SEW EURODRIVE

System Manual

Decentralized MOVISAFE® HM31 Safety Controller

Edition 07/2014

21233799/EN

Contents

21233799/EN – 07/2014

21233799/EN – 07/2014

21233799/EN – 07/2014

21233799/EN – 07/2014

21233799/EN – 07/2014

21233799/EN – 07/2014

1 General information

This manual contains information on the designated use of the safety controller.

The following conditions are required for safe installation, startup and safety during operation and maintenance:

- Knowledge of regulations
- Proper technical implementation of the safety instructions in this manual through qualified personnel.

Under the following circumstances, disruption or impairment of safety functions can cause severe injury to persons, damage to property or damage to the environment, for which SEW-EURODRIVE cannot assume liability:

- Unskilled access to the units
- De-activating or bypassing safety functions
- Non-observance of instructions in this manual

SEW-EURODRIVE develops, manufactures and tests safety controllers in compliance with the pertinent safety standards. The units may only be used if the following requirements have been met:

- They are only used for the intended applications
- They are only operated under the specified environmental conditions
- They are only operated in conjunction with approved non-SEW units

1.1 Documentation design and use

This system manual covers the following topics:

- General information
- System features
- Communication
- Safeethernet
- PROFINET IO
- Modbus TCP/UDP
- Send and receive TCP
- SNTP protocol
- Com user task (CUT)
- Operating system
- User program
- Startup
- Operation
- Maintenance

The manual describes the following variant:

Programming tool	Processor operating system	Communication operating system
SILworX®	CPU-BS V.8 and later	COM-BS V.13 and later

21233799/EN – 07/2014

1.2 Target group

This document was written for planners, project planners, and programmers of automation systems as well as for persons authorized to start up, operate, and service the units and systems. Specific knowledge of safety-related automation systems is required.

1.3 Text conventions

The following notation is used in this document to enhance readability and comprehensibility:

Notation	Meaning
Bold	To highlight important text.
[...]	Names of buttons and menu commands that you can click in the programming tool.
Italics	Parameters and system variables.
`Courier`	Actual user entries.
RUN	Names of operating states in capital letters.

1.4 Structure of the safety notes

1.4.1 Meaning of signal words

The following table shows the grading and meaning of the signal words for safety notes.

Signal word	Meaning	Consequences if disregarded
⚠ **DANGER**	Imminent hazard	Severe or fatal injuries
⚠ **WARNING**	Possible dangerous situation	Severe or fatal injuries
⚠ **CAUTION**	Possible dangerous situation	Minor injuries
NOTICE	Possible damage to property	Damage to the drive system or its environment
INFORMATION	Useful information or tip: Simplifies handling of the drive system.	

1.4.2 Structure of section-related safety notes

Section-related safety notes do not apply to a specific action but to several actions pertaining to one subject. The hazard symbols used either indicate a general hazard or a specific hazard.

This is the formal structure of a safety note for a specific section:

SIGNAL WORD

Type and source of hazard.

Possible consequence(s) if disregarded.

- Measure(s) to prevent hazard.

21233799/EN – 07/2014

1.4.3 Structure of embedded safety notes

Embedded safety notes are directly integrated into the instructions just before the description of the dangerous action.

This is the formal structure of an embedded safety note:

- ⚠ **SIGNAL WORD** Type and source of hazard.

  Possible consequence(s) if disregarded.

  – Measure(s) to prevent hazard.

1.5 Rights to claim under limited warranty

A requirement of fault-free operation and fulfillment of any rights to claim under limited warranty is that you adhere to the information in the documentation. Read the documentation before you start working with the unit.

1.6 Exclusion of liability

You must comply with the information contained in this documentation to ensure safe operation and to achieve the specified product characteristics and performance features. SEW-EURODRIVE assumes no liability for injury to persons or damage to equipment or property resulting from non-observance of these operating instructions. In such cases, any liability for defects is excluded.

1.7 Other applicable documentation

Observe the following applicable documents:

- "Decentralized Safety Controller MOVISAFE® HM31" operating instructions
- "Decentralized Safety Controller MOVISAFE® HM31" safety manual
- Drive Engineering – Practical Implementation, Electromagnetic Compatibility (EMC) in Drive Engineering

If you want to use the Com user task (CUT) function, also observe the following applicable documents:

- "Com User Task for MOVISAFE® HM31" manual
- "MOVIVISION® Parameter and Diagnostics Tool Version 2.0" manual

You require software that is **not** included in the delivery. You can order the software together with the documentation on a data storage medium (CD/DVD) from SEW-EURODRIVE using the following order information:

Designation	Part number
SILworX® for MOVISAFE® HM31 • Hardware: SILworX® license dongle • Software: SILworX® 4.64.0 or later	19500114
Motion Library MOVISAFE® HM31 Function block library for safety-related position detection	17106400

Also observe the applicable documentation for the connected drive technology.

21233799/EN – 07/2014

The latest documentation versions are available for download from the SEW website (www.sew-eurodrive.com) under "Documentation".

1.8 Copyright

1.9 Product names and trademarks

The brands and product names in this documentation are trademarks or registered trademarks of their respective titleholders.

21233799/EN – 07/2014

2 System properties

The safety controller includes a safety-related processor system, a number of inputs and outputs, as well as communication connections in a single housing unit.

For details, refer to the "Decentralized Safety Controller MOVISAFE® HM31" operating instructions.

2.1 Monitoring the operating voltage

The device monitors the DC 24 V voltage during operation. The devices respond according to the voltage levels listed in the table.

Voltage level	Response of the devices
DC 24 V −20% / +25% (19.2 V – 30 V)	Normal operation
< 18.0 V (circuit board voltage read by the software)	Alarm status (internal variables are described and sent to inputs and outputs)
< 12.0 V (circuit board voltage read by the software)	Inputs and outputs are turned off

The *power supply status* system variable enables you to analyze the operating voltage status using the programming tool, or in the user program.

2.2 Monitoring the temperature status

The temperature is measured by sensors at relevant locations inside the unit or system and is output by the software.

It has a delta amount relative to the ambient temperature; this amount depends on many factors.

If the temperature measured inside the unit exceeds the defined switching points, the value of the "temperature status" system variable changes as follows:

Temperature (inside the unit)	Temperature range	Temperature status [BYTE]
<60 °C	Normal	0x00
60 °C to 70 °C	High temperature	0x01
> 70 °C	Very high temperature	0x03
Return to 64 °C to 54 °C[1)]	High temperature	0x01
Return to < 54 °C[1)]	Normal	0x00

1) The sensors have a hysteresis of 6 °C.

If there is little or no air circulation and natural convection is insufficient, the threshold for the “high temperature” range in the safety controller can be tripped at ambient temperatures of < 35 °C. This could be caused by local warming or poor heat dissipation. Especially for digital outputs, warming depends to a large extent on the load. The

21233799/EN – 07/2014

"temperature status" system variable enables users to read the internal temperature. If the condition "very high temperature" occurs frequently, SEW-EURODRIVE recommends that you improve the heat dissipation of the system, for example through additional cooling or ventilation to ensure a long service life of the safety controller.

INFORMATION

Transition to "high temperature" or "very high temperature" status does not mean that the safety of the system is impaired.

2.2.1 Setting a temperature threshold for messages

You can set a threshold for the safety controller to define a value at which a message will be issued to indicate that the temperature is exceeded. You set this parameter in the SILworX® Hardware Editor in "Detailed view of the controller".

2.2.2 Short-circuit behavior of output channels

In the event of a short circuit in an output channel, the safety controller disables the affected channel. If several short circuits occur, the safety controller disables the channels individually according to the current consumption. If the maximum permitted total current is exceeded for all outputs, then all outputs are disabled and are switched on again cyclically.

INFORMATION

Do not insert the plug connectors for output circuits while load is connected. The reason is that the short-circuit current can damage the terminals in the event of a short circuit.

2.3 Alarm and event recording

The safety controller can record alarms and events (sequence of events recording, SER).

2.3.1 Alarms and events

Events are changes in the status of the system or controller that are marked with a time stamp.

Alarms are events that signal an increased risk potential.

The safety controller records changes in status as events, including the time when they occurred.

The safety controller distinguishes between Boolean and scalar events.

Boolean events:

- Changes in Boolean variables, such as of digital inputs.
- Alarm and normal status; these states can be freely assigned to the variable states.

Scalar events:

- Exceeding of limits that have been defined for a scalar variable.
- Scalar variables have a numerical data type, such as INT, REAL.

21233799/EN – 07/2014

- Two upper and two lower limits can be set.
- The following must apply to the limits:

  Maximum limit = upper limit = normal range = lower limit = lowest limit.
- A hysteresis can take effect in the following cases:
  - If a value falls below an upper limit.
  - If a value exceeds a lower limit.

Entering a hysteresis prevents an unnecessarily large number of events if the global variable fluctuates a great deal around a limit.

The safety controller can only create events if they have been defined in SILworX®, see the "Configuring alarms and events" chapter. Up to 4000 alarms and events can be defined.

2.3.2 Creating events

The processor system can create events. The processor system creates the events from global variables and stores them in the buffer; see "Event recording". Events are created in the user program cycle. Each event that has been read can be overwritten by a new event.

System events:

In addition to events that register changes in global variables or input signals, the processor systems create the following types of system events:

- Overflow: Events have not been stored because of a buffer overflow. The time stamp of the overflow event is the time when the event that caused the overflow occurred.
- Init: The event buffer has been initialized.

System events include the SRS identification of the unit that caused the event. Status variables make the event status of scalar events available to the user program. A BOOLEAN-type global variable can be assigned to each of the following states as a status variable:

- Normal.
- Fell below lower limit.
- Fell below lowest limit. Upper limit exceeded.
- Maximum limit exceeded.

The assigned status variable becomes TRUE when the corresponding status is reached.

2.3.3 Event recording

The processor system collects the events: The processor system stores all the events in its buffer. The buffer is located in the non-volatile memory and holds 1000 events. If the buffer is full, no new events are saved until more events have been read, and as a result marked for overwriting.

2.3.4 Transferring events

The events can be transferred to the drive control (Beck PC) via the MODBUS protocol, or to the higher-level safety controller via safeethernet. For this purpose, the corresponding variables must have been previously linked in the user program. Advanced diagnostics is performed via the PADT (SILworX®).

21233799/EN – 07/2014

3 Communication

The safety controller communicates using the following protocols:

- safeethernet

  Safety-related protocol for controllers communicating with each other
- Modbus TCP/UDP fieldbus protocol for connecting external units or systems
- Communication with the programming unit

The communication system is connected to the safety-related processor system.

The communication system and the fieldbus interfaces are connected to the safe microprocessor system via dual-port RAM. Only units that ensure safe electrical separation may be connected to the interfaces.

The communication system controls the controller's communication with other systems via high-performance interfaces:

Available protocols

The following protocols are available for the MOVISAFE® HM31 safety controller. These protocols can be activated as follows:

Protocol	Interfaces	Activation
safeethernet	Ethernet	The function is activated by default with MOVISAFE® HM31.
SNTP server/client		
Modbus TCP/UDP master		
COM user task	CAN (X4111_1/2) RS485 (X4011)	
OPC server (runs on host PC)	Ethernet	On request, SEW-EURODRIVE generates a license that can be used to activate the protocol.
PROFINET IO Controller		
PROFINET IO Device		
Modbus TCP/UDP slave		
TCP send/receive		
PROFIsafe host		
PROFIsafe device		

INFORMATION

You can use the optional protocols in MOVISAFE® HM31 without activation for testing purposes for 5000 operating hours. The "ERROR" system LED lights up red continuously when protocols that were not enabled are used.

After the 5000 operating hours have elapsed, the controller no longer starts.

- Therefore, order the license for enabling the required functions in time.

3.1 Ethernet

The safety controller features an Ethernet switch with connections. Other units can be connected to the controller via these connections using Ethernet cables.

21233799/EN – 07/2014

The following interfaces are available:

- **2 Ethernet interfaces:** X4233_1 and X4233_2

  Both interfaces are located on the terminal strip of the unit.

- **1 Ethernet service interface:** X4223

  For connecting a programming unit (PADT).

Ethernet switch

- Unlike a hub, an Ethernet switch can analyze data packets and store them temporarily. The switch then establishes a temporary dedicated connection between two communication partners (sender/receiver) to send the data. The purpose is to avoid collisions that often occur with hubs, and to reduce the load on the network. Each switch requires an address/port assignment table to know where to send the data. This table is automatically generated by the switch in a self-learning process. In the table, MAC addresses are assigned to a specific port on the Ethernet switch. Incoming data packets are sent directly to the corresponding port based on this table.
- The Ethernet switch automatically switches between 10 and 100 MBit/s transmission rates, and between full duplex and half-duplex connections. This means the full bandwidth is available to each data transfer direction (full duplex operation).
- An Ethernet switch controls the communication between two different communication terminals. The Ethernet switch can address up to 1000 absolute MAC addresses.
- Auto crossing detects the connection of cables with crossed wires, and the switch automatically adjusts to this situation.

INFORMATION

Observe the notes in the safety manual when configuring the safety-related communication.

3.2 Communication with the programming tool

The safety controller communicates with a PADT via Ethernet. A PADT is a PC or laptop on which the SILworX® programming tool is installed.

The controller can communicate with up to 5 PADTs at a time. However, only one programming tool can write to the controller. All other PADTs can only read information. The controller provides read-only access to any other PADT attempting to write.

21233799/EN – 07/2014

3.3 Registering and activating protocols

The following protocols are available for the MOVISAFE® HM31 safety controller. These protocols can be activated as follows:

Protocol	Interfaces	Activation
safeethernet	Ethernet	The function is activated by default for MOVISAFE® HM31.
SNTP server/client		
Modbus TCP/UDP master		
COM user task	CAN (X4111_1/2) RS485 (X4011)	
OPC server (runs on host PC)	Ethernet	Upon request to SEW-EURODRIVE
PROFINET IO Controller		
PROFINET IO Device		
Modbus TCP/UDP slave		
TCP send/receive		
PROFIsafe host		
PROFIsafe device		

Do the following to enter the software activation code in SILworX®:

1. From the structure tree, choose [Configuration] / [Resource] / [License Management].
2. Select the entry "License Management" and open the context menu. Form the context menu, choose [New] / [License Key]. A new license key is added.
3. Select "License Key" and open the context menu. From the context menu, choose [Properties].
4. In the "Activation Code" field, enter the generated software activation code.

INFORMATION

The license is linked with this specific system ID. This means you can use a license only once for a certain system ID. Therefore, do not activate the software until you are not absolutely sure about the system ID. A software activation code can include a maximum of 32 licenses. In the License Management, you can enter more than one activation code. A maximum of 64 licenses can be loaded to a controller.

INFORMATION

With unit option PFF-HM31A1-E61-I111-00/000/000, you can use the Ethernet protocols without activation for testing purposes for 5000 operating hours. The "ERROR" system LED lights up red continuously when protocols that were not enabled are used.

After the 5000 operating hours have elapsed, the controller no longer starts.

- Therefore, order the unit option with the required protocols in time.

21233799 / EN – 07/2014

3.4 Load limitation

You can specify a computing time budget in % (µP budget) for each communication protocol. The available computing time can in this way be distributed among the configured protocols. The total amount of the computing time budgets of all configured communication protocols of a CPU module or COM module must not exceed 100%.

The computing time budgets specified for the individual communication protocols are monitored. If a communication protocol has reached or exceeded its computing time budget and if no additional computing time is available as a reserve, then the communication protocol is not processed completely.

If there is still sufficient additional computing time available, then this reserve is used to finish processing a communication protocol that has reached or exceeded its computing time budget. In this case, a communication protocol uses a greater computing time budget than assigned to it.

More than 100% computing time budget might be displayed online. This is not an error but indicates that the amount above the 100% is the computing time used in addition to the assigned one.

INFORMATION

The additional computing time budget in no way is reliable for a specific communication protocol and can be revoked by the system at any time.

21233799 / EN – 07/2014

4 safeethernet

The safety controller is safeethernet-capable. It can communicate via Ethernet (100 Mbit/s) in a safety-related manner according to SIL 3.

The Ethernet interfaces of the safety controller can also be used simultaneously for other protocols.

The safeethernet communication between the controllers can be implemented with various Ethernet network topologies. Adjust the parameters of the safeethernet protocol to the Ethernet network used to increase the speed and efficiency of data transfer.

You can set these parameters using so-called network profiles. The factory setting of the parameters ensures communication so that the user does not have to learn network configuration details right away.

INFORMATION

The safeethernet protocol is safety-related and TÜV-certified up to SIL 3 according to IEC 61508.

safeethernet properties

Element	Features	Description
Required module/ controller	Controller's integrated processor module	safeethernet is run on the safety-related processor module.
Ethernet interfaces	100 Mbit/s	The Ethernet interfaces used can also be used simultaneously for other protocols.
Connections	128	safeethernet connections
Redundant connections	128	2-channel operation. Redundant safeethernet connections between controllers can be set in safeethernet editor.
Redundant transmission paths	Limitation because only one unit	Redundant safeethernet transmission paths.
Amount of process data per connection	1100 bytes	Per safeethernet connection.

INFORMATION

For safeethernet connections with resources outside a project, an Export function is available (see chapter "Communication across projects").

21233799/EN – 07/2014

4.1 What is safeethernet?

Requirements such as determinism, reliability, interchangeability, expandability and, above all, safety, are key issues in the process and automation technology sector.

safeethernet is a transmission protocol for transmitting safety-related data up to SIL 3 using Ethernet technology.

safeethernet includes mechanisms that recognize the following errors and cause a safety-related response to them:

- Corruption of transmitted data (duplicate, lost, modified bits)
- Incorrect message addressing (sender, receiver)
- Incorrect data sequence (repetition, loss, wrong sequence)
- Incorrect time behavior (delay, echo)

safeethernet is based on the IEEE 802.3 standard.

safeethernet uses "unsafe data transfer channels" (Ethernet) in accordance with the black channel principle, and monitors them at the sender's and receiver's ends through safety-related protocol mechanisms. This makes it possible to use Ethernet network components such as hubs, switches and routers within a safety-related network.

safeethernet uses the functions of standard Ethernet in a way that enables safety and real-time capability. A special protocol mechanism guarantees deterministic behavior even when communication stations are dropped or enter the network. The system then automatically integrates new components while it is running. All network components can be exchanged during runtime. Transmission times can be clearly defined using switches. This enables Ethernet's real-time capability.

Connections to the company's internal Intranet, as well as connections to the Internet, are possible with safeethernet. This way, only one network is required for both safe and non-safe data transmission.

INFORMATION

The network can also be used by other participants if sufficient transmission capacity is available.

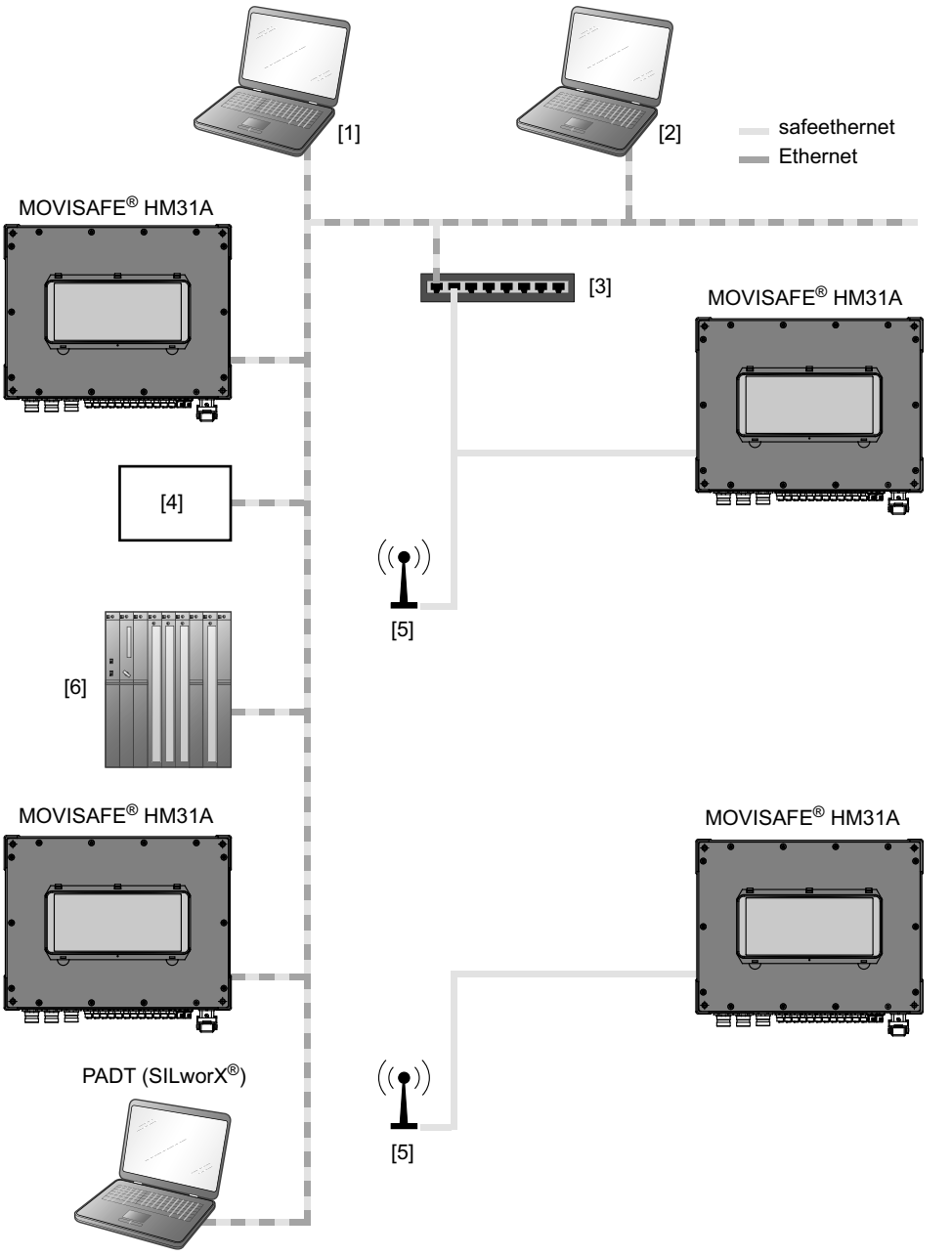
⚠ WARNING

Manipulation of safety-related data transfer.

Severe or fatal injuries.

21233799/EN – 07/2014

safeethernet enables flexible system structures for decentralized automation with specified response times. Depending on your requirements, you can choose to have central or decentralized distribution of intelligence to participants within the network.



[1] PC for the DCS control system
[2] PADT (SILworX®)
[3] Network switch
[4] DCS control system
[5] Radio, satellite, WLAN, fiber optics, ISDN or DSL
[6] PLC

21233799/EN – 07/2014

INFORMATION

Unintentional change to safe state is possible.

- Take care that no network loops are created during interconnection. Data packets can reach the controller only on **one** path.
- Use only managed switches when setting up an Ethernet ring topology.

4.2 safeethernet editor

You can create and configure the safeethernet connections to communication partners (resources) in the safeethernet editor.

Do the following to open the safeethernet editor of the local resource:

1. Open [Configuration]/[Resource] in the structure tree.
2. Right-click on "safeethernet" and choose [Edit] from the context menu.

The safeethernet editor contains the work area and the selection of objects.

You can create and configure the safeethernet connections to communication partners (resources) in the safeethernet editor. To do so, drag the resources from the selection of objects into the work area.

To configure the safeethernet connection, set the following safeethernet protocol parameters:

Parameter	Description
Partner	Resource name of the link partner.
IF CH...	Available Ethernet interfaces on the resource (local) and resource (target).
Profile	Combination of matching safeethernet parameters, see also chapter "Safeethernet profiles".
Response time [ms]	Time until receipt of a message is acknowledged at the sender; see also chapter "Response time".
Receive timeout [ms]	Monitoring time on PES1 during which a correct answer must be received from PES2, see also chapter "Receive Timeout".
Resend timeout [ms]	Monitoring time on PES1 during which PES2 must acknowledge receipt of a data packet. Else, the data packet is resent, see also chapter "Resend Timeout".
Acknowledge timeout [ms]	Interval during which a received data packet must be acknowledged by the CPU; see also chapter "Acknowledge timeout".
Prod. rate	Production rate: Smallest interval between two data packets; see also chapter "Production rate".
Memory (queue depth)	Number of data packets that can be sent without acknowledgement; see also chapter "Memory".

21233799/EN – 07/2014

Parameter	Description
Freeze data in case of dropped connection [ms]	Behavior of the input variables of this safeethernet connection when the connection is disrupted[1]. • Use initial data: The initial data is used for the input variables. Unlimited input variables are frozen at their current values and used until the connection is re-established. • Limited Entry: Double-click the drop-down menu and enter the time. The input variables are frozen at their current values and used until after the timeout parameter set. Afterwards the initial data is used. The timeout can be extended by up to one CPU cycle.
Fragments per cycle	Fixed setting: One fragment per controller cycle is sent to the communication partner. Fragment ≤ 900 bytes
Event priority	Function is not supported.
Priority of status values	
Number of ignored warnings	This is the number of warnings that must occur consecutively during the "Warning period [ms]" time span before the warnings are entered into diagnostics or the communication error statistics.
Warning period [ms]	0 ms is currently the only permitted value.
Activate SER	Default: disabled

1) Observe the following warning note:

⚠ WARNING

Behavior of input variables with disrupted connection.

Severe or fatal injuries.

Object selection

Object selection provides all resources within this project that the resource can connect to via safeethernet.

INFORMATION

For safeethernet connections with resources outside a project, an Export function is available (see chapter "Communication across projects").

21233799/EN – 07/2014

4.3 Detail view of the safeethernet editor

The detailed view always refers to the local resource for which you started the safeethernet editor.

Do the following to open the detailed view of a safeethernet connection:

1. Open the context menu by right-clicking on [safeethernet connection].
2. Click [Detailed view].

   The detailed view includes the system variables tab, fragment definitions, and resource (local) ↔ resource (target).

4.3.1 Register: System variables

You can control the safeethernet connection and analyze its status using system variables in the user program.

<table>
<tr><th>System variable</th><th colspan="2">Description</th></tr>
<tr><td>Ack. frame no.</td><td colspan="2">Receipt counter (revolving).</td></tr>
<tr><td>Number of faulty messages</td><td colspan="2" rowspan="2">Number of faulty messages per channel (incorrect CRC, incorrect header, other errors).</td></tr>
<tr><td>Number of faulty messages in the red. Channel</td></tr>
<tr><td>Number of successful connections</td><td colspan="2">Number of successful connections since the statistics were reset.</td></tr>
<tr><td>Number of lost messages</td><td colspan="2" rowspan="2">Number of messages lost on one of the two transmission paths since the statistics were reset. The counter is only run until a channel fails completely.</td></tr>
<tr><td>Number of lost messages in the red. channel</td></tr>
<tr><td>Early queue usage</td><td colspan="2">Number of messages that were placed into the early queue since the statistics were reset; see also chapter "Memory".</td></tr>
<tr><td>Faulty messages</td><td colspan="2">Number of messages rejected since the statistics were reset.</td></tr>
<tr><td>Frame no.</td><td colspan="2">Sent counter (revolving).</td></tr>
<tr><td rowspan="5">Channel status</td><td colspan="2">Current status of channel 1. The channel status is the current status of channel 1 at the time (seq. no. X-1) a message with seq. no. X was received.</td></tr>
<tr><td>Status</td><td>Description</td></tr>
<tr><td>0</td><td>No message regarding the status of channel 1.</td></tr>
<tr><td>1</td><td>Channel 1 OK.</td></tr>
<tr><td>2</td><td>Last message was faulty; current one is OK.</td></tr>
<tr><td></td><td>3</td><td>Error on channel 1.</td></tr>
<tr><td>Layout version</td><td colspan="2">Signature of the data layout used in the communication.</td></tr>
</table>

21233799/EN – 07/2014

<table>
<tr><th>System variable</th><th colspan="3">Description</th></tr>
<tr><td>Last channel latency</td><td colspan="3" rowspan="8">The channel latency specifies the delay between the two redundant transmission paths at the time when messages with identical SeqNo were received. Statistics of the average, minimum, maximum and most recent latencies are kept for this purpose.
If the min. value > max value, the statistical values are not valid. The most recent channel latency and the average channel latency are then 0.</td></tr>
<tr><td>Most recent latency of the red. channel</td></tr>
<tr><td>Max. channel latency</td></tr>
<tr><td>Max. channel latency of the red. Channel</td></tr>
<tr><td>Min. channel latency</td></tr>
<tr><td>Min. channel latency of the red. Channel</td></tr>
<tr><td>Average channel latency</td></tr>
<tr><td>Average channel latency of the red. Channel</td></tr>
<tr><td>Monoticity</td><td colspan="3">User data sent counter (revolving).</td></tr>
<tr><td>New layout version</td><td colspan="3">Signature of the new data layout.</td></tr>
<tr><td rowspan="6">Quality of channel 1</td><td colspan="3">Status of the main transmission path.</td></tr>
<tr><td>Bit no.</td><td>Bit = 0</td><td>Bit = 1</td></tr>
<tr><td>0</td><td>Transmission path not activated</td><td>Transmission path activated</td></tr>
<tr><td>1</td><td>Transmission path not used</td><td>Transmission path used actively</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 delivers message first</td></tr>
<tr><td>4-7</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>Quality of channel 2</td><td colspan="3">Status of the redundant transmission path; see status of channel 1 (main transmission path).</td></tr>
<tr><td>Receive timeout</td><td colspan="3">Time in milliseconds (ms) on PES1 during which a correct answer must be received from PES2, see also chapter "Receive Timeout".</td></tr>
<tr><td>Response time</td><td colspan="3">Time in milliseconds (ms) until receipt of a message is acknowledged at the sender; see also chapter "Response time".</td></tr>
<tr><td rowspan="3">Reset safeethernet statistics</td><td colspan="3">Reset statistical values for the communications connection in the user program (such as number of faulty messages, channel status, timestamp of the most recent error in the red. channel, retries).</td></tr>
<tr><td>Value</td><td colspan="2">Function</td></tr>
<tr><td>0</td><td colspan="2">No reset</td></tr>
<tr><td>1-255</td><td colspan="2">Reset of the safeethernet statistics</td></tr>
<tr><td rowspan="3">Transmission control channel 1</td><td colspan="3">Transmission control of channel 1.</td></tr>
<tr><td>Bit 0</td><td colspan="2">Function</td></tr>
<tr><td>FALSE</td><td colspan="2">Transmission path activated for tests</td></tr>
</table>

21233799/EN – 07/2014

System variable	Description	
	TRUE	Transmission path blocked
	Bits 2 to 7 reserved.	
Transmission control channel 2	See transmission control channel 1.	
Connection control	The user program can control the safeethernet connection using these system variables.	
	Command	**Description**
	Auto connect (0x0000)	Default: After safeethernet communication has been lost, the controller attempts to re-establish connection during the next CPU cycle.
	Toggle mode 0 (0x0100) Toggle mode 1 (0x0101)	After communication has been lost, a program-controlled change in toggle mode can re-establish the connection. • TOGGLE MODE_0 (0x100) set: Set to TOGGLE MODE 1 (0x101) to re-establish the connection. • TOGGLE MODE 1 (0x101) set: Set to TOGGLE_MODE_0 (0x100) to re-establish the connection.
	Disabled (0x8000)	safeethernet connection turned off.
Connection status	The connection status analyzes the status of the communication between two controllers in the user program.	
	Status/value	**Description**
	Closed (0)	Connection is closed and no attempt is made to open it.
	Try_open (1)	An attempt is being made to open the connection, but it is not yet open. This status applies to both the active and the passive side.
	Connected (2)	The connection has been established and is in operation (active time monitoring and data exchange).
Retries	Number of retries since the statistics were reset.	
Timestamp of the last error in the red. channel [ms]	Millisecond portion of the timestamp (current system time).	
Timestamp of the last error in the red. channel [s]	Seconds portion of the timestamp (current system time).	
Timestamp of the last error [ms]	Millisecond portion of the timestamp (current system time).	
Timestamp of the last error [s]	Seconds portion of the timestamp (current system time).	
Status of the red. channel	Current status of channel 2. The channel status is the current status of channel 2 at the time (seq. no. X-1) a message with seq. no. X was received.	
	Status	**Description**
	0	No message regarding the status of channel 2

21233799/EN – 07/2014

System variable	Description	
	1	Channel 2 OK
	2	Last message was faulty; current one is OK.
	3	Error on channel 2.

4.4 safeethernet parameters

You can set up safety-related communication in the safeethernet editor. To do so, you must set the parameters described in this chapter. The following condition applies to calculating the *Receive Timeout* and *Response Time* safeethernet parameters: The communication time slice must be large enough to process all safeethernet connections during one CPU cycle; see chapter "Maximum communications slice".

4.4.1 Maximum cycle time of the safety controller

SEW-EURODRIVE recommends the following method for determining the maximum cycle time of a MOVISAFE® HM31 safety controller.

Determining the maximum cycle time of the MOVISAFE® HM31 safety controller:

1. Operate the system under maximum load. All communication connections must be operating via safe**ethernet** and via standard protocols. Frequently read the cycle time from the control panel and note the maximum cycle time.
2. Repeat step 1 for the communication partner (second safety controller).
3. The required maximum cycle time is the greater of the two determined maximum cycle times.

The maximum cycle time has now been determined and is used in the calculations below.

4.4.2 Receive timeout

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

If the communication partner does not receive a valid response during the *ReceiveTMO*, safety-related communication is terminated. The input variables of this safe**ethernet** connection behave according to the setting made for *Freeze data on lost connection [ms]*.

For safety-related functions which are implemented via safe**ethernet,** only the *Use initial data* setting may be used.

Because *ReceiveTMO* is a safety-relevant component of the worst case response time T_R (for maximum response time, see safety manual, chapter 8.2.4), the *ReceiveTMO* must be calculated as described below and entered in the safe**ethernet** Editor:

ReceiveTMO ≥ 4 × delay + 5 × max. cycle time

Condition: The communication time slice must be large enough to process all safe**ethernet** connections during one CPU cycle.

Delay: Delay on the transmission path, for instance due to switch or satellite

Max. cycle time: Maximum cycle time of both controllers

21233799/EN – 07/2014

INFORMATION

- A desired error tolerance of the communication can be achieved by increasing *Receive TMO* provided this is permitted for the application process in terms of time.
- The maximum value permitted for *ReceiveTMO* depends on the application process and is set in the safe**ethernet** Editor together with the maximum expected *response time* and the profile.

4.4.3 Response time

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

To set parameters using a safe**ethernet** profile, you have to specify an expected *ResponseTime* based on the physical conditions of the transmission path.

The specified *ResponseTime* affects the configuration of all parameters for the safe**ethernet** connection. These parameters must be calculated as follows:

ResponseTime ≤ ReceiveTMO / n

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

The ratio of *ReceiveTMO* to *ResponseTime* affects the capability to tolerate faults, for instance if packets are lost (resending lost data packets) or there are delays along the transmission path.

In a network where packets could be lost, the following condition must be met:

Min. response time ≤ receiveTMO / 2 ≥ 2 × delay + 2.5 × max. cycle time

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

INFORMATION

- If this condition has not been met, the availability of a safeethernet connection can only be guaranteed in a network free of collisions and interference. However, this does not mean that the processor module has a safety problem.
- Make sure that the communication system complies with the set *ResponseTime*. In cases where this cannot always be guaranteed, a corresponding system variable for the connection is available to monitor the *ResponseTime*. If the *ResponseTime* measured is exceeded by half of the *ReceiveTMO* more often than in exceptional cases, then the *ResponseTime* set must be increased. The *Receive Timeout* must be adjusted to the newly set *ResponseTime*.
- In the examples below, the formulas for calculating the maximum response time for a connection with the safety controller only apply if the safety time for these has been set to = 2 × watchdog time.

4.4.4 Sync/Async

Sync: Currently not supported.

Async: Is the default setting. When the parameter is set to Async, the safeethernet protocol instance receives during the CPU's input phase and sends according to its sending rules during the CPU's output phase.

21233799/EN – 07/2014

4.4.5 Resend timeout

ResendTMO cannot be entered manually but is calculated from the profile and the response time. Monitoring time in milliseconds (ms) on PES1 during which PES2 must acknowledge receipt of a data packet; otherwise the data packet is resent.

Rule: *ResendTMO* ≤ *ReceiveTimeout*

If resend timeouts have been configured differently for the communication partners, then the active protocol partner (lower SRS) determines the actual value of the protocol connection's resend timeout.

4.4.6 Acknowledge timeout

AckTMO cannot be entered manually but is calculated from the profile and the *ResponseTime*. *AckTMO* is the latest time after which a received data packet must be acknowledged by the CPU.

In a fast network *AckTMO* is zero, which means the receipt of a data packet is acknowledged immediately. In a slow network (such as a telephone modem path), *AckTMO* is greater than zero. In this case, the system tries to send the acknowledgement along with the process data to reduce the load on the network by avoiding addressing and safety blocks.

Rules:

AckTMO must be ≤ *ReceiveTimeout*

AckTMO must be ≤ *ResendTimeout*, if *ProductionRate* is > *ResendTimeout*.

4.4.7 Production rate

ProdRate cannot be entered manually, but is calculated from the profile and the *ResponseTime*.

Smallest interval in milliseconds (ms) between two data packets.

The goal of *ProdRate* is to limit the number of data packets to a quantity that does not overload a (slow) communication channel. The purpose is to achieve a consistent utilization of the transmission medium and prevents that old data is received at the receiving end.

Rules:

- ProdRate ≤ *ReceiveTimeout*
- ProdRate ≤ *ResendTimeout*, if *AcknowledgeTimeout* > *ResendTimeout*

INFORMATION

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

4.4.8 Memory

Memory cannot be entered manually but is calculated from the profile and the response time.

Memory (queue depth) is the number of data packets that can be sent without having to wait for acknowledgement.

The value depends on the network's transmission capacity and possible delays due to network runtimes.

21233799/EN – 07/2014

The message memory available in the CPU is divided among all safeethernet connections.

4.5 Maximum response time for safeethernet

In the examples below, the formulas for calculating the maximum response time apply only if the safety time has been set to = 2x watchdog time.

INFORMATION

The maximum permitted response time depends on the process and must be coordinated with the relevant inspection authorities.

Term	Meaning
ReceiveTMO	Monitoring time in PES1 during which a valid answer must be received from PES2. When this time has elapsed, safety-related communication is terminated.
Production rate	Minimum interval between two data transmissions.
Watchdog time	Maximum permitted duration of a RUN cycle of a controller. The duration of the RUN cycle depends on the complexity of the user program and the number of safeethernet connections. Watchdog time (WDT) must be entered in the properties of the resource.
Worst case reaction time	Maximum response time for sending the signal change of a physical input (In) of a PES 1 until the change of the physical output (Out) of a PES 2.
Delay	Delay in a transmission path, for example for a modem or satellite connection. For a direct connection, a delay of 2 ms can initially be assumed. The actual delay on a transmission path can be measured by the network administrator in charge.

The following conditions apply to the calculation of maximum allowable response times below:

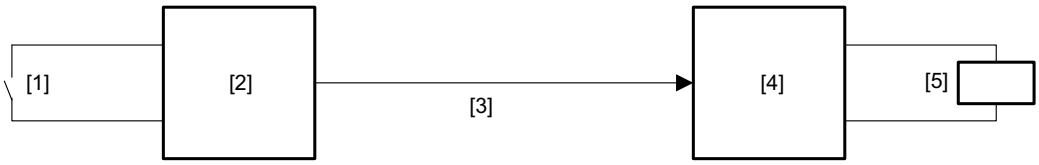
- The variables that are sent via safeethernet must be processed in the respective controllers within one CPU cycle.
- The response times for the sensors and actuators must also be added.

The calculations also apply to signals in the opposite direction.

21233799/EN – 07/2014

4.5.1 Calculating the maximum response time

The maximum response time T_R (worst case) from the change of an input in PES 1 until the response of the output in PES 2 can be calculated as follows:



4784751883

[1] Input

[2] Safety controller PES 1

[3] Safety-related protocol

[4] Safety controller PES 2

[5] Output

$T_R = t_1 + t_2 + t_3$

T_R Worst case reaction time

t_1 2 × watchdog time of safety controller 1

t_2 ReceiveTMO

t_3 2 × watchdog time of safety controller 2

The maximum response time depends on the process and has to be agreed with the test authority responsible for final acceptance.

4.5.2 safeethernet profiles

safeethernet profiles are combinations of matching parameters that are set automatically when a safeethernet profile is selected. To set the parameters, you only have to individually configure the *Receive timeout* and the expected *Response time*.

The goal of a safeethernet profile is to optimize data throughput in the network while taking into account physical conditions.

The following conditions are required for effective optimization:

- The communication time slice must be large enough to process all safeethernet connections during one CPU cycle.
- Mean CPU cycle time < *Response time*
- Mean CPU cycle time < *ProdRate* or *ProdRate* = 0

NOTICE

Improper combinations of CPU cycle, communication time slice, *Response time* and *ProdRate* are not rejected during code generation and during download/reload. However, these combinations can cause faults, including failure of the safeethernet communication.

Possible damage to the drive system.

- Check the "**Faulty messages**" and "**Retries**" displays in the control panels of both controllers.

The safeethernet profile appropriate for this transmission path can be selected from 6 available safeethernet profiles. Note the following warning:

21233799/EN – 07/2014

⚠ WARNING

Only the **noisy profiles** are suitable for safety-related process data communication.

Severe or fatal injuries.

- Suitable safeethernet profiles: Fast & Noisy, Medium & Noisy and Slow & Noisy.

The following table shows the available safeethernet profiles:

safeethernet profile	Application
Fast & Cleanroom	Only recommended for interference-free networks.
Fast & Noisy	Recommended for high availability of the safeethernet connection.
Medium & Cleanroom	Only recommended for interference-free networks.
Medium & Noisy	Recommended for high availability of the safeethernet connection.
Slow & Cleanroom	Only recommended for interference-free networks.
Slow & Noisy	Recommended for high availability of the safeethernet connection.

4.5.3 Profile I (Fast & Cleanroom)

⚠ WARNING

Only the **noisy profiles** are suitable for safety-related process data communication.

Severe or fatal injuries.

- Suitable safeethernet profiles: Fast & Noisy, Medium & Noisy and Slow & Noisy.

Application

The **Fast & Cleanroom** profile is suitable for applications in an ideal environment, such as a laboratory.

- For extremely fast data throughput.
- For applications that require fast data transmission.
- For applications that require the lowest possible worst case response time.

Network requirements

- Fast: 100 Mbit technology (100 Base TX), 1 Gbit technology.
- Clean: Interference-free network.
- Data losses due to network overload, outside influences and network manipulation must be avoided.
- LAN switches required.

Communication path characteristics

- Minimum delays.
- Expected response time ≤ ReceiveTMO

  (otherwise ERROR when setting parameters)

4.5.4 Profile II (Fast & Noisy)

Application

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

- For extremely fast data throughput.

21233799/EN – 07/2014

- For applications that require fast data transmission.
- For applications that require the lowest possible worst case response time.

Network requirements
- Fast: 100 Mbit technology (100 Base TX), 1 Gbit technology.
- Noisy: Network is not interference-free.

  Low likelihood of data packet loss.

  Time for ≥ 1 retries.
- LAN switches required.

Communication path characteristics
- Minimum delays.
- Expected response time ≤ ReceiveTMO / 2

  (otherwise ERROR when setting parameters)

4.5.5 Profile III (Medium & Cleanroom)

⚠ WARNING

Only the **noisy profiles** are suitable for safety-related process data communication.

Severe or fatal injuries.

- Suitable safeethernet profiles: Fast & Noisy, Medium & Noisy and Slow & Noisy.

Application

The **Medium & Cleanroom** profile is suitable for applications in an interference-free network that require only moderately fast data transmission.

- For medium data throughput.
- Appropriate for virtual private networks (VPN) where data exchange is slow but error-free due to intermediate safety systems (firewalls, encryption).
- Appropriate for applications where 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 required.
- Clean: Interference-free network.

  Data losses due to network overload, outside influences and network manipulation must be avoided.

  Time for ≥ 0 retries.

Communication path characteristics
- Moderate delays.
- Expected response time ≤ ReceiveTMO

  (otherwise ERROR when setting parameters)

4.5.6 Profile IV (Medium & Noisy)

Application

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

- For medium data throughput.
- For applications that require only moderately fast data transmission.
- Appropriate for applications where 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

21233799/EN – 07/2014

- LAN switches required.
- Noisy: Network is not interference-free.

  Low likelihood of data packet loss.

  Time for ≥ 1 retries.

Communication path characteristics
- Moderate delays
- Expected response time ≤ ReceiveTMO / 2

  (otherwise ERROR when setting parameters)

4.5.7 Profile V (Slow & Cleanroom)

⚠ WARNING

Only the **noisy profiles** are suitable for safety-related process data communication.

Severe or fatal injuries.

- Suitable safeethernet profiles: Fast & Noisy, Medium & Noisy and Slow & Noisy.

Application

The **Slow & Cleanroom** profile is suitable for applications in an interference-free network that require only slow data transmission.

- For slow data throughput.
- For applications that require only slow data transmission to (possibly far away) controllers and where the conditions of the communication path cannot be predicted.

Network requirements
- Slow: Data transfer via ISDN, dedicated line or radio relay connection.
- Clean: Interference-free network.

  Data losses due to network overload, outside influences and network manipulation must be avoided.

  Time for ≥ 0 retries.

Communication path characteristics
- Moderate delays.
- Expected response time ≤ ReceiveTMO

  (otherwise ERROR when setting parameters)

4.5.8 Profile VI (Slow & Noisy)

Application

The **Slow & Noisy** profile is for applications that require only slow data transmission to (possibly far away) controllers.

- For slow data throughput.
- For applications, mainly for data transfer via bad telephone wires or faulty radio relay paths.

Network requirements
- Slow: data transfer via telephone, satellite, radio, etc.
- Noisy: Network is not interference-free.

  Low likelihood of data packet loss.

  Time for ≥ 1 retries.

Communication path characteristics
- Moderate to long delays.
- Expected response time ≤ ReceiveTMO / 2

  (otherwise ERROR when setting parameters)

21233799/EN – 07/2014

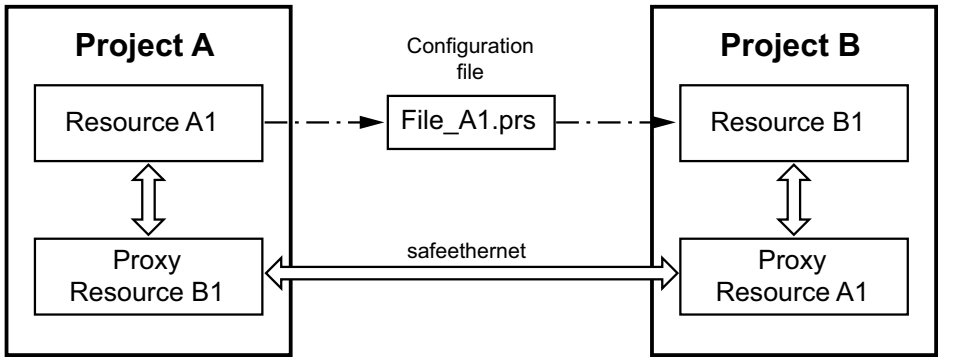
4.6 Communication across projects

Communication across projects is used for the following purposes:

- To connect resources from different projects with each other.
- To connect controllers with the SILworX® operating system and controllers via safeethernet.

The two projects communicate via safeethernet and the communication is configured in the safeethernet editor.

safeethernet connection between resource A1 in project A and resource B1 in project B:



5306777483

The project for which you configure the safeethernet connection and create the configuration file is called the local project.

The project into which you import the configuration file is called the target project.

During data exchange, the local project and the target project are equal communication partners.

Each proxy resource serves as a placeholder for the corresponding resource from the external project and is used for importing and exporting the safeethernet connections.

The *Proxy Resource B1* in project A is the placeholder for the *Resource B1* in project B.

The *Proxy Resource A1* in project B is the placeholder for the *Resource A1* in project A.

You must create and configure the proxy resource (here *Proxy Resource B1*) manually in the local project (here *Project A*). After configuration, import the configuration file (here `File_A1.prs`) into the target project (here *Resource B1*).

The configuration file `File_A1.prs` contains the complete description of *Resource A1*. for safeethernet connection with *Resource B1*. After having imported the configuration file `File_A1.prs` into *Resource B1*, the *Proxy Resource A1* is automatically created in Project B.

4.6.1 Variants of communication across projects

In the two variants below, projects A and B communicate with each other via safeethernet.

In the first variant, project A is the local project, and in the second variant project B is the local project. In general, the user may decide which of the two projects to create the configuration in.

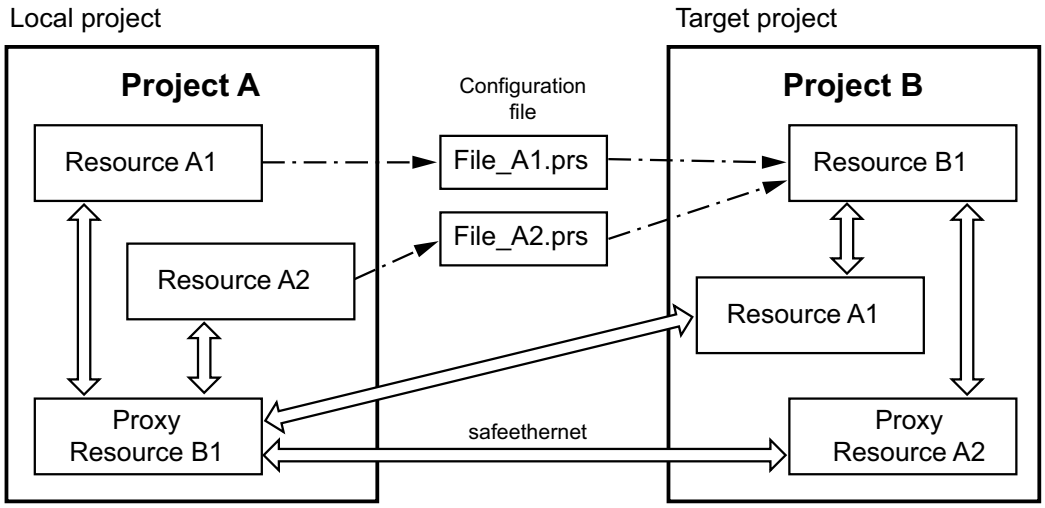
The effort of configuration is about the same either way, and results in the same configuration.

21233799/EN – 07/2014

Local project A

In the local project A, you configure the communication with target project B and create the configuration files. The advantage of this method is that you only have to create *Proxy resource B1* manually in the local project.

