFINET IO Device (Properties)                             | 106        |
| 5.7.3    | PROFINET IO Device (within the Controller)                       | 108        |
| 5.7.3.1  | Tab: Parameters (Properties)                                     | 108        |
| 5.7.4    | DAP Module (Device Access Point Module)                          | 108        |
| 5.7.4.1  | DAP Submodule (Properties)                                       | 109        |
| 5.7.4.2  | DAP Submodule (Edit)   | 109        |
| 5.7.4.3  | Header Parameter   | 110        |
| 5.7.5    | Input/Output PROFINET IO Modules                                 | 110        |
| 5.7.6    | Input Submodule  | 111        |
| 5.7.6.1  | Submodule Input (Properties)                                     | 111        |
| 5.7.6.2  | Input Submodule (Edit)   | 112        |
| 5.7.6.3  | F Parameters of Submodule Input (for PROFIsafe Modules only)     | 115        |
| 5.7.7    | Submodule Output   | 116        |
| 5.7.7.1  | Submodule Output (Properties)                                    | 116        |
| 5.7.7.2  | Submodule Output (Edit)  | 117        |
| 5.7.7.3  | F Parameters of Submodule Output (for PROFIsafe Modules only)    | 117        |
| 5.7.8    | Submodule Inputs and Outputs                                     | 118        |
| 5.7.8.1  | Submodule Inputs and Outputs (Properties)                        | 118        |
| 5.7.8.2  | Submodule Inputs and Outputs (Edit)                              | 119        |
| 5.7.8.3  | Submodule Input/Output F Parameters (for PROFIsafe Modules only) | 119        |
| 5.7.9    | Application Relation (Properties)                                | 120        |
| 5.7.10   | Alarm CR (Properties)  | 121        |
| 5.7.11   | Input CR (Properties)  | 122        |
| 5.7.11.1 | Input CR (Edit)  | 123        |
| 5.7.11.2 | Output CR (Properties)   | 124        |
| 5.8      | <b>HIMA PROFINET IO Device</b>                                   | 125        |
| 5.9      | <b>System Requirements</b>                                       | 125        |
| 5.10     | <b>PROFINET IO/PROFIsafe Example</b>                             | 126        |
| 5.10.1   | Configuring the PROFINET IO Device in SILworX                    | 126        |
| 5.10.1.1 | Configuring the PROFINET IO Device Input Module                  | 127        |
| 5.10.1.2 | Configuring the PROFIsafe Device Output Module                   | 127        |
| 5.10.1.3 | Verifying the PROFINET IO Device Configuration                   | 128        |
| 5.11     | <b>Menu Functions for PROFINET IO Device</b>                     | 129        |
| 5.11.1   | Menu Function: Properties  | 129        |
| 5.11.2   | HIMA PROFINET IO Modules   | 130        |
| 5.11.3   | HIMA PROFIsafe Modules   | 131        |
| 5.11.4   | PROFINET IO and PROFIsafe module                                 | 132        |
| <b>6</b> | <b>PROFIBUS DP</b>   | <b>135</b> |
| 6.1      | <b>HIMA PROFIBUS DP Master</b>                                   | 136        |
| 6.1.1    | Creating a HIMA PROFIBUS DP Master                               | 136        |
| 6.2      | <b>PROFIBUS DP: Example</b>                                      | 137        |
| 6.2.1    | Configuring the PROFIBUS DP Slave                                | 137        |
| 6.2.2    | Configuring the PROFIBUS DP Master                               | 139        |
| 6.2.2.1  | Creating the HIMax PROFIBUS DP Modules                           | 140        |
| 6.2.2.2  | Configuring the Input and Output Modules                         | 141        |
| 6.2.2.3  | Creating the User Data within the PROFIBUS DP Master             | 142        |
| 6.2.2.4  | Optimizing the PROFIBUS DP parameters                            | 143        |
| 6.3      | <b>Menu Functions of the PROFIBUS DP Master</b>                  | <b>146</b> |

|             |  |            |
|-------------|--|------------|
| 6.3.1       | Edit   | 146        |
| 6.3.2       | Menu Function: Properties                                      | 147        |
| 6.3.2.1     | Tab: General   | 148        |
| 6.3.2.2     | Tab: Timings   | 149        |
| 6.3.2.3     | Tab: CPU/COM   | 150        |
| 6.3.2.4     | Tab: Other   | 151        |
| <b>6.4</b>  | <b>PROFIBUS DP Bus Access Method</b>                           | <b>152</b> |
| 6.4.1       | Master/Slave Protocol  | 152        |
| 6.4.2       | Token Protocol   | 152        |
| 6.4.3       | Target Token Rotation Time (Ttr)                               | 152        |
| 6.4.4       | Calculating the Target Token Rotation Time (Ttr)               | 153        |
| <b>6.5</b>  | <b>Isochronous PROFIBUS DP Cycle (DP V2 and Higher)</b>        | <b>155</b> |
| 6.5.1       | Isochronous Mode (DP V2 and higher)                            | 156        |
| 6.5.2       | Isochronous Sync Mode (DP V2 and higher)                       | 156        |
| 6.5.3       | Isochronous Freeze Mode (DP V2 and higher)                     | 156        |
| <b>6.6</b>  | <b>Menu Functions of the PROFIBUS DP Slave (in the Master)</b> | <b>157</b> |
| 6.6.1       | Creating a PROFIBUS DP Slave (in the Master)                   | 157        |
| 6.6.2       | Edit   | 157        |
| 6.6.3       | Properties   | 158        |
| 6.6.3.1     | Tab: Parameters  | 158        |
| 6.6.3.2     | Tab: Groups  | 159        |
| 6.6.3.3     | Tab: DP V1   | 159        |
| 6.6.3.4     | Tab: Alarms  | 160        |
| 6.6.3.5     | Tab: Data  | 160        |
| 6.6.3.6     | Tab: Model   | 161        |
| 6.6.3.7     | Tab: Features  | 161        |
| 6.6.3.8     | Tab: Baud Rates  | 162        |
| 6.6.3.9     | Tab: Acyclic   | 162        |
| <b>6.7</b>  | <b>Importing the GSD File</b>                                  | <b>163</b> |
| <b>6.8</b>  | <b>Configuring User Parameters</b>                             | <b>164</b> |
| <b>6.9</b>  | <b>PROFIBUS Function Blocks</b>                                | <b>166</b> |
| 6.9.1       | MSTAT Function Block   | 167        |
| 6.9.2       | RALRM Function Block   | 170        |
| 6.9.3       | RDIAG Function Block   | 174        |
| 6.9.4       | RDREC Function Block   | 178        |
| 6.9.5       | SLACT Function Block   | 181        |
| 6.9.6       | WRREC Function Block   | 184        |
| <b>6.10</b> | <b>PROFIBUS Auxiliary Function Blocks</b>                      | <b>187</b> |
| 6.10.1      | Auxiliary Function Block: ACTIVE                               | 187        |
| 6.10.2      | Auxiliary Function Block: ALARM                                | 188        |
| 6.10.3      | Auxiliary Function Block: DEID                                 | 189        |
| 6.10.4      | Auxiliary Function Block: ID                                   | 190        |
| 6.10.5      | Auxiliary Function Block: NSLOT                                | 191        |
| 6.10.6      | Auxiliary Function Block: SLOT                                 | 191        |
| 6.10.7      | Auxiliary Function Block: STDDIAG                              | 193        |
| <b>6.11</b> | <b>Error Codes of the Function Blocks</b>                      | <b>195</b> |
| <b>6.12</b> | <b>Control Panel (PROFIBUS DP Master)</b>                      | <b>196</b> |
| 6.12.1      | Context Menu (PROFIBUS DP Master)                              | 196        |
| 6.12.2      | Context Menu (PROFIBUS DP Slave)                               | 196        |
| 6.12.3      | View Box (PROFIBUS Master)                                     | 197        |
| 6.12.4      | PROFIBUS DP Master State                                       | 198        |

|             |   |            |
|-------------|---|------------|
| 6.12.5      | Behavior of the PROFIBUS DP Master                                  | 198        |
| 6.12.6      | Function of the FBx LED in the PROFIBUS Master                      | 199        |
| 6.12.7      | Function of the FAULT LED in the PROFIBUS Master (HIMax only)       | 199        |
| <b>6.13</b> | <b>HIMA PROFIBUS DP Slave</b>                                       | <b>200</b> |
| 6.13.1      | Creating a HIMA PROFIBUS DP Slave                                   | 200        |
| <b>6.14</b> | <b>Menu Functions of the PROFIBUS DP Slave</b>                      | <b>201</b> |
| 6.14.1      | Edit  | 201        |
| 6.14.2      | Properties  | 203        |
| <b>6.15</b> | <b>Control Panel (Profibus DP Slave)</b>                            | <b>205</b> |
| 6.15.1      | View Box (PROFIBUS DP Slave)  | 205        |
| <b>6.16</b> | <b>Function of the FBx LED in the PROFIBUS Slave</b>                | <b>206</b> |
| <b>6.17</b> | <b>Function of the FAULT LED in the PROFIBUS Slave (HIMax only)</b> | <b>206</b> |
| <b>7</b>    | <b>Modbus</b>   | <b>207</b> |
| <b>7.1</b>  | <b>RS485 Bus Topology</b>   | <b>207</b> |
| 7.1.1       | H 7506 Terminal Assignment  | 208        |
| 7.1.2       | Bus Cable   | 208        |
| 7.1.3       | Properties of the RS485 Transmission                                | 209        |
| <b>7.2</b>  | <b>HIMA Modbus Master</b>   | <b>210</b> |
| 7.2.1       | Modbus Example  | 211        |
| 7.2.1.1     | Configuring the Modbus TCP Slave                                    | 212        |
| 7.2.1.2     | Configuring the Modbus TCP Master                                   | 213        |
| 7.2.2       | Example of Alternative Register/Bit Addressing                      | 214        |
| 7.2.3       | Menu Functions of the HIMA Modbus Master                            | 216        |
| 7.2.3.1     | Edit  | 216        |
| 7.2.3.2     | Properties  | 217        |
| 7.2.4       | Modbus Function Codes of the Master                                 | 219        |
| 7.2.4.1     | Modbus Standard Function Codes                                      | 219        |
| 7.2.4.2     | HIMA-Specific Function Codes  | 220        |
| 7.2.4.3     | Format of Request and Response Header                               | 221        |
| 7.2.4.4     | Read Request Telegrams  | 222        |
| 7.2.4.5     | Read/Write Request Telegram   | 223        |
| 7.2.4.6     | Write Request Telegram  | 224        |
| 7.2.5       | Ethernet Slaves (TCP/UDP Slaves)                                    | 226        |
| 7.2.5.1     | System Variables for TCP/UDP Slaves                                 | 227        |
| 7.2.5.2     | TCP/UDP Slave Properties  | 228        |
| 7.2.6       | Modbus Gateway (TCP/UDP Gateway)                                    | 229        |
| 7.2.6.1     | Gateway Properties  | 231        |
| 7.2.6.2     | System Variables for the Gateway Slave                              | 231        |
| 7.2.6.3     | Gateway Slave Properties  | 231        |
| 7.2.7       | Serial Modbus   | 232        |
| 7.2.7.1     | Serial Modbus Properties  | 233        |
| 7.2.7.2     | System Variables for the Modbus Slave                               | 234        |
| 7.2.7.3     | Modbus Slave Properties   | 234        |
| 7.2.8       | Control Panel (Modbus Master)                                       | 234        |
| 7.2.8.1     | Context Menu (Modbus Master)  | 235        |
| 7.2.8.2     | View Box (Modbus Master)  | 235        |
| 7.2.9       | Control Panel (Modbus Master->Slave)                                | 235        |
| 7.2.10      | FBx LED Function in the Modbus Master                               | 236        |

|            |  |            |
|------------|--|------------|
| 7.2.11     | Function of the FAULT LED in the Modbus Master (HIMax only)          | 236        |
| <b>7.3</b> | <b>HIMA Modbus Slave</b>   | <b>237</b> |
| 7.3.1      | Configuring the Modbus TCP Slave                                     | 239        |
| 7.3.2      | Configuring the Redundant Modbus TCP Slave                           | 239        |
| 7.3.3      | Rules for Redundant Modbus TCP Slaves                                | 240        |
| 7.3.3.1    | Slots Allowed for the Redundant Modbus Slave COM Modules             | 240        |
| 7.3.3.2    | Redundant Modbus Slave COM Modules in Different Base Plates          | 240        |
| 7.3.4      | Menu Functions of the HIMA Modbus Slave Set                          | 241        |
| 7.3.4.1    | Modbus Slave Set Properties  | 241        |
| 7.3.4.2    | Register Variable  | 242        |
| 7.3.4.3    | Bit Variables  | 242        |
| 7.3.5      | Assigning Send/Receive Variables                                     | 243        |
| 7.3.6      | Modbus Slave Set System Variables                                    | 243        |
| 7.3.7      | Modbus Slave and Modbus Slave Redundant                              | 243        |
| 7.3.8      | Modbus Function Codes of the HIMA Modbus Slaves                      | 246        |
| 7.3.8.1    | Modbus Function Codes  | 246        |
| 7.3.9      | HIMA-Specific Function Codes   | 248        |
| 7.3.9.1    | Format of Request and Response Header                                | 249        |
| 7.3.10     | Addressing Modbus using Bit and Register                             | 250        |
| 7.3.10.1   | Register Area  | 250        |
| 7.3.10.2   | Bit Area   | 251        |
| 7.3.11     | Offsets for Alternative Modbus Addressing                            | 252        |
| 7.3.11.1   | Access to the Register Variables in the Bit Area of the Modbus Slave | 253        |
| 7.3.11.2   | Access to the Bit Variables in the Register Area of the Modbus Slave | 254        |
| 7.3.12     | Control Panel (Modbus Slave)   | 255        |
| 7.3.12.1   | Context Menu (Modbus Slave)  | 255        |
| 7.3.12.2   | View Box (Modbus Slave)  | 256        |
| 7.3.12.3   | View Box (Master Data)   | 256        |
| 7.3.13     | FBx LED Function in the Modbus Slave                                 | 257        |
| 7.3.14     | Function of the FAULT LED in the Modbus Slave (HIMax only)           | 257        |
| 7.3.15     | Error Codes of the Modbus TCP/IP Connection                          | 257        |
| <b>8</b>   | <b>Send &amp; Receive TCP</b>  | <b>258</b> |
| <b>8.1</b> | <b>System Requirements</b>   | <b>258</b> |
| 8.1.1      | Creating a S&R TCP Protocol  | 258        |
| <b>8.2</b> | <b>Example: S&amp;R TCP Configuration</b>                            | <b>259</b> |
| 8.2.1      | S&R TCP Configuration of the Siemens Controller SIMATIC 300          | 261        |
| 8.2.2      | S&R TCP Configuration of the HIMax Controller                        | 265        |
| <b>8.3</b> | <b>TCP S&amp;R Protocols Menu Functions</b>                          | <b>268</b> |
| 8.3.1      | Edit   | 268        |
| 8.3.2      | Properties   | 268        |
| <b>8.4</b> | <b>Menu Functions for TCP Connection</b>                             | <b>271</b> |
| 8.4.1      | Edit   | 271        |
| 8.4.2      | System Variables   | 271        |
| 8.4.3      | Properties   | 272        |
| <b>8.5</b> | <b>Data exchange</b>   | <b>274</b> |
| 8.5.1      | TCP Connections  | 274        |
| 8.5.2      | Cyclic Data Exchange   | 275        |
| 8.5.3      | Acyclic Data Exchange with Function Blocks                           | 275        |

|             |   |                      |
|-------------|---|----------------------|
| <b>1</b>    | <b>Introduction</b>                                     | <b>Communication</b> |
| 8.5.4       | Simultaneous Cyclic and Acyclic Data Exchange           | 275                  |
| 8.5.5       | Flow Control  | 275                  |
| <b>8.6</b>  | <b>Third-Party Systems with Pad Bytes</b>               | <b>276</b>           |
| <b>8.7</b>  | <b>S&amp;R TCP Function Blocks</b>                      | <b>277</b>           |
| 8.7.1       | TCP_Reset   | 278                  |
| 8.7.2       | TCP_Send  | 281                  |
| 8.7.3       | TCP_Receive   | 284                  |
| 8.7.4       | TCP_ReceiveLine   | 288                  |
| 8.7.5       | TCP_ReceiveVar  | 292                  |
| <b>8.8</b>  | <b>Control Panel (Send/Receive over TCP)</b>            | <b>297</b>           |
| 8.8.1       | View box for General Parameters                         | 297                  |
| 8.8.2       | View box for TCP connections                            | 297                  |
| 8.8.3       | Error Code of the TCP Connection                        | 298                  |
| 8.8.4       | Additional Error Code Table for the Function Blocks     | 299                  |
| 8.8.5       | Connection State  | 299                  |
| 8.8.6       | Partner Connection State                                | 299                  |
| <b>9</b>    | <b>SNTP Protocol</b>                                    | <b>300</b>           |
| 9.1         | <b>SNTP Client</b>                                      | <b>300</b>           |
| 9.2         | <b>SNTP Client (Server Information)</b>                 | <b>302</b>           |
| 9.3         | <b>SNTP Server</b>                                      | <b>302</b>           |
| <b>10</b>   | <b>X OPC Server</b>                                     | <b>304</b>           |
| 10.1        | <b>Equipment and System Requirements</b>                | <b>304</b>           |
| 10.2        | <b>X-OPC Server Properties</b>                          | <b>305</b>           |
| 10.3        | <b>HIMax Controller Properties for X-OPC Connection</b> | <b>306</b>           |
| 10.4        | <b>Actions Required as a Result of Changes</b>          | <b>307</b>           |
| 10.5        | <b>Forcing Global Variables on I/O Modules</b>          | <b>307</b>           |
| 10.6        | <b>Configuring an OPC Server Connection</b>             | <b>308</b>           |
| 10.6.1      | Software required:                                      | 308                  |
| 10.6.2      | Requirements for Operating the X-OPC Server:            | 308                  |
| 10.6.3      | Installation on Host PC                                 | 309                  |
| 10.6.4      | Configuring the OPC Server in SILworX                   | 312                  |
| 10.6.5      | Settings for the OPC Server in the safeethernet Editor  | 313                  |
| 10.6.6      | Configuring the X-OPC Data Access Server in SILworX     | 314                  |
| 10.6.7      | Configuring the X-OPC Alarm&Event Server in SILworX     | 316                  |
| 10.6.8      | Configuring the Fragments and Priorities in SILworX     | 319                  |
| 10.6.8.1    | Priority of Fragments for Events (Alarm&Event)          | 320                  |
| 10.6.8.2    | Priorities of Data Access Fragments                     | 321                  |
| 10.7        | <b>Alarm &amp; Event Editor</b>                         | <b>322</b>           |
| 10.7.1      | <b>Boolean Events</b>                                   | <b>322</b>           |
| 10.7.2      | <b>Scalar Events</b>                                    | <b>323</b>           |
| <b>10.8</b> | <b>Parameters for the X-OPC Server Properties</b>       | <b>326</b>           |
| 10.8.1      | OPC Server Set  | 326                  |
| 10.8.2      | OPC Server  | 330                  |
| <b>10.9</b> | <b>Uninstalling the X-OPC Server</b>                    | <b>330</b>           |

|             |  |            |
|-------------|--|------------|
| <b>11</b>   | <b>Synchronous Serial Interface</b>                                | <b>331</b> |
| 11.1        | System Requirements  | 331        |
| 11.2        | Block Diagram  | 331        |
| 11.3        | D-Sub Connectors FB1 and FB2                                       | 332        |
| 11.4        | Configuring Data Exchange between COM Module and SSI Submodule     | 332        |
| 11.5        | Configuration of the SSI   | 332        |
| 11.5.1      | Wire Lengths and Recommended Clock Rates                           | 333        |
| 11.6        | Application Notes  | 334        |
| <b>12</b>   | <b>HART</b>  | <b>335</b> |
| 12.1        | System Requirements  | 335        |
| 12.2        | Creating a HART Over IP Protocol Instance                          | 335        |
| 12.2.1      | Properties   | 336        |
| 12.3        | Structure Overview of the HART over IP Installation                | 337        |
| 12.4        | Start-up   | 338        |
| 12.4.1      | X-AI/O Module  | 338        |
| 12.4.2      | X-HART Module  | 338        |
| 12.4.3      | X-COM Module   | 338        |
| 12.4.4      | Creating the SILworX User Program, Generating and Loading the Code | 338        |
| 12.4.5      | HART/OPC Server or FDT/DTM Asset Management System                 | 339        |
| <b>12.5</b> | <b>Online View of the X-COM Module</b>                             | <b>339</b> |
| 12.5.1      | View Box (HART Protocol)   | 340        |
| 12.5.2      | Online View of the Device List                                     | 341        |
| 12.5.3      | HART Field Device Addressing                                       | 342        |
| <b>13</b>   | <b>ComUserTask</b>   | <b>343</b> |
| 13.1        | System Requirements  | 343        |
| 13.1.1      | Creating a ComUserTask   | 343        |
| <b>13.2</b> | <b>Requirements</b>  | <b>344</b> |
| <b>13.3</b> | <b>Abbreviations</b>   | <b>344</b> |
| <b>13.4</b> | <b>CUT Interface in SILworX</b>                                    | <b>345</b> |
| 13.4.1      | Schedule Interval [ms]   | 345        |
| 13.4.2      | Scheduling Preprocessing   | 345        |
| 13.4.3      | Scheduling Postprocessing  | 345        |
| 13.4.4      | STOP_INVALID_CONFIG  | 345        |
| 13.4.5      | CUT Interface Variables (CPU<->CUT)                                | 346        |
| 13.4.6      | Menu Function: Properties  | 347        |
| 13.4.7      | Menu Function: Edit  | 348        |
| 13.4.7.1    | System Variables   | 348        |
| 13.4.7.2    | Process Variables  | 349        |
| <b>13.5</b> | <b>CUT Functions</b>   | <b>350</b> |
| 13.5.1      | COM User Callback Functions  | 350        |
| 13.5.2      | COM User Library Functions   | 350        |
| 13.5.3      | Header Files   | 350        |
| 13.5.4      | Code/Data Area and Stack for CUT                                   | 350        |
| 13.5.5      | Start Function CUCB_TaskLoop                                       | 351        |
| 13.5.6      | RS485 / RS232 IF Serial Interfaces                                 | 352        |
| 13.5.6.1    | CUL_AscOpen  | 352        |
| 13.5.6.2    | CUL_AscClose   | 354        |

|             |   |            |
|-------------|---|------------|
| 13.5.6.3    | CUL_AscRcv                                    | 355        |
| 13.5.6.4    | CUCB_AscRcvReady                              | 357        |
| 13.5.6.5    | CUL_AscSend                                   | 358        |
| 13.5.6.6    | CUCB_AscSendReady                             | 358        |
| 13.5.7      | UDP/TCP Socket IF                             | 360        |
| 13.5.7.1    | CUL_SocketOpenUdpBind                         | 360        |
| 13.5.7.2    | CUL_SocketOpenUdp                             | 361        |
| 13.5.7.3    | CUL_NetMessageAlloc                           | 362        |
| 13.5.7.4    | CUL_SocketSendTo                              | 363        |
| 13.5.7.5    | CUCB_SocketUdpRcv                             | 364        |
| 13.5.7.6    | CUL_NetMessageFree                            | 365        |
| 13.5.7.7    | CUL_SocketOpenTcpServer_TCP                   | 366        |
| 13.5.7.8    | CUCB_SocketTryAccept                          | 367        |
| 13.5.7.9    | CUL_SocketAccept                              | 368        |
| 13.5.7.10   | CUL_SocketOpenTcpClient                       | 369        |
| 13.5.7.11   | CUCB_SocketConnected                          | 370        |
| 13.5.7.12   | CUL_SocketSend                                | 371        |
| 13.5.7.13   | CUCB_SocketTcpRcv                             | 372        |
| 13.5.7.14   | CUL_SocketClose                               | 373        |
| 13.5.8      | Timer-IF                                      | 374        |
| 13.5.8.1    | CUL_GetTimeStampMS                            | 374        |
| 13.5.8.2    | CUL_GetDateAndTime                            | 374        |
| 13.5.9      | Diagnosis                                     | 375        |
| <b>13.6</b> | <b>Functions for HIMatrix L3 and HM31</b>     | <b>376</b> |
| 13.6.1      | ComUserIRQTask                                | 376        |
| 13.6.1.1    | CUCB_IrqService                               | 376        |
| 13.6.1.2    | CUL_IrqServiceDisable                         | 377        |
| 13.6.1.3    | CUL_DeviceBaseAddr                            | 377        |
| 13.6.2      | NVRam IF                                      | 378        |
| 13.6.2.1    | CUL_NVRamWrite                                | 378        |
| 13.6.2.2    | CUL_NVRamRead                                 | 378        |
| 13.6.3      | Semaphore IF                                  | 379        |
| 13.6.3.1    | CUL_SemaRequest()                             | 379        |
| 13.6.3.2    | CUL_SemaRelease                               | 380        |
| 13.6.3.3    | CUL_SemaTry                                   | 381        |
| <b>13.7</b> | <b>Installing the Development Environment</b> | <b>382</b> |
| 13.7.1      | Installing the Cygwin Environment             | 382        |
| 13.7.2      | Installing the GNU Compiler                   | 384        |
| <b>13.8</b> | <b>Creating New CUT Projects</b>              | <b>386</b> |
| 13.8.1      | CUT Makefiles                                 | 387        |
| 13.8.1.1    | Makefile with ".mke" Extension                | 387        |
| 13.8.1.2    | Makefile with the 'makeinc.inc.app' Extension | 388        |
| 13.8.2      | Adapting C Source Codes                       | 389        |
| 13.8.2.1    | Configure Input and Output Variables          | 390        |
| 13.8.2.2    | Start Function in CUT                         | 390        |
| 13.8.2.3    | Example Code "example_cut.c"                  | 390        |
| 13.8.2.4    | Creating Executable Codes (ldb file)          | 392        |
| 13.8.3      | Implementing the ComUserTask in the Project   | 393        |
| 13.8.3.1    | Creating the ComUserTask                      | 393        |
| 13.8.3.2    | Loading Program Code into the Project         | 393        |
| 13.8.3.3    | Connecting Process Variables                  | 394        |

---

|             |   |            |
|-------------|---|------------|
| 13.8.3.4    | Creating the SILworX User Program                             | 395        |
| 13.8.4      | DCP: Error in loading a configuration with CUT                | 396        |
| <b>14</b>   | <b>General</b>  | <b>397</b> |
| <b>14.1</b> | <b>Configuring the Function Blocks</b>                        | <b>397</b> |
| 14.1.1      | Purchasing Function Block Libraries                           | 397        |
| 14.1.2      | Configuring the Function Blocks in the User Program           | 397        |
| 14.1.3      | Configuring the Function Blocks in the SILworX Structure Tree | 398        |
|             | <b>Appendix</b>   | <b>401</b> |
|             | <b>Glossary</b>   | <b>401</b> |
|             | <b>Index of Figures</b>                                       | <b>402</b> |
|             | <b>Index</b>  | <b>413</b> |

## 1 Introduction

The communication manual describes the characteristics and configuration of the communication protocols of the safety-related HIMax and HIMatrix controller systems with the programming tool SILworX.

Using the provided protocols, HIMax/HIMatrix controllers can be connected to one another or to controllers from other manufacturers.

Knowledge of regulations and the proper technical implementation of the instructions detailed in this manual performed by qualified personnel are prerequisites for planning, engineering, programming, installing, starting up, operating and maintaining the HIMax/HIMatrix automation devices.

HIMA will not be held liable for severe personal injuries, damage to property or the surroundings caused by any of the following: unqualified personnel working on or with the devices, deactivation or bypassing of safety functions, or failure to comply with the instructions detailed in this manual (resulting in faults or impaired safety functionality).

HIMax/HIMatrix automation devices have been developed, manufactured and tested in compliance with the pertinent safety standards and regulations. They may only be used for the intended applications under the specified environmental conditions.

### 1.1 Structure and Use of this Manual

The manual contains the following chapters:

- Introduction
- Safety
- Product Description
- Description of the available communication protocols
- General

Additionally, the following documents must be taken into account:

| Name                           | Content   | Document no. |
|--------------------------------|---|--------------|
| HIMax System Manual            | Hardware description of the HIMax system        | HI 801 001 E |
| HIMax Safety Manual            | Safety functions of the HIMax system            | HI 801 003 E |
| HIMatrix Safety manual         | Safety functions of the HIMatrix system         | HI 800 023 E |
| HIMatrix Compact System Manual | Hardware description HIMatrix Compact System    | HI 800 141 E |
| HIMatrix Modular System Manual | Hardware description HIMatrix Modular System    | HI 800 191 E |
| HIMatrix M45 System Manual     | Hardware description of the HIMatrix M45 System | HI 800 651 E |
| HIMatrix M45 Safety Manual     | Safety functions of the HIMatrix M45 system     | HI 800 653 E |
| First Steps                    | Introduction to SILworX                         | HI 801 103 E |

Table 1: Additional Valid Manuals

The latest manuals can be downloaded from the HIMA website at [www.hima.com](http://www.hima.com). The revision index on the footer can be used to compare the current version of existing manuals with the Internet edition.

## 1.2 Target Audience

This document addresses system planners, configuration engineers, programmers of automation devices and personnel authorized to implement, operate and maintain the devices and systems. Specialized knowledge of safety-related automation systems is required.

## 1.3 Formatting Conventions

To ensure improved readability and comprehensibility, the following fonts are used in this document:

|                 |  |
|-----------------|--|
| <b>Bold:</b>    | To highlight important parts<br>Names of buttons, menu functions and tabs that can be clicked and used in the programming tool.  |
| <i>Italics:</i> | For parameters and system variables  |
| Courier         | Literal user inputs  |
| RUN             | Operating state are designated by capitals   |
| Chapter 1.2.3   | Cross references are hyperlinks even though they are not particularly marked. When the cursor hovers over a hyperlink, it changes its shape.<br>Click the hyperlink to jump to the corresponding position. |

Safety notes and operating tips are particularly marked.

### 1.3.1 Safety Notes

The safety notes are represented as described below.

These notes must absolutely be observed to reduce the risk to a minimum. The content is structured as follows:

- Signal word: warning, caution, notice
- Type and source of risk
- Consequences arising from non-observance
- Risk prevention

#### **⚠ SIGNAL WORD**

Type and source of risk!

Consequences arising from non-observance

Risk prevention



The signal words have the following meanings:

- Warning indicates hazardous situation which, if not avoided, could result in death or serious injury.
- Warning indicates hazardous situation which, if not avoided, could result in minor or modest injury.
- Notice indicates a hazardous situation which, if not avoided, could result in property damage.

#### **NOTE**

Type and source of damage!

Damage prevention



**1.3.2 Operating Tips**

Additional information is structured as presented in the following example:



The text corresponding to the additional information is located here.

---

Useful tips and tricks appear as follows:



The tip text is located here.

---

## 2 Safety

All safety information, notes and instructions specified in this document must be strictly observed. The HIMax/HIMatrix controllers may only be used if all guidelines and safety instructions are adhered to.

HIMax/HIMatrix controllers are operated with SELV or PELV. No imminent risk results from the module itself. The use in Ex-Zone is permitted if additional measures are taken.

### 2.1 Intended use

For using the HIMax/HIMatrix controllers, all corresponding requirements must be met, see the specific manuals Table 1.

### 2.2 Residual Risk

No imminent risk results from a HIMax/HIMatrix system itself.

Residual risk may result from:

- Faults related to engineering
- Faults related to the user program
- Faults related to the wiring

### 2.3 Safety Precautions

Observe all local safety requirements and use the protective equipment required on site.

### 2.4 Emergency Information

A HIMax/HIMatrix system is a part of the safety equipment of a plant. If the controller fails, the system adopts the safe state.

In case of emergency, no action that may prevent the HIMax/HIMatrix systems from operating safely is permitted.

## 3 Product Description

The HIMax/HIMatrix systems can exchange process data with one another and with third-party systems via a data connection. The protocols described in this manual are configured in the SILworX programming tool.

To program HIMatrix systems in SILworX, the HIMatrix systems must be loaded with CPU OS V7 and higher and COM OS V12 and higher. A **safeethernet** connection to HIMatrix systems with previous operating system versions can be established via cross-project communication, see Chapter 4.8.

HIMA is continuously improving the operating systems. The improved versions can be loaded into the module using SILworX, see the system manuals for HIMax (HI 801 001 E) and HIMatrix (HI 801 041 E, HI 801 091 D).

The following two chapters (3.1 and 3.2) provide an overview of the safety-related protocol **safeethernet** and the available standard protocols.

### 3.1 Safety-Related Protocol (**safeethernet**)

All HIMax and HIMatrix systems can safely communicate via Ethernet in accordance with SIL 3. The **safeethernet** protocol ensures safety-related communication.

The safety-related **safeethernet** protocol is used to ensure that HIMax and HIMatrix controllers can safely exchange process data in an Ethernet network.

For more on the safety-related protocol **safeethernet**, refer to Chapter 4.

#### **⚠ WARNING**



**Manipulation of safety-related data transfer!**

**Physical injury**

**The operator is responsible for ensuring that the Ethernet used for safeethernet is sufficiently protected against manipulations (e.g., from hackers). The type and extent of the measures must be agreed upon together with the responsible test authority.**

### 3.2 Standard Protocols

Numerous proven standard protocols are available to ensure that field devices and control systems are optimally integrated in the HIMax/HIMatrix systems. In this scenario, both Ethernet and fieldbus protocols can be used.

Many communication protocols only ensure a non-safety-related data transmission. These protocols can only be used for the non-safety-related aspects of an automation task.

| Element  | HIMax  | HIMatrix F*03 und M45  | HIMatrix standard  | Description  |
|--|--|--|--|--|
| Required Module/controller   | A maximum of 20 communication modules per HIMax controller.  | F*03: Integrated communication module of the controller<br>M45: a maximum of 3 M-COM modules per controller  | Integrated communication module of the controller              | Standard protocols are run on the communication module   |
| Interfaces   | Ethernet and fieldbus interfaces of the X-COM modules.   | Ethernet interfaces and fieldbus interfaces for <ul style="list-style-type: none"> <li>▪ F*03: Integrated communication module of the controller</li> <li>▪ M45: M-COM modules.</li> </ul> | Ethernet interfaces and fieldbus interfaces of the controller. | For further information, refer to Table 8.   |
| Maximum number of standard protocols   | <ul style="list-style-type: none"> <li>▪ 20<sup>1)</sup> for each HIMax controller.</li> <li>▪ 6<sup>1)</sup> for each X-COM module</li> </ul> | <ul style="list-style-type: none"> <li>▪ 6<sup>1)</sup> for each F*03</li> <li>▪ 6<sup>1)</sup> For each M-COM module</li> <li>▪ 12<sup>1)</sup> max. for each M45 system</li> </ul>       | 4 <sup>1)</sup>  | Available standard protocols.<br>See Table 3.  |
| Process data volume <sup>2)</sup> for all standard protocols of a controller | 128 kB send<br>128 kB receive  | 64 kB send<br>64 kB receive  | 16 kB send<br>16 kB receive                                    | The maximum process data volume of the controller must not be exceeded. If it is exceeded, the controller configuration is rejected during the load process. |

<sup>1)</sup> X-OPC Server, SNTP client and SNTP server are not taken into account in this calculation.

<sup>2)</sup> The process data volume of the standard protocols contains the exchanged data and the system variables of the standard protocols.

Table 2: Standard Protocols

#### **⚠ WARNING**

##### **Use of non-safe import data**

**Non-safe data must not be used for performing the safety functions of the user program.**



The following standard protocols are available:

| Protocol                        | per HIMax module | per HIMatrix                         | Description  |
|---------------------------------|------------------|--------------------------------------|--------------|
| PROFINET IO controller          | 1                | ---                                  | Chapter 5.5  |
| PROFINET IO device              | 1                | ---                                  | Chapter 5.8  |
| PROFIBUS DP Master              | 2                | 1 for F20,<br>2 for F30, F35,<br>F60 | Chapter 6.1  |
| PROFIBUS DP slave               | 1                | 1                                    | Chapter 6.13 |
| Modbus master                   | 1                | 1                                    | Chapter 7.2  |
| Modbus slave                    | 1                | 1                                    | Chapter 7.3  |
| S&R TCP                         | 1                | 1                                    | Chapter 8    |
| HIMA X-OPC Server <sup>1)</sup> | ---              | ---                                  | Chapter 10   |
| HART                            | 2                | 2                                    | Chapter 12   |
| ComUserTask                     | 1                | 1                                    | Chapter 13   |

<sup>1)</sup> The HIMA X-OPC server is installed on a host PC and is used as a transfer interface for up to 255 HIMax/HIMatrix controllers and third-party systems that have an OPC interface.

Table 3: Available Standard Protocols



- A maximum of 1280 TCP connections per HIMax controller with 20 COM modules.
- A maximum of 64 TCP connections per HIMatrix F\*03 controller, M-COM or HIMax COM module.
- A maximum of 32 TCP connections per HIMatrix standard controller.

#### Maximum number of active protocols on one HIMatrix or one HIMax COM module

A maximum of 64 TCP sockets are available for each HIMatrix F\*03, M-COM or HIMax COM module.

Example 1:

| Protocol        | Connections                |
|-----------------|----------------------------|
| 1 Modbus master | TCP: 44 slave connections  |
| 1 Modbus slave  | TCP: 20 master connections |

Table 4: Protocols on one Communication Module

Example 2:

| Protocol             | Connections           |
|----------------------|-----------------------|
| 1 PROFIBUS DP master | 122 slave connections |
| 1 PROFIBUS DP slave  | 1 master connection   |

Table 5: Protocols on one Communication Module

### 3.3 Redundancy

The HIMax system conceptual design is characterized by high availability and also provides redundancy for the purpose of communication. A communication connection is redundant if two physical transmission paths exist and 2 HIMax modules are used to this end.

Redundant communication on HIMatrix standard is only ensured via safeethernet and one Ethernet interface.

#### Redundancy via safeethernet

Redundancy is configured in the **safeethernet** Editor by selecting the Ethernet interface for the second transmission path, see Chapter 4.5.2).

#### Redundancy via standard protocols

|  |  |
|--|--|
| PROFIBUS DP Master<br>PROFIBUS DP slave<br>PROFINET IO<br>TCP S&R<br>Modbus master | Redundancy of the corresponding standard protocol must be configured in the user program such that the user program monitors the redundant transmission paths and assigns the redundantly transmitted process data to the corresponding transmission path. |
| Modbus slave   | Redundancy for HIMax can be set in SILworX.  |

### 3.4 Protocol Registration and Activation

The protocols specified below are available for the HIMax/HIMatrix systems and can be activated as follows:

| Protocol  | Interfaces<br>HIMax<br>HIMatrix standard<br>HIMatrix F*03 | Interfaces<br>M45 | Activation        |
|---|---|-------------------|-------------------|
| HIMA safeethernet   | Ethernet  | Ethernet          | [1]               |
| SNTP server/client  | Ethernet  | Ethernet          | [1]               |
| HIMA X-OPC server (it runs on the host PC)                    | Ethernet  | Ethernet          | [4]               |
| PROFINET IO controller <sup>1)</sup>                          | Ethernet  | Ethernet          | [4]               |
| PROFINET IO device <sup>1)</sup>                              | Ethernet  | Ethernet          | [4]               |
| PROFIsafe host <sup>1)</sup>                                  | Ethernet  | Ethernet          | [4] <sup>3)</sup> |
| PROFIsafe device <sup>1)</sup>                                | Ethernet  | Ethernet          | [4] <sup>3)</sup> |
| Modbus TCP master   | Ethernet  | Ethernet          | [4]               |
| Modbus TCP slave  | Ethernet  | Ethernet          | [4]               |
| TCP Send/Receive  | Ethernet  | Ethernet          | [4]               |
| PROFIBUS DP Master  | FB1 and FB2   | <sup>4)</sup>     | [2]               |
| PROFIBUS DP slave   | FB1 or FB2  | <sup>4)</sup>     | [2]               |
| Modbus master RS485   | FB1, FB2, FB3 <sup>5)</sup>                               | <sup>4)</sup>     | [3]               |
| Modbus slave RS485  | FB1, FB2, FB3 <sup>5)</sup>                               | <sup>4)</sup>     | [3]               |
| ComUserTask RS232, RS485,<br>RS422, SSI and CAN <sup>2)</sup> | FB1, FB2, FB3 <sup>5)</sup>                               | <sup>4)</sup>     | [3]               |

<sup>1)</sup> Not available for HIMatrix standard  
<sup>2)</sup> Only available for HIMatrix F\*03 and M45  
<sup>3)</sup> additional PROFINET license needed  
<sup>4)</sup> depending on the used M-COM module, see Table 7  
<sup>5)</sup> FB3 only for HIMatrix Compact Systems

Table 6: Protocols Available for the HIMax/HIMatrix Systems

For M45 a total of four communication modules with the following communication options are available:

| M45 Modul   | FB1                | FB2   | FB3         |
|-------------|--------------------|-------|-------------|
| M-COM 010 2 | PROFIBUS DP Master | RS485 | RS422/RS485 |
| M-COM 010 3 | PROFIBUS DP slave  | RS232 | RS422/RS485 |
| M-COM 010 7 | SSI                | RS485 | RS422/RS485 |
| M-COM 010 8 | CAN-Bus            | RS485 | RS422/RS485 |

Table 7: M-COM 010 x Communication Module

[1]. The function is activated by default in all HIMax/HIMatrix systems.

[2]. The PROFIBUS master and PROFIBUS slave are activated by installing one fieldbus submodule.

[3]. Additionally, a license (software activation code) is required for the selected fieldbus protocol used with RS485 fieldbus submodules (Modbus RS485), RS232, RS422 and SSI (ComUserTask).

[4]. The software activation code with the required license can be generated on the HIMA website using the system ID (e.g., 60000) of the controller. Follow the instructions provided on the HIMA website at [www.hima.com](http://www.hima.com) -> Products -> Registration -> Communication options SILworX

- 
- i Each protocol on a COM requires its own license. Each COM can only process each protocol one time (instantiating).  
A Modbus master RS485 or Modbus slave RS485 operated on one COM through two interfaces is considered a single Modbus slave instance and therefore only requires one license.
- 

**To enter the software activation code in SILworX:**

1. In the structure tree, select Configuration, Resource, License Management.
2. Right-click **License Management**, and select New, License Key from the context menu.  
 A new license key is created.
3. Right-click **License Key** and select **Properties** from the context menu.
4. Enter the new software activation code in the Activation Code field.

- 
- i The license is intrinsically bound to this system ID. One license can only be used one time for a specific system ID. For this reason, only activate the code when the system ID has been uniquely defined.  
A software activation code may include a maximum of 32 licenses. It is also possible to specify multiple activation codes in the license management. A maximum of 64 licenses may be loaded to one controller.
- 

- 
- i Order the license on time!  
All Ethernet protocols can be tested without license for 5000 operating hours. With HIMatrix F20, F30 and F35, additionally also via RS485.  
If Ethernet protocols are operates with no valid license, the COM LED (for HIMax) or the Error LED (for HIMatrix) is lit.  
After 5000 operating hours, communication continues until the controller is stopped.  
Afterwards, the user program cannot be started without a valid license for the protocols used in the project (invalid configuration).
- 

### 3.5

### Ethernet interfaces

The Ethernet interfaces of the HIMatrix modules and HIMatrix controllers are used to communicate with external systems. Each individual Ethernet interface can simultaneously process multiple protocols.

- 
- i Process data cannot be transferred over the Ethernet interface of the HIMax X-SB 01 system bus module. The UP and DOWN Ethernet interfaces are exclusively intended for interconnecting HIMax base plates.  
If the system bus is operated in a network structure, the DIAG Ethernet interface can also be used for connecting to a HIMax base plate. Refer to the system manual (HI 801 001 E)!
- 

Each HIMax COM and CPU module and each HIMatrix controller has an Ethernet switch with a freely configurable IP address.

To transfer data, the Ethernet switch establishes a targeted connection between two communication partners. This prevents collisions and reduces the load on the network.

For targeted data forwarding, a MAC/IP address assignment table (ARP cache) is generated in which the MAC addresses are assigned to specific IP addresses. From now on, data packets are only forwarded to the IP addresses specified in the ARP cache.

- **i** Replacement of one HIMax processor or communication module or a HIMatrix controller with identical IP address.  
If a device has its *ARP Aging Time* set to 5 minutes and its *MAC Learning* set to *Conservative*, its communication partner does not adopt the new MAC address until a period of 5 to 10 minutes after the module is replaced. Until the new MAC address has been adopted, no communication is possible using the replaced device.

In addition to the configurable ARP Aging time, the user must wait at least the non-configurable MAC Aging time of the switch (approx. 10 seconds) before the replaced device is able to communicate again.

### 3.5.1 HIMax Ethernet Interfaces

| Property            | HIMax CPU module   | HIMax COM module                          |
|---------------------|--|---|
| Ports               | 4  | 4   |
| Transfer standard   | 10/100/1000 Base-T,<br>half and full duplex                        | 10/100 Base-T<br>Half and full duplex     |
| Auto negotiation    | Yes  | Yes                                       |
| Auto crossover      | Yes  | Yes                                       |
| Connection Socket   | RJ45   | RJ45                                      |
| IP address          | Freely configurable <sup>1)</sup>                                  | Freely configurable <sup>1)</sup>         |
| Subnet Mask         | Freely configurable <sup>1)</sup>                                  | Freely configurable <sup>1)</sup>         |
| Supported protocols | <b>safeethernet</b><br>Programming and debugging tool (PADT), SNTP | <b>safeethernet</b><br>Standard Protocols |

<sup>1)</sup> The general rules for assigning IP address and subnet masks must be adhered to.

Table 8: HIMax Ethernet Interfaces

### 3.5.2 HIMatrix Ethernet Interfaces

| Property            | M45 CPU module   | M45 COM module  | HIMatrix   |
|---------------------|--|---|--|
| Ports               | 4  | 4 per M-COM 01<br>(a maximum of 3 M-COM 01 modules per controller)                        | 4,<br>2 for F20 and remote I/Os  |
| Transfer standard   | 10BASE-T/<br>100BASE-Tx,<br>half and full duplex                   | 10BASE-T/<br>100BASE-Tx,<br>Half and full duplex  | 10BASE-T/<br>100BASE-Tx,<br>Half and full duplex   |
| Auto negotiation    | Yes  | Yes   | Yes  |
| Auto crossover      | Yes  | Yes   | Yes  |
| Connection Socket   | RJ45   | RJ45  | RJ45   |
| IP address          | Freely configurable <sup>1)</sup>                                  | Freely configurable <sup>1)</sup>   | Freely configurable <sup>1)</sup>  |
| Subnet Mask         | Freely configurable <sup>1)</sup>                                  | Freely configurable <sup>1)</sup>   | Freely configurable <sup>1)</sup>  |
| Supported protocols | <b>safeethernet</b><br>Programming and debugging tool (PADT), SNTP | <b>safeethernet</b><br>Standard protocols:<br>Programming and debugging tool (PADT), SNTP | <b>safeethernet</b><br>Standard Protocols<br>Programming and debugging tool (PADT), SNTP |

<sup>1)</sup> The general rules for assigning IP address and subnet masks must be adhered to.

Table 9: HIMatrix Ethernet Interfaces

### 3.5.3 Configuring the Ethernet Interfaces

The Ethernet interfaces are configured in the Detail View of the CPU or COM module.

For HIMax/HIMatrix controllers, the *Speed Mode* and *Flow Control Mode* parameters are set per default to Autoneg.



Communication loss!

With an inappropriate Ethernet parameters setting, the device might no longer be reachable.  
Reset the device!

---

#### To open the Detail View of the communication module:

1. In the structure tree, select **Configuration, Resource, Hardware**.
2. Right-click, and then click **Edit** to open the Hardware Editor.
3. Right-click **Communication Module**, and then click **Detail View** from the context menu. The Detail View opens.



The parameters set in the properties of the COM or the CPU modules are not available for the HIMax/HIMatrix system communication, until they have been re-compiled with the user program and transferred to the controller.

---

## Module

| Designation                                  | Description  |
|--|--|
| Name   | Name of the communication module.  |
| Activate Max. $\mu$ P Budget for HH Protocol | <ul style="list-style-type: none"> <li>▪ Activated: Use CPU load limit from the field <i>Max. <math>\mu</math>P Budget for HH Protocol [%]</i>.</li> <li>▪ Deactivated: Do not use the CPU Load limit for <b>safeethernet</b>.</li> </ul>  |
| Max. $\mu$ P Budget for HH Protocol [%]      | <p>Maximum CPU load of module that can be used for processing the <b>safeethernet</b> protocols.</p> <p><b>i</b> The maximum load must be distributed among all the implemented protocols that use this communication module.</p>  |
| IP Address                                   | IP address of the Ethernet interface.  |
| Subnet Mask                                  | 32 bit address mask to split up the IP address in network and host address.  |
| Standard Interface                           | Activated: the interface is used as standard interface for the system login.<br>Default setting: Deactivated   |
| Default Gateway                              | IP address of the default gateway.   |
| ARP Aging Time [s]                           | <p>A processor or COM module stores the MAC addresses of the communication partners in a MAC/IP address assignment table (ARP cache).</p> <p>If in a period of <math>1x\dots2x</math> <i>ARP Aging Time</i> ...</p> <ul style="list-style-type: none"> <li>▪ ... messages of the communication are received, the MAC address remains stored in the ARP cache.</li> <li>▪ ... no messages of the communication partner are received, the MAC address is erased from the ARP cache.</li> </ul> <p>The typical value for the <i>ARP Aging Time</i> in a local network ranges from 5...300 s.</p> <p>The user cannot read the contents of the ARP cache.</p> <p>Range of values: 1...3600 s<br/>Default value: 60 s</p> <p><b>Note:</b><br/>If routers or gateways are used, the user must adjust (increase) the <i>ARP Aging Time</i> due to the additional time required for two-way transmission.<br/>If the <i>ARP Aging Time</i> is too low, the MAC address of the communication partner is erased from the ARP cache, the communication is delayed or interrupted. For an efficient performance, the ARP aging time value must be less than the receive timeout set for the protocols in use.</p> |

| Designation   | Description  |
|---------------|--|
| MAC Learning  | <p>MAC Learning and <i>ARP Aging Time</i> are used to set how quick the Ethernet switch should learn the MAC address.</p> <p>The following settings are possible:</p> <ul style="list-style-type: none"> <li>▪ <b>Conservative (recommended):</b><br/>If the ARP cache already contains MAC addresses of communication partners, these are locked and cannot be replaced by other MAC addresses for at least one <i>ARP Aging Time</i> and a maximum of two <i>ARP Aging Time</i> periods. This ensures that data packets cannot be intentionally or unintentionally forwarded to external network participants (ARP spoofing).</li> <li>▪ <b>Tolerant:</b><br/>When a message is received, the IP address contained in the message is compared to the data in the ARP cache and the MAC address stored in the ARP cache is immediately overwritten with the MAC address from the message. The <i>Tolerant</i> setting must be used if the availability of communication is more important than the authorized access to the controller.</li> </ul> <p>Default setting: Conservative</p> |
| IP Forwarding | <p>Allow a system bus module to operate as router and to forward data packets to other network nodes.</p> <p>Default setting: Deactivated</p>  |
| ICMP Mode     | <p>The Internet Control Message Protocol (ICMP) allows the higher protocol layers to detect error states on the network layer and optimize the transmission of data packets.</p> <p>Message types of Internet Control Message Protocol (ICMP) supported by the processor module:</p> <ul style="list-style-type: none"> <li>▪ <b>No ICMP Responses</b><br/>All the ICMP commands are deactivated. This ensures a high degree of safety against potential sabotage that might occur over the network.</li> <li>▪ <b>Echo Response</b><br/>If Echo Response is activated, the node responds to a ping command. It is thus possible to determine if a node can be reached. Safety is still high.</li> <li>▪ <b>Host Unreachable</b><br/>Not important for the user. Only used for testing at the manufacturer's facility.</li> <li>▪ <b>All Implemented ICMP Responses</b><br/>All ICMP commands are activated. This allows a more detailed diagnosis of network malfunctions.</li> </ul> <p>Default setting: Echo Response</p>   |

Table 10: Configuration Parameters

## Routings

The **Routings** tab contains the routing table. This table is empty if the module is new. A maximum of 8 routing entries are possible.

| Designation | Description  |
|-------------|--|
| Name        | Denomination of the routing settings   |
| IP Address  | Target IP address of the communication partner (with direct host routing) or network address (with subnet routing).<br>Range of values: 0.0.0.0 ... 255.255.255.255<br>Default value: 0.0.0.0                                |
| Subnet Mask | Define the target address range for a routing entry.<br>255.255.255.255 (with direct host routing) or subnet mask of the addressed subnet.<br>Range of values: 0.0.0.0 ... 255.255.255.255<br>Default value: 255.255.255.255 |
| Gateway     | IP address of the gateway to the addressed network.<br>Range of values: 0.0.0.0 ... 255.255.255.255<br>Default value: 0.0.0.1  |

Table 11: Routing Parameters

## Ethernet switch

| Designation                    | Description   |
|--------------------------------|---|
| Name                           | Port number as printed on the housing; per port, only one configuration may exist.<br>Range of values: 1...4  |
| Speed [Mbit/s]                 | 10 Mbit/s: Data rate 10 Mbit/s<br>100 Mbit/s: Data rate 100 Mbit/s<br>1000 Mbit/s: Data rate 1000 Mbit/s (processor module)<br>Autoneg (10/100/1000): Automatic baud rate setting<br>Default value: Autoneg   |
| Flow Control                   | Full duplex: Simultaneous communication in both directions<br>Half duplex: Communication in one direction<br>Autoneg: Automatic communication control<br>Default value: Autoneg   |
| Autoneg also with fixed values | The <i>Advertising</i> function (forwarding the speed and flow control properties) is also performed if the parameters <i>Speed</i> and <i>Flow Control</i> have fixed values. This allows other devices with ports set to <i>Autoneg</i> to recognize the HIMax port settings. |
| Limit                          | Limit the inbound multicast and/or broadcast packets.<br>Off: No limitation<br>Broadcast: Limit broadcast packets (128 kbit/s)<br>Multicast and Broadcast: Limit multicast and broadcast packets (1024 kbit/s)<br>Default value: Broadcast                                      |

Table 12: Ethernet Switch Parameters

## VLAN (Port-Based VLAN)

For configuring the use of port-based VLAN.

- i** Should VLAN be supported, port-based VLAN should be off to enable each port to communicate with the other switch ports.

For each port on one switch, the user can define which other ports of the switch received Ethernet frames may be sent to.

The table in the VLAN tab contains entries through which the connection between two ports can be set as *active* or *inactive*.

Default setting: All connection between ports *active*

## LLDP

With LLDP (Link Layer Discovery Protocol), information such as MAC address, device name, port number is sent per multicast in periodic intervals via the own device and is received from the neighboring devices.

LLDP uses the following values depending on whether PROFINET is configured on the communication module:

| PROFINET on the COM module | ChassisID   | TTL (Time to Live) |
|----------------------------|-------------|--------------------|
| Used                       | Device name | 20 s               |
| Not used                   | MAC address | 120 s              |

Table 13: Values for LLDP

The processor and communication modules support LLDP on the Eth1, Eth2, Eth3 and Eth4 ports

The following parameters define how a given port should work:

- |              |   |
|--------------|---|
| Off          | LLDP is disabled on this port.  |
| Send         | LLDP sends LLDP Ethernet frames, received LLDP Ethernet frames are deleted without being processed. |
| Receive      | LLDP sends no LLDP Ethernet frames, but received LLDP Ethernet frames are processed.                |
| Send/Receive | LLDP sends and processes received LLDP Ethernet frames.   |

Default setting: OFF

### Mirroring

Mirroring is used to configure whether the module should duplicate Ethernet packets on a given port such that they can be read from a device connected to that port, e.g., for test purposes.

The following parameters define how a given port should work:

Off This port does not participate to the mirroring process.

Egress: Outgoing data of this port are duplicated.

Ingress: Incoming data of this port are duplicated.

Ingress/Egress: Incoming and outgoing data of this port are duplicated.

Dest Port: This port is used to send duplicated data.

Default setting: OFF

## 3.5.4 Network Ports Used for Ethernet Communication

### UDP Ports / Use

|       |  |
|-------|--|
| 123   | SNTP (time synchronization between PES and remote I/O, PES and external devices) |
| 502   | Modbus slave (can be modified by the user)                                       |
| 6010  | safeethernet and OPC   |
| 8001  | Configuring the remote I/O using the PES   |
| 8000  | Programming and operation with SILworX   |
| 34964 | PROFINET endpoint mapper (required for establishing the connection)              |
| 49152 | PROFINET RPC server  |
| 49153 | PROFINET RPC client  |

### TCP Ports / Use

|     |  |
|-----|--|
| 502 | Modbus slave (can be modified by the user) |
| Xxx | TCP SR assigned by the user                |



All ports listed above are destination ports.

The ComUserTask can use any port if it is not already used by another protocol.

---

### 3.6 Fieldbus interfaces

The fieldbus submodules allow communication via the fieldbus interfaces of the HIMax X-COM 01 as well as the HIMatrix controllers F20, F30, F35 and the F60 CPU 01. The fieldbus submodules are optional and must be mounted by the manufacturer.

Factory-made, the fieldbus interface FB3 of HIMatrix controllers is equipped with RS485 for Modbus (master or slave) or ComUserTask.

No safety-related communication can be ensured with the available fieldbus protocols.

The following table provides an overview of the available fieldbus submodules:

| Designation     | Description  |
|-----------------|--|
| PROFIBUS master | Fieldbus submodule PROFIBUS DP master                                      |
| PROFIBUS slave  | Fieldbus submodule PROFIBUS DP slave                                       |
| RS485 module    | RS485 fieldbus submodule for Modbus (master or slave) or ComUserTask       |
| RS232 module    | RS232 fieldbus submodule for ComUserTask                                   |
| RS422 module    | RS422 fieldbus submodule for ComUserTask                                   |
| SSI module      | SSI fieldbus submodule for ComUserTask                                     |
| CAN module      | CAN fieldbus submodule for ComUserTask<br>Only available for HIMatrix F*03 |

Table 14: Available Fieldbus Submodules

#### 3.6.1 Installation

The fieldbus submodules are optional and must be mounted by the manufacturer. The definition is made with the order by the part number. Additionally, the protocols used must be activated.

##### **⚠ CAUTION**



**Improper opening of the COM module oder HIMatrix**

**Damage to COM module or HIMatrix**

**Only HIMA is authorized to retrofit the fieldbus submodules.**

##### 3.6.1.1 Part Number Structure

The following sections present how the part number for the HIMax X-COM 01 or a HIMatrix controller changes if fieldbus interfaces are used.

Numbers are allocated to the fieldbus to create the part numbers, see Table 15.

| Options for FB1 and FB2 | Description   |
|-------------------------|---|
| 0                       | No fieldbus submodule inserted                          |
| 1                       | RS485 for Modbus (master or slave) or ComUserTask       |
| 2                       | PROFIBUS DP master                                      |
| 3                       | PROFIBUS DP slave                                       |
| 4                       | INTERBUS master (only with HIMatrix in ELOP II Factory) |
| 5                       | RS232 for ComUserTask                                   |
| 6                       | RS422 for ComUserTask                                   |
| 7                       | SSI for ComUserTask                                     |
| 8                       | CAN for ComUserTask (only with HIMatrix F*03)           |

Table 15: Options for Fieldbus Interfaces FB1 and FB2

### 3.6.1.2 HIMax COM Module Part Number

The COM module forms a functional unit with the X-CB 001 02 connector board. Note that the connector board must be separately purchased.

When the module is equipped with one or multiple fieldbus submodules, the part number and also the module name changes from X-COM 01 to X-COM 010 XY.

The following table specifies the available components:

| Designation                | Description                                      |
|----------------------------|--|
| X-COM 01                   | Communication module without fieldbus submodules |
| X-COM 010 XY <sup>1)</sup> | Communication module with fieldbus submodule     |
| X-CB 001 02                | Connector board                                  |

<sup>1)</sup> X: Option for fieldbus interface FB1 according to Table 15  
Y: Option for fieldbus interface FB2 according to Table 15

Table 16: Available HIMax Components

The following table shows examples for part numbers and names:

| Part no.   | Designation  | Fieldbus submodule 1 (FB1)     | Fieldbus submodule 2 (FB2)     |
|------------|--------------|--------------------------------|--------------------------------|
| 98 5260021 | X-COM 010 21 | PROFIBUS master (max. 12 Mbit) | RS485                          |
| 98 5260023 | X-COM 010 23 | PROFIBUS master (max. 12 Mbit) | PROFIBUS slave (max. 1.5 Mbit) |
| 98 5260011 | X-COM 010 11 | RS485                          | RS485                          |
| 98 5260000 | X-COM 01     | ---                            | ---                            |

Table 17: Examples of COM Module Part Numbers and Names



HIMA recommends operating the PROFIBUS DP using the FB1 fieldbus interface (maximum transfer rate 12 Mbit). The maximum transfer rate permitted for the FB2 fieldbus interface is 1.5 Mbit.

The designation and part number (part no.) are printed on the type label of the module.

For further information, refer to the SILworX Communication Manual (HI 801 101 E).

### 3.6.1.3 HIMatrix Controller Part Number

The HIMatrix controllers can be equipped with fieldbus submodules in accordance with the following table:

| Controller | FB1               | FB2               | FB3              |
|------------|-------------------|-------------------|------------------|
| F20        | Free configurable | Integrated RS485  | ---              |
| F30        | Free configurable | Free configurable | Integrated RS485 |
| F35        | Free configurable | Free configurable | Integrated RS485 |
| F60        | Free configurable | Free configurable | ---              |

Table 18: Equipment of HIMatrix Controller with Fieldbus Submodules

With the selection of the appropriate fieldbus submodule the part number changes:

98 22xy...

X: Option for fieldbus interface FB1 according to Table 15

Y: Option for fieldbus interface FB2 according to Table 15

The following table shows examples for part numbers:

| Part no.   | Controller         | FB1             | FB2             |
|------------|--------------------|-----------------|-----------------|
| 98 2210417 | F20 01             | RS485           | ---             |
| 98 2232415 | F30 01             | PROFIBUS Slave  | PROFIBUS Master |
| 98 2232472 | F30 01 SILworX     | PROFIBUS Slave  | PROFIBUS Master |
| 98 2242416 | F35 01             | INTERBUS Master | PROFIBUS Master |
| 98 2232497 | F35 03 SILworX     | PROFIBUS Slave  | PROFIBUS Master |
| 98 2212126 | F60 CPU 01         | RS485           | PROFIBUS Master |
| 98 2212137 | F60 CPU 01 SILworX | RS485           | PROFIBUS Master |
| 98 0012139 | F60 CPU 03 SILworX | RS485           | PROFIBUS Master |

Table 19: Examples of HIMatrix Controller Part Numbers

### 3.6.2 Registration and Activation

The communication options are activated in accordance with the fieldbus submodules, see Chapter 3.4.

### 3.6.3 Pin Assignment of the D-Sub Connectors

The following tables describe the Pin assignments of the fieldbus interfaces.

#### 3.6.3.1 RS485 Fieldbus Interfaces for Modbus Master, Slave or ComUserTask

| Connection                     | Signal    | Function                      |
|--------------------------------|-----------|-------------------------------|
| 1                              | - - -     | - - -                         |
| 2                              | RP1)      | 5 V decoupled with diodes     |
| 3                              | RxD/TxD-A | Receive/send data A           |
| 4                              | CNTR-A    | Control signal A              |
| 5                              | DGND      | Data reference potential      |
| 6                              | VP1)      | 5 V, plus pole supply voltage |
| 7                              | - - -     | - - -                         |
| 8                              | RxD/TxD-B | Receive/send data B           |
| 9                              | CNTR-B    | Control signal B              |
| 1) HIMatrix M45 not supported! |           |                               |

Table 20: Pin Assignment of D-Sub Connectors for RS485

#### 3.6.3.2 Fieldbus Interface PROFIBUS DP Master or Slave

| Connection | Signal    | Function                      |
|------------|-----------|-------------------------------|
| 1          | - - -     |                               |
| 2          | - - -     |                               |
| 3          | RxD/TxD-A | Receive/send data A           |
| 4          | RTS       | Control signal                |
| 5          | DGND      | Data reference potential      |
| 6          | VP        | 5 V, plus pole supply voltage |
| 7          | - - -     | - - -                         |
| 8          | RxD/TxD-B | Receive/send data B           |
| 9          | - - -     | - - -                         |

Table 21: Pin Assignment of D-sub Connectors for PROFIBUS DP

### 3.6.3.3 RS232 Fieldbus Interface for ComUserTask

| Connection                     | Signal | Function                 |
|--------------------------------|--------|--------------------------|
| 1                              | ---    | ---                      |
| 2                              | TxD    | Send data                |
| 3                              | RxD    | Receive data             |
| 4                              | ---    | ---                      |
| 5                              | DGND   | Data reference potential |
| 6                              | ---    | ---                      |
| 7                              | RTS1)  | Request to send          |
| 8                              | ---    | ---                      |
| 9                              | ---    | ---                      |
| 1) HIMatrix M45 not supported! |        |                          |

Table 22: Pin Assignment of D-Sub Connectors for RS232

### 3.6.3.4 RS422 Fieldbus Interface for ComUserTask

| Connection                     | Signal | Function                   |
|--------------------------------|--------|----------------------------|
| 1                              | ---    | ---                        |
| 2                              | RP1)   | +5 V decoupled with diodes |
| 3                              | RxA    | Receive data A             |
| 4                              | TxA    | Send data A                |
| 5                              | DGND   | Data reference potential   |
| 6                              | VP1)   | +5 V supply voltage        |
| 7                              | ---    | ---                        |
| 8                              | RxB    | Receive data B             |
| 9                              | TxB    | Send data B                |
| 1) HIMatrix M45 not supported! |        |                            |

Table 23: Pin Assignment of D-Sub Connectors for RS422

### 3.6.3.5 SSI Fieldbus Interface

| Connection | Signal   | Function  |
|------------|----------|---|
| 1          | D2+      | Data input channel 2+                           |
| 2          | D1-      | Data input channel 1-                           |
| 3          | CL2+/D3+ | Clock output channel 2+ or data input channel 3 |
| 4          | CL1+     | Clock output channel 1+                         |
| 5          | GND      | Ground  |
| 6          | D1+      | Data input channel 1+                           |
| 7          | D2-      | Data input channel 2-                           |
| 8          | CL2-/D3- | Clock output channel 2- or data input channel 3 |
| 9          | CL1-     | Clock output channel 1-                         |

Table 24: Pin Assignment of D-Sub Connectors for SSI

### 3.6.3.6 CAN Fieldbus Interface

| Connection | Signal | Function |
|------------|--------|----------|
| 1          | ---    | ---      |
| 2          | CAN-L  | CAN-Low  |
| 3          | GND    | Ground   |
| 4          | ---    | ---      |
| 5          | ---    | ---      |
| 6          | ---    | ---      |
| 7          | CAN-H  | CAN-High |
| 8          | ---    | ---      |
| 9          | ---    | ---      |

Table 25: Pin Assignment of D-Sub Connectors for CAN

## 4 safeethernet

All HIMax and HIMatrix systems can safely communicate via **safeethernet** in accordance with SIL 3 (HIMax 1 Gbit/s, HIMatrix 100 Mbit/s). For railway applications, **safeethernet** for HIMatrix is approved for use up to SIL 4 in accordance with CENELEC.

- i** The **safeethernet** protocol meets all requirements for safety protocols in accordance with IEC 61508-2 Edition 2 and IEC 61784-3. The TÜV has tested this feature and verified the protocols used in the controllers. The TÜV has tested this feature and verified the protocols used in the controllers.

With a bit error probability ( $P_e$ ) of  $10^{-2}$  of the transmission medium, e.g., due to a disturbed network, residual data error rate  $\lambda_{SL}$  ( $P_e$ ) of an safety-related message is less than,

- 1% von SIL 3
- 1% of SIL 4 in accordance with CENELEC for railway applications.

During a safety function 1000 **safeethernet** communications and 1000 Ethernet-Switches/Router/Gateways may be used.

The corresponding Ethernet interfaces of the HIMax CPU, HIMax COM and HIMatrix controllers can be also used simultaneously for other protocols.

Various Ethernet network topologies can be used to ensure **safeethernet** communication between controllers. To this end, so called **safeethernet** profile suitable for the Ethernet network in use can be selected in SILworX to increase the data transmission speed and efficiency. These **safeethernet** profiles ensure **safeethernet** communication, even if users are not experts of network configuration.

### Equipment and system requirements

|                 |   |
|-----------------|---|
| HIMA controller | HIMax with processor module<br>HIMatrix Standard, F*03, M45         |
| Activation      | The function is activated by default in all HIMax/HIMatrix systems. |

**safeethernet (Properties):**

| Element                                 | HIMax  | HIMatrix F*03 und M45  | HIMatrix standard  | Description  |
|---|--|--|--|--|
| Required Module/controller              | per HIMax<br>1 up to max. 4 CPU modules            | For F*03<br>Integrated CPU module of the controller<br>For M45<br>CPU module and optionally up to 3 M-COM 01 modules | Integrated CPU module of controller                              | safeethernet is run on the safety-related CPU module.  |
| Ethernet Interfaces:                    | CPU module<br>1 GBit/s<br>COM module<br>100 Mbit/s | CPU module:<br>100 Mbit/s<br>COM module<br>100 Mbit/s  | CPU module:<br>100 Mbit/s  | The Ethernet interfaces in use can simultaneously be used for additional protocols.  |
| Connections:                            | for each HIMax 255                                 | 128  | 64   | safeethernet connections   |
| Connections between two controllers     | max. 1 up to V6<br>max. 64 with V6 and higher      | max. 1 up to V10<br>max. 64 with V10 and higher  | max. 1   | safeethernet connections   |
| Redundant Connections                   | for each HIMax 255                                 | 128  | 64<br>(Redundancy via one Transmission Path)<br>Remote I/O: n. a | Two-channel operation<br>Redundant safeethernet connections between HIMax and HIMatrix controllers can be configured in the safeethernet Editor. |
| Process data volume for each connection | 1100 bytes   | 1100 bytes   | 900 bytes  | for each safeethernet connection.  |
| n.a.: not applicable                    |  |  |  |  |

Table 26: Safety-Related Protocol (safeethernet)



Cross-project communication!

safeethernet connections to a resource in another project or a HIMatrix controller with CPU operating system up to V7 and COM operating system up to V12 can be configured in SILworX, see Chapter 4.8.

#### 4.1 General information to safeethernet

Requirements as determinism, reliability, interchangeability, extendibility and above all safety, are central issues within the process and automation technology.

**safeethernet** is a transfer protocol for transmitting safety-related data up to SIL 3 when Ethernet technology is used.

**safeethernet** implements mechanisms that can detect and safely respond to the following faults:

- Data signature
  - Corruption
  - Fragmentation
  - Masquerading
- Connection authentication
  - Addressing
  - Masquerading
  - Diverging configuration of the communication partners
  - Insertions
  - Unintentional restart
- Sequence number
  - Delay
  - Queueing
  - Loss
  - Sequence
  - Unintentional resending
  - Insertion of individual messages and sequences
  - Unintentional restart
- Time expectations
  - Loss
  - Delay
  - Queueing
- Safe addressing
  - Masquerading
  - Addressing

safeethernet is based on the IEEE 802.3 standard.

The standard Ethernet protocol frame is used to transmit safety-related data.

**safeethernet** uses "unsafe data transfer channels" (Ethernet) in accordance with the black channel approach and monitors them on the transmitter and receiver side by using safety-related protocol mechanism. This allows the users to use normal Ethernet network components such as switches, routers and wireless LAN devices within a safety-related network.

**safeethernet** uses the abilities of standard Ethernet so that security and real time ability are made possible. A special protocol mechanism ensures a deterministic behavior even if faults occur or new communication subscribers join the network. The system automatically integrates new components in the running system. All network components can be replaced during operation. Transmission times can be clearly defined using switches. If properly configured, Ethernet is thus real-time capable.

The possible transfer rate of up to 1 Gbit/s offers automation applications sufficient transmission capacity for safety-related data. Transmission media such as copper lines and fiber optic cables can be used.

**safeethernet** can use the existing company Ethernet network to transmit safe data as well as non-safe data on the Ethernet network.

- i The network may be shared with other subscribers if sufficient transfer capacity is available.  
The plant manufacturer and the operator are responsible for ensuring that the Ethernet network used for safeethernet is sufficiently protected against manipulations (e.g., from hackers).  
The type and extent of the measures must be agreed upon together with the responsible test authority.
- 

safeethernet allows the user to create flexible system structures for decentralize automation with defined response times. Depending on the requirements, the intelligence can be distributed to the network subscribers in a centralized or decentralized manner.

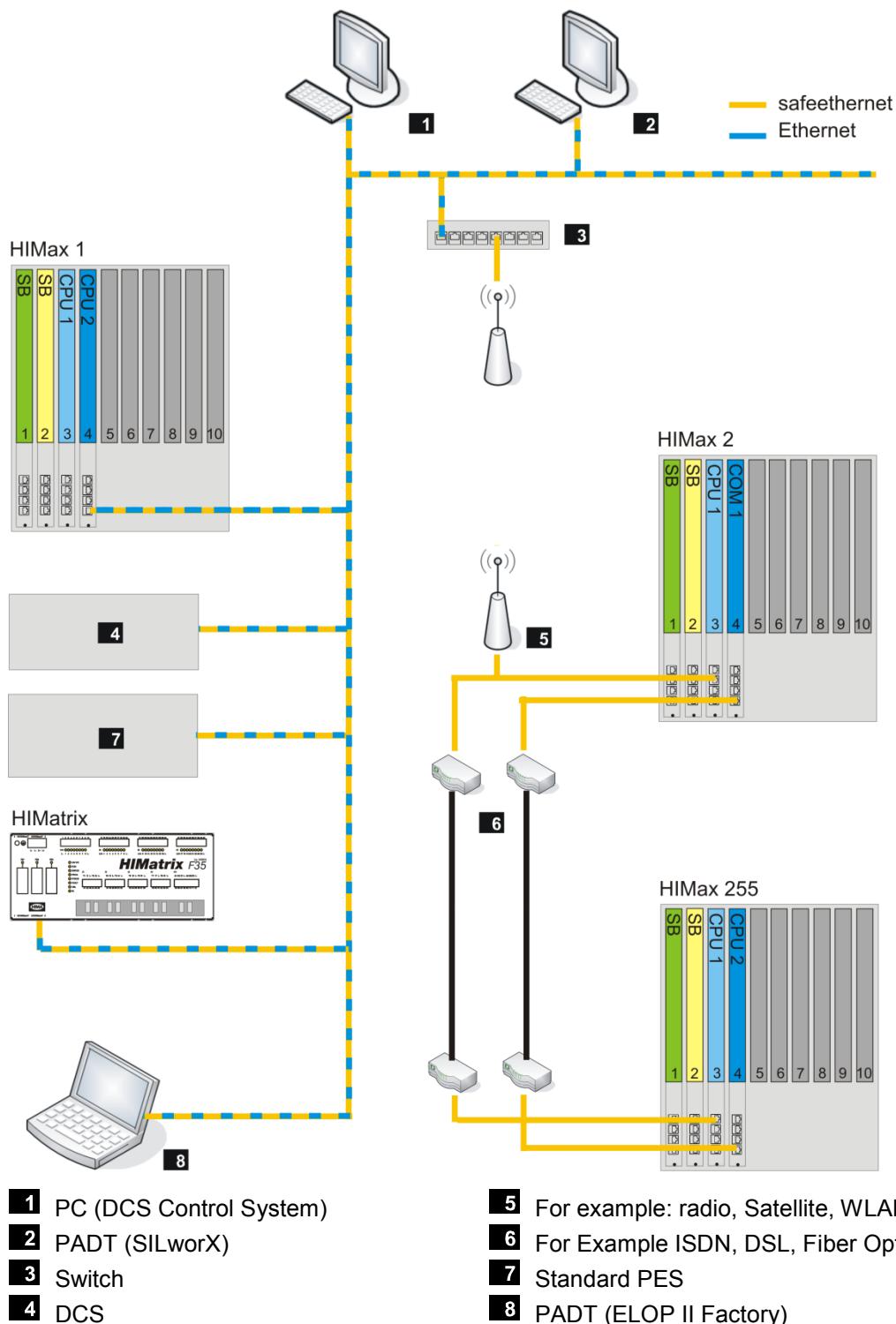


Figure 1: System Structures



Unintentional transition to the safe state possible!

In accordance with the generally accepted regulations for developing Ethernet networks, no network loop may occur. Data packets may only reach a controller over a single path.

## 4.2 Configuring a Redundant safeethernet Connection

This example shows how to configure a redundant HIMax/HIMax safeethernet connection.

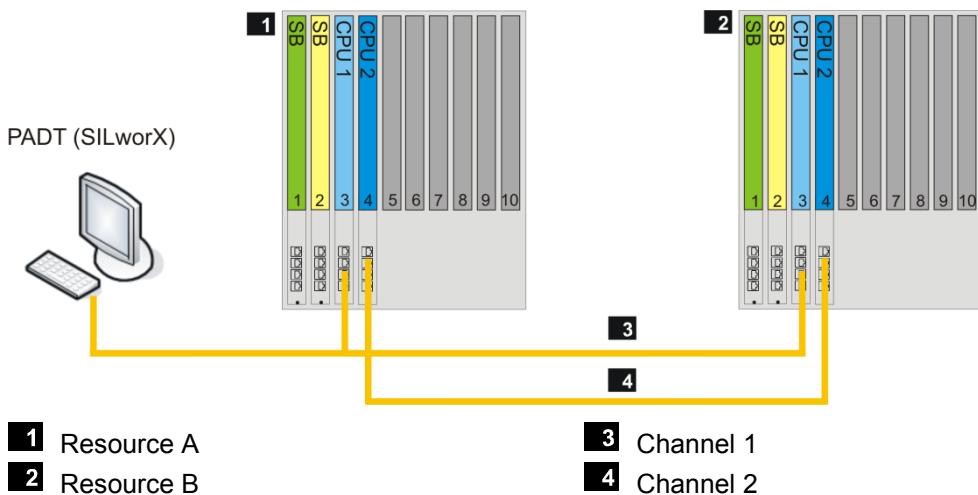


Figure 2: Structure for Configuring a Redundant Connection



For redundant safeethernet connections, HIMA recommends to implement the two transmission paths (channel 1 and channel 2) via two Ethernet networks completely separated from one another. In doing so, the bandwidth and the delay on the respective transmission paths must be nearly identical.

### Establishing the safeethernet Connection

In the safeethernet Editor, create a safeethernet connection between the resource A and the target resource.

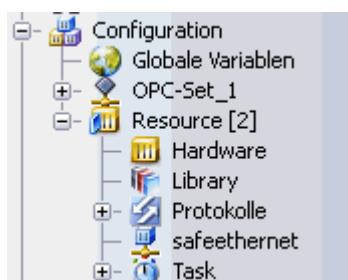


Figure 3: Resource Structure Tree

#### To open the safeethernet Editor of the resource A

1. In the structure tree, open **Configuration**, **Resource**.
  2. Right-click **safeethernet** and select **Edit** from the context menu.
- The target resources are located in the Object Panel.

#### To create the safeethernet connection to the target resource

1. Drag the **target resource** from the Object Panel onto a free space within the workspace of the **safeethernet** Editor.

- 
- i** The reciprocal communication path is automatically added in the safeethernet Editor of the target resource.
- 

#### Configuring the safeethernet Connection:

1. Select **Ethernet Interfaces Channel 1** for the local and target resource.
2. Select **Ethernet Interfaces Channel 2** for the local and target resource.
3. Select the **Network Profile** for the safeethernet connection (e.g., Fast&Noisy).
4. Calculate and enter **Receive Timeout** and **Response Time** (see Chapter 4.6).

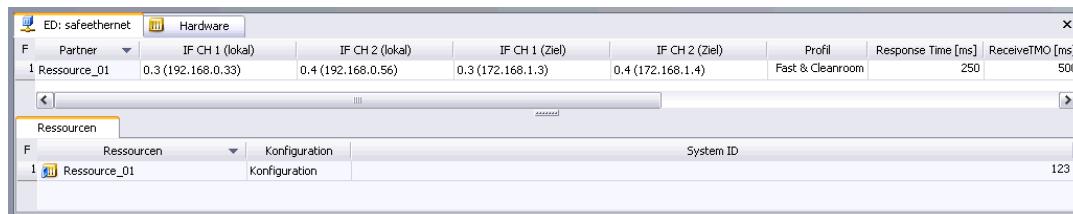


Figure 4: Parameter Values for a Redundant safeethernet Connection

#### 4.2.1 Connecting Process Variables

To add the process variables in the editor of the safeethernet connection.

- 
- i** Only global variables from the configuration context may be used, and not from the resource context!
- 

#### To open the connection editor:

The safeethernet Editor of the resource A is open.

1. Right-click **Resource B** line and open context menu.
2. Select **Edit** from the context menu to open the connection editor of the safeethernet connection.
3. Select the **Resource A<->Resource B** tab.
4. Select the **Resource A->Resource B** area.
5. In the Object Panel, select a **Global Variable** for this transport direction and drag it onto the **Resource A->Resource B** column.
6. Repeat this step for every further variable.
7. Select the **Resource B->Resource A** area.
8. In the Object Panel, select a **Global Variable** for this transport direction and drag it onto the **Resource B->Resource A** column.
9. Repeat this step for further variables.
10. Add other the global variables Connection State, Quality Channel 1 and Quality Channel 2 for each transport direction.

#### To verify the safeethernet connection

1. In the structure tree, select **Configuration, Resource, safeethernet**.
2. Right-click and select **Verification** from the context menu.
3. Thoroughly verify the messages displayed in the logbook and correct potential errors.

- i** The configuration of the **safeethernet** connection must be compiled with the user program of the resource A and resource B and transferred to the controllers. The new configuration can only be used for communicating with the HIMax system upon completion of this step.
- 

#### 4.2.2 Verifying safeethernet Communication

In the Control Panel, reset the *Bad Messages* and *Resends* values to zero.

1. Use the HIMax system under the maximum load:
    - All communication protocols are operating (**safeethernet** and standard protocols).
    - Remove and re-insert the processor module such as described in Chapter 4.6.1.
    - Perform a reload to load the user program (the **safeethernet** cannot be modified by performing a reload).
- 

- i** To verify that the redundant **safeethernet** connection was established properly, disconnect and reconnect one redundant connection and then repeat this test for the other connection. During this test, no faults must occur in the **safeethernet** communication.
- 

2. In the Control Panels of the two controllers, verify the *Bad Messages* and *Resends* values.  
If the counter for *Bad Messages* and *Resends*  
 $= 0$ , then the **safeethernet** settings are OK.  
 $\geq 0$ , the **safeethernet** settings must be rechecked.
    - Recalculate the *Receive Timeout* using the maximum cycle time, see Chapters 4.6.1 and 4.6.3.
    - Vary the *Response Time* such as described in Chapter 4.6.4.
- 

- i** Additional causes for *bad messages* and *resends*!

Verify the correct network design (e.g., lines, switches, PCs). If the Ethernet network is not exclusively used for **safeethernet**, also verify the network load (probable data collisions).

---

#### 4.3 safeethernet Editor

The **safec** Editor is used to create and configure the safeethernet connections to the communication partners (resources).

##### To open the safeethernet Editor of the local resource

1. In the structure tree, open **Configuration, Resource**.
2. Right-click **safeethernet** and select **Edit** from the context menu.

The **safeethernet** Editor includes the workspace and the Object Panel.

To do this, drag the communication partners (resource) from the Object Panel onto the workspace.

The following **safeethernet** protocol parameters must be set to configure the **safeethernet** connection:

| Parameter     | Description   |
|---------------|---|
| Name          | Name of the <b>safeethernet</b> connection  |
| ID            | <b>safeethernet</b> connection ID<br>Range of values: 0...63  |
| Partner       | Resource name of the link partner   |
| IF Channel... | Ethernet interfaces available on the (local) and (remote) resource, see also Chapter 3.5.   |
| Timing Master | The timing master provides the <i>receive timeout</i> , <i>resend timeout</i> and the <i>acknowledge timeout</i> for this <b>safeethernet</b> connection. The opposite controller is the timing slave and resume these values.<br>If no <i>timing master</i> is selected, the controller with the smaller IP address determines these <b>safeethernet</b> parameters. |
| Profile       | Combination of matching <b>safeethernet</b> parameters, see also Chapter 4.7.7.   |
| Rsp t         | Time until the message acknowledgment is received by the sender, see also Chapter 4.6.4.<br>Default value: 500 ms   |
| Rcv TMO       | Monitoring time of PES 1 within which a correct response from PES 2 must be received, see also Chapter 4.6.3. Default value: 1000 ms  |
| Rsnd TMO      | Monitoring time expressed in milliseconds (ms) and set in PES1 within which PES2 must have acknowledged the reception of a data packet; upon expiration of this period, the data packet is sent again 4.6.6.  |
| Ack TMO       | Time period within which the CPU must acknowledge the reception of a data packet, see also Chapter 4.6.7.   |
| Prod Rate     | Production rate: Minimum time interval between two data packets, see also Chapter 4.6.8.  |
| Queue         | Number of data packets that can be sent without acknowledgment, see also Chapter 4.6.9.   |

|   |   |
|---|---|
| Behavior  | Behavior of the input variables for this safeethernet connection if the connection is interrupted.  |
|   | Use Initial Value<br>The initial data are used for the input variables.   |
|   | Freeze Process Value with no Limits<br>The input variables are freezed to the current value and used until a new connection is established.   |
|   | Limited<br>Input: Double-click the drop-down field and enter the time value.<br><br>The input variables are freezed to the current value and used until the configured timeout.<br>Afterwards, the initial data are used for the input variables.<br><br>The timeout can be extended by up to a CPU cycle.                |
| <b>⚠ CAUTION</b><br> <p>The Use Initial Data setting may only be used for safety-related functions implemented via safeethernet.</p> |   |
| Diag.Entry  | The number of warnings that must occur in sequence within the <i>Warning Period [ms]</i> before the warnings are recorded in the diagnosis or communication fault statistic.  |
| Prio A&E  | The function is only activated for the connection to the X-OPC server. This is done to define the priority with which the X-OPC server requires events from the controller.<br><br>Fragments with the priority <b>n</b> and fragments with the priority <b>m</b> are sent at a ratio of <b>n</b> to <b>m</b> times.       |
| Prio Sync   | The function is only activated for the connection to the X-OPC server. This is done to define the priority with which the X-OPC server requires state values from the controller.<br><br>Fragments with the priority <b>n</b> and fragments with the priority <b>m</b> are sent at a ratio of <b>n</b> to <b>m</b> times. |
| A&E activ   | Default value: Deactivated  |
| Codegen   | Default value: V6 and higher<br>V6 and higher: optimized safeethernet Signatur<br>up to V6: standard safeethernet Signatur  |

Table 27: safeethernet Protocol Parameters

### Object Panel

The Object Panel contains all the project resources with which the current resource can be connected via safeethernet.

- i** An export function is available to establish safeethernet connections to resources outside the project or to a HIMatrix controller (planned in ELOP II Factory) (see Chapter 4.8).

#### 4.4 Detail View of the safeethernet Editor

The **detail view** has always a reference to the local resource for which the safeethernet Editor was started.

**To open the detail view of a safeethernet connection:**

1. Right-click the safeethernet connection to open the context menu.
  2. Select **Detail View**.
- The **Detail View** includes the three tabs *Peer1 <-> Peer2*, *Peer1* and *Peer2*.

##### 4.4.1 Tab: *Peer1<->Peer2*

The *Peer1<->Peer2* tab is divided into two areas for the required transport direction: *Peer1->Peer2* and *Peer2->Peer1*.

For transport via safeethernet, *Global Variables* can be dragged onto these two areas.

##### 4.4.2 Tab: *Peer1 (2)*

The *Peer1 (2)* tab contains the two tabs *System Variables* and *Fragment Definitions: Peer->Peer*.

##### 4.4.2.1 Tab System Variables

The safeethernet connection can be controlled and evaluated using system variables.

| Name   | Data type | R/W | Description   |
|--|-----------|-----|---|
| The following statuses and parameters can be assigned global variables and used in the user program. |           |     |   |
| Ack.Frame No.  | UDINT     | R   | Receive Counter (revolving)   |
| Number of bad messages   | UDINT     | R   | Number of all the bad messages per channel (invalid CRC, invalid header, other faults)      |
| Number of bad messages for the redundant channel   | UDINT     | R   |   |
| Number of Successful Connections   | UDINT     | R   | Number of successful connections since statistics reset.                                    |
| Number of lost messages  | UDINT     | R   | Number of messages dropped out on one of the two transmission paths since statistics reset. |
| Number of lost messages for the redundant channel  | UDINT     | R   | The counter only continues to run until a channel completely fails.                         |
| Early Queue Usage  | UDINT     | R   | Number of messages stored in Early Queue since statistical reset, see also Chapter 4.6.9.   |
| Bad Messages   | UDINT     | R   | Number of rejected messages since statistics reset.   |
| Frame No.  | UDINT     | R   | Send counter (revolving).   |

| Channel state                       | USINT  | R  | <p>Current state of Channel 1.<br/>The channel state is the current state of channel 1 to the time (Seq. no X-1) when a message with Seq. no. X is received.</p> <table border="1"> <thead> <tr> <th>Status</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No message on the state of Channel 1.</td></tr> <tr> <td>1</td><td>Channel 1 OK.</td></tr> <tr> <td>2</td><td>The last message was wrong, the current one is OK.</td></tr> <tr> <td>32</td><td>Fault on Channel 1.</td></tr> </tbody> </table>  | Status  | Description | 0       | No message on the state of Channel 1. | 1                               | Channel 1 OK.                 | 2 | The last message was wrong, the current one is OK. | 32                              | Fault on Channel 1. |                                 |                             |   |   |  |       |          |          |
|-------------------------------------|--|--|---|---------|-------------|---------|---------------------------------------|---------------------------------|-------------------------------|---|--|---------------------------------|---------------------|---------------------------------|-----------------------------|---|---|--|-------|----------|----------|
| Status                              | Description  |  |   |         |             |         |                                       |                                 |                               |   |  |                                 |                     |                                 |                             |   |   |  |       |          |          |
| 0                                   | No message on the state of Channel 1.              |  |   |         |             |         |                                       |                                 |                               |   |  |                                 |                     |                                 |                             |   |   |  |       |          |          |
| 1                                   | Channel 1 OK.                                      |  |   |         |             |         |                                       |                                 |                               |   |  |                                 |                     |                                 |                             |   |   |  |       |          |          |
| 2                                   | The last message was wrong, the current one is OK. |  |   |         |             |         |                                       |                                 |                               |   |  |                                 |                     |                                 |                             |   |   |  |       |          |          |
| 32                                  | Fault on Channel 1.                                |  |   |         |             |         |                                       |                                 |                               |   |  |                                 |                     |                                 |                             |   |   |  |       |          |          |
| Layout Version                      | UDINT  | R  | Signature of the data layout used within communication.   |         |             |         |                                       |                                 |                               |   |  |                                 |                     |                                 |                             |   |   |  |       |          |          |
| Last channel latency                | UDINT  | R  | The channel latency specifies the delay between two redundant transmission paths to the reception time of messages with identical SeqNo. A statistic is kept specifying the average, minimum, maximum and last latency.   |         |             |         |                                       |                                 |                               |   |  |                                 |                     |                                 |                             |   |   |  |       |          |          |
| Last latency of the red. channel    | UDINT  | R  | If the minimum value is greater than the maximum value, the statistic values are invalid.   |         |             |         |                                       |                                 |                               |   |  |                                 |                     |                                 |                             |   |   |  |       |          |          |
| Max. channel latency                | UDINT  | R  | The last channel latency and the average channel latency are then 0.  |         |             |         |                                       |                                 |                               |   |  |                                 |                     |                                 |                             |   |   |  |       |          |          |
| Max. latency of the red. channel    | UDINT  | R  |   |         |             |         |                                       |                                 |                               |   |  |                                 |                     |                                 |                             |   |   |  |       |          |          |
| Min. channel latency                | UDINT  | R  |   |         |             |         |                                       |                                 |                               |   |  |                                 |                     |                                 |                             |   |   |  |       |          |          |
| Min. latency of the red. channel    | UDINT  | R  |   |         |             |         |                                       |                                 |                               |   |  |                                 |                     |                                 |                             |   |   |  |       |          |          |
| Average channel latency             | UDINT  | R  |   |         |             |         |                                       |                                 |                               |   |  |                                 |                     |                                 |                             |   |   |  |       |          |          |
| Average latency of the red. channel | UDINT  | R  |   |         |             |         |                                       |                                 |                               |   |  |                                 |                     |                                 |                             |   |   |  |       |          |          |
| Monotony                            | UDINT  | R  | User data send counter (revolving).   |         |             |         |                                       |                                 |                               |   |  |                                 |                     |                                 |                             |   |   |  |       |          |          |
| New Layout Version                  | UDINT  | R  | Signature of the new data layout.   |         |             |         |                                       |                                 |                               |   |  |                                 |                     |                                 |                             |   |   |  |       |          |          |
| Quality of Channel 1                | BYTE   | R  | <p>State of the main transmission path.</p> <table border="1"> <thead> <tr> <th>Bit no.</th> <th>Bit = 0</th> <th>Bit = 1</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Transmission path not activated</td> <td>Transmission path enabled</td> </tr> <tr> <td>1</td> <td>Transmission path not used</td> <td>Transmission path actively used</td> </tr> <tr> <td>2</td> <td>Transmission path not connected</td> <td>Transmission path connected</td> </tr> <tr> <td>3</td> <td>-</td> <td>Transmission path first provides message</td> </tr> <tr> <td>4 - 7</td> <td>Reserved</td> <td>Reserved</td> </tr> </tbody> </table> | Bit no. | Bit = 0     | Bit = 1 | 0                                     | Transmission path not activated | Transmission path enabled     | 1 | Transmission path not used                         | Transmission path actively used | 2                   | Transmission path not connected | Transmission path connected | 3 | - | Transmission path first provides message | 4 - 7 | Reserved | Reserved |
| Bit no.                             | Bit = 0  | Bit = 1                                  |   |         |             |         |                                       |                                 |                               |   |  |                                 |                     |                                 |                             |   |   |  |       |          |          |
| 0                                   | Transmission path not activated                    | Transmission path enabled                |   |         |             |         |                                       |                                 |                               |   |  |                                 |                     |                                 |                             |   |   |  |       |          |          |
| 1                                   | Transmission path not used                         | Transmission path actively used          |   |         |             |         |                                       |                                 |                               |   |  |                                 |                     |                                 |                             |   |   |  |       |          |          |
| 2                                   | Transmission path not connected                    | Transmission path connected              |   |         |             |         |                                       |                                 |                               |   |  |                                 |                     |                                 |                             |   |   |  |       |          |          |
| 3                                   | -  | Transmission path first provides message |   |         |             |         |                                       |                                 |                               |   |  |                                 |                     |                                 |                             |   |   |  |       |          |          |
| 4 - 7                               | Reserved   | Reserved                                 |   |         |             |         |                                       |                                 |                               |   |  |                                 |                     |                                 |                             |   |   |  |       |          |          |
| Quality of Channel 2                | BYTE   | R  | State of the redundant transmission path, see state of Channel 1 (main transmission path).  |         |             |         |                                       |                                 |                               |   |  |                                 |                     |                                 |                             |   |   |  |       |          |          |
| Receive Timeout                     | UDINT  | R  | <p>Time in milliseconds (ms) of PES1 within which PES2 must receive a valid response.<br/>See also Chapter 4.6.3.</p>   |         |             |         |                                       |                                 |                               |   |  |                                 |                     |                                 |                             |   |   |  |       |          |          |
| Response Time                       | UDINT  | R  | <p>Time in milliseconds (ms) until the acknowledgment of a message is received by the sender, see also Chapter 4.6.4.</p>   |         |             |         |                                       |                                 |                               |   |  |                                 |                     |                                 |                             |   |   |  |       |          |          |
| Reset safeethernet statistics       | BYTE   | W  | <p>Reset the statistical values for the communication connection in the user program (e.g., <i>Number of Bad Messages</i>, <i>Channel State</i>, <i>Timestamp for the Last Fault on the Red. Channel</i>, <i>Resends</i>).</p> <table border="1"> <thead> <tr> <th>Value</th><th>Function</th></tr> </thead> <tbody> <tr> <td>0</td><td>No reset</td></tr> <tr> <td>1-255</td><td>Reset safeethernet statistics</td></tr> </tbody> </table>   | Value   | Function    | 0       | No reset                              | 1-255                           | Reset safeethernet statistics |   |  |                                 |                     |                                 |                             |   |   |  |       |          |          |
| Value                               | Function   |  |   |         |             |         |                                       |                                 |                               |   |  |                                 |                     |                                 |                             |   |   |  |       |          |          |
| 0                                   | No reset   |  |   |         |             |         |                                       |                                 |                               |   |  |                                 |                     |                                 |                             |   |   |  |       |          |          |
| 1-255                               | Reset safeethernet statistics                      |  |   |         |             |         |                                       |                                 |                               |   |  |                                 |                     |                                 |                             |   |   |  |       |          |          |

| Transmission Control Channel1                         | BYTE   | W | <p>Transmission control of channel 1</p> <table border="1"> <tr><td>Bit 0</td><td>Function</td></tr> <tr><td>FALSE</td><td>Transmission path enabled</td></tr> <tr><td>TRUE</td><td>Transmission path locked</td></tr> </table><br><table border="1"> <tr><td>Bit 1</td><td>Function</td></tr> <tr><td>FALSE</td><td>Transmission path enabled for tests</td></tr> <tr><td>TRUE</td><td>Transmission path locked</td></tr> </table> <p>Bits 2...7 reserved.</p>  | Bit 0        | Function    | FALSE                | Transmission path enabled  | TRUE              | Transmission path locked   | Bit 1         | Function   | FALSE | Transmission path enabled for tests | TRUE | Transmission path locked |
|---|--|---|--|--------------|-------------|----------------------|--|-------------------|--|---------------|--|-------|-------------------------------------|------|--------------------------|
| Bit 0   | Function   |   |  |              |             |                      |  |                   |  |               |  |       |                                     |      |                          |
| FALSE   | Transmission path enabled  |   |  |              |             |                      |  |                   |  |               |  |       |                                     |      |                          |
| TRUE  | Transmission path locked   |   |  |              |             |                      |  |                   |  |               |  |       |                                     |      |                          |
| Bit 1   | Function   |   |  |              |             |                      |  |                   |  |               |  |       |                                     |      |                          |
| FALSE   | Transmission path enabled for tests  |   |  |              |             |                      |  |                   |  |               |  |       |                                     |      |                          |
| TRUE  | Transmission path locked   |   |  |              |             |                      |  |                   |  |               |  |       |                                     |      |                          |
| Transmission Control Ch2                              | BYTE   | W | Transmission control of channel 2,<br>see Transmission control of channel 1.   |              |             |                      |  |                   |  |               |  |       |                                     |      |                          |
| Connection Control                                    | WORD   | W | <p>Use this system variable to control the <b>safeethernet</b> connection from within the user program.</p> <table border="1"> <tr><th>Command</th><th>Description</th></tr> <tr><td>Autoconnect (0x0000)</td><td>Default value:<br/>After a <b>safeethernet</b> communication loss, the controller attempts to re-establish the connection in the following CPU cycle.</td></tr> <tr><td>Disabled (0x8000)</td><td><b>safeethernet</b> communication is disabled.</td></tr> </table>  | Command      | Description | Autoconnect (0x0000) | Default value:<br>After a <b>safeethernet</b> communication loss, the controller attempts to re-establish the connection in the following CPU cycle. | Disabled (0x8000) | <b>safeethernet</b> communication is disabled.   |               |  |       |                                     |      |                          |
| Command   | Description  |   |  |              |             |                      |  |                   |  |               |  |       |                                     |      |                          |
| Autoconnect (0x0000)                                  | Default value:<br>After a <b>safeethernet</b> communication loss, the controller attempts to re-establish the connection in the following CPU cycle. |   |  |              |             |                      |  |                   |  |               |  |       |                                     |      |                          |
| Disabled (0x8000)                                     | <b>safeethernet</b> communication is disabled.   |   |  |              |             |                      |  |                   |  |               |  |       |                                     |      |                          |
| Connection State                                      | UINT   | R | <p>The connection state evaluates the status of the communication between two controllers from within the user program.</p> <table border="1"> <tr><th>Status/Value</th><th>Description</th></tr> <tr><td>Closed (0)</td><td>The connection is closed and no attempt is made to open it.</td></tr> <tr><td>Try_open (1)</td><td>An attempt is made to open the connection, but it is still closed. This state applies for both the active and the passive sides.</td></tr> <tr><td>Connected (2)</td><td>The connection is established and functioning (active time monitoring and data exchange)</td></tr> </table> | Status/Value | Description | Closed (0)           | The connection is closed and no attempt is made to open it.  | Try_open (1)      | An attempt is made to open the connection, but it is still closed. This state applies for both the active and the passive sides. | Connected (2) | The connection is established and functioning (active time monitoring and data exchange) |       |                                     |      |                          |
| Status/Value  | Description  |   |  |              |             |                      |  |                   |  |               |  |       |                                     |      |                          |
| Closed (0)  | The connection is closed and no attempt is made to open it.  |   |  |              |             |                      |  |                   |  |               |  |       |                                     |      |                          |
| Try_open (1)  | An attempt is made to open the connection, but it is still closed. This state applies for both the active and the passive sides.                     |   |  |              |             |                      |  |                   |  |               |  |       |                                     |      |                          |
| Connected (2)   | The connection is established and functioning (active time monitoring and data exchange)   |   |  |              |             |                      |  |                   |  |               |  |       |                                     |      |                          |
| Version State   | UINT   | R | <p>Reload version state of this <b>safeethernet</b> connection, see also status of <i>reload</i> in chapter 4.11.1.</p> <table> <tr><td>unknown</td><td>0x0000</td></tr> <tr><td>uptodate:</td><td>0x0001</td></tr> <tr><td>updated:</td><td>0x0002</td></tr> <tr><td>outdated:</td><td>0x0003</td></tr> </table>  | unknown      | 0x0000      | uptodate:            | 0x0001   | updated:          | 0x0002   | outdated:     | 0x0003   |       |                                     |      |                          |
| unknown   | 0x0000   |   |  |              |             |                      |  |                   |  |               |  |       |                                     |      |                          |
| uptodate:   | 0x0001   |   |  |              |             |                      |  |                   |  |               |  |       |                                     |      |                          |
| updated:  | 0x0002   |   |  |              |             |                      |  |                   |  |               |  |       |                                     |      |                          |
| outdated:   | 0x0003   |   |  |              |             |                      |  |                   |  |               |  |       |                                     |      |                          |
| Resends   | UDINT  | R | Number of resends since statistics reset.  |              |             |                      |  |                   |  |               |  |       |                                     |      |                          |
| Timestamp for the last fault on the red. channel [ms] | UDINT  | R | Millisecond fraction of the timestamp (current system time)  |              |             |                      |  |                   |  |               |  |       |                                     |      |                          |
| Timestamp for the last fault on the red. channel [s]  | UDINT  | R | Second fraction of the timestamp (current system time)   |              |             |                      |  |                   |  |               |  |       |                                     |      |                          |
| Timestamp for the last fault [ms]                     | UDINT  | R | Millisecond fraction of the timestamp (current system time)  |              |             |                      |  |                   |  |               |  |       |                                     |      |                          |
| Timestamp for the last fault [s]                      | UDINT  | R | Second fraction of the timestamp (current system time)   |              |             |                      |  |                   |  |               |  |       |                                     |      |                          |

| State of the red.<br>channel | USINT   | R | Current state of Channel 2.<br>The channel state is the current state of channel 2 to the time<br>(Seq. no X-1) when a message with Seq. no. X is received.<br><table border="1"><thead><tr><th>Status</th><th>Description</th></tr></thead><tbody><tr><td>0</td><td>No message on the state of channel 2</td></tr><tr><td>1</td><td>Channel 2 OK.</td></tr><tr><td>2</td><td>The last message was wrong, the current one<br/>is OK.</td></tr><tr><td>3</td><td>Fault on Channel 2.</td></tr></tbody></table> | Status | Description | 0 | No message on the state of channel 2 | 1 | Channel 2 OK. | 2 | The last message was wrong, the current one<br>is OK. | 3 | Fault on Channel 2. |
|------------------------------|---|---|---|--------|-------------|---|--------------------------------------|---|---------------|---|---|---|---------------------|
| Status                       | Description   |   |   |        |             |   |                                      |   |               |   |   |   |                     |
| 0                            | No message on the state of channel 2                  |   |   |        |             |   |                                      |   |               |   |   |   |                     |
| 1                            | Channel 2 OK.   |   |   |        |             |   |                                      |   |               |   |   |   |                     |
| 2                            | The last message was wrong, the current one<br>is OK. |   |   |        |             |   |                                      |   |               |   |   |   |                     |
| 3                            | Fault on Channel 2.                                   |   |   |        |             |   |                                      |   |               |   |   |   |                     |

Table 28: System Variables Tab in the safeethernet Editor

#### 4.4.2.2 Tab: Fragment Definitions

The *Fragment Definitions* tab outputs the status and includes parameters for the fragments sent by opposite controller.

