Manual

HIMax®

System Manual

Contact

HIMA Paul Hildebrandt GmbH

P.O. Box 1261

68777 Brühl

Phone: +49 6202 709-0

Fax: +49 6202 709-107

E-mail: info@hima.com

Document designation	Description
HI 801 000 D, Rev. 11.00 (1925)	German original document
HI 801 001 E, Rev. 11.00.00 (1927)	English translation of the German original document

Table of Contents

1 Introduction

The system manual describes the configuration and mode of operation of the safety-related HIMax controller system. The system is designed for safety-related applications up to SIL 3 (IEC 61508), PL e (EN ISO 13849) and for maximum availability.

HIMax can be used for various control tasks within the process and factory automation industry.

1.1 Structure and Use of the Document

This system manual is composed of the following main chapters:

Introduction	Introduction to this manual.
Safety	Information on how to safely use the HIMax system.
Product description	Structure of the HIMax systems.
Redundancy	Options for increasing availability.
Programming	Important instructions on how to create a user program.
Diagnostics	Summary of the diagnostic options.
Product data, dimensioning.	Specifications related to the entire system. Specifications for the individual components are included in the corresponding manual.
Lifecycle	Phases of a HIMax system lifecycle: ▪ Installation. ▪ Start-up. ▪ Maintenance and repairs.
Documentation	Overview of the documentation.
Appendix	▪ Examples of how to configure the HIMax systems ▪ Glossary. ▪ Index of tables and index of figures. ▪ Index.

1.2 Target Audience

This document is aimed at the planners, design engineers, programmers and the persons authorized to start up, operate and maintain the automation systems. Specialized knowledge of safety-related automation systems is required.

All specialist staff (planning, installation, start-up) must be instructed concerning the risks and the associated possible consequences which can arise as a result of modifications to a safety-related automation system.

Planners and configuration engineers must have additional knowledge about the selection and use of electrical and electronic safety systems in automated plants, e.g., to prevent improper connections or faulty programming.

The operator is responsible for qualifying the operating and maintenance personnel and providing them with appropriate safety instructions.

Only staff members with knowledge of industrial process measurement and control, electrical engineering, electronics and the implementation of PES and ESD protective measures may modify or extend the system wiring.

1.3 Writing Conventions

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

Bold	To highlight important parts. Names of buttons, menu functions and tabs that can be clicked and used in the programming tool.
Italics	Parameters and system variables, references.
`Courier`	Literal user inputs.
RUN	Operating states are designated by capitals.
Chapter 1.2.3	Cross-references are hyperlinks even if they are not specially marked. In the electronic document (PDF): When the mouse pointer hovers over a hyperlink, it changes its shape. Click the hyperlink to jump to the corresponding position.

Safety notices and operating tips are specially marked.

1.3.1 Safety Notices

Safety notices must be strictly observed to ensure the lowest possible risk.

The safety notices are represented as described below.

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

The signal words have the following meanings:

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

⚠ SIGNAL WORD

Type and source of risk!

Consequences arising from non-observance.

Risk prevention.

NOTICE

Type and source of damage!

Damage prevention.

1.3.2 Operating Tips

Additional information is structured as presented in the following example:

i The text giving additional information is located here.

Useful tips and tricks appear as follows:

TIP The tip text is located here.

1.4 Safety Lifecycle Services

HIMA provides support throughout all the phases of the plant's safety lifecycle, from planning and engineering through commissioning to maintenance of safety and security.

HIMA's technical support experts are available for providing information and answering questions about our products, functional safety and automation security.

To achieve the qualification required by the safety standards, HIMA offers product or customer-specific seminars at HIMA's training center or on site at the customer's premises. The current seminar program for functional safety, automation security and HIMA products can be found on HIMA's website.

Safety Lifecycle Services:	
Onsite+ / On-Site Engineering	In close cooperation with the customer, HIMA performs changes or extensions on site.
Startup+ / Preventive Maintenance	HIMA is responsible for planning and executing preventive maintenance measures. Maintenance actions are carried out in accordance with the manufacturer's specifications and are documented for the customer.
Lifecycle+ / Lifecycle Management	As part of its lifecycle management processes, HIMA analyzes the current status of all installed systems and develops specific recommendations for maintenance, upgrading and migration.
Hotline+ / 24 h Hotline	HIMA's safety engineers are available by telephone around the clock to help solve problems.
Standby+ / 24 h Call-Out Service	Faults that cannot be resolved over the phone are processed by HIMA's specialists within the time frame specified in the contract.
Logistics+/ 24 h Spare Parts Service	HIMA maintains an inventory of necessary spare parts and guarantees quick, long-term availability.

Contact details:	
Safety Lifecycle Services	https://www.hima.com/en/about-hima/contacts-worldwide/
Technical Support	https://www.hima.com/en/products-services/support/
Seminar Program	https://www.hima.com/en/products-services/seminars//

2 Safety

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

For further information on safety, observe the instructions provided in the HIMax safety manual (HI 801 003 E).

The product is operated with SELV or PELV. The HIMax X-DO 12 01 relay module can switch external voltages of up to 250 VDC/VAC. No imminent risk results from the product itself. Use in the Ex zone is only permitted if additional measures are taken.

2.1 Intended Use

This chapter describes the intended use of the safety-related automation system HIMax.

The automation system is designed for the industrial process market to control and regulate processes, protective systems, burner control applications, machine controllers and process plants, as well as for factory automation plants. SILworX, HIMA's programming tool, is used for programming, configuring, monitoring, operating and documenting the HIMax system.

Redundancy operation of HIMax modules does not preclude simultaneous operation of other non-redundant modules.

2.1.1 Application in Accordance with the De-Energize to Trip Principle

The HIMax system is designed in accordance with the de-energize to trip principle.

A system operating in accordance with the de-energize to trip principle switches off, for instance, an actuator to perform its safety function.

2.1.2 Application in Accordance with the Energize to Trip Principle

The HIMax system can also be used in applications that operate in accordance with the energize to trip principle.

A system operating in accordance with the energize to trip principle switches on, for instance, an actuator to perform its safety function.

When designing the automation system, the requirements specified in the application standards must be taken into account. For instance, line monitoring (SC/OC) for inputs and outputs or message reporting a triggered safety function may be required.

2.1.3 Use in Fire Alarm Systems

The HIMax systems with analog inputs are tested and certified for use in fire alarm systems in accordance with DIN EN 54-2 and NFPA 72.

The conditions of use provided in the safety manual (HI 801 003 E) must be observed.

2.1.4 Explosion Protection

The HIMax automation system is suitable for mounting in zone 2.

The special conditions provided in the HIMax safety manual (HI 801 003 E) must be observed.

2.2 ESD Protective Measures

Only personnel with knowledge of ESD protective measures may work on the HIMax system.

NOTICE

Damage to the HIMax system due to electrostatic discharge!

- **When performing the work, make sure that the workspace is free of static, and wear a grounding strap.**
- **If not used, ensure that the modules are protected from electrostatic discharge, e.g., by storing them in their packaging.**

2.3 Residual Risk

No imminent risk results from a HIMA system itself.

Residual risk may result from:

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

2.4 Safety Precautions

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

2.5 Emergency Information

A HIMA system is a part of the safety equipment of an overall system. If the controller fails, the system enters the safe state.

In case of emergency, no action that may prevent the HIMA system from operating safely is permitted.

3 Product Description

HIMax is a safety-related control system that is intended for continuous operation and maximum availability.

HIMax is a modular system. Functions such as processing, communication or inputs and outputs, are distributed on pluggable modules. These are inserted in one or more base planes. A HIMax controller tailored to the concrete application can be created by selecting appropriate modules. The base plates are connected to one another via patch cables (system bus).

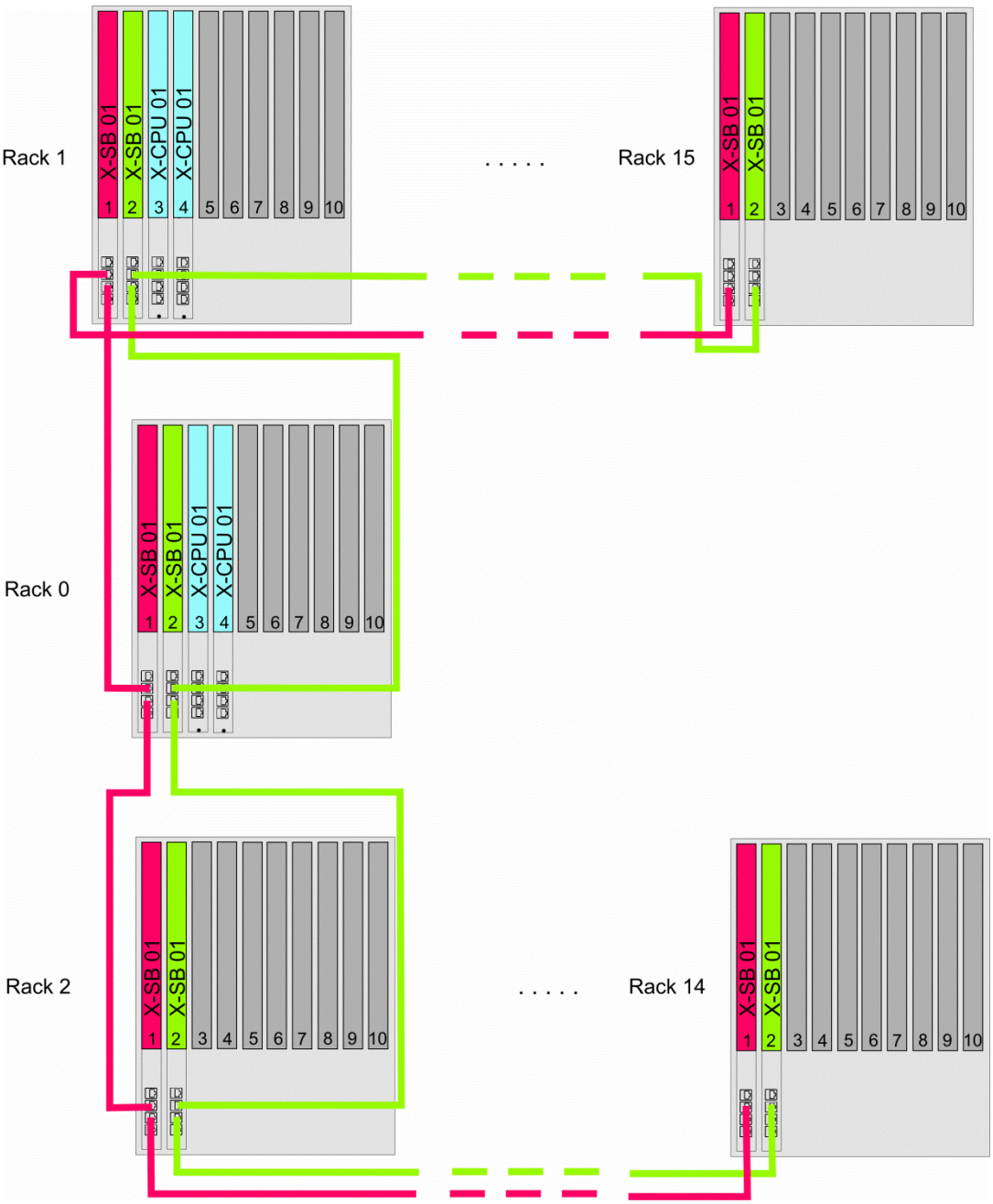
The controller can be easily adapted for future extensions of the process to be controlled, e.g., by adding individual modules or base plates containing modules.

Figure 1 shows the structure of the HIMax system. The figure shows the racks, both system buses, the system bus modules, the processor modules and the connector boards of the modules.

To increase availability, HIMax is intended for redundant operation. For further details, refer to Chapter 3.9.

The system can also be used as mono, non-redundant system. For further details, refer to Chapter 3.3.2, Variant 1, and the appendix.

In either case, if the corresponding module types are used, safety-related operation up to SIL 3 is possible.