Variant with project A as local project:

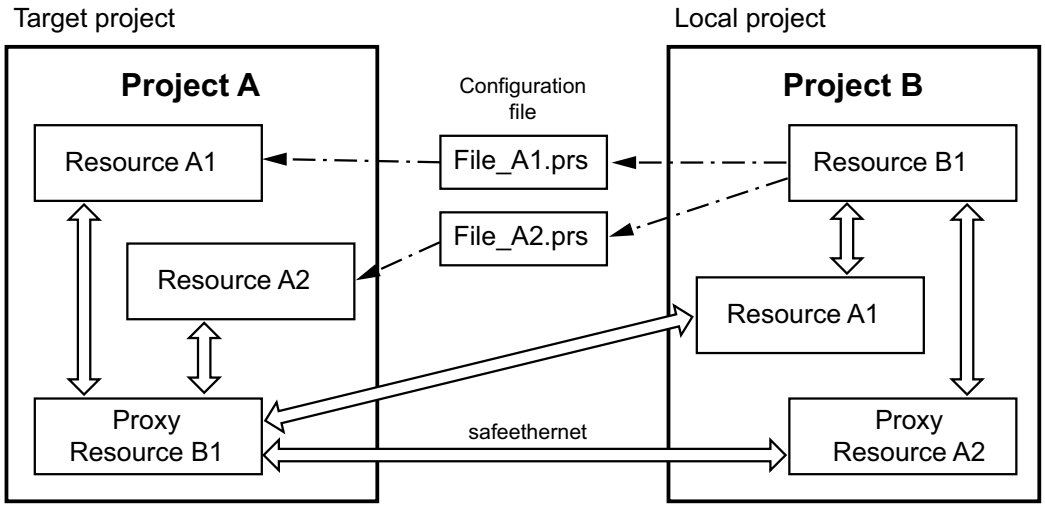


5306781963

Local project B

In the local project B, you configure the communication with target project A and create the configuration files. The disadvantage is that you have to create two *Proxy Resources* (*A1* and *A2*) manually in local project B.

Variant with project B as local project:



5306786827

4.7 Control panel (safeethernet)

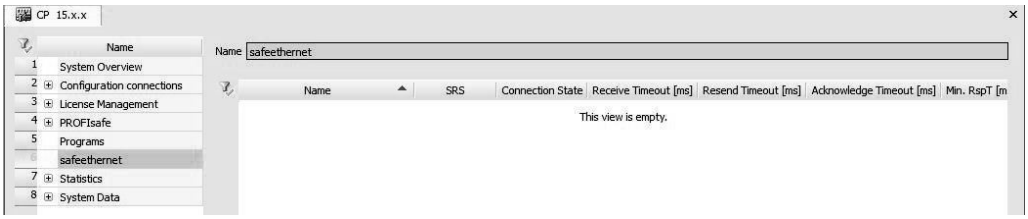
The user can verify and control the settings for the safeethernet connection in the control panel. Current status information (such as cycle time, bus status, etc.) for the safeethernet connection is also displayed.

Do the following to open the control panel to monitor the safeethernet connection:

1. In the structure tree, select [Resource].
2. From the context menu of the resource, choose [Online].
3. To open the control panel of the resource, enter access data in the system login.

21233799/EN – 07/2014

4. In the structure tree of the control panel, choose [safeethernet].



5428388235

Resetting statistical values:

Using the context menu function lets you reset the statistical data (cycle time min., max., etc.) to zero.

Do the following to reset the statistical data of the safeethernet connection:

- In the structure tree, select safeethernet connection.
- From the context menu of the safeethernet connection, choose [Reset safeethernet statistics].

4.7.1 Display field (safeethernet connection)

The following values for the selected safeethernet connection are shown in the display field:

Element	Description
Name	Resource name of the communication partner.
SRS	System.Rack.Slot
Connection status	Status of the safeethernet connection (see also chapter "Detailed view of the safeethernet editor").
Receive timeout [ms]	See chapter "safeethernet parameters".
Resend timeout [ms]	See chapter "safeethernet parameters".
Acknowledge timeout [ms]	See chapter "safeethernet parameters".
Min. RspT [ms]	Actual *Response time* as minimum, maximum, last and average values (see chapter "safeethernet parameters").
Max. RspT [ms]	
Last RspT [ms]	
Average RspT [ms]	
Faulty messages	Number of messages rejected since the statistics were reset.
Retries	Number of retries since the statistics were reset.
Number of successful connections	Number of successful connections since the statistics were reset.
Early queue usage	Number of messages that were placed into the *early queue* since the statistics were reset (see also chapter "safeethernet parameters").
Frame no.	Revolving sent counter.
Ack. frame no.	Revolving receipt counter.
Monoticity	Revolving user data sent counter.
Layout version	Signature of the current communication end point.

21233799/EN – 07/2014

Element	Description
New layout version	Signature of the new communication end point.
Connection control	Status of the connection control.
Channel 1 transmission control	Enabling Channel 1 transmission route (see chapter "safeethernet parameters").
Channel 2 transmission control	Enabling Channel 2 transmission route (see chapter "safeethernet parameters").
Quality of channel 1	Status of Channel 1 transmission route (see chapter "safeethernet parameters").
Quality of channel 2	Status of Channel 2 transmission route (see chapter "safeethernet parameters").
Redundant messages received late	For redundant transmission routes. Number of messages received late since the statistics were reset.
Lost redundant messages	For redundant transmission routes. Number of messages received on only one of the two transmission routes since the statistics were reset.
Protocol version	2: New protocol version for CPU operating system V7 or later.

4.8 Maximum communication time slice

The maximum communication time slice is the allotted time in milliseconds (ms) per cycle during which the processor system processes the communication tasks. If not all communication tasks scheduled for a cycle can be processed, the complete communication data is sent over several cycles (number of communication time slices > 1).

INFORMATION

The condition applies that the number of communication time slices = 1. Set the duration of the communication time slice high enough so that the cycle cannot exceed the watchdog time specified by the process when using the entire communication time slice (see also chapter "Maximum response time for safeethernet").

4.9 Connections for safeethernet/Ethernet

The safety controller has the following interfaces for networking via safeethernet/ Ethernet:

The following interfaces are available:

- **2 Ethernet interfaces:** X4233_1 and X4233_2

  Both interfaces are located on the terminal strip of the unit.

- **1 Ethernet service interface:** X4223

  For connecting a programming unit (PADT).

The different systems can be freely linked to each other via Ethernet (star or linear configuration). A programming device (PADT) can also be connected at any location.

21233799/EN – 07/2014

INFORMATION

Disruptions can occur in Ethernet operation.

- Take care that no network rings are created during interconnection.
- Data packets can only reach a system on **one** path.

21233799/EN – 07/2014

5 PROFINET IO

PROFINET IO is an Ethernet-based transmission protocol of the PROFIBUS user organization for automation technology. With PROFINET IO, the decentral field units are integrated in SILworX® by means of a device description (GSDML file up to V2.2), just like with PROFIBUS DP.

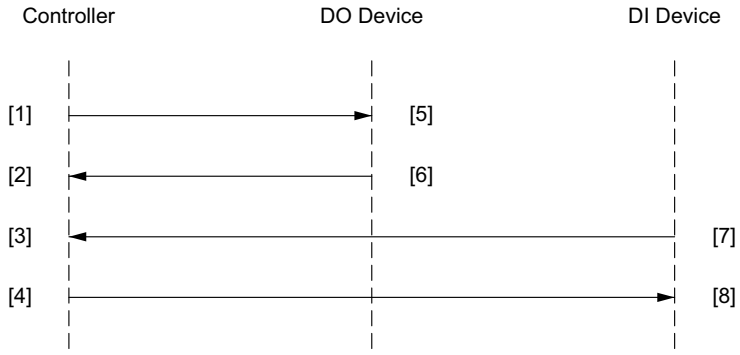
The PROFINET IO controller complies with conformance class A and supports non real-time (NRT) and real-time (RT) communication with the PROFINET IO devices. Real-time communication is used for time-critical data exchange whereas non real-time communication is used for processes that are not time-critical (such as acyclical reading/writing).

A redundant PROFINET IO connection can be achieved by configuring a second PROFINET IO controller I/O device and by adjustments in the user program.

5.1 Controlling the consumer/provider status (IOxS)

The consumer/provider status (IOxS) can be controlled via the user program using system variables. These system variables are described in this chapter. If you do not want to control the consumer/provider status via user program, you must assign the output variables a constant set to TRUE. The states are set to GOOD as soon as the communication module has received valid process values from the processor module.

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



[1] Valid output data

[2] Output data accepted by the device

[3] Valid input data

[4] Input data accepted by the controller

[5] Output data accepted by the controller

[6] Valid output data

[7] Input data accepted by the device

[8] Valid input data

5.1.1 Control variables in the controller

The consumer/provider status (IOxS) can be controlled via the user program using the output variables *Valid output data* [1] and *Input data accepted by the controller* [4].

The consumer/provider status (IOxS) can be read via the user program using the input variables *Output data accepted by the device* [2] and *Input data valid* [3].

21233799/EN – 07/2014

5.1.2 Control variables in the DO device

The consumer/provider status (IOxS) can be controlled via the user program using the output variable *Valid output data* [6]. The consumer/provider status (IOxS) can be read via the user program using the input variable *Output data accepted by the controller* [5].

5.1.3 Control variables in the DI device

The consumer/provider status (IOxS) can be controlled via the user program using the output variable *Input data accepted by the device* [7]. The consumer/provider status (IOxS) can be read via the user program using the input variable *Valid input data* [3].

5.2 PROFIsafe

Knowledge of the PROFIsafe specification of the PROFIBUS user organization is a prerequisite. PROFIsafe uses the PROFINET protocol for sending safety-related data up to SIL 3 based on Ethernet technology.

PROFIsafe is a protocol superimposed on the PROFINET protocol and contains safe user data as well as information about data integrity. The safe PROFIsafe data is sent to the lower-level PROFINET protocol together with the non safety-relevant PROFINET data.

Similar to the black channel principle, PROFIsafe uses "unsafe data transmission channels" (Ethernet) for transmitting safe data. In this way, F-host and F-device exchange the safe PROFIsafe data.

According to the PROFIsafe specification, the F-host repeatedly sends a message packet until the F-device acknowledges receipt to the F-host. The F-host does not send a new message packet to the F-device until it receives the acknowledgement.

The current process value is sent in each repeated PROFIsafe message packet. This is the reason why the same process signal can have different values in the repeated message packets.

In SEW devices, PROFIsafe is implemented on the receiver side in such a way 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 the same sequential number of the message packet) are rejected.

If the connection is lost and *F_WD_Time* has expired, the PROFIsafe process variables adopt their initial values. The process value must remain unchanged for at least the following time so that the remote end (F-host/F-device) receives a certain process value:

2 × *F_WD_Time* + *F_WD_Time2*

The user has to parameterize the PROFIsafe system in such a way that the SFRT (Safety Function Response Time) is appropriate for performing the corresponding safety function (see chapter "Remark regarding SFRT calculations").

21233799/EN – 07/2014

INFORMATION

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

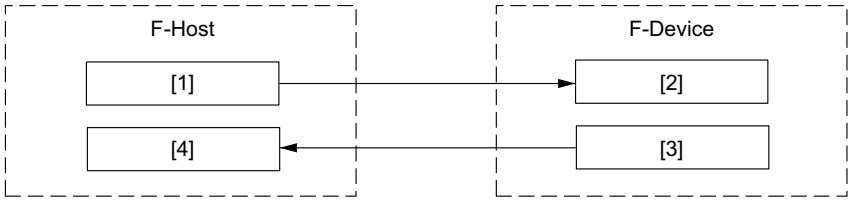
The default settings in SILworX® ensure a behavior consistent with PROFIsafe.

- The initial values of the process value variables must be set to "0".
- The parameter *AutoAwaitFParamsOnConnLoss* must be disabled (see chapter "Menu functions of the PROFINET IO Device").

5.2.1 PROFIsafe control byte and status byte

Each PROFIsafe submodule contains the two system variables *Control byte* and *Status byte*. These variables are exchanged during communication between F-host and F-device (see chapters "F-parameters from submodule input" and "PROFINET IO and PROFIsafe module").

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.



12214037131

[1] Writing control byte
[2] Reading control byte
[3] Writing status byte
[4] Reading status byte

INFORMATION

The system variables *Control byte* and *Status byte* have additional features that deviate from the PROFIsafe specification.

5.2.2 PROFIsafe watchdog time (F_WD_Time)

The following inequation applies to a functional PROFIsafe connection between an SEW F-host and an F-device:

F_WD_Time > Sum of the following components:

- 3 × CPU cycle time × number of communication time slices
- 2 × PROFINET controller production interval
- 1 × DAT (F-device acknowledgement time)
- 2 × internal bus time of the F-device
- 2 × PROFINET device production interval
- 2 × Ethernet delay

The production interval of PROFINET controller and PROFINET device is usually identical and is calculated as follows:

Reduction factor × SendClockFaktor × 31.25 µs

21233799/EN – 07/2014

INFORMATION

For the DAT (F-Device Acknowledgement Time) and the internal bus time of the device, refer to the device description provided by the F-device manufacturer.

Information regarding the PROFIsafe watchdog time (F_WD_Time)

- DAT (F-Device Acknowledgement Time) is the time an F-device requires to respond to a received PROFIsafe message. An F-device is a safe unit (in SEW-EURODRIVE systems the CPU module) that executes the F-device stack. If modular systems/devices are used, they do not include the time values for the non-safety 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 bus of the F-device.
  - DAT does not include the portion of PROFINET device production intervals.
  - DAT does not include any delays, for example due to input or output value filters or the physical properties of the inputs and outputs.
  - Depending on the connection, DAT refers to DATin (input) or DATout (output).
  - Use the corresponding maximum value for all time parameters.
- Internal bus time of the F-device for the safety controller:

  (maximum number of communication time slices − 1) × WDT − CPU
- Requirement: the F-device runs cyclically and its DAT is:

  DAT = 2 × max. cycle
  - F-device **does not** operate with communication time slices. If (SEW CPU cycle time × number of communication time slices) < F-device cycle time, then delta = (F-device cycle time) − (SEW CPU cycle time × number of communication time slices) must be added to the SEW CPU cycle time of the *F_WD_Time* calculation.

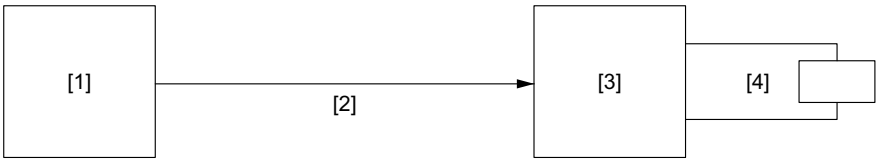
    F-device works **with** communication time slices. If (SEW CPU cycle time × number of communication time slices) < F-device cycle time × number of F-device communication time slices), then delta = (F-device cycle time × number of device communication time slices) − (SEW CPU cycle time × number of communication time slices) must be added to the SEW CPU cycle time of the *F_WD_Time* calculation.

21233799/EN – 07/2014

5.2.3 Safety function response time (SFRT)

Calculating the SFRT between an F-device and an SEW F-host

The permitted SFRT for a PROFIsafe connection between F-device and SEW F-host with local output is calculated as follows:



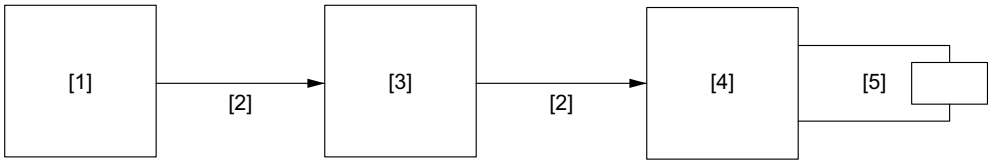
12214416523

[1] F-device
[2] PROFIsafe protocol
[3] SEW F-host
[4] Output

$SFRT \leq MaxDateAgeIn + 2 \times F\_WD\_Time + MaxDataAgeOut + T_u$

Calculating the SFRT with F-devices and an SEW F-host

The permitted SFRT for a PROFIsafe connection between an SEW host and an F-device with local output is calculated as follows:



12214421899

[1] F-device
[2] PROFIsafe protocol
[3] SEW F-host
[4] F-device
[5] Output

$SFRT \leq MaxDateAgeIn + 2 \times F\_WD\_Time\ (Input) + 2 \times WDT\text{-}CPU + 2 \times F\_WD\_TIME\ (Output) + MaxDataAgeOut + T_u$

Information regarding safety function response time calculations

1. Definition of SFRT according to IEC 61784-3 Ed.2.
2. All additional delays in the user program (for example due to TOF or TON function blocks) or in the modules (for example due to output filters, input filters, relays, etc.) must be added.
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 an F-device. But only the portion is added that is not already contained in *DATin*.

INFORMATION

In the safety controller, *MaxDataAgeIn* has the following values:

- For physical inputs up to FTT-CPU − 2 × WDT − CPU − DATin (FTT: fault tolerance time of the CPU module)
- For data created by the user program = 0 ms

4. MaxDataAgeOut

21233799/EN – 07/2014

This is the worst case response time of an F-output device or F-host for

- Outputting received process values to a physical output
- Controlling the physical outputs after expiry of F_WD_Time
- Deactivating the physical outputs in the event of device failure

After expiry of *F_WD_TIME*, the safety controller responds without faults after 2 × WDT − CPU at the latest.

- If the F-host/F-device fails immediately before this response, then the outputs of the safety controller are de-energized once WDT − CPU has expired.
- Assuming that only one fault occurs, then 1 × WDT − CPU can be subtracted from *MaxDataAgeOut*.

5. T_u is the minimum value of DATin, DATout and WDT − CPU. Theoretically you can use half of the value of DATin and DATout but the manufacturer must specify to which inaccuracy degree the device monitors *F_WD_Time*. If the device runs cyclically, then DAT= 2 × max. device cycle.
6. F_WD_TIME, see chapter "F_WD_Time (PROFIsafe watchdog time)".

5.3 Requirements for safe operation of PROFIsafe

5.3.1 Addressing

The SEW-EURODRIVE PROFIsafe network corresponds to the PROFINET Ethernet network that can be used to send PROFIsafe messages. In this context, network refers to a logical network that can include several physical subnetworks.

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 example, 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, for example via port-based VLANs, one-to-one addressing should be aimed at. One-to-one addressing ensures that no connections are established accidentally between PROFIsafe networks during upgrade or maintenance.

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

- One-to-one correspondence must be ensured between the F-addresses of the PROFIsafe devices/modules in a PROFIsafe network.
- We recommend that you ensure one-to-one correspondence between F-addresses even in separated PROFIsafe networks.
- When starting up or modifying safety functions, ensure that the safety functions use the proper inputs and outputs of the corresponding PROFIsafe devices throughout the entire PROFIsafe network.
- Configure the PROFIsafe F-modules in such a way (for example by assigning suitable F-addresses or F_WD_Time) that F-modules with identical input and output data lengths that operate in a PROFIsafe network have different CRC1 signatures. You can read the CRC1 in SILworX®.

  This is in any case ensured for F-modules operating in a PROFIsafe network if only one F-host is used and if the F-parameters differ only in the F-address. To avoid accidental generation of the same CRC1 signature, make sure that for example the *F_WD_Time* and *F_Prm_Flag1/2* parameters are identical for all F-modules and that the F-modules do not use an iPar CRC.

21233799/EN – 07/2014

Risk associated with PROFIsafe devices with identical input and output data length

PROFIsafe devices may only be operated when the F-INPUT data length does not equal the F-OUTPUT data length of the same PROFIsafe connection. Else, addressing faults in standard components and/or the standard transmission technology might not be detected and could result in safety-related malfunctions.

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

5.3.2 Network aspects

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

INFORMATION

If PROFIsafe detects reduced transmission quality that is not recognized by the standard transmission facilities (Ethernet), then a safety response is triggered.

The problems causing the safety response must be eliminated to ensure sufficient transmission quality again. Take the required measures first before triggering the acknowledgement for restarting PROFIsafe. To trigger acknowledgement, use the "Operator Acknowledge" or "Reset signal".

INFORMATION

Use "Operator-Acknowledge" and "Reset" only when hazardous conditions no longer exist.

Protect the PROFIsafe network against unauthorized access (such as DoS, hacker). The measures must be agreed upon with the supervising authority. This is particularly important when using wireless transmission technologies. For more information, see PROFIsafe Specification V2.5c, tables 23 and 24.

Availability with respect to added messages

Message packets can be stored, for example, using network components such as switches, and can be added (sent) at a later point in time. These message packets cause a shutdown if they are older than the last message packet received by the PROFIsafe device (see consecutive number tables 46 and 63).

5.4 PROFINET IO controller and PROFIsafe F-host

This chapter describes the characteristics of the PROFINET IO controller and PROFIsafe F-host as well as the menu functions and dialog boxes in SILworX® required to configure the PROFINET IO controller and PROFIsafe host.

- PROFINET IO controller system requirements:

Element	Description
Controller	PFF-HM31A safety controller
CPU module	The Ethernet interfaces of the processor module cannot be used for PROFINET IO.

21233799/EN – 07/2014

Element	Description
COM module	Ethernet 10/100BaseT.
Activation	Software activation code required, see chapter "Communication".

- Properties of the PROFINET IO controller:

Properties	Description
Safety-related	No
Transmission rate	100 Mbit/s full duplex
Transmission path	Ethernet interfaces of the COM module. Ethernet interfaces in use can also be used simultaneously for other 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 controllers	One PROFINET IO controller can be configured for each COM module.
Max. number of PROFINET IO devices applications relations (ARs)	One 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 communication relation (CR)	Output: max. 1440 bytes Input: max. 1440 bytes
Send cycle	Possible at device level by setting the reduction rate.

- Properties of the PROFIsafe host:

Properties	Description
Max. number of F-hosts	1012 or 512
Max. process data length of a communication relation (CR)	Output: max. 123 bytes user data + 5 bytes management data Input: max. 123 bytes user data + 5 bytes management data
Max. user data size	512 x 123 bytes = 62976 bytes

5.5 Example of PROFINET IO / PROFIsafe (controller)

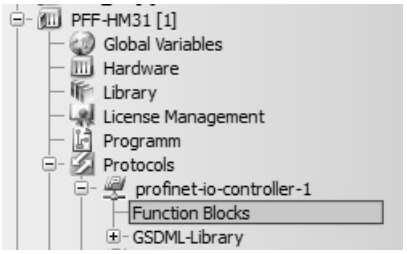
This example shows how to configure a PROFINET IO controller, which has a connection to the PROFINET IO device, on a safety controller. The PROFINET IO device is equipped with a PROFINET IO module and a PROFIsafe module. In the PROFINET IO controller, the PROFINET IO device must be configured according to its actual structure.

21233799/EN – 07/2014

5.5.1 Creating a PROFINET IO controller in SILworX®

Proceed as follows:

1. In the structure tree, choose [Configuration] / [Resource] / [Protocols].
2. To add a new PROFINET IO controller, open the context menu of Protocols and choose [New] / [PROFINET IO controller].
3. From the context menu of the PROFINET IO controller, choose [Properties].
4. In the "Name" field, enter the device name of the controller.
5. Select a COM module.



12843370507

Configuring the device in the PROFINET IO controller

Do the following to create a PROFINET IO device in the PROFINET IO controller:

1. From the context menu of the PROFINET IO controller, choose [New] / [PROFINET IO device].

Reading a GSDML library file from an external data source

Do the following to read the GSDML library file from an external data source (such as CD, USB flash drive, internet):

1. In the structure tree, choose [Configuration] / [Resource] / [Protocols] / [PROFINET IO Controller] / [GSDML library].
2. From the context menu of the GSDML library, choose [New] and add the GSDML file for the PROFINET IO device.

Loading a GSDML file for a new PROFINET IO device

Do the following to load the GSDML file for a new PROFINET IO device:

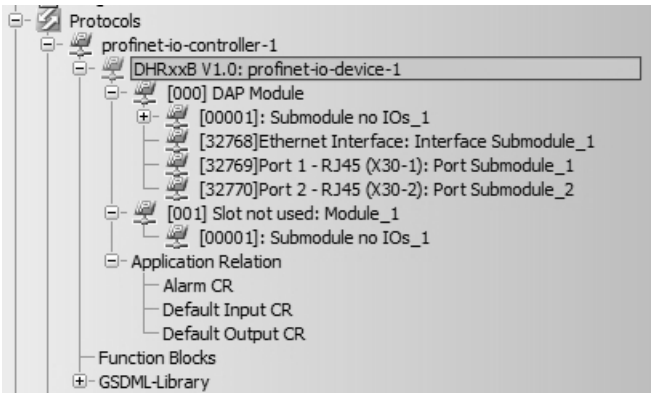
1. In the structure tree, choose [Configuration] / [Resource] / [Protocols] / [PROFINET IO Controller] / [PROFINET IO device].
2. From the context menu of the PROFINET IO device, choose [Properties], and open the "Parameters" tab.
3. In the "Name" field, enter the device name of the device.
4. In the "IP address" field, enter the IP address of the PROFINET IO device.
5. From the dropdown menu for "GSDML file", choose the GSDML library for the PROFINET IO device. Close the "Properties" window.

Selecting the device access point (DAP) for the PROFINET IO device

Do the following to select the device access point (DAP) for the PROFINET IO device:

1. In the structure tree, choose [Protocols] / [PROFINET IO controller] / [PROFINET IO device] / [DAP module].

2. From the context menu, choose [Edit] and choose the suitable device access point (DAP) data set for the PROFINET IO device.



12843376011

INFORMATION

The GSDML library file often contains several device access points (DAP) of a manufacturer.

Configuring module slots

Do the following to configure module slots:

1. In the structure tree, open [Protocols] / [PROFINET IO device].
2. To open the module list, choose [New] from the context menu of the PROFINET IO device.
3. From the module list, choose suitable modules for the PROFINET IO device and click [Add Module(s)] to confirm the action.

Numbering PROFINET IO device modules

Do the following to number PROFINET IO device modules:

The device access point (DAP) module has slot 0 by default. All other PROFINET IO device modules must be numbered.

1. From the context menu of the PROFINET IO device module, choose [Properties].
2. In the "Slot" field, enter the device module slots in the same order as they are arranged on the actual PROFINET IO device.
3. Repeat this step for the other PROFINET IO device modules.

The "Model" and "Features" tabs show further details of the GSDML file.

Configuring an application relation

Do the following to configure an application relation:

1. In the structure tree, open [PROFIsafe IO Device] / [Application Relation].
2. From the context menu of "Default Input CR", choose [Properties].
3. Adjust the *reduction factor* parameter (set it to 4, for example).
4. From the context menu of "Default Output CR", choose [Properties].
5. Adjust the *reduction factor* parameter (set it to 4, for example).

21233799/EN – 07/2014

Configuring the device access point (DAP) module

Proceed as follows:

1. Choose [000] DAP Module, [xxxxx] DAP Submodule.
2. From the context menu of [xxxxx] DAP Submodule, choose [Edit].
3. In the "Edit" dialog, click the "System Variables" tab. If you do not want to control the consumer/provider status using a special user program logic, then assign a global variable with the initial value TRUE to the *Input data accepted by controller* output variable.

INFORMATION

You can use these system variables to control the consumer/provider status, see chapter "Controlling the consumer/provider status (IOxS)".

Setting the header parameters of the DAP for the PROFINET IO device

Do the following to set the header parameters of the device access point (DAP) for the PROFINET IO device:

1. In the structure tree, choose [Protocols] / [PROFINET IO Controller] / [PROFINET IO-Device] / [DAP Module] / [[xxxxx] DAP Submodule] / [Alarm Settings (Header): Parameters].
2. From the context menu, choose [Properties].
3. In the "Name" field, enter the parameter name of the header parameter.
4. Click the [Edit] button to open a dialog for setting or changing the settings for interfaces and diagnostics/alarms.

Configuring the PROFINET IO device module

INFORMATION

The sum of the variables in bytes must be identical with the size of the module in bytes.

Do the following to configure the PROFINET IO device module:

1. Select the submodule [[001] PROFINET IO Device Module] / [[xxxxx] PROFINET IO Device Submodule].
2. From the context menu of [xxxxx] submodule, choose [Edit].
3. In the "Edit" dialog, click the "Process Variables" tab.
4. In the object selection, select a suitable variable and drag it onto the Input Signals area.
5. Make a right mouse click anywhere in the Inputs Signals area and choose [New Offsets] from the context menu to re-generate the offsets of the variables.
6. In the "Edit" dialog, click the "System Variables" tab. If you do not want to control the consumer/provider status using a special user program logic, then assign a global variable with the initial value TRUE to the *Input data accepted by controller* and *Valid output data* output variables.

21233799/EN – 07/2014

INFORMATION

You can use these system variables to control the consumer/provider status, see chapter "Controlling the consumer/provider status (IOxS)".

Configuring the PROFIsafe IO device module

INFORMATION

The sum of the variables in bytes must be identical with the size of the module in bytes.

Do the following to configure the PROFIsafe IO device module:

1. In the structure tree, select [[001] PROFIsafe IO Device Module] / [[xxxxx] PROFIsafe IO Device Submodule].
2. From the context menu of [xxxxx] submodule, choose [Edit].
3. In the "Edit" dialog, click the "Process Variables" tab.
4. In the object selection, select a suitable variable and drag it onto the Input Signals area.
5. Make a right mouse click anywhere in the Inputs Signals area and choose [New Offsets] from the context menu to re-generate the offsets of the variables.
6. In the "Edit" dialog, click the "System Variables" tab. If you do not want to control the consumer/provider status using a special user program logic, then assign a global variable with the initial value TRUE to the *Input data accepted by controller* and *Valid output data* output variables.

INFORMATION

You can use these system variables to control the consumer/provider status, see chapter "Controlling the consumer/provider status (IOxS)".

Configuring F-parameters

Do the following to configure F-parameters:

1. In the structure tree, select [[001] PROFIsafe IO Device Module] / [[xxxxx] PROFIsafe IO Device Submodule] / [F-Parameters].
2. From the context menu of [xxxxx] submodule, choose [Edit].
3. In the "Edit" dialog, click the "Process Variables" tab.
4. In the object selection, select a suitable variable and drag it onto the Input Signals area.
5. Make a right mouse click anywhere in the Inputs Signals area and choose [New Offsets] from the context menu to re-generate the offsets of the variables.
6. In the "Edit" dialog, click the "System Variables" tab. If you do not want to control the consumer/provider status using a special user program logic, then assign a global variable with the initial value TRUE to the *Input data accepted by controller* and *Valid output data* output variables.

INFORMATION

You can use these system variables to control the consumer/provider status, see chapter "Controlling the consumer/provider status (IOxS)".

21233799/EN – 07/2014

Verifying the PROFINET IO configuration

1. In the structure tree, open [Configuration] / [Resource] / [Protocols] / [PROFINET IO Controller].
2. From the context menu of the PROFINET IO controller, choose [Verification].
3. Thoroughly verify the entries in the logbook and correct any errors.

INFORMATION

Compile the resource again and load it into the controller to ensure that the new configuration can be used for PROFINET IO communication.

Identifying a PROFINET IO device in the Ethernet network

Do the following to identify a PROFINET IO device in the Ethernet network:

1. Log in to the communication module containing the PROFINET IO controller.
2. In the structure tree of the online view, select [PROFINET IO Controller] / [PROFINET IO Station].
3. From the context menu, choose [PROFINET Network Station]. A list opens with all PROFINET devices in the network of this PROFINET IO controller.

Configuring the PROFINET IO device in online view

Do the following to configure the PROFINET IO device in online view:

1. In the list, open the context menu of the PROFINET IO device to be configured to change the settings.
2. From the context menu, choose [Name the PROFINET IO Device]. Enter the device name. Make sure that the PROFINET IO device name matches the project (only lowercase letters are permitted).
3. From the context menu, choose [Network Settings]. Set the IP address, subnet mask, and gateway.

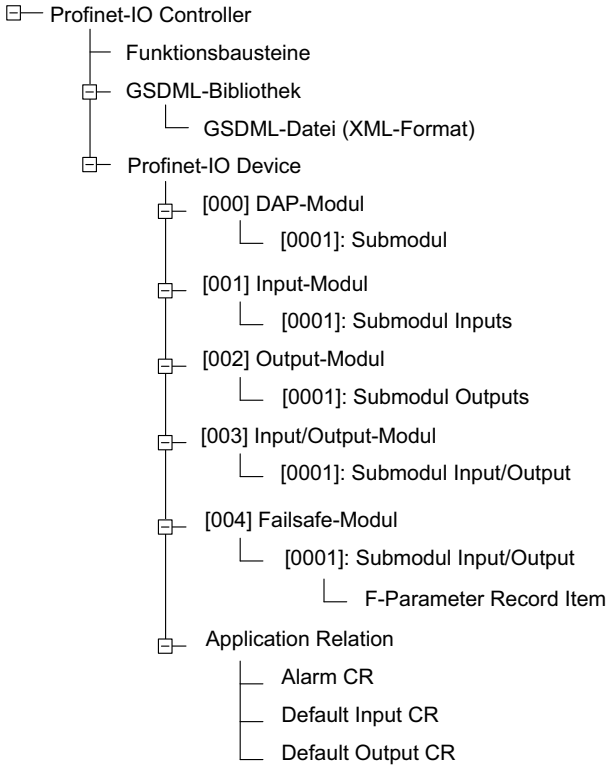
INFORMATION

Communication is only possible when the PROFINET IO device name and the network settings of the PROFINET IO device were configured in the PROFINET IO controller.

21233799/EN – 07/2014

5.6 Menu functions for the PROFINET IO controller

5.6.1 Example of a structure tree for the PROFINET IO controller



12760425611

5.6.2 PROFINET IO controller

To open the "Properties" dialog, choose [Properties] from the context menu of the PROFINET IO controller. The dialog contains the following tab.

PROFINET IO (properties) tab

Element	Description
Type	PROFINET IO controller
Name	Device name of the PROFINET IO controller.
Refresh rate for process data [ms]	Refresh rate in milliseconds at which the COM and CPU exchange protocol data. If the *Refresh Rate* is zero or lower than the cycle time for the controller, then data is exchanged as fast as possible. Range of values: 4 to (2^{31} − 1) Default: 0

21233799/EN – 07/2014

Element	Description
Force process data consistency	• Activated (default): All the protocol data is transferred from the CPU to the COM within a CPU cycle. • Deactivated: All the protocol data is transferred from the CPU to the COM, distributed over several CPU cycles, each with 1100 bytes per data direction. In this way, the cycle time of the controller can be reduced.
Module	Selection of the COM module on which the protocol is processed.
Activate max. µP budget	• Activated: Adopt the µP budget limit from the *Max. µP Budget in [%]* field. • Deactivated: Do not use a limit of the µP budget for this protocol.
Max. µP budget in [%]	Maximum µP budget for the module that can be used for processing the protocol. Range of values: 1 – 100% Default: 30%
RPC port server	Remote procedure call port Range of values: 1024 – 65535 Default: 49152 RPC port server and RPC port client must not be identical.
RPC port client	Remote procedure call port Range of values: 1024 – 65535 Default: 49153 RPC port server and RPC port client must not be identical.
F_Source_Add	Address of the controller (F-host). Users must use unique PROFIsafe controller/device addresses in a PROFIsafe network (see also IEC 61784-3-3 V2.5c chapter 9.7).

5.6.3 PROFINET IO device (in the controller)

To open the "Properties" dialog, choose [Properties] from the context menu of the PROFINET IO device. The dialog contains the following tabs:

Parameters (properties) tab

Element	Description
Name	Device name of the PROFINET IO device.
IP address	IP address of the communication partner. Range of values: 0.0.0.0 – 255.255.255.255 Default: 192.168.0.99 Do not use IP addresses that are already in use (see chapter "Network ports used for Ethernet communication").
Subnet mask	Subnet mask for the addressed subnet containing the device. Range of values: 0.0.0.0 – 255.255.255.255 Default: 255.255.255.0
GSDML file	GSDML is the abbreviation for generic station description markup language and refers to an XML-based description language. The GSDML file contains the master data of the PROFINET device.

Model and Features tabs

The "Model" and "Features" tabs show additional details of the GSDML file, such as *Manufacturer Name*, *Device Description* or *Supported Factors*. This additional information is intended to support the users when configuring the device, and must not be changed.

5.6.4 DAP module (device access point module)

Within a PROFINET device, a DAP module is used for connecting the bus. The DAP module is a default and cannot be deleted.

To open the "Properties" dialog, choose [Properties] from the context menu of the DAP module. The dialog contains the following tabs:

Parameters (properties) tab

Element	Description
Name	Name for the DAP module
Slot	Cannot be changed. Default: 0

Model and Features tabs

The "Model" and "Features" tabs show additional details of the GSDML file, such as *Module ID*, *Hardware/Software Version*. This additional information is intended to support the users when configuring the device, and must not be changed.

DAP submodule (properties)

To open the "Properties" dialog, choose [Properties] from the context menu of the DAP submodule. The dialog contains the following tabs:

Parameters tab

Element	Description
Name	Name of the input submodule.

21233799/EN – 07/2014

Element	Description
Subslot	Default: 1
IO data CR, inputs	Selection of the communication relation (CR) to which the submodule inputs should be transferred: • None • Default: Input CR
Input data accepted by the controller	Selection of the communication relation (CR) to which the consumer status (CS) of the submodule should be transferred: • None • Default: Output CR

Model and Features tabs

The "Model" and "Features" tabs show additional details of the GSDML file, such as *Submodule ID*, *Data Length*. This additional information is intended to support the users when configuring the device, and must not be changed.

21233799/EN – 07/2014

DAP submodule (edit)

To open the "Edit" dialog, choose [Edit] from the context menu of the input submodules. The dialog contains the following tabs:

"System variables" tab

The "System variables" tab provides the following system variables that allow users 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 the controller	• TRUE Valid input data (GOOD) • FALSE Invalid input data (BAD)

INFORMATION

You can use these system variables to control the consumer/provider status, see chapter "Consumer/provider status".

Header parameters

Some devices contain so-called header parameters that are used to activate/deactivate parameters, such as diagnostics, alarm, and interfaces.

5.6.5 Input/output PROFINET IO modules

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

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

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

Do the following to create the required PROFINET IO modules:

1. In the structure tree, open [Configuration] / [Protocols] / [PROFINET IO Device].
2. From the context menu of the PROFINET IO device, choose [New].
3. Choose the required modules.

To open the "Properties" dialog, choose [Properties] from the context menu of the input/output PROFINET IO modules. The dialog contains the following tabs:

Element	Description
Name	Name of the input/output PROFINET IO module
Slot	0 – 32767 Default: 1

21233799/EN – 07/2014

Model and Features tabs

The "Model" and "Features" tabs show additional details of the GSDML file, such as *Module ID*, *Hardware/Software Version*. This additional information is intended to support the users when configuring the device, and must not be changed.

5.6.6 Input submodule

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

Submodule input (properties)

To open the "Properties" dialog, choose [Properties] from the context menu of the input submodules. The dialog contains the following tabs:

Element	Description
Name	Name of the submodule input.
Subslot	Cannot be changed; default: 1
IO data CR, inputs	Selection of the communication relation (CR) to which the submodule inputs should be transferred: • None • Default input CR
Input data accepted by the controller	Selection of the communication relation (CR) to which the IO consumer status (CS) of the submodule should be transferred: • None • Default input CR
Shared input	• Activated Several PROFINET IO controllers can access the inputs. • Deactivated Only one PROFINET IO controller can access the inputs.
Input values if IO CR is disconnected	Behavior of the input variables for this PROFINET IO submodule when the connection is interrupted. • Retain last value The input variables are frozen at their current values and used until the connection is re-established. • Adopt initial values The initial data is used for the input variables.

"Model" and "Features" tabs

The "Model" and "Features" tabs show 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 when configuring the device, and must not be changed.

Submodule input (edit)

To open the "Edit" dialog, choose [Edit] from the context menu of the input submodules. The dialog contains the following tabs:

"System variables" tab

The "System variables" tab provides the following system variables that allow users to evaluate the state of the PROFINET IO submodule from within the user program.

INFORMATION

You can use these system variables to control the consumer/provider status, see chapter "Consumer/provider status".

Element	Description
Valid input data	• TRUE Valid input data (GOOD) • FALSE Invalid input data (BAD)
Input data accepted by the controller	• TRUE Valid input data (GOOD) • FALSE Invalid input data (BAD)
The following parameters are only available for PROFIsafe modules:	
Valid output data	• TRUE Valid output data (GOOD) • FALSE Invalid output data (BAD)
Output data accepted by the device	• TRUE Valid output data (GOOD) • FALSE Invalid output data (BAD)

21233799/EN – 07/2014

Element	Description
PROFIsafe control	With each message, PROFIsafe sends the PROFIsafe control byte from the controller to the device that can be set in the user program (see chapter "PROFIsafe control byte and status byte"). • Bit 0 iPar_EN_C: To load new parameters into the F-device, the iPar_EN_C must be set to TRUE to unlock the F-device. As long as iPar_EN_C is TRUE, failsafe values "0" are exchanged between F-host and F-device. • Bit 1 OA_C: Operator Acknowledge After a PROFIsafe error (for example CRC error or timeout), bit 1 must be set to TRUE for at least one PROFIsafe cycle. If you want to start or restart a PROFIsafe connection, the operator acknowledgement may only be issued when no dangerous states are present. • Bit 2, bit 3 Reserved • Bit 4 Activate_FV_C: – FALSE Process values are exchanged between F-host and F-device. – TRUE Failsafe values "0" are exchanged between F-host and F-device. • Bit 5 Reserved • Bit 6 • Bit 7 Reset_Comm: PROFIsafe communication reset; the protocol stack is set to the initial state. Bit 7 must be set to TRUE until the PROFIsafe status *Bit 2 Reset_Comm* has read back the value TRUE.

21233799/EN – 07/2014

Element	Description
PROFIsafe RoundTrip Time last	For an F-host, this is the time between sending a data message (with consecutive number N) and receiving the corresponding acknowledgement (with consecutive number N), measured in milliseconds.

21233799/EN – 07/2014

Element	Description
PROFIsafe status	Each message received by PROFIsafe on the host contains the PROFIsafe status byte that can be evaluated in the user program. • Bit 0 iPar_OK_S – TRUE New parameters received – FALSE No change • Bit 1 OA_Req_S Operator acknowledge requested • Bit 2 Reset_Comm This 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 – TRUE The F-device has reported a device fault. – FALSE The F-device has not reported any device fault. • Bit 6 WD_Timeout – TRUE Either the F-device has reported a watchdog timeout, or the host timeout on the F-host has elapsed. – FALSE A timeout has occurred neither on the F-device nor on the F-host. • Bit 7 CRC – TRUE

Element	Description
	Either the F-device has reported a CRC error, or a CRC error has occurred on the F-host. – FALSE A CRC error occurred neither on the F-device nor on the F-host.

The Process Variables tab is used to enter the process variables.

F-parameters of submodule input (for PROFIsafe modules only)

PROFIsafe F-devices need normalized F-parameters to exchange process data safely. The F-device does not establish communication until valid F-parameters are configured. Gray parameters are disabled and cannot be edited because some of them are preset by the GSDML file or are calculated automatically.

Element	Description
Name	Module name.
Index	Module index.
F_Par_Version	Only V2 mode is supported. V1 mode is rejected. Is determined by the GSDML file.
F_Source_Add	The *F_Source_Address* of the F-host must be unique within the PROFIsafe network. Range of values: 1 – 65534
F_Dest_Add	The *F_Destination_Adresse* of the F-device must be unique within the PROFIsafe network. Range of values: 1 – 65534
F_WD_Time	Watchdog time. Range of values: 1 ms to 65534 ms
F_iPar_CRC	Enter the *F_iPar_CRC* of the F-device into this field.
F_SIL	Displays the SIL level. 0: SIL 1 1: SIL 2 2: SIL 3 3: No SIL Is determined by the GSDML file.
F_Check_iPar	Displays the *iParameter CRC*. Is determined by the GSDML file.
F_Block_ID	Structure of the *F-parameters*. Is determined by the GSDML file.
F_CRC_Length	Indicates whether the 3-byte CRC is used or the 4-byte CRC. Is determined by the GSDML file.

21233799/EN – 07/2014

Element	Description
F_Par_CRC	Displays the P-parameter CRC (CRC1). Is calculated based on the current F-parameters.

5.6.7 Submodule output

The submodule parameters are used to define the communication relations of the modules and their behavior after connection is interrupted.

Submodule output (properties)

To open the "Properties" dialog, choose [Properties] from the context menu of the output submodules. The dialog contains the following tabs:

"Parameters" tab

Element	Description
Name	Name of the output submodule.
Subslot	Cannot be changed for PROFINET IO controllers. Default: 1
IO data CR, outputs	Selection of the communication relation (CR) to which the submodule outputs should be transferred. • None • Default input CR
Output data accepted by the device	Selection of the communication relation (CR) to which the submodule outputs should be transferred. • None • Default output CR

"Model" and "Features" tabs

The "Model" and "Features" tabs show 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 when configuring the device, and must not be changed.

Submodule output (edit)

To open the "Edit" dialog, choose [Edit] from the context menu of the output submodules. The dialog contains the following tabs:

"System Variables" tab

The "System Variables" tab provides the following system variables that allow users to evaluate the state of the PROFINET IO submodule from within the user program.

Element	Description
Valid output data	• TRUE Valid output data (GOOD) • FALSE Invalid output data (BAD)

21233799/EN – 07/2014

Element	Description
Output data accepted by the device	• TRUE Valid output data (GOOD) • FALSE Invalid output data (BAD)
The following parameters are only available for PROFIsafe modules:	
Valid input data	• TRUE Valid input data (GOOD) • FALSE Invalid input data (BAD)
Input data accepted by the controller	• TRUE Valid input data (GOOD) • FALSE Invalid input data (BAD)

For more parameters for PROFIsafe modules, see chapter "Submodule input (edit)".

INFORMATION

You can use these system variables to control the consumer/provider status, see chapter "Consumer/provider status".

The Process Variables tab is used to enter the outputs variables.

F-parameters of submodule output (for PROFIsafe modules only)

For a description of the F-parameters, see chapter "F-parameters of submodule input (for PROFIsafe modules only)".

21233799/EN – 07/2014

5.6.8 Submodule inputs and outputs

The submodule parameters are used to define the communication relations of the modules and their behavior after connection is interrupted.

Submodule inputs and outputs (properties)

To open the "Properties" dialog, choose [Properties] from the context menu of the input and output submodules. The dialog contains the following tabs:

"Parameters" tab

Element	Description
Name	Name of the input/output submodule.
Subslot	Cannot be changed for PROFINET IO controllers. Default: 1
IO data CR, inputs	Selection of the communication relation (CR) to which the submodule inputs should be transferred. • None • Default input CR
IO data CR, outputs	Selection of the communication relation (CR) to which the submodule outputs should be transferred. • None • Default output CR
Input data accepted by the controller	Selection of the communication relation (CR) to which the IO consumer status (CS) of the submodule should be transferred. • None • Default output CR
Output data accepted by the device	Selection of the communication relation (CR) to which the IO consumer status (CS) of the submodule should be transferred. • None • Default input CR

"Model" and "Features" tabs

The "Model" and "Features" tabs show 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 when configuring the device, and must not be changed.

Submodule inputs and outputs (edit)

To open the "Edit" dialog, choose [Edit] from the context menu of the input/output submodules. The dialog contains the following tabs:

21233799/EN – 07/2014

"System Variables" tab

The "System Variables" tab provides the following system variables that allow users to evaluate the state of the PROFINET IO submodule from within the user program.

Element	Description
Valid output data	• TRUE Valid output data (GOOD) • FALSE Invalid output data (BAD)
Output data accepted by the device	• TRUE Valid output data (GOOD) • FALSE Invalid output data (BAD)
Valid input data	• TRUE Valid input data (GOOD) • FALSE Invalid input data (BAD)
Input data accepted by the controller	• TRUE Valid input data (GOOD) • FALSE Invalid input data (BAD)

For more parameters for PROFIsafe modules, see chapter "Submodule input (edit)".

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

INFORMATION

You can use these system variables to control the consumer/provider status, see chapter "Consumer/provider status".

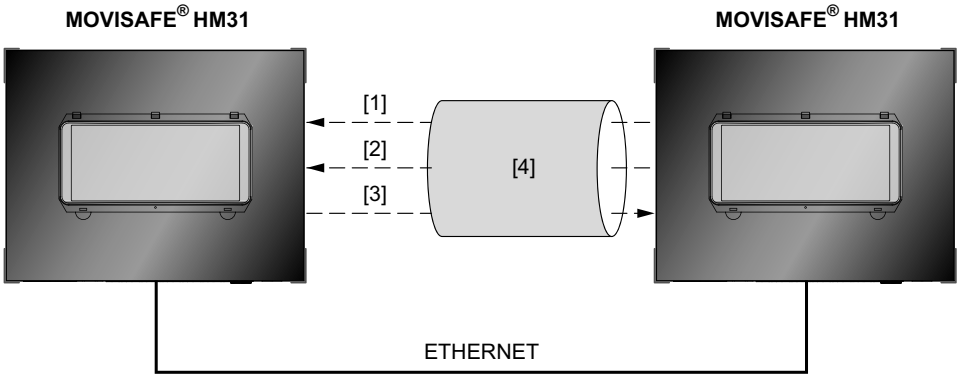
F-parameters of submodule inputs/outputs (for PROFIsafe modules only)

For a description of the F-parameters, see chapter "F-parameters of submodule input (for PROFIsafe modules only)".

21233799/EN – 07/2014

5.6.9 Application relation (properties)

An application relation (AR) is a logical construct for enabling data exchange between controller and device. In the example (see figure below), data is sent within the application relation via the standard communication relations (alarm CR, default input CR, and default output CR). These communication relations are already configured in the input and output modules by default.



[1] Alarm CR

[2] Default input CR

[3] Default output CR

[4] Application relation (AR)

To open the "Properties" dialog, choose [Properties] from the context menu of Application Relation.

Element	Description
Name	Cannot be changed.
AR UUID	Code for unique identification of the application relation (AR). Cannot be changed.
Connection establishment timeout factor	From perspective of a PROFINET IO device, this parameter is used to calculate the maximum time that is allowed while establishing a connection between sending the response on the connect request and receiving a new request from the PROFINET IO controller. Range of values: 1 – 1000 (× 100 ms) Default: 600
Supervisor may adopt AR	Definition whether a PROFINET IO supervisor may adopt the application relation (AR). • 0: Not allowed • 1: Allowed Default: 0

5.6.10 Alarm CR (properties)

Several communication relations (CR) can be established within an application relation. The PROFINET IO device uses the alarm CR to send alarms to the PROFINET IO controller.