At this level, one can set for this controller (or X-OPC server) the required synchronization rate of the received fragments from all the connected controller. The priority setting is mostly intended for the X-OPC server, which process a large data volume from various controller.

| Name   | Data type   | R/W | Description  |          |             |           |                              |          |  |           |   |
|--|---|-----|--|----------|-------------|-----------|------------------------------|----------|--|-----------|---|
| The following statuses and parameters can be assigned global variables and used in the user program. |   |     |  |          |             |           |                              |          |  |           |   |
| Fragment Definition  | -   | -   | <p>The Priority column is used to define how often this fragment should be received compared to the other fragments</p> <ul style="list-style-type: none"> <li>▪ A HIMax fragment is a fragment of <math>\leq 1100</math> bytes.</li> <li>▪ A HIMatrix fragment is a fragment of <math>\leq 900</math> bytes.</li> </ul> <p>Default setting: priority 1<br/>Range of values for the priorities: 1 (highest) to 65 535 (lowest)</p>   |          |             |           |                              |          |  |           |   |
| Fragment Version State   | UINT  | R   | <p>Reload version state of this safeethernet fragment, see also status of <i>reload</i> in chapter 4.11.1.</p> <table> <tr> <td>unknown:</td> <td>0x0000</td> </tr> <tr> <td>upToDate:</td> <td>0x0001</td> </tr> <tr> <td>updated:</td> <td>0x0002</td> </tr> <tr> <td>outdated:</td> <td>0x0003</td> </tr> </table>  | unknown: | 0x0000      | upToDate: | 0x0001                       | updated: | 0x0002   | outdated: | 0x0003  |
| unknown:   | 0x0000  |     |  |          |             |           |                              |          |  |           |   |
| upToDate:  | 0x0001  |     |  |          |             |           |                              |          |  |           |   |
| updated:   | 0x0002  |     |  |          |             |           |                              |          |  |           |   |
| outdated:  | 0x0003  |     |  |          |             |           |                              |          |  |           |   |
| Fragment timestamp [ms]  | UDINT   | R   | Millisecond fraction of the timestamp (current system time)  |          |             |           |                              |          |  |           |   |
| Fragment timestamp [s]   | UDINT   | R   | Second fraction of the timestamp (current system time)   |          |             |           |                              |          |  |           |   |
| Fragment state   | UINT  | R   | <table border="1"> <thead> <tr> <th>Status</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>CLOSED: Connection is closed</td> </tr> <tr> <td>1</td> <td>TRY OPEN: An attempt is made to open the connection, but it is still closed.</td> </tr> <tr> <td>2</td> <td>CONNECTED: The connection exists and current fragment data was received (cp. timestamp). As long as no fragment data is received, the fragment state is set to TRY_OPEN when establishing the connection.</td> </tr> </tbody> </table> <p><b>i</b> The connection state of the safeethernet Editor is set to CONNECTED as soon as the connection is open. Unlike Fragment State, no data may have been exchanged here.</p> | Status   | Description | 0         | CLOSED: Connection is closed | 1        | TRY OPEN: An attempt is made to open the connection, but it is still closed. | 2         | CONNECTED: The connection exists and current fragment data was received (cp. timestamp). As long as no fragment data is received, the fragment state is set to TRY_OPEN when establishing the connection. |
| Status   | Description   |     |  |          |             |           |                              |          |  |           |   |
| 0  | CLOSED: Connection is closed  |     |  |          |             |           |                              |          |  |           |   |
| 1  | TRY OPEN: An attempt is made to open the connection, but it is still closed.  |     |  |          |             |           |                              |          |  |           |   |
| 2  | CONNECTED: The connection exists and current fragment data was received (cp. timestamp). As long as no fragment data is received, the fragment state is set to TRY_OPEN when establishing the connection. |     |  |          |             |           |                              |          |  |           |   |

Table 29: Tab Fragment Definitions

## 4.5 Possible safeethernet Connections

A **safeethernet** connection between two HIMax controllers can be configured as mono or redundant. The Ethernet interfaces available for a **safeethernet** connection are always displayed related to the resource (local) for which the **safeethernet** Editor was opened. All Ethernet interfaces available for a controller are showed in the drop-down menu for the **IF Channel...** parameter.

| Element               | Description                            |
|-----------------------|--|
| IF Channel 1 (local)  | Ethernet interface of the resource     |
| IF Channel 2 (local)  | Ethernet interface of the resource     |
| IF Channel 1 (remote) | Ethernet interface of the link partner |
| IF Channel 2 (remote) | Ethernet interface of the link partner |

Table 30: Available Ethernet Interfaces

### 4.5.1 Mono safeethernet Connection (Channel 1)

In the local resource, set the Ethernet interfaces IF Channel 1 (local) and IF Channel 1 (remote) for a mono connection.

### 4.5.2 Redundant safeethernet Connection (Channel 1 and Channel 2)

Redundant **safeethernet** transmission paths between two HIMax/HIMatrix controllers are possible.

For a redundant connection, the following Ethernet interfaces can be used:

- The Ethernet interfaces IF Channel 1 (local) and IF Channel 1 (remote) for channel 1.
- The Ethernet interfaces IF Channel 2 (local) and IF Channel 2 (remote) for channel 2.
- For a redundant connection via channel 1 and channel 2 using only one Ethernet interface, select the same Ethernet interface IF Channel1 (local) and IF Channel 2 (local) in the **safeethernet** Editor.



The redundant transmission paths must be sufficiently homogeneous to ensure that the bandwidth and the delay on the two transmission paths are nearly identical.

Once the offset of the received messages becomes too large or the messages require longer than the response time to arrive, the transmission path diagnosis no longer operates as intended and considers these delays transmission path as faults.

To evaluate the transmission path diagnosis, refer to the system variables *State of the Red. Channel* and *Channel State*.

### 4.5.3 Permitted Combinations

The following chapters list the possible combinations for redundant **safeethernet** connections.



In accordance with the generally accepted regulations for developing Ethernet networks, no network loop may occur. Data packets may only reach a controller over a single path.

#### 4.5.3.1 Redundant safeethernet with 2 Separated Transmission Paths

A redundant connection to two separated transmission paths (channel 1 and channel 2) can be established with HIMax and HIMatrix F\*03 controllers if these controllers exhibit one of the following interface combinations.

| Interface combination | IF Channel 1 (local)      | IF Channel 2 (local)      |
|-----------------------|---------------------------|---------------------------|
| a)                    | IP of HIMax CPU 1         | IP of HIMax CPU 2         |
| b)                    | IP of HIMax CPU 1         | IP of HIMax COM 1         |
| c)                    | IP of HIMax COM 1         | IP of HIMax COM 2         |
| d)                    | IP of HIMatrix F*03 CPU1) | IP of HIMatrix F*03 COM1) |

<sup>1)</sup> With HIMatrix F\*03, the switch ports are separated from one another via VLAN, see Chapter 4.5.3.4.

Table 31: Potential Subscribers are HIMax and HIMatrix F\*03 Controllers

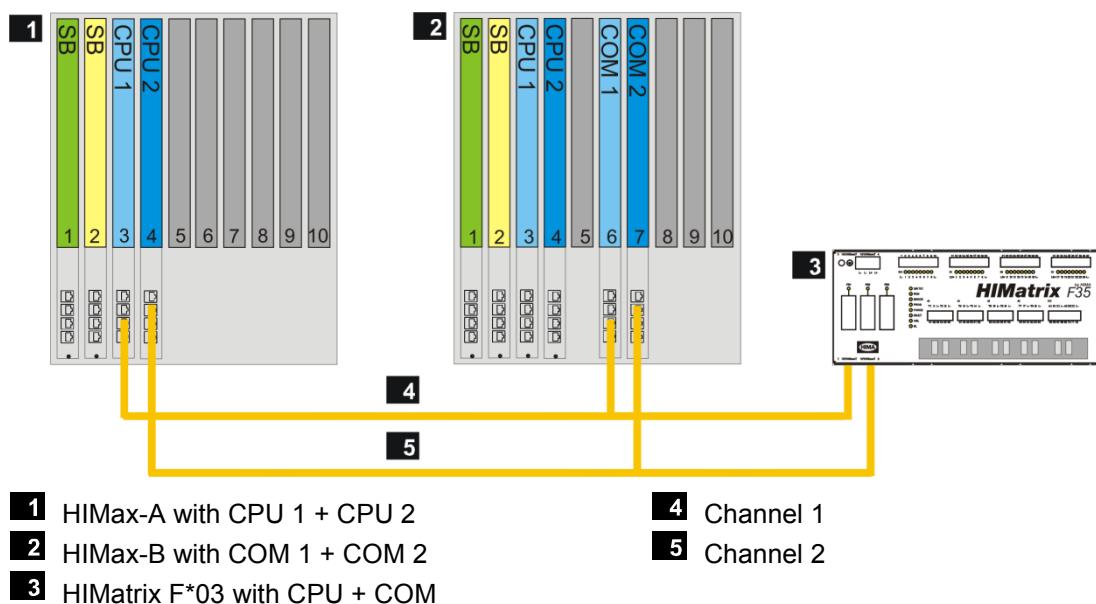


Figure 5: Redundant Connection between HIMax and HIMatrix F\*03 Controllers

#### 4.5.3.2 Redundant safeethernet with HIMA Ring Master

A redundant connection via two transmission paths (channel 1 and channel 2) is also possible with a ring structure. In such a network, a controller with two IP addresses must exist and operate as ring master, see Figure 6. Restriction: In a ring structure, the entire safeethernet communication must run through the ring master. safeethernet communication between the ring slaves is not allowed!

The ring master controller must exhibit one of the following interface combinations (two IP addresses each), suitable are, e.g., HIMax or HIMatrix F\*03.

| Interface combination | IF Channel 1 (local)                   | IF Channel 2 (local)                  |
|-----------------------|--|---------------------------------------|
| a)                    | IP of HIMax CPU 1                      | IP of HIMax CPU 2                     |
| b)                    | IP of HIMax CPU 1                      | IP of HIMax COM 1                     |
| c)                    | IP of HIMax COM 1                      | IP of HIMax COM 2                     |
| d)                    | IP von HIMatrix F*03 CPU <sup>1)</sup> | IP of HIMatrix F*03 COM <sup>1)</sup> |

<sup>1)</sup> With HIMatrix F\*03, the switch ports are separated from one another via VLAN, see Chapter 4.5.3.4.

Table 32: Potential Ring Master are HIMax and HIMatrix F\*03 Controllers

Possible ring slaves are HIMax, HIMatrix F\*03, HIMatrix standard. These controllers use one of the following interface combinations (one IP address each) for safeethernet communication.

| Interface combination | IF Channel 1 (local)        | IF Channel 2 (local)        |
|-----------------------|-----------------------------|-----------------------------|
| a)                    | IP of HIMax CPU             | IP of HIMax CPU             |
| b)                    | IP of HIMax COM             | IP of HIMax COM             |
| c) <sup>1)</sup>      | IP of HIMatrix Standard COM | IP of HIMatrix Standard COM |
| d)                    | IP of HIMatrix F*03 CPU     | IP of HIMatrix F*03 COM     |
| e)                    | IP of HIMatrix F*03 COM     | IP of HIMatrix F*03 CPU     |

<sup>1)</sup> Only with HIMax as ring master with the interface combinations in accordance to table 32.

Table 33: Possible ring slaves are HIMax, HIMatrix F\*03, HIMatrix standard.

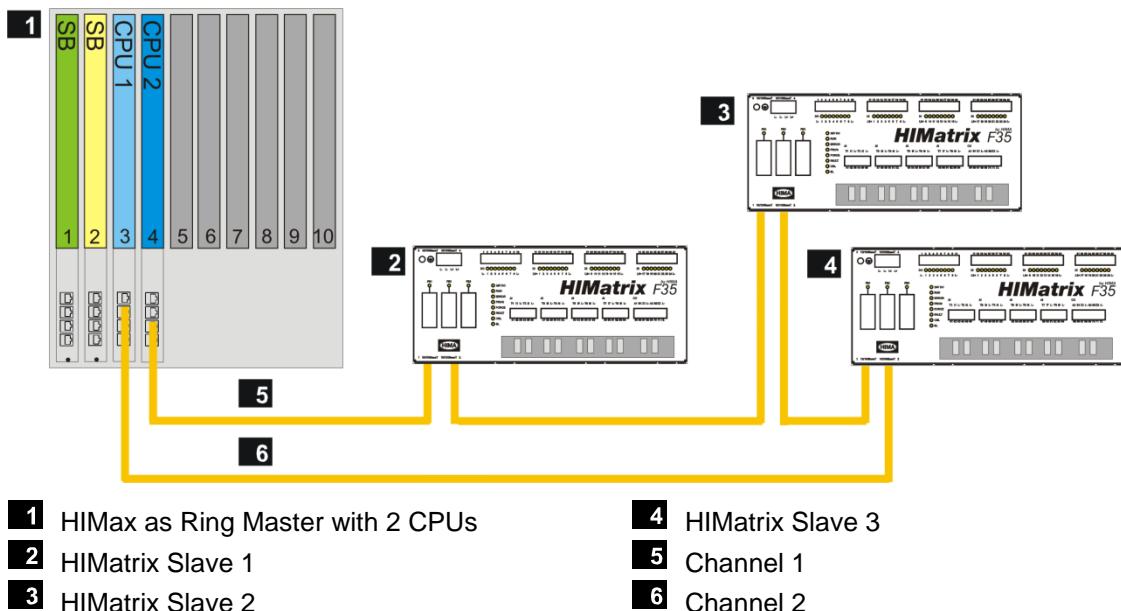


Figure 6: Ring Connection between HIMax and three HIMatrix standard via two Transmission Paths (Channel 1 and Channel 2)

#### 4.5.3.3 Redundant safeethernet with one Transmission Path

A redundant connection can also be configured if only one transmission path is available or the controller only has one Ethernet interface (IP address). In such cases, channel 1 and channel 2 are transferred consecutively across the same transmission path. The constant data throughput increases the connection's interference immunity, e.g., against EMC interferences.

These controllers use one of the following interface combinations, allowing usage of only one Ethernet interface (IP address).

The safeethernet partner can also use only a single Ethernet interface (IP address).

| Interface combination | IF Channel 1 (local)        | IF Channel 2 (local)        |
|-----------------------|-----------------------------|-----------------------------|
| a)                    | IP of HIMax CPU 1           | IP of HIMax CPU 1           |
| b)                    | IP of HIMax COM 1           | IP of HIMax COM 1           |
| c)                    | IP of HIMatrix F*03 CPU     | IP of HIMatrix F*03 CPU     |
| d)                    | IP of HIMatrix F*03 COM     | IP of HIMatrix F*03 COM     |
| e)                    | IP of HIMatrix Standard COM | IP of HIMatrix Standard COM |

Table 34: Interface Combinations with one Ethernet Interface (IP Address)

The safeethernet partner, however, can also use two Ethernet interfaces (IP addresses).

| Interface combination | IF Channel 1 (local)       | IF Channel 2 (local)      |
|-----------------------|----------------------------|---------------------------|
| a)                    | IP of HIMax CPU 1          | IP of HIMax CPU 2         |
| b)                    | IP of HIMax CPU 1          | IP of HIMax COM 1         |
| c)                    | IP of HIMax COM 1          | IP of HIMax COM 2         |
| d)                    | IP von HIMatrix F*03 CPU1) | IP of HIMatrix F*03 COM1) |

<sup>1)</sup> With HIMatrix F\*03, the switch ports are separated from one another via VLAN, see Chapter 4.5.3.4.

Table 35: Interface Combinations with two Ethernet Interfaces (IP Addresses)

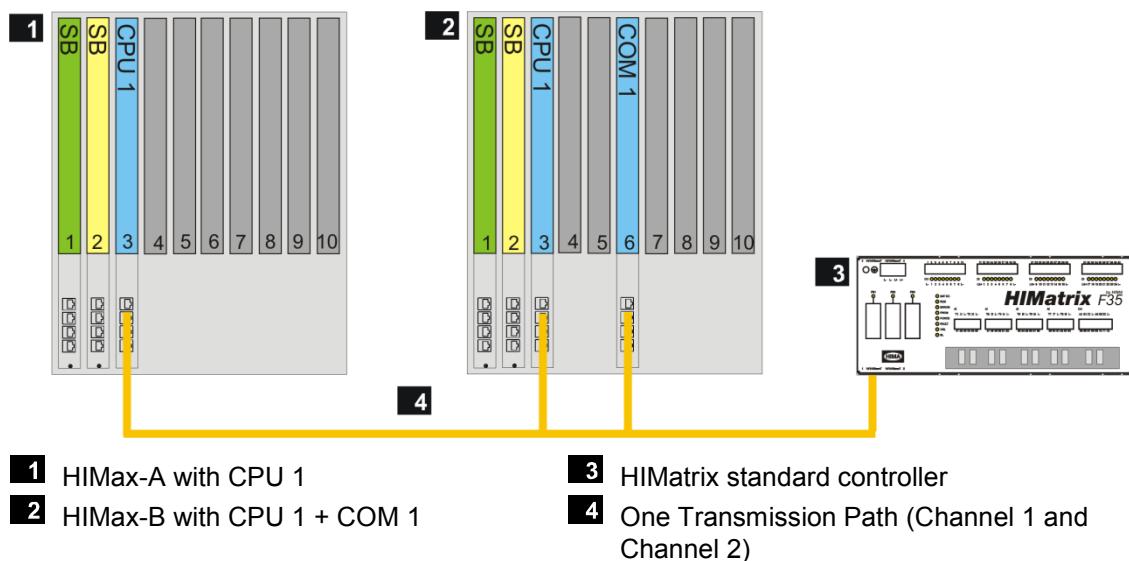
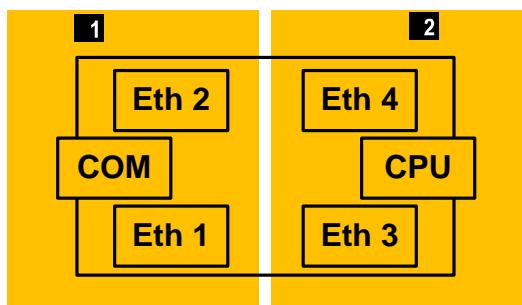


Figure 7: Redundant Connection of Two HIMax and one HIMatrix standard Controllers using one Transmission Path

#### 4.5.3.4 Separating Switch Ports via VLAN

Using HIMatrix controllers with enhanced performance (F\*03), the 4 available Ethernet ports (Eth1...Eth4) can be separated in accordance with the required application. It is therefore possible to establish a connection with two IP addresses or to separate safe communication through the CPU from non-safe communication through the COM.



- 1** Eth1 and Eth 2 are assigned to the COM
- 2** Eth3 and Eth 4 are assigned to the CPU

Figure 8: Example of Switch Ports separated via VLAN

The switch ports are configured in the detail view of the CPU or COM module, see Chapter 3.5.3.

Parameters for separating the switch ports are set in the VLAN tab such as depicted above.

|      | Eth1     | Eth2     | Eth3     | Eth4     | COM      |
|------|----------|----------|----------|----------|----------|
| Eth1 |          |          |          |          |          |
| Eth2 | active   |          |          |          |          |
| Eth3 | inactive | inactive |          |          |          |
| Eth4 | inactive | inactive | active   |          |          |
| COM  | active   | active   | inactive | inactive |          |
| CPU  | inactive | inactive | active   | active   | inactive |

Table 36: VLAN Tab

The PADT communication takes place via an externally accessible Ethernet port. If this Ethernet port is excluded per VLAN configuration, the communication between the PADT and the controller via this Ethernet port is not possible after download or reload.

If all Ethernet port connections to the processor of the controller were stopped by the VLAN configuration, then the controller must be reset. Then the controller is accessible via the default IP address again.

## 4.6 safeethernet Parameters

The safety-related communication is configured in the safeethernet Editor. The parameters described in this chapter must be set.

For determining the *Receive Timeout* and *Response Time* safeethernet parameters, the following condition applies:

The communication time slice must be sufficiently high to allow all the safeethernet connections to be processed within one CPU cycle, see Chapter .

### 4.6.1 Maximum Cycle Time (Minimum Watchdog Time) of the HIMax Controller

To determine the maximum cycle time for a HIMax controller (minimum watchdog time), HIMA recommends proceeding as follows when all the processor modules of the system are inserted.

1. Set the watchdog time high for testing.
2. Use the system under the maximum load. In the process, all communication connections must be operating both via safeethernet and standard protocols. Frequently read the cycle time in the Control Panel and note the variations of the cycle time.
3. In succession, remove and reinsert every processor module in the base plate. Prior to removing one processor module, wait that the processor module that has just been inserted is synchronized.

**i** When a processor module is inserted in the base plate, it automatically synchronizes itself with the configuration of the existing processor modules. The time required for the synchronization process extends the controller cycle up to the maximum cycle time.

The synchronization time increases with the number of processor modules that have already been synchronized.

For more information on how to insert and remove a processor module, refer to the X-CPU 01 Manual (HI 801 009 E).

4. In the diagnostic history, read the synchronization time from n to n+1 processor modules in every synchronization process and note down.
5. Repeat these steps for the next communication partner (the second HIMax controller). The greatest synchronization time value is used to determine the watchdog time.

**i** Note down the synchronization times of both HIMax controllers!

6. Calculate the minimum watchdog time from the longest synchronization time + 12 ms spare + spare for the noted variations of the cycle time.

A suitable value for the maximum cycle time (minimum watchdog time) has been thus determined for the following calculations.

**TIP**

Perform the calculations specified in step 6 for both HIMax controllers and use the corresponding synchronization time value previously noted down.

The maximum cycle times (minimum watchdog time) calculated as described above can be used as watchdog time in the corresponding resource, see safety manual (HI 801 003).

#### 4.6.2 Maximum Cycle Time of the HIMatrix Controller

To determine the maximum cycle time for a HIMatrix controller, HIMA recommends to proceed as follows:

##### To determine the maximum cycle time for the HIMatrix controller

1. Use the system under the maximum load. In the process, all communication connections must be operating both via safeethernet and standard protocols. Frequently read the cycle time in the Control Panel and note the maximum cycle time.
2. Repeat step 1 for the next communication partner (i.e., the second HIMatrix controller).
3. The required maximum cycle time is the greatest value between the two cycle times previously determined.

The maximum cycle time was determined and is used in the following calculations.

#### 4.6.3 Receive Timeout

*ReceiveTMO* is the monitoring time in milliseconds (ms) within which a correct response from the communication partner must be received.

If a correct response is not received from the communication partner within *ReceiveTMO*, safety-related communication is terminated. The input variables of this safeethernet connection react in accordance with the preset parameter *Freeze Data on Lost Connection [ms]*.

The *Use Initial Data* setting may only be used for safety-related functions implemented via safeethernet.

Since *ReceiveTMO* is a safety-relevant component of the Worst Case Reaction Time  $T_R$  (see Chapter 4.7.1 et seqq.), its value must be determined as described below and entered in the safeethernet Editor.

##### **ReceiveTMO $\geq$ 4\*delay + 5\*max. cycle time**

Condition: The Communication Time Slice must be sufficiently high to allow all the safeethernet connections to be processed within one CPU cycle.

Delay: Delay on the transmission path, e.g., due to switch or satellite.

Max. Cycle Time Maximum cycle time of both controllers.



The availability of the safeethernet communication can be increased by incrementing the *ReceiveTMO* value (e.g., by doubling it), provided that the configured time is still sufficient to perform the safety function (worst case reaction time).

The plant manufacturer and the operator are responsible for ensuring that the safeethernet connection complies at least with the following condition:  $ReceiveTMO \geq 2 * Response\ Time$ .

#### 4.6.4 Response Time

*ResponseTime* is the time in milliseconds (ms) that elapses until the sender of the message receives acknowledgement from the recipient.

When configuring using a safeethernet profile, a *Response Time* parameter must be set based on the physical conditions of the transmission path.

The preset *ResponseTime* affects the configuration of all the safeethernet connection parameters and is calculated as follows:

$$\text{ResponseTime} \leq \text{ReceiveTMO} / n$$

**n = 2, 3, 4, 5, 6, 7, 8.....**

The ratio between *ReceiveTMO* and *Response Time* influences the capability to tolerate faults, e.g., when packets are lost (resending lost data packets) or delays occur on the transmission path.

In networks where packets can be lost, the following condition must be given:

$$\text{min. Response Time} \leq \text{ReceiveTMO} / 2 \geq 2 \cdot \text{Delay} + 2.5 \cdot \text{max. Cycle Time}$$

If this condition is met, the loss of at least one data packet can be intercepted without interrupting the **safeethernet** connection.



If this condition is **not met**, the availability of a **safeethernet** connection can only be ensured in a collision and fault-free network. However, this is not a safety problem for the processor module!

---



Make sure that the transmission path complies with the configured *Response Time*!

If these conditions cannot always be ensured, a corresponding system variable is available for the **safeethernet** connection allowing for the response time monitoring. If more than once occasion the measured response time exceeds the *ReceiveTMO* by more than a half, the configured response time must be increased.

*ReceiveTMO* must be adjusted according to the new value configured for *Response Time*.

The plant manufacturer and the operator are responsible for ensuring that the **safeethernet** connection complies at least with the following condition:  $\text{ReceiveTMO} \geq 2 \cdot \text{Response Time}$ .

---

#### 4.6.5 Sync/Async

Sync Currently not supported.

Async It is the default setting.

If *Async* is set, data is received by the **safeethernet** protocol instance during the CPU input phase and is sent during the CPU output phase in accordance with the sending rules.

#### 4.6.6 Resend Timeout

*ResendTMO* cannot be set manually, but it is calculated based on the profile and *Response Time*.

Monitoring time expressed in milliseconds (ms) and set in PES1 within which PES2 must have acknowledged the reception of a data packet; upon expiration of this period, the data packet is sent again.

Rule:  $\text{Resend Timeout} \leq \text{Receive Timeout}$

If the *resend timeout* values set in the communication partners differ from one another, the active protocol partner (with the lowest SRS) determines the *resend time* for the protocol connection.

#### 4.6.7 Acknowledge Timeout

*AckTMO cannot be set manually, but it is calculated based on the profile and Response Time.*

AckTMO is the time period within which the CPU must acknowledge the reception of a data packet.

In a rapid network, AckTMO is zero, i.e., the reception of a data packet is immediately acknowledged. In a slow network (e.g., a telephone modem line), AckTMO is greater than zero. In this case, the system attempts to transmit the acknowledgment message together with the process data to reduce the network load by avoiding addressing and security blocks.

Rules:

- *AckTMO must be  $\leq$  Receive Timeout*
- *AckTMO must be  $\leq$  Resend Timeout, if ProdRate is  $>$  Resend Timeout.*

#### 4.6.8 Production Rate

*ProdRate cannot be set manually, but it is calculated based on the profile and Response Time.*

Minimum time interval in milliseconds (ms) between two data packets.

The *ProdRate* is used to limit the amount of data packets such that a (slow) communication channel will not be overloaded. This ensures a uniform load of the transmission medium and prevents the receiver from receiving obsolete data.

Rules:

- *ProdRate  $\leq$  Receive Timeout*
- *ProdRate  $\leq$  Resend Timeout, if AckTMO > Resend Timeout.*



A zero production rate means that data packets can be transmitted in each user program cycle.

---

#### 4.6.9 Queue

*Queue cannot be set manually, but it is calculated based on the profile and Response Time.*

Queue is the number of data packets that can be sent with no need to wait for their acknowledgement.

The value depends on the network's transfer capacity and potential network delays.

All safeethernet connections share the message queue available in the CPU.

#### 4.7 Worst Case Reaction Time for safeethernet

In the following examples, the formulas for calculating the worst case reaction time only apply for a connection with HIMatrix controllers if the parameter Safety Time = 2 \* Watchdog Time is set. These formulas always apply to HIMax controllers.



The allowed worst case reaction time depends on the process and must be agreed upon together with the test authority responsible for the final inspection.

##### Terms

|                          |   |
|--------------------------|---|
| ReceiveTMO:              | Monitoring time of PES 1 within which a correct response from PES 2 must be received. Otherwise, safety-related communication is terminated after the time has expired.   |
| Production Rate:         | Minimum interval between two data transmissions.  |
| Watchdog Time:           | Maximum duration permitted for a controller's RUN cycle. The duration of the RUN cycle depends on the complexity of the user program and the number of <b>safeethernet</b> connections. The watchdog time (WDT) must be entered in the resource properties. |
| Worst Case Reaction Time | The worst case reaction time is the time between a change in a physical input signal (in) of PES 1 and a reaction on the corresponding output (out) of PES 2.   |
| Delay:                   | Delay of a transmission path, e.g., with a modem or satellite connection.<br>For direct connections, an initial delay of 2 ms can be assumed.<br>The responsible network administrator can measure the actual delay on a transmission path.                 |

To the calculations of the maximum reaction times specified below, the following conditions apply:

- The signals transmitted over **safeethernet** must be processed in the corresponding controllers within one CPU cycle.
- Further, the reaction time of the sensors and actuators must be added.

The calculations also apply to signals in the opposite direction.

#### 4.7.1 Calculating the Worst Case Reaction Time of Two HIMax controllers

The worst case reaction time  $T_R$  is the time between a change on the sensor input signal (in) of controller 1 and a reaction on the corresponding output (out) of controller 2. It is calculated as follows:

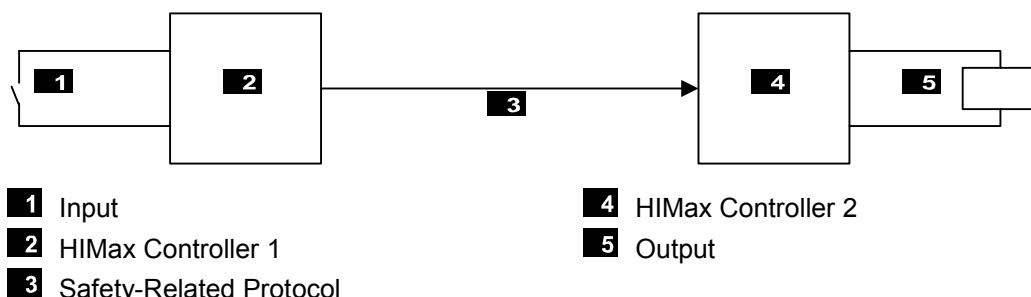


Figure 9: Reaction Time with Interconnection of Two HIMax Controllers

$$T_R = t_1 + t_2 + t_3$$

$T_R$  Worst case reaction time

$t_1$  Safety time of HIMax controller 1.

$t_2$  *ReceiveTMO*

$t_3$  Safety time of HIMax controller 2.

#### 4.7.2 Calculating the Worst Case Reaction Time in Connection with One HIMatrix controller

The worst case reaction time  $T_R$  is the time between a change on the sensor input signal (in) of HIMax controller and a reaction on the corresponding output (out) of HIMatrix controller. It is calculated as follows:

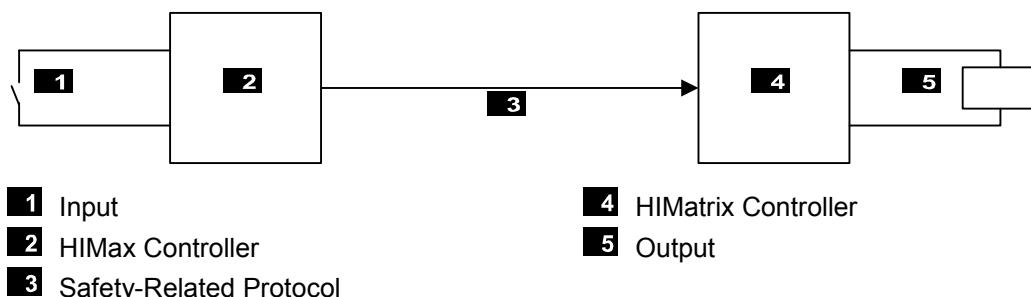


Figure 10: Reaction Time for Connection between One HIMax and One HIMatrix Controller

$$T_R = t_1 + t_2 + t_3$$

$T_R$  Worst case reaction time

$t_1$  Safety time of HIMax controller

$t_2$  *ReceiveTMO*

$t_3$  2 \* Watchdog time of the HIMatrix controller

#### 4.7.3 Calculating the Worst Case Reaction Time with two HIMatrix Controllers or Remote I/Os

The worst case reaction time  $T_R$  is the time between a change on the sensor input signal (in) of the first HIMatrix controller or remote I/O (e.g., F3 DIO 20/8 01) and a reaction on the corresponding output (out) of the second HIMatrix controller or remote I/O (out). It is calculated as follows:

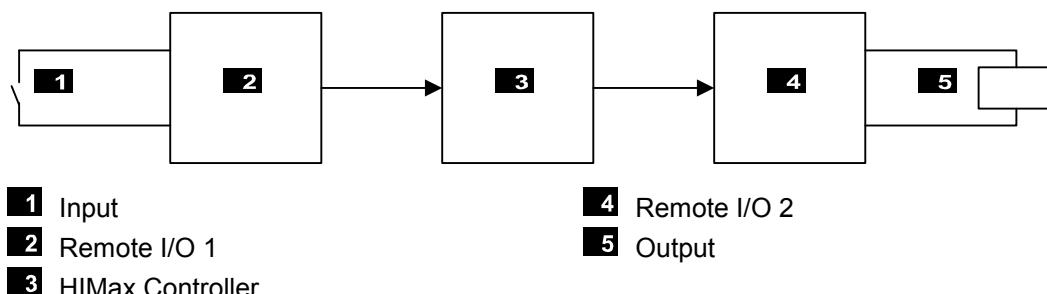


Figure 11: Reaction Time with Two Remote I/Os and One HIMax controller

$$T_R = t_1 + t_2 + t_3 + t_4 + t_5$$

$T_R$  Worst case reaction time

$t_1$  2 \* watchdog time of remote I/O 1

$t_2$  *ReceiveTMO1*

$t_3$  2 \* watchdog time of the HIMax controller.

$t_4$  *ReceiveTMO2*

$t_5$  2 \* watchdog time of remote I/O 2



Remote I/O 1 and Remote I/O 2 can also be identical. The time values still apply if a HIMax controller is used instead of a remote I/O.

#### 4.7.4 Calculating the Worst Case Reaction Time with two HIMax and one HIMatrix Controller

The worst case reaction time  $T_R$  is the time between a change on the sensor input signal (in) of the first HIMax controller and a reaction on the corresponding output (out) of the second HIMax controller. It is calculated as follows:

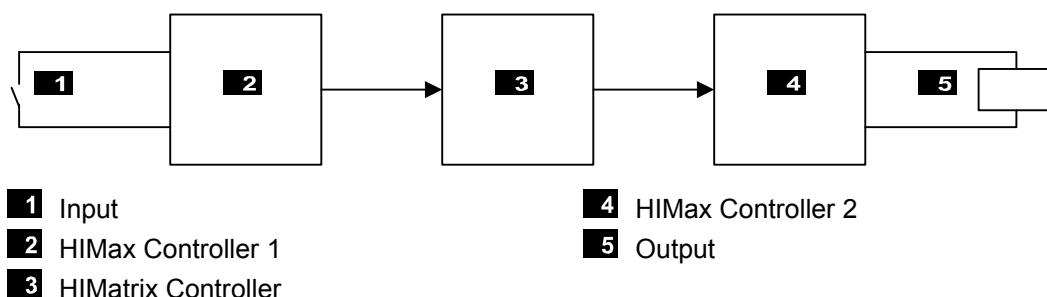


Figure 12: Reaction Time with Two HIMax Controllers and One HIMatrix controller

$$T_R = t_1 + t_2 + t_3 + t_4 + t_5$$

$T_R$  Worst case reaction time

$t_1$  Safety time of HIMax controller 1.

$t_2$  *ReceiveTMO1*

$t_3$  2 \* watchdog time of the HIMatrix controller

$t_4$  *ReceiveTMO2*

$t_5$  Safety time of HIMax controller 2.

- 1 HIMax controller 1 and HIMax controller 2 can also be identical.  
The HIMatrix controller can also be a HIMax controller.

#### 4.7.5 Calculating the Worst Case Reaction Time of Two HIMatrix Controllers

The worst case reaction time  $T_R$  is the time between a change on the sensor input signal of controller 1 and a reaction on the corresponding output of controller 2. It is calculated as follows:

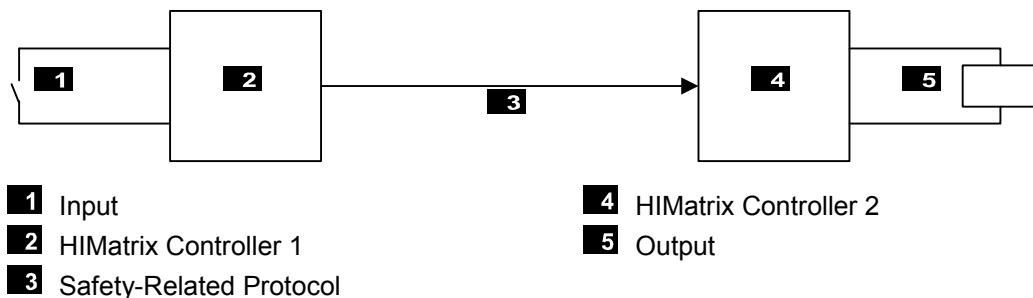


Figure 13: Reaction Time with Interconnection of Two HIMatrix Controllers

$$T_R = t_1 + t_2 + t_3$$

$T_R$  Worst case reaction time

$t_1$  2 \* watchdog time of the HIMatrix controller 1.

$t_2$  *ReceiveTMO*

$t_3$  2 \* watchdog time of the HIMatrix controller 2.

#### 4.7.6 Calculating the Worst Case Reaction Time with two Remote I/Os

The worst case reaction time TR is the time between a change on the sensor input signal (in) of the first HIMatrix controller or remote I/O (e.g., F3 DIO 20/8 01) and a reaction on the corresponding output of the second HIMatrix controller or remote I/O (out). It is calculated as follows:

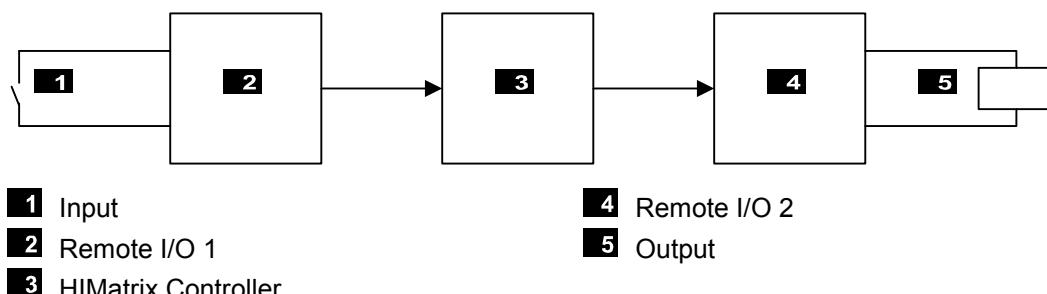


Figure 14: Reaction Time with Remote I/Os

$$T_R = t_1 + t_2 + t_3 + t_4 + t_5$$

$T_R$  Worst case reaction time

$t_1$  2 \* watchdog time of remote I/O 1

$t_2$  ReceiveTMO<sub>1</sub>

$t_3$  2 \* watchdog time of the HIMatrix controller.

$t_4$  ReceiveTMO<sub>2</sub>

$t_5$  2 \* watchdog time of remote I/O 2

Note: Remote I/O 1 and remote I/O 2 can also be identical. The time values still apply if a HIMatrix controller is used instead of a remote I/O.

#### 4.7.7 safeethernet Profile

safeethernet profiles are combinations of parameters compatible with one another that are automatically set when one of the safeethernet profiles is selected. When configuring, only the receive timeout and the expected response time parameters must be set individually. A safeethernet profile is used to optimize the data throughput within a network taking the physical conditions into account.

To ensure that the optimization is effective the following conditions must be met:

- The communication time slice value must be sufficiently high to allow all the safeethernet connections to be processed within one CPU cycle.
- average CPU cycle time < response time.
- average CPU cycle time < ProdRate or ProdRate = 0.



Unsuitable combinations of CPU cycle, communication time slice, response time and ProdRate are not rejected during code generation and download/reload. These combinations can cause communication disturbances until a failure of the safeethernet communication.

In the Control Panels of the two controllers, verify the *Bad Messages* and *Resends* values.

Six safeethernet profiles are available. Select the safeethernet profile most suitable for the transmission path.

HIMA recommends using the *Fast&Noisy*, *Medium&Noisy* or *Slow&Noisy* profile to ensure high availability of the safeethernet connection.

|                    |   |
|--------------------|---|
| Fast & Cleanroom   | Recommended only for noisy free network   |
| Fast & Noisy       | Recommended for high availability of the safeethernet connection.   |
| Medium & Cleanroom | Recommended only for noisy free network   |
| Medium & Noisy     | Recommended for high availability of the safeethernet connection.   |
| Slow & Cleanroom   | Recommended only for noisy free network   |
| Slow & Noisy       | Recommended for high availability of the safeethernet connection.   |
| Fixed              | Starting with V4, a modified calculation applies to all Cleanroom profiles. If a project created with a SILworX version prior to V4 should be converted, the configured profile must be set to <i>Fixed</i> to ensure that the CRC does not change. |

##### 4.7.7.1 safeethernet in a Noisy Network

The plant manufacturer and the operator must involve in their risk analysis the influence of the noisy network to the application.

The following settings are not recommended by HIMA due the possibly reduced availability:

- the configured ResponseTime is always or frequently not observed.
- and/or a cleanroom profile is used

Nevertheless, to use safeethernet under these conditions, the ReceiveTMO must be set such that the worst case response time for the safety function is also appropriate if twice the value of the ReceiveTMO would be used for calculating the worst case response time.

The factor  $n$  in  $\text{ResponseTime} \leq \text{ReceiveTMO} / n$ , where  $n > 4$ , can be for instance configured to increase the availability of the safe**ethernet** connection. The value of  $n$  depends on the availability actually required or necessary. The characteristics of the transmission system must be considered.

#### 4.7.8 Profile I (Fast & Cleanroom)



HIMA recommends using the *Fast&Noisy*, *Medium&Noisy* or *Slow&Noisy* profile to ensure high availability of the **safeethernet** connection.

The use of the Cleanroom profile is only recommended for networks free from interference, see chapter 4.7.7.1.

---

##### Use

The *Fast & Cleanroom* profile is suitable applications in ideal environments such as laboratories!

- For the fastest data throughput
- For applications requiring fast data transmission
- For application requiring a worst case reaction time as low as possible

##### Network requirements:

- Fast: 100 Mbit technology (100 Base TX), 1 Gbit technology
- Clean: Network free from interference. Prevent data from being lost due to network overload, external influences or network manipulation.
- LAN switches are necessary!

##### Communication path characteristics:

- Minimum delays
- Expected ResponseTime  $\leq$  ReceiveTMO  
(otherwise ERROR during configuration)

#### 4.7.9 Profile II (Fast & Noisy)

##### Use

The *Fast&Noisy* profile is the SILworX default profile for communicating via **safeethernet**.

- For fast data throughput
- For applications requiring fast data transmission
- For application requiring a worst case reaction time as low as possible

##### Network requirements:

- Fast: 100 Mbit technology (100 Base TX), 1 Gbit technology
- Noisy: Interference within the network.  
Low probability of data packet loss  
time for  $\geq 1$  resends
- LAN switches are necessary!

##### Communication path characteristics:

- Minimum delays
- Expected ResponseTime  $\leq$  ReceiveTMO / 2  
(otherwise ERROR during configuration)

#### 4.7.10 Profile III (Medium & Cleanroom)

##### NOTE



HIMA recommends using the Fast&Noisy, Medium&Noisy or Slow&Noisy profile to ensure high availability of the safeethernet connection.

The use of the Cleanroom profile is only recommended for networks free from interference, see chapter 4.7.7.1.

##### Use

The Medium & Cleanroom profile is only suitable for applications in a network free from interference and requiring a moderately fast data transmission rate.

- For medium data throughput
- Suitable for Virtual Private Networks (VPN) in which data is exchanged slowly but without faults since intermediate safety devices (e.g., firewalls, encryption) are used.
- Suitable for applications in which the worst case reaction time is not a critical factor

##### Network requirements:

- Medium: 10 Mbit (10 Base T), 100 Mbit (100 Base TX), 1 Gbit technology
- LAN switches are necessary!
- Clean: Network free from interference.  
Prevent data from being lost due to network overload, external influences or network manipulation.  
Time for  $\geq 0$  resends

##### Communication path characteristics:

- Moderate delays
- Expected ResponseTime  $\leq$  ReceiveTMO  
(otherwise ERROR during configuration)

#### 4.7.11 Profile IV (Medium & Noisy)

##### Use

The *Medium & Noisy* profile is suitable for applications that require moderate fast data transmission.

- For medium data throughput
- For applications requiring moderate fast data transmission
- Suitable for applications in which the worst case reaction time is not a critical factor

##### Network requirements:

- Medium: 10 Mbit (10 Base T), 100 Mbit (100 Base TX), 1 Gbit technology
- LAN switches are necessary!
- Noisy: Interference within the network.  
Low probability of data packet loss  
time for  $\geq 1$  resends

##### Communication path characteristics:

- Moderate delays
- Expected ResponseTime  $\leq$  ReceiveTMO / 2  
(otherwise ERROR during configuration)

#### 4.7.12 Profile V (Slow & Cleanroom)

##### NOTE



HIMA recommends using the *Fast&Noisy*, *Medium&Noisy* or *Slow&Noisy* profile to ensure high availability of the safeethernet connection.

The use of the Cleanroom profile is only recommended for networks free from interference, see chapter 4.7.7.1.

##### Use

The Slow & Cleanroom profile is suitable for applications in a network free from interference and requiring a slow data transmission rate.

- For slow data throughput
- For applications that only require a slow data transmission rate to controllers (potentially located far away) or if the communication path conditions cannot be defined in advance.

##### Network requirements:

- Slow: Data transfer via ISDN, dedicated line or radio relay.
  - Clean: Network free from interference.  
Prevent data from being lost due to network overload, external influences or network manipulation.
- Time for  $\geq 0$  resends

##### Communication path characteristics:

- Moderate delays
- Expected ResponseTime = ReceiveTMO  
(otherwise ERROR during configuration)

#### 4.7.13 Profile VI (Slow & Noisy)

##### Use

The Slow & Noisy profile is suitable for applications that only require a slow data transmission rate to the controllers (potentially located far away).

- For slow data throughput
- Generally for data transfer via bad telephone lines or disturbed radio relays.

##### Network requirements:

- Slow: Data transfer via telephone, satellite, radio etc.
- Noisy: Interference within the network.  
Low probability of data packet loss  
time for  $\geq 1$  resends

##### Communication path characteristics:

- Moderate to significant delays
- Expected ResponseTime  $\leq$  ReceiveTMO / 2  
(otherwise ERROR during configuration)

## 4.8 Cross-Project Communication

Cross-project communication is used to connect resources from various projects. The connection between resources is established via **safeethernet** and is configured in the **safeethernet** Editor.

A cross-project safeethernet connections can be established between

- two resources located in different SILworX projects, see 4.9.
- one resources located in SILworX and one resource located in ELOP II Factory, see 4.10.

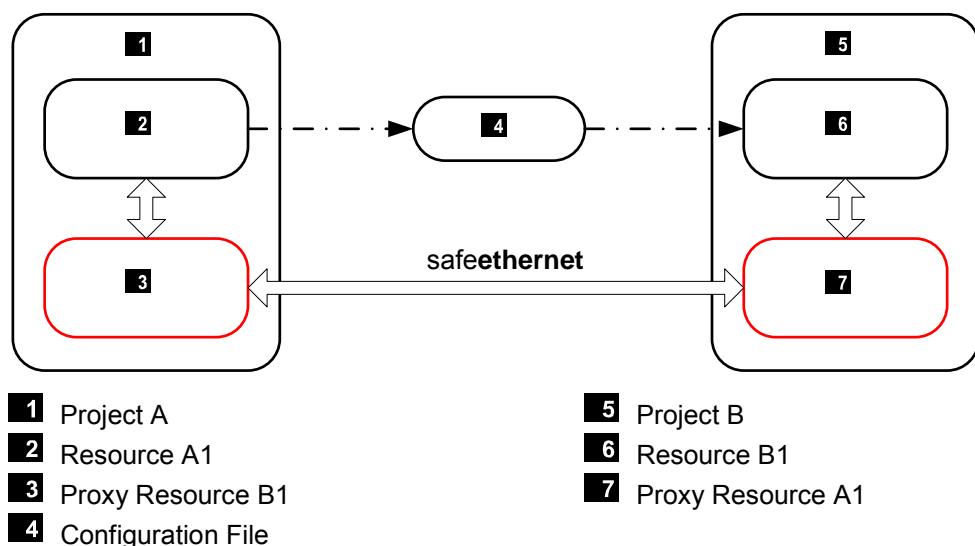


Figure 15: **safeethernet** Connection between Resource A1 in Project A and Resource B1 in Project B

The project in which the **safeethernet** connection is configured and the configuration file is created (archived) is referred to as project A.

The project to which the configuration file is imported (restored) is referred to as Project B.

The corresponding proxy resource serves as placeholder for the corresponding resource from the external project.

To maintain a clear structure in SILworX, HIMA recommends to create two configurations in both SILworX projects.

### Project A

- Config A
  - Resource A
- Config B
  - Resource B (as proxy)
- Global variables

### Project B

- Config B
  - Resource B
- Config A
  - Resource A (as proxy)
- Global Variables  
(global variables only results from restoring an archive)

#### 4.9 Cross-project communication SILworX<->SILworX

This example shows the configuration of the project cross communication between two HIMA controllers, which are located in two separate SILworX projects.

The **safeethernet** connection configured in the Project A must be archived and then restored in the Project B.

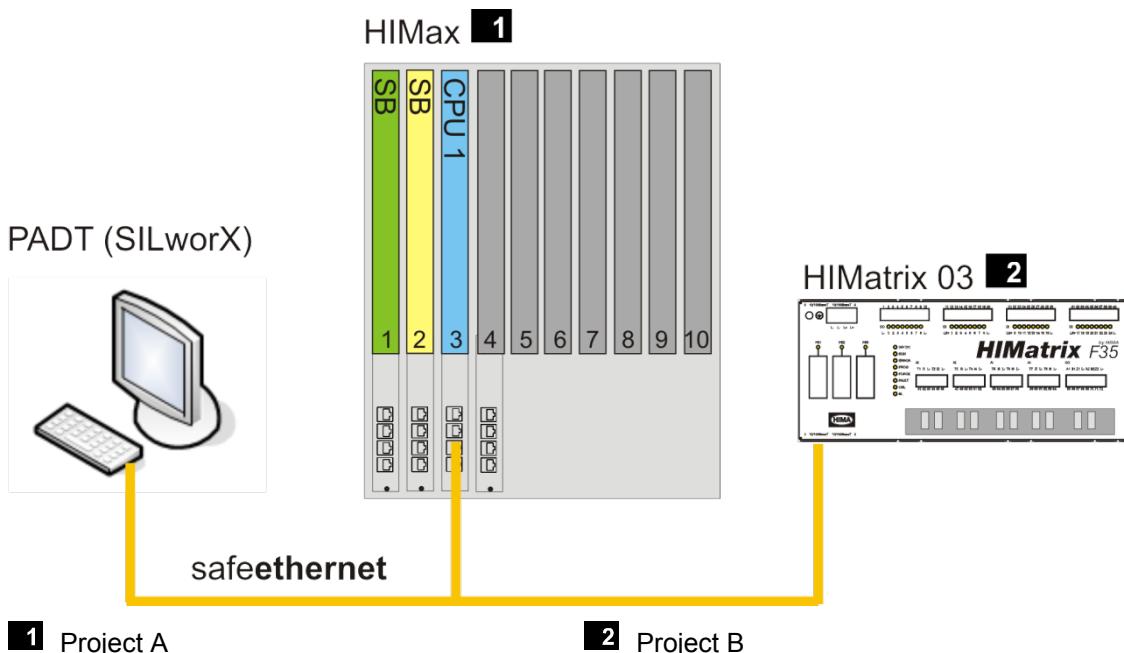


Figure 16: Cross-Project communication between two separate SILworX projects

For the configuration, the following parameters of both HIMA controllers must be determine:

- Name
- System ID [SRS]
- IP address

##### 4.9.1 Configuration B in Project A

Create separate Configuration B for Proxy Resource B in Project A.

###### 4.9.1.1 Creating the Proxy Resource B in Project A

A Proxy Resource B serves as placeholder for a Resource from an external Project B and is used for exchanging process data via **safeethernet**.

###### Creating the Proxy Resource:

1. Open project A, in which the proxy resource B should be created.
2. Right-click **Project A**, and then select **New, Configuration**.
  - A new configuration (Configuration B) is created.
3. Right-click **Configuration B**, and select **New, Proxy Resource SILworX** from the context menu.
  - A new proxy resource (Proxy Resource B) is created.

###### Configuring the Proxy Resource:

1. Right-click **Proxy Resource B**, and select **Properties**.
2. Enter a unique name in the **Name** field.  
Use the name of the Resource B in the Project B for the Proxy Resource B in the project A.

3. Readout the **system ID** from Project B and enter this system ID in Proxy Resource B.
4. Click **OK** to confirm.

**To open the structure tree for the Proxy Resource:**

1. Right-click **Hardware**, and then click **Edit**.
2. Select the resource type that is used in the Project B:
  - **HIMatrix 03 Proxy**
  - HIMatrix Proxy
  - HIMax System Proxy
3. Click **OK** to confirm. The Hardware Editor for the proxy resource appears.
4. For HIMatrix proxy 03, successively double-click **CPU-** and **COM module**, through which the redundant connection to the proxy resource is established.

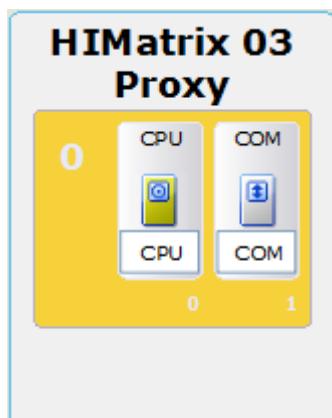


Figure 17: HIMatrix Proxy Resource

5. Enter the *IP addresses* and click **save**.
6. Repeat these steps for every further proxy resource contained in the project A.

#### 4.9.1.2 Create and archive global variables for the safeethernet Connection

**To create global variables for the safeethernet Connection:**

1. Right-click **Project A**, and then select **New, Global Variables**.  
 Objekt Globale Variable wird auf Projektebene angelegt.
2. Right click on **Global Variables**, and select **Edit** in the context menu to open the Variable Editor.
3. Right click onto a free space within the workspace of the variable editor and select **New Global Variable** from context menu to create a new global variable.
4. Repeat these steps for each additional New Global Variable for the safeethernet connection.
5. Add other the global variables *Connection State*, *Quality Channel 1* and *Quality Channel 2* for each transport direction.

**To archive the Global Variables:**

**TIP** When the *Global Variable* object already exists on project level in Project B, then the SILworX function Save Table Content as CSV can be used as an alternative. With the appropriate filter settings the required global variables can be exported selectively, see online help.

1. In structure tree, select **Project A, Global Variables**.
2. Select **Archive** from the context menu. The SILworX dialog box to archive an object appears.
3. Enter archive name for the Global Variables object in dialog box. Archive is saved with the file extension **\*.A3** in the selected archive folder.  
 The archived Global Variables object contains all global variables, created in the Project A.
4. Close Project A.

#### 4.9.1.3 Create a connection between the Resource A and the Proxy Resource B.

In the safeethernet Editor, create a safeethernet connection between the Resource A and the Proxy Resource B.

**To open the safeethernet Editor of the resource A**

1. In the structure tree, select **Configuration A, Resource A, safeethernet**.
2. Right-click **safeethernet**, then select **Edit**.  
 The new proxy resource B is created in the Object Panel.

**To create the safeethernet connection to the proxy resource:**

1. Drag the **Proxy Resource B** from the Object Panel onto a free space within the workspace of the safeethernet Editor.  
Select an appropriate name for this connection, immediately.
2. Select proper Ethernet interfaces **IF Kanalx** of the Resource and Proxy Resource.  
The following parameters determine the data throughput and the fault and collision tolerance of the safeethernet connection.
3. Select the **Profile** for the safeethernet connection (e.g., Fast&Noisy).
4. Calculate and enter **Receive Timeout** and **Response Time** (see Chapter 4.6.3 and Chapter 4.6.4).

#### 4.9.1.4 Connecting Process Variables

To add the process variables in the editor of the safeethernet connection.

**To open the connection editor:**

The safeethernet editor of Resource A is opened.

1. Right-click **Proxy Resource B** line and open context menu.
2. Select **Edit** from the context menu to open the connection editor of the safeethernet connection.
3. Select the **Resource A<->Proxy Resource B** tab.
4. Select the **Resource A->Resource B** area.
5. In the Object Panel, select a **Global Variable** for this transport direction and drag it onto the **Resource A->Resource B** column.
6. Repeat this step for every further variable.
7. Select the **Proxy Resource B->Resource A** area.

8. In the Object Panel, select a **Global Variable** for this transport direction and drag it onto the **Proxy Resource B->Resource A** column.
9. Repeat this step for further variables.
10. Add other the global variables *Connection State*, *Quality Channel 1* and *Quality Channel 2* for each transport direction.

#### 4.9.1.5 Archive safeethernet connection in Project A

The **safeethernet** connection configured in the Project A must be archived and then restored in the Project B.

Verifying the safeethernet Connection:

1. In the structure tree, select **Project A**, **safeethernet** and open context menu.
2. Select **Verification** from the context menu, and click **OK** to confirm.
3. Thoroughly verify the messages contained in the logbook and correct potential errors.

##### Archive the safeethernet connection

1. In the structure tree, select **Project A**, **safeethernet** and open context menu.
2. Select **Archive** from the context menu. The SILworX dialog box to archive an object appears.
3. Enter archive name for the **safeethernet** object in dialog box. Archive is saved with the file extension **\*.A3** in the selected archive folder.  
 All **safeethernet** connections contained in the **safeethernet** object are now archived.  
**safeethernet** connections can also be archived separately.
4. Close Project A.



The configuration of the **safeethernet** connection must be recompiled with the user program of the HIMax resource and transferred to the controller. The new configuration can only be used for communicating with the HIMax system upon completion of this step.

#### 4.9.2 Configuration A in Project B

Create separate Configuration A for Proxy Resource A in Project B.

The Project B looks now as the first project in SILworX. The resource from the first project is now the proxy resource.

#### 4.9.2.1 Creating the Proxy Resource A in Project B

A Proxy Resource A serves as placeholder for a Resource A from an external Project A and is used for exchanging process data via safeethernet.

##### Creating the Proxy Resource:

1. Open Project B, in which the Proxy Resource A should be created.
2. Right-click Project B, and then select **New, Configuration**.  
 A new configuration (Rename Configuration A) is created.
3. Right-click **Configuration B**, and select **New, Proxy Resource SILworX** from the context menu.  
 A new proxy resource (Proxy Resource A) is created.

### Configuring the Proxy Resource:

1. Right-click Proxy Resource A, and select **Properties**.
2. Enter a unique name in the **Name** field. Use the name of the Resource A in the Project A for the Proxy Resource A in the Project B.
3. Readout the **System ID** from Project A and enter this in Proxy Resource A.
4. Click **OK** to confirm.

### To open the structure tree for the Proxy Resource:

1. Right-click **Hardware**, and then click **Edit**, HIMatrix Proxy.
2. Select the resource type that is used in the Project A:
  - HIMatrix 03 Proxy
  - HIMatrix Proxy
  - **HIMax System Proxy**
3. Click **OK** to confirm. The Hardware Editor for the proxy resource appears.
4. Select **generic module** for HIMax system proxy and drag it to the base module on the proper slot, that is corresponding to the slot of the CPU / COM in the Project A.

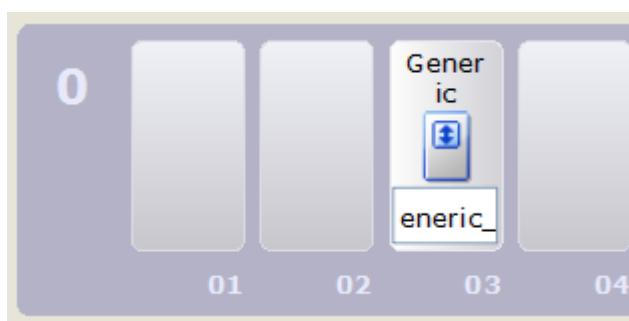


Figure 18: HIMax Proxy Resource

5. Double-click **Generic Module**, and enter the *IP address* of the CPU or COM module.
6. Click **save**.
7. Repeat these steps for every further Proxy Resource in Project B.

#### 4.9.2.2 Create and archive global variables for the safeethernet Connection

The same global variables as in the Project A must be created in the project B.

**TIP** When the global variable object already exists on project level, then the SILworX function Save Table Content as CSV can be used as an alternative, see online help.

### Restore global variables in Project B:

1. Right-click **Project**, and select **Restore**.
  - The SILworX dialog box to restore an object appears.
2. Open the archive directory and select the archived *Global Variables* object with the file extension **\*.A3**, that has been created in the Project A.
  - The restored Global Variables object contains all global variables, created in the Project A.

#### 4.9.2.3 Restore safeEthernet connection in Project B

##### Restore safeEthernet connection in Project B

1. Right-click **Project**, and select **Restore**.  
 The SILworX dialog box to restore an object appears.
2. Open the archive directory and select the archived **safeEthernet** object with the file extension **\*.A3**, that has been created in the Project A.  
 The restored **safeEthernet** object contains all connections between Ressource A and Proxy Resource B in Project B.

## 4.10 Cross-Project Communication SILworX<->ELOP II Factory

This example shows how to configure a safeethernet connection between a HIMax/HIMatrix controller with operating system V7 and higher (SILworX) and a HIMatrix controller with operating system prior to V7 (ELOP II Factory).

When connecting a SILworX Project A with the a ELOP II factory Project B, create a configuration file \*.prs in the Project A.

The Proxy Resource B must be created and configured in the Project A by the user. The configuration file contains the Resource A description for the safeethernet connection to the Resource B.

Once the configuration file has been imported, the Resource A is automatically created as Proxy Resource A in the ELOP II Factory Project B.

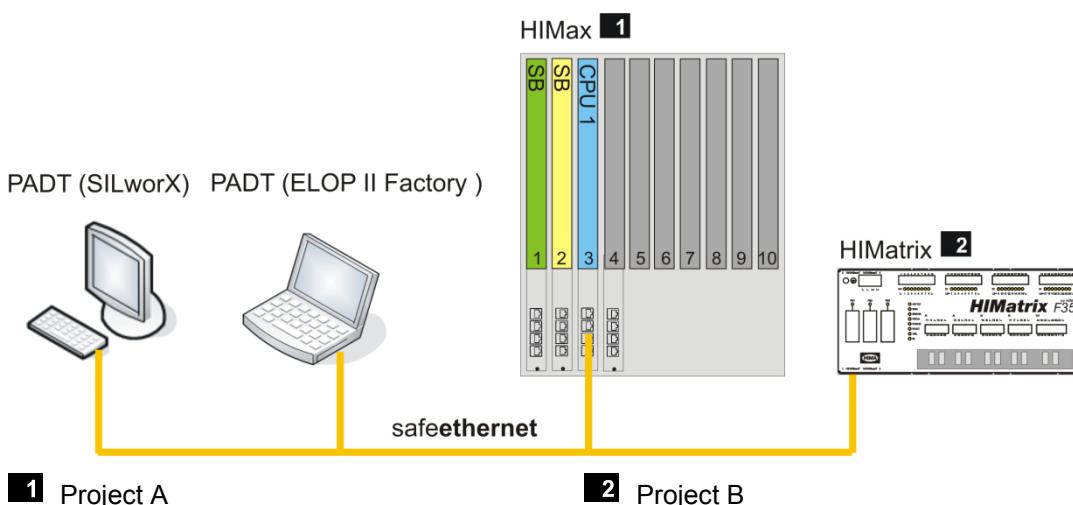


Figure 19: Cross-Project Communication between SILworX and ELOP II Factory

Open the resource in the Project B (HIMatrix) that should serve as proxy resource in the Project A (HIMax).

For this Resource B, determine the following parameters:

- Name
- System ID [SRS]
- Safety Time [ms]
- Watchdog Time [ms]
- IP Address



The resource properties are safety-relevant and are subjected to restrictions. For more information, refer to the safety manuals for the corresponding controller.

### 4.10.1 Configuration B in SILworX Project A

Create separate Configuration B for ELOP II Factory Proxy Resource B in Project A.

#### 4.10.1.1 Create ELOP II Factory Proxy Resource B in Project A.

A Proxy Resource B serves as placeholder for a Resource from an external Project B and is used for exchanging process data via safeethernet.

**Creating the Proxy Resource:**

1. Open Project A, in which the Proxy Resource B should be created.
2. Right-click **Project A**, and then select **New, Configuration**.  
 A new configuration (Configuration B) is created.
3. Right-click **Configuration B** and select **New, Proxy Resource ELOP II Factory**.  
 A new proxy resource (Proxy Resource B) is created.

**Configuring the Proxy Resource:**

Right-click Proxy Resource B, and select **Properties**.

4. Enter a unique name in the **Name** field.  
Use the name of the Resource B in the Project B for the Proxy Resource B in the project A.
5. Readout the **System ID** from Project B and enter this in Proxy Resource B.
6. Click **OK** to confirm.

**To open the structure tree for the Proxy Resource:**

1. Right-click **Hardware**, and then click **Edit, HIMatrix Proxy**.
2. Select the resource type that is used in the Project B:
  - HIMatrix 03 Proxy
  - **HIMatrix Proxy**
  - HIMax System Proxy
3. Click **OK** to confirm. The Hardware Editor for the proxy resource appears.
4. For HIMatrix Proxy, successively double-click **COM module**, through which the connection on the proxy resource is established.



Figure 20: HIMatrix Proxy Resource

5. Enter the *IP address* of the Proxy Resource and click **save**.
6. Repeat these steps for every further proxy resource contained in the project A.

#### 4.10.1.2 Create a connection between the Resource A and the Proxy Resource B.

In the safeethernet Editor, create a safeethernet connection between the Resource A and the Proxy Resource B.

**To open the safeethernet Editor of the resource A**

1. In the structure tree, select **Configuration A, Resource A, safeethernet**.
2. Right-click **safeethernet**, then select **Edit**.  
 The new Proxy Resource B is created in the Object Panel.

**To create the safeethernet connection to the proxy resource:**

1. Drag the **Proxy Resource B** from the Object Panel onto a free space within the workspace of the **safeethernet** Editor.  
Select an appropriate name for this connection, immediately.
2. Select proper Ethernet interfaces **IF Kanalx** of the Resource and Proxy Resource.  
The following parameters determine the data throughput and the fault and collision tolerance of the **safeethernet** connection.
3. Select **Profile** for the **safeethernet** connection (e.g., Fast&Noisy).
4. Calculate and enter **Receive Timeout** and **Response Time** (see Chapter 4.6.3 and Chapter 4.6.4).

#### 4.10.1.3 Connecting Process Variables

To add the process variables in the editor of the **safeethernet** connection.

**To open the connection editor:**

The **safeethernet** editor of Resource A is opened.

1. Right-click **Proxy Resource B** line and open context menu.
2. Select **Edit** from the context menu to open the connection editor of the **safeethernet** connection.
3. Select the **Resource A<->Proxy Resource B** tab.
4. Select the **Resource A->Proxy Resource B** area.
5. In the Object Panel, select a **Global Variable** for this transport direction and drag it onto the **Resource A->Proxy Resource B** column.
6. Repeat this step for every further variable.
7. Select the **Proxy Resource B->Resource A** area.
8. In the Object Panel, select a **Global Variable** for this transport direction and drag it onto the **Proxy Resource B->Resource A** column.
9. Repeat this step for further variables.
10. Add other the global variables *Connection State*, *Quality Channel 1* and *Quality Channel 2* for each transport direction.

**Verifying the safeethernet Connection:**

1. In the structure tree, select **Project A, Configuration A, Resource A, safeethernet** and open context menu.
2. Right-click and select **Verification** from the context menu. Click **OK** to confirm.
3. Thoroughly verify the messages contained in the logbook and correct potential errors.



Recompile the resource configuration with the **safeethernet** connection, and load it to the controllers. The new configuration can only be used for communicating with HIMax system after this step.

#### 4.10.1.4 Exporting the Configuration File from SILworX

The **safeethernet** connection configured in SILworX must be exported as configuration file with the extension **\*.prs**. This configuration file can be imported to ELOP II Factory to establish the **safeethernet** connection for the HIMatrix controller.

##### To export a safeethernet connection:

1. Select **Proxy Resource** in the safeethernet Editor.
2. Right-click and select **Export Connection to Proxy Resource** from the context menu.  
☒ A standard dialog box for saving a file appears.
3. Enter a file name for the configuration file and save it with the extension **\*.prs**.
4. Close Project A.

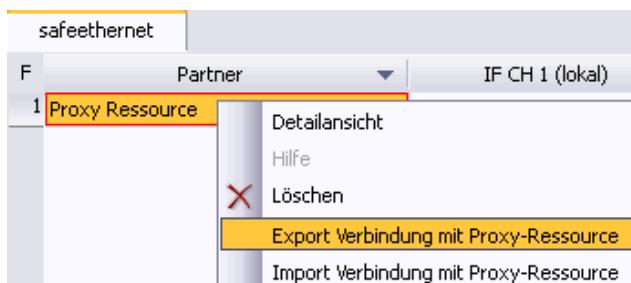


Figure 21: safeethernet Connection Export

#### 4.10.2 Configuring a HIMatrix in an ELOP II Factory Project

**To import the configuration file to the ELOP II Factory Project B:**

1. Start ELOP II Factory.
  2. Open Project B to which the configuration file should be imported.
  3. In the structure tree, select Resource B and open the context menu.
  4. Select **Import Connections** from the context menu. A dialog box for importing a file with the extension **\*.prs** appears.
  5. Select the configuration file created in the Project A, and click **OK**.
- Once the configuration file has been imported, the Resource A is automatically created as Proxy Resource A in the Project B.



Figure 22: Importing Connections in ELOP II Factory

##### 4.10.2.1 Assigning ELOP II Factory Process Signals

Connect process signals in the ELOP II Factory Resource B.

Select **Signals, Editor** on the menu bar to open the **Signal Editor**.

**To open the ELOP II Factory Peer-to-Peer Editor for the Resource B:**

1. In the structure tree, open **Configuration, Resource, Peer-to-Peer-Editor**.
2. Enter the **HH Network** for this connection in the **Peer-to-Peer-Editor**.
3. In the **Peer-to-Peer-Editor**, click **Connect Process Signals**.

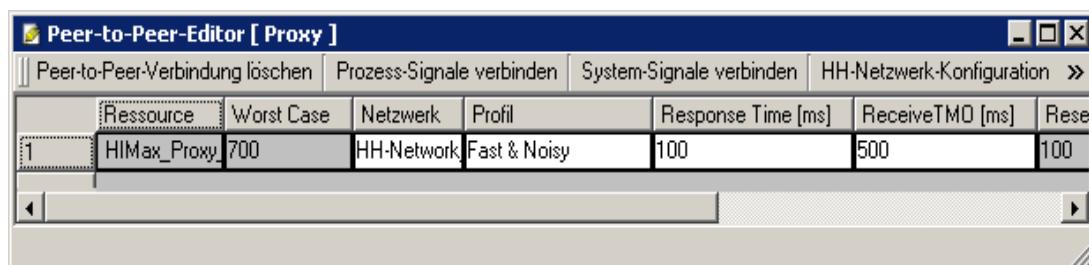


Figure 23: Peer-to-Peer-Editor Editor in ELOP II Factory



Note that both communication partners must use the same profile and the same settings (automatically adopted while importing the configuration file).

### To assign Peer-to-Peer signals:

The **Signal Editor** is opened

1. Select the **HIMatrix Resource B->HIMax Proxy Resource A** area.
2. Select a **process signal** for this transport direction from the *Signal editor* and drag the process signal to the dialog box *Peer-to-Peer-Signals* to a proper **signal**.
3. Repeat this step for further Peer to Peer Signals of this transport direction.

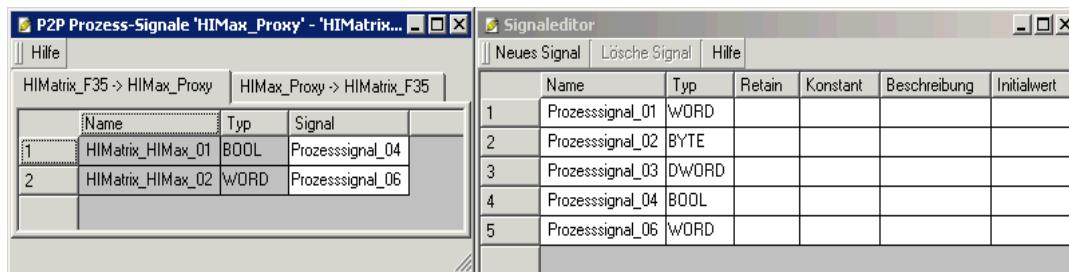


Figure 24: Transport direction HIMax Proxy Resource A->HIMatrix Resource B area.

4. Select the **HIMax Proxy Resource A->HIMatrix Resource B** area.
5. Select a **process signal** for this transport direction from the *Signal editor* and drag the process signal to the dialog box *Peer-to-Peer-Signals* to a proper **signal**.
6. Repeat this step for further Peer to Peer Signals of this transport direction.

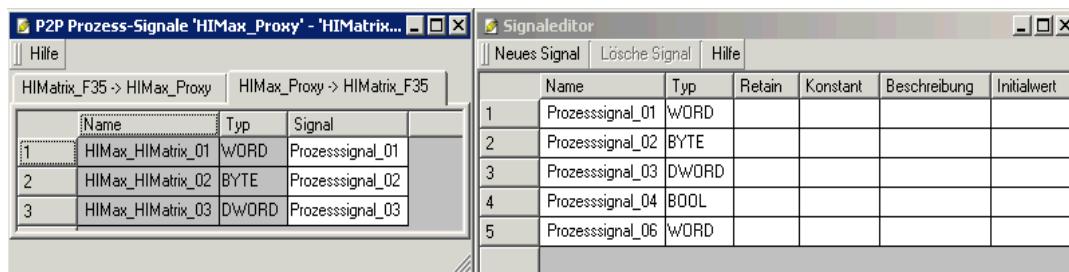


Figure 25: Transport direction HIMax Proxy Resource A->HIMatrix Resource B.



For more information on how to connect process signals in ELOP II Factory, refer to the ELOP II Factory online help.



The configuration of the P2P connection and HIMatrix user program must be once again compiled and loaded to the controller. Only after this step, the P2P connection is active for the HIMatrix system.

## 4.11 Control Panel (safeethernet)

The Control Panel can be used to verify and control the safeethernet connection settings. Details on the current status of the safeethernet connection (e.g., cycle time, bus status, etc.) are displayed.

### To open Control Panel for monitoring the safeethernet connection:

1. Select a **Resource** in the structure tree.
2. Right click and select **Online** from context menu.
3. In the **System Log-in** window, enter the access data to open the Control Panel for the resource.
4. In the structure tree associated with the Control Panel, select **safeethernet**.

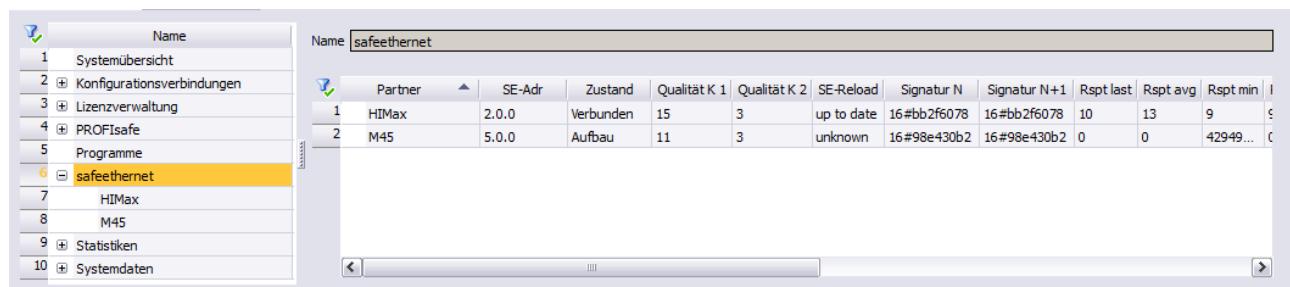


Figure 26: Control Panel for Connection Control

### To reset the statistical data of the safeethernet connection

This function is used to reset the statistical data (cycle [min], cycle [max], etc.) to zero.

1. In the structure tree, select the safeethernet connection.
2. Right-click and select **Reset safeethernet Statistics** from the context menu.

### 4.11.1 View Box (safeethernet Connection)

The view box displays the following values of the selected safeethernet connection:

| Element      | Description  |
|--------------|--|
| Partner      | Resource name of the communication partner                     |
| Address      | SRS<br>System.Rack.Slot  |
| State        | State of the safeethernet connection<br>(See also Chapter 4.4) |
| Quality Ch 1 | State of transmission path Ch1<br>See also Chapter 4.4.        |
| Quality Ch 2 | State of transmission path Ch2<br>See also Chapter 4.4.        |

| Element       | Description  |  |
|---------------|--|--|
| Reload        | <b>safeethernet</b> Reload Status<br>unknown      Version state is unknown:<br>-no connection is established<br>-no V6 connection<br>updated: <i>Partner must be updated</i><br>outdated:     Own configuration must be updated<br>uptodate     Both partners are operating with the current configuration |  |
| Signature N   | Previous signature of the <b>safeethernet</b> configuration.   |  |
| Signature N+1 | Current signature of the <b>safeethernet</b> configuration.  |  |
| Rsp t last    | Actual response time as minimum, maximum, last and average value. See also Chapter 4.6.4.  |  |
| Rsp t avg     |  |  |
| Rsp t min     |  |  |
| Rsp t max     |  |  |
| Faults        | Bad Messages<br>Number of rejected messages since statistics reset.  |  |
| Rsnd          | Number of resends since statistics reset [UDINT].  |  |
| Succeeded     | Number of successful connections since statistics reset.   |  |
| Early         | Early Queue Usage<br>Number of messages stored in the memory since statistics reset. See also Chapter 4.6.9.   |  |
| Frame         | Frame No.<br>Revolving send counter.   |  |
| Ack Frame     | Ack.Frame No.<br>Revolving receive counter   |  |
| Monotony      | Revolving user data send counter   |  |
| Rcv TMO       | Receive Timeout [ms]<br>(See also Chapter 4.6.3)   |  |
| Rsnd TMO      | Resend Timeout [ms]<br>(See also Chapter 4.6.6)  |  |
| Ack TMO       | Acknowledge Timeout [ms]<br>(See also Chapter 4.6.7)   |  |
| Conn Ctrl     | Connection Control   |  |
| Ctrl Ch 1     | Transmission Control Ch1<br>See also Chapter 4.4.  |  |
| Ctrl Ch 2     | Transmission Control Ch2<br>See also Chapter 4.4.  |  |
| Protocol      | 0-1      Previous protocol version for HIMatrix with CPU operating system up to V7<br>2          New protocol version for HIMax and HIMatrix with CPU operating system higher than V7  |  |

Table 37: View Box of the **safeethernet** Connection

## 4.12 safeEthernet Reload

Thanks to this feature, changes performed to a safeEthernet configuration can be loaded during operation by performing a reload while the safeEthernet connection continues to run with no interruptions.

### 4.12.1 Requirements

safeEthernet reload is supported for HIMax, HIMatrix F\*03 and HIMatrix M45. The following system requirements apply for all controllers participating to the safeEthernet:

- HIMax CPU OS V6 and higher, and COM OS V6 and higher
- HIMatrix M45 and F\*03 CPU OS V10 and higher, and COM OS V14 and higher

The current COM OS V6 is required to ensure that safeEthernet connections are properly routed via the X-COM module, see Chapter 4.12.8.

In the properties of the safeEthernet connection, set the *Codegen* parameter to **V6 and higher**.



If a redundant module is available, the operating systems of HIMax modules can be updated during operation. This ensures that the conversion to safeEthernet reload is performed without interruptions, even in all HIMax plants using previous operating systems.

### 4.12.2 Technical Concept

The safeEthernet signature is a CRC code used to uniquely identify the safeEthernet configuration. The safeEthernet signature is created during the code generation and is part of the loaded configuration.

safeEthernet communication between communication partners can only occur if both partners have the same safeEthernet configuration with identical signature.

To use reload to perform changes to a safeEthernet connection, the controller must be provided with two safeEthernet configurations and corresponding signatures (N and N+1). This is supported for SILworX V6 and higher.

In the two controllers, configuration E1 is connected to a safeEthernet signature



After the second reload for controller 1, configurations E1 and E2 are available. The previous safeEthernet configuration E1 with signature N continues to be active in controller 1.

Changing the safeEthernet configuration results in a dual configuration (in the example: E1+E2). The safeEthernet version state of controller 1 is **updated** and the version state of controller 2 is **outdated**, which signalizes that a reload must be performed in controller 2.



1) safeEthernet Version State, see Chapter 4.12.6

Upon completion of the reload process for controller 2, the new **safeethernet** configuration E2 is active with signature N+1. The dual configuration (E1+E2) is now available for both controllers and should be deleted as recommended by performing an additional reload, see Chapter 4.12.4.1.



<sup>1)</sup> **safeethernet** Version State, see Chapter 4.12.6

- i** With respect to reload, HIMA recommends to always start the process for the controller that is configured as *Timing Master* of the **safeethernet** connection. The new **safeethernet** connection becomes active after completion of the reload procedure for both controllers.

#### 4.12.3 Changes to the **safeethernet** configuration

The following table provides an overview of the changes to the **safeethernet** configuration and their effects on the **safeethernet** reload.

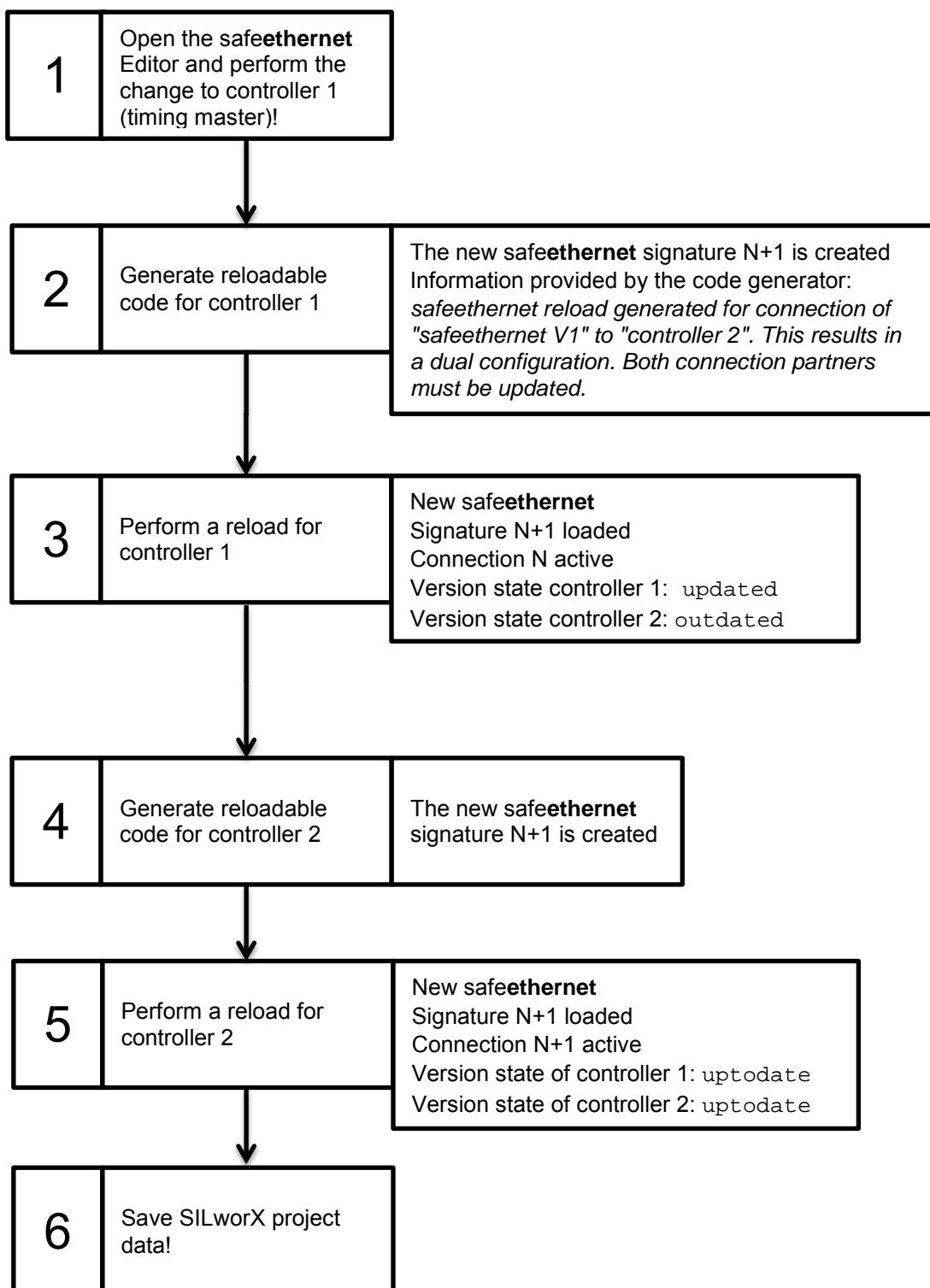
| Changes to  | CPU  | COM             | X-OPC          |
|---|------|-----------------|----------------|
| Adding or deleting global variables for <b>safeethernet</b>   |      |                 |                |
| <b>safeethernet</b>   | •    | •               | <sup>-2)</sup> |
| X-OPC (DA)  | •    | •               | <sup>-2)</sup> |
| X-OPC (events)  | •    | •               | <sup>-2)</sup> |
| To change the number of views (X-OPC)   | •    | •               | <sup>-2)</sup> |
| Adding or deleting a new <b>safeethernet</b> connection   | •    | • <sup>1)</sup> | <sup>-2)</sup> |
| <b>safeethernet</b> parameters, e.g., timing master, receive timeout  | •    | •               | <sup>-2)</sup> |
| IP addresses (change to the transmission path)  | •    | • <sup>1)</sup> | <sup>-2)</sup> |
| Changes to the standard protocols on the COM  | n.a. | • <sup>1)</sup> | n.a.           |
| <b>safeethernet</b> parameters ( <i>profile</i> )   | -    | n. a.           | n. a.          |
| <b>safeethernet</b> parameters ( <i>behavior</i> )  | -    | n. a.           | n. a.          |
| <ul style="list-style-type: none"> <li>• <b>safeethernet</b> reload possible</li> <li>- <b>safeethernet</b> reload not possible</li> </ul>              |      |                 |                |
| n.a.: not applicable  |      |                 |                |
| <sup>1)</sup> Only in connection with <i>cold reload</i> , i.e., with a stopped communication module.   |      |                 |                |
| <sup>2)</sup> The X-OPC configuration can be modified on the CPU/COM by performing a reload, the X-OPC server must be updated by performing a download. |      |                 |                |

Table 38: **safeethernet** Reload after Changes

#### 4.12.4 Procedure to be observed

safeEthernet connections are to be considered holistically, i.e., changes should always be performed on both partners and in direct succession to ensure the consistency of the safeEthernet.

The previous safeEthernet configuration is active up to step 4. The new safeEthernet configuration becomes active after a successful reload in step 5.



#### 4.12.4.1 Align signatures N and N+1

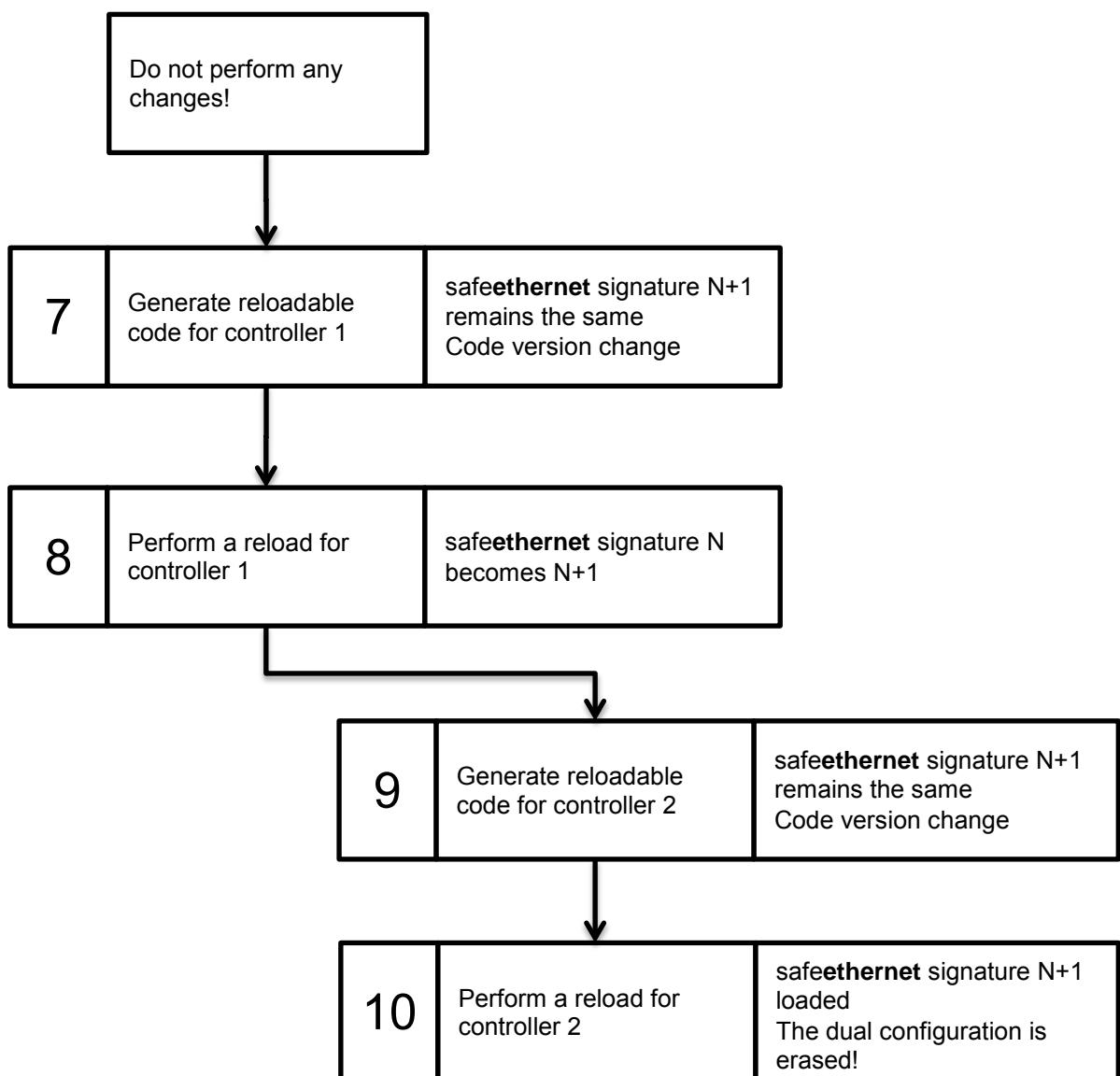
Changes to the **safeethernet** configuration such as described in Chapter 4.12.4 result in a dual configuration. The controllers contains the two following configurations:

- The previous configuration with **safeethernet** signature N through which **safeethernet** communication is running, remains active until both controllers have been updated.
- The new configuration with **safeethernet** signature N+1 through which **safeethernet** communication is running, becomes active after both controllers have been updated.

After completion of a new code generation without **safeethernet** changes, a reload must be performed. This deletes the dual configuration. At this point, only one configuration with one **safeethernet** signature exists (i.e., the same CRC code is set in the *Signature N* and *Signature N+1* system variables).

- i** A dual configuration is part of the configuration file and results in a changed code version! Therefore, HIMA recommends to always erase the dual configuration.

To erase a dual configuration, perform the step 7 through 10 such as described below!



#### 4.12.5 Integrated Protective Mechanisms

The protective mechanisms integrated in SILworX and in the controller's operating system ensure early detection of unintended interruption or resumption of a safeEthernet connection and generate a warning message.

##### 4.12.5.1 Automatic Test during Code Generation

The following table contains the messages output during a code generation and connected to safeEthernet reload and informing the user about the current safeEthernet version state.

| Information in the code generator dialog  | Description  |
|---|--|
| <i>safeEthernet reload generated for connection of "safeEthernet V1" to "controller2". This results in a dual configuration. Both connection partners must be updated.</i>  | <b>Procedure OK!</b><br>This information is provided after a change performed to the safeEthernet connection and the code generation!<br>Follow the recommended procedure, see Chapter 4.12.2  |
| <i>Dual configuration safeEthernet reload for connection of "safeEthernet V1" to "controller2" has been removed.</i>  | <b>Procedure OK!</b><br>A reload has been performed after completion of a new code generation without any safeEthernet change. The dual configuration has been erased, i.e., there is once again only one configuration with one safeEthernet signature, see Chapter 4.12.4.1.                   |
| <i>The safeEthernet connection of "safeEthernet V1" to "controller2" could be interrupted. Please update this partner. No matching connection version could be found in the partner's download configuration.</i> | <b>Caution!</b><br>Do not perform any reload to ensure that the connection will not be interrupted! Please contact HIMA technical support!<br>With the partner controller, there is no longer a common configuration with identical signature such that no safeEthernet reload can be performed. |

Table 39: Messages from the Code Generator

##### 4.12.5.2 Automatic Test during the Controller's Reload

Warning messages are only output before a safeEthernet reload if suitable CPU operating systems are loaded in the controllers.

- HIMax CPU OS V6 and higher
- HIMatrix M45 and F\*03 CPU OS V10 and higher

Prior to performing a reload, the operating system checks the safeEthernet version state to ensure that it is suitable for a reload. If a controller detects that a reload could result in the interruption of the safeEthernet connection, it generates a corresponding warning message displayed in SILworX. In this scenario, reload can be aborted by the user. After an aborted reload, the controllers continue to operate with the last suitable safeEthernet configuration.

| Information in the Dialog Box   | Description  |
|---|--|
| <i>A reload is to be performed although a safeEthernet connection reports the version state updated, partner's safeEthernet address: x/x/x. The connection might be lost by activating the configuration. Check for potential consequences.</i>   | <b>Caution!</b><br>Do not perform a reload. Please contact HIMA technical support!<br>If reload is performed anyway, the safeEthernet connection can be interrupted!   |
| <i>A reload is to be performed although a safeEthernet connection reports the version state unknown (i.e., no connection exists to the partner), partner's safeEthernet address: x/c/x. If a connection is established before the configuration has been activated, the configuration activation could cause the connection to be lost again. Check for potential consequences.</i> | <b>Caution!</b><br>The <i>unknown</i> version state is reported, if a safeEthernet connection is interrupted, see Chapter 4.12.6. Prior to performing a new reload, check the physical connection, e.g., if all the Ethernet cables are properly plugged in. |

Table 40: Messages from the Operating System

#### 4.12.6 safeEthernet Version State

The version state provides information about the current state of the **safeEthernet** connection and about whether suitable **safeEthernet** configurations are loaded or are to be loaded. The consistent procedure applied to **safeEthernet** reload is a requirement for ensuring that the version state is properly displayed, see Chapter 4.12.4.

The following **safeEthernet** version state is displayed:

|             |  |
|-------------|--|
| unknown     | Version state is unknown:<br>- No connection is established<br>- No V6 connection<br>- No suitable <b>safeEthernet</b> signature available |
| updated:    | Partner must be updated  |
| outdated:   | Own configuration must be updated  |
| up-to-date: | Both partners are operating with the current configuration   |

If no suitable configuration is available after a reload, a warning is output informing the user that the reload can be aborted.

If, however, the reload process is continued in spite of the warning messages described in Chapter 4.12.5, a suitable configuration could no longer be present in the partner controller. The **safeEthernet** connection to the partner controller could be interrupted (CLOSED)!

The **safeEthernet** version state is called *Reload* in the SILworX Online View of the **safeEthernet** connection. The same information is provided by the *Version State* system variable, which can be assigned a global variable and used as such in the user program.

#### 4.12.7 Maximum Number of safeEthernet Connections during Reload

During the reload process, the number of **safeEthernet** connections contained in the controller can be greater than configured. Not only the added **safeEthernet** connections are maintained, but also the deleted **safeEthernet** connections, since they must remain active until the reload is completed.

The maximum number of simultaneous **safeEthernet** connections during reload is as follows:

- For HIMax: 300 (max. 255 **safeEthernet** connections + 45 (reload buffer))
- For HIMatrix F\*03/M45: 150 (max. 128 **safeEthernet** connections + 22 (reload buffer))

These limits are defined to restrict the maximum storage space required during a reload.



If during the reload code generation, the maximum number of **safeEthernet** connections allowed for reload is exceeded, the reload code generation is aborted and an error message is output. For the maximum number of **safeEthernet** connections between two controllers refer to Table 26.

If multiple changes are required, these must be performed through multiple consecutive reloads.

#### 4.12.8 safeEthernet Connection via the Communication Module

HIMA recommends to set the *Code Generation* parameter of the communication module to *V6 and higher* to prevent, as far as possible, a cold reload of the communication module. In doing so, the **safeEthernet** connections routed through this communication module are not interrupted, even if changes are performed to variables or parameters, e.g., profiles.

For more details on the **safeEthernet** reload behavior in connection with the communication module and additional changes, refer to Chapter 4.12.3.

## 5 PROFINET IO

PROFINET IO is the transfer protocol provided by PNO Germany and is based on Ethernet technology.

With PROFINET IO, such as with PROFIBUS DP, the remote field devices are integrated in SILworX via a device description (GSDML file).

The HIMA PROFINET IO controller complies with Conformance Class A and supports non-real time (NRT) and real time (RT) communication with the PROFINET IO devices. In particular, real time communication is used for time critical data exchange and non-real time communication for non time critical processes, such as acyclic read/write operations.

A redundant PROFINET IO connection can only be implemented by configuring a second PROFINET IO controller/device and adjusting it in the user program.

### 5.1 PROFINET IO Function Blocks

To acyclically exchange data, function blocks with the same functionality as with PROFIBUS DP are available in SILworX.

The following PROFINET IO function blocks are available:

| Function block | Function description                                    |
|----------------|---|
| MSTAT 6.9.1    | Controlling the controller state using the user program |
| RALRM 6.9.2    | Reading the alarm messages of the devices               |
| RDREC 6.9.4    | Reading the data records of the devices                 |
| SLACT 6.9.5    | Controlling the device states using the user program    |
| WRREC 6.9.6    | Writing the data records of the devices                 |

Table 41: Overview of PROFINET IO Function Blocks

The PROFINET IO function blocks are configured such as the PROFIBUS DP function blocks, see Chapter 6.9.

## 5.2 Controlling the Consumer/Provider Status (IOxS)

The system variables described in this chapter, the consumer/provider status (IOxS) can be controlled via the user program. If the consumer/provider status should not be controlled via the user program, the output variables must be assigned a constant set to TRUE. The statuses are then set to GOOD as soon as the communication module obtained valid process values from the processor module.

The following picture shows how system variables are exchanged between the HIMA controller and a DO device or a DI device, respectively.

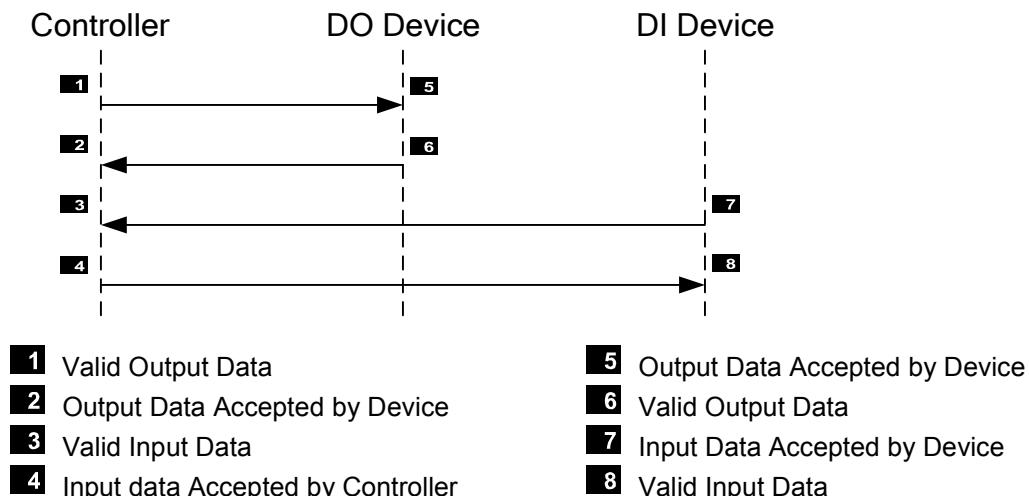


Figure 27: Controlling the Consumer/Provider Status (IOxS)

### 5.2.1 Control Variables in the HIMA Controller

The *Valid Output Data* **1** and *Input Data Accepted by Controller* **4** input variables can be used to control the consumer/provider status (IOxS) via the user program.

The *Output Data Accepted by Device* **2** and *Valid Input Data* **3** input variables can be used to read the consumer/provider status (IOxS) via the user program.