Rack 0: Base plate 0.
Rack 1: Base plate 1.
Rack 2: Base plate 2.
Rack 14: Base plate 14.
Rack 15: Base plate 15.

Figure 1: System Overview

A HIMax system is composed of at least one rack, i.e., rack 0. It has rack ID (base plate number) 0 and contains at least one processor module. All other racks are extension racks. Among these, only rack 1 may contain a total of 2 processor modules. The remaining extension racks must not include any processor modules.

Rack 0 may be expanded by up to 15 extension racks. Patch cables (Chapter 3.2) are used to interconnect the two system buses A and B on all the racks.

i In HIMax documents, the terms *base plate* and *rack* are used as follows:
- *Base plate* is used to describe the hardware of an individual base plate.
- *Rack* is used to indicate a base plate within a system.

3.1 Base Plates and Base Plate Types

HIMax base plates differ in the number of slots.

Each base plate used to compose a HIMax controller can have 10, 15 or 18 slots.

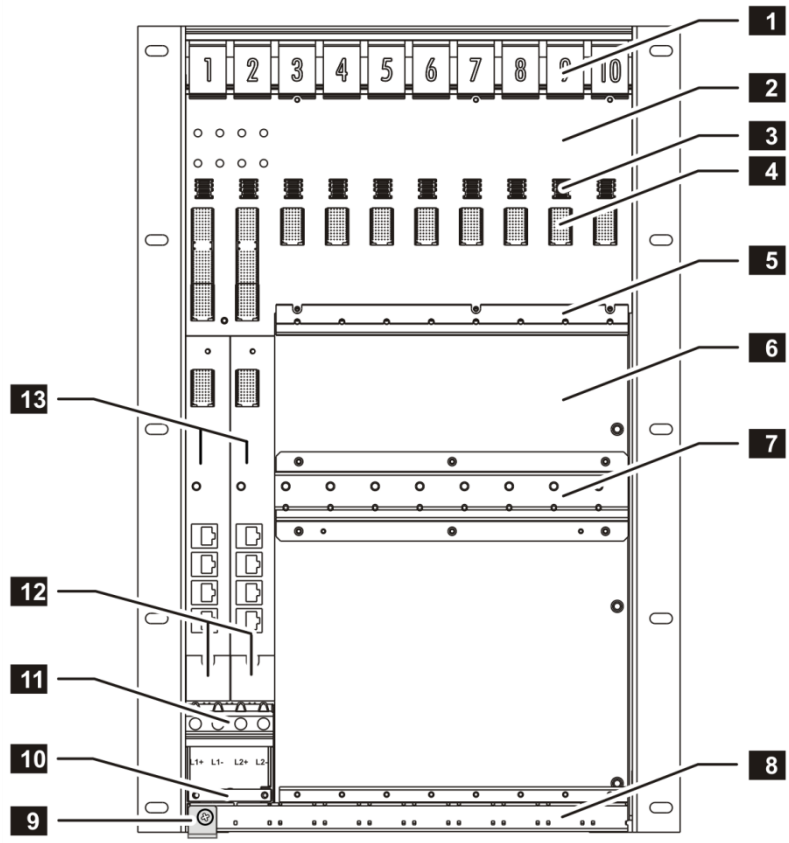
Base plate types:

- With 10 slots: X-BASE PLATE 10 01, X-BASE PLATE 10 31
  for mounting on a backplane, e.g., a mounting plate.
- With 15 slots: X-BASE PLATE 15 01, X-BASE PLATE 15 31
  for mounting on a flat base.
- With 15 slots: X-BASE PLATE 15 02, X-BASE PLATE 15 32
  for 19-inch mounting.
- With 18 slots: X-BASE PLATE 18 01, X-BASE PLATE 18 31
  for mounting on a flat base.

A total of one module and one connector board can be plugged in to each slot.

Patch cables (system bus) are used to interconnect the base plates.

3.1.1 Base Plate Structure



1 Hook-in rail with slot numbers.
2 Backplane bus.
3 24 VDC power supply for the modules.
4 System bus connection.
5 Guiding rail for connector boards.
6 Backplane with wall flanges or 19-Inch mounting flanges.
7 Mounting rail
8 Cable shield rail.
9 Shield connection bracket.
10 Strain relief for supply.
11 Clamp terminal block.
12 Filters for high voltage transients.
13 Connector boards for system bus modules.

Figure 2: Base Plate Structure

Slot 1 and slot 2 are reserved for system bus modules. In X-BASE PLATE 10 31, X-BASE PLATE 15 31 and X-BASE PLATE 18 31, slot 1 and slot 2 are reserved for X-CPU 31 processor modules. The remaining slots can be used for other modules, but observe the restrictions for positioning processor modules, see Chapter 3.3.2.

A connector board specific to each module is available, ensuring connection to external devices such as sensors, actuators and other controllers. Both connector boards for the system bus modules are included within the scope of delivery of the base plate.

The clamp terminal blocks of the base plate are used to connect the power supply. Two redundant 24 VDC power supply units can be connected.

3.1.2 Ventilation

A suitable system fan located above the rack ensures ventilation of the modules.

The air flows from below, through the connection space in front of the connector boards and through the modules to the system fan. To ensure proper ventilation, insert blank modules in all the unused rack slots!

HIMA recommends operating the HIMax system with the corresponding system fan (X-FAN). HIMA system fans ensure the HIMax system's high availability!

A system fan with suitable dimensions is available for each base plate type. Depending on the width, the system fans are equipped with 2, 3 or 4 fans. For further details, refer to the X-FAN manual (HI 801 033 E).

In a control cabinet, the heat lost within the cabinet must be taken into account. If necessary, measures must be taken to reduce the heat loss, e.g., by using roof-mounted fans. Low ambient temperature increases the product life and the reliability of the electronic components within the system, see Chapter 8.1.7.1.

3.1.3 Temperature Monitoring

The modules monitor their own temperature. Use the SILworX programming tool to display the temperature level and evaluate it for programming the corresponding response. For further details, see Chapter 8.1.7.6.

3.1.4 Power Supply

The HIMax system requires a power supply of 24 VDC.

The safe electrically protective separation must be guaranteed within the 24 V system supply. Use power supply units of type PELV or SELV only. When used in accordance with UL regulations, an adjustable power supply unit with a maximum voltage of 150 V and a maximum performance of 10 kVA is allowed.

The power supply units must compensate for voltage dropouts of up to 20 ms. HIMA power supply units are appropriately equipped. Before using power supply units from other manufacturers, ensure their adequate testing.

Two redundant power supply units can be connected.

To ensure high availability, operate the HIMax systems with a power supply complying with the following requirements:

- **If a fault occurs, the maximum overvoltage at the power supply output is 35 V.**
- **Each base plate is fitted with back-up fuses to protect it against currents higher than 63 A.**

The modules monitor both supply voltages. Use the SILworX programming tool to display the voltage level and evaluate it for programming responses.

3.1.4.1 Estimating the Required Power

Use the following rule of thumb to estimate the power required for the power supply.

$$\mathbf{P}_{Total} = \mathbf{n}_{CPU} * 35 + \mathbf{n}_{Modules} * 20 + \mathbf{n}_{Fans} * 20 + P_{External}$$

P_{Total}:	Total required power.
n_{CPU}:	Number of processor modules in use.
$n_{Modules}$:	Number of modules used without processor modules.
n_{Fans}:	Number of fans in use. Each system fan contains 2, 3 or 4 fans.
$P_{External}$:	Power delivered from the output modules to the connected actuators.

The formula includes the following typical power consumptions:

- Power consumption of a HIMax processor module: approx. 35 W.
- Power consumption of additional HIMax modules: approx. 20 W (except for processor module)
- Power consumption of a fan: approx. 20 W
- Power consumption of the actuators connected to and supplied by the output modules.

The calculated value is a rough estimate of the required power of a HIMax system, expressed in watts.

For an exact calculation of the power required, use the power consumption values of the individual modules as specified in the corresponding manuals. The power consumption values of the other consumer loads are specified in the corresponding manuals.

3.2 System Bus

The HIMax system operates with two redundant system buses, system bus A and system bus B.

The system buses run within a base plate on a backplane PCB. The module is connected with the system buses by inserting it into the base plate. System buses A and B interconnect the modules via the system bus modules. The failure of one module does not affect the system bus connections to the remaining modules.

The system bus connections to the modules are galvanically separated from the base plate. An insulation voltage of at least 1500 V is ensured between the processor module and each input and output module.

A system bus module is required to manage a system bus. The system bus module in slot 1 operates system bus A and the system bus module in slot 2 operates system bus B.

i If only one system bus module is inserted in the base plate, only one system bus is available!

If both system bus modules are used to operate the HIMax system, communication runs on both system buses simultaneously.

If the HIMax system is composed of various base plates, use patch cables to interconnect the system buses on the base plates. These cables must be inserted in the RJ-45 socket located on the connector boards of the system bus modules. System bus A and system bus B must not be crossed or connected.

It is not allowed to interconnect the system buses of multiple different HIMax systems!

NOTICE

System malfunction possible!

The UP, DOWN and DIAG system bus ports may only be used to connect to HIMax racks. The system buses cannot be used as normal Ethernet connections.

The UP, DOWN and DIAG ports must not be connected to local networks or other devices with LAN port such as the PADT!

System bus A and system bus B must never be interconnected or crossed!

In line structure, all racks must either be operated redundantly or not redundantly!

Racks with processor modules or system bus modules set to *Responsible* must be structured with a redundant system bus, irrespective of the line or network structure.

The patch cables for the system buses must meet the following requirements:

- At least Cat. 5e (in accordance with IEEE 802.3) for 1 Gbit/s, for industrial applications.
- Industrial RJ-45 connectors on both sides.
- The cable shielding must comply with at least Class D in accordance with ISO/IEC 11801.
- Autocrossover allows the use of both crossover and straight through cables.

Suitable patch cables with industrial connector are available from HIMA in standard lengths.

NOTICE

Communication interference possible!

Use patch cables compliant with industrial standard Cat. 5e or better!

In harsh environments (e.g., subject to temperature changes, electromagnetic interference), low-quality patch cables may cause communication to fail.

The system bus can be organized in two different structures:

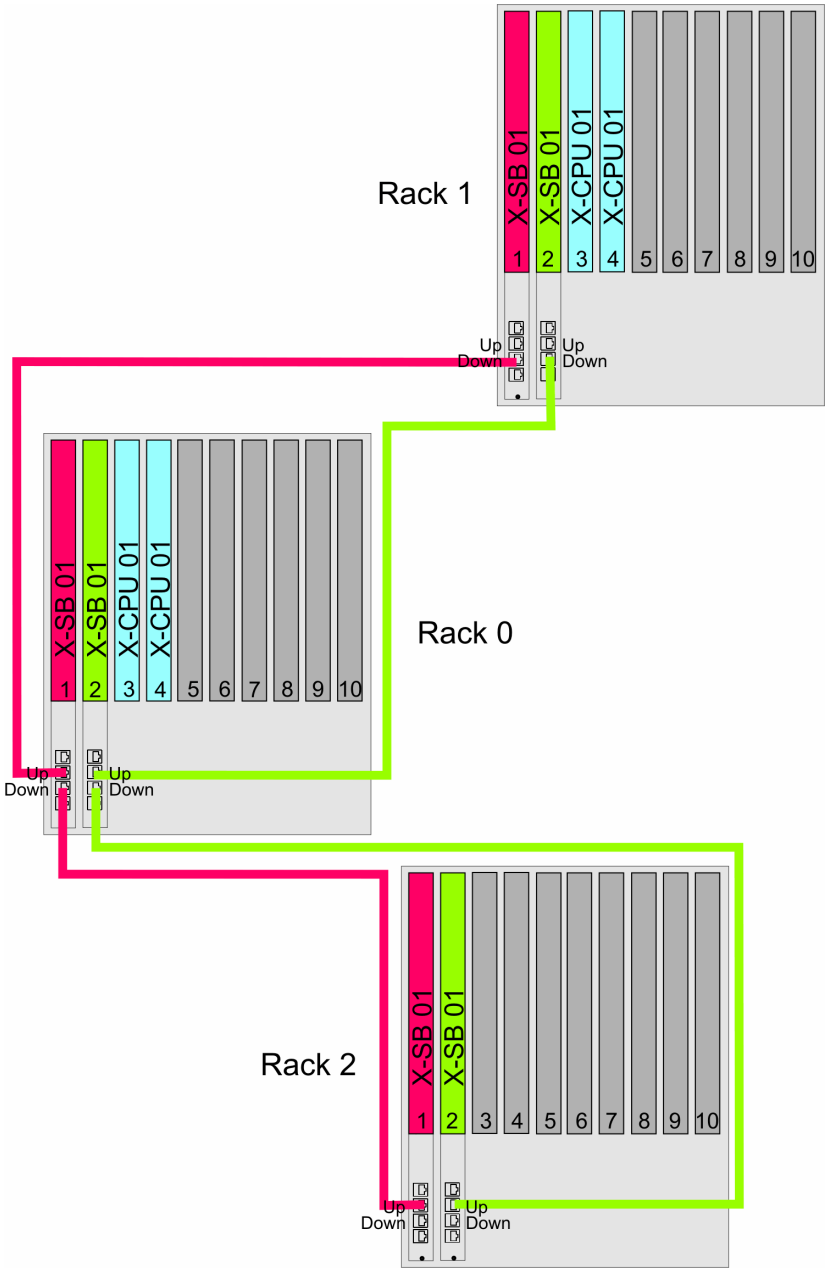
- Line structure

  The line structure is the default structure.
- Network structure

  If the network structure is used, racks can be shut down and replaced during operation, without interrupting the connection to other racks.