21233799/EN – 07/2014

To open the "Properties" dialog, choose [Properties] from the context menu of the application relation. The dialog contains the following tabs:

Element	Description
Name	Cannot be changed.
VLAN ID, high-priority alarms	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 with ID = 2, 3, ... Range of values, see also IEC 61158-6 • 0x000: No VLAN • 0x001: Standard VLAN • 0x002 – 0xFFF: See IEEE 802.1 Q • Default: 0
VLAN ID, low-priority alarms	For a description, see VLAN ID, high-priority alarms Default: 0
Alarm 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 always a low priority.
Alarm resends	Maximum number of resend attempts by the device if the controller does not respond. Range of values: 3 – 15 Default: 10
Alarm timeout factor	The RTA timeout factor is used to calculate the maximum device time that may elapse after sending an RTA_Data (alarm) frame and receiving the RTA_ACK frame. Range of values: 1 – 65535 Default: 5

5.6.11 Input CR (properties)

The PROFINET IO device uses the input CR to send variables to the PROFINET IO controller.

To open the "Properties" dialog, choose [Properties] from the context menu of the input CR. The dialog contains the following tabs:

Element	Description
Name	Any unique name for an input CR. The default input CR cannot be changed.
Type	1 (cannot be changed).

21233799/EN – 07/2014

Element	Description
Send clock factor	The send clock factor defines the send clock for the cyclic IO CR data transfer. Send clock = send clock factor × 31.25 µs Range of values: 1 – 128 Default: 32
Reduction factor	The reduction factor lets you 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: Send cycle = reduction factor × send clock Range of values: 1 – 16384 Default: 32 (depending on the device)
Watchdog factor	From perspective of the IO CR consumer, the watchdog factor is used to calculate the maximum time allowed between the reception of two frames: Watchdog time = watchdog factor × send cycle Range of values: 1 – 7680 Default: 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 with ID = 2, 3, ... Range of values, see also IEC 61158-6 • 0x000: No VLAN • 0x001: Standard VLAN • 0x002 – 0xFFF: See IEEE 802.1 Q • Default: 0

21233799/EN – 07/2014

Input CR (edit)

To open the "Edit" dialog, choose [Edit] from the context menu of the output submodules. The dialog contains the following tabs:

Element	Value	Description
Data status input CR	0	State With redundant connections, *primary* writes to the leading channel. • 1 = Primary • 0 = Backup With mono connections: • 1 = Connected • 0 = Not connected
	1	Not used
	2	Data valid *Invalid* is set during the startup phase, or when the application is not able to report faults via IOPS. • 1 = Valid • 0 = Invalid
	3	Not used
	4	Process state For information only. The actual data validity is reported via IOPS. • 1 = Run • 0 = Stop
	5	Problem indicator *Problem detected* provides details on the diagnostic data of the alarm CR. • 1 = Regular operation • 0 = Problem detected
	6	Not used
	7	Not used

Output CR (properties)

Several communication relations (CR) can be established within an application relation. The PROFINET IO device uses the output CR to send variables to the PROFINET IO controller.

To open the "Properties" dialog, choose [Properties] from the context menu of the output CR. The dialog contains the following tabs:

Element	Description
Name	Any unique name for an output CR. The default output CR cannot be changed.

21233799/EN – 07/2014

Element	Description
Type	2 (cannot be changed).
Send clock factor	The send clock factor defines the send clock for the cyclic IO CR data transfer. Send clock = send clock factor × 31.25 µs Range of values: 1 – 128 Default: 32
Reduction factor	For setting the transmission frequency. The reduction factor lets you reduce the actual cycle time needed for sending the data of an IO CR. The actual data cycle time is calculated as follows: Send cycle = reduction factor × send clock Range of values: 1 – 16384 Default: 32
Watchdog factor	From perspective of the IO CR consumer, the watchdog factor is used to calculate the maximum time allowed between the reception of two frames: Watchdog time = watchdog factor × send cycle Range of values: 1 – 7680 Default: 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 with ID = 2, 3, ... Range of values, see also IEC 61158-6 • 0x000: No VLAN • 0x001: Standard VLAN • 0x002 – 0xFFF: See IEEE 802.1 Q • Default: 0

5.7 PROFINET IO device

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

5.7.1 System requirements

- Equipment and system requirements

Element	Description
Controller	MOVISAFE® HM31 safety controller

21233799/EN – 07/2014

Element	Description
CPU module	The Ethernet interfaces of the processor module cannot be used for PROFINET IO.
COM module	Ethernet 10/100BaseT
Activation	A software activation code is required (see chapter "Communication" in the "MOVISAFE® HM31 Safety Controller" system manual).

- PROFINET IO device properties

Element	Description
Safety-related	No
Transmission rate	100 Mbit/s full duplex
Transmission path	Ethernet interfaces of the COM module. Ethernet interfaces in use can also be used simultaneously for other 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.
Maximum 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.
Maximum number of communication relations (CRs for each AR)	Standard: 1 input, 1 output, 1 alarm
Maximum process data length of all configured PROFINET IO modules	Output: max. 1440 bytes Input: max. 1440 bytes
Data prioritization	Possible at device level by setting the *Reduction Rate*.

5.8 Example of PROFINET IO / PROFIsafe (device)

This chapter describes how to configure the PROFINET-IO / PROFIsafe device.

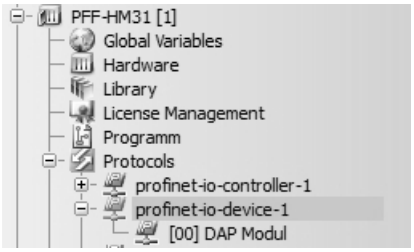
5.8.1 Configuring the PROFINET IO device in SILworX®

Proceed as follows:

1. In the structure tree, choose [Configuration] / [Resource] / [Protocols].
2. To add a new PROFINET IO device, open the context menu of Protocols and choose [New] / [PROFINET IO Device].
3. From the context menu of the PROFINET IO device, choose [Properties].
4. In the "Name" field, enter the device name of the PROFINET IO device.

21233799/EN – 07/2014

5. Select a COM module.



12843484683

Creating PROFINET IO modules

Do the following to create the required PROFINET IO modules:

1. In the structure tree, choose [Configuration] / [Resource] / [Protocols] / [PROFINET IO Device].
2. From the context menu of the PROFINET IO device, choose [New].
3. For this example, choose the following modules:

PROFINET IO / PROFIsafe module	Slot
In 1 byte	1
Safe Out 1Byte	2

Numbering PROFINET IO device modules

Do the following to number the PROFINET device modules:

1. From the context menu of the first PROFINET IO device module, choose [Properties].
2. In the "Slot" field, enter the value "1".
3. Repeat this step for all other PROFINET IO device modules and number the modules consecutively.

INFORMATION

Number the modules according to their actual position in the PROFINET IO device.

4. The following step is only required for PROFIsafe modules:

   In the "PROFIsafe F_Destination_Address" field, enter the address of the PROFIsafe module.

Configuring the PROFINET IO device input module

INFORMATION

The sum of the variables in bytes must be identical with the size of the module in bytes.

Do the following 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 the input module [01] In 1 Byte and choose [Edit] from the context menu.

21233799/EN – 07/2014

3. In the "Edit" dialog, choose the "Process Variables" tab.
4. In the object selection, select a suitable variable and drag it into the "Input Signals" area.
5. Make a right mouse click anywhere in the "Input Signals" area and choose [New Offsets] from the context menu to re-generate the offsets of the variables.
6. In the "Edit" dialog, choose the "System Variables" tab. If you do not want to control the consumer/provider status, then assign a global variable with the initial value TRUE to the *Input data accepted by the device* output variable.

INFORMATION

You can use these system variables to control the consumer/provider status, see chapter "Controlling the consumer/provider status (IOxS)".

Configuring the PROFIsafe device output module

Do the following to configure the output module [02] Safe Out 1 Byte:

1. In the PROFIsafe device, choose the output module [02] Safe Out 1 Byte.
2. Right-click [02] Safe 1 Byte and choose [Edit] from the context menu.
3. In the "Edit" dialog, select the "Process Variables" tab.
4. In the object selection, select a suitable variable and drag it into the "Output Signals" area.
5. Make a right mouse click anywhere in the "Output Signals" area and choose [New Offsets] from the context menu to re-generate the offsets of the variables.
6. In the "Edit" dialog, select the "System Variables" tab.

INFORMATION

You can use these system variables to control the consumer/provider status, see chapter "Controlling the consumer/provider status (IOxS)".

Verifying the PROFINET IO device configuration

Do the following to verify the PROFINET IO device configuration:

1. In the structure tree, open [Configuration] / [Resource] / [PROFINET IO Device].
2. Click the [Verification} button on the action bar, and click [OK] to confirm the action.
3. Thoroughly verify the entries in the logbook and correct any errors.

INFORMATION

Use the user program of the PROFINET IO device resource to recompile the configuration of the PROFINET IO device and transfer it to the controller. Only after this step, the new configuration can be used for communication with the PROFINET IO.

21233799/EN – 07/2014

5.9 Menu functions of the PROFINET IO device

5.9.1 "Properties" menu

To open the "Properties" dialog, choose [Properties] from the context menu of the PROFINET IO device.

Element	Description
Type	PROFINET IO device
Name	Any unique name for a PROFINET IO device.
Process data refresh rate in ms	Refresh rate in milliseconds at which the COM and CPU exchange protocol data. If the refresh rate is zero or lower than the cycle time for the controller, then data is exchanged as fast as possible. • Range of values: 4 to (2^{31} − 1) • Default: 0
Force process data consistency	• Activated (default): All the protocol data is transferred from the CPU to the COM within a CPU cycle. • Deactivated: The entire data of the protocol is transferred from the CPU to the COM and distributed via several CPU cycles with 1100 bytes each per data direction. This might also reduce the cycle time of the controller.
Module	Selection of the COM module on which this protocol is processed.
Activate max. µP budget	• Activated: Adopt the µP budget limit from the *Max. µP Budget in [%]* field. • Deactivated: Do not use a limit of the µP budget for this protocol.
Max. µP budget in %	Maximum µP budget for the module that can be used for processing the protocol. • Range of values: 1 – 100% • Default: 30%
RPC port server	Remote procedure call port • Range of values: 1024 – 65535 • Default: 49152 RPC port server and RPC port client must not be identical.
RPC port client	Remote procedure call port • Range of values: 1024 – 65535 • Default: 49153 RPC port server and RPC port client must not be identical.

21233799/EN – 07/2014

Element	Description
AutoAwaitFParamsOn-ConnLoss	This parameter is **only used for PROFIsafe modules**. Whenever the connection to the F-host is lost, the F-parameters must be reloaded into the F-device. You can activate this parameter to simplify PROFIsafe startup. Afterwards, the F-device automatically sets the required F-parameters at restart or after a connection loss. After startup, it is essential that you deactivate this parameter for PROFIsafe-compliant behavior. • Activated (default): The F-device automatically enters the *Wait for F-parameters* state. • Deactivated: The user must send an online command to have the F-device enter *Wait for F-parameters* state.

5.9.2 PROFINET IO modules

The following PROFINET IO modules are available in the PROFINET IO device:

PROFINET IO module	Max. input variable size	Max. output variable size
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

21233799/EN – 07/2014

PROFINET IO module	Max. input variable size	Max. output variable size
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

5.9.3 PROFIsafe modules

The following PROFIsafe modules are available in the PROFINET IO device:

PROFIsafe module	Max. input variable size	Max. output variable size
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

21233799/EN – 07/2014

PROFIsafe module	Max. input variable size	Max. output variable size
Safe Out 123 bytes		123 bytes

Creating a PROFINET module or a PROFIsafe module:

1. In the structure tree, open [Configuration] / [Resource] / [Protocols] / [PROFINET IO Device].
2. From the context menu of the PROFINET IO device, choose [Insert Modules].
3. Select a suitable module for transporting the required process data.
4. Right-click the selected module and choose [Edit] from the context menu.
   - In the "Process Variables" tab, enter the input and/or output variables.
   - If you do not want to control the consumer/provider status using a special user program logic, then assign a global variable with the initial value TRUE to the *Input data accepted by controller* and *Valid output data* output variables in the "System Variables" tab.

INFORMATION

You can use these system variables to control the consumer/provider status, see chapter "Controlling the consumer/provider status (IOxS)".

5. The following setting is only required for PROFIsafe modules:

   In the "Properties" tab, enter the slot and the *F_Destination_Address*.

INFORMATION

You can use these system variables to control the consumer/provider status, see chapter "Controlling the consumer/provider status (IOxS)".

5.9.4 PROFINET IO module and PROFIsafe module

The module parameters are used to define the communication relations of the modules and their behavior after connection is interrupted.

- "Properties" dialog

  To open the "Properties" dialog, choose [Properties] from the context menu of the modules. The "Properties" dialog contains the following tabs:

Element	Description
Name	Name of the device module.
Slot	0 to 32767.
Module ID	Unique number.
Subslot	Number of subslots.
Process data behavior	Process data value after the connection is interrupted. • Retain last valid process data • Adopt initial data
Length of IO input data	1 to 123
Length of IO output data	1 to 123

21233799/EN – 07/2014

Element	Description
PROFIsafe F_destination_address	The F_Destination_Address of the F-device must be unique within the PROFIsafe network. Range of values: 1 – 65534

- "Edit" dialog

  To open the "Edit" dialog, choose [Edit] from the context menu of the submodules.

  The "System Variables" tab provides the following system variables that allow users to evaluate the state of the submodule from within the user program.

INFORMATION

You can use these system variables to control the consumer/provider status, see chapter "Controlling the consumer/provider status (IOxS)".

Element	Description
Valid output data	• TRUE: Valid output data • FALSE: Invalid output data
Output data accepted by the controller	• TRUE: Valid output data • FALSE: Invalid output data
Valid input data	• TRUE: Valid input data • FALSE: Invalid input data
Input data accepted by the device	• TRUE: Valid input data • FALSE: Invalid input data

- The following parameters are only available for PROFIsafe modules:

21233799/EN – 07/2014

Element	Description
PROFIsafe control	With each message, PROFIsafe sends the PROFIsafe control byte from the controller to the device. The control byte can be set in the user program (see chapter "PROFIsafe control byte and status byte"). • Bit 0: iPar_EN_DC Enable from the controller lets the device load new iParameters into the device. • Bit 1: OA_Req_DC (Operator Acknowledge) Operator acknowledge from the host control byte. • Bit 2: Reset_Comm This is the Reset_Comm value read back from the host control byte. • Bit 3: activate_FV_DC FALSE: Process values are exchanged between F-host and F-device. TRUE: Failsafe values "0" are exchanged between F-host and F-device. • Bit 4: Toggle_d Toggle bit of the F-device. • Bit 5: Cons_nr_R The consecutive number is adopted whenever there is a change between two consecutive control bytes of the Toggle_d bit. This is independent of the occurrence of a fault. • Bit 6: F_ParamValid TRUE: F-parameters were set FALSE: else • Bit 7: F_Param_ConfiguredTwice TRUE: The F-device was configured with different F-parameters more than once. FALSE: else
PROFIsafe F_Par_iPar_CRC	iParameters are independent or technology-specific F-device parameters. The iPar_CRC results from the configuration of the F-device. Note: Users are responsible for setting the valid iPar_CRC after having configured the iParameters and changing to "hot" operation.
PROFIsafe F_SIL	• 0: SIL 1 • 1: SIL 2 • 2: SIL 3 • 3: No SIL

21233799/EN – 07/2014

Element	Description
PROFIsafe RoundTrip Time last	For an F-host, this is the time between sending a data message (with consecutive number N) and receiving the corresponding acknowledgement (with consecutive number N), measured in milliseconds.
PROFIsafe status	With each message, PROFIsafe sends the PROFIsafe status byte from the device to the controller. The PROFIsafe status byte can be set in the user program of the device. • Bit 0: iPar_OK_DS New iParameters received. • Bit 1: Default_Fault_DS TRUE: Device fault FALSE: No device fault Is only taken into account from the state 21 *Await Message*. • Bit 2: Reserved • Bit 3: Reserved • Bit 4: FV_activated_DS Failsafe value is active. • Bit 5: Reserved • Bit 6: • Bit 7: Reset_Comm The protocol stack is reset to its initial state.

The Process Variables tab is used to enter the process variables.

5.10 PROFINET IO function blocks

The following PROFINET IO function blocks are available in SILworX® for acyclic data exchange.

The function blocks are configured in the user program in such a way that the functions of the controller and of the devices (alarms, diagnostic data, states) can be set and read in the user program.

Function block	Description of the function
MSTAT	Controlling the function state using the user program.
RALRM	Reading the alarm messages of the IO devices.
RDREC	Reading the data records of the IO devices.
SLACT	Controlling the device states using the user program.
WRREC	Writing the data records of the IO devices.

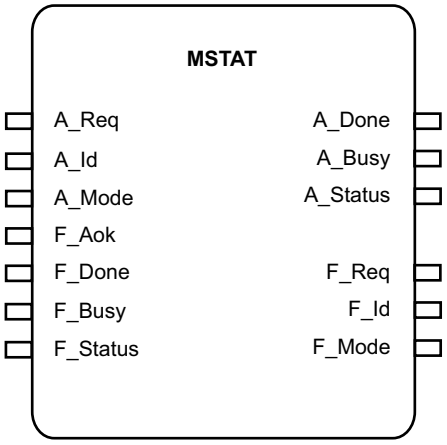
21233799/EN – 07/2014

INFORMATION

- The PROFINET IO function blocks must be integrated using the *PROFIlib.A3* library. You find the *PROFIlib.A3* library on the installation CD for SILworX®.
- For a description of how to set the parameters of the function blocks, refer to chapter "Configuring the function blocks".

5.10.1 MSTAT function block

The following figure shows the layout of the MSTAT function block with all inputs (left) and outputs (right).



7435913099

The user program uses the MSTAT function block to control the PROFINET IO controller. This means the PROFINET IO controller can be set to one of the following states using a timer or a mechanical switch connected to a physical input:

- 0: OFFLINE
- 1: OPERATE

INFORMATION

To configure the function block, drag it from the function block library and drop it in the user program (see chapter "Configuring function blocks")

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	A positive edge starts the function block	BOOLEAN
A_ID	Master ID (not used)	DWORD
A_Mode	The PROFINET IO controller can be set to the following states: • 0: OFFLINE • 1: OPERATE	INT

21233799/EN – 07/2014

A-outputs	Description	Type
A_Done	TRUE: The PROFINET IO controller was set to the state defined at the A_Mode input.	BOO-LEAN
A_Busy	TRUE: The PROFINET IO controller is still being set.	BOO-LEAN
A_Status	Status or error code (see chapter "Error codes of function blocks").	DWORD

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 the structure tree. The prefix "F" means Field.

INFORMATION

Common variables are used to connect the MSTAT function block in the structure tree (in the "Function blocks" folder) with the MSTAT function block (in the user program). These common variables must first be created in the Variable Editor.

Connect the F-inputs of the MSTAT function block in the user program to the same variables that will be connected to the outputs of the MSTAT function block in the structure tree.

F-inputs	Type
F_ACK	BOOLEAN
F_DONE	BOOLEAN
F_BUSY	BOOLEAN
F_STATUS	DWORD

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	BOOLEAN
F_ID	DWORD
F_MODE	INT

How to create the MSTAT function block in the structure tree:

1. In the structure tree, open [Configuration] / [Resource] / [Protocols] / [PROFINET IO Controller] / [Function Blocks] / [New].
2. Select the MSTAT function block.
3. Right-click the MSTAT function block and choose [Edit] from the context menu. The window for assigning variables to the function block opens.
4. 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

21233799/EN – 07/2014

Inputs	Type
MODE	INT
REQ	BOOLEAN

5. 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	BOOLEAN
BUSY	BOOLEAN
DONE	BOOLEAN
STATUS	DWORD

Using the MSTAT function block

Do the following to use the MSTAT function block:

1. In the user program, set the *A_Mode* input to the required state. If you do not set *A_Mode*, an error code is output on the *A_Status* output after step 2, and the PROFINET IO controller will not be set.
2. In the user program, set the *A_Req* input to TRUE. The function block responds to a positive edge on *A_Req*. The *A_Busy* output is TRUE until the MSTAT command has been processed. Afterwards, the *A_Busy* output is set to FALSE and the *A_Done* output to TRUE.

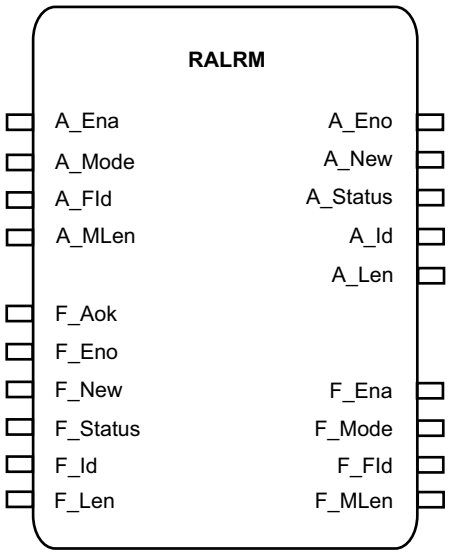
INFORMATION

If the specified mode could not be set, an error code is output on the *A_Status* output. The current controller mode is indicated by the *Controller_Status* variable (see chapter "PROFINET IO auxiliary function blocks").

21233799/EN – 07/2014

5.10.2 RALRM function block

The following figure shows the layout of the RALRM function block with all inputs (left) and outputs (right).



7436172555

The RALRM function block is used to evaluate alarms.

Alarms are a special kind of diagnostic messages that are handled with high priority. Alarms report important events to which the application must response (for example a WRREC). How the application responds, depends on the manufacturer. For more information, refer to the manual of the PROFINET IO device.

As long as the RALRM function block is active, it waits for alarm messages from the slaves. Once 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 once cycle. All alarms are acknowledged implicitly. No alarms are lost.

When using several RALRM function blocks, the user program must be configured in such a way that only one RALRM function block is active at any given time.

INFORMATION

To configure the function block, drag it from the function block library and drop it in the user program (see chapter "Configuring function blocks").

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.	BOO-LEAN
A_Mode	Not used.	INT
A_FID	Not used.	DWORD
A_MLen	Maximum expected length of the received alarm data in bytes.	INT

A-outputs	Description	Type
A_Eno	TRUE: The function block is active. FALSE: The function block is not active.	BOO-LEAN
A_New	TRUE: A new alarm was received. FALSE: No new alarm.	BOO-LEAN
A_Status	Status or error code (see chapter "Error codes of function blocks").	DWORD
A_ID	Identification number of the IO device triggering the alarm.	DWORD
A_Len	Length of the received alarm data in bytes.	INT

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 the structure tree. The prefix "F" means Field.

INFORMATION

Common variables are used to connect the RALRM function block in the structure tree (in the "Function blocks" folder) with the RALRM function block (in the user program). These common variables must first be created in 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	BOOLEAN
F_ENO	BOOLEAN
F_NEW	BOOLEAN
F_STATUS	DWORD
F_ID	DWORD
F_LEN	INT

21233799/EN – 07/2014

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	BOOLEAN
F_MODE	INT
F_FID	DWORD
F_MLEN	INT

How to create the RALRM function block in the structure tree:

1. In the structure tree, open [Configuration] / [Resource] / [Protocols] / [PROFINET IO Controller] / [Function Blocks] / [New].
2. Select the RALRM function block.
3. Right-click the RALRM function block and choose [Edit] from the context menu. The window for assigning variables to the function block opens.
4. 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	BOOLEAN
F_ID	DWORD
MLEN	INT
MODE	INT

5. 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	BOOLEAN
ENO	BOOLEAN
ID	DWORD
LEN	INT
NEW	BOOLEAN
STATUS	DWORD

Alarm data

You have to define variables in the Process Variables tab of the RALRM function block in the structure tree. The structure of these variables must match the alarm data. If no variables are defined, alarm data can be requested but cannot be read.

An alarm message contains at least 4 bytes. The first 4 bytes of the alarm messages contain the standard alarm data. To decode standard alarms, SEW-EURODRIVE provides the auxiliary function block ALARM (see chapter "Auxiliary function block ALARM").

21233799/EN – 07/2014

INFORMATION

If an alarm telegram contains more bytes than defined in the "Data" tab, then only those bytes are accepted that were defined. The rest is cut off.

Alarm data	Description
Byte 0	Length of the alarm message in bytes (4 – 126)
Byte 1	Identification for the alarm type: • 1: Diagnostic alarm • 2: Process alarm • 3: Pull alarm • 4: Plug alarm • 5: Status alarm • 6: Update alarm • 31: Failure of a controller's expansion or IO device expansion • 32 – 126: Manufacturer specific Consult the device manual provided by the manufacturer for more information on the specific meaning.
Byte 2	Slot number of the component that triggers the alarm.
Byte 3	• 0: No further information • 1: Incoming alarm, slot malfunction • 2: Outgoing alarm, slot no longer malfunctioning • 3: Outgoing alarm, continued slot malfunction
Bytes 4 to 126	Consult the device manual provided by the manufacturer for more information on the specific meaning.

INFORMATION

The structure of the standard alarms (bytes 0 to 3) is standardized and identical for all manufacturers. For information on the manufacturer-specific bytes 4 to 126, refer to the manual of the PROFINET IO device.

Do the following to use the RALRM function block:

1. In the user program, define the maximum expected amount of alarm data in bytes for the *A_Mlen* output. *A_Mlen* cannot be changed during operation.
2. In the user program, set the *A_Ena* input to TRUE. Unlike the other function blocks, the RALRM function block is only active as long as the *A_Ena* input is set to TRUE.

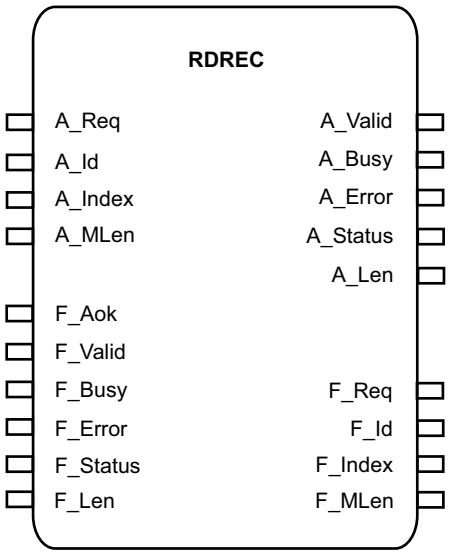
   If the function block was started successfully, then the *A_Eno* output is set to TRUE. If the function block could not be started, an error code is output on the *A_Status* output.

   If a new alarm is received, the *A_New* output is set to TRUE for at least one cycle. During this time, the outputs contain the alarm data of the IO device that has triggered the alarm. The alarm data can be evaluated. Afterwards, the *A_New* output returns to FALSE for at least one cycle. The outputs *A_ID* and *A_Len* are reset to zero before the next alarm message can be received and evaluated.

21233799/EN – 07/2014

5.10.3 RDREC function block

The following figure shows the layout of the RDREC function block with all inputs (left) and outputs (right).



7436364427

The RDREC function block is used for acyclic reading of an addressed data record of an IO device on the *A_Index* index. Refer to the operating instructions of the IO device for information on the data that can be read.

This function is optional. Up to 32 RDREC and/or WRREC function blocks can be active at the same time in the PROFINET IO controller.

INFORMATION

To configure the function block, drag it from the function block library and drop it in the user program (see chapter "Configuring function blocks").

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 positive edge starts the read request.	BOO-LEAN
A_ID	Identification number of the IO device (see chapter "PROFINET IO auxiliary function blocks").	DWORD
A_Index	Data record number of the data record to be read. Consult the device manual provided by the manufacturer for more information on the specific meaning.	INT
A_MLen	Maximum expected length of the data to be read in bytes.	INT

A-outputs	Description	Type
A_Valid	A new diagnostic message has been received and is valid.	BOO-LEAN
A_Busy	TRUE: Data is still being read.	BOO-LEAN

A-outputs	Description	Type
A_Error	TRUE: An error occurred while reading. FALSE: No error.	BOO-LEAN
A_Status	Status or error code (see chapter "Error codes of function blocks").	DWORD
A_Len	Length of the received diagnostic data in bytes.	INT

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 the structure tree. The prefix "F" means Field.

INFORMATION

Common variables are used to connect the RDREC function block in the structure tree (in the "Function blocks" folder) with the RDREC function block (in the user program). These common variables must first be created in 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	BOOLEAN
F_VALID	BOOLEAN
F_BUSY	BOOLEAN
F_ERROR	BOOLEAN
F_STATUS	DWORD
F_LEN	INT

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	BOOLEAN
F_ID	DWORD
F_INDEX	INT
F_MLEN	INT

How to create the RDREC function block in the structure tree:

1. In the structure tree, open [Configuration] / [Resource] / [Protocols] / [PROFINET IO Controller] / [Function Blocks] / [New].
2. Select the RDREC function block.
3. Right-click the RDREC function block and choose [Edit] from the context menu. The window for assigning variables to the function block opens.
4. 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.

21233799/EN – 07/2014

Inputs	Type
ID	DWORD
INDEX	INT
MLEN	INT
REQ	BOOLEAN

5. 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	BOOLEAN
BUSY	BOOLEAN
ERROR	BOOLEAN
LEN	INT
STATUS	DWORD
VALID	BOOLEAN

Data	Description
No predefined variables	A user-specific data structure can be defined in the "Process Variable" tab. However, the structure must match the data record structure. For more information on the data record structure, refer to the operating instructions provided by the manufacturer of the IO device.

Using the RDREC function block

Do the following to use the RDREC function block:

1. In the user program, set the IO device address on the *A_ID* input.
2. In the user program, set the IO device-specific index for the data record (refer to the manual provided by the manufacturer) on the *A_Index* input.
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 responds to a positive edge on *A_Req*.

The *A_Busy* output is TRUE until the data record request has been processed. Afterwards, the *A_Busy* output is set to FALSE and *A_Valid* or *A_Error* is set to TRUE.

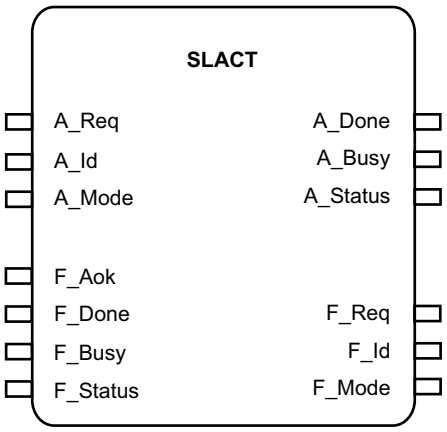
If the data record is valid, the *A_Valid* record is set to TRUE. The data record can be evaluated using the variables defined in the "Data" tab. The *A_Len* output contains the actual length of the data record that has been read.

If the data record could not be read successfully, the *A_Error* output is set to TRUE, and an error code is output on the *A_Status* output.

21233799/EN – 07/2014

5.10.4 SLACT function block

The following figure shows the layout of the SLACT function block with all inputs (left) and outputs (right).



7436368651

The SLACT function block is used for activating and deactivating an IO device from within the user program of the PROFINET IO controller. This means the IO device can be set to one of the following states using a timer or a mechanical switch connected to a physical input of the PROFINET IO controller.

- ≠ 0: Active
- 0: Not active

When using several SLACT function blocks, the user program must be configured in such a way that only one SLACT function block is active at any given time.

INFORMATION

To configure the function block, drag it from the function block library and drop it in the user program (see chapter "Configuring function blocks")

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	A positive edge starts the function block.	BOO-LEAN
A_ID	Identification number of the IO device (see chapter "PROFINET IO auxiliary function blocks").	DWORD
A_Mode	Target state for the PROFIBUS IO device: • ≠ 0: Active (connected) • 0: Not active (deactivated)	INT

A-outputs	Description	Type
A_Done	TRUE: The PROFINET IO device was set to the state defined at the *A_Mode* input.	BOO-LEAN
A_Busy	TRUE: The PROFINET IO device is still being set.	BOO-LEAN

21233799/EN – 07/2014

A-outputs	Description	Type
A_Status	Status or error code (see chapter "Error codes of function blocks").	DWORD

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 the structure tree. The prefix "F" means Field.

INFORMATION

Common variables are used to connect the SLACT function block in the structure tree (in the "Function blocks" folder) with the SLACT function block (in the user program). These common variables must first be created in 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	BOOLEAN
F_DONE	BOOLEAN
F_BUSY	BOOLEAN
F_STATUS	DWORD

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	BOOLEAN
F_ID	DWORD
F_MODE	INT

How to create the SLACT function block in the structure tree:

1. In the structure tree, open [Configuration] / [Resource] / [Protocols] / [PROFINET IO Controller] / [Function Blocks] / [New].
2. Select the SLACT function block.
3. Right-click the SLACT function block and choose [Edit] from the context menu. The window for assigning variables to the function block opens.
4. 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	BOOLEAN

21233799/EN – 07/2014

5. 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	BOOLEAN
BUSY	BOOLEAN
DONE	BOOLEAN
STATUS	DWORD

Using the SLACT function block

Do the following to use the SLACT function block:

1. In the user program, set the *A_Mode* input to the required state.
2. In the user program, set the IO device identifier on the *A_ID* input.
3. In the user program, set the *A_Req* input to TRUE. The function block responds to a positive edge on *A_Req*.

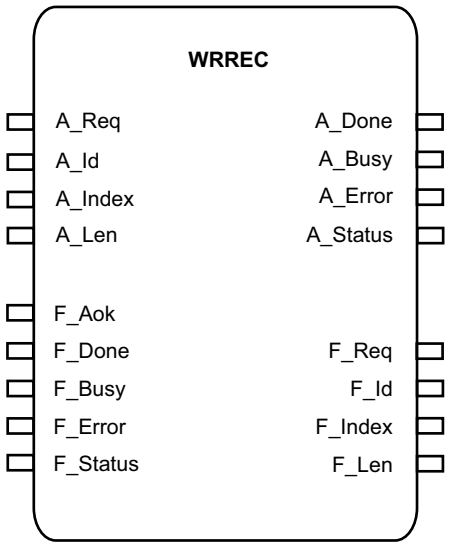
The *A_Busy* output is TRUE until the SLACT command has been processed. Afterwards, the outputs *A_Busy* are set to FALSE and *A_Done* to TRUE.

If the slave mode could be set, the slave mode is output on *A_Status*.

If the slave mode could not be set, an error code is output on *A_Status*.

5.10.5 WRREC function block

The following figure shows the layout of the WRREC function block with all inputs (left) and outputs (right).



7436372875

The WRREC function block is used for acyclic writing of a data record to an IO device addressed with *A_Index*. For information on the data that can be written, refer to the operating instructions for the IO device.

Up to 32 RDREC and/or WRREC function blocks can be active at the same time in the PROFINET IO controller.

21233799/EN – 07/2014

INFORMATION

To configure the function block, drag it from the function block library and drop it in the user program (see chapter "Configuring function blocks")

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 positive edge starts the request for writing a data record.	BOOLEAN
A_ID	Identification number of the IO device (see chapter "PROFINET IO auxiliary function blocks").	DWORD
A_Index	Data record number of the data record to be written. 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

A-outputs	Description	Type
A_Done	TRUE: The PROFINET IO device was set to the state defined at the *A_Mode* input.	BOOLEAN
A_Busy	TRUE: The PROFINET IO device is still being set.	BOOLEAN
A_Status	Status or error code (see chapter "Error codes of function blocks").	DWORD
A_Status	Status or error code (see chapter "Error codes of function blocks").	DWORD

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 the structure tree. The prefix "F" means Field.

INFORMATION

Common variables are used to connect the WRREC function block in the structure tree (in the "Function blocks" folder) with the WRREC function block (in the user program). These common variables must first be created in 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 outputs of the WRREC function block in the structure tree.

F-inputs	Type
F_ACK	BOOLEAN
F_DONE	BOOLEAN
F_ERROR	BOOLEAN
F_BUSY	BOOLEAN

21233799/EN – 07/2014

F-inputs	Type
F_STATUS	DWORD

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	BOOLEAN
F_ID	DWORD
F_INDEX	INT
F_LEN	INT

How to create the WRREC function block in the structure tree:

1. In the structure tree, open [Configuration] / [Resource] / [Protocols] / [PROFINET IO Controller] / [Function Blocks] / [New].
2. Select the WRREC function block.
3. Right-click the WRREC function block and choose [Edit] from the context menu. The window for assigning variables to the function block opens.
4. 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	BOOLEAN

5. 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	BOOLEAN
BUSY	BOOLEAN
DONE	BOOLEAN
ERROR	BOOLEAN
STATUS	DWORD

Data	Description
No predefined variables	A user-specific data structure can be defined in the "Process Variable" tab. However, the structure must match the data record structure. For more information on the data record structure, refer to the operating instructions provided by the manufacturer of the IO device.

21233799/EN – 07/2014

Using the WRREC function block

Do the following to use the WRREC function block:

1. In the user program, set the IO device address on the *A_ID* input.
2. In the user program, set the IO device-specific index for the data record (refer to the manual provided by the manufacturer) on the *A_Index* input.
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 responds to a positive edge on *A_Req*.

The *A_Busy* output is TRUE until the data record has been written. Afterwards, the outputs *A_Busy* are set to FALSE and *A_Done* 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 on the *A_Status* output.

21233799/EN – 07/2014

5.11 Configuring function blocks

The fieldbus protocols and the associated function blocks run on the COM module of the safety controller. This is the reason why the function blocks must be created in the SILworX® structure tree under [Configuration] / [Resource] / [Protocols...].

To being able to control the function blocks on the COM module, you can create function blocks in the user program of SILworX® that can be used like standard function blocks.

Common variables are used to connect the function blocks in the user program of SILworX® to the corresponding function blocks in the structure tree of SILworX®. These common variables must first be created in the Variable Editor.

5.11.1 Adding function block libraries

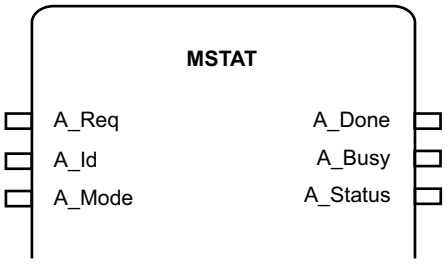
To add the function block libraries for PROFINET and Send & Receive TCP (PROFIlib.A3 and *TCPlib.A3*) to the project, choose "Restore..." from the context menu of the project. You find the function block libraries on the installation CD for SILworX® under "Binaries/Communic-Libs". From there, you can insert the libraries using the "Restore" function.

5.11.2 Configuring function blocks in the user program

You can use drag and drop to add the required function blocks to the user program. Configure the inputs and outputs as described for the individual function block.

Upper part of the function block

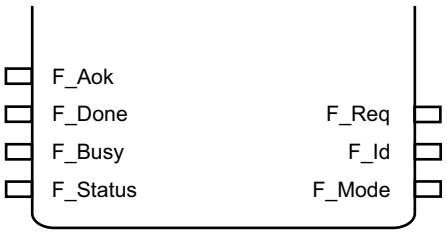
The upper part of the function block corresponds to the user interface that is used by the user program to control the function block. This is where the variables in the user program are connected. The prefix "A" means Application.



7441757323

Lower part of the function block

The lower part of the function block represents the connection to the function block (in the SILworX® structure tree). This is where the variables that must be connected to the function block in the SILworX® structure tree are connected. The prefix "F" means Field.



7441760779

21233799/EN – 07/2014

5.11.3 Configuring the function blocks in the structure tree of SILworX®

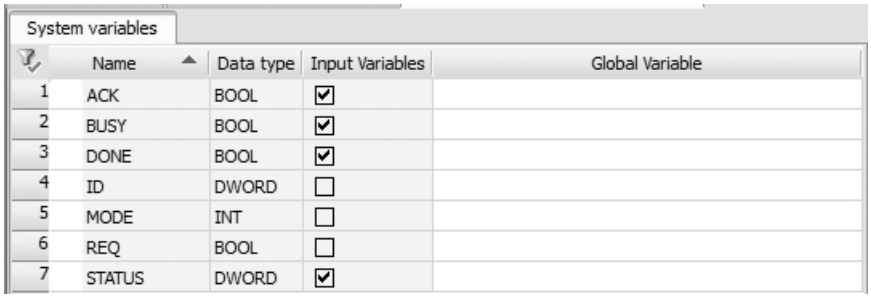
Do the following to configure the function blocks in the structure tree of SILworX®:

1. In the structure tree, choose [Configuration] / [Resource] / [Protocols].
2. Right-click "Function Blocks" and choose [New] from the context menu.
3. In the SILworX® structure tree, select the suitable function block.

12843489675

The inputs of the function block (checkmark in the "Input Variables" column) must be connected to the same variables that are connected to the F-outputs of the function block in the user program.

The outputs of the function block (no checkmark in the "Input Variables" column) must be connected to the same variables that are connected to the F-inputs of the function block in the user program.



	Name	Data type	Input Variables	Global Variable
1	ACK	BOOL	☑	
2	BUSY	BOOL	☑	
3	DONE	BOOL	☑	
4	ID	DWORD	☐	
5	MODE	INT	☐	
6	REQ	BOOL	☐	
7	STATUS	DWORD	☑	

12845617035

5.12 PROFINET IO auxiliary function blocks

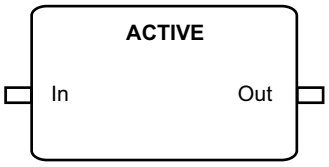
Auxiliary function blocks are used to configure and evaluate the inputs and outputs of the function blocks. The following auxiliary function blocks are available:

Auxiliary function block	Description of the function
ACTIVE	Checks whether the PROFINET IO device is active or inactive.
ALARM	Decodes alarm data.
DEID	Decodes the identification number.
ID	Generates a 4-byte identifier.

Auxiliary function block	Description of the function
NSLOT	Creates a continuous identification number for the slots.
SLOT	Creates a SLOT identification number using a slot number.
STDDIAG	Decodes the standard diagnosis of a slave.
LATCH	Is used only within other function blocks.
PIG	Is used only within other function blocks.
PIGII	Is used only within other function blocks.

5.12.1 ACTIVE auxiliary function block

The ACTIVE auxiliary function block uses the standard diagnosis of a PROFINET IO device to determine whether the slave is active or inactive.



7441573387

INFORMATION

To configure the function block, drag it from the function block library and drop it in the user program (see chapter "Configuring function blocks")

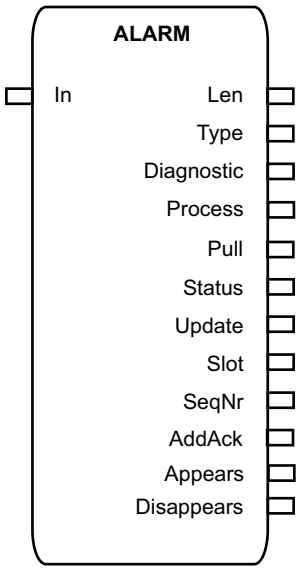
Input	Description	Type
IN	Standard diagnosis of the slave.	DWORD

Output	Description	Type
OUT	TRUE: The slave is active. FALSE: The slave is inactive.	BOOLEAN

21233799/EN – 07/2014

5.12.2 ALARM auxiliary function block

The ALARM auxiliary function block decodes the standard alarm data of a PROFINET IO device.



7440806667

INFORMATION

To configure the function block, drag it from the function block library and drop it in the user program (see chapter "Configuring function blocks")

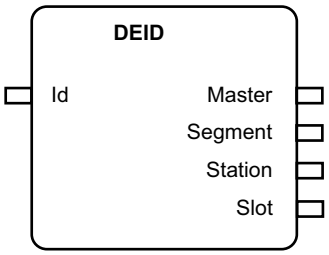
Input	Description	Type
IN	Standard alarm	DWORD

Outputs	Description	Type
LEN	Total length of the alarm message.	SINT
Type	1: Diagnostic alarm 2: Process alarm 3: Pull alarm 4: Plug alarm 5: Status alarm 6: Update alarm 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
Diagnostics	True = Diagnostic alarm	BOOLEAN
Process	True = Process alarm	BOOLEAN
Pull	True = The module was pulled	BOOLEAN
Plug	True = The module was plugged	BOOLEAN

Outputs	Description			Type
Status	True = Status alarm			BOO-LEAN
Update	True = Update alarm			BOO-LEAN
Slot	Module that triggers the alarm			BYTE
SeqNo	Alarm sequence number			SINT
AddAck	TRUE means that the slave that triggered this alarm expects an additional acknowledgement from the application. For more information, refer to the slave manual of the manufacturer.			BOO-LEAN
Appears Disap-pears	Output	Value	Description	BOO-LEAN
	Appears	TRUE	If both are FALSE, no error has occurred yet.	BOO-LEAN
	Disap-pears	FALSE		
	Appears	TRUE	An error occurred and is still pending.	
	Disap-pears	FALSE		
	Appears	TRUE	An error occurred and is disappearing.	
	Disap-pears	FALSE		
	Appears	TRUE	If both are TRUE, the error disappears but the slave is still faulty.	
	Disap-pears	FALSE		

5.12.3 DEID auxiliary function block

The DEID auxiliary function block decodes the identification number and splits it into its 4 parts.



7440815371

INFORMATION

To configure the function block, drag it from the function block library and drop it in the user program (see chapter "Configuring function blocks")

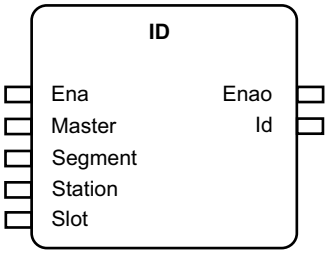
Input	Description	Type
ID	Identification number of the slave	DWORD

21233799/EN – 07/2014

Outputs	Description	Type
Master	Master bus address	BYTE
Segment	Segment	BYTE
Station	Slave bus address	BYTE
Slot	Slot or module number	BYTE

5.12.4 ID auxiliary function block

The ID auxiliary function block uses 4 bytes to generate an identifier (identification number) that is used by other auxiliary function blocks.



7441561227

INFORMATION

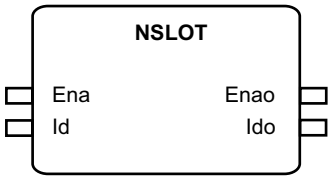
To configure the function block, drag it from the function block library and drop it in the user program (see chapter "Configuring function blocks")

Inputs	Description	Type
Ena	Not used	BOO-LEAN
Master	Bus address	BYTE
Segment	Segment	BYTE
Station	Slave bus address	BYTE
Slot	Slot or module number	BYTE

Outputs	Description	Type
Enao	Not used	BOO-LEAN
ID	Identification number of the slave	DWORD

5.12.5 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. The *Ena* input must be set to TRUE to have the auxiliary function block run. The *Enao* output is set to TRUE when the result on the *Ido* output is valid.



7441569931

INFORMATION

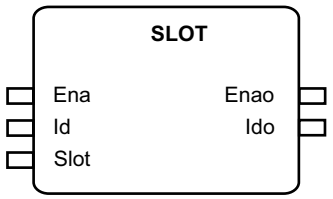
To configure the function block, drag it from the function block library and drop it in the user program (see chapter "Configuring function blocks")

Inputs	Description	Type
Ena	The auxiliary function block is running as long as Ena is set to TRUE.	BOO-LEAN
ID	Identification number of the slave	DWORD

Outputs	Description	Type
Enao	TRUE = The result is valid FALSE = No further slot numbers	BOO-LEAN
Ido	Identification number of the slave	DWORD

5.12.6 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.



7441564683

INFORMATION

To configure the function block, drag it from the function block library and drop it in the user program (see chapter "Configuring function blocks")

Inputs	Description	Type
Ena	Not used	BOO-LEAN
ID	Logical address of the slave component (slave ID and slot number)	DWORD

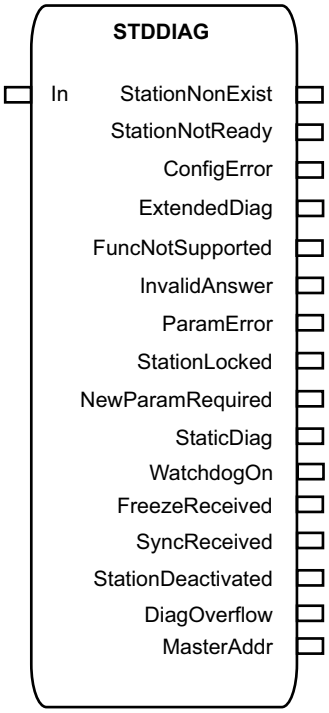
21233799/EN – 07/2014

Inputs	Description	Type
Slot	New slot or module number	BYTE

Outputs	Description	Type
Enao	Not used	BOO-LEAN
Ido	Identification number of the slave	DWORD

5.12.7 STDDIAG auxiliary function block

The STDDIAG auxiliary function block decodes the standard diagnosis of a PROFINET IO device. The outputs of the type BOOLEAN of the STDDIAG function block are set to TRUE if the associated bit is set in the standard diagnosis.



7440811915

INFORMATION

To configure the function block, drag it from the function block library and drop it in the user program (see chapter "Configuring function blocks")

Input	Description	Type
IN	Standard diagnostics of the slave	DWORD

Outputs	Description	Type
StationNonExist	The slave does not exist	BOO-LEAN
StationNotReady	Slave not ready	BOO-LEAN
ConfigError	Configuration error	BOO-LEAN

Outputs	Description	Type
ExtendedDiag	Extended diagnostics follows	BOO-LEAN
FuncNotSupported	The function is not supported	BOO-LEAN
InvalidAnswer	Invalid reply from slave	BOO-LEAN
ParamError	Parameter error	BOO-LEAN
StationLocked	The slave is locked by another master	BOO-LEAN
NewParamRequired	New configuration data is required	BOO-LEAN
StaticDiag	Static diagnostics	BOO-LEAN
WatchdogOn	The watchdog is active	BOO-LEAN
FreezeReceived	Freeze command received	BOO-LEAN
SyncReceived	Sync command received	BOO-LEAN
StationDeactivated	The slave was deactivated	BOO-LEAN
DiagOverflow	Diagnostic overflow	BOO-LEAN
MasterAdd	Bus address of the master	BYTE

Do the following to read the standard diagnosis of the PROFINET IO device:

1. In the structure tree, choose [Configuration] / [Resource] / [Protocols] / [PROFINET IO Controller].
2. Right-click "PROFINET IO Device" and choose [Edit] from the context menu.
3. Drag the global variable of the type DWORD onto the *Standard Diagnostics* field. In the SILworX® structure tree, select the suitable function block.
4. Connect the global variable with the input of the STDDIAG function block.