### 5.2.2 Control Variables in the HIMA DO Device

The *Valid Output Data* **6** output variable can be used to control the consumer/provider status (IOxS) via the user program.

The *Output Data Accepted by Controller* **5** input variable can be used to read the consumer/provider status (IOxS) via the user program.

### 5.2.3 Control Variables in the HIMA DI Device

The *Input Data Accepted by Device* **7** output variable can be used to control the consumer/provider status (IOxS) via the user program.

The *Valid Input Data* **8** input variable can be used to read the consumer/provider status (IOxS) via the user program.

### 5.3 PROFIsafe

The PROFIsafe specification from the PNO is deemed to be known!

PROFIsafe uses the PROFINET protocol to transfer safety-related data up to SIL 3, based on Ethernet technology.

The PROFIsafe protocol is superposed the PROFINET protocol and contains safe user data as well as data backup information. The safe PROFIsafe data is transferred together with the non-safety-related PROFINET data via the subsidiary PROFINET protocol.

As with the black-channel principle, PROFIsafe uses unsafe data transfer channels (Ethernet) to transfer safe data. This allows the F host and F device to exchange safe PROFIsafe data.

#### PROFIsafe in connection with HIMA controllers

According to the PROFIsafe specification, the F-Host repeatedly sends a message packet until the F-Device acknowledges receipt to the F-Host. Only then does the F-Host send a new message packet to the F-Device.

The current process value is sent in each repeated PROFIsafe message packet. It can thus happen that the same process signal has different values in the repeated message packets.

In HIMA devices, PROFIsafe is implemented on the receiver side such that process values can only be adopted when the message packet is received for the first time. The process values of the resent message packets (with identical progressive number of the message packet) are rejected.

If the connection is lost and  $F\_WD\_Time$  has expired, the PROFIsafe process values adopts their initial values.

A given process value is only received on the opposite side (F-Host/F-Device) if the process value remains unchanged for at least the following time:

$$2 * F\_WD\_Time + F\_WD\_Time2$$

The PROFIsafe system must be configured to ensure the SFRT (Safety Function Response Time) is suitable for performing the corresponding safety function.

Calculation formula, see Chapter 5.3.3.



The following conditions must be met to ensure a behavior consistent with PROFIsafe:

- The initial values of the process value variables must be set to 0.
  - The *AutoAwaitFParamsOnConnLoss* parameter must be deactivated, see Chapter 5.11.
- The SILworX default settings are configured to ensure a behavior consistent with PROFIsafe!
-

### 5.3.1 PROFIsafe Control Byte and Status Byte

Both system variables Control Byte and Status Byte are contained in each PROFIsafe submodule and are exchanged during communication between F-Host and F-Device, see Chapter 5.7.6.3 and Chapter 5.11.4.

The PROFIsafe control byte is written in the F-Host and read in the F-Device.  
The PROFIsafe status byte is written in the F-Device and read in the F-Host.

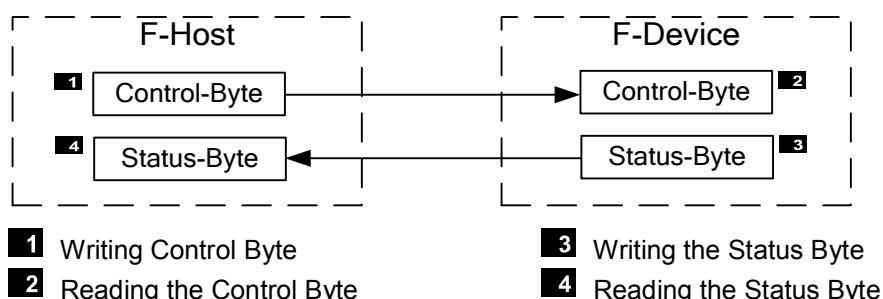


Figure 28: PROFIsafe Control Byte and Status Byte



In the HIMA's systems, the system variables Control Byte and Status Byte have additional features deviating from the PROFIsafe specification, see Table 51 and Table 68: Edit Dialog Box for the Input Submodule.

### 5.3.2 F\_WD\_Time (PROFIsafe watchdog time)

The following inequation applies to functional PROFIsafe connection between a HIMA F-Host and a F-Device:

$$\begin{aligned}
 F\_WD\_Time > \\
 & 3 * CPU \text{ cycle time} * \text{number of communication time slices} + \\
 & 2 * \text{PROFINet controller production interval}^1 + \\
 & 1 * DAT (\text{F-Device acknowledgement time}) + \\
 & 2 * \text{internal F-Device bus time} + \\
 & 2 * \text{PROFINet device production interval}^1 + \\
 & 2 * \text{Ethernet delay}
 \end{aligned}$$

<sup>1)</sup>PROFINet controller and PROFINet device production intervals are generally identical and are calculated as follows: reduction factor \* send clock factor \* 31.25  $\mu$ s



Refer to the device description provided by the F-Device manufacturer for the values of DAT (F-device acknowledgement time) and the internal device bus time!

For HIMax and HIMatrix L3 F-Devices,  $DAT = DAT_{out} = DAT_{in} = 2 * WDT \text{ CPU}$

### 5.3.2.1 Remarks about F\_WD\_Time (PROFIsafe watchdog time)

1. DAT (F-Device Acknowledgement Time) is the time required by an F-Device to respond to a received PROFIsafe message. F-Devices are the safe units (in HIMA systems the CPU module) processing the F-Device stacks. In particular if modular systems/devices are used, they do not include the time values for the non safety-related functions/components. This DAT definition differs from that provided in the PROFIsafe specification V2.5c, Chapter 9.3.3, in the following points!
  - DAT does not include the time values for the internal F-Device bus.
  - DAT does not include the portion of PROFINet device production intervals.
  - DAT does not include any delays, e.g., due to input or output value filters or the physical properties of the inputs and outputs.
  - DAT refers to DATin (input) or DATout (output), depending on the connection.
  - The corresponding maximum value must be used for all time parameters.
2. In HIMatrix L3 and in HIMax, the internal F-Device bus time is:  
*(max. Number of Communication Time Slices - 1) \* WDT CPU.*
3. Requirement: The F-Device runs cyclically and its DAT is:  
 $DAT = 2 * \text{max. cycle}$ 
  - F-Device does not operate with *communication time slices*.  
If *HIMA CPU cycle time \* number of communication time slices* is less than the *F-Device cycle time*,  
*Delta = F-Device cycle time - HIMA CPU cycle time \* number of communication time slices* must be added to the *HIMA CPU cycle time* specified in the *F\_WD\_Time* equation.
  - F-Device operates with *communication time slices*.  
If *(HIMA CPU cycle time \* number of communication time slices)* is less than the *F-Device cycle time \* number of F-Device communication time slices*,  
for the *HIMA CPU cycle time*  
*Delta = (F-Device cycle time \* number of F-Device communication time slices) - (HIMA CPU cycle time \* number of communication time slices)* must be added to the calculation of the *F\_WD\_Time*.

### 5.3.3 SFRT (Safety Function Response Time)

#### 5.3.3.1 Calculation of the SFRT between an F-Device and a HIMA F-Host

The maximum SFRT allowed for a PROFIsafe connection between an F-Device and a HIMA F-Host with local output is calculated as follows:

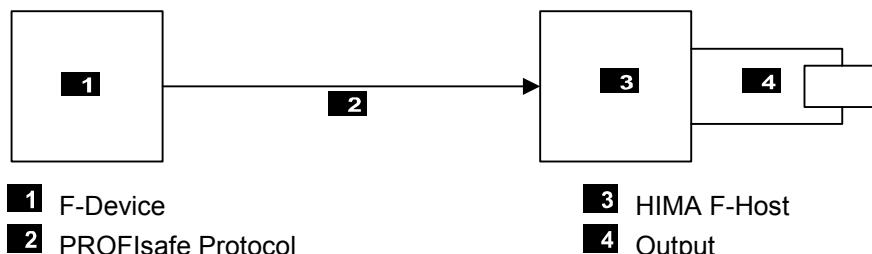


Figure 29: Reaction Time between an F-Device and a HIMA F-Host

$$\text{SFRT} \leq \text{MaxDataAgeIn} + 2 * \text{F_WD_TIME} + \text{MaxDataAgeOut} + \text{Tu}$$

#### Remarks:

With HIMax/HIMatrix L3 and if data is used locally, i.e., if it is not output to a HIMax I/O, the fault tolerance time of the CPU module can be replaced by  $2 * \text{WDT\_CPU}$ . See also the *Remark to the SFRT calculations* in Chapter 5.3.3.3.

#### 5.3.3.2 Calculation of the SFRT using an F-Device and a HIMA F-Host

The maximum SFRT allowed for a PROFIsafe connection between an F-Host and a HIMA F-Device with local output is calculated as follows:

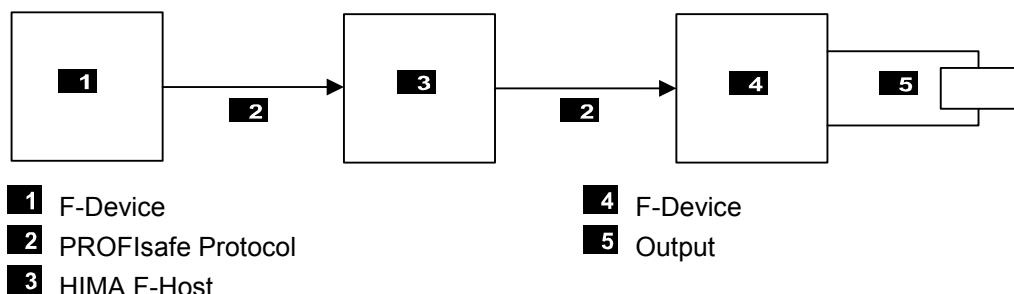


Figure 30: Reaction Time using a HIMA F-Host and two F-Devices

$$\text{SFRT} \leq \text{MaxDataAgeIn} + 2 * \text{F_WD_TIME(input)} + 2 * \text{WDT\_CPU} + \\ 2 * \text{F_WD_TIME(output)} + \text{MaxDataAgeOut} + \text{Tu}$$

#### 5.3.3.3 Remark to the SFRT Calculations

1. Definition of **SFRT** in accordance with IEC 61784-3, Ed.2
2. All additional delays in the user program must be added, e.g., due to TOF or TON function blocks, or in the modules, due to output filters, input filters, relay, etc.
3. **MaxDataAgeIn** is the maximum age of a process value that is read on a physical input and is added to a PROFIsafe message by a F-Device, but only the portion that is not already contained in DATin.

- 
- i In HIMatrix L3, MaxDataAgeIn must be set to 0 ms.  
In a HIMax system, the MaxDataAgeIn value
- does not exceed  $FTT\ CPU^{1)} - 2 * WDT\ CPU - DATin$  (for physical inputs).
  - must be set to 0 (for data created by the user program).
- 

#### 4. **MaxDataAgeOut**

is the worst case reaction time of an output F-Device or F-Host for

- a) outputting the received process values to a physical output,
- b) controlling the physical outputs after  $F\_WD\_Time$  has expired and
- c) deactivating the physical outputs if the devices fail.

- In HIMatrix L3,  $MaxDataAgeOut = 3 * WDT\ CPU$  (for physical outputs)
- In HIMax,  $MaxDataAgeOut = WDT\ CPU + FTT\ CPU^{1)}$  (for physical outputs)
- After  $F\_WD\_TIME$  has expired, the HIMax and HIMatrix L3 respond without faults at the latest after  $2 * WDT\ CPU$ .
  - If the F-Host/F-Device (HIMA CPU module) fails immediately prior to this reaction, the outputs of HIMatrix L3 are de-energized once  $WDT\ CPU$  has expired. In the HIMax system, this occurs at the latest after  $FTT\ CPU^{1)} - WDT\ CPU$  since the output module has not received any message for the duration of a  $WDT\ CPU$ .
  - Assuming that only **one** fault occurs in HIMatrix L3 or HIMax,  $1 * WDT\ CPU$  can be subtracted from MaxDataAgeOut.

#### 5. **Tu** is the minimum value of $DATin$ , $DATout$ , $WDT\ CPU$ . Theoretically, it is possible to use half of the value of $DATin$ and half of the value of $DATout$ , but the manufacturer must specify to which inaccuracy degree $F\_WD\_Time$ is monitored by the device. If the device runs cyclically,

$DAT = 2 * \text{max. device cycle}$ . For HIMA HIMatrix L3 and HIMax F-Devices,  $DATout = DATin = 2 * WDT\ CPU$

#### 6. **F\_WD\_TIME**, see Chapter 5.3.2.

<sup>1)</sup> Fault tolerance time of the CPU module

## 5.4 Requirements for Safely Operating PROFIsafe

### 5.4.1 Addressing

The HIMA PROFIsafe network corresponds to the PROFINET Ethernet network that can be used to transfer the PROFIsafe messages. In this context, network refers to a logical network that can include multiple physical sub-networks.

A separation is suitable for a PROFIsafe network if PROFIsafe messages cannot override the network separation.

This would be the case, if an IP-based router is used and the networks are connected to different Ethernet interfaces of the router.

The PROFIsafe networks are not separated if, for instance, the networks are connected via one port router, switches, hubs or Ethernet bridges.

Even if manageable switches are used and the PROFIsafe networks are separated, e.g., via port-based VLANs, one-to-one addressing should be striven for. This ensures that no connections between PROFIsafe networks are accidentally established during upgrade or maintenance.

The following conditions must be met for addressing the PROFIsafe devices:

- A one-to-one correspondence between the F addresses of the PROFIsafe device/modules in a PROFIsafe network must be ensured.
- Additionally, HIMA recommends to selecting the F addresses bijective, even in separated PROFIsafe networks, to ensure addressing safety.
- When starting up and modifying safety functions, ensure that the safety functions use the proper inputs and outputs of the corresponding PROFIsafe devices throughout the entire PROFIsafe network.
- The PROFIsafe F modules must be configured such that F modules with identical input and output data lengths that operate in a given PROFIsafe network have different CRC1 signatures, e.g., by assigning suitable F addresses or F\_WD\_Time. The CRC1 can be read in SILworX.

**Remark:**

This is definitely ensured for F modules operating in a given PROFIsafe network if only one F-Host is used and their F parameters only differ in the F address.

To avoid accidental generation of the same CRC1 signature, make sure that, e.g., the parameters *F\_WD\_Time*, *F\_Prm\_Flag1/2* are identical for all F modules and the F modules do not use an iPar CRC.

### Risk Associated with PROFIsafe Devices with Identical Input and Output Data Length

PROFIsafe device may only be operated if the F INPUT data length does not equal the F OUTPUT data length of the same PROFIsafe connection.

Otherwise, potential addressing faults in standard components and/or standard transmission technologies might not be detected and could cause safety-related malfunctions.

In HIMA F modules configured for the HIMA controller, the F input data length must differ from the F output data length. To prevent the risk of a safety-related malfunction, only use F-input modules or F-output modules. Do not use F-input/output modules

### 5.4.2 Network Aspects

The network used for transferring PROFIsafe messages must ensure sufficient availability and transmission quality.

- 
- i** A safety reaction is triggered if reduced transmission quality is detected by PROFIsafe, but is not recognized by the standard transmission facilities (Ethernet).
- 

After a safety reaction due to reduced transmission quality, the problems must be resolved to once again ensure sufficient transmission quality. Only after the required measures were taken, a PROFIsafe restart may be acknowledged. To do so, the Operator Acknowledge or Reset signal must be used.

**⚠ WARNING**

**Operator Acknowledge and Reset may only be used if dangerous states no longer exist.**



A PROFIsafe network must be protected against unauthorized access and actions (e.g., DoS, hacker, etc.). The measures must be agreed upon together with the supervising authority. This is particularly important if wireless transmission technologies are used.

For further details, refer to the PROFIsafe Specification V2.5c, Table 23 and Table 24.

**Availability with respect to added messages**

Message packets can be stored, e.g., using network components such as switches, and can be added (sent) to a later point in time<sup>1)</sup>. These message packets cause a shutdown if they are older than the last message packet received by the PROFIsafe device (see Consecutive Number Table 51 and Table 68: Edit Dialog Box for the Input Submodule).

<sup>1)</sup>Fault assumption for safety-related communication

## 5.5 HIMA PROFINET IO Controller and PROFIsafe F-Host

This chapter describes the features of the HIMA PROFINET IO controller and PROFIsafe F-Host, and the menu functions and dialog boxes required to configure the HIMA PROFINET IO controller and PROFIsafe in SILworX.

### System Requirements PROFINET IO Controller

Equipment and system requirements

| Element          | Description   |
|------------------|---|
| Controller       | HIMax with COM module<br>HIMatrix L3  |
| Processor module | The Ethernet interfaces on the processor module may not be used for PROFINET IO . |
| COM module       | Ethernet 10/100BaseT.   |
| Activation       | Software activation code required, see Chapter 3.4.                               |

Table 42: Equipment and System Requirements for the PROFINET IO Controller.

### PROFINET IO Controller and PROFIsafe Host Properties

| Properties   | Description  |
|--|--|
| Safety-related   | No   |
| Transfer rate  | 100 Mbit/s full duplex   |
| Transmission path  | Ethernet interfaces on the COM module<br>Ethernet interfaces in use can simultaneously be used for additional protocols. |
| Conformity class   | The PROFINET IO controller meets the requirements for Conformance Class A.   |
| Real Time Class  | RT Class 1   |
| Max. number of PROFINET IO controller                                | One PROFINET-IO controller can be configured for each COM module.  |
| Max. number of PROFINET IO devices application relations (ARs)       | A PROFINET IO controller can establish an application relation (AR) with a maximum of 64 PROFINET IO devices.            |
| Number of communication relations (CRs for each AR)                  | Standard: 1 input CR, 1 output CR, 1 alarm CR  |
| Max. process data length of a CR                                     | Output: max. 1440 bytes<br>Input: max. 1440 bytes  |
| Transmit clocking  | Possible at device level using the <i>Reduction Rate</i> setting.  |
| <b>The following features apply to PROFIsafe</b>                     |  |
| Max. number of F-Hosts (HIMax)                                       | 1024   |
| Max. number of F-Hosts (HIMatrix L3)                                 | 512  |
| Max. process data length of a CR                                     | Output: max. 123 bytes user data + 5 bytes <sup>1)</sup><br>Input: max. 123 bytes user data + 5 bytes <sup>1)</sup>      |
| Max. user data size<br>HIMax:<br>HIMatrix L3:                        | 1024 x 123 bytes = 125 952 bytes<br>512 x 123 bytes = 62 976 bytes   |
| <sup>1)</sup> 5 bytes management data (status/control bytes and CRC) |  |

Table 43: PROFINET IO Controller Properties

## 5.6 PROFINET IO/PROFIsafe Example

This example shows how to configure a HIMA PROFINET IO controller with a connection to the PROFINET IO device, on a HIMax controller.

The PROFINET IO device is equipped with a PROFINET IO module and a PROFIsafe module. In the HIMA PROFINET IO controller, the PROFINET IO device must be configured as it is actually structured.

### 5.6.1 Creating a HIMA PROFINET IO Controller in SILworX

#### To create a new HIMA PROFINET IO controller

1. In the structure tree, select **Configuration, Resource, Protocols**.
2. Right-click **Protocols** and select **New, PROFINET IO Controller** from the context menu to add a new PROFINET IO controller.
3. Select **Properties** from the context menu for PROFINET IO controller.
4. Enter the controller's device name in the **Name** field.
5. Click **COM Module**.

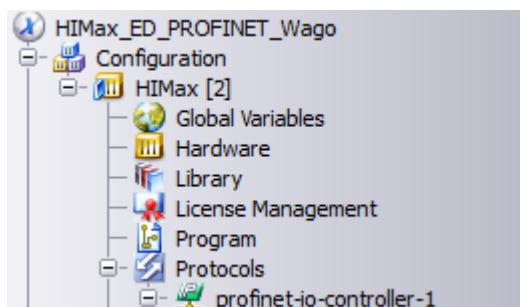


Figure 31: Structure Tree for the PROFINET IO Controller

#### 5.6.1.1 Configuring the Device in the HIMA PROFINET IO Controller

##### To create a HIMax PROFINET IO Device within the PROFINET IO controller

1. Select **New, PROFINET IO Device** from the context menu for the PROFINET IO controller.

##### To read the GSDML library file from an external data source (e.g., CD, USB stick, Internet):

1. On the structure tree, select **Configuration, Resource, Protocols, PROFINET IO Controller, GSDML Library**.
2. Right-click the GSDML library and select **New** from the context menu to add the GSDML file for to the PROFINET IO device.

##### To load the GSDML file for a new PROFINET IO device

1. On the structure tree, select **Configuration, Resource, Protocols, PROFINET IO Controller, PROFINET IO Device**.
2. Select **Properties** from the context menu and open the Parameter tab.
3. Enter the device name in the **Name** field.
4. Enter the IP address of the PROFINET IO device in the **IP Address** field.
5. On the drop-down menu for **GSDML File**, select the GSDML library file specific to PROFINET IO device and close **Properties**.

**To select the device access point (DAP) for the PROFINET IO device**

1. In the structure tree, select **Protocols**, **PROFINET IO Controller**, **PROFINET IO Device**, **DAP Module**.
2. Select **Edit** from the context menu and choose the suitable DAP data record for the PROFINET IO device (*DAP = Device Access Point*).



The GSDML library file often contains multiple *DAPs* from one manufacturers.

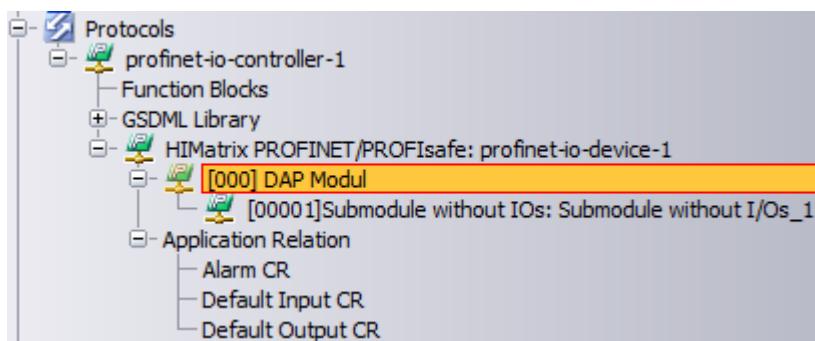


Figure 32: Device Access Point (DAP) for the PROFINET-IO Device

**To configure the module slots:**

1. In the structure tree, open **Protocols**, **PROFINET IO Device**.
2. Right-click PROFINET IO Device and select **New** from the context menu to open the module list.
3. From the module list, select suitable modules for the PROFINET IO device and click **Add Module(s)** to confirm the action.

**To number the PROFINET IO device modules**

The **Device Access Point (DAP) module** is associated by default with slot 0. All other PROFINET IO device modules must be numbered.

1. Right-click PROFINET IO Device and select **Properties** from the context menu.
2. In the **Slot** field, enter the device module slots in the same order as arranged on the actual PROFINET IO device.
3. Repeat these steps for every further **PROFINET IO device modules**.

The **Model and Features** tabs display additional details of the GSDML file.

**To configure the application relation**

1. In the structure tree, open **PROFIsafe IO Device**, **Application Relation**.
2. Right-click **Default Input CR** and select **Properties** from the context menu.
3. Adjust the reduction factor parameter, e.g., set it to 4.
4. Right-click **Default Output CR** and select **Properties** from the context menu.
5. Adjust the reduction factor parameter, e.g., set it to 4.

### 5.6.1.2 Configuring the Device Access Point (DAP) Module

#### To configure the DAP module (Device Access Point)

1. Select [000] DAP Module, [xxxxx] DAP Submodule.
2. Right-click [xxxxx] DAP Submodule and select **Edit** from the context menu.
3. If the consumer/provider status should not be controlled with a special user program logic, select the **System Variables** tab located in the **Edit** dialog box and assign the *Input Data Accepted by Controller* output variable a global variable with the TRUE initial value.



These system variables can be used to control the Consumer/Provider Status, see Chapter 5.2.

#### Setting the header parameters, e.g., for a DAP with alarm settings.

#### To set the header parameters for the device access point (DAP) for the PROFINET IO device

1. In the structure tree, select **Protocols, PROFINET IO Controller, PROFINET IO Device, DAP Module, [xxxxx] DAP Submodule, Alarm Settings (Header): Parameters**.
2. Select Properties from the context menu.
3. Enter the header parameter name in the **Name** field.
4. Click the **Edit** button to open a dialog box for setting or changing the parameters for the interfaces, diagnosis or alarms.

### 5.6.1.3 Configuring the PROFINET IO Device Modules



The sum of the variables (in bytes), must identical with the size of the module (in bytes).

#### To configure the PROFINET IO device module

1. Select [001] PROFINET IO Device Module, [xxxxx] PROFINET IO Device Submodule.
2. Right-click [xxxxx] Submodule and select **Edit** from the context menu.
3. In the **Edit** dialog box, select the **Process Variables** tab.
4. Drag the suitable variable from the Object Panel onto the **Input Signals** area.
5. Right-click anywhere in the **Inputs Signals** area and click **New Offsets** from the context menu to re-generate the variable offsets.
6. If the consumer/provider status should not be controlled, select the **System Variables** tab located in the **Edit** dialog box and assign the *Valid Output Data* and *Input Data Accepted by Controller* output variables a global variable with the TRUE initial value.

- 
- i These system variables can be used to control the Consumer/Provider Status, see Chapter 5.2.
- 

#### 5.6.1.4 Configuring the PROFIsafe IO Device Modules

- 
- i The sum of the variables (in bytes), must identical with the size of the module (in bytes).
- 

##### To configure the PROFIsafe IO device modules

1. Select [001] PROFIsafe IO Device Module, [xxxxx] PROFIsafe IO Device Submodule.
2. Right-click [xxxxx] Submodule and select Edit from the context menu.
3. In the Edit dialog box, select the Process Variables tab.
4. Drag the suitable variable from the Object Panel onto the Input Signals area.
5. Right-click anywhere in the Inputs Signals area and click New Offsets from the context menu to re-generate the variable offsets.
6. If the consumer/provider status should not be controlled, select the System Variables tab located in the Edit dialog box and assign the Valid Output Data and Input Data Accepted by Controller output variables a global variable with the TRUE initial value.

- 
- i These system variables can be used to control the Consumer/Provider Status, see Chapter 5.2.
- 

##### To configure the F parameters

1. Select [001] PROFIsafe IO Device Module, [xxxxx] PROFIsafe IO Device Submodule, F Parameters.
2. Right-click F Parameters and select Properties from the context menu.
3. Set the following parameters:
  - F\_Dest\_Add: Destination address of the device module
  - F\_WD\_Time: Watchdog time for the connection to this device module

##### To verify the PROFINET IO configuration

1. In the structure tree, open Configuration, Resource, Protocols, PROFINET IO Controller.
2. Right-click PROFINET IO Controller and select Verification from the context menu.
3. Thoroughly verify the messages displayed in the logbook and correct potential errors.

- 
- i Recompile the resource and load it into the controller to ensure that the new configuration can be used for communication with the PROFINET IO.
-

### 5.6.1.5 Identifying the PROFINET IO Devices within the Network

#### To find the PROFINET IO device within the Ethernet network

1. Log-in to the communication module containing the **PROFINET IO controller**.
2. In the structure tree for to the online view, select **PROFINET IO Controller**, **PROFINET IO Station**.
3. Select **Determine PROFINET IO Network Member** from the context menu.
  - A list appears specifying all the PROFINET devices in the network of the current PROFINET IO controller.

#### To configure the PROFINET IO device in the online view

1. In the list, right-click the PROFINET IO device to be configured, to change the settings.
2. Name the device using the **Name the PROFINET IO Device** context menu function.
  - Make sure that the PROFINET IO device name match the project. (Only lower case letters may be used!)
3. Use the **Network Settings** context menu function to set the IP address, subnet mask and gateway.



The PROFINET IO device name and network settings must be configured in the PROFINET IO controller, or no communication is possible.

## 5.7 Menu Functions for the PROFINET IO Controller

### 5.7.1 Example of Structure Tree for the PROFINET IO Controller

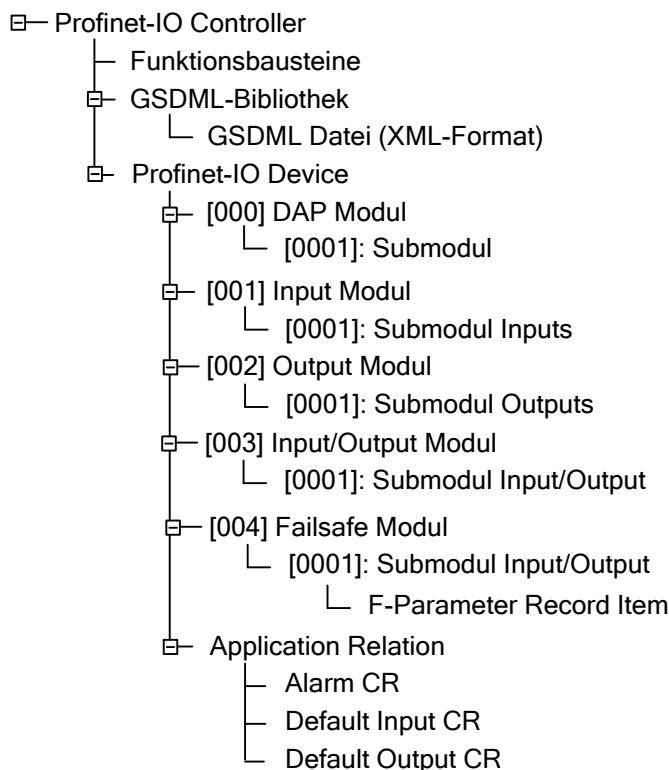


Figure 33: Structure Tree for the PROFINET IO Controller

### 5.7.2 PROFINET IO Controller

The **Properties** function of the context menu for the PROFINET IO controller is used to open the **Properties** dialog box. The dialog box contains the following tabs:

#### 5.7.2.1 Tab: PROFINET IO Device (Properties)

| Element                                | Description  |
|--|--|
| Type                                   | PROFINET IO controller   |
| Name                                   | Name of the PROFINET IO controller   |
| Refresh interval for process data [ms] | Refresh rate in milliseconds at which the COM and CPU exchange protocol data.<br>If the <i>Refresh Rate</i> is zero or lower than the cycle time for the controller, data is exchanged as fast as possible.<br>Range of values: 4...(2 <sup>31</sup> -1).<br>Default value: 0  |
| Force Process Data Consistency         | Activated: Transfer of all protocol data from the CPU to the COM within a CPU cycle.<br>Deactivated: Transfer of all protocol data from the CPU to the COM, distributed over multiple CPU cycles, each with 1100 byte per data direction. This can also allow lowering the cycle time of the controller.<br>Default value: activated |

|                              |  |
|------------------------------|--|
| Module                       | Selection of the COM module within which the protocol is processed.  |
| Activate Max. $\mu$ P Budget | Activated: Adopt the $\mu$ P budget limit from the <i>Max. <math>\mu</math>P Budget in [%]</i> field.<br>Deactivated: Do not use the $\mu$ P budget limit for this protocol.                           |
| Max. $\mu$ P budget in [%]   | Maximum $\mu$ P budget for the module that can be used for processing the protocols.<br>Range of values: 1...100%<br>Default value: 30%  |
| RPC Port Server              | Remote Procedure Call Port<br>Range of values: 1024...65535<br>Default value: 49152<br>RPC port server and RPC port client must not be identical!  |
| RPC Port Client              | Remote Procedure Call Port<br>Range of values: 1024...65535<br>Default value: 49153<br>RPC port server and RPC port client must not be identical!  |
| F_Source_Add                 | Address of the controller (F-Host).<br>Unique PROFIsafe controller/device addresses must be used in a PROFIsafe network. Also refer to the <i>IEC 61784-3-3 V2.5c Chapter 9.7</i> for further details. |

Table 44: The PROFINET IO Device (Properties) Tab

### 5.7.3 PROFINET IO Device (within the Controller)

The **Properties** function of the context menu for the PROFINET IO device is used to open the **Properties** dialog box, which contains the following tabs:

#### 5.7.3.1 Tab: Parameters (Properties)

| Element     | Description   |
|-------------|---|
| Name        | Name of the PROFINET IO device  |
| IP Address  | IP address of the communication partner.<br>Range of values: 0.0.0.0 ... 255.255.255.255<br>Default value: 192.168.0.99<br>Do not use IP addresses already in use, see Chapter 3.5.4. |
| Subnet Mask | Subnet mask for the addressed subnet containing the device.<br>Range of values: 0.0.0.0 ... 255.255.255.255<br>Default value: 255.255.255.0   |
| Gsdml File  | GSDML stands for Generic Station Description Markup Language and refers to an XML-based description language.<br>The GSDML file contains the PROFINET device master data              |

Table 45: Parameters (Properties) Tab

#### Tabs: Model and Features

The **Model and Features** tabs display additional details of the GSDML file such as *Manufacturer Name*, *Device Description* or *Supported Factors*. This additional information is intended to support the users during the device configuration and may not be changed.

### 5.7.4 DAP Module (Device Access Point Module)

Within a PROFINET device, a DAP module is always used for connecting the bus (DAP: Device Access Point). The DAP module is a default and cannot be deleted.

The **Properties** function located on the context menu for the DAP module is used to open the **Properties** dialog box, which contains the following parameters:

#### Tab: Parameters (Properties)

| Element | Description                        |
|---------|------------------------------------|
| Name    | Name for the DAP module            |
| Slot    | Not changeable<br>Default value: 0 |

Table 46: Parameters (Properties) Tab

#### Tabs: Model and Features

The **Model and Features** tabs display additional details of the GSDML file such as *Module ID*, *Hardware/Software Version*. This additional information is intended to support the users during the device configuration and may not be changed.

#### 5.7.4.1 DAP Submodule (Properties)

The **Properties** function located on the context menu for the DAP submodule is used to open the **Properties** dialog box, which contains the following parameters:

##### Tab: Parameters

| Element                           | Description   |
|-----------------------------------|---|
| Name                              | Name of the input submodule   |
| Sub-Slot                          | Default value: 1  |
| IO Data CR, Inputs                | Selection of the communication relation (CR) to which the submodule inputs should be transferred.<br>- None<br>- Default Input CR                   |
| Input Data Accepted by Controller | Selection of the communication relation (CR) to which the submodule IO consumer status (CS) should be transferred.<br>- None<br>- Default Output CR |

Table 47: Parameters Tab

##### Tabs: Model and Features

The **Model** and **Features** tabs display additional details of the GSDML file such as *Submodule ID*, *Data Length*. This additional information is intended to support the users during the device configuration and may not be changed.

#### 5.7.4.2 DAP Submodule (Edit)

The **Edit** function of the context menu for input submodules is used to open the **Edit** dialog box, which contains the following tabs:

##### Tab: System Variables

The **System Variables** tab contains the following system variables that are required to evaluate or control the state of the PROFINET IO submodule from within the user program.

| Element                           | Description |                    |      |
|-----------------------------------|-------------|--------------------|------|
| Valid input data                  | True        | Valid Input Data   | GOOD |
|                                   | False       | Invalid Input Data | BAD  |
| Input Data Accepted by Controller | True        | Valid Input Data   | GOOD |
|                                   | False       | Invalid Input Data | BAD  |

Table 48: System Variables Tab



These system variables can be used to control the Consumer/Provider Status, see Chapter 5.2.

#### 5.7.4.3 Header Parameter

Some devices contain so-called header parameters used to activate/deactivate parameters such as Diagnosis, Alarm and Interfaces.

#### 5.7.5 Input/Output PROFINET IO Modules

An input/output PROFINET IO module can have multiple submodules. HIMax PROFINET IO controllers have one submodule in each input/output PROFINET IO module.

The PROFINET IO input modules are used to enter the HIMax PROFINET IO controller input variables that are sent by the PROFINET IO device.

The PROFINET IO output modules are used to enter the HIMax PROFINET IO controller output variables that are sent to the PROFINET IO device.

##### To create the required PROFINET IO modules

1. In the structure tree, open **Configuration, Resource, Protocols, PROFINET IO Device**.
2. Right-click **PROFINET IO Device** and select **New** from the context menu.
3. Select the modules required.

The **Properties** function of the context menu for the input/output PROFINET IO modules is used to open the **Properties** dialog box, which contains the following tabs:

##### Tab: Parameters

| Element | Description                                 |
|---------|---|
| Name    | Name of the input/output PROFINET IO module |
| Slot    | 0 to 32767<br>Default value: 1              |

Table 49: Parameter Tab of the I/O PROFINET IO Module

##### Tabs: Model and Features

The **Model and Features** tabs display additional details of the GSDML file such as *Module ID*, *Hardware/Software Version*. This additional information is intended to support the users during the device configuration and may not be changed.

## 5.7.6 Input Submodule

The submodule parameters are used to define the communication relation of the module and its behavior after connection is interrupted.

### 5.7.6.1 Submodule Input (Properties)

The **Properties** function located on the context menu for the *Input Submodules* is used to open the **Properties** dialog box, which contains the following parameters:

Tab: Parameters

| Element                               | Description   |  |
|---------------------------------------|---|--|
| Name                                  | Name of the input submodule   |  |
| Sub-Slot                              | Not changeable for HIMax PROFINET IO controller.<br>Default value: 1  |  |
| IO Data CR, Inputs                    | Selection of the communication relation (CR) to which the submodule inputs should be transferred.<br>- None<br>- Default Input CR                   |  |
| Input data accepted by Controller     | Selection of the communication relation (CR) to which the submodule IO consumer status (CS) should be transferred.<br>- None<br>- Default Output CR |  |
| Shared Input                          | Activated   | Multiple PROFINET IO controllers can access the inputs.  |
|                                       | Deactivate d  | Only one PROFINET IO controller can access the inputs.   |
| Input Values if IO CR is Disconnected | Retain Last Value   | The input variables are freezed to the current value and used until a new connection is established. |
|                                       | Adopt Initial Values  | The initial data are used for the input variables.   |

Table 50: Parameters Tab

## Tabs: Model and Features

The **Model** and **Features** tabs display additional details of the GSDML file such as *Submodule ID*, *Hardware/Software Version* or *Data Length*. This additional information is intended to support the users during the device configuration and may not be changed.

### 5.7.6.2 Input Submodule (Edit)

The **Edit** function of the context menu for the input submodule is used to open the **Edit** dialog box. The dialog box contains the following tabs:

#### Tab: System Variables

The **System Variables** tab contains the following system variables that are required to evaluate the state of the PROFINET IO submodule from within the user program.

| Element  | Description |                           |                                     |
|--|-------------|---------------------------|-------------------------------------|
| Valid Input Data   | True        | Valid Input Data<br>GOOD  | False<br>Invalid Input Data<br>BAD  |
| Input Data Accepted by Controller  | True        | Valid Input Data<br>GOOD  | False<br>Invalid Input Data<br>BAD  |
| <b>The following parameters are only available for PROFIsafe modules</b> |             |                           |                                     |
| Valid Output Data  | True        | Valid Output Data<br>GOOD | False<br>Invalid Output Data<br>BAD |
| Output Data Accepted by Device   | True        | Valid Output Data<br>GOOD | False<br>Invalid Output Data<br>BAD |



These system variables can be used to control the Consumer/Provider Status, see Chapter 5.2.

|                               |   |
|-------------------------------|---|
| PROFIsafe Control             | <p>With each message, PROFIsafe sends the PROFIsafe control byte from the controller to the device that can be set in the user program.</p> <p>See also Chapter 5.3.1.</p> <p>Bit 0      iPar_EN_C:<br/>To load new iParameters into the F-Device, the iPar_EN_C must be set to TRUE to allow the F-Device to be unlocked.<br/>As long as iPar_EN_C is TRUE, failsafe values 0 are exchanged between F-Host and F-Device.</p> <p>Bit 1      OA_C: Operator Acknowledge.<br/>After a PROFIsafe error (e.g., CRC error or timeout), bit must be set to TRUE for at least a PROFIsafe cycle.<br/>If a PROFIsafe connection should be (re-)started, the operator acknowledgment can only be sent if no dangerous states exist.</p> <p>Bit 2      Reserved</p> <p>Bit 3      Reserved</p> <p>Bit 4      Activate_FV_C:<br/>FALSE: Process values are exchanged between F-Host and F-Device.<br/>TRUE: Failsafe values 0 are exchanged between F-Host and F-Device.</p> <p>Bit 5      Reserved</p> <p>Bit 6      </p> <p>Bit 7      Reset_Comm:<br/>PROFIsafe communication reset; the protocol stack is set to the initial state.<br/>The bit must be set to TRUE until the PROFIsafe status <i>Bit 2 Reset_Comm</i> has read back the TRUE value.</p> |
| PROFIsafe RoundTrip Time last | For a F-Host, it is the time between a data message transmission (with consecutive number N) and the reception of the corresponding acknowledgment (with consecutive number N), measured in milliseconds.   |

|                  |  |       |   |       |   |       |  |       |                |       |          |       |   |       |   |       |  |
|------------------|--|-------|---|-------|---|-------|--|-------|----------------|-------|----------|-------|---|-------|---|-------|--|
| PROFIsafe Status | <p>Each message received by the PROFIsafe on the host contains the PROFIsafe status byte, which can be evaluated in the user program.</p> <table> <tbody> <tr> <td>Bit 0</td><td>iPar_OK_S<br/>TRUE: New iParameters received<br/>FALSE: No change</td></tr> <tr> <td>Bit 1</td><td>OA_Req_S<br/>Operator Acknowledge Requested.</td></tr> <tr> <td>Bit 2</td><td>Reset_Comm<br/>is the Reset_Comm value read back from the host control byte. This bit indicates whether Reset_Comm has arrived.</td></tr> <tr> <td>Bit 3</td><td>FV_activated_S</td></tr> <tr> <td>Bit 4</td><td>Toggle_h</td></tr> <tr> <td>Bit 5</td><td>Device_Fault<br/>TRUE:<br/>The F-Device reported a device fault.<br/>FALSE:<br/>The F-Device did not report any device fault.</td></tr> <tr> <td>Bit 6</td><td>WD_timeout<br/>TRUE:<br/>Either the F-Device reported a watchdog timeout or the host timeout occurred on the F-Host.<br/>FALSE:<br/>No timeout occurred on the F-Device or on the F-Host.</td></tr> <tr> <td>Bit 7</td><td>CRC<br/>TRUE:<br/>Either the F-Device reported a CRC error or a CRC error occurred on the F-Host.<br/><br/>FALSE:<br/>No CRC fault occurred on the F-Device or on the F-Host.</td></tr> </tbody> </table> | Bit 0 | iPar_OK_S<br>TRUE: New iParameters received<br>FALSE: No change | Bit 1 | OA_Req_S<br>Operator Acknowledge Requested. | Bit 2 | Reset_Comm<br>is the Reset_Comm value read back from the host control byte. This bit indicates whether Reset_Comm has arrived. | Bit 3 | FV_activated_S | Bit 4 | Toggle_h | Bit 5 | Device_Fault<br>TRUE:<br>The F-Device reported a device fault.<br>FALSE:<br>The F-Device did not report any device fault. | Bit 6 | WD_timeout<br>TRUE:<br>Either the F-Device reported a watchdog timeout or the host timeout occurred on the F-Host.<br>FALSE:<br>No timeout occurred on the F-Device or on the F-Host. | Bit 7 | CRC<br>TRUE:<br>Either the F-Device reported a CRC error or a CRC error occurred on the F-Host.<br><br>FALSE:<br>No CRC fault occurred on the F-Device or on the F-Host. |
| Bit 0            | iPar_OK_S<br>TRUE: New iParameters received<br>FALSE: No change  |       |   |       |   |       |  |       |                |       |          |       |   |       |   |       |  |
| Bit 1            | OA_Req_S<br>Operator Acknowledge Requested.  |       |   |       |   |       |  |       |                |       |          |       |   |       |   |       |  |
| Bit 2            | Reset_Comm<br>is the Reset_Comm value read back from the host control byte. This bit indicates whether Reset_Comm has arrived.   |       |   |       |   |       |  |       |                |       |          |       |   |       |   |       |  |
| Bit 3            | FV_activated_S   |       |   |       |   |       |  |       |                |       |          |       |   |       |   |       |  |
| Bit 4            | Toggle_h   |       |   |       |   |       |  |       |                |       |          |       |   |       |   |       |  |
| Bit 5            | Device_Fault<br>TRUE:<br>The F-Device reported a device fault.<br>FALSE:<br>The F-Device did not report any device fault.  |       |   |       |   |       |  |       |                |       |          |       |   |       |   |       |  |
| Bit 6            | WD_timeout<br>TRUE:<br>Either the F-Device reported a watchdog timeout or the host timeout occurred on the F-Host.<br>FALSE:<br>No timeout occurred on the F-Device or on the F-Host.  |       |   |       |   |       |  |       |                |       |          |       |   |       |   |       |  |
| Bit 7            | CRC<br>TRUE:<br>Either the F-Device reported a CRC error or a CRC error occurred on the F-Host.<br><br>FALSE:<br>No CRC fault occurred on the F-Device or on the F-Host.   |       |   |       |   |       |  |       |                |       |          |       |   |       |   |       |  |

Table 51: System Variables Tab

The **Process Variables** tab is used to enter the input variables.

### 5.7.6.3 F Parameters of Submodule Input (for PROFIsafe Modules only)

To exchange process data safely, the PROFIsafe F-Devices need normalized F parameters. The F-Device only establishes communication if valid F parameters were configured. Grayed-out parameters are disabled and, to some extent, preset by the GSDML file or automatically calculated.

| Element       | Description   |
|---------------|---|
| Name          | Module name   |
| Index         | Module index  |
| F_Par_Version | Only V2-mode is supported. V1-mode is rejected.<br>It is determined through the GSDML file.                                     |
| F_Source_Add  | The F source address of the F-Host must be unique within the PROFIsafe network!<br>Range of values: 1 to 65534                  |
| F_Dest_Add    | The F destination address of the F-Device must be unique within the PROFIsafe network!<br>Range of values: 1 to 65534           |
| F_WD_Time     | Watchdog time<br>Range of values: 1 ms to 65534 ms  |
| F_iPar_CRC    | The F_iPar_CRC of the F-Device is entered in this field.  |
| F_SIL         | The SIL is displayed in this field<br>0 - SIL1<br>1 - SIL2<br>2 - SIL3<br>3 - NoSIL<br>It is determined through the GSDML file. |
| F_Check_iPar  | The CRC iParameter is displayed in this field<br>It is determined through the GSDML file.                                       |
| F_Block_ID    | Structure of the F parameters<br>It is determined through the GSDML file.   |
| F_CRC_Length  | It indicates if the 3-byte CRC or 4-byte CRC is used.<br>It is determined through the GSDML file.                               |
| F_Par_CRC     | The CRC F parameter (CRC1) is displayed in this field.<br>It is calculated based on the current F parameters                    |

Table 52: F Parameters for Submodule Input (Properties)

## 5.7.7 Submodule Output

The submodule parameters are used to define the communication relation of the module and its behavior after connection is interrupted.

### 5.7.7.1 Submodule Output (Properties)

The **Properties** function located on the context menu for the *Output Submodules* is used to open the **Properties** dialog box, which contains the following parameters:

Tab: Parameters

| Element                        | Description   |
|--------------------------------|---|
| Name                           | Name of the output submodule  |
| Sub-Slot                       | Not changeable for HIMax PROFINET IO controller.<br>Default value: 1  |
| IO Data CR, Outputs            | Selection of the communication relation (CR) to which the submodule outputs should be transferred.<br>- None<br>- Default Input CR                  |
| Output Data Accepted by Device | Selection of the communication relation (CR) to which the submodule IO consumer status (CS) should be transferred.<br>- None<br>- Default Output CR |

Table 53: Parameters Tab

### Tabs: Model and Features

The **Model** and **Features** tabs display additional details of the GSDML file such as *Submodule ID*, *Hardware/Software Version* or *Data Length*. This additional information is intended to support the users during the device configuration and may not be changed.

### 5.7.7.2 Submodule Output (Edit)

The **Edit** function of the context menu for Output Submodule is used to open the **Edit** dialog box, which contains the following tabs:

#### Tab: System Variables

The **System Variables** tab contains the following system variables that are required to evaluate the state of the PROFINET IO submodule from within the user program.

| Element   | Description |                            |  |
|---|-------------|----------------------------|--|
| Valid Output Data   | True        | Valid Output Data<br>GOOD  |  |
|   | False       | Invalid Output Data<br>BAD |  |
| The following parameters are only available for PROFIsafe modules                 |             |                            |  |
| Valid Input Data  | True        | Valid Input Data<br>GOOD   |  |
|   | False       | Invalid Input Data<br>BAD  |  |
| Input Data Accepted by Controller   | True        | Valid Input Data<br>GOOD   |  |
|   | False       | Invalid Input Data<br>BAD  |  |
| For more details on additional PROFIsafe module parameters, see <b>Table 51</b> . |             |                            |  |

Table 54: System Variables Tab



These system variables can be used to control the Consumer/Provider Status, see Chapter 5.2.

The **Process Variables** tab is used to enter the output variables.

### 5.7.7.3 F Parameters of Submodule Output (for PROFIsafe Modules only)

For a description of the F parameters, see Chapter 5.7.6.3.

## 5.7.8 Submodule Inputs and Outputs

The submodule parameters are used to define the communication relation of the module and its behavior after connection is interrupted.

### 5.7.8.1 Submodule Inputs and Outputs (Properties)

The **Properties** function located on the context menu for the *Input and Output Submodules* is used to open the **Properties** dialog box, which contains the following parameters:

Tab: Parameters

| Element                                 | Description   |
|---|---|
| Name                                    | Name of the input/output submodule  |
| Sub-Slot                                | Not changeable for HIMax PROFINET IO controller.<br>Default value: 1  |
| IO Data CR, Inputs                      | Selection of the communication relation (CR) to which the submodule inputs should be transferred.<br>- None<br>- Default Input CR                   |
| IO Data CR, Outputs                     | Selection of the communication relation (CR) to which the submodule outputs should be transferred.<br>- None<br>- Default Output CR                 |
| Input Data Accepted by Controller       | Selection of the communication relation (CR) to which the submodule IO consumer status (CS) should be transferred.<br>- None<br>- Default Output CR |
| Output Data Accepted by Device          | Selection of the communication relation (CR) to which the submodule IO consumer status (CS) should be transferred.<br>- None<br>- Default Input CR  |
| Input Values When IO CR is Disconnected | - Retain Last Value<br>- Adopt Initial Values   |

Table 55: Parameters Tab

### Tabs: Model and Features

The **Model** and **Features** tabs display additional details of the GSDML file such as *Submodule ID*, *Hardware/Software Version* or *Data Length*. This additional information is intended to support the users during the device configuration and may not be changed.

### 5.7.8.2 Submodule Inputs and Outputs (Edit)

The **Edit** function of the context menu for Input/Output Submodules is used to open the **Edit** dialog box, which contains the following tabs:

#### Tab: System Variables

The **System Variables** tab contains the following system variables that are required to evaluate the state of the PROFINET IO submodule from within the user program.

| Element                           | Description |                            |
|-----------------------------------|-------------|----------------------------|
| Valid Output Data                 | True        | Valid Output Data<br>GOOD  |
|                                   | False       | Invalid Output Data<br>BAD |
| Output Data Accepted by Device    | True        | Valid Output Data<br>GOOD  |
|                                   | False       | Invalid Output Data<br>BAD |
| Valid Input Data                  | True        | Valid Input Data<br>GOOD   |
|                                   | False       | Invalid Input Data<br>BAD  |
| Input Data Accepted by Controller | True        | Valid Input Data<br>GOOD   |
|                                   | False       | Invalid Input Data<br>BAD  |

For PROFIsafe module parameters, see **Table 51**.

Table 56: System Variables Tab

The **Process Variables** tab is used to enter the input and output variables in their corresponding area.



These system variables can be used to control the Consumer/Provider Status, see Chapter 5.2.

### 5.7.8.3 Submodule Input/Output F Parameters (for PROFIsafe Modules only)

For a description of the F parameters, see Chapter 5.7.6.3.

### 5.7.9 Application Relation (Properties)

An application relation (AR) is a logic construct for enabling data exchange between controller and device. In this example (see Figure 34), data is transferred within the application relation via the standard communication relations (alarm CR, default input CR and default output CR). This communication relations are already configured per default in the input and output modules.

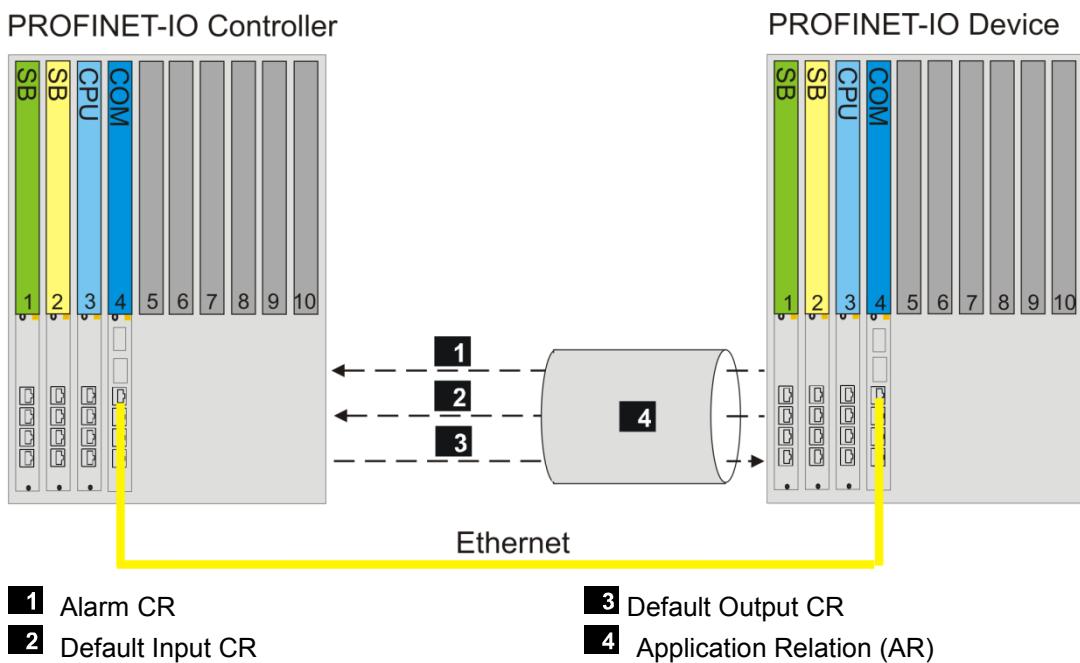


Figure 34: Communication via PROFINET IO/PROFIsafe

The **Properties** function of the context menu for application relation is used to open the **Properties** dialog box.

| Element                                 | Description   |
|---|---|
| Name                                    | Not changeable  |
| AR UUID                                 | Code for unambiguously identifying the application relation (AR).<br>Not changeable   |
| Connection Establishment Timeout Factor | From the perspective of the PROFINET IO device, this parameter is used during the creation of a connection to calculate the maximum time allowed between sending the response on the connect request and receiving a new request from the a PROFINET IO controller.<br>Range of values: 1...1000 (x 100 ms)<br>Default value: 600 |
| Supervisor may adopt the AR             | Definition whether a PROFINET IO supervisor may adopt the application relation (AR).<br>0 Not Allowed<br>1 Allowed<br>Default value: 0  |

Table 57: Application Relation (Properties)

### 5.7.10 Alarm CR (Properties)

Multiple communication relations (CR) can be established within an application relation.

The alarm CR is used by a PROFINET IO device to transmit alarms to the PROFINET IO controller.

The **Properties** function of the context menu for application relation opens the **Properties** dialog box, which contains the following tabs:

| Element                | Description  |  |                   |  |                      |   |       |                  |                 |  |
|------------------------|--|--|-------------------|--|----------------------|---|-------|------------------|-----------------|--|
| Name                   | Not changeable   |  |                   |  |                      |   |       |                  |                 |  |
| VLAN ID, High Priority | <p>Each virtual LAN (VLAN) is assigned a unique number to ensure separation. A device in the VLAN with ID=1 can communicate with any other device in the same VLAN, but not with a device in another VLAN (e.g., ID=2, 3, ...).</p> <p>Range of values, see also IEC 61158-6</p> <table> <tr> <td>0x000</td> <td>No VLAN</td> </tr> <tr> <td>0x001</td> <td>Standard VLAN</td> </tr> <tr> <td>0x002</td> <td>See IEEE 802.1 Q</td> </tr> <tr> <td>Up to<br/>0xFFFF</td> <td></td> </tr> </table> <p>Default value: 0</p> |  | 0x000             | No VLAN                                    | 0x001                | Standard VLAN   | 0x002 | See IEEE 802.1 Q | Up to<br>0xFFFF |  |
| 0x000                  | No VLAN  |  |                   |  |                      |   |       |                  |                 |  |
| 0x001                  | Standard VLAN  |  |                   |  |                      |   |       |                  |                 |  |
| 0x002                  | See IEEE 802.1 Q   |  |                   |  |                      |   |       |                  |                 |  |
| Up to<br>0xFFFF        |  |  |                   |  |                      |   |       |                  |                 |  |
| VLAN ID, Low Priority  | <p>Description, see VLAN ID, High Priority</p> <p>Default value: 0</p>   |  |                   |  |                      |   |       |                  |                 |  |
| Alarm Priority         | <table> <tr> <td>Use User Priority</td> <td>The priority assigned by the user is used.</td> </tr> <tr> <td>Ignore User Priority</td> <td>The priority assigned by the user is ignored. The generated alarm has low priority.</td> </tr> </table>   |  | Use User Priority | The priority assigned by the user is used. | Ignore User Priority | The priority assigned by the user is ignored. The generated alarm has low priority. |       |                  |                 |  |
| Use User Priority      | The priority assigned by the user is used.   |  |                   |  |                      |   |       |                  |                 |  |
| Ignore User Priority   | The priority assigned by the user is ignored. The generated alarm has low priority.  |  |                   |  |                      |   |       |                  |                 |  |
| Alarm Resends          | <p>Maximum number of F-Device resends if the controller does not respond.</p> <p>Range of values 3 to 15</p> <p>Default value: 10</p>  |  |                   |  |                      |   |       |                  |                 |  |
| Alarm Timeout Factor   | <p>The RTA timeout factor is used to calculate the maximum device time that may elapse after sending a RTA data (alarm) frame and receiving the RTA ack frame.</p> <p>RTA timeout = RTA timeout factor x 100 ms</p> <p>Range of values: 1...65535</p> <p>Default value: 5</p>  |  |                   |  |                      |   |       |                  |                 |  |

Table 58: Alarm CR (Properties)

### 5.7.11 Input CR (Properties)

The input CR is used by a PROFINET IO device to transmit variables to the PROFINET IO controller.

The **Properties** function of the context menu for the input CR appears the **Properties** dialog box. The dialog box contains the following parameters:

| Element           | Description  |       |         |       |               |       |                  |              |  |
|-------------------|--|-------|---------|-------|---------------|-------|------------------|--------------|--|
| Name              | Any unique name for an input CR<br>The default input CR cannot be changed  |       |         |       |               |       |                  |              |  |
| Type              | 1 (not changeable)   |       |         |       |               |       |                  |              |  |
| Send Clock Factor | The send clock factor defines the send clock for the cyclic IO CR data transfer.<br><b>Send clock = send clock factor x 31.25 µs</b><br>Range of values: 1...128<br>Default value: 32  |       |         |       |               |       |                  |              |  |
| Reduction Factor  | The reduction factor allows one to reduce the actual cycle time needed for sending the data of an IO CR to send clock. The actual data cycle time is calculated as follows:<br><b>Sending cycle = reduction factor x send clock</b><br>Range of values: 1...16384<br>Default value: 32 (depending on the device)   |       |         |       |               |       |                  |              |  |
| Watchdog Factor   | From the perspective of an IO CR consumer, the watchdog factor is used to calculate the maximum time allowed between the reception of two frames:<br><b>Watchdog time = watchdog factor x send cycle</b><br>Range of values: 1...7680<br>Default value: 3  |       |         |       |               |       |                  |              |  |
| VLAN ID           | Each virtual LAN (VLAN) is assigned a unique number to ensure separation. A device in the VLAN with ID=1 can communicate with any other device in the same VLAN, but not with a device in another VLAN (e.g., ID=2, 3, ...).<br>Range of values, see also IEC 61158-6:<br><table style="margin-left: 20px;"> <tr> <td>0x000</td> <td>No VLAN</td> </tr> <tr> <td>0x001</td> <td>Standard VLAN</td> </tr> <tr> <td>0x002</td> <td>See IEEE 802.1 Q</td> </tr> <tr> <td>Up to 0xFFFF</td> <td></td> </tr> </table><br>Default value: 0 | 0x000 | No VLAN | 0x001 | Standard VLAN | 0x002 | See IEEE 802.1 Q | Up to 0xFFFF |  |
| 0x000             | No VLAN  |       |         |       |               |       |                  |              |  |
| 0x001             | Standard VLAN  |       |         |       |               |       |                  |              |  |
| 0x002             | See IEEE 802.1 Q   |       |         |       |               |       |                  |              |  |
| Up to 0xFFFF      |  |       |         |       |               |       |                  |              |  |

Table 59: Input CR (Properties)

### 5.7.11.1 Input CR (Edit)

The **Edit** function of the context menu for the default input CR is used to open the **System Variables** dialog box, and contains the following system variables:

| Element              | Description |   |
|----------------------|-------------|---|
|                      | Value       | Description   |
| Data Status Input CR | 0           | <p>State<br/>With redundant connections, primary writes to the leading channel<br/>1 = Primary<br/>0 = Backup<br/>With mono connection:<br/>1 = Connected<br/>0 = Not connected</p> |
|                      | 1           | Not used  |
|                      | 2           | <p>Data Valid<br/>Invalid is used during the start-up phase or if the application is not able to report faults via IOPS.<br/>1 = Valid<br/>0 = Invalid</p>                          |
|                      | 3           | Not used  |
|                      | 4           | <p>Process State<br/>It has only an informative character, the actual data validity is reported via IOPS.<br/>1 = Run<br/>0 = Stop</p>  |
|                      | 5           | <p>Problem Indicator<br/>'Problem detected' provides details on the diagnostic data of the alarm CR.<br/>1 = Regular operation<br/>0 = Problem detected</p>                         |
|                      | 6           | Not used  |
|                      | 7           | Not used  |

Table 60: Input CR (Edit)

### 5.7.11.2 Output CR (Properties)

Multiple communication relations (CR) can be established within an application relation.

The output CR is used by the PROFINET IO device to transmit variables to the PROFINET IO controller.

The **Properties** function of the context menu for the output CR is used to open the **Properties** dialog box, which contains the following parameters:

| Element           | Description  |
|-------------------|--|
| Name              | Any unique name for an output CR<br>The default output CR cannot be changed  |
| Type              | 2 (not changeable)   |
| Send Clock Factor | The send clock factor defines the send clock for the cyclic IO CR data transfer.<br><b>Send clock = send clock factor x 31.25 µs</b><br>Range of values: 1...128<br>Default value: 32  |
| Reduction Factor  | For setting the transmission frequency.<br>The redundant factor allows the reduction of the actual cycle time needed for sending the data of an IO CR. The actual data cycle time is calculated as follows:<br><b>Sending cycle = reduction factor x send clock</b><br>Range of values: 1...16384<br>Default value: 32   |
| Watchdog Factor   | From the perspective of an IO CR consumer, the watchdog factor is used to calculate the maximum time allowed between the reception of two frames:<br><b>Watchdog time = watchdog factor x send cycle</b><br>Range of values: 1...7680<br>Default value: 3  |
| VLAN ID           | Each virtual LAN (VLAN) is assigned a unique number to ensure separation. A device in the VLAN with ID=1 can communicate with any other device in the same VLAN, but not with a device in another VLAN (e.g., ID=2, 3, ...).<br>Range of values, see also IEC 61158-6:<br>0x000      No VLAN<br>0x001      Standard VLAN<br>0x002      See IEEE 802.1 Q<br>Up to<br>0xFFFF<br><br>Default value: 0 |

Table 61: Output CR (Properties)

## 5.8 HIMA PROFINET IO Device

This chapter describes the characteristics of the HIMA PROFINET IO device and the menu functions and dialog boxes required to configure the HIMA PROFINET IO controller in SILworX.

## 5.9 System Requirements

Equipment and system requirements

| Element          | Description   |
|------------------|---|
| Controller       | HIMax with COM module<br>HIMatrix L3  |
| Processor module | The Ethernet interfaces on the processor module may not be used for PROFINET IO . |
| COM module       | Ethernet 10/100BaseT  |
| Activation       | Software activation code required, see Chapter 3.4.                               |

Table 62: Equipment and System Requirements for the PROFINET IO Controller.

## PROFINET IO Device Properties

| Element  | Description  |
|--|--|
| Safety-related   | No   |
| Transfer rate  | 100 Mbit/s full duplex   |
| Transmission path  | Ethernet interfaces on the COM module<br>Ethernet interfaces in use can simultaneously be used for additional protocols. |
| Conformity class   | The PROFINET IO device meets the requirements for Conformance Class A.   |
| Real Time Class  | RT Class 1   |
| Max. number of PROFINET IO devices                                       | One PROFINET-IO device can be configured for each COM module.  |
| Max. number of application relations (ARs) to the PROFINET IO controller | A PROFINET IO device can establish a maximum of 5 application relations (ARs) to a PROFINET IO controller.               |
| Max. number of communication relations (CRs for each AR)                 | Standard: 1 input, 1 output, 1 alarm   |
| Max. process data length of all configured PROFINET IO modules           | Output: max. 1440 bytes<br>Input: max. 1440 bytes  |
| Data Priorization  | Possible at device level using the <i>Reduction Rate</i> setting.  |
| <b>The following features apply to PROFIsafe</b>                         |  |
| Max. number of F-Devices on each COM module (HIMax and HIMatrix L3)      | 63   |
| Max. process data length of a CR   | Output: max. 123 bytes user data + 5 bytes <sup>1)</sup><br>Input: max. 123 bytes user data + 5 bytes <sup>1)</sup>      |
| Max. user data size<br>HIMax:<br>HIMatrix L3                             | 1024 x 123 bytes = 125 952 bytes<br>512 x 123 bytes = 62 976 bytes   |
| <sup>1)</sup> 5 bytes management data (status/control bytes and CRC)     |  |

Table 63: PROFINET IO Controller Properties

## 5.10 PROFINET IO/PROFIsafe Example

The following chapter describes how to configure the HIMA PROFINET IO/ PROFIsafe device.

### 5.10.1 Configuring the PROFINET IO Device in SILworX

#### To create a new HIMA PROFINET IO device

1. In the structure tree, open **Configuration, Resource, Protocols**.
2. Select **New, PROFINET IO Device** from the context menu for protocols to add a new PROFINET IO device.
3. Right-click PROFINET IO Device and select **Properties** from the context menu.
4. Enter the PROFINET IO device name in the **Name** field.
5. Click **COM Module**.

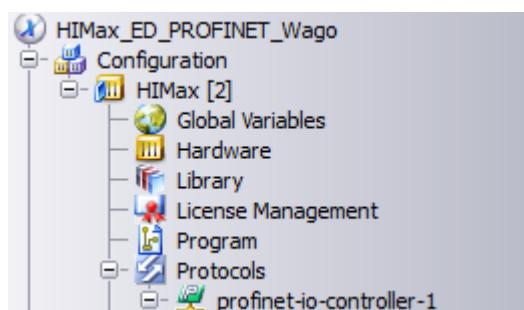


Figure 35: Structure Tree for the PROFINET IO Device

#### To create the required PROFINET IO modules

1. In the structure tree, open **Configuration, Resource, Protocols, PROFINET IO Device**.
2. Select **New** from the context menu for the PROFINET IO device.
3. For this example, select the following modules.

| PROFINET IO/ PROFIsafe module | Slot |
|-------------------------------|------|
| In 1 byte                     | 1    |
| Safe Out 1 Byte               | 2    |

#### To number the PROFINET IO device modules

1. Right-click the first **PROFINET IO device module**, and then click **Properties**.
2. Enter **1** into the **Slot** field.
3. Repeat these steps for every further **PROFINET IO device modules** and number the modules consecutively.



**Number** the modules such as actually positioned in the PROFINET IO device.

#### The following step is only required for PROFIsafe module!

4. Enter the PROFIsafe module address in the *PROFIsafe F\_Destination\_Address* field.

### 5.10.1.1 Configuring the PROFINET IO Device Input Module

- 
- i** The sum of the variables (in bytes), must be identical with the size of the module (in bytes).
- 

#### To configure the input module [01] In 1 Byte

1. In the PROFINET IO device, select the input module **[01] In 1 Byte**.
2. Right-click **[01] In 1 Bytes** and select **Edit** from the context menu.
3. In the **Edit** dialog box, select the **Process Variables** tab.
4. Drag the suitable variable from the Object Panel onto the **Input Signals** area..
5. Right-click anywhere in the **Input Signals** area to open the context menu.
6. Click **New Offsets** to re-generate the variable offsets.
7. If the consumer/provider status should not be controlled, select the **System Variables** tab located in the **Edit** dialog box and assign the *Input Data Accepted by Device* output variable a global variable with the TRUE initial value.

- 
- i** These system variables can be used to control the Consumer/Provider Status, see Chapter 5.2.
- 

### 5.10.1.2 Configuring the PROFIsafe Device Output Module

#### To configure the output module [02] Out 1 Bytes

1. In the PROFINET IO device, select the output module **[02] Safe Out 1 Byte**.
2. Right-click **[02] Out 1 Bytes** and select **Edit** from the context menu.
3. In the **Edit** dialog box, select the **Process Variables** tab.
4. Drag the suitable variable from the Object Panel onto the **Output Signals** area.
5. Right-click anywhere in the **Output Signals** area to open the context menu.
6. Click **New Offsets** to re-generate the variable offsets.
7. If the consumer/provider status should not be controlled, select the **System Variables** tab located in the **Edit** dialog box and assign the *Valid Output Variable* and *Input Data Accepted by Device* output variables a global variable with the TRUE initial value.

- 
- i** These system variables can be used to control the Consumer/Provider Status, see Chapter 5.2.
-

### 5.10.1.3 Verifying the PROFINET IO Device Configuration

#### To verify the PROFINET IO device configuration

1. In the structure tree, open **Configuration, Resource, Protocols, PROFINET IO Device**.
2. Click the **Verification** button on the Action Bar, and then click **OK** to confirm the action.
3. Thoroughly verify the messages displayed in the logbook and correct potential errors.



Use the user program of the PROFINET IO device resource to recompile the configuration of the PROFINET IO device and transfer it to the controllers. Only after this step, the new configuration can be used for communication with the PROFINET IO

## 5.11 Menu Functions for PROFINET IO Device

### 5.11.1 Menu Function: Properties

The **Properties** function of the context menu for the PROFINET IO device is used to open the **Properties** dialog box.

| Element                        | Description   |
|--------------------------------|---|
| Type                           | PROFINET IO device  |
| Name                           | Any unique name for a PROFINET IO device  |
| Process Data Refresh Rate [ms] | Refresh rate in milliseconds at which the COM and CPU exchange protocol data.<br>If the refresh rate is zero or lower than the cycle time for the controller, data is exchanged as fast as possible.<br>Range of values: 4...(2 <sup>31</sup> -1)<br>Default value: 0   |
| Force Process Data Consistency | Activated: Transfer of all protocol data from the CPU to the COM within a CPU cycle.<br>Deactivated: Transfer of all protocol data from the CPU to the COM, distributed over multiple CPU cycles, each with 1100 byte per data direction. This can also allow lowering the cycle time of the controller.<br>Default value: activated  |
| Module                         | Selection of the COM module within which the protocol is processed.   |
| Activate Max. $\mu$ P Budget   | Activated: Adopt the $\mu$ P budget limit from the <i>Max. <math>\mu</math>P Budget in [%]</i> field.<br>Deactivated: Do not use the $\mu$ P budget limit for this protocol.  |
| Max. $\mu$ P budget in [%]     | Maximum $\mu$ P budget for the module that can be used for processing the protocols.<br>Range of values: 1...100%<br>Default value: 30%   |
| RPC Port Server                | Remote Procedure Call Port<br>Range of values: 1024...65535<br>Default value: 49152<br>RPC port server and RPC port client must not be identical!   |
| RPC Port Client                | Remote Procedure Call Port<br>Range of values: 1024...65535<br>Default value: 49153<br>RPC port server and RPC port client must not be identical!   |
| AutoAwaitF ParamsOn ConnLoss   | <b>This parameter is only used for PROFIsafe modules.</b><br>Whenever the connection to the F-Host is lost, the F parameters must be reloaded into the F-Device.<br>This parameter can be activated to simplify the PROFIsafe start-up. Afterwards, the F-Device automatically sets the required F parameters at restart or after a connection loss. <b>After start-up, this parameter must absolutely be deactivated to ensure a PROFIsafe-compliant behavior.</b><br>Activated: The F-Device automatically enters the <i>Wait for F Parameters</i> state.<br>Deactivated: The users must send the online command to allow the F-Device to enter the <i>Wait for F Parameters</i> state.<br>Default value: Deactivated |

Table 64: PROFINET IO Device General Properties

### 5.11.2 HIMA PROFINET IO Modules

The following PROFINET IO modules are available in the HIMA PROFINET IO device.

| PROFINET IO module | Max. size for the input variables | Max. size of the output variables |
|--------------------|-----------------------------------|-----------------------------------|
| In 1 byte          | 1 byte                            |                                   |
| In 2 bytes         | 2 bytes                           |                                   |
| In 4 bytes         | 4 bytes                           |                                   |
| In 8 bytes         | 8 bytes                           |                                   |
| In 16 bytes        | 16 bytes                          |                                   |
| In 32 bytes        | 32 bytes                          |                                   |
| In 64 bytes        | 64 bytes                          |                                   |
| In 128 bytes       | 128 bytes                         |                                   |
| In 256 bytes       | 256 bytes                         |                                   |
| In 512 bytes       | 512 bytes                         |                                   |
| In 1024 bytes      | 1024 bytes                        |                                   |
| In-Out 1 byte      | 1 byte                            | 1 byte                            |
| In-Out 2 bytes     | 2 bytes                           | 2 bytes                           |
| In-Out 4 bytes     | 4 bytes                           | 4 bytes                           |
| In-Out 8 bytes     | 8 bytes                           | 8 bytes                           |
| In-Out 16 bytes    | 16 bytes                          | 16 bytes                          |
| In-Out 32 bytes    | 32 bytes                          | 32 bytes                          |
| In-Out 64 bytes    | 64 bytes                          | 64 bytes                          |
| In-Out 128 bytes   | 128 bytes                         | 128 bytes                         |
| In-Out 256 bytes   | 256 bytes                         | 256 bytes                         |
| In-Out 512 bytes   | 512 bytes                         | 512 bytes                         |
| Out 1 byte         |                                   | 1 byte                            |
| Out 2 bytes        |                                   | 2 bytes                           |
| Out 4 bytes        |                                   | 4 bytes                           |
| Out 8 bytes        |                                   | 8 bytes                           |
| Out 16 bytes       |                                   | 16 bytes                          |
| Out 32 bytes       |                                   | 32 bytes                          |
| Out 64 bytes       |                                   | 64 bytes                          |
| Out 128 bytes      |                                   | 128 bytes                         |
| Out 256 bytes      |                                   | 256 bytes                         |
| Out 512 bytes      |                                   | 512 bytes                         |
| Out 1024 bytes     |                                   | 1024 bytes                        |

Table 65: PROFINET IO Modules

### 5.11.3 HIMA PROFIsafe Modules

The following PROFIsafe modules are available in the HIMA PROFINET IO device:

| PROFIsafe module      | Max. size for the input variables | Max. size of the output variables |
|-----------------------|-----------------------------------|-----------------------------------|
| Safe In 1 Byte        | 1 byte                            |                                   |
| Safe In 2 bytes       | 2 bytes                           |                                   |
| Safe In 4 bytes       | 4 bytes                           |                                   |
| Safe In 8 bytes       | 8 bytes                           |                                   |
| Safe In 16 bytes      | 16 bytes                          |                                   |
| Safe In 32 bytes      | 32 bytes                          |                                   |
| Safe In 64 bytes      | 64 bytes                          |                                   |
| Safe In 123 bytes     | 123 bytes                         |                                   |
| Safe In-Out 1 Byte    | 1 byte                            | 1 byte                            |
| Safe In-Out 2 bytes   | 2 bytes                           | 2 bytes                           |
| Safe In-Out 4 bytes   | 4 bytes                           | 4 bytes                           |
| Safe In-Out 8 bytes   | 8 bytes                           | 8 bytes                           |
| Safe In-Out 16 bytes  | 16 bytes                          | 16 bytes                          |
| Safe In-Out 32 bytes  | 32 bytes                          | 32 bytes                          |
| Safe In-Out 64 bytes  | 64 bytes                          | 64 bytes                          |
| Safe In-Out 123 bytes | 123 bytes                         | 123 bytes                         |
| Safe Out 1 Byte       |                                   | 1 byte                            |
| Safe Out 2 bytes      |                                   | 2 bytes                           |
| Safe Out 4 bytes      |                                   | 4 bytes                           |
| Safe Out 8 bytes      |                                   | 8 bytes                           |
| Safe Out 16 bytes     |                                   | 16 bytes                          |
| Safe Out 32 bytes     |                                   | 32 bytes                          |
| Safe Out 64 bytes     |                                   | 64 bytes                          |
| Safe Out 123 bytes    |                                   | 123 bytes                         |

Table 66: PROFIsafe Modules

#### To create a PROFINET or PROFIsafe module

1. In the structure tree, open **Configuration, Resource, Protocols, PROFINET IO Device**.
2. Right-click **PROFINET IO Device** and select **Insert Modules** from the context menu.
3. Select a suitable module to transport the required process data.
4. Right-click the module selected and select **Edit**.
  - Enter the input and/or output variables in the **Process Variables** tab.
  - If the consumer/provider status should not be controlled, assign the *Valid Output Variable* and *Input Data Accepted by Device* output variables a global variable with the TRUE initial value in the **System Variables** tab.



These system variables can be used to control the Consumer/Provider Status, see Chapter 5.2.

#### The following setting is only required for PROFIsafe module!

- Enter the *Slot* and *F\_Destination\_Address Register* in the **Properties** tab.

### 5.11.4 PROFINET IO and PROFIsafe module

The module parameters are used to define the communication relations of the module and its behavior after connection is interrupted.

#### Properties

The **Properties** function of the context menu for the modules opens the **Properties** dialog box. The dialog box contains the following tabs:

| Element                            | Description   |
|------------------------------------|---|
| Name                               | Name od the device module   |
| Slot                               | 0 to 32767  |
| Module ID                          | Unique number   |
| Sub-Slot                           | Number of sub-slots   |
| Process data behavior              | Process data value after the connection is interrupted<br>- Retain last valid process data<br>- Adopt initial data    |
| Length of IO Input Data            | 1...123   |
| Length of IO Output Data           | 1...123   |
| PROFIsafe<br>F_Destination_Address | The F destination address of the F-Device must be unique within the PROFIsafe network!<br>Range of values: 1 to 65534 |

Table 67: Device Module General Properties

#### Edit

The **Edit** function of the context menu for the submodule is used to open the **Edit** dialog box.

The **System Variables** tab contains the following system variables that are required to evaluate the state of the submodule from within the user program.



These system variables can be used to control the Consumer/Provider Status, see Chapter 5.2.

| Element                        | Description |                            |
|--------------------------------|-------------|----------------------------|
| Valid Output Data              | True        | Valid Output Data<br>GOOD  |
|                                | False       | Invalid Output Data<br>BAD |
| Output Data Accepted by Device | True        | Valid Output Data<br>GOOD  |
|                                | False       | Invalid Output Data<br>BAD |
| Valid Input Data               | True        | Valid Input Data<br>GOOD   |
|                                | False       | Invalid Input Data<br>BAD  |
| Input Data Accepted by Device  | True        | Valid Input Data<br>GOOD   |
|                                | False       | Invalid Input Data<br>BAD  |

| <b>The following variables are only used for PROFIsafe modules</b> |  |   |      |   |      |   |      |   |        |
|--|--|---|------|---|------|---|------|---|--------|
| PROFIsafe Control  | <p>The PROFIsafe control byte sent by the controller read within the device, see also Chapter 5.3.1.</p> <p>Bit 0      iPar_EN_DC<br/>Enabling the controller releases the device to load new iParameters into the device.</p> <p>Bit 1      OA_Req_DC<br/>Operator Acknowledge from host control byte.</p> <p>Bit 2      Reset_Comm<br/>is the Reset_Comm value read back from the host control byte.</p> <p>Bit 3      activate_FV_DC<br/>FALSE: Process values are exchanged between F-Host and F-Device.<br/>TRUE: Failsafe values 0 are exchanged between F-Host and F-Device.</p> <p>Bit 4      Toggle_d<br/>Toggle bit of the F-Device</p> <p>Bit 5      Cons_nr_R<br/>The consecutive number is adopted whenever there is a change between two consecutive control bytes of the Toggle_d bit, e.g., it does not depend on the fault occurrence.</p> <p>Bit 6      F_ParamValid<br/>TRUE: F parameters were set<br/>FALSE: Otherwise</p> <p>Bit 7      F_Param_ConfiguredTwice<br/>TRUE: The F-Device was configured more than one time with different F parameters.<br/>FALSE: Otherwise</p> |   |      |   |      |   |      |   |        |
| PROFIsafe F_iPar_CRC   | <p>iParameters are independent or technology-specific F-Device parameters. The iPar_CRC results from the configuration of the F-Device.</p> <p><b>i</b> The users are responsible for setting the valid iPar_CRC after configuring the iParameters and moving to hot operation.</p>  |   |      |   |      |   |      |   |        |
| PROFIsafe F_SIL  | <table> <tr> <td>0</td><td>SIL1</td></tr> <tr> <td>1</td><td>SIL2</td></tr> <tr> <td>2</td><td>SIL3</td></tr> <tr> <td>3</td><td>No SIL</td></tr> </table>   | 0 | SIL1 | 1 | SIL2 | 2 | SIL3 | 3 | No SIL |
| 0  | SIL1   |   |      |   |      |   |      |   |        |
| 1  | SIL2   |   |      |   |      |   |      |   |        |
| 2  | SIL3   |   |      |   |      |   |      |   |        |
| 3  | No SIL   |   |      |   |      |   |      |   |        |
| PROFIsafe RoundTrip Time last                                      | PROFIsafe must determine the RoundTripTimeLast on a F-Host. For a F-Host, it is the time between a data message transmission (with consecutive number N) and the reception of the corresponding acknowledgment (with consecutive number N), measured in milliseconds.  |   |      |   |      |   |      |   |        |

|                  |  |
|------------------|--|
| PROFIsafe Status | With each message, PROFIsafe sends the PROFIsafe status byte that can be set in the device user program from the device to the controller.<br>See also Chapter 5.3.1.<br>Bit 0      iPar_OK_DS<br>New iParameters received.<br>Bit 1      Device_Fault_DS<br>TRUE: Device fault<br>FALSE: No device fault<br>It is only taken into account from the 21<br><i>Await Message PROFIsafe state.</i><br>Bit 2      Reserved<br>Bit 3      Reserved<br>Bit 4      FV_activated_DS<br>Activate the failsafe value<br>Bit 5      Reserved<br>Bit 6<br>Bit 7      Reset_Comm<br>Protocol stack is reset to its initial state. |
|------------------|--|

Table 68: Edit Dialog Box for the Input Submodule

The **Process Variables** tab is used to enter the input variables.

## 6 PROFIBUS DP

PROFIBUS DP is an international, open fieldbus standard that is used when a fast reaction time is required for small amounts of data.

The HIMA PROFIBUS DP master and the HIMA PROFIBUS DP slave meet the criteria specified in the European EN 50170 standard [7] and the internationally binding IEC standard 61158 for PROFIBUS DP.

The HIMA PROFIBUS DP master can exchange data with the PROFIBUS DP slaves cyclically and acyclically.

Different function blocks are available in SILworX to acyclically exchange data. These function blocks are used to tailor the HIMA PROFIBUS DP master and the PROFIBUS DP slaves to best meet the project requirements.

A redundant PROFIBUS DP connection can only be implemented by configuring a second PROFIBUS DP master/slave and adjusting it in the user program.

- PROFIBUS DP master (see Chapter 6.1)
- PROFIBUS DP slave (see Chapter 6.13)

## 6.1 HIMA PROFIBUS DP Master

This chapter describes the characteristics of the HIMA PROFIBUS DP master and the menu functions and dialog boxes required for configuring the HIMA PROFIBUS DP master in SILworX.

### Equipment and System Requirements:

| Element         | Description   |
|-----------------|---|
| HIMA controller | HIMax with COM module<br>HIMatrix: CPU OS version 7 and beyond and COM OS version 12 and beyond   |
| COM Module      | The serial fieldbus interface (FB1 or FB2) used on the COM module must be equipped with an optional HIMA PROFIBUS DP master submodule, see Chapter 3.6. |
| Activation      | Activation via Fieldbus Submodule, see Chapter 3.4.   |

Table 69: Equipment and System Requirements

- 
- i** For the HIMax system, HIMA recommends operating the PROFIBUS DP using the FB1 fieldbus interface (maximum transfer rate 12 Mbit). The maximum transfer rate permitted for the FB2 fieldbus interface is 1.5 Mbit.
- 

### PROFIBUS DP Master Properties:

| Element                                | Description   |
|--|---|
| Type of HIMA PROFIBUS DP master        | DP-V1 Class 1 Master with additional DP-V2 functions  |
| Transfer rate                          | 9.6 kbit/s...12 Mbit/s  |
| Bus address                            | 0...125   |
| Max. number of PROFIBUS DP master      | Two PROFIBUS DP masters can be configured for each HIMax COM module or HIMatrix F20, F30, F35, F60. Only one PROFIBUS DP master can be configured for the HIMatrix F20.       |
| Max. number of PROFIBUS DP slaves      | Up to 122 slaves can be configured for each resource (in all master protocol instances). However, a maximum of 31 slaves can be connected to a bus segment without repeaters. |
| Maximum process data length to a slave | DP output=: max. 244 bytes<br>DP input=: max. 244 bytes   |

Table 70: PROFIBUS DP Master Properties

According to the standard, a total of three repeaters may be used such that a maximum of 122 stations are possible per serial interface on a master.

### 6.1.1 Creating a HIMA PROFIBUS DP Master

#### To create a new HIMA PROFIBUS DP master

1. In the structure tree, open **Configuration, Resource, Protocols**.
2. Select **New, PROFIBUS DP Master** from the context menu for protocols to add a new PROFIBUS DP master.
3. Select **Properties, General** from the context menu for the PROFIBUS DP master.
4. Select **Module and Interfaces**.

## 6.2 PROFIBUS DP: Example

In this example, a HIMA PROFIBUS DP master exchanges variables with a HIMA PROFIBUS DP slave.

The example shows how to create and configure a HIMA PROFIBUS DP master and a PROFIBUS DP slave.

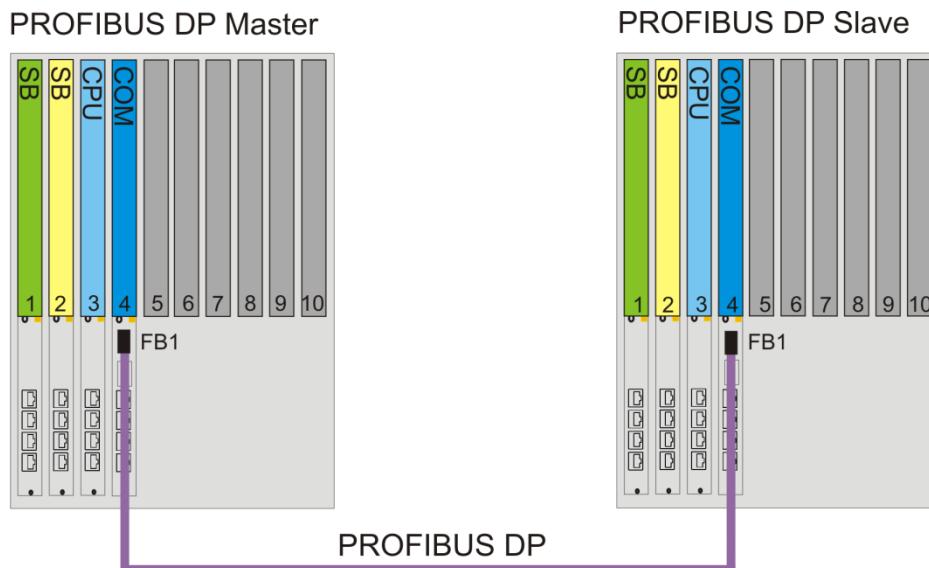


Figure 36: Communication via PROFINET IO

Fieldbus interface 1 on the COM modules of both HIMax controllers must be equipped with the corresponding PROFIBUS DP submodule, see Chapter 3.6.

For this example, the following global variables must be created in SILworX:

| Global Variable  | Type  |
|------------------|-------|
| PB_Slave_Master1 | UINT  |
| PB_Slave_Master2 | DWORD |
| PB_Slave_Master3 | DWORD |
| PB_Slave_Master4 | BYTE  |
| PB_Master_Slave1 | DWORD |
| PB_Master_Slave2 | BYTE  |

### 6.2.1 Configuring the PROFIBUS DP Slave

Configuration of the PROFIBUS DP slave.

#### To create a new HIMA PROFIBUS DP Slave

1. In the structure tree, open **Configuration, Resource, Protocols**.
2. Select **New, PROFIBUS DP Slave** from the context menu for protocols to add a new PROFIBUS DP slave .
3. Select **Edit** from the context menu for the PROFIBUS DP slave.
4. In the **Properties** tab, select **COM Module** and **Interfaces** (e.g., FB1).

### To assign variables in the HIMA PROFIBUS DP slave

1. Select **Edit** from the context menu for the PROFIBUS DP slave.
2. In the **Edit** dialog box, select the **Process Variables** tab.



The start address of the input and output variables in the HIMA PROFIBUS DP slave always begin with 0. If the PROFIBUS DP master (from another manufacturer) expects a higher start address, dummy variables must be added to the reference variables.

### Outputs in the HIMA PROFIBUS DP Slave

| Name             | Type  | Offset | Global Variable  |
|------------------|-------|--------|------------------|
| PB_Slave_Master1 | UINT  | 0      | PB_Slave_Master1 |
| PB_Slave_Master2 | DWORD | 2      | PB_Slave_Master2 |
| PB_Slave_Master3 | DWORD | 6      | PB_Slave_Master3 |
| PB_Slave_Master4 | BYTE  | 10     | PB_Slave_Master4 |

Table 71: Outputs in the HIMA PROFIBUS DP Slave

1. Drag the global variable to be sent from the Object Panel onto the **Output Variables** area.



In this example, the output variables of the HIMA PROFIBUS DP slave are composed of **four variables** with a total of **11 bytes**. The start address of the output variable with the lowest offset is **0**.

2. Right-click anywhere in the **Output Variables** area to open the context menu.
3. Click **New Offsets** to re-generate the variable offsets.

### Inputs in the HIMA PROFIBUS DP Slave

| Name             | Type  | Offset | Global Variable  |
|------------------|-------|--------|------------------|
| PB_Master_Slave1 | DWORD | 14     | PB_Master_Slave1 |
| PB_Master_Slave2 | BYTE  | 18     | PB_Master_Slave2 |

Table 72: Inputs in the HIMA PROFIBUS DP Slave

1. Drag the global variables to be received from the Object Panel onto the **Input Variables** area.



In this example, the input variables of the HIMA PROFIBUS DP slave are composed of **two variables** with a total of **5 bytes**. The start address of the input variable with the lowest offset is **0**.

2. Right-click anywhere in the the **Input Variables** area to open the context menu.
3. Click **New Offsets** to re-generate the variable offsets.

### To verify the configuration of the PROFIBUS DP slave

1. In the structure tree, open **Configuration, Resource, Protocols, PROFIBUS DP Slave**.
2. Click the **Verification** button on Action Bar, and then click **OK** to confirm the action.

- 
3. Thoroughly verify the messages displayed in the logbook and correct potential errors.

- 
- i** Use the user program of the PROFIBUS DP slave resource to once again compile the configuration of the PROFIBUS DP slave and load it to the controllers. Only after this step, the new configuration can be used for communication with the PROFIBUS DP
- 

## 6.2.2 Configuring the PROFIBUS DP Master

### To create a new HIMA PROFIBUS DP master

1. In the structure tree, open **Configuration, Resource, Protocols**.
2. Select **New, PROFIBUS DP Master** from the context menu for protocols to add a new PROFIBUS DP master.
3. Select **Properties, General** from the context menu for the PROFIBUS DP master.
4. In the **General** tab, select **COM Module and Interfaces** (e.g., FB1).

- 
- i** Perform these steps to configure the HIMax PROFIBUS DP slave from within the HIMax PROFIBUS DP master.
- 

### To create a new HIMax PROFIBUS DP Slave in the PROFIBUS DP master

1. Select **New, PROFIBUS DP Slave** from the context menu for the PROFIBUS DP master.

### To read the GSD file for the new PROFIBUS DP slave

1. In the structure tree, open **Configuration, Resource, Protocols, PROFIBUS DP Master, PROFIBUS Slave**.
2. Select **Read GSD File** from the context menu for the PROFIBUS DP master and choose the GSD file for the PROFIBUS slave (e.g., hax100ea.gsd).

- 
- i** The GSD files for HIMax controllers are available on HIMA website at [www.hima.com](http://www.hima.com).
-

### 6.2.2.1 Creating the HIMax PROFIBUS DP Modules

The number of bytes that must actually be transferred, must be configured in the PROFIBUS DP master. To do this, add *Modules* until the physical configuration of the slave is achieved.



The number of modules used to achieve the necessary number of bytes is not important as long as the maximum of 32 modules is not exceeded.

To avoid unnecessarily complicating the PROFIBUS DP master configuration, HIMA recommends keeping the number of selected modules to a minimum.

#### To create the required PROFIBUS DP modules

1. In the structure tree, open **Configuration, Resource, Protocols, PROFIBUS DP Master, PROFIBUS Slave**.
2. On the menu bar, select **PROFIBUS DP Master, Add modules**.
3. For this example, select the following modules to receive **11 bytes** from the PROFIBUS DP slave and to send 3 bytes.

#### To number the PROFIBUS DP modules

1. Right-click the first **PROFIBUS DP Module**, and then click **Properties**.
2. Enter **0** into the **Slot** field.
3. Repeat these steps for every further **PROFIBUS DP Module** and number the modules consecutively.

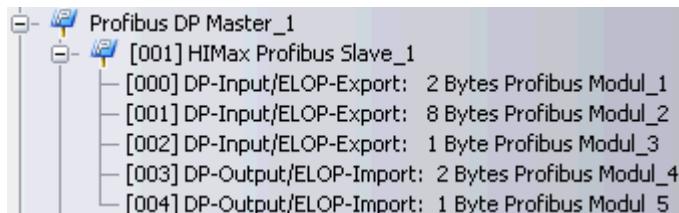


Figure 37: HIMax PROFIBUS DP Slave with Modules



**Number** the HIMax PROFIBUS DP modules without gaps and in ascending order, starting with **0**.

The order in which the PROFIBUS DP modules are arranged is not important for operation. However, HIMA recommends organizing the DP input and output modules in an orderly manner to ensure than an overview is maintained.

### 6.2.2.2 Configuring the Input and Output Modules

- i** The sum of the variables (in bytes), must identical with the size of the module (in bytes).

#### To configure the input module [000] DP Input/ELOP Export: 2 bytes

1. In the PROFIBUS DP slave, select the input module **[000] DP Input/ELOP Export: 2 Bytes**
2. Right-click the input module, then click **Edit**.
3. In the **Edit** dialog box, select the **Process Variables** tab.
4. Drag the suitable variable from the Object Panel onto the **Input Signals** area of the input module **[000] DP-Input/ELOP-Export: 2 Bytes**.

| Name             | Type | Offset | Global Variable  |
|------------------|------|--------|------------------|
| PB_Slave_Master1 | UINT | 0      | PB_Slave_Master1 |

Table 73: Variables of the Input Module [000] DP Input/ELOP Export: 2 Bytes

5. Right-click anywhere in the **Input Signals** area to open the context menu.
6. Click **New Offsets** to re-generate the variable offsets.

#### To configure the input module [001] DP Input/ELOP Export: 8 bytes

1. In the PROFIBUS DP slave, select the input module **[001] DP Input/ELOP Export: 8 Bytes**
2. Right-click the input module, then click **Edit**.
3. In the **Edit** dialog box, select the **Process Variables** tab.
4. Drag the suitable variable from the Object Panel onto the **Input Signals** area of the input module **[001] DP-Input/ELOP-Export: 8 Bytes**.

| Name             | Type  | Offset | Global Variable  |
|------------------|-------|--------|------------------|
| PB_Slave_Master2 | DWORD | 0      | PB_Slave_Master2 |
| PB_Slave_Master3 | DWORD | 4      | PB_Slave_Master3 |

Table 74: Variables of the Input Module [001] DP Input/ELOP Export: 8 Bytes

5. Right-click anywhere in the **Input Signals** area to open the context menu.
6. Click **New Offsets** to re-generate the variable offsets.

#### To configure the input module [002] DP Input/ELOP Export: 1 byte

1. In the PROFIBUS DP slave, select the input module **[002] DP Input/ELOP Export: 1 Byte**
2. Right-click the input module, then click **Edit**.
3. In the **Edit** dialog box, select the **Process Variables** tab.
4. Drag the suitable variable from the Object Panel onto the **Input Signals** area of the input module **[002] DP-Input/ELOP-Export: 1 Byte**.

| Name             | Type | Offset | Global Variable  |
|------------------|------|--------|------------------|
| PB_Slave_Master4 | BYTE | 0      | PB_Slave_Master4 |

Table 75: Variables of the Input Module [002] DP Input/ELOP Export: 1 Byte

5. Right-click anywhere in the **Input Signals** area to open the context menu.
6. Click **New Offsets** to re-generate the variable offsets.

**To configure the output module [003] DP Output/ELOP Import: 2 Bytes**

1. In the PROFIBUS DP slave, select the output module **[003] DP Output/ELOP Import: 2 Bytes**
2. Right-click the output module, then click **Edit**.
3. In the **Edit** dialog box, select the **Process Variables** tab.
4. Drag the suitable variable from the Object Panel onto the **Output Signals** area of the output module **[003] DP-Output/ELOP-Import: 2 Bytes**.

| Name             | Type | Offset | Global Variable  |
|------------------|------|--------|------------------|
| PB_Master_Slave1 | UINT | 0      | PB_Master_Slave1 |

Table 76: Variables of the Output Module [003] DP Output/ELOP Import: 2 Bytes

5. Right-click anywhere in the **Output Signals** area to open the context menu.
6. Click **New Offsets** to re-generate the variable offsets.

**To configure the output module [004] DP Output/ELOP Import: 1 Byte**

1. In the PROFIBUS DP slave, select the output module **[004] DP Output/ELOP Import: 1 Byte**
2. Right-click the output module, then click **Edit**.
3. In the **Edit** dialog box, select the **Process Variables** tab.
4. Drag the suitable variable from the Object Panel onto the **Output Signals** area of the output module **[004] DP-Output/ELOP-Import: 1 Byte**.

| Name             | Type | Offset | Global Variable  |
|------------------|------|--------|------------------|
| PB_Master_Slave2 | BYTE | 0      | PB_Master_Slave2 |

Table 77: Variables of the Output Module [004] DP Output/ELOP Import: 1 Byte

5. Right-click anywhere in the **Output Signals** area to open the context menu.
6. Click **New Offsets** to re-generate the variable offsets.

**6.2.2.3 Creating the User Data within the PROFIBUS DP Master****To create the user data in the PROFIBUS DP master**

1. In the structure tree, open **Configuration, Resource, Protocols, PROFIBUS DP Master, PROFIBUS DP Slave**.
2. Right-click **PROFIBUS Slave**, and then click **Properties**.
3. Select the **Data** tab and click the **Edit** button next to the user data.

The group's start address and number of variables are defined in the 32 bytes long user data field (see also Chapter 6.8).

4. For this example, create the following user data:  
 4, to ensure that **four variables** are received by the PROFIBUS DP master.  
 2, to ensure that two variables that are sent by the PROFIBUS DP master.  
 The start address of the input and output groups begins with **0**.

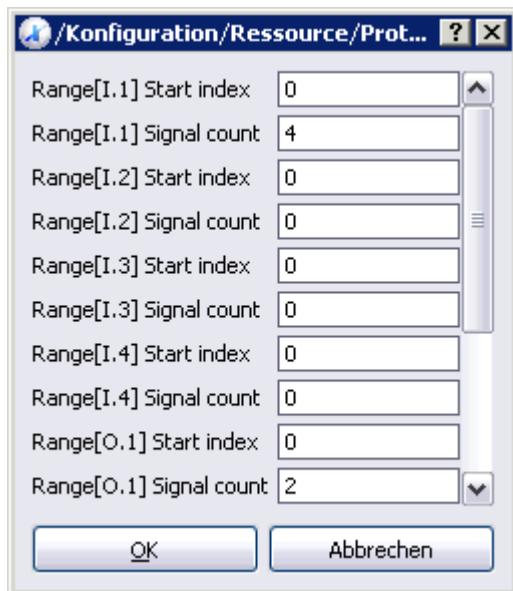


Figure 38: User Data Field

#### To verify the configuration of the PROFIBUS DP slave

1. In the structure tree, open **Configuration, Resource, Protocols, PROFIBUS DP Master**.
2. Click the **Verification** button located on Action Bar, and then click **OK** to confirm the action.
3. Thoroughly verify the messages displayed in the logbook and correct potential errors.

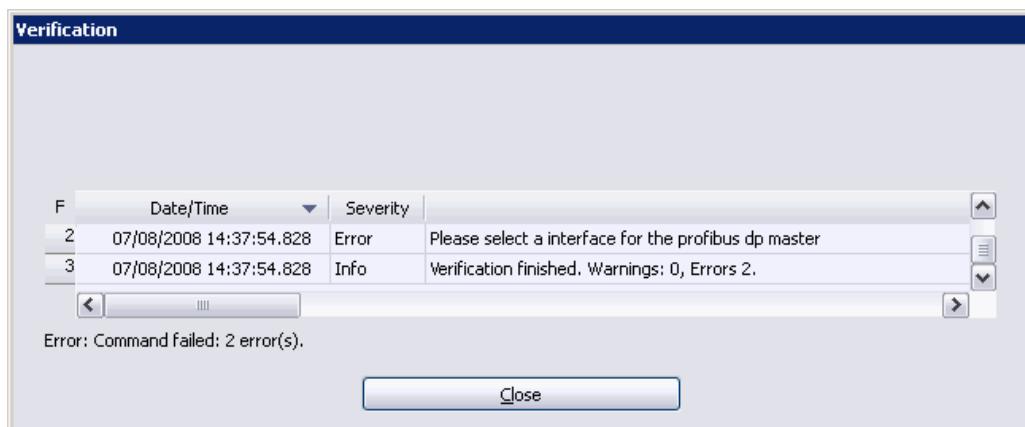


Figure 39: Verification Dialog Box



Use the user program of the PROFIBUS DP master resource to recompile the configuration of the PROFIBUS DP master and transfer it to the controllers. Only after this step, the new configuration can be used for communication with the PROFIBUS DP

#### 6.2.2.4 Optimizing the PROFIBUS DP parameters

Using the default values for the PROFIBUS parameters, smooth PROFIBUS communication is generally not a problem. However, the settings should be further optimized to achieve faster data exchange rates and improve fault detection.