3.2.1 System Bus with Line Structure

Two adjacent racks can be connected to one rack.



Rack 0: Base plate 0. Rack 2: Base plate 2.
Rack 1: Base plate 1.

Figure 3: Arrangement of Racks on the System Bus

A rack sequence results from interconnecting the racks – see Figure 3.

- Start with rack ID 0.
- The extension rack connected to the *UP* connector of rack 0 has rack ID 1.
  - All additional racks connected to rack 0 through rack 1 have odd rack IDs up to 15.
- The extension rack connected to the *DOWN* connector of rack 0 has rack ID 2.
  - All additional racks connected to rack 0 through rack 2 have even rack IDs up to 14.

3.2.2 System Bus with Network Structure

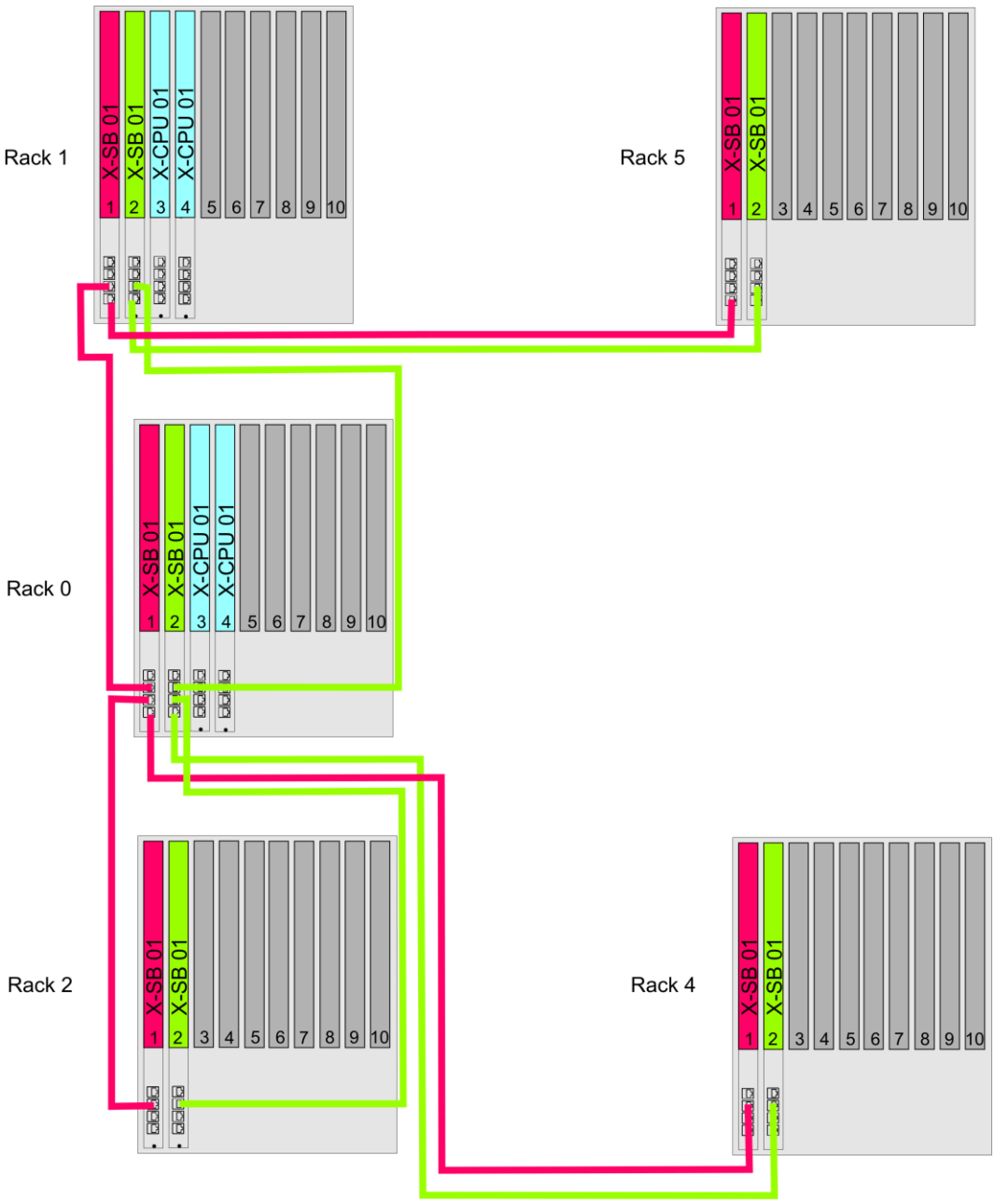
When operating the system bus in a network structure, the UP, DOWN and DIAG ports of the system bus module are of the same value so that any number of structures can be created. Additionally, other network components such as switches can be used. The network components must have the following properties.

- Support of 1 Gbit/s and Ethernet flow control.
- Sufficient storage space so that all messages can be forwarded with no overflow. Discarded messages are considered as system bus errors and displayed in the diagnostic panel of the processor module.

In contrast to the line structure, the rack ID can be almost freely assigned in the network structure. It is, however, required that the (redundant) processor modules are inserted in rack 0 and rack 1. Rack 0 and rack 1 must be directly connected to one another, i.e., using patch cables only or using patch cables and media converters, if both racks are equipped with processor modules or responsible system bus modules. The connection may allow an additional latency of up to 10 µs.

HIMA recommends directly connecting rack 0 and rack 1, even if only rack 0 is equipped with a processor module. Later extension with processor modules in rack 1 is thus possible.

Ethernet ring structures are not allowed for the system bus. The network path of a module to a processor module must always be unique, i.e., data packets may only reach a controller over a single path.



Rack 0: Base plate 0.
Rack 1: Base plate 1.
Rack 2: Base plate 2.
Rack 4: Base plate 4.
Rack 5: Base plate 5.

Figure 4: System Bus with Network Structure

NOTICE

HIMax system malfunctions are possible!

The rack IDs of all the racks directly or indirectly connected to the system bus must be unique! In a network structure, the HIMax system is not always able to recognize ambiguous rack IDs.

Only the interconnection of racks within the same HIMax system is allowed. The racks of different HIMax systems must never be connected to one another via a system bus.

Failure to comply with these instructions may possibly lead to safety problems!

- **Prior to starting safety-related operation, verify that the used rack IDs are unique and properly defined.**
- **The operating company is responsible for this action.**

In a network structure, the system bus module cannot prevent Ethernet rings from occurring.

i A faulty network structure can cause a part of or the entire HIMax system to shut down.

3.2.3 Extending the System Bus, System Bus Latency

The system bus is based on Ethernet technology. For this reason, the system bus can be extended with Ethernet components. The HIMax system can thus stretch across an extensive production line or pipeline's length. All components used in the system must allow a data rate of 1 Gbit/s.

Fiber optic cables are suitable for extending system buses over wider distances.

Wide extensions and a large system structure lead to a message time delay on the system bus, the system bus latency. All delays caused by the physical structure of a system are included in the system bus latency.

The system bus latency is the time delay accumulated by a message on the route between processor module and a module in an I/O rack.

The maximum system bus latency is the maximum delay permitted. A message accumulates this delay over the route to the I/O rack, which presents most of the delaying network components. Delaying components are:

- Rack with system bus modules' switches.
- User-side switches and media converters for fiber optic cables.
- Long patch cables.
- Long fiber optic cables

i Contact HIMA technical support for more details on which switches and media converters have been released by HIMA for extending the system bus!

Use the *Maximum System Bus Latency [µs]* system parameter located in the resource properties to set the maximum system bus latency in the range 100…50 000 µs. When the *Maximum System Bus Latency [µs]* is set to *System Defaults*, the maximum system bus latency is determined by the system. A license is required if the latency is set to a value ≠ *System Defaults*.

With a license, the maximum system bus latency can also be set online.

The HIMax system measures the actual system bus latency during operation and displays it in the SILworX Control Panel.

3.2.3.1 Default Values for the Maximum System Bus Latency

The HIMax system uses the maximum system bus latency default values in the following cases:

- The parameter *Maximum System Bus Latency [µs]* is set to *System Defaults*.
- The project was created with a SILworX version ≤ V4.

For a HIMax system, which is only equipped with HIMax components and using no more than 100 m patch cables for each two-rack connection, the default values of the maximum system bus latency are used in accordance with the *Min* columns of Table 1.

If additional network infrastructure is used, such as fiber optic cables, media converters or switches, the resulting **additional** latency may be max. 50 µs. The *Max* columns of Table 1 show the default values for the maximum system system bus latency, including the additional latency.

Maximum rack distance	Maximum system bus latency in µs				Example: the system consists of the mentioned racks
	X-CPU 01		X-CPU 31		
	Min	Max [1)]	Min	Max [1)]	
0	49.1	-	665.2	-	Only rack 0
1	105.5	155.5	721.6	771.6	Racks 0 and 1
2	161.9	211.9	778.0	828.0	Racks 0, 1, 3
3	218.4	268.4	834.4	884.4	Racks 0, 1, 3, 5
4	274.8	324.8	890.8	940.8	Racks 0, 1, 3, 5, 7
5	331.2	381.2	947.2	997.2	Racks 0, 1, 3, 5, 7, 9
6	387.6	437.6	1003.6	1053.6	Racks 0, 1, 3, 5, 7, 9, 11
7	444.0	494.0	1060.9	1110.9	Racks 0, 1, 3, 5 , 7, 9, 11, 13,
8	500.4	550.4	1116.4	1166.4	Racks 1, 0, 2, 4, 6, 8, 10, 12, 14

1) Maximum system bus latency including the maximum additional latency caused by the network infrastructure

Table 1: Default Values for Maximum System Bus Latency

HIMax uses these default values for the maximum system bus latency, irrespective of whether the line structure or network structure has been configured.

To calculate the maximum rack distance **for more than one** rack, the system assumes the worst case. This means that rack 1, which could contain processor modules, is always taken into consideration, even if it is not configured or does not exist. Therefore, for a system consisting of racks 0, 2, and 4, HIMax assumes a rack distance of 3 racks, not of 2.

3.2.3.2 System Bus Extension for Max. Latency Default Settings

If the maximum system bus latency is set to the default value, and thus with no specific license required, the system bus can already be extended over a long distance using fiber optic cables. The cable length is limited due to the additional signal delay in the fiber optic cable and in the converters between patch cable and fiber optic cable.