5.13 Function block error codes

If a function block is unable to properly execute a command, an error code is issued on the *A_Status* output. The meaning of the error codes is described in the following table:

Error code	Symbol	Description
16#40800800	TEMP_NOT_AVAIL	Service temporarily not available
16#40801000	INVALID_PARA	Invalid parameter
16#40801100	WRONG_STATE	Slave does not support PROFINET

21233799/EN – 07/2014

Error code	Symbol	Description
16#40808000	FATAL_ERR	Fatal program error
16#40808100	BAD_CONFIG	Configuration error in data area
16#40808200	PLC_STOPPED	Controller was stopped
16#4080A000	READ_ERR	Error while reading record
16#4080A100	WRITE_ERR	Error while writing record
16#4080A200	MODULE_FAILURE	Error cannot be specified in detail
16#4080B000	INVALID_INDEX	Index not valid
16#4080B100	WRITE_LENGTH	Wrong length while writing
16#4080B200	INVALID_SLOT	Slot number not valid
16#4080B300	TYPE_CONFLICT	Wrong type
16#4080B400	INVALID_AREA	Invalid read or write area
16#4080B500	STATE_CONFLICT	Master in invalid state
16#4080B600	ACCESS_DENIED	Slave not active (or similar)
16#4080B700	INVALID_RANGE	Invalid read or 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 restriction
16#4080C100	WRITE_CONSTRAIN	Write restriction
16#4080C200	BUSY	A function block of this type is already active
16#4080C300	NO_RESOURCE	Slave not active

21233799/EN – 07/2014

6 Modbus TCP/UDP

6.1 Modbus master

Data is transferred between Modbus master and Modbus slaves via TCP/UDP (Ethernet).

The following table shows the properties of the Modbus master:

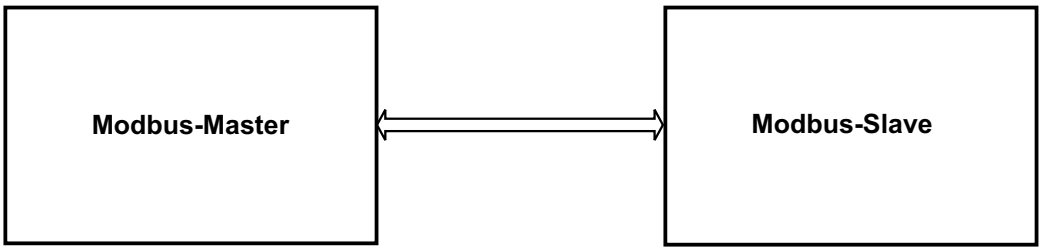
Property	Description
Modbus master	One Modbus master can be configured for each COM module or controller. The Modbus master can simultaneously exchange data with the TCP/UDP slaves.
Max. number of Modbus slaves	One Modbus master can operate up to 247 slaves. • 64 TCP slaves via TCP/IP connection • 247 UDP slaves via UDP/IP connection The maximum number of UDP slaves is limited because 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 per request telegram	The process data length for SEW-specific request telegrams is 1100 bytes (see chapter "SEW-specific function codes").
Max. size of send data	64 kB send 64 kB receive Note: The status byte of the master and the status bytes of each associated slave must be subtracted from the max. size of the send data.
Max. size of receive data	
Modbus data display format	The safety controller uses big endian format. Example: 32-bit data (for example DWORD, DINT):

32-bit data (hex)	0x12345678			
Memory offset	0	1	2	3
Big endian	12	34	56	78
Middle endian	56	78	12	34
Little endian	78	56	34	12

21233799/EN – 07/2014

6.1.1 Example of a Modbus

In this example, the Modbus master exchanges data with a Modbus slave via Modbus TCP. Both controllers are connected via the Ethernet interfaces of the communication modules.



5452807563

INFORMATION

If Modbus master and Modbus slave are located in different subnets, then the routing table must contain the corresponding user-defined routes.

For this example, you have to create the following global variables in SILworX®.

Global variables	Type
Master->Slave_BOOL_00	BOOLEAN
Master->Slave_BOOL_01	BOOLEAN
Master->Slave_BOOL_02	BOOLEAN
Master->Slave_WORD_00	WORD
Master->Slave_WORD_01	WORD
Slave->Master_WORD_00	WORD
Slave->Master_WORD_01	WORD

Configuring the Modbus TCP slave

Do the following to create a new Modbus slave:

1. In the structure tree, open [Configuration] / [Resource] / [Protocols].
2. To add a new Modbus slave set, open the context menu of Protocols and choose [New] / [Modbus Slave Set].
3. In the context menu of the Modbus Slave Set, choose [Edit] and [Modbus Slave Set Properties], and retain the default values.
4. Select the "Modbus Slave" tab and make the following settings:
   - Select [COM Module]
   - Activate [Enable TCP]
   - The remaining parameters retain the default values.

Configuring the bit input variables of the Modbus slave

INFORMATION

In the "Bit Variables" tab, enter the Boolean variables that the master addresses bit-by-bit (function codes 1, 2, 5, 15).

Do the following to configure the bit input variables of the Modbus slave:

1. From the context menu of the Modbus slave, choose [Edit] and then [Bit Variables].
2. In the object selection, select the following global variables and drag them to the "Bit Inputs" area.

Bit address	Bit variable	Type
0	Master->Slave_BOOL_00	BOOLEAN
1	Master->Slave_BOOL_01	BOOLEAN
2	Master->Slave_BOOL_02	BOOLEAN

3. Right-click anywhere in the "Open Inputs" tab and choose "New Offsets" from the context menu to renumber the variable offsets.

Configuring the register output variables of the Modbus slave

Do the following to configure the register output variables of the Modbus slave:

INFORMATION

In the "Register Variables" tab, enter the variables that the master addresses register-by-register (function codes 3, 4, 6, 16, 23).

1. From the context menu of the Modbus slave, choose [Edit] and then [Register Variables].
2. In the object selection, select the following variables and drag them to the "Register Inputs" area.

Bit 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 choose [New Offsets] from the context menu to renumber the variable offsets.

Configuring the register output variables of the Modbus slave

Do the following to configure the register output variables of the Modbus slave:

INFORMATION

In the "Register Variables" tab, enter the variables that the master addresses register-by-register (function codes 3, 4, 6, 16, 23).

1. From the context menu of the Modbus slave, choose [Edit] and then [Register Variables].
2. In the object selection, select the following variables and drag them to the "Register Outputs" area.

Bit address	Register variables	Type
0	Slave → Master_WORD_00	WORD
1	Slave → Master_WORD_01	WORD

3. Right-click anywhere in the "Register Outputs" tab and choose [New Offsets] from the context menu to renumber the variable offsets.

21233799/EN – 07/2014

Checking the Modbus TCP slave configuration

Do the following to check the Modbus TCP slave configuration:

1. Open the context menu for the Modbus TCP slave and choose [Verification].
2. Thoroughly check the messages in the logbook and correct any errors.

Configuring the Modbus TCP master

Do the following to create the Modbus master:

1. In the structure tree, open [Configuration] / [Resource] / [Protocols].
2. Select [New]/[Modbus master] in the context menu of Protocols to add a new Modbus master.
3. From the context menu of the Modbus master, choose [Properties] and then [General].
4. Select [COM module].

   The remaining parameters retain the default values.

Establishing a connection from Modbus master to Modbus TCP slave

Do the following to establish the connection to the Modbus TCP slave in the Modbus master:

1. Open [Resource]/[Protocols]/[Modbus master]/[Ethernet slaves] in the structure tree.
2. Right-click [Ethernet slaves] and choose [New] from the context menu.
3. Choose "TCP/UDP slave" from the list and confirm with [OK].
4. Do the following to configure the TCP/UDP slave in the Modbus master:
   - Choose [Edit] to assign the system variables, see chapter "System variables gateway slave".
   - Choose [Properties] to configure the properties, see chapter "Properties of gateway slave".

     Enter the IP address of the TCP/UDP slave in the properties of the slave.

     The remaining parameters retain the default values.

Configuring the request telegram for writing bit output variables

Do the following to configure the request telegram for writing the bit output variables:

1. Right-click TCP/UDP slave and choose [New] from the context menu.
2. From the list of the request telegram, choose "Write Multiple Coils (15)".
3. Right-click the request telegram "Write Multiple Coils (15)" and choose [Properties] from the context menu.

   Enter "0" as the start address for the write area.
4. Right-click the request telegram "Write Multiple Coils (15)" and choose [Edit] from the context menu.
5. In the object selection, select the following variables and drag them to the "Output Variables" tab.

Offset	Bit variable	Type
0	Master->Slave_BOOL_00	BOOLEAN

21233799/EN – 07/2014

Offset	Bit variable	Type
1	Master->Slave_BOOL_01	BOOLEAN
2	Master->Slave_BOOL_02	BOOLEAN

6. Right-click anywhere in the "Output Variables" area and choose [New Offsets] from the context menu to renumber the variable offsets.

Configuring the request telegram for writing register output variables

Do the following to configure the request telegram for writing the register output variables:

1. Right-click TCP/UDP slave and choose [New] from the context menu.
2. From the list of the request telegram, choose "Write Multiple Registers (16)".
3. Right-click the request telegram "Write Multiple Registers (16)" and choose [Properties] from the context menu.

   Enter "0" as the start address for the read area.
4. Right-click the request telegram "Write Multiple Registers (16)" and choose [Edit] from the context menu.
5. In the object selection, select the following variables and drag them to 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 and choose [New Offsets] from the context menu to renumber the variable offsets.

Defining the request telegram in the Modbus master for reading the input variables

Do the following to define the request telegram in the Modbus master for reading the input variables:

1. Right-click TCP/UDP slave and choose [New] from the context menu.
2. From the list of the request telegram, choose "Read Holding Register (03)".
3. Right-click the request telegram "Read Holding Register (03)" and choose [Properties] from the context menu.

   Enter "0" as the start address for the read area.
4. Right-click the request telegram "Read Holding Register (03)" and choose [Edit] from the context menu.
5. In the object selection, select the following variables and drag them to 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 and choose [New Offsets] from the context menu to renumber the variable offsets.

21233799/EN – 07/2014

Checking the Modbus master configuration

Do the following to check the Modbus TCP master configuration:

1. Open the context menu for the Modbus TCP master and choose [Verification].
2. Thoroughly check the messages in the logbook and correct any errors.

Creating codes for controllers

Do the following to create the codes for the controllers:

1. Start the code generator for the master and slave resources.
2. Make sure that the codes were generated without any errors.
3. Load the codes into the respective master and slave controllers.

6.1.2 Example of alternative register/bit addressing

In this example, the configuration of chapter "Modbus example" is expanded by 16 Boolean variables in the register area. The 16 Boolean variables are read using the Write Multiple Coils (15) request telegram.

Configuring input variables in the Modbus slave

Do the following to configure the input variables in the Modbus slave:

1. From the context menu of the Modbus slave, choose [Edit] and then [Register Variables].
2. In the object selection, select the 16 new Boolean variables and drag them into the "Register Inputs" area.

Register address	Register variables	Type
0	Master -> Slave_WORD_00	WORD
1	Master -> Slave_WORD_01	WORD
2	Master -> Slave_WORD_03 to _18	BOOLEAN

3. Make a right mouse click anywhere in the "Register Inputs" area and choose [New Offsets] from the context menu to renumber the offsets of the variables.

Configuring alternative register/bit addressing in the Modbus slave

Do the following to configure the alternative register/bit addressing in the Modbus slave:

- From the context menu of the Modbus, choose [Edit] and [Offset]. Next, activate "Use Alternative Register/Bit Addressing".
- For 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

21233799/EN – 07/2014

INFORMATION

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.

Configuring the request telegram for writing output variables

Do the following to configure the request telegram for writing the output variables (BOOLEAN) in the Modbus master:

1. Right-click TCP/UDP slave and choose [New] from the context menu.
2. From the list of the request telegram, choose "Write Multiple Coils (15)".
3. Right-click the request telegram "Write Multiple Coils (15)" and choose [Properties] from the context menu.

   Enter "8032" as the start address for the write area.
4. Right-click the request telegram "Write Multiple Coils (15)" and choose [Edit] from the context menu.
5. In the object selection, select the following variables and drag them to the "Output Variables" tab.

Offset	Mirrored register variable	Type
0 to 15	Master->Slave_BOOL_03..._18	BOOLEAN

6. Right-click anywhere in the "Output Variables" area and choose [New Offsets] from the context menu to renumber the variable offsets.

6.1.3 Menu functions of the Modbus master

Edit

The Modbus "Edit" dialog of 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 to analyze the state from within the user program.

Element	Description
Slave connection error count	Number of faulty connections with Modbus slaves that are in activated state. Deactivated Modbus slaves are not taken into account.
Modbus master activation control	To stop or start the Modbus master from within the user program. 0: Activate 1: Deactivate (edge-triggered. The Modbus master can be activated using the PADT even if Modbus master activation control is set to 1)
Modbus master bus error	Bus error, for example message error (unknown codes, etc.), length error.
Modbus master state	The Modbus master indicates the present protocol state: 1: OPERATE 0: OFFLINE

21233799/EN – 07/2014

Element	Description
Reset all slave errors	A change from FALSE to TRUE resets all slave and bus errors.

Properties

To open the "Properties" dialog, choose [Properties] from the context menu of the Modbus master.

The dialog 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 UDP gateway.

Parameter	Description
Type	Modbus master.
Name	Name for the Modbus master.
Module	Selection of the COM module within which the protocol is processed.
Activate max. µP budget	**Activated:** Adopt the µP budget limit from the *Max. µP budget* in [%] field. **Deactivated:** Do not use the µP budget limit for this protocol.
Max. µP budget in [%]	Maximum µP load for the module that can be used for processing the protocols. Range of values: 1 – 100% Default: 30%
Behavior on CPU/COM connection loss	If the connection between the processor module and the communication module is lost, the input variables are initialized or continue to be used unchanged in the processor module, based on this parameter. (For example, if the communication module is disconnected while communication is in progress.) **Adopt initial data:** Input variables are reset to their initial values. **Retain last value**: Input variables retain their last values.
Enable TCP gateway	This function may not be enabled because the RS485 interface is being used by the Com user task.
TCP server port	Default: 502 Other TCP ports can also be configured. In this case, observe the port assignment provided by the *Internet Corporation for Assigned Names and Numbers (ICANN)*.
Maximum number of TCP connections operating as server	Maximum number of TCP connections opened simultaneously and operating as server. Range of values: 1 – 64 Default: 5
Enable UDP gateway	This function may not be enabled because the RS485 interface is being used by the Com user task.

Parameter	Description
UDP port	Default: 502 Other TCP ports can also be configured. In this case, observe the port assignment provided by the *Internet Corporation for Assigned Names and Numbers (ICANN)*.
Maximum queue length	Length of the gateway queue for unanswered request messages from other masters. This option is only taken into account if a gateway has been activated. Range of values: 1 – 20 Default: 3

CPU/COM

The default values for the parameters provide the fastest possible exchange of Modbus data between the COM module and the CPU module in the safety controller.

These parameters should only be changed if a reduction of the COM and/or CPU load is required for an application, and the process allows this change.

INFORMATION

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 increased.

- The system time requirements must be checked.

Parameter	Description
Process data refresh rate [ms]	Refresh rate in milliseconds during which COM and CPU exchange protocol data. If the process data refresh rate is zero or lower than the cycle time for the controller, then data is exchanged as fast as possible. Range of values: 0 – (2^{31}−1) Default: 0
Force process data consistency	**Activated:** All the protocol data is transferred from the CPU to the COM within a CPU cycle. **Deactivated:** The entire data of the protocol is transferred from the CPU to the COM and distributed via several CPU cycles with 1100 bytes each per data direction. This might also reduce the cycle time of the controller. Default: Activated

6.1.4 Modbus function codes of the master

With the Modbus function codes (request messages), you can write or read variables in both directions. Individual variables or several consecutive variables can be read or written.

Do the following to create a new request telegram for a TCP/UDP slave:

1. In the structure tree, open [Resource]/[Protocols]/[Modbus master]/[Ethernet Slaves] and select a TCP/UDP slave.
2. Right-click TCP/UDP slave and choose [New] from the context menu.
3. Choose a request telegram from the “New object” dialog.

21233799/EN – 07/2014

Modbus default function codes

The Modbus master supports the following Modbus standard function codes:

Element	Code	Type	Meaning
READ COILS	01	BOOLEAN	Read multiple variables (BOOLEAN) from the slave.
READ DISCRETE INPUTS	02	BOOLEAN	Read multiple variables (BOOLEAN) 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	BOOLEAN	Write one single variable (BOOLEAN) to the slave.
WRITE SINGLE REGISTER	06	WORD	Write one single variable (WORD) to the slave.
WRITE MULTIPLE COILS	15	BOOLEAN	Write multiple variables (BOOLEAN) to the slave.
WRITE MULTIPLE REGISTERS	16	WORD	Write multiple variables of any type to the slave.
READ WRITE HOLDING REGISTERS	23	WORD	Write and read multiple variables of any type to and from the slave.

INFORMATION

For more information on Modbus, refer to the Modbus Application Protocol Specification at *www.modbus.org.*

SEW-EURODRIVE-specific function codes

The SEW-EURODRIVE-specific function codes correspond to the standard Modbus function codes. The only differences are the maximum allowable process data length of 1100 bytes and the format of request and response headers.

Element	Code	Type	Meaning
Read Coils Extended	100 (0x64)	BOOLEAN	Corresponds to function code 01. Read multiple variables (BOOLEAN) from the import or export area of the slave. Maximum length of the process data: 1100 bytes
Read Discrete Inputs Extended	101 (0x65)	BOOLEAN	Correspond to function code 02. Read multiple variables (BOOLEAN) from the import or export area of the slave. Maximum length of the process data: 1100 bytes

21233799/EN – 07/2014

Element	Code	Type	Meaning
Read Holding Registers Extended	102 (0x66)	WORD	Corresponds to function code 03. Read multiple variables (BOOLEAN) from the import or export area of the slave. Maximum length of the process data: 1100 bytes
Read Input Registers Extended	103 (0x67)	WORD	Corresponds to function code 04. Read multiple variables (BOOLEAN) from the import or export area of the slave. Maximum length of the process data: 1100 bytes
Write Multiple Coils Extended	104 (0x68)	BOOLEAN	Corresponds to function code 15. Writing multiple variables (BOOL) to the import are of the slave. Maximum length of the process data: 1100 bytes
Write Multiple Registers Extended	105 (0x69)	WORD	Corresponds to function code 16. Write multiple variables (BOOL) to the import area of the slave. Maximum length of the process data: 1100 bytes
Read/Write Multiple Registers Extended	106 (0x6A)	WORD	Corresponds to function code 23. Write and read multiple variables of any type to and from the import or export area of the slave. Maximum length of the process data: 1100 bytes (request telegram from the Modbus master) 1100 bytes (response to the master).

6.1.5 Format of the request and response headers

The request and response headers of the SEW-specific Modbus function code are structured as follows:

Code	Request	Response
100 (0x64)	1 byte function code 0x64 2 bytes start address 2 bytes number of coils 1 – 8800 (0x2260)	1 byte function code 0x64 2 bytes number of bytes = N N bytes coil data (8 coils are packed into one byte)
101 (0x65)	1 byte function code 0x65 2 bytes start address 2 bytes number of discrete inputs 1 – 8800 (0x2260)	1 byte function code 0x65 2 bytes number of bytes = N N bytes discrete inputs data (8 discrete inputs are packed into one byte)

21233799/EN – 07/2014

Code	Request	Response
102 (0x66)	1 byte function code 0x66 2 bytes start address 2 bytes number of registers 1 – 550 (0x226)	1 byte function code 0x66 2 bytes number of bytes = N N bytes register data
103 (0x67)	1 byte function code 0x67 2 bytes start address 2 bytes number of registers 1 – 550 (0x226)	1 byte function code 0x67 2 bytes number of bytes = N N bytes register data
104 (0x68)	1 byte function code 0x68 2 bytes start address 2 bytes number of coils 1 – 8800 (0x2260) 2 bytes number of bytes = N N bytes coil data	1 byte function code 0x68 2 bytes start address 2 bytes number of coils 1 – 8800 (0x2260)
105 (0x69)	1 byte function code 0x69 2 bytes start address 2 bytes number of registers 1 – 550 (0x226) 2 bytes number of bytes = N N bytes register data	1 byte function code 0x69 2 bytes start address 2 bytes number of registers 1 – 550 (0x226)
106 (0x6A)	1 byte function code 0x6a 2 bytes read start address 2 bytes number of read registers 1 – 550 (0x226) 2 bytes write start address 2 bytes number of write registers 1 – 550 (0x226) 2 bytes number of write bytes = N N bytes register data	1 byte function code 0x6a 2 bytes number of bytes = N N bytes register data

6.1.6 Read request telegrams

Variables can be read from the slave with the read function codes. In addition to the Modbus function, the request telegram of a Modbus master 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 by sending back a response telegram with the required variables.

Do the following to configure a read request telegram:

1. In the structure tree, select the [Request Telegram] you want to configure.
2. Right-click "Request Telegram" and choose [Edit] from the context menu.

21233799/EN – 07/2014

3. In the object selection, select the global variable to be used as Modbus receive variable and drag it anywhere in the Input Signals area.
4. Repeat this step for each additional Modbus receive variable.
5. Open the context menu by right-clicking anywhere in the "Input signals" area and choose [New offsets] to renumber the offsets of the variables.

The following read request telegrams are available:

Read Coils (01) and Extended (100)

Read multiple variables (BOOLEAN) from the slave.

Element	Meaning
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.

Read Discrete Inputs (02) and Extended (101)

Read multiple variables (BOOLEAN) from the slave.

Element	Meaning
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.

Read Holding Registers (03) and Extended (102)

Read multiple variables of any type from the slave.

Element	Meaning
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.

Read Input Registers (04) and Extended (103)

Read multiple variables of any type from the slave

Element	Meaning
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.

6.1.7 Read and write request telegrams

To read and write variables, the Modbus master sends a read/write request telegram to the Modbus slave.

The Modbus master first writes the write variables into the defined import area of the Modbus slave.

21233799/EN – 07/2014

The Modbus master then reads the read variables from the defined export area of the Modbus slave.

INFORMATION

In the read/write request telegram, the write and read functions are independent of one another; they are just sent in the same request telegram.

However, the read/write request telegram is often used in such a way that the variables written by the Modbus master are read back. This ensures that the sent variables were written correctly.

Do the following to configure a read and write request telegram:

1. In the structure tree, select the [Request Telegram] you want to configure.
2. Right-click "Request Telegram" and choose [Edit] from the context menu.

Configuring read variables

Do the following to configure the read variables:

1. In the object selection, select a global variable to be connected to the new Modbus receive variable and drag it onto the "Global Variable" column of the Modbus receive variable.
2. Repeat step 1 for each additional Modbus receive variable.
3. Open the context menu by right-clicking anywhere in the “Input Signals” area and choose [New offsets] to renumber the offsets of the variables.

Configuring write variables

Do the following to configure write variables:

1. In the object selection, select a global variable to be connected to the new Modbus send variable and drag it onto the "Global Variable" column of the Modbus send variable.
2. Repeat step 1 for each additional Modbus send variable.
3. Open the context menu by right-clicking on an empty spot in the “Output signals” area and select [New offsets] to renumber the offsets of the variables.

Read Write Holding Register (23) and Extended (106)

Write and read multiple variables of any type to and from the import area of the slave.

Element	Meaning
Type	Modbus function *Read Write Holding Registers*.
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.

6.1.8 Write request telegram

Using the write function codes, variables are written only to the import area of a slave.

In addition to the Modbus function, the request telegram of a Modbus master also contains the start address for the read/write area.

21233799/EN – 07/2014

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 Assign Variables" dialog for a write request telegram.

Do the following to configure a write request telegram:

1. In the structure tree, select the [Request Telegram] you want to configure.
2. Right-click "Request Telegram" and choose [Edit] from the context menu.
3. In the object selection, select the global variable to be used as Modbus send variable and drag it anywhere in the "Send Signals" area.
4. Repeat step 3 for each additional Modbus send variable.
5. Open the context menu by right-clicking anywhere in the "Send Signals" area and choose [New offsets] to renumber the offsets of the variables.

The following write request telegrams are available:

Write Multiple Coils (15) and Extended (104)

Write multiple variables (BOOLEAN) to the import area of the slave.

Element	Meaning
Type	Modbus function *Write Multiple Coils*.
Name	Any unique name for the Modbus function.
Description	Description for the Modbus function.
Start address of the write area	0 – 65535.

Write Multiple Registers (16) and Extended (105)

Write multiple variables of any type to the import area of the slave.

Element	Meaning
Type	Modbus function *Write Multiple Registers*.
Name	Any unique name for the Modbus function.
Description	Description for the Modbus function.
Start address of the write area	0 – 65535.

Write Single Coil (05)

Write a single variable (BOOLEAN) to the import area of the slave.

Element	Meaning
Type	Modbus function *Write Single Coil*.
Name	Any unique name for the Modbus function.
Description	Description for the Modbus function.
Start address of the write area	0 – 65535.

Write Single Register (06)

Write a single variable (WORD) to the import area of the slave.

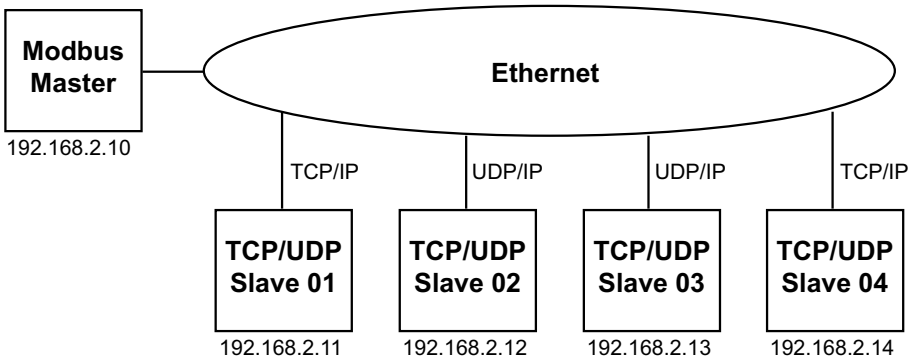
Element	Meaning
Type	Modbus function *Write Single Register*.
Name	Any unique name for the Modbus function.
Description	Description for the Modbus function.

21233799/EN – 07/2014

Element	Meaning
Start address of the write area	0 – 65535.

6.1.9 Ethernet slaves (TCP/UDP slaves)

The Modbus master can communicate with up to 64 TCP/IP and 247 UDP/IP slaves.



5307419275

Do the following 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] and choose [New] from the context menu.
3. Choose "TCP/UDP Slaves" from the list and confirm with [OK].
4. Do the following to configure the TCP/UDP slave in the Modbus master:

   Choose [Edit] to assign the system variables; see chapter "System variables of TCP/UDP slaves".

   Choose [Properties] to configure the properties; see chapter "Properties of TCP/UDP slaves".

INFORMATION

If the TCP/UDP slaves and the Modbus master are located in different subnets, then the routing table must contain the corresponding user-defined routes.

The telegrams that the Modbus TCP master sends to the Modbus TCP slave always include the IP address as well as a Modbus slave address (unit identifier). This address is always FF_{Hex} (255).

System variables of the TCP/UDP slaves

The "System Variables" tab provides the following system variables that are required to control the TCP/UDP slave and evaluate its state from within the user program.

You can use the following status variables to evaluate the TCP/UDP slave status from within the user program:

Element	Description	
Modbus slave activation control	Using this function, the user program activates or deactivates the TCP/UDP slave.	0: Activate 1: Deactivate (edge-triggered – The Modbus slave can be activated using the PADT even if Modbus slave activation control is set to 1)
Modbus slave error	Error code	The error codes 0x01 – 0x0b correspond to the exception codes of the Modbus protocol specification. 0x00: No error
	Exception codes:	0x01: Invalid function code 0x02: Invalid addressing 0x03: Invalid data 0x04: (Not used) 0x05: (Not used) 0x06: Device busy (gateway only, not supported) 0x08: (Not used) 0x0a: (Not used) 0x0b: No response from slave (gateway only, not supported)
	SEW-specific codes	0x10: Faulty frame received 0x11: Frame with wrong transaction ID received 0x12: Unexpected response received 0x13: Response about wrong connection received 0x14: Wrong response to a write request 0xff: Slave timeout
Modbus slave state	Connection status of the TCP/UDP slave	0: Deactivated 1: Not connected 2: Connected

21233799/EN – 07/2014

Properties of the TCP/UDP slaves

To configure the connection to the TCP/UDP slave, you have to set the following parameters in the Modbus master:

Parameter	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 through (2^{31}−1). If the *Maximum number of retries* was exceeded and the slave could not be reached, the value for *Master-slave Data exchange* is set four times higher.
TCP connection only on demand	If the type of transmission protocol is TCP, then this parameter is used to define whether the connection to the slave should be terminated automatically whenever data is exchanged. TRUE: Terminate the connection. FALSE: Do not terminate the connection. Default: FALSE
Receive timeout [ms]	Receive timeout for this slave [ms]. After this time has elapsed, a resend is attempted.
IP address	IP address of the TCP/UDP slave.
Port	Default: 502 Other TCP/UDP ports can also be configured. In this case, observe the port assignment provided by the *Internet Corporation for Assigned Names and Numbers (ICANN)*.
IP protocol communication type	TCP or UDP. Default: TCP
Maximum number of resends	Maximum number of send retries if the slave does not respond. The number of resends can be set as required (0 – 65535). With TCP/IP, the value is always set to 0 and cannot be changed. We recommend setting a value between 0 and 8.

6.1.10 Control Panel (Modbus master)

The control panel can be used to verify and control the settings for the Modbus master. Current status information of the master (such as master state, etc.) is also displayed.

Do the following to open the control panel to monitor the Modbus master:

1. In the structure tree, right-click [Hardware] and choose [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 [Modbus master] in the structure tree.

Context menu (Modbus master)

The following commands are available on the context menu of the selected Modbus master:

Offline: Use this command to stop the Modbus master.

Operate: Use this command to start the Modbus master.

Reset statistical data: Use this command to reset the statistical data (such as number of bus errors, cycle time min. and max., etc.) to zero.

Display field (Modbus master)

The display field shows the following values of the selected Modbus master:

Element	Description
Name	Name of the Modbus master.
Master state	The Modbus master indicates the present protocol state: OPERATE OFFLINE
Bus error count	Counter for bus errors.
Interrupted connections	Counter for interrupted connections.
µP load (planned)	See features in chapter "Menu functions of the Modbus master".
µP load (actual)	

6.1.11 Control Panel (Modbus master → slave)

You can use the control panel to verify and activate/deactivate the settings for the communication partners of the Modbus master.

Current status information (such as slave status, etc.) of the communication partner is also displayed.

Do the following to open the control panel to monitor the Modbus connection:

- In the structure tree, right-click [Hardware] and choose [Online] from the context menu.
- In the System Login window, enter the access data to open the online view for the hardware.
- Double-click "COM module" and select [Modbus master]/[Slave] in the structure tree.

6.2 Modbus slave

The Modbus slave can operate several Modbus masters simultaneously using Ethernet (TCP/UDP).

Property	Description
Modbus slave	One Modbus slave can be configured.

21233799/EN – 07/2014

Property	Description
Number of master accesses	• TCP A maximum of 20 Modbus masters can access the slave. • UDP An unlimited number of Modbus masters can access the slave.
Max. size of send data	64 kB send 64 kB receive Note: The status byte of the master and the status bytes of each associated slave must be subtracted from the maximum size of the send data.
Max. size of receive data	
Modbus data display format	The safety controller uses big endian format. Example: 32-bit data (for example DWORD, DINT):

32-bit data (hex)	0x12345678			
Memory offset	0	1	2	3
Big endian	12	34	56	78
Middle endian	56	78	12	34
Little endian	78	56	34	12

6.2.1 Configuring the Modbus TCP slave

Do the following to create a new Modbus slave:

1. In the structure tree, open [Configuration] / [Resource] / [Protocols].
2. To add a new Modbus slave set, choose [New] / [Modbus master] from the context menu of Protocols.
3. In the context menu of the Modbus Slave Set, choose [Edit] and [Modbus Slave Set Properties], and retain the default values.
4. Select the "Modbus Slave" tab and make the following settings:
   - Select [COM Module]
   - Activate [Enable TCP]
   - The remaining parameters retain the default values.

INFORMATION

For an example of how to configure the connection between a Modbus TCP slave and a Modbus TCP master, refer to chapter "Modbus example".

6.2.2 Menu functions of the Modbus slave set

To open the "Modbus Slave Set Properties" dialog, choose [Edit] from the context menu of the Modbus slave set. The dialog contains the following tabs:

21233799/EN – 07/2014

Properties of the Modbus slave set

You can set the following parameters for the Modbus slave in the "Modbus Slave Set Properties" tab:

Element	Description
Name	Name of the Modbus slave set.
Activate max. µP budget	**Activated:** Adopt the µP budget limit from the *Max. µP budget* in [%] field. **Deactivated:** Do not use the µP budget limit for this protocol.
Maximum µP budget in [%]	Maximum µP load of the COM module that can be used for processing the protocols. Range of values: 1 – 100% Default: 30%
Activate redundant operation	Activated: Redundant operation (not supported) Deactivated: Mono operation Default: Deactivated
Max. response time [ms]	Time period after reception of a request within which the Modbus slave may respond. For the MOVISAFE® HM31 safety controller, this value must be set to 0 ms. Range of values: 0 – (2^{31}−1) ms Default: 5000 ms (0 = no limitation)
Area for reading function codes 1, 3, 100, 102	This parameter defines from which data area the data should be read for function codes 1, 3, 100, 102. Range of values: • Import area • Export area
Area for reading function codes 23, 106	Here you can specify the Modbus slave area from which function code 23 should be read. **Import area:** The master has read and write access to the import area of the slave. **Export area:** The master reads from the export area of the slave, and writes to the import area of the slave. Note: The master reads and writes during a single CPU cycle. This means that the read data was provided during the last CPU cycle.
COM: Values at connection loss to master	If the connection between the communication module and the Modbus master is lost, the input variables are sent to the processor module either initialized or unchanged, depending on this parameter. **Adopt initial data:** Input variables are reset to their initial values. **Retain last value:** Input variables retain their last values.

21233799/EN – 07/2014

Element	Description
CPU: Values at connection loss to COM	If the connection between the processor module and the communication module is lost, the input variables are initialized or continue to be used unchanged in the processor module, depending on this parameter. **Same behavior as COM to master:** See settings in parameter "COM: Values at connection loss to master". **Retain last value:** Input variables retain their last values. **Default:** Input variables retain their last values.
Use alternative register/bit addressing	**Activated:** Use alternative addressing **Deactivated:** Do not use alternative addressing **Default:** Deactivated, see chapter "Offsets for alternative Modbus addressing".
Register area Offset Bits Input	Range of values: 0 – 65535 Default: 0
Register area Offset Bits Output	
Bit area Offset Register Input	
Bit area Offset Register Output	
Process data refresh rate [ms]	Refresh rate in milliseconds during which COM and CPU exchange protocol data. If the *Refresh Rate* is zero or lower than the cycle time for the controller, then data is exchanged as fast as possible. Range of values: 0 – (2^{31}−1) Default: 0
Force process data consistency	**Activated:** All the protocol data is transferred from the CPU to the COM within a CPU cycle. **Deactivated:** All the protocol data is transferred from the CPU to the COM, distributed over several CPU cycles, each with 1100 bytes per data direction. This might also reduce the cycle time of the controller. **Default:** Activated

Register variable ("Access" tab)

In the "Register Variables" tab, enter the variables that the master addresses 16 bitwise (function codes 3, 4, 6, 16, 23, 102, 103, 105, 106).

Bit variables (bit and/or coil access)

In the "Register Variables" tab, enter the variables that the master addresses 1 bitwise (function codes 1, 2, 5, 15, 100, 101, 104).

21233799/EN – 07/2014

6.2.3 Assigning send/receive variables

In the "Inputs" tab, enter all variables that the Modbus slave receives from the Modbus master.

Do the following to configure the send variables of the Modbus slave:

1. In the structure tree, select the Modbus slave you want to configure.
2. Right-click Modbus slave and choose [Edit] from the context menu.
3. Choose the "Register Variables" or "Bit Variables" tab.
4. In the object selection, select a variable and drag it to the "Register Outputs" area.
5. Repeat step 4 for every further variable you want to define as send variable for the Modbus slave.
6. Right-click the "Register Outputs" area and choose [New Offsets] from the context menu.

Configuring the receive variables of the Modbus slave

Do the following to configure the receive variables of the Modbus slave:

1. In the structure tree, select the Modbus slave you want to configure.
2. Right-click "Modbus slave" and choose [Edit] from the context menu.
3. Choose the "Register Variables" or "Bit Variables" tab.
4. In the object selection, select a variable and drag it to the "Register Inputs" area.
5. Repeat step 4 for every further variable you want to define as receive variable for the Modbus slave.
6. Right-click the "Register Inputs" area and choose [New Offsets] from the context menu.

6.2.4 Modbus slave set system variables

The "Modbus Slave Set System Variables" tab provides the following system variables:

Element	Description
Redundancy state (not supported)	This parameter describes the redundancy state of the redundant Modbus slave communication module pair. 0: Redundant Modbus slave COM module active 1: First Modbus slave COM module not active 2: Redundant Modbus slave COM module not active 3: Both Modbus slave COM modules are not active

The "Modbus Slave" tab contains the two tabs "Properties" and "System Variables".

Element	Description
Module	Selection of the COM module within which the protocol is processed.

21233799/EN – 07/2014

Element	Description
Master monitoring time [ms]	Time period after reception of a request within which the Modbus slave must respond. If the connection between the communication module and the Modbus master is lost, the input variables are sent to the processor module either initialized or unchanged, depending on the parameter *COM: Values at connection loss to master*, see chapter "Properties of the Modbus slave set". Value range 1 – (2^{31}−1) [ms] Default: 0 ms (no limitation)
Activate TCP	**Activated:** TCP/IP connection is activated **Deactivated:** TCP/IP connection is deactivated **Default**: Deactivated
TCP port	Default: 502
Maximum number of TCP connections	Maximum number of TCP connections opened simultaneously and operating as server. Range of values: 1 – 20. Default: 3
Activate UDP	**Activated:** UDP/IP connection is activated **Deactivated:** UDP/IP connection is deactivated **Default**: Deactivated
UDP port	Default: 502.

The "System Variables" tab contains system variables that are required to control the Modbus slave and to analyze the 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 power-on of the controller.
Master requests	Total number of all master requests since the last reset of all counters or last power-on of the controller.
Master monitoring time [ms]	Time period after reception of a request within which the Modbus slave must respond, see chapter 7.3.7.
Master connection state	FALSE: Not connected 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. A change from 0 to 1 triggers the reset function. Values greater than 1 are treated as 1.

21233799/EN – 07/2014

Element	Description
Invalid master requests	Number of invalid master requests since the last reset of all counters or last power-on of the controller. An invalid request is a request upon which the Modbus slave responds by sending an error code to the Modbus master. Faulty transmissions that were already detected and filtered out at driver level (framing errors, CRC errors, length errors) are not included in invalid requests but are reported through diagnostics.
Rejected requests	Number of rejected master requests since the last reset of all counters or last power-on of the controller.
Response timeout	Number of response timeouts since the last reset of all counters or last power-on of the controller. The *response timeout* is the maximum time within which the sending station must receive the message acknowledgement.

6.2.5 Modbus function code of the Modbus slave

The Modbus slave supports the following Modbus function codes:

Element	Code	Type	Meaning
READ COILS	01	BOOLEAN	Read multiple variables (BOOLEAN) from the import or export area of the slave. Maximum length of the process data: 251 bytes[1)]
READ DISCRETE INPUT	02	BOOLEAN	Read multiple variables (BOOLEAN) from the export area of the slave. Maximum length of the process data: 251 bytes
READ HOLDING REGISTER	03	WORD	Read multiple variables of any type from the import or export area of the slave. Maximum length of the process data: 250 bytes[1)]
READ INPUT REGISTER	04	WORD	Read multiple variables of any type from the export area of the slave. Maximum length of the process data: 250 bytes
WRITE SINGLE COIL	05	BOOLEAN	Write a single signal (BOOLEAN) to the import area of the slave. Maximum length of the process data: 1 byte
WRITE SINGLE REGISTER	06	WORD	Write a single signal (WORD) to the import area of the slave. Maximum length of the process data: 2 bytes
WRITE MULTIPLE COILS	15	BOOLEAN	Write multiple variables (BOOLEAN) to the import area of the slave. Maximum length of the process data: 247 bytes
WRITE MULTIPLE REGISTER	16	WORD	Write multiple variables of any type to the import area of the slave. Maximum length of the process data: 246 bytes

21233799/EN – 07/2014

Element	Code	Type	Meaning
READ WRITE MULTIPLE REGISTER	23	WORD	Write and read multiple variables of any type to and from the import or export area of the slave. Maximum length of the process data: 242 bytes (request telegram from the Modbus master); 250 bytes (response to the master).
Read device identification	43	Any	Send the identification data of the slave to the master. For details, refer to the information regarding the Modbus function: Read device identification (43)

1) The export area can only be selected in MOVISAFE® HM31 slaves.

The function codes 03, 04, 16 and 23 support all other data types in addition to the WORD (2 bytes) data type.

The start address of the first variable to be sent and the number of registers/bits of the variables to be sent must be entered for each request.

Error codes:

- If the master sends a telegram with unknown function code, the controller responds with error code 1 (invalid code).
- If the request telegram length does not match the variable limit, the slave responds with error code 2 (invalid data).
- If the master sends a telegram with faulty values (such as length field), the slave responds with error code 3 (invalid value).

Communication can only take place when the COM module is in RUN state. If the COM module is in any other operation state, the master does not respond to any request.

Information regarding the Modbus function: Read device identification (43)

The Modbus slave provides the identification data to the master and supports the following object IDs:

Basic:

```
0x00 VendorName "SEW-EURODRIVE GmbH + Co KG"
0x01 ProductCode "<Module serial number>"
0x02 MajorMinorRevision "<COM Vx.y CRC>"
```

Regular:

```
0x03 VendorUrl "http://www.sew-eurodrive.de"
0x04 ProductName "SEW F-PLC"
0x05 ModelName "PFF-HM31A1"
0x06 UserApplicationName "--------[S.R.S]"
```

Extended:

```
0x80 blank "--------"
0x81 blank "--------"
0x82 blank "--------"
0x83 blank "--------"
0x84 blank "--------"
0x85 blank "--------"
```

21233799/EN – 07/2014

`0x86 CRC of the file modbus.config "<0x234adcef>"`

(Configuration file for the Modbus slave protocol in the file system of the CPU. To be compared with the information specified in SILworX under 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 regular 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.

6.2.6 SEW-specific function codes

The SEW-specific function codes correspond to the standard Modbus function codes. The only differences are the maximum allowable process data length of 1100 bytes and the format of request and response headers.

Element	Code	Type	Meaning
Read Coils Extended	100 (0x64)	BOOLEAN	Corresponds to function code 01 Read multiple variables (BOOLEAN) from the import or export area of the slave. Maximum length of the process data: 1100 bytes[1)]
Read Discrete Inputs Extended	101 (0x65)	BOOLEAN	Corresponds to function code 02 Read multiple variables (BOOLEAN) from the export area of the slave. Maximum length of the process data: 1100 bytes
Read Holding Registers Extended	102 (0x66)	WORD	Corresponds to function code 03 Read multiple variables of any type from the import or export area of the slave. Maximum length of the process data: 1100 bytes
Read Input Registers Extended	103 (0x67)	WORD	Corresponds to function code 04 Read multiple variables of any type from the export area of the slave. Maximum length of the process data: 1100 bytes
Write Multiple Coils Extended	104 (0x68)	BOOLEAN	Corresponds to function code 15 Write multiple variables (BOOLEAN) to the import area of the slave. Maximum length of the process data: 1100 bytes
Write Multiple Registers Extended	105 (0x69)	WORD	Corresponds to function code 16 Write multiple variables of any type to the import area of the slave. Maximum length of the process data: 1100 bytes

21233799/EN – 07/2014

Element	Code	Type	Meaning
Read/Write Multiple Registers Extended	106 (0x6A)	WORD	Corresponds to function code 23 Write and read multiple variables of any type to and from the import or export area of the slave. Maximum length of the process data: 1100 bytes (request telegram from the Modbus master); 1100 bytes (response to the master).

1) The export area can only be selected as slave with PFF-HM31A.

Format of request and response headers

The request and response headers of the SEW-specific Modbus function code are structured as follows:

Code	Request	Response
100 (0x64)	1 byte function code 0x64 2 bytes start address 2 bytes number of coils 1 – 8800 (0x2260)	1 byte function code 0x64 2 bytes number of bytes = N N bytes coil data (8 coils are packed into one byte)
101 (0x65)	1 byte function code 0x65 2 bytes start address 2 bytes number of coils 1 – 8800 (0x2260)	1 byte function code 0x65 2 bytes number of bytes = N N bytes coil data (8 coils are packed into one byte)
102 (0x66)	1 byte function code 0x66 2 bytes start address 2 bytes number of registers 1 – 550 (0x226)	1 byte function code 0x66 2 bytes number of bytes = N N bytes register data
103 (0x67)	1 byte function code 0x67 2 bytes start address 2 bytes number of registers 1 – 550 (0x226)	1 byte function code 0x67 2 bytes number of bytes = N N bytes register data
104 (0x68)	1 byte function code 0x68 2 bytes start address 2 bytes number of coils 1 – 8800 (0x2260) 2 bytes number of bytes = N N bytes coil data	1 byte function code 0x66 2 bytes start address 2 bytes number of coils 1 – 8800 (0x2260)
105 (0x69)	1 byte function code 0x69 2 bytes start address 2 bytes number of registers 1 – 550 (0x226) 2 bytes number of bytes = N N bytes register data	1 byte function code 0x69 2 bytes start address 2 bytes number of registers 1 – 550 (0x226)

Code	Request	Response
106 (0x6A)	1 byte function code 0x6a 2 bytes read start address 2 bytes number of read registers 1 – 550(0x226) 2 bytes write start address 2 bytes number of write registers 1 – 550(0x226) 2 bytes number of write bytes = N N bytes register data	1 byte function code 0x6a 2 bytes number of bytes = N N bytes register data

6.2.7 Addressing Modbus using bit and register

This addressing mode corresponds to the Modbus addressing standard and knows only the two data lengths bit (1 bit) and register (16 bits) that are used to send all the permitted data types.

There are two areas in 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 areas differ in the Modbus function codes permitted for accessing these areas.

INFORMATION

Modbus addressing using bit and register does not ensure variable integrity. This means that any portion of a variable can be read or written using this access mode. BOOLEAN variables are stored in a packed format. This means that each variable of this type is stored as a bit within a byte.

Register area

The variables in the register area are created in the "Register Variables" tab. Refer also to chapter "Assigning send/receive variables".

INFORMATION

To access variables in the register area using the Modbus function codes 1, 2, 5, 15, the variables must be mirrored in the bit area, see chapter "Access to register variables in the bit area of the Modbus slave".

The variables in the register area can only be accessed using Modbus function codes 3, 4, 6, 16, 23. For this purpose, 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

21233799/EN – 07/2014

Register variables	Register bit	Bit
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

Modbus master configuration of the request telegram

Do the following to read the variables *01_Register_Area_SINT* to *03_Register_Area_REAL* in the Modbus master:

1. Right-click "TCP/UDP Slave" and choose [New] from the context menu.
2. From the list, choose "Read Holding Registers (3)".
3. Right-click "Read Holding Registers (3)" and choose [Properties] from the context menu.

   Enter "1" as the start address for the read area.
4. Right-click "Read Holding Registers (3)" and choose [Edit] from the context menu.
5. Drag the following variables from the Object Selection 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 and choose [New Offsets] from the context menu to renumber the variable offsets.

Bit area

The variables in the register area are created in the "Bit Variables" tab. Refer also to chapter "Assigning send/receive variables".

INFORMATION

To access variables in the bit area using the Modbus function codes 3, 4, 6, 16, 23, the variables must be mirrored in the register area, see chapter "Accessing bit variables in the register area of the Modbus slave".

The variables in the bit area can only be accessed using Modbus function codes 1, 2, 5, 15. For this purpose, 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

Modbus master configuration of the request telegram

Do the following to read the variables *04_BIT_Area_BOOL* to *06_BIT_Area_BOOL* in the Modbus master:

1. Right-click "TCP/UDP Slave" and choose [New] from the context menu.
2. From the list, choose "Read Coils (1)".
3. Right-click "Read Coils (1)" and choose [Properties] from the context menu.

   Enter "64" as the start address for the read area.
4. Right-click "Read Coils (1)" and choose [Edit] from the context menu.
5. Drag the following variables from the Object Selection onto the "Input Variables" tab:

21233799/EN – 07/2014

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 and choose [New Offsets] from the context menu to renumber the variable offsets.

6.2.8 Offsets for alternative Modbus addressing

To access the variables in the bit area using the Modbus function code (register type), the variables must be mirrored in the register area. To access the variables in the register area using the Modbus function codes (bit type), the variables must be mirrored in the bit area. The offsets for the mirrored variables are entered in the "Properties/ Offsets" tab.

Do the following to mirror the variables in the bit area and register area:

1. Right-click the Modbus slave and choose [Edit] and [Offset] from the context menu. Next, 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 the register area.

INFORMATION

The mirrored variables in the bit/register area and the existing variables in the bit/ register area must not overlap with respect to the Modbus addresses.

Element	Description and range of values
Use alternative register/bit addressing	**Activated:** Use alternative addressing **Deactivated:** Do not use alternative addressing **Default:** Deactivated
Register area offset / bit inputs	0 – 65535
Register area offset / bit outputs	
Bit area offset / register inputs	
Bit area offset / register outputs	

Accessing register variables in the bit area of the Modbus slave

To access the register variables with the Modbus function codes (bit type) 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

21233799/EN – 07/2014

This is where the variables mirrored from the register area to the bit area are located 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

Modbus master configuration of the request telegram

Do the following to read the variables *04_Register_Area_BOOL* to *06_Register_Area_BOOL* in the Modbus master:

1. Right-click "TCP/UDP Slave" and choose [New] from the context menu.
2. From the list, choose "Read Coils (1)".
3. Right-click "Read Coils (1)" and choose [Properties] from the context menu.

   Enter "8064" as the start address for the read area.
4. Right-click "Read Coils (1)" and choose [Edit] from the context menu.
5. Drag the following variables from the Object Selection onto the "Input Variables" tab:

Bit 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 and choose [New Offsets] from the context menu to renumber the variable offsets.

21233799/EN – 07/2014

Accessing bit variables in the register area of the Modbus slave

To access the bit variables with the Modbus function codes (register type) 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

This is where the variables mirrored from the bit area to the register area are located 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

Modbus master configuration of the request telegram

Do the following to read the variables *01_BIT_Area_SINT* to *03_BIT_Area_REAL* in the Modbus master:

1. Right-click "TCP/UDP Slave" and choose [New] from the context menu.
2. From the list, choose "Read Holding Registers (3)".
3. Right-click "Read Holding Registers (3)" and choose [Properties] from the context menu.

   Enter "1001" as the start address for the read area.
4. Right-click "Read Holding Registers (3)" and choose [Edit] from the context menu.

5. Drag the following variables from the Object Selection onto the "Input Variables" tab:

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 and choose [New Offsets] from the context menu to renumber the variable offsets.

6.2.9 Control panel (Modbus slave)

The control panel can be used to verify and control the settings for the Modbus slave. Current status information of the slave (such as master state, etc.) is also displayed.

Do the following to open the control panel to monitor the Modbus slave:

1. In the structure tree, right-click [Hardware] and choose [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 [Modbus Slave] in the structure tree.