**To determine the actual target rotation time TTR [ms]**

1. Open the Control Panel associated with the HIMax PROFIBUS DP master controller.
2. In the structure tree associated with the Control Panel, click **PROFIBUS DP Master** and read the actual **Target Rotation Time TTR [ms]**. Note down this value.

**To determine the parameters required for the PROFIBUS DP slave**

1. In the structure tree, open **Configuration, Resource, Protocols, PROFIBUS DP Master, PROFIBUS DP Slave**.
2. Right-click **HIMax PROFIBUS Slave**, and then click **Properties**.
3. Select the **Features** tab and read **Min. Slave Interval MSI [ms]** for this PROFIBUS DP slave. Note down this value.
4. Select the **Transfer Rate** tab and read **Max. Tsdr** for the transfer rate used. Note down this value.

**To enter the parameters previously determined**

1. Right-click **PROFIBUS Master**, and then click **Properties**.
2. Select the **Timings** tab.

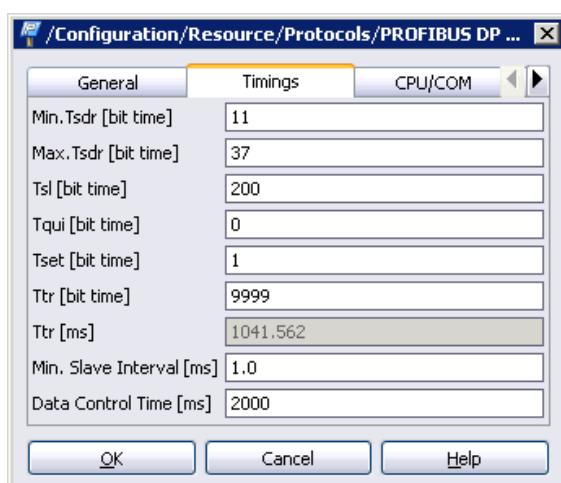


Figure 40: PROFIBUS DP Master Properties

3. Convert the **Max. Tsdr** that was previously noted down in **bit Time**
4. Convert the **Target Rotation Time TTR [ms]** that was previously noted down in **bit Time**, add 1/3 safety margin and enter the resulting value in the **Target Rotation Time TTR [ms]** field.
5. Enter the **Min. Slave Interval MSI [ms]** that was previously noted down.

**i** If various slaves are configured, the highest values of the parameters MaxTsdr [bit time] and Min. Slave Interval [ms] must be used.

6. The data control time [ms] must be set to  $\geq 6 \cdot Ttr$  ms

**To enter the watchdog time for the PROFIBUS DP slave**

1. In the structure tree, open **Configuration, Resource, Protocols, PROFIBUS DP Master, PROFIBUS DP Slave..**
2. Right-click **HIMax PROFIBUS Slave**, and then click **Properties**.

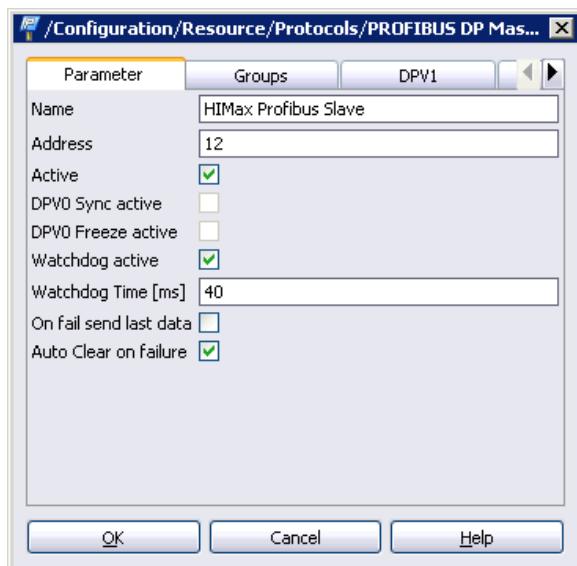


Figure 41: PROFIBUS DP Slave Properties

3. Select the **Parameter** tab and mark the **Watchdog Active** checkbox.
4. Enter the watchdog time [ms]  $\geq 6 \cdot T_{tr}$  [ms] in the **Watchdog Time [ms]** field.



Use the user program of the PROFIBUS DP master and slave resources to recompile the configurations of the PROFIBUS DP master and slave and transfer them to the controllers. Only after this step, the new configurations can be used for communication with the PROFIBUS DP.

## 6.3 Menu Functions of the PROFIBUS DP Master

### 6.3.1 Edit

The **Edit** function of the context menu for the PROFIBUS DP master is used to open the **Edit** dialog box.

The **System Variables** tab contains the following system variables that are required to evaluate the state of the PROFIBUS DP master from within the user program.

| Element              | Description  |      |             |   |                  |   |  |   |   |   |   |   |  |   |   |   |   |
|----------------------|--|------|-------------|---|------------------|---|--|---|---|---|---|---|--|---|---|---|---|
| Number of IO Errors  | Number of errors since statistics reset.   |      |             |   |                  |   |  |   |   |   |   |   |  |   |   |   |   |
| Baud Rate            | Baud rate (bit/s) used for the bus.  |      |             |   |                  |   |  |   |   |   |   |   |  |   |   |   |   |
| Bus Error            | If a bus error occurs, an error code is set in the <i>Bus Error</i> system variable. An error code retains its value until the bus error has been eliminated. <table border="1" data-bbox="603 718 1270 1392"> <thead> <tr> <th>Code</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>OK, no bus error</td></tr> <tr> <td>1</td><td>Address error:<br/>The master address is already available on the bus</td></tr> <tr> <td>2</td><td>Bus malfunction<br/>Malfunction detected on the bus, (e.g., bus not properly terminated, multiple stations are sending data simultaneously).</td></tr> <tr> <td>3</td><td>Protocol error<br/>An incorrectly coded packet was received.</td></tr> <tr> <td>4</td><td>Hardware fault<br/>The hardware reported a fault, e.g., too short time periods.</td></tr> <tr> <td>5</td><td>Unknown error<br/>The master changed the status for an unknown reason.</td></tr> <tr> <td>6</td><td>Controller reset<br/>The controller chip is reset if a serious bus error occurs.</td></tr> </tbody> </table> To evaluate the <i>Bus Error</i> status variable from within the user program, it must be connected to a variable. | Code | Description | 0 | OK, no bus error | 1 | Address error:<br>The master address is already available on the bus | 2 | Bus malfunction<br>Malfunction detected on the bus, (e.g., bus not properly terminated, multiple stations are sending data simultaneously). | 3 | Protocol error<br>An incorrectly coded packet was received. | 4 | Hardware fault<br>The hardware reported a fault, e.g., too short time periods. | 5 | Unknown error<br>The master changed the status for an unknown reason. | 6 | Controller reset<br>The controller chip is reset if a serious bus error occurs. |
| Code                 | Description  |      |             |   |                  |   |  |   |   |   |   |   |  |   |   |   |   |
| 0                    | OK, no bus error   |      |             |   |                  |   |  |   |   |   |   |   |  |   |   |   |   |
| 1                    | Address error:<br>The master address is already available on the bus   |      |             |   |                  |   |  |   |   |   |   |   |  |   |   |   |   |
| 2                    | Bus malfunction<br>Malfunction detected on the bus, (e.g., bus not properly terminated, multiple stations are sending data simultaneously).  |      |             |   |                  |   |  |   |   |   |   |   |  |   |   |   |   |
| 3                    | Protocol error<br>An incorrectly coded packet was received.  |      |             |   |                  |   |  |   |   |   |   |   |  |   |   |   |   |
| 4                    | Hardware fault<br>The hardware reported a fault, e.g., too short time periods.   |      |             |   |                  |   |  |   |   |   |   |   |  |   |   |   |   |
| 5                    | Unknown error<br>The master changed the status for an unknown reason.  |      |             |   |                  |   |  |   |   |   |   |   |  |   |   |   |   |
| 6                    | Controller reset<br>The controller chip is reset if a serious bus error occurs.  |      |             |   |                  |   |  |   |   |   |   |   |  |   |   |   |   |
| Average cycle time   | Measured average bus cycle time in milliseconds.   |      |             |   |                  |   |  |   |   |   |   |   |  |   |   |   |   |
| Last cycle time      | Measured bus cycle time in milliseconds.   |      |             |   |                  |   |  |   |   |   |   |   |  |   |   |   |   |
| Master State         | Indicate the current protocol state.<br>0: OFFLINE<br>1: STOP<br>2: CLEAR<br>3: OPERATE<br>Connect the status variable <i>Master State</i> to a variable to evaluate it in the user program.   |      |             |   |                  |   |  |   |   |   |   |   |  |   |   |   |   |
| Maximum Cycle Time   | Measured maximum bus cycle time in milliseconds.   |      |             |   |                  |   |  |   |   |   |   |   |  |   |   |   |   |
| Min. Slave Interval  | Minimum slave interval measured for one of the slaves assigned to this master.   |      |             |   |                  |   |  |   |   |   |   |   |  |   |   |   |   |
| Minimum Cycle Time   | Measured minimum bus cycle time in milliseconds.   |      |             |   |                  |   |  |   |   |   |   |   |  |   |   |   |   |
| Target Rotation Time | Target token rotation time   |      |             |   |                  |   |  |   |   |   |   |   |  |   |   |   |   |

Table 78: System Variables in the PROFIBUS DP Master

### 6.3.2 Menu Function: Properties

The **Properties** function of the context menu for the PROFIBUS DP master is used to open the **Properties** dialog box.

The dialog box contains the following tabs:

## 6.3.2.1 Tab: General

| Element                             | Description   |       |           |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
|-------------------------------------|---|-------|-----------|-----|-----|------|------------|---|---|-------|-------------|---|---|-------|--------------|---|---|-------|--------------|---|---|--------|--------------|---|---|--------|------------|---|---|---------|------------|---|---|---------|----------|---|---|---------|----------|---|---|----------|-----------|---|---|
| Type                                | PROFIBUS DP master  |       |           |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
| Name                                | Any unique name for a PROFIBUS DP master  |       |           |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
| Module                              | Selection of the COM module within which the protocol is processed.   |       |           |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
| Max. $\mu$ P Budget                 | Activated:<br>Adopt the $\mu$ P budget limit from the <i>Max. <math>\mu</math>P Budget in [%]</i> field.<br>Deactivated:<br>Do not use the $\mu$ P budget limit for this protocol.  |       |           |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
| Max. $\mu$ P budget in [%]          | Maximum $\mu$ P budget for the module that can be used for processing the protocols.<br><br>Range of values: 1...100%<br>Default value: 30%   |       |           |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
| Behavior on CPU/COM Connection Loss | If the connection of the processor module to the communication module is lost, the input variables are initialized or continue to be used unchanged in the process module, depending on this parameter.<br><br>For instance, if the communication module is removed when communication is running.<br><br><b>If a project created with a SILworX version prior to V3 should be converted, this value must be set to Retain Last Value to ensure that the CRC does not change.</b><br><b>For HIMatrix controllers with CPU OS version prior to V8, this value must always be set to Retain Last Value.</b><br><br>Adopt initial data      Input variables are reset to their initial values.<br>Retain Last Value      The input variables retains the last value.   |       |           |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
| Address                             | Master station address.<br>Only one master station address may be available on the bus.<br><br>Range of values: 0...125<br>Default value: 0   |       |           |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
| Interface                           | COM interface that should be used for the master.<br>Range of values: FB1, FB2  |       |           |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
| Baud Rate                           | Baud rate (bit/s) used for the bus.<br>Possible values:<br><table border="1"> <thead> <tr> <th>Value</th> <th>Baud Rate</th> <th>FB1</th> <th>FB2</th> </tr> </thead> <tbody> <tr> <td>9600</td> <td>9.6 kbit/s</td> <td>X</td> <td>X</td> </tr> <tr> <td>19200</td> <td>19.2 kbit/s</td> <td>X</td> <td>X</td> </tr> <tr> <td>45450</td> <td>45.45 kbit/s</td> <td>X</td> <td>X</td> </tr> <tr> <td>93750</td> <td>93.75 kbit/s</td> <td>X</td> <td>X</td> </tr> <tr> <td>187500</td> <td>187.5 kbit/s</td> <td>X</td> <td>X</td> </tr> <tr> <td>500000</td> <td>500 kbit/s</td> <td>X</td> <td>X</td> </tr> <tr> <td>1500000</td> <td>1.5 Mbit/s</td> <td>X</td> <td>X</td> </tr> <tr> <td>3000000</td> <td>3 Mbit/s</td> <td>X</td> <td>-</td> </tr> <tr> <td>6000000</td> <td>6 Mbit/s</td> <td>X</td> <td>-</td> </tr> <tr> <td>12000000</td> <td>12 Mbit/s</td> <td>X</td> <td>-</td> </tr> </tbody> </table> | Value | Baud Rate | FB1 | FB2 | 9600 | 9.6 kbit/s | X | X | 19200 | 19.2 kbit/s | X | X | 45450 | 45.45 kbit/s | X | X | 93750 | 93.75 kbit/s | X | X | 187500 | 187.5 kbit/s | X | X | 500000 | 500 kbit/s | X | X | 1500000 | 1.5 Mbit/s | X | X | 3000000 | 3 Mbit/s | X | - | 6000000 | 6 Mbit/s | X | - | 12000000 | 12 Mbit/s | X | - |
| Value                               | Baud Rate   | FB1   | FB2       |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
| 9600                                | 9.6 kbit/s  | X     | X         |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
| 19200                               | 19.2 kbit/s   | X     | X         |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
| 45450                               | 45.45 kbit/s  | X     | X         |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
| 93750                               | 93.75 kbit/s  | X     | X         |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
| 187500                              | 187.5 kbit/s  | X     | X         |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
| 500000                              | 500 kbit/s  | X     | X         |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
| 1500000                             | 1.5 Mbit/s  | X     | X         |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
| 3000000                             | 3 Mbit/s  | X     | -         |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
| 6000000                             | 6 Mbit/s  | X     | -         |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
| 12000000                            | 12 Mbit/s   | X     | -         |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |

Table 79: General Properties for PROFIBUS DP Master

### 6.3.2.2 Tab: Timings

| Element                     | Description  |
|-----------------------------|--|
| MinTsdr<br>[bit time]       | Min. Station Delay Time:<br>Minimum time period that a PROFIBUS DP slave must wait before it may respond.<br>Range of values: 11...1023<br>Default value: 11   |
| MaxTsdr<br>[bit time]       | Max. Station Delay Time:<br>Maximum time period that a PROFIBUS DP slave may need to respond.<br>$\text{Max Tsdr} \geq \text{Tsdr}$ (of the connected slave with the highest Tsdr)<br>The MaxTsdr values of the slaves are read from the GSD files and are displayed in the <b>Baud Rates</b> tab located in the slave's <b>Properties</b> dialog box<br>Range of values: 37...65535<br>Default value: 100   |
| Tsl<br>[bit time]           | Slot Time<br>Maximum time span that the master waits for a slave's acknowledgment.<br>$\text{Tsl} > \text{MaxTsdr} + 2 * \text{Tset} + \text{Tqui} + 13$<br>Range of values: 37...16383<br>Default value: 200  |
| Tqui<br>[bit time]          | Quiet Time for Modulator<br>Time that a station may need to switch from sending to receiving.<br>Range of values: 0...493<br>Default value: 0  |
| Tset<br>[bit time]          | Setup Time<br>Time for reacting to an event.<br>Range of values: 1...494<br>Default value: 1   |
| Ttr<br>[bit time]           | Time configured for a token cycle.<br>Maximum time available for a token rotation.<br>A lower estimate of the Ttr can be obtained with a specific calculation, see Chapter 6.4.4.<br>Range of values: 256...16777215<br>Default value: 9999  |
| Ttr [ms]                    | Actual token rotation time in ms   |
| Min. Slave Interval<br>[ms] | Minimum time between two cyclical requests of a slave. The master observes the Min. Slave Interval and does not fall below it.<br>However, the PROFIBUS DP cycle can be extended if Isochronous Mode is inactive and the portion of acyclic telegrams increases within a cycle. The value for the <i>Min. Slave Interval</i> is read from the GSD file and appears in the <b>Features</b> tab located in the <b>Properties</b> dialog box. In Isochronous Mode, the value for <i>Min. Slave Interval</i> defines the time period for an isochronous cycle.<br>Isochronous Mode is activated if the options Isochronous Sync Mode or Isochronous Freeze Mode are activated.<br>See also Refresh Rate between CPU and COM (CPU/COM tab).<br>Range of values: 0...6553.5 (step size 0.1 ms)<br>Default value: 1.0 |

| Element                        | Description  |
|--------------------------------|--|
| User Data Monitoring Time [ms] | Time span within which the master must report its current state on the bus<br>Standard value: User Data Monitoring Time = WDT of the slave<br>Range of values: 0...655350 (step size 10 ms)<br>Default value: 2000 |

Table 80: Timings Tab in the Properties Dialog Box for the PROFIBUS DP Master

### 6.3.2.3 Tab: CPU/COM

The default values of the parameters provide the fastest possible data exchange of PROFIBUS DP data between the COM module (COM) and the processor module (CPU) within the HIMax controller. These parameters should only be changed if it is necessary to reduce the COM or CPU load for an application, and the process allows this change.

- 
- i** Only experienced programmers should modify the parameters. Increasing the COM and CPU refresh rate means that the effective refresh rate of the PROFIBUS DP data is also increased. The system time requirements must be verified.
- 

Take also the parameter *Min. Slave Interval [ms]* into account which defines the refresh rate of the PROFIBUS DP data from/to the PROFIBUS DP slave. The refresh rate of the PROFIBUS DP data can be increased according to the CPU/COM refresh rate.

| Element                        | Description  |
|--------------------------------|--|
| Process Data Refresh Rate [ms] | Refresh rate in milliseconds at which the COM and CPU exchange protocol data. If the <i>Refresh Rate</i> is zero or lower than the cycle time for the controller, data is exchanged as fast as possible. Range of values: 0...(2 <sup>31</sup> -1)<br>Default value: 0   |
| Force Process Data Consistency | Activated:<br>Transfer of all protocol data from the CPU to the COM within a CPU cycle.<br><br>Deactivated:<br>Transfer of all protocol data from the CPU to the COM, distributed over multiple CPU cycles, each with 1100 byte per data direction. This can also allow lowering the cycle time of the controller.<br><br>Default value: activated |

Table 81: CPU/COM Tab in the Properties Dialog Box for the PROFIBUS DP Master

## 6.3.2.4 Tab: Other

| Element                  | Description   |
|--------------------------|---|
| Max. Number of Resends   | Maximum number of resends attempted by a master if a slave does not respond.<br>Range of values: 0...7<br>Default value: 1  |
| Highest Active Address   | Highest Station Address (HSA)<br>Highest station address to be expected for one master. Masters having a station address beyond the HSA are not included into the token ring.<br>Range of values: 0...125<br>Default value: 125   |
| Isochronous Sync Mode    | Isochronous Sync Mode allows both a clock-controlled synchronization of the master and the slaves and a simultaneous activation of the physical outputs of multiple slaves.<br>If Isochronous Sync Mode is active, the master sends the Sync control command as a broadcast telegram to all slaves. As soon as the slaves supporting Isochronous Sync Mode receive the "Sync" control command, they synchronously switch the data from the user program to the physical outputs.<br>The physical outputs' values remain frozen up to the next Sync control command.<br>The cycle time is determined by the Min. Slave Interval.<br>Condition: Ttr < Min. Slave Interval<br><br>Default value: Deactivated |
| Isochronous Freeze Mode  | Isochronous Freeze Mode allows the user to simultaneously accept the input data of multiple slaves.<br>If Isochronous Freeze Mode is active, the master sends the "Freeze" control command as a broadcast telegram to all slaves. As soon as the slaves supporting Isochronous Freeze Mode receive the "Freeze" control command , the physical inputs' variables are frozen to the current value.<br>The master can thus read the values. The input data is only updated when the next Freeze control command is sent.<br>The cycle time is determined by the "Min. Slave Interval".<br>Condition: Ttr < Min. Slave Interval<br><br>Default value: Deactivated  |
| Auto Clear on Error      | If Auto Clear on Error is set in a slave that fails, the master adopts the CLEAR state.<br><br>Default value: Deactivated   |
| Time Master              | The master is also time master and periodically sends the system time via the bus.<br>Default value: Deactivated  |
| Clock Sync Interval [ms] | Clock synchronization interval.<br>Time interval within which the time master sends the system time over the bus.<br>Range of values: 0...655350 (step size 10 ms)<br>Default value: 0 (no time master)   |

Table 82: Other Properties for the PROFIBUS DP Master

## 6.4 PROFIBUS DP Bus Access Method

The bus access method provides a defined time window to every station within which the station can perform its communication tasks.

### 6.4.1 Master/Slave Protocol

The bus assignment between a PROFIBUS DP master and a PROFIBUS DP slave is ensured by the master/slave method.

An active PROFIBUS DP master communicates with passive PROFIBUS DP slaves.

The PROFIBUS DP master with the token is authorized to send and may communicate with the PROFIBUS DP slaves assigned to it. The master assigns the bus to a slave for a certain time and the slave must respond within this time period.

### 6.4.2 Token Protocol

The bus assignment between automation devices (class 1 masters) and/or programming devices (class 2 masters) is ensured via token passing.

All PROFIBUS DP masters connected to a common bus form a token ring. As long as the active PROFIBUS DP master has the token, it assumes the master function on the bus.

In a token ring, the PROFIBUS DP masters are organized in ascending order according to their station addresses. The token is passed on in this order until it is received by the PROFIBUS DP master with the highest station address.

This master passes the token on to the master with the lowest station address to close the token ring.

The token rotation time corresponds to one token cycle through all the PROFIBUS DP masters. The target rotation time (Ttr) is the maximum time permitted for a token cycle.

### 6.4.3 Target Token Rotation Time (Ttr)

Default Values for different transfer rates

While configuring the PROFIBUS DP master, take into account that some parameters set in the **Timings** tab depend on the baud rate set in the **General** tab. For the first (initial) configuration, use the default values specified in the following table. The values are optimized at a later point in time.

|               | 9.6k | 19.2k | 45.45k | 93.75k | 187.5k | 500k | 1.5M | 3M  | 6M  | 12M  |
|---------------|------|-------|--------|--------|--------|------|------|-----|-----|------|
| MinTsdr       | 11   | 11    | 11     | 11     | 11     | 11   | 11   | 11  | 11  | 11   |
| MaxTsdr       | 60   | 60    | 400    | 60     | 60     | 100  | 150  | 250 | 450 | 800  |
| Tsl bit time  | 100  | 100   | 640    | 100    | 100    | 200  | 300  | 400 | 600 | 1000 |
| Tqui bit time | 0    | 0     | 0      | 0      | 0      | 0    | 0    | 3   | 6   | 9    |
| Tset bit time | 1    | 1     | 95     | 1      | 1      | 1    | 1    | 4   | 8   | 16   |

Table 83: Default Values for Token Rotation Time Used with Different Transfer Rates

All time values specified are expressed in Tbit (1Tbit = 1/[bit/s]).

MinTsdr is at least 11 Tbits long as a character has 11 bits (1 start bit, 1 stop bit, 1 parity bit, 8 data bits).

Transmission Time for a Character

| Baud Rate  | Tbit bit = 1/baud rate      | Time                              |
|------------|-----------------------------|-----------------------------------|
| 9600 bit/s | $1 / 9600 = 104.166 \mu s$  | $11 * 104.166 \mu s = 1.14583 ms$ |
| 6 Mbit/s   | $1 / 6 * 1066 = 166.667 ns$ | $11 * 166.667 ns = 1.833 \mu s$   |

Table 84: Transmission Time for a Character Used with different Transfer Rates

#### 6.4.4 Calculating the Target Token Rotation Time (Ttr)

Calculate the minimum target token rotation time Ttr as follows:

$$Ttr_{min} = n * (198 + T1 + T2) + b * 11 + 242 + T1 + T2 + Tsl$$

| Element | Description   |
|---------|---|
| n       | Number of active slaves   |
| b       | Number of I/O data bytes of the active slaves (input plus output)           |
| T0      | $35 + 2 * Tset + Tqui$  |
| T1      | If $T0 < MinTsdr$ : $T1 = MinTsdr$<br>If $T0 > MinTsdr$ : $T1 = T0$         |
| T2      | If $T0 < MaxTsdr$ : $T2 = MaxTsdr$<br>If $T0 > MaxTsdr$ : $T2 = T0$         |
| Tsl     | Slot Time: Maximum time period that the master waits for a slave to respond |
| 198     | Twice a telegram header with variable length (for request and response)     |
| 242     | Global_Control, FDL_Status_Req and token passing                            |

Table 85: Elements Required for Calculating the Target Token Rotation Time



The estimate of the token rotation time  $Ttr$  is only valid if the following conditions are met: only one master is operating on the bus, no transmissions are repeated and no acyclic data is transmitted.

Never set  $Ttr$  to a value less than that calculated with the above formula. Otherwise fault-free operation can no longer be ensured. HIMA recommends using a value two or three times greater than the result.

#### Example of Calculating the Token Rotation Time Ttr

The following configuration is available:

5 active slaves

(n = 5)

20 I/O data bytes per slave

(b = 100)

The following time constants for a transmission rate of 6 Mbit/s are taken from Table 85:

- $MinTsdr = 11 T_{bit}$
- $MaxTsdr = 450 T_{bit}$
- $Tsl$  bit time =  $600 T_{bit}$
- $Tqui$  bit time =  $6 T_{bit}$
- $Tset$  bit time =  $8 T_{bit}$

$$T0 = 35 + 2 * Tset + Tqui$$

$$T0 = 35 + 2 * 8 + 6$$

$$T_0 = 57 \text{ T}_{\text{bit}}$$

As  $T_0 > \text{MinTsdr}$ :  **$T_1 = T_0 = 57 \text{ T}_{\text{bit}}$**

As  $T_0 < \text{MaxTsdr}$ :  **$T_2 = \text{MaxTsdr} = 450 \text{ T}_{\text{bit}}$**

Use the computed values in the formula for the minimum target token rotation time:

$$T_{tr,\min} = n * (198 + T_1 + T_2) + b * 11 + 242 + T_1 + T_2 + TsI$$

$$T_{tr,\min} = 5 * (198 + 57 + 450) + 100 * 11 + 242 + 57 + 450 + 600$$

$$T_{tr,\min} [\text{T}_{\text{bit}}] = 5974 \text{ T}_{\text{bit}}$$

Result:

$$T_{tr,\min} [\mu\text{s}] = 5974 \text{ T}_{\text{bit}} * 166.67 \text{ ns} = 995.68 \mu\text{s}$$



*T<sub>tr</sub>* is verified when it is entered into the dialog box.

If the value set for *T<sub>tr</sub>* is lower than the value calculated by SILworX, an error message appears in the logbook. A minimum value for *T<sub>tr</sub>* is also suggested.

If *Isochronous Sync Mode* or *Isochronous Freeze Mode* is activated, the cycle time is defined by the parameter *MinSlaveInterval*. *T<sub>tr</sub>* must be lower than *Minimum Slave Interval*.

If this condition is not met in the isochronous mode, an error message appears.

---

## 6.5

### Isochronous PROFIBUS DP Cycle (DP V2 and Higher)

The PROFIBUS DP cycle consists of two telegram phases: a fixed and cyclical phase and an event-driven and acyclic phase.

The acyclic phase can extend the corresponding PROFIBUS DP cycle. This effect is not wanted in specific applications and areas, such as drive technology.

To achieve a constant cycle time ( $t_{\text{const}}$ ), Isochronous Mode is activated in the master such that *Min. Slave Interval [ms]* defines the constant cycle time ( $t_{\text{const}}$ ). Configured in this way, the isochronous PROFIBUS DP cycle offers clock accuracy with a difference of < 10 ms.

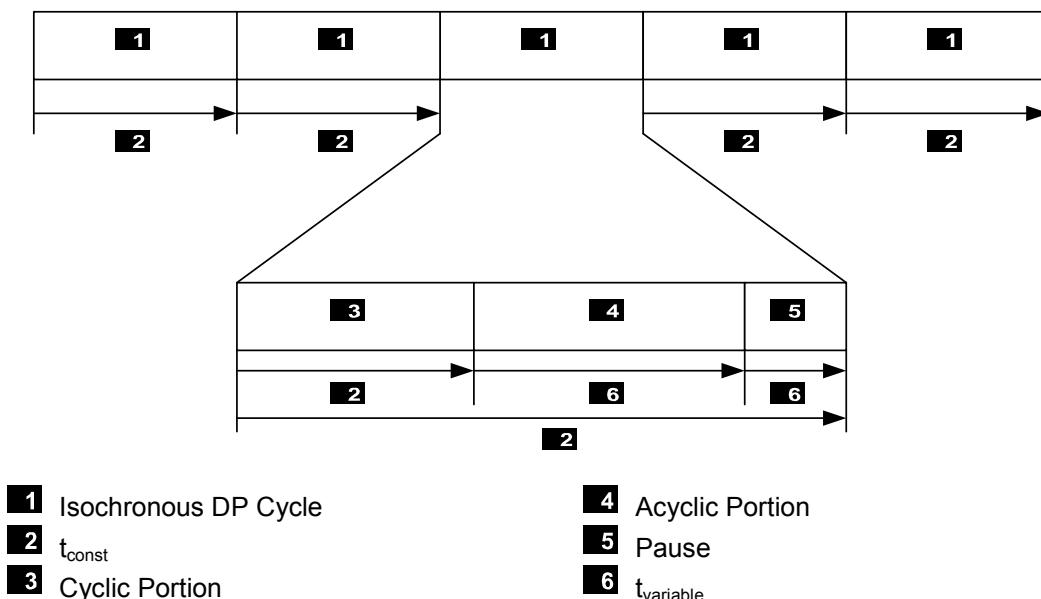


Figure 42: Isochronous PROFIBUS DP Cycle

To determine the cyclical phase, the minimum target token rotation time must be calculated.

Further, a sufficiently large time interval (typically two to three times the minimum target token rotation time  $T_{\text{tr}}$ ) must be reserved for the acyclic phase. If the reserved time is not needed, a break is taken prior to starting the next cycle to ensure the cycle time remains constant. See also Chapter 6.4.3, Target Token Rotation Time ( $T_{\text{tr}}$ ).



The master is configured entering the DP cycle time determined by the user into *Min. Slave Interval [ms]*.

To operate in the *Isochronous Mode*, one of the two parameters *Isochronous Sync Mode* or *Isochronous Freeze Mode* must be activated in the master.

On the bus, only one master may simultaneously operate in the Isochronous Mode. Additional masters are not permitted.

### 6.5.1 Isochronous Mode (DP V2 and higher)

This function allows a clock-controlled synchronization in the master and the slaves, irrespective of congestion on the bus. The bus cycle is synchronized with a clock difference of <10 ms. Highly precise positioning processes can be thus implemented.

- 
- i** To a certain degree, slaves (DP V0 slaves) that do not support *Isochronous Mode* can also benefit from its advantages. To do so, the slaves must be assigned to Group 8 and the parameters *Sync* and/or *Freeze* must be activated.  
The *Sync Mode* and *Freeze Mode* are normally used simultaneously.
- 

### 6.5.2 Isochronous Sync Mode (DP V2 and higher)

*Isochronous Sync Mode* allows both a clock-controlled synchronization of the master and the slave and a simultaneous activation of the outputs of multiple slaves.

### 6.5.3 Isochronous Freeze Mode (DP V2 and higher)

*Isochronous Freeze Mode* allows the user to simultaneously accept the input data of multiple slaves.

## 6.6 Menu Functions of the PROFIBUS DP Slave (in the Master)

### 6.6.1 Creating a PROFIBUS DP Slave (in the Master)

#### To create a PROFIBUS DP slave in the HIMA PROFIBUS DP Master

1. In the structure tree, open **Configuration, Resource, Protocols, PROFIBUS DP Master**.
2. On the context menu for PROFIBUS DP master, click **New, PROFIBUS Slave** to add a new PROFIBUS slave.

### 6.6.2 Edit

The **Edit** function of the context menu for the PROFIBUS DP master is used to open the **System Variables** dialog box.

The **System Variables** tab contains the following system variables that are required to evaluate the state of the PROFIBUS DP slave from within the user program.

| Element                  | Description   |       |             |   |  |   |   |   |   |  |
|--------------------------|---|-------|-------------|---|--|---|---|---|---|--|
| Activation Control       | A change from 0 to 1 deactivates the slave.<br>A change from 1 to 0 activates the slave previously deactivated.<br>Activated = 0<br>Deactivated = 1   |       |             |   |  |   |   |   |   |  |
| PNO Ident Number         | 16 bit unique number assigned by the PNO Germany to a product (field device) and identifying it.  |       |             |   |  |   |   |   |   |  |
| Standard Diagnosis       | With Standard Diagnosis, the slave informs the master about its current state. This variable always contains the last received standard diagnosis. The parameters comply with the diagnostic telegram in accordance with IEC 61158.   |       |             |   |  |   |   |   |   |  |
| Connection Count         | It increases with each new connection. It counts from count reset.  |       |             |   |  |   |   |   |   |  |
| Connection State         | <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Deactivated:<br/>The parameter sets are loaded for these slaves, but the slaves are completely ignored. The input data is reset to their initial values, no activity related to these slaves is noted on the bus.</td> </tr> <tr> <td>1</td> <td>Inactive (not connected):<br/>If a slave can no longer be reached, the input data is reset to their initial values.<br/>The following options can be selected for each slave:           <ul style="list-style-type: none"> <li>▪ The master continues to send output data or</li> <li>▪ the master attempts to re-configure the slave.</li> </ul> </td> </tr> <tr> <td>2</td> <td>Active (connected):<br/>The slaves are exchanging I/O data with the CPU.</td> </tr> </tbody> </table> | Value | Description | 0 | Deactivated:<br>The parameter sets are loaded for these slaves, but the slaves are completely ignored. The input data is reset to their initial values, no activity related to these slaves is noted on the bus. | 1 | Inactive (not connected):<br>If a slave can no longer be reached, the input data is reset to their initial values.<br>The following options can be selected for each slave: <ul style="list-style-type: none"> <li>▪ The master continues to send output data or</li> <li>▪ the master attempts to re-configure the slave.</li> </ul> | 2 | Active (connected):<br>The slaves are exchanging I/O data with the CPU. |  |
| Value                    | Description   |       |             |   |  |   |   |   |   |  |
| 0                        | Deactivated:<br>The parameter sets are loaded for these slaves, but the slaves are completely ignored. The input data is reset to their initial values, no activity related to these slaves is noted on the bus.  |       |             |   |  |   |   |   |   |  |
| 1                        | Inactive (not connected):<br>If a slave can no longer be reached, the input data is reset to their initial values.<br>The following options can be selected for each slave: <ul style="list-style-type: none"> <li>▪ The master continues to send output data or</li> <li>▪ the master attempts to re-configure the slave.</li> </ul>   |       |             |   |  |   |   |   |   |  |
| 2                        | Active (connected):<br>The slaves are exchanging I/O data with the CPU.   |       |             |   |  |   |   |   |   |  |
| Counter Slave Alarm      |   |       |             |   |  |   |   |   |   |  |
| Standard Diagnosis Count |   |       |             |   |  |   |   |   |   |  |

Table 86: System Variables in the PROFIBUS DP Slave

### 6.6.3 Properties

The **Properties** function of the context menu for the PROFIBUS DP slave is used to open the **Properties** dialog box. The dialog box contains the following tabs:

#### 6.6.3.1 Tab: Parameters

| Element                   | Description   |
|---------------------------|---|
| Name                      | Name of the slave   |
| Address                   | Address of the slave<br>Range of values: 0...125<br>Default value: 0  |
| Active                    | Slave State<br>Only an active slave can communicate with a PROFIBUS DP master.<br>Default value: activated  |
| DP V0 Sync active         | <b>Sync Mode</b> allows the user to simultaneously activate the outputs of various DP V0 slaves.<br><b>Important</b><br>This field must be deactivated in DP V2 slaves operating in <i>Isochronous Sync Mode</i> .<br>Default value: Deactivated  |
| DP V0 Freeze active       | <b>Freeze Mode</b> allows the user to simultaneously accept the input data of multiple DP V0 slaves.<br><b>Important</b><br>This field must be deactivated in DP V2 slaves operating in <i>Isochronous Freeze Mode</i> .<br>Default value: Deactivated  |
| Watchdog Active           | If the Watchdog Active checkbox is ticked, the slave detects a master's failure and enters the safe state.<br>Default value: Deactivated  |
| Watchdog Time [ms]        | The Watchdog Active checkbox must be ticked.<br>If master and slave do not exchange any data within this time interval, the slave disconnects itself and resets all DP output data to their initial values.<br>0 = Deactivated<br>Standard value: Slave's watchdog time > 6 * Ttr<br>Range of values: 0...65535<br>Default value: 0 |
| On failure send last data | FALSE: If a fault occurs, the connection is terminated and re-established.<br>TRUE: If a fault occurs, the data continues to be sent, even without the slave's acknowledgement.<br>Default value: Deactivated   |
| Auto Clear on Failure     | If <i>Auto Clear on Failure</i> is set to TRUE for the current slave and the current slave fails, the master switches the entire PROFIBUS DP to the safe state.<br>Default value: activated   |

Table 87: Parameters Tab in the PROFIBUS DP Slave

### 6.6.3.2 Tab: Groups

In this tab, the slaves can be organized into various groups. The global control commands, Sync and Freeze, can systematically address one or multiple groups.

| Element           | Description  |
|-------------------|--|
| Member of Group 1 | Member of Group 1<br>Member of Group 2<br>Member of Group 3<br>Member of Group 4<br>Member of Group 5<br>Member of Group 6<br>Member of Group 7<br>Member of Group 8 |
| Member of Group 2 |  |
| Member of Group 3 |  |
| Member of Group 4 |  |
| Member of Group 5 |  |
| Member of Group 6 |  |
| Member of Group 7 |  |
| Member of Group 8 |  |

Table 88: Groups Tab in the Properties Dialog Box for the PROFIBUS DP Slave

### 6.6.3.3 Tab: DP V1

This tab contains the parameters set with DP V1 and higher. In DP V0 slaves, no parameters can be selected in this tab. The Supp column shows which parameters are supported by the slave.

| Element                 | Description   |
|-------------------------|---|
| DP V1                   | If the DP V1 mode is not activated, no DP V1 features can be used. In this case, the slave acts like a DP V0 slave. The configuration data may have to be changed (refer to the slave manual).<br>Default value: Deactivated  |
| Failsafe                | If this mode is activated, a master in the CLEAR state does not send zeros as output data; rather, it sends an empty data packet (failsafe data packet) to the slave.<br>The slave recognizes that it must place the safe output data on the outputs (the value of the safe output data is not necessarily zero).<br>Default value: Deactivated |
| Isochronous Mode        | This function allows a clock-controlled synchronization in the master and the slaves, irrespective of congestion on the bus. The bus cycle is synchronized with a clock difference of < 1 ms. Highly precise positioning processes can be thus implemented.<br>Default value: Deactivated   |
| Publisher Active        | This function is required for the slave intercommunication. This allows the slaves to communicate with one another in a direct and time saving manner via broadcast without detouring through the master.<br>Default value: Deactivated   |
| Prm Block Struct. Supp. | The slave supports structured configuration data (read only).<br>Default value: Deactivated   |
| Check Cfg Mode          | Reduced configuration control: If Check Cfg Mode is activated, the slave can operate with an incomplete configuration.<br>This field should be deactivated during start-up.<br>Default value: Deactivated   |

Table 89: DP V1 Tab in the Properties Dialog Box for the PROFIBUS DP Slave

#### 6.6.3.4 Tab: Alarms

This tab is used to activate alarms. This, however, is only possible with DP V1 slaves, if DP V1 is activated and the slave supports alarms. The checkmarks in the **Supp** column designate which alarms are supported by the slave. Mandatory alarms are noted in the **Required** column.

| Element           | Description   |                               |
|-------------------|---|-------------------------------|
| Update Alarm      | Alarm, if the module parameters changed.  | Default value:<br>Deactivated |
| Status Alarm      | Alarm, if the module state changed.   |                               |
| Vendor Alarm      | Vendor specific alarm.  |                               |
| Diagnostic Alarm  | Alarm, if specific events occur in a module, e.g., short circuits, over temperature, etc. |                               |
| Process Alarm     | Alarm, if important events occur in the process.  |                               |
| Pull & Plug Alarm | Alarm, if a module is removed or inserted.  |                               |
|                   |   |                               |

Table 90: Alarms Tab in the Properties Dialog Box for the PROFIBUS DP Slave

#### 6.6.3.5 Tab: Data

This tab specifies details about the supported data lengths and about the user data (extended configuration data).

| Element             | Description  |
|---------------------|--|
| Max. Input Len      | Maximum length of the input data   |
| Max. Output Len     | Maximum length of the output data  |
| Max. Data Len       | Maximum total length of the input and output data  |
| User Data Len       | Length of the user data.   |
| User Data           | Configuration data. HIMA does not recommend editing at this level. Use the <i>User Parameters</i> dialog box instead, see Chapter 6.8. |
| Max. Diag. Data Len | Maximum length of the diagnostic data sent by the slave.   |

Table 91: Data Tab in the Properties Dialog Box for the PROFIBUS DP Slave

### 6.6.3.6 Tab: Model

This tab displays self-explanatory details.

| Element          | Description  |
|------------------|--|
| Model            | Manufacturer identification of the PROFIBUS DP slave |
| Manufacturer     | Manufacturer of the field device                     |
| Ident Number     | Slave identification provided by PNO Germany         |
| Revision         | Issue status of the PROFIBUS DP slave                |
| Hardware release | Hardware issue status of the PROFIBUS DP slave       |
| Software release | Software issue status of the PROFIBUS DP slave       |
| GSD file name    | File name of the GSD file                            |
| Info Text        | Additional details about the PROFIBUS DP slave.      |

Table 92: Model Tab in the Properties Dialog Box for the PROFIBUS DP Slave

### 6.6.3.7 Tab: Features

| Element                     | Description   |
|-----------------------------|---|
| Modular Station             | TRUE: Modular station<br>FALSE: Compact station   |
| First slot number           | The modules (slots) must be numbered without gaps, starting with this value.  |
| Max Modules                 | Maximum number of modules that can be installed in a modular station.   |
| Support for 'Set Slave Add' | The slave supports dynamic address allocation.  |
| Min. Slave Interval [ms]    | The minimum time period that must elapse between two cyclic calls of the slave.                                     |
| Diag. Update                | Number of polling cycles until the slave's diagnosis mirrors the current state.                                     |
| Support for WDBase1ms       | The slave supports 1 ms as time base for the watchdogs  |
| Support for DP V0 Sync      | The slave supports DP V0 Sync   |
| Support for DP V0 Freeze    | The slave supports DP V0 Freeze   |
| DP V1 Data Types            | The slave supports the DP V1 data types.  |
| Extra Alarm SAP             | The slave supports SAP 50 for acknowledging the alarm.  |
| Alarm Seq. Mode Count       | Indicate the number of active alarms that the slave can simultaneously process. Zero means one alarm of each model. |

Table 93: Features Tab in the Properties Dialog Box for the PROFIBUS DP Slave

### 6.6.3.8 Tab: Baud Rates

This tab specifies the baud rates that the slave supports and the corresponding *MaxTsdr*.

*MaxTsdr* is the time within which the slave must acknowledge a request from the master. The range of values depends on the slave and the transfer rate, and ranges between 15 and 800 Tbit.

| Element | Description   |
|---------|---------------|
| 9.6k    | MaxTsdr = 60  |
| 19.2k   | MaxTsdr = 60  |
| 31.25k  | Not supported |
| 45.45k  | MaxTsdr = 60  |
| 93.75k  | MaxTsdr = 60  |
| 187.5k  | MaxTsdr = 60  |
| 500k    | MaxTsdr = 70  |
| 1.5M    | MaxTsdr = 75  |
| 3M      | MaxTsdr = 90  |
| 6M      | MaxTsdr = 100 |
| 12M     | MaxTsdr = 120 |

Table 94: Baud Rates Tab in the Properties Dialog Box for the PROFIBUS DP Slave

### 6.6.3.9 Tab: Acyclic

This tab contains some parameters for the acyclic data transfer.

| Element                   | Description                                   |
|---------------------------|---|
| Support for C1 Read/Write | The slave supports the acyclic data transfer. |
| C1 Read/Write required    | The slave requires the acyclic data transfer. |
| C1 Max Data Len[Byte]     | Maximum length of an acyclic data packet.     |
| C1 Response Timeout [ms]  | Time out for the acyclic data transfer.       |

Table 95: Acyclic Tab in the Properties Dialog Box for the PROFIBUS DP Slave

## 6.7 Importing the GSD File

The GSD file contains data for configuring the PROFIBUS DP slave.

### To read the GSD file for the new PROFIBUS DP slave

1. In the structure tree, open **Configuration, Resource, Protocols, PROFIBUS DP Master, PROFIBUS Slave**.
2. Select **Read GSD File** from the context menu for the PROFIBUS DP master and choose the GSD file for the PROFIBUS slave (e.g., hax100ea.gsd).



The GSD files for HIMax/HIMatrix controllers are available on HIMA website at [www.hima.com](http://www.hima.com). The manufacturer of the field device is responsible for the correctness of the GSD file.

The GSD files for HIMax (hax100ea.gsd) and HIMatrix (hix100ea.gsd) include the following modules:

| PROFIBUS DP Master Input Modules  | Type  | Number |
|-----------------------------------|-------|--------|
| DP Input/ELOP Export              | Byte  | 1      |
| DP Input/ELOP Export              | Bytes | 2      |
| DP Input/ELOP Export              | Bytes | 4      |
| DP Input/ELOP Export              | Bytes | 8      |
| DP Input/ELOP Export              | Bytes | 16     |
| DP Input/ELOP Export              | Word  | 1      |
| DP Input/ELOP Export              | Words | 2      |
| DP Input/ELOP Export              | Words | 4      |
| DP Input/ELOP Export              | Words | 8      |
| DP Input/ELOP Export              | Words | 16     |
| PROFIBUS DP Master Output Modules | Type  | Number |
| DP Output/ELOP Import             | Byte  | 1      |
| DP Output/ELOP Import             | Bytes | 2      |
| DP Output/ELOP Import             | Bytes | 4      |
| DP Output/ELOP Import             | Bytes | 8      |
| DP Output/ELOP Import             | Bytes | 16     |
| DP Output/ELOP Import             | Word  | 1      |
| DP Output/ELOP Import             | Words | 2      |
| DP Output/ELOP Import             | Words | 4      |
| DP Output/ELOP Import             | Words | 8      |
| DP Output/ELOP Import             | Words | 16     |

Table 96: GSD File of the HIMax/HIMatrix PROFIBUS DP Slave

## 6.8 Configuring User Parameters

The group's **start address** and the **number of variables** are defined in the user data field.

The number of bytes that must actually be transferred, must also be configured in the PROFIBUS DP master. This is done by choosing the PROFIBUS DP modules defined in the GDS file of the PROFIBUS DP slave (see also Chapter 6.2.2).

### To open the Edit User Parameters dialog box

1. In the structure tree, open **Configuration, Resource, Protocols, PROFIBUS DP Master**.
2. Right-click **PROFIBUS Slave**, and then click **Properties**.
3. Select the **Data** tab and click the ... button next to the user data.



The structure of the **Edit User Parameters** dialog box depends on the GSD file of the slave.

---

### Structure of the 32-byte user data field:

The 32-byte user data field is structured as follows:

The 32 bytes are allocated in **eight groups**, with **four bytes per group**.

Groups 1...4 define which and how many variables the PROFIBUS DP master receives from the PROFIBUS DP slave.

Groups 5...8 define which and how many variables the PROFIBUS DP master sends to the PROFIBUS DP slave.

The first two bytes of each group specify the start address for the first variables to be read or to be written.

The last two bytes in each group specify the number of variables that should be received or sent.

### Configuring the user data in different groups:

Usually, it is not necessary to allocate variables (user data) into various groups. It is enough to define only the first signal group of the input and output variables, and to read or write the data *en bloc*.

In applications requiring that only selected variables are read and written, up to four variable groups for both the input and the output variables can be defined.

### Example

The PROFIBUS DP master sends and receives the following variables from the PROFIBUS DP slave:

1st group: **4** input variables from start address **0** and up.

2nd group: **6** input variables from start address **50** and up.

4th group: **9** input variables from start address **100** and up.

5th group: **2** output variables from start address **10** and up.

### User data configuration in the PROFIBUS DP master:

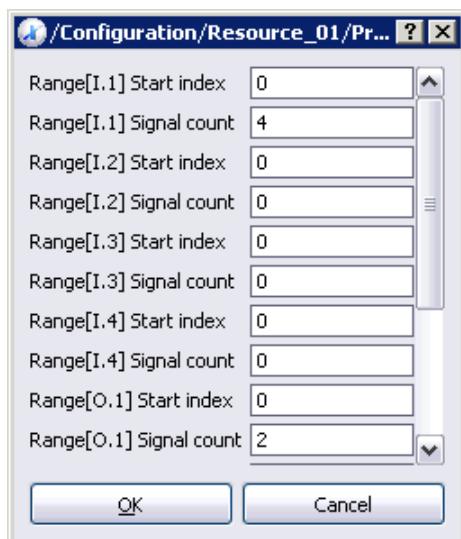
| Master Import/Slave Export | Start Address | Number Variable |
|----------------------------|---------------|-----------------|
| 1st group (byte 0...3)     | 0.0           | 0.4             |
| 2nd group (byte 4...7)     | 0.50          | 0.6             |
| 3rd group (byte 8...11)    | 0.0           | 0.0             |
| 4th group (byte 12...15)   | 0.100         | 0.9             |

Table 97: Example: Group 1...4 of the User Data Field

| Master Export/Slave Import | Start Address | Number of variables |
|----------------------------|---------------|---------------------|
| 5th group (byte 16...19)   | 0.10          | 0.2                 |
| 6th group (byte 20...23)   | 0.0           | 0.0                 |
| 7th group (byte 24...27)   | 0.0           | 0.0                 |
| 8th group (byte 28...31)   | 0.0           | 0.0                 |

Table 98: Example: Group 1...4 of the User Data Field

*Edit User Parameters* dialog box of one HIMatrix or HIMax PROFIBUS DP slave.

Figure 43: *Edit User Parameters* Dialog Box

## 6.9 PROFIBUS Function Blocks

The PROFIBUS function blocks are used to tailor the HIMA PROFIBUS DP master and the corresponding PROFIBUS DP slaves to best meet the project requirements.

The function blocks are configured in the user program such that the master and slave functions (alarms, diagnostic data, and states) can be set and read in the user program.

- 
- i** Function blocks are required for special applications. They are not needed for the normal cyclic data traffic between master and slave!
- For more information on the conceptual configuration of the PROFIBUS DP function blocks, refer to Chapter 14.1.
- 

The following function blocks are available:

| Function block | Function description                                | Suitable beginning with stage of extension DP |
|----------------|---|---|
| MSTAT 6.9.1    | Controlling the master state using the user program | DP V0   |
| RALRM 6.9.2    | Reading the alarm messages of the slaves            | DP V1   |
| RDIAG 6.9.3    | Reading the diagnostic messages of the slaves       | DP V0   |
| RDREC 6.9.4    | Reading the acyclic data records of the slaves      | DP V1   |
| SLACT 6.9.5    | Controlling the slave states using the user program | DP V0   |
| WRREC 6.9.6    | Writing the acyclic data records of the slaves      | DP V1   |

Table 99: Overview of the PROFIBUS DP Function Blocks

- 
- i** HIMA PROFIBUS DP masters operate with the stage of extension DP V1.  
HIMA PROFIBUS DP slaves operate with stage of extension DP V0.  
For this reason, not all function blocks of the HIMA PROFIBUS DP master can be used to control a HIMA PROFIBUS DP slaves.
-

### 6.9.1 MSTAT Function Block

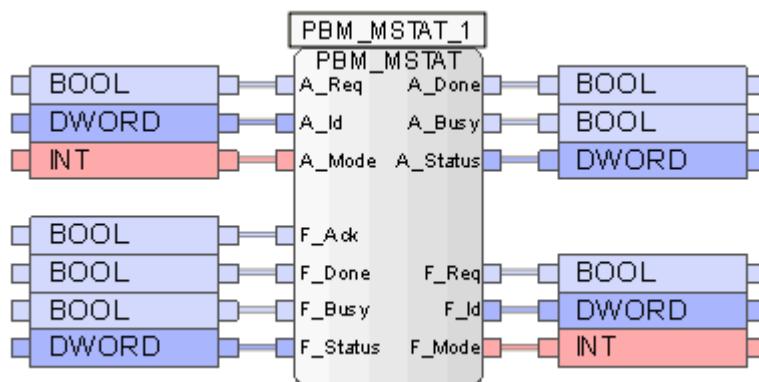


Figure 44: MSTAT Function Block

The user program uses the **MSTAT** function block (DP V0 and higher) to control the PROFIBUS DP master. The master can thus be set to one of the following states using a timer or a mechanical switch connected to a physical input.

- 0: OFFLINE
- 1: STOP
- 2: CLEAR
- 3: OPERATE

**i** To configure the function block, drag it from the function block library onto the user program, see also Chapter 14.1.

#### Inputs and Outputs of the Function Block with Prefix A:

These inputs and outputs can be used to control and evaluate the function block using the user program. The prefix A means Application.

| A_Inputs | Description   | Type  |
|----------|---|-------|
| A_Req    | Rising edge starts the function block   | BOOL  |
| A_ID     | Master ID (not used)  | DWORD |
| A_Mode   | The PROFIBUS DP master can be set to the following states:<br>0: OFFLINE<br>1: STOP<br>2: CLEAR<br>3: OPERATE | INT   |

Table 100: A-Inputs for the MSTAT Function Block

| A_Outputs | Description   | Type  |
|-----------|---|-------|
| DONE      | TRUE: The PROFIBUS DP master has been set to the state defined on the A_Mode input. | BOOL  |
| A_Busy    | TRUE: The PROFIBUS DP master is still being set.                                    | BOOL  |
| A_Status  | Status or error code (see Chapter 6.11)   | DWORD |

Table 101: A-Outputs for the MSTAT Function Block

#### Inputs and Outputs of the Function Block with Prefix F:

These inputs and outputs of the function block establish the connection to the MSTAT function block in structure tree. The prefix F means Field.

- 
- i** Common variables are used to connect the MSTAT function block (in the Function Blocks directory) to the MSTAT function block (in the user program). These must be created beforehand using the Variable Editor.
- 

Connect the *F-Outputs* of the **MSTAT** function block in the user program to the same variables that will be connected to the inputs of the **MSTAT** function block in the structure tree.

| F-Inputs | Type  |
|----------|-------|
| F_ACK    | BOOL  |
| F_DONE   | BOOL  |
| F_BUSY   | BOOL  |
| F_STATUS | DWORD |

Table 102: F-Inputs for the MSTAT Function Block

Connect the *F-Outputs* of the **MSTAT** function block in the user program to the same variables that will be connected to the inputs of the **MSTAT** function block in the structure tree.

| F-Outputs | Type  |
|-----------|-------|
| F_REQ     | BOOL  |
| F_ID      | DWORD |
| F_MODE    | INT   |

Table 103: F-Outputs for the MSTAT Function Block

#### To create the MSTAT function block in the structure tree

1. In the structure tree, open **Configuration, Resource, Protocols, PROFIBUS DP Master, Function Blocks, New**.
2. Select the **MSTAT** function block and click **OK**.
3. Right-click the **MSTAT** function block, and then click **Edit**.
  - The window for assigning variables to the function blocks appears.

Connect the inputs of the **MSTAT** function block in the structure tree to the same variables that have been previously connected to the *F-Outputs* of the **MSTAT** function block in the user program.

| Inputs | Type  |
|--------|-------|
| M_ID   | DWORD |
| MODE   | INT   |
| REQ    | BOOL  |

Table 104: Input System Variables

Connect the outputs of the **MSTAT** function block in the structure tree to the same variables that have been previously connected to the *F-Inputs* of the **MSTAT** function block in the user program.

| Outputs | Type  |
|---------|-------|
| ACK     | BOOL  |
| BUSY    | BOOL  |
| DONE    | BOOL  |
| STATUS  | DWORD |

Table 105: Output System Variables

#### To use the MSTAT function block

1. In the user program, set the *A\_Mode* input to the desired state.  
If *A\_Mode* is not set, an error code is output after step 2 on the *A\_Status* output and the PROFIBUS DP master state is not set.
2. In the user program, set the *A\_Req* input to TRUE.



The function block reacts to a rising edge on *A\_Req*.

- 
- The *A\_Busy* output is TRUE until the MSTAT command has been processed. Afterwards, *A\_Busy* is set to FALSE and *A\_Done* is set to TRUE.



- If the preset mode could not be set successfully, an error code is output to *A\_Status*.  
The mode of the current master can be derived from the Master State variable (see Chapter 6.10).
-

## 6.9.2 RALRM Function Block

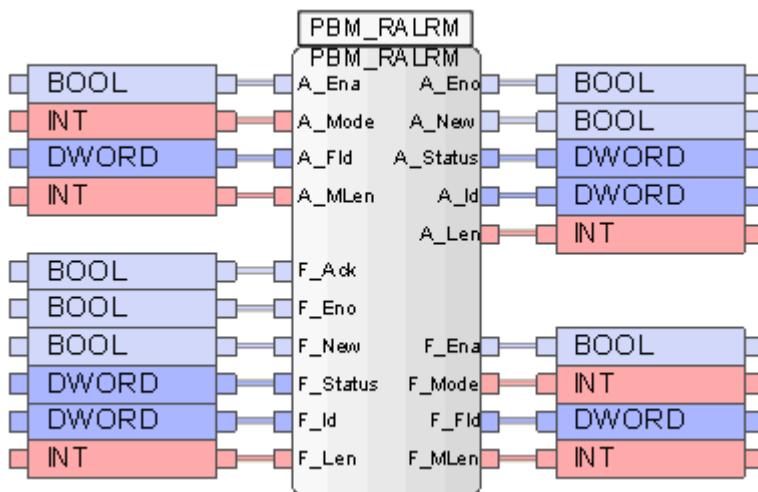


Figure 45: RALRM Function Block

The **RALRM** function block (DP V1 and higher) is used to evaluate the alarms.

Alarms are a special type of diagnostic messages that are handled with a high priority. Alarms report important events to which the application must react (e.g., a WRREC). How the application react, however, depends on the manufacturer. Refer to the manual of the PROFIBUS DP slave for more information.

As long as the **RALRM** function block is active, it waits for alarm messages from the slaves. If an alarm is received, the *A\_NEW* output is set to TRUE for at least one cycle and the alarm data can be read from an alarm telegram. Before the next alarm is received, *A\_NEW* is set to FALSE for at least one cycle. All alarms are acknowledged implicitly. No alarms are lost.

If multiple **RALRM** function blocks are used, the user program must be configured such that only one **RALRM** function block is active at any given time.

- i** To configure the function block, drag it from the function block library onto the user program, see also Chapter 14.1.

### Inputs and Outputs of the Function Block with Prefix A:

These inputs and outputs can be used to control and evaluate the function block using the user program. The prefix A means Application.

| A_Inputs | Description   | Type  |
|----------|---|-------|
| A_Ena    | TRUE enables the function block.                                      | BOOL  |
| A_Mode   | Not used  | INT   |
| A_Fld    | Not used  | DWORD |
| A_MLen   | Maximum expected length of the received alarm data expressed in bytes | INT   |

Table 106: A-Inputs for the RDIAG Function Block

| A_Outputs | Description   | Type  |
|-----------|---|-------|
| A_Eno     | TRUE: The function block is active<br>FALSE: The function block is not active | BOOL  |
| A_New     | TRUE: New alarm was received<br>FALSE: No new alarm                           | BOOL  |
| A_Status  | Status or error code<br>(see Chapter 6.11)                                    | DWORD |
| A_ID      | Identification number of the slave triggering the alarm                       | DWORD |
| A_Len     | Length of the received alarm data in bytes                                    | INT   |

Table 107: A-Outputs for the RDIAG Function Block

**Inputs and Outputs of the Function Block with Prefix F:**

These inputs and outputs of the function block establish the connection to the RALRM function block in structure tree. The prefix F means Field.



Common variables are used to connect the **RALRM** function block (in the Function Blocks directory) to the **RALRM** function block (in the user program). These must be created beforehand using the Variable Editor.

Connect the *F-Inputs* of the **RALRM** function block in the user program to the same variables that will be connected to the outputs of the **RALRM** function block in the structure tree.

| F-Inputs | Type  |
|----------|-------|
| F_ACK    | BOOL  |
| F_ENO    | BOOL  |
| F_NEW    | BOOL  |
| F_STATUS | DWORD |
| F_ID     | DWORD |
| F_LEN    | INT   |

Table 108: F-Inputs for the RALRM Function Block

Connect the *F-Outputs* of the **RALRM** function block in the user program to the same variables that will be connected to the inputs of the **RALRM** function block in the structure tree.

| F-Outputs | Type  |
|-----------|-------|
| F_Ena     | BOOL  |
| F_MODE    | INT   |
| F_FID     | DWORD |
| F_MLEN    | INT   |

Table 109: F-Outputs for the RALRM Function Block

**To create the RALRM function block in the structure tree**

1. In the structure tree, open **Configuration, Resource, Protocols, PROFIBUS DP Master, Function Blocks, New.**
2. Select the **RALRM** function block and click **OK**.
3. Right-click the **RALRM** function block, and then click **Edit**.
  - The window for assigning variables to the function blocks appears.

Connect the inputs of the **RALRM** function block in the structure tree to the same variables that have been previously connected to the *F-Outputs* of the **RALRM** function block in the user program.

| Inputs | Type  |
|--------|-------|
| EN     | BOOL  |
| F_ID   | DWORD |
| MLEN   | INT   |
| MODE   | INT   |

Table 110: Input System Variables

Connect the outputs of the **RALRM** function block in the structure tree to the same variables that have been previously connected to the *F-Inputs* of the **RALRM** function block in the user program.

| Outputs | Type  |
|---------|-------|
| ACK     | BOOL  |
| ENO     | BOOL  |
| ID      | DWORD |
| LEN     | INT   |
| NEW     | BOOL  |
| STATUS  | DWORD |

Table 111: Output System Variables

The Process Variables tab of the **RALRM** function block located in the structure tree contains variables that must be defined and whose structure must match the alarm data. If no variables are defined, alarm data can be requested but not read.

An alarm message contains at least four bytes. The first four bytes of the alarm message contain the standard alarm data.

To decode standard alarms, HIMA provides the auxiliary function block **ALARM** (see Chapter 6.10).



If an alarm telegram contains more bytes than defined in the Data tab, only the preset number of bytes is accepted. The rest is cut off.

| Alarm Data    | Description   |
|---------------|---|
| Byte 0        | Length of the alarm message expressed in bytes (4...126)  |
| Byte 1        | Identification for the alarm type<br>1: Diagnostic alarm<br>2: Process alarm<br>3: Pull alarm<br>4: Plug alarm<br>5: Status alarm<br>6: Update alarm<br>31: Failure of a master's or a slave's extension<br>32...126: Manufacturer specific<br>Consult the device manual provided by the manufacturer for more information on the specific meaning. |
| Byte 2        | Slot number of the component triggering the alarm   |
| Byte 3        | 0: No further information<br>1: Inbound alarm, slot malfunction<br>2: Outbound alarm, slot no longer malfunctioning<br>3: Outbound alarm, continued slot malfunction  |
| Byte 4 to 126 | Consult the device manual provided by the manufacturer for more information on the specific meaning.  |

Table 112: Alarm Data

- i** The structure of the standard alarms (bytes 0...3) is standardized and identical for all manufacturers. See the manual of the PROFIBUS DP slave for more information on bytes 4...126, since their use is manufacturer specific.  
 Devices built in accordance with the DP-V0 standard do not support alarm telegrams.

#### To use the RALRM function block

1. On the *A\_Mlen* input of the user program, define the maximum amount of alarm data in bytes that must be expected. *A\_Mlen* cannot be changed during operation.
2. In the user program, set the *A\_Ena* input to TRUE.

- i** In contrast to other function blocks, the **RALRM** function block is only active as long as the *A\_Ena* input is set to TRUE.

If the function block was started successfully, the *A\_Eno* output is set to TRUE. If the function block could not be started, an error code is output to *A\_Status*.

If a new alarm is received, the *A\_New* output is set to TRUE for at least one cycle. During this time period, the alarm data of the slave triggering the alarm are contained in the outputs and can be evaluated.

Afterwards, the *A\_New* output returns to FALSE for at least one cycle.

The *A\_Id* and *A\_Len* outputs are reset to zero before the next alarm message can be received and evaluated.

### 6.9.3 RDIAG Function Block

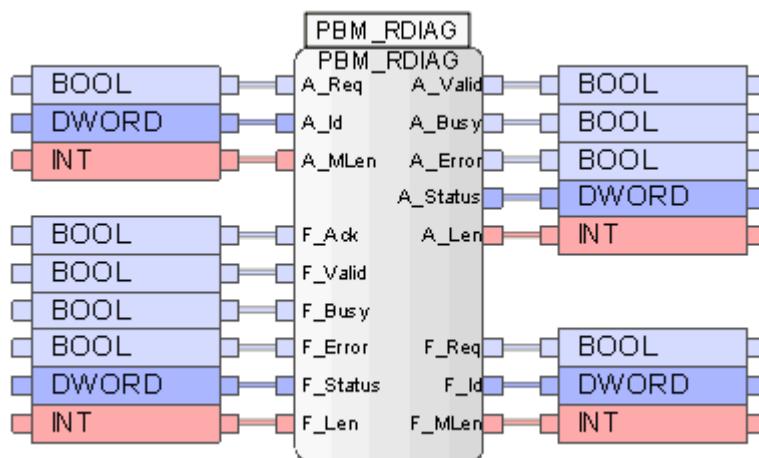


Figure 46: RDIAG Function Block

The **RDIAG** function block (DP V0 and higher) is used for reading the current diagnostic message of a slave (6...240 bytes).

As many **RDIAG** function blocks as desired may be simultaneously active within the HIMA PROFIBUS DP master.

- i** To configure the function block, drag it from the function block library onto the user program (see also Chapter 14.1).

#### Inputs and Outputs of the Function Block with Prefix A:

These inputs and outputs can be used to control and evaluate the function block using the user program. The prefix A means Application.

| A_Inputs | Description   | Type  |
|----------|---|-------|
| A_Req    | The rising edge starts the reading request of a diagnostic message      | BOOL  |
| A_ID     | Slave's identification number (see Chapter 6.10)                        | DWORD |
| A_MLen   | Maximum length (in bytes) of the diagnostic message expected to be read | INT   |

Table 113: A-Inputs for the RDIAG Function Block

| A_Outputs | Description   | Type  |
|-----------|---|-------|
| A_Valid   | A new diagnostic message has been received and is valid | BOOL  |
| A_Busy    | TRUE: Data is still being read                          | BOOL  |
| ERROR     | TRUE: An error occurred during the reading process      | BOOL  |
| A_Status  | Status or error code (see Chapter 6.11)                 | DWORD |
| A_Len     | Length of the read diagnostic data in bytes             | INT   |

Table 114: A-Outputs for the RDIAG Function Block

#### Inputs and Outputs of the Function Block with Prefix F:

These inputs and outputs of the function block establish the connection to the **RDIAG** function block in structure tree. The prefix F means Field.

- 
- i** Common variables are used to connect the **RDIAG** function block (in the Function Blocks directory) to the **RDIAG** function block (in the user program). These must be created beforehand using the Variable Editor.
- 

Connect the *F-Inputs* of the **RDIAG** function block in the user program to the same variables that will be connected to the outputs of the **RDIAG** function block in the structure tree.

| F-Inputs | Type  |
|----------|-------|
| F_ACK    | BOOL  |
| F_VALID  | BOOL  |
| F_BUSY   | BOOL  |
| F_ERROR  | BOOL  |
| F_Status | DWORD |
| F_LEN    | INT   |

Table 115: F-Inputs for the RDIAG Function Block

Connect the *F-Outputs* of the **RDIAG** function block in the user program to the same variables that will be connected to the inputs of the **RDIAG** function block in the structure tree.

| F-Outputs | Type  |
|-----------|-------|
| F_Req     | BOOL  |
| F_Id      | DWORD |
| F_Mlen    | INT   |

Table 116: F-Outputs for the RDIAG Function Block

#### To create the RDIAG function block in the structure tree

1. In the structure tree, open **Configuration, Resource, Protocols, PROFIBUS DP Master, Function Blocks, New**.
2. Select the **RDIAG** function block and click **OK**.
3. Right-click the **RDIAG** function block, and then click **Edit**.
  - The window for assigning variables to the function blocks appears.

Connect the inputs of the **RDIAG** function block in the structure tree to the same variables that have been previously connected to the *F-Outputs* of the **RDIAG** function block in the user program.

| Inputs | Type  |
|--------|-------|
| ID     | DWORD |
| MLEN   | INT   |
| REQ    | BOOL  |

Table 117: Input System Variables

Connect the outputs of the **RDIAG** function block in the structure tree to the same variables that have been previously connected to the *F-Inputs* of the **RDIAG** function block in the user program.

| Outputs | Type  |
|---------|-------|
| ACK     | BOOL  |
| BUSY    | BOOL  |
| ERROR   | BOOL  |
| LEN     | INT   |
| Status  | DWORD |
| VALID   | BOOL  |

Table 118: Output System Variables

### Diagnostic Data

The **Data** tab contains variables that must be defined and whose structure must match the alarm data. A diagnostic message contains at least six bytes and a maximum of 240 bytes. The first four bytes of the diagnostic message contain the standard diagnosis.

To decode the standard alarms, HIMA provides the auxiliary function block **STDDIAG** (see Chapter 6.10).

- 
- i** If a diagnostic telegram contains more bytes than defined in the Data tab, only the preset number of bytes is accepted. The rest is cut off.
- 

| Diagnostic Data | Description   |
|-----------------|---|
| Byte 0          | Byte 0...3 contains the standard diagnosis. Use the <b>STDDIAG</b> auxiliary function block to decode the standard diagnosis as a variable of the type DWORD. |
| Byte 1          |   |
| Byte 2          |   |
| Byte 3          | Bus address of the master to which a slave is assigned.   |
| Byte 4          | High byte (manufacturer ID)   |
| Byte 5          | Low byte (manufacturer ID)  |
| Byte 6...240    | Consult the device manual provided by the manufacturer for more information on the specific meaning.  |

Table 119: Diagnostic Data

- 
- i** The HIMA slaves send a diagnostic telegram of six bytes in length. The meaning of these bytes is standardized.  
The first six bytes of slaves from other manufacturers are only functionally identical.  
For more information on the diagnostic telegram, refer to the description of the slave provided by the manufacturer.
- 

### To use the RDIAG function block

1. In the user program, set the slave address on the *A\_ID* input.
  2. On the user program's *A\_Mlen*, define the maximum amount of alarm data in bytes that must be expected.
  3. In the user program, set the *A\_Req* input to TRUE.
- 

- i** The function block reacts to a rising edge on *A\_Req*.
- 

The *A\_Busy* output is set to TRUE until the diagnostic request has been processed. Afterwards, *A\_Busy* is set to FALSE and *A\_Valid* or *A\_Error* to TRUE.

If the diagnostic telegram is valid, the *A\_Valid* output is set to TRUE. The diagnostic data can be evaluated using the variables defined in the Data tab. The *A\_Len* output contains the amount of diagnostic data in bytes that was actually read.

If the diagnostic telegram could not be read successfully, the *A\_Error* output is set to TRUE and an error code is output to *A\_Status*.

### 6.9.4 RDREC Function Block

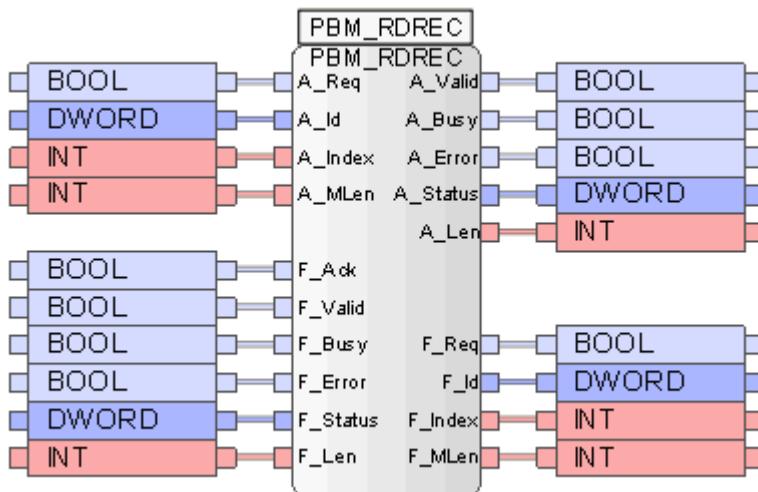


Figure 47: RDREC Function Block

The **RDREC** function block is used for acyclically reading a data record from a slave addressed on the *A\_Index* input. Consult the slave's manual to find out which data can be read.

This functionality is optional and is only defined with DP V1 and higher!

Up to 32 **RDREC** and/or **WRREC** function blocks can simultaneously be active in the HIMA PROFIBUS DP master.



To configure the function block, drag it from the function block library onto the user program (see also Chapter 14.1).

#### Inputs and Outputs of the Function Block with Prefix A:

These inputs and outputs can be used to control and evaluate the function block using the user program. The prefix A means Application.

| A_Inputs | Description   | Type  |
|----------|---|-------|
| A_Req    | The rising edge starts the reading request.   | BOOL  |
| A_Id     | Slave identification number, (see Chapter 6.10)   | DWORD |
| A_Index  | Number of the data record to be read.<br>Consult the device manual provided by the manufacturer for more information on the specific meaning. | INT   |
| A_MLen   | Maximum length of the data to be read in bytes.   | INT   |

Table 120: A-Inputs for the RDREC Function Block

| A_Outputs | Description  | Type  |
|-----------|--|-------|
| A_Valid   | A new data record was received and is valid.         | BOOL  |
| A_Busy    | TRUE: Data is still being read.                      | BOOL  |
| ERROR     | TRUE: An error occurred<br>FALSE: No error           | BOOL  |
| A_Status  | Status or error code, see Chapter 6.11.              | DWORD |
| A_Len     | Length of the read data record information in bytes. | INT   |

Table 121: A-Outputs for the RDREC Function Block

### Inputs and Outputs of the Function Block with Prefix F

These inputs and outputs of the function block establish the connection to the **RDREC** function block in structure tree. The prefix F means Field.

- 
- i** Common variables are used to connect the **RDREC** function block (in the Function Blocks directory) to the RDREC function block (in the user program). These must be created beforehand using the Variable Editor.
- 

Connect the *F-Inputs* of the **RDREC** function block in the user program to the same variables that will be connected to the outputs of the **RDREC** function block in the structure tree.

| F-Inputs | Type  |
|----------|-------|
| F_Ack    | BOOL  |
| F_Valid  | BOOL  |
| F_Busy   | BOOL  |
| F_Error  | BOOL  |
| F_Status | DWORD |
| F_Len    | INT   |

Table 122: F-Inputs for the RDREC Function Block

Connect the *F-Outputs* of the **RDREC** function block in the user program to the same variables that will be connected to the inputs of the **RDREC** function block in the structure tree.

| F-Outputs | Type  |
|-----------|-------|
| F_Req     | BOOL  |
| F_Id      | DWORD |
| F_Index   | INT   |
| F_Mlen    | INT   |

Table 123: F-Outputs for the RDREC Function Block

### To create the RDREC function block in the structure tree

1. In the structure tree, open **Configuration, Resource, Protocols, PROFIBUS DP Master, Function Blocks, New**.
2. Select the **RDREC** function block and click **OK..**
3. Right-click the **RDREC** function block, and then click **Edit**.
  - The window for assigning variables to the function blocks appears.

Connect the inputs of the **RDREC** function block in the structure tree to the same variables that have been previously connected to the *F-Outputs* of the **RDREC** function block in the user program.

| Inputs | Type  |
|--------|-------|
| ID     | DWORD |
| INDEX  | INT   |
| MLEN   | INT   |
| REQ    | BOOL  |

Table 124: Input System Variables

Connect the outputs of the **RDREC** function block in the structure tree to the same variables that have been previously connected to the *F-Inputs* of the **RDREC** function block in the user program.

| Outputs | Type  |
|---------|-------|
| ACK     | BOOL  |
| BUSY    | BOOL  |
| ERROR   | BOOL  |
| LEN     | INT   |
| STATUS  | DWORD |
| VALID   | BOOL  |

Table 125: Output System Variables

| Data                    | Description  |
|-------------------------|--|
| No predefined variables | A user-specific data structure can be defined in the <i>Process Variables</i> tab; however, the structure must match the data record structure. For more information on the record structure, refer to the operating instructions provided by the manufacturer of the slave. |

Table 126: Data

#### To use the RDREC function block

1. In the user program, set the slave address on the *A\_ID* input.
2. In the user program, set the slave-specific index for the data record on the *A\_Index* input (see the manual provided by the manufacturer).
3. In the user program, set the length of the data record to be read on the *A\_Len* input.
4. In the user program, set the *A\_Req* input to TRUE.



The function block reacts to a rising edge on *A\_Req*.

The *A\_Busy* output is set to TRUE until the data record request has been processed. Afterwards, *A\_Busy* is set to FALSE and *A\_Valid* or *A\_Error* to TRUE.

If the data record is valid, the *A\_Valid* output is set to TRUE. The data set can be evaluated using the variables defined in the Data tab. The *A\_Len* output contains the actual length of the data record that has been read.

If the data record could not be read successfully, the *A\_Error* output is set to TRUE and an error code is output to *A\_Status*.

### 6.9.5 SLACT Function Block

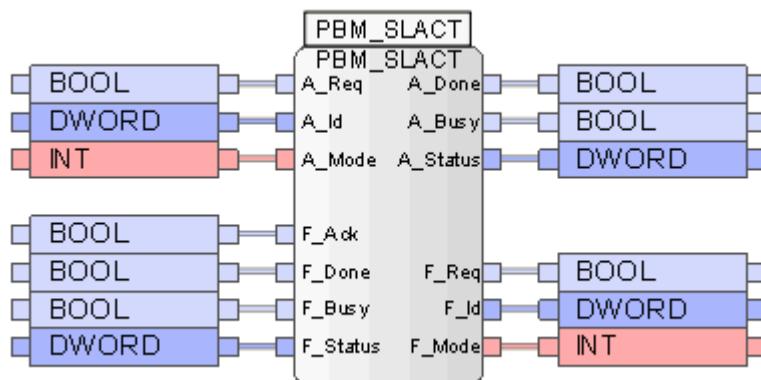


Figure 48: SLACT Function Block

The **SLACT** function block (DP V0 and higher) is used for activating and deactivating a slave from within the user program of the PROFIBUS DP master. The slave can thus be set to one of the following states using a timer or a mechanical switch connected to a physical input of the PROFIBUS DP master.

$\neq 0$ : Active

$= 0$ : Inactive

If various **SLACT** function blocks are used, the user program must be configured such that only one **SLACT** function block is active at a time.



To configure the function block, drag it from the function block library onto the user program (see also Chapter 14.1).

#### Inputs and Outputs of the Function Block with Prefix A:

These inputs and outputs can be used to control and evaluate the function block using the user program. The prefix A means Application.

| A_Inputs | Description   | Type  |
|----------|---|-------|
| A_Req    | Rising edge starts the function block   | BOOL  |
| A_Id     | Slave's identification number<br>(see Chapter 6.10)   | DWORD |
| A_Mode   | Target state for the slave PROFIBUS DP slave<br>$\neq 0$ : Active (Connected)<br>$= 0$ : Not active (Deactivated) | INT   |

Table 127: A-Inputs for the SLACT Function Block

| A_Outputs | Description  | Type  |
|-----------|--|-------|
| DONE      | TRUE: The PROFIBUS DP slave has been set to the state defined on the A_Mode input. | BOOL  |
| A_Busy    | TRUE: The PROFIBUS DP slave is still being set.                                    | BOOL  |
| A_Status  | Status or error code<br>(see Chapter 6.11)   | DWORD |

Table 128: A-Outputs for the SLACT Function Block

**Inputs and Outputs of the Function Block with Prefix F:**

These inputs and outputs of the function block establish the connection to the **SLACT** function block in structure tree. The prefix F means Field.

- i** Common variables are used to connect the SLACT function block (in the Function Blocks directory) to the SLACT function block (in the user program). These must be created beforehand using the Variable Editor.

Connect the *F-Inputs* of the **SLACT** function block in the user program to the same variables that will be connected to the outputs of the **SLACT** function block in the structure tree.

| F-Inputs | Type  |
|----------|-------|
| F_Ack    | BOOL  |
| F_Done   | BOOL  |
| F_Busy   | BOOL  |
| F_Status | DWORD |

Table 129: F-Inputs for the SLACT Function Block

Connect the *F-Outputs* of the **SLACT** function block in the user program to the same variables that will be connected to the inputs of the **SLACT** function block in the structure tree.

| F-Outputs | Type  |
|-----------|-------|
| F_Req     | BOOL  |
| F_Id      | DWORD |
| F_Mode    | INT   |

Table 130: F-Outputs for the SLACT Function Block

**To create the SLACT function block in the structure tree**

1. In the structure tree, open **Configuration, Resource, Protocols, PROFIBUS DP Master, Function Blocks, New.**
2. Select the **SLACT** function block and click **OK**.
3. Right-click the **SLACT** function bloc, and then click **Edit**.
  - The window for assigning variables to the function blocks appears.

Connect the inputs of the **SLACT** function block in the structure tree to the same variables that have been previously connected to the *F-Outputs* of the **SLACT** function block in the user program.

| Inputs | Type  |
|--------|-------|
| ID     | DWORD |
| MODE   | INT   |
| REQ    | BOOL  |

Table 131: Input System Variables

Connect the outputs of the **SLACT** function block in the structure tree to the same variables that have been previously connected to the *F-Inputs* of the **SLACT** function block in the user program.

| Outputs | Type  |
|---------|-------|
| ACK     | BOOL  |
| BUSY    | BOOL  |
| DONE    | BOOL  |
| STATUS  | DWORD |

Table 132: Output System Variables

#### To use the SLACT function block

1. In the user program, set the *A\_Mode* input to the desired state.
2. In the user program, set the slave address identifier on the *A\_ID* input.
3. In the user program, set the *A\_Req* input to TRUE.



The function block reacts to a rising edge on *A\_Req*.

---

*A\_Busy* output is set to TRUE until the *SLACT* command has been processed. Afterwards, *A\_Busy* is set to FALSE and *A\_Done* is set to TRUE.

If the slave mode could be set successfully, it is output to *A\_Status*.

If the slave mode could not be set successfully, an error code is output to *A\_Status*.

### 6.9.6 WRREC Function Block

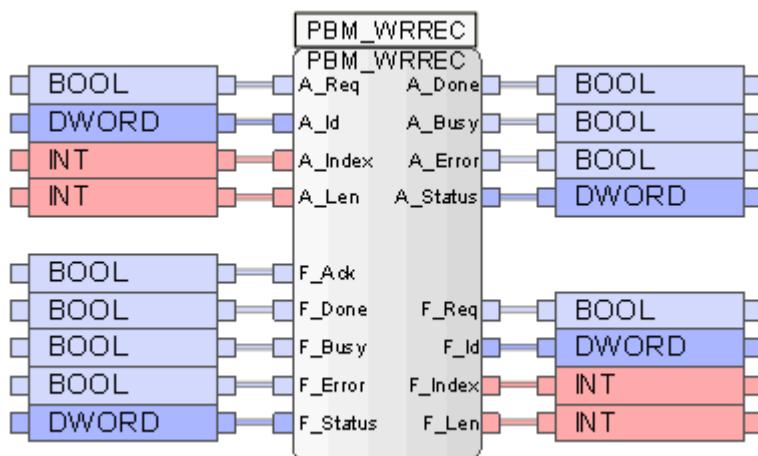


Figure 49: WRREC Function Block

The **WRREC** function block (DP V1 and higher) is used for acyclically writing a data record to a slave addressed with *A\_Index*. Consult the slave's manual to find out which data can be written.

Up to 32 **RDREC** and/or **WRREC** function blocks can simultaneously be active in the HIMA PROFIBUS DP master.



To configure the function block, drag it from the function block library onto the user program (see also Chapter 14.1).

#### Inputs and Outputs of the Function Block with Prefix A:

These inputs and outputs can be used to control and evaluate the function block using the user program. The prefix A means Application.

| A_Inputs | Description  | Type  |
|----------|--|-------|
| A_Req    | The rising edge starts the request for writing a data record.  | BOOL  |
| A_ID     | Identification number of the slave<br>(see Chapter 6.10)   | DWORD |
| A_Index  | Number of the data record to be written.<br>Consult the device manual provided by the manufacturer for more information on the specific meaning. | INT   |
| A_Len    | Length of the data record to be written in bytes   | INT   |

Table 133: A-Inputs for the WRREC Function Block

| A_Outputs | Description  | Type  |
|-----------|--|-------|
| DONE      | TRUE: The function block completed the writing process.            | BOOL  |
| A_Busy    | TRUE: The function block has not yet completed the writing process | BOOL  |
| ERROR     | TRUE: An error occurred  | BOOL  |
| A_STATUS  | Status or error code<br>(see Chapter 6.11)                         | DWORD |

Table 134: A-Outputs for the WRREC Function Block

#### Inputs and Outputs of the Function Block with Prefix F:

These inputs and outputs of the function block establish the connection to the **WRREC** function block in structure tree. The prefix F means Field.

- 
- i** Common variables are used to connect the WRREC function block (in the Function Blocks directory) to the WRREC function block (in the user program). These must be created beforehand using the Variable Editor.
- 

Connect the *F-Inputs* of the **WRREC** function block in the user program to the same variables that will be connected to the inputs of the **WRREC** function block in the structure tree.

| F-Inputs | Type  |
|----------|-------|
| F_Ack    | BOOL  |
| F_Done   | BOOL  |
| F_Busy   | BOOL  |
| F_Error  | BOOL  |
| F_Status | DWORD |

Table 135: F-Inputs for the WRREC Function Block

Connect the *F-Outputs* of the **WRREC** function block in the user program to the same variables that will be connected to the inputs of the **WRREC** function block in the structure tree.

| F-Outputs | Type  |
|-----------|-------|
| F_Req     | BOOL  |
| F_Id      | DWORD |
| F_Index   | INT   |
| F_Len     | INT   |

Table 136: F-Outputs for the WRREC Function Block

#### To create the WRREC function block in the structure tree

1. In the structure tree, open **Configuration, Resource, Protocols, PROFIBUS DP Master, Function Blocks, New**.
2. Select the **WRREC** function block and click **OK**.
3. Right-click the **WRREC** function block, and then click **Edit**.
  - The window for assigning variables to the function blocks appears.

Connect the inputs of the **WRREC** function block in the structure tree to the same variables that have been previously connected to the *F-Outputs* of the **WRREC** function block in the user program.

| Inputs | Type  |
|--------|-------|
| ID     | DWORD |
| INDEX  | INT   |
| LEN    | INT   |
| REQ    | BOOL  |

Table 137: Input System Variables

Connect the outputs of the **WRREC** function block in the structure tree to the same variables that have been previously connected to the *F-Inputs* of the **WRREC** function block in the user program.

| Outputs | Type  |
|---------|-------|
| ACK     | BOOL  |
| BUSY    | BOOL  |
| DONE    | BOOL  |
| ERROR   | BOOL  |
| STATUS  | DWORD |

Table 138: Output System Variables

| Data                    | Description   |
|-------------------------|---|
| No predefined variables | A user-specific data structure can be defined in the <i>Process Variables</i> tab; however, the structure must match the data record structure.<br>For more information on the record structure, refer to the operating instructions provided by the manufacturer of the slave. |

Table 139: Data

**To operate the WRREC function block, the following steps are essential:**

1. In the user program, set the slave address on the *A\_ID* input.
2. In the user program, set the slave-specific index for the data record on the *A\_Index* input (see the manual provided by the manufacturer).
3. In the user program, set the length of the data record to be written on the *A\_Len* input.
4. In the user program, set the data record as defined in the Data tab.
5. In the user program, set the *A\_Req* input to TRUE.



The function block reacts to a rising edge on *A\_Req*.

---

The *A\_Busy* output is TRUE until to the data record is written. Afterwards, *A\_Busy* is set to FALSE and *A\_Done* is set to TRUE.

If the data record could not be written successfully, the *A\_Error* output is set to TRUE and an error code is output to *A\_Status*.

## 6.10 PROFIBUS Auxiliary Function Blocks

The auxiliary function blocks are used to configure and evaluate the inputs and outputs of the function blocks.

The following auxiliary function blocks are available:

| Auxiliary function blocks    | Function description                                    |
|------------------------------|---|
| ACTIVE (see Chapter 6.10.1)  | Determine if the slave is active or inactive            |
| ALARM (see Chapter 6.10.2)   | Decode the alarm data                                   |
| DEID (see Chapter 6.10.3)    | Decode the identification number                        |
| ID (see Chapter 6.10.4)      | Generate a four byte identifier                         |
| NSLOT (see Chapter 6.10.5)   | Create a continuous identification number for the slots |
| SLOT (see Chapter 6.10.6)    | Create a SLOT identification number using a slot number |
| STDDIAG (see Chapter 6.10.7) | Decode the standard diagnosis of a slave                |
| LATCH                        | Only used within other function blocks                  |
| PIG                          | Only used within other function blocks                  |
| PIGII                        | Only used within other function blocks                  |

Table 140: Overview of the Auxiliary Function Blocks

### 6.10.1 Auxiliary Function Block: ACTIVE

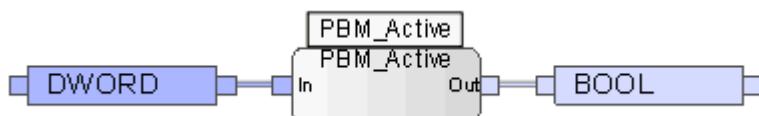


Figure 50: The ACTIVE Auxiliary Function Block

The ACTIVE auxiliary function block uses the standard diagnosis of a PROFIBUS DP slave to determine if the slave is active or inactive.



To configure the function block, drag it from the function block library onto the user program (see also Chapter 14.1).

| Inputs | Description              | Type  |
|--------|--------------------------|-------|
| IN     | Slave standard diagnosis | DWORD |

Table 141: Inputs for the ACTIVE Auxiliary Function Block

| Outputs | Description   | Type |
|---------|---|------|
| OUT     | TRUE: The slave is active<br>FALSE: The slave is inactive | BOOL |

Table 142: Outputs for the ACTIVE Auxiliary Function Block

### 6.10.2 Auxiliary Function Block: ALARM

(Decode the Alarm Data)

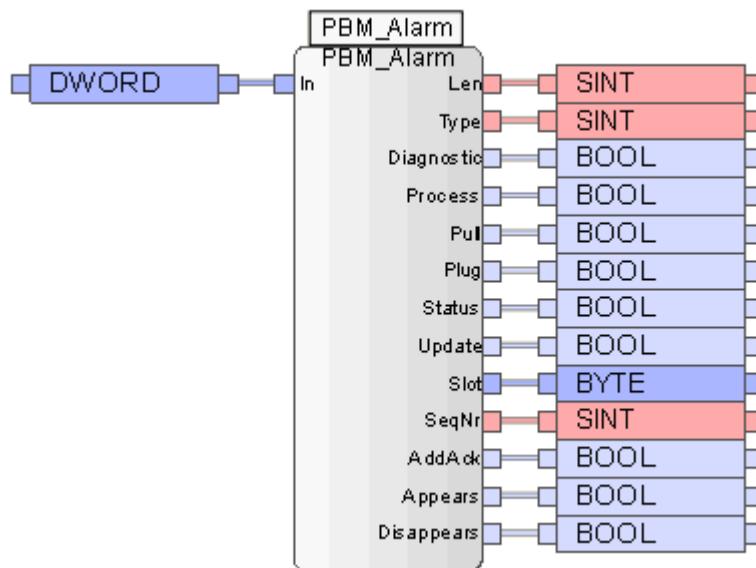


Figure 51: The ALARM Auxiliary Function Block

The **ALARM** auxiliary function block decodes the standard alarm data of a PROFIBUS DP slave.

- i** To configure the function block, drag it from the function block library onto the user program (see also Chapter 14.1).

| Inputs | Description    | Type  |
|--------|----------------|-------|
| IN     | Standard alarm | DWORD |

Table 143: Inputs for the ALARM Auxiliary Function Block

| Output     | Description   | Type |
|------------|---|------|
| Len        | Total length of the alarm message.  | SINT |
| Type       | 1: Diagnostic alarm<br>2: Process alarm<br>3: Pull alarm<br>4: Plug alarm<br>5: Status alarm<br>6: Update alarm<br><br>The other numbers are either reserved or manufacturer specific. Consult the device manual provided by the manufacturer for more information on the specific meaning. | SINT |
| Diagnostic | True = Diagnostics alarm  | BOOL |
| Process    | True = Process alarm  | BOOL |
| Pull       | True = The module was pulled  | BOOL |
| Plug       | True = The module was plugged   | BOOL |
| Status     | True = Status alarm   | BOOL |
| Update     | True = Update alarm   | BOOL |
| Slot       | Alarm Triggering Module   | BYTE |
| SeqNr      | Alarm Sequence Number   | SINT |

| Output                | Description   |       |  | Type |  |
|-----------------------|---|-------|--|------|--|
| AddAck                | TRUE means that the slave that triggered this alarm requires an additional acknowledgement from the application. For more information on this, consult the slave manual provided by the manufacturer. |       |  | BOOL |  |
| Appears<br>Disappears | Output  | Value | Description  | BOOL |  |
|                       | Appears   | TRUE  | If both are FALSE, no error has occurred yet.                                    |      |  |
|                       | Disappears  | FALSE | An error occurred and is still present.  |      |  |
|                       | Appears   | TRUE  | An error occurred and is disappearing.   |      |  |
|                       | Disappears  | FALSE | If both are TRUE, the fault disappears but the slave is still in a faulty state. |      |  |
|                       | Appears   | TRUE  | An error occurred and is disappearing.   |      |  |
|                       | Disappears  | FALSE |  |      |  |

Table 144: Outputs for the ALARM Auxiliary Function Block

### 6.10.3 Auxiliary Function Block: DEID

(Decode the identification number)

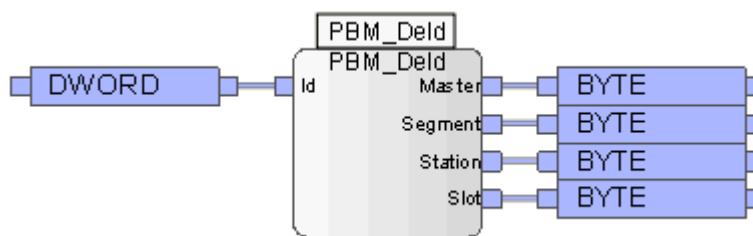


Figure 52: The DEID Auxiliary Function Block

The **DEID** auxiliary function block decodes the identification number and disjoins it into its four component parts.

- i To configure the function block, drag it from the function block library onto the user program (see also Chapter 14.1).

| Inputs | Description                        | Type  |
|--------|------------------------------------|-------|
| ID     | Identification number of the slave | DWORD |

Table 145: Inputs for the DEID Auxiliary Function Block

| Outputs | Description           | Type |
|---------|-----------------------|------|
| Master  | Master bus address    | BYTE |
| Segment | Segment               | BYTE |
| Stop    | Slave bus address     | BYTE |
| Slot    | Slot or module number | BYTE |

Table 146: Outputs for the DEID Auxiliary Function Block

#### 6.10.4 Auxiliary Function Block: ID

(Generate the identification number)

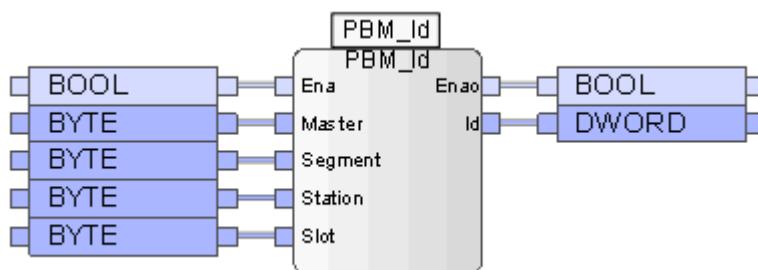


Figure 53: The ID Auxiliary Function Block

The **ID** auxiliary function block uses four bytes to generate an identifier (identification number) used by other auxiliary function blocks.

- 
- i** To configure the function block, drag it from the function block library onto the user program (see also Chapter 14.1).
- 

| Inputs  | Description           | Type |
|---------|-----------------------|------|
| Ena     | Not used              | BOOL |
| Master  | Bus address           | BYTE |
| Segment | Segment               | BYTE |
| Stop    | Slave bus address     | BYTE |
| Slot    | Slot or module number | BYTE |

Table 147: Inputs for the ID Auxiliary Function Block

| Outputs | Description                        | Type  |
|---------|------------------------------------|-------|
| Enao    | Not used                           | BOOL  |
| ID      | Identification number of the slave | DWORD |

Table 148: Outputs for the ID Auxiliary Function Block

### 6.10.5 Auxiliary Function Block: NSLOT

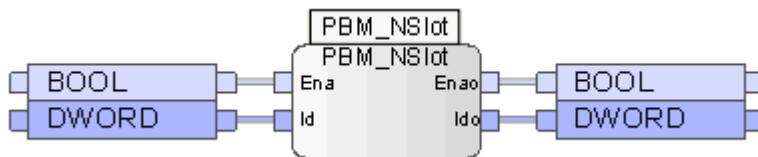


Figure 54: The NSLOT Auxiliary Function Block

The **NSLOT** auxiliary function block uses an identifier to generate a new identifier that addresses the next slot within the same slave. Ena must be set to TRUE to allow the auxiliary function block to run.

Enao is set to TRUE if the result on the Ido output is valid.

- To configure the function block, drag it from the function block library onto the user program (see also Chapter 14.1).

| Inputs | Description  | Type  |
|--------|--|-------|
| Ena    | The auxiliary function block runs as long as Ena is set to TRUE. | BOOL  |
| ID     | Identification number of the slave                               | DWORD |

Table 149: Inputs for the NSLOT Auxiliary Function Block

| Outputs | Description  | Type  |
|---------|--|-------|
| Enao    | TRUE = The result is valid<br>FALSE = No further slot number | BOOL  |
| Ido     | Identification number of the slave                           | DWORD |

Table 150: Outputs for the NSLOT Auxiliary Function Block

### 6.10.6 Auxiliary Function Block: SLOT

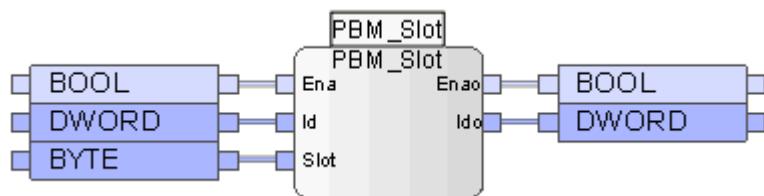


Figure 55: The SLOT Auxiliary Function Block

The **SLOT** auxiliary function block uses an identifier and a slot number to generate a new identifier that addresses the same slave as the first identifier but with the new slot number.

- 
- i To configure the function block, drag it from the function block library onto the user program (see also Chapter 14.1).
-

| Inputs | Description   | Type  |
|--------|---|-------|
| Ena    | Not used  | BOOL  |
| ID     | Logical address of the slave component (slave ID and slot number) | DWORD |
| Slot   | New slot or module number   | BYTE  |

Table 151: Inputs for the SLOT Auxiliary Function Block

| Outputs | Description                        | Type  |
|---------|------------------------------------|-------|
| Enao    | Not used                           | BOOL  |
| Ido     | Identification number of the slave | DWORD |

Table 152: Outputs for the SLOT Auxiliary Function Block

### 6.10.7 Auxiliary Function Block: STDDIAG

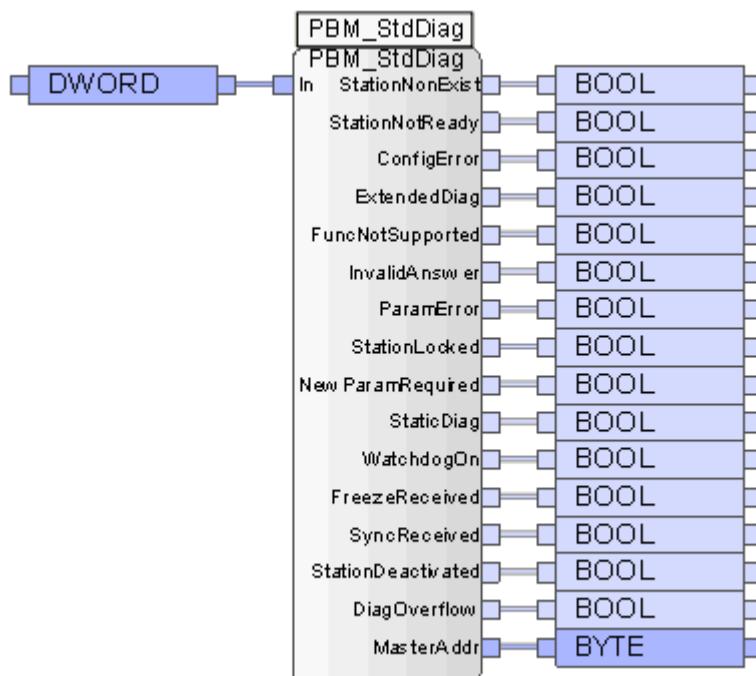


Figure 56: The STDDIAG Auxiliary Function Block

The **STDDIAG** auxiliary function block decodes the standard diagnosis of a PROFIBUS DP slave.

The outputs of type BOOL in the **STDDIAG** auxiliary function block are set to TRUE if the corresponding bit has been set in the standard diagnosis.

- 
- i** To configure the function block, drag it from the function block library onto the user program (see also Chapter 14.1).
-

| Inputs | Description              | Type  |
|--------|--------------------------|-------|
| IN     | Slave standard diagnosis | DWORD |

Table 153: Inputs for the STDDIAG Auxiliary Function Block

| Outputs            | Description                     | Type |
|--------------------|---------------------------------|------|
| StationNonExist    | The slave does not exist        | BOOL |
| StationNotReady    | Slave not ready                 | BOOL |
| ConfigError        | Configuration error             | BOOL |
| ExtendedDiag       | Extended diagnosis follows      | BOOL |
| FuncNotSupported   | The function is not supported   | BOOL |
| InvalidAnswer      | Invalid reply from slave        | BOOL |
| ParamError         | Parameter error                 | BOOL |
| StationLocked      | Slave locked by another master  | BOOL |
| NewParamRequired   | New configuration data required | BOOL |
| StaticDiag         | Static diagnosis                | BOOL |
| WatchdogOn         | Watchdog active                 | BOOL |
| FreezeReceived     | Freeze command received         | BOOL |
| SyncReceived       | Sync command received           | BOOL |
| StationDeactivated | The slave was deactivated       | BOOL |
| DiagOverflow       | Diagnostic overflow             | BOOL |
| MasterAddr         | Master bus address              | BYTE |

Table 154: Outputs for the STDDIAG Auxiliary Function Block

**To read the standard diagnosis of the PROFIBUS DP slave:**

1. In the structure tree, open **Configuration, Resource, Protocols, PROFIBUS DP Master**.
2. Right-click **PROFIBUS Slave**, and then click **Edit**.
3. Drag the global variable of type DWORD onto the *Standard Diagnosis* field.
4. Connect this global variable with the input of the **STDDIAG** function block.