If the default latency value is used, HIMax allows the following maximum **additional** delays between modules:

- A maximum of 10 µs between two processor modules.
- A maximum of 50 µs. between a processor module and the furthermost I/O module.

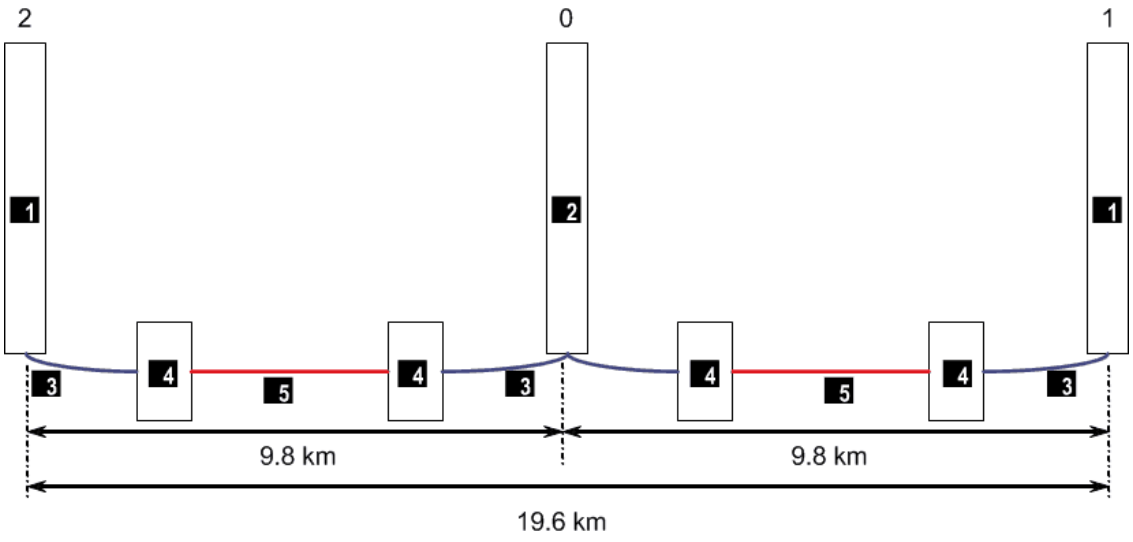
The use of a fiber optic cable causes the following delays:

- Within the fiber optic cable depending on the type, e.g., 5 µs/km.
- 1 µs through the media converters, fiber optic cable and two patch cables.

The delay due to the short patch cables between system bus modules and converters corresponds to the delay of the fiber optic cable. The length of these patch cables is included in the total length.

If all processor modules are arranged directly near one another, either in rack 0 or distributed over rack 0 and rack 1, the two furthermost racks with I/O modules may be located up to 9.8 km from the processor modules.

The HIMax system may have a maximum extension of 19.6 km (Figure 5).



1 Rack with I/O modules.
2 Rack with processor modules.
3 Patch cable.
4 Patch cable <-> fiber optic cable converter.
5 Fiber optic cable.

Figure 5: Maximum Extension for Default Latency Value

The delay time between processor modules in rack 0 and, for instance, the left rack (rack ID 2) with I/O modules is composed of the delay due to the converters (1 µs) and the delay due to the fiber optic cable length (a maximum of 50 µs **-** 1 µs). The following formula applies to the delay due to the fiber optic cable and its length:

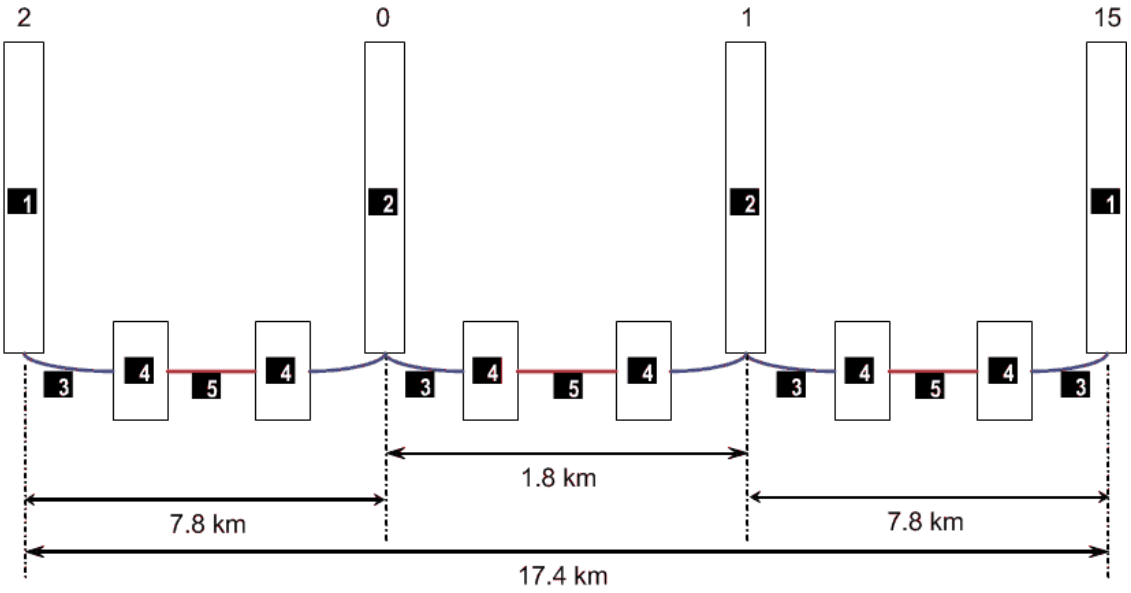
49 µs ≥ length * 5 µs / km, i.e., length **≤** 9800 m

The same formula applies to the length between processor modules in rack 0 and the right rack (rack ID 1) with I/O modules, the maximum length of the fiber optic cable is also 9800 m.

3.2.3.3 Maximum Distance between Processor Modules

If the processor modules are distributed among rack 0 and rack 1, these racks can be positioned at a distance from each other and interconnected using fiber optic cables (Figure 6).

The two racks with processor modules may be located at a distance of up to 1.8 km from each other.
In this case, the HIMax system can have a maximum extension of 17.4 km.



1 Rack with I/O modules.
2 Rack with processor modules.
3 Patch cable.
4 Patch cable <-> fiber optic cable converter.
5 Fiber optic cable.

Figure 6: Maximum Distance between Processor Modules with Latency Default Value

- The delay time between rack 0 and rack 1 is composed of the delay due to both converters (1 µs) and the delay due to the fiber optic cable (a maximum of 10 µs - 1 µs). The following formula applies to the delay due to the fiber optic cable and its length:
  9 µs **≥** length * 5 µs/km, i.e., length **≤** 1800 m
- The delay time between the left rack with I/O modules (rack ID 2) and the rack with processor modules (rack ID 1) is composed of:
  - The delay on the distance between the two racks with processor modules (rack ID 0 and rack ID 1), see Figure 6.
  - The delay on the distance between the left rack (rack ID 2) with I/O modules and the rack with processor modules (rack ID 0). The maximum delay must not exceed 50 µs **-** 10 µs = 40 µs.
    It is composed of the delay due to both converters (1 µs) and the delay due to the fiber optic cable (a maximum of 39 µs). The following formula applies to the delay due to the fiber optic cable and its length:
    39 µs **≥** length * 5 µs/km, i.e., length **≤** 7800 m

The same formula applies to the length of the fiber optic cable between rack 1 and rack 15, the maximum length of the fiber optic cable is also 7800 m.

3.2.3.4 Calculating a User-Specific Maximum System Bus Latency

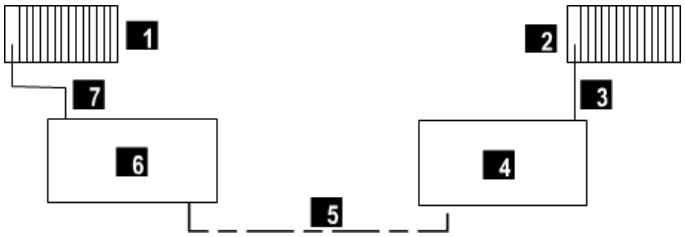
The following factors must be taken into account when calculating the maximum system bus latency:

- The latency of additional network components, if used.
- 65 µs must be added for each rack.

To determine the maximum system bus latency, take all rack connections between racks with processor modules and I/O racks into account.

The largest value, obtained by adding the latencies of all network components between the racks with a processor module and a considered rack, corresponds to the minimum value of the maximum latency.

Figure 7 presents an example of a connection between two racks, rack A and rack B, through a fiber optic cable path.



1 Rack A, connected to switch X.
2 Rack B, connected to switch Y
3 Patch cables between racks B and switch Y.
4 Switch Y.
5 Fiber optic cable.
6 Switch X.
7 Patch cables between racks A and switch X.

Figure 7: Connection of Two Racks through a Fiber Optic Cable

For connecting a fiber optic cable with approved switches, the latency between the connector on the system bus module in rack A and the connector on system bus module in rack B must be calculated in accordance with the following formula:

$$t_{Latency} = t_{Patch}1 + t_{Message} + t_{Switch\ X} + t_{Fiber\ optic\ cable} + t_{Message} + t_{Switch\ Y} + t_{Patch}2 + t_{Message}$$

$t_{Latency}$	Connection latency	
t_{Patch1}	Patch cable latency between racks A and switch X	See below
$t_{Switch\ X}$	Switch X latency	5 µs
$t_{Fiber\ optic\ cable}$	Fiber optic cable latency	See below
$t_{Switch\ Y}$	Switch Y latency	5 µs
t_{Patch2}	Patch cable latency between racks B and switch Y	See below
$t_{Message}$	Runtime for 1 Gbit/s message, considered once for each route	6.592 µs

The patch cable latency 3 and 7, and the fiber optic cable latency 5 must be calculated as follows:

t = Damping * l/c

t	Patch or fiber optic cable latency	$t_{Patch}1$ or $t_{Patch}2$ or $t_{Fiber\ optic\ cable}$
L	Length of patch or fiber optic cable	$l_{Patch}1$ or $l_{Patch}2$ or $l_{Fiber\ optic\ cable}$
c	Light velocity	Approx. 300,000 km/s
Damping	Damping of patch or fiber optic cable	2 (for both, assumed value)

Observe the following points when setting up the system bus:

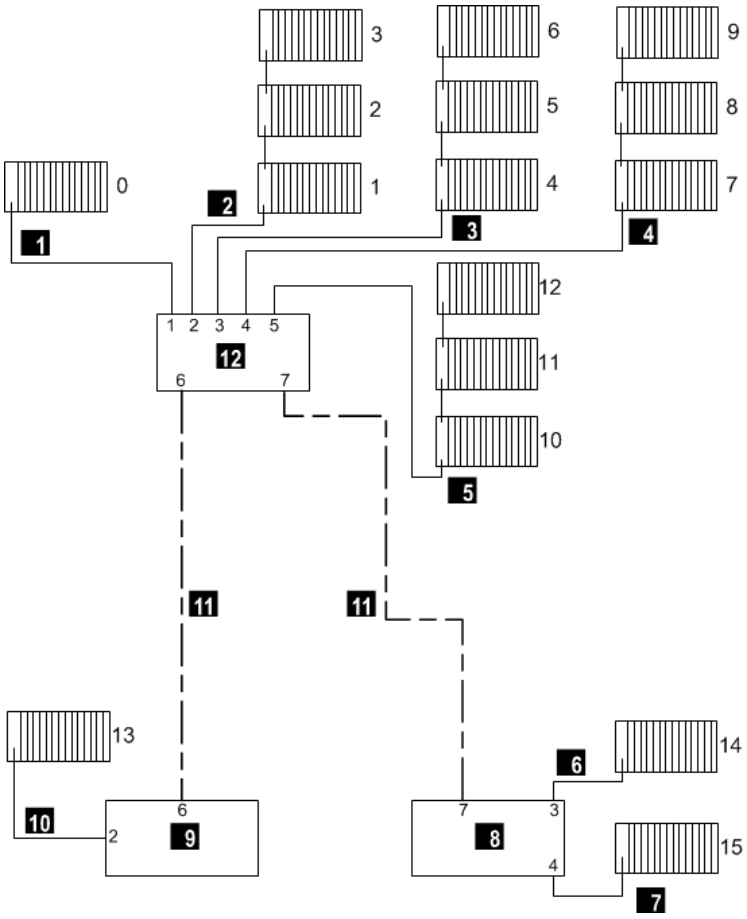
- The maximum latency between processor modules, communication modules and system bus modules, which the PADT can be connected to, is only calculated in accordance with Table 1 based on the distance to the racks with processor modules. The calculated maximum latency is not a function of the value of the *Maximum System Bus Latency [µs]* SILworX system parameter.
- With respect to a standard wiring, i.e., a direct connection via patch cable with a maximum length of 100 m, the maximum latency between the two racks with processor modules or with system bus modules set to *Responsible* may be increased by a maximum of 10 µs.