Context menu (Modbus slave)

The following command is available on the context menu of the selected Modbus slave:

Reset statistics: Use this command to reset the statistical data (min./max. cycle time, etc.) to zero.

Display field (Modbus slave)

The display field shows the following values of the selected Modbus slave:

Element	Description
Name	Name of the Modbus slave.
Planned µP budget [%]	See chapter "Menu functions of the Modbus slave set".
Current µP budget [%]	
SRS of redundant module (not supported)	SRS of the redundant COM module.
Response time [ms]	Time period after reception of a request within which the Modbus slave must respond.

Display field (master data)

The master data display field shows the following values:

Element	Description
Name	Name of the master data.
Requests	Total number of master requests since the last counter reset.

21233799/EN – 07/2014

Element	Description
Valid requests	Number of valid master requests since the last counter reset.
Invalid requests	Number of invalid master requests since the last counter reset. Invalid requests include only requests that were acknowledged by the master. Requests received with CRC error are automatically rejected.
Master timeout [ms]	Timeout time. Timeout within which the slave must receive at least one request from the master. If the slave does not receive a request within the timeout, the *Master Connection Status* is set to "not connected".
Connection state	0 = Not monitored (master request timeout is zero) 1 = Not connected 2 = Connected
Response timeout	Number of response timeouts since the last reset of all counters or last power-on of the controller. The *response timeout* is the maximum time within which the sending station must receive the message acknowledgement.
Rejected requests	Number of rejected master requests since the last reset of all counters or last power-on of the controller.
Maximum buffer fill level for requests	Maximum number of concurrent master requests.
Average buffer fill level for requests	Average number of concurrent master requests.

6.2.10 Error codes of the Modbus TCP/IP connection

The error codes of the Modbus TCP/IP connection are displayed in the "Diagnostics" dialog window.

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	Timeout occurred. Connection terminated.
61	Connection rejected (by peer).
65	No route entry to peer.

21233799/EN – 07/2014

7 Send and receive TCP

Send and receive TCP (S&R TCP) is a manufacturer-independent standard protocol for cyclic and acyclic data exchange. This protocol does not use any specific protocols other than TCP/IP.

With the send and receive TCP protocol, the controller supports nearly every third-party system as well as PCs with socket interface (such as Winsock.dll) to TCP/IP.

Send and receive 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).

7.1 System requirements

Equipment and system requirements

Element	Description
Controller	MOVISAFE® HM31 from CPU BS V7 and later, and COM BS V12.
CPU module	The Ethernet interfaces of the processor module cannot be used for send and receive TCP.
COM module	Ethernet 10/100BaseT. One send and receive TCP protocol can be configured for each COM module.
Activation	Software activation code required (see chapter "Registering and activating the protocols").

Properties of the send and receive TCP protocol

Element	Description
Safety-related	No.
Data exchange	Cyclic and acyclic data exchange over TCP/IP.
Function blocks	The send and receive TCP function blocks must be used for exchanging data acyclically.
TCP connections	Up to 32 TCP connections can be configured in one controller if the maximum size of send or receive data is not exceeded.
Max. size of send data	**Note:** To determine the maximum amount of useful data, the value for all status variables of the TCP connections used and for the TCP/SR function blocks must be subtracted from the value for the maximum amount of send data. The data can be assigned as required to the individual TCP connections.
Max. size of receive data	

7.1.1 Creating a send and receive TCP protocol

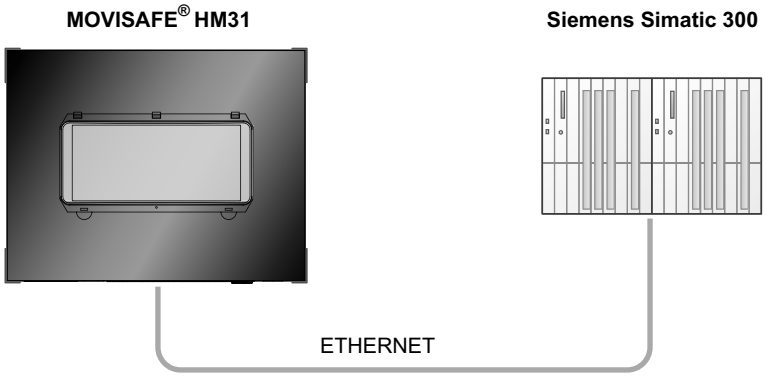
Do the following to create a send and receive TCP protocol:

1. In the structure tree, open [Configuration] / [Resource] / [Protocols].
2. To add a new send and receive protocol, right-click "Protocols" and choose [New] / [Send/Receive over TCP] from the context menu.

21233799/EN – 07/2014

3. From the context menu of [Send/Receive over TCP], choose [Properties] / [General], and select the COM module.

7.2 Example of a send and receive TCP configuration



12757488907

In this example, the protocol send/receive over TCP is installed in MOVISAFE®. MOVISAFE® HM31 is to cyclically communicate with a Siemens controller (such as SIMATIC 300) via S&R TCP.

In this example, MOVISAFE® HM31 (client) is the active station that establishes the TCP connection to the passive Siemens SIMATIC 300 (server). Once the connection has been established, both controllers are equal and can send and receive data at any time.

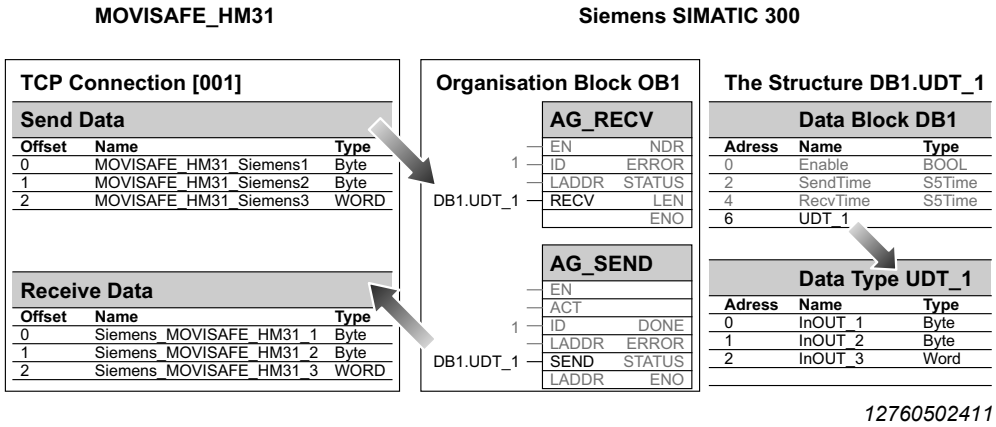
Take the following points into account when connecting the MOVISAFE® HM31 controller to the Siemens SIMATIC 300 controller:

- The requirements described in the "System requirements" chapter apply to MOVISAFE® HM31.
- MOVISAFE® HM31 and Siemens SIMATIC 300 are connected to one another via their Ethernet interfaces.
- The MOVISAFE® HM31 and Siemens SIMATIC 300 controllers must be located in the same subnet, or they must have the corresponding routing settings if a router is used.

In the example, the MOVISAFE® HM31 controller is supposed to send two BYTES and one WORD to the Siemens SIMATIC 300 controller. The variables are received in the SIMATIC 300 controller by the function block AG_RECV (FC 6), and are passed on internally to the function block AG_SEND (FC 5). SIMATIC 300 sends the variables unchanged to the MOVISAFE® HM31 controller using the function block AG_SEND (FC 5).

21233799/EN – 07/2014

Once the configuration is completed, the user can verify successful transmission using the Force Editor.



12760502411

Description of the MOVISAFE® HM31 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 MOVISAFE® HM31 controller must be identical with the address and variable types in the *UDT_1* data type in the SIMATIC 300 controller.
Receive data	The variable offsets and types in the MOVISAFE® HM31 controller must be identical with the address and variable types in the *UDT_1* data type in the SIMATIC 300 controller.

Description of the SIMATIC 300 configuration:

Element	Description
Organization block OB1	The function blocks *AG_RECV (FC 6)* and *AG_SEND (FC 5)* must be created and configured in the *OB1* organization block.
AG_RECV (FC 6)	The AG_RECV (FC 6) function block accepts the data received from the communication partner in data type DB1.UDT_1. The *ID* and *LADDR* inputs must be appropriately configured for communication with the communication partner.
AG_SEND (FC 5)	The *AG_SEND (FC 5)* function block sends the data from data type *DB1.UDT_1* to the communication partner. The *ID* and *LADDR* inputs must be appropriately configured for communication with the communication partner.
Data block DB1	The data type *UDT_1* is defined in the *DB1* data block.
Data type UDT_1	The addresses and types of the variables in the SIMATIC 300 controller must be identical with the offsets and types of the controller. Data type *UDT_1* accepts the received user data and stores them until they are sent to the communication partner.

21233799/EN – 07/2014

7.2.1 Send and receive TCP configuration of the SIMATIC 300 controller

INFORMATION

The following step-by-step instructions for configuring the Siemens controller are not exhaustive. The information is provided without guarantee. Refer to the Siemens documentation when planning projects with Siemens controllers.

Do the following to create the send and receive TCP server in the SIMATIC 300 project:

1. Start the SIMATIC manager.
2. In the SIMATIC manager, open the project of the SIMATIC 300 controller.
3. In this project, create and configure the "Industrial Ethernet" and "MPI connections".

Creating the UDT1 data type

Do the following to create the *UDT1* data type using the following variables:

1. Open the "Function Blocks" folder in the Siemens SIMATIC manager.
2. From the main menu, choose [Add] / [S7 Function Block] / [Data Type], and create a data type.
3. Name the data type *UDT1*.
4. Assign the symbolic name *UDT1* to the data type.
5. In the *UDT_1* data type, create three variables *InOut_x* (see variable list below).

Address	Name	Type	Initial value
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	

INFORMATION

During cyclic and acyclic data exchange, bear in mind that some controllers (such as SIMATIC 300) add so-called "pad bytes". These 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 added in the correct position of the MOVISAFE® HM31 controller (see chapter "Third-party systems with pad bytes").

Creating DB1 data block for FC5 and FC6 function blocks

Do the following to create the DB1 data block for the FC5 and FC6 function blocks:

1. From the main menu, choose [Add] / [S7 Function Block] / [Data Block], and create a data block.
2. Name the data block "DB1".
3. Assign the symbolic name "DB1" to the data block.
4. Assign the *UDT_1* data type to the "DB1" data block.

21233799/EN – 07/2014

5. In the DB1 data block, configure the data types (see variable list below).

Address	Name	Type	Initial value
0.0		STRUCT	
+0.0	Enable	BOOLEAN	FALSE
+2.0	SendTime	SSTIME	SST#100MS
+4.0	RecvTime	SSTIME	SST#10MS
+6.0	UDT_1	„UDT_1“	
=10.0		END_STRUCT	

Creating symbols in the symbol editor

Do the following to create the following symbols:

1. Double-click the OB1 organization block to open "LD/IL/FBD".
2. From the main menu, choose [Extras] / [Symbol Table] to open the symbol editor.
3. Add variables M 1.0 to MW 5 in the symbol editor (see overview below).

Symbol	Address	Type
RecDone	M 1.0	BOOLEAN
RecError	M 1.1	BOOLEAN
SendDone	M 1.2	BOOLEAN
SendError	M 1.3	BOOLEAN
RecStatus	MW 1	WORD
RecLen	MW 3	INT
SendStatus	MW 5	WORD

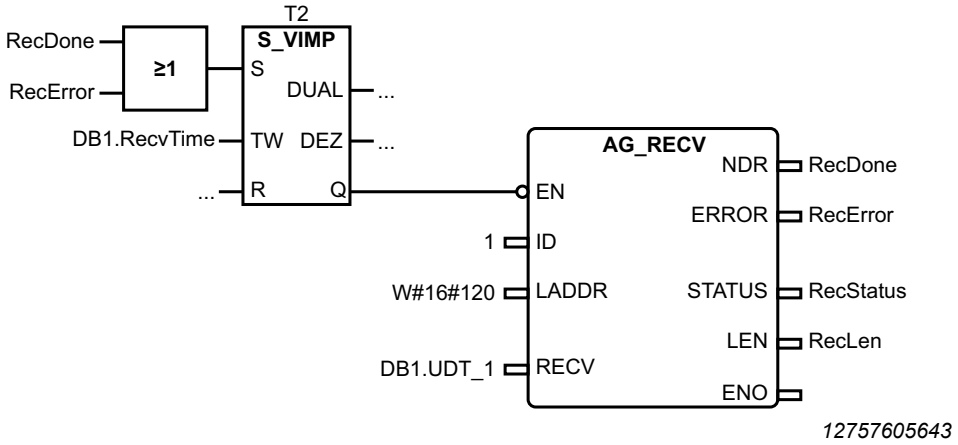
Creating the function block AG_RECV (FC6)

Do the following to create the function block AG_RECV (FC6):

1. Open the dialog box "LD/IL/FBD".
2. From the structure tree on the left side of the symbol manager, choose the following function blocks in the following order:
   - 1 OR gate
   - 1 S_VIMP
   - 1 AG_RECV (FC6)
3. Drag these function blocks onto the *OB1* organization block.
4. Connect and configure the function blocks as shown in the figure below.
5. Right-click the function block *AG_RECV (FC6)* and choose "Properties".
6. Deactivate "Active Connection Setup" and configure the ports.

21233799/EN – 07/2014

7. Note down the *LADDR* function block parameter and enter it in the function chart on the *AG_RECV (FC6)* function block.

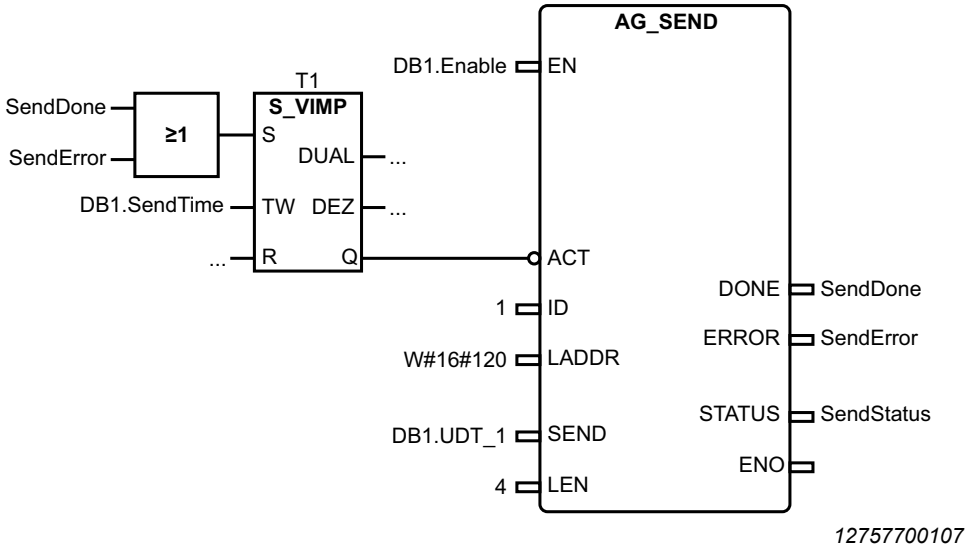


12757605643

Creating the function block AG_SEND (FC5)

Do the following to create the function block AG_SEND (FC5):

1. Open the dialog box "LD/IL/FBD".
2. From the structure tree on the left side of the symbol manager, choose the following function blocks in the following order:
   - 1 OR gate
   - S_VIMP
   - 1 AG_SEND (FC5)
3. Drag these function blocks onto the *OB1* organization block.
4. Connect and configure the function blocks as shown in the figure below.
5. Right-click the function block *AG_SEND (FC5)* and choose "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 (FC5)* function block.



12757700107

Loading the code into the SIMATIC 300 controller

Do the following to load the code into the SIMATIC 300 controller:

1. Start the code generator for the program.
2. Make sure that the codes were generated without any errors.
3. Load the code into the SIMATIC 300 controller.

7.2.2 Send and receive TCP configuration of MOVISAFE® HM31

For more information on how to configure the MOVISAFE® HM31 controller and how to use the SILworX® programming tool, refer to the "SILworX® First Steps" manual.

Do the following to create the following global variables in the variable editor:

1. In the structure tree, choose [Configuration] / [Global Variables].
2. From the context menu, choose "Edit".
3. Create global variables (see table below).

Name	Type
Siemens_PFF-HM31	Byte
Siemens_PFF-HM31	Byte
Siemens_PFF-HM31	WORD
PFF-HM31_Siemens1	Byte
PFF-HM31_Siemens2	Byte
PFF-HM31_Siemens3	WORD

Creating a send and receive TCP protocol in the resource

Do the following to create a send and receive TCP protocol in the resource:

1. Open [Configuration]/[Resource] in the structure tree.
2. Right-click "Protocols" and choose "New" from the context menu. Choose "Send/ Receive over TCP" and enter a name for the protocol.
3. Confirm your entries with "OK" to create the new protocol.
4. Right-click "Send/Receive over TCP" and choose "Properties" from the context menu. Then select "COM Module". The remaining parameters retain the default values.

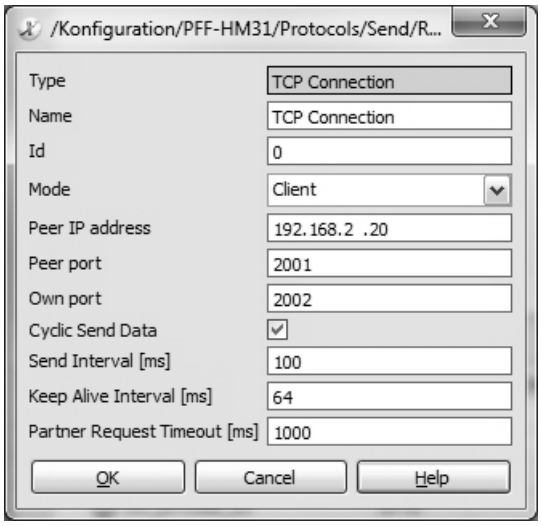
Creating a TCP connection

Do the following to create a TCP connection:

1. Right-click "Send/Receive over TCP" and choose "New" from the context menu. Then select "TCP Connection".

21233799/EN – 07/2014

2. Right-click "TCP Connection" and choose "Properties" from the context menu. Configure the properties as shown in the figure below.



12846254859

INFORMATION

If you want to set parameters for cyclic data exchange between two controllers, activate the option "Cyclic data transfer" in the "Properties" dialog box for the TCP connection.

Configuring receive data of the MOVISAFE® HM31 controller

Do the following to configure the receive data of the MOVISAFE® HM31 controller:

1. Right-click "TCP Connection" and choose "Edit" from the context menu. Then choose the "Process Variables" tab.
2. In the object selection, select the following global variables and drag them to the "Input Signals" area.

Global variable	Type
Siemens_PFF-HM31	Byte
Siemens_PFF-HM31	Byte
Siemens_PFF-HM31	WORD

3. Right-click anywhere in the "Register Inputs" area and choose [New Offsets] from the context menu to renumber the variable offsets.

INFORMATION

It is important that the variable offsets in the MOVISAFE® HM31 controller are identical with the variable addresses in the *UDT_1* data type of the SIMATIC 300 controller.

Configuring send data of the MOVISAFE® HM31 controller

Do the following to configure the send data of the MOVISAFE® HM31 controller:

1. Right-click "TCP Connection" and choose "Edit" from the context menu. Then choose the "Process Variables" tab.

2. In the object selection, select the following global variables and drag them to the "Input Signals" area.

Global variable	Type
PFF-HM31_Siemens1	Byte
PFF-HM31_Siemens2	Byte
PFF-HM31_Siemens3	WORD

3. Right-click anywhere in the "Register Inputs" area and choose [New Offsets] from the context menu to renumber the variable offsets.

INFORMATION

It is important that the variable offsets in the MOVISAFE® HM31 controller are identical with the variable addresses in the *UDT_1* data type of the SIMATIC 300 controller.

Verifying the send and receive TCP configuration

Do the following to verify the send and receive TCP configuration:

1. In the structure tree, open [Configuration] / [Resource] / [Protocols] / [Send/Receive over TCP].
2. Click the [Verification] button on the action bar. Then click "OK" to confirm the action.

7.3 Menu functions in the send and receive TCP protocol

7.3.1 Edit

The "Edit" dialog of the send and receive TCP protocol contains the following tab:

System variables

System variables are used to evaluate the state of the send and receive TCP protocol in the user program.

Element	Description
Active connection count	System variable that provides the number of active (not disturbed) connections.
Disturbed connection count	System variable that provides the number of disturbed connections. Disrupted means that the TCP connection was interrupted by a timeout or error.
Status	No function.

7.3.2 Properties

Data is exchanged cyclically or acyclically over a TCP connection. For acyclic data exchange, the send and receive TCP function blocks are required.

Data cannot be exchanged cyclically and acyclically simultaneously on one connection.

21233799/EN – 07/2014

General

Name	Description
Type	Send and receive over TCP.
Name	Name for the send and receive over TCP protocol. Limited to a maximum of 31 characters.
Module	Selection of the COM module within which the protocol is processed.
Activate max. µP budget	• Activated: Adopt the µP budget limit from the *Max. µP Budget in [%]* field. • Deactivated: Do not use a limit of the µP budget for this protocol.
Max. µP budget in %	Maximum µP budget for the module that can be used for processing the protocol. Range of values: 1 – 100% Default: 30%
Behavior on CPU/COM connection loss	If the connection between the processor module and the communication module is lost, the input variables are initialized or continue to be used unchanged in the processor module, depending on this parameter. • Adopt initial data: Input variables are reset to their initial values. • Retain last value: Input variables retain their last values.

CPU/COM

The default parameter values provide the fastest possible data exchange of send and receive TCP data between the COM module (COM) and the CPU module (CPU) in the controller. Change these parameters only if a reduction of the COM and/or CPU load is required for an application, and the process allows this change.

INFORMATION

Only experienced programmers should modify the parameters. Increasing the COM and CPU refresh rate means that the effective refresh rate of the send and receive TCP data is increased. The system time requirements must be checked.

Name	Description
Process data refresh rate in ms	Refresh rate in milliseconds during which COM and CPU exchange send and receive TCP protocol data. If the refresh rate is zero or lower than the cycle time for the controller, then data is exchanged as fast as possible. Range of values: 0 to (2^{31} − 1) Default: 0

21233799/EN – 07/2014

Name	Description
Force process data consistency	• Activated: All the send and receive TCP protocol data is transferred from the CPU to the COM within a CPU cycle. • Deactivated: Transfer of the send and receive TCP data (a maximum of 1100 bytes per data direction) from the CPU to the COM distributed over several CPU cycles.

7.4 Menu functions for TCP connection

7.4.1 Edit

The "Edit" menu function is used to open the tabs "Process Variables" and "System Variables".

Process variables

- Input signals

  The variables for cyclic data exchange to be received by this controller are entered in the "Input Signals" area. Any variable can be created in the "Input Signals" tab. The offsets and types of the variables must be identical with the offsets and types of the variables (send data) of the communication partner.

- Output signals

  The variables for cyclic data exchange to be sent by this controller are entered in the "Output Signals" area. Any variable can be created in the "Output Signals" tab. The offsets and types of the variables must be identical with the offsets and types of the variables (receive data) of the communication partner.

7.4.2 System variables

Using the variables in the "System Variables" tab, you can evaluate the state of the TCP connection in 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.
Error code timestamp in ms	Millisecond fraction of the timestamp. Time when the error occurred.
Error code timestamp in s	Second fraction of the timestamp. Time when the error occurred.
Partner request timeout	With cyclic data transmission: Timeout during which the communication partner must receive at least one data package after data has been sent. 0: Off 1: 1 to (2^{31} − 1) ms

21233799/EN – 07/2014

Name	Description
Partner connection status	If no data is received during the timeout, the partner connection state is set to "not connected" and the connection is re-established. 0: No connection 1: Connection OK
Status	Connection status of the TCP connection.

7.4.3 Properties

Data is exchanged cyclically or acyclically over a TCP connection. For acyclic data exchange, the send and receive TCP function blocks are required. Send and receive TCP function blocks cannot be used for cyclic data exchange.

Name	Description
Type	TCP connection.
Name	Any unique name for a TCP connection. Limited to a maximum of 31 characters.
ID	Any unique identification number (ID) for each TCP connection. The ID is also required as reference for the send and receive TCP function blocks. Range of values: 0 – 255 Default: 0
Mode	• Server (default value) This station operates as a server, which means in passive mode. The connection must be initiated by the communication partner (client). Once communication has been established, both communication partners are equal and can send data at any time. The own port must be specified. • Server with defined partner This station operates as a server, which means in passive mode. The connection must be initiated by the communication partner (client). Once communication has been established, both communication partners are equal and can send data at any time. If you enter the IP address and/or the port of the communication partner here, then only the specified communication partner can connect to the server. All other stations are ignored. If one of the parameters (IP address or port) is set to zero, then this parameter is not verified. • Client This station operates as a client, which means it initiates the connection to the communication partner. You have to specify the IP address and port of the communication partner. Instead, you can define an own port.

21233799/EN – 07/2014

Name	Description
Partner IP address	IP address of the communication partner. 0.0.0.0: Any IP address is permitted Valid range: 1.0.0.0 to 223.255.255.255, except for: 127.x.x.x Default: 0
Partner port	Port of the communication partner. 0: Any port. Ports that are reserved or already used (1 – 1024) are rejected by the COM-BS. Range of values: 0 – 65535 Default: 0
Own port	Own port. 0: Any port Ports that are reserved or already used (1 – 1024) are rejected by the COM-BS. Range of values: 0 – 65535 Default: 0
Cyclic data transmission	• Deactivated (default) Cyclic data transmission is deactivated. Data exchange over this TCP connection must be programmed with function blocks. No cyclic I/O data may be defined on this connection. • Activated Cyclic data transmission is active. Data is defined in the "Process Variables" dialog box for the TCP connection. Receive data must have been defined. No function blocks can be used on this connection.
Send interval	Can only be edited for cyclic data transmission. This is where the send interval is set. Range of values: 10 – 2 147 483 647 ms (lower values are rounded to 10 ms) Default: 0
Keep alive interval in s	Time until the connection monitoring provided by the TCP is active. Zero deactivates connection monitoring. If no data is exchanged during the specified keep alive interval, keep alive samples are sent to the communication partner. If the connection is still established, the keep alive samples are acknowledged by the communication partner. If no data is exchanged between the partners within a period of > 10 keep alive intervals, the connection is closed. If no response is received after a data packet was sent, the data packet will be resent at predefined intervals. The connection is aborted after 12 unsuccessful resend attempts (approximately 7 minutes).

21233799/EN – 07/2014

Name	Description
Partner request timeout in ms	With cyclic data transmission: Timeout during which the communication partner must receive at least one data package after data has been sent. If no data is received during the timeout, the partner connection state is set to "not connected" and the connection is re-established. After having closed the connection due to timeout or another error, the active side re-establishes the connection with a delay of 10 x "PartnerRequestTimeout" or a delay of 10 seconds, if "PartnerRequestTimeout" is 0. The passive side opens the port already after half this time.

7.5 Data exchange

Send and receive TCP operates according to the client/server principle. The connection must be established by the communication partner that is configured as the client. Once communication has been established, both communication partners are equal and can send data at any time.

Send and receive TCP does not have its own data protection protocol but directly uses the TCP/IP protocol. As TCP sends the data in a data stream, it must be ensured that offsets and types of variables to be exchanged are identical both on the receiver side and the send side.

Send and receive 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).*

Additionally, 5 send and receive TCP function blocks are provided for controlling and individually configuring communication using the user program. Using the send and receive TCP function blocks, any protocol (for example Modbus) can be sent over TCP.

7.5.1 TCP connections

At least one TCP connection must be created in the MOVISAFE® HM31 controller for each connection to a communication partner over send and receive TCP.

You have to set the identification number of the TCP connection and the addresses/ports of the own controller and of the controller of the communication partner in the "Properties" dialog box for the TCP connection.

You can establish a maximum of 32 TCP connections in a MOVISAFE® HM31 controller. These TCP connections must have different identification numbers and different addresses/ports.

Do the following to create a new TCP connection:

INFORMATION

The MOVISAFE® HM31 controller and the third-party controller must be located in the same subnet, or they must have the corresponding routing settings if a router is used.

1. Open [Configuration]/[Resource] in the structure tree.
2. Right-click "Protocols" and choose "New" from the context menu. Then select "Send/Receive over TCP". Enter a name for the protocol.
3. Confirm your entries with "OK" to create the new protocol.

21233799/EN – 07/2014

4. Right-click "Send/Receive over TCP" and choose "Properties" from the context menu. Then select "COM Module".
5. The remaining parameters retain the default values.

7.5.2 Cyclic data exchange

If data is exchanged cyclically, you have to define a send interval in the MOVISAFE® HM31 controller and in the communication partner.

The send interval defines the cyclic time period during which the sending communication partner sends the variables to the receiving communication partner.

- To ensure a continuous data exchange, you should define almost the same send interval for both communication partners (see chapter "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.
- Assign the variables to be sent and received in the "Process Variables" dialog box for the TCP connection. Receive data must exist, send data are optional.

INFORMATION

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.

7.5.3 Acyclic data exchange with function blocks

In MOVISAFE® HM31 controllers, acyclic data exchange is controlled by the user program via the send and receive TCP function blocks. This means data exchange can be controlled using a timer or a mechanical switch connected to a physical input of the MOVISAFE® HM31 controller.

- For acyclic data exchange, the "Cyclic data transfer" option must be deactivated in the TCP connection in use.
- Only one send and receive TCP function block may send at a given time.
- You assign the variables to be sent and received in the "Process Variables" dialog box for the send and receive TCP function blocks (all except for reset).

INFORMATION

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.

7.5.4 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 different partner ports.

A single TCP connection cannot be used simultaneously for cyclic and acyclic data exchange.

21233799/EN – 07/2014

7.5.5 Flow control

Flow control is part of TCP and monitors the continuous data traffic between two communication partners.

With cyclic data transmission, at least one packet must be received after a maximum of 3 to 5 packets have been sent. Else, transmission is blocked until a packet is received or the connection monitoring process terminates the connection.

The number (3 to 5) of potential transmissions without packet reception depends on the size of the packets to be sent.

- Number = 5 for small packets < 4 kB
- Number = 3 for large packets ≥ 4 kB
- When planning the project, bear in mind that no station sends more data than the other station can process simultaneously.
- For cyclic data exchange, set almost the same send interval for both communication partners.

7.6 Third-party systems with pad bytes

During cyclic and acyclic data exchange, bear in mind that some controllers (such as SIMATIC 300) add so-called pad bytes. These 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 packets (in bytes) is even.

In the MOVISAFE® HM31 controller, dummy bytes must be added for the pad bytes at the corresponding positions.

Address	Name	Type	Initial value
0.0		STRUCT	
+0.0	InOut_1	BYTE	B#16#0
+2.0	InOut_2	WORD	W#16#0
=4.0		END_STRUCT	

In the Siemens controller, a pad byte is added (not visible) so that the *InOut_3* variable begins at an even offset (see figure below).

Output Signals

	Name	Data type	Offset	Global Variable
1	Dummy	BYTE	1	Dummy
2	InOut_1	BYTE	0	InOut_1
3	InOut_3	WORD	2	InOut_3

12846249227

In the MOVISAFE® HM31 controller, a dummy byte must be added so that the *InOut_3* variable has the same offset as in the Siemens controller.

7.7 Send and receive TCP function blocks

If cyclic data transmission is not flexible enough, data can also be sent and received using send and receive TCP function blocks. The "Cyclic data transfer" option must be deactivated in the TCP connection in use.

Using send and receive TCP function blocks, you can optimally adjust data transmission over TCP/IP to the requirements of your project.

21233799/EN – 07/2014

The function blocks are parameterized in the user program. In this way you can set and evaluate the functions (send, receive, reset) of the MOVISAFE® HM31 controller in the user program.

Send and receive TCP function blocks are only required for acyclic data exchange. They are not required for cyclic data exchange between server and client.

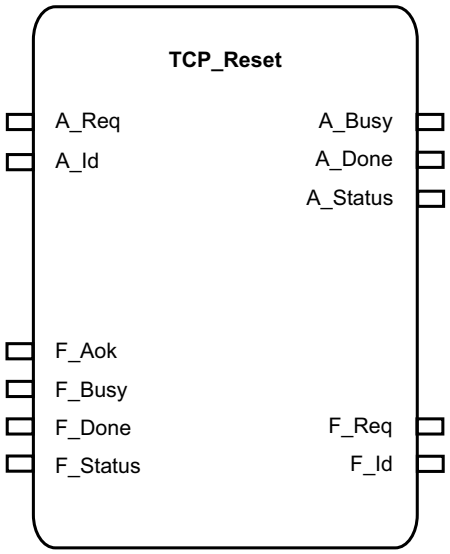
INFORMATION

- The send and receive TCP function blocks must be integrated using the *TCPlib.A3* library. You find the *TCPlib.A3* library on the installation CD for SILworX®.
- For a description of how to set the parameters of the send and receive TCP function blocks, refer to chapter "Configuring the function blocks".

The following function blocks are available:

Function block	Description of the function
TCP_Reset (see chapter "TCP_Reset")	TCP connection reset.
TCP_Send (see chapter TCP_Send")	Sending of data.
TCP_Receive (see chapter "TCP_Receive")	Reception of data packets with fixed length.
TCP_ReceiveLine (see chapter "TCP_ReceiveLine")	Reception of an ASCII line.
TCP_ReceiveVar (see chapter "TCP_ReceiveVar")	Reception of data packets of variable length (with length field).
LATCH	Is used only within other function blocks.
PIG	Is used only within other function blocks.
PIGII	Is used only within other function blocks.

7.7.1 TCP_Reset



12757709579

21233799/EN – 07/2014

The TCP_Reset function block is used to re-establish a disturbed connection if a send or receive function block reports a timeout (16#8A).

INFORMATION

To configure the function block, drag it from the function block library and drop it in the user program (see chapter "Configuring function blocks").

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	A positive edge starts the function block.	BOO-LEAN
A_Id	Identification number *ID* of the disturbed TCP connection to be reset.	INT

A-outputs	Description	Type
A_Busy	TRUE: The function block is still being reset.	BOO-LEAN
A_Done	TRUE: Data transmission completed successfully.	BOO-LEAN
A_Status	The status and error code of the function block and of the TCP connection are output at the *A_Status* output.	DWORD

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 the structure tree. The prefix "F" means Field.

INFORMATION

Common variables are used to connect the "Reset" function block in the structure tree (in the "Function blocks" folder) with the "TCP_Reset" function block (in the user program). These common variables must first be created in the 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	BOOLEAN
F_Busy	BOOLEAN
F_Done	BOOLEAN
F_Status	DWORD

21233799/EN – 07/2014

Connect the F-outputs of the "TCP_Reset" function block in the user program to the same variables that will be connected to the inputs of the "Reset" function block in the structure tree.

F-outputs	Type
F_Req	BOOLEAN
F_Id	INT

Creating the "Reset" function block in the structure tree

Do the following 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. Right-click the "Reset" function block and open "Edit" from the context menu.
3. The window for assigning variables to the function block opens.

   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	INT
REQ	BOOLEAN

4. 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	BOOLEAN
BUSY	BOOLEAN
DONE	BOOLEAN
STATUS	DWORD

Operating the "TCP_Reset" function block

The following steps are required for operating the "TCP_Reset" function block:

1. In the user program, set the identification number for the disturbed TCP connection at the *A_ID* input.
2. In the user program, set the *A_Req* input to TRUE.

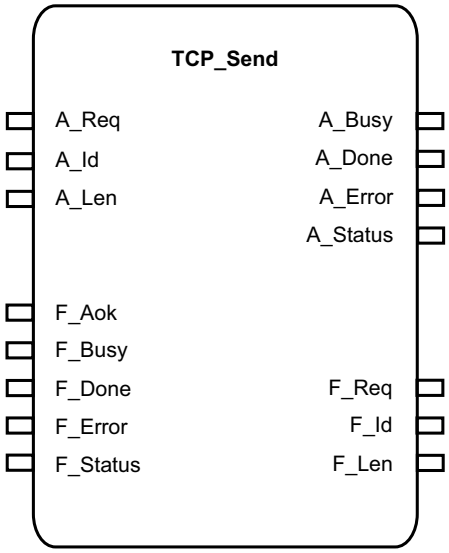
INFORMATION

The function block responds to a positive edge on *A_Req*.

3. The *A_Busy* output is set to TRUE until a reset is sent to the specified TCP connection. Afterwards, the *A_Busy* output is set to FALSE and the *A_Done* output to TRUE.

21233799/EN – 07/2014

7.7.2 TCP_Send



12757780619

The "TCP_Send" function block is used for sending variables acyclically to a communication partner. A function block with the same variables and offsets, such as "Receive" must be configured in the communication partner.

INFORMATION

To configure the function block, drag it from the function block library and drop it in the user program (see chapter "Configuring function blocks").

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	A positive edge starts the function block.	BOOLEAN
A_Id	Identification number of the configured TCP connection to the communication partner to which data should be sent.	INT
A_Len	Number in bytes of variables to be sent. *A_Len* must be greater than zero and must not end within a variable.	INT

A-outputs	Description	Type
A_Busy	TRUE: Data is still being sent.	BOOLEAN
A_Done	TRUE: Data transmission completed successfully.	BOOLEAN
A_Error	• TRUE: An error occurred • FALSE: No error	BOOLEAN
A_Status	The status and error code of the function block and of the TCP connection are output at the *A_Status* output.	DWORD

21233799/EN – 07/2014

These inputs and outputs of the function block establish the connection to the "Send" function block in the structure tree. The prefix "F" means Field.

INFORMATION

Common variables are used to connect the "Send" function block in the structure tree (in the "Function blocks" folder) with the "TCP_Send" function block (in the user program). These common variables must first be created in 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	BOOLEAN
F_Busy	BOOLEAN
F_Done	BOOLEAN
F_Error	BOOLEAN
F_Status	DWORD

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	INT
F_Len	INT
F_Req	BOOLEAN

Creating the "Send" function block in the structure tree

Do the following 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. Right-click the "Send" function block and open "Edit" from the context menu.
3. The window for assigning variables to the function block opens.

   Connect the inputs 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	INT
LEN	INT
REQ	BOOLEAN

4. 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.

21233799/EN – 07/2014

Outputs	Type
ACK	BOOLEAN
BUSY	BOOLEAN
DONE	BOOLEAN
ERROR	BOOLEAN
STATUS	DWORD

Data	Description
Send data	Any variable can be created in the "Process Variables" tab. The offsets and types of the variables must be identical with the offsets and types of the variables of the communication partner.

Using the "TCP_Send" function block

The following steps are required for using the "TCP_Send" function block:

INFORMATION

The variables to be sent must be created in the "Process Variables" tab of the "Send" dialog box. The offsets and types of the variables must be identical with the offsets and types of the variables of the communication partner.

1. In the user program, set the identification number of the TCP connection at the *A_ID* input.
2. In the user program, set the expected length in bytes of the variables to be sent at the *A_Req* input.
3. In the user program, set the *A_Req* input to TRUE.

INFORMATION

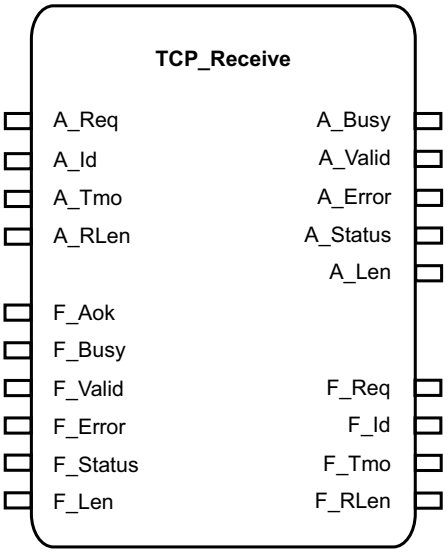
The function block responds to a positive edge on *A_Req*.

The *A_Busy* output is TRUE until the variables have been sent. Afterwards, the *A_Busy* output is set to FALSE and the *A_Done* output to TRUE.

If sending was not successful, the *A_Error* output is set to TRUE, and an error code is output on the *A_Status* output.

21233799/EN – 07/2014

7.7.3 TCP_Receive



12757827339

The "TCP_Receive" function block is used to receive defined variables from the communication partner. A function block with the same variables and offsets, such as "TCP_Send" must be configured in the communication partner.

INFORMATION

To configure the function block, drag it from the function block library and drop it in the user program (see chapter "Configuring function blocks").

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	A positive edge starts the function block.	BOOLEAN
A_Id	Identification number of the configured TCP connection to the communication partner from which data should be received.	INT
A_Tmo	Receive timeout. If no data is received during the timeout, the function block stops and an error message is issued. If the A_Tmo input is not used or is set to zero, the timeout is deactivated.	TIME
A_RLen	*A_RLen* is the expected length of the variables to be received in bytes. *A_RLen* must be greater than zero and must not end within a variable.	INT

A-outputs	Description	Type
A_Busy	TRUE: Data is still being received.	BOOLEAN
A_Done	TRUE: Data has been received successfully.	BOOLEAN

21233799/EN – 07/2014

A-outputs	Description	Type
A_Error	• TRUE: An error occurred • FALSE: No error	BOO-LEAN
A_Status	The status and error code of the function block and of the TCP connection are output at the *A_Status* output.	DWORD
A_Len	Number of received bytes.	INT

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 the structure tree. The prefix "F" means Field.

INFORMATION

Common variables are used to connect the "Receive" function block in the structure tree (in the "Function blocks" folder) with the "TCP_Receive" function block (in the user program). These common variables must first be created in 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	BOOLEAN
F_Busy	BOOLEAN
F_Valid	BOOLEAN
F_Error	BOOLEAN
F_Status	DWORD
F_Len	INT

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	BOOLEAN
F_Id	INT
F_Tmo	TIME
F_RLen	INT

Creating the "Receive" function block in the structure tree

Do the following to create the "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. Right-click the "Receive" function block and open "Edit" from the context menu.

21233799/EN – 07/2014

3. The window for assigning variables to the function block opens.

   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	BOOLEAN
RLEN	INT
TIMEOUT	TIME

4. 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	BOOLEAN
BUSY	BOOLEAN
ERROR	BOOLEAN
LEN	INT
STATUS	DWORD
VALID	BOOLEAN

Data	Description
Receive data	Any variable can be created in the "Process Variables" tab. The offsets and types of the variables must be identical with the offsets and types of the variables of the communication partner.

Using the "TCP_Receive" function block

The following steps are required for using the "TCP_Receive" function block:

INFORMATION

The receive variables must be created in the "Process Variables" tab of the "Receive" dialog box. The offsets and types of the receive variables must be identical with the offsets and types of the variables of the communication partner.

1. In the user program, set the identification number for the TCP connection at the *A_ID* input.
2. In the user program, set the receive timeout at the *A_Tmo* input.
3. In the user program, set the expected length in bytes of the variables to be received at the *A_RLen* input.
4. In the user program, set the *A_Req* input to TRUE.

INFORMATION

The function block responds to a positive edge on *A_Req*.

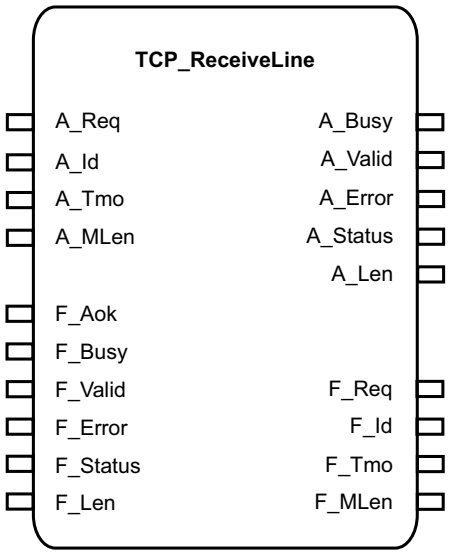
21233799/EN – 07/2014

The *A_Busy* output is set to TRUE until the variables have been received or the receive timeout has expired. Afterwards, the *A_Busy* output is set to FALSE and the *A_Valid* or *A_Error* output is set to TRUE.

If the variables were received successfully, the *A_Valid* output is set to TRUE. The variables defined in the "Data" tab can be evaluated.

If receiving was not successful, the *A_Error* output is set to TRUE, and an error code is output on the *A_Status* output.

7.7.4 TCP_ReceiveLine



12757834891

The "TCP_ReceiveLine" function block is used for receiving an ASCII character string with LineFeed (16#0A) from a communication partner.

INFORMATION

To configure the function block, drag it from the function block library and drop it in the user program (see chapter "Configuring function blocks").

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	A positive edge starts the function block. The connection is ready.	BOOLEAN
A_Id	Identification number of the configured TCP connection to the communication partner from which data should be received.	INT
A_Tmo	Receive timeout. If no data is received during the timeout, the function block stops and an error message is issued. If the input is not used or is set to zero, the timeout is deactivated.	TIME

A-inputs	Description	Type
A_MLen	*A_MLen* is the maximum length of a line to be received in bytes. The receive variables must be created in the "Data" tab in the COM function block. Transmitted bytes = Min (*A_MLen*, line length, length of data range).	INT

A-outputs	Description	Type
A_Busy	TRUE: Data is still being received.	BOOLEAN
A_Valid	TRUE: Data has been received successfully.	BOOLEAN
A_Error	• TRUE: An error occurred • FALSE: No error	BOOLEAN
A_Status	The status and error code of the function block and of the TCP connection are output at the *A_Status* output.	DWORD
A_Len	Number of received bytes.	INT

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 the structure tree. The prefix "F" means Field.

INFORMATION

Common variables are used to connect the "ReceiveLine" function block in the structure tree (in the "Function blocks" folder) with the "TCP_ReceiveLine" function block (in the user program). These common variables must first be created in 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	BOOLEAN
F_Busy	BOOLEAN
F_Valid	BOOLEAN
F_Error	BOOLEAN
F_Status	DWORD
F_Len	INT

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	BOOLEAN
A_Id	INT

21233799/EN – 07/2014

F-outputs	Type
A_Tmo	TIME
A_MLen	INT

Creating the "ReceiveLine" function block in the structure tree

Do the following to create the "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. Right-click the "ReceiveLine" function block and open "Edit" from the context menu.
3. The window for assigning variables to the function block opens.

   Connect the inputs 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
REQ	BOOLEAN
MLEN	INT
TIMEOUT	TIME

4. 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	BOOLEAN
BUSY	BOOLEAN
ERROR	BOOLEAN
LEN	INT
STATUS	DWORD
VALID	BOOLEAN

Data	Description
Receive variables	Create only variables of the type "BYTE" in the "Process Variables" tab. The offsets of the variables must be identical with the offsets of the variables of the communication partner.

Using the "TCP_ReceiveLine" function block

The following steps are required for using the "TCP_ReceiveLine" function block:

INFORMATION

The receive variables of the type "Byte" must be created in the "Process Variables" tab of the "ReceiveLine" dialog box. The offsets of the receive variables must be identical with the offsets of the send variables of the communication partner.

1. In the user program, set the identification number for the TCP connection at the *A_ID* input.
2. In the user program, set the receive timeout at the *A_Tmo* input.
3. In the user program, set the maximum length of the line to be received at the *A_MLen* input.

INFORMATION

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 read process is stopped without error message. The number of received bytes is output to the *A_Len* output:

Received bytes = Min (*A_MLen*, line length, length of data range).

4. In the user program, set the *A_Req* input to TRUE.

INFORMATION

The function block responds to a positive edge on *A_Req*.

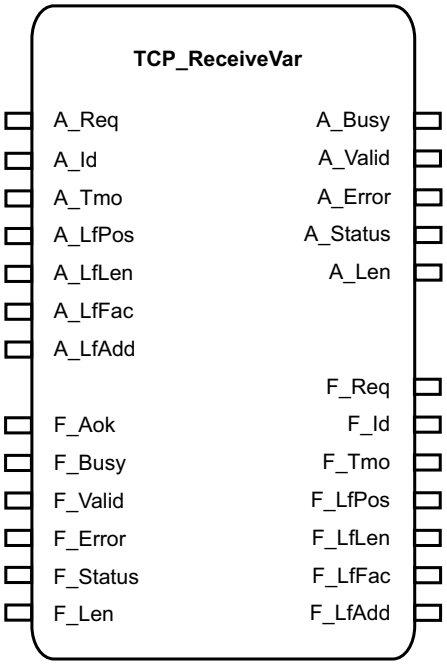
The *A_Busy* output is set to TRUE if the receive buffer is full or the end of line LineFeed is received, or the receive timeout has expired. Afterwards, the *A_Busy* output is set to FALSE and the *A_Valid* or *A_Error* output is set to TRUE.

If the lines were received successfully, the *A_Valid* output is set to TRUE. The variables defined in the "Data" tab can be evaluated.

If the line was not received successfully, the *A_Error* output is set to TRUE, and an error code is output on the *A_Status* output.

21233799/EN – 07/2014

7.7.5 TCP_ReceiveVar



12757935755

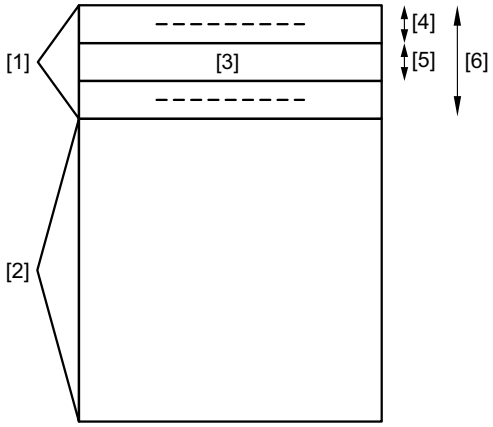
The "TCP_ReceiveVar" function block is used to evaluate data packets of variable length and containing a length field.

INFORMATION

To configure the function block, drag it from the function block library and drop it in the user program (see chapter "Configuring function blocks").

Functional description

The received data packets must have the structure shown in the figure below (for example Modbus protocol). You can adjust them to any protocol format by modifying the input parameters *A_LfPos*, *A_LfLen*, *A_LfFac*, *A_LfLen*.



12757940619

[1] Header
[2] User data range
[3] Length field
[4] A_LfPos
[5] A_LfLen
[6] A_LfAdd

21233799/EN – 07/2014

The received data packet consists of a header and a user data range. The header contains data, such as station address, telegram function, length field, etc. required for establishing communication. To evaluate the user 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 user data range must be read from the length field of the data packet that is currently being read. The position of the length field is entered in the *A_LfPos* parameter. The size of the length field is entered in bytes in *LfLen*. If the length is not specified in bytes, you have to enter the corresponding conversion factor in *A_LfFac* (for example 2 for Word or 4 for Double Word).