## 6.11 Error Codes of the Function Blocks

If a function block is unable to correctly execute a command, an error code is output to **A\_Status**. The meaning of the error codes is described in the following table.

| Error code  | Symbol            | Description                                     |
|-------------|-------------------|---|
| 16#40800800 | TEMP_NOT_AVAIL    | Service temporary not available                 |
| 16#40801000 | INVALID_PARA      | Invalid parameter                               |
| 16#40801100 | WRONG_STATE       | The slave does not support DP V1                |
| 16#40808000 | FATAL_ERR         | Fatal program error                             |
| 16#40808100 | BAD_CONFIG        | Configuration error in the data area            |
| 16#40808200 | PLC_STOPPED       | The controller was stopped                      |
| 16#4080A000 | READ_ERR          | Error while reading a record                    |
| 16#4080A100 | WRITE_ERR         | Error while writing a record                    |
| 16#4080A200 | MODULE_FAILURE    | The error cannot be specified in greater detail |
| 16#4080B000 | INVALID_INDEX     | Index is invalid                                |
| 16#4080B100 | WRITE_LENGTH      | Wrong length while writing                      |
| 16#4080B200 | INVALID_SLOT      | Slot number invalid                             |
| 16#4080B300 | TYPE_CONFLICT     | Wrong type                                      |
| 16#4080B400 | INVALID_AREA      | Wrong read/write range                          |
| 16#4080B500 | STATE_CONFLICT    | Master in invalid state                         |
| 16#4080B600 | ACCESS_DENIED     | Slave not active (or similar)                   |
| 16#4080B700 | INVALID_RANGE     | Invalid read/write range                        |
| 16#4080B800 | INVALID_PARAMETER | Invalid parameter value                         |
| 16#4080B900 | INVALID_TYPE      | Invalid parameter type                          |
| 16#4080C300 | NO_RESOURCE       | Slave not available                             |
| 16#4080BA00 | BAD_VALUE         | Invalid Value                                   |
| 16#4080BB00 | BUS_ERROR         | Bus error                                       |
| 16#4080BC00 | INVALID_SLAVE     | Invalid slave ID                                |
| 16#4080BD00 | TIMEOUT           | Timeout occurred                                |
| 16#4080C000 | READ_CONSTRAIN    | Read constraint                                 |
| 16#4080C100 | WRITE_CONSTRAIN   | Write constraint                                |
| 16#4080C200 | BUSY              | A function block of this type is already active |
| 16#4080C300 | NO_RESOURCE       | Slave inactive                                  |

Table 155: Error Codes of the Function Blocks

## 6.12 Control Panel (PROFIBUS DP Master)

The Control Panel can be used to verify and control the settings for the PROFIBUS DP master. Details about the current status of the master or slave associated with it (e.g., cycle time, bus state, etc.) are displayed.

### To open Control Panel for monitoring the PROFIBUS DP master

1. Right-click the **Hardware** structure tree node and select **Online** from the context menu.
2. In the **System Login** window, enter the access data to open the online view for the hardware.
3. Double-click **COM Module** and select the **PROFIBUS DP Master** structure tree node.

### 6.12.1 Context Menu (PROFIBUS DP Master)

The following commands can be chosen from the context menu for the selected PROFIBUS DP master:

#### Offline:

Switch off the selected PROFIBUS DP master. If the master is switched off, it is completely disabled.

#### Stop:

Stop the selected PROFIBUS DP master. The master participates in the token protocol but does not send any data to the slaves.

#### Clear:

By clicking the CLEAR button, the selected PROFIBUS DP master adopts a safe state and exchanges safe data with the slaves. The output data sent to the slaves only contains zeros. Failsafe slaves receive failsafe telegrams containing no data. The PROFIBUS DP master ignores the input data from the slaves, and uses the initial values in the user program instead.

#### Operate:

Start the selected PROFIBUS DP master. The PROFIBUS DP master cyclically exchanges I/O data with the slaves.

#### Reset Statistics

The *Reset Statistics* button is used to reset the statistical data such as min. or max. cycle time to zero.

### 6.12.2 Context Menu (PROFIBUS DP Slave)

The following commands can be chosen from the context menu for the selected PROFIBUS DP slave:

#### Activate:

Activate the selected slave which can now exchange data with the PROFIBUS DP master.

#### Deactivate:

Deactivate the selected slave. The communication is terminated.

### 6.12.3 View Box (PROFIBUS Master)

The view box displays the following values of the selected PROFIBUS DP master.

| Element                    | Description   |
|----------------------------|---|
| Name                       | Name of the PROFIBUS DP master  |
| Master State               | Indicate the current protocol state (see Chapter 6.12.4).<br>0 = OFFLINE<br>1 = STOP<br>2 = CLEAR<br>3 = OPERATE<br>100 = UNDEFINED   |
| Bus state                  | Bus error code 0...6:<br>0 = OK<br>1 = Address error:<br>The master address is already available on the bus<br>2 = Bus malfunction:<br>A malfunction was detected on the bus, e.g., bus was not properly terminated, and multiple stations are sending data simultaneously.<br>3 = Protocol error:<br>An incorrectly coded packet was received.<br>4 = Hardware fault:<br>The hardware reported a fault, e.g., too short time periods.<br>5 = Unknown error:<br>The master changed the state for an unknown reason.<br>6 = Controller Reset:<br>With severe bus malfunctions, the controller chip could adopt an undefined state and is reset.<br>The error code retains its value until the bus error has been eliminated. |
| Fieldbus Interface         | FB1, FB2  |
| $\mu$ P load (planned) [%] | Load of the COM module planned for this protocol.   |
| $\mu$ P load (actual) [%]  | Actual load of the COM module for this protocol.  |
| Baud rate [bps]            | Baud rate of the master<br>The master can communicate using all baud rates specified in the standard. Cycle times can be set up to a lower limit of 2 ms.   |
| Fieldbus address           | Master bus address (0 ... 125)  |
| PNO-IdentNo                | 16 bit unique number assigned by the PNO Germany to a product (field device) and identifying it.  |
| Bus Error Count            | Number of the bus error, so far.  |
| MSI [ms]                   | Min. Slave Interval in ms, resolution 0.1 ms  |
| TTR [ms]                   | Target Token Rotation Time in ms, resolution 0.1 ms   |

| Element                 | Description                         |
|-------------------------|-------------------------------------|
| Last Cycle Time [ms]    | Last PROFIBUS DP cycle time [ms]    |
| Minimum Cycle Time [ms] | Minimum PROFIBUS DP cycle time [ms] |
| Average Cycle Time [ms] | Average PROFIBUS DP cycle time [ms] |
| Maximum Cycle Time [ms] | Maximum PROFIBUS DP cycle time [ms] |

Table 156: View Box of the PROFIBUS Master

#### 6.12.4 PROFIBUS DP Master State

The master status is displayed in the view box of the Control Panel and can be evaluated in the user program using the Master Connection State status variable.

| Master State | State of the master  |
|--------------|--|
| OFFLINE      | The master is switched off; no bus activity.   |
| STOP         | The master participates in the token protocol but does not send any data to the slaves.  |
| CLEAR        | The master is in the safe state and exchanges data with the slaves. <ul style="list-style-type: none"> <li>▪ The output data sent to the slaves only contains zeros.</li> <li>▪ Failsafe slaves receive failsafe telegrams containing no data.</li> <li>▪ The input data of the slaves are ignored and instead initial values are used.</li> </ul> |
| OPERATE      | The master is operating and cyclically exchanges I/O data with the slaves.   |
| UNDEFINED    | The firmware for the PROFIBUS DP master module is being updated.   |

Table 157: PROFIBUS DP Master State

#### 6.12.5 Behavior of the PROFIBUS DP Master

Behavior of the PROFIBUS DP master according to the controller operating state.

| State of the Controller | Behavior of the HIMA PROFIBUS DP Master   |
|-------------------------|---|
| STOP *)                 | If the controller is in STOP, the master is in the OFFLINE state.   |
| RUN                     | If the controller is in RUN, the master tries to enter the OPERATE state.   |
| STOP                    | If the controller enters the STOP state, the master adopts the CLEAR state.<br>If the master is already in the STOP or OFFLINE state, it remains in this state. |

\*) After powering up the controller or loading the configuration

Table 158: Behavior of the PROFIBUS DP Master

### 6.12.6 Function of the FBx LED in the PROFIBUS Master

The FBx LED of the corresponding fieldbus interface indicates the state of the PROFIBUS DP protocol. The states of the FBx LED are specified in the following table:

| FBx LED                     | Description   |
|-----------------------------|---|
| OFF                         | No configuration or invalid configuration of the PROFIBUS DP master.                            |
| Blinking<br>Every 2 seconds | Valid configuration.<br>The PROFIBUS DP master is in the OFFLINE or STOP state.                 |
| ON                          | The PROFIBUS DP master is in the OPERATE or CLEAR state, if all activated slaves are connected. |
| Blinking, every second      | At least one slave failed.  |

Table 159: The FBx LED

### 6.12.7 Function of the FAULT LED in the PROFIBUS Master (HIMax only)

The FAULT LED of the corresponding fieldbus interface indicates that the PROFIBUS DP protocol has failed. The states of the FAULT LED are specified in the following table:

| FAULT LED | Color | Description   |
|-----------|-------|---|
| OFF       | Red   | The PROFIBUS DP protocols is not disturbed.   |
| Blinking  | Red   | The protocol is disturbed. <ul style="list-style-type: none"> <li>▪ At least one slave has failed.</li> <li>▪ Bus error detected.</li> <li>▪ Calculating time budget exceeded.</li> </ul> If no faults occur for a period longer than 5 seconds, the state changes to <i>Protocol not disturbed</i> . |

Table 160: The FAULT FBx

## 6.13 HIMA PROFIBUS DP Slave

This chapter describes the characteristics of the HIMA PROFIBUS DP slave and the menu functions and dialog boxes in SILworX required for configuring the HIMA PROFIBUS DP slave.

### Equipment and System Requirements

| Element         | Description   |
|-----------------|---|
| HIMA controller | HIMax with COM module<br>HIMatrix: CPU OS version 7 and beyond and COM OS version 12 and beyond   |
| COM Module      | The serial fieldbus interface (FB1 or FB2) used on the COM module must be equipped with an optional HIMA PROFIBUS DP slave submodule.<br>For more information on the interface assignment, see Chapter 3.6. |
| Activation      | Activation through the plug-in module, see Chapter 3.4.   |

Table 161: Equipment and System Requirements for the HIMA PROFIBUS DP Slave

- i** For the HIMax system, HIMA recommends operating the PROFIBUS DP using the FB1 fieldbus interface (maximum transfer rate 12 Mbit). The maximum transfer rate permitted for the FB2 fieldbus interface is 1.5 Mbit.

### PROFIBUS DP Slave Properties

| Element   | Description  |
|---|--|
| Type of HIMA PROFIBUS DP slave                  | DP V0  |
| Transfer rate                                   | 9.6 kbit/s...12 Mbit/s   |
| Bus address                                     | 0...125  |
| Max. number of slaves                           | One HIMA PROFIBUS DP slave can be configured for each COM module.  |
| Process data volume of a HIMA PROFIBUS DP slave | DP-Output: max. 192 bytes<br>DP-Input: max. 240 bytes<br>Total: max. 256 bytes   |
| Protocol watchdog                               | If the COM is in RUN and the connection to the PROFIBUS DP master is lost, the DP slave detects this once the watchdog timeout has expired (the watchdog timeout must be set in the master). In this case, the DP output data (or input data from the perspective of the resource) are reset to their initial value and the <i>Data Valid</i> flag (status variable of the DP slave protocol) is set to FALSE. |

Table 162: Properties of the HIMA PROFIBUS DP Slave

#### 6.13.1 Creating a HIMA PROFIBUS DP Slave

##### To create a new HIMA PROFIBUS DP Slave

1. In the structure tree, open **Configuration, Resource, Protocols**.
2. Select **New, PROFIBUS DP Slave** from the context menu for protocols to add a new PROFIBUS DP slave .
3. Select **Edit** from the context menu for the PROFIBUS DP slave.
4. In the **Properties** tab, click **Module** and **Interface**.

## 6.14 Menu Functions of the PROFIBUS DP Slave

### 6.14.1 Edit

The **Edit** dialog box for the PROFIBUS DP master contains the following tabs:

#### Process Variables

The send and receive variables are created in the **Process Variables** tab.

#### Input Variables

The variables that the current controller should receive are entered in the *Input Signals* area.

Any variables can be created in the *Input Signals* area. Offsets and types of the received variables must be identical with offsets and types of the send variables of the communication partner.

#### Output Variables

The variables for cyclic data exchange sent by this controller are entered in the *Output Signals* area.

Any variables can be created in the *Output Signals* area. Offsets and types of the received variables must be identical with offsets and types of the receive variables of the communication partner.

#### System Variables

The variables that should be read in the controller are defined in the **System Variables** tab.

The **System Variables** tab contains the following system variables that are required to evaluate the state of the PROFIBUS DP slave from within the user program.

| Element           | Description   |
|-------------------|---|
| Current baud rate | Baud rate currently used by the PROFIBUS DP slave protocol.   |
| Data valid        | If the status variable <i>Data Valid</i> is set to TRUE, the slave received valid import data from the master.<br>The status variable is set to FALSE if the watchdog time within the slave has expired.<br><br>Default value: FALSE<br><br>Note:<br>If the master did not activate the slave's watchdog and the connection is lost, the <i>Data Valid</i> status variable retains the value TRUE since the PROFIBUS DP slave has no means to recognize that the connection was lost.<br><br>This fact must be taken into account when using this variable! |
| Error Code        | If an error occurred in the PROFIBUS DP slave protocol, the error is transferred to this variable. The last occurred error is displayed.<br><br>Possible (hexadecimal) value:<br>0x00: No error<br>0xE1: Faulty configuration by the PROFIBUS Master<br>0xD2: Faulty configuration by the PROFIBUS DP Master<br><br>Default value: 0x00   |
| Master Address    | This is the address of the PROFIBUS master that configured its own PROFIBUS DP slave.<br><br>Possible (decimal) values:<br>0-125: Master Address<br>255: The slave is not assigned to any master<br><br>Default value: 255  |
| Protocol State    | Describe the state of the PROFIBUS DP slave protocol<br><br>Possible (hexadecimal) value:<br>0xE1: The controller is disconnected from the bus or not active.<br>0xD2: The controller waits for a configuration from the master.<br>0xC3: The controller cyclically exchanges data with the master.<br><br>Default value: 0xE1  |
| Slave Address     | Address of the controller's PROFIBUS DP slave. This address was previously configured by the user using the PADT.<br><br>Possible (decimal) values:<br>0-125: Slave Address   |
| Watchdog Time     | Watchdog time in milliseconds configured in the master. See Chapter 6.6.3.  |

Table 163: System Variables in the PROFIBUS DP Slave

### 6.14.2 Properties

The **Properties** tab for the HIMA PROFIBUS DP slave contains the following parameters for configuring the PROFIBUS DP slave.

The default values for *Within one cycle* and *Process Data Refresh Rate [ms]* provide a fast means of exchanging PROFIBUS DP data between the COM module (COM) and the PROFIBUS DP slave hardware of the HIMax/HIMatrix controller.

These parameters should only be changed if it is necessary to reduce the COM load for an application, and the process allows this change.

- 
- i** Only experienced programmers should modify the parameters.  
Increasing the refresh rate for the COM and PROFIBUS DP hardware means that the effective refresh rate of the PROFIBUS DP data is also increased. The system time requirements must be verified.
- 

The **Min. Slave Interval [ms]** parameter defines the minimum refresh rate of the PROFIBUS DP data between PROFIBUS DP master and PROFIBUS DP slave, see Timings Tab, Chapter 6.3.2).

| Element                             | Description   |
|-------------------------------------|---|
| Type                                | PROFIBUS DP slave   |
| Name                                | Name of the PROFIBUS DP Slave   |
| Module                              | Selection of the COM module within which the protocol is processed.   |
| Activate Max. $\mu$ P Budget        | Activated:<br>Adopt the $\mu$ P budget limit from the Max. $\mu$ P Budget in [%] field.<br><br>Deactivated:<br>Do not use the $\mu$ P budget limit for this protocol.   |
| Max. $\mu$ P budget in [%]          | Maximum $\mu$ P budget for the module that can be used for processing the protocols.<br>Range of values: 1...100%<br>Default value: 30%   |
| Behavior on CPU/COM connection loss | If the connection of the processor module to the communication module is lost, the input variables are initialized or continue to be used unchanged in the process module, depending on this parameter.<br>For instance, if the communication module is removed when communication is running.<br><b>If a project created with a SILworX version prior to V3 should be converted, this value must be set to Retain Last Value to ensure that the CRC does not change.</b><br><b>For HIMatrix controllers with CPU OS version prior to V8, this value must always be set to Retain Last Value.</b><br><br>Adopt Initial Data      Input variables are reset to their initial values.<br>Retain Last Value      The input variables retains the last value. |
| Station address                     | Slave station address.<br>Only one slave station address may be available on the bus.<br>Range of values: 1...125<br>Default value: 0   |

| Interface                      | Fieldbus interface that should be used for the PROFIBUS DP slave.<br>Range of values: fb1, fb2<br>Default value: None   |       |           |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
|--------------------------------|---|-------|-----------|-----|-----|------|------------|---|---|-------|-------------|---|---|-------|--------------|---|---|-------|--------------|---|---|--------|--------------|---|---|--------|------------|---|---|---------|------------|---|---|---------|----------|---|---|---------|----------|---|---|----------|-----------|---|---|
| Baud rate [bps]                | <p>Baud rate used for the bus.</p> <p>Possible values:</p> <table border="1"> <thead> <tr> <th>Value</th><th>Baud Rate</th><th>FB1</th><th>FB2</th></tr> </thead> <tbody> <tr><td>9600</td><td>9.6 kbit/s</td><td>X</td><td>X</td></tr> <tr><td>19200</td><td>16.2 kbit/s</td><td>X</td><td>X</td></tr> <tr><td>45450</td><td>45.45 kbit/s</td><td>X</td><td>X</td></tr> <tr><td>93750</td><td>93.75 kbit/s</td><td>X</td><td>X</td></tr> <tr><td>187500</td><td>187.5 kbit/s</td><td>X</td><td>X</td></tr> <tr><td>500000</td><td>500 kbit/s</td><td>X</td><td>X</td></tr> <tr><td>1500000</td><td>1.5 Mbit/s</td><td>X</td><td>X</td></tr> <tr><td>3000000</td><td>3 Mbit/s</td><td>X</td><td>-</td></tr> <tr><td>6000000</td><td>6 Mbit/s</td><td>X</td><td>-</td></tr> <tr><td>12000000</td><td>12 Mbit/s</td><td>X</td><td>-</td></tr> </tbody> </table> | Value | Baud Rate | FB1 | FB2 | 9600 | 9.6 kbit/s | X | X | 19200 | 16.2 kbit/s | X | X | 45450 | 45.45 kbit/s | X | X | 93750 | 93.75 kbit/s | X | X | 187500 | 187.5 kbit/s | X | X | 500000 | 500 kbit/s | X | X | 1500000 | 1.5 Mbit/s | X | X | 3000000 | 3 Mbit/s | X | - | 6000000 | 6 Mbit/s | X | - | 12000000 | 12 Mbit/s | X | - |
| Value                          | Baud Rate   | FB1   | FB2       |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
| 9600                           | 9.6 kbit/s  | X     | X         |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
| 19200                          | 16.2 kbit/s   | X     | X         |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
| 45450                          | 45.45 kbit/s  | X     | X         |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
| 93750                          | 93.75 kbit/s  | X     | X         |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
| 187500                         | 187.5 kbit/s  | X     | X         |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
| 500000                         | 500 kbit/s  | X     | X         |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
| 1500000                        | 1.5 Mbit/s  | X     | X         |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
| 3000000                        | 3 Mbit/s  | X     | -         |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
| 6000000                        | 6 Mbit/s  | X     | -         |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
| 12000000                       | 12 Mbit/s   | X     | -         |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
| Process Data Refresh Rate [ms] | Refresh rate in milliseconds at which the COM and the PROFIBUS DP slave hardware exchange protocol data.<br>Range of values: 4...1000<br>Default value: 10  |       |           |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |
| Force Process Data Consistency | <p>Activated:<br/>Transfer of all protocol data from the CPU to the COM within a CPU cycle.</p> <p>Deactivated:<br/>Transfer of all protocol data from the CPU to the COM, distributed over multiple CPU cycles, each with 1100 byte per data direction. This can also allow lowering the cycle time of the controller.</p> <p>Default value: activated</p>   |       |           |     |     |      |            |   |   |       |             |   |   |       |              |   |   |       |              |   |   |        |              |   |   |        |            |   |   |         |            |   |   |         |          |   |   |         |          |   |   |          |           |   |   |

Table 164: Slave Properties: General Tab

## 6.15 Control Panel (Profibus DP Slave)

The Control Panel can be used to verify and control the settings for the PROFIBUS DP slave. Details about the slave's current status (e.g., cycle time, bus state, etc.) are displayed.

### To open Control Panel for monitoring the PROFIBUS DP Slave

1. Right-click the **Hardware** structure tree node and select **Online** from the context menu.
2. In the **System Login** window, enter the access data to open the online view for the hardware.
3. Double-click **COM Module** and select the **PROFIBUS DP Slave** structure tree node.

### 6.15.1 View Box (PROFIBUS DP Slave)

The view box displays the following values of the selected PROFIBUS DP slave.

| Element                      | Description  |
|------------------------------|--|
| Name                         | Name of the PROFIBUS DP slave  |
| Fieldbus Interface           | Subordinated slave's fieldbus interface  |
| Protocol State               | Connection State<br>0 = Deactivated,<br>1 = Inactive (connection attempt)<br>2 = Connected |
| Error State                  | See Chapter 6.14.1   |
| Timeout                      | Watchdog time in milliseconds configured in the master. See Chapter 6.6.3                  |
| Watchdog Time [ms]           | It is set in the master. See Chapter 6.6.3.  |
| Fieldbus address             | See Chapter 6.14.2.  |
| Master Address               | Address of the PROFIBUS DP master.   |
| Baud rate [bps]              | Current baud rate. See Chapter 6.14.2.   |
| $\mu$ P budget (planned) [%] | Load of the COM module planned for this protocol.  |
| $\mu$ P budget (actual) [%]  | Actual load of the COM module for this protocol.   |

Table 165: View Box of the PROFIBUS DP Slave

### 6.16 Function of the FBx LED in the PROFIBUS Slave

The FBx LED of the corresponding fieldbus interface indicates the state of the PROFIBUS DP protocol. The states of the FBx LED are specified in the following table:

| FBx LED                     | Color  | Description   |
|-----------------------------|--------|---|
| OFF                         | Yellow | The PROFIBUS DP protocol is not active!<br>This means that the controller is in the STOP state or no PROFIBUS DP slave is configured. |
| Blinking<br>every 2 seconds | Yellow | No data traffic!<br>The PROFIBUS DP slave is configured and ready.  |
| ON                          | Yellow | The PROFIBUS DP protocol is active and is exchanging data with the PROFIBUS DP master.  |

Table 166: The FBx LED

### 6.17 Function of the FAULT LED in the PROFIBUS Slave (HIMax only)

The FAULT LED of the corresponding fieldbus interface indicates that the PROFIBUS DP protocol has failed. The states of the FAULT LED are specified in the following table:

| FAULT LED | Color | Description  |
|-----------|-------|--|
| OFF       | Red   | The PROFIBUS DP protocols is not disturbed.  |
| Blinking  | Red   | The protocol is disturbed. <ul style="list-style-type: none"> <li>▪ The configurations of the PROFIBUS DP master and slave are faulty.</li> <li>▪ Calculating time budget exceeded.</li> </ul> If no faults occur for a period longer than 5 seconds, the state changes to <i>Protocol not disturbed</i> . |

Table 167: The FAULT FBx

## 7 Modbus

The Modbus coupling of HIMax/HIMatrix systems to almost any process control and visual display system can be achieved either directly, using the RS485 interfaces, or indirectly, using the Ethernet interfaces of the controllers. HIMax/HIMatrix systems can operate both as master and slave.

The Modbus functionality makes it particularly easy to connect to Control Panels or other controllers. Given its intensive use in projects worldwide, Modbus has been proven through countless applications.

Modbus master (see Chapter 7.1)

Redundancy of the Modbus master must be configured in the user program such that the user program monitors the redundant transmission paths and assigns the redundantly transmitted process data to the corresponding transmission path.

Modbus slave (see Chapter 7.3)

The Modbus slave can be configured redundantly.



HIMax/HIMatrix controllers and the communication partner must be located in the same subnet, if the Ethernet interfaces are used as transmission channel, or they must have the corresponding routing settings if a router is used.

### 7.1 RS485 Bus Topology

The following picture shows the structure of an RS485 bus topology using HIMA components. H 7506 are used as bus terminals. The permissible maximum total bus length is 1200 m. Repeaters such as H 7505<sup>1)</sup> must be used for distances greater than 1200 m. A total of 3 repeaters may be used. The bus may be extended up to 4800 m.

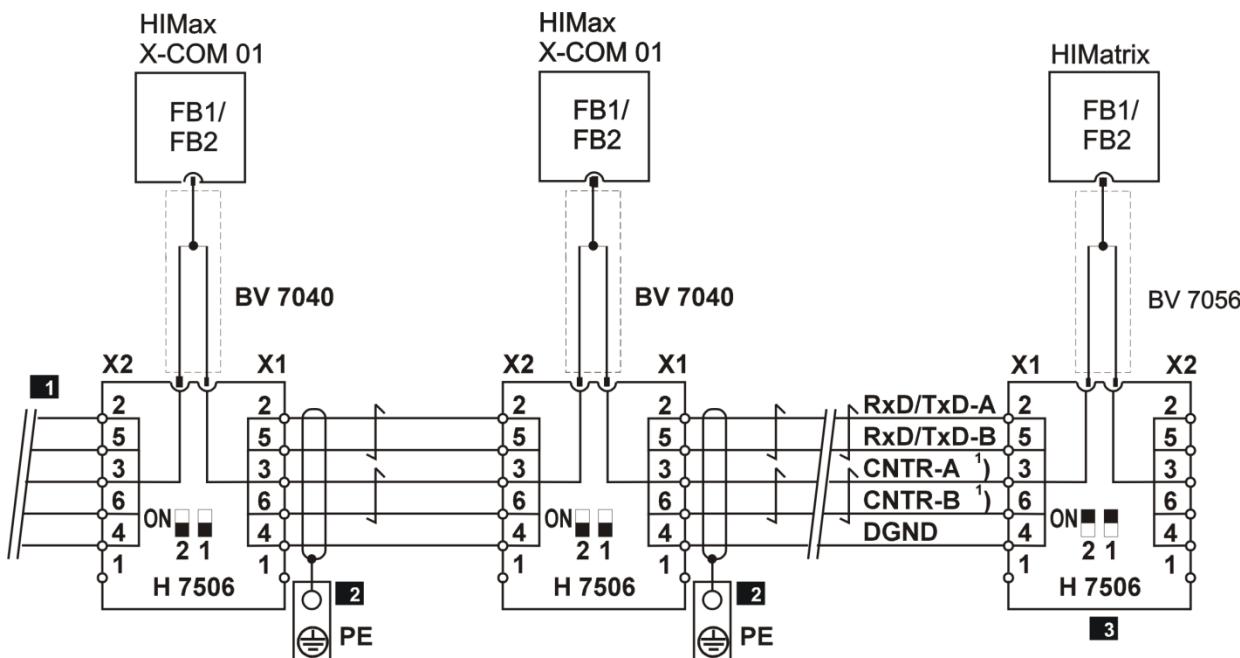


Figure 57: RS485 Bus Topology

<sup>1)</sup>If fibre optic cable/RS485 converters are used in the bus, H 7505 must not be used (no automatic switching of data direction).

- 
- i** Potential bonding should be used if the bus extends over larger distances.
- 

### 7.1.1 H 7506 Terminal Assignment

The following table shows the terminal assignment of the H 7506 bus terminal. The HIMA BV 7040 cable connects the H 7506 to the FBx fieldbus interface of the controller.

| X1/X2 | Color | Description                          |
|-------|-------|--------------------------------------|
| 1     | -     | -                                    |
| 2     | WH    | RxD/TxD-A, data line                 |
| 3     | GN    | CNTR-A, controller line for repeater |
| 4     | GY    | DGND                                 |
| 5     | BN    | RxD/TxD-B, data line                 |
| 6     | YE    | CNTR-B, controller line for repeater |

Table 168: Terminal Assignment for H 7506

- 
- i** Refer to the HIMA's website for more information on this topic and on additional HIMA RS485 components.
- 

### 7.1.2 Bus Cable

For bus cabling, HIMA recommends using shielded twisted pair wires with the following characteristics:

| Element            | Description                        |
|--------------------|------------------------------------|
| Cable type         | LiYCY 3 x 2 x 0.25 mm <sup>2</sup> |
| Wire cross-section | > 0.25 mm <sup>2</sup>             |
| Impedance          | 100...120 Ω                        |

Table 169: Bus Cable

### 7.1.3 Properties of the RS485 Transmission

| Element                        | Description  |
|--------------------------------|--|
| Network Topology               | Linear bus, active bus termination on both ends                                |
| Medium                         | Shielded, twisted pair wires   |
| Connector                      | 9 pin SUB-D connector. Refer to Chapter 3.6 for details on the pin assignment. |
| Stations for each segment      | 32 stations in every segment, without repeaters <sup>1)</sup>                  |
| Total of stations for each bus | 1 Modbus master, 3 repeaters <sup>1)</sup><br>121 Modbus slaves                |
| Max. length of a bus segment   | 1200 m for each segment  |
| Max. length of the bus         | 4800 m, 4 segments with 3 repeaters <sup>1)</sup>                              |
| Max. baud rate                 | HIMax: 38400 bit/s<br>HIMatrix: 115000 bit/s                                   |

<sup>1)</sup> For each repeater used, the maximum number of stations in this segment decreases by 1. This means that a maximum of 31 stations may be operated on this segment. According to the standard, a total of three repeaters may be used such that a maximum of 121 Modbus slaves may be connected for each serial Modbus master interface.

Table 170: Properties of the RS485 Transmission

## 7.2

### HIMA Modbus Master

The data transfer between HIMA Modbus master and the Modbus slaves can be configured via the serial interface (RS485) as well as via the TCP/UDP (Ethernet). Additionally, the HIMA Modbus master can be used as a gateway (Modbus: TCP/UDP -> RS485).

#### Equipment and System Requirements

| Element          | Description   |
|------------------|---|
| HIMA controller  | HIMax with COM module<br>HIMatrix: CPU OS version 7 and beyond and COM OS version 12 and beyond   |
| Processor module | The Ethernet interfaces on the processor module may not be used for Modbus TCP.   |
| COM module       | Ethernet 10/100BaseT<br>Pin assignment of the D-Sub connectors FB1 and FB2<br>If Modbus RTU is used, each of the serial fieldbus interfaces (FB1 and FB2) used on the COM module must be equipped with an optional HIMA RS485 submodule. For more information on the interface assignment, see Chapter 3.6. |
| Activation       | Each of the two Modbus master functions must be enabled individually, see Chapter 3.4.<br>Modbus master RTU (RS485) and Modbus master TCP<br>The Modbus master RTU license is required for the Modbus gateway.  |

Table 171: Equipment and System Requirements for the Modbus Master

#### Modbus Master Properties

| Property   | Description   |
|--|---|
| Modbus Master                                      | One Modbus master can be configured for each COM module or each HIMatrix controller.<br>The Modbus master can simultaneously: <ul style="list-style-type: none"><li>- Exchange data with TCP/UDP slaves</li><li>- Exchange data with serial slaves</li><li>- Be used as gateway from Modbus TCP to Modbus RTU.</li></ul>  |
| Max. number of Modbus slaves HIMax/HIMatrix        | One Modbus master can operate up to 247 slaves. <ul style="list-style-type: none"><li>- 121 Modbus slaves per serial interface (FB1, FB2)</li><li>- 64 TCP slaves through the TCP/IP connection</li><li>- 247 UDP slaves through the UDP/IP connection</li></ul> The maximum number of UDP slaves is limited since the slaves must be managed on the master side. |
| Max. number of request telegrams                   | Up to 988 request telegrams can be configured per Modbus master.  |
| Max. process data length for each request telegram | The process data length for HIMA-specific request telegrams is 1100 bytes, see Chapter 7.2.4.2.   |
| Max. Size of Send Data                             | See Table 2 Standard Protocols  |
| Max. Size of Receive Data                          | <b>i</b> The status bytes of the master and the status bytes of each slave assigned to it must be subtracted from the max. size of send data.   |

|   |  |                   |            |    |  |  |               |   |   |   |   |                             |    |    |    |    |                      |    |    |    |    |               |    |    |    |    |
|---|--|-------------------|------------|----|--|--|---------------|---|---|---|---|-----------------------------|----|----|----|----|----------------------|----|----|----|----|---------------|----|----|----|----|
| Display format of the Modbus data<br>The HIMax/HIMatrix controllers use the big endian format.<br>Example: 32-bit data (e.g., DWORD, DINT): | <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">32-bit data (hex)</td><td colspan="4" style="text-align: center;">0x12345678</td></tr> <tr> <td>Memory offset</td><td>0</td><td>1</td><td>2</td><td>3</td></tr> <tr> <td>Big endian (HIMax/HIMatrix)</td><td>12</td><td>34</td><td>56</td><td>78</td></tr> <tr> <td>Middle endian (H51q)</td><td>56</td><td>78</td><td>12</td><td>34</td></tr> <tr> <td>Little endian</td><td>78</td><td>56</td><td>34</td><td>12</td></tr> </table> | 32-bit data (hex) | 0x12345678 |    |  |  | Memory offset | 0 | 1 | 2 | 3 | Big endian (HIMax/HIMatrix) | 12 | 34 | 56 | 78 | Middle endian (H51q) | 56 | 78 | 12 | 34 | Little endian | 78 | 56 | 34 | 12 |
| 32-bit data (hex)   | 0x12345678   |                   |            |    |  |  |               |   |   |   |   |                             |    |    |    |    |                      |    |    |    |    |               |    |    |    |    |
| Memory offset   | 0  | 1                 | 2          | 3  |  |  |               |   |   |   |   |                             |    |    |    |    |                      |    |    |    |    |               |    |    |    |    |
| Big endian (HIMax/HIMatrix)   | 12   | 34                | 56         | 78 |  |  |               |   |   |   |   |                             |    |    |    |    |                      |    |    |    |    |               |    |    |    |    |
| Middle endian (H51q)  | 56   | 78                | 12         | 34 |  |  |               |   |   |   |   |                             |    |    |    |    |                      |    |    |    |    |               |    |    |    |    |
| Little endian   | 78   | 56                | 34         | 12 |  |  |               |   |   |   |   |                             |    |    |    |    |                      |    |    |    |    |               |    |    |    |    |

Table 172: Properties of the Modbus Master

According to the standard, a total of three repeaters may be used such that a maximum of 121 slaves are possible per serial master interface.

### 7.2.1 Modbus Example

In this example, the HIMA Modbus master exchanges data with a HIMA Modbus slave through Modbus TCP. Both controllers are connected via the Ethernet interface of the communication modules.

- i If the Modbus slave and the Modbus master are located in different subnets, the routing table must contain the corresponding user-defined routes.

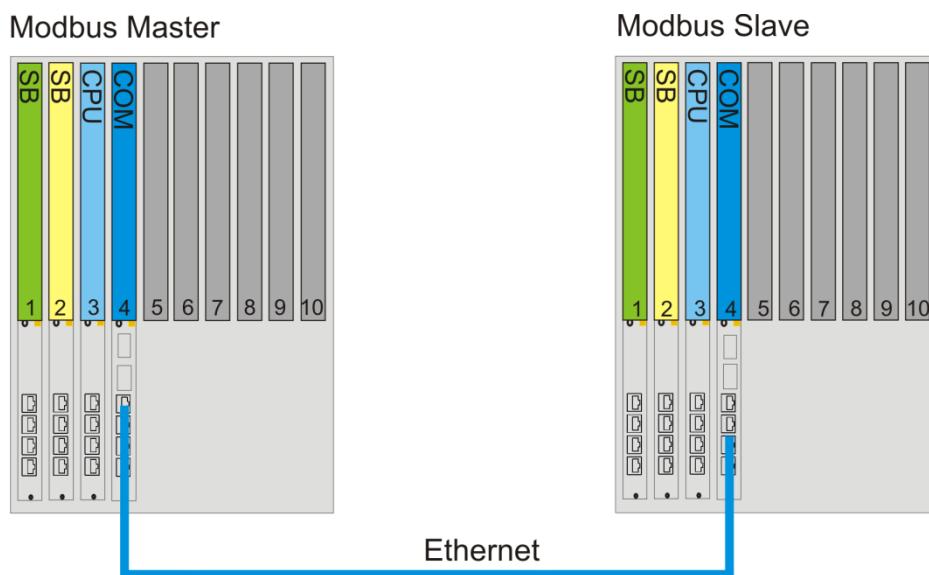


Figure 58: Communication via Modbus TCP/IP

For this example, the following global variables must be created in SILworX:

| Global Variables      | Type |
|-----------------------|------|
| Master->Slave_BOOL_00 | BOOL |
| Master->Slave_BOOL_01 | BOOL |
| Master->Slave_BOOL_02 | BOOL |
| Master->Slave_WORD_00 | WORD |
| Master->Slave_WORD_01 | WORD |
| Slave->Master_WORD_00 | WORD |
| Slave->Master_WORD_01 | WORD |

### 7.2.1.1 Configuring the Modbus TCP Slave

#### To create a new HIMA Modbus slave

1. In the structure tree, open **Configuration, Resource, Protocols**.
2. On the context menu for protocols, select **New, Modbus Slave Set** to add a new Modbus slave set.
3. Select **Edit** from the context menu for the Modbus slave set, open the **Modbus Slave Set Properties**, and retain the default values.
4. Select the **Modbus slave** tab and perform the following actions:
  - Select **COM Module**
  - Activate **Enable TCP**
  - The remaining parameters retain the default values.

#### To configure the bit input variables of the Modbus slave



The Boolean variables that the master addresses bit by bit are entered in the Bit Variables tab (function code 1, 2, 5, 15).

1. From the context menu for the Modbus slave, click **Edit** and then select the **Bit Variables** tab.
2. Drag the following global variables from the **Object Panel** onto the **Bit Inputs** area.

| Bit address | Bit variable          | Type |
|-------------|-----------------------|------|
| 0           | Master->Slave_BOOL_00 | BOOL |
| 1           | Master->Slave_BOOL_01 | BOOL |
| 2           | Master->Slave_BOOL_02 | BOOL |

3. Right-click anywhere in the **Register Inputs** area, and then click **New Offsets** to renumber the variable offsets.

#### To configure the register input variables of the Modbus slave



The variables that the master addresses 16 register by register are entered in the Register Variables tab (function code 3, 4, 6, 16, 23).

1. From the context menu for the Modbus slave, click **Edit** and then select the **Register Variables** tab.
2. Drag the following variables from the **Object Panel** onto the **Register Inputs** area.

| Register address | Register variables    | Type |
|------------------|-----------------------|------|
| 0                | Master->Slave_WORD_00 | WORD |
| 1                | Master->Slave_WORD_01 | WORD |

3. Right-click anywhere in the **Register Inputs** area, and then click **New Offsets** to renumber the variable offsets.

#### To configure the register output variables of the Modbus slave

1. From the context menu for the Modbus slave, click **Edit** and then select the **Register Variables** tab.
2. Drag the following variables from the **Object Panel** onto the **Register Outputs** area.

| Register address | Register variables    | Type |
|------------------|-----------------------|------|
| 0                | Slave->Master_WORD_00 | WORD |
| 1                | Slave->Master_WORD_01 | WORD |

3. Right-click anywhere in the **Register Outputs** area, and then click **New Offsets** to renumber the variable offsets.

#### To check the Modbus TCP slave configuration

1. Open the context menu for the Modbus TCP master and click **Verification**.
2. Thoroughly verify the messages displayed in the logbook and correct potential errors.

### 7.2.1.2 Configuring the Modbus TCP Master

#### To create the HIMA Modbus master

1. In the structure tree, open **Configuration, Resource, Protocols**.
2. On the context menu for protocols, click **New, Modbus Master** to add a new Modbus master.
3. From the context menu for the Modbus master, **Properties, General**.
4. Click **COM Module**.  
The remaining parameters retain the default values.

#### To create the connection to the Modbus TCP slave in the Modbus master

1. In the structure tree, open **Resource, Protocols, Modbus Master, Ethernet Slaves**.
  2. Right-click **Ethernet Slaves**, then click **New**.
  3. Select **TCP/UDP Slaves** from the list and click **OK** to confirm.
  4. To configure the TCP/UDP slave in the Modbus master:
    - Click **Edit** to assign the system variables, see Chapter 7.2.6.2.
    - Click **Properties** to configure the properties, see Chapter 7.2.6.3.  
Enter the **IP address** of the TCP/UDP slave in the slave's properties.
- The remaining parameters retain the default values.

#### To configure the write request telegram for the bit output variable:

1. Right-click **TCP/UDP slaves**, then click **New**.
2. From the list, select **Write Multiple Coils (15)**.
3. Right-click **Write Multiple Coils (15)**, then click **Properties**.
  - Enter **0** in the **start address of the write area**
4. Right-click **Read Multiple Coils (15)**, then click **Edit**.
5. Drag the following variables from the **Object Panel** onto the **Output Variables** tab..

| Offset | Bit variables         | Type |
|--------|-----------------------|------|
| 0      | Master->Slave_BOOL_00 | BOOL |
| 1      | Master->Slave_BOOL_01 | BOOL |
| 2      | Master->Slave_BOOL_02 | BOOL |

6. Right-click anywhere in the **Output Variables** area to open the context menu and click **New Offsets** to renumber the variable offsets.

**To configure the write request telegram for the register output variable:**

1. Right-click **TCP/UDP slaves**, then click **New**.
2. From the list, select **Write Multiple Registers (16)**.
3. Right-click **Write Multiple Register (16)**, then click **Properties**.
  - Enter **0** in the **start address of the write area**
4. Right-click **Write Multiple Registers (16)**, then click **Edit**.
5. Drag the following variables from the **Object Panel** onto the **Output Variables** tab..

| Offset | Register variables    | Type |
|--------|-----------------------|------|
| 0      | Master->Slave_WORD_00 | WORD |
| 1      | Master->Slave_WORD_01 | WORD |

6. Right-click anywhere in the **Output Variables** area to open the context menu and click **New Offsets** to renumber the variable offsets.

**To define the request telegram for reading the input variables in the Modbus master**

1. Right-click **TCP/UDP slaves**, then click **New**.
2. From the list, select **Read Holding Registers (03)**.
3. Right-click **Read Holding Registers (03)**, then click **Properties**.
  - Enter **0** in the **start address of the read area**.
4. Right-click **Read Holding Registers (03)**, then click **Edit**.
5. Drag the following variables from the **Object Panel** onto the **Input Variables** tab..

| Offset | Register variables    | Type |
|--------|-----------------------|------|
| 0      | Slave->Master_WORD_00 | WORD |
| 1      | Slave->Master_WORD_01 | WORD |

6. Right-click anywhere in the **Input Variables** area to open the context menu and click **New Offsets** to renumber the variable offsets.

**To check the Modbus TCP master configuration**

1. Open the context menu for the Modbus TCP master and click **Verification**.

**To check the Modbus TCP master configuration**

1. Open the context menu for the Modbus TCP master and click **Verification**.
2. Thoroughly verify the messages displayed in the logbook and correct potential errors.

**To create the code for the controllers**

1. Start the code generator for the master and slave resource.
2. Make sure that the code was generated without error.
3. Load the codes into the master and slave controllers respectively.

## 7.2.2 Example of Alternative Register/Bit Addressing

In this example, the configuration defined in Chapter 7.2.1 is extended by 16 Boolean variables in the Register Area. The 16 Boolean variables are read with the request telegram **Write Multiple Coils (15)**, see also Chapter 7.3.11.

### To configure the input variables in the Modbus slave

1. From the context menu for the Modbus slave, click **Edit** and then select the **Register Variables** tab.
2. Drag the 16 new Boolean variables from the **Object Panel** onto the **Register Inputs** area.

| Register address | Register variables           | Type |
|------------------|------------------------------|------|
| 0                | Master->Slave_WORD_00        | WORD |
| 1                | Master->Slave_WORD_01        | WORD |
| 2                | Master->Slave_BOOL_03 ..._18 | BOOL |

Add one again

3. Right-click anywhere in the **Register Inputs** area, and then click **New Offsets** to renumber the variable offsets.

### To configure the alternative register/bit addressing in the Modbus slave

1. Right-click the Modbus slave and select **Edit**, and **Offsets**, then activate **Use Alternative Register/Bit Addressing**.
2. In this example, use the following offsets for the alternative areas:

|                                  |      |
|----------------------------------|------|
| Register Area Offset Bits Input  | 1000 |
| Register Area Offset Bits Output | 1000 |
| Bit Area Offset Register Input   | 8000 |
| Bit Area Offset Register Output  | 8000 |



To use the Modbus request telegram **Write Multiple Coils (15)** to access the Boolean variables in the **Register Variables** area, the variables must be mirrored in the **Bit Variables** area.

### To configure the write request telegram for the output variable (BOOL) in the Modbus master

1. Right-click **TCP/UDP slaves**, then click **New**.
2. From the list, select **Write Multiple Coils (15)**.
3. Right-click **Write Multiple Coils (15)**, then click **Properties**.
  - Enter **8032** in the **start address of the write area**
4. Right-click **Read Multiple Coils (15)**, then click **Edit**.
5. Drag the following variables from the **Object Panel** onto the **Output Variables** tab..

| Offset  | Mirrored Register Variable  | Type |
|---------|-----------------------------|------|
| 0 to 15 | Master->Slave_BOOL_03..._18 | BOOL |

6. Right-click anywhere in the **Output Variables** area to open the context menu and click **New Offsets** to renumber the variable offsets.

## 7.2.3 Menu Functions of the HIMA Modbus Master

### 7.2.3.1 Edit

The **Edit** dialog box for the Modbus master contains the following tab:

#### System Variables

The **System Variables** tab contains system variables that are required to control the Modbus master and evaluate its state from within the user program.

| Element                          | Description  |
|----------------------------------|--|
| Slave Connection Error Count     | Number of faulty connections with Modbus slaves that are in the Activated state. Deactivated Modbus slaves are not taken into account.   |
| Modbus Master Activation Control | Stop or start the Modbus master from within the user program.<br>0: Activate<br>1: Deactivate<br>(Edge triggered!<br>The Modbus master can be activated using the PADT even if the Modbus master activation control is 1). |
| Modbus Master Bus Error          | Bus error on RS485, e.g., telegram error (code unknown etc), length error.   |
| Modbus Master State              | It indicates the current protocol state:<br>1: OPERATE<br>0: OFFLINE   |
| Reset All Slave Errors           | A change from FALSE to TRUE resets all slave and bus errors.   |

Table 173: System Variables for the Modbus Master

### 7.2.3.2 Properties

The **Properties** function of the context menu for the Modbus master is used to open the *Properties* dialog box.

The dialog box contains the following tabs:

#### General

The **General** tab contains the name and a description for the Modbus master. This tab is also used to set the parameters for specifying whether the Modbus master should also operate as a TCP and/or a UDP gateway.

| Element  | Description   |
|--|---|
| Type   | Modbus Master   |
| Name   | Name for the Modbus master  |
| Description  | A description for the Modbus master.  |
| Module   | Selection of the COM module within which the protocol is processed.   |
| Activate Max. $\mu$ P Budget                           | Activated:<br>Adopt the $\mu$ P budget limit from the Max. $\mu$ P Budget in [%] field.<br><br>Deactivated:<br>Do not use the $\mu$ P budget limit for this protocol.   |
| Max. $\mu$ P budget in [%]                             | Maximum $\mu$ P budget for the module that can be used for processing the protocols.<br><br>Range of values: 1...100%<br>Default value: 30%   |
| Behavior on CPU/COM Connection Loss                    | If the connection of the processor module to the communication module is lost, the input variables are initialized or continue to be used unchanged in the process module, depending on this parameter.<br><br>For instance, if the communication module is removed when communication is running.<br><br><b>If a project created with a SILworX version prior to V3 should be converted, this value must be set to Retain Last Value to ensure that the CRC does not change.</b><br><b>For HIMatrix controllers with CPU OS version prior to V8, this value must always be set to Retain Last Value.</b><br><br>Adopt Initial Data      Input variables are reset to their initial values.<br>Retain Last Value      The input variables retains the last value. |
| Enable TCP Gateway                                     | If the TCP Modbus gateway is enabled, at least one Modbus RS485 interface must be configured.   |
| TCP Server Port  | Standard: 502<br>Additional TCP ports may also be configured. Observe the port assignment provided by the ICANN ( <i>Internet Corporation for Assigned Names and Numbers</i> ).   |
| Maximum Number of TCP Connections Operating as Server. | Maximum number of TCP connections opened simultaneously and operating as server.<br><br>Range of values: 1...64<br>Default value: 5   |

|                             |  |
|-----------------------------|--|
| Enable UDP Gateway          | If the UDP Modbus gateway is enabled, at least one Modbus RS485 interface must be configured.  |
| UDP Port                    | Standard: 502<br>Additional UDP ports may also be configured. Observe the port assignment provided by the ICANN (Internet Corporation for Assigned Names and Numbers).   |
| Maximum length of the queue | Length of the gateway queue for other masters' request telegrams that have not been answered yet. This option is only taken into account if a gateway has been activated.<br><br>Range of values: 1...20<br>Default value: 3 |

Table 174: General Properties of the Modbus Master

### CPU/COM

The default values of the parameters provide the fastest possible data exchange of Modbus data between the COM module (COM) and the processor module (CPU) within the HIMax/HIMatrix controller.

These parameters should only be changed if it is necessary to reduce the COM or CPU load for an application, and the process allows this change.



Only experienced programmers should modify the parameters.

Increasing the COM and CPU refresh rate means that the effective refresh rate of the Modbus data is also increased. The system time requirements must be verified.

| Element                        | Description   |
|--------------------------------|---|
| Process Data Refresh Rate [ms] | Refresh rate in milliseconds at which the COM and CPU exchange protocol data.<br><br>If the <i>Process Data Refresh Rate [ms]</i> is zero or lower than the cycle time for the controller, data is exchanged as fast as possible.<br><br>Range of values: 0...(2 <sup>31</sup> -1)<br>Default value: 0  |
| Force Process Data Consistency | Activated:<br>Transfer of all protocol data from the CPU to the COM within a CPU cycle.<br><br>Deactivated:<br>Transfer of all protocol data from the CPU to the COM, distributed over multiple CPU cycles, each with 1100 byte per data direction.<br>This can also allow lowering the cycle time of the controller.<br><br>Default value: activated |

Table 175: Parameters of COM/CPU

## 7.2.4 Modbus Function Codes of the Master

Individual variables or multiple consecutive variables can be written or read with the Modbus function codes (request telegrams).

### To create a new request telegram for a TCP/UDP slave

1. In the structure tree, open **Resource, Protocols, Modbus Master, Ethernet Slaves**, and then select a **TCP/UDP Slave**.
2. Right-click **TCP/UDP slaves**, then click **New**.
3. In the **New Object** dialog box, click a **Request Telegram**.

### To create a new request telegram for a gateway slave

1. In the structure tree, open **Resource, Protocols, Modbus Master, Modbus Gateway**, , and then click a **gateway slave**.
2. Right-click **Gateway Slave**, then click **New**.
3. In the **New Object** dialog box, click a **Request Telegram**.

### To create a new request telegram for a RS485 Modbus slave

1. In the structure tree, open **Resource, Protocols, Modbus Master, Serial Modbus**, , and then click a **Modbus slave**.
2. Right-click **Modbus Slave**, then click **New**.
3. In the **New Object** dialog box, click a **Request Telegram**.

## 7.2.4.1 Modbus Standard Function Codes

The HIMA Modbus master supports the following Modbus standard function codes:

| Element                      | Code | Type | Description  |
|------------------------------|------|------|--|
| READ COILS                   | 01   | BOOL | Read multiple variables (BOOL) from the slave.                       |
| READ DISCRETE INPUTS         | 02   | BOOL | Read multiple variables (BOOL) from the slave.                       |
| READ HOLDING REGISTERS       | 03   | WORD | Read multiple variables of any type from the slave.                  |
| READ INPUT REGISTERS         | 04   | WORD | Read multiple variables of any type from the slave.                  |
| WRITE SINGLE COIL            | 05   | BOOL | Write one single signal (BOOL) in the slave.                         |
| WRITE SINGLE REGISTER        | 06   | WORD | Write one single signal (WORD) in the slave.                         |
| WRITE MULTIPLE COILS         | 15   | BOOL | Write multiple variables (BOOL) in the slave.                        |
| WRITE MULTIPLE REGISTERS     | 16   | WORD | Write multiple variables of any type in the slave.                   |
| READ WRITE HOLDING REGISTERS | 23   | WORD | Write and read multiple variables of any type in and from the slave. |

Table 176: Modbus Function Codes



For more information on Modbus, refer to the *Modbus Application Protocol Specification* at [www.modbus.org](http://www.modbus.org)

### 7.2.4.2 HIMA-Specific Function Codes

HIMA-specific function codes correspond to the standard Modbus function codes. The two differences are the maximum permissible process data length of 1100 bytes and the format of the request and response headers.

| Element                                | Code          | Type | Description  |
|--|---------------|------|--|
| Read Coils Extended                    | 100<br>(0x64) | BOOL | Correspond to function code 01.<br>Read multiple variables (BOOL) from the slave's import or export <sup>1)</sup> area.<br>Maximum length of the process data: 1100 bytes.   |
| Read Discrete Inputs Extended          | 101<br>(0x65) | BOOL | Correspond to function code 02.<br>Read multiple variables (BOOL) from the slave's export area.<br>Maximum length of the process data: 1100 bytes.   |
| Read Holding Registers Extended        | 102<br>(0x66) | WORD | Correspond to function code 03.<br>Read multiple variables of any type from the slave's import or export <sup>1)</sup> area.<br>Maximum length of the process data: 1100 bytes.  |
| Read Input Registers Extended          | 103<br>(0x67) | WORD | Correspond to function code 04.<br>Read multiple variables of any type from the slave's export area.<br>Maximum length of the process data: 1100 bytes.  |
| Write Multiple Coils Extended          | 104<br>(0x68) | BOOL | Correspond to function code 15.<br>Write multiple variables (BOOL) in the slave's import area.<br>Maximum length of the process data: 1100 bytes.  |
| Write Multiple Registers Extended      | 105<br>(0x69) | WORD | Correspond to function code 16.<br>Write multiple variables of any type in the slave's import area.<br>Maximum length of the process data: 1100 bytes.   |
| Read/Write Multiple Registers Extended | 106<br>(0x6A) | WORD | Correspond to function code 23.<br>Write and read multiple variables of any type in and from the slave's import or export area.<br>Maximum length of the process data:<br>1100 bytes (request telegram from the master Modbus master).<br>1100 bytes (response to the master). |

### 7.2.4.3 Format of Request and Response Header

The request and response header of the HIMA-specific Modbus function codes are structured as follows:

| Code          | Request   | Response   |
|---------------|---|--|
| 100<br>(0x64) | 1 byte function code 0x64<br>2 bytes start address<br>2 bytes number of coils 1...8800(0x2260)  | 1 byte function code 0x64<br>2 bytes number of bytes = N<br>N bytes coil data<br>(8 coils are packed in one byte)                      |
| 101<br>(0x65) | 1 byte function code 0x65<br>2 bytes start address<br>2 bytes number of discrete inputs 1...8800<br>(0x2260)  | 1 byte function code 0x65<br>2 bytes number of bytes = N<br>N bytes discrete inputs data<br>(8 discrete inputs are packed in one byte) |
| 102<br>(0x66) | 1 byte function code 0x66<br>2 bytes start address<br>2 bytes number of registers 1...550 (0x226)   | 1 byte function code 0x66<br>2 bytes number of bytes = N<br>N bytes register data  |
| 103<br>(0x67) | 1 byte function code 0x67<br>2 bytes start address<br>2 bytes number of registers 1...550 (0x226)   | 1 byte function code 0x67<br>2 bytes number of bytes = N<br>N bytes register data  |
| 104<br>(0x68) | 1 byte function code 0x68<br>2 bytes start address<br>2 bytes number of coils 1...8800(0x2260)<br>2 bytes number of bytes = N<br>N bytes coil data  | 1 byte function code 0x68<br>2 bytes start address<br>2 bytes number of coils<br>1...8800(0x2260)                                      |
| 105<br>(0x69) | 1 byte function code 0x69<br>2 bytes start address<br>2 bytes number of registers 1...550 (0x226)<br>2 bytes number of bytes = N<br>N bytes register data   | 1 byte function code 0x69<br>2 bytes start address<br>2 bytes number of registers<br>1...550 (0x226)                                   |
| 106<br>(0x6A) | 1 byte function code 0x6a<br>2 bytes read start address<br>2 bytes number of read registers 1...550<br>(0x226)<br>2 bytes write start address<br>2 bytes number of write registers 1...550<br>(0x226)<br>2 bytes number of write bytes = N<br>N bytes register data | 1 byte function code 0x6a<br>2 bytes number of bytes = N<br>N bytes register data  |

#### 7.2.4.4 Read Request Telegrams

The read function codes are used to read variables from the slave.

In addition to the Modbus function, a Modbus master's request telegram also contains the start address for the read/write area.

To read variables, the Modbus master sends a *Read Request Telegram* to the Modbus slave.

The Modbus slave responds to the Modbus master sending back a response telegram with the variables required.

##### To configure a read request telegram

1. In the structure tree, click the **Request Telegram** to be configured.
2. Right-click the **request telegram**, then click **Edit**.
3. Select the global variable that should be used as Modbus receive variable and drag it from the **Object Panel** onto anywhere in the **Input Signals** area.
4. Repeat these steps for every further Modbus receive variable.
5. Right-click anywhere in the **Inputs Signals** area, and then click **New Offsets** to renumber the variable offsets.

**The following Read Request Telegrams are available:**

##### Read Coils (01) and Extended (100)

Read multiple variables (BOOL) from the slave.

| Element                        | Description                             |
|--------------------------------|---|
| Type                           | Modbus Function Read Coils              |
| Name                           | Any unique name for the Modbus function |
| Description                    | Description for the Modbus function     |
| Start address of the read area | 0...65535                               |

Table 177: Request Telegram Read Coils

##### Read Discrete Inputs (02) and Extended (101)

Read multiple variables (BOOL) from the slave.

| Element                        | Description                             |
|--------------------------------|---|
| Type                           | Modbus Function Read Discrete Inputs    |
| Name                           | Any unique name for the Modbus function |
| Description                    | Description for the Modbus function     |
| Start address of the read area | 0...65535                               |

Table 178: Request Telegram Read Discrete Inputs

##### Read Holding Registers (03) and Extended (102)

Read multiple variables of any type from the slave.

| Element                        | Description                             |
|--------------------------------|---|
| Type                           | Modbus Function Read Holding Registers  |
| Name                           | Any unique name for the Modbus function |
| Description                    | Description for the Modbus function     |
| Start address of the read area | 0...65535                               |

Table 179: Request Telegram Read Holding Registers

### Read Input Registers (04) and Extended (103)

Read multiple variables of any type from the slave.

| Element                        | Description                             |
|--------------------------------|---|
| Type                           | Modbus Function Read Input Registers    |
| Name                           | Any unique name for the Modbus function |
| Description                    | Description for the Modbus function     |
| Start address of the read area | 0...65535                               |

Table 180: Request Telegram Read Input Registers

#### 7.2.4.5 Read/Write Request Telegram

For reading and writing variables, the Modbus master sends a *Read/Write Request Telegram* to the Modbus slave.

First, the Modbus master writes the write variables into the defined import area of the Modbus slave.

Afterwards, the Modbus master reads the read signals from the defined export area of the Modbus slave.



In the Read/Write Request Telegram, the Write and Read functions are independent of one another.

However, the *Read/Write Request Telegram* is often used such that the variables written by the Modbus master are read back. This ensures that the transferred variables were written correctly.

#### To configure a read/write request telegram

1. In the structure tree, click the **Request Telegram** to be configured.
2. Right-click the **request telegram**, then click **Edit**.

#### To configure the read variables

1. In the **Object Panel**, select the global variable that should be connected to one new Modbus receive variable and drag it onto the **Global Variable** column of the Modbus receive variable.
2. Repeat these steps for every further Modbus receive variable.
3. Right-click anywhere in the **Inputs Signals** area, and then click **New Offsets** to renumber the variable offsets.

#### To configure the write variables

1. In the **Object Panel**, select the global variable that should be connected to one new Modbus send variable and drag it onto the **Global Variable** column of the Modbus send variable.
2. Repeat these steps for every further Modbus send variable.
3. Right-click anywhere in the **Outputs Signals** area, and then click **New Offsets** to renumber the variable offsets..

### Read Write Holding Register (23) and Extended (106)

Write and read multiple variables of any type in and from the slave's import area.

| Element                         | Description   |
|---------------------------------|---|
| Type                            | Modbus function <i>Read Write Holding Registers</i> |
| Name                            | Any unique name for the Modbus function             |
| Description                     | Description for the Modbus function                 |
| Start address of the read area  | 0...65535   |
| Start address of the write area | 0...65535   |

Table 181: Read Write Holding Register

#### 7.2.4.6 Write Request Telegram

Using the write function codes, variables may only be written in a slave's import area.

In addition to the Modbus function, a Modbus master's request telegram also contains the start address for the read/write area.

To write variables, the Modbus master sends a *Write Request Telegram* to the Modbus slave.

The Modbus slave writes the received variables into its import area.

The variables that a Modbus master writes to a Modbus slave must be entered in the *Variable Connections* dialog box for a *write request telegram*.

#### To configure a write request telegram

1. In the structure tree, click the **Request Telegram** to be configured.
2. Right-click the **request telegram**, then click **Edit**.
3. Select the global variable that should be used as Modbus send variable and drag it from the **Object Panel** onto anywhere in the **Send Signals** area.
4. Repeat these steps for every further Modbus send variable.
5. Right-click anywhere in the **Send Signals** area, and then click **New Offsets** to renumber the variable offsets.

#### The following *Write Request telegrams* are available:

### Write Multiple Coils (15) and Extended (104)

Write multiple variables (BOOL) in the slave's import area.

| Element                         | Description                                 |
|---------------------------------|---|
| Type                            | Modbus function <i>Write Multiple Coils</i> |
| Name                            | Any unique name for the Modbus function     |
| Description                     | Description for the Modbus function         |
| Start address of the write area | 0...65535                                   |

Table 182: Request Telegram Write Multiple Coils

**Write Multiple Registers (16) and Extended (105)**

Write multiple variables of any type in the slave's import area.

| Element                         | Description                                     |
|---------------------------------|---|
| Type                            | Modbus function <i>Write Multiple Registers</i> |
| Name                            | Any unique name for the Modbus function         |
| Description                     | Description for the Modbus function             |
| Start address of the write area | 0...65535                                       |

Table 183: Request Telegram Write Multiple Registers

**Write Single Coil (05)**

Write one single variable (BOOL) in the slave's import area.

| Element                         | Description                              |
|---------------------------------|--|
| Type                            | Modbus function <i>Write Single Coil</i> |
| Name                            | Any unique name for the Modbus function  |
| Description                     | Description for the Modbus function      |
| Start address of the write area | 0...65535                                |

Table 184: Request Telegram Write Single Coil (05)

**Write Single Register (06)**

Write one single variable (WORD) in the slave's import area.

| Element                         | Description                                  |
|---------------------------------|--|
| Type                            | Modbus function <i>Write Single Register</i> |
| Name                            | Any unique name for the Modbus function      |
| Description                     | Description for the Modbus function          |
| Start address of the write area | 0...65535                                    |

Table 185: Request Telegram Write Single Register

## 7.2.5 Ethernet Slaves (TCP/UDP Slaves)

The Modbus masters can communicate with up to 64 TCP/IP and 247 UDP/IP slaves.

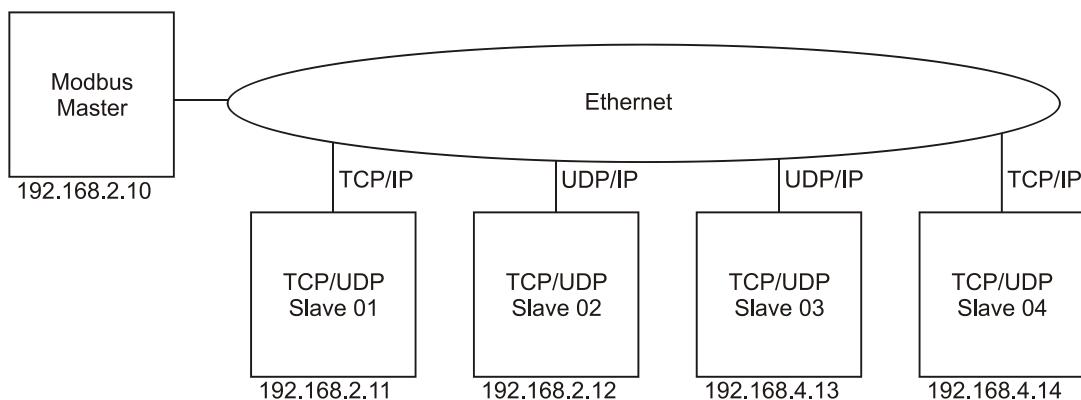


Figure 59: Modbus Network

### To create a new connection to a TCP/UDP slave within the Modbus master

1. In the structure tree, open **Resource, Protocols, Modbus Master, Ethernet Slaves**.
2. Right-click **Ethernet Slaves**, then click **New**.
3. Select **TCP/UDP Slaves** from the list and click **OK** to confirm.
4. To configure the TCP/UDP slave in the Modbus master:  
Click **Edit** to assign the system variables, see Chapter 7.2.5.1.  
Click **Properties** to configure the properties, see Chapter 7.2.5.2.



If the TCP/UDP slaves and the Modbus master are located in different subnets, the routing table must contain the corresponding user-defined routes.

The telegrams that the HIMax/HIMatrix Modbus TCP master sends to the Modbus TCP slave always include the IP address as well as a Modbus slave address (Unit Identifier) which is always FF (255).

### 7.2.5.1 System Variables for TCP/UDP Slaves

The *System Variables* tab contains system variables that are required to control the TCP/UDP slave and evaluate its state from within the user program.

The following status variables can be used to evaluate the TCP/UDP slave status from within the user program:

| Element                         | Description  |
|---------------------------------|--|
| Modbus Slave Activation Control | <p>The user program activates or deactivates the TCP/UDP slave using this function.<br/>           0: Activate<br/>           1: Deactivate<br/>           (Edge triggered!<br/>           The Modbus slave can be activated using the PADT even if the <i>Modbus Slave Activation Control</i> is 1).</p>  |
| Modbus Slave Error              | <p>Error Code<br/>           The error codes 0x01...0x0b correspond to the Exception Codes of the Modbus protocol specification.<br/>           0x00: No error<br/> <br/>           Exception Codes:<br/>           0x01: Invalid function code<br/>           0x02: Invalid addressing<br/>           0x03: Invalid data<br/>           0x04: (not used)<br/>           0x05: (not used)<br/>           0x06: Device busy (only gateway)<br/>           0x08: (not used)<br/>           0x0a: (not used)<br/>           0x0b: No Response from Slave (only gateway)<br/> <br/>           HIMA-specific codes:<br/>           0x10: Defective frame received<br/>           0x11: Frame with wrong transaction ID received<br/>           0x12: Unexpected response received<br/>           0x13: Response about wrong connection received<br/>           0x14: Wrong response to a write request<br/> <br/>           0xff: Slave Timeout</p> |
| Modbus Slave State              | Connection status of the TCP/UDP slave:<br>0: Disabled<br>1: Not connected<br>2: Connected   |

Table 186: System Variables for TCP/UDP Slaves

### 7.2.5.2 TCP/UDP Slave Properties

To configure the connection to the TCP/UDP slave, the following parameters must be set in the Modbus master.

| Element                              | Description  |
|--------------------------------------|--|
| Type                                 | TCP/UDP slave  |
| Name                                 | Any unique name for the TCP/UDP slave  |
| Description                          | Any unique description for the TCP/UDP slave   |
| Master-Slave Data Exchange [ms]      | Time interval for exchanging data with this slave 1 to $(2^{31}-1)$ . If the <i>Maximal Number of Retries</i> was exceeded and the slave could not be reached, the value for <i>Master-Slave Data Exchange</i> is set four times higher.                             |
| TCP connection only on demand        | If the type of transmission protocol is TCP, this parameter can be used to define if the connection to the slave should be automatically closed whenever data is exchanged:<br>TRUE: Close the connection.<br>FALSE: Do not close connection<br>Default value: FALSE |
| Receive Timeout [ms]                 | Receive timeout [ms] for this slave. Once this time period has expired, a resend is attempted.   |
| IP Address                           | IP address of the TCP/UDP slave  |
| Port                                 | Standard: 502<br>Additional TCP/UDP ports may also be configured. Observe the port assignment provided by the ICANN (Internet Corporation for Assigned Names and Numbers).   |
| Type of communication<br>IP protocol | TCP or UDP<br>Default value: TCP   |
| Maximum Number of Resends            | Maximal number of send retries if the slave does not respond.<br>The number of resends can be freely set (0...65535).<br>With TCP/IP, the value is always set to 0 and cannot be changed.<br>HIMA recommends setting a value between 0 and 8.                        |

Table 187: Configuration Parameters

### 7.2.6 Modbus Gateway (TCP/UDP Gateway)

The Modbus master RTU license is required to enable the Modbus master to operate as Modbus gateway. In this mode, master requests that the gateway receives via Ethernet are forwarded to the RS485 und Ethernet slaves connected to the gateway. Accordingly, the slave's responses are forwarded to the Modbus master through the gateway.

Up to 121 serial Modbus slaves can be addressed per serial interface.

The slave's address ranges from 1 to 247. Even if only the Modbus gateway is used, a Modbus master license is required for Modbus master 2 (Modbus gateway).

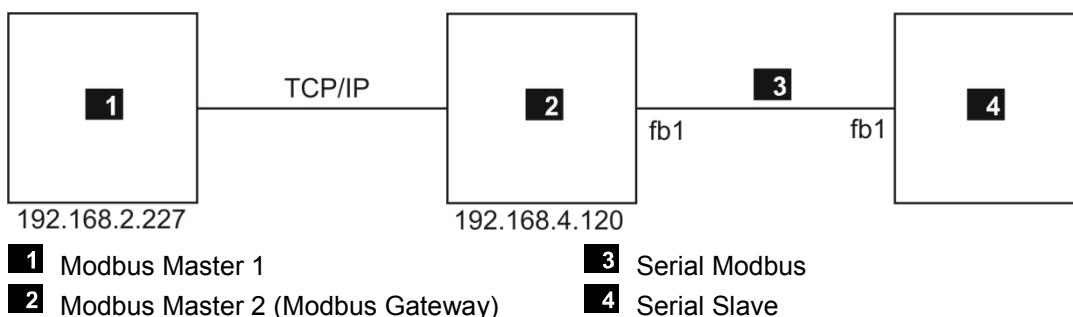


Figure 60: Modbus Gateway

- 
- i** If the Modbus gateway and the Modbus master are located in different subnets, the routing table must contain the corresponding user-defined routes.
- 

#### Modbus Master 1

##### To create the connection to the Modbus slave within Modbus master 1

1. In the structure tree, open **Resource, Protocols, Modbus Master**
2. Right-click **Modbus Master**, then click **New**.
3. Select **Modbus Gateway** from the list and click **OK** to confirm.
4. To configure the Modbus gateway in Modbus Master 1:  
click **Properties** to configure the properties, see Chapter 7.2.7.3.  
In the properties, enter the **IP Address** for Modbus master 2 (Modbus gateway).

##### To create the connection to the gateway slave within Modbus master 1

In Modbus master 1, the serial slave must be created as gateway slave.

1. In the structure tree, open **Resource, Protocols, Modbus Master, Modbus Gateway**.
2. Right-click **Modbus Gateway**, then click **New**.
3. Select **Gateway Slave** from the list and click **OK** to confirm.
4. To configure the gateway slave in Modbus master 1:  
Click **Edit** to assign the system variables, see Chapter 7.2.6.2.  
Click **Properties** to configure the properties, see Chapter 7.2.6.3.  
In the slave's properties, enter the **serial address** for the gateway slave.

**To define input and output variables to the serial slave in Modbus master 1**

1. Right-click **Gateway Slave**, then click **New**.
2. Select the required **request telegrams** from the list.
3. Right-click the corresponding **request telegram**, then click **Edit**. Enter the input or output variables in the *Process Variables* tab.

**Modbus Master 2 (Modbus Gateway):**

The gateway function must be enabled in the properties of Modbus master 02. The gateway slaves configured in master 01 is connected to the serial slaves.

**To activate the gateway function in Modbus master 2**

1. In the structure tree, open **Resource, Protocols, Modbus Master**
2. Right-click **Modbus Master**, then click **Properties**.
3. Activate the **Enable TCP Gateway** parameter to allow the Modbus master to additionally operate as TCP gateway.
4. Activate the **Enable UDP Gateway** parameter to allow the Modbus master to additionally operate as UDP gateway.

**To configure the serial Modbus in Modbus master 2**

1. In the structure tree, open **Resource, Protocols, Modbus Master**
2. Right-click **Modbus Master**, then click **New**.
3. Select **Serial Modbus** from the list and click **OK** to confirm.
4. Select **Properties** to configure the **serial Modbus**, then enter the interface, the baud rate, etc.

**To configure the connection to the serial slave in Modbus master 2**

1. In the structure tree, open **Resource, Protocols, Modbus Master, Serial Modbus**.
2. Right-click **Serial Modbus**, then click **New**.
3. Select **Modbus Slave** from the list and click **OK** to confirm.
4. Select **Properties** to configure the **Modbus Slave**, then enter the **Slave Address** for the serial slave.

**Serial Slave****To configure the serial Modbus slave**

1. In the structure tree, open **Resource, Protocols, Modbus Slave**
2. Right-click **Modbus Slave**, then click **Edit**.
3. Select **Properties** to configure the **Modbus Slave**, then enter the **Slave Address** for the serial slave.

### 7.2.6.1 Gateway Properties

The Modbus gateway allows the Modbus master to communicate with its Modbus slave.

To configure the connection to the Modbus gateway, the following parameters must be set in the Modbus master.

| Element                   | Description  |
|---------------------------|--|
| Type                      | Modbus Gateway   |
| Name                      | Any unique name for the gateway  |
| Description               | Any unique description for the TCP/UDP slave   |
| Communication IP Protocol | TCP or UDP<br>Default value: TCP   |
| IP address                | Gateway's IP address that the Modbus master should use to communicate with its Modbus slave.<br>Default value: (0.0.0.0) |
| Port                      | Default value: 502   |

Table 188: Connection Parameters for the Modbus Gateway

### 7.2.6.2 System Variables for the Gateway Slave

The editor contains three status variables:

| Element                         | Description  |
|---------------------------------|--|
| Modbus Slave Activation Control | Using this function, the user program can enable or disable the gateway slave.<br>0: Activate<br>1: Deactivate<br>(Edge triggered!<br>The Modbus slave can be activated using the PADT even if the <i>Modbus Slave Activation Control</i> is 1). |
| Modbus Slave Error              | Parameters: the same as for TCP/UDP slave, see Chapter7.2.5.1.   |
| Modbus Slave State              | Connection status of the gateway slave:<br>0: Disabled<br>1: Not connected<br>2: Connected   |

Table 189: Status Variables for the Gateway Slave

### 7.2.6.3 Gateway Slave Properties

To configure the connection to the gateway slave, the following parameters must be set in the Modbus master.

| Element   | Description                                  |
|---|--|
| Type  | Gateway Slave                                |
| Name  | Any unique name for the gateway slave        |
| Description   | Any unique description for the gateway slave |
| Slave Address   | 1...247                                      |
| The remaining parameters are the same as for TCP/UDP slave, see Chapter7.2.5.2. |  |

Table 190: Connection Parameters for the Gateway Slave

### 7.2.7 Serial Modbus

The Modbus master can communicate with up to 247 serial slaves. According to the standard, a total of three repeaters may be used such that a maximum of 121 stations are possible per serial interface on a master.

- i** For more information on the pin assignment of the X-COM module's D-sub connectors (fb1, fb2), refer to Chapter 3.6.

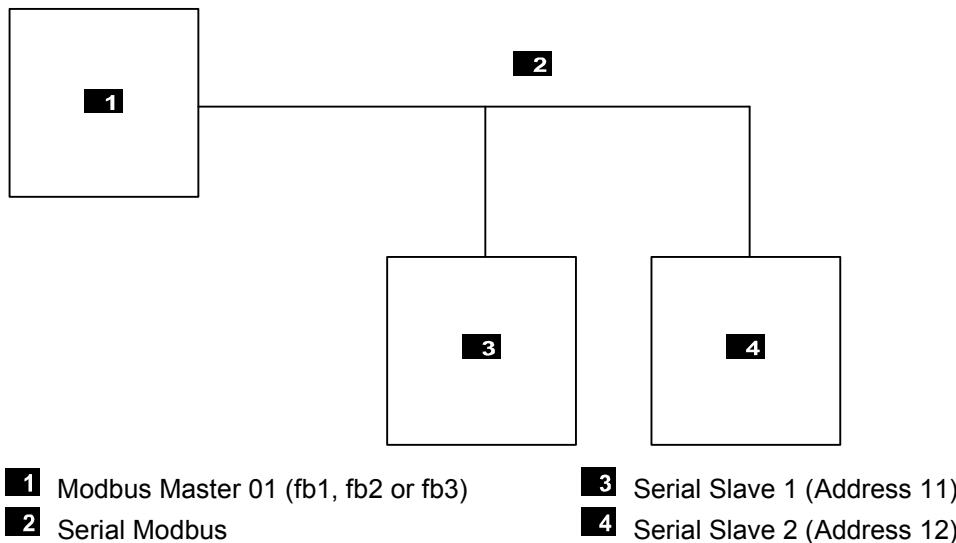


Figure 61: Serial Modbus

The HIMA Modbus master supports data transfer in RTU format (Remote Terminal Unit).

The RTU telegram frame starts and ends with the idle characters set by the user (default value: 5 idle characters).

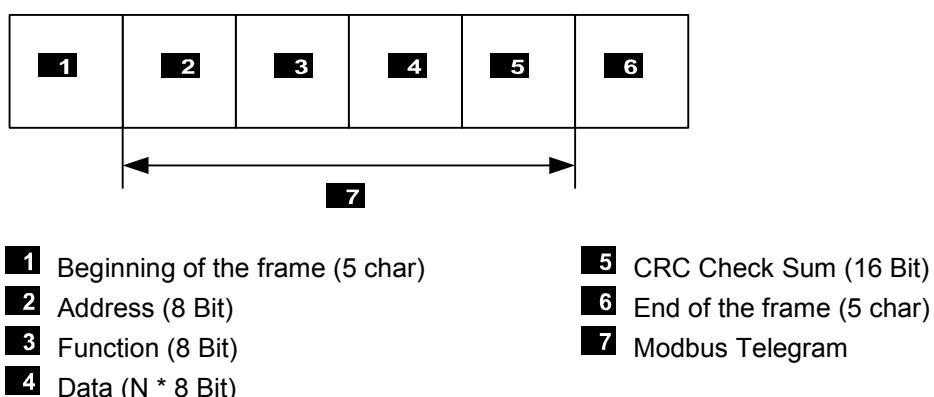


Figure 62: Modbus Telegram

**To create a serial Modbus within the Modbus master**

1. In the structure tree, open **Resource**, **Protocols**, **Modbus Master**, **Serial Modbus**.
2. Right-click **Serial Modbus**, then click **New**.
3. Select **Modbus Slave** from the list and click **OK** to confirm.
4. To configure the Modbus slave in the Modbus master:  
 Click **Edit** to assign the system variables, see Chapter 7.2.7.2.  
 Click **Properties** to configure the properties, see Chapter 7.2.7.3.

#### 7.2.7.1 Serial Modbus Properties

To configure the serial Modbus master, the following parameters must be set.

| Element              | Description  |
|----------------------|--|
| Type                 | Serial Modbus  |
| Name                 | The serial Modbus name may be selected by the user   |
| Description          | Any unique description for the serial Modbus   |
| Interface            | Fieldbus interface that should be used for the Modbus master (fb1, fb2).   |
| Baud rate [bps]      | Possible value for transfer rate for RS485:<br>300 bit/s<br>600 bit/s<br>1200 bit/s<br>2400 bit/s<br>4800 bit/s<br>9600 bit/s<br>19200 bit/s<br>38400 bit/s<br>57600 bit/s (maximum baud rate for HIMax V4 and beyond)<br>62500 bit/s (HIMatrix only)<br>76800 bit/s (HIMatrix only)<br>115000 bit/s (HIMatrix only) |
| Parity               | None<br>Odd<br>Even<br>Default value: Even   |
| Stop Bits            | Standard (adapts the number of stop bits to the parity:<br>with parity = 1 stop bit, no parity = 2 stop bits)<br>One stop bit<br>Two stop bits<br>Default value: Default   |
| Number of Idle Chars | Number of idle characters at the start and the end of a RTU telegram frame.<br>Range of values: 0...65535<br>Default value: 5 characters   |

Table 191: Parameters for the Serial Modbus Master

### 7.2.7.2 System Variables for the Modbus Slave

The editor contains three status variables (system variables).

| Element                         | Description  |
|---------------------------------|--|
| Modbus Slave Activation Control | Activate or deactivate the Modbus slave in the user program.<br>0: Activate<br>1: Deactivate<br>(Edge triggered!<br>The Modbus slave can be activated using the PADT even if the <i>Modbus Slave Activation Control</i> is 1). |
| Modbus Slave Error              | Parameters: the same as for TCP/UDP slave, see Chapter 7.2.5.2.  |
| Modbus Slave State              | Connection status of the Modbus slave:<br>0: Disabled<br>1: Not connected<br>2: Connected  |

Table 192: System Variables in the Modbus Slave

### 7.2.7.3 Modbus Slave Properties

To configure the connection to the serial slave, the following parameters must be set in the Modbus master.

| Element  | Description                                       |
|--|---|
| Type   | Modbus slave                                      |
| Name   | The Modbus slave name may be selected by the user |
| Description  | Any unique description for the Modbus slave       |
| Slave Address  | 1...247   |
| The remaining parameters are the same as for TCP/UDP slave, see Chapter 7.2.5.2. |   |

Table 193: Connection Parameters for the Modbus Master



In the serial Modbus slave, the Receive Timeout depends on the transfer rate which has been set.

If the baud rate is 19200 [bit/s] or higher, the default value for Receive Timeout may be used. If the baud rate is lower than 19200 [bit/s], the value for Receive Timeout must be increased.

### 7.2.8 Control Panel (Modbus Master)

The Control Panel can be used to verify and control the settings for the Modbus master. Details about the master's current state (e.g., master state, etc.) are displayed.

#### To open the Control Panel for monitoring the Modbus master

1. Right-click the **Hardware** structure tree node and select **Online** from the context menu.
2. In the **System Login** window, enter the access data to open the online view for the hardware.
3. Double-click **COM Module** and select the **Modbus Master** structure tree node.

### 7.2.8.1 Context Menu (Modbus Master)

The following commands can be chosen from the context menu for the selected Modbus master:

#### Offline

This command is used to stop the Modbus master.

#### Operate

This command is used to start the Modbus master.

#### Reset statistical data

Reset the statistical data (e.g., number of bus errors, min./max. cycle time etc.) to 0.

### 7.2.8.2 View Box (Modbus Master)

The view box displays the following values of the selected Modbus master.

| Element               | Description  |
|-----------------------|--|
| Name                  | Modbus master name   |
| Master State          | It indicates the current protocol state:<br>OPERATE<br>OFFLINE |
| Bus Error Count       | Counter for bus errors   |
| Disturbed Connections | Counter for disturbed connections                              |
| µP load (planned)     | See Chapter 7.2.3.2  |
| µP load (actual)      |  |

Table 194: View Box of the Modbus Master

### 7.2.9 Control Panel (Modbus Master->Slave)

The Control Panel is used to verify and activate/deactivate the settings for the communication partner. Details about the current status of the communication partner (e.g., slave state, etc.) are displayed.

#### To open Control Panel for monitoring the Modbus connection

1. Right-click the **Hardware** structure tree node and select **Online** from the context menu.
2. In the **System Login** window, enter the access data to open the online view for the hardware.
3. Double-click **COM Module** and select the **Modbus Master, Slave** in the structure tree.

### 7.2.10 FBx LED Function in the Modbus Master

The FBx LED of the corresponding fieldbus interface indicates the state of the Modbus protocol. The states of the FBx LED are specified in the following table:

| FBx LED  | Color  | Description   |
|----------|--------|---|
| OFF      | Yellow | The Modbus master protocol is not active!<br>This means that the controller is in STOP or no Modbus master is configured. |
| Blinking | Yellow | The Modbus master protocol is active and is exchanging data with the Modbus slave.  |

Table 195: The FBx LED

### 7.2.11 Function of the FAULT LED in the Modbus Master (HIMax only)

The FAULT LED of the corresponding fieldbus interface indicates a failure of the Modbus protocol. The states of the FAULT LED are specified in the following table:

| FAULT LED | Color | Description   |
|-----------|-------|---|
| OFF       | Red   | The Modbus master protocol is not disturbed.  |
| Blinking  | Red   | <p>The following events result in a malfunction.</p> <ul style="list-style-type: none"> <li>▪ Incorrect response or error message from the slave received</li> <li>▪ Timeout for one or more slaves</li> <li>▪ Calculating time budget exceeded</li> </ul> <p>If no faults occur for a period longer than 5 seconds, the state changes to "Protocol not disturbed".</p> |

Table 196: The FAULT FBx

## 7.3

### HIMA Modbus Slave

The HIMA Modbus slave can simultaneously use the serial interface (RS485) and the TCP/UDP (Ethernet) to operate various Modbus masters.

#### Equipment and System Requirements

| Element          | Description  |
|------------------|--|
| HIMA controller  | HIMax with COM module<br>HIMatrix: CPU OS version 7 and beyond and COM OS version 12 and beyond  |
| Processor module | The Ethernet interfaces on the processor module may not be used for Modbus TCP.  |
| COM module       | Ethernet 10/100BaseT<br>Pin assignment of the D-sub connectors FB1 and FB2<br>If Modbus RTU is used, each of the serial fieldbus interfaces (FB1 and FB2) used on the COM module must be equipped with a HIMA RS485 submodule.<br>For more information on the interface assignment, see Chapter 3.6. |
| Activation       | Each of the two Modbus slave functions must be enabled individually, see Chapter 3.4.<br>Modbus Slave RTU (RS485)<br>Modbus Slave TCP<br>Two licenses are required for the redundant Modbus slave, one for each X-COM module.  |

Table 197: Equipment and System Requirements for the HIMA Modbus Slave

#### Modbus Slave (Properties)

| Element                           | Description   |                   |            |               |                  |                             |                      |                      |                      |               |                      |
|-----------------------------------|---|-------------------|------------|---------------|------------------|-----------------------------|----------------------|----------------------|----------------------|---------------|----------------------|
| Modbus slave                      | One TCP Modbus slave and one RTU Modbus slave (RS485) can be configured for each COM module.  |                   |            |               |                  |                             |                      |                      |                      |               |                      |
| Redundancy                        | For HIMax only!<br>A maximum of 10 redundant Modbus slave communication module pairs can be operated in a HIMax system.<br>As long as the Modbus slave communication module pair operates redundantly, the same input and output data is exchanged with the Modbus master via both communication modules, see Chapter 7.3.3.  |                   |            |               |                  |                             |                      |                      |                      |               |                      |
| Number of Master Accesses         | RTU: Due to the RS485 transmission technology, only one Modbus master can access an installed RS485 interface<br>TCP: A maximum of 20 Modbus masters can access the slave.<br>TCP: Unlimited number of Modbus masters can access the slave.   |                   |            |               |                  |                             |                      |                      |                      |               |                      |
| Max. Size of Send Data            | See Table 2 Standard Protocols  |                   |            |               |                  |                             |                      |                      |                      |               |                      |
| Max. Size of Receive Data         |   |                   |            |               |                  |                             |                      |                      |                      |               |                      |
| Display format of the Modbus data | <p>The HIMax/HIMatrix controllers use the big endian format.<br/>Example: 32-bit data (e.g., DWORD, DINT):</p> <table border="1"> <thead> <tr> <th>32-bit data (hex)</th> <th>0x12345678</th> </tr> </thead> <tbody> <tr> <td>Memory offset</td> <td>0    1    2    3</td> </tr> <tr> <td>Big endian (HIMax/HIMatrix)</td> <td>12    34    56    78</td> </tr> <tr> <td>Middle endian (H51q)</td> <td>56    78    12    34</td> </tr> <tr> <td>Little endian</td> <td>78    56    34    12</td> </tr> </tbody> </table> | 32-bit data (hex) | 0x12345678 | Memory offset | 0    1    2    3 | Big endian (HIMax/HIMatrix) | 12    34    56    78 | Middle endian (H51q) | 56    78    12    34 | Little endian | 78    56    34    12 |
| 32-bit data (hex)                 | 0x12345678  |                   |            |               |                  |                             |                      |                      |                      |               |                      |
| Memory offset                     | 0    1    2    3  |                   |            |               |                  |                             |                      |                      |                      |               |                      |
| Big endian (HIMax/HIMatrix)       | 12    34    56    78  |                   |            |               |                  |                             |                      |                      |                      |               |                      |
| Middle endian (H51q)              | 56    78    12    34  |                   |            |               |                  |                             |                      |                      |                      |               |                      |
| Little endian                     | 78    56    34    12  |                   |            |               |                  |                             |                      |                      |                      |               |                      |

Table 198: Properties of the Modbus Slave

### 7.3.1 Configuring the Modbus TCP Slave

#### To create a new HIMA Modbus Slave

1. In the structure tree, open **Configuration, Resource, Protocols**.
2. Right-click **Protocols** and select **New, Modbus Slave Set** from the context menu to add a new Modbus slave set.
3. Select **Edit** from the context menu for the Modbus slave set, open the **Modbus Slave Set Properties**, and retain the default values.
4. Select the **Modbus slave** tab and perform the following actions:
  - Select **COM Module**
  - Activate **Enable TCP**
  - The remaining parameters retain the default values.



Chapter 7.2.1 provides an example of how to configure the connection between an HIMA Modbus TCP slave and an HIMA Modbus TCP master.

### 7.3.2 Configuring the Redundant Modbus TCP Slave

#### To create a redundant HIMA Modbus slave

1. In the structure tree, open **Configuration, Resource, Protocols, Modbus Slave Set**.
2. Select **Edit** from the context menu for the Modbus slave set, open the **Modbus Slave Set Properties**, and perform the following actions:
  - Activate **Set Redundant Operation**.
  - The **Modbus Slave Redundant** tab is automatically added.
3. Select the **Modbus Slave Redundant** tab and perform the following actions:
  - Select **COM Module**
  - Activate **Enable TCP**

The remaining parameters retain the default values.



The send and receive variables assigned in the Modbus slave set are valid for both Modbus slaves.

### 7.3.3 Rules for Redundant Modbus TCP Slaves

The redundant configuration of the HIMax system is recommended for operating the HIMax Modbus slave communication modules redundantly, see the System Manual (HI 801 001 E) for more details.

Otherwise, the consistent behavior of the Modbus slave communication module pairs towards their external partner (Modbus master) can no longer be ensured once the first fault has occurred within the HIMax system.

#### 7.3.3.1 Slots Allowed for the Redundant Modbus Slave COM Modules

To minimize the risk of potential collisions on the HIMax system bus, the system bus segments (1 to 3) on the base plate must be taken into account. For this reason, the redundant Modbus slave communication modules should only be inserted in the same segment of a base plate in the following slots:

| Segment | Slot  |
|---------|---|
| 1       | 3...6 (as long as no processor module has been planned) |
| 2       | 7...14  |
| 3       | 15...18   |

Table 199: Slots Allowed for the Redundant Modbus Slave COM Modules

#### 7.3.3.2 Redundant Modbus Slave COM Modules in Different Base Plates

A maximum of two redundant Modbus slave communication module pairs may be operated if their redundant Modbus slave communication modules are located in different base plates (0...15).

In this case, these redundant Modbus slave communication modules may only be located in adjacent base plates.

Furthermore, 8 additional Modbus slave communication module pairs may be operated on the same HIMax system in accordance with the rules specified in Chapter 7.3.3.1.

#### NOTE



**System malfunction possible!**

**Only use slots for redundant Modbus slave communication modules if the specified rules are observed!**

**Between an X-COM module and the X-CPU modules, a maximum additional delay of 50 µs is allowed (due to cable length, switches, etc).**

**Recommendation: Operate the X-COM modules as close as possible to the X-CPU modules (e.g., rack 0, rack 1)**

### 7.3.4 Menu Functions of the HIMA Modbus Slave Set

The Edit function of the context menu for the Modbus Slave Set is used to open the *Modbus Slave Set Properties* dialog box. The dialog box contains the following tabs:

#### 7.3.4.1 Modbus Slave Set Properties

The following parameters for the Modbus slave can be set in the *Modbus Slave Set Properties* tab.

| Element  | Description   |
|--|---|
| Name   | Name of the Modbus slave set  |
| Activate Max. $\mu$ P Budget                   | Activated:<br>Adopt the $\mu$ P budget limit from the <i>Max. <math>\mu</math>P Budget in [%]</i> field.<br><br>Deactivated:<br>Do not use the $\mu$ P budget limit for this protocol.  |
| Max. $\mu$ P budget in [%]                     | Maximum $\mu$ P load of the COM module that can be used for processing the protocols.<br>Range of values: 1...100%<br>Default value: 30%  |
| Set Redundant Operation                        | Activated: Redundant Operation<br>Deactivated: Mono Operation<br>Default value: Deactivated   |
| Max. Response Time [ms]                        | Time period after the reception of a request within which the Modbus slave may respond.<br><b>This value must be set to 0 in HIMatrix controllers.</b><br>Range of values: 0...-( $2^{31}$ -1) ms<br>Default value: 5000 ms (0 = No limitation)   |
| Area for reading function codes 1, 3, 100, 102 | This parameter defines from which data field the data should be read for function code 1, 3, 100, 102.<br>Range of values: <ul style="list-style-type: none"><li>▪ Import area</li><li>▪ Export area (compatible with 51q)</li></ul>  |
| Area for Reading Function Code 23, 106         | The user can specify the Modbus slave's area from which the function code 23 should be read.<br>Import area: The Master has read/write access to the slave's import area.<br>Export area: The master reads from the slave's export area and writes to the slave's import area.<br>Note: writing and reading take place within a single CPU cycle.<br>This means that the read data was provided during the <u>last</u> CPU cycle. |
| COM: Values at Connection Loss to Master       | If the connection of the communication module to the Modbus master is lost, the input variables are forwarded to the process module initialized or unchanged, depending on this parameter.<br>Adopt Initial Data      Input variables are reset to their initial values.<br>Retain Last Value      The input variables retains the last value.  |

|  |   |
|--|---|
| CPU: Values at Connection Loss to Master | If the connection of the processor module to the communication module is lost, the input variables are initialized or continue to be used unchanged in the process module, depending on this parameter.<br><br>For instance, if the communication module is removed when communication is running.<br><b>If a project created with a SILworX version prior to V3 should be converted, this value must be set to Retain Last Value to ensure that the CRC does not change.</b><br><b>For HIMatrix controllers with CPU OS version prior to V8, this value must always be set to Retain Last Value.</b><br>The same behavior as the COM to the master<br>See the setting of the COM:<br><i>Values at Connection Loss to Master</i> parameter<br>Retain Last Value<br>The input variables retains the last value.<br>Default value: The same behavior as the COM to the master |
| Alternative register / bit addressing    | Activated<br>Deactivate<br>d<br>Default value: Deactivated, see Chapter 7.3.11.   |
| Register Area Offset Bits Input          | Range of values: 0...65535<br>Default value: 0  |
| Register Area Offset Bits Output         | Range of values: 0...65535<br>Default value: 0  |
| Bit Area Offset Register Input           | Range of values: 0...65535<br>Default value: 0  |
| Bit Area Offset Register Output          | Range of values: 0...65535<br>Default value: 0  |
| Refresh Rate [ms]                        | Refresh rate in milliseconds at which the COM and CPU exchange protocol data.<br>If the <i>Refresh Rate</i> is zero or lower than the cycle time for the controller, data is exchanged as fast as possible.<br>Range of values: 0...(2 <sup>31</sup> -1)<br>Default value: 0  |
| Force Process Data Consistency           | Activated:<br>Transfer of all protocol data from the CPU to the COM within a CPU cycle.<br>Deactivate<br>d:<br>Transfer of all protocol data from the CPU to the COM, distributed over multiple CPU cycles, each with 1100 byte per data direction.<br>This can also allow lowering the cycle time of the controller.<br>Default value: activated   |

Table 200: Modbus Slave Properties Set Tab

#### 7.3.4.2 Register Variable

(Tab: Access)

The variables that the master addresses 16 bit by bit are entered in the Register Variables tab (function code 3, 4, 6, 16, 23, 102, 103, 105, 106).

#### 7.3.4.3 Bit Variables

(Bit and/or coil access)

The variables that the master addresses bit by bit are entered in the Bit Variables tab (function code 1, 2, 5, 15, 100, 101, 104).

### 7.3.5 Assigning Send/Receive Variables

All the variables that the Modbus slave receives from the Modbus master are entered in the **Inputs** tab.

#### To configure the send variables of the Modbus slave

1. In the structure tree, select the **Modbus Slave** that should be configured.
2. Right-click **Modbus Slave**, and then click **Edit**.
3. Select the **Register Variables** or **Bit Variables** tab.
4. Drag one **variable** from the *Object Panel* onto the *Register Outputs* area.
5. Repeat these steps for every further variable that should be defined as send variable for the Modbus slave.
6. Right-click the *Register Outputs* area, and then click **New Offsets**.

#### To configure the receive variables of the Modbus slave

1. In the structure tree, select the **Modbus Slave** that should be configured.
2. Right-click **Modbus Slave**, and then click **Edit**.
3. Select the **Register Variables** or **Bit Variables** tab.
4. Drag one **variable** from the *Object Panel* onto the *Register Inputs* area.
5. Repeat these steps for every further variable that should be defined as receive variable for the Modbus slave.
6. Right-click the *Register Inputs* area, and then click **New Offsets**.

### 7.3.6 Modbus Slave Set System Variables

The **Modbus Slave Set System Variables** includes the following system variables.

| Element         | Description   |
|-----------------|---|
| Redundant State | This parameter describes the redundancy state of the redundant Modbus slave communication module pair.<br>0: Redundant Modbus Slave COM Modules active<br>1: First Modbus Slave COM Module not active<br>2: Redundant Modbus slave COM module not active<br>3: Both Modbus Slave COM Modules not active |

Table 201: Modbus Slave Set Tab

### 7.3.7 Modbus Slave and Modbus Slave Redundant

The **Modbus Slave** tab contains the two tabs **Properties** and **System Variables**.



The pin assignment of the D-sub connectors (fb1, fb2) is described in the manuals of the corresponding HIMax modules and HIMatrix controllers.

### Properties

| Element                                      | Description  |
|--|--|
| Module                                       | Selection of the COM module within which the protocol is processed.  |
| Master Monitoring Time [ms]                  | <p>Time period after the reception of a request within which the Modbus slave must respond.</p> <p>If the connection of the connection module to the Modbus master is lost, the input variables are forwarded initialized or unchanged to the process module, depending on the parameter X-COM: <i>Values at Connection Loss to Master</i>.</p> <p>Note: If multiple master of the same type are used (Eth or RS485), the master monitoring function can not distinguish between masters. It may therefore occur, that a master fails without being noticed.</p> <p>See Chapter 7.3.4.1.<br/> Range of values 1...(2<sup>31</sup>-1) [ms]<br/> Default value: 0 ms (no limitation)</p> |
| <b>Parameters for the Ethernet interface</b> |  |
| Enable the TCP/IP connection                 | <p>Activated      The TCP/IP connection is enabled<br/> Deactivated      TCP/IP connection disabled<br/> Default value: Deactivated</p>  |
| TCP Port                                     | Default value: 502   |
| Maximum number of TCP Connections            | <p>Maximum number of TCP connections opened simultaneously and operating as server.<br/> Range of values: 1...20<br/> Default value: 3</p>   |
| UDP Enable                                   | <p>Activated      UDP/IP connection is enabled<br/> Deactivate      UDP/IP connection disabled<br/> Default value: Deactivated</p>   |
| UDP Port                                     | Default value: 502   |
| <b>Parameters for the serial interface</b>   |  |
| Name   | Name of the serial interface   |
| Interface                                    | Selection of the fieldbus interfaces that are available and can be used for the Modbus slave (none, fb1 and/or fb2).   |
| Slave Address                                | <p>Slave bus address<br/> Range of values: 1...247</p>   |
| Baud rate [bps]                              | <p>Possible value for transfer rate for RS485:</p> <ul style="list-style-type: none"> <li>300 bit/s</li> <li>600 bit/s</li> <li>1200 bit/s</li> <li>2400 bit/s</li> <li>4800 bit/s</li> <li>9600 bit/s</li> <li>19200 bit/s</li> <li>38400 bit/s</li> <li>57600 bit/s (maximum baud rate for HIMax V4 and beyond)</li> <li>62500 bit/s (HIMatrix only)</li> <li>76800 bit/s (HIMatrix only)</li> <li>115000 bit/s (HIMatrix only)</li> </ul>   |

|                      |  |
|----------------------|--|
| Parity               | Range of values: <ul style="list-style-type: none"><li>▪ none</li><li>▪ Odd</li><li>▪ Even</li></ul> Default value: Even   |
| Stop Bits            | Range of values:<br>Standard (adapts the number of stop bits to the parity:<br>with parity = 1 stop bit, no parity = 2 stop bits)<br>One stop bit<br>Two stop bits<br>Default value: Default |
| Number of Idle Chars | Number of idle characters at the start and the end of a RTU telegram frame.<br>Range of values: 1...65535<br>Default value: 5 characters   |

Table 202: TCP and UDP Ports Tab for HIMA Modbus Slave

The **System Variables** tab contains system variables that are required to control the Modbus Slave and evaluate its state from within the user program.

| Element                                | Description   |
|--|---|
| Average Buffer Fill Level for Requests | Average number of concurrent master requests  |
| Valid Master Requests                  | Number of valid master requests since the last reset of all counters or last HIMax controller's start-up.   |
| Master Requests                        | Total number of master requests since the last reset of all counters or last HIMax controller's start-up.   |
| Master Monitoring Time [ms]            | Time period after the reception of a request within which the Modbus slave must respond, see Chapter 7.3.7.   |
| Master Connection State                | FALSE: Not Connected<br>TRUE: Connected   |
| Maximum Buffer Fill Level for Requests | Maximum number of concurrent master requests  |
| Reset All Counters                     | This system variable is used to reset all counters in the user program.<br>A change from 0 to 1 triggers the reset function.<br>Values greater than 1 are treated as 1.   |
| Invalid Master Requests                | Number of invalid master requests since the last reset of all counters or last HIMax controller's start-up.<br>An invalid request is a request upon which the Modbus slave sends an error code to the Modbus master.<br>Incorrect transmissions that were already detected and filtered out at the driver level (framing errors, CRC errors, length errors) are not included here, but they are reported through the diagnosis. |
| Rejected Requests                      | Number of rejected master requests since the last reset of all counters or last HIMax controller's start-up.  |
| Response Timeout                       | Number of response timeouts since the last reset of all counters or last HIMax controller's start-up. The response timeout is the maximum time within which the sending station must receive the message acknowledgment.  |

Table 203: System Variables Tab for the HIMA Modbus Slave

### 7.3.8 Modbus Function Codes of the HIMA Modbus Slaves

#### 7.3.8.1 Modbus Function Codes

The HIMA Modbus slave supports the following Modbus function codes:

| Element   | Code | Type | Description   |
|---|------|------|---|
| READ COILS  | 01   | BOOL | Read multiple variables (BOOL) from the slave's import or export <sup>1)</sup> area.<br>Maximum length of the process data: 251 bytes.  |
| READ DISCRETE INPUT   | 02   | BOOL | Read multiple variables (BOOL) from the slave's export area.<br>Maximum length of the process data: 251 bytes.  |
| READ HOLDING REGISTER   | 03   | WORD | Read multiple variables of any type from the slave's import or export <sup>1)</sup> area.<br>Maximum length of the process data: 250 bytes.   |
| READ INPUT REGISTER   | 04   | WORD | Read multiple variables of any type from the slave's export area.<br>Maximum length of the process data: 250 bytes.   |
| WRITE SINGLE COIL   | 05   | BOOL | Write one single signal (BOOL) in the slave's import area.<br>Maximum length of the process data: 1 byte  |
| WRITE SINGLE REGISTER   | 06   | WORD | Write one single signal (WORD) in the slave's import area.<br>Maximum length of the process data: 2 bytes.  |
| Diagnosis   | 08   | x    | Only sub-code 0: Loop-back function of the slave.   |
| WRITE MULTIPLE COILS  | 15   | BOOL | Write multiple variables (BOOL) in the slave's import area.<br>Maximum length of the process data: 247 bytes.   |
| WRITE MULTIPLE REGISTER   | 16   | WORD | Write multiple variables of any type in the slave's import area.<br>Maximum length of the process data: 246 bytes.  |
| READ WRITE MULTIPLE REGISTER                                      | 23   | WORD | Write and read multiple variables of any type in and from the slave's import or export area.<br>Maximum length of the process data:<br>242 bytes (request telegram from the master Modbus Master).<br>250 bytes (response to the Master). |
| Read Device Identification  | 43   | x    | Transmit the slave identification data to the master.   |
| <sup>1)</sup> The export area can only be selected in HIMA slaves |      |      |   |

Table 204: Modbus Function Codes of the HIMA Modbus Slave

In addition to the WORD data type (2 bytes), the function codes 03, 04, 16 and 23 support all other data types.