The network parameters, such as switch latency or damping, are indicated in the specifications or must be determined through a measurement, and then be used in the calculation.

i When designing the network structure and calculating the maximum latency, HIMA recommends consulting a network expert.

3.2.3.5 Example for Calculating a User-Specific Maximum Latency

Figure 8 shows racks that are connected to one another and to the switches with 100 m patch cable each. 10 km fiber optic cables are used to connect the switches to one another. The system is equipped with X-CPU 01 processor modules. Switches Hirschmann SPIDER II Giga 5T/2S EEC Rail Switch were used for the example.



1. Rack 0 connected to port 1 of switch A.
2. Racks 1, 2, 3 connected to port 2 of switch A.
3. Racks 4, 5, 6 connected to port 3 of switch A.
4. Racks 7, 8, 9 connected to port 4 of switch A.
5. Racks 10, 11, 12 connected to port 5 of switch A.
6. Rack 14 connected to port 3 of switch C.
7. Rack 15 connected to port 4 of switch C.
8. Switch C with port numbers.
9. Switch B with port numbers.
10. Rack 13 connected to port 2 of switch C.
11. 10 km fiber optic cable.
12. Switch A with port numbers.

Figure 8: Example for Calculating the System Bus Latency

In this example, the maximum system bus latency is calculated based on the following values:

t_{Switch}	Internal switch latency	5 µs
c	Light velocity	300 000 km/s
$Damping_{Fiber\ optic\ cable}$	Fiber optic cable damping	2 is assumed
$Damping_{Patch}$	Damping of the patch cable	2 is assumed
l_{Patch}	Length of the patch cable, identical for all cables	100 m
$l_{Fiber\ optic\ cable}$	Length of the fiber optic cable, identical for all cables	10 km
$t_{Fiber\ optic\ cable}$	Runtime over 10 km fiber optic cable	$=l_{Fiber\ optic\ cable}*Damping_{Fiber\ optic\ cable}/c$ = 66.7 µs
t_{Patch}	Runtime over 100 m patch cable	$= l_{Patch}*Damping_{Patch}/c$ = 0.667 µs
t_{Rack}	Latency per rack with I/O modules	65 µs
$t_{Message}$	Runtime for 1 Gbit/s message, considered once for each route	6.592 µs

The latency is calculated for the following connections:

- Connection between rack 3 and rack 0. With respect to the type and number of the network components, this corresponds to the connections between rack 6, 9 or 12 and rack 0.
- Connection between rack 15 and rack 0. With respect to the type and number of the network components, this corresponds to the connections between rack 13 or 14 and rack 0.

The connections to the remaining racks have less network components and therefore a lower latency.

Calculation of the latency $t_{Latency}$ between rack 3 and rack 0:

$$t_{Latency} = 4 * t_{Patch} + t_{Switch} + (n_{Racks} - 1) * t_{Rack} + 4 * t_{Message}$$
$$= 4 * 0.667\ µs + 5\ µs + 15 * 65\ µs + 4 * 6.592\ µs$$
$$= 2.667\ µs + 5\ µs + 975\ µs + 26.368\ µs$$
$$= 1009.036\ µs$$

Explanation:

$4 * t_{Patch}$	4 patch cables between racks 3, 2, 1, switch A and rack 0
n_{Racks}	Number of racks, here 16
$(n_{Racks} - 1) * t_{Rack}$	Latency for the following racks: ▪ Rack 3 itself ▪ 11 additional racks (1, 2, 4…12) connected to switch A ▪ Rack 13 connected to switch B ▪ 2 racks (14 and 15) connected to switch C

Calculation of the latency $t_{Latency}$ between rack 15 and rack 0:

$$t_{Latency} = 2 * t_{Patch} + 2 * t_{Switch} + t_{Fiber\ optic\ cable} + (n_{Racks} - 1) * t_{Rack} + 3 * t_{Message}$$
$$= 2 * 0.667\ µs + 2 * 5\ µs + 66.7\ µs + 15 * 65\ µs + 3 * 6.592\ µs$$
$$= 1072.81\ µs$$

Explanation:

2 * t_{Patch}	2 patch cables between racks 8, 7 and switch A and rack 0
2 * t_{Switch}	Delay due to switches A and B
n_{Racks}	Number of racks, here 16
(n_{Racks} - 1) * t_{Rack}	Latency for the following racks: ▪ Rack 15 itself ▪ 12 racks (1…12) connected to switch A ▪ Rack 13 connected to switch B ▪ Rack 14 connected to switch C

In this example, at least 1073 µs must be used for *Maximum System Bus Latency [µs]*.

If this example includes a communication module inserted in rack 8, the following considerations are to be made:

- The maximum latency allowed between processor and communication modules is 274.8 µs in accordance with Table 1.
- The latency between rack 0 and rack 8 is calculated as follows:

  $t_{Latency} = 3*t_{Patch} + t_{Switch} + (n_{Racks} - 1)*t_{Rack} + 3*t_{Message} = 3*0.667\ µs + 5\ µs + 15*65\ µs + 3*6.592\ µs = 1001.776\ µs$

Explanation:

3 * t_{Patch}	3 patch cables between racks 8, 7, switch A and rack 0
t_{Switch}	Delay due to switch A
n_{Racks}	Number of racks, here 16
(n_{Racks} - 1) * t_{Rack}	Latency for the following racks: ▪ 12 racks (1…12) connected to switch A, including rack 8 itself ▪ Rack 13 connected to switch B ▪ 2 racks (14 and 15) connected to switch C

Result: The resulting maximum latency of 1001.776 µs is significantly greater than the maximum allowed latency of 274.8 µs. For this reason, the communication module must not be inserted in rack 8!

3.3 Modules and Connector Boards

The HIMax system is a modular system including various module types. The following module types are available:

- Processor modules for processing the user programs.
- The X-CPU 31 processor module combining the functionality of a processor module with that of a system bus module.
- System bus modules for managing the system buses.
- Input modules for measuring and preprocessing the process values.
- Output modules for converting the results of the user program into control commands for actuators.
- Special I/O modules such as the overspeed trip module (X-MIO 7/6 01) or the HART communication module (X-HART 32 01).
- Communication modules
  - For communicating with external devices or systems operating with standard data transmission protocols (e.g., Modbus, PROFIBUS).
  - Physical interfaces for safe**ethernet** for connecting to additional HIMA controllers.

A protective coating protects the electronic module components against corrosion and dust.

Each module forms a functional unit with the connector board. A connector board establishes the connection between module and field level or ensures communication to other controllers or devices. When replacing a module, the connector board remains in the rack. In this way, the cables or wires connected to the connector board need not be released and reconnected.

Each module type is related to one or several types of connector boards.

The connectors between I/O modules and the corresponding connector boards are mechanically coded. This ensures that a module of a certain type can only be plugged in to the corresponding connector board and prevents them from being equipped with improper modules. Coding is performed with wedges on the female connector located on the connector boards, see also the manuals for I/O modules.

Two types of connector boards exist for I/O modules:

- Connector boards for directly connecting to the supply lines of the field devices.
- Connector boards for directly connecting to field termination assemblies (FTAs).

The FTAs are used to connect the field devices and can be separated from the controller, e.g., in their on marshalling cabinet.

For further details on connector boards and FTAs, refer to the module-specific manuals or the manuals for the FTAs.

3.3.1 Identifying the Modules via SRS

The HIMax system uses the parameters **S**ystem, **R**ack, **S**lot (SRS) to identify the modules.

Designation	Range of values	Description
System	1…65535	Resource identification
Rack	0…15	Rack identification
Slot	1…18	Slot identification

Table 2: Identifying a Module via the SRS

i Every device, e.g., remote I/O that can be reached in a network must be assigned a unique SRS.

3.3.2 Permissible Slot Assignments

The slot assignment is defined as follows:

1. Slot 1 and slot 2 of each rack are reserved for system bus modules. In the base plate of type X-BASE PLATE 1x 31, slot 1 and slot 2 are reserved for X-CPU 31 processor modules. Do not insert any other module into these slots!
2. Processor modules are only allowed in specific slots in accordance with the rules provided in the next section.
3. Once the slots for the processor modules are determined, I/O and communication modules may be inserted into all remaining slots.

3.3.2.1 Slots Permitted for Processor Modules

The following rules apply when assigning the slots to the processor modules, including in the Hardware Editor:

1. A HIMax system may contain a maximum of 4 processor modules of type X-CPU 01 **or** 2 processor modules of type X-CPU 31 on X-BASE PLATE 1x 31.
2. X-CPU 01 processor modules may only be inserted in the following slots:
   - Slots 3 through 6 in rack 0.
   - Slots 3 through 4 in rack 1.
3. Slot 5 in rack 0 and slot 4 in rack 1 may not simultaneously contain processor modules.
4. Slot 6 in rack 0 and slot 3 in rack 1 may not simultaneously contain processor modules.
5. X-CPU 31 processor modules may only be inserted in slot 1 and slot 2 of rack 0.

   In this case, no processor modules may be inserted in other slots, even in rack 1!

NOTICE

System malfunction possible!

Only slots complying with these rules may be used for processor modules.

The table specifies the recommended variants complying with the rules:

Variant	Rack 0 Processor module(s) in slot:	Rack 1 Processor module(s) in slot:	Required system buses
1	3 for mono operation [1)]	---	A
2	3	---	A + B
3	3, 4	---	A + B
4	3, 4, 5	---	A + B
5	3, 4, 5, 6	---	A + B
6	3	3	A + B
7	3, 4	3	A + B
8	3, 4	3, 4	A + B
9	3, 4, 5	3	A + B
10	1 for mono operation (X-CPU 31) [2)]	---	A
11	1, 2 (X-CPU 31)	---	A + B

[1)] Mono operation: The project is configured in SILworX for mono operation and has only one processor module in slot 3, at least one system bus module in slot 1, I/O modules and possibly communication modules. The switch for mono start-up must be set in SILworX. It is always possible (and recommended!) to configure the system bus modules redundantly!

[2)] Mono operation: The project is configured in SILworX for mono operation and has only one processor module X-CPU 31 in slot 1, and I/O modules and possibly communication modules. The switch for mono start-up must be set in SILworX. It is always possible (and recommended!) to configure the X-CPU 31 modules redundantly!

Table 3: Slot Positions Recommended for Processor Modules

HIMA recommends using variant 3 even if variant 1 would be possible. In doing so, the processor module can be replaced without interrupting operation.

If X–CPU 31 are used, variant 11 is to be preferred to variant 10.

Since the operating system is designed to ensure maximum availability, other combinations are possible, but not recommended. HIMax is thus able to offer more flexibility, e.g., when replacing modules or modifying the system. Either way, once the system has been configured, it must have a structure corresponding to one of the recommended variants in Table 3.

If X-CPU 01 processor modules are used, rack 0 must be of type X-BASE PLATE 10 01, X-BASE PLATE 15 01, X-BASE PLATE 15 02 or X-BASE PLATE 18 01.