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 positive edge starts the CPU function block.	BOO-LEAN
A_Id	Identification number *ID* of the configured TCP connection to the communication partner from which data packet should be received.	INT
A_Tmo	Receive timeout. If no data is received during the timeout, the function block stops and an error message is issued. If the input is not used or is set to zero, the timeout is deactivated.	TIME
A_LfPos	Start position of the length field in the data packet; numbering begins with zero (measured in bytes).	USINT
A_LfLen	Size of the length field *A_LfLen* in bytes. Permitted are 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 is set to zero, "1" is used as default.	USINT
A_LfAdd	Size of the header in bytes.	USINT

A-outputs	Description	Type
A_Busy	TRUE: Data is still being received.	BOO-LEAN
A_Valid	TRUE: Data has been received successfully.	BOO-LEAN
A_Error	• TRUE: An error occurred while reading. • FALSE: No error	BOO-LEAN
A_Status	The status and error code of the function block and of the TCP connection are output at the *A_Status* output.	DWORD
A_Len	Number of received bytes.	INT

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 the structure tree. The prefix "F" means Field.

21233799/EN – 07/2014

INFORMATION

Common variables are used to connect the "ReceiveVar" function block in the structure tree (in the "Function blocks" folder) with the "TCP_ReceiveVar" function block (in the user program). These common variables must first be created in 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	BOOLEAN
F_Busy	BOOLEAN
F_Valid	BOOLEAN
F_Error	BOOLEAN
A_Status	DWORD
A_Len	INT

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	BOOLEAN
F_Id	INT
F_Tmo	TIME
F_LfPos	USINT
A_LfLen	USINT
A_LfFac	USINT
A_LfAdd	USINT

Creating the "ReceiveVar" function block in the structure tree

Do the following 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. Right-click the "ReceiveVar" function block and open "Edit" from the context menu.
3. The window for assigning variables to the function block opens.

   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
LfAdd	USINT
LfFac	USINT

21233799/EN – 07/2014

Inputs	Type
LfLen	USINT
LfPos	USINT
REQ	BOOLEAN
TIMEOUT	TIME

4. Connect the outputs 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	BOOLEAN
BUSY	BOOLEAN
ERROR	BOOLEAN
LEN	INT
STATUS	DWORD
VALID	BOOLEAN

Data	Description
Receive variables	Any variable can be created in the "Process Variables" tab. The offsets and types of the variables must be identical with the offsets and types of the variables of the communication partner.

Using the "TCP_ReceiveVar" function block

The following steps are required for using the "TCP_ReceiveVar" function block:

INFORMATION

The receive variables must be created in the "Process Variables" tab of the "ReceiveVar" dialog box. The offsets and types of the receive variables must be identical with the offsets and types of the send variables of the communication partner.

1. In the user program, set the identification number for the TCP connection at the *A_ID* input.
2. In the user program, set the receive timeout at 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.

INFORMATION

The function block responds to a positive 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, the *A_Busy* output is set to FALSE and the *A_Valid* or *A_Error* output is set to TRUE.

21233799/EN – 07/2014

If the variables were received successfully, the *A_Valid* output is set to TRUE. The variables defined in the "Data" tab can be evaluated. The *A_Len* output contains the number of bytes that has actually been read.

If receiving was not successful, the *A_Error* output is set to TRUE, and an error code is output on the *A_Status* output.

7.8 Control panel (send/receive over TCP)

The control panel can be used to verify and control the settings for the send/receive protocol. Current status information of the send/receive protocol (such as disturbed connections, etc.) is also displayed.

Do the following to open the control panel to monitor the send/receive protocol:

1. In the structure tree, select [Resource].
2. Click "Online" on the action bar.
3. In the System Login window, enter the access data to open the control panel for the resource.
4. In the structure tree for the control panel, choose [Send/Receive Protocol].

7.8.1 Display field for general parameters

The display field shows the following values of the send/receive protocol:

Element	Description
Name	TCP SR protocol.
µP load (planned) in %	See chapter "Load limitation".
µP load (actual) in %	See chapter "Load limitation".
Undisturbed connections	Number of undisturbed connections.
Disturbed connections	Number of disturbed connections.

7.8.2 Display field for TCP connections

The display field shows the following values of the selected TCP connections:

Element	Description
Name	TCP connection.
Partner timeout	• Yes: Partner request timeout expired • No: Partner request timeout not expired
Connection state	Current state of this connection. • 0x00: Connection OK • 0x01: Connection terminated • 0x02: Server waits for connection to be established • 0x04: Client attempts to establish connection • 0x08: Connection is blocked
Peer address	IP address of the communication partner.
Peer port	Port of the communication partner.

21233799/EN – 07/2014

Element	Description
Own port	Port of this controller.
Watchdog time in ms	Current partner request timeout during which the communication partner received data at least once after data has been sent.
Error code	See chapter "Error code of the TCP connection".
Timestamp error code in ms	Timestamp for the last reported error. Range of values: Seconds since 1.1.1970 in milliseconds
Received bytes	Number of bytes received in this TCP connection.
Sent bytes	Number of bytes sent in this TCP connection.

7.8.3 Error code of the TCP connection

The error codes can be read from the *Error Code* variable. For each configured connection, the connection state comprises 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	Unknown device
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 is busy
22	16#16	Invalid value (for example 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#2 B	Protocol not supported
45	16#2D	Operation on socket not supported
47	16#2F	Address not supported by protocol

21233799/EN – 07/2014

Error code (decimal)	Error code (hexadecimal)	Description
48	16#30	Address already in use
49	16#31	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#36	Socket already connected
57	16#39	Socket not connected
58	16#3 A	Socket closed
60	16#3C	Operation time expired
61	16#3D	Connection rejected (by peer)
65	16#41	No route to peer
78	16#4E	Function not available
254	16#FE	Timeout occurred
255	16#FF	Connection closed by peer

7.8.4 Additional error code table for the function blocks

The error codes for the function blocks are only output to *A_Status* of the send and receive TCP function blocks.

Error code (decimal)	Error code (hexadecimal)	Description
129	16#81	No connection exists with this identifier
130	16#82	Length greater than or equal to zero
131	16#83	Only cyclic data are permitted for this connection
132	16#84	Invalid state
133	16#85	Timeout value too large
134	16#86	Internal program error
135	16#87	Configuration error
136	16#88	Transmitted data do not match data area
137	16#89	Function block stopped
138	16#8 A	Timeout occurred or transmission blocked
139	16#8 B	Another function block of this type is already active on this connection

21233799/EN – 07/2014

7.8.5 Connection state

Error code (hexadecimal)	Description
16#00	Connection OK
16#01	Connection terminated
16#02	Server waits for connection to be established
16#04	Client attempts to establish connection
16#08	Connection is blocked

7.8.6 Partner connection state

Protocol state (decimal)	Description
0	No connection
1	Connection OK

21233799/EN – 07/2014

8 Simple Network Time Protocol (SNTP)

With the SNTP protocol (Simple Network Time Protocol), the SNTP server synchronizes the time on the SNTP clients via Ethernet. The safety controller can be configured and used as an SNTP server and/or SNTP client.

The SNTP standard according to RFC 2030 (SNTP version 4) applies, with the limitation that only Unicast mode is supported.

- The function is enabled by default.
- Ethernet 10/100-BASE-T is required as the network transmission standard.

8.1 SNTP client

The SNTP client uses only the accessible SNTP server with the highest priority for time synchronization.

One SNTP client in each resource can be configured for time synchronization.

INFORMATION

Time synchronization of one safety controller by another safety controller.

If an SNTP client is set up on a safety controller, the safety controller's internal SNTP server is turned off.

8.1.1 Creating a new SNTP client

Do the following to create a new SNTP client:

1. In the structure tree, open [Configuration] / [Resource] / [Protocols].
2. Right-click "Protocols" and choose [New]/[SNTP client] from the context menu.
   A new SNTP client is added.
3. Select the COM module in the context menu of [SNTP client]/[Properties].

The SNTP client dialog window contains the following parameters:

Parameter	Description
Type	SNTP client.
Name	Name for the SNTP client, limited to a maximum of 32 characters
Module	Selecting the CPU or COM module where this protocol is processed.
Behavior on CPU/COM connection loss	If the connection between the processor module and the communication module is lost, the input variables are initialized or continue to be used unchanged in the processor module, depending on this parameter. (For example, if the communication module is disconnected while communication is in progress.) **Adopt initial data:** Input variables are reset to their initial values. **Retain last value**: Input variables retain their last values.

21233799/EN – 07/2014

Parameter	Description
Activate max. µP budget	Is disregarded by the module's operating system. Parameter was maintained for CRC and reload stability.
Max. µP budget in [%]	Is disregarded by the module's operating system. Parameter was maintained for CRC and reload stability.
Description	Any unique description for the SNTP.
Current SNTP version	Displays the current SNTP version.
Reference stratum	The stratum of an SNTP client indicates how precise the client's local time is. The lower the stratum, the more precise its local time. Zero means an unspecified or unavailable stratum (not valid). The SNTP server currently used by an SNTP client is the server that is accessible and has the highest priority. If the stratum of the current SNTP server is smaller than that of the SNTP client, the resource adopts the time of the current SNTP server. If the stratum of the current SNTP server is larger than that of the SNTP client, the resource does not adopt the time of the current SNTP server. If the stratum of the current SNTP server is the same as that of the SNTP client, there are two different possibilities: • If the SNTP client (resource) only operates as an SNTP client, the resource adopts the time of the current SNTP server. • If the SNTP client (resource) also functions as an SNTP server, the resource adopts half of the time difference to the current SNTP server for each SNTP client request (time adjusts gradually). Range of values: 16 s – 16384 s (default: 16 s)
Client request time interval [s]	Time interval during which the current SNTP server synchronizes time. The client request time interval in the SNTP client must be greater than the timeout in the SNTP server. Range of values: 16 s – 16384 s (default: 16 s)

8.2 SNTP client (server info)

The connection to the SNTP server is configured in the SNTP server info. One to four pieces of SNTP server info can be configured under an SNTP client.

8.2.1 Creating new SNTP server info

Do the following to create a new SNTP server info:

1. Open [Configuration]/[Resource]/[Protocols]/[SNTP client] in the structure tree.
2. Right-click "Protocols" and choose [New]/[SNTP server info] from the context menu.

   A new SNTP server info is added.

21233799/EN – 07/2014

3. Select the COM module in the context menu of [SNTP server info]/[Properties].

The SNTP server info dialog window contains the following parameters:

Parameter	Description
Type	SNTP server info.
Name	Name for the SNTP server info. Limited to a maximum of 31 characters.
Description	Description of the SNTP server. Limited to a maximum of 31 characters.
IP address	IP address of the resource or the PC on which the SNTP server is configured. Default: 0.0.0.0
SNTP server priority	Priority with which the SNTP client handles this SNTP server. SNTP servers configured for a single SNTP client should have different priorities. Range of values: 0 (lowest priority) through 4294967295 (highest priority) Default: 1 s
SNTP server timeout [s]	The timeout in the SNTP server must be set lower than the value for the time request interval in the SNTP client. Range of values: 1 s – 16384 s Default: 1 s

8.3 SNTP server

The SNTP server accepts the request from the SNTP client and sends its current time back to the SNTP client.

8.3.1 Creating a new SNTP server

Do the following to create a new SNTP server:

1. In the structure tree, open [Configuration] / [Resource] / [Protocols].
2. Right-click "Protocols" and choose [New] / [SNTP server] from the context menu.

   A new SNTP server is added.
3. Select the COM module in the context menu of [SNTP server] / [Properties].

The SNTP server dialog window contains the following parameters:

Parameter	Description
Type	SNTP server.
Name	Name for the SNTP server, limited to a maximum of 31 characters.
Module	Selecting the CPU or COM module where this protocol is processed.

Parameter	Description
Activate max. µP budget	**Activated:** Adopt limitation of the µP budget from the field max. µP budget in [%]. **Deactivated:** Do not use a µP budget limit for this protocol.
Max. µP budget in [%]	Maximum µP load for the module that can be used for processing the protocols. Range of values: 1 – 100% Default: 30%
Behavior on CPU/COM connection loss	If the connection between the processor module and the communication module is lost, the input variables are initialized or continue to be used unchanged in the processor module, depending on this parameter. (For example, if the communication module is disconnected while communication is in progress.) **Adopt initial data:** Input variables are reset to their initial values. **Retain last value**: Input variables retain their last values.
Description	Any unique description for the SNTP.
Current SNTP version	Displays the current SNTP version.
Time server stratum	The stratum of an SNTP client indicates how precise the client's local time is. The lower the stratum, the more precise its local time. Zero means an unspecified or unavailable stratum (not valid). The SNTP server stratum must be lower than or the same as the stratum of the querying SNTP client. Otherwise the SNTP server's time will not be applied to the SNTP client. Range of values: 1 – 15 Default: 14

21233799/EN – 07/2014

9 Com user task (CUT)

In addition to the user program created with SILworX®, a C program can also be run on the controller. This non-safe C program runs as a Com user task and does not impact the safe processor module of the controller's communication module.

The Com user task has its own cycle which is independent of the CPU cycle.

9.1 CUT features

The following table describes the features of the CUT

Element	Description
Com user task	One Com user task can be configured for each safety controller.
Safety-related	No

9.2 Requirements

You will need the following to set up a SILworX® program with a Com user task:

Designation	Part number
SILworX® for MOVISAFE® HM31 • Hardware: SILworX® license dongle • Software: SILworX® 4.64.0 or later	19500114
Motion Library MOVISAFE® HM31 Function block library for safety-related position and velocity detection	17106400
Com User Task MOVISAFE® HM31 Refer to the "Com User Task for MOVISAFE® HM31" manual for detailed information.	28202430

- The above software is **not** included in the delivery:

  You can order this software together with the documentation on a data storage medium (CD/DVD) from SEW-EURODRIVE by quoting the above-mentioned order numbers.
- You need the MOVIVISION® Parameter and Diagnostics Tool Version 2.0 software for diagnostics for Com user task applications (not included in the delivery).

21233799/EN – 07/2014

10 Operating system

The operating system includes all basic safety controller functions.

The user functions that each PES is to execute are set in the user program. A code generator translates the user program into machine code. The programming tool transfers this machine code into the controller's flash memory.

10.1 Processor operating system functions

The main functions of the operating system for the processor system and the connections to the user program are shown in the table below.

Operating system functions	Connections to the user program
Cyclical processing of the user program.	Acts on variables, function modules.
Configuration of the programmable controller.	Set by selecting the controller.
Processor tests.	—
I/O module tests.	—
Responses in case of error.	Fixed. The user program is responsible for process responses.
Diagnostics for processor system and inputs/outputs.	Use of system variables for error messages.
Safe communication: • Peer-to-peer Non-safe communication: • Modbus	Defining the use of communication variables.
PADT interface: • Permitted actions	Set in the programming tool: • Configuration of protective functions • User login

Each operating system is verified by the relevant TÜV (German Technical Control Board) and approved for operation with safety-related controllers. Each valid version of the operating system and its associated signatures (CRCs) are documented in a list that SEW-EURODRIVE creates together with the German Technical Control Board.

10.2 Behavior if errors occur

The response to errors that have been identified by tests is important. The following types of errors can occur:

- Permanent input and output errors
- Temporary input and output errors
- Internal errors

21233799/EN – 07/2014

10.2.1 Permanent input and output errors

An error that occurs in an input or output channel has no effect on the controller. The operating system only regards the channel as faulty, not the entire controller. The other safety functions are not affected by this and remain active.

In case of faulty input channels, the operating system transfers the safe value “0” or the initial value to processing.

The operating system switches off faulty output channels. If it is not possible to disable only one channel, the entire output module is regarded as faulty.

The operating system sets the error status signal and notifies the user program of the error type.

If the controller cannot disable the corresponding output and also the second path for disabling the output is ineffective, the controller goes to STOP. The processor system’s watchdog then disables the outputs.

If errors in the I/O modules have been pending for more than 24 hours, the controller permanently disables only the corresponding I/O modules.

10.2.2 Temporary input and output errors

If an error occurs in an input or output module and disappears on its own, the operating system resets the error status and resumes regular operation.

The operating system statistically analyzes the frequency with which errors occur. It sets the module status to permanently faulty if the configured error frequency is exceeded. This causes the module to stop working even after the error disappears. When the controller’s operating state is changed from STOP to RUN, the module is enabled and the error statistic is cleared. This change acknowledges the fault in the module.

10.2.3 Internal errors

INFORMATION

On the rare occasion that a safety controller discovers an internal error, the following error response is performed:

The safety controller is automatically rebooted.

10.3 The processor system

The processor system is the central component of the controller and communicates with the I/O modules within the controller via the I/O bus.

The processor system monitors the process and the logically correct execution of the operating system and the user program. The following functions are monitored with respect to time:

- Self-tests for the hardware and software of the processor system
- RUN cycle of the processor system (including user program)
- I/O tests and processing of I/O signals

21233799/EN – 07/2014

10.3.1 Operating states of the processor system

LEDs on the front panel of the controller indicate the operating state of the processor system. The programming device can also indicate the operating state together with other parameters of the processor module and the user program.

Stopping the processor interrupts the execution of the user program and sets the outputs of the controller and all remote I/Os to safe values.

Setting the EMERGENCY SWITCHING OFF system parameter to TRUE using a program logic makes the processor system enter STOP state. The table below provides a summary of the most important operating states.

Operating mode	Description
INIT	Safe state of the processor system during initialization. Hardware and software tests are performed.
STOP/VALID CONFIGURATION	Safe state of the processor system without execution of a user program. All outputs of the controller are reset. Hardware and software tests are performed.
STOP/INVALID CONFIGURATION	Safe state of the processor system without loaded configuration or after a system error. All outputs of the controller are reset and the hardware watchdog is not triggered. The processor system can only be rebooted using the PADT.
RUN	The processor system is active: • The user program is executed, I/O signals are processed. • The processor system performs safety-related and non-safety-related communication (if configured). • Hardware and software tests are performed as well as tests for configured I/O modules.

10.3.2 Programming

A PADT (programming device) is used for programming the controller. The programming device is a PC with installed SILworX® programming tool.

SILworX® supports the following programming languages according to IEC 61131-3:

- Function block language
- Sequential function chart (SFC)

The programming tools are suited for creating safety-related programs and for operating the controller. For more information on the programming tools, refer to the SILworX® online help.

21233799/EN – 07/2014

11 User program

The PES user program must be created and loaded using a programming device on which the SILworX® programming tool has been installed according to the requirements of IEC 61131-3.

The user program must first be created and configured for safety-related operation of the controller using the programming tool. Observe the requirements in the "Decentralized Safety Controller MOVISAFE® HM31" safety manual.

After the program has been compiled, the programming device loads the user program (logic) and the configuration (connection parameters such as IP address, subnet mask and system ID) into the controller, and starts the controller.

The programming device offers the following options for working with the controller while the controller is in operation:

- Starting and stopping the user program
- Displaying and forcing variables with the force editor
- Step-by-step execution of the user program in test mode – not in safety-related operation
- Reading the diagnostics history

The programming device must have the same user program as the controller for the diagnostics history to be read.

The following optional functions are available for the user program:

- **Multitasking**

  Multitasking refers to the capability of the safety controller to process up to 32 user programs within the processor module.

  This allows for separating sub-functions within a single project. Individual user programs can be started, stopped, and reloaded independently of one other.

- **Reload**

  If changes were made to the user programs, these changes can be transferred to the PES during ongoing operation. The operating system verifies and activates the modified user program, which then takes over the control function.

INFORMATION

In MOVISAFE® HM31, the optional functions can be used for testing purposes without activation for 5000 hours of operation. The "ERROR" system LED lights up red continuously when functions are used that were not enabled.

After 5000 hours of operation have elapsed, the controller no longer starts.

- Therefore, order the license for enabling the required functions in time.

11.1 User program operating modes

Only one user program at a time can be loaded into each controller. The user program can be in one of the following operating modes:

Operating mode	Description
RUN	The processor system is in RUN operating mode. The user program is executed cyclically; I/O signals are processed.

21233799/EN – 07/2014

Operating mode	Description
Test mode (individual step)	The processor system is in RUN operating mode. The user program is executed cyclically by manual request; I/O signals are processed. **Not allowed for safety-related operation.**
STOP	The processor system is in STOP operating mode. The user program is not (or no longer) executed; the outputs are reset.
Error	A loaded user program has been stopped because of an error. The outputs are reset. Note: The program can only be restarted using the PADT.

11.2 Multitasking

Multitasking refers to the capability of the safety controller to process up to 32 user programs within the processor module. This allows for separating sub-functions within a single project. Individual user programs can be started, stopped, and reloaded independently of one other. SILworX® shows the states of the individual user program in the control panel and allows the user to operate them.

11.2.1 CPU cycle without multitasking

The cycle of the processor module (CPU cycle) for only one user program comprises the following phases (simplified):

1. Processing input data
2. Executing the user program
3. Supplying output data

Special tasks (such as reload), which might be performed within a CPU cycle, are not depicted.

11.2.2 CPU cycle with multitasking

With multitasking, the second phase differs and results in the following CPU cycle:

1. Processing input data
2. Executing all user programs
3. Supplying output data

In the second phase, the safety controller can process up to 32 user programs at a time. Two cases are possible for each user program:

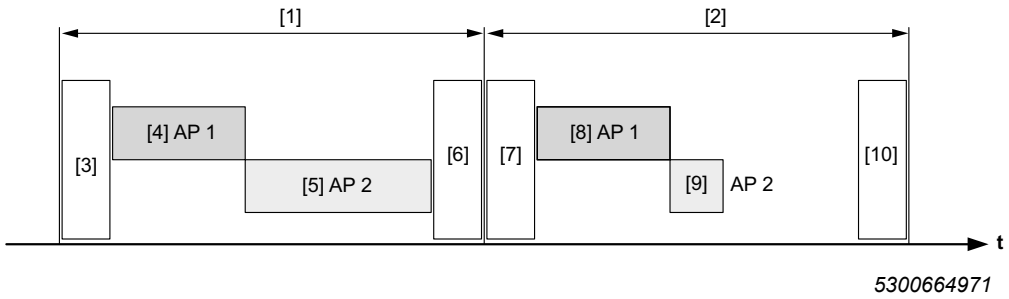
- A complete user program cycle is processed within a single CPU cycle
- A complete user program cycle requires several CPU cycles for processing

The two cases are also possible if only one user program is available.

It is not possible to exchange global data between user programs within a single CPU cycle. Data written by a user program is provided immediately before phase 3 and after complete execution of the user program. This means that this data can only be used as input values when another user program is started.

21233799/EN – 07/2014

The figure below shows an example of both cases in a project containing two user programs (AP 1 and AP 2).



5300664971

[1] First CPU cycle examined.

[2] Second CPU cycle examined.

[3] Input processing in the first CPU cycle.

[4] First portion of the examined cycle of user program 1 (AP 1).

[5] First portion of the examined cycle of user program 2 (AP 2).

[6] Output processing in the first CPU cycle.

[7] Input processing in the second CPU cycle.

[8] Second portion of the examined cycle of user program 1 (AP 1).

[9] Second portion of the examined cycle of user program 2 (AP 2).

[10] Output processing in the second CPU cycle.

Each cycle of user program 1 (AP 1) is processed completely during each CPU cycle. AP 1 processes an input change that the system has noticed at the beginning of the CPU cycle [1], and provides a response at the end of the cycle.

One cycle of the AP 2 user program requires two CPU cycles to be processed. AP 2 also requires the second CPU cycle [2] for processing an input change noticed by the system at the beginning of the first CPU cycle [1]. This is the reason why the response to this input change is only available at the end of the second CPU cycle [2]. The response time of AP 2 is twice as long as that of AP 1.

Upon completion of the first part [5] of the examined cycle of AP 2, the processing of AP 2 is aborted **completely** and is not continued until [9] starts. During its cycle, AP 2 processes the data provided by the system during [3]. The results of AP 2 are available to the system during [10] (for example for process output). The data that is exchanged between user program and system is always consistent.

Program processing can be controlled by assigning a priority that indicates the importance of a user program compared to the others (see multitasking mode 2).

To specify the user program execution order, set the following parameters in the resources and programs or in the Multitasking Editor:

INFORMATION

Multitasking can only be used by activating a license.

Parameter	Meaning	Can be set for
Max. duration per cycle [µs]	Time permitted for executing the user program within a CPU cycle.	User program, Multitasking Editor

Parameter	Meaning	Can be set for
Program ID	ID for identifying the program when displayed in SILworX®.	User program, Multitasking Editor
Watchdog time	Watchdog time of the resource	Resource, Multitasking Editor
Target cycle time [ms]	Required or maximum cycle time.	Resource
Multitasking mode	Using the execution duration not required by the user program, which means the difference between actual execution duration within one CPU cycle and the defined *Max. duration per cycle [µs]*. Mode 1: The duration of a CPU cycle depends on the execution time required by all user programs. Mode 2: The processor makes available the execution time not required by user programs with a low priority to user programs with a high priority. Operating mode for high availability. Mode 3: During the execution time not required by the user programs, the processor waits for the time to expire and in this way extends the cycle.	Resource, Multitasking Editor
Target cycle time mode	Using the *Target cycle time [ms]*.	Resource
Priority	Importance of a user program, highest priority: 0.	User program, Multitasking Editor
Maximum number of cycles	Maximum number of CPU cycles required for processing one user program cycle.	Multitasking Editor

Bear in mind the following rules when specifying the parameters:

- If *Max. duration per cycle [µs]* is set to "0", the execution time of the user program is not limited. This means the user program is always executed completely. Therefore, the number of cycles may be set to "1" in this case.

21233799/EN – 07/2014

- The sum of the *Max. duration per cycle [µs]* parameters of all user programs must not exceed the resource watchdog time. Provide for a sufficient reserve for processing the remaining system tasks.
- The sum of the *Max. duration per cycle [µs]* parameters of all user programs must be large enough to include a reserve for maintaining the target cycle time.
- The *program IDs* of all user programs must be unique.

SILworX® monitors compliance with the rules during verification and code generation. Also adhere to these rules when changing the parameters online.

SILworX® uses these parameters to calculate the watchdog time of the user program:

Watchdog time of the user program = w*atchdog time* × *maximum number of cycles*

INFORMATION

The sequence control for executing the user programs operates in steps of 250 µs. The values set for *Max. duration per cycle [µs]* can therefore be exceeded or undershot by up to 250 µs.

The individual user programs generally run without interfering one another. However, reciprocal influence can be caused in the following cases:

- When using the same global variables in several user programs.
- When runtimes of individual user programs are longer than predicted if a limit is not set in *Max duration per cycle.*

NOTICE

Reciprocal influence of user programs.

Using the same global variable in several user programs might result in reciprocal influence of the user programs and can have various consequences.

- Carefully plan the use of the same global variables in several user programs.
- Use the cross references in SILworX® to check the use of global data. Global data may only be written with values either in a user program or from the hardware.

INFORMATION

SEW-EURODRIVE recommends that you set the *Max. duration per cycle [µs]* parameter to an appropriate value ≠ 0. This setting ensures that a user program with an excessively long runtime is stopped during the current CPU cycle and is resumed in the next cycle without affecting the other user programs.

An unusually long runtime of one or several user programs can result in exceeding the target cycle time or even the watchdog time of the resource, and consequently leads to a fault stop.

The operating system specifies the execution sequence of the user programs as follows:

- The system processes user programs with lower priority before user programs with higher priority.
- If user programs have the same priority, the system executes them according to their *program ID* in ascending order.

This sequence also applies to starting and stopping the user programs during the start and stop of the PES.

11.2.3 Multitasking mode

You can choose one of three multitasking modes for each resource. The modes differ in the utilization of the execution times that are not required for executing the CPU cycle of the user programs.

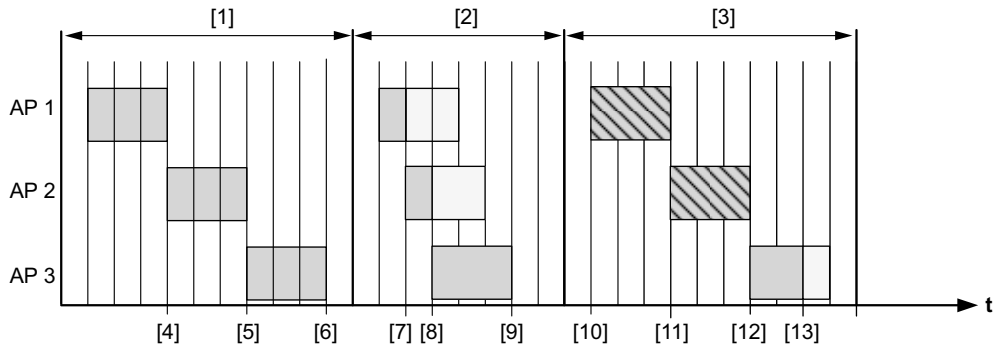
INFORMATION

In the following examples of multitasking modes, the input and output processing is indicated by empty areas at the beginning and end of each CPU cycle.

- Multitasking mode 1

  Multitasking mode 1 uses the unneeded time to reduce the CPU cycle. Once the user program is completely processed, the next user program is processed immediately. This results in a shorter cycle in total.

  Example: Three user programs (AP 1, AP 2 and AP 3) where a cycle of the user program may last up to 3 CPU cycles.



5300668683

[1] First CPU cycle examined.

[2] Second CPU cycle examined.

[3] Third CPU cycle examined.

[4] *Max. duration per cycle [µs]* of AP 1 has expired, AP 2 starts.

[5] *Max. duration per cycle [µs]* of AP 2 has expired, AP 3 starts.

[6] *Max. duration per cycle [µs]* of AP 3 has expired, completion of the first CPU cycle.

[7] Completion of the AP 1 cycle, AP 2 is resumed.

[8] Completion of the AP 2 cycle, AP 3 is resumed.

[9] *Max. duration per cycle [µs]* of AP 3 has expired, completion of the second CPU cycle.

[10] Next user program cycle of AP 1 starts.

[11] *Max. duration per cycle [µs]* of AP 1 has expired, next user program cycle of AP 2 starts.

[12] *Max. duration per cycle [µs]* of AP 2 has expired, AP 3 starts.

21233799/EN – 07/2014

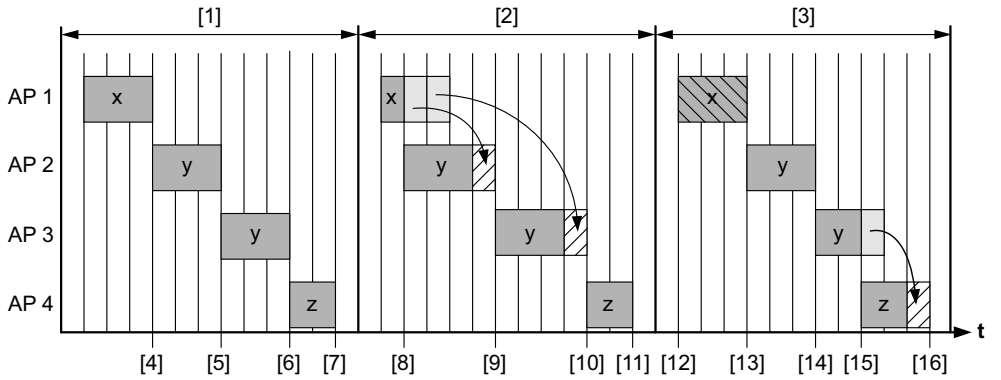
[13] Completion of the AP 3 cycle.

- Multitasking mode 2

  Multitasking mode 2 distributes the unneeded duration of lower-priority user programs among higher-priority user programs. This means that programs with higher priority have available a portion of the unneeded duration in addition to the set *max. duration per cycle [µs]*. This procedure ensures a high availability.

  Four user programs (AP 1 – AP 4) are used in the following example. The following priorities are assigned to the user programs:

  – AP 1 has the lowest priority x
  – AP 2 and AP 3 have medium priority y
  – AP 4 has the highest priority z



5301376139

[1] First CPU cycle examined.

[2] Second CPU cycle examined.

[3] Third CPU cycle examined.

[4] *Max. duration per cycle [µs]* of AP 1 has expired, AP 2 starts.

[5] *Max. duration per cycle [µs]* of AP 2 has expired, AP 3 starts.

[6] *Max. duration per cycle [µs]* of AP 3 has expired, AP 4 starts.

[7] *Max. duration per cycle [µs]* of AP 4 has expired, completion of the first CPU cycle.

[8] Completion of the AP 1 cycle, AP 2 is resumed. The remaining duration is distributed (see arrows) to the *Max. duration per cycle [µs]* of AP 2 and AP 3 (higher priority y).

[9] *Max. duration per cycle [µs]* of AP 2 plus portion of remaining duration of AP 1 have expired, AP 3 is resumed.

[10] *Max. duration per cycle [µs]* of AP 3 plus portion of remaining duration of AP 1 have expired, AP 4 starts.

[11] *Max. duration per cycle [µs]* of AP 4 has expired, completion of the second CPU cycle.

[12] Next user program cycle of AP 1 starts.

[13] *Max. duration per cycle [µs]* of AP 1 has expired, AP 2 is resumed.

[14] *Max. duration per cycle [µs]* of AP 2 has expired, AP 3 is resumed.

[15] Completion of the AP 3 cycle, AP 4 is resumed. The remaining duration is added to AP 4 (higher priority z).

21233799/EN – 07/2014

[16] *Max. duration per cycle [µs]* of AP 4 plus remaining duration of AP 3 have expired, completion of the third cycle.

INFORMATION

The unused execution time of user programs that were not executed cannot be used as remaining time for other user programs.

User programs are not executed in one of the following states:

- STOP
- ERROR
- TEST_MODE

An increase in the number of CPU cycles required for processing the cycle of another user program might be the consequence.

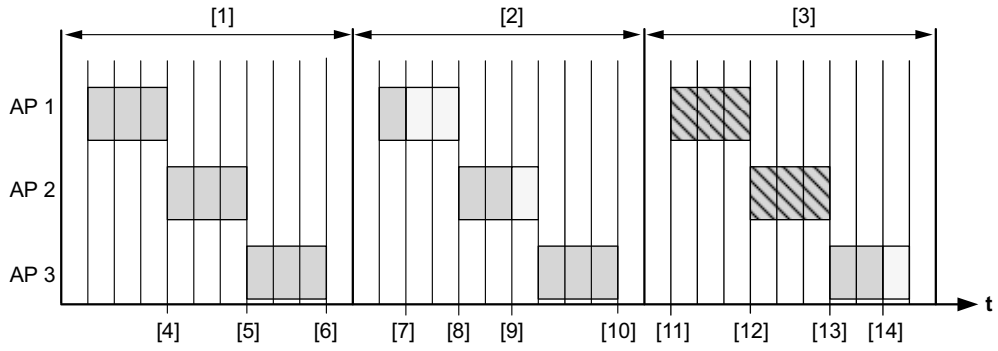
In this case, the maximum processing time of the user program might be exceeded and results in a fault stop if the value for the maximum number of cycles was set too low.

Maximum processing time = max. duration per cycle [µs] × maximum number of cycles. Use multitasking mode 3 to verify the parameter setting.

- Multitasking mode 3

  Multitasking mode 3 does not use the unneeded duration for executing user programs but waits until the *Max. duration per cycle [µs]* of the user program, and then starts processing the next user program. This behavior results in CPU cycles of the same duration. Multitasking mode 3 allows users to verify whether multitasking mode 2 ensures proper program execution even in the worst case scenario.

  Example:



5301379595

[1] First CPU cycle examined.

[2] Second CPU cycle examined.

[3] Third CPU cycle examined.

[4] *Max. duration per cycle [µs]* of AP 1 has expired, AP 2 starts.

[5] *Max. duration per cycle [µs]* of AP 2 has expired, AP 3 starts.

[6] *Max. duration per cycle [µs]* of AP 3 has expired, completion of the first CPU cycle. User program 1 (AP 1) is resumed.

[7] Completion of the AP 1 cycle, AP 2 is resumed. The remaining duration is started.

21233799/EN – 07/2014

[8] *Max. duration per cycle [µs]* of AP 1 has expired, AP 2 is resumed.

[9] Completion of the AP 2 cycle. The remaining duration is waited.

[10] *Max. duration per cycle [µs]* of AP 3 has expired, completion of the second CPU cycle.

[11] Next user program cycle of AP 1 starts.

[12] Next *Max. duration per cycle [µs]* of AP 2 has expired, next AP 2 cycle starts.

[13] *Max. duration per cycle [µs]* of AP 2 has expired, AP 3 is resumed.

[14] Completion of the AP 3 cycle, wait time until the *Max. duration per cycle [µs]* of AP 3 has expired. Completion of the third CPU cycle.

11.3 Reload

If changes were made to the user programs, these changes can be transferred to the PES during operation. The operating system checks and activates the modified user program, which then takes over the control task.

INFORMATION

Adhere to the following information when reloading step chains:

The reload information for step chains does not take account of the current sequence status. This is the reason why the step sequence might be changed accordingly and set to an undefined state when performing a reload. The user is responsible for this action.

- Deleting the active step. As a result, no step of the step chain has the state *active*.
- Renaming the initial step while another step is active. As a result, the step chain has two active steps.

INFORMATION

Adhere to the following information when reloading actions:

During reload, actions are loaded with their entire data. Analyze the consequences carefully before performing a reload.

- Deleting a timer action qualifier due to the reload causes the timer to expire immediately. As a result, the Q output can change to TRUE depending on the other settings.
- When deleting the action qualifier (for example the *S* action qualifier) for a set element, the element remains set.
- Deleting a *P0* action qualifier set to TRUE, actuates the trigger function.

Before performing a reload, the operating system checks whether the required additional tasks would increase the cycle time of the current user programs to such an extent that the defined watchdog time is exceeded. In this case, the reload process is aborted with an error message and the controller continues to operate with the previous project configuration.

INFORMATION

The controller can abort a reload process.

To achieve a successful reload, plan a sufficient reserve for the reload when determining the watchdog time, or increase the controller watchdog time temporarily by a reserve.

Any temporary increase of the watchdog time must be agreed upon with the responsible test authority. Exceeding the target cycle time can result in aborting the reload.

The reload can only be performed if the *Reload Allowed* system parameter is set to "ON" and the *Reload Deactivation* system variable is set to "OFF".

INFORMATION

The user is responsible for ensuring that the watchdog time includes a sufficient reserve time. This should allow the user to manage the following situations:

- Variations in the cycle time of the user program.
- Sudden, strong cycle loads, for example due to communication.
- Expiration of time limits during communication.

During a reload, the global and local variables are assigned the values of the corresponding variables from the previous project version. Names of local variables contain the instance name of the program organization unit (POE, according to IEC 61131).

This procedure has the following consequences if names are changed and are loaded to the PES using a reload:

- Renaming a variable has the same effect as deleting the variable and creating a new one, which means that doing so results in an initialization process also in the case of retain variables. As a consequence, the variables lose their current values.
- Renaming a function block instance results in initializing all variables, even retain variables, and all contained function block instances.
- Renaming a program results in initializing all contained variables and function block instances.

This behavior can have unintended effects on one or several user programs and consequently on the plant to be controlled.

11.3.1 Conditions for using the reload function

A license is required to use the reload function.

By performing a reload, the following project modifications can be transferred to the controller:

- Changes to the user program parameters.
- Changes to the logic of the program, function blocks, and functions.
- Changes that allow a reload according to the table below.

Changes made to	Type of change			
	Add	Delete	Change initial value	Assign other variable
Assigning global variables to				
• User programs	X	X	X	X

21233799/EN – 07/2014

Changes made to	Type of change			
	Add	Delete	Change initial value	Assign other variable
• System variables	X	X	X	X
• I/O channels	X	X	X	X
• Communication protocols	—	—	—	—
• safeethernet	—	—	X	—
• SER	—	—		
Communication protocols	—	—	n.a.	n.a.
User programs	X	X[1)]	n.a.	n.a.
System rack, rack ID	—			
IP addresses	—			
User accounts and licenses	X			

1) Reload possible but at least one user program must remain in the controller

X Reload possible

— Reload not possible

n.a. Not applicable

A reload may only be performed in accordance with the above described conditions. In all other cases, stop the controller and perform a download.

INFORMATION

To be able to perform a reload even if global variable assignments have been added, do the following:

- Assign unused global variables to communication protocols already when creating the user program.
- Assign a safe value to the unused global variables as initial value.

In this way, you later only have to change the assignments instead of adding them so that you can perform a reload.

11.4 General information about forcing

Forcing means that the current value of a variable is replaced by a force value. A variable can receive its current value from the following sources:

- From a physical input
- From the communication
- From a logic operation

21233799/EN – 07/2014

When a variable is forced, the user specifies the value. Forcing is used for the following purposes:

- Testing the user program, particularly for rare cases that cannot be tested otherwise.
- Simulation of unavailable sensors in cases where the initial value is not appropriate.

⚠ WARNING

Personal injury may result from forced values.

Severe or fatal injuries.

- Force values only after consulting the inspection authority responsible for approving the system.
- Remove limitations to forcing only after consultation with the inspection authority responsible for approving the system.

During forcing, the person in charge must ensure sufficiently safe monitoring of the process through other technical and organizational measures. SEW-EURODRIVE recommends forcing only for a limited time.

⚠ WARNING

Using forced values can disrupt safety-related operation.

Severe or fatal injuries.

- Forced values can lead to incorrect output values.
- Forcing extends the cycle time. The watchdog time might be exceeded as a result.

Basic information about forcing is provided in the "Maintenance Override" document from the German Technical Control Board. The document is available on the following German Technical Control Board sites:

http://www.tuv-fs.com or http://www.tuvasi.com.

11.5 Forcing

Forcing can occur on two levels:

- Global forcing

  Global variables are forced for all applications.
- Local forcing

  The values of local variables are forced for an individual user program.

The following conditions must be met for a global or local variable to be forced:

- The corresponding force switch is set.
- Forcing has been started.

If forcing has been started, a change to the force switch takes effect immediately. If forcing has been started and the force switch has been set, a change to the force value takes effect immediately. Local forcing can be started and stopped separately for each user program.

21233799/EN – 07/2014

11.5.1 Time limit

Different time limits can be set for global and local forcing. Once the defined time has expired, the controller stops forcing. The behavior of the safety controller after the time limit has elapsed can be configured.

- For global forcing, the following settings can be selected:
  - The resource stops.
  - The resource continues.
- For local forcing, the following settings can be selected:
  - The user program stops.
  - The user program continues.

It is also possible to use forcing without a time limit. In this case, forcing must be stopped manually. When a variable is no longer forced, the process value is used again for the process variable.

11.5.2 Force editor

The Force editor of SILworX® displays all the variables for which forcing is possible. Global and local variables are displayed separately in different tabs. Force values and force switches can be set in the tabs.

11.5.3 Restrictions on forcing

To avoid possible disruptions to safety-related operation by improper forcing, the following measures can be taken in the configuration to restrict the use of forcing:

- Configuring different user accounts with and without forcing rights
- Prohibiting global forcing for a resource
- Prohibiting local forcing or entering of process values
- In addition, forcing can be stopped immediately using a keyswitch. For such a switch to take effect, the *Deactivate forcing* system variable must be connected to a digital input to which a keyswitch is connected.

  The *Deactivate forcing* system variable prevents the start of forcing for global and local variables and immediately turns off forces that have already been started.

21233799/EN – 07/2014

12 Startup

Starting safety controller operations includes the following stages:

- Mechanical installation. Observe the chapter "Mechanical installation" in the "MOVISAFE® HM31 Decentralized Safety Controller" operating instructions.
- Electrical installation. Observe the chapter "Electrical installation" in the "MOVISAFE® HM31 Decentralized Safety Controller" operating instructions.
- Configuration
  - Creating the user program
  - Setting safety parameters, communication parameters, and other parameters

12.1 Checklist for project planning, programming and startup

This checklist is a recommendation for the user

- for the project planning, programming and startup of safety-related inputs and outputs,
- for creating a user program with the SILworX® programming tool.

Filling in the checklist can ensure that requirements have been fully accounted for in an orderly format. The checklist also serves to document the connection between external wiring and user program.

The *PFF_HM31A_Checkliste_DE.pdf* checklist can be downloaded as a PDF document from the SEW-EURODRIVE homepage (www.sew-eurodrive.com). You find the checklist in the “safetyDRIVE” area of the “Documentation” category.

12.2 Configuration with SILworX®

The hardware editor of the SILworX® programming tool displays MOVISAFE® HM31 similarly to a base carrier, equipped with the following modules:

- Processor module (CPU)
- Communication module (COM)
- Digital input module (DI 26)
- Digital output module (DO 8)
- Counter module (HSC 2)

Double-clicking the modules opens the detailed view with tabs. In the tabs, the global variables configured in the user program can be assigned to the system variables of the relevant module.

12.2.1 Processor module

Double-clicking the modules opens the detailed view with tabs. In the tabs, the global variables configured in the user program can be assigned to the system variables of the relevant module.

21233799/EN – 07/2014

Module tab

The module tab includes the following parameters:

Parameter	Description
Name	Module name.
Use max. µP budget for HH protocol	• Activated: Copy CPU load limit from the *Max. µP budget for HH protocol [%]* field. • Deactivated: Do not use a CPU load limit for safeethernet. Default setting: Deactivated
Max. µP budget for HH protocol [%]	Maximum CPU load for the module that may be generated when processing the safeethernet protocol. **Note:** The maximum load must be distributed among all the protocols used that utilize this communication module.
IP address	IP address of the Ethernet interface. Default: 192.168.0.99
Subnet mask	32-bit address mask for subdividing an IP address into network and host addresses. Default: 255.255.252.0
Default interface	Activated: Interface is used as the default interface for a system login. Default setting: Deactivated
Default gateway	IP address of the default gateway. Default: 0.0.0.0
ARP aging time [s]	A CPU or COM module saves the MAC addresses of its communication partners in a MAC/IP address allocation table (ARP cache). If during a period of 1× to 2× *ARP aging time* • messages arrive from the communication partner, the MAC address remains in the ARP cache. • If no messages arrive from the communication partner, the MAC address is deleted from the ARP cache. A typical value for the *ARP aging Time* in a local network is 5 to 300 s. The content of the ARP cache cannot be read by the user. When routers or gateways are used, the *ARP aging time* should be adjusted (increased) to the additional delays in the back and forth transmissions. If the *ARP Aging Time* is too low, the CPU/COM module deletes the MAC address of the communication partner from the ARP cache, and communication is either delayed or stopped. For efficient use, the *ARP aging time* must be greater than the *ReceiveTimeouts* of the protocols used. Range of values: 1 s – 3600 s Default: 60 s

21233799/EN – 07/2014

Parameter	Description
MAC learning	ARP cache learning behavior: • Conservative: MAC addresses of stored ARP entries are not overwritten by received messages. • Tolerant: MAC addresses of stored ARP entries are overwritten by received messages. Default setting: conservative
IP forwarding	Allows a processor module to work as a router and forward data packets to other network nodes. Default setting: Deactivated
ICMP mode	Internet Control Message Protocol (ICMP) message types supported by the processor module: • No ICMP responses • Echo response • Host unavailable • All implemented ICMP responses Default setting: Echo response
Max. com. time slice ASYNC [ms]	Maximum value in ms of the time slice that is used for communication during the cycle of the resource. Setting range: 2 – 5000 ms
Max. duration of configuration connections [ms]	Defines how much time is available during a CPU cycle for process data communication. Setting range: 6 – 5000 ms
Target cycle time [ms]	Desired or maximum cycle time, see *Target cycle time mode*. The *Target cycle time* must not exceed the configured watchdog time (6 ms); otherwise it is rejected by the PES. Setting range: 0 – 7500 ms
Target cycle time mode	Using the *Target cycle time [ms]*. **Fixed:** The PES adheres to the *Target cycle time* and, if necessary, extends the cycle. This does not apply if the user programs' processing time exceeds the *Target cycle time*. **Fixed tolerant:** Similar to fixed, but the *target cycle time* is ignored during the first activation cycle of the reload function (function is subject to license). **Dynamic tolerant:** The PES adheres to the *target cycle time* if possible but executes the cycle in the shortest possible time. In the 1st activation cycle of the reload function (function is subject to license), the *target cycle time* is ignored.
Maximum system bus latency [µs]	Must not be used for MOVISAFE® HM31 safety controllers.

21233799/EN – 07/2014

Parameter	Description
safeethernet CRC	**Current version:** The CRC for safeethernet is created with the current algorithm.

Routings tab

The routings tab contains the following parameters:

Parameter	Description
Name	Name of the routing setting.
IP address	Target IP address of the communication partner (for direct host routing) or network address (for subnet routing). Range of values: 0.0.0.0 – 255.255.255.255 Default: 0.0.0.0
Subnet mask	Specified target address range for a routing entry. 255.255.255.255 (for direct host routing) or subnet mask of the subnet being addressed. Range of values: 0.0.0.0 – 255.255.255.255 Default: 255.255.255.255
Gateway	IP address of the gateway to the addressed network. Range of values: 0.0.0.0 – 255.255.255.255 Default: 0.0.0.1

Ethernet switch tab

The Ethernet switch tab contains the following parameters:

Parameter	Description
Name	Name of the port (Eth1 through Eth4) as printed on the housing; only one configuration may exist for each port.
Speed [Mbit/s]	10 Mbit/s: Data rate 10 Mbit/s 100 Mbit/s: Data rate 100 Mbit/s 1000 Mbit/s: Data rate 1000 Mbit/s (not supported) Autoneg: Baud rate is set automatically Default: Autoneg
Flow control	Full duplex: Simultaneous communication in both directions Half duplex: Communication in one direction Autoneg: Automatic communication control Default: Autoneg
Autoneg also with fixed values	The *Advertising* function (forwarding the *Speed* and *Flow control* properties) is also performed if *Speed* and *Flow control* have fixed values. This allows other devices with ports set to *Autoneg* to detect the setting of the safety controller's ports.

21233799/EN – 07/2014

Parameter	Description
Limit	Limit incoming multicast and/or broadcast packets. Off: No limitation Broadcast: Limit broadcast (128 kbit/s) Multicast and broadcast: Limit multicast and broadcast (1024 kbit/s) Default: Broadcast

VLAN (port-based LAN) tab

Configures the use of port-based VLAN.

INFORMATION

In order for VLAN to be supported, port-based VLAN must be disabled so that each port can communicate with any other port on the switch.

For each port on a switch, you can configure which other port on the switch received Ethernet frames can be sent to. The table in the VLAN tab contains entries that allow the connection between two ports to be set to active or inactive.

	Port				
Port **(Ethernet interface on MOVISAFE® HM31)**	**Eth 1** **(X4233_1)**	**Eth 2** **(X4233_2)**	**Eth 3** **(X4223)**	**Eth 4**	**COM**
Eth 1 (X4233_1)					
Eth 2 (X4233_2)	Active				
Eth 3 (X4223)	Active	Active			
Eth 4	Active	Active	Active		
COM	Active	Active	Active	Active	
CPU	Active	Active	Active	Active	Active

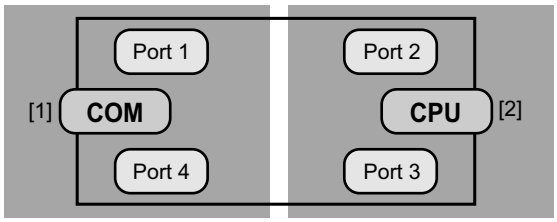
INFORMATION

Port Eth 4 has no function.