The start address of the first variable to be transferred and the number of registers/bits of the variables to be transferred must be entered for each request.

**Error Codes:**

- If the master sends a telegram with unknown function codes, the controller responds with error code 1 (invalid code).
- If the telegram does not match the Modbus slave configuration (i.e., the request telegram does not end at one variable limit), the slave responds with error code 2 (invalid data).
- If the master sends a telegram with incorrect values (e.g., length fields), the slave responds with error code 3 (invalid value).

Communication only takes place if the COM module is in the RUN state. If the COM module is in any other operating state, the master does not respond to any requests.

**Note for Modbus Function: Read Device Identification (43)**

The HIMax Modbus slave provides the identification data to the master and supports the following object IDs:

**Basic:**

0x00 VendorName "HIMA Paul Hildebrandt GmbH"  
0x01 ProductCode "<Module serial number>"  
0x02 MajorMinorRevision "<COM Vx.y CRC>"

**Regular:**

0x03 VendorUrl "http://www.hima.com"  
0x04 ProductName "HIMax"  
0x05 ModelName "HIMax"  
0x06 UserApplicationName "-----[S.R.S]"

**Extended:**

0x80 blank "-----"  
0x81 blank "-----"  
0x82 blank "-----"  
0x83 blank "-----"  
0x84 blank "-----"  
0x85 blank "-----"  
0x86 CRC of the file modbus.config "<0x234adcef>"  
(configuration file for the Modbus slave protocol in the CPU file system. To compare with the information specified in SILworX, Online/Version comparison).

The following ReadDevice ID Codes are supported:

- (1) Read Basic device identification (stream access)
- (2) Read regular device identification (stream access)
- (3) Read extended device identification (stream access)
- (4) Read one specific identification object (individual access)

For more information on Modbus, refer to the *Modbus Application Protocol Specification* at [www.modbus.org](http://www.modbus.org)

### 7.3.9 HIMA-Specific Function Codes

HIMA-specific function codes correspond to the standard Modbus function codes. The only differences are the maximum permissible process data length of 1100 bytes and the format of the request and response headers.

| Element                                | Code          | Type | Description  |
|--|---------------|------|--|
| Read Coils Extended                    | 100<br>(0x64) | BOOL | Correspond to function code 01.<br>Read multiple variables (BOOL) from the slave's import or export <sup>1)</sup> area.<br>Maximum length of the process data: 1100 bytes.   |
| Read Discrete Inputs Extended          | 101<br>(0x65) | BOOL | Correspond to function code 02.<br>Read multiple variables (BOOL) from the slave's export area.<br>Maximum length of the process data: 1100 bytes.   |
| Read Holding Registers Extended        | 102<br>(0x66) | WORD | Correspond to function code 03.<br>Read multiple variables of any type from the slave's import or export <sup>1)</sup> area.<br>Maximum length of the process data: 1100 bytes.  |
| Read Input Registers Extended          | 103<br>(0x67) | WORD | Correspond to function code 04.<br>Read multiple variables of any type from the slave's export area.<br>Maximum length of the process data: 1100 bytes.  |
| Write Multiple Coils Extended          | 104<br>(0x68) | BOOL | Correspond to function code 15.<br>Write multiple variables (BOOL) in the slave's import area.<br>Maximum length of the process data: 1100 bytes.  |
| Write Multiple Registers Extended      | 105<br>(0x69) | WORD | Correspond to function code 16.<br>Write multiple variables of any type in the slave's import area.<br>Maximum length of the process data: 1100 bytes.   |
| Read/Write Multiple Registers Extended | 106<br>(0x6A) | WORD | Correspond to function code 23.<br>Write and read multiple variables of any type in and from the slave's import or export area.<br>Maximum length of the process data:<br>1100 bytes (request telegram from the master Modbus master).<br>1100 bytes (response to the master). |

### 7.3.9.1 Format of Request and Response Header

The request and response header of the HIMA-specific Modbus function codes are structured as follows:

| Code          | Request   | Response  |
|---------------|---|---|
| 100<br>(0x64) | 1 byte function code 0x64<br>2 bytes start address<br>2 bytes number of coils 1...8800(0x2260)  | 1 byte function code 0x64<br>2 bytes number of bytes = N<br>N bytes coil data<br>(8 coils are packed in one byte) |
| 101<br>(0x65) | 1 byte function code 0x65<br>2 bytes start address<br>2 bytes number of coils 1...8800(0x226)   | 1 byte function code 0x65<br>2 bytes number of bytes = N<br>N bytes coil data<br>(8 coils are packed in one byte) |
| 102<br>(0x66) | 1 byte function code 0x66<br>2 bytes start address<br>2 bytes number of registers 1...550 (0x226)   | 1 byte function code 0x66<br>2 bytes number of bytes = N<br>N bytes register data                                 |
| 103<br>(0x67) | 1 byte function code 0x67<br>2 bytes start address<br>2 bytes number of registers 1...550 (0x226)   | 1 byte function code 0x67<br>2 bytes number of bytes = N<br>N bytes register data                                 |
| 104<br>(0x68) | 1 byte function code 0x68<br>2 bytes start address<br>2 bytes number of coils 1...8800(0x2260)<br>2 bytes number of bytes = N<br>N bytes coil data  | 1 byte function code 0x66<br>2 bytes start address<br>2 bytes number of coils<br>1...8800(0x2260)                 |
| 105<br>(0x69) | 1 byte function code 0x69<br>2 bytes start address<br>2 bytes number of registers 1...550 (0x226)<br>2 bytes number of bytes = N<br>N bytes register data   | 1 byte function code 0x69<br>2 bytes start address<br>2 bytes number of registers<br>1...550 (0x226)              |
| 106<br>(0x6A) | 1 byte function code 0x6a<br>2 bytes read start address<br>2 bytes number of read registers 1...550<br>(0x226)<br>2 bytes write start address<br>2 bytes number of write registers 1...550<br>(0x226)<br>2 bytes number of write bytes = N<br>N bytes register data | 1 byte function code 0x6a<br>2 bytes number of bytes = N<br>N bytes register data                                 |

### 7.3.10 Addressing Modbus using Bit and Register

The addressing mode adheres to the Modbus addressing standard and only knows the two data lengths bit (1 bit) and register (16 bits) that are used to transfer all the data types allowed.

There are two areas within the Modbus slave: a **Register Area** (inputs and outputs) and a **Bit Area** (inputs and outputs). Both areas are separated from one another and can accept all permitted data types. The difference between these areas resides in the Modbus function codes permitted for accessing them.

- 
- i** The Modbus addressing using bit and register does not guarantee the variable integrity, meaning that any arbitrary portion of a variable can be read or written with this access mode. Variables of type BOOL are stored in a packed format, i.e., each variable of type BOOL is stored as a bit within a byte.
- 

#### 7.3.10.1 Register Area

The variables in the Register Area are created in the **Register Variables** tab. For more information on how to assign send/receive variables, refer to Chapter 7.3.5.

- 
- i** To access variables in the Register Area with the Modbus function codes 1, 2, 5, 15, the variables must be mirrored in the Bit Area, see Chapter 7.3.11.1.
- 

The variables in the Register Area can only be accessed using the Modbus function codes 3, 4, 6, 16, 23. To do this, enter the start address of the first variable in the properties of the function code.

Example: Accessing the Variables in the Register Area of the Modbus Slave

| Register variables    | Register.Bit | Bit |
|-----------------------|--------------|-----|
| 00_Register_Area_WORD | 0.0          | 0   |
| 01_Register_Area_SINT | 1.8          | 16  |
| 02_Register_Area_SINT | 1.0          | 24  |
| 03_Register_Area_REAL | 2.0          | 32  |
| 04_Register_Area_BOOL | 4.8          | 64  |
| 05_Register_Area_BOOL | 4.9          | 65  |
| 06_Register_Area_BOOL | 4.10         | 66  |
| 07_Register_Area_BOOL | 4.11         | 67  |
| 08_Register_Area_BOOL | 4.12         | 68  |
| 09_Register_Area_BOOL | 4.13         | 69  |
| 10_Register_Area_BOOL | 4.14         | 70  |
| 11_Register_Area_BOOL | 4.15         | 71  |
| 12_Register_Area_BOOL | 4.0          | 72  |
| 13_Register_Area_BOOL | 4.1          | 73  |
| 14_Register_Area_BOOL | 4.2          | 74  |
| 15_Register_Area_BOOL | 4.3          | 75  |
| 16_Register_Area_BOOL | 4.4          | 76  |
| 17_Register_Area_BOOL | 4.5          | 77  |
| 18_Register_Area_BOOL | 4.6          | 78  |
| 19_Register_Area_BOOL | 4.7          | 79  |

Table 205: Register Variables in the Register Area of the Modbus Slave

### HIMA Modbus Master Configuration of the Request Telegram

To read the variables **01\_Register\_Area\_SINT** to **03\_Register\_Area\_REAL** in the Modbus master

1. Right-click **TCP/UDP slaves** and select **New** from the context menu.
2. Select **Read Holding Registers (3)** from the list.
3. Right-click **Read Holding Registers (3)** and select **Properties**.
  - Enter 1 in the **start address of the read area**.
4. Right-click **Read Holding Registers (3)** and select **Edit**.
5. Drag the following variables from the **Object Panel** onto the Input Variables tab..

| Register variables    | Offset |
|-----------------------|--------|
| 01_Register_Area_SINT | 0      |
| 02_Register_Area_SINT | 1      |
| 03_Register_Area_REAL | 2      |

6. Right-click anywhere in the **Output Variables** area to open the context menu and click **New Offsets** to renumber the variable offsets.

#### 7.3.10.2 Bit Area

The variables in the Bit Area are created in the **Bit Variables** tab. For more information on how to assign send/receive variables, refer to Chapter 7.3.5.



To access variables in the Register Area with the Modbus function codes 3, 4, 6, 16 and 23, the variables must be mirrored in the Bit Area, see Chapter 7.3.11.2.

The variables in the Bit Area can only be accessed using the Modbus function codes 1, 2, 5, 15. To do this, enter the start address of the first variable in the properties of the function code. Example: Accessing the Variables in the Bit Area of the Modbus Slave

| Bit Variables    | Bit | Register.Bit |
|------------------|-----|--------------|
| 00_BIT_Area_WORD | 0   | 0.0          |
| 01_BIT_Area_SINT | 16  | 1.8          |
| 02_BIT_Area_SINT | 24  | 1.0          |
| 03_BIT_Area_REAL | 32  | 2.0          |
| 04_BIT_Area_BOOL | 64  | 4.8          |
| 05_BIT_Area_BOOL | 65  | 4.9          |
| 06_BIT_Area_BOOL | 66  | 4.10         |
| 07_BIT_Area_BOOL | 67  | 4.11         |
| 08_BIT_Area_BOOL | 68  | 4.12         |
| 09_BIT_Area_BOOL | 69  | 4.13         |
| 10_BIT_Area_BOOL | 70  | 4.14         |
| 11_BIT_Area_BOOL | 71  | 4.15         |
| 12_BIT_Area_BOOL | 72  | 4.0          |
| 13_BIT_Area_BOOL | 73  | 4.1          |
| 14_BIT_Area_BOOL | 74  | 4.2          |
| 15_BIT_Area_BOOL | 75  | 4.3          |
| 16_BIT_Area_BOOL | 76  | 4.4          |
| 17_BIT_Area_BOOL | 77  | 4.5          |
| 18_BIT_Area_BOOL | 78  | 4.6          |
| 19_BIT_Area_BOOL | 79  | 4.7          |

Table 206: Bit Variables in the Bit Area of the Modbus Slave

### HIMA Modbus Master Configuration of the Request Telegram

**To read the variables 04\_BIT\_Area\_BOOL to 06\_Area\_BOOL in the Modbus master**

1. Right-click **TCP/UDP slaves**, then click **New**.
2. Select **Read Coils (1)** from the list.
3. Right-click **Read Coils (1)** and select **Properties** from the context menu.
  - Enter **64** in the **start address of the read area**.
4. Right-click **Read Coils (1)** and select **Edit** from the context menu.
5. Drag the following variables from the **Object Panel** onto the Input Variables tab.

| Bit Variables    | Offset |
|------------------|--------|
| 04_BIT_Area_BOOL | 0      |
| 05_BIT_Area_BOOL | 1      |
| 06_BIT_Area_BOOL | 2      |

6. Right-click anywhere in the **Output Variables** area to open the context menu and click **New Offsets** to renumber the variable offsets.

#### 7.3.11 Offsets for Alternative Modbus Addressing

To access the variables in the **Bit Area** using the Modbus function codes (of type Register), the variables must be mirrored in the **Register Area**, and to access the variables in the **Register Area** using the Modbus function codes (of type Bit), the variables must be mirrored in the **Bit Area**. The offsets for the mirrored variables are entered in the **Properties/Offsets** tab.

**To mirror the variables in the Bit Area and Register Area**

1. Right-click the Modbus slave and select **Edit**, and **Offsets**, then activate **Use Alternative Register/Bit Addressing**.
  - This action mirrors the variables in the corresponding area.
2. Enter the offset for the mirrored variables in the Bit Area and Register Area.



The existing variables and the corresponding variables mirrored in the Bit/Register Area must not overlap with respect to the Modbus addresses.

| Element                               | Description / Range of values |   |
|---------------------------------------|-------------------------------|---|
| Alternative register / bit addressing | Activated<br>Deactivate<br>d  | Use the alternative addressing<br>Do not use the alternative addressing<br>Default value: Deactivated |
| Register area offset / bit inputs     | 0...65535                     |   |
| Register area offset / bit outputs    | 0...65535                     |   |
| Bit area offset / register inputs     | 0...65535                     |   |
| Bit area offset / register outputs    | 0...65535                     |   |

Table 207: Offsets Tab for HIMA Modbus Slave

### 7.3.11.1 Access to the Register Variables in the Bit Area of the Modbus Slave

To access the Register Area with the Modbus function codes (of type Bit) 1, 2, 5, 15, the register variables must be mirrored in the **Bit Area**. The offsets for the mirrored register variables must be entered in the **Properties/Offsets** tab.

Example:

|                                    |      |
|------------------------------------|------|
| Bit area offset / register inputs  | 8000 |
| Bit area offset / register outputs | 8000 |

The variables mirrored from the Register Area to the Bit Area are located here starting with Bit Address 8000.

| Mirrored Register Variables | Bit  |
|-----------------------------|------|
| 00_Register_Area_WORD       | 8000 |
| 01_Register_Area_SINT       | 8016 |
| 02_Register_Area_SINT       | 8024 |
| 03_Register_Area_REAL       | 8032 |
| 04_Register_Area_BOOL       | 8064 |
| 05_Register_Area_BOOL       | 8065 |
| 06_Register_Area_BOOL       | 8066 |
| 07_Register_Area_BOOL       | 8067 |
| 08_Register_Area_BOOL       | 8068 |
| 09_Register_Area_BOOL       | 8069 |
| 10_Register_Area_BOOL       | 8070 |
| 11_Register_Area_BOOL       | 8071 |
| 12_Register_Area_BOOL       | 8072 |
| 13_Register_Area_BOOL       | 8073 |
| 14_Register_Area_BOOL       | 8074 |
| 15_Register_Area_BOOL       | 8075 |
| 16_Register_Area_BOOL       | 8076 |
| 17_Register_Area_BOOL       | 8077 |
| 18_Register_Area_BOOL       | 8078 |
| 19_Register_Area_BOOL       | 8079 |

Table 208: Variables Mirrored from the Register Area to the Bit Area

#### HIMA Modbus Master Configuration of the Request Telegram

**To read the variables 04\_Register\_Area\_BOOL to 06\_Register\_Area\_BOOL in the Modbus master**

1. Right-click **TCP/UDP slaves**, then click **New**.
2. Select **Read Coils (1)** from the list.
3. Right-click **Read Coils (1)** and select **Properties** from the context menu.
  - Enter **8064** in the **start address of the read area**.
4. Right-click **Read Coils (1)** and select **Edit** from the context menu.
5. Drag the following variables from the **Object Panel** onto the Input Variables tab..

| Mirrored Register Variables | Offset |
|-----------------------------|--------|
| 04_Register_Area_BOOL       | 0      |
| 05_Register_Area_BOOL       | 1      |
| 06_Register_Area_BOOL       | 2      |

6. Right-click anywhere in the **Output Variables** area to open the context menu and click **New Offsets** to renumber the variable offsets.

### 7.3.11.2 Access to the Bit Variables in the Register Area of the Modbus Slave

To access the bit variables with the Modbus function codes (of type Register) 3, 4, 6, 16, 23, the bit variables must be mirrored in the **Register Area**. The offsets for the mirrored bit variables must be entered in the **Properties/Offsets** tab.

Example:

Register area offset / bit inputs: 1000

Register area offset / bit outputs: 1000

The variables mirrored from the Bit Area to the Register Area are located here starting with Register Address 1000.

| Mirrored Bit Variables | Register.Bit |
|------------------------|--------------|
| 00_BIT_Area_WORD       | 1000.0       |
| 01_BIT_Area_SINT       | 1001.8       |
| 02_BIT_Area_SINT       | 1001.0       |
| 03_BIT_Area_REAL       | 1002.0       |
| 04_BIT_Area_BOOL       | 1004.8       |
| 05_BIT_Area_BOOL       | 1004.9       |
| 06_BIT_Area_BOOL       | 1004.10      |
| 07_BIT_Area_BOOL       | 1004.11      |
| 08_BIT_Area_BOOL       | 1004.12      |
| 09_BIT_Area_BOOL       | 1004.13      |
| 10_BIT_Area_BOOL       | 1004.14      |
| 11_BIT_Area_BOOL       | 1004.15      |
| 12_BIT_Area_BOOL       | 1004.0       |
| 13_BIT_Area_BOOL       | 1004.1       |
| 14_BIT_Area_BOOL       | 1004.2       |
| 15_BIT_Area_BOOL       | 1004.3       |
| 16_BIT_Area_BOOL       | 1004.4       |
| 17_BIT_Area_BOOL       | 1004.5       |
| 18_BIT_Area_BOOL       | 1004.6       |
| 19_BIT_Area_BOOL       | 1004.7       |

Table 209: Variables Mirrored from the Bit Area to the Register Area

#### HIMA Modbus Master Configuration of the Request Telegram

##### To read the variables 01\_BIT\_Area\_SINT to 03\_BIT\_Area\_REAL in the Modbus master

1. Right-click **TCP/UDP slaves**, then click **New**.
2. Select **Read Holding Registers (3)** from the list.
3. Right-click **Read Holding Registers (3)** and select **Properties**.
  - Enter **1001** in the **start address of the read area**.
4. Right-click **Read Holding Registers (3)** and select **Edit**.
5. Drag the following variables from the **Object Panel** onto the Input Variables tab..

| Mirrored Bit Variables | Offset |
|------------------------|--------|
| 01_BIT_Area_SINT       | 0      |
| 02_BIT_Area_SINT       | 1      |
| 03_BIT_Area_REAL       | 2      |

6. Right-click anywhere in the **Output Variables** area to open the context menu and click **New Offsets** to renumber the variable offsets.

### 7.3.12 Control Panel (Modbus Slave)

The Control Panel can be used to verify and control the settings for the Modbus slave. Details about the slave's current state (e.g., master state, etc.) are displayed.

#### To open the Control Panel for monitoring the Modbus slave

1. Right-click the **Hardware** structure tree node and select **Online** from the context menu.
2. In the **System Login** window, enter the access data to open the online view for the hardware.
3. Double-click **COM Module** and select the **Modbus Slave** structure tree node.

#### 7.3.12.1 Context Menu (Modbus Slave)

The following command is available from the context menu for the selected Modbus slave:

##### **Reset Statistics**

Reset statistical data (min./max. cycle time etc.) to 0.

### 7.3.12.2 View Box (Modbus Slave)

The view box displays the following values of the selected Modbus slave.

| Element                 | Description   |
|-------------------------|---|
| Name                    | Modbus slave name   |
| Planned µP Budget [%]   | See Chapter 7.3.4   |
| Current µP Budget [%]   |   |
| SRS of Redundant Module | SRS of the redundant COM module   |
| Response Time [ms]      | Time period after the reception of a request within which the Modbus slave may respond. |

Table 210: View Box of the Modbus Slave

### 7.3.12.3 View Box (Master Data)

The Master Data view box displays the following values.

| Element                                | Description  |
|--|--|
| Name                                   | Name of master data  |
| Requests                               | Total number of master requests since the last counter reset.  |
| Valid Requests                         | Number of valid master requests since the last counter reset.  |
| Invalid Requests                       | Number of invalid master requests since the last counter reset.<br>The invalid requests only include requests that were acknowledged by the master.<br>Requests with CRC error that were improperly received are automatically rejected. |
| Master Timeout [ms]                    | Timeout within which the slave must receive at least one request from the master.<br>If the slave receives no request within the timeout, the <i>Master Connection Status</i> is set to <i>not connected</i> .                           |
| Connection State                       | 0 = Not monitored<br>1 = Not connected<br>2 = Connected  |
| Response Timeout                       | Number of response timeouts since the last reset of all counters or last HIMax controller's start-up. The response timeout is the maximum time within which the sending station must receive the message acknowledgment.                 |
| Rejected Requests                      | Number of rejected master requests since the last reset of all counters or last HIMax controller's start-up.   |
| Maximum Buffer Fill Level for Requests | Maximum number of concurrent master requests   |
| Average Buffer Fill Level for Requests | Average number of concurrent master requests   |

Table 211: Master Data View Box

### 7.3.13 FBx LED Function in the Modbus Slave

The FBx LED of the corresponding fieldbus interface indicates the state of the Modbus protocol. The states of the FBx LED are specified in the following table:

| FBx LED  | Color  | Description  |
|----------|--------|--|
| OFF      | Yellow | The Modbus slave protocol is not active!<br>This means that the controller is in STOP or no Modbus master is configured. |
| Blinking | Yellow | The Modbus slave protocol is active and is exchanging data with the Modbus master.                                       |

Table 212: The FBx LED

### 7.3.14 Function of the FAULT LED in the Modbus Slave (HIMax only)

The FAULT LED of the corresponding fieldbus interface indicates a failure of the Modbus protocol. The states of the FAULT LED are specified in the following table:

| FAULT LED | Color | Description   |
|-----------|-------|---|
| OFF       | Red   | PROFIBUS DP slave protocol is not disturbed.  |
| Blinking  | Red   | The Modbus slave protocol is disturbed.<br>The following events result in a malfunction. <ul style="list-style-type: none"> <li>▪ Unknown function code received</li> <li>▪ Request with incorrect addressing received</li> <li>▪ Calculating time budget exceeded</li> </ul> If no faults occur for a period longer than 5 seconds, the state changes to <i>Protocol not disturbed</i> . |

Table 213: The FAULT FBx

### 7.3.15 Error Codes of the Modbus TCP/IP Connection

The error codes of the Modbus TCP/IP connection are output in the **Diagnosis** dialog box.

| Error Code | Description                                |
|------------|--|
| 35         | Operation is blocked                       |
| 48         | Port already in use                        |
| 50         | Network is down                            |
| 53         | Software caused connection abort           |
| 54         | Peer caused connection abort               |
| 55         | No buffer space available                  |
| 60         | Operation timed out, connection terminated |
| 61         | Connection refused (from peer)             |
| 65         | No route to peer host                      |

Table 214: Error Codes of Modbus TCP/IP

## 8 Send & Receive TCP

S&R TCP is a manufacturer-independent, standard protocol for cyclic and acyclic data exchange and does not use any specific protocols other than TCP/IP.

With the S&R TCP protocol, the HIMax/HIMatrix controller supports almost every third-party system as well as PCs with implemented socket interface to TCP/IP (for example Winsock.dll).

S&R TCP is compatible with the Siemens SEND/RECEIVE interface and ensures communication with Siemens controllers via TCP/IP. Data is exchanged using the S7 function blocks AG\_SEND (FC5) and AG\_RECV (FC6).

### 8.1 System Requirements

#### Equipment and system requirements

| Element          | Description   |
|------------------|---|
| Controller       | HIMax with COM module<br>HIMatrix: CPU OS version 7 and beyond and COM OS version 12 and beyond |
| Processor module | The Ethernet interfaces on the processor module may not be used for S&R TCP.                    |
| COM module       | Ethernet 10/100BaseT<br>One S&R TCP protocol can be configured for each COM module.             |
| Activation       | Software activation code required, see Chapter 3.4.   |

Table 215: Equipment and System Requirements for the S&R TCP

#### Properties of the S&R TCP protocol

| Element                   | Description  |
|---------------------------|--|
| Safety-related            | No   |
| Data exchange             | Cyclic and acyclic data exchange over TCP/IP.  |
| Function Blocks           | The S&R TCP function blocks must be used for acyclically exchanging data.  |
| TCP Connections           | Up to 32 TCP connections can be configured in one controller, provided that the maximum size of send or received data is not exceeded.   |
| Max. Size of Send Data    | See Table 2 Standard Protocols   |
| Max. Size of Receive Data | <p><b>i</b> To determine the maximum amount of reference data, the value for all status variables of the configured TCP connections and TCP/SR function blocks must be subtracted from the value for the maximum amount of send data. The data can be freely allocated among multiple TCP connections.</p> |

Table 216: S&R TCP Properties

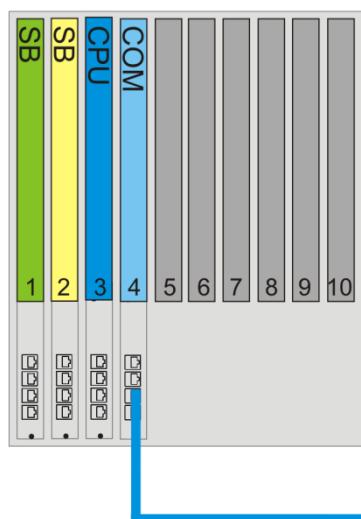
#### 8.1.1 Creating a S&R TCP Protocol

##### To create a new S&R TCP protocol

1. In the structure tree, open **Configuration, Resource, Protocols**.
2. Right-click **Protocols** and select **New, Send/Receive over TCP** from the context menu to add a new S&R TCP protocol.
3. Right-click **Send/Receive over TCP** and select **Properties**. In the **General** tab, select **COM Module**.

## 8.2 Example: S&R TCP Configuration

HIMax



Siemens Simatic 300

Figure 63: Connecting a HIMax to a Siemens Controller

In this example, the protocol Send/Receive over TCP is installed in a HIMax controller. The HIMax is supposed to cyclically communicate via S&R TCP with a Siemens controller (e.g., SIMATIC 300).

In this example, HIMax (client) is the active station that establishes the TCP connection to the passive Siemens SIMATIC 300 (server). Once the connection has been established, both stations are equal and can send and receive data at any point in time.

When connecting the HIMax to the Siemens SIMATIC 300, the following points must be taken into account:

The requirements described in Chapter 8.1 System Requirements apply for the HIMax.

HIMax and Siemens SIMATIC 300 are connected to one another via Ethernet interfaces.

HIMax and Siemens SIMATIC 300 must be located in the same subnet or must have the corresponding routing settings if a router is used.

In this example, the HIMA controller is supposed to send two BYTES and one WORD to the Siemens SIMATIC 300. The variables are received in the Siemens SIMATIC 300 by the function block AG\_RECV (FC 6) and are internally transmitted to the function block AG\_SEND (FC 5). Siemens SIMATIC 300 sends the variables (unchanged) back to the HIMax controller using the function block AG\_SEND (FC 5).

Once the configuration is completed, the user can verify the variable transmission using the HIMA Force Editor.

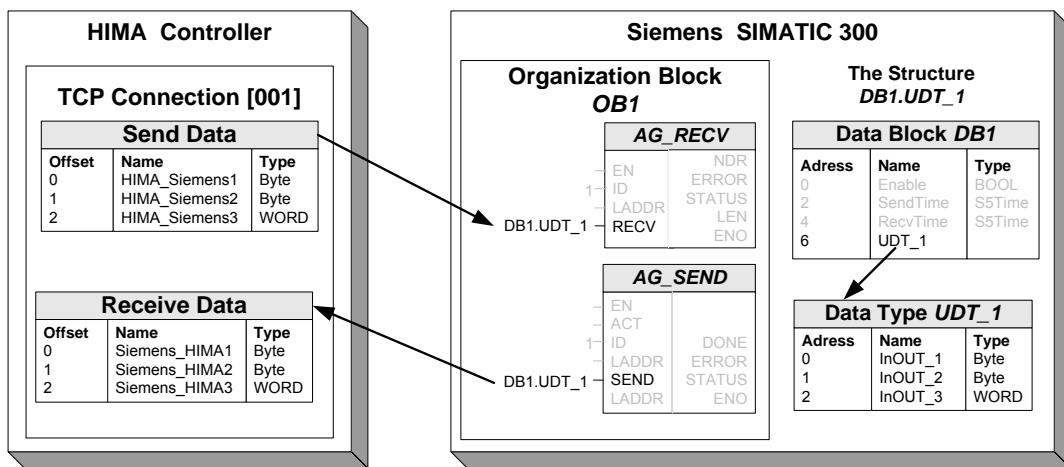


Figure 64: Data Transfer between a HIMax and a Siemens Controller

#### Description of the HIMax controller configuration

| Element              | Description  |
|----------------------|--|
| TCP connection [001] | This dialog box contains all parameters required for communicating with the communication partner (Siemens SIMATIC 300).                                       |
| Send Data            | The variable offsets and types in the controller must be identical with the variable addresses and types with data type <i>UDT_1</i> in the SIMATIC 300.       |
| Receive Data         | The variable offsets and types in the HIMax controller must be identical with the variable addresses and types with data type <i>UDT_1</i> in the SIMATIC 300. |

Table 217: HIMax Controller Configuration

#### Description of the Siemens SIMATIC 300 Configuration

| Element                | Description   |
|------------------------|---|
| Organization block OB1 | The function blocks AG_RECV (FC6) and AG_SEND (FC 5) must be created and configured in the OB1 organization block.  |
| AG_RECV (FC 6)         | The function block AG_RECV (FC 6) accepts the data received from the communication partner with data type DB1.UDT_1. The inputs ID and LADDR must be appropriately configured for communication with the communication partner.                     |
| AG_SEND (FC 5)         | The function block AG_SEND (FC 5) transfers the data from data type DB1.UDT_1 to the communication partner. The inputs ID and LADDR must be appropriately configured for communication with the communication partner.                              |
| Data block DB1         | The data type UDT_1 is defined in the data block DB1.   |
| Data type UDT_1        | The addresses and types of the variables in SIMATIC 300 must be identical with the offsets and types of the controller. The data type UDT_1 accepts the received user data and stores them until they are transmitted to the communication partner. |

Table 218: Siemens SIMATIC 300 Configuration

## 8.2.1 S&R TCP Configuration of the Siemens Controller SIMATIC 300

- i** The following step by step instructions for configuring the Siemens controller are not to be considered exhaustive.

This information is provided without guarantee (errors and omissions excepted); refer to the Siemens documentation when developing projects with Siemens controllers.

### To create the S&R TCP server in the SIMATIC 300 project

1. Start the SIMATIC manager.
2. In the SIMATIC manager, open the project associated with the SIMATIC 300 controller.
3. In this project, create and configure the *Industrial Ethernet* and the *MPI* connections.

### To create the UDT1 data type using the following variables

1. Open the *Function Blocks* directory in the Siemens SIMATIC manager.
2. Select **Add, S7 Block, Data Type** from the main menu and create a data type.
3. Name the data type **UDT1**
4. Give the symbolic name **UDT\_1** to the data type.
5. In data type **UDT\_1**, create the three **InOut\_x** variables as described in the figure below.

| Address | Name    | Type       | Initial value | Comment |
|---------|---------|------------|---------------|---------|
| 0.0     |         | STRUCT     |               |         |
| +0.0    | InOut_1 | BYTE       | B#16#0        |         |
| +1.0    | InOut_2 | BYTE       | B#16#0        |         |
| +2.0    | InOut_3 | WORD       | W#16#0        |         |
| =4.0    |         | END_STRUCT |               |         |

Figure 65: List of Variables in the Siemens UDT1 Block

- i** During cyclic and acyclic data exchange, note that some controllers (e.g., SIMATIC 300) add so-called *pad bytes*. Pad bytes ensure that all data types greater than one byte always begin at an even offset and that also the total size of the defined variables is even.  
In such a case, dummy bytes must be used on the correct place of the HIMax controller (see Chapter 8.6 Third-Party Systems with Pad Bytes).

**To create the DB1 data block for the FC 5 and FC 6 function blocks**

1. Select **Add, S7 Function Block, Data Block** on the main menu and create a data block.
2. Enter the name **DB1** for the data block.
3. Enter the symbolic name **DB1** for the data block.
4. Assign the *UDT\_1* data type to the *DB1* data block.
5. In the *DB1* data block, configure the data types such as described in the figure below.

| Address | Name     | Type       | Initial val | Comment |
|---------|----------|------------|-------------|---------|
| 0.0     |          | STRUCT     |             |         |
| +0.0    | Enable   | BOOL       | TRUE        |         |
| +2.0    | SendTime | S5TIME     | S5T#100MS   |         |
| +4.0    | RecvTime | S5TIME     | S5T#10MS    |         |
| +6.0    | UDT_1    | "UDT_1"    |             |         |
| =10.0   |          | END_STRUCT |             |         |

Figure 66: List of Variables in the Siemens DB1 Function Block

**To create the following symbols in the Symbol Editor**

1. Double-click the **OB1** organization block to open the *KOP/AWL/FUP* dialog box.
2. Select **Extras, Symbol Table** from the main menu to open the Symbol Editor.
3. Add the variables **M 1.0...MW 5** in the *Symbol Editor* such as specified in the figure below.

|    | Status | Symbol          | Address | Data type |
|----|--------|-----------------|---------|-----------|
| 1  |        | DB1             | DB 1    | DB 1      |
| 2  |        | RecDone         | M 1.0   | BOOL      |
| 3  |        | RecError        | M 1.1   | BOOL      |
| 4  |        | SendDone        | M 1.2   | BOOL      |
| 5  |        | SendError       | M 1.3   | BOOL      |
| 6  |        | RecStatus       | MW 1    | WORD      |
| 7  |        | RecLen          | MW 3    | INT       |
| 8  |        | SendStatus      | MW 5    | WORD      |
| 9  |        | Cycle Execution | OB 1    | OB 1      |
| 10 |        | UDT_1           | UDT 1   | UDT 1     |
| 11 |        |                 |         |           |

Figure 67: SIMATIC Symbol Editor

**To create the AG\_RECV (FC 6) function block**

1. Open the *KOP/AWL/FUP* dialog box.
2. From the structure tree located on the left side of the Symatic Manager, select the following function blocks in the given order:  
*one OR gate*  
*one S\_VIMP*  
*one AG\_RECV (FC 6)*
3. Drag these function blocks onto the *OB1* organization block.
4. Connect and configure the *function blocks* as described in the figure below.
5. Right-click the **AG\_RECV (FC 6)** function block, and then click **Properties**.
6. Deactivate **Active Connection Setup** and configure the ports.

7. Note down the *LADDR* function block parameter and enter it in the function chart on the *AG\_RECV (FC 6)* function block.

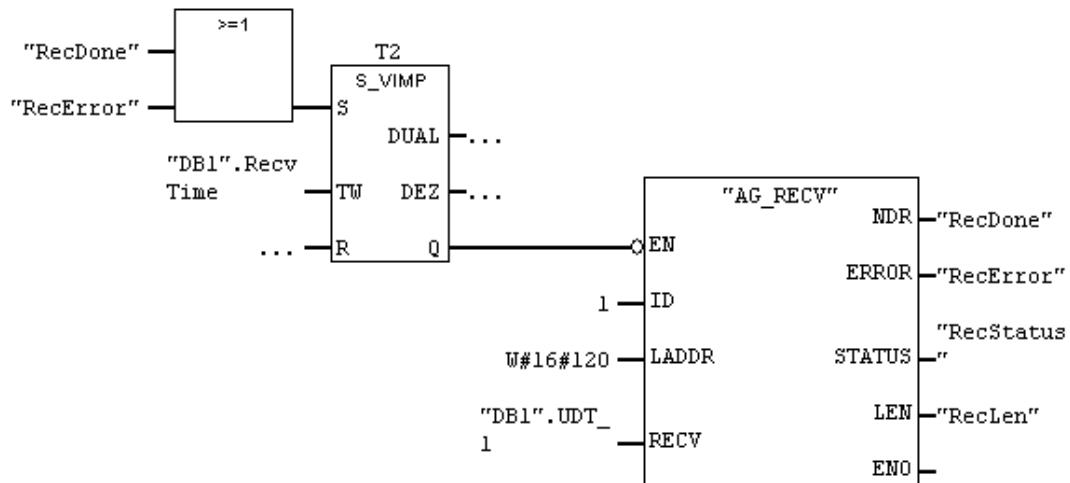


Figure 68: Receive Function Chart

### To create the AG\_SEND (FC 5) function block

1. Open the KOP/AWL/FUP dialog box.
2. From the structure tree located on the left side of the Symatic Manager, select the following function blocks in the given order:  
*one OR gate*  
*one S\_VIMP*  
*one AG\_SEND (FC 5)*
3. Drag these function blocks onto the OB1 organization block.
4. Connect and configure the *function blocks* as described in the figure below.
5. Right-click the **AG\_SEND (FC 5)** function block, and then click **Properties**.
6. Deactivate **Active Connection Setup** and configure the ports.
7. Note down the *LADDR* function block parameter and enter it in the function chart on the **AG\_SEND (FC 5)** function block.

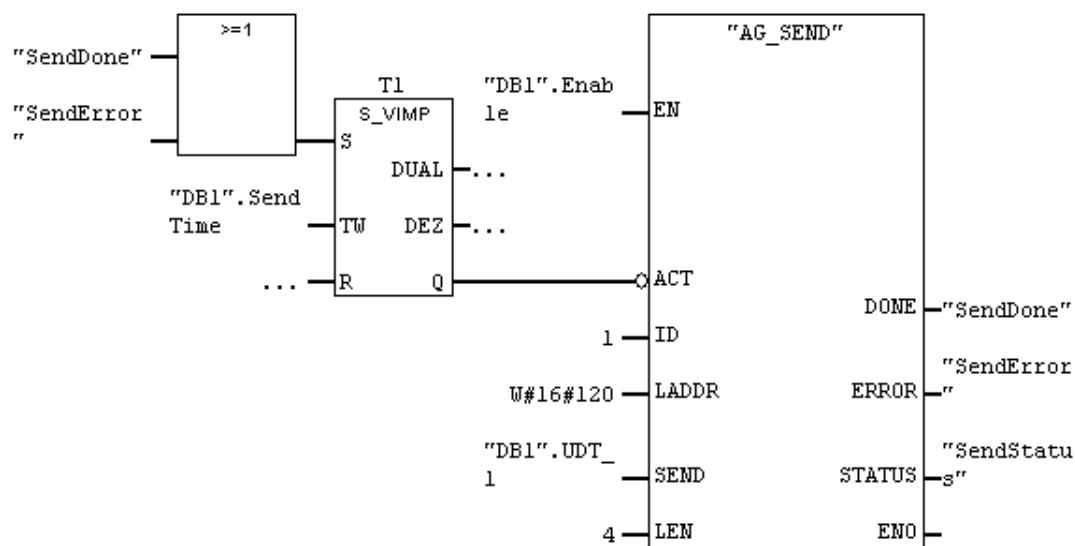


Figure 69: Send Function Chart

### To load the code into the SIMATIC 300 controller

1. Start the **Code Generator** for the program.
2. Make sure that the code was generated without error.
3. Load the code into the SIMATIC 300 controller.

## 8.2.2 S&R TCP Configuration of the HIMax Controller

For more information on how to configure the HIMax controllers and use the SILworX programming tool, refer to the SILworX First Steps manual.

### To create the following global variables in the Variable Editor

1. In the structure tree, open **Configuration, Global Variables**.
2. Right-click the **Global Variables**, and then click **Edit**.
3. Create the global variables as described in Table 219.

| Name          | Type |
|---------------|------|
| Siemens_HIMA1 | Byte |
| Siemens_HIMA2 | Byte |
| Siemens_HIMA3 | WORD |
| HIMA_Siemens1 | Byte |
| HIMA_Siemens2 | Byte |
| HIMA_Siemens3 | WORD |

Table 219: Global Variables

### To create the S&R TCP protocol in the resource

1. In the structure tree, open **Configuration, Resource**.
2. Right-click **Protocols**, then click **New**.
3. Select **Send/Receive over TCP** and enter a name for the protocol.
4. Click **OK** to create a new protocol.
5. Right-click **Send/Receive over TCP**, then click **Properties**.
6. Click **COM Module**. The remaining parameters retain the default values.

### To create the TCP connection

1. Right-click **Send/Receive over TCP**, then click **New**.
2. Right-click **TCP Connection**, then click **Properties**.
3. Configure the properties such as specified in the figure.

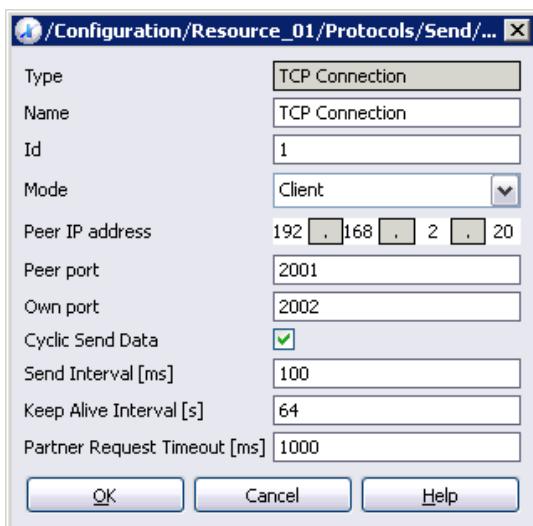


Figure 70: TCP Connection Properties in SILworX

- If parameters for cyclic data exchange between two controllers should be set, the option *Cyclic data transfer* must be activated in the *Properties* dialog box for the TCP Connection.
-

**To configure the receive data of the HIMax controller**

1. Right-click **TCP Connection**, then click **Edit**.
2. Select the **Process Variables** tab.
3. Drag the following global variables from the **Object Panel** onto the **Input Signals** area.

| Global Variable | Type |
|-----------------|------|
| Siemens_HIMA1   | Byte |
| Siemens_HIMA2   | Byte |
| Siemens_HIMA3   | WORD |

Table 220: Variables for Receive Data

4. Right-click anywhere in the **Register Inputs** area, and then click **New Offsets** to renumber the variable offsets.

- i** Take into account that the variable offsets in the HIMax controller must be identical with the variable addresses with *UDT\_1* data type in the SIMATIC 300.

**To configure the send data of the HIMax controller**

1. Right-click **TCP Connection**, then click **Edit**.
2. Select the **Process Variables** tab.
3. Drag the following global variables from the **Object Panel** onto the **Input Signals** area.

| Global Variable | Type |
|-----------------|------|
| HIMA_Siemens1   | Byte |
| HIMA_Siemens2   | Byte |
| HIMA_Siemens3   | WORD |

Table 221: Variables for Send Data

4. Right-click anywhere in the **Register Inputs** area, and then click **New Offsets** to renumber the variable offsets.

- i** Note that the variable offsets in the HIMax controller must be identical with the variable addresses with *UDT\_1* data type in the SIMATIC 300.

**To verify the S&R TCP configuration**

1. In the structure tree, open **Configuration, Resource, Protocols, Send/Receive Protocol over TCP**.
2. Click the **Verification** button located on the Action Bar, and then click **OK** to confirm the action.
3. Thoroughly verify the messages displayed in the logbook and correct potential errors.

### 8.3 TCP S&R Protocols Menu Functions

#### 8.3.1 Edit

The **Edit** dialog box for the S&R TCP protocol contains the following tab:

##### System Variables

*System variables* are used to evaluate the state of the TCP Send Receive protocol from within the user program.

| Element                    | Description   |
|----------------------------|---|
| Active Connection Count    | System variable providing the number of active (not disturbed) connections.   |
| Disturbed Connection Count | System variable providing the number of disturbed connections.<br>Disturbed means that the TCP connection was interrupted due to a timeout or an error. |
| Status                     | No function   |

Table 222: System Variables S&R TCP

#### 8.3.2 Properties

Over a TCP connection, data is exchanged cyclically or acyclically. The S&R TCP function blocks are required for the acyclic data exchange.

On a connection, data cannot be simultaneously exchanged cyclically and acyclically.

## General

| Name                                | Description  |                    |  |                   |   |
|-------------------------------------|--|--------------------|--|-------------------|---|
| Type                                | Send/Receive over TCP  |                    |  |                   |   |
| Name                                | Name for the current Send/Receive over TCP Protocol. A maximum of 31 characters.   |                    |  |                   |   |
| Module                              | Selection of the COM module within which the protocol is processed.  |                    |  |                   |   |
| Activate Max. $\mu$ P Budget        | <p>Activated:<br/>Adopt the <math>\mu</math>P budget limit from the Max. <math>\mu</math>P Budget in [%] field.</p> <p>Deactivated:<br/>Do not use the <math>\mu</math>P budget limit for this protocol.</p>   |                    |  |                   |   |
| Max. $\mu$ P budget in [%]          | <p>Maximum <math>\mu</math>P budget for the module that can be used for processing the protocols.</p> <p>Range of values: 1...100%<br/>Default value: 30%</p>  |                    |  |                   |   |
| Behavior on CPU/COM Connection Loss | <p>If the connection of the processor module to the communication module is lost, the input variables are initialized or continue to be used unchanged in the process module, depending on this parameter. Example: the communication module is removed when communication is running.</p> <p><b>If a project created with a SILworX version prior to V3 should be converted, this value must be set to Retain Last Value to ensure that the CRC does not change.</b></p> <p><b>For HIMatrix controllers with CPU OS version prior to V8, this value must always be set to Retain Last Value.</b></p> <table> <tr> <td>Adopt Initial Data</td> <td>Input variables are reset to their initial values.</td> </tr> <tr> <td>Retain Last Value</td> <td>The input variables retains the last value.</td> </tr> </table> | Adopt Initial Data | Input variables are reset to their initial values. | Retain Last Value | The input variables retains the last value. |
| Adopt Initial Data                  | Input variables are reset to their initial values.   |                    |  |                   |   |
| Retain Last Value                   | The input variables retains the last value.  |                    |  |                   |   |

Table 223: S&amp;R TCP General Properties

### CPU/COM

The default values of the parameters provide the fastest possible data exchange of S&R TCP data between the COM module (COM) and the processor module (CPU) within the controller. These parameters should only be changed if it is necessary to reduce the COM and CPU loads for an application, and the process allows this change.

- 
- i** Only experienced programmers should modify the parameters. Increasing the COM and CPU refresh rate means that the effective refresh rate of the S&R TCP data is also increased. The system time requirements must be verified.
- 

| Name                           | Description  |
|--------------------------------|--|
| Process Data Refresh Rate [ms] | Refresh rate in milliseconds at which the COM and CPU exchange S&R TCP protocol data. If the Refresh Rate is zero or lower than the cycle time for the controller, data is exchanged as fast as possible. Range of values: 0...( $2^{31}-1$ ), default value: 0. |
| Force Process Data Consistency | Activated:<br>Transfer of the S&R TCP data from the CPU to the COM within a CPU cycle.<br><br>Deactivated:<br>Transfer of the S&R TCP data from the CPU to the COM, distributed over multiple CPU cycles, each with 1100 bytes per data direction.               |

Table 224: Parameters of COM/CPU

## 8.4 Menu Functions for TCP Connection

### 8.4.1 Edit

The **Edit** menu function is used to open the tabs Process Variables and System Variables.

#### Process Variables

##### **Input Signals**

The *Input Signals* area contains the variables for cyclic data exchange that this controller should receive.

Any variables can be created in the *Input Signals* tab. Offsets and types of the received variables must be identical with offsets and types of the transmitted variables (send data) of the communication partner.

##### **Output Signals**

The variables for cyclic data exchange sent by this controller are entered in the *Output Signals* area.

Any variables can be created in the *Output Signals* tab. Offsets and types of the received variables must be identical with offsets and types of the transmitted variables (receive data) of the communication partner.

### 8.4.2 System Variables

Using the variables in the *System Variables* tab, the state of the TCP connection can be assessed from within the user program.

| Name                      | Description   |
|---------------------------|---|
| Bytes received            | Number of bytes received so far.  |
| Bytes sent                | Number of bytes sent so far.  |
| Error Code                | Error code of the TCP connection.<br>See Chapter 8.8.3.   |
| Error Code Timestamp [ms] | Millisecond fraction of the timestamp.<br>Point in time when the error occurred.  |
| Error Code Timestamp [s]  | Second fraction of the timestamp.<br>Point in time when the error occurred.   |
| Partner Request Timeout   | With acyclic data transfer: Timeout within which the communication partner must receive at least one time data after data sending.<br><br>0=Off<br>1... $2^{31}-1$ [ms]       |
| Partner Connection State  | If no data is received within the timeout, Partner Connection State is set to <i>Not Connected</i> and the connection is restarted.<br><br>0=No connection<br>1=Connection OK |
| Status                    | TCP connection status.<br>(see Chapter 8.8.5)   |

Table 225: System Variables

### 8.4.3 Properties

Over a TCP connection, data is exchanged cyclically or acyclically. The S&R TCP function blocks are required for the acyclic data exchange. The S&R TCP function blocks cannot be used for cyclic data exchange.

| Name               | Description   |
|--------------------|---|
| Type               | TCP connection  |
| Name               | Any unique name for one TCP connection. A maximum of 31 characters.   |
| ID                 | Any unique identification number (ID) for each TCP connection. The ID is also required as a reference for the S&R TCP function blocks.<br>Range of values: 0..255<br>Default value: 0   |
| Mode               | <p>Server (default value):<br/>This station operates as a server (passive mode). The connection is established by the communication partner (client). Once the connection has been established, both communication partners are equal and can send data at any time.<br/>The own port must be specified.</p> <p>Server with defined partner:<br/>This station operates as a server (passive mode). The connection is established by the communication partner (client). Once the connection has been established, both communication partners are equal and can send data at any time.<br/>If the IP address and/or port of the communication partner are defined here, only the specified communication partner can connect to the server. All other stations are ignored.<br/>If one of the parameters (IP address or port) is set to zero, the parameter is not verified.</p> <p>Client:<br/>This station operates as a client, i.e., the station establishes the connection to the communication partner.<br/>IP address and port of the communication partner must be specified.<br/>Also an own port can optionally be defined.</p> |
| Partner IP address | IP address of the communication partner.<br>0.0.0.0: any IP address is permitted.<br>Valid range: 1.0.0.0...223.255.255.255,<br>Except for: 127.x.x.x<br>Default value: 0   |
| Partner port       | Port of the communication partner.<br>0: Any port<br>Ports that are reserved or already used (1...1024), are rejected by the COM OS.<br>Range of values: 0 ... 65535<br>Default value: 0  |

|                              |   |
|------------------------------|---|
| Own Port                     | <p>Own port.<br/>0: Any port<br/>Ports that are reserved or already used (1...1024), are rejected by the COM OS.<br/>Range of values: 0...65535<br/>Default value: 0</p>  |
| Cyclic data transfer         | <p>Deactivated (default value)<br/>Cyclic data transfer is deactivated.<br/>Function blocks must be used to program the data exchange over this TCP connection.<br/>No cyclic E/A data may be defined on this connection.</p>   |
|                              | <p>Activated:<br/>Cyclic data transfer is active.<br/>Data is defined in the Process Variable dialog box for the TCP connection.<br/>Receive data must be defined.<br/>No function blocks can be used on this connection.</p>   |
| Send Interval [ms]           | <p>Only editable with cyclic data transfer.<br/>The send interval is set here.<br/>Range of values 10...2147483647 ms (lower values are rounded to 10 ms)<br/>Default value: 0</p>  |
| Keep Alive Interval [s]      | <p>Time period until the connection monitoring provided by the TCP is activated.<br/>Zero deactivates the connection monitoring.<br/>If no data is exchanged within the specified KeepAlive interval, the KeepAlive samples are sent to the communication partner. If the connection still exists, the KeepAlive samples are acknowledged by the communication partner.<br/>If no data is exchanged between the partners within a period of &gt; 10 KeepAlive interval, the connection is closed.<br/>If no response is received after a data packet sending, the data packet is resent in predefined intervals. The connection is aborted after 12 unsuccessful resends (approx. 7 minutes).<br/>Range of values 1...65535s<br/>Default value: 0 = deactivated</p> |
| Partner Request Timeout [ms] | <p>With acyclic data transfer: Timeout within which the communication partner must receive at least one time data after data sending. If no data is received within the timeout, <i>Partner connection state</i> is set to <i>not connected</i> and the connection is restarted.<br/>After a timeout or another error closed the connection, the active side re-establishes the connection with a delay of 10 x PartnerRequestTimeout or a delay of 10 seconds if PartnerRequestTimeout is equal to 0. The passive side opens the port within half of this time.<br/>0=Off<br/>Range of values: 1...<math>2^{31}-1</math> [ms]<br/>Default value: 0</p>   |

Table 226: S&amp;R TCP Connection Properties

## 8.5 Data exchange

S&R TCP operates according to the client/server principle. The connection is established by the communication partner which is configured as a client. Once the connection has been established, both communication partners are equal and can send data at any point in time.

S&R TCP does not have its own data protection protocol; rather, it uses TCP/IP directly. As the data sent by the TCP are arranged in a data stream, it must be ensured that offsets and types of the variables to be exchanged on the receiving station are identical with the ones on the sending station.

S&R TCP is compatible with the Siemens SEND/RECEIVE interface and allows cyclical data exchange with the Siemens S7 function blocks AG\_SEND (FC5) and AG\_RECV (FC6) (see Chapter 8.2, Example of S&R TCP Configuration).

Further, HIMA provides five S&R TCP function blocks for controlling and individually configuring communication using the user program. With the S&R TCP function blocks, any arbitrary protocol transferred over TCP (e.g., Modbus) can be sent and received.

### 8.5.1 TCP Connections

For each connection to a communication partner over S&R TCP, at least one TCP connection must exist in the HIMax controller.

The identification number of the TCP connection and the addresses/ports of the own controller and of the communication partner's controller must be set in the *Properties* dialog box for the TCP connection.

A maximum of 32 TCP connections can be established in a HIMax controller.

These TCP connections must have different identification numbers and different addresses/ports.

#### To create a new TCP connection

1. In the structure tree, open **Configuration, Resource**.
2. Right-click **Protocols**, then click **New**.
3. Select **Send/Receive over TCP** and enter a name for the protocol.
4. Click **OK** to create a new protocol.
5. Right-click **Send/Receive over TCP**, then click **Properties**.
6. Select **COM Module**. The remaining parameters retain the default values.

---

**TIP**

The HIMax/HIMatrix controller and the third-party system must be located in the same subnet or must have the corresponding routing settings if a router is used.

---

### 8.5.2 Cyclic Data Exchange

If data is exchanged cyclically, a send interval must be defined in the HIMax/HIMatrix controller and in the communication partner.

The send interval defines the cyclic time period within which the sending communication partner sends the variables to the receiving communication partner.

- To ensure a continuous data exchange, both communication partners should define almost the same send interval (see Chapter 8.5.5, Flow Control).
- For cyclic data exchange, the *Cyclic Data Transfer* option must be activated in the TCP connection in use.
- If the *Cyclic Data Transfer* option is activated in a TCP connection, no function blocks may be used.
- The variables to be sent and received are assigned in the *Process Variable* dialog box for the TCP connection. Receive data **must** exist, send data is optional.

- 
- i** The same variables (same offsets and types) that are defined as send data in a station, must be defined as receive data in the other station.
- 

### 8.5.3 Acyclic Data Exchange with Function Blocks

In HIMax/HIMatrix controllers, the acyclic data exchange is controlled by the user program via the TCP S&R function blocks. Data exchange can thus be controlled using a timer or a mechanical switch connected to a physical input of the HIMax/HIMatrix controller.

- The *Cyclic Data Transfer* option must be deactivated in the TCP connection in use.
- Only one S&R TCP function block may send at any given time.
- The variables to be sent or received are assigned in the *Process Variables* dialog box for the S&R TCP function blocks (all except for *Reset*).

- 
- i** The same variables (same offsets and types) that are defined as send data in a station, must be defined as receive data in the other station.
- 

### 8.5.4 Simultaneous Cyclic and Acyclic Data Exchange

For this purpose, one TCP connection must be configured for cyclic data and one TCP connection for acyclic data. The two TCP connections must use different partner IP addresses and partner ports.

One individual TCP connection cannot be simultaneously used for cyclic and acyclic data exchange.

### 8.5.5 Flow Control

The flow control is a component of the TCP and monitors the continuous data traffic between two communication partners.

With cyclic data transfer, at least one packet must be received after a maximum of 3 to 5 packets have been sent; otherwise, transmission is blocked until a packet is received or the connection monitoring process terminates the connection.

The number (3...5) of potential transmissions without packet reception depends on the size of the packets to be sent.

Number=5 for small packets < 4kB.

Number=3 for big packets  $\geq$  4kB.

- While planning the project, it must be ensured that no station sends more data than the other station can simultaneously process.
- To ensure a cyclical data exchange, both communication partners should define almost the same send interval

## 8.6 Third-Party Systems with Pad Bytes

During cyclic and acyclic data exchange, note that some controllers (e.g., SIMATIC 300) add so-called *pad bytes*. Pad bytes ensure that all data types exceeding one byte always begin at an even offset and that the total size of the packets (in bytes) is also even.

In the HIMax/HIMatrix controller, dummy bytes must be added in place of pad bytes in the corresponding positions.

| Address | Name    | Type       | Initial value |
|---------|---------|------------|---------------|
| 0.0     |         | STRUCT     |               |
| +0.0    | InOut_1 | BYTE       | B#16#0        |
| +2.0    | InOut_3 | WORD       | W#16#0        |
| =4.0    |         | END_STRUCT |               |

Figure 71: Siemens List of Variables

In the Siemens controller, a *pad byte* is added (not visible) such that the *InOut\_3* variable begins at an even offset.

| Output Signals |         |           |        |   |                 |
|----------------|---------|-----------|--------|---|-----------------|
| F              | Name    | Data type | Offset | ▼ | Global Variable |
| 1              | InOut_1 | BYTE      | 0      |   | InOut_1         |
| 2              | Dummy   | BYTE      | 1      |   | Dummy           |
| 3              | InOut_3 | WORD      | 2      |   | InOut_3         |

Figure 72: HIMax/HIMatrix List of Variables

In the HIMax/HIMatrix controller, a *dummy byte* must be added such that the variable *InOut\_3* has the same offset as in the Siemens controller.

## 8.7 S&R TCP Function Blocks

If the cyclic data transfer is not sufficient flexible, data can also be sent and received using the S&R TCP function blocks. The *Cyclic Data Transfer* option must be deactivated in the TCP connection in use.

The S&R TCP function blocks is used to tailor the data transfer over TCP/IP to best meet the project requirements.

The function blocks are configured in the user program. The functions (Send, Receive, Reset) of the HIMax/HIMatrix controller can thus be set and evaluated in the user program.

S&R TCP function blocks are required for the acyclic data exchange. These function blocks are not required for the cyclic data exchange between server and client.



The configuration of the S&R TCP function blocks is described in Chapter 14.1.

---

The following function blocks are available:

| Function block                         | Function description   |
|--|--|
| TCP_Reset<br>(see Chapter 8.7.1)       | TCP connection reset   |
| TCP_Send<br>(see Chapter 8.7.2)        | Sending of data  |
| TCP_Receive<br>(see Chapter 8.7.3)     | Reception of data packets with fixed length                        |
| TCP_ReceiveLine<br>(see Chapter 8.7.4) | Reception of an ASCII line   |
| TCP_ReceiveVar<br>(see Chapter 8.7.5)  | Reception of data packets with variable length (with length field) |
| LATCH                                  | Only used within other function blocks                             |
| PIG                                    | Only used within other function blocks                             |
| PIGII                                  | Only used within other function blocks                             |

Table 227: Function Blocks for S&R TCP Connections

### 8.7.1 TCP\_Reset

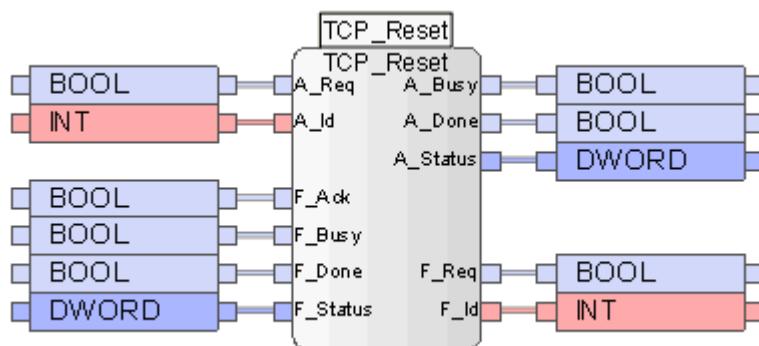


Figure 73: Function Block TCP\_Reset

The **TCP Reset** function block is used to re-establish a disturbed connection if a send or receive function block reports a timeout error (16#8A).

- To configure the function block, drag it from the function block library onto the user program (see also Chapter 14.1).

#### Inputs and Outputs of the Function Block with Prefix A:

These inputs and outputs can be used to control and evaluate the function block using the user program. The prefix A means Application.

| A-Inputs | Description  | Type |
|----------|--|------|
| A_Req    | Rising edge starts the function block  | BOOL |
| A_Id     | Identification number ( <i>ID</i> ) of the disturbed TCP connection to be reset. | INT  |

Table 228: A-Inputs for the TCP\_Reset Function Block

| A_Outputs | Description   | Type  |
|-----------|---|-------|
| A_Busy    | TRUE: The TCP connection is still being reset.  | BOOL  |
| DONE      | TRUE: The data transmission ended without error.  | BOOL  |
| A_Status  | The status and error code of the function block and of the TCP connection are output on <i>A_Status</i> . | DWORD |

Table 229: A-Outputs for the TCP\_Reset Function Block

### Inputs and Outputs of the Function Block with Prefix F:

These inputs and outputs of the function block establish the connection to the **Reset** function block in structure tree. The prefix F means Field.

- 
- i** Common variables are used to connect the **Reset** function block (in the Function Blocks directory) to the **TCP\_Reset** function block (in the user program). These must be created beforehand using the Global Variable Editor.
- 

Connect the *F-Inputs* of the **TCP\_Reset** function block in the user program to the same variables that will be connected to the outputs of the **Reset** function block in the structure tree.

| F-Inputs | Type  |
|----------|-------|
| F_Ack    | BOOL  |
| F_Busy   | BOOL  |
| F_Done   | BOOL  |
| F_Status | DWORD |

Table 230: F-Inputs for the TCP\_Reset Function Block

Connect the *F-Outputs* of the **TCP\_Reset** function block in the user program to the same variables that will be connected to the outputs of the **Reset** function block in the structure tree.

| F-Outputs | Type  |
|-----------|-------|
| F_Req     | BOOL  |
| F_Id      | DWORD |

Table 231: F-Outputs for the TCP\_Reset Function Block

### To create the Reset function block in the structure tree

1. In the structure tree, open **Configuration, Resource, Protocols, Send/Receive over TCP, Function Blocks, New.**
2. Select the **Reset** function block and click **OK**.
3. Right-click the **Reset** function block, and then click **Edit**.
  - The window for assigning variables to the function blocks appears.

Connect the inputs of the **Reset** function block in the structure tree to the same variables that have been previously connected to the *F-Outputs* of the **TCP\_Reset** function block in the user program.

| Inputs | Type  |
|--------|-------|
| ID     | DWORD |
| REQ    | BOOL  |

Table 232: Input System Variables

Connect the outputs of the **Reset** function block in the structure tree to the same variables that have been previously connected to the *F-Inputs* of the **TCP\_Reset** function block in the user program.

| Outputs | Type  |
|---------|-------|
| ACK     | BOOL  |
| BUSY    | BOOL  |
| DONE    | BOOL  |
| STATUS  | DWORD |

Table 233: Output System Variables

**To operate the TCP\_Reset function block, the following steps are essential:**

1. In the user program, set the identification number for the disturbed TCP connection on the *A\_ID* input.
2. In the user program, set the *A\_Req* input to TRUE.



The function block reacts to a rising edge on *A\_Req*.

---

The *A\_Busy* output is set to TRUE until a reset is sent to the specified TCP connection. Afterwards, *A\_Busy* is set to FALSE and *A\_Done* is set to TRUE.

### 8.7.2 TCP\_Send

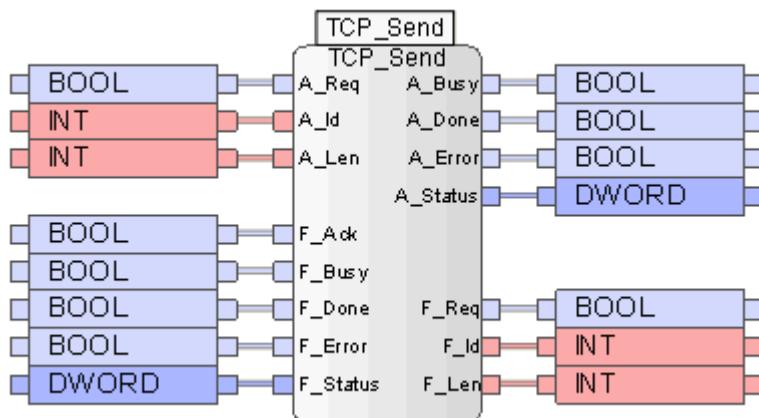


Figure 74: Function Block TCP\_Send

The **TCP\_Send** function block is used for acyclically send variables to a communication partner. A function block with the same variables and offsets, e.g., *Receive*, must be configured in the communication partner.

- To configure the function block, drag it from the function block library onto the user program (see also Chapter 14.1).

#### Inputs and Outputs of the Function Block with Prefix A:

These inputs and outputs can be used to control and evaluate the function block using the user program. The prefix A means Application.

| A-Inputs | Description   | Type |
|----------|---|------|
| A_Req    | The rising edge starts the function block.  | BOOL |
| A_Id     | Identification number of the configured TCP connection to the communication partner to which data should be sent.           | INT  |
| A_Len    | Number of transmitted variables, expressed in bytes.<br>A_Len must be greater than zero and must not end within a variable. | INT  |

Table 234: A-Inputs for the TCP\_Send Function Block

| A_Outputs | Description   | Type  |
|-----------|---|-------|
| A_Busy    | TRUE: Data is still being transmitted.  | BOOL  |
| DONE      | TRUE: The data transmission ended without error.  | BOOL  |
| ERROR     | TRUE: An error occurred<br>FALSE: No error  | BOOL  |
| A_Status  | The status and error code of the function block and of the TCP connection are output on A_Status. | DWORD |

Table 235: A-Outputs for the TCP\_Send Function Block

#### Inputs and Outputs of the Function Block with Prefix F:

These inputs and outputs of the function block establish the connection to the **Send** function block in structure tree. The prefix F means Field.

- 
- i** Common variables are used to connect the **Send** function block (in the Function Blocks directory) to the **TCP\_Send** function block (in the user program). These must be created beforehand using the Variable Editor.
- 

Connect the *F-Inputs* of the **TCP\_Send** function block in the user program to the same variables that will be connected to the outputs of the **Send** function block in the structure tree.

| F-Inputs | Type  |
|----------|-------|
| F_Ack    | BOOL  |
| F_Busy   | BOOL  |
| F_Done   | BOOL  |
| F_Error  | BOOL  |
| F_Status | DWORD |

Table 236: F-Inputs for the TCP\_Send Function Block

Connect the *F-Outputs* of the **TCP\_Send** function block in the user program to the same variables that will be connected to the inputs of the **Send** function block in the structure tree.

| F-Outputs | Type  |
|-----------|-------|
| F_Id      | DWORD |
| F_Len     | INT   |
| F_Req     | BOOL  |

Table 237: F-Outputs for the TCP\_Send Function Block

#### To create the Send function block in the structure tree

1. In the structure tree, open **Configuration, Resource, Protocols, Send/Receive over TCP, Function Blocks, New.**
2. Select the **Send** function block and click **OK**.
3. Right-click the **Send** function block, and then click **Edit**.
  - The window for assigning variables to the function blocks appears.

Connect the outputs of the **Send** function block in the structure tree to the same variables that have been previously connected to the *F-Outputs* of the **TCP\_Send** function block in the user program.

| Inputs | Type  |
|--------|-------|
| ID     | DWORD |
| LEN    | INT   |
| REQ    | BOOL  |

Table 238: Input System Variables

Connect the outputs of the **Send** function block in the structure tree to the same variables that have been previously connected to the *F-Inputs* of the **TCP\_Send** function block in the user program.

| Outputs | Type  |
|---------|-------|
| Ack     | BOOL  |
| Busy    | BOOL  |
| Done    | BOOL  |
| ERROR   | BOOL  |
| STATUS  | DWORD |

Table 239: Output System Variables

| Data      | Description   |
|-----------|---|
| Send data | Any variables can be created in the <i>Process Variables</i> tab. Offsets and types of the received variables must be identical with offsets and types of the transmitted variables of the communication partner. |

Table 240: Send Data

**The following steps are essential to operate the TCP\_Send function block:**

- i The send variables must be created in the *Process Variables* tab of the *Send* dialog box. Offsets and types of the received variables must be identical with offsets and types of the transmitted variables of the communication partner.

1. In the user program, set the identification number of the TCP connection on the *A\_ID* input.
2. In the user program, set the expected length (in bytes) of the variables to be sent on the *A\_Len* input.
3. In the user program, set the *A\_Req* input to TRUE.

- i The function block reacts to a rising edge on *A\_Req*.

The *A\_Busy* output is set to TRUE until the variables have been sent. Afterwards, *A\_Busy* is set to FALSE and *A\_Done* is set to TRUE.

If the sending process was not successful, the *A\_Error* output is set to TRUE and an error code is output to *A\_Status*.

### 8.7.3 TCP\_Receive

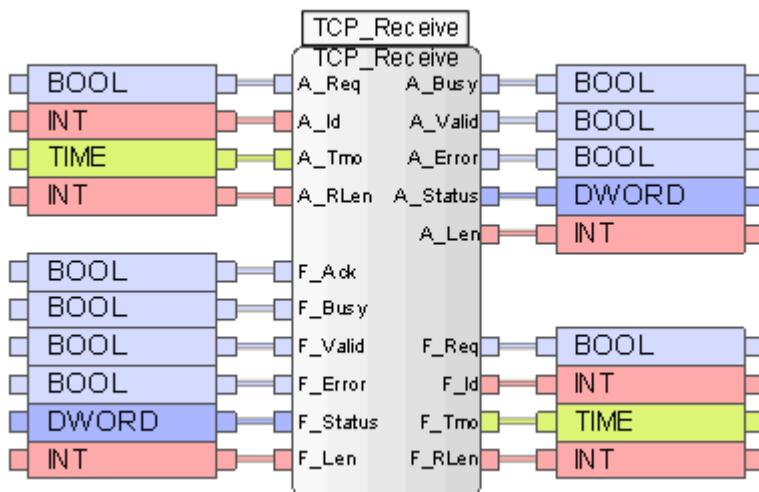


Figure 75: Function Block TCP\_Receive

The **TCP\_Receive** function block is used to receive predefined variables from the communication partner.

A function block with the same variables and offsets, e.g., **TCP\_Send**, must be configured in the communication partner.

- 
- i** To configure the function block, drag it from the function block library onto the user program (see also Chapter 14.1).
- 

#### Inputs and Outputs of the Function Block with Prefix A:

These inputs and outputs can be used to control and evaluate the function block using the user program. The prefix A means Application.

| A-Inputs | Description  | Type |
|----------|--|------|
| A_Req    | The rising edge starts the function block.   | BOOL |
| A_Id     | Identification number of the configured TCP connection to the communication partner from which data should be received.  | INT  |
| A_Tmo    | Receive timeout<br>If no data are received within the timeout, the function block stops and an error message appears. If the A_Tmo input is not used or set to zero, the timeout is deactivated. | TIME |
| A_RLen   | A_RLen is the expected length of the variables to be received, expressed in bytes.<br>A_RLen must be greater than zero and must not end within a variable.                                       | INT  |

Table 241: A-Inputs for the TCP\_Receive Function Block

| A_Outputs | Description   | Type  |
|-----------|---|-------|
| A_Busy    | TRUE: Data is still being received.   | BOOL  |
| A_Valid   | TRUE: The data reception ended without error.   | BOOL  |
| ERROR     | TRUE: An error occurred<br>FALSE: No error  | BOOL  |
| A_Status  | The status and error code of the function block and of the TCP connection are output on A_Status. | DWORD |
| A_Len     | Number of received bytes.   | INT   |

Table 242: A-Outputs for the TCP\_Receive Function Block

**Inputs and Outputs of the Function Block with Prefix F:**

These inputs and outputs of the function block establish the connection to the **Receive** function block in structure tree. The prefix F means Field.

- 
- i** Common variables are used to connect the **Receive** function block (in the Function Blocks directory) to the **TCP\_Receive** function block (in the user program). These must be created beforehand using the Variable Editor.
- 

Connect the *F-Inputs* of the **TCP\_Receive** function block in the user program to the same variables that will be connected to the outputs of the **Receive** function block in the structure tree.

| F-Inputs | Type  |
|----------|-------|
| F_Ack    | BOOL  |
| F_Busy   | BOOL  |
| F_Valid  | BOOL  |
| F_Error  | BOOL  |
| F_Status | DWORD |
| F_Len    | INT   |

Table 243: A-Inputs for the TCP\_Receive Function Block

Connect the *F-Outputs* of the **TCP\_Receive** function block in the user program to the same variables that will be connected to the inputs of the **Receive** function block in the structure tree.

| F-Outputs | Type  |
|-----------|-------|
| F_Req     | BOOL  |
| F_Id      | DWORD |
| F_Tmo     | INT   |
| F_RLen    | INT   |

Table 244: F-Outputs for the TCP\_Receive Function Block

**To create the corresponding Receive function block in the structure tree:**

1. In the structure tree, open **Configuration, Resource, Protocols, Send/Receive over TCP, Function Blocks, New.**
2. Select the **Receive** function block and click **OK**.
3. Right-click the **Receive** function block, and then click **Edit**.  
 The window for assigning variables to the function blocks appears.

Connect the inputs of the **Receive** function block in the structure tree to the same variables that have been previously connected to the *F-Outputs* of the **TCP\_Receive** function block in the user program.

| Inputs  | Type |
|---------|------|
| ID      | INT  |
| REQ     | BOOL |
| RLEN    | INT  |
| TIMEOUT | TIME |

Table 245: Input System Variables

Connect the outputs of the **Receive** function block in the structure tree to the same variables that have been previously connected to the *F-Inputs* of the **TCP\_Receive** function block in the user program.

| Outputs | Type  |
|---------|-------|
| Ack     | BOOL  |
| Busy    | BOOL  |
| ERROR   | BOOL  |
| LEN     | INT   |
| STATUS  | DWORD |
| VALID   | BOOL  |

Table 246: Output System Variables

| Data              | Description   |
|-------------------|---|
| Receive variables | Any variables can be created in the <i>Process Variables</i> tab. Offsets and types of the received variables must be identical with offsets and types of the transmitted variables of the communication partner. |

Table 247: Receive Variables

To operate the TCP\_Receive function block, the following steps are essential:

- 
- i The receive variables must be created in the *Process Variables* tab located in the *Receive* dialog box. Offsets and types of the receive variables must be identical with offsets and types of the send variables of the communication partner.
- 

1. In the user program, set the identification number for the TCP connection on the *A\_ID* input.
2. In the user program, set the receive timeout on the *A\_Tmo* input.
3. In the user program, set the expected length of the variables to be received on the *A\_RLen* input.
4. In the user program, set the *A\_Req* input to TRUE.

- 
- i The function block starts with a rising edge on *A\_Req*.
- 

The *A\_Busy* output is set to TRUE until the variables have been received or the receive timeout has expired. Afterwards, *A\_Busy* is set to FALSE and *A\_Valid* or *A\_Error* to TRUE.

If no error occurred during the variable reception, the *A\_Valid* output is set to TRUE. The variables defined in the *Data* tab can be evaluated.

If an error occurred during the variable reception, the *A\_Error* output is set to TRUE and an error is output to *A\_Status*.

### 8.7.4 TCP\_ReceiveLine

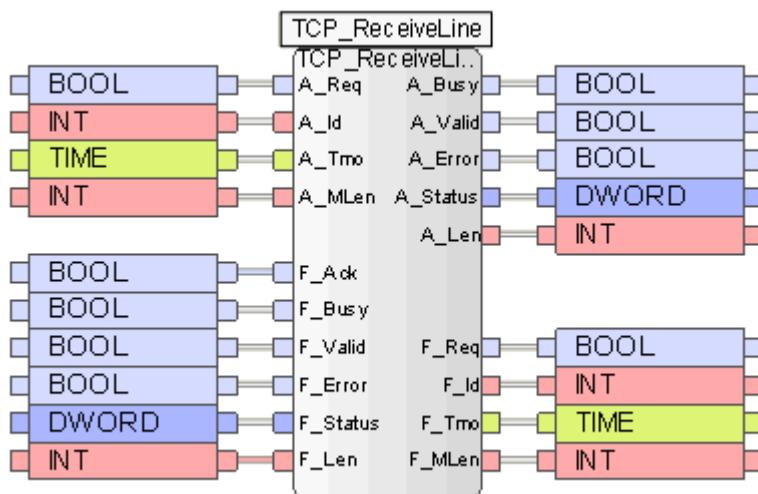


Figure 76: Function Block TCP\_ReceiveLine

The **TCP\_ReceiveLine** function block is used for receiving an ASCII character string with LineFeed (16#0A) from a communication partner.

- i** To configure the function block, drag it from the function block library onto the user program (see also Chapter 14.1).

#### Inputs and Outputs of the Function Block with Prefix A:

These inputs and outputs can be used to control and evaluate the function block using the user program. The prefix A means Application.

| A-Inputs | Description   | Type |
|----------|---|------|
| A_Req    | Rising edge starts the function block.  | BOOL |
| A_Id     | Identification number of the configured TCP connection to the communication partner from which data should be received.   | INT  |
| A_Tmo    | Receive timeout<br>If no data are received within the timeout, the function block stops and an error message appears. If the input is not used or set to zero, the timeout is deactivated.  | TIME |
| A_MLen   | Maximum length of a line to be received, expressed in bytes.<br>The receive variables must be created in the <i>Data</i> tab located in the COM function block.<br>Transmitted bytes = Min (A_MLen, line length, length of the data range). | INT  |

Table 248: A-Inputs for the TCP\_ReceiveLine Function Block

| A_Outputs | Description   | Type  |
|-----------|---|-------|
| A_Busy    | TRUE: Data is still being received.   | BOOL  |
| A_Valid   | TRUE: The data reception ended without error.   | BOOL  |
| ERROR     | TRUE: An error occurred<br>FALSE: No error  | BOOL  |
| A_Status  | The status and error code of the function block and of the TCP connection are output on A_Status. | DWORD |
| A_Len     | Number of received bytes.   | INT   |

Table 249: A-Outputs for the TCP\_ReceiveLine Function Block

**Inputs and Outputs of the Function Block with Prefix F:**

These inputs and outputs of the function block establish the connection to the **ReceiveLine** function block in structure tree. The prefix F means Field.



Common variables are used to connect the **ReceiveLine** function block in the structure tree (located in the Function Blocks directory) to the **TCP\_ReceiveLine** function block (in the user program). These must be created beforehand using the Variable Editor.

Connect the *F-Inputs* of the **TCP\_ReceiveLine** function block in the user program to the same variables that will be connected to the outputs of the **ReceiveLine** function block in the structure tree.

| F-Inputs | Type  |
|----------|-------|
| F_Ack    | BOOL  |
| F_Busy   | BOOL  |
| F_Valid  | BOOL  |
| F_Error  | BOOL  |
| F_Status | DWORD |
| F_Len    | INT   |

Table 250: F-Inputs for the TCP\_ReceiveLine Function Block

Connect the *F-Outputs* of the **TCP\_ReceiveLine** function block in the user program to the same variables that will be connected to the inputs of the **ReceiveLine** function block in the structure tree.

| F-Outputs | Type |
|-----------|------|
| A_Req     | BOOL |
| A_Id      | INT  |
| A_Tmo     | TIME |
| A_MLen    | INT  |

Table 251: A-Outputs for the TCP\_ReceiveLine Function Block

**To create the corresponding ReceiveLine function block in the structure tree**

1. In the structure tree, open **Configuration, Resource, Protocols, Send/Receive over TCP, Function Blocks, New**.
2. Select the **ReceiveLine** function block and click **OK**.
3. Right-click the **ReceiveLine** function block, and then click **Edit**.  
 The window for assigning variables to the function blocks appears.

Connect the outputs of the **ReceiveLine** function block in the structure tree to the same variables that have been previously connected to the *F-Outputs* of the **TCP\_ReceiveLine** function block in the user program.

| Inputs  | Type |
|---------|------|
| ID      | INT  |
| MLEN    | INT  |
| REQ     | BOOL |
| TIMEOUT | TIME |

Table 252: Input System Variables

Connect the outputs of the **ReceiveLine** function block in the structure tree to the same variables that have been previously connected to the *F-Inputs* of the **TCP\_ReceiveLine** function block in the user program.

| Outputs | Type  |
|---------|-------|
| ACK     | BOOL  |
| BUSY    | BOOL  |
| ERROR   | BOOL  |
| LEN     | INT   |
| STATUS  | DWORD |
| VALID   | BOOL  |

Table 253: Output System Variables

| Data              | Description   |
|-------------------|---|
| Receive variables | The <i>Process Variables</i> tab should only contain variables of type BYTE. Offsets of the variables must be identical with offsets of the variables of the communication partner. |

Table 254: Receive Variables

To operate the **TCP\_ReceiveLine** function block, the following steps are essential:

- 
- i** The receive variables of type BYTE must be created in the tab *Process Variables* located in the *ReceiveLine* dialog box. Offsets of the receive variables must be identical with offsets of the send variables of the communication partner.
- 

1. In the user program, set the identification number for the TCP connection on the *A\_ID* input.
  2. In the user program, set the receive timeout on the *A\_Tmo* input.
  3. In the user program, set the maximum length of the line to be received on the *A\_MLen* input.
- 

- i** *A\_Mlen* must be greater than zero and determines the size of the receive buffer in bytes.  
If the receive buffer is full and a line end has not yet occurred, the reading process ends and no error message appears.  
The number of received bytes is output to the *A\_Len* output:  
Received bytes = Min (A\_MLen, line length, length of the data range).
- 

4. In the user program, set the *A\_Req* input to TRUE.
- 

- i** The function block reacts to a rising edge on *A\_Req*.
- 

The *A\_Busy* output is set to TRUE if the receive buffer is full or the end of line *LineFeed* is received or the receive time-put has expired. Afterwards, *A\_Busy* is set to FALSE and *A\_Valid* or *A\_Error* to TRUE.

If no error occurred during the line reception, the *A\_Valid* output is set to TRUE. The variables defined in the Data tab can be evaluated.

If an error occurred during the line reception, the *A\_Error* output is set to TRUE and an error is output to *A\_Status*.

### 8.7.5 TCP\_ReceiveVar

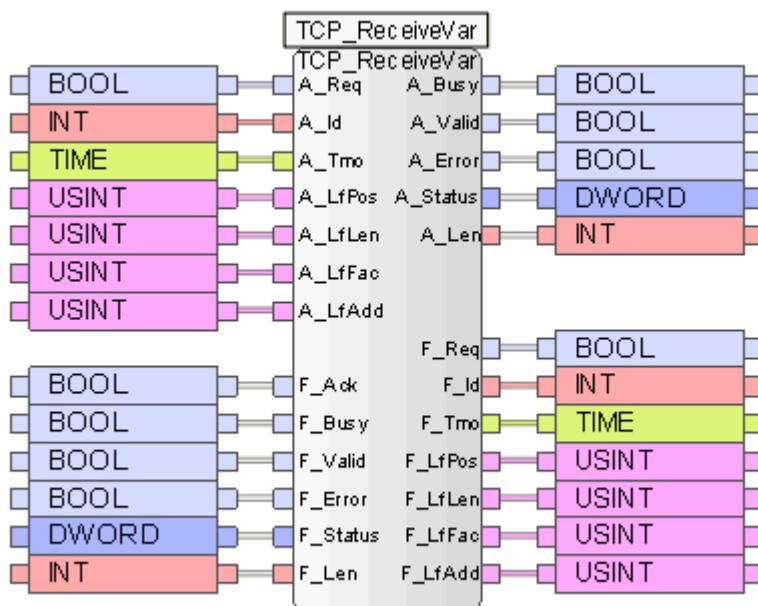


Figure 77: Function Block TCP\_ReceiveVar

The **TCP\_ReceiveVar** function block is used to evaluate data packets with variable length and containing a length field.

- i To configure the function block, drag it from the function block library onto the user program (see also Chapter 14.1).

#### Functional Description

The received data packets must have the structure represented in the figure below (e.g., Modbus protocol). Modifying the input parameters *A\_LfPos*, *A\_LfLen*, *A\_LfFac*, *A\_LfLen*, the received data packets can be adapt to any protocol format.

The received data packet is composed of a header and a data range. The header contains data such as subscriber address, telegram function, length field etc. required for establishing communication. To evaluate the data range, separate the header and read the length field.

The size of the header is entered in the *A\_LfAdd* parameter.

The length of the data range must be read from the length field of the data packet currently read. The position of the length field is entered in *A\_LfPos*. The size of the length field expressed in bytes is entered in *LfLen*. If the length is not expressed in bytes, the corresponding conversion factor must be entered in *A\_LfFac* (e.g., 2 for WORD or 4 for DOUBLE WORD).

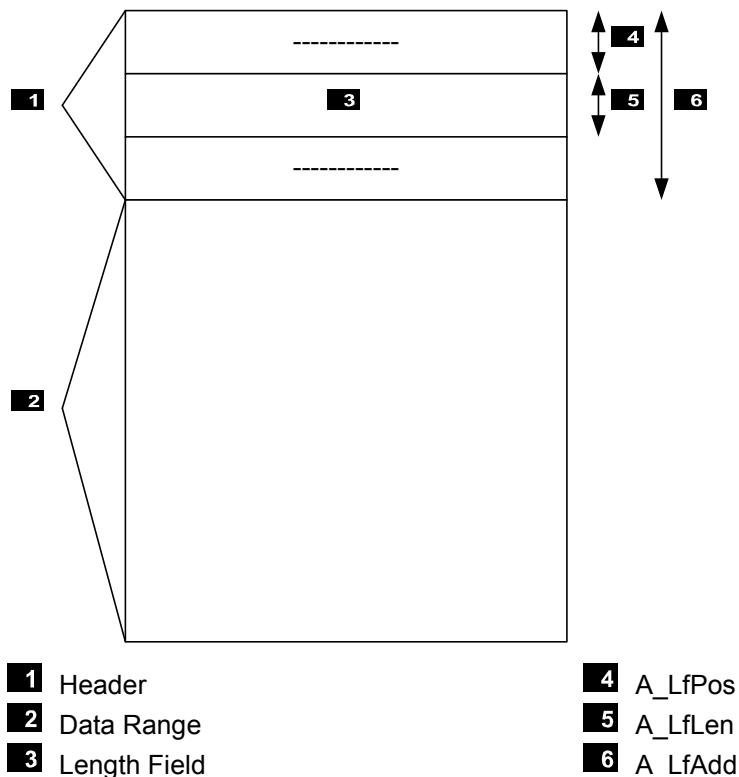


Figure 78: Data Packet Structure

### Inputs and Outputs of the Function Block with Prefix A

These inputs and outputs can be used to control and evaluate the function block using the user program. The prefix A means Application.

| A-Inputs | Description  | Type  |
|----------|--|-------|
| A_Req    | Rising edge starts the function block.   | BOOL  |
| A_Id     | Identification number ( <i>ID</i> ) of the configured TCP connection to the communication partner from which the data should be received.  | DWORD |
| A_Tmo    | Receive timeout<br>If no data are received within the timeout, the function block stops and an error message appears. If the input is not used or set to zero, the timeout is deactivated. | INT   |
| A_LfPos  | Start position of the length field in the data packet; the numbering begins with zero.<br>(measured in bytes)  | USINT |
| A_LfLen  | Size of the <i>A_LfLen</i> length field in bytes.<br>Permitted: 1, 2 or 4 bytes.   | USINT |
| A_LfFac  | Conversion factor in bytes if the value set in the length field is not expressed in bytes. If the input is not used or set to zero, 1 is used as default value.                            | USINT |
| A_LfAdd  | Size of the header in bytes.   | USINT |

Table 255: A-Inputs for the TCP\_ReceiveVar Function Block

| A-Outputs | Description   | Type  |
|-----------|---|-------|
| A_Busy    | TRUE: Data is still being received.   | BOOL  |
| A_Valid   | TRUE: The data reception ended without error.   | BOOL  |
| ERROR     | TRUE: An error occurred during the reading process<br>FALSE: No error                             | BOOL  |
| A_Status  | The status and error code of the function block and of the TCP connection are output on A_Status. | DWORD |
| A_Len     | Number of received bytes.   | INT   |

Table 256: A-Outputs for the TCP\_ReceiveVar Function Block

**Inputs and Outputs of the Function Block with Prefix F:**

These inputs and outputs of the function block establish the connection to the **ReceiveVar** function block in structure tree. The prefix F means Field.

- i** Common variables are used to connect the **ReceiveVar** function block in the structure tree (located in the Function Blocks directory) to the **TCP\_ReceiveVar** function block (in the user program). These must be created beforehand using the Variable Editor.

Connect the *F-Inputs* of the **TCP\_ReceiveVar** function block in the user program to the same variables that will be connected to the outputs of the **ReceiveVar** function block in the structure tree.

| F-Inputs | Type  |
|----------|-------|
| F_Ack    | BOOL  |
| F_Busy   | BOOL  |
| F_Valid  | BOOL  |
| F_Error  | BOOL  |
| A_Status | DWORD |
| A_Len    | INT   |

Table 257: F-Inputs for the TCP\_ReceiveVar Function Block

Connect the *F-Outputs* of the **TCP\_ReceiveVar** function block in the user program to the same variables that will be connected to the inputs of the **ReceiveVar** function block in the structure tree.

| F-Outputs | Type  |
|-----------|-------|
| F_Req     | BOOL  |
| F_Id      | INT   |
| F_Tmo     | TIME  |
| F_LfPos   | USINT |
| A_LfLen   | USINT |
| A_LfFac   | USINT |
| A_LfAdd   | USINT |

Table 258: F-Outputs for the TCP\_ReceiveVar Function Block

**To create the ReceiveVar function block in the structure tree**

1. In the structure tree, open **Configuration, Resource, Protocols, Send/Receive over TCP, Function Blocks, New**.
2. Select the **ReceiveVar** function block and click **OK**.
3. Right-click the **Receive** function block, and then click **Edit**.
  - The window for assigning variables to the function blocks appears.

Connect the inputs of the **ReceiveVar** function block in the structure tree to the same variables that have been previously connected to the *F-Outputs* of the **TCP\_ReceiveVar** function block in the user program.

| Inputs  | Type  |
|---------|-------|
| ID      | INT   |
| Lf Add  | USINT |
| Lf Fac  | USINT |
| Lf Len  | USINT |
| Lf Pos  | USINT |
| REQ     | BOOL  |
| TIMEOUT | TIME  |

Table 259: Input System Variables

Connect the inputs of the **ReceiveVar** function block in the structure tree to the same variables that have been previously connected to the *F-Inputs* of the **TCP\_ReceiveVar** function block in the user program.

| Outputs | Type  |
|---------|-------|
| ACK     | BOOL  |
| BUSY    | BOOL  |
| ERROR   | BOOL  |
| LEN     | INT   |
| STATUS  | DWORD |
| VALID   | BOOL  |

Table 260: Output System Variables

| Data              | Description   |
|-------------------|---|
| Receive variables | Any variables can be created in the <i>Process Variables</i> tab. Offsets and types of the received variables must be identical with offsets and types of the transmitted variables of the communication partner. |

Table 261: Receive Variables

To operate the **TCP\_ReceiveVar** function block, the following steps are essential:

- 
- i** The receive variables must be created in the *Process Variables* tab located in the *Variables* dialog box. Offsets and types of the receive variables must be identical with offsets and types of the send variables of the communication partner.
- 

1. In the user program, set the identification number for the TCP connection on the *A\_ID* input.
  2. In the user program, set the receive timeout on the *A\_Tmo* input.
  3. In the user program, set the parameters *A\_LfPos*, *A\_LfLen*, *A\_LfFac* and *A\_LfAdd*.
  4. In the user program, set the *A\_Req* input to TRUE.
- 

- i** The function block starts with a rising edge on *A\_Req*.
- 

The *A\_Busy* output is set to TRUE until the variables have been received or the receive timeout has expired. Afterwards, *A\_Busy* is set to FALSE and *A\_Valid* or *A\_Error* to TRUE.

If no error occurred during the variable reception, the *A\_Valid* output is set to TRUE. The variables defined in the Data tab can be evaluated. The *A\_Len* output contains the amount of data in bytes that was actually read.

If an error occurred during the variable reception, the *A\_Error* output is set to TRUE and an error is output to *A\_Status*.

## 8.8 Control Panel (Send/Receive over TCP)

The Control Panel can be used to verify and control the settings for the Send/Receive protocol. Details about the current status of the Send/Receive protocol (e.g., disturbed connections) are displayed.

### To open Control Panel for monitoring the Send/Receive protocol

1. In the structure tree, click **Resource**.
2. Click **Online** on the **Action Bar**.
3. In the **System Log-in** window, enter the access data to open the Control Panel for the resource.
4. In the structure tree for the Control Panel, select **Send/Receive Protocol**.

### 8.8.1 View box for General Parameters

The view box displays the following values of the Send/Receive protocol.

| Element                    | Description  |
|----------------------------|--|
| Name                       | TCP SR Protocol  |
| $\mu$ P Load (planned) [%] | If the connection of the connection module to the Modbus master is lost, the input variables are forwarded initialized or unchanged to the process module, depending on the parameter X-COM: <i>Values at Connection Loss to Master</i> . See Chapter 7.2.3.2. |
| $\mu$ P load (actual) [%]  |  |
| Undisturbed Connections    | Number of undisturbed connections  |
| Disturbed Connections      | Disturbed Connection Count   |

Table 262: S&R Protocol View Box

### 8.8.2 View box for TCP connections

The view box displays the following values of the selected TCP connections.

| Element                   | Description   |
|---------------------------|---|
| Name                      | TCP connection  |
| Partner timeout           | Yes: Partner request timeout expired<br>No: Partner request timeout not expired   |
| Connection State          | Current state of this connection<br>0x00: Connection OK<br>0x01: Connection closed<br>0x02: Server waits for the connection to be established<br>0x04: Client attempts to establish connection<br>0x08: Connection is blocked |
| Peer Address              | IP address of the communication partner.  |
| Peer Port                 | Port of the communication partner.  |
| Own Port                  | Port of this controller   |
| Watchdog Time [ms]        | It is the actual partner request timeout within which the communication partner received data at least one time after data has been sent.   |
| Error Code                | Error code (see Chapter 8.8.3)  |
| Timestamp Error Code [ms] | Timestamp for the last reported fault<br>Range of values: Seconds since 1/1/1970 in milliseconds  |
| Received Bytes [Bytes]    | Number of bytes received in this TCP connection   |
| Transmitted Bytes [Bytes] | Number of bytes sent in this TCP connection   |

Table 263: View Box of the Modbus Slave

### 8.8.3 Error Code of the TCP Connection

The error codes can be read from the *Error Code* variable.

For each configured connection: The connection state is composed of the connection state and error code of the last operation.

| Error Code Decimal | Error Code Hexadecimal | Description                                  |
|--------------------|------------------------|--|
| 0                  | 16#00                  | OK   |
| 4                  | 16#04                  | Interrupted system call                      |
| 5                  | 16#05                  | I/O Error                                    |
| 6                  | 16#06                  | Device unknown                               |
| 9                  | 16#09                  | Invalid socket descriptor                    |
| 12                 | 16#0C                  | No memory available                          |
| 13                 | 16#0D                  | Access Denied                                |
| 14                 | 16#0E                  | Invalid address                              |
| 16                 | 16#10                  | Device occupied                              |
| 22                 | 16#16                  | Invalid value (e.g., in the length field)    |
| 23                 | 16#17                  | Descriptor table is full                     |
| 32                 | 16#20                  | Connection aborted                           |
| 35                 | 16#23                  | Operation is blocked                         |
| 36                 | 16#24                  | Operation currently in process               |
| 37                 | 16#25                  | Operation already in process                 |
| 38                 | 16#27                  | Target address required                      |
| 39                 | 16#28                  | Message too long                             |
| 40                 | 16#29                  | Incorrect protocol type for the socket       |
| 42                 | 16#2A                  | Protocol not available                       |
| 43                 | 16#2B                  | Protocol not supported                       |
| 45                 | 16#2D                  | Operation on socket not supported            |
| 47                 | 16#2F                  | The address is not supported by the protocol |
| 48                 | 16#30                  | Address already in use                       |
| 49                 | 16#31                  | The address cannot be assigned               |
| 50                 | 16#32                  | Network is down                              |
| 53                 | 16#35                  | Software caused connection abort             |
| 54                 | 16#36                  | Connection reset by peer                     |
| 55                 | 16#37                  | No buffer space available                    |
| 56                 | 16#38                  | Socket already connected                     |
| 57                 | 16#39                  | Socket not connected                         |
| 58                 | 16#3A                  | Socket closed                                |
| 60                 | 16#3C                  | Operation time expired                       |
| 61                 | 16#3D                  | Connection refused (from peer)               |
| 65                 | 16#41                  | No route to peer host                        |
| 78                 | 16#4E                  | Function not available                       |
| 254                | 16#FE                  | Timeout occurred                             |
| 255                | 16#FF                  | Connection closed by peer                    |

Table 264: Error Codes of the TCP Connection

#### 8.8.4 Additional Error Code Table for the Function Blocks

The error codes for the function blocks are only output to A\_Status of the S&R TCP function blocks.

| Error Code Decimal | Error Code Hexadecimal | Description  |
|--------------------|------------------------|--|
| 129                | 16#81                  | No connection exists with this identifier.                               |
| 130                | 16#82                  | Length is greater than or equal to null.                                 |
| 131                | 16#83                  | Only cyclic data is permitted for this connection                        |
| 132                | 16#84                  | IOC: Invalid state   |
| 133                | 16#85                  | Timeout value too large  |
| 134                | 16#86                  | Internal program error   |
| 135                | 16#87                  | Configuration error  |
| 136                | 16#88                  | Transferred data does not match data area                                |
| 137                | 16#89                  | Function block stopped   |
| 138                | 16#8A                  | Timeout occurred or transmission blocked                                 |
| 139                | 16#8B                  | Another function block of this type is already active on this connection |

Table 265: Additional Error Codes

#### 8.8.5 Connection State

| Error Code Hexadecimal | Description                                       |
|------------------------|---|
| 16#00                  | Connection OK                                     |
| 16#01                  | Connection closed                                 |
| 16#02                  | Server waits for the connection to be established |
| 16#04                  | Client attempts to establish connection           |
| 16#08                  | Connection is blocked                             |

Table 266: Connection State

#### 8.8.6 Partner Connection State

| Protocol State Decimal | Description   |
|------------------------|---------------|
| 0                      | No connection |
| 1                      | Connection OK |

Table 267: Partner's Connection State

## 9 SNTP Protocol

(Simple Network Time Protocol)

The SNTP protocol is used to synchronize the time of the SNTP client over the SNTP server.

HIMax/HIMatrix controllers can be configured and used as **SNTP server** and/or as **SNTP client**. The SNTP standard in accordance with RFC 2030 (SNTP version 4) applies with the limitation that only the unicast mode is supported.

### Equipment and system requirements

| Element    | Description   |
|------------|---|
| Controller | HIMax with COM module or only with processor module<br>HIMatrix: CPU OS version 7 and beyond and COM OS version 12 and beyond |
| Activation | This function is activated by default in all HIMax/HIMatrix systems.  |
| Interface  | Ethernet 10/100/1000BaseT   |

Table 268: Equipment and System Requirements for the S&R TCP

### 9.1 SNTP Client

To synchronize time settings, the SNTP client always uses the available SNTP server having the highest priority.