If X-CPU 31 processor modules are used, rack 0 must be of type X-BASE PLATE 10 31, X-BASE PLATE 15 31, X-BASE PLATE 15 32 or X-BASE PLATE 18 31.

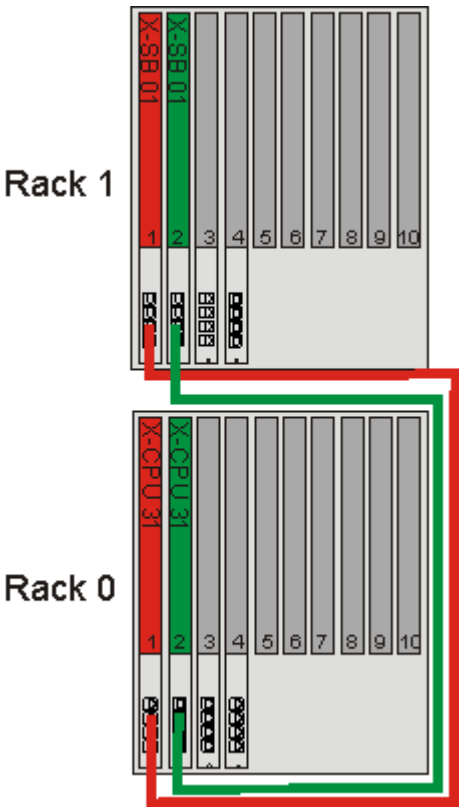


Figure 9: Use of X-CPU 31 Processor Modules

3.3.3 Faults Handling with I/O Modules

If the self-test detects faults, the module behaves as follows:

- If faults occur on individual channels, the module switches off the affected channels and sets their *Channel OK* system variable to FALSE.
- If faults occur on all channels, the module triggers its complete shutdown. The processor module sets the *Module OK* system variable in the faulty module to FALSE.
- Faults detected in I/O modules are signaled via LEDs and by diagnostic messages.

If faults occur, HIMA recommends to check the connected wiring, including the sensors and actuators. If it can be excluded that the fault occurred in the wiring, sensors or actuators, HIMA recommends to replace the module.

For details on the error codes of the individual modules, refer to the corresponding module-specific manual.

3.4 Processor Module

The CPU operating system controls the user programs running in a processor module.

3.4.1 Operating system.

Tasks:

- Controlling the cyclic run of the user programs.
- Performing the self-tests of the module.
- Controlling safety-related communication via safe**ethernet.**
- Managing redundancy together with other processor modules (synchronization).

3.4.1.1 General Cycle Sequence

Phases:

1. Reading of the input data.
2. Processing of the user program.
3. Writing of the output data.
4. Other activities, e.g., reload processing.

3.4.1.2 Operating System States

States that can be recognized by the user:

- LOCKED
- STOP/VALID CONFIGURATION
- STOP/INVALID CONFIGURATION
- STOP/LOADING OS
- RUN
- RUN/UP STOP

Use the LEDs on the module to recognize the operating state. To this end, multiple LEDs must be taken into account. For further details, refer to the module-specific manuals.
SILworX displays the operating states in the online view.

Table 4 provides an overview of the operating system states and indicates the conditions for entering them.

State	Description	The state is entered by...
LOCKED	The processor module is reset to the factory settings, e.g., SRS, network settings.	Connecting the supply voltage to the processor module while the mode switch is set to *Init*.
STOP/VALID CONFIGURATION	Processor module stopped: A valid configuration is available in the memory.	Stopping the processor module using SILworX.
		Applying the supply voltage ▪ Autostart is disabled in the project configuration. ▪ Mode switch is set to *Stop* and the processor module starts by itself.
		From the LOCKED state: Setting the mode switch to *Stop* if one single processor module exists.
		A fault occurred.
STOP/INVALID CONFIGURATION	Processor module stopped: No valid configuration is available in the memory.	Loading with error.
		From the LOCKED state: Setting the mode switch to *Stop* if one single processor module exists.
STOP/LOADING OS	Processor module stopped: The operating system is loaded in the non-volatile memory.	Loading the operating system using SILworX
RUN	The user program is running.	From the STOP/VALID CONFIGURATION state: SILworX command.
		Applying the supply voltage, the following conditions must be met: ▪ A valid project configuration is loaded. ▪ Autostart is enabled in the project configuration. ▪ The mode switch is not set to *Init*. ▪ The mode switch is set to *Run* if the processor module starts by itself.
		From the LOCKED state: Setting the mode switch from *Init* to *Stop* or *Run* if an additional processor module is operating in RUN.
RUN/UP STOP	The user program is not running. This state is used for testing the inputs/outputs and communication.	From the STOP/VALID CONFIGURATION state: SILworX commandSILworX.

Table 4: Operating System States, States Entered

Table 5 specifies how the user may intervene during the corresponding states.

State	Possible user interventions:
LOCKED	▪ Changing the factory settings. ▪ Turning the mode switch to enter the STOP[1)] state. ▪ Turning the mode switch to enter the RUN state. ▪ Using a PADT command to stop (STOP state). ▪ Using a PADT command to start (RUN state).
STOP/VALID CONFIGURATION	▪ Loading the user program. ▪ Starting the user program. ▪ Loading the operating system. ▪ Taking preliminary actions for forcing variables.
STOP/INVALID CONFIGURATION	▪ Loading the user program. ▪ Loading the operating system.
STOP/LOADING OS	None. Once the loading process is completed, the processor module stops (STOP state).
RUN	▪ Stopping the user program. ▪ Forcing variables. ▪ Performing the test.
RUN/UP STOP	▪ Using a PADT command to stop (STOP state).
1) STOP/VALID CONFIGURATION or STOP/INVALID CONFIGURATION, depending on whether the processor module has a valid configuration.	

Table 5: Operating System States, Possible User Interventions

i The cycle time increases by the number of modules used in the system. This applies irrespective of whether or not the modules are included in the configuration.

- **Connecting additional racks with several modules during operation can cause the watchdog time to be exceeded!**

3.4.2 Behavior in the Event of Faults

If faults occur, the processor module enters the error stop state and tries to restart. It performs a complete self-test which can also cause another error stop.

If a fault is still present, the module restarts with reduced functionality to prevent a reboot loop.

Once the processor module has properly run for one minute, the next error stop to occur is considered the first *error stop* attempting a restart.

i Use the PADT for troubleshooting and removing the cause of the fault, e.g., by loading a new application.

3.4.3 Processor Module X-CPU 31

X-CPU 31 combines the functions of a system bus module (X-SB 01) and a processor module (X-CPU 01). For this reason, it can only be inserted into rack 0, slot 1 and slot 2. It is not possible to insert it into different racks or slots.

It is not possible to insert one X-CPU 01 and one X-SB 01 into rack 0 at the same time. X-CPU 01 processor modules in slots 3…6 cannot operate redundantly to X-CPU 31 in slots 1 or 2.

Due to its dual functionality, the X-CPU 31 processor module has considerably lower performance for user programs than the X-CPU 01. It is thus only suitable for being equipped with up to 64 I/O modules.

3.5 Noise Blanking

This chapter describes how noise blanking of I/O modules operates in the HIMax system.

3.5.1 Effects of Noise Blanking

Noise blanking suppresses transient interference to increase the system availability. It ensures that the system triggers a safety-related response to existing interferences within the configured time.

Noise blanking can be activated for each individual I/O module. The default setting is *Activated* for all I/O module types, except for counter modules.

If an interference is blanked out, the system automatically processes the last valid input and output values instead of the currently disturbed values. The time in which noise can be blanked out is limited by the safety time, watchdog time and the cycle time.

The maximum noise blanking time can be calculated using the following equation:

Maximum noise blanking time = safety time - (2 x watchdog time)

The greater the noise blanking time value, the longer the interference can be blanked out. Since an interference can be present for up to one cycle before it is detected while reading in the values, the minimum noise blanking time can be determined by subtracting a cycle from the maximum noise blanking time value.

Minimum noise blanking time = maximum noise blanking time - cycle time

Noise blanking is effective if the cycle time value is less than the noise blanking time.

3.5.2 Configuring Noise Blanking

To blank out as many cycles as possible, the safety time must be set as large as possible taking the process safety time into account. At the same time, the value set for the watchdog time should be as low as possible, but sufficiently large to allow reload and synchronization of an additional processor module. Refer to the safety manual (HI 801 003 E) for further details on the various time parameters and their application.

Configure noise blanking in accordance with the following examples:

Example	1 [1)]	2	3 [2)]
Safety Time [ms]	600	2000	1000
Watchdog Time [ms]	200	500	500
Cycle Time [ms]	100	200	200
Maximum noise blanking time [ms]	200	1000	0
Minimum noise blanking time [ms]	100	800	0

1) Default setting in SILworX.
2) No noise blanking is possible in example 3 since the noise blanking time is less than the cycle time.

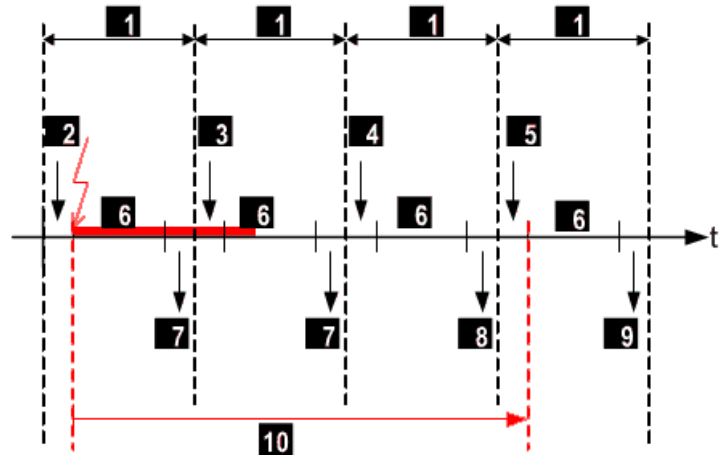
Table 6: Examples of Calculating the Min. and Max. Noise Blanking Time

3.5.3 Noise Blanking Sequence

The examples illustrate the sequence of noise blanking:

- A transient interference is blanked out.
- An interference present for longer than the maximum noise blanking time triggers the safe response.

Example 1: Transient interference is successfully blanked out



1 Cycle, duration = watchdog time.
2 Reading in cycle 1.
3 Reading in cycle 2.
4 Reading in cycle 3.
5 Reading in cycle 4.
6 Processing (in all cycles).
7 Output process in cycle 1 and 2.
8 Output process in cycle 3.
9 Output process in cycle 4.
10 Duration of safety time.

Figure 10: Transient Interference

In example 1, valid input values 2 are read within one cycle. For this cycle, the system processes the valid input values, even though an interference occurred directly upon completion of the read-in process.

If the interference is still present in the following cycle during the read-in process 3, the module detects the interference and the system decides if noise blanking can be performed at this point in time based on the following rule:

Safety time - elapsed time - (2 x watchdog time) > 0

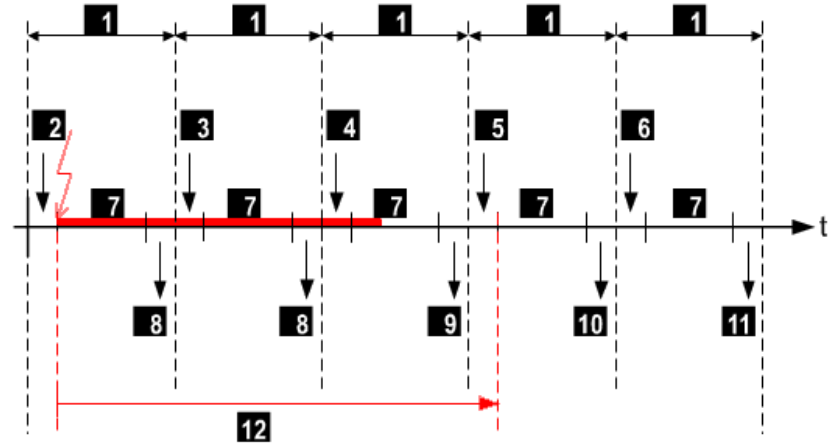
Elapsed time = Time interval between the moment, in which the last valid values were read in, and the moment, in which the interference was detected.