21233799/EN – 07/2014

Separating switch ports via VLAN

Available Ethernet ports can be separated according to the required application. This is how a connection can be established with two IP addresses, or safe communication can be separated via CPU from non-safe communication via COM.



12209301643

[1] Ports 1 and 4 are assigned to the COM

[2] Ports 2 and 3 are assigned to the CPU

The switch ports are configured in SILworX® in the detailed view of the CPU or COM module. Settings in the VLAN tab according to the above figure.

LLDP tab (Link Layer Discovery Protocol)

LLDP (Link Layer Discovery Protocol) periodically sends information about its own device (such as MAC address, device name, port number) via multicast and receives the same information from neighboring devices.

The processor and communication modules support LLDP on ports Eth1, Eth2 and Eth3. Settings for Port Eth4 have no function.

The parameters below specify how the relevant port works.

- 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.

Mirroring tab

Configures whether the module duplicates Ethernet packets on a port so that these packets can also be read by a device connected there, for example for testing.

The parameters below specify how the relevant port works.

- Off

  This port does not participate in mirroring.

- Egress

  Data leaving this port is duplicated.

21233799/EN – 07/2014

- Ingress/egress

  Incoming and outgoing data of this port is duplicated.
- Dest. port

  Duplicated data is sent to this port.

12.2.2 Communications module

The communication module (COM) contains the “Module” and “Routings” tabs with the same parameters as the processor module. The default IP address is 192.168.0.100.

12.2.3 Resource configuration

The properties of the resource and the hardware output variables must be configured.

Resource properties

These parameters determine the behavior of the controller during operation and are set in SILworX® in the "Properties" dialog of the resource.

Parameter/ switch	Description	Default	Setting for safe operation
Name	Name of the resource.		Any
System ID [SRS]	System ID of the resource. The system ID value must be different from the default, else the project cannot be run. Setting range: 1 – 65535	60000	Unique value within the network of controllers including all controllers that can potentially interconnected.
Safety time [ms]	Safety time in milliseconds. Setting range: 20 – 22500 ms	600 ms	Application-specific
Watchdog time [ms]	Watchdog time in milliseconds. Setting range: 8 – 5000 ms	200 ms	Application-specific

21233799/EN – 07/2014

Parameter/ switch	Description	Default	Setting for safe operation
Main enable	The main enable can only be set to "ON" if the PES is stopped. **ON:** The following switches/parameters can be changed during operation (= RUN) using the PADT: • *System ID* • *Watchdog time of the resource* • *Safety time* • *Target cycle time* • *Target cycle time mode* • *Autostart* • *Global forcing allowed* • *Global force timeout response* • *Reload function allowed (function available by activating a license)* • *Start allowed* **OFF:** The parameters cannot be changed during operation.	ON	OFF recommended
Autostart	**ON:** The user program starts automatically when connecting the processor system to the supply voltage. **OFF:** The user program does not start automatically when connecting the system to the supply voltage.	OFF	Application-specific
Start allowed	**ON:** Cold start or warm start using the PADT is permitted in RUN or STOP condition. **OFF:** Start not allowed.	ON	Application-specific
Loading allowed	**ON:** Downloading the user program is permitted. **OFF:** Downloading the user program is not permitted.	ON	Application-specific

Parameter/ switch	Description	Default	Setting for safe operation
Reload	**ON:** Reload function (available by activating a license) of the user program is allowed. **OFF:** Reload function (available by activating a license) of the user program is not allowed. A reload (function available by activating a license) that is already running is not terminated when switching to OFF.	ON	OFF recommended
Global forcing allowed	**ON:** Global forcing permitted for this resource. **OFF:** Global forcing not permitted for this resource.	ON	Application-specific
Global force timeout response	Specifies how the resource behaves when the timeout of global forcing has expired: • Stop forcing • Stopping a resource	Stop forcing	Application-specific
Max. com. time slice ASYNC [ms]	Maximum value in ms of the time slice that is used for communication during the cycle of the resource. Setting range: 2 – 5000 ms	60 ms	Application-specific
Max. duration of configuration connections [ms]	Defines how much time is available during a CPU cycle for process data communication. Setting range: 6 – 5000 ms	6 ms	—
Target cycle time [ms]	Desired or maximum cycle time, see *Target cycle time mode*. The *Target cycle time* must not exceed the configured watchdog time (6 ms); otherwise it is rejected by the PES. Setting range: 0 – 7500 ms	0 ms	—

21233799/EN – 07/2014

Parameter/ switch	Description	Default	Setting for safe operation
Target cycle time mode	Using the *Target cycle time [ms]*. **Fixed:** The PES adheres to the *Target cycle time* and, if necessary, extends the cycle. This does not apply if the user programs' processing time exceeds the *Target cycle time*. **Fixed tolerant:** Similar to fixed, but the *target cycle time* is ignored during the first activation cycle of the reload function (subject to license). **Dynamic tolerant:** Similar to fixed, but the *target cycle time* is ignored during the first activation cycle of the reload function (subject to license). **Dynamic:** The safety controller adheres to the *Target cycle time* if possible, but executes the cycle in the shortest possible time.	Fixed	—
Maximum configuration version	—	SILworX® V4	—
Maximum system bus latency [µs]	Cannot be used for the safety controller.	0 ms	—
safeethernet CRC	In the current version, the CRC for safeethernet is created with the current algorithm.	Current version	Application-specific

Hardware system variables for creating parameters

These variables are used to change the behavior of the controller during operation if specific states occur. These variables are located in the hardware editor in SILworX® in the detailed view of the hardware.

Variable	Function	Default setting	Setting for safe operation
Force deactivation	Is used to prevent forcing and to stop it immediately.	FALSE	Application-specific
Spare 2 – spare 16	No function.	—	—
Emergency off 1 – emergency off 4	EMERGENCY OFF switch for shutting down the controller in the event of malfunctions detected by the user program.	FALSE	Application-specific

21233799/EN – 07/2014

Variable	Function	Default setting	Setting for safe operation
Read-only in RUN	After starting the controller, no operating action such as stop, start, or download is permitted in SILworX®. Exceptions: Forcing and reload function (available by activating a license).	FALSE	Application-specific
Relay contact 1 – 4	No function.	—	—
Reload deactivation	Prevents the controller from being loaded by using the reload function (available by activating a license).	FALSE	Application-specific
User LED 1 – 2	Controls the corresponding LED, if present.	FALSE	Application-specific

Global variables can be assigned to these system variables; the value of the global variables can be changed using a physical input or the user program logic.

Hardware system variables for reading parameters

These system variables can be accessed in the hardware editor of SILworX®. To open the detailed view, select the gray background outside the (yellow) subrack representation and double-click or open the context menu.

Variable	Description	Data type
Number of I/O errors	Number of current I/O errors.	UDINT
Number of I/O errors, historic count	Total number of I/O errors (counter can be reset).	UDINT
Number of I/O warnings	Number of current I/O warnings.	UDINT
Number of I/O warnings, historic count	Total number of I/O warnings (counter can be reset).	UDINT
Number of communication errors	Number of current communication errors.	UDINT
Communication error historic count	Total number of communication errors (counter can be reset).	UDINT
Number of communication warnings	Number of current communication warnings.	UDINT
Communication warning historic count	Total number of communication warnings (counter can be reset).	UDINT
System error count	Number of current system errors.	UDINT
System error historic count	Total number of system errors (counter can be reset).	UDINT
System warning count	Number of current system warnings.	UDINT
System warning historic count	Total number of system warnings (counter can be reset).	UDINT

21233799/EN – 07/2014

Variable	Description	Data type
Autostart CPU release	**ON:** When the supply voltage is applied, the processor system automatically starts the user program. **OFF:** When the supply voltage is applied, the processor system is switched to STOP state.	BOOLEAN
OS major	Output of the operating system into the processor system.	UINT
OS minor		UINT
CRC	Checksum of the project configuration.	UDINT
Date/time [ms portion]	System date and time in s and ms since 01/01/1970.	UDINT
Date/time [sec. portion]		UDINT
Force deactivation	**ON:** Forcing is deactivated. **OFF:** Forcing is possible.	BOOLEAN
Forcing is active	**ON:** Global or local forcing is active. **OFF:** Global and local forcing are not active.	BOOLEAN
Force switch status	Status of the force switch. 0xFFFFFFFE: No force switch set 0xFFFFFFFF: At least one force switch set	UDINT
Global forcing started	**ON:** Global forcing is active. **OFF:** Global forcing is not active.	BOOLEAN
Spare 0 – 16	Reserved.	USINT
Spare on17		BOOLEAN
Last I/O warning [ms]	Date and time of the last I/O warning in s and ms since 01/01/1970.	UDINT
Last I/O warning [s]		UDINT
Last communication warning [ms]	Date and time of the last communication warning in s and ms since 01/01/1970.	UDINT
Last communication warning [s]		UDINT
Last system warning [ms]	Date and time of the last system warning in s and ms since 01/01/1970.	UDINT
Last system warning [s]		UDINT
Last I/O error [ms]	Date and time of the last I/O error in s and ms since 01/01/1970.	UDINT
Last I/O error [s]		UDINT

21233799/EN – 07/2014

Variable	Description	Data type
Last communication error [ms]	Date and time of the last communication error in s and ms since 01/01/1970.	UDINT
Last communication error [s]		UDINT
Last system error [ms]	Date and time of the last system error in s and ms since 01/01/1970.	UDINT
Last system error [s]		UDINT
Fan status	0xFF: Not available	BYTE
Major CPU release	Main enable switch for the processor system: **ON:** The lower-level enable switches can be changed. **OFF:** The lower-level enable switches cannot be changed.	BOOLEAN
Read-only in RUN	**ON:** The stop, start and download operator actions are locked. **OFF:** The stop, start and download operator actions are not blocked.	BOOLEAN
Reload release	**ON:** The controller can be reloaded using the reload function (can be activated using a license). **OFF:** The controller cannot be reloaded using the reload function (can be activated using a license).	BOOLEAN
Reload deactivation	**ON:** Loading the controller by performing a reload (can be activated using a license) is locked. **OFF:** Loading the controller by performing a reload (can be activated using a license) is possible.	BOOLEAN
Reload cycle	TRUE during the first cycle after a reload (can be activated using a license), else FALSE.	BOOLEAN
CPU safety time [ms]	Safety time set for the controller in ms.	UDINT

21233799/EN – 07/2014

Variable	Description		Data type
Start CPU release	**ON:** Starting the processor system using PADT is allowed. **OFF:** Starting the processor system using PADT is not allowed.		BOOLEAN
Start cycle	ON during the first cycle after start, otherwise OFF.		BOOLEAN
Power supply state	Bit-coded state of the power supply.		BYTE
	Value	State	
	0x00	Normal	
	0x01	Undervoltage at 24 V supply voltage.	
	0x02	(Undervoltage with battery) not used.	
	0x04	Undervoltage with internally generated 5 V voltage.	
	0x08	Undervoltage with internally generated 3.3 V voltage.	
	0x10	Overvoltage with internally generated 3.3 V voltage.	
System ID	System ID of the controller. Setting range: 1 – 65535		UINT
System tick HIGH	Revolving millisecond counter (64-bit).		UDINT
System tick LOW			UDINT
Temperature state	Bit-coded temperature state of the processor system.		BYTE
	Value	State	
	0x00	Normal temperature	
	0x01	Temperature threshold 1 exceeded	
	0x03	Temperature threshold 2 exceeded	
	0xFF	Not available	
Remaining global force duration [ms]	Time in ms until the global force time limit elapses.		DINT
CPU watchdog time [ms]	Longest allowable duration of a RUN cycle in ms.		UDINT
Cycle time, last [ms]	Current cycle time in ms.		UDINT
Cycle time, max. [ms]	Maximum cycle time in ms.		UDINT
Cycle time, min. [ms]	Minimum cycle time in ms.		UDINT

Variable	Description	Data type
Cycle time, average [ms]	Average cycle time in ms.	UDINT

User program configuration

The following user program switches and parameters can be set in the "Properties" dialog of the user program.

Parameter/switch	Description	Default	Setting for safe operation
Name	Name of the user program.		Any
Safety integrity level	Safety level: SIL 0, SIL 3	SIL 3	Application-specific
Start allowed	**ON:** Starting the user program via PADT allowed. **OFF:** Starting the user program via PADT not allowed.	ON	Application-specific
Program main enable	Enable of the change on other user program switches: Only the enable switch of the resource is relevant.	ON	Application-specific
Autostart	Enabled autostart type: Cold start, warm start, off.	Warm start	Application-specific
Test operation allowed	**ON:** Test operation is allowed for the user program. **OFF:** Test operation is not allowed for the user program.	OFF	Application-specific
Local forcing allowed	**ON:** Forcing at program level allowed. **OFF:** Forcing at program level not allowed.	OFF	OFF recommended
Local force timeout response	Behavior of the user program after the force time has elapsed: • Stop only forcing • Stop program	Stop only forcing	Application-specific
Reload allowed	**ON:** Reload function (available by activating a license) of the user program is allowed. **OFF:** Reload function (available by activating a license) of the user program is not allowed.	ON	Application-specific
Maximum CPU cycles for program	Maximum number of CPU cycles that a cycle in the user program can last. A value > 1 is allowed.	1	Application-specific

21233799/EN – 07/2014

Parameter/switch	Description	Default	Setting for safe operation
Max. duration per cycle [µs]	Maximum execution time per processor module cycle for a user program. Setting range: 1 – 7 500000 µs 0: No limit	0 µs	0 µs
Program ID	ID for identifying the program when displayed in SILworX®. Setting range: 1 – 32	1	Application-specific
Watchdog time [ms] (calculated)	Non-changeable monitoring time of the user program, calculated from *Maximum CPU cycles program* and *Resource watchdog time*. **Note:** **When using counter inputs, it is important that the watchdog time of the user program is ≤ 5 000 ms.**		—

12.2.4 Configuring inputs and outputs

The inputs and outputs are configured in the hardware editor by assigning global variables to the system variables for the inputs or outputs.

To find the channels' system variables, do the following:

1. View the desired resource in the hardware editor.
2. Open the detail view by double-clicking on the desired input or output module.
3. In the detailed view, open the tab with the required channels. The system variables of the channels are displayed.

Using digital inputs

The following steps are necessary to use the value of a digital input in the user program:

1. Define a global variable of the type BOOLEAN.
2. Enter an appropriate initial value when defining the global variable.
3. Assign the global variable to the channel value of the input.
4. In the user program, program a safety-related fault response using the error code -> *Error code [byte]*.

The global variable provides values to the user program.

By assigning global variables to *DI.ErrorCode* and *ModuleErrorCode*, you can program additional fault responses in the user program. For details about the error codes, refer to chapter "Parameters and error codes for inputs and outputs".

Using safety-related counter inputs

You can use the counter reading or the speed/frequency as an integer value or as a scaled floating-point value. In the following sections, "xx" refers to the respective channel number.

Do the following to use an integer value:

1. Define a global variable of the type UDINT.

21233799/EN – 07/2014

2. Enter an appropriate initial value when defining the global variable.
3. Assign the global variable to the integer value *Counter[xx].Value* of the input.
4. In the user program, program a safety-related fault response using the error code *Counter[xx].Error code [byte]*.

The global variable provides values to the user program.

By assigning global variables to *Counter.ErrorCode* and *ModuleErrorCode*, you can program additional fault responses in the user program. For details about the error codes, refer to chapter "Parameters and error codes for inputs and outputs".

Using digital outputs

The following steps are required to write a user program value to a digital output:

1. Define a global variable of the type BOOLEAN that contains the value to be output.
2. Enter an appropriate initial value when defining the global variable.
3. Assign the global variable to the channel value *[BOOL] ->* of the output.
4. In the user program, program a safety-related fault response using the error code *-> Error code [byte]*.

The global variable provides values to the digital output.

By assigning global variables to *DO.ErrorCode* and *ModuleErrorCode*, you can program additional fault responses in the user program.

12.2.5 Generating the resource configuration

Proceed as follows:

1. Select the resource in the structure tree.
2. Click the [Code generation] button in the action bar or select [Code generation] from the context menu. The "Code generation" dialog opens.
3. In the "Start code generation" dialog, click [OK]. Another "Start code generation" dialog opens, shows the code generation progress, and closes again. A line showing the result of the code generation appears in the log.
4. With the resource still selected, select [Version comparison] from the [Extras] menu. The "Version overview" dialog opens. It contains the CRC of the generated code.
5. Click [Export]. An "Archive" dialog opens where you can enter a comment about the project status and the name of the archive file.
6. Generate code again as described in steps 2 and 3.
7. With the resource still selected, select [Version comparison] from the [Extras] menu. The "Version overview" dialog opens.
8. Click [Import]. In the "Restore" dialog, import the archive file exported in step 5. The "Version overview" window now contains information about the last generated project status and the imported project status.
9. Click [OK]. The result of the version comparison is displayed in the work area. If "ok" is displayed in the "Comparison of CRCs" column, the code generated for both project states is the same and may be used for safety-related operation. Differences are highlighted in red.

The code for resource configuration is now generated.

21233799/EN – 07/2014

NOTICE

Errors may occur during code generation if a non-safe PC is used.

For safety-related applications, the code generator must generate code twice and the checksums (CRCs) of both generating processes must match. Only then is error-free code ensured.

12.2.6 Configuring system ID and connection parameters

Proceed as follows:

1. Select the resource in the structure tree
2. Click the [Online] button in the action bar or select "Online" from the context menu.

   The system login dialog window opens.
3. Click [Search].

   The "Search by MAC" dialog opens.
4. Enter the valid MAC address for the controller (see label on the housing) and click [Search].

   The dialog shows the values for the IP address, subnet mask and SRS set in the controller.
5. To import the values, click the [Import] button.
6. Log in with the "Administrator" user account.
7. Select the [Online]/[Startup]/[Set system ID] menu and assign the desired system ID.

   The change is effective immediately so that the connection is terminated.
8. If this has not taken place: Assign the IP address of COM and CPU via hardware and compile the project.
9. To enter additional IP addresses/system IDs, repeat steps 1 through 6.
10. Load the program into the controller.

    The change is effective immediately so that the connection is terminated.

Now you can log in with the IP addresses and system IDs configured in the project.

For a system network of several safety controllers, we recommend configuring the individual controllers consecutively before incorporating them into the network. In the device configuration of the safety controller, you have to enter a default value for the IP address. This means the controller can only be assigned using the MAC address of the individual controller.

⚠ WARNING

Risk of interchanging the addressed controller. In this case, a wrong user program might be loaded into the safety controller.

Severe or fatal injuries.

12.2.7 Loading the resource configuration from the programming device

Before a user program can be loaded into the controller together with the connection parameters of the controller (IP address, subnet mask, and system ID), the code for the resource must have been generated, and the programming device and the resource must have valid connection parameters (see chapter "Configuring system ID and connection parameters").

Do the following to load the resource configuration from the programming device:

1. Select the resource in the structure tree.
2. In the action bar, click [Online], or choose [Online] from the context menu.
3. In the "System Login" window, enter a user group with administrator rights or write access. The control panel opens in the workspace and displays the controller state.
4. In the [Online] menu, choose [Resource Download]. The "Resource Download" dialog opens.
5. Click "OK" to confirm the download. SILworX® loads the configuration into the controller.
6. Once the configuration has been loaded, start the user program by clicking [Resource Cold Start] in the [Online] menu. After the cold start, "System State" and "Program State" enter RUN mode.

The resource configuration is loaded from the programming device. The "Start", "Stop", and "Load" functions are also available as icons in the symbol bar.

12.2.8 Loading the resource configuration from the communication system's flash memory

In the event of data errors in the NVRAM resulting in the watchdog time to be exceeded, it can be useful to load the resource configuration from the flash memory for the communication system instead of loading it from the programming device.

If the control panel (CP) is no longer accessible, you have to reset the connection parameters for the project in the controller, see chapter "Configuring system ID and connection parameters".

When the controller adopts the STOP/VALID CONFIGURATION state after restart, you can start the user program again.

When the controller adopts the STOP/INVALID CONFIGURATION state after restart, you have to reload the user program into the NVRAM.

Use the [Load Configuration from Flash] command to read a backup copy of the last executable configuration from the flash memory of the communication system and to transfer it to the NVRAM of the processor. You can now start the user program again by choosing [Online] / [Resource Cold Start] without having to download the project.

Do the following to load the resource configuration from the flash memory of the communication system:

1. Log in to the required resource.
2. From the [Online] menu, choose [Maintenance/Service] and then [Load Configuration from Flash].
3. Confirm the action in the dialog that opens.

The controller loads the resource configuration from the flash memory of the communication system into the NVRAM.

21233799/EN – 07/2014

12.2.9 Cleaning up the resource configuration in the communication system's flash memory

The flash memory of the communication system might contain remainders of invalid configurations after temporary hardware faults.

To delete these invalid configurations, use the [Clean Up Configuration].

Do the following to clean up the resource configuration:

1. Select the resource in the structure tree.
2. In the action bar, click [Online], or choose [Online] from the context menu.
3. In the "System Login" window, enter a user group with administrator rights or write access. The control panel opens in the workspace and displays the controller state.
4. In the [Online] menu, choose [Maintenance/Service] and then [Clean Up Configuration].
5. Click "OK" in the [Clean Up Configuration] window to confirm the action.

The configuration in the flash memory of the communication system has been cleaned up. Cleaning up the configuration is needed only in rare cases. A valid configuration is not affected by the clean-up process.

12.2.10 Setting date and time

Do the following to set date and time:

1. Select the resource in the structure tree.
2. In the action bar, click [Online], or choose [Online] from the context menu.
3. In the "System Login" window, enter a user group with administrator rights or write access. The control panel opens in the workspace and displays the controller state.
4. In the [Online] menu, choose [Startup] and then [Set Date/Time]. The "Set Date/Time" dialog opens.
5. Choose one of the following options:
   - Use the date and time of the programming device.

     The time and date displayed for the programming device is transferred to the controller.
   - User-defined.

     The time and date of the two edit boxes are transferred to the controller. Make sure that the format used for the date and time is correct.
6. Click [OK] to confirm the action. The date and time are now set for the controller.

12.3 User management with SILworX®

SILworX® can set up and maintain an own user management scheme for each project and each controller.

12.3.1 User management for SILworX® projects

A PADT user management scheme can be added to each SILworX® project. This user management scheme controls the access to the project in SILworX®.

21233799/EN – 07/2014

Without PADT user management, every user can open a project and modify any part of it. If a user management scheme has been defined for a project, only authorized users can open the project. This means that only users with the corresponding rights can modify a project. The following authorization levels are available:

Level	Meaning
Security administrator (Sec Adm)	Can modify the user management: Set up, delete, change user accounts and user groups as well as the PADT user management scheme, set up the default user account. Can also perform any other functions in SILworX®.
Read/write (R/W)	All SILworX® functions, except for the user management.
Read only (RO)	Read-only access, which means users may not change or archive projects.

The user management allocates the rights to the user groups. The user groups allocate the rights to the user accounts assigned to them.

User group properties:

- The name must be unique in the project and must have a length of 1 to 31 characters.
- A user group is assigned an authorization level.
- A user group can be assigned any number of user accounts.
- A project can include up to 100 user groups.

User account properties:

- The name must be unique in the project and must have a length of 1 to 31 characters.
- A user account is assigned a user group.
- A project can include up to 1000 user accounts.
- A user account can be the default user of the project.

12.3.2 User administration for the controller

The user management for a controller (PES user management) is used to protect a safety controller from unauthorized access. Users and their access rights are part of the project. They are defined with SILworX® and are loaded into the processor module.

The user management can be used to set and manage the access rights to a controller for up to 10 users. The access rights are stored in the controller and are not lost when switching off the operating voltage.

Each user account consists of name, password, and access right. Once the project has been downloaded to the controller, this information is available for login. Users log in to the controller using their name and password.

User accounts need not be created but contribute to safe operation. If a user management scheme has been defined for a resource, then this scheme must contain at least one user with administrator rights.

Default user

If no user-specific user accounts were created for a resource, then the factory settings apply.

21233799/EN – 07/2014

Factory settings:

- Number of users: 1
- User ID: Administrator
- Password: None
- Access right: Administrator

INFORMATION

Note that default settings cannot be maintained when defining own user accounts.

User account parameters

Define the following parameters when setting up new user accounts:

Parameter	Description
User name	Name and ID of the user to log in to a controller. The user name must not contain more than 32 characters (recommended: a maximum of 16 characters) and may only consist of letters (A to Z, a to z), numbers (0 to 9), and the special characters underscore "_" and hyphen "-". The password is case sensitive.
Password	Password assigned to a user name required for login. The password must not contain more than 32 characters and may only consist of letters (A to Z, a to z), numbers (0 to 9), and the special characters underscore "_" and hyphen "-". The password is case sensitive.
Confirm password	Repeat the password to confirm the entry.

21233799/EN – 07/2014

Parameter	Description
Access type	Access types define the privileges a user might have. The following access types are available: • Read: Users may only read information from the controller but may not make changes. • Read and operator: Like *Read* but users may also: – Perform a download to load and start user programs. – Configure processor modules as redundant. – Reset cycle time and fault statistics. – Set the system time. – Use the forcing function. – Restart and reset modules. – Start system operation for processor modules. • Read and write: Like *read and operator* but users may also create programs, compile programs, load programs into the controller, and test them. • Administrator: Like *read and write* but users may also: – Load operating systems. – Modify the main enable switch setting. – Change the SRS. – Change the IP settings. At least one user must have administrator rights, else the controller does not accept the settings. The administrator can revoke a user's permission to access a controller by deleting the user name from the list.

Setting up user accounts

A user with administrator rights can access all user accounts. Observe the following information when creating user accounts:

- Make sure that at least one user account is created with administrator rights. Define a password for the user account with administrator rights.
- When the administrator has created a user account in the user management and wants to edit it, then he/she must enter the password of the user account.
- To check the created user accounts, use the "Verification" function in SILworX®.
- The new user accounts take effect as soon as the code has been generated and a download has been performed to load the project into the controller. This means that all previously saved user accounts, such as default settings, are no longer valid.

12.4 Configuring the communication with SILworX®

This chapter describes how to configure the communication using the SILworX® programming tool.

21233799/EN – 07/2014

Depending on the application, configure the following:

- Ethernet/safeethernet
- Standard protocols

  For more information on how to configure the standard protocols, refer to the chapter "Modbus TCP/UDP".

12.4.1 Configuring the Ethernet interfaces

The configuration is made in the detailed view of the communication module (COM).

INFORMATION

SILworX® represents the processor system and the communication system within a device or module as the processor module and communication module.

For the safety controller, set the parameters *Speed [Mbit/s]* and *Flow Control* in the Ethernet switch settings to "Autoneg". For a detailed explanation of the parameters *ARP Aging Time*, *MAC Learning*, *IP Forwarding*, *Speed [Mbit/s]* and *Flow Control*, refer to the online help of SILworX®.

INFORMATION

Replacing a controller with the same IP address:

When replacing a controller for which the *ARP Aging Time = 5 minutes* and *MACLearning = conservative*, the communication partner does not adopt the new MAC address until a period of 5 to 10 minutes after the controller is replaced. During this period, no communication is possible with the replaced controller.

You can configure the port settings of the integrated Ethernet switch of the safety controller individually. On the "Ethernet Switch" tab, a table entry can be created for each switch port.

Port configuration parameters	Explanation
Port	Port number as printed on the housing. Only one configuration may exist for each port. Range of values: 1 – n, depending on the resource.
Speed [Mbit/s]	10 Mbit/s: Data rate 10 Mbit/s 100 Mbit/s: Data rate 100 Mbit/s Autoneg (10/100): Baud rate is set automatically Default: Autoneg
Flow control	Full duplex: Simultaneous communication in both directions Half duplex: Communication in one direction at a time Autoneg: Automatic communication control Default: Autoneg
Autoneg also with fixed values	Forwarding the *Speed* and *Flow control* properties (*Advertising*) is also performed when *Speed* and *Flow control* have fixed values. This allows other devices with ports set to *Autoneg* to detect the setting of the safety controller's ports.

Port configuration parameters	Explanation
Limit	Limit incoming multicast and/or broadcast packets. Off: No limitation Broadcast: Limit broadcast (128 kbit/s) Multicast and broadcast: Limit multicast and broadcast (1024 kbit/s) Default: Broadcast

You can change the parameters and enter them in the configuration of the communication system by double-clicking each table cell. The parameters take effect for the communication of the safety controller once they have been compiled again with the user program and have been transferred to the controller.

You can also change the properties of the communication system and of the Ethernet switch online using the control panel. These settings take effect immediately but are not adopted by the user program.

For more information about configuring the safeethernet communication, refer to the chapter "safeethernet".

12.5 Configuring alarms and events

Defining events:

1. Define a global variable for each event. Generally use global variables that have already been defined for the program.
2. Below the resource, create a new "Alarm and Events" branch if it does not already exist.
3. Define events in the alarm and event editor.
   - Drag global variables into the event window for Boolean or scalar events.
   - Define the details of the events, see the following two tables.

The events are now defined. For more information, refer to the SILworX® online help.

Enter the **parameters of the Boolean events** in a table with the following columns:

Column	Description	Value range
Name	Name for the event definition. The name must be unique within the resource.	Text, max. 32 characters
Global variable	Name of the assigned global variable (for example added using drag and drop).	
Data type	Data type of the global variable; cannot be modified.	BOOLEAN
Event source	CPU event: The processor module creates the timestamp. It creates all events in each of its cycles. Auto event: Like CPU event Default: Auto event	CPU, auto

21233799/EN – 07/2014

Column	Description	Value range
Alarm when FALSE	Activated: An event is triggered as soon as the value changes from TRUE to FALSE. Deactivated: An event is triggered as soon as the value changes from FALSE to TRUE. Default: Deactivated	Checkbox activated, deactivated
Alarm text	Text that specifies the alarm state.	Text
Alarm priority	Priority of the alarm state. Default: 500	0 – 1000
Alarm acknowledgement required	Activated: The user must acknowledge the alarm state (acknowledgement). Deactivated: The user need not acknowledge the alarm state (acknowledgement). Default: Deactivated	Checkbox activated, deactivated
Return to normal text	Text that specifies the alarm state.	Text
Return to normal severity	Priority of the normal state. Default: 500	0 – 1000
Return to normal ack required	The user must acknowledge the normal state (acknowledgement). Default: Deactivated	Checkbox activated, deactivated

Enter the **parameters of the scaled events** in a table with the following columns:

Column	Description	Value range
Name	Name for the event definition. The name must be unique within the resource.	Text, max. 32 characters
Global variable	Name of the assigned global variable (for example added using drag and drop).	
Data type	Data type of the global variable; cannot be modified.	Depending on the global variable type
Event source	CPU event: The processor module creates the timestamp. It creates all events in each of its cycles. Auto event: Like CPU event Default: Auto event	CPU, auto
HH alarm text	Text that specifies the alarm state of the maximum limit.	Text
HH alarm value	Maximum limit that triggers an event. Condition: (HH alarm value − hysteresis) > H alarm value or HH alarm value = H alarm value	Depending on the global variable type

Column	Description	Value range
HH alarm priority	Priority of the maximum limit. Default: 500	0 – 1000
HH alarm acknowledgement required	Activated: The user must acknowledge that the maximum limit has been exceeded. Deactivated: The user need not acknowledge that the maximum limit has been exceeded. Default: Deactivated	Checkbox activated, deactivated
H alarm text	Text that specifies the alarm state of the maximum limit.	Text
H alarm value	Upper limit that triggers an event. Condition: (H alarm value − hysteresis) > (L alarm value + hysteresis) or H alarm value = L alarm value	Depending on the global variable type
H alarm priority	Priority of the upper limit. Default: 500	0 – 1000
H alarm acknowledgement required	Activated: The user must acknowledge that the maximum limit has been exceeded. Deactivated: The user need not acknowledge that the maximum limit has been exceeded. Default: Deactivated	Checkbox activated, deactivated
Return to normal text	Text that specifies the alarm state.	Text
Return to normal severity	Priority of the normal state. Default: 500	0 – 1000
Return to normal ack required	The user must acknowledge the normal state (acknowledgement). Default: Deactivated	Checkbox activated, deactivated
L alarm text	Text that specifies the alarm state of the lower limit.	Text
L alarm value	Lower limit that triggers an event. Condition: (L alarm value + hysteresis) < (H alarm value − hysteresis) or L alarm value = H alarm value	Depending on the global variable type
L alarm priority	Priority of the lower limit. Default: 500	0 – 1000

21233799/EN – 07/2014

Column	Description	Value range
L alarm acknowledgement required	Activated: The user must acknowledge that the lower limit has been exceeded. Deactivated: The user need not acknowledge that the lower limit has been exceeded. Default: Deactivated	Checkbox activated, deactivated
LL alarm text	Text that specifies the alarm state of the lowest limit.	Text
LL alarm value	Lowest limit that triggers an event. Condition: (LL alarm value + hysteresis) < (L alarm value) or LL alarm value = L alarm value	Depending on the global variable type
LL alarm priority	Priority of the lowest limit. Default: 500	0 – 1000
LL alarm acknowledgement required	Activated: The user must acknowledge that the lowest limit has been exceeded. Deactivated: The user need not acknowledge that the lowest limit has been exceeded. Default: Deactivated	Checkbox activated, deactivated
Alarm hysteresis	The hysteresis prevents the permanent creation of many events in case the process value often oscillates around a limit.	Depending on the global variable type

NOTICE

Faulty events are possible due to improper parameter setting.

Setting the parameters *L alarm value* and *H alarm value* to the same value can cause an unexpected behavior of event creation because no normal range exists in this case.

12.6 Working with the user program

You can use the programming device to influence the program's functions in the controller.

12.6.1 Setting parameters and switches

During configuration of the user program, the parameters and switches are set to offline and are loaded into the controller together with the code-generated program. The parameters and switches can instead also be set when the controller is in STOP or RUN state provided that the *main enable* switch has been activated. Only the elements in the NVRAM can be modified. All other elements are activated during loading.

12.6.2 Starting the program from STOP/VALID CONFIGURATION

Starting the program has the same effect as switching the controller's mode of operation from STOP/VALID CONFIGURATION to RUN. The program also enters RUN mode. The program enters test mode if test mode is active when starting the program. In accordance with IEC 61131, a cold or warm start can also be performed in addition to starting in test mode.

INFORMATION

The program can only be started when the *Start/restart allowed* switch was activated.

12.6.3 Program restart after error

If the program enters STOP/INVALID CONFIGURATION state, for example due to unauthorized access to operating system areas, then it starts again. If the user program enters STOP/INVALID CONFIGURATION state again within approximately one minute after restart, then it remains in this state. In this case you can restart it using the start button of the control panel. After restart, the operating system checks the entire program.

12.6.4 Stopping the program

When stopping the user program, the controller switches from RUN to STOP/VALID CONFIGURATION mode.

12.6.5 Program test mode

Test mode is started via control panel by choosing [Test Mode] / [Test Mode with Hot Start] (or cold start/warm start). Each *Cycle step* command activates a single cycle (one complete logic cycle).

Behavior of variable values in test mode:

Choosing cold start, warm start, or hot start determines the variable values used for the first cycle in test mode.

- Cold start: All variables are set to their initial values.
- Warm start: Retain variables retain their values, others are set to their initial values.
- Hot start: All variables retain their current values.

You can then use the *Cycle step* command to start the user program in single step mode. All current values are retained for the next cycle (frozen state).

⚠ WARNING

Actuators in unsafe state.

Severe or fatal injuries.

Do not use the test mode function during safety-related operation.

21233799/EN – 07/2014

13 Operation

This chapter describes the handling and diagnostics during normal operation of the controller.

13.1 Operating

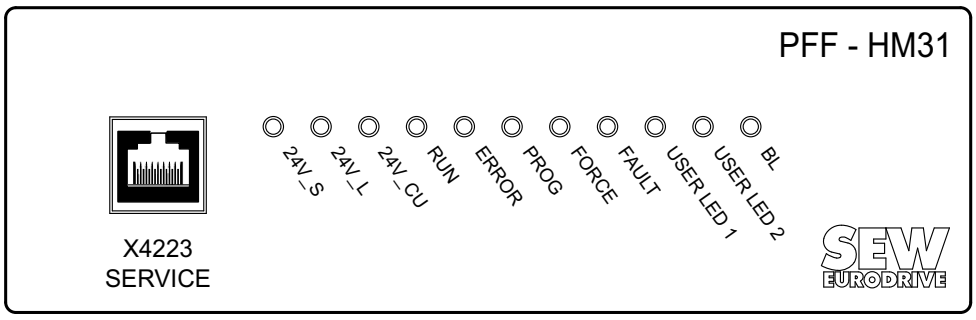
The controller need not be operated during normal operation. Intervention with the programming device is only necessary when problems occur.

13.2 Diagnostics

A first, rough diagnosis can be made by means of light emitting diodes (LEDs). A detailed analysis of the operating or error state can be made using the diagnostic history, which can be displayed with the programming device.

13.2.1 LED display

The system LEDs are located on the service unit and indicate the status of the fieldbus and the unit. There are also 2 user LEDs that can be configured by the user as required.



4867138571

The following table shows the status and the meaning of the LED.

Designation	Status LED	Meaning
BL	Flashing red	• BL (boot loader) defect or hardware error. • External process data communication error. • Duplicate IP address detected[1)]. • PROFINET has received an identify request.[1)]
	Off	None of the events described have occurred.
USER LED 2	Lights up red	Coding: 1 (see section "User LEDs")
USER LED 1	Flashing red	Coding: 2 (see section "User LEDs")
	Off	Coding: 0 or 3 – 255

21233799/EN – 07/2014

Designation	Status LED	Meaning
FAULT	Lights up yellow / flashing yellow[2)]	• The new operating system is corrupted (after downloading). • Error while loading a new operating system. • Incorrect configuration loaded. • One or more I/O errors have occurred. • Duplicate IP address detected.[1)] • PROFINET has received an identify request.[1)]
	Off	None of the events described have occurred.
FORCE	Yellow light	Forcing prepared: • The force switch of a variable has been set, • and the main force switch has not yet been enabled. • The unit is in RUN or STOP state.
	Flashing yellow	• Forcing is active: At least one local or global variable has assumed its force value. • Duplicate IP address detected.[1)] • PROFINET has received an identify request.[1)]
	Off	None of the events described have occurred.
PROG	Yellow light	• A new configuration is loaded to the unit. • A new operating system is loaded. • Changes to the watchdog time or safety time. • Check for duplicate IP address. • Changes to the SRS.
	Flashing yellow	• Reload function is being performed (the function must be enabled by means of a license). • Duplicate IP address detected.[1)] • PROFINET has received an identify request.[1)]
	Off	None of the events described have occurred.
ERROR	Red light / flashing red[2)]	• The unit status is set to FAULT STOP. Internal error detected by self-tests, such as hardware error, software error or voltage supply error. Remedy: The processor system can only be restarted by a command from the PADT (reboot). • Activated protocols/functions are not used (warning). • Error while loading the operating system.
	Off	None of the events described have occurred.

21233799/EN – 07/2014

Designation	Status LED	Meaning
RUN	Green light	• Unit status set to RUN, normal operation. • A loaded user program is running.
	Flashing green	• Unit status set to STOP. • A new operating system is being loaded.
	Off	The unit status is not set to RUN or STOP.
24V_CU	Green light	24 V is applied between X1541.1 and X1541.2.
24V_L	Green light	24 V is applied between X1541.3 and X1541.4.
24V_S	Green light	24 V is applied between X2312.1 and X2312.3.

1) If both LEDs are flashing at the same time: PROG, FORCE, FAULT and BL

2) The status "lit" signals a warning, and "flashing" signals an alarm.

Whenever the voltage supply is connected, all of the LEDs light up for a short time while they are tested.

User LEDs

The two freely configurable user LEDs (USER LED 1/2) are controlled using system variables. For this purpose, global variables of the USINT data type must be assigned to the corresponding system variables.

13.2.2 Diagnostic history

The diagnostic history records the various states of the processor system and the communication system, and stores these states in a non-volatile memory. Both systems include a short-term and a long-term diagnostics according to the table below:

	CPU	COM
Entries in the long-term diagnostics	700	300
Entries in the long-term diagnostics	700	700

Long-term diagnostics of the processor system includes the following events:

- Reboot
- Change of operating mode
  (INIT, RUN, STOP/VALID CONFIGURATION, STOP/INVALID CONFIGURATION)
- Change of program mode
  (START, RUN, ERROR, TEST MODE)
- Loading/deleting a configuration
- Setting/resetting switches
- Processor system errors
- Loading an operating system
- Forcing (setting and resetting the force switch is allowed)
- Diagnostics of power supply and temperature

Long-term diagnostics of the communication system includes the following events:

- Rebooting the communication system
- Change of operating mode (INIT, RUN, STOP/VALID CONFIGURATION, STOP/INVALID CONFIGURATION)
- Login of users
- Loading an operating system

If the memory for long-term diagnostics is full, all data older than three days is deleted so that new entries can be stored. If no data is older than three days, then new entries cannot be stored and are lost. A message in the long-term diagnostics indicates that it was not possible to store data.

Short-term diagnostics of the processor system includes the following events:

- Diagnostics of the processor system (setting force switch and force values)
- Diagnostics of the user program (cyclic operation)
- Diagnostics of the communication
- Diagnostics of power supply and temperature

Short-term diagnostics of the communication system includes the following events:

- safeethernet-related events
- Start/stop while writing the flash memory
- Errors that can occur when loading a configuration from the flash memory
- Unsuccessful time synchronization between communication system and processor system

Parameter errors associated with inputs or outputs are possibly not detected during code generation. In the event of a parameter error, the message INVALID CONFIG appears in the feedback window for the diagnostics indicating the error source and error code. This message helps to analyze errors caused by incorrect parameters set for inputs and outputs.

If the memory for the short-term diagnostics is full, the oldest entries are deleted so that new entries can be saved. No message appears to indicate that old entries are deleted.

The recording of diagnostic data is not safety-related. You can read the data recorded in chronological order for analysis purposes using the programming tool. Reading does not delete the data in the controller. The programming tool can store the content of the diagnostics window.

13.2.3 Diagnostics in SILworX®

Diagnostics is accessed via the online view in the hardware editor of SILworX®.

Do the following to open diagnostics:

1. Select the "Hardware" branch under the required resource.
2. Click [Online] in the context menu or on the action bar. The system login window opens.
3. In the system login window, select or enter the following information:
   - IP address of the controller
   - User name and password

   The online view of the hardware editor opens.
4. In the online view, select the required module, which is usually the processor module or the communication module.
5. Choose [Diagnostics] from the context menu or from the [Online] menu.

The diagnostics for the respective module opens.

During controller operation, messages appear at specific, user-defined time intervals. These messages indicate the state of the processor system, the communication system and the I/O modules.

21233799/EN – 07/2014

13.3 Parameters and error codes for inputs and outputs

The following overviews list the system parameters of the inputs and outputs that can be read and set, as well as error codes. The error codes can be read in the user program by means of the variables assigned in the logic. You can also have the error codes displayed in SILworX®.

13.3.1 Digital inputs of MOVISAFE® HM31

The following tables show the states and parameters for the digital input and output module (DI 26) in the same order as in the hardware editor.

Module tab

The module tab contains the following system parameters:

System parameter	Data type	R/W	Description	
DI number of pulsed channels	USINT	W	Number of pulse outputs (supply outputs).	
			Coding	Description
			0	No pulse output provided for SC/OC detection.[1)]
			1	Pulse output 1 provided for SC/OC detection.[1)]
			2	Pulse outputs 1 and 2 provided for SC/OC detection.[1)]
			...	...
			6	Pulse outputs 1 through 6 provided for SC/OC detection.[1)]
			Do not use pulse outputs as safety-related outputs.	
DI supply [01]	BOOLEAN	W	Activation of the supply output. TRUE: Supply is enabled FALSE: Supply is disabled	
DI pulse module slot	UDINT	W	Pulse module slot (SC/OC detection); set the value to "2".[1)]	
DI pulse delay [µs]	UINT	W	Waiting time for line control (detection of short circuits or cross circuits).	
DI.ErrorCode	WORD	R	Error codes for all digital inputs.	
			Coding	Description
			0x0001	Module fault.
			0x0002	FTT test of test pattern faulty.
			0x2000	Faulty parameter setting of SC/SO detection.[1)]
DI.ErrorCode supply	WORD	R	Module error codes of the supply output.	
			Coding	Description
			0x0001	Module fault.

21233799/EN – 07/2014

System parameter	Data type	R/W	Description	
DI[01].Error code supply	BYTE	R	Channel error codes of the supply output.	
			Coding	Description
			0x01	DI supply unit fault.
			0x02	Supply disconnected due to overcurrent.
			0x04	Error while reading back supply.
DO.ErrorCode	WORD	R	Error codes of all digital outputs.	
			Coding	Description
			0x0001	Module fault.
Module error code	WORD	R	Error code of the module.	
			Coding	Description
			0x0000	I/O processing, possibly with errors; see other error codes.
			0x0001	No I/O processing (CPU not in RUN state).
			0x0002	No I/O processing during booting test.
			0x0004	Manufacturer interface in operation.
			0x0010	No I/O processing: invalid parameter setting.
			0x0020	No I/O processing: fault rate exceeded.
			0x0040 / 0x0080	No I/O processing: configured module not plugged.
Module SRS	UDINT	R	Slot number (system rack slot).	
Module type	UINT	R	Module type, target value: 0x001A [26_{dec}].	

1) SC = short circuit / OC = open circuit (line control)

DI26 tab: DI channels

DI 26 tab: DI channels contain the following system parameters:

System parameter	Data type	R/W	Description	
Channel no.	—	R	Channel number, predefined.	
→ error code [BYTE]	BYTE	R	Error code of the digital input channels.	
			Coding	Description
			0x01	Fault in the digital input module.
			0x10	Short circuit of the channel.
			0x80	Interruption between pulse output (DO channel) and digital input DI, such as • Open circuit • Open switch • L+ undervoltage (+24 V_CU)

21233799/EN – 07/2014

System parameter	Data type	R/W	Description	
→ value [BOOLEAN]	BOOLEAN	R	Input values for the digital inputs. 0 = Input de-energized 1 = Input energized	
Pulsed channel [USINT] →	USINT	W	Source channel for pulsed supply.	
			Coding	Description
			0	Input channel
			1	Pulse of first DO channel
			2	Pulse of second DO channel
			...	...
			6	Pulse of sixth DO channel

DI26 tab: DO channels

DI 26 tab: DO channels contain the following system parameters:

System parameter	Data type	R/W	Description	
Channel no.	—	R	Channel number, predefined.	
→ error code [BYTE]	BYTE	R	Error codes of the digital outputs.	
			Coding	Description
			0x01	Fault in the digital output module or the module.
→ value [BOOLEAN]	BOOLEAN	W	Output value of DO channels.	
			Coding	Description
			0	Output de-energized
			1	Output energized

13.3.2 Digital outputs of MOVISAFE® HM31

The following tables show the states and parameters for the digital input and output module (DO 8) in the same order as in the hardware editor.

21233799/EN – 07/2014

Module tab

The module tab includes the following parameters:

System parameter	Data type	R/W	Description	
DO. Error code	WORD	R	Error codes of all digital outputs.	
			Coding	Description
			0x0001	Module fault.
			0x0002	Test of safety shutdown returns a fault.
			0x0004	Test of auxiliary voltage returns a fault.
			0x0008	FTT test of test pattern faulty.
			0x0010	Output switch test pattern faulty.
			0x0020	Output switch test pattern faulty (shutdown test of the outputs).
			0x0040	Active shutdown via WD faulty.
			0x0400	FTT test: 1st temperature threshold exceeded.
			0x0800	FTT test: 2nd temperature threshold exceeded.
			0x4000	Invalid parameter setting for 2-pole monitoring.
Activation delay	UINT	W	Activation delay for 2-pole tests due to conductor capacitances, inductive and capacitive load. Range of values: 0 – 30 ms	
Module error code	WORD	R	Error code of the module.	
			Coding	Description
			0x0000	I/O processing, possibly with errors; see other error codes.
			0x0001	No I/O processing (CPU not in RUN state).
			0x0002	No I/O processing during booting test.
			0x0004	Manufacturer interface in operation.
			0x0010	No I/O processing: invalid parameter setting.
			0x0020	No I/O processing: fault rate exceeded.
			0x0040/ 0x0080	No I/O processing: configured module not plugged.
Module SRS	UDINT	R	Slot number (system rack slot).	
Module type	UINT	R	Module type, target value: 0x0029 [41_{dec}].	

21233799/EN – 07/2014

DO 8 tab: Channels

DO 8 tab: Channels contains the following system parameters:

System parameter	Data type	R/W	Description	
Channel no.	—	R	Channel number, predefined.	
→ + error code [BYTE]	WORD	R	Error codes of the digital outputs.	
			Coding	Description
			0x0001	Fault in the digital input module.
			0x0002	Output disabled due to overcurrent.
			0x0004	Error while reading back the digital outputs.
			0x0008	Error while reading back the state of the digital outputs.
			0x0020	External short circuit or fault in the EMC protection provides an error (L+ short circuit (24V_L)).
			0x0040	External short circuit or fault in the EMC protection provides an error (L− short circuit (24V_L)).
			0x0080	Channel disabled due to error in assigned DO channel.
			0x0100	Output short-circuit test against L+ (24V_L) not performed due to changed target values or undervoltage.
			0x0200	Output short-circuit test against L− (0V24) not performed due to changed target values or undervoltage.
			0x0400	All channels disabled, total current exceeded.
			0x0800	FTT test: Monitoring of auxiliary voltage 1: undervoltage.
→ − error code [BYTE]	WORD	R	See -> + *error code [BYTE]*	
→ value [BOOLEAN]	BOOLEAN	W	Output value of the DO channels. 0 = Output de-energized 1 = Output energized	
2-pole switched off [BOOLEAN] →	BOOLEAN	W	Parameter setting whether the channel is used in 2-pole operation. 0 = 2-pole use of channel 1 = 1-pole use of channel	

13.3.3 Counters of MOVISAFE® HM31

The following tables show the states and parameters for the counter module (HSC 2) in the same order as in the hardware editor.