One SNTP client can be configured for time synchronization in each resource.

- **i** Time synchronization of a remote I/O performed by a HIMax controller.  
If a SNTP client is configured on a HIMax controller, the internal SNTP server of the HIMax controller is shut down.  
An SNTP server must be set up on the HIMax communication module connected to the remote I/O to ensure that the HIMax controller performs the time synchronization of the remote I/O.

#### To create a new SNTP client

1. In the structure tree, open **Configuration, Resource, Protocols**.
2. Right-click **Protocols**, then click **New, SNTP Client**.
  - A new SNTP Client is created.
3. Right-click the SNTP client, and click **Properties** and select the **COM module**.

The dialog box for the SNTP client contains the following parameters:

| Element                             | Description   |                    |  |                   |   |
|-------------------------------------|---|--------------------|--|-------------------|---|
| Type                                | SNTP Client   |                    |  |                   |   |
| Name                                | Name for the SNTP client, composed of a maximum of 32 characters.   |                    |  |                   |   |
| Module                              | Selection of the COM or processor module within which the protocol is processed.  |                    |  |                   |   |
| Behavior on CPU/COM Connection Loss | <p>If the connection of the processor module to the communication module is lost, the input variables are initialized or continue to be used unchanged in the process module, depending on this parameter.</p> <p>For instance, if the communication module is removed when communication is running.</p> <p><b>If a project created with a SILworX version prior to V3 should be converted, this value must be set to Retain Last Value to ensure that the CRC does not change.</b></p> <p><b>For HIMatrix controllers with CPU OS version prior to V8, this value must always be set to Retain Last Value.</b></p> <table> <tr> <td>Adopt Initial Data</td> <td>Input variables are reset to their initial values.</td> </tr> <tr> <td>Retain Last Value</td> <td>The input variables retains the last value.</td> </tr> </table>   | Adopt Initial Data | Input variables are reset to their initial values. | Retain Last Value | The input variables retains the last value. |
| Adopt Initial Data                  | Input variables are reset to their initial values.  |                    |  |                   |   |
| Retain Last Value                   | The input variables retains the last value.   |                    |  |                   |   |
| Activate Max. $\mu$ P Budget        | <p>It is not taken into account by the operating system.</p> <p>Parameter was retained due to CRC and reload stability.</p>   |                    |  |                   |   |
| Max. $\mu$ P budget in [%]          | <p>It is not taken into account by the operating system.</p> <p>Parameter was retained due to CRC and reload stability.</p>   |                    |  |                   |   |
| Description                         | Any unique description for the SNTP client  |                    |  |                   |   |
| Current SNTP Version                | The current SNTP version is displayed.  |                    |  |                   |   |
| Reference Stratum                   | <p>The stratum of an SNTP client specifies the precision of its local time. The lowest the stratum, the more precise its local time.</p> <p>Zero means an unspecified or not available stratum (not valid). The SNTP server currently used by an SNTP client is the one that can be reached and has the highest priority.</p> <p>If the stratum of the current SNTP server is lower than the stratum of the SNTP client, the resource adopts the time of the current SNTP server.</p> <p>If the stratum of the current SNTP server is higher than the stratum of the SNTP client, the resource does not adopt the time of the current SNTP server.</p> <p>If the stratum of the current SNTP server is identical with the stratum of the SNTP client, two different cases result:</p> <ul style="list-style-type: none"> <li>▪ If the SNTP Client (resource) only operates as SNTP client, the resource adopts the time of the current SNTP server.</li> <li>▪ If the SNTP client (resource) also operates as SNTP server, the resource adopts the half value of the time difference to the current SNTP server per SNTP client request (time approaches slowly).</li> </ul> <p>Range of values: 1...16<br/>Default value: 15</p> |                    |  |                   |   |
| Client Time Request Interval [s]    | <p>Time interval within which the current SNTP Server performs the time synchronization.</p> <p>The Client Request Time Interval set in the SNTP client must be higher than the timeout set in the SNTP server.</p> <p>Range of values: 16...16384s<br/>Default value: 16</p>   |                    |  |                   |   |

Table 269: SNTP Client Properties

## 9.2 SNTP Client (Server Information)

The connection to the SNTP server is configured in the SNTP Server Info.

1 to 4 SNTP Server Info can be subordinate to a SNTP client.

### To create a new SNTP Server Info

1. In the structure tree, open **Configuration, Resource, Protocols, SNTP Client**.
2. Right-click **SNTP Client**, and then click **New, SNTP Server Info**.  
 A new SNTP server Info is created.
3. Right-click the **SNTP-Server Info**, click **Properties**, and then select the **COM Module**.

The dialog box for the SNTP Server Info contains the following parameters:

| Element                | Description  |
|------------------------|--|
| Type                   | SNTP Server Info   |
| Name                   | Name for the SNTP server. A maximum of 31 characters.  |
| Description            | Description for the SNTP server. A maximum of 31 characters.   |
| IP Address             | IP address of the resource or PC in which the SNTP Server is configured.<br>Default value: 0.0.0.0   |
| SNTP Server Priority   | Priority with which the SNTP client addresses this SNTP server.<br>The SNTP servers configured for the SNTP client should have different priorities.<br>Range of values: 0 (lowest priority) to 4294967295 (highest priority).<br>Default value: 1 |
| SNTP Server Timeout[s] | The timeout in the SNTP server must be set lower than the value for the <i>Time Request Interval</i> in the SNTP client.<br>Range of values: 1...16384s<br>Default value: 1  |

Table 270: SNTP Server Info Properties

## 9.3 SNTP Server

The SNTP server accepts the requests from a SNTP client and sends back to the SNTP client its current time.

### To create a new SNTP server

1. In the structure tree, open **Configuration, Resource, Protocols**.
2. Right-click **Protocols**, and then click **New, SNTP Server**.  
 A new SNTP server is created.
3. Right-click the **SNTP Server**, click **Properties**, and then click the **COM Module**.

The dialog box for the SNTP server contains the following parameters:

| Element                             | Description   |
|-------------------------------------|---|
| Type                                | SNTP Server   |
| Name                                | Name for the SNTP server, composed of a maximum of 31 characters.   |
| Module                              | Selection of the COM or processor module within which the protocol is processed.  |
| Activate Max. $\mu$ P Budget        | Activated:<br>Adopt the $\mu$ P budget limit from the Max. $\mu$ P Budget in [%] field.<br><br>Deactivated:<br>Do not use the $\mu$ P budget limit for this protocol.   |
| Max. $\mu$ P budget in [%]          | Maximum $\mu$ P load of the module that can be used for processing the protocols.<br><br>Range of values: 1...100%<br>Default value: 30%  |
| Behavior on CPU/COM Connection Loss | If the connection of the processor module to the communication module is lost, the input variables are initialized or continue to be used unchanged in the process module, depending on this parameter.<br>For instance, if the communication module is removed when communication is running.<br><b>If a project created with a SILworX version prior to V3 should be converted, this value must be set to Retain Last Value to ensure that the CRC does not change.</b><br><b>For HIMatrix controllers with CPU OS version prior to V8, this value must always be set to Retain Last Value.</b><br><br>Adopt Initial Data      Input variables are reset to their initial values.<br>Retain Last Value      The input variables retains the last value. |
| Description                         | Description for the SNTP  |
| Current SNTP Version                | The current SNTP version is displayed.  |
| Stratum of Timeserver               | The stratum of an SNTP server specifies the precision of its local time. The lowest the stratum, the more precise the local time.<br>Zero means an unspecified or not available stratum (not valid).<br>The value for the SNTP server stratum must be lower or equal to the stratum value of the requesting SNTP client. Otherwise, the SNTP client does not accept the SNTP server time.<br><br>Range of values: 1...15<br>Default value: 14   |

Table 271: SNTP Server Properties

## 10 X OPC Server

The HIMA X-OPC server serves as transmission interface between HIMax/HIMatrix controllers and third-party systems that are equipped with an OPC interface.

OPC means Openess, Productivity & Collaboration and is based on the technology developed by Microsoft. This technology allows the user to interconnect process control systems, visualization systems and controllers from different manufacturers, and it enables them to exchange data with one another (see also [www.opcfoundation.org](http://www.opcfoundation.org)). After installation, the HIMA X OPC server is run on a PC as Windows service.

- 
- i** SILworX is used to configure and operate the entire X OPC server. Like a controller, the X-OPC server can be loaded, started and stopped from within the SILworX Control Panel.
- 

The X OPC server supports the following specifications:

- **Data Access (DA) versions 1.0, 2.05a and 3.0**  
DA is used to ensure process data transmission from the HIMax/HIMatrix controller to the OPC client. Each global variable of the HIMax/HIMatrix controller can be transferred to a OPC client.
- **Alarm&Event (A&E) version 1.10**  
A&E is used to transfer alarms and events from HIMax/HIMatrix controller to the OPC client. Each global variable of the HIMax/HIMatrix controller can be monitored using sequence of events recording.  
Events are state modifications of a variable that are performed by the plant or controllers and are provided with a timestamp.  
Alarms are events that signalize an increasing risk potential.  
Events are divided in Boolean and scalar events, see Chapter 10.7.

### 10.1 Equipment and System Requirements

| Element                     | Description   |
|-----------------------------|---|
| Activation                  | Software activation code required, see Chapter 3.4. The following licenses can be activated on an individual basis: <ul style="list-style-type: none"> <li>▪ Data Access (DA) Server</li> <li>▪ Alarm and Events (A&amp;E) Server</li> </ul>  |
| PC Operating System         | The OPC server can run on an x86 based PC with the following operating systems: <ul style="list-style-type: none"> <li>▪ Windows XP Professional (mind. Service Pack 2) (32-bit)</li> <li>▪ Windows Server 2003 (32-bit)</li> <li>▪ Windows Vista Ultimate (32-bit)</li> <li>▪ Windows Vista Business (32-bit)</li> </ul>   |
| Requirements to the host PC | Minimum requirements to the host PC: <ul style="list-style-type: none"> <li>▪ Pentium 4</li> <li>▪ 1 GByte (XP) or 2.5 GByte RAM (Vista)</li> <li>▪ The network card must be designed according to the data traffic (100 Mbit/s or 1 Gbit/s).</li> </ul> <p><b>i</b> The minimum requirements only apply to the operation for a X OPC server if no additional applications, such as SILworX or Word, is run on the host PC.</p> |

Table 272: Equipment and System Requirements for the X-OPC Server

## 10.2 X-OPC Server Properties

| Element                         | Description   |
|---------------------------------|---|
| OPC server                      | The X OPC server supports the following functions: <ul style="list-style-type: none"><li>▪ OPC Data Access Custom Interface, versions 1.0, 2.05a and 3.0.</li><li>▪ OPC Alarm &amp; Event Interfaces 1.10</li></ul> |
| Safety-related                  | The X-OPC server is run on a PC and is not safety-related.  |
| Interface                       | Recommended: Ethernet 1GBit/s   |
| Data exchange                   | Data exchange via <b>safeethernet</b> .   |
| Ethernet Network                | The underlying Ethernet network speed must be designed according to the data traffic (min. 100 Mbit/s, recommended 1GBit/s).  |
| Global Variables                | Only global variables from the configuration context may be used!   |
| Permissible Types of variables  | All data types that can be created in SILworX are permitted.  |
| Impermissible ASCII characters  | The following characters are reserved and must not be used (e.g., for global variables): ! " # ' , . / \`   |
| HIMax/HIMatrix Controllers      | An X-OPC server can support a maximum of 255 HIMax/HIMatrix controllers.  |
| safeethernet Connection         | The X-OPC server can exchange 128 kB in each <b>safeethernet</b> connection.  |
| X OPC Server                    | 10 X-OPC servers can be operated on a host PC.  |
| X-OPC Clients                   | An X-OPC server supports 10 X-OPC clients.  |
| Data Access Tags                | A Data Access server supports a maximum of 100 000 DA tags.<br>Definition:<br>Tags: Data provided by the X-OPC server. Tags correspond to the defined global variables.<br>Items: Data required by the OPC client.  |
| Alarm & Event Event Definitions | An X-OPC Alarm & Event server supports a maximum of 100 000 event definitions.  |

Table 273: X-OPC Server Properties

### 10.3 HIMax Controller Properties for X-OPC Connection

| Element                                     | HIMax                | HIMatrix L3                             | HIMatrix L2     | Description  |
|---|----------------------|---|-----------------|--|
| Process data volume                         | 128 kB               | 128 kB                                  | 16 kB           | Process data volume that a HIMA controller may exchange for each <b>safeethernet</b> connection to one X-OPC server.   |
| Fragment size                               | 1100 byte            | 1100 bytes                              | 900 byte        | In each HIMax cycle, only a view is sent to the X-OPC server.  |
| Ethernet interfaces                         | 10/100/<br>1000BaseT | 10/<br>100BaseT                         | 10/<br>100BaseT | Ethernet interfaces in use can simultaneously be used for additional protocols.  |
| Maximum number of system events (CPU event) | 20 000               | 4000                                    | n. a            | The events defined as CPU events are created on the processor module. The processor module creates all the events in each of its cycle.<br>This allows one to record and evaluate the value of each global variable as an event.   |
| Max. Number of I/O Events (I/O event)       | 6000                 | n. a                                    | n. a            | The event defined as I/O events can only be created on SOE I/O modules (e.g., AI 32 02 or DI 32 04). The processor module creates all the events in each of its cycle.   |
| Event memory size                           | 5000                 | 1000<br>(with enabled SOE license only) | n. a            | Non-volatile event buffer of the HIMax processor module.<br>If the event buffer is full, no new events can be stored as long as at least one X-OPC A&E server has not read an event entry and thus marked it as to be overwritten. |
| Max. Number of X-OPC Server                 | 4                    | 4                                       | 4               | Maximum number of X-OPC servers can access a HIMA controller and simultaneously read events from the event buffer of the processor module.   |
| n. a. non-applicable                        |                      |   |                 |  |

Table 274: HIMax Controller Properties for X-OPC Connection

Range of values for the UTC timestamp (Universal Time Coordinated):

sec fraction since 1970 in [udword]

ms fraction of the seconds as [udword] of 0-999

Default value: 01/01/2000 / 00:00:00

An automatic change to/out of daylight-saving time is not supported.

## 10.4 Actions Required as a Result of Changes

The following table shows the actions that must be performed after a change in the individual systems.

| Type of Change  | Changes to |          |       |
|---|------------|----------|-------|
|   | HIMax      | HIMatrix | X-OPC |
| <b>DA</b>   |            |          |       |
| Add tags  | C+D        | C+D      | C+D   |
| Tag name (GV change of name)                            | C+D        | C+D      | C+D   |
| Delete tags   | C+D        | C+D      | C+D   |
| Change fragments (Parameters and Add/Delete)            | C+D        | C+D      | C+D   |
| <hr/>   |            |          |       |
| <b>A&amp;E</b>  |            |          |       |
| Add event definition                                    | C+D        | n.a.     | C+D   |
| Delete Event Definition                                 | C+D        | n.a.     | C+D   |
| Change Event Source                                     | C+D        | n.a.     | C+D   |
| Change Alarm Texts                                      | -          | n.a.     | C+D   |
| Change Alarm Severity                                   | -          | n.a.     | C+D   |
| Change Ack Required parameter                           | -          | n.a.     | C+D   |
| Change Alarm Values with scalar events                  | C+D        | n.a.     | C+D   |
| Change the Alarm at False parameter with Boolean events | C+D        | n.a.     | C+D   |
| Change name   | C+D        | n.a.     | C+D   |
| Connect I/O channel to GV                               | C+R        | n.a.     | -     |
| Connect state variables to GV                           | C+R        | n.a.     | -     |
| <hr/>   |            |          |       |
| <b>In general</b>                                       |            |          |       |
| Change safeethernet parameters                          | C+D        | C+D      | C+D   |

Table 275: Actions Required as a Result of Changes

C: Code generation required

R: Reload required

D: Download required

n.a.: non-applicable

-: No action required

## 10.5 Forcing Global Variables on I/O Modules



If global variables connected to a process value are forced, the forced global variables have no effect on the global variables connected to the parameters

->**State LL, L,- N, H, HH.**

This particularity also applies if these alarms are configured in the Alarm&Event Editor.  
When testing, these variables must be forced individually.

## 10.6 Configuring an OPC Server Connection

This example shows how to configure a redundant X-OPC server connection to a HIMax controller.

The X-OPC servers provide the process variables and event values of the HIMax controller to the OPC clients. The OPC clients access these process variables and event values and represent them on their user interface.

### 10.6.1 Software required:

- SILworX
- X-OPC server
- OPC client

**i** SILworX is used to configure and operate the entire X OPC server. Like a controller, the X-OPC server can be loaded, started and stopped from within the SILworX Control Panel.

### 10.6.2 Requirements for Operating the X-OPC Server:

- The Ethernet network should have a bandwidth of at least 100 Mbit/s (better 1GBit/s).
- The system time of computer and server must be synchronized, e.g., using SNTP.
- Make sure that the data records for Data Access and Alarm & Events on the controller, X-OPC servers and OPC clients match one another.

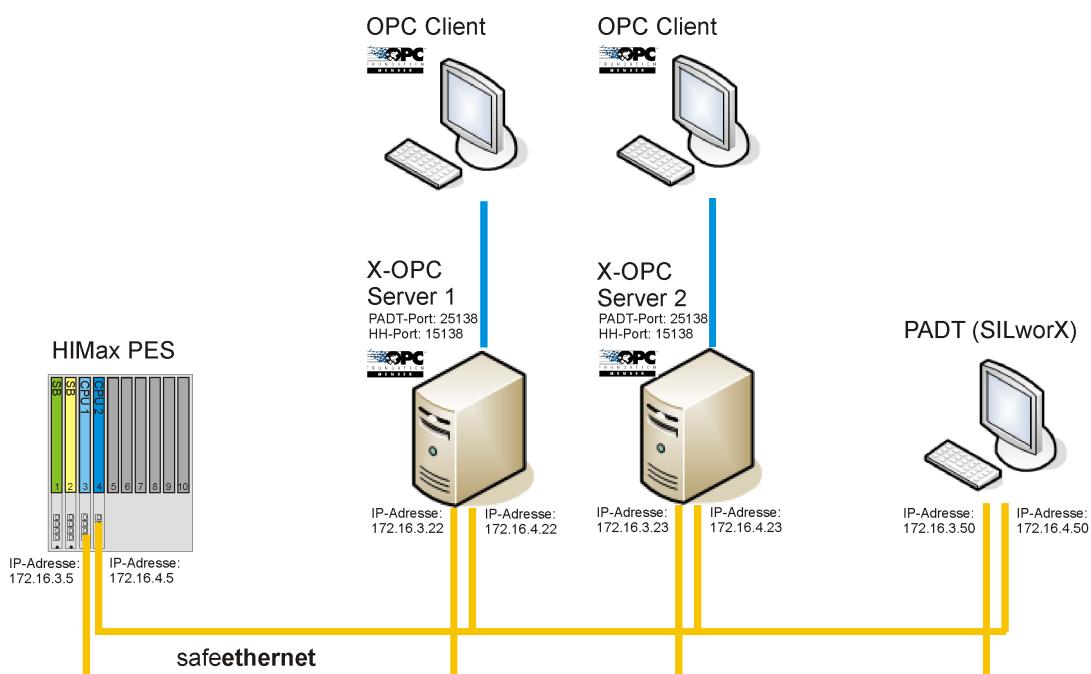


Figure 79: Redundant X-OPC Operation

### 10.6.3 Installation on Host PC

The X-OPC server must be installed on the respective host PCs.

- i** Note down the system ID and the number of the PADT port. These values are required for generating the license key!

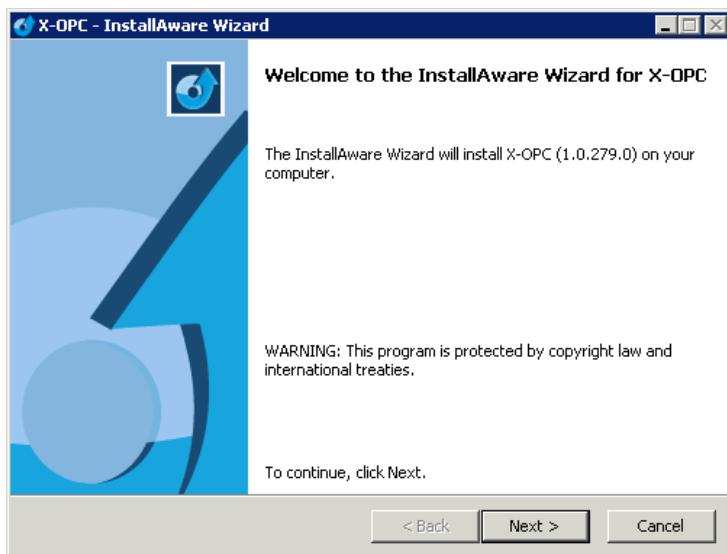


Figure 80: Wizard for Installing the X-OPC Server

#### To install the X-OPC server on the first host PC

Start the X-OPC.exe on each host PC and follow the instructions of the install wizard.

1. Enter the following data for the X-OPC server:
  - System ID: 100
  - PADT port: 25138
  - An arbitrary name for the X-OPC server (it is displayed in the OPC client).
2. To install the X-OPC server, click **Next>**

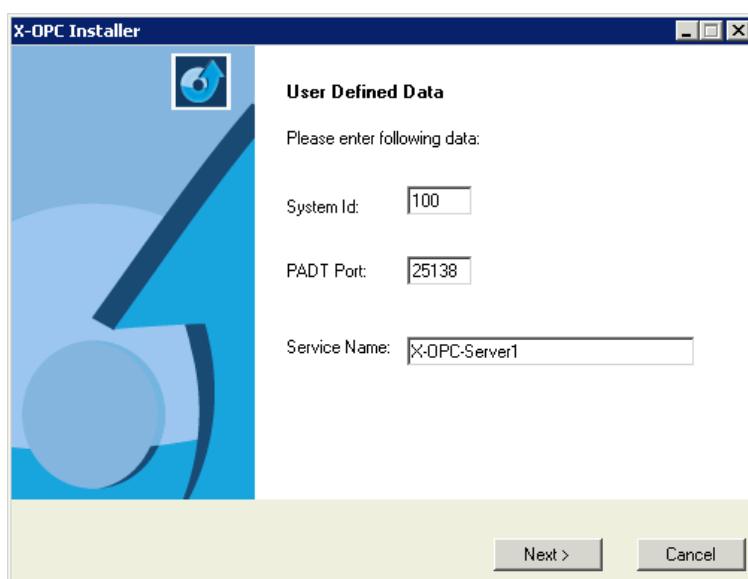


Figure 81: Wizard for Installing the X-OPC Server

**To install the X-OPC server on the second host PC**

**i** The Class ID of the first X-OPC server must be determined before the second X-OPC server is installed!

If one OPC client is redundantly connected to two X-OPC servers, some OPC client systems expect that the class IDs of the two X-OPC servers are identical. First determine the class ID of the first X-OPC server (e.g., using the OPC client) and note it down.

Start the X-OPC.exe file on the second host PC and follow the instructions of the install wizard.

1. Enter the following data for the X-OPC server:

- System ID: 110
- PADT port: 25138
- An arbitrary name for the X-OPC server (it is displayed in the OPC client).

**i** PADT port and HH port of the second X-OPC server may be identical with the first one, if the X-OPC servers are run on different PCs.

2. Click **Continue>** to confirm.

**To set the same class ID on the second host PC**

1. Choose the CLSID setting **manual** for DA and AE.
2. Enter the class ID of the first X-OPC server in the **CLSID** fields.
3. To install the X-OPC server, click **Next>**

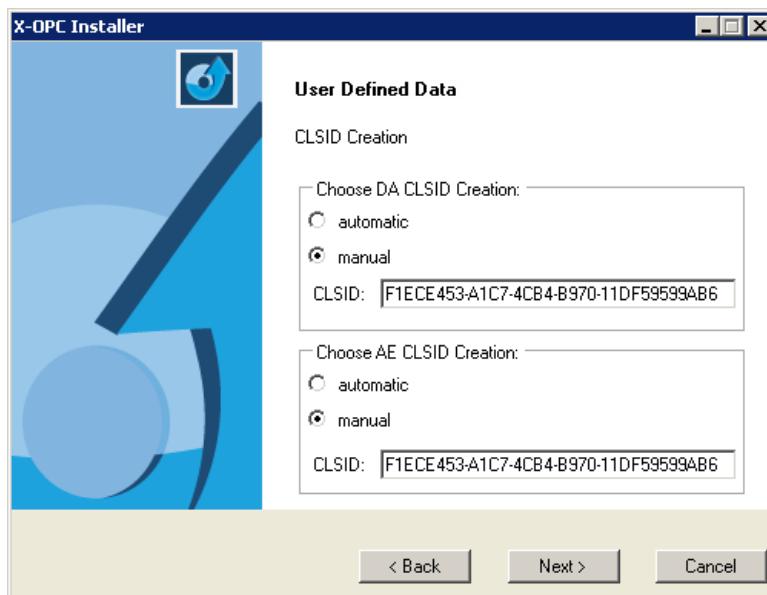


Figure 82: Setting the Class ID of the Second X-OPC Server Manually

**To automatically start the X-OPC servers after restarting the PCs**

1. In Windows, go to **Start, Settings, Control Panel, Administration, Services** and select **X-OPC Server** from the list.
2. Right-click the OPC Server, then select **Properties..**.
3. In the **General** tab, select the **Automatic** start type.

**To verify if the X-OPC server is running on the PC**

1. Open the *Windows Task Manager* and select the *Processes* tab.
2. Check if the *X-OPC.exe* process is running on the PC.



If the OPC client and OPC server are not running on the same PC, the DCOM interface must be adjusted.

To do this, observe the instructions given in the manual of the OPC Foundation *Using OPC via DCOM with Microsoft Windows XP Service Pack 2 Version 1.10* (see [www.opcfoundation.org](http://www.opcfoundation.org)).

---

#### 10.6.4 Configuring the OPC Server in SILworX

##### To create a new OPC server set in SILworX

1. In the structure tree, open **Configuration**.
2. Right-click **Configuration**, and then click **New, OPC-Server Set**.  
 A new OPC-Server set is added.
3. Right-click **OPC Server Set**, select **Properties** and accept the default values

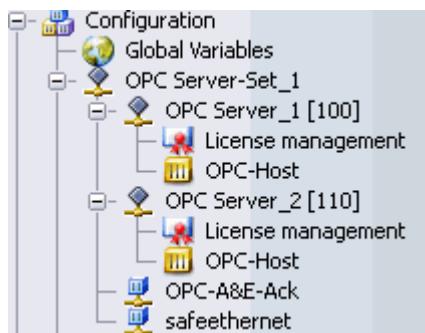


Figure 83: Redundant X-OPC Operation

##### To configure the first X-OPC server in SILworX

1. In the structure tree, open **Configuration, OPC Server Set**.
2. Right-click **OPC Server Set** and select **New, OPC Server**.  
 A new OPC server is added.
3. Right-click **OPC Server** and select **Properties**.
  - Enter the system ID [SRS] (e.g., 100)
  - Accept the default settings.
3. Right-click **OPC Host** and select **Edit**.  
 The OPC host dialog box for configuring the IP interfaces appears.
4. Right-click anywhere in the OPC host window and select **New IP Device**.
  - Set the PADT port (e.g., 25138).
  - IP address of the PC on which the X-OPC server is installed (e.g., 172.16.3.22).
  - IP address of the PC on which the X-OPC server is installed (e.g., 172.16.4.22).
  - Define it as Standard Interface checking the corresponding checkbox.
  - Set the HH Port (e.g., 15138)

Configure the redundant OPC server in the same OPC server set.

#### To configure the second OPC server

1. In the structure tree, open **Configuration, OPC Server Set**.
2. Right-click **OPC Server Set** and select **New, OPC Server**.
  - A new OPC server is created.
3. Right-click **OPC Server** and select **Properties**.
  - Enter the system ID [SRS] (e.g., 110)
  - Accept the default settings.
5. Right-click **OPC Host** and select **Edit**.
  - The OPC host dialog box for configuring the IP interfaces appears.
6. Right-click anywhere in the OPC host window and select **New IP Device**.
  - Set the PADT port (e.g., 25138).
  - IP address of the PC on which the X-OPC server is installed (e.g., 172.16.3.23).
  - IP address of the PC on which the X-OPC server is installed (e.g., 172.16.4.23).
  - Define it as standard interface checking the corresponding checkbox.
  - Set the HH port (e.g., 15138)

**i** If a firewall is installed on the PC, the TCP/UDP PADT and HH ports for the X-OPC servers must be selected on the Exception tab of the firewall configuration.

### 10.6.5 Settings for the OPC Server in the safeethernet Editor

#### To create the safeethernet connection between the OPC server and the resource (HIMax controller)

1. In the OPC server set, open the **safeethernet Editor**.
2. In the Object Panel, drag the resource anywhere onto the workspace of the **safeethernet Editor**.
3. For Alarm & Events, the *Activate SOE* parameter is activated by default.

|   | IF CH 1 (local)             | IF CH 2 (local)             | IF CH 1 (target)         | IF CH 2 (target)         | Profile      | Sync/Asycn |
|---|-----------------------------|-----------------------------|--------------------------|--------------------------|--------------|------------|
| 1 |                             |                             |                          |                          | Fast & Noisy | ASYNC      |
| 2 | 110.x.x (122.16.4.23:15138) | 110.x.x (122.16.4.23:15138) | 2.0.15 (172.16.4.5:6010) | 2.0.15 (172.16.4.5:6010) |              |            |
| 3 | 100.x.x (172.16.3.22:15138) | 100.x.x (172.16.3.22:15138) | 2.0.5 (172.16.3.5:6010)  | 2.0.5 (172.16.3.5:6010)  |              |            |

Figure 84: Redundant X-OPC Operation

**i** The used Ethernet interfaces of the PCs are represented in the **IF CH1 (local)** column. The Ethernet interfaces of the HIMax controller must be selected in the **IF CH1 (target)** column.

The **safeethernet** parameters of the X-OPC server communication are set by default for ensuring the maximum availability.

Receive Timeout = 1000 ms, Response Time = 500 ms etc.

For more information on the **safeethernet** parameters, refer to Chapter 4.6.

## 10.6.6 Configuring the X-OPC Data Access Server in SILworX

### To create the fragment definitions for the safeethernet connection

Requirement: The safeethernet Editor of the OPC server must be opened.

1. Right-click the row corresponding to the **resource** to open its context menu.
2. Select **Detail View** to open the detail view of the safeethernet connection.
3. Click the **Fragment Definitions** tab.
4. Right-click anywhere in the workspace and select **New Fragment Definition**.  
The Priority column is used to define how often this fragment should be sent compared to the other fragments (a fragment's size is 1100 bytes).  
For fragment definitions, first use the standard setting for Priority 1, see also Chapter 10.6.8.
5. Click the **OPC Server Set<->Resource** tab.

### To add the OPC receive variables

OPC receive variables are sent from the resource to the OPC server.

1. Open the detail view of the X-OPC safeethernet Editor and select the **OPC Server Set<->Resource** tab.
2. In the Object Panel, drag a **Global Variable** onto the **OPC Server Set <-Resource** area.
3. Double-click the **Fragment Name** column and select the **Fragment Definition** created beforehand.
4. Repeat these steps for every further OPC receive variables.

### To add the OPC send variables

OPC send variables are sent from the OPC server to the resource.

1. Open the detail view of the X-OPC safeethernet Editor and select the **OPC Server Set<->Resource** tab.
2. In the Object Panel, drag a **Global Variable** onto the **OPC Server Set->Resource** area.
3. Double-click the **Fragment Name** column and select the **Fragment Definition** created beforehand.
4. Repeat these steps for every further OPC send variables.



The OPC send and receive variables must be created in the OPC server set one time only. The variables are automatically used by both X-OPC servers in the OPC server set.

### To generate the code and load a resource

1. In the structure tree, select **Configuration, Resource**.
2. Click **Code Generation** in the Action Bar and click **OK** to confirm.
3. Thoroughly verify the messages displayed in the logbook and correct potential errors.
4. Load the generated code into the resource.

### To generate the code and verify the OPC set

1. In the structure tree, open **Configuration, OPC Server Set**.
2. Click **Code Generation** in the Action Bar and click **OK** to confirm.
3. Thoroughly verify the messages displayed in the logbook and correct potential errors.

### To load the generated code into the X-OPC server

1. Right-click **OPC Server** and select **Online** to perform a **System Login**.
2. Enter the access data:

- IP address of the PC on which the X-OPC server is installed (e.g., 172.16.3.23).
  - User name: Administrator
  - Password: <empty>
  - Rights: Administrator
3. Click **Login** to open the Control Panel.
  4. In the SILworX menu bar, click the **Resource Download** symbol.
    - The code is loaded into the X-OPC server.
  5. In the SILworX menu bar, click the **Resource Cold Start** symbol.
    - The X-OPC server is running.

#### To open the OPC client

The name of the X-OPC server displayed in the OPC client is composed of :  
**HIMA** (manufacturer).**Service name** (see Chapter 10.6.3) **DA** (Data Access).

Connect to the X-OPC server. At this point, the configured DA data should be transferred to the OPC client.

### 10.6.7 Configuring the X-OPC Alarm&Event Server in SILworX

This example shows how to connect an X-OPC A&E server to a HIMax controller. The X-OPC A&E server records the events from the HIMax controller via **safeethernet** and provide them to the OPC client. The OPC client accesses these event variables and displays them on its user interface.

#### To create an Alarm&Event Editor

1. In the structure tree, open **Configuration, Resource**.
2. Right-click **Resource** and select **New, Alarm&Events**.  
 A new Alarm&Event Editor is created.

#### To create the Alarm&Events

1. In the structure tree, open **Configuration, Resource**.
2. Right-click **Alarm & Events** and select **Edit**.
3. Select the **Event Definition Bool** tab for Boolean events, see Chapter 10.7.1.
4. Select the **Event Definition Scalar** tab for scalar events, see Chapter 10.7.2.
5. In the Object Panel, drag the **global variable** anywhere onto the workspace of the Alarm & Event Editor.
6. Enter the event priority in the **safeethernet** Editor, see Chapter 10.6.8.

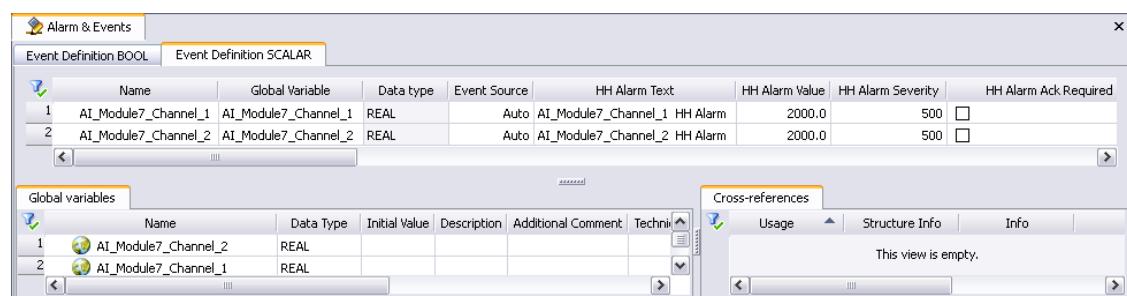


Figure 85: Alarm & Event Editor

**To establish the acknowledge connection between both Alarm&Event X-OPC servers:**

- 
- i** If two Alarm&Event X-OPC servers are operated redundantly, the acknowledgements to confirm the alarms on both X-OPC servers can be synchronized. To do this, an acknowledge connection is created.
- 

1. In the structure tree, open **Configuration, OPC Server Set, New**.
2. Right-click **OPC Server Set** and select **New, OPC A&E Ack**.
3. Select the following IP connections in the OPC A&E Ack dialog box.
  - IF CH1 (OPC Server 1, e.g., 172.16.3.22).
  - IF CH2 (OPC server 1, e.g., 172.16.4.22).
  - IF CH1 (OPC server 2, e.g., 172.16.3.23).
  - IF CH2 (OPC server 2, e.g., 172.16.4.23).



Figure 86: Redundant X-OPC Operation

**To generate the code and load a resource**

1. In the structure tree, select **Configuration, Resource**.
2. Click **Code Generation** located on the Action Bar and click **OK** to confirm.
3. Thoroughly verify the messages displayed in the logbook and correct potential errors.
4. Load the generated code into the resource.

**To generate the code and verify the OPC set**

1. In the structure tree, open **Configuration, OPC Server Set**.
2. Click **Code Generation** located on the Action Bar and click **OK** to confirm.
3. Thoroughly verify the messages displayed in the logbook and correct potential errors.

**To load the generated code into the X-OPC server**

1. Right-click the **OPC Server** and select **Online** to log in to the system.
2. Enter the access data:
  - IP address of the PC on which the X-OPC server is installed (e.g., 172.16.3.23).
  - User name: Administrator
  - Password: <empty>
  - Rights: Administrator
3. Click **Login** to open the Control Panel.
4. In the SILworX menu bar, click the **Resource Download** symbol.  
 The code is loaded into the OPC server.
5. In the SILworX menu bar, click the **Resource Cold Start** symbol.  
 The OPC server is running.

**To open the OPC client**

The name of the X-OPC server displayed in the OPC client is composed of:  
**HIMA** (manufacturer).**Service name** (see Chapter 10.6.3) **AE (Alarm&Event)**.

Connect to the X-OPC server. At this point, the configured alarms and events should be transferred to the OPC client.



If a controller and an X-OPC A&E server are connected, the X-OPC A&E server must synchronize with the controller. To do this, the X-OPC A&E server reads the current state of all the variables defined as events and transfers the upcoming alarms to the OPC client. An image of the current controller state can thus be created in the OPC client. The events are only read at this moment.

After the X-OPC server has synchronized with the controller, all the events on the OPC client are updated. Entries for events with an older timestamp are overwritten with the currently read states of the event variables

---

### 10.6.8 Configuring the Fragments and Priorities in SILworX

Depending on its type and for each **safeethernet** connection, a HIMA controller can send 128 kB or 16 kB to an X-OPC server, but only one fragment (1100 bytes or 900 bytes) per HIMA CPU cycle. To send more data via a **safeethernet** connection, data must be fragmented. The Priority parameter associated with these fragments can be used to define how often these fragments should be refreshed.

- 
- i** Fragments with the priority **n** and fragments with the priority **m** are sent at a ratio of **n** to **m** times.
- 

For the reaction time from the controller to the X-OPC server, also observe the number of SOE fragments and commands (e.g., stop, start).

$T_R = t_1 + t_2 + t_3 + t_4$  only applies if the priority of all fragments for state data is equal to 1

|       |  |
|-------|--|
| $T_R$ | Worst Case Reaction Time   |
| $t_1$ | Safety Time of PES 1   |
| $t_2$ | <i>Number of Fragments * Receive TMO</i>   |
| $t_3$ | Safety Time of X-OPC Server  |
| $t_4$ | Delay due to SOE function; depending on the number of events and on how the connection is established. |

The response time for the inverse direction can be determined using the same formula, but only one fragment is usually relevant in this case, since the X- OPC server only transfers the data written by OPC clients.

Maximum number of fragments: 1024

Maximum size of a fragment: 1100 bytes or 900 bytes

Range of values for the priorities: 1 (highest) to 65 535 (lowest)

| Designation  | Priority default value |
|--|------------------------|
| Priority of events (Alarm&Event)                   | 1                      |
| Priority of state values (Alarm&Event)             | 10                     |
| Priority of the fragment definitions (Data Access) | 1                      |

Table 276: Default Values Associated with the Priorities

### 10.6.8.1 Priority of Fragments for Events (Alarm&Event)

The events created in the Alarm&Event Editor are automatically fragmented and transferred in fragments.

The event priority are entered in the columns **Priority of Events** and **Priority of State Values** of the safeethernet Editor. These priorities are thus valid for all the Alarm&Event fragments of that given safeethernet connection.

#### To set the priority values for Alarm&Event Fragments

1. In the OPC server set, open the safeethernet Editor.

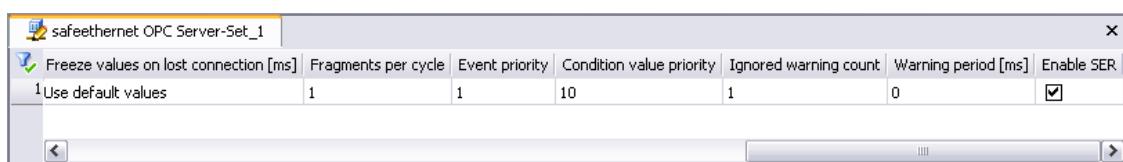


Figure 87: safeethernet Editor

2. Double-click **Priority of Events** to modify the priority of the events.  
All event fragments receive the priority entered in the **Priority of Events** column (e.g., 1). This is done to define the priority with which the X-OPC server requires events from the controller. If no events exist in the controller at a given point in time, none is transferred.
3. Double-click **Priority of State Values** to modify the priority of the event state values. The fragments for the event state values are assigned with the priority entered in the **Priority of State Values** column (e.g., 10).



The state values of the events are only required for synchronization reasons (e.g., when the connection is being established); they can thus be transferred in larger intervals than the events.

### 10.6.8.2 Priorities of Data Access Fragments

Global variables can be assigned to each data access fragment and the fragment priority can be set. The priority is used to determine how often these variables must be refreshed.

#### To set the priority values for Data Access Fragments

Requirement: The safeethernet Editor of the OPC server must be opened.

1. Right-click the row corresponding to the **resource** to open its context menu.
2. Select **Detail View** to open the detail view of the safeethernet connection.
3. Click the **Fragment Definitions** tab.
4. Set the priority for Data Access Views in the Priority column.
  - Assign a high priority to Data Access Fragments with global variables that should be refreshed frequently (e.g., 1).
  - Assign a lower priority to data access fragments with global variables that should be refreshed less frequently (e.g., 10)

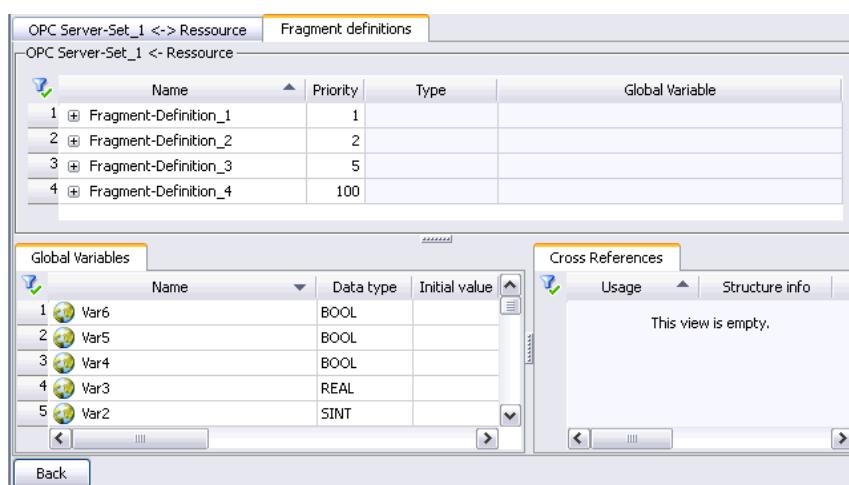


Figure 88: Detailed View of the safeethernet Connection

Further, the state and timestamp of each data access fragment can be read using the following variables.

| Name                    | Description   |   |
|-------------------------|---|---|
| Fragment timestamp [ms] | Millisecond fraction of the timestamp (current system time) |   |
| Fragment timestamp [s]  | Second fraction of the timestamp (current system time)      |   |
| Fragment status         | Status  | Description   |
|                         | 0   | CLOSED: Connection is closed  |
|                         | 1   | TRY OPEN: An attempt is made to open the connection, but it is still closed.  |
|                         | 2   | CONNECTED: The connection exists and current fragment data was received (cp. timestamp). As long as no fragment data is received, the fragment state is set to TRY_OPEN when establishing the connection. |
|                         | i   | The connection state of the safeethernet Editor (see Chapter 4.4) is set to CONNECTED as soon as the connection is open. Unlike Fragment State, no data may have been exchanged here.                     |

Table 277: State and Timestamp for the Data Access Fragments

## 10.7 Alarm & Event Editor

The Alarm & Event Editor is used to set the parameters for the alarms and events of the HIMax controller.

### To create an Alarm&Event Editor

1. In the structure tree, open **Configuration, Resource**.
2. Right-click **Resource** and select **New, Alarm&Events**.  
 A new Alarm & Events object is added.

### To create the Alarm&Events

1. In the structure tree, open **Configuration, Resource**.
2. Right-click **Alarm & Events** and select **Edit**.
- 3 Select the **Event Definition Bool** tab for Boolean events, see Chapter 10.7.1.
4. Select the **Event Definition Scalar** tab for scalar events, see Chapter 10.7.2.
5. In the Object Panel, drag the **global variable** anywhere onto the workspace of the Alarm & Event Editor.
6. Enter the event priority in the **safeethernet** Editor, see Chapter 10.6.8.

HIMax differentiates between Boolean and scalar events.

### 10.7.1 Boolean Events

- Changes of Boolean variables, e.g., of digital inputs.
- Alarm and normal state: They can be arbitrarily assigned to the variable states.

The parameters for the **Boolean** events are entered in the Alarm & Event Editor of the resource; the editor contains the following tabs:

| Column           | Description  |  | Range of values                 |
|------------------|--|--|---------------------------------|
| Name             | Name of the event definition   |  | Text, max. 31 characters.       |
| Global Variable  | Name of the assigned global variable (added using a drag&drop operation) |  |                                 |
| Data Type        | Data type of the global variable; it cannot be modified.                 |  | BOOL                            |
| Event source     | CPU event<br>IO event<br>Auto event<br><br>Default value: CPU event      | The timestamp is created on a processor module. The processor module creates all the events in each of its cycle.<br>The time stamp is created on a suitable I/O module (e.g., DI 32 04).<br>A CPU event and, if available, IO events of the I/O module are created. | CPU Event, IO Event, Auto Event |
| Alarm when FALSE | Activated<br>Deactivated<br><br>Default value: Deactivated               | If the global variable value changes from TRUE to FALSE, an event is triggered.<br>If the global variable value changes from FALSE to TRUE, an event is triggered.   | Checkbox activated, deactivated |
| Alarm Text       | Text specifying the alarm state  |  | Text                            |
| Alarm priority   | Priority of the alarm state<br>Default value: 1                          |  | 1...1000                        |

|                               |   |  |   |
|-------------------------------|---|--|---|
| Alarm Acknowledgment Required | Activated<br>Deactivate<br>Default value: Deactivated | The alarm state must be confirmed by the user (acknowledgement)<br>The alarm state may not be confirmed by the user<br>Text specifying the alarm state | Checkbox activated, deactivated<br>Text |
| Return to Normal Text         |   | Priority of the normal state   | 1...1000                                |
| Return to Normal Ack Required |   | The normal state must be confirmed by the user (acknowledgement)<br>Default value: Deactivated   | Checkbox activated, deactivated         |

Table 278: Parameters for Boolean Events

### 10.7.2 Scalar Events

- Exceedance of the limits defined for a scalar variable, e.g., an analog input.
- Two upper limits and two lower limits are possible.  
For the limit values, the following condition must be met:  
Superior limit  $\geq$  upper limit  $\geq$  normal area  $\geq$  lower limit  $\geq$  inferior limit.  
An hysteresis is effective in the following cases:
  - If the value falls below the upper limit
  - If the value exceeds the lower limit
 An hysteresis is defined to avoid a needless large number of events when a global variable strongly oscillate around a limit.

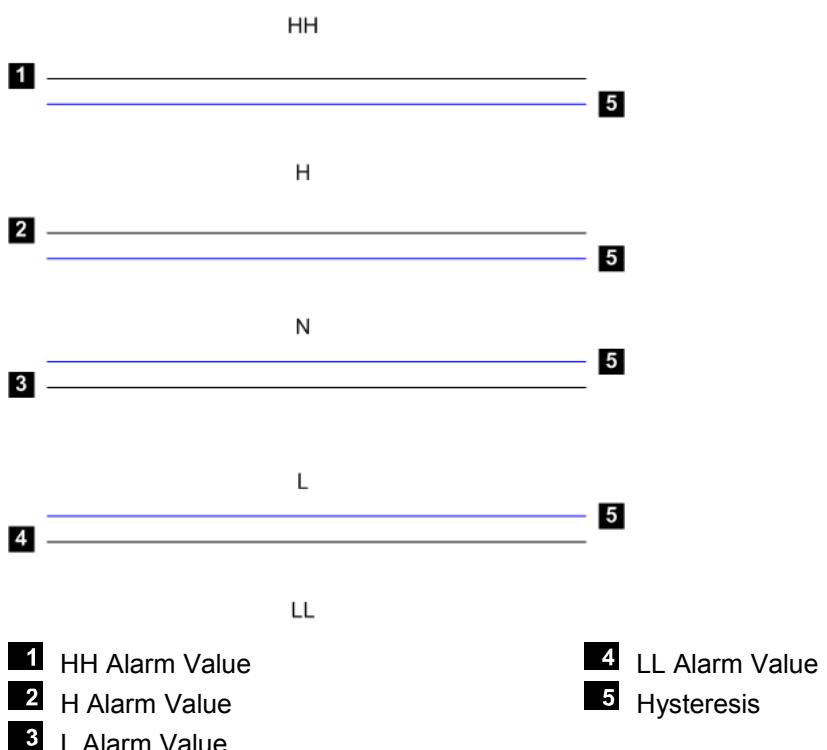


Figure 89: Five Areas of a Scalar Event

The parameters for the **scalar** events are entered in the Alarm & Event Editor of the resource; the editor contains the following tabs:

| Column                           | Description   | Range of values                       |
|----------------------------------|---|---------------------------------------|
| Name                             | Name of the event definition  | Text, max. 31 characters.             |
| Global Variable                  | Name of the assigned global variable (added using a drag&drop operation)  |                                       |
| Data Type                        | Data type of the global variable; it cannot be modified.  | depending on the global variable type |
| Event source                     | CPU event The timestamp is created on a processor module.<br>The processor module creates all the events in each of its cycle.<br><br>I/O Event The timestamp is built on an appropriate I/O module (e.g., AI 32 02).<br><br>Auto event A CPU event and, if available, IO events of the I/O module are created.<br><br>Default value: CPU Event | CPU Event, IO Event, Auto Event       |
| HH Alarm Text                    | Text specifying the alarm state of the upper limit.   | Text                                  |
| HH Alarm Value                   | Upper limit triggering an event. Condition:<br>$(\text{HH Alarm Value} - \text{Hysteresis}) > \text{H Alarm Value}$ or $\text{HH Alarm Value} = \text{H Alarm Value}$   | depending on the global variable type |
| HH Alarm Priority                | Priority of the upper limit; default value: 1   | 1...1000                              |
| HH Alarm Acknowledgment Required | Activated The user must confirm that the upper limit has been exceeded (acknowledgment).<br><br>Deactivate The user may not confirm that the upper limit has been exceeded.<br><br>Default value: Deactivated   | Checkbox activated, deactivated       |
| H Alarm Text                     | Text specifying the alarm state of the upper limit.   | Text                                  |
| H Alarm Value                    | Upper limit triggering an event. Condition:<br>$(\text{H Alarm Value} - \text{Hysteresis}) > (\text{L Alarm Value} + \text{Hysteresis})$ or $\text{H Alarm Value} = \text{L Alarm Value}$   | depending on the global variable type |
| H Alarm Priority                 | Priority of the upper limit; default value: 1   | 1...1000                              |
| H alarm acknowledgment required  | Activated The user must confirm that the upper limit has been exceeded (acknowledgment).<br><br>Deactivate The user may not confirm that the upper limit has been exceeded.<br><br>Default value: Deactivated   | Checkbox activated, deactivated       |
| Return to Normal Text            | Text specifying the alarm state   | Text                                  |
| Return to Normal Severity        | Priority of the normal state; default value: 1  | 1...1000                              |
| Return to Normal Ack Required    | The normal state must be confirmed by the user (acknowledgement); default value: Deactivated  | Checkbox activated, deactivated       |
| L Alarm Text                     | Text specifying the alarm state of the lower limit.   | Text                                  |

|                                  |   |                                       |
|----------------------------------|---|---------------------------------------|
| L Alarm Value                    | Lower limit triggering an event. Condition:<br>(L Alarm Value + Hysteresis) < (H Alarm Value - Hysteresis)<br>or L Alarm Value = H Alarm Value  | depending on the global variable type |
| L Alarm Priority                 | Priority of the lower limit; default value: 1   | 1...1000                              |
| L alarm acknowledgment required  | Activated The user must confirm that the lower limit has been exceeded (acknowledgment).<br>Deactivate The user may not confirm that the lower limit has d been exceeded.<br>Default value: Deactivated   | Checkbox activated, deactivated       |
| LL Alarm Text                    | Text specifying the alarm state of the lowest limit.  | Text                                  |
| LL Alarm Value                   | Lower limit triggering an event. Condition:<br>(LL Alarm Value + Hysteresis) < (L Alarm Value) or<br>LL Alarm Value = L Alarm Value   | depending on the global variable type |
| LL Alarm Priority                | Priority of the lowest limit; default value: 1  | 1...1000                              |
| LL alarm acknowledgment required | Activated The user must confirm that the lowest limit has been exceeded (acknowledgment).<br>Deactivate The user may not confirm that the lowest limit has d been exceeded.<br>Default value: Deactivated | Checkbox activated, deactivated       |
| Alarm Hysteresis                 | The hysteresis avoids that many events are continuously created when the process value often oscillate around a limit.  | depending on the global variable type |

Table 279: Parameters for Scalar Events

## 10.8 Parameters for the X-OPC Server Properties

- i** SILworX is used to configure and operate the entire X OPC server. Like a controller, the X-OPC server can be loaded, started and stopped from within the SILworX Control Panel.

### 10.8.1 OPC Server Set

The OPC set is used as common platform for configuring up to two OPC servers.

The OPC Server Set properties for the two redundant X-OPC servers are automatically set identical.

#### To create a new OPC server set

1. In the structure tree, open **Configuration**.
2. Right-click **Configuration** and select **New, OPC Server Set** to create a new OPC server set.
3. Right-click **OPC Server Set** and select **Properties**. Accept the default values.

The Properties dialog box for the OPC server set contains the following parameters:

| Element            | Description  |
|--------------------|--|
| Name               | Name of the OPC server set. A maximum of 31 characters.  |
| Safety Time [ms]   | <p>The safety time is the time in milliseconds within which the X-OPC server must react to an error.</p> <p>Condition:<br/> <math>\text{safety time} \geq 2 * \text{watchdog time}</math></p> <p>Range of values:<br/>           2000...400 000 ms</p> <p>Default value: 20 000 ms</p>   |
| Watchdog Time [ms] | <p>The watchdog time is the maximum time that the OPC server may require to complete a program cycle. If the defined watchdog time is exceeded (the execution of a program cycle takes too long), the X-OPC server is stopped.</p> <p>Condition:<br/> <math>\text{WDT} \geq 1000 \text{ ms and } \leq 0.5 * \text{safety time}</math></p> <p>Range of values:<br/>           1000...200 000 ms</p> <p>Default value: 10 000 ms</p> |

|                        |  |
|------------------------|--|
| Main Enable            | <p>The setting of the 'Main Enable' OPC switch affects the function of the other OPC switches.</p> <p>If 'Main Enable' is deactivated, the parameters set for the other OPC switches cannot be modified while the user program is being processed (the controller is in RUN).</p> <p>Default value: activated</p>  |
| Autostart              | <p>Autostart defines if the OPC configurations may be automatically started with a cold start or a warm start after powering up or booting the controller or should not be started (Off).</p> <p>If Autostart is deactivated, the X-OPC server adopts the STOP/VALID CONFIGURATION state after booting.</p> <p>Default value: Deactivated</p>  |
| Start Allowed          | <p>Only if <i>Start Allowed</i> is activated, an X-OPC server can be started from within the programming tool.</p> <p>If <i>Start Allowed</i> is deactivated, the X-OPC server cannot be started from within the programming tool. In this case, the X-OPC server can only be started if <i>Autostart</i> is activated and the host PC is switched on or rebooted.</p> <p>If neither <i>Autostart</i> nor <i>Start Allowed</i> is activated, the X-OPC server cannot be started. This can be required, for instance, during maintenance actions to prevent the system from starting.</p> <p>Default value: activated</p> |
| Load Allowed           | <p>If <i>Load Allowed</i> is deactivated, no (new) OPC configuration can be loaded into the controller.</p> <p>Deactivate <i>Load Allowed</i> to avoid that the OPC configuration loaded into the X-OPC server is overwritten.</p> <p>Default value: activated</p>   |
| Reload Allowed         | No function yet!   |
| Global Forcing Allowed | <p><i>Global forcing</i> can only be started if <i>Global Forcing Allowed</i> is activated.</p> <p><b>i</b> The Force Editor can also be used to display variable contents if <i>Global Forcing Allowed</i> is deactivated.</p> <p>Default value: Deactivated</p>  |

|                                |   |
|--------------------------------|---|
| Global Force Timeout Reaction  | <p>If <i>Global Force Timeout Reaction, Stop Resource</i> is selected, the X-OPC server enters the STOP state after the preset force time has expired. All the outputs of the X-OPC server are set to LOW.</p> <p>If <i>Global Force Timeout Reaction, Stop Forcing Only</i> is selected, the X-OPC server continues executing the OPC configuration after the force time has expired.</p> <p><b>i</b> If forcing is allowed, carefully check the setting for 'Stop at Force Timeout'. Also observe the instructions provided in the Safety Manual.</p> <p>Default value: Stop Resource</p> |
| Max.Com. Time Slice ASYNC [ms] | <p><i>Max. Com. Time Slice ASYNC [ms]</i> is the time in milliseconds that is reserved in each X-OPC server cycle for performing all the communication tasks scheduled for P2P communication.</p> <p>Default value: 500 ms</p>  |
| Target Cycle Time [ms]         | <p>Target cycle time for the X-OPC Server</p> <p>Default value: 50 ms</p>   |
| safeethernet CRC               | <p>-Current version<br/>SILworX V2.36</p>   |
| Namespace Separator            | <p>Dot .<br/>Slash /<br/>Colon :<br/>Backslash \</p> <p>Default value: Dot</p>  |
| Namespace Type                 | <p>Depending on the OPC client requirements, the following name space types can be set:</p> <ul style="list-style-type: none"> <li>- Hierarchical Namespace</li> <li>- Flat Namespace</li> </ul> <p>Default value: Hierarchical Namespace</p>   |
| Changeless Update              | <p>Setting according to the OPC client requirement</p> <p>Activated:<br/>If <i>Changeless Update</i> is selected and 'OPC Group UpdateRate' has expired, the X-OPC server provides all items to the OPC client.</p> <p>Deactivated:<br/>If <i>Changeless Update</i> is not selected, only the modified values are provided to the OPC client (this behavior is in accordance with the OPC Specification).</p>   |
| Cycle Delay [ms]               | <p>The cycle delay limits the CPU load of the PC due to the X-OPC server to allow other programs to be run.</p> <p>Range of values: 1...100 ms</p> <p>Default value: 5 ms</p>   |

|                                  |  |
|----------------------------------|--|
| Short Tag Names for DA           | This parameter can only be activated if <i>Flat Namespace</i> is selected.<br>It is an option in which data and event are offered to the OPC client without any further context (path name).<br>Default value: Deactivated         |
| Simple-Events for CPU I/O Events | Never<br>Only at Start<br>Allways  |
| Short tag-names for A&E          | This parameter can only be activated if <i>Flat Namespace</i> is selected.<br>It is an option in which data and event sources are offered to the OPC client without any further context (path name).<br>Default value: Deactivated |

Table 280: Properties

## 10.8.2 OPC Server

### To create a new OPC server

1. In the structure tree, open **Configuration, OPC Server Set**.
2. Right-click **OPC Server Set** and select **New, OPC Server** to create a new OPC server.
3. Right-click **OPC Server** and select **Properties**.

The Properties dialog box for the OPC server contains the following parameters:

| Element          | Description  |
|------------------|--|
| Name             | Name for the OPC server. A maximum of 31 characters. |
| System ID [SRS]  | Default value: 60000                                 |
| Namespace Prefix |  |

Table 281: Properties

### To open the OPC host

1. In the structure tree, open **Configuration, OPC Server Set, OPC Server**.
2. Right-click **OPC Host** and select **Edit** to get an overview of the IP interfaces.

The Edit dialog box for the OPC host contains the following parameters:

| Element            | Description  |
|--------------------|--|
| PADT Port          | Default value: 25138   |
| Name               | Name for the OPC server set. A maximum of 31 characters.   |
| IP Address         | IP address of the host PC.<br>Default value: 192.168.0.1   |
| Standard Interface | It must be selected if the host PC is equipped with more than one Ethernet port.<br>Default value: activated |
| HH Port            | Default value: 15138   |

Table 282: Edit

## 10.9 Uninstalling the X-OPC Server

### To uninstall the X-OPC server

1. Open **Start, Settings, Control Panel, Software** in Windows.
2. From the list, select the X-OPC server to be uninstalled and click the **Remove** button.
3. Follow the instructions of the Uninstall Wizard.

## 11 Synchronous Serial Interface

The synchronous serial interface (SSI) is a non-safety-related interface Schnittstelle for absolute encoders (angle measuring system).

The SSI submodule allows synchronization of up to three absolute encoders with a common pulse and thus obtaining the X-Y-Z position simultaneously.

Prior to starting up the SSI submodule, the COM must be configured using the ComUserTask and the user program logic must be created using the programming tool SILworX.

### 11.1 System Requirements

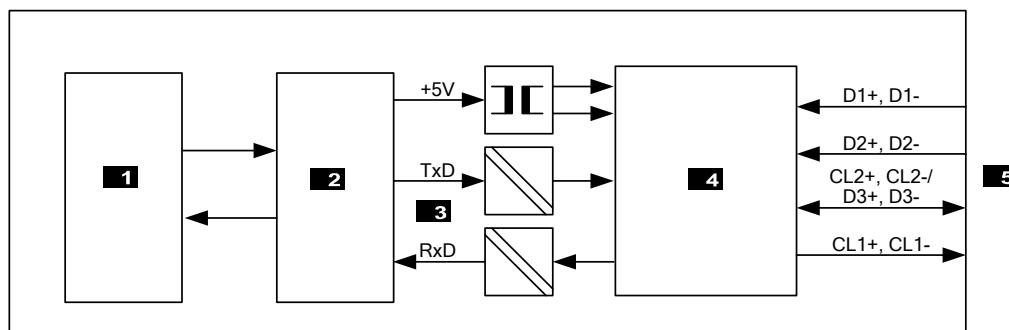
Equipment and system requirements

| Element          | Description   |
|------------------|---|
| Controller       | HIMax with COM module<br>HIMatrix: CPU OS version 7 and beyond.24 and COM OS version 12 and beyond.30                   |
| Processor module | The interfaces on the processor module may not be used for SSI.   |
| COM module       | If the serial fieldbus interfaces (FB1 or FB2) are used, they must be equipped with the SSI submodule, see Chapter 3.6. |
| Activation       | Software activation code required, see Chapter 3.4.   |

Table 283: Equipment and System Requirements for the ComUserTask

### 11.2 Block Diagram

The SSI submodule is electrically isolated from the HIMA controller.



- 1** Processor Module (CPU)
- 2** Communication Module (COM)
- 3** Internal Serial Interface

- 4** SSI Submodule
- 5** SSI, FB1 or FB2

Figure 90: Block Diagram

### 11.3 D-Sub Connectors FB1 and FB2

If the SSI submodule is used, apply the pin assignment of D-SUB connectors FB1 and FB2 as specified in Chapter 3.6.

### 11.4 Configuring Data Exchange between COM Module and SSI Submodule

Data is exchanged between the communication module (COM) **2** and the SSI submodule **4** via the internal serial interface **3** as configured with the ComUserTask, see Chapter 13 *ComUserTask*.

The data protocol created in the ComUserTask for exchanging data between the COM module and the SSI submodule must be set to the following parameters:

- Baud rate 115.2 kBit/s
- Data length: 8 bits
- Parity even
- 1 stop bit

### 11.5 Configuration of the SSI

The SSI **5** is configured using the SSISTART start command byte, see Table 284.

Whenever a new sensor data record (current position) is required, the SSISTART start command byte must be set in the user program and forwarded to the the SSI submodule **4**.

The SSISTART start command byte to be sent has the following format:

| Bit                 | Description   |
|---------------------|---|
| 7                   | Auxiliary bit<br>This bit defines the assignment of the start command byte.<br><br>If bit 7 = 1   |
| <b>If bit 7 = 1</b> |   |
| 6 ... 4             | The following setting is possible for the SSI clock<br>000 62.5 kHz<br>001 125 kHz<br>010 250 kHz<br>011 500 kHz  |
| 3                   | It switches pins 3 and 8 either for use as a data input or as a pulsed output.<br>0: D3+, D3- switched for use as an input<br>1: CL2+, CL2- switched for use as a pulsed output |
| 2                   | Pulsed output CL1<br>0: Activated<br>1: Deactivated:  |
| 1                   | Not used  |
| 0                   | Pulsed output CL2<br>0: Activated<br>1: Deactivated:  |
| <b>If bit 7 = 0</b> |   |
| 6 ... 0             | Not used  |

Table 284: Data Format of the SSISTART Start Command Bytes

The SSI submodule **3** transmits the sensor data determined via the SSI to the COM **4** via the internal serial interface **2** in the following format and order.

The sensor data can be evaluated in the user program for determining the X-Y-Z position (note: the data bits are transferred to the user program in the same order as provided by the sensor).

| No. | Channel   | Data bit   |
|-----|-----------|------------|
| 1   | Channel 1 | D47 .. D40 |
| 2   |           | D39 .. D32 |
| 3   |           | D31 .. D24 |
| 4   |           | D23 .. D16 |
| 5   |           | D15 .. D8  |
| 6   |           | D7 .. D0   |
| 7   | Channel 2 | D47 .. D40 |
| 8   |           | D39 .. D32 |
| 9   |           | D31 .. D24 |
| 10  |           | D23 .. D16 |
| 11  |           | D15 .. D8  |
| 12  |           | D7 .. D0   |
| 13  | Channel 3 | D47 .. D40 |
| 14  |           | D39 .. D32 |
| 15  |           | D31 .. D24 |
| 16  |           | D23 .. D16 |
| 17  |           | D15 .. D8  |
| 18  |           | D7 .. D0   |

Table 285: Format and Order of the Sensor Data

### 11.5.1 Wire Lengths and Recommended Clock Rates

The following table specifies the recommended clock rates for the SSI according with the field wire lengths.

| Wire length / m | Clock rate /kHz |
|-----------------|-----------------|
| < 25            | $\leq 500$      |
| < 50            | < 400           |
| < 100           | < 300           |
| < 200           | < 200           |
| < 400           | < 100           |

Table 286: Recommended Clock Rates According to the Field Wire Lengths

## 11.6 Application Notes

The SSI sensors must be supplied in the field with an external power supply since the SSI submodule is electrically isolated from the HIMA controller.

The following applications are possible with an SSI submodule:

- Connection of 3 sensors for simultaneously determining the x, y and z coordinates  
1 SSI clock (CL1+, CL1-) and 3 data channels (D1+, D1-, D2+, D2-, D3+, D3-) for simultaneously determining the x, y and z coordinates.  
All the 3 sensors are energized from the same clock (CL1+, CL1-) and thus from the same clock rate.
- Connection of 2 sensors for simultaneously determining the x and y coordinates  
1 SSI clocks (CL1+, CL1-, CL2+, CL2-) and 2 data channels (D1+, D1-, D2+, D2-) for simultaneously determining the x and y coordinates.  
Both sensors are energized from 2 different clocks (CL1+, CL1-, CL2+, CL2-). They have the same clock rate.
- Connection of 1 sensor  
1 SSI clock (CL1+, CL1- or CL2+, CL2-) and 1 data channel (D1+, D1- or D2+, D2-).



If the special conditions X are observed, the SSI submodule may be mounted in zone 2 (EC Directive 94/9/EC, ATEX).

## 12 HART

The HART (Highway Addressable Remote Transducer) protocol allows digital fieldbus communication during which the HART signal is superimposed onto the (4...20 mA) analog current signal.

The connector boards can be used to allow the X-HART 32 01 module to be combined with analog input or output modules.

The HART signal is used to transfer the measuring and device data to sensors or actuators. The X-HART modules transfer the HART data to the assigned X-COM module within the HIMax system. The X-COM module transfers the HART data via the HART over IP protocol to the host (asset management system or HART OPC server).

The X-COM module and the assigned X-HART modules form a I/O system as referred to in the HART specification.

The HIMax system allows use of up to 2 I/O systems (i.e., 2 X-COM modules with HART over IP protocol). A X-HART module can only be assigned to one X-COM module.

### 12.1 System Requirements

Equipment and system requirements for HART over IP:

| Element          | Description   |
|------------------|---|
| Controller       | HIMax with X-COM module and X-HART module   |
| Processor module | The Ethernet interfaces on the processor module are not be used for HART over IP. |
| COM module       | Ethernet 10/100BaseT  |
| Activation       | The HART over IP protocol need not be licensed on the X-COM module.               |

Table 287: Equipment and System Requirements for HART over IP

### HART over IP Features

| Properties                                    | Description  |
|---|--|
| Safety-related                                | No   |
| Transfer rate                                 | 100 Mbit/s full duplex   |
| Transmission path                             | Ethernet interfaces on the X-COM module<br>Ethernet interfaces in use can simultaneously be used for additional protocols. |
| Max. number of HART Over IP protocol instance | A maximum of one HART over IP protocol instances can be configured for each X-COM module.                                  |
| Max. number of HART over IP sessions via UDP  | Max. 2 on each COM module  |
| Max. number of HART over IP sessions via TCP  | Max. 2 on each COM module  |

Table 288: Properties of the HART Over IP Protocol Instance

### 12.2 Creating a HART Over IP Protocol Instance

#### To create a HART over IP protocol instance

1. In the structure tree, open **Configuration, Resource, Protocols**.
2. Select **New, HART Protocol** from the context menu for Protocols to add a new HART protocol.
3. Right-click the HART protocol and select **Properties of the X-Com Module**.  
The default settings may be retained for the first configuration.

### 12.2.1 Properties

The Properties dialog box for the HART protocol contains the following parameters:

| Element                           | Description  |                                |
|-----------------------------------|--|--------------------------------|
| Name                              | Name for the HART protocol, composed of a maximum of 32 characters.  |                                |
| Module                            | Selection of the COM module within which HART over IP is processed.  |                                |
| Activate Max. $\mu$ P Budget      | Activated:<br>Adopt the $\mu$ P budget limit from the <i>Max. <math>\mu</math>P Budget in [%]</i> field.<br>Deactivated:<br>Do not use the $\mu$ P budget limit for this protocol.<br>Default value: activated |                                |
| Max. $\mu$ P budget in [%]        | Maximum $\mu$ P budget for the module that can be used for processing the protocols.<br><br>Range of values: 1...100%<br>Default value: 30%  |                                |
| Polling Address                   | Polling Address of X-COM<br>Range of values: 0...63<br>Default value: 0  |                                |
| Use Standard HART TCP Port (5094) | Activated  | The TCP connection is enabled  |
|                                   | Deactivated  | The TCP connection is disabled |
|                                   | Default value: Activated, TCP port 5094 is used  |                                |
| Use Second HART TCP Port          | Activated  | The TCP connection is enabled  |
|                                   | Deactivated  | The TCP connection is disabled |
|                                   | Default value: Deactivated   |                                |
| Second HART TCP Port              | Default value: 20004   |                                |
| Use Standard HART UDP Port (5094) | Activated  | The UDP connection is enabled  |
|                                   | Deactivated  | The UDP connection is disabled |
|                                   | Default value: Activated, UDP port 5094 is used  |                                |
| Use Second HART UDP Port          | Activated  | The UDP connection is enabled  |
|                                   | Deactivated  | The UDP connection is disabled |
|                                   | Default value: Deactivated   |                                |
| Second HART UDP Port              | Default value: 20004   |                                |

Table 289: HART Protocol Properties

### 12.3 Structure Overview of the HART over IP Installation

This chapter provides an overview of the systems, from the HART field device up to engineering and asset management tool, and the type of exchanged data.

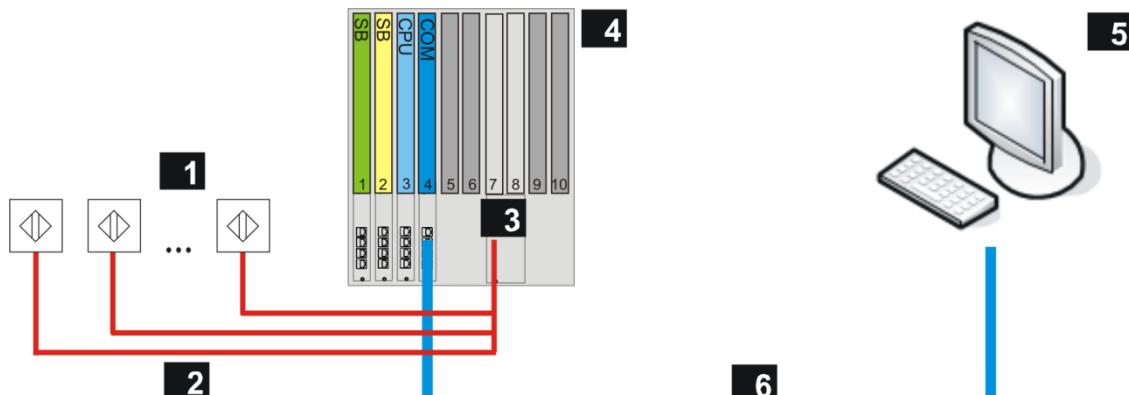


Figure 91: Structure of the HART over IP Installation



For further information on how to wire and configure the X-HART module with analog input or output modules, refer to the corresponding manual (HI 801 307 E).

## 12.4 Start-up

The start-up process includes the following phases:

1. Set the parameters for the analog HIMax module, X-HART module and X-COM module
2. Create the SILworX user program, generate and load the code
3. Install and configure
  - a HART/OPC server and OPC client
  - or a FDT/DTM asset management system

### 12.4.1 X-AI/O Module

For HART communication, the HART field devices must be connected to the analog input or output HIMax modules.

### 12.4.2 X-HART Module

The connector boards can be used to allow the X-HART module to be combined with the analog HIMax input or output modules.

Configuration notes:

| Parameter                           | Description   |
|-------------------------------------|---|
| HART ID                             | The HART ID corresponds to the IO Card Number when addressing field devices using HART command 77.  |
| X-COM                               | Select the module to be used for transmitting the HART over IP protocol.  |
| HART commando variables (submodule) | These variables are used to control the HART command filter function via the user program, see the module-specific manual for X-HART 32 01. If the variables are connected via digital inputs, the read/write rights can be activated, e.g., using a key switch.<br>The shared lock filter function must be deactivated to allow for HART-conform behavior. |
| HART commando variables (channels)  | These variables can be used to activate the individual HART channels of a sensor/actuator.  |

Table 290: Parameters and Variables Required for the Configuration



For detailed information on HART commands, refer to the documentation provided by HART Foundation (HCF).

### 12.4.3 X-COM Module

The X-COM module used to transmit the HART over IP protocol must be configured in SILworX through the protocol configuration or via online command.

The X-COM module supports the following HART commandos:

0, 1, 2, 3, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 38, 41, 42, 48, 71, 74, 75, 76, 77, 78, 84, 85, 86, 87, 88, 89, 94, 106, 512, 513

### 12.4.4 Creating the SILworX User Program, Generating and Loading the Code

The user programs and the corresponding global variables are created with SILworX.

**To complete the HIMax controller's start-up for HART communication**

1. Generate the code for the resource and user program(s)
2. Load the generated code into the HIMax controller.
3. Start the HIMax controller.

The HIMax controller started operation.

**12.4.5 HART/OPC Server or FDT/DTM Asset Management System**

For configuring and monitoring the HART field devices

A suitable HART OPC server can be obtained by HART Foundation.



The two device drivers *CommDTM* and *DeviceDTM* for the HIMax system can be obtained by HIMA.

---

**12.5 Online View of the X-COM Module**

The settings for the HART protocol can be controlled and verified from within the online view of the X-COM module. Details about the current status of the field devices and X-COM module are displayed.

**To open the online view of the Hardware Editor for monitoring the HART protocol:**

1. Right-click the **Hardware** structure tree node and select **Online** from the context menu.
2. In the **System Login** window, enter the access data to open the online view for the hardware.
3. Double-click **X-COM Module** and select the **HART Protocol** structure tree node.

**To update the device list**

1. In the structure tree, select **HART Protocol, Device List**.
2. Right-click and select **Update Device List**.

### 12.5.1 View Box (HART Protocol)

The view box displays the following values of the selected HART protocol.

| Element                               | Description  |
|---------------------------------------|--|
| Name                                  | Name for the HART protocol, composed of a maximum of 32 characters.  |
| Planned µP Budget [%]                 | Value displayed for the planned maximum µP budget of the X-COM module, which may be produced during the protocol's processing.   |
| Current µP Budget [%]                 | Value displayed for the current maximum µP budget of the X-COM module, which is being produced during the protocol's processing.   |
| Polling Address                       | The polling address of X-COM is displayed<br>Range of values: 0...63   |
| Unique address for X-COM              | The five-byte address of the X-COM (unique address) is displayed.  |
| Standard HART TCP Port Number         | The standard TCP port used for HART over IP is displayed online  |
| Second HART TCP Port Number           | The TCP port additionally or alternatively used for HART over IP is displayed online   |
| Standard HART UDP Port Number         | The standard UDP port used for HART over IP is displayed online  |
| Second HART UDP Port Number           | The UDP port additionally or alternatively used for HART over IP is displayed online   |
| Number of HART Devices                | The number of HART devices currently detected as connected is displayed.<br>The X-COM, which is also a HART device, is not included in this number.  |
| Number of X-HART Modules              | The number of X-HART 32 01 modules (IO cards) detected and belonging to this X-COM 01 module is displayed  |
| Status Device Interlock               | It displays the device interlock status. The device interlock (interlock for the HART IO subsystem) is triggered by the host through HART command 71.<br><br>Range of values:<br>See HCF_SPEC-183 (Common Tables) Table 25<br>Zero - The device is not locked<br>Not equal to zero - Lock device status code<br>Default value: 0 |
| Device Interlock through Host with IP | With the device interlock (device interlock status is not zero), this field displays the IP address of the host that triggered the interlock (i.e., which sent HART command 71)<br><br>Range of values: <IP address><br>Default value: 0   |

Table 291: Online View of the HART Protocol

### 12.5.2 Online View of the Device List

The Device List view box displays the following values.

| Element                   | Description   |
|---------------------------|---|
| Device Index              | The device index is displayed online<br>Range of values 0...65535 (decimal, 2 bytes)  |
| I/O Card Number           | The IO card number to which the device is connected is displayed online<br>Range of values: 1...249 (decimal, 1 byte) for connected devices<br>Range of values: 251 (None) for the X-COM itself                     |
| Channel no.               | The channel number to which the device is connected is displayed online<br>Range of values: 1...32 (decimal, 1 byte) for connected devices, see Chapter 12.5.3.<br>Range of values: 251 (None) for the X-COM itself |
| Manufacturer ID           | The ID of the device manufacturer is displayed online<br>Range of values: 0x00...0xFFFF (hexadecimal, 2 bytes)  |
| Expanded Device Type Code | The expanded device type code of the device is displayed online<br>Range of values: 0x00...0xFFFF (hexadecimal 2 bytes)   |
| Device ID                 | The ID of the device is displayed online<br>Range of values: 0x00 0x00 0x00 ... 0xFF 0xFF 0xFF (hexadecimal)  |
| HART Version              | The HART version (Universal Command Revision Level) of the device is displayed online.<br>Range of values 0...255 (decimal 1 byte)  |
| Long Tag                  | The device long tag is displayed online (HART V.7 and beyond)<br>The device message is displayed for devices prior to HART V.7, which do not support the long tag.<br>Range of values 32 characters (Latin-1)       |
| Rack.Slot I/O Card        | The IO card slot (RackSlot) is displayed online.<br>Format: Rack.Slot<br>Range of values: 0-15.0-18   |
| Telegram Counter STX      | The telegram counter for the device's commands (Stx) are displayed online<br>Range of values 0...65535 (decimal 2 bytes revolving)  |
| Telegram Counter ACK      | The telegram counter for the device's responds (Ack) are displayed online<br>Range of values 0...65535 (decimal 2 bytes revolving)  |
| Telegram Counter BACK     | The telegram counter for the device's burst responds (Back) are displayed online<br>Range of values 0...65535 (decimal 2 bytes revolving)   |

Table 292: Online View of the HART Protocol

### 12.5.3 HART Field Device Addressing

| Channel number<br>(X-HART 32 01 front plate) | Channel address<br>(decimal) | Channel address<br>(hexadecimal) |
|--|------------------------------|----------------------------------|
| 1...32                                       | 0...31                       | 0x00...0x1f                      |

Table 293: HART Field Device Addressing

The channel numbers displayed in the X-COM 01 online view correspond to the channel numbers specified on the front plate of the X-HART 32 01 module (channel count starting with 1).

If a connected HART field device is addressed, the channel address (channel number upon command 77) = channel number – 1

Example:

Channel number = 15

The field device is addressed with channel address = 14 (0x0e)

## 13 ComUserTask

In addition to the user program created in SILworX, a C program can also be run in the controller.

As ComUserTask, this non-safety-related C program operates interference-free to the safe processor module on the controller's communication module.

The ComUserTask has its own cycle time that does not depend on the CPU cycle.

This allows the users to program in C any kind of applications and implement them as ComUserTask, for example:

- Communication interfaces for special protocols (TCP, UDP, etc.).
- Gateway function between TCP/UDP and serial communication.

### 13.1 System Requirements

Equipment and system requirements

| Element              | Description  |
|----------------------|--|
| Controller           | HIMax with COM module<br>HIMatrix: CPU OS versions beyond 7 and COM OS versions beyond 12  |
| Processor module     | The Ethernet interfaces on the processor module may not be used for ComUserTask.   |
| Communication module | Ethernet 10/100BaseT<br>Pin assignment of the D-sub connectors FB1 and FB2 e.g., for RS 232<br>If the serial fieldbus interfaces (FB1 or FB2) oder FB3 for HIMatrix (RS 485) are used, they must be equipped with an optional HIMA submodule, see Chapter 3.6. |
| Activation           | Software activation code required, see Chapter 3.4.  |

Table 294: Equipment and System Requirements for the ComUserTask

Properties of the ComUserTask:

| Element            | Description   |
|--------------------|---|
| ComUserTask        | One ComUserTask can be configured for each HIMax/HIMatrix controller.                       |
| Safety-related     | No  |
| Data Exchange      | Configurable  |
| Code and data area | See Chapter 13.5.4  |
| Stack              | The stack is located in a reserved memory outside the code/data area.<br>See Chapter 13.5.4 |

Table 295: ComUserTask Properties

#### 13.1.1 Creating a ComUserTask

##### To create a new ComUserTask

1. In the structure tree, open **Configuration, Resource, Protocols**.
2. On the context menu for protocols, click **New, ComUserTask** to add a ComUserTask.
3. Right-click the **ComUserTask**, click **Properties** and select the **COM module**.  
Accept the default settings for the first configuration.

## 13.2 Requirements

In addition to the normal C commands, a specific library with defined functions is available for programming a ComUserTask (see Chapter 13.4).

### Development Environment

The development environment comprises the GNU C Compiler and Cygwin which are provided with the HIMA DVD (not included in ELOP II Factory) and are subject to the conditions of the GNU General Public License, (see [www.gnu.org](http://www.gnu.org)).

### Controller

In the HIMax/HIMatrix controllers, the ComUserTask has no access to the safe hardware inputs and outputs. If an access to the safe hardware inputs and outputs is required, a CPU user program must exist for connecting the variables (see 13.4.5).

## 13.3 Abbreviations

| Abbreviation | Description   |
|--------------|---|
| CUCB         | COM user callback<br>(CUCB_ Functions invoked by the COM) |
| CUIT         | COM user IRQ task   |
| CUL          | COM user library<br>(CUL_ Functions invoked in the CUT)   |
| CUT          | ComUserTask   |
| GNU          | GNU project   |
| IF           | Interface   |
| FB           | Fieldbus interface of the controller                      |
| FIFO         | First in first out (Data memory)                          |
| NVRam        | Non volatile random access memory,<br>non volatile memory |
| SSI          | Synchronous serial interface                              |

Table 296: Abbreviations

## 13.4 CUT Interface in SILworX

The process data communication of the ComUserTask runs between COM and CPU.

### ⚠ WARNING



The CUT code operates in the COM in an interference-free manner with the safe CPU. This ensures that the safe CPU is protected against the CUT code. Note that errors in the CUT code can disturb the entire COM function, thus affecting or stopping the controller's function. The CPU safety functions, however, are not compromised.

### 13.4.1 Schedule Interval [ms]

The ComUserTask is invoked in a predefined schedule interval [ms] during the controller's states RUN and STOP\_VALID\_CONFIG (COM module).

In SILworX, Schedule Interval [ms] can be set in the ComUserTask *Properties*.

| Schedule Interval [ms] |              |
|------------------------|--------------|
| Range of values:       | 10 .. 255 ms |
| Default value:         | 15 ms        |

Table 297: Schedule Interval [ms]



The COM processor time available to the CUT depends on the other configured COM functions such as `safeethernet` or Modbus TCP.

If CUT is not finished within the schedule interval, each call to restart the CUT is ignored until CUT has been processed.

### 13.4.2 Scheduling Preprocessing

#### In the controller's state RUN:

Before each CUT call, the COM module provides the process data of the safe CPU to the CUT in a memory area defined by CUT.

#### In the controller's state STOP:

No process data is exchanged between COM and safe CPU.

### 13.4.3 Scheduling Postprocessing

#### In the controller's state RUN:

After each CUT call, the COM module provides the process data of the CUT to the safe CPU.

#### In the controller's state STOP:

No process data is exchanged between COM and safe CPU.

### 13.4.4 STOP\_INVALID\_CONFIG

If the COM is in the STOP\_INVALID\_CONFIG state, CUT is not executed.

If the COM module enters the STOP\_INVALID\_CONFIG state and executes CUT or CUIT, these functions are terminated.

### 13.4.5 CUT Interface Variables (CPU<->CUT)

Configuration of non-safety-related process data communication between safe CPU and COM (CUT). In this context, the maximum process data volume may be exchanged per data direction. The maximum process data volume for standard protocols depends on the controller type, see Table 2.

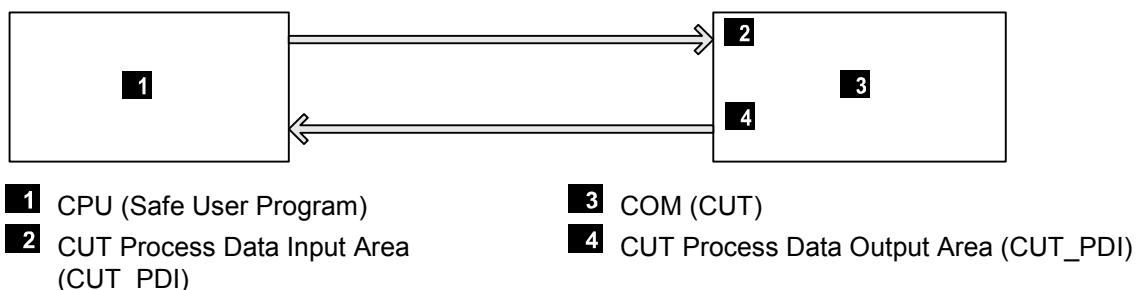


Figure 92: Process Data Exchange between CPU and COM (CUT)

All data types that are used in SILworX can be exchanged.

The data structure must be configured in SILworX.

The size of the data structures CUT\_PDI and CUT\_PDO (in the compiled CUT C code) must correspond to the size of the data structures configured in SILworX.



- If the data structures CUT\_PDI and CUT\_PDO are not available in the compiled C code or do not have the same size as the process data configured in SILworX, the configuration is invalid and the COM module enters the STOP\_INVALID\_CONFIG state.
- Process data communication takes place in the RUN state only.

### 13.4.6 Menu Function: Properties

The **Properties** function of the context menu for the ComUserTask appears the Properties dialog box.

| Element                             | Description   |                    |  |                   |   |
|-------------------------------------|---|--------------------|--|-------------------|---|
| Type                                | ComUserTask   |                    |  |                   |   |
| Name                                | Any unique name for a ComUserTask   |                    |  |                   |   |
| Force Process Data Consistency      | Activated:<br>Transfer of all protocol data from the CPU to the COM within a CPU cycle.<br>Deactivated:<br>Transfer of all protocol data from the CPU to the COM, distributed over multiple CPU cycles, each with 1100 bytes per data direction.<br>The cycle time of the controller can thus be reduced.<br>Default value: activated   |                    |  |                   |   |
| Module                              | Selection of the COM module within which the protocol is processed.   |                    |  |                   |   |
| Activate Max. µP Budget             | Not used  |                    |  |                   |   |
| Max. µP Budget in [%]               | Not used  |                    |  |                   |   |
| Behavior on CPU/COM Connection Loss | If the connection of the processor module to the communication module is lost, the input variables are initialized or continue to be used unchanged in the process module, depending on this parameter.<br>Example: the communication module is removed when communication is running.<br><b>If a project created with a SILworX version prior to V3 should be converted, this value must be set to Retain Last Value to ensure that the CRC does not change.</b><br><b>For HIMatrix controllers with CPU OS version prior to V8, this value must always be set to Retain Last Value.</b><br><br><table> <tr> <td>Adopt Initial Data</td> <td>Input variables are reset to their initial values.</td> </tr> <tr> <td>Retain Last Value</td> <td>The input variables retains the last value.</td> </tr> </table> | Adopt Initial Data | Input variables are reset to their initial values. | Retain Last Value | The input variables retains the last value. |
| Adopt Initial Data                  | Input variables are reset to their initial values.  |                    |  |                   |   |
| Retain Last Value                   | The input variables retains the last value.   |                    |  |                   |   |
| Schedule Interval [ms]              | The ComUserTask is invoked in a predefined schedule interval [ms] of controller (COM module), see Chapter 13.4.1.<br>Range of values: 10...255 ms<br>Default value: 15 ms   |                    |  |                   |   |
| User Task                           | Loadable file path, if already loaded   |                    |  |                   |   |

Table 298: PROFINET IO Device General Properties

### 13.4.7 Menu Function: Edit

The **Edit** menu function is used to open the tabs Process Variables and System Variables.

#### 13.4.7.1 System Variables

The **System Variables** tab contains the following system parameters for monitoring and controlling the CUT:

| Name                           | Function   |          |             |                    |   |                         |   |
|--------------------------------|--|----------|-------------|--------------------|---|-------------------------|---|
| Execution Time [DWORD]         | Execution time of the ComUserTask in $\mu$ s   |          |             |                    |   |                         |   |
| Real Schedule Interval [DWORD] | Time between two ComUserTask cycles in ms  |          |             |                    |   |                         |   |
| User Task State Control [WORD] | The following table shows how the user can control the ComUserTask with the <i>User Task State Control</i> system parameter: <table border="1" data-bbox="651 696 1333 898"> <thead> <tr> <th>Function</th><th>Description</th></tr> </thead> <tbody> <tr> <td>DISABLED<br/>0x8000</td><td>The application program locks the CUT (CUT is not started).</td></tr> <tr> <td>Autostart<br/>Default: 0</td><td>After termination a new start of the CUT is automatically allowed if the error is eliminated.</td></tr> </tbody> </table> | Function | Description | DISABLED<br>0x8000 | The application program locks the CUT (CUT is not started). | Autostart<br>Default: 0 | After termination a new start of the CUT is automatically allowed if the error is eliminated. |
| Function                       | Description  |          |             |                    |   |                         |   |
| DISABLED<br>0x8000             | The application program locks the CUT (CUT is not started).  |          |             |                    |   |                         |   |
| Autostart<br>Default: 0        | After termination a new start of the CUT is automatically allowed if the error is eliminated.  |          |             |                    |   |                         |   |
| State of the User Task [BYTE]  | 1 = RUNNING (CUT is running)<br>0 = ERROR (CUT is not running due to an error)   |          |             |                    |   |                         |   |

Table 299: ComUserTask System Variables

---

**TIP** If the CUT is terminated and restarted, the COM state STOP / LOADING DATA FROM FLASH is displayed from the flash memory even if the ComUserTask is in the RUN state.

---

### 13.4.7.2 Process Variables

#### Input Signals (COM->CPU)

The **Input Signals** tab contains the signals that should be transferred from the COM module (CUT) to the CPU (CPU input area).

#### CAUTION

**ComUserTask is not safety-related!**

**Non-safe variables of the ComUserTask must not be used for the safety functions of the CPU user program.**

#### Required entry in the C code

The C code of the COM User Tasks must have the following CUT\_PDO data structure for the COM outputs (CPU input area):

```
/* SILworX Input Records (COM->CPU) */
uword CUT_PDO[1] __attribute__ ((section("CUT_PD_OUT_SECT"), aligned(1)));
```

The size of the CUT\_PDO data structure must correspond to the size of the data inputs configured in SILworX.

#### Output Signals (CPU->COM)

The **Output Signals** tab contains the signals that should be transferred from the CPU (CPU output area) to the COM module (CUT).

#### Required entry in the C code

The C code of the ComUserTask must have the following CUT\_PDI data structure for the COM inputs (CPU output area):

```
/* SILworX Output Records (CPU->COM) */
uword CUT_PDI[1] __attribute__ ((section("CUT_PD_IN_SECT"), aligned(1)));
```

The size of the CUT\_PDI data structure must correspond to the size of the data outputs configured in SILworX.

## 13.5 CUT Functions

### 13.5.1 COM User Callback Functions

The COM User callback functions have all the **CUCB\_** prefix and are directly invoked from the COM when events occur.

- 
- i** All COM user callback functions must be defined in the user's C code!  
Also the CUCB\_IrqService function, which is not supported for HIMax and HIMatrix L2 (but for HIMatrix L3 with CAN module), must be defined in the user C-Code for reasons of compatibility.  
Funktion prototype: void CUCB\_IrqService(udword devNo) {}
- 

The COM user callback (CUCB) and the COM user library (CUL) functions share the stack and the same code and data memory. These functions mutually ensure the consistency of the data shared (variables).

### 13.5.2 COM User Library Functions

All COM user library functions and variables have the **CUL\_** prefix and are invoked in the CUT.  
All the CUL functions are available via the **libcut.a** object file.

### 13.5.3 Header Files

The two header files **cut.h** and **cut\_types.h** contain all function prototypes for CUL/CUCB and the related data types and constants.

For the data types, the following short cuts are defined in the header file **cut\_types.h**:

```
typedef unsigned long      udword;
typedef unsigned short     uword;
typedef unsigned char      ubyte;
typedef signed long       dword;
typedef signed short      word;
typedef signed char       sbyte;
#ifndef HAS_BOOL
typedef unsigned char      bool; // with 0=FALSE, otherwise TRUE
#endif
```

### 13.5.4 Code/Data Area and Stack for CUT

The code/data area is a coherent memory area that begins with the code segment and the initial data segment and continues with the data segments. In the HIMA linker files (**makeinc.inc.app**

and **section.dId**), the written segment sequence and the available storage capacity are predefined (this applies to HIMax/HIMatrix and HM31).

| Element       | HIMax V.4 and beyond | HIMax prior to V.4<br>HIMatrix L2 | HIMatrix L3 |
|---------------|----------------------|-----------------------------------|-------------|
| Start address | 0x780000             | 0x790000                          | 0x800000    |
| Length        | 512 kBytes           | 448 kBytes                        | 4 MB        |

Table 300: Memory Area for Code and Data

The stack is located in a reserved memory area defined when the COM operating system is started.

| Element     | HIMax / HIMatrix L2                         | HIMatrix L3                                 |
|-------------|---|---|
| End address | Dynamically (from the point of view of CUT) | Dynamically (from the point of view of CUT) |
| Length      | approx. 10 kByte                            | approx. 600 kByte                           |

Table 301: Stack Memory

### 13.5.5 Start Function CUCB\_TaskLoop

CUCB\_TaskLoop( ) is the starting function associated with the ComUserTask.

The ComUserTask program execution begins when this function is invoked (see Chapter 13.4.1 Schedule Interval[ms]).

#### Function Prototype:

void CUCB\_TaskLoop(udword mode)

#### Parameter:

The function has the following parameter (returns the value):

| Parameter | Description   |
|-----------|---|
| mode      | 1 = MODE_STOP corresponds to the mode STOP_VALID_CONFIG<br>2 = MODE_RUN controller's normal operation |

Table 302: Parameter

### 13.5.6 RS485 / RS232 IF Serial Interfaces

The used fieldbus interfaces must be equipped with the corresponding fieldbus submodules (hardware), see System documentation HIMax/HIMatrix.

#### Receive Telegrams from the perspective of the CUT application

The number of idle characters for identification of telegram boundaries (idle time) is set to 5 characters for the serial receiving via the COM user task.

A communication partner which consecutive sends several telegrams to CUT application, must insert at least 5 idle characters between the sent telegrams. By this, the UART driver identifies these telegrams as single telegrams.

The telegrams received by the UART driver are only forwarded to the CUT application, if at least 5 idle characters have been received.

#### Send telegrams from the perspective of the CUT application

The number of idle characters for identification of telegram boundaries (idle time) is set to 5 characters for the serial sending via the COM user task.

The COM user task application interface ensures that the serial telegrams are sent by the application consecutively in idle time intervals. It should be considered that the storage capacity of the CUT for sending telegrams is limited to one telegram . If the sending occurs in shorter intervals than the physical transmission of telegrams requires, the storage capacity of the CUT is exceeded and the error code CUL\_WOULDBLOCK is returned, see chapter 13.5.6.5

### 13.5.6.1 CUL\_AscOpen

The CUL\_AscOpen( ) function initializes the entered serial interface (*comId*) with the given parameters. After invoking the CUL\_AscOpen( ) function, the COM immediately begins receiving data via this interface.

The received data is stored in a FIFO software with a size of 1kByte for **each** initialized serial interface.

The data is stored until it is read out with the CUT function CUL\_AscRcv( ).



If data is read out of the FIFO more slowly than it is received, the new data is rejected.

---

#### Function Prototype:

```
udword CUL_AscOpen( Udword comId,  
                      Ubyte duplex,  
                      udword baudRate,  
                      ubyte parity,  
                      ubyte stopBits)
```

#### Parameters:

The function has the following parameters:

| Parameter  | Description   |
|--|---|
| comId  | Fieldbus interface (RS485, RS 232)<br>1 = FB1<br>2 = FB2<br>3 = FB3   |
| duplex   | 0 = Full duplex<br>1 = Half duplex  |
| baudRate   | 1 = 1200 Bit<br>2 = 2400 Bit<br>3 = 4800 Bit<br>4 = 9600 Bit<br>5 = 19200 Bi<br>6 = 38400 bits (maximum baud rate for HIMax prior to V.4)<br>7 = 57600 bits (maximum baud rate for HIMax V.4 and beyond)<br>8 = 115000 bits (HIMatrix only) |
| The data bit length is fixed and set to 8 data bits. Start, parity and stop bits must be added to these 8 data bits. |   |
| parity   | 0 = NONE<br>1 = EVEN<br>2 = ODD   |
| stopBits   | 1 = 1 bit<br>2 = 2 bits   |

Table 303: Parameter

**Return Value:**

An error code (udword) is returned.

The error codes are defined in the cut.h header file.

| Error Code         | Description  |
|--------------------|--|
| CUL_OKAY           | The interface was initialized successfully.                      |
| CUL_ALREADY_IN_USE | The interface is used by other COM functions or is already open. |
| CUL_INVALID_PARAM  | Incorrect parameters or parameter combinations transmitted.      |
| CUL_DEVICE_ERROR   | Other errors   |

Table 304: Return Value

### 13.5.6.2 CUL\_AscClose

The `CUL_AscClose()` function closes the serial interface entered in `comId`. In doing so, the data that has already been received but not read out with the function `CUL_AscRcv()` is deleted in FIFO.

#### Function Prototype:

```
Udword CUL_AscClose(udword comId)
```

#### Parameter:

The function has the following parameter:

| Parameter | Description   |
|-----------|---|
| comId     | Fieldbus interface (RS485, RS 232)<br>1 = FB1<br>2 = FB2<br>3 = FB3 |

Table 305: Parameter

#### Return Value:

An error code (udword) is returned.

The error codes are defined in the cut.h header file.

| Error Code        | Description   |
|-------------------|---|
| CUL_OKAY          | The interface was closed successfully .                     |
| CUL_NOT_OPENED    | The interface was not opened (by the CUT).                  |
| CUL_INVALID_PARAM | Incorrect parameters or parameter combinations transmitted. |
| CUL_DEVICE_ERROR  | Other errors  |

Table 306: Return Value

### 13.5.6.3 CUL\_AscRcv

The `CUL_AscRcv()` function instructs the COM to provide a defined data volume from the FIFO.

As soon as the requested data is available (and the CUL or the scheduling allows it), the COM invokes the `CUCB_AscRcvReady()` function .

If not enough data is contained in FIFO, the `CUL_AscRcv()` function returns immediately.

The instruction to receive data is stored until:

- The instruction was completely processed or
- the `CUL_AscClose()` function is invoked or
- redefined due to a new instruction

- i** Until the instruction is completely processed, the `*pBuf` content may only be changed using the `CUCB_AscRcvReady()` function.

#### Function Prototype:

```
Udword CUL_AscRcv(udword comId, CUCB_ASC_BUFFER *pBuf)
```

```
typedef struct CUCB_AscBuffer {
    bool bAscState;      // for using by CUT/CUCB
    bool bError;         // for using by CUT/CUCB
    uword reserved1;    // 2 unused bytes
    udword mDataIdx;   // byte offset in aData from which the data
                        // are available
    udword mDataMax;   // max. byte offset 1 for data in aData,
                        //
    udword aData[0];    // Start point of the data copy range
}CUCB_ASC_BUFFER;
```

#### Parameter:

The function has the following parameters:

| Parameter | Description  |
|-----------|--|
| comId     | Fieldbus interface (RS485, RS 232)<br>1 = FB1<br>2 = FB2<br>3 = FB3  |
| pBuf      | It defines the requested amount of data and the location, to which the data should be copied, before <code>CUCB_AscReady()</code> is invoked. If sufficient data has already been received in the FIFO, the <code>CUCB_AscRcvReady()</code> function is invoked during <code>CUL_AscRcv()</code> . |

Table 307: Parameter

**Return Value:**

An error code (udword) is returned.

The error codes are defined in the cut.h header file.

| Error Code        | Description   |
|-------------------|---|
| CUL_OKAY          | If the order was successful, otherwise error code.          |
| CUL_NOT_OPENED    | If the interface was not opened by the CUT.                 |
| CUL_INVALID_PARAM | Incorrect parameters or parameter combinations transmitted. |
| CUL_DEVICE_ERROR  | Other errors  |

Table 308: Return Value

**Restrictions:**

- If the memory area (defined by CUCB\_ASC\_BUFFER) is not allocated in the CUT data segment, CUT and CUIT are terminated.
- A maximum of 1024 bytes of data can be requested.
- A maximum of 1 byte of data can be requested.