In this example, noise blanking is possible since the interference is present for less than 1 cycle (= elapsed time) and 2 additional cycles (2 x watchdog time) are available for triggering a safe response. For this cycle, the system processes the last valid input values of 2 and no fault response is triggered. The transient interference was successfully blanked out.

If the interference is no longer present in 4, new valid values are read in and processed.

If noise blanking is not active, the system immediately triggers the defined fault response during the read-in process 3.

Example 2: Triggering a safety-related response when interference occurs



1	Cycle, duration = watchdog time	7	Processing (in all cycles)
2	Reading in cycle 1	8	Output process in cycle 1 and 2
3	Reading in cycle 2	9	Output process in cycle 3
4	Reading in cycle 3	10	Output process in cycle 4
5	Reading in cycle 4	11	Output process in cycle 5
6	Reading in cycle 5	12	Duration of safety time

Figure 11: Interference Triggers Safe Response

In example 2, valid input values 2 are read within one cycle. For this cycle, the system processes the valid input values, even though an interference occurred directly upon completion of the read-in process.

If the interference is still present in the following cycle during the read-in process 3, the module detects the interference and the system decides if noise blanking can be performed at this point in time based on the following rule:

Safety time - elapsed time - (2 x watchdog time) > 0

Noise blanking is possible in the 1st and 2nd cycle since the interference is present for less than 1 cycle (= elapsed time) and 2 additional cycles (2 x watchdog time) are available for triggering a safe response. For this cycle, the system processes the last valid input values of 2 and no defined fault response is triggered. The transient interference was successfully blanked out.

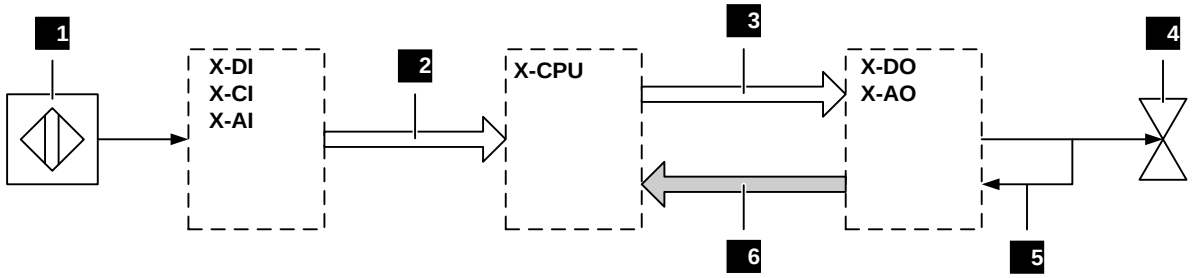
In case of a ratio of safety time/watchdog time = 3/1, as in example 2, 2 cycles are still available for the safe response

If the interference is still present in the next read-in process 4, the fault response must be triggered in that cycle. The fault response must be triggered no later than when the outputs are written to 9. At the next output moment 10, the safety time has expired.

If noise blanking is not active, the system immediately triggers the defined fault response during the read-in process 3.

3.5.4 Noise Blanking Effective Direction

The effective direction must be observed when considering noise blanking and output noise blanking, see Figure 12 and the following chapters.



1 Sensor.
2 Effective direction from the input module to the processor module.
3 Effective direction from the processor module to the output module.
4 Actuator.
5 Output noise blanking.
6 Effective direction from the output module to the processor module.

Figure 12: Effective Direction Associated with Noise Blanking and Output Noise Blanking

3.5.4.1 Effective Direction from the Input Module to the Processor Module (2)

Noise blanking with effective direction from the input module to the processor module is performed by the processor module. Noise blanking suppresses the transient interference on the input module and on the system bus. Noise blanking on the input module can be deactivated in the properties (SILworX) (default = Activated), see the input module manuals. Noise blanking on the system bus is always active and cannot be deactivated in SILworX.

3.5.4.2 Effective Direction from the Processor Module to the Output Module (3)

Noise blanking with effective direction from the processor module to the output module is performed by the output module and is always active. Noise blanking suppresses the transient interference on the system bus.

3.5.4.3 Effective Direction from the Output Module to the Processor Module (6)

Noise blanking with effective direction from the output module to the processor module on the system bus is performed by the processor module. Noise blanking suppresses status acknowledgments of the output module such as SC/OC detection. Noise blanking on the output module can be deactivated in the properties (SILworX) (default = Activated), see the output module manuals.

3.5.4.4 Output Noise Blanking (5)

Output noise blanking is performed by the output module itself. Noise blanking suppresses the switch-off response of a channel to a deviation between the output channel's default and read-back values. Output noise blanking can be activated for each individual output module (Default = Activated), see the output module manuals.

i If the Output Noise Blanking option is activated, the safe response to an existing interference is delayed by up to (safety time – watchdog time).

This interval is a worst case time period that is usually not completely exploited during trouble-free operation of the output module.

3.6 Alarm and Sequence of Events Recording

The HIMax system is able to record alarms and sequences of events (SOE) .

3.6.1 Alarms and Events

Events are changes in the state of a system or controller that are provided with a timestamp.

Alarms are events that signalize increased risk potential.

The HIMA system records the state changes as events specifying the time point when they occurred. The X-OPC Server transfers the events to other systems such as control systems that display or evaluate the events.

HIMax differentiates between Boolean and scalar events.

Boolean events:

- Changes of Boolean variables, e.g., of digital inputs.
- Alarm and normal state can be arbitrarily assigned to the variable states.

Scalar events:

- Crossing the upper and lower limit values defined for a scalar variable.
- Scalar variables have a numeric data type, e.g., INT, REAL.
- Two upper and two lower limits are possible.
- The following condition must be met for the limits:
  Highest limit (HH) ≥ high limit (H) ≥ normal range ≥ low limit (L) ≥ lowest limit (LL).
- A hysteresis can be effective in the following cases:
  - If the value falls below one of the upper limits.
  - If the value exceeds one of the lower limits.

A hysteresis is defined to avoid a needlessly large number of events when a global variable strongly oscillates around a limit.

HIMax can only create events if they are configured in SILworX, see Chapter 5.2.11. The number of definable events is 20 000.

3.6.2 Creating Events

Both the processor module and certain types of I/O modules are able to create events. In the following sections, these I/O modules are referred to as SOE modules.

3.6.2.1 Creating Events on the Processor Module

The processor module uses global variables to create the events and stores them in the buffer, see Chapter 3.6.3. The events are created in the user program cycle.

3.6.2.2 Creating Events on SOE Modules

SOE modules can create events using the input states. The events are created in the SOE module cycle.

The SOE module stores the events in the intermediate buffer that the processor modules use to read them. The intermediate buffer is part of the volatile memory so that the events are lost if the supply voltage is switched off.

Every event that has been read can be overwritten by a new event.

3.6.2.3 System Events

In addition to events, which record changes of global variables or input signals, processor and SOE modules create the following types of system events:

- Overflow: Some events were not stored due to buffer overflow. The timestamp of the overflow event corresponds to that of the event causing the overflow.
- Init: The event buffer was initialized.
- Operating mode Stop: A SOE module changed its operating mode to STOP.
- Operating mode Run: A SOE module changed its operating mode to RUN.
- Establishing communication: Communication between processor module and SOE module has started.
- Losing communication: Communication between processor module and SOE module was terminated.

System events contain the SRS identifier of the module causing the events.

3.6.2.4 Status Variables

Status variables provide the user program with the state of scalar events. Each of the following states is connected to a status variable and can be assigned a global variable of type BOOL:

- Normal.
- Value below low limit (L).
- Value below lowest limit (LL).
- High limit (H) exceeded.
- Highest limit (HH) exceeded.

The assigned status variable becomes TRUE when the corresponding state is achieved.

3.6.3 Recording Events

The processor module records the following events:

- Events created by I/O modules.
- Events created by the processor module itself.

The processor module stores all the events in its buffer. The buffer is part of the non-volatile memory and has a capacity of 5000 events.

The processor module arranges the events from different sources by the time of their arrival and does not sort them by their timestamp.

If the event buffer is full, no new events can be stored as long as no further events are read and thus marked for overwriting.

Refer to Chapter 5.3.7 for more details on forcing and scalar events.

3.6.4 Transfer of Events

The X-OPC Server reads the events from the buffer and transfers them to a third-party system for evaluation and representation purposes. 4 X-OPC Servers can simultaneously read events from a processor module.

3.7 Communication

Communication with other HIMA systems or third-party systems occurs via communication modules. HIMax supports the following communication protocols.

- safe**ethernet** (safety-related)
- HIPRO-S V2 (safety-related) for the connection to HIQuad PES.
- Standard protocols.

safe**ethernet** connections are also possible via the Ethernet interfaces of the processor module.

For details on communication and supported standard protocols, refer to the communication manual (HI 801 101 E) and HIPRO-S V2 manual (HI 800 723 E).

3.7.1 ComUserTask (CUT)

Programs cyclically running on the communication module can be written in C programming language. This allows users to implement their own communication protocols. C programs are not safety-related.

3.7.2 Licensing Protocols

Standard protocols and ComUserTask can only be run in the long term with a valid license. For some protocols, a software activation code is required. For activation, see Chapter 3.9.

i Order the software activation code on time!
After 5000 operating hours, communication continues until the controller is stopped. Afterwards, the user program cannot be started without a valid software activation code for the protocols used in the project (invalid configuration).

3.8 Communication with the Programming and Debugging Tool

A HIMax controller communicates with a PADT via Ethernet. A PADT is a computer that is running the SILworX programming tool.

The computer must be able to reach the controller via Ethernet.

Ethernet to the PADT can be connected to the following HIMax system's interfaces:

- The system bus module's RJ-45 socket labeled PADT.
- A communication module's RJ-45 socket.
- A processor module's RJ-45 socket.

A HIMax system can simultaneously communicate with up to 5 PADTs. If this is the case, only one programming tool can access the controller with write permission. The remaining PADTs can only read information. If they try to establish a writing connection, the controller only allows them a read-only access.

3.9 Licensing

Some HIMax system functions need be licensed:

- Remote rack

  This license unlocks the following functions:
  - Network system structure.
  - Entering a maximum system bus latency ≥ 100 µs.
- Some communication protocols, see the communication manual (HI 801 101 E).

The licenses can be obtained from HIMA for a fee. To activate the function, HIMA provides an activation code which can be entered with the PADT in the configuration. The activation code is bound to the system ID of the PES.

The activation code is generated on the HIMA website at: `www.hima.com/en` under Products & Services -> Product Registration. Refer to the corresponding page for more details.

To activate a function with an activation code

1. Generate the software activation code on the HIMA website `www.hima.com/en` using the system ID of the controller (e.g., 10 000) and the license numbers received from HIMA. To do so, follow the instructions provided on the HIMA website:

i The software activation code is intrinsically bound to this system ID. A license can only be used once for a specific system ID. For this reason, only activate the code when the system ID has been uniquely defined.

2. In SILworX, create a license management for the resource, if not existing.
3. Create a license key in the license management and enter the activation code.
4. Compile the project and load it into the controller.

► The function is activated.

4 Redundancy

The conceptual design of the HIMax system is characterized by high availability. To this end, all system components can be operated redundantly. The following chapter describes redundancy aspects for the various system components.

i A redundant system only increases the system availability, but not its safety integrity level (SIL)!

4.1 Processor Module

A HIMax system can be configured as a mono system with just 1 processor module or as a high-availability system with up to 4 redundant processor modules.

A system with redundant processor modules always requires a redundant system bus.

A digital image of the controller must be created in the Hardware Editor (SILworX) to represent the processor module redundancy. The configuration must be compiled and transferred to the processor modules. Redundant operation can be started once the configuration has been validated in the processor modules.