Module tab

The module tab includes the following parameters:

System parameter	Data type	R/W	Description	
Module error code	WORD	R	Error code of the module.	
			Coding	Description
			0x0000	I/O processing, possibly with errors; see other error codes.
			0x0001	No I/O processing (CPU not in RUN state).
			0x0002	No I/O processing during booting test.
			0x0004	Manufacturer interface in operation.
			0x0010	No I/O processing: invalid parameter setting.
			0x0020	No I/O processing: fault rate exceeded.
			0x0040/ 0x0080	No I/O processing: configured module not plugged.
Module SRS	UDINT	R	Slot number (system rack slot).	
Module type	UINT	R	Module type, target value: 0x0003 [3_{dec}].	
Counter.ErrorCode	WORD	R	Error code of the counter module.	
			Coding	Description
			0x0001	Module fault.
			0x0002	Error while comparing the time base.
			0x0004	Address error while reading the time base.
			0x0008	Invalid parameter for the time base.
			0x0010	Address error while reading the counter state.
			0x0020	Corrupted parameter setting for counter.
			0x0040	Address error while reading the gray code.
			0x0080	FTT test of test pattern faulty.
			0x0100	FTT test fault while checking the coefficients.
			0x0200	Error during initial configuration of the module.

HSC 2 tab: Channels

HSC 2 tab: Channels contains the following system parameters:

System parameter	Data type	R/W	Description
Counter[0x]. 5/24V mode	BOOLEAN	R/W	Counter input 5 V or 24 V. TRUE: 24 V FALSE: 5 V

21233799/EN – 07/2014

System parameter	Data type	R/W	Description
Counter [0x]Autom. direction of rotation detection	BOOLEAN	R/W	Automatic detection of the direction of rotation. TRUE: Automatic detection of rotation direction ON FALSE: Manual setting of rotation direction
Counter[0x].Error Code	BYTE	R	Error codes of the counter channels. Coding: Description 0x01: Fault in the counter module. 0x02: Error while comparing the counter states. 0x08: Error while setting the configuration (reset).
Counter[0x].Gray Code	BOOLEAN	R/W	Decoder/pulse mode. TRUE: Gray code decoder FALSE: Pulse mode **Decoder mode is not permitted.**
Counter[0x].Spare1	BOOLEAN	R/W	No function.
Counter[0x].Spare2	BOOLEAN	R/W	
Counter[0x].Spare3	BOOLEAN	R/W	
Counter[0x].Reset	BOOLEAN	R/W	Reset of the counter channel (only if *Counter[0x].Autom. Direction of rotation detection = FALSE*) TRUE: No reset FALSE: Reset
Counter[0x].Direction	BOOLEAN	R/W	Counting direction of the counter (only if *Counter[0x].Autom. Direction of rotation detection = FALSE*). TRUE: Decrement FALSE: Increment
Counter[0x].Value	UDINT	R	Counter state of the counters: 24 bits for pulse counter.
Counter[0x].Value Overflow	BOOLEAN	R	Counter overflow indication. TRUE: Overflow since the last cycle (only if *Counter[0x].Autom. Direction of rotation detection = FALSE*) FALSE: No overflow since the last cycle
Counter[0x].Timestamp	UDINT	R	Timestamp for *Counter[0x].Value*, 24 bits, time resolution 1 µs.

21233799/EN – 07/2014

System parameter	Data type	R/W	Description
Counter[0x].Time Overflow	BOOLEAN	R	Overflow indication for the timestamp of the counter. TRUE: 24-bit overflow since the last cycle FALSE: No overflow since the last cycle

21233799/EN – 07/2014

14 Service

14.1 Inspection and maintenance

The safety controller does not require any maintenance. SEW-EURODRIVE does not stipulate any regular inspection work. However, it is recommended that you check the following parts regularly:

- Connection cables:

  Damaged or fatigued cables must be replaced immediately.

INFORMATION

Only SEW-EURODRIVE is authorized to carry out repairs.

14.2 Unit replacement

14.2.1 Requirements

To replace a MOVISAFE® HM31 safety controller, the following requirements must be met:

- All connections to inputs and outputs must be disconnected.
- All communication connections must be disconnected.
- External wiring is not allowed for the safety controller.
- The documentation/description of electrical connections must be at hand.
- A programming tool (PC/notebook) with installed SILworX® software (version 4.116, SILworX® full version) and a USB dongle (hardlock including SEW option) must be at hand.
- The corresponding SILworX® project must be available.
- In SILworX®, the project compiled with no errors (code generation done twice) must be open.
- An Ethernet cable must be available.
- SEW-EURODRIVE employees use the respective form for documentation purposes during unit replacement.

INFORMATION

It is not possible to upload a compiled user program from the previous controller to a programming and debugging tool (PADT) and then download this to the new controller.

14.2.2 Connection to the safety controller

Connect the programming tool to the safety controller (X4223: Ethernet service interface).

Login

To log in, enter the corresponding target IP address. However, the IP address is not required for system operation. The IP address of a module is stored in a non-volatile memory within the module.

21233799/EN – 07/2014

The IP address is selected according to the following priorities:

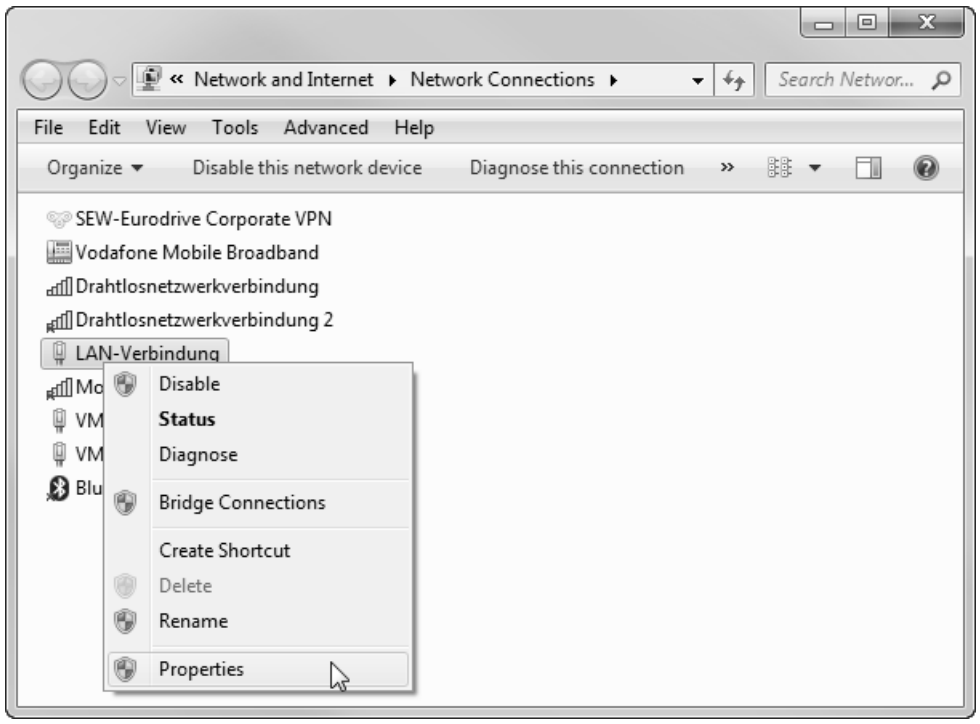
- If a valid SILworX® configuration is loaded, the IP addresses are taken from this configuration.
- If no valid configuration is available, the last valid IP address of the module is used. Bear this in mind if modules are used that have previously been used in another application.
- Factory settings of MOVISAFE® HM31:
  - Standard IP address of the CPU module: 192.168.0.99
  - Standard IP address of the COM module: 192.168.0.100
  - Subnet mask: 255.255.252.0
  - Standard system ID: 60000
- To ensure that a unique IP address is defined for the current module, we recommend that you read out the IP address using the "Search via MAC" dialog box, and then use this IP address for the first login.
- The IP address of the programming tool must match the subnet mask and be located in the same network as the IP address of the module to be connected. It might be necessary to correct the IP address of the PC.

Setting the IP address of the programming tool

The following example shows how to set the IP address of the programming tool. If the programming tool is equipped with multiple network cards, make sure that you select the correct network card required for the application.

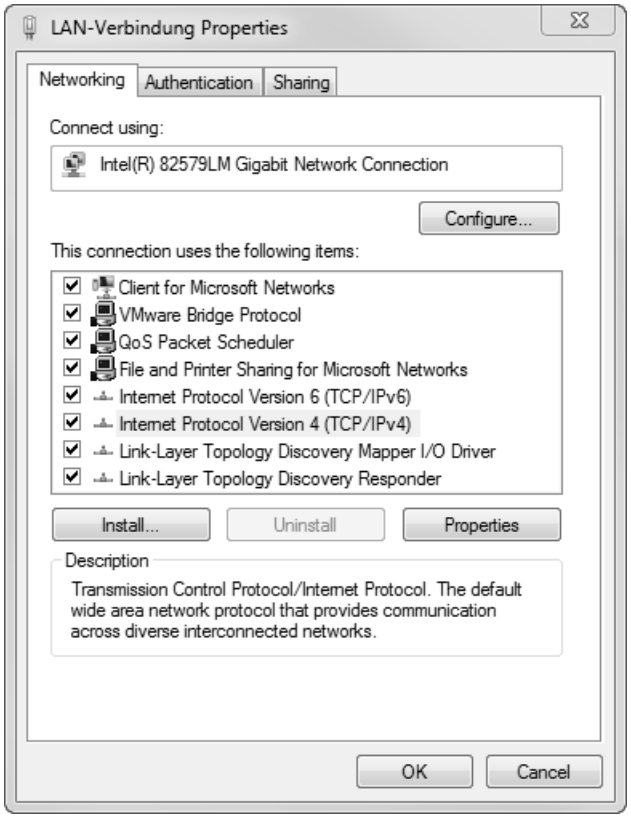
Proceed as follows:

1. Open the context menu of the network card and choose [Properties].



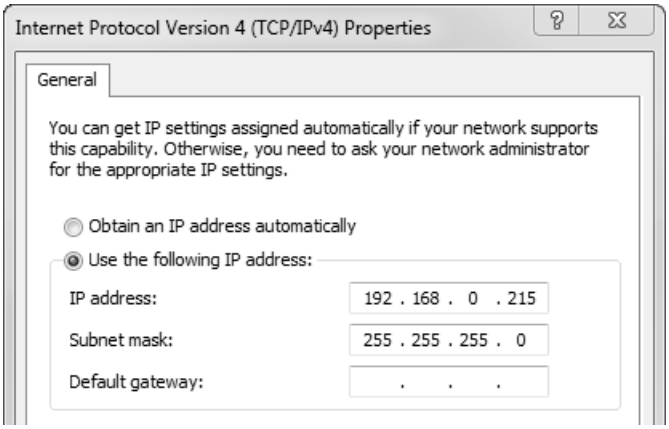
12085296395

2. On the "Networking" tab, choose "Internet Protocol Version 4" and click [Properties].



12085301387

3. The window "Internet Protocol Version 4 (TCP/IPv4) Properties" opens. Choose "Use the following IP address" and enter the IP address required in your project as well as the subnet mask.



12085359243

4. Confirm your entries with [OK]. You have now set the IP address of the programming tool.

14.2.3 Verifying system data

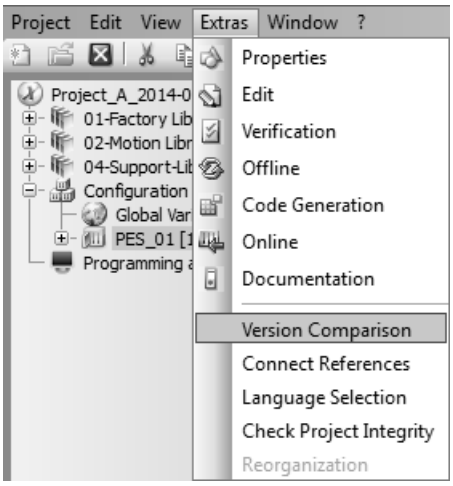
If the safety controller is still in working order, the current configuration CRC that is active in the safety controller and the "Configuration path in FS" parameter must be compared and documented to provide proof for the operating license (acceptance).

This additionally requires a version comparison of the last loaded version for the corresponding resource. This comparison ensures that the documentation contains the current CRCs of code generation.

Version comparison

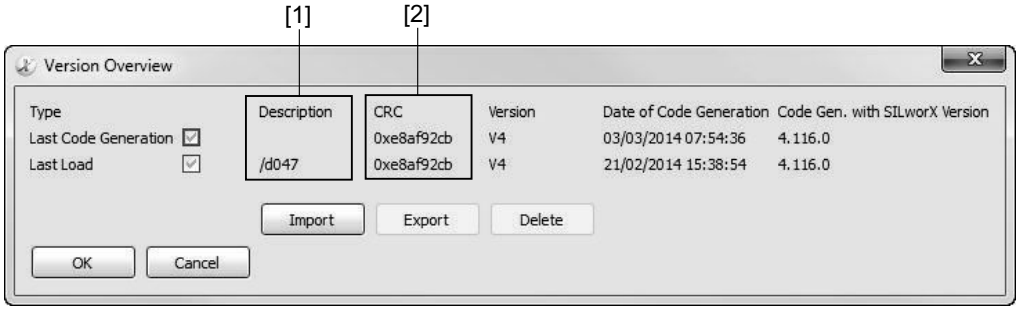
Proceed as follows to perform the version comparison for a resource.

1. In the structure tree, select the respective resource (for example "Project B"). Choose [Version Comparison] from the [Extras] menu.



12085364235

2. The "Version Overview" window opens. The two CRCs [2] for "Last Code Generation" and "Last Load" must be identical. These CRCs must also correspond with the CRC from the project documentation. In addition, the "Description" parameter [1] must be compared under "last load".



12085370123

3. You have now finished comparing the resource versions.

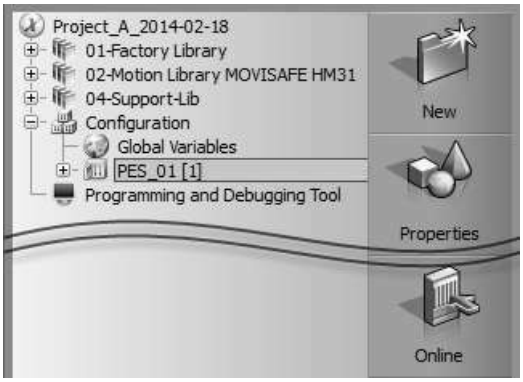
Control panel (online)

If the safety controller is still in working order, the parameters "CRC" and "Description" from the version comparison must correspond with the respective parameters in the system data (online).

Proceed as follows to perform a system login:

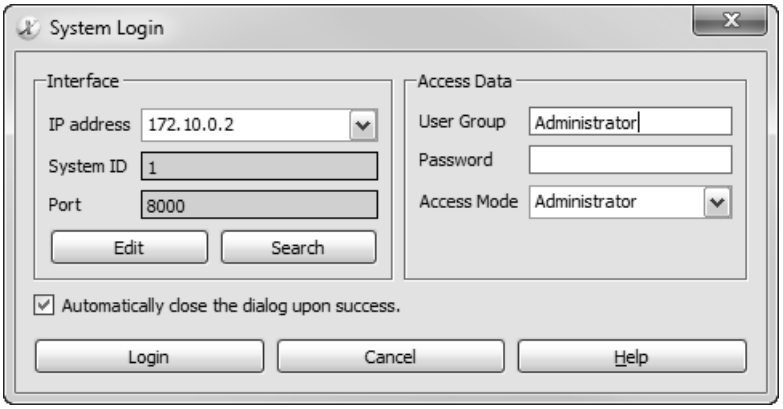
1. In the structure tree, select the corresponding resource (for example "Project B"), and click [Online] in the action bar. The "System Login" dialog box appears.

21233799/EN – 07/2014



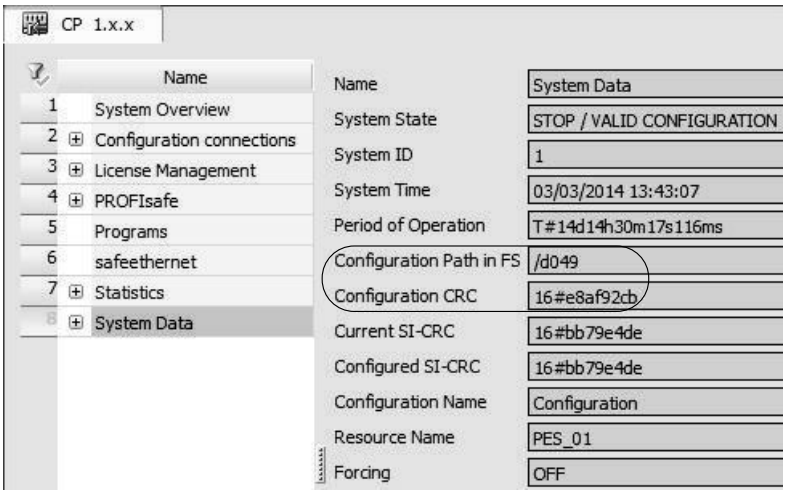
12085478283

2. In the "Interface" group box, check whether the correct IP address is displayed in the "IP Address" drop-down list. For authorization, enter the data for the default user group in the "Access Data" group box:
   - Click the "User Group" field and press <Ctrl + A>.
   - The "User Group" and "Access Mode" fields are automatically filled in.



12085484171

3. Click the [Login] button. The control panel for the resource opens.
4. Under "System Data", you find the parameters "Configuration Path in FS" and "Configuration CRC".



12085629579

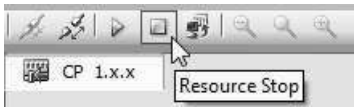
5. These parameter values must match the version comparison values (see chapter "Version comparison") as well as the project documentation values. The "Configuration Path in FS" parameter corresponds with the "Description" parameter in the version comparison.

Stopping a resource

The safety controller must be in "STOP" state before it can be replaced. The reason for this is that, for example during debugging, the diagnostics entries of the resource are overwritten in case of a faulty output.

Proceed as follows to stop the resource:

1. Click the "Resource Stop" button on the tool bar.



12085635339

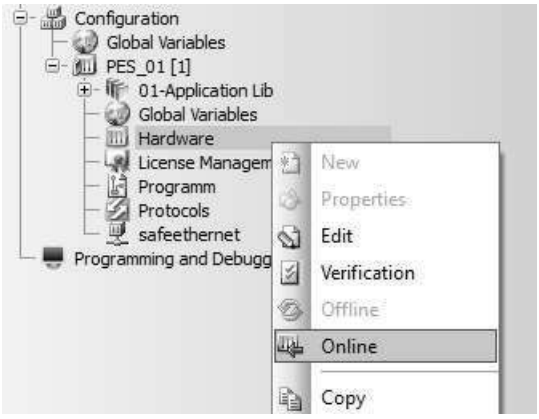
2. Disconnect the voltage supply of the safety controller and replace the controller.

14.2.4 Saving diagnostic data (CPU and COM)

If the safety controller is still in working order, you have to save the diagnostic entries of the COM module and the CPU module for subsequent analysis.

Proceed as follows:

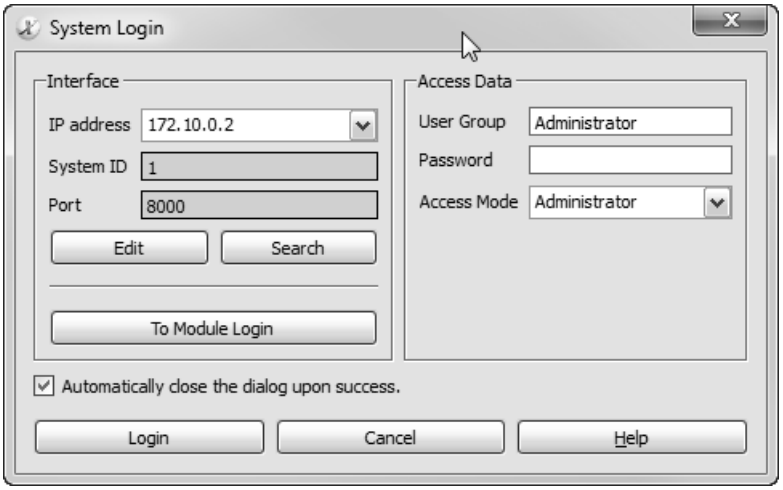
1. In the structure tree, select the hardware. Next, click [Online] in the action bar, or open the context menu and choose [Online]. The "System Login" dialog box appears.



12085999627

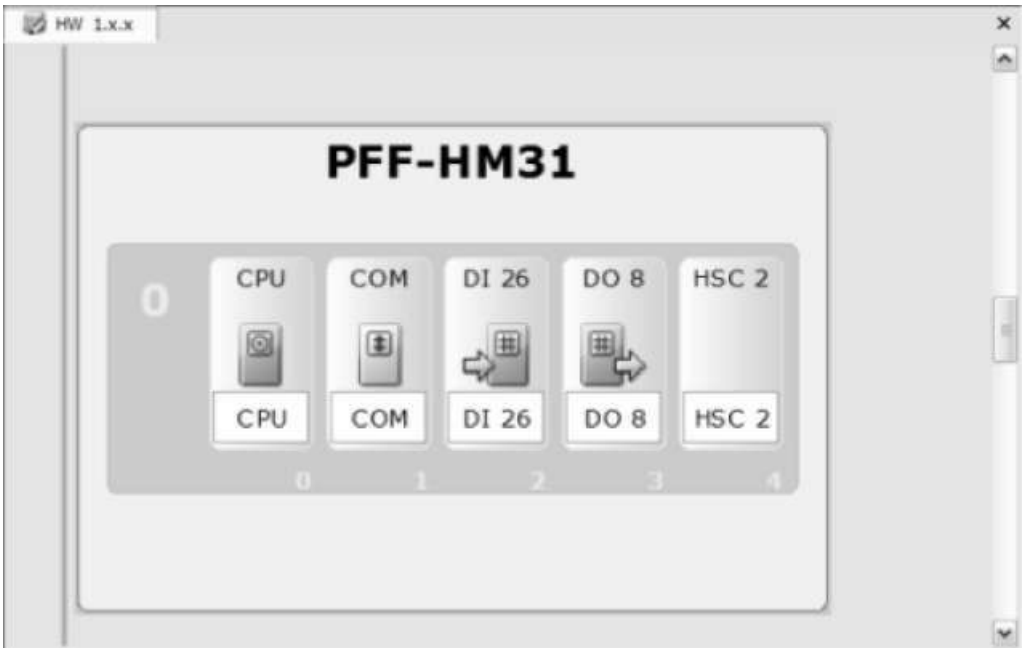
2. In the "Interface" group box, check whether the correct IP address is displayed in the "IP Address" drop-down list. For authorization, enter the data for the default user group in the "Access Data" group box:
   - Click the "User Group" field and press <Ctrl + A>.
   - The "User Group" and "Access Mode" fields are automatically filled in.

21233799/EN – 07/2014



12086004235

3. Click the [Login] button. The hardware view of the resource opens.



12086008843

4. In the online view of the Hardware Editor, right-click a module icon. Choose [Diagnostics] from the context menu. The diagnostics view opens.

   Observe the following notes:

   - **Modules with warnings** are displayed in **yellow**
   - **Modules with errors or faults** are displayed in **red**

5. Choose [All entries] to display the entire content of the diagnostic memory.

   Choose [Entries since] and modify the date and time to have displayed only entries from a certain date.

6. To save the diagnostic memory in a file for further evaluation, do the following:

   - Right-click anywhere in the list to open the context menu and choose [Save].

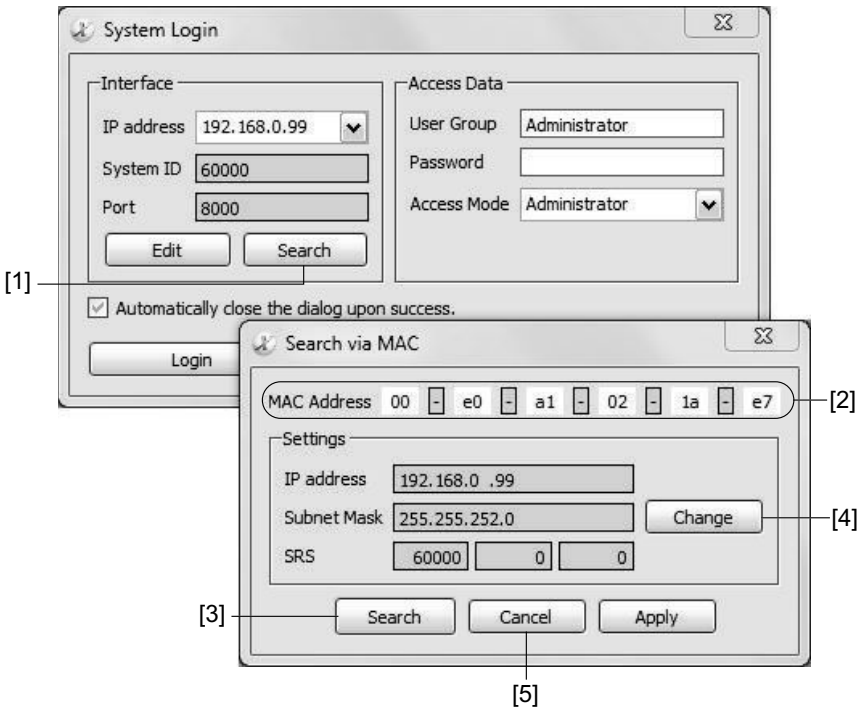
     The data is saved as readable XML file including some basic data of the module.

- Save the diagnostic file with a unique file name and, if required, send the file to SEW-EURODRIVE.

14.2.5 Starting up MOVISAFE® HM31 with factory settings

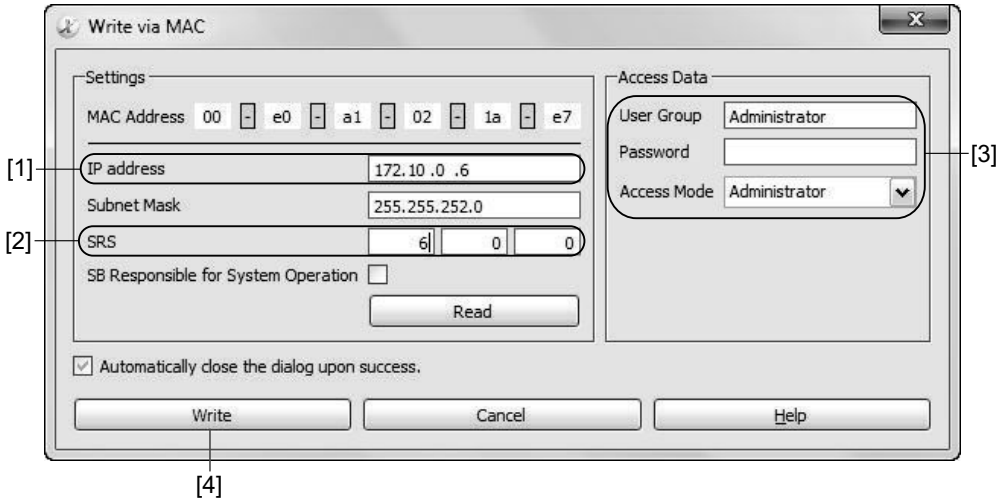
To start up a MOVISAFE® HM31 safety controller with factory settings, proceed as follows:

1. Disconnect all communication connections and all connections to inputs and outputs. External wiring is not allowed for the safety controller.
2. Switch on the voltage supply and wait for the initialization process to be completed (RUN LED is flashing). Connect the programming tool to the safety controller using an Ethernet cable.
3. Start SILworX® and open your project.
4. Select the resource name in the structure tree and click "Online" in the action bar. The "System Login" dialog box opens.
5. In the "Interface" group box, click the [Search] button [1]. The "Search via MAC" dialog box opens.
6. In the "MAC address" field [2], enter the MAC address of the CPU. You find the MAC address on a label on the safety controller.
7. Click the [Search] button [3]. The data for "IP address", "Subnet Mask" and "SRS" are read out and displayed in the "Settings" group box.



12086117899

8. Click the [Change] button [4] (see previous figure). The "Write via MAC" dialog box opens (see figure below).



12086382091

9. Use the system ID and the IP address of the system login dialog. Enter the system ID (in the example: 6) in the "SRS" field [2], and enter the IP address [1] (in the example 172.10.0.6).
10. For authorization, enter the data for the default user group in the "Access Data" group box [3]. Click the "User Group" field and press [Ctrl + A]. The "User Group" and "Access Mode" fields are automatically filled in.
11. Click the [Write] button [4].
12. Next, close the "Search via MAC" dialog box by clicking [Cancel] [5]. You can now load a program into the controller and start it.

14.2.6 Starting up MOVISAFE® HM31 without factory settings

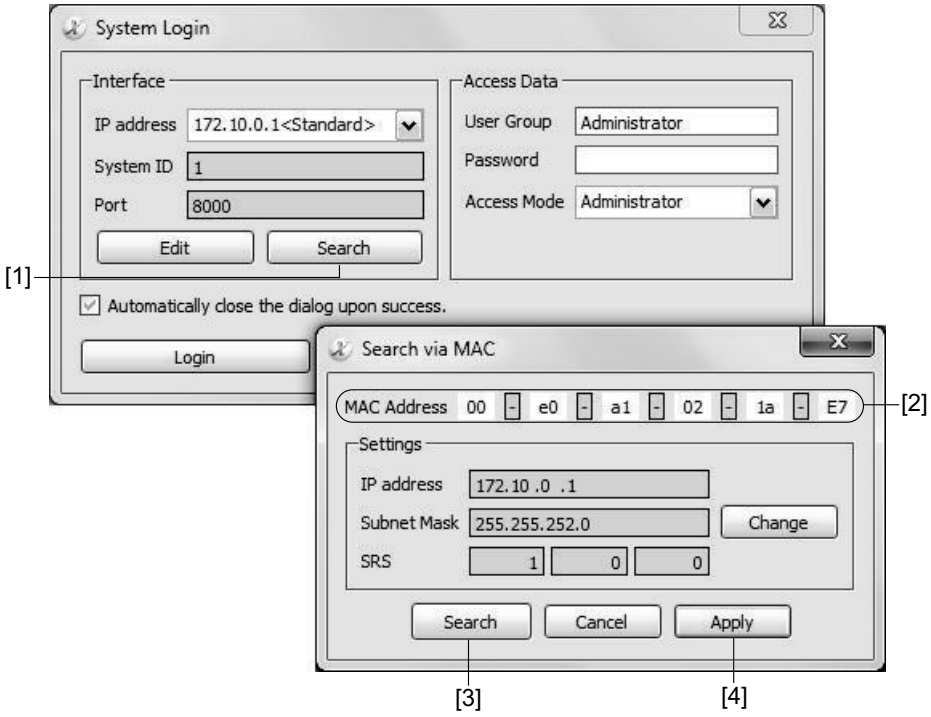
To start up a MOVISAFE® HM31 safety controller without factory settings, proceed as follows:

1. Disconnect all communication connections and all connections to inputs and outputs. External wiring is not allowed for the safety controller.
2. Switch on the voltage supply and wait for the initialization process to be completed (RUN LED is flashing). Connect the programming tool to the safety controller using an Ethernet cable.
3. Start SILworX® and open your project.
4. Select the resource name in the structure tree and click "Online" in the action bar. The "System Login" dialog box appears and displays the Ethernet parameters in accordance with the project settings.

The Ethernet parameters of the controller are unknown

Proceed as follows if the current Ethernet parameters of the controller are not known:

1. In the "System Login" window, click [Search] [1]. The "Search via MAC" window opens.



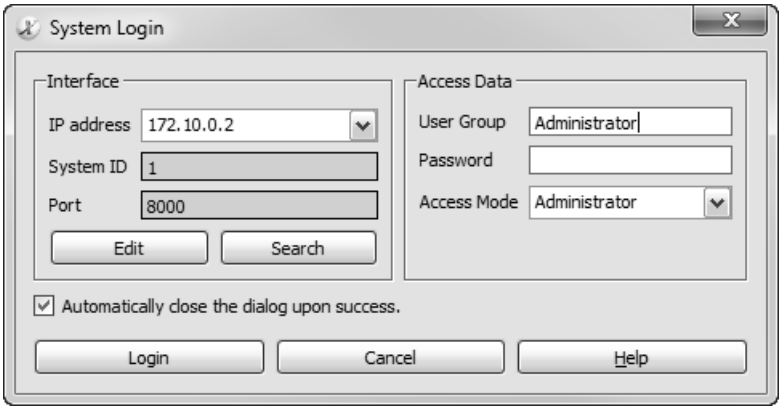
12086505227

2. In the "MAC address" field [2], enter the MAC address of the CPU. You find the MAC address on a label on the safety controller.
3. Click the [Search] button [3]. The data for "IP Address", "Subnet Mask" and "SRS" is read out and displayed in the "Settings" group box.
4. Click [Apply] [4]. The read-out data is inserted in the "System Login" dialog box.

Logging in to the system

Proceed as follows:

1. For authorization, enter the data for the default user group in the "Access Data" group box:
   - Click the "User Group" field and press <Ctrl + A>.
   - The "User Group" and "Access Mode" fields are automatically filled in.



12085484171

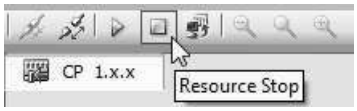
21233799/EN – 07/2014

2. Click [Login]. The control panel for the resource opens. If the data specified for the default user group is not accepted, a user management scheme is configured on the safety controller. The administrator data defined in this user management scheme must be used for the login. If this information is not known, reset the safety controller to the factory settings.

Setting the system ID

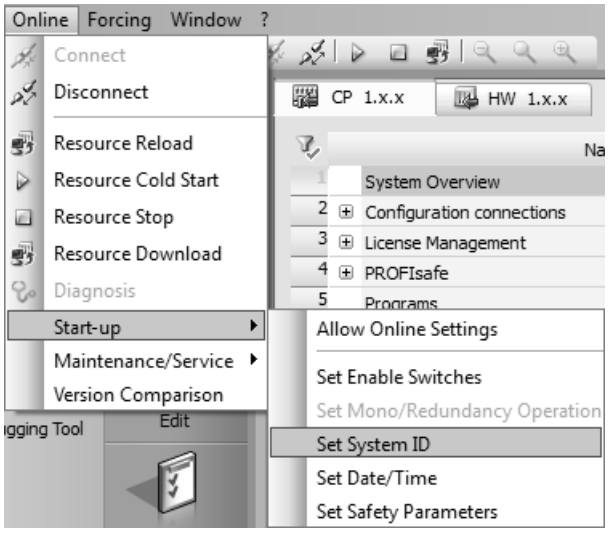
Proceed as follows:

1. Ensure that the system is in "STOP" state. Otherwise, the system ID cannot be modified.
2. Click the "Resource Stop" button on the tool bar.



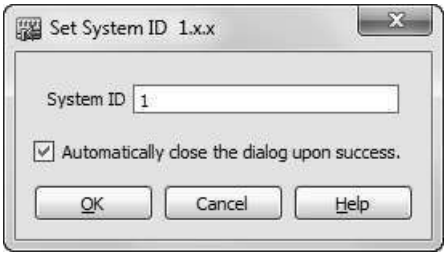
12085635339

3. From the [Online] menu, choose [Start-up] / [Set System ID].



12086834187

4. The "Set System ID" window opens. The current system ID is displayed in the header.



12088041483

5. Enter the required system ID and click [OK].
6. Close the Control Panel and continue with chapter "Loading and starting the resource (MOVISAFE® HM31)".

INFORMATION

Changing the system ID interrupts the communication between PADT and safety controller because the system login was performed with another system ID.

14.2.7 Loading and starting the resource (MOVISAFE® HM31)

Requirements

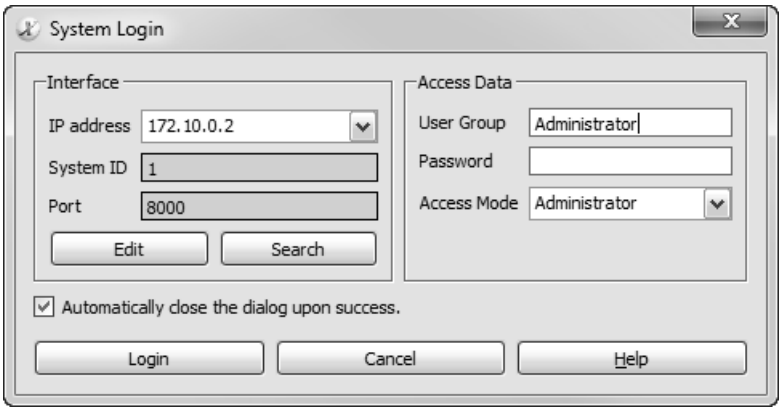
To load and start a resource, the controller must be started up as described in the chapters "Starting up MOVISAFE® HM31 with/without factory settings". The following requirements must be met:

1. The system ID used in the project must be configured in the MOVISAFE® HM31 safety controller.
2. A project compiled without errors must be open in SILworX®.
3. The user must be authorized to perform a system login with write access.

Logging in to the system

Proceed as follows:

1. For authorization, enter the data for the default user group in the "Access Data" group box:
   - Click the "User Group" field and press <Ctrl + A>.
   - The "User Group" and "Access Mode" fields are automatically filled in.



12085484171

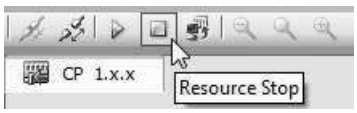
2. Click the [Login] button. The control panel for the resource opens.

Performing a download

A download can only be performed if the system is in "STOP" state. The system state is displayed in the "System Information" group box of the control panel.

Proceed as follows:

1. On the tool bar, click the "Resource Stop" icon.



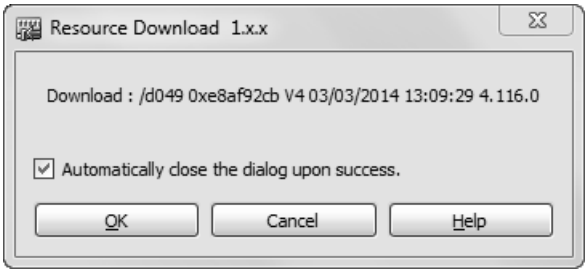
12085635339

2. Click "Resource Download" in the tool bar.

21233799/EN – 07/2014

12088151691

3. The "Resource Download" dialog box opens. Click [OK] to start the download.



12088159883

INFORMATION

After successful download, the IP addresses configured in the project take effect. If the new IP address of the resource differs from the IP address used for the login, then the connection between the programming tool and the resource is interrupted.

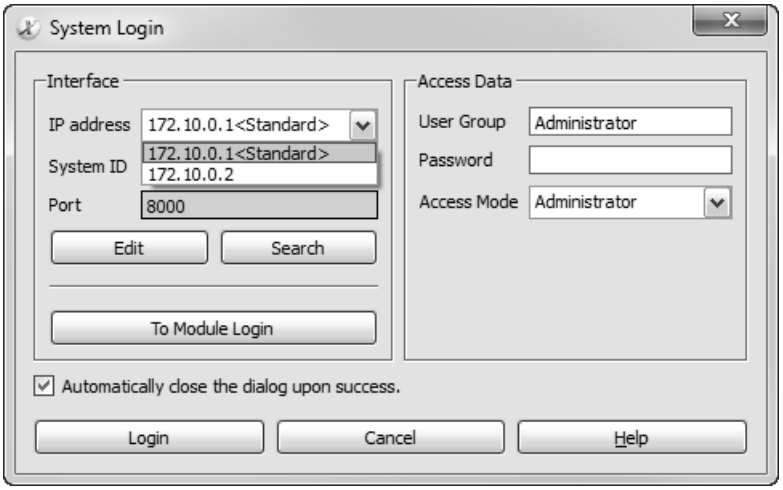
Resource cold start

Proceed as follows:

1. If the connection is lost after performing a download, log in again. For this purpose, click "Connect" in the tool bar.

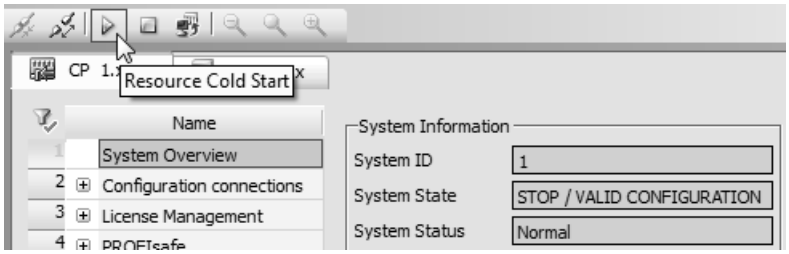
12088190475

2. The "System Login" dialog box opens. In the "Interface" group box, select the required IP address from the "IP address" drop-down list.



12088307083

3. Click the [Login] button. In the control panel, click "Resource Cold Start" in the tool bar.



12088313355

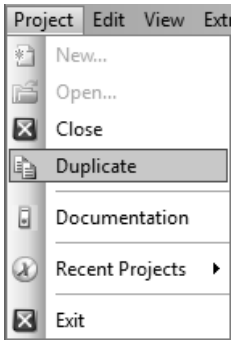
4. The CPU changes to "RUN" state. Also refer to the system information (system ID, system state, system status) in the control panel.

Creating a backup

Always create a backup copy of your project in a separate directory after every loading. Activate write protection for the backup. Doing so ensures that the backup copy (a project) is not changed accidentally.

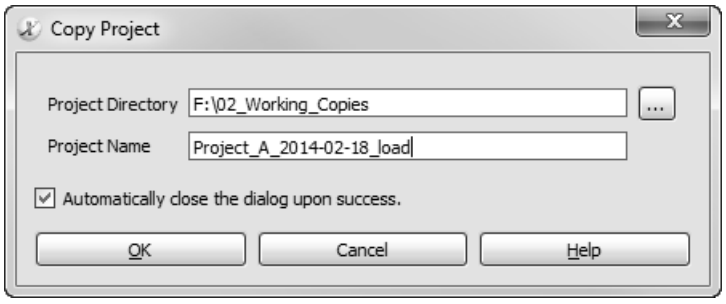
Proceed as follows to create a backup:

1. In SILworX®, choose [Duplicate] from the [Project] menu.



12088433547

2. The "Copy Project" window opens. Select the project directory in which you want to create the project copy. Specify a project name with comments.



12088438155

3. Click [OK]. A project copy is created.

14.2.8 Electrical installation

Once the safety controller has changed to "RUN" state without faults, you can switch off the voltage supply of the controller and re-establish all communication connections an all connections to inputs and outputs. Note the corresponding documentation/ description of electrical connections.

14.2.9 Verification

After replacing the unit and establishing all connections to inputs and outputs and to communication interfaces, you have to verify all relevant parameters of the safety controller. This includes, for example, the system data (configuration CRC, configuration path in FS), the encoder signals, and the communication interface. SEW-EURODRIVE uses the respective form for this purpose.

14.3 Fault information

Most faults in the processor system (CPU) cause the whole control unit to shut down. Faults are indicated by the "ERROR" status LED.

You can delete the display by executing the "Reboot resource" command in the [Extras] menu on the SILworX® Control Panel. The control unit is booted and re-started. The system automatically detects faults in the input and output channels during operation. These faults are signaled by the "FAULT" status LED on the top of the unit.

Even if the controller is stopped, the diagnostic history of the PADT (SILworX®) can be used to read out detected faults unless the communication is disrupted as well.

- Before replacing a control unit, check for an external line fault and ensure that the corresponding sensor/actuator is in working order.

14.4 Loading operating systems

Different operating systems are used for the processor system and communication system. These operating systems are stored on rewritable flash memories and can be replaced if necessary.

⚠ WARNING

Interruption to fail-safe operation caused by new operating systems being loaded by the programming tool.

Severe or fatal injuries.

- The controller must be in STOP state to enable the programming tool to load new operating systems.
- The operator must ensure that the safety of the system is guaranteed during this time, for example by taking the necessary organizational measures.

INFORMATION

- The programming tool prevents operating systems from being loaded during RUN state, and signals this accordingly.
- If the loading process is interrupted or incorrectly terminated, the control unit no longer functions. However, you can reload an operating system.

You must load the operating system for the processor system (CPU operating system) before you load the operating system for the communication system (COM operating system). In order for a new operating system to be loaded, it must be stored in a directory that can be accessed using the programming tool.

21233799/EN – 07/2014

14.4.1 Loading operating systems with SILworX®

Proceed as follows to load a new operating system:

1. Set the controller to STOP state unless this has already taken place.
2. Open the online view for the hardware and log into the control unit with administrator rights.
3. Right-click the module you want to load (processor module or communication module).
4. From the context menu, choose [Maintenance/Service] / [Load operating system module].
5. Choose the type of operating system you want to load from the “Load operating system module” dialog box.
6. In the open file selection window, select the file containing the operating system to be loaded and click [Open].

SILworX® then loads the new operating system into the controller.

14.5 Shutdown

To shut down the safety controller, disconnect the unit using appropriate measures.

21233799/EN – 07/2014

15 Appendix

15.1 Glossary

Term	Description
DC 24 V	The safety controller has the following DC 24 V input voltage potential: 24V_CU: DC 24 V input – controller 24V_L: DC 24 V input – load 24V_S: DC 24 V input – sensor supply Reference potential: 0V24
A3	SILworX® archive file
ARP	Address resolution protocol (network protocol for assigning network addresses to hardware addresses)
BS	Operating system
BL	Boot loader
BWS	Contactless protection device
COM	Communication module
COE	CANopen software module
CRC	Cyclic redundancy check (checksum)
CUT	COM user task
DCS	Distributed control system (process control system)
DI	Digital input (binary input)
DO	Digital output (binary output)
EMC	Electromagnetic compatibility
EN	European standard
ESD	Electrostatic discharge
FB	Fieldbus interface of the controller
FBD	Function block language
FIFO	First-in first-out (data memory)
FTA	Field termination assembly
FTT	Fault tolerance time
ICMP	Internet control message protocol (network protocol for status and error messages)
IEC	International Electrotechnical Commission
IF	InterFace
MAC address	Media access control address (hardware address of a network connection)
NVRAM	Non volatile random access memory

21233799/EN – 07/2014

Term	Description
PADT	Programming and debugging tool (in accordance with IEC 61131-3), PC with SILworX®
PE	Protective earth
PELV	Protective extra low voltage
PES	Programmable electronic system
POU	Program organizational units (in accordance with IEC 61131-1)
PFD	Probability of failure on demand
PFF-HM31A	Safety controller
PFH	Probability of failure per hour
R	Read (system variable provides a value, for example to the user program)
Non-reactive	Supposed two input circuits are connected to the same source (for example a transmitter). In this case, an input circuit is referred to as non-reactive if it does not distort the signals of the other input circuit.
R/W	Read/Write (column title for system variable type)
SB	System bus (module)
SELV	Safety extra low voltage
SFF	Safe failure fraction
SIL	Safety integrity level (according to IEC 61508)
SILworX®	Programming tool for PFF-HM31A safety controllers
SNTP	Simple network time protocol (RFC 1769)
S.R.S	System.Rack.Slot (addressing of a module)
SW	Software
S&R	Send and Receive; in connection with TCP protocols
TMO	Timeout
W	Write (system variable is provided with a value, for example from the user program)
Watchdog (WD)	Time monitoring for modules or programs. A fault stop will occur in the module or program if the watchdog time is exceeded.
WDT	Watchdog time

21233799/EN – 07/2014

Index

A

C

D

E

F

21233799/EN – 07/14

G

H

I

L

M

N

O

21233799/EN – 07/14

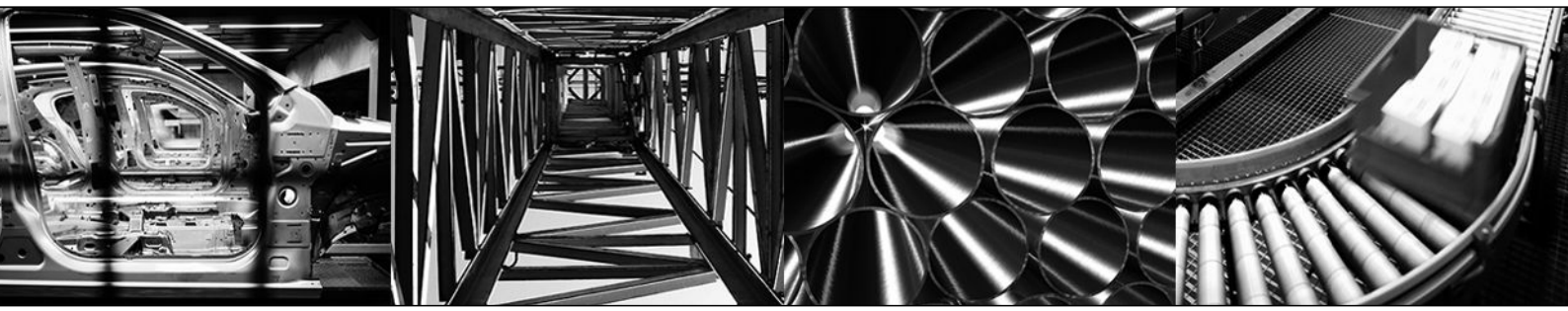
21233799/EN – 07/14

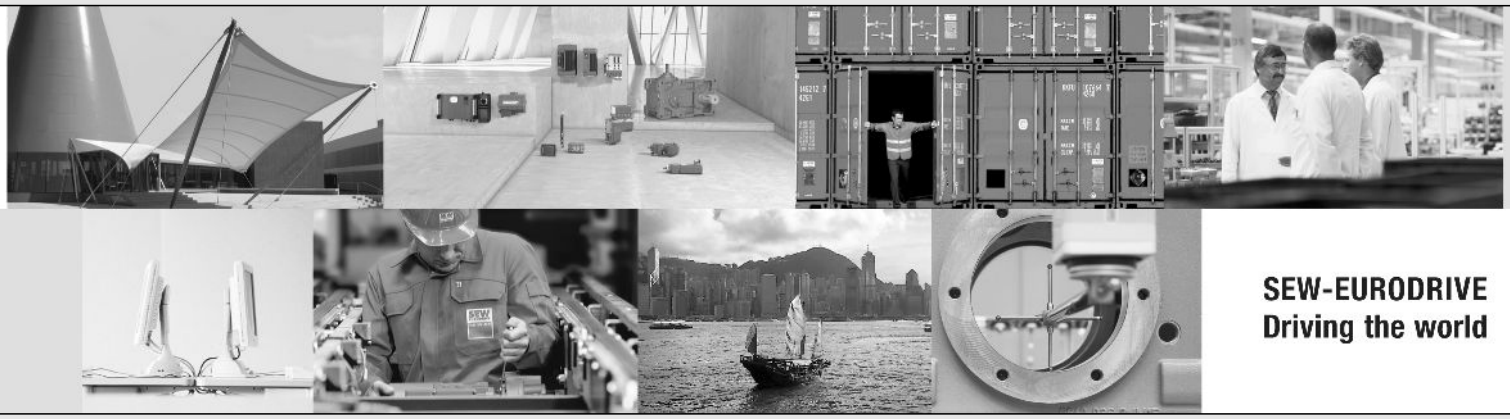
T

U

W

21233799/EN – 07/14





SEW-EURODRIVE
Driving the world

SEW-EURODRIVE GmbH & Co KG
P.O. Box 3023
76642 BRUCHSAL
GERMANY
Phone +49 7251 75-0
Fax +49 7251-1970
sew@sew-eurodrive.com
➞ **www.sew-eurodrive.com**