#### 13.5.6.4 CUCB\_AscRcvReady

If the COM module invokes calls the CUCB\_AscRcvReady( ) function, the requested amount of data is ready in FIFO (data from the serial interface defined in the comId parameter).

The data has been previously requested with the CUL\_AscRcv( ) function.

The CUCB\_AscRcvReady( ) function can be invoked before, after or while invoking the CUL\_AscRcv( ) function. The task context is always that of CUT.

The CUCB\_AscRcvReady( ) function may invoke all CUT library functions.

These actions are also permitted:

- Increasing mDataMax or
- Reconfiguring mDataIdx and mDataMax by the \*.pBuf data assigned to comId (for further reading)

The structure element of CUCB\_ASC\_BUFFER.mDataIdx has the value of CUCB\_ASC\_BUFFER.mDataMax.

##### Function Prototype:

```
void CUCB_AscRcvReady(udword comId)
```

##### Parameter:

The function has the following parameter:

| Parameter | Description   |
|-----------|---|
| comId     | Fieldbus interface (RS485, RS 232)<br>1 = FB1<br>2 = FB2<br>3 = FB3 |

Table 309: Parameter

##### Restrictions:

If the memory area (defined by CUCB\_ASC\_BUFFER) is not located in the CUT data segment, CUIT and CUT are terminated.

### 13.5.6.5 CUL\_AscSend

The `CUCB_AscSend` function sends the data set defined by the parameter `pBuf` via the serial interface `comId`.

The defined data set must be  $\geq 1$  byte and  $\leq 1\text{kByte}$ .

After data have been sent, the `CUCB_AscSendReady( )` function is invoked. If an error occurs,

- it is not be sent and
- the `CUCB_AscSendReady( )` function will not be invoked.

**Function Prototype:**

```
Udword CUL_AscSend( udword comId, CUCB_ASC_BUFFER *pBuf)
```

#### Parameter:

The function has the following parameters:

| Parameter          | Description   |
|--------------------|---|
| <code>comId</code> | Fieldbus interface (RS485, RS 232)<br>1 = FB1<br>2 = FB2<br>3 = FB3 |
| <code>pBuf</code>  | Defines the data amount to be sent                                  |

Table 310: Parameter

#### Return Value:

An error code (udword) is returned.

The error codes are defined in the `cut.h` header file.

| Error Code                     | Description   |
|--------------------------------|---|
| <code>CUL_OKAY</code>          | If data sending was successfully                            |
| <code>CUL_WOULDBLOCK</code>    | If a message previously sent has not been sent yet          |
| <code>CUL_NOT_OPENED</code>    | If the interface was not opened by the CUT                  |
| <code>CUL_INVALID_PARAM</code> | Incorrect parameters or parameter combinations transmitted. |
| <code>CUL_DEVICE_ERROR</code>  | Other errors  |

Table 311: Return Value

#### Restrictions:

If the memory area (defined by `CUCB_ASC_BUFFER`) is not located in the CUT data segment, CUIT and CUT are terminated.

### 13.5.6.6 CUCB\_AscSendReady

If the COM invokes the `CUCB_AscSendReady( )` function, data is completely sent with the `CUCB_AscSend( )` function via the serial interface.

The task context is always that of CUT. The CUCB\_AscSendReady( )function may invoke all CUT library functions.

**Function Prototype:**

```
void CUCB_AscSendReady(udword comId)
```

**Parameter:**

The function has the following parameter:

| Parameter | Description   |
|-----------|---|
| comId     | Fieldbus interface (RS485, RS 232)<br>1 = FB1<br>2 = FB2<br>3 = FB3 |

Table 312: Parameter

### 13.5.7 UDP/TCP Socket IF

A maximum of 8 sockets can simultaneously be used irrespective of the used protocol.

The physical connection runs over the 10/100BaseT Ethernet interfaces of the controller.

#### 13.5.7.1 CUL\_SocketOpenUdpBind

The `CUL_SocketOpenUdpBind()` function creates a socket of UDP type and binds the socket to the selected port.

The binding address is always `INADDR_ANY`, i.e., all messages for UDP/port addressed to the COM module are received. Sockets are always run in non-blocking mode, i.e., this function does not block.

##### **Function Prototype:**

```
dword CUL_SocketOpenUdpBind( uword port, uword *assigned_port_ptr )
```

##### **Parameter:**

The function has the following parameters:

| Parameter         | Description   |
|-------------------|---|
| port              | An available port number, not occupied by the COM $\geq 0$ .<br>If the port parameter = 0, the socket is bound to the first available port. |
| assigned_port_ptr | Address to which the bounded port number should be copied, if the port parameter = 0 or NULL if not.  |

Table 313: Parameter

##### **Return Value:**

An error code (udword) is returned.

The error codes are defined in the cut.h header file.

| Error Code                       | Description   |
|----------------------------------|---|
| Socket number                    | Socket number assigned to UDP if $> 0$<br>Error codes are $< 0$ |
| <code>CUL_ALREADY_BOUND</code>   | Impossible binding to a port for UDP                            |
| <code>CUL_NO_MORE_SOCKETS</code> | No more resources are available for socket                      |
| <code>CUL SOCK_ERROR</code>      | Other socket errors   |

Table 314: Return Value

##### **Restrictions:**

If the CUT does not possess `assigned_port_ptr` then CUT/CUIT are terminated.

### 13.5.7.2 CUL\_SocketOpenUdp

The `CUL_SocketOpenUdp()` function creates a socket of UDP type without binding to a port. Afterwards, messages can only be sent, but not received via the socket.

**Function Prototype:**

```
dword CUL_SocketOpenUdp ( void )
```

**Parameter:**

None

**Return Value:**

An error code (udword) is returned.

The error codes are defined in the `cut.h` header file.

| Error Code                       | Description   |
|----------------------------------|---|
| Socket number                    | Socket number assigned to UDP if > 0<br>Error codes are < 0 |
| <code>CUL_NO_MORE_SOCKETS</code> | No more resources are available for socket                  |
| <code>CUL SOCK_ERROR</code>      | Other socket errors   |

Table 315: Return Value

### 13.5.7.3 CUL\_NetMessageAlloc

The `CULMessageAlloc()` function allocates message memory for using

- `CUL_SocketSendTo()` with UDP and
- `CUL_SocketSend()` with TCP

A maximum of 10 messages can be simultaneously used in CUT.

#### **Function Prototype:**

```
void *CUL_NetMessageAlloc(udword size, ubyte proto)
```

#### **Parameter:**

The function has the following parameters:

| Parameter | Description                     |                                   |
|-----------|---------------------------------|-----------------------------------|
| size      | Memory requirements in bytes    |                                   |
|           | HIMatrix L2/ HIMax prior to V.4 | HIMatrix L3/ HIMax V.4 and beyond |
|           | ≥ 1 byte and ≤ 1400 bytes       | ≥ 1 byte and ≤ 1472 bytes         |
| proto     | 0 = TCP<br>1 = UDP              |                                   |

Table 316: Parameter

#### **Return Value:**

Buffer address to which the data to be sent must be copied. Memory ranges must never be written outside the allocated area. There are no ranges for the used transmission protocols (EtherNet/IP/UDP or TCP).

#### **Restrictions:**

If no more memory resources are available or if the parameter size is too big or proto > 1, CUT and CUIT are terminated.

### 13.5.7.4 CUL\_SocketSendTo

The CUL\_SocketSendTo( ) function sends the message previously allocated and filled with the CUL\_NetMessageAlloc( ) function as UDP package to the destIp/destPort target address.

After the message has been sent, the pMsg message memory is released automatically.

Whenever messages are sent, firstly the message memory must be allocated with the CULMessageAlloc( ) function.

#### **Function Prototype:**

```
dword CUL_SocketSendTo(     dword socket,
                           void *pMsg,
                           udword size,
                           udword destIp,
                           uwrd destPort )
```

#### **Parameter:**

The function has the following parameters:

| Parameter | Description   |
|-----------|---|
| Socket    | Socket created with CUL_SocketOpenUdp()                             |
| pMsg      | UDP user data memory previously reserved with CUL_NetMessageAlloc() |
| Size      | Memory size in bytes, it must be $\leq$ than the allocated memory   |
| destIp    | Target address != 0, also 0xffffffff is allowed as broadcast        |
| destPort  | Target port != 0  |

Table 317: Parameter

#### **Return Value:**

An error code (udword) is returned.

The error codes are defined in the cut.h header file.

| Error Code     | Description                                 |
|----------------|---|
| CUL_OKAY       | Message was sent successfully               |
| CUL_NO_ROUTE   | No routing available to obtain destIp       |
| CUL_WRONG SOCK | Invalid socket type or socket not available |
| CUL_SOCK_ERROR | Other socket errors                         |

Table 318: Return Value

#### **Restrictions:**

If the CUT does not possess the pMsg message or if the size of pMsg is too high, CUT and CUIT are terminated.

### 13.5.7.5 CUCB\_SocketUdpRcv

The COM invokes the CUCB\_SocketUdpRcv() function if data from the socket is available. In callback, data must be copied from \*pMsg to CUT data, if required. After the function return, no access to \*pMsg is allowed.

#### Function Prototype:

```
void CUCB_SocketUdpRcv( dword socket,
                         void *pMsg,
                         udword packetLength,
                         udword dataLength)
```

#### Parameter:

The function has the following parameters:

| Parameter    | Description  |
|--------------|--|
| socket       | Socket created with CUL_SocketOpenUdp()  |
| pMsg         | pMsg points to the UDP package begin including the Ethernet header.<br>The transmitter of the message can be identified via the Ethernet header. |
| packetLength | The length of the package is stored in packetLength, included the length of the header.  |
| dataLength   | The length of the UDP user data part is stored in dataLength.  |

Table 319: Parameter

### 13.5.7.6 CUL\_NetMessageFree

The `CUL_NetMessageFree()` function releases the message previously allocated with `CUL_NetMessageAlloc()`.

This function is usually not required since invoking the `CUL_SocketSendTo()` function results in an automatic release.

#### Function Prototype:

```
void CUL_NetMessageFree(void *pMsg)
```

#### Parameter:

The function has the following parameter:

| Parameter | Description  |
|-----------|--|
| pMsg      | TCP user data memory previously reserved with <code>CUL_NetMessageAlloc()</code> |

Table 320: Parameter

#### Restrictions:

If the CUT does not possess the pMsg message, CUT and CUIT are terminated.

### 13.5.7.7 CUL\_SocketOpenTcpServer\_TCP

The `CUL_SocketOpenServer()` function creates a socket of type TCP and binds the socket to the selected port.

The address for binding is always `INADDR_ANY`. Additionally, the COM is requested to perform a *listen* on the stream socket. Sockets are always running in non-blocking mode, i.e., this function does not block.

For further information on how to use the socket, refer to `CUCB_SocketTryAccept()` and `CUL_SocketAccept()`.

#### **Function Prototype:**

`dword CUL_SocketOpenTcpServer(uword port, udword backlog)`

#### **Parameter:**

The function has the following parameters:

| Parameter | Description  |
|-----------|--|
| port      | Port number not occupied by the COM > 0  |
| backlog   | Max. number of waiting connections for socket .<br>If the value is 0, the default value 10 is used. The maximum upper limit for this parameter is 50. Higher values are limited to 50. |

Table 321: Parameter

#### **Return Value:**

An error code (udword) is returned.

The error codes are defined in the `cut.h` header file.

| Error Code                       | Description   |
|----------------------------------|---|
| Socket number                    | Socket number already assigned to TCP if > 0<br>Error codes are < 0 |
| <code>CUL_ALREADY_BOUND</code>   | Binding to a port/proto not possible                                |
| <code>CUL_NO_MORE_SOCKETS</code> | No more resources are available for socket                          |
| <code>CUL SOCK_ERROR</code>      | Other socket errors   |

Table 322: Return Value

#### **Restrictions:**

If successfully one socket is used.

### 13.5.7.8 CUCB\_SocketTryAccept

The COM invokes the `CUCB_SocketTryAccept()` function if a TCP connection request is present.

This request can be used to create a socket with the `CUL_SocketAccept()` function.

#### Function Prototype:

```
void CUCB_SocketTryAccept(dword serverSocket)
```

#### Parameter:

The function has the following parameter:

| Parameter    | Description   |
|--------------|---|
| serverSocket | Socket previously created by <code>CUL_SocketOpenTcpServer()</code> . |

Table 323: Parameter

### 13.5.7.9 CUL\_SocketAccept

The `CUL_SocketAccept()` function creates a new socket for the connection request previously signalized with `CUCB_SocketTryAccept()`.

#### Function Prototype:

```
dword CUL_SocketAccept( dword serverSocket,
                        udword *pIpAddr,
                        uword *pTcpPort)
```

#### Parameter:

The function has the following parameters:

| Parameter    | Description   |
|--------------|---|
| serverSocket | serverSocket signalized with <code>CUCB_SocketTryAccept()</code>              |
| pIpAddr      | Address to which the IP address of the peer should be copied or 0 if not      |
| pTcpPort     | Address to which the TCP port number of the peer should be copied or 0 if not |

Table 324: Parameter

#### Return Value:

An error code (udword) is returned.

The error codes are defined in the cut.h header file.

| Error Code          | Description  |
|---------------------|--|
| Socket              | if > 0, then new created socket . If < 0, then error code. |
| CUL_WRONG SOCK      | Invalid socket type or socket not available                |
| CUL_NO_MORE_SOCKETS | No more socket resources are available                     |
| CUL_SOCK_ERROR      | Other socket error   |

Table 325: Return Value

#### Restrictions:

If the CUT does not possess the messages pIpAddr and pTcpPort, CUT and CUIT are terminated.

### 13.5.7.10 CUL\_SocketOpenTcpClient

The `CUL_SocketOpenTcpClient()` function creates a socket of type TCP with free local port and orders a connection to `destIp` and `destPort`. Sockets are always run in non-blocking mode, i.e., this function does not block. The `CUCB_SocketConnected()` function is invoked as soon as the connection has been established.

#### Function Prototype:

```
dword CUL_SocketOpenTcpClient(udword destIp, uword destPort)
```

#### Parameter:

The function has the following parameters:

| Parameter             | Description                               |
|-----------------------|---|
| <code>destIp</code>   | IP address of the communication partner   |
| <code>destPort</code> | Port number of the communication partners |

Table 326: Parameter

#### Return Value:

An error code (udword) is returned.

The error codes are defined in the `cut.h` header file.

| Error Code                       | Description                                       |
|----------------------------------|---|
| Socket number                    | if > 0; error codes are < 0                       |
| <code>CUL_NO_MORE_SOCKETS</code> | No more resources are available for socket        |
| <code>CUL_NO_ROUTE</code>        | No routing available to reach <code>destIp</code> |
| <code>CUL SOCK_ERROR</code>      | Other socket error                                |

Table 327: Return Value

### 13.5.7.11 CUCB\_SocketConnected

The CUCB\_SocketConnected( ) function is invoked by the COM module if a TCP connection was established with the CUL\_SocketOpenTcpClient( ) function.

#### Function Prototype:

```
void CUCB_SocketConnected(dword socket, bool successfully )
```

#### Parameter:

The function has the following parameters:

| Parameter    | Description  |
|--------------|--|
| socket       | Socket previously created and instructed by CUL_SocketOpenTcpClient( ) |
| successfully | TRUE if the connection attempt was successfully, otherwise FALSE       |

Table 328: Parameter

### 13.5.7.12 CUL\_SocketSend

The `CUL_SocketSend()` function sends the message allocated and filled with the `CUL_NetMessageAlloc()` function as TCP package.

After the message has been sent, the pMsg message memory is released automatically.

Whenever messages are sent, firstly the message memory must be allocated with the `CULMessageAlloc()` function.

#### Function Prototype:

```
dword CUL_SocketSend(    dword socket,
                        void *pMsg,
                        udword size)
```

#### Parameter:

The function has the following parameters:

| Parameter | Description  |
|-----------|--|
| socket    | Socket previously created by <code>CUL_SocketOpenTcpClient()</code>              |
| pMsg      | TCP user data memory previously reserved with <code>CUL_NetMessageAlloc()</code> |
| size      | Memory size in bytes, it must be ≤ than the allocated memory                     |

Table 329: Parameter

#### Return Value:

An error code (udword) is returned.

The error codes are defined in the cut.h header file.

| Error Code                   | Description   |
|------------------------------|---|
| <code>CUL_OKAY</code>        | Message was sent successfully                                 |
| <code>CUL_WRONG SOCK</code>  | Invalid socket type or socket not available                   |
| <code>CUL_WOULD_BLOCK</code> | Message cannot be sent, otherwise the socket would be blocked |
| <code>CUL SOCK_ERROR</code>  | Other socket error  |

Table 330: Return Value

#### Restrictions:

If the CUT does not possess the pMsg message or if the size of pMsg is too high, CUT and CUIT are terminated.

### 13.5.7.13 CUCB\_SocketTcpRcv

The CUCB\_SocketTcpRcv( ) function is invoked by the COM module if socket user data are placed.

After quitting the function CUCB\_SocketTcpRcv( ), \*pMsg must no longer be accessed.

If user data are also required outside the CUCB\_SocketTcpRcv( ) function, it must be copied from \*pMsg in an area created ad hoc.



- If the TCP connection is closed asynchronously (after an error or due to a request from the other side), the CUCB\_SocketTcpRcv( ) function with dataLength = 0 is selected.  
The call signalized to CUT that the socket must be closed to re-synchronize communication.

#### Function Prototype:

```
void CUCB_SocketTcpRcv( dword socket,  
                        void *pMsg,  
                        udword dataLength)
```

#### Parameter:

The function has the following parameters:

| Parameter  | Description   |
|------------|---|
| socket     | Socket used to receive the user data.   |
| pMsg       | The pMsg parameter points to the user data begin <b>without</b> Ethernet /IP /TCP header. |
| dataLength | Length of the user data in bytes  |

Table 331: Parameter

### 13.5.7.14 CUL\_SocketClose

The `CUL_SocketClose()` function closes a socket previously created.

The socket is closed within 90 seconds. Only if the socket has been closed, the `SocketOpen` function can be executed once again.

#### Function Prototype:

```
dword CUL_SocketClose(dword socket)
```

#### Parameter:

The function has the following parameter:

| Parameter | Description               |
|-----------|---------------------------|
| socket    | Socket previously created |

Table 332: Parameter

#### Return Value:

An error code (udword) is returned.

The error codes are defined in the `cut.h` header file.

| Error Code                  | Description                                       |
|-----------------------------|---|
| <code>CUL_OKAY</code>       | Socket closed and one socket resource free again. |
| <code>CUL_WRONG SOCK</code> | Socket not available                              |

Table 333: Return Value

### 13.5.8 Timer-IF

#### 13.5.8.1 CUL\_GetTimeStampMS

The `CUL_GetTimeStampMS()` function provides a millisecond tick. This tick is suitable for implementing an own timer in CUT/CUIT. The counter is derived from the quartz of the COM processor and has therefore the same precision.

**Function Prototype:**

```
udword CUL_GetTimeStampMS(void)
```

#### 13.5.8.2 CUL\_GetDateAndTime

The `CUL_GetDateAndTime()` function provides the seconds since the 1st January 1970, 00:00 to the `*pSec` memory and the milliseconds to the `*pMsec` memory. The values are compared with the safe CPU and, depending on the configuration, they can be externally synchronized via SNTP (see Chapter 9).

The values for `CUL_GetDateAndTime()` should **not** be used for time measurement, timer or something else because the values can be provided by the synchronization and/or by the user during operation.

**Function Prototype:**

```
void CUL_GetDateAndTime(udword *pSec, udword *pMsec)
```

**Restrictions:**

If the memory of `pSec` or `pMsec` are not allocated in the CUT data segment, CUT and CUIT are terminated.

### 13.5.9 Diagnosis

The CUL\_DiagEntry( ) function records an event in the COM short time diagnosis that can be read out using the PADT.

#### Function Prototype:

```
void CUL_DiagEntry( udword severity,  
                     udword code,  
                     udword param1,  
                     udword param2)
```

#### Parameter:

The function has the following parameters:

| Parameter         | Description   |
|-------------------|---|
| severity          | It is used to classify events<br>0x45 ('E') == error,<br>0x57 ('W') == warning,<br>0x49 ('I') == information  |
| code              | The user defines the parameter code with an arbitrary number for the corresponding events. If the event occurs, the number is displayed in the diagnosis. |
| param1,<br>param2 | Additional information on the event   |

Table 334: Parameter

## 13.6 Functions for HIMatrix L3 and HM31

The following functions only apply for the HIMatrix L3 and HM31 controllers.

### 13.6.1 ComUserIRQTask

The ComUserIRQTask (CUT) shares the code and data memory with the CUT. It has its own stack and is configured as well as the CUT stack by the COM OS (considered by the CUIT dynamically). The stack has a size of 32kByte. There is only one CUIT in the COM. Initially the IRQServices are disabled, e.g. after power on or after loading the configuration.



The CUIT stack size must not exceed 32 kByte!

If the maximum CUIT stack allowed is exceeded, data is written to the COM memory and can lead to malfunctions!

#### Restrictions:

From the CUIT only the CUL functions for the semaphore handling are invoked.

#### 13.6.1.1 CUCB\_IrqService

The `CUCB_IrqService()` function is invoked by the COM after activation of one of the two possible CAN IRQs.

This function is responsible for the operation of the `devNo` IRQ source of the corresponding CAN chip and must ensure that the CAN chip cancels its IRQ request.

#### Function Prototype:

```
void CUCB_IrqService(udword devNo)
```

#### Restrictions:

The IRQ management of the COM processor is performed by the COM OS and must not be performed by the `CUCB_IrqService()` function.



The `CUCB_IrqService()` function must be implemented very efficiently to minimize unneeded latencies of other COM processor functions.

Otherwise, functions could no longer be performed at high COM processor load which could interfere with the safe CPU's safe communication.

The CUT library allows the COM IRQ channel to which the CAN controller is connected, to be unlocked and switched off.

#### CUL\_IrqServiceEnable

The `CUL_IrqServiceEnable()` function enables the COM IRQ channel for the selected `devNo` CAN controller. From now on, CAN IRQs trigger the call of the CUT IRQ task.

#### Function Prototype:

```
void CUL_IrqServiceEnable(udword devNo)
```

#### Parameter:

The function has the following parameter:

| Parameter | Description                                  |
|-----------|--|
| devNo     | 1 = CAN controller A<br>2 = CAN controller B |

Table 335: Parameter

**Restrictions:**

If devNo values other than 1 or 2 are used or a field bus interface not set with CAN is employed, CUIT/CUT are terminated.

### 13.6.1.2 CUL\_IrqServiceDisable

The `CUL_IrqServiceDisable()` function locks the COM IRQ channel for the *devNo* CAN controller. From now on, the CAN IRQs no longer trigger the call of the CUT IRQ task. However, incompletely handled IRQs are processed.

**Function Prototype:**

```
void CUL_IrqServiceDisable(udword devNo)
```

**Parameter:**

The function has the following parameter:

| Parameter | Description                                  |
|-----------|--|
| devNo     | 1 = CAN controller A<br>2 = CAN controller B |

Table 336: Parameter

**Restrictions:**

If devNo values other than 1 or 2 are used or a field bus interface not set with CAN is employed, CUIT/CUT are terminated.

### 13.6.1.3 CUL\_DeviceBaseAddr

The `CUL_DeviceBaseAddr()` function provides the 32-bit basic address of the CAN controllers.

**Function Prototype:**

```
void* CUL_DeviceBaseAddr(udword devNo)
```

**Parameter:**

The function has the following parameter:

| Parameter | Description                                  |
|-----------|--|
| devNo     | 1 = CAN controller A<br>2 = CAN controller B |

Table 337: Parameter

**Restrictions:**

If devNo values other than 1 or 2 are used or a field bus interface not set with CAN is employed, CUIT/CUT are terminated.

### 13.6.2 NVRam IF

The CUT and the CUIT can read and write the available range.

The size of the available NVRam depends on the controller in use.

| Element    | HIMax V.4 and beyond                | HIMatrix L3                          | HIMatrix L2   |
|------------|-------------------------------------|--------------------------------------|---------------|
| NVRam size | 24576 bytes<br>(64 kByte -40 kByte) | 483328 bytes<br>(512 kByte-40 kByte) | Not available |

Table 338: Memory Area for Code and Data



The COM **cannot** ensure data consistency if the operation voltage fails while accessing it and if two tasks are accessing it simultaneously.

No difference is made between memory areas that are written often and those that are written seldom.

#### 13.6.2.1 CUL\_NVRamWrite

The `CUL_NVRamWrite()` function writes data to the NVRam.

##### Function Prototype:

```
void CUL_NVRamWrite(udword offset, void *source, udword size)
```

##### Parameter:

The function has the following parameters:

| Parameter | Description  |
|-----------|--|
| offset    | Values valid in the NVRam range, see Chapter 13.6.2                  |
| source    | Memory area in CUT data segment which should be copied to the NVRam. |
| size      | Number of bytes that should be copied                                |

Table 339: Parameter

##### Restrictions:

With invalid parameters, the CUT and the CUIT are terminated, for example:

- $\text{offset} \geq \text{size of the available NVRam}$
- $\text{offset+size} \geq \text{size of the available NVRam}$
- Source out of the CUT data segment
- Source+size out of the CUT data segment

#### 13.6.2.2 CUL\_NVRamRead

The function `CUL_NVRamRead()` reads data from the NVRam.

**Function Prototype:**

```
void CUL_NVRamRead(udword offset, void *destination, udword size)
```

**Parameter:**

The function has the following parameters:

| Parameter   | Description  |
|-------------|--|
| offset      | Values valid in the NVRam range, see Chapter 13.6.2                  |
| destination | Memory area in CUT data segment which should be copied to the NVRam. |
| size        | Number of bytes that should be copied                                |

Table 340: Parameter

**Restrictions:**

With invalid parameters, the CUT and the CUIT are terminated, for example:

- $\text{offset} \geq \text{size of the available NVRam}$
- $\text{offset} + \text{size} \geq \text{size of the available NVRam}$
- Destination out of the CUT data segment
- Destination+size out of the CUT data segment

### 13.6.3 Semaphore IF

The CUT and the CUIT have **one** common semaphore for process synchronization.

A semaphore must be used to protect the data of CUCB and CUL functions that is shared with the CUCB\_IrqService( )function. This guarantees that data shared with the CUCB\_IrqService( )function is consistent.

**Restriction:**

If the controller has processing the semaphore function, is stopped and rebooted, the COM could be rebooted.

#### 13.6.3.1 CUL\_SemaRequest()

The CUL\_SemaRequest( ) function sends a request to the CUT/CUIT semaphore.

If the semaphore

- is available, the function returns with the pContext value.
- is not available, the invoking task is blocked until the semaphore is released by another task and is returns with the pContext value.

The context, which is referenced by the pContext parameter, is only used by the CUL functions for the invoking task and must not change between request and release.

**Function Prototype:**

```
void CUL_SemaRequest(udword *pContext)
```

**Parameter:**

The function has the following parameter:

| Parameter | Description   |
|-----------|---|
| pContext  | Only used by the CUL function within the invoking task.<br>The context is returned via pContext and must be specified in the CUL_SemaRelease( ) function. |

Table 341: Parameter

**Restrictions:**

If the number of admissible recursions is exceeded then the CUT/CUIT are terminated.

If the CUT is blocked by a semaphore no more CUCB\_’s are executed with the exception of the function CUCB\_IrqService( ).

### 13.6.3.2 CUL\_SemaRelease

The function CUL\_SemaRelease( ) releases again the semaphore defined by \*pContext.

**Function Prototype:**

```
void CUL_SemaRelease(udword *pContext)
```

**Parameter:**

The function has the following parameter:

| Parameter | Description   |
|-----------|---|
| pContext  | With the same value for *pContext as written by CUL_SemaRequest or CUL_SemaTry. |

Table 342: Parameter

**Restrictions:**

If Release is invoked more times than Request/Try, then the CUT/CUIL are terminated.

### 13.6.3.3 CUL\_SemaTry

The function `CUL_SemaTry()` tries to request the semaphore of the CUT/CUIT.

If the semaphore

- is free the function returns with the value TRUE and reserves the semaphore.
- is not free the function returns with the value FALSE and does not reserve the semaphore.

The udword referenced by `pContext` is only used by CUL for the invoking task and must not be changed between request and release.

#### **Function Prototype:**

```
bool CUL_SemaTry(udword *pContext)
```

#### **Parameter:**

The function has the following parameter:

| Parameter             | Description   |
|-----------------------|---|
| <code>pContext</code> | Only used by the CUL function within the invoking task. |

Table 343: Parameter

#### **Return Value:**

An error code (udword) is returned.

The error codes are defined in the `cut.h` header file.

| Return Value | Description                     |
|--------------|---------------------------------|
| TRUE         | Semaphore could be reserved     |
| FALSE        | Semaphore could not be reserved |

Table 344: Return Value

The context is returned via `pContext` and must be specified in the `CUL_SemaRelease()` function.

#### **Restrictions:**

If the number of admissible recursions is exceeded then the CUT/CUIT are terminated.

If the CUT is blocked by a semaphore no more CUCB\_’s are executed with the exception of the function `CUCB_IrqService()`.



The functions `CUL_SemaTry()` and `CUL_SemaRequest()` may also be invoked without blockade if the invoking task has already reserved the semaphore; in such a case, however, the same number of `CUL_SemaRelease` must occur until the semaphore is available again. The recursion allows a minimum of 32000 steps. If more steps are allowed, depends on the corresponding COM version.

## 13.7 Installing the Development Environment

This chapter describes how to install the development environment and create a ComUserTask.

The latest development environment is included in the current HIMA DVD. For HIMatrix L3 valid from HIMA DVD (Edition 2012), see Chapter 12.2.1.

### 13.7.1 Installing the Cygwin Environment

The Cygwin environment is required since the GNU C compiler tools only runs under the Cygwin environment.

Cygwin environment must be installed under Windows 2000/XP/Vista/Win7.

- 
- i** For further details on the installation requirements, refer to Chapter 13.2. Deactivate the **virus scanner** on the PC on which Cygwin should be installed to avoid problems when installing Cygwin.
- 

Perform the following steps to install the Cygwin environment:

#### Start the setup program for installing Cygwin:

1. Copy the Cygwin installation archive `cygwin_1.7.5-1` from the installation CD to the local hard disk (e.g., drive `C:\`).
2. Open the Cygwin index in Windows Explorer  
`C:\ cygwin_1.7.5-1`.
3. Double-click the **setup-2.697.exe** file to start the Cygwin installation.
4. Click the **Next** button to start the setup.



Figure 93: *Cygwin Setup* Dialog Box

**The Disable Virus Scanner dialog box appears if the virus scanner was not deactivated.**

Follow these steps to deactivate the virus scanner during the installation of Cygwin.



Deactivate the virus scanner before installing Cygwin since, depending on the virus scanner in use, the warning dialog box could not appear although the virus scanner is running.

---

1. Select **Disable Virus Scanner** to prevent potential problems during the installation due to the virus scanner.
2. Click the **Next** button to confirm.

**Select the installation source in the *Choose Installation Type* dialog box:**

1. Select **Install from Local Directory** as installation source.
2. Click the **Next** button to confirm.

**In the *Choose Installation Directory* dialog box, select the installation target directory for Cygwin:**

1. Enter the directory in which Cygwin should be installed.
2. Accept all the defaults of the dialog box.
3. Click the **Next** button to confirm.

**In the *Select Local Package Directory* dialog box, select the Cygwin installation archive.**

1. Enter the name for the Cygwin installation archive in the field **Local Package Directory**.  
Select the archive with the installation files.
2. Click the **Next** button to confirm.

**In the *Select Packages* dialog box, select all installation packages:**

1. Select the **Curr** radio button.
2. In the view box, slowly click the installation option next to **All** until **Install** is displayed for a complete installation of all packages (approx. 1.86 GB memory requirements).



Make sure that **Install** is placed behind each package.  
If the packages are not completely installed, important functions might be missing for compiling the CUT C code!

---

3. Click the **Next** button to confirm.

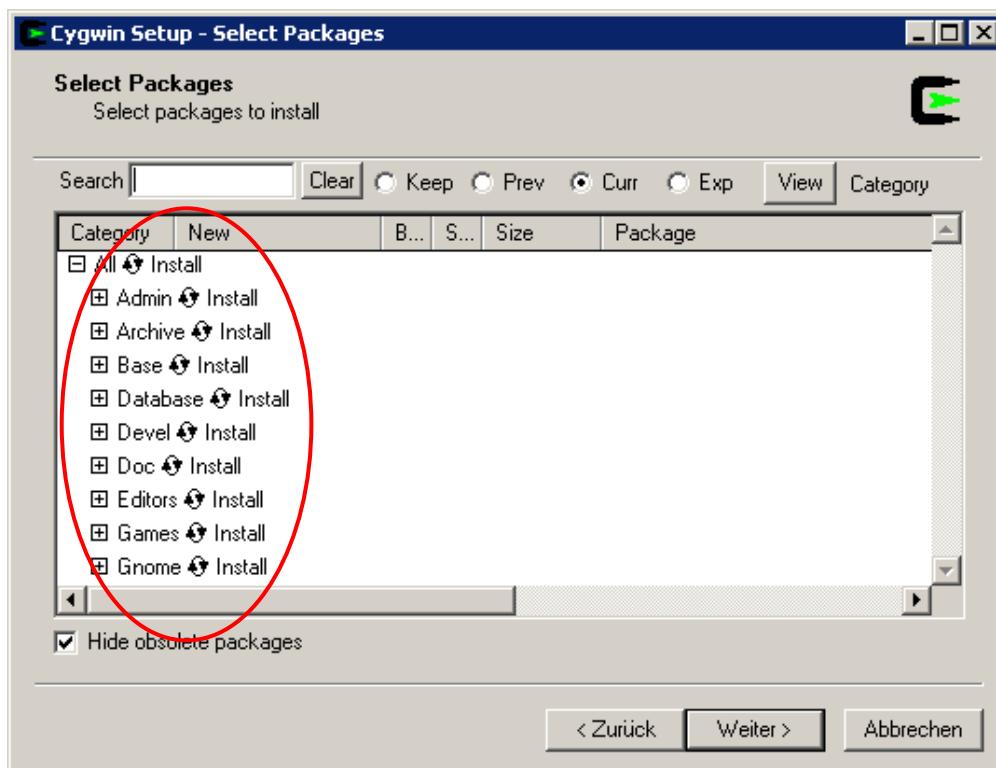


Figure 94: *Select Packages* Cygwin Setup Dialog Box

**Perform the following steps to complete the Cygwin installation:**

1. Select Entry in the **Start Menu**.
2. Select **Desktop Icon**.
3. Click the **Finish** button to complete the Cygwin installation.

| Cygwin Commands            | Description                      |
|----------------------------|----------------------------------|
| <b>cd (directory name)</b> | Change directory                 |
| <b>cd ..</b>               | Move to parent directory         |
| <b>ls -l</b>               | Display all files of a directory |
| <b>help</b>                | Overview of bash shell commands  |

Table 345: Commands in Cygwin (Bash Shell)

### 13.7.2 Installing the GNU Compiler

**Perform the following steps to install the GNU Compiler:**

1. In Windows Explorer, open the directory of the installation CD.
2. Double-click `gcc-ppc-v3.3.2_binutils-v2.15.zip`.
3. Extract all files in the Cygwin directory (e.g., `C:\cygwin\...`).  
The GNU compiler is unpacked in the **gcc-ppc** subdirectory.
4. Set the environment variables in the system control:

- Use the Windows start menu **Settings->System Control->System** to open the system properties.
- Select the **Advanced** tab.
- Click the **Environment Variables** button.
- Select the **Path** system variable in the *System Variables* box and extend the system variable with `C:\cygwin\gcc-ppc\bin`.

Copy the `cut_src` directory from the installation CD to the home directory.

The `cut_src` directory contains all the 'include' and 'lib' directories required for creating a ComUserTask.

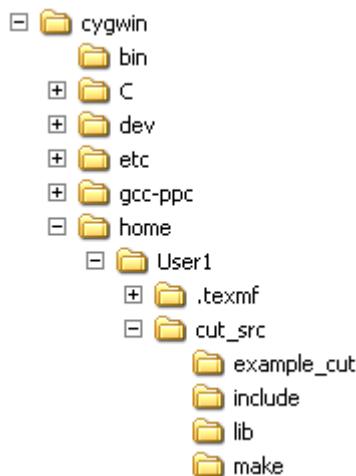


Figure 95: Cygwin Structure Tree

- 
- i** If the home directory was not created automatically, create it with Windows Explorer (e.g., `C:\cygwin\home\User1`).

To create another home directory for cygwin bash shell, add the `set Home` command to the `cygwin.bat` batch file.

---

```

@echo off
C:
chdir C:\cygwin\bin
set Home=C:\User1
bash --login -i
  
```

Figure 96: *Cygwin.bat* Batch File

- 
- i** Refer to Chapter 13.8.2.4 for further information on how to generate executable code for the `example_cut` program provided on the DVD.
-

## 13.8 Creating New CUT Projects

This chapter shows how to create a new CUT project and specifies which files must be adapted.



The **example cut** CUT project located on the installation CD is fully adapted.

For creating new CUT projects, HIMA recommends to create a new subdirectory of ... \cut\_src\ for each CUT project.

**Example:**

As a test, create the *example cut* directory, name the C source **example\_cut.c**, the ldb file created in the make directory is named **example\_cut ldb**.

**Create the directory *example\_cut* for the new ComUserTask.**

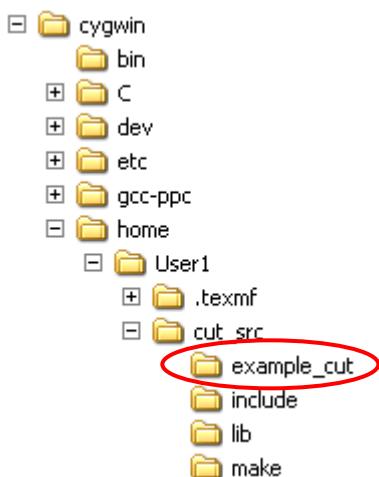


Figure 97: Cygwin Structure Tree

1. Copy the files
  - Example\_cut.c
  - Example\_cut.mke
  - makefile
 in the directory *example\_cut*.

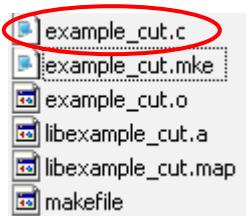


Figure 98: C Code File in the *example\_cut* Folder

As described in the following chapters, perform all changes to .mke file and makefile for every new project.

### 13.8.1 CUT Makefiles

Configuration of CUT makefiles for different source files and ldb files.

As described in the following chapters, a total of three makefiles must be adapted.

#### 13.8.1.1 Makefile with ".mke" Extension

The .mke file is located in the corresponding source code directory, e.g., *cut\_src\example\_cut\example\_cut.mke*.

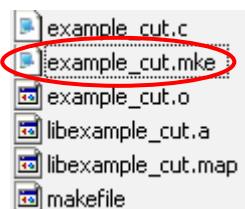


Figure 99: .mke File in the example\_cut Folder

#### Change the mke file as described below:

1. The module variable must have the same loadable name as the corresponding .mke file (e.g., **example\_cut**).
2. One or multiple C files required for creating the target code (loadable file) are assigned to the **c\_sources** variable .

```
#####
#  
# make file (DOS/NT)  
# $Id: example_cut.mke 58869 2005-10-11 12:35:46Z es_fp $  
#  
# assign name of module here (e.g. nl for NetworkLayer)  
module= example_cut  
#  
# assign module sources here  
sources=  
  
c_sources= $(module).c  
asm_sources=
```

Figure 100: .mke File Starting with Line 1

### Makefile

The makefile file is located in the corresponding source code directory, e.g., cut\_src\example\_cut\makefile.

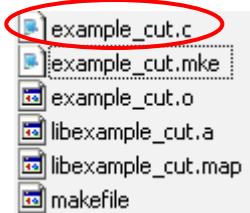


Figure 101: makefile in the *example\_cut* Folder

#### Change the makefile file as described below:

1. Drag the include line for the .mke file upwards and enter the current name for the the .mke file.
2. Expand the make call with the two variables **SUBMOD\_DIRS** and **CUT\_NAME**.

all\_objects:

**include example\_cut.mke**

cut:

**\$(MAKE) -C ..\make elf SUBMOD\_DIRS=cut\_src\\$(module) CUT\_NAME=\$(module)**

all\_objects: \$(c\_objects) \$(asm\_objects) \$(objects)

Figure 102: makefile Starting with Line 34

#### 13.8.1.2 Makefile with the 'makeinc.inc.app' Extension

The only change made to this and all the following CUT projects is that the name of the CUT loadable is made changeable via a make variable.

The makeinc.inc.app file is located in the cut\_src directory  
e.g., cut\_src\makeinc.inc.app.

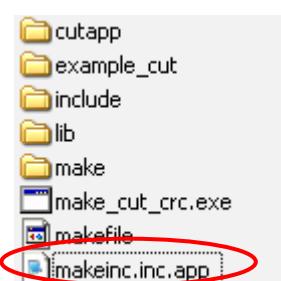


Figure 103: makeinc.inc.app file in the *example\_cut* Folder

**Change the makeinc.inc.app file as described below:**

1. Expand the file with the **CUT\_NAME** variable.

```

all: lib$(module).$(LIBEXT)
@echo 'did make for module ['lib$(module).$(LIBEXT)']'

lib$(module).$(LIBEXT): $(objects) $(c_objects) $(asm_objects) $(libraries)

SUBMOD2_LIBS=$(foreach lib,$(SUBMOD_LIBS),../$(lib))

CUT_NAME=cut

makeAllLibs:
$(MAKE) -C ../../cut_src cut_src

makeLoadable:
@echo; \
BGTYPE=" $(CUT_NAME)"; \
if [ ! -f $$BGTYPE.map ] ; then \
echo "Error: MAP-Datei $$BGTYPE.map existiert nicht"; \
exit 1; \
fi; \
OS_LENGTH=$$(gawk '/__OS_LENGTH/ {print substr($$1,3,8)}' $$BGTYPE.map); \
echo; \
$(OBJCOPY) --strip-all --strip-debug -O binary $$BGTYPE.elf $$BGTYPE.bin; \
echo; \
echo "Building C3-Loadable-Binary ..."; \
$(MCRC) $$BGTYPE.bin 0 $$OS_LENGTH $$OS_LENGTH $$BGTYPE.lbd; \
echo; \
$(CUT_NAME).elf: makeAllLibs $(SUBMOD2_LIBS)

elf:
@echo; test -f section.dld && $(MAKE) $(CUT_NAME).elf && $(MAKE) makeLoadable \
|| { echo "ERROR: Invalid subdir. Please invoke elf target only from make/ subdirectory." &&
echo && false ; } ;

# end of file: makeinc.inc

```

Figure 104: makeinc.inc.app Starting with Line 247

### 13.8.2 Adapting C Source Codes

**Perform the following steps to open the source code file:**

1. Open the project directory *cut\_src\example\_cut* that has been created and configured in the previous steps.
2. Open the C source code file with the extension **.c** with an editor (e.g. notepad).

### 13.8.2.1 Configure Input and Output Variables

**Perform the following steps to configure the input and output variables in the source code file:**

1. The data size of the variables that should be created in the SILworX **Output Variables** tab must be set in the **CUT\_PDI[X]** array of the source code file.
2. The data size of the variables that should be created in the SILworX **Input Variables** tab must be set in the **CUT\_PDO[X]** array of the source code file.

### 13.8.2.2 Start Function in CUT

The `void CUCB_TaskLoop (udword mode)` C function is the start function and is cyclically invoked by the user program.

### 13.8.2.3 Example Code "example\_cut.c"

The following C code copies the value from the **CUT\_PDI[0]** input to the **CUT\_PDO[0]** output and returns the value unchanged to the SILworX user program.



The C code **example\_cut.c** is located on the installation CD.

```
/* Example for the CUT implementation */
#include "include/cut_types.h"
#include "include/cut.h"
#ifndef __cplusplus
extern "C" {
#endif
/*********************************************
/* SILworX Output Records (CPU->COM) */
uword CUT_PDI[1] __attribute__ ((section("CUT_PD_IN_SECT"), aligned(1)));
/* SILworX Input Records (COM->CPU) */
uword CUT_PDO[1] __attribute__ ((section("CUT_PD_OUT_SECT"), aligned(1)));
/********************************************

/* Callback function for starting the CUT */
void CUCB_TaskLoop(udword mode)
{
    if (CUT_PDI[0] > CUT_PDO[0]) /*This is executed only, if the
                                    /*SILworX application program
                                    /*was processed.
                                    /*The SILworX application program*/
                                    /*adds the value 1 to CUT_PDO[0] and
                                    /*writes the result into CUT_PDI[0]
                                    */

        CUT_PDO[0] = CUT_PDI[0]; /*Copies the value from input CUT_PDI[0]
                                /*into output CUT_PDO[0] of the SPS
        if (CUT_PDO[0] == 65535)
            (CUT_PDO[0] = 0);
    }
}
/*********************************************
```

Figure 105:Resource Structure Tree

```
/************************************************************************/
void CUCB_AscRcvReady(udword comId)
{
    CUL_DiagEntry(0x49, 1, comId, 0);
}
/************************************************************************/
void CUCB_AscSendReady(udword comId)
{
    CUL_DiagEntry(0x49, 2, comId, 0);
}
/************************************************************************/
void CUCB_SocketTryAccept(dword serverSocket)
{
    CUL_DiagEntry(0x49, 3, serverSocket, 0);
}
/************************************************************************/
void CUCB_SocketConnected(dword socket, bool Okay)
{
    CUL_DiagEntry(0x49, 4, socket, Okay);
}
/************************************************************************/
void CUCB_SocketTcpRcv(dword socket, void *pMsg, udword dataLength)
{
    CUL_DiagEntry(0x49, 5, socket, dataLength);
}
/************************************************************************/
void CUCB_SocketUdpRcv(dword socket, void *pMsg, udword packetLength,
                        udword dataLength)
{
    CUL_DiagEntry(0x49, 6, socket, dataLength);
}
/************************************************************************/
void CUCB_IrqService(udword devNo)
{
    CUL_DiagEntry(0x49, 7, devNo, 0);
}
/************************************************************************/

#ifndef __cplusplus
/* end extern "C" */
#endif

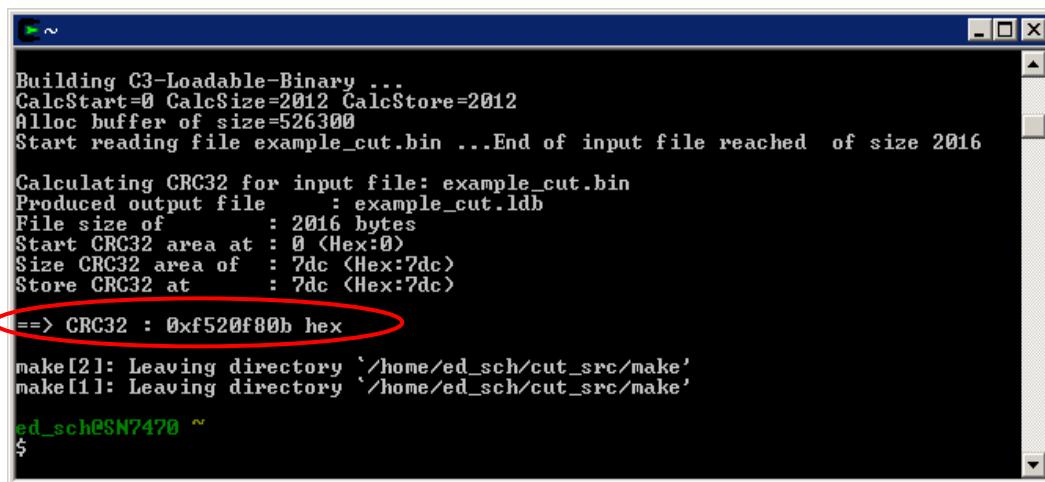
/* end of file */
```

Figure 106: C Code example\_cut.c

### 13.8.2.4 Creating Executable Codes (ldb file)

**Perform the following steps to create the executable code (ldb file):**

1. Start **Cygwin Bash Shell**.
2. Move to the directory .../cut\_src/example\_cut/.
3. Start the code generation by specifying:  
*make cut\_himax* for HIMax  
*make cut\_l2* for HIMatrix L2  
*make cut\_l3* for HIMatrix L3.  
The **cut.ldb** binary file located in the /cut\_src/make/ directory is generated automatically.
4. If CRC32 was generated, also the executable code was generated (see red marking in Figure 107).



```
Building C3-Loadable-Binary ...
CalcStart=0 CalcSize=2012 CalcStore=2012
Alloc buffer of size=526300
Start reading file example_cut.bin ...End of input file reached of size 2016
Calculating CRC32 for input file: example_cut.bin
Produced output file : example_cut.ldb
File size of : 2016 bytes
Start CRC32 area at : 0 <Hex:0>
Size CRC32 area of : 7dc <Hex:7dc>
Store CRC32 at : 7dc <Hex:7dc>
==> CRC32 : 0xf520f80b hex
make[2]: Leaving directory '/home/ed_sch/cut_src/make'
make[1]: Leaving directory '/home/ed_sch/cut_src/make'
ed_sch@SN7470 ~
```

Figure 107: Cygwin Bash Shell

This executable code (ldb file) must be loaded into the ComUserTask, see Chapter 13.8.3.

### 13.8.3 Implementing the ComUserTask in the Project

Perform the following steps in SILworX to integrate the ComUserTask into the project:

#### 13.8.3.1 Creating the ComUserTask

##### To create a new ComUserTask

1. In the structure tree, open **Configuration, Resource, Protocols**.
2. On the context menu for protocols, click **New, ComUserTask** to add a ComUserTask.
3. Right-click the **ComUserTask**, click **Properties** and select the **COM module**.  
Accept the default settings for the first configuration.



Only one ComUserTask per resource may be created.

---

#### 13.8.3.2 Loading Program Code into the Project

##### To load a ComUserTask into the project

1. In the structure tree, open **Configuration, Resource, Protocols**.
2. Right-click **ComUserTask** and select **Load User Task**. Open the directory  
`.../cut_src/make/`
3. Select the **ldb** file that should be processed in the ComUserTask.



Different versions of the ldb file can be integrated by reloading the executable code (ldb file). The correctness of the ldb file's content is not checked when loading. The ldb file is then compiled in the project together with the resource configuration and can be loaded into the controller. If the ldb file is changed, the project must be recompiled and reloaded.

---

### Connecting Variables to CUT

The user can define a not safety-related process communication between the safe CPU and the not safe COM (CUT). Depending on the controller, a given data amount can be exchanged in each direction (see Chapter see Chapter 3.2).

Create the following global variables:

| Variable | Type |
|----------|------|
| COM_CPU  | UINT |
| CPU_COM  | UINT |

### 13.8.3.3 Connecting Process Variables

#### Process variables in the ComUserTask:

1. Right-click **ComUserTask** and select **Edit**.
2. In the **Edit** dialog box, select the **Process Variables** tab.

#### Output Variables (CPU->COM)

The **Output Variables** tab contains the variables that should be transferred from the CPU to the COM module.

| Name    | Type | Offset | Global Variable |
|---------|------|--------|-----------------|
| CPU_COM | UINT | 0      | CPU_COM         |

Table 346: Output Variables (CPU->COM)

1. Drag the global variables to be sent from the Object Panel onto the **Output Variables** tab.
2. Right-click anywhere in the **Output Variables** area to open the context menu.
3. Click **New Offsets** to re-generate the variable offsets.

#### Input Variables (COM->CPU)

The **Input Variables** tab contains the variables that should be transferred from the COM to the CPU module.

| Name    | Type | Offset | Global Variable |
|---------|------|--------|-----------------|
| COM_CPU | UINT | 0      | COM_CPU         |

Table 347: Input Variables(COM->CPU)

1. Drag the global variables to be received from the Object Panel onto the **Input Variables** area.
2. Right-click anywhere in the the **Input Variables** area to open the context menu.
3. Click **New Offsets** to re-generate the variable offsets.

#### To verify the ComUserTask configuration

1. In the structure tree, open **Configuration, Resource, Protocols, ComUserTask**.
2. Right-click **Verification** to verify the CUT configuration.
3. Thoroughly verify the messages displayed in the **logbook** and correct potential errors.

### 13.8.3.4 Creating the SILworX User Program

#### To create the SILworX user program

1. In the structure tree, open **Configuration**, right-click **Resource**, then select **Edit**.
2. Drag the global variables **COM\_CPU** and **CPU\_COM** from the Object Panel to the drawing area.
3. Create the user program as specified in the following figure.

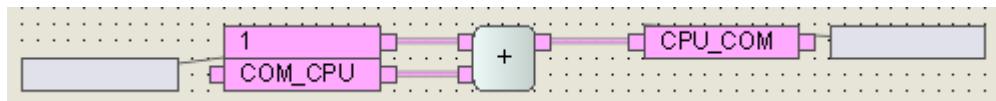


Figure 108:SILworX Program Editor

#### To configure the schedule interval [ms]

1. Right-click **ComUserTask**, then click **Properties**.
2. In the *Schedule Interval [ms]* input box, specify in which intervals the ComUserTask should be invoked.

- 
- i** Use the resource's user program to recompile the ComUserTask configuration and load it into the controller. The new configuration can only be used with the HIMax/HIMatrix system after this step is completed.
- 

#### To check the ComUserTask with the online test

1. In the structure tree, select **Configuration**, **Resource**, **Program**.
2. Right-click **Program** and select **Online**. Log in to the system.



Figure 109:SILworX Online Test

#### Function of the SILworX user program:

The SILworX user program adds the value **1** to the signal **COM\_CPU** (data inputs) and transmits the result to the signal **CPU\_COM** (data outputs).

With the next CUT call (schedule interval [ms]), the signal **CPU\_COM** is transmitted to the CUT function (see example code in Chapter 13.8.2).

The ComUserTask receives the signal **CPU\_COM** and returns the value unchanged with the signal **COM\_CPU**.

### 13.8.4 DCP: Error in loading a configuration with CUT

**Run Time Problems (e.g., ComUserTask in infinite loop):**

**Reason for run time problems:**

Programming a loop, which runs for a long time, in the corresponding CUT source code results in a “deadlock” of the COM processor.

As a consequence, no connection can be established to the controller and the resource configuration can no longer be deleted.

**Solution:** Reset the HIMax communication module or the HIMatrix controller:

- In the online view associated with the Hardware Editor, use the **Maintenance/Service, Module (Restart)** function to reset the communication module (or the reset push-button to reset the HIMatrix system, see the controller's data sheet).
- Create a new CUT (without run time errors, endless loops).
- Load the CUT (lrb file) into the project.
- Generate the code.
- Load the code into the controller.

## 14 General

### 14.1 Configuring the Function Blocks

The fieldbus protocols and the corresponding function blocks operate in the HIMax/HIMatrix controller's COM module. Therefore, the function blocks must be created in SILworX. To do this, open **Configuration, Resource, Protocols...** in the structure tree.

To control the function blocks on the COM module, function blocks can be created in the SILworX user program (see Chapter 14.1.1). These can be used as standard function blocks.

Common variables are used to connect the function blocks in the SILworX user program to the corresponding function blocks in the SILworX structure tree. These must be created beforehand using the Variable Editor.

#### 14.1.1 Purchasing Function Block Libraries

The function block libraries for PROFIBUS DP and TCP Send/Receive must be added to the project using the *Restore...* function (from the context menu for the project).

The function block library is available from HIMA support upon request.

**Phone:** +49 (0)6202-709 -185

-259

-261

**E-mail:** support@hima.com

#### 14.1.2 Configuring the Function Blocks in the User Program

Drag the required function blocks onto the user program. Configure the inputs and outputs as described for the individual function block.

##### Upper Part of the Function Block

The upper part of the function block corresponds to the user interface that the user program uses for controlling it.

The variables used in the user program are connected here. The prefix "A" means Application.

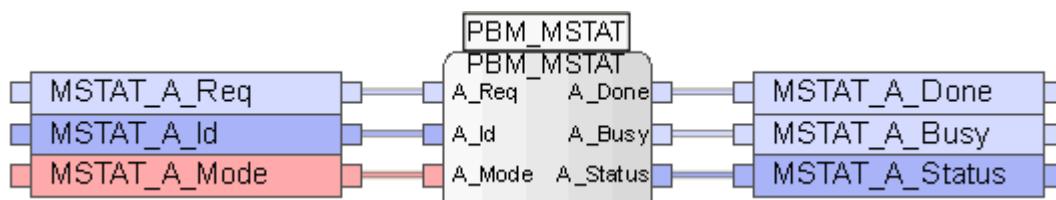


Figure 110: PNM\_MSTST Function Block (Upper Part)

### Lower Part of the Function Block

The function block's lower part represents the connection to the function block (in the SILworX structure tree).

The variables that must be connected to the function block located in SILworX structure tree are connected here.

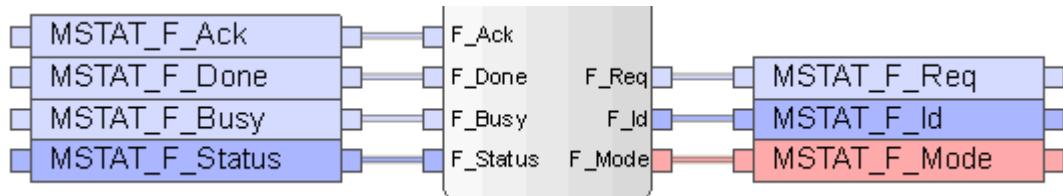


Figure 111: PNM\_MSTST Function Block (Lower Part)

#### 14.1.3 Configuring the Function Blocks in the SILworX Structure Tree

##### To configure the function block in the SILworX structure tree

1. In the structure tree, open **Configuration, Resource, Protocols**, e.g., **PROFIBUS Master**.
2. Right-click **Function Blocks**, and then click **New**.
3. In the SILworX structure tree, click the suitable function block.

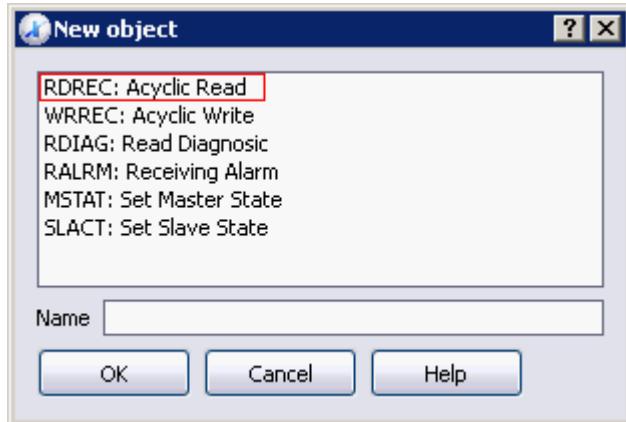


Figure 112: Choosing Function Blocks

The inputs of the function block (checkmark in the Input Variables column) must be connected to the same variables that are connected in the user program to the *F\_Outputs* of the function block.

The outputs of the function block (no checkmark in the Input Variables column) must be connected to the same variables that are connected in the user program to the *F\_Inputs* of the function block.

| System Variables |        |           |                                     |                 |  |
|------------------|--------|-----------|-------------------------------------|-----------------|--|
| F                | Name   | Data type | Input variable                      | Global Variable |  |
| 1                | ACK    | BOOL      | <input checked="" type="checkbox"/> | MSTAT_F_Ack     |  |
| 2                | BUSY   | BOOL      | <input checked="" type="checkbox"/> | MSTAT_F_Busy    |  |
| 3                | DONE   | BOOL      | <input checked="" type="checkbox"/> | MSTAT_F_Done    |  |
| 4                | M_ID   | DWORD     | <input type="checkbox"/>            | MSTAT_F_Id      |  |
| 5                | MODE   | INT       | <input type="checkbox"/>            | MSTAT_F_Mode    |  |
| 6                | REQ    | BOOL      | <input type="checkbox"/>            | MSTAT_F_Req     |  |
| 7                | STATUS | DWORD     | <input checked="" type="checkbox"/> | MSTAT_F_Status  |  |

Figure 113: System Variables of the MSTAT Function Block

## 14.2 Maximum Communication Time Slice

The maximum communication time slice is the time period in milliseconds (ms) and per CPU cycle assigned to the processor module for processing the communication tasks. If not all upcoming communication tasks can be processed within one CPU cycle, the whole communication data is transferred over multiple CPU cycles (number of communication time slices > 1).

### 14.2.1 For HIMax Controllers

#### To determine the maximum communication time slice

1. Use the system under the maximum load:  
All communication protocols are operating (safeethernet and standard protocols).
2. Open the **Control Panel** and select the **Statistics** structure tree element.
3. Open **Cyc.Async** to read the number of communication time slices.
4. Open **Time.Async** to read the duration of one communication time slice.



For calculating the maximum response time, the number of communication time slices must be equal to 1, see Chapter 4.6.

The duration of the communication time slice must be set such that, when using the communication time slice, the CPU cycle cannot exceed the watchdog time specified by the process.

## 14.3 Load Limitation

A computing time budget expressed in % ( $\mu P$ -Budget) can be defined for each communication protocol. It allows one to distribute the available computing time among the configured protocols. The sum of the computing time budgets configured for all communication protocols on a CPU or COM modules may not exceed 100%.

The defined computing time budgets of the individual communication protocols are monitored. If a communication protocol already achieved or exceeded its budget and no reserve computing time is available, the communication protocol cannot be processed.

If sufficient additional computing time is available, it is used to process the communication protocols that already achieved or exceeded their budget. It can therefore happen that a communication protocol uses more budget than it has been allocated to it.

It is possible that more than 100 % computing time budget is displayed online. This is not a fault; the budget exceeding 100 % indicates the additional computing time used.



The additional computing time budget is not a guarantee for a certain communication protocol and can be revoked from the system at any time.

## Appendix

### Glossary

| Term              | Description   |
|-------------------|---|
| ARP               | Address resolution protocol: Network protocol for assigning the network addresses to hardware addresses   |
| AI                | Analog input  |
| Connector board   | Connector board for the HIMax module  |
| COM               | Communication module  |
| CRC               | Cyclic redundancy check   |
| DI                | Digital input   |
| DO                | Digital output  |
| EMC               | Electromagnetic compatibility   |
| EN                | European norm   |
| ESD               | Electrostatic discharge   |
| FB                | Fieldbus  |
| FBD               | Function block diagrams   |
| FTA               | Field termination assembly  |
| FTT               | Fault tolerance time  |
| ICMP              | Internet control message protocol: Network protocol for status or error messages  |
| IEC               | International electrotechnical commission   |
| MAC Address       | Media access control address: Hardware address of one network connection  |
| PADT              | Programming and debugging tool (in accordance with IEC 61131-3),<br>PC with SILworX   |
| PE                | Protective earth  |
| PELV              | Protective extra low voltage  |
| PES               | Programmable electronic system  |
| R                 | Read  |
| Rack ID           | Base plate identification (number)  |
| Interference-free | Supposing that two input circuits are connected to the same source (e.g., a transmitter). An input circuit is termed "interference-free" if it does not distort the signals of the other input circuit. |
| R/W               | Read/Write  |
| SB                | System bus (module)   |
| SELV              | Safety extra low voltage  |
| SFF               | Safe failure fraction, portion of faults that can be safely controlled  |
| SIL               | Safety integrity level (in accordance with IEC 61508)   |
| SILworX           | Programming tool for HIMax and HIMatrix   |
| SNTP              | Simple network time protocol (RFC 1769)   |
| SRS               | System.Rack.Slot  |
| SW                | Software  |
| TMO               | Timeout   |
| W                 | Write   |
| WD                | Watchdog  |
| WDT               | Watchdog time   |

## Index of Figures

|   |     |
|---|-----|
| <b>Figure 1: System Structures</b>  | 40  |
| <b>Figure 2: Structure for Configuring a Redundant Connection</b>   | 41  |
| <b>Figure 3: Resource Structure Tree</b>  | 41  |
| <b>Figure 4: Parameter Values for a Redundant safeethernet Connection</b>   | 42  |
| <b>Figure 5: Redundant Connection between HIMax and HIMatrix F*03 Controllers</b>   | 52  |
| <b>Figure 6: Ring Connection between HIMax and three HIMatrix standard via two Transmission Paths<br/>(Channel 1 and Channel 2)</b> | 53  |
| <b>Figure 7: Redundant Connection of Two HIMax and one HIMatrix standard Controllers using one<br/>Transmission Path</b>            | 54  |
| <b>Figure 8: Example of Switch Ports separated via VLAN</b>   | 55  |
| <b>Figure 9: Reaction Time with Interconnection of Two HIMax Controllers</b>  | 61  |
| <b>Figure 10: Reaction Time for Connection between One HIMax and One HIMatrix Controller</b>  | 61  |
| <b>Figure 11: Reaction Time with Two Remote I/Os and One HIMax controller</b>   | 62  |
| <b>Figure 12: Reaction Time with Two HIMax Controllers and One HIMatrix controller</b>  | 62  |
| <b>Figure 13: Reaction Time with Interconnection of Two HIMatrix Controllers</b>  | 63  |
| <b>Figure 14: Reaction Time with Remote I/Os</b>  | 64  |
| <b>Figure 15: safeethernet Connection between Resource A1 in Project A and Resource B1 in Project B</b>                             | 70  |
| <b>Figure 16: Cross-Project communication between two separate SILworX projects</b>   | 71  |
| <b>Figure 17: HIMatrix Proxy Resource</b>   | 72  |
| <b>Figure 18: HIMax Proxy Resource</b>  | 75  |
| <b>Figure 19: Cross-Project Communication between SILworX and ELOP II Factory</b>   | 77  |
| <b>Figure 20: HIMatrix Proxy Resource</b>   | 78  |
| <b>Figure 21: safeethernet Connection Export</b>  | 80  |
| <b>Figure 22: Importing Connections in ELOP II Factory</b>  | 81  |
| <b>Figure 23: Peer-to-Peer-Editor Editor in ELOP II Factory</b>   | 81  |
| <b>Figure 24: Transport direction HIMax Proxy Resource A-&gt;HIMatrix Resource B area.</b>  | 82  |
| <b>Figure 25: Transport direction HIMax Proxy Resource A-&gt;HIMatrix Resource B.</b>   | 82  |
| <b>Figure 26: Control Panel for Connection Control</b>  | 83  |
| <b>Figure 27: Controlling the Consumer/Provider Status (IOxS)</b>   | 92  |
| <b>Figure 28: PROFIsafe Control Byte and Status Byte</b>  | 94  |
| <b>Figure 29: Reaction Time between an F-Device and a HIMA F-Host</b>   | 96  |
| <b>Figure 30: Reaction Time using a HIMA F-Host and two F-Devices</b>   | 96  |
| <b>Figure 31: Structure Tree for the PROFINET IO Controller</b>   | 101 |
| <b>Figure 32: Device Access Point (DAP) for the PROFINET-IO Device</b>  | 102 |
| <b>Figure 33: Structure Tree for the PROFINET IO Controller</b>   | 106 |
| <b>Figure 34: Communication via PROFINET IO/PROFIsafe</b>   | 120 |
| <b>Figure 35: Structure Tree for the PROFINET IO Device</b>   | 126 |
| <b>Figure 36: Communication via PROFINET IO</b>   | 137 |
| <b>Figure 37: HIMax PROFIBUS DP Slave with Modules</b>  | 140 |

|  |     |
|--|-----|
| <b>Figure 38: User Data Field</b>  | 143 |
| <b>Figure 39: Verification Dialog Box</b>                                | 143 |
| <b>Figure 40: PROFIBUS DP Master Properties</b>                          | 144 |
| <b>Figure 41: PROFIBUS DP Slave Properties</b>                           | 145 |
| <b>Figure 42: Isochronous PROFIBUS DP Cycle</b>                          | 155 |
| <b>Figure 43: <i>Edit User Parameters</i> Dialog Box</b>                 | 165 |
| <b>Figure 44: MSTAT Function Block</b>                                   | 167 |
| <b>Figure 45: RALRM Function Block</b>                                   | 170 |
| <b>Figure 46: RDIAG Function Block</b>                                   | 174 |
| <b>Figure 47: RDREC Function Block</b>                                   | 178 |
| <b>Figure 48: SЛАCT Function Block</b>                                   | 181 |
| <b>Figure 49: WRREC Function Block</b>                                   | 184 |
| <b>Figure 50: The ACTIVE Auxiliary Function Block</b>                    | 187 |
| <b>Figure 51: The ALARM Auxiliary Function Block</b>                     | 188 |
| <b>Figure 52: The DEID Auxiliary Function Block</b>                      | 189 |
| <b>Figure 53: The ID Auxiliary Function Block</b>                        | 190 |
| <b>Figure 54: The NSLOT Auxiliary Function Block</b>                     | 191 |
| <b>Figure 55: The SLOT Auxiliary Function Block</b>                      | 191 |
| <b>Figure 56: The STDDIAG Auxiliary Function Block</b>                   | 193 |
| <b>Figure 57: RS485 Bus Topology</b>                                     | 207 |
| <b>Figure 58: Communication via Modbus TCP/IP</b>                        | 211 |
| <b>Figure 59: Modbus Network</b>   | 226 |
| <b>Figure 60: Modbus Gateway</b>   | 229 |
| <b>Figure 61: Serial Modbus</b>  | 232 |
| <b>Figure 62: Modbus Telegram</b>  | 232 |
| <b>Figure 63: Connecting a HIMax to a Siemens Controller</b>             | 259 |
| <b>Figure 64: Data Transfer between a HIMax and a Siemens Controller</b> | 260 |
| <b>Figure 65: List of Variables in the Siemens UDT1 Block</b>            | 261 |
| <b>Figure 66: List of Variables in the Siemens DB1 Function Block</b>    | 262 |
| <b>Figure 67: SIMATIC Symbol Editor</b>                                  | 262 |
| <b>Figure 68: Receive Function Chart</b>                                 | 263 |
| <b>Figure 69: Send Function Chart</b>                                    | 264 |
| <b>Figure 70: TCP Connection Properties in SILworX</b>                   | 265 |
| <b>Figure 71: Siemens List of Variables</b>                              | 276 |
| <b>Figure 72: HIMax/HIMatrix List of Variables</b>                       | 276 |
| <b>Figure 73: Function Block TCP_Reset</b>                               | 278 |
| <b>Figure 74: Function Block TCP_Send</b>                                | 281 |
| <b>Figure 75: Function Block TCP_Receive</b>                             | 284 |
| <b>Figure 76: Function Block TCP_ReceiveLine</b>                         | 288 |
| <b>Figure 77: Function Block TCP_ReceiveVar</b>                          | 292 |

---

|  |     |
|--|-----|
| Figure 78: Data Packet Structure   | 293 |
| Figure 79: Redundant X-OPC Operation   | 308 |
| Figure 80: Wizard for Installing the X-OPC Server                                    | 309 |
| Figure 81: Wizard for Installing the X-OPC Server                                    | 309 |
| Figure 82: Setting the Class ID of the Second X-OPC Server Manually                  | 310 |
| Figure 83: Redundant X-OPC Operation   | 312 |
| Figure 84: Redundant X-OPC Operation   | 313 |
| Figure 85: Alarm & Event Editor  | 316 |
| Figure 86: Redundant X-OPC Operation   | 317 |
| Figure 87: safeethernet Editor   | 320 |
| Figure 88: Detailed View of the safeethernet Connection                              | 321 |
| Figure 89: Five Areas of a Scalar Event  | 323 |
| Figure 90: Block Diagram   | 331 |
| Figure 91: Structure of the HART over IP Installation                                | 337 |
| Figure 92: Process Data Exchange between CPU and COM (CUT)                           | 346 |
| Figure 93: Cygwin Setup Dialog Box   | 382 |
| Figure 94: Select Packages Cygwin Setup Dialog Box                                   | 384 |
| Figure 95: Cygwin Structure Tree   | 385 |
| Figure 97: Cygwin Structure Tree   | 386 |
| Figure 98: C Code File in the <code>example_cut</code> Folder                        | 386 |
| Figure 99: <code>.mke</code> File in the <code>example_cut</code> Folder             | 387 |
| Figure 100: <code>.mke</code> File Starting with Line 1                              | 387 |
| Figure 101: <code>makefile</code> in the <code>example_cut</code> Folder             | 388 |
| Figure 102: <code>makefile</code> Starting with Line 34                              | 388 |
| Figure 103: <code>makeinc.inc.app</code> file in the <code>example_cut</code> Folder | 388 |
| Figure 104: <code>makeinc.inc.app</code> Starting with Line 247                      | 389 |
| Figure 105: Resource Structure Tree  | 390 |
| Figure 106: C Code <code>example_cut.c</code>  | 391 |
| Figure 107: Cygwin Bash Shell  | 392 |
| Figure 108: SILworX Program Editor   | 395 |
| Figure 109: SILworX Online Test  | 395 |
| Figure 110: PNM_MSTST Function Block (Upper Part)                                    | 397 |
| Figure 111: PNM_MSTST Function Block (Lower Part)                                    | 398 |
| Figure 112: Choosing Function Blocks   | 398 |

**Index of Tables**

|                  |   |    |
|------------------|---|----|
| <b>Table 1:</b>  | <b>Additional Valid Manuals</b>   | 14 |
| <b>Table 2:</b>  | <b>Standard Protocols</b>   | 19 |
| <b>Table 3:</b>  | <b>Available Standard Protocols</b>                                       | 20 |
| <b>Table 4:</b>  | <b>Protocols on one Communication Module</b>                              | 20 |
| <b>Table 5:</b>  | <b>Protocols on one Communication Module</b>                              | 20 |
| <b>Table 6:</b>  | <b>Protocols Available for the HIMax/HIMatrix Systems</b>                 | 22 |
| <b>Table 7:</b>  | <b>M-COM 010 x Communication Module</b>                                   | 22 |
| <b>Table 8:</b>  | <b>HIMax Ethernet Interfaces</b>  | 24 |
| <b>Table 9:</b>  | <b>HIMatrix Ethernet Interfaces</b>                                       | 24 |
| <b>Table 10:</b> | <b>Configuration Parameters</b>   | 27 |
| <b>Table 11:</b> | <b>Routing Parameters</b>   | 28 |
| <b>Table 12:</b> | <b>Ethernet Switch Parameters</b>   | 28 |
| <b>Table 13:</b> | <b>Values for LLDP</b>  | 29 |
| <b>Table 14:</b> | <b>Available Fieldbus Submodules</b>                                      | 31 |
| <b>Table 15:</b> | <b>Options for Fieldbus Interfaces FB1 and FB2</b>                        | 31 |
| <b>Table 16:</b> | <b>Available HIMax Components</b>   | 32 |
| <b>Table 17:</b> | <b>Examples of COM Module Part Numbers and Names</b>                      | 32 |
| <b>Table 18:</b> | <b>Equipment of HIMatrix Controller with Fieldbus Submodules</b>          | 32 |
| <b>Table 19:</b> | <b>Examples of HIMatrix Controller Part Numbers</b>                       | 33 |
| <b>Table 20:</b> | <b>Pin Assignment of D-Sub Connectors for RS485</b>                       | 33 |
| <b>Table 21:</b> | <b>Pin Assignment of D-sub Connectors for PROFIBUS DP</b>                 | 33 |
| <b>Table 22:</b> | <b>Pin Assignment of D-Sub Connectors for RS232</b>                       | 34 |
| <b>Table 23:</b> | <b>Pin Assignment of D-Sub Connectors for RS422</b>                       | 34 |
| <b>Table 24:</b> | <b>Pin Assignment of D-Sub Connectors for SSI</b>                         | 34 |
| <b>Table 25:</b> | <b>Pin Assignment of D-Sub Connectors for CAN</b>                         | 35 |
| <b>Table 26:</b> | <b>Safety-Related Protocol (safeethernet)</b>                             | 37 |
| <b>Table 27:</b> | <b>safeethernet Protocol Parameters</b>                                   | 45 |
| <b>Table 28:</b> | <b>System Variables Tab in the safeethernet Editor</b>                    | 49 |
| <b>Table 29:</b> | <b>Tab Fragment Definitions</b>   | 50 |
| <b>Table 30:</b> | <b>Available Ethernet Interfaces</b>                                      | 51 |
| <b>Table 31:</b> | <b>Potential Subscribers are HIMax and HIMatrix F*03 Controllers</b>      | 52 |
| <b>Table 32:</b> | <b>Potential Ring Master are HIMax and HIMatrix F*03 Controllers</b>      | 53 |
| <b>Table 33:</b> | <b>Possible ring slaves are HIMax, HIMatrix F*03, HIMatrix standard.</b>  | 53 |
| <b>Table 34:</b> | <b>Interface Combinations with one Ethernet Interface (IP Address)</b>    | 54 |
| <b>Table 35:</b> | <b>Interface Combinations with two Ethernet Interfaces (IP Addresses)</b> | 54 |
| <b>Table 36:</b> | <b>VLAN Tab</b>   | 55 |
| <b>Table 37:</b> | <b>View Box of the safeethernet Connection</b>                            | 84 |
| <b>Table 38:</b> | <b>safeethernet Reload after Changes</b>                                  | 86 |
| <b>Table 39:</b> | <b>Messages from the Code Generator</b>                                   | 89 |

|  |     |
|--|-----|
| <b>Table 40: Messages from the Operating System</b>                                  | 89  |
| <b>Table 41: Overview of PROFINET IO Function Blocks</b>                             | 91  |
| <b>Table 42: Equipment and System Requirements for the PROFINET IO Controller.</b>   | 100 |
| <b>Table 43: PROFINET IO Controller Properties</b>                                   | 100 |
| <b>Table 44: The PROFINET IO Device (Properties) Tab</b>                             | 107 |
| <b>Table 45: Parameters (Properties) Tab</b>   | 108 |
| <b>Table 46: Parameters (Properties) Tab</b>   | 108 |
| <b>Table 47: Parameters Tab</b>  | 109 |
| <b>Table 48: System Variables Tab</b>  | 109 |
| <b>Table 49: Parameter Tab of the I/O PROFINET IO Module</b>                         | 110 |
| <b>Table 50: Parameters Tab</b>  | 111 |
| <b>Table 51: System Variables Tab</b>  | 114 |
| <b>Table 52: F Parameters for Submodule Input (Properties)</b>                       | 115 |
| <b>Table 53: Parameters Tab</b>  | 116 |
| <b>Table 54: System Variables Tab</b>  | 117 |
| <b>Table 55: Parameters Tab</b>  | 118 |
| <b>Table 56: System Variables Tab</b>  | 119 |
| <b>Table 57: Application Relation (Properties)</b>                                   | 120 |
| <b>Table 58: Alarm CR (Properties)</b>   | 121 |
| <b>Table 59: Input CR (Properties)</b>   | 122 |
| <b>Table 60: Input CR (Edit)</b>   | 123 |
| <b>Table 61: Output CR (Properties)</b>  | 124 |
| <b>Table 62: Equipment and System Requirements for the PROFINET IO Controller.</b>   | 125 |
| <b>Table 63: PROFINET IO Controller Properties</b>                                   | 125 |
| <b>Table 64: PROFINET IO Device General Properties</b>                               | 129 |
| <b>Table 65: PROFINET IO Modules</b>   | 130 |
| <b>Table 66: PROFIsafe Modules</b>   | 131 |
| <b>Table 67: Device Module General Properties</b>                                    | 132 |
| <b>Table 68: Edit Dialog Box for the Input Submodule</b>                             | 134 |
| <b>Table 69: Equipment and System Requirements</b>                                   | 136 |
| <b>Table 70: PROFIBUS DP Master Properties</b>                                       | 136 |
| <b>Table 71: Outputs in the HIMA PROFIBUS DP Slave</b>                               | 138 |
| <b>Table 72: Inputs in the HIMA PROFIBUS DP Slave</b>                                | 138 |
| <b>Table 73: Variables of the Input Module [000] DP Input/ELOP Export: 2 Bytes</b>   | 141 |
| <b>Table 74: Variables of the Input Module [001] DP Input/ELOP Export: 8 Bytes</b>   | 141 |
| <b>Table 75: Variables of the Input Module [002] DP Input/ELOP Export: 1 Byte</b>    | 141 |
| <b>Table 76: Variables of the Output Module [003] DP Output/ELOP Import: 2 Bytes</b> | 142 |
| <b>Table 77: Variables of the Output Module [004] DP Output/ELOP Import: 1 Byte</b>  | 142 |
| <b>Table 78: System Variables in the PROFIBUS DP Master</b>                          | 146 |
| <b>Table 79: General Properties for PROFIBUS DP Master</b>                           | 148 |

---

|  |     |
|--|-----|
| <b>Table 80: Timings Tab in the Properties Dialog Box for the PROFIBUS DP Master</b>       | 150 |
| <b>Table 81: CPU/COM Tab in the Properties Dialog Box for the PROFIBUS DP Master</b>       | 150 |
| <b>Table 82: Other Properties for the PROFIBUS DP Master</b>                               | 151 |
| <b>Table 83: Default Values for Token Rotation Time Used with Different Transfer Rates</b> | 152 |
| <b>Table 84: Transmission Time for a Character Used with different Transfer Rates</b>      | 153 |
| <b>Table 85: Elements Required for Calculating the Target Token Rotation Time</b>          | 153 |
| <b>Table 86: System Variables in the PROFIBUS DP Slave</b>                                 | 157 |
| <b>Table 87: Parameters Tab in the PROFIBUS DP Slave</b>                                   | 158 |
| <b>Table 88: Groups Tab in the Properties Dialog Box for the PROFIBUS DP Slave</b>         | 159 |
| <b>Table 89: DP V1 Tab in the Properties Dialog Box for the PROFIBUS DP Slave</b>          | 159 |
| <b>Table 90: Alarms Tab in the Properties Dialog Box for the PROFIBUS DP Slave</b>         | 160 |
| <b>Table 91: Data Tab in the Properties Dialog Box for the PROFIBUS DP Slave</b>           | 160 |
| <b>Table 92: Model Tab in the Properties Dialog Box for the PROFIBUS DP Slave</b>          | 161 |
| <b>Table 93: Features Tab in the Properties Dialog Box for the PROFIBUS DP Slave</b>       | 161 |
| <b>Table 94: Baud Rates Tab in the Properties Dialog Box for the PROFIBUS DP Slave</b>     | 162 |
| <b>Table 95: Acyclic Tab in the Properties Dialog Box for the PROFIBUS DP Slave</b>        | 162 |
| <b>Table 96: GSD File of the HIMax/HIMatrix PROFIBUS DP Slave</b>                          | 163 |
| <b>Table 97: Example: Group 1...4 of the User Data Field</b>                               | 165 |
| <b>Table 98: Example: Group 1...4 of the User Data Field</b>                               | 165 |
| <b>Table 99: Overview of the PROFIBUS DP Function Blocks</b>                               | 166 |
| <b>Table 100: A-Inputs for the MSTAT Function Block</b>                                    | 167 |
| <b>Table 101: A-Outputs for the MSTAT Function Block</b>                                   | 167 |
| <b>Table 102: F-Inputs for the MSTAT Function Block</b>                                    | 168 |
| <b>Table 103: F-Outputs for the MSTAT Function Block</b>                                   | 168 |
| <b>Table 104: Input System Variables</b>   | 168 |
| <b>Table 105: Output System Variables</b>  | 169 |
| <b>Table 106: A-Inputs for the RDIAG Function Block</b>                                    | 170 |
| <b>Table 107: A-Outputs for the RDIAG Function Block</b>                                   | 171 |
| <b>Table 108: F-Inputs for the RALRM Function Block</b>                                    | 171 |
| <b>Table 109: F-Outputs for the RALRM Function Block</b>                                   | 171 |
| <b>Table 110: Input System Variables</b>   | 172 |
| <b>Table 111: Output System Variables</b>  | 172 |
| <b>Table 112: Alarm Data</b>   | 173 |
| <b>Table 113: A-Inputs for the RDIAG Function Block</b>                                    | 174 |
| <b>Table 114: A-Outputs for the RDIAG Function Block</b>                                   | 174 |
| <b>Table 115: F-Inputs for the RDIAG Function Block</b>                                    | 175 |
| <b>Table 116: F-Outputs for the RDIAG Function Block</b>                                   | 175 |
| <b>Table 117: Input System Variables</b>   | 175 |
| <b>Table 118: Output System Variables</b>  | 176 |
| <b>Table 119: Diagnostic Data</b>  | 176 |

---

|  |     |
|--|-----|
| <b>Table 120: A-Inputs for the RDREC Function Block</b>            | 178 |
| <b>Table 121: A-Outputs for the RDREC Function Block</b>           | 178 |
| <b>Table 122: F-Inputs for the RDREC Function Block</b>            | 179 |
| <b>Table 123: F-Outputs for the RDREC Function Block</b>           | 179 |
| <b>Table 124: Input System Variables</b>                           | 179 |
| <b>Table 125: Output System Variables</b>                          | 180 |
| <b>Table 126: Data</b>   | 180 |
| <b>Table 127: A-Inputs for the SLACT Function Block</b>            | 181 |
| <b>Table 128: A-Outputs for the SLACT Function Block</b>           | 182 |
| <b>Table 129: F-Inputs for the SLACT Function Block</b>            | 182 |
| <b>Table 130: F-Outputs for the SLACT Function Block</b>           | 182 |
| <b>Table 131: Input System Variables</b>                           | 182 |
| <b>Table 132: Output System Variables</b>                          | 183 |
| <b>Table 133: A-Inputs for the WRREC Function Block</b>            | 184 |
| <b>Table 134: A-Outputs for the WRREC Function Block</b>           | 184 |
| <b>Table 135: F-Inputs for the WRREC Function Block</b>            | 185 |
| <b>Table 136: F-Outputs for the WRREC Function Block</b>           | 185 |
| <b>Table 137: Input System Variables</b>                           | 185 |
| <b>Table 138: Output System Variables</b>                          | 186 |
| <b>Table 139: Data</b>   | 186 |
| <b>Table 140: Overview of the Auxiliary Function Blocks</b>        | 187 |
| <b>Table 141: Inputs for the ACTIVE Auxiliary Function Block</b>   | 187 |
| <b>Table 142: Outputs for the ACTIVE Auxiliary Function Block</b>  | 187 |
| <b>Table 143: Inputs for the ALARM Auxiliary Function Block</b>    | 188 |
| <b>Table 144: Outputs for the ALARM Auxiliary Function Block</b>   | 189 |
| <b>Table 145: Inputs for the DEID Auxiliary Function Block</b>     | 189 |
| <b>Table 146: Outputs for the DEID Auxiliary Function Block</b>    | 189 |
| <b>Table 147: Inputs for the ID Auxiliary Function Block</b>       | 190 |
| <b>Table 148: Outputs for the ID Auxiliary Function Block</b>      | 190 |
| <b>Table 149: Inputs for the NSLOT Auxiliary Function Block</b>    | 191 |
| <b>Table 150: Outputs for the NSLOT Auxiliary Function Block</b>   | 191 |
| <b>Table 151: Inputs for the SLOT Auxiliary Function Block</b>     | 193 |
| <b>Table 152: Outputs for the SLOT Auxiliary Function Block</b>    | 193 |
| <b>Table 153: Inputs for the STDDIAG Auxiliary Function Block</b>  | 194 |
| <b>Table 154: Outputs for the STDDIAG Auxiliary Function Block</b> | 194 |
| <b>Table 155: Error Codes of the Function Blocks</b>               | 195 |
| <b>Table 156: View Box of the PROFIBUS Master</b>                  | 198 |
| <b>Table 157: PROFIBUS DP Master State</b>                         | 198 |
| <b>Table 158: Behavior of the PROFIBUS DP Master</b>               | 198 |
| <b>Table 159: The FBx LED</b>                                      | 199 |

---

|  |     |
|--|-----|
| <b>Table 160: The FAULT FBx</b>  | 199 |
| <b>Table 161: Equipment and System Requirements for the HIMA PROFIBUS DP Slave</b> | 200 |
| <b>Table 162: Properties of the HIMA PROFIBUS DP Slave</b>                         | 200 |
| <b>Table 163: System Variables in the PROFIBUS DP Slave</b>                        | 202 |
| <b>Table 164: Slave Properties: General Tab</b>                                    | 204 |
| <b>Table 165: View Box of the PROFIBUS DP Slave</b>                                | 205 |
| <b>Table 166: The FBx LED</b>  | 206 |
| <b>Table 167: The FAULT FBx</b>  | 206 |
| <b>Table 168: Terminal Assignment for H 7506</b>                                   | 208 |
| <b>Table 169: Bus Cable</b>  | 208 |
| <b>Table 170: Properties of the RS485 Transmission</b>                             | 209 |
| <b>Table 171: Equipment and System Requirements for the Modbus Master</b>          | 210 |
| <b>Table 172: Properties of the Modbus Master</b>                                  | 211 |
| <b>Table 173: System Variables for the Modbus Master</b>                           | 216 |
| <b>Table 174: General Properties of the Modbus Master</b>                          | 218 |
| <b>Table 175: Parameters of COM/CPU</b>  | 218 |
| <b>Table 176: Modbus Function Codes</b>  | 219 |
| <b>Table 177: Request Telegram Read Coils</b>                                      | 222 |
| <b>Table 178: Request Telegram Read Discrete Inputs</b>                            | 222 |
| <b>Table 179: Request Telegram Read Holding Registers</b>                          | 222 |
| <b>Table 180: Request Telegram Read Input Registers</b>                            | 223 |
| <b>Table 181: Read Write Holding Register</b>                                      | 224 |
| <b>Table 182: Request Telegram Write Multiple Coils</b>                            | 224 |
| <b>Table 183: Request Telegram Write Multiple Registers</b>                        | 225 |
| <b>Table 184: Request Telegram Write Single Coil (05)</b>                          | 225 |
| <b>Table 185: Request Telegram Write Single Register</b>                           | 225 |
| <b>Table 186: System Variables for TCP/UDP Slaves</b>                              | 227 |
| <b>Table 187: Configuration Parameters</b>   | 228 |
| <b>Table 188: Connection Parameters for the Modbus Gateway</b>                     | 231 |
| <b>Table 189: Status Variables for the Gateway Slave</b>                           | 231 |
| <b>Table 190: Connection Parameters for the Gateway Slave</b>                      | 231 |
| <b>Table 191: Parameters for the Serial Modbus Master</b>                          | 233 |
| <b>Table 192: System Variables in the Modbus Slave</b>                             | 234 |
| <b>Table 193: Connection Parameters for the Modbus Master</b>                      | 234 |
| <b>Table 194: View Box of the Modbus Master</b>                                    | 235 |
| <b>Table 195: The FBx LED</b>  | 236 |
| <b>Table 196: The FAULT FBx</b>  | 236 |
| <b>Table 197: Equipment and System Requirements for the HIMA Modbus Slave</b>      | 237 |
| <b>Table 198: Properties of the Modbus Slave</b>                                   | 238 |
| <b>Table 199: Slots Allowed for the Redundant Modbus Slave COM Modules</b>         | 240 |

|   |     |
|---|-----|
| <b>Table 200: Modbus Slave Properties Set Tab</b>                             | 242 |
| <b>Table 201: Modbus Slave Set Tab</b>  | 243 |
| <b>Table 202: TCP and UDP Ports Tab for HIMA Modbus Slave</b>                 | 245 |
| <b>Table 203: System Variables Tab for the HIMA Modbus Slave</b>              | 245 |
| <b>Table 204: Modbus Function Codes of the HIMA Modbus Slave</b>              | 246 |
| <b>Table 205: Register Variables in the Register Area of the Modbus Slave</b> | 250 |
| <b>Table 206: Bit Variables in the Bit Area of the Modbus Slave</b>           | 251 |
| <b>Table 207: Offsets Tab for HIMA Modbus Slave</b>                           | 252 |
| <b>Table 208: Variables Mirrored from the Register Area to the Bit Area</b>   | 253 |
| <b>Table 209: Variables Mirrored from the Bit Area to the Register Area</b>   | 254 |
| <b>Table 210: View Box of the Modbus Slave</b>                                | 256 |
| <b>Table 211: Master Data View Box</b>  | 256 |
| <b>Table 212: The FBx LED</b>   | 257 |
| <b>Table 213: The FAULT FBx</b>   | 257 |
| <b>Table 214: Error Codes of Modbus TCP/IP</b>                                | 257 |
| <b>Table 215: Equipment and System Requirements for the S&amp;R TCP</b>       | 258 |
| <b>Table 216: S&amp;R TCP Properties</b>                                      | 258 |
| <b>Table 217: HIMax Controller Configuration</b>                              | 260 |
| <b>Table 218: Siemens SIMATIC 300 Configuration</b>                           | 260 |
| <b>Table 219: Global Variables</b>  | 265 |
| <b>Table 220: Variables for Receive Data</b>                                  | 267 |
| <b>Table 221: Variables for Send Data</b>                                     | 267 |
| <b>Table 222: System Variables S&amp;R TCP</b>                                | 268 |
| <b>Table 223: S&amp;R TCP General Properties</b>                              | 269 |
| <b>Table 224: Parameters of COM/CPU</b>                                       | 270 |
| <b>Table 225: System Variables</b>  | 271 |
| <b>Table 226: S&amp;R TCP Connection Properties</b>                           | 273 |
| <b>Table 227: Function Blocks for S&amp;R TCP Connections</b>                 | 277 |
| <b>Table 228: A-Inputs for the TCP_Reset Function Block</b>                   | 278 |
| <b>Table 229: A-Outputs for the TCP_Reset Function Block</b>                  | 278 |
| <b>Table 230: F-Inputs for the TCP_Reset Function Block</b>                   | 279 |
| <b>Table 231: F-Outputs for the TCP_Reset Function Block</b>                  | 279 |
| <b>Table 232: Input System Variables</b>                                      | 279 |
| <b>Table 233: Output System Variables</b>                                     | 280 |
| <b>Table 234: A-Inputs for the TCP_Send Function Block</b>                    | 281 |
| <b>Table 235: A-Outputs for the TCP_Send Function Block</b>                   | 281 |
| <b>Table 236: F-Inputs for the TCP_Send Function Block</b>                    | 282 |
| <b>Table 237: F-Outputs for the TCP_Send Function Block</b>                   | 282 |
| <b>Table 238: Input System Variables</b>                                      | 282 |
| <b>Table 239: Output System Variables</b>                                     | 283 |

|  |     |
|--|-----|
| <b>Table 240: Send Data</b>  | 283 |
| <b>Table 241: A-Inputs for the TCP_Receive Function Block</b>            | 284 |
| <b>Table 242: A-Outputs for the TCP_Receive Function Block</b>           | 285 |
| <b>Table 243: A-Inputs for the TCP_Receive Function Block</b>            | 285 |
| <b>Table 244: F-Outputs for the TCP_Receive Function Block</b>           | 285 |
| <b>Table 245: Input System Variables</b>                                 | 286 |
| <b>Table 246: Output System Variables</b>                                | 286 |
| <b>Table 247: Receive Variables</b>                                      | 286 |
| <b>Table 248: A-Inputs for the TCP_ReceiveLine Function Block</b>        | 288 |
| <b>Table 249: A-Outputs for the TCP_ReceiveLine Function Block</b>       | 289 |
| <b>Table 250: F-Inputs for the TCP_ReceiveLine Function Block</b>        | 289 |
| <b>Table 251: A-Outputs for the TCP_ReceiveLine Function Block</b>       | 289 |
| <b>Table 252: Input System Variables</b>                                 | 290 |
| <b>Table 253: Output System Variables</b>                                | 290 |
| <b>Table 254: Receive Variables</b>                                      | 290 |
| <b>Table 255: A-Inputs for the TCP_ReceiveVar Function Block</b>         | 293 |
| <b>Table 256: A-Outputs for the TCP_ReceiveVar Function Block</b>        | 294 |
| <b>Table 257: F-Inputs for the TCP_ReceiveVar Function Block</b>         | 294 |
| <b>Table 258: F-Outputs for the TCP_ReceiveVar Function Block</b>        | 294 |
| <b>Table 259: Input System Variables</b>                                 | 295 |
| <b>Table 260: Output System Variables</b>                                | 295 |
| <b>Table 261: Receive Variables</b>                                      | 295 |
| <b>Table 262: S&amp;R Protocol View Box</b>                              | 297 |
| <b>Table 263: View Box of the Modbus Slave</b>                           | 297 |
| <b>Table 264: Error Codes of the TCP Connection</b>                      | 298 |
| <b>Table 265: Additional Error Codes</b>                                 | 299 |
| <b>Table 266: Connection State</b>                                       | 299 |
| <b>Table 267: Partner's Connection State</b>                             | 299 |
| <b>Table 268: Equipment and System Requirements for the S&amp;R TCP</b>  | 300 |
| <b>Table 269: SNTP Client Properties</b>                                 | 301 |
| <b>Table 270: SNTP Server Info Properties</b>                            | 302 |
| <b>Table 271: SNTP Server Properties</b>                                 | 303 |
| <b>Table 272: Equipment and System Requirements for the X-OPC Server</b> | 304 |
| <b>Table 273: X-OPC Server Properties</b>                                | 305 |
| <b>Table 274: HIMax Controller Properties for X-OPC Connection</b>       | 306 |
| <b>Table 275: Actions Required as a Result of Changes</b>                | 307 |
| <b>Table 276: Default Values Associated with the Priorities</b>          | 319 |
| <b>Table 277: State and Timestamp for the Data Access Fragments</b>      | 321 |
| <b>Table 278: Parameters for Boolean Events</b>                          | 323 |
| <b>Table 279: Parameters for Scalar Events</b>                           | 325 |

| Appendix  | Communication |
|---|---------------|
| <b>Table 280: Properties</b>  | <b>329</b>    |
| <b>Table 281: Properties</b>  | <b>330</b>    |
| <b>Table 282: Edit</b>  | <b>330</b>    |
| <b>Table 283: Equipment and System Requirements for the ComUserTask</b>       | <b>331</b>    |
| <b>Table 284: Data Format of the SSISTART Start Command Bytes</b>             | <b>332</b>    |
| <b>Table 285: Format and Order of the Sensor Data</b>                         | <b>333</b>    |
| <b>Table 286: Recommended Clock Rates According to the Field Wire Lengths</b> | <b>333</b>    |
| <b>Table 287: Equipment and System Requirements for HART over IP</b>          | <b>335</b>    |
| <b>Table 288: Properties of the HART Over IP Protocol Instance</b>            | <b>335</b>    |
| <b>Table 289: HART Protocol Properties</b>                                    | <b>336</b>    |
| <b>Table 290: Parameters and Variables Required for the Configuration</b>     | <b>338</b>    |
| <b>Table 291: Online View of the HART Protocol</b>                            | <b>340</b>    |
| <b>Table 292: Online View of the HART Protocol</b>                            | <b>341</b>    |
| <b>Table 293: HART Field Device Addressing</b>                                | <b>342</b>    |
| <b>Table 294: Equipment and System Requirements for the ComUserTask</b>       | <b>343</b>    |
| <b>Table 295: ComUserTask Properties</b>                                      | <b>343</b>    |
| <b>Table 296: Abbreviations</b>   | <b>344</b>    |
| <b>Table 297: Schedule Interval [ms]</b>                                      | <b>345</b>    |
| <b>Table 298: PROFINET IO Device General Properties</b>                       | <b>347</b>    |
| <b>Table 299: ComUserTask System Variables</b>                                | <b>348</b>    |
| <b>Table 300: Memory Area for Code and Data</b>                               | <b>351</b>    |
| <b>Table 301: Stack Memory</b>  | <b>351</b>    |
| <b>Table 302: Parameter</b>   | <b>351</b>    |
| <b>Table 303: Parameter</b>   | <b>353</b>    |
| <b>Table 304: Return Value</b>  | <b>353</b>    |
| <b>Table 305: Parameter</b>   | <b>354</b>    |
| <b>Table 306: Return Value</b>  | <b>354</b>    |
| <b>Table 307: Parameter</b>   | <b>355</b>    |
| <b>Table 308: Return Value</b>  | <b>356</b>    |
| <b>Table 309: Parameter</b>   | <b>357</b>    |
| <b>Table 310: Parameter</b>   | <b>358</b>    |
| <b>Table 311: Return Value</b>  | <b>358</b>    |
| <b>Table 312: Parameter</b>   | <b>359</b>    |
| <b>Table 313: Parameter</b>   | <b>360</b>    |
| <b>Table 314: Return Value</b>  | <b>360</b>    |
| <b>Table 315: Return Value</b>  | <b>361</b>    |
| <b>Table 316: Parameter</b>   | <b>362</b>    |
| <b>Table 317: Parameter</b>   | <b>363</b>    |
| <b>Table 318: Return Value</b>  | <b>363</b>    |
| <b>Table 319: Parameter</b>   | <b>364</b>    |

|   |     |
|---|-----|
| <b>Table 320: Parameter</b>                       | 365 |
| <b>Table 321: Parameter</b>                       | 366 |
| <b>Table 322: Return Value</b>                    | 366 |
| <b>Table 323: Parameter</b>                       | 367 |
| <b>Table 324: Parameter</b>                       | 368 |
| <b>Table 325: Return Value</b>                    | 368 |
| <b>Table 326: Parameter</b>                       | 369 |
| <b>Table 327: Return Value</b>                    | 369 |
| <b>Table 328: Parameter</b>                       | 370 |
| <b>Table 329: Parameter</b>                       | 371 |
| <b>Table 330: Return Value</b>                    | 371 |
| <b>Table 331: Parameter</b>                       | 372 |
| <b>Table 332: Parameter</b>                       | 373 |
| <b>Table 333: Return Value</b>                    | 373 |
| <b>Table 334: Parameter</b>                       | 375 |
| <b>Table 335: Parameter</b>                       | 377 |
| <b>Table 336: Parameter</b>                       | 377 |
| <b>Table 337: Parameter</b>                       | 377 |
| <b>Table 338: Memory Area for Code and Data</b>   | 378 |
| <b>Table 339: Parameter</b>                       | 378 |
| <b>Table 340: Parameter</b>                       | 379 |
| <b>Table 341: Parameter</b>                       | 380 |
| <b>Table 342: Parameter</b>                       | 380 |
| <b>Table 343: Parameter</b>                       | 381 |
| <b>Table 344: Return Value</b>                    | 381 |
| <b>Table 345: Commands in Cygwin (Bash Shell)</b> | 384 |
| <b>Table 346: Output Variables (CPU-&gt;COM)</b>  | 394 |
| <b>Table 347: Input Variables(COM-&gt;CPU)</b>    | 394 |

## Index

|                       |    |                           |    |
|-----------------------|----|---------------------------|----|
| activation code ..... | 22 | dual configuration .....  | 85 |
| license .....         | 22 | reload .....              | 85 |
| part number           |    | signature .....           | 85 |
| HIMatrix .....        | 32 | version state .....       | 90 |
| HIMax .....           | 32 | standard protocols        |    |
| safeethernet          |    | process data volume ..... | 19 |

HI 801 101 E  
© 2013 HIMA Paul Hildebrandt GmbH  
® = registered trademark of:  
HIMA Paul Hildebrandt GmbH

HIMA Paul Hildebrandt GmbH  
Albert-Bassermann-Str. 28 | 68782 Brühl, Germany  
Phone +49 6202 709-0 | Telefax +49 6202 709-107  
[info@hima.com](mailto:info@hima.com) | [www.hima.com](http://www.hima.com)



SAFETY  
NONSTOP



For a detailed list of all our subsidiaries and representatives,  
please visit our website: [www.hima.com/contact](http://www.hima.com/contact)