4.1.1 Reducing Redundancy

A HIMax system with dual to fourfold redundancy of processor modules continues its safety-related operation even if one of the processor modules is no longer available, e.g., because a module failed or was removed. Safety-related operation is still ensured even if multiple processor modules fail.

4.1.2 Increasing Redundancy

If a new processor module is added to a running HIMax system, it automatically synchronizes with the configuration of the existing processor modules. Safety-related operation is ensured.

Requirements:

- The user program run by the processor module is redundantly configured (default setting).
- One slot among 4, 5, 6 on rack 0 or among 3, 4 on rack 1 is still available.
- At least one system bus is operating.
- The mode switch of the processor module that was added is set to *Stop* or *Run*.
- The operating system of the added processor module has either the same version as the existing processor modules or a higher one.

4.1.3 Processor Module X-CPU 31

Only dual redundancy can be implemented in the X-CPU 31 processor module since it can only be used in slot 1 and 2 of rack 0. A X-CPU 01 processor module cannot operate redundantly to an X-CPU 31 processor module.

4.2 Redundancy of I/O Modules

In terms of redundancy, two cases can be distinguished for input and output modules:

- Module redundancy.
- Channel redundancy.

HIMA recommends defining module redundancy before channel redundancy. HIMax supports dual redundancy of input and output modules. Higher redundancy levels are possible using the corresponding programming logic. For redundant connection of a sensor or actuator to several I/O modules, observe the input and output values permitted for I/O modules.

4.2.1 Module Redundancy

2 I/O modules of the same type are defined in the programming system as redundant to one another. They create a redundancy group.

4.2.1.1 Spare Modules

In SILworX, modules that are redundant to one another can have the attribute *Spare Module*. This avoids that an error message is issued if a module fails or is missing.

4.2.2 Channel Redundancy

Requirement: Two modules were configured as redundant to one another.

Channels with the same channel number can be set up as redundant to one another.

In such cases, the programming tool automatically allocates a global variable, which is assigned to a channel (channel number), to both channels of the redundant modules. For further details on the Hardware Editor in SILworX, refer to the online help.

For input channels, users can indicate how the controller should link the signals of both redundant channels to a resulting value. The global variable adopts this value.

Not all channels of two redundant modules need to be redundant.

4.2.3 Connector Boards for Redundant Modules

In several application cases, all channels of two redundant modules are redundant, but the connected transmitters or actuators are not.

Additional wiring effort can be saved by proceeding as follows:

- Use a connector board that is intended for this purpose and occupies two slots.
- Plug the two redundant modules in to adjacent slots.
- Field connections must be created on the connector board once only.

For further details on the connector boards, refer to the module-specific manuals.

4.3 System Bus Redundancy

The HIMax system can be operated with redundant system bus A and system bus B.

Requirements for redundant operation:

- Use of two system bus modules per rack.
- Suitable configuration of the system bus modules.
- Connection of the racks in a controller, see Chapter 3.2.

HIMA recommends using system bus A and system bus B redundantly even if non-redundant operation would be possible, see variant 1 in Chapter 3.3.2.

4.4 Communication Redundancy

For further details, refer to the SILworX online help and communication manual (HI 801 101 E).

4.4.1 safe**ethernet**

Redundancy is configured in the SILworX safe**ethernet** Editor. A communication connection is redundant if two identical physical transmission paths exist.

4.4.2 Standard Protocols

If standard protocols are used, the user program must manage redundancy, except for Modbus slaves.

4.5 Power Supply

The HIMax system can be operated with redundant power supply units. The power supply units are connected to the terminal block, with terminals L1+/L1- used for the first power supply unit and L2+/L2- used for the redundant power supply unit.
Each module supports internal decoupling of the supply voltage from the two terminals.

A redundant supply external to the HIMax system must be provided for connector boards with external supply.

For further details, refer to the corresponding module-specific manual.

4.6 Mono Operation

Mono operation is suitable for applications that do not require redundant operation.

Requirements for mono operation:

- The project contains only one processor module, either type X-CPU 01 in slot 3 or type X-CPU 31 in slot 1.
- Data traffic between the modules runs via system bus A only.
- The project in SILworX is configured accordingly.

The following configurations of rack 0 are appropriate for mono operation:

- One X-CPU 01 processor module in slot 3, one system bus module in slot 1.
- 1 X-CPU 31 processor module in slot 1.

Additionally, the system is equipped with I/O modules and, depending on the project requirements, with communication modules.

Upon successful connection to the controller, the mono start-up switch must be activated. This switch remains active after a power failure. It is reset when a redundant project configuration is loaded into the controller (PES). For further details on how to set the mono start-up switch, refer to Chapter 8.4.1.

HIMA recommends using redundant system buses and system bus modules!

5 Programming

The user programs for the HIMax system must be created using a PADT which is composed of one PC with the programming tool SILworX. A user program is composed of standard function blocks in accordance with IEC 61131-3, of user-defined function blocks and of variables and connectors. The elements are placed in the SILworX FBD editor and graphically interconnected. Based on the resulting graphical representation, SILworX generates an executable program that can be loaded into the controller.

For further details on the programming tool, refer to the SILworX online help.

Up to 32 user programs can be loaded into the controller. The controller processes the user programs simultaneously. The user programs can be processed with tunable priorities.

5.1 Connection of the Programming Tool

The PC with the SILworX programming tool (PADT) is connected to the HIMax system via Ethernet. The following interfaces are available:

- The PADT Ethernet interface of the system bus modules. Only patch cables may be used in connection with these interfaces.
- Ethernet interfaces of the communication modules (X-COM).
- Ethernet interfaces of the processor modules (X-CPU).

5.1.1 Use of Ethernet Interfaces

Depending on the system structure, the Ethernet interfaces must be used in the following order for reasons of automation security:

1. **PADT** interface to an X-SB, which is not set to *Responsible*.
2. Ethernet interface to an X-COM.
3. Ethernet interface to an X-CPU.
4. **PADT** interface to an X-SB, which is set to *Responsible*.

5.2 Using Variables in a Project

A variable is a placeholder for a value within the program logic. The variable name is used to symbolically address the storage space containing the stored value.

Two essential advantages result from using symbolic names instead of physical addresses:

- The names of inputs and outputs used in the process can also be used in the user program.
- The modification of how the variables are assigned to the input and output channels does not affect the user program.

Local and global variables exist. Local variables are valid in a delimited project area, in a user program or function block. Global variables can be used in several function blocks or user programs and can exchange data between the function blocks.

Global variables can be created at different project tree levels. The global variables apply to all subordinate levels within the scope.

Example: If a project contains several resources, the global variables created under a resource are only valid for the branches subordinated to that resource.

Hierarchy of the levels at which global variables can be defined.

1. Project.
2. Configuration.
3. Resource.

Values may only be written to global data in one program location! The possible sources are:

- Logic in a user program.
- Inputs.
- System variables.
- Communication protocols.

Writing to global variables at multiple positions within the program can result in unintended effects!

In the Global Variable Editor, check the usage of global data with the *Cross-Reference in Column* function.

5.2.1 Variable Types

SILworX supports the following variable types:

- VAR, a variable within a logic (read and write).
- VAR with the CONST attribute, a variable that was defined as constant and cannot be modified.
- VAR with the RETAIN attribute, a variable that retains its value after a power outage.
- VAR_EXTERNAL, reference to global variables (read and write).
- VAR_GLOBAL, global variable (read and write) to exchange values between programs and subordinated functions and function blocks.
- VAR_INPUT, input variable (read) of a POU. It is also displayed in the interface viewer.
- VAR_OUTPUT, output variable (write) of a POU. It is also displayed in the interface viewer.
- VAR_TEMP, temporary variable (read and write).
- VAR_ACTION, action declaration (read and write).

Which type can be assigned to a variable depends on the hierarchy of the variables in the structure tree. The following table shows the permissible variable types in connection with the structure tree nodes.

Type	Project	Configuration	Resource	Program type	Function block type	Function type
VAR				X		
VAR_EXTERNAL				X		
VAR_GLOBAL	X	X	X	(1)		
VAR_INPUT				(1)	X	X
VAR_OUTPUT				(1)	X	X
VAR_TEMP				(2)	X	
VAR_ACTION				(2)	X	

(1) Contrary to the standard, this function is not supported.
(2) Contrary to the standard, VAR_ACTION is supported.

Table 7: Supported Variable Types

5.2.2 Initial Value

An initial value can be allocated to any variable. The variable adopts this value if no other value was assigned by the program:

- While starting the program.
- If a fault occurs in one of the following sources from which the variable derived its value, for example:
  - Physical input.
  - Communication interface.
  - User program in the STOP state.

Initial values are specified as follows:

- The initial value that is in accordance with the data type declaration is used. For elementary data types, the initial values in accordance with IEC 61131-3, Table *Ranges of Values and Initial Values for the Data Types* apply. This initial value has the lowest priority.
- The initial value specified by the user for a derived data type, applies. The initial value of the basis type is no longer inherited.
- The initial value explicitly specified by the user for a variable in the variable declaration, applies.
- The instance-specific initial value that was already defined at the time of initialization, applies. This initial value has the highest priority.

The value that the connected variable should adopt can be set for safe**ethernet** and communication protocols.

i In safety-related applications, a safe value must be assigned as initial value to all variables that receive their value from a physical input or from the communication.

5.2.3 System Variables and System Parameters

System variables are pre-defined variables for processing properties or states of the HIMax system in the user program. To define them, they are assigned global variables used in the user program.

The system parameters are used to configure properties of the controller (only possible in SILworX). System parameters that can only have the values TRUE and FALSE are also referred to as switches.

System variables and system parameters are defined at different project levels. The system variables and parameters are configured in SILworX, either in the Properties dialog box of the corresponding structure tree node or in the detail view of the Hardware Editor.

Project level	Description of the system variables and parameters
Resource	See Table 9.
Hardware, in general	▪ System variables for configuring the controller, see Table 9. ▪ System variables providing information, see Table 12 and Table 13.
Hardware: Modules	Refer to the manual of the corresponding module type. The system variables and system parameters are configured in the module's detail view of the Hardware Editor.
User Program	See Chapter 5.2.6.

Table 8: System Variables at Different Project Levels

5.2.4 Resource System Parameters

The system parameters of the resource determine how the controller will behave during operation. The system parameters can be set in SILworX, in the *Properties* dialog box of the resource.

System parameter	S [1)]	Description	Setting for safe operation
Name	N	Name of the resource.	Any
System ID [SRS]	Y	System ID of the resource. Range of values: 1…65535 Default value: 60 000 The value assigned to the system ID must differ from the default value, otherwise the project is not able to run!	Unique value within the controller network. This network includes all controllers that can potentially be interconnected.
Safety Time [ms]	Y	For details on the safety time of the resource (in milliseconds), refer to the safety manual (HI 801 003 E). Range of values: 20…22500 ms Default value: 600 ms (can be changed online)	Application-specific
Watchdog Time [ms]	Y	For details on the watchdog time of the resource (in milliseconds), refer to the safety manual (HI 801 003 E). Range of values: 6...7500 ms Default value: 200 ms (can be changed online)	Application-specific
Target Cycle Time [ms]	N	Target or maximum cycle time, see *Target Cycle Time Mode*. Range of values: 0...7500 ms Default value: 0 ms (can be changed online) The maximum target cycle time value may not exceed the configured *Watchdog Time [ms]* minus the minimum value that can be set for *Watchdog Time [ms]* (6 ms, see above); otherwise the entry is rejected. If the default value is set to 0 ms, the target cycle time is not taken into account. For further details, refer to the following chapters.	Application-specific
Target Cycle Time Mode	N	For details on the use of the *Target Cycle Time [ms],* see the following chapters. The default setting is *Fixed-tolerant* (can only be changed online).	Application-specific
Multitasking Mode	N	Mode 1 The duration of a CPU cycle is based on the required execution time for all user programs. Mode 2 The processor provides the execution time portion not needed by lower priority user programs to higher priority user programs. Operation mode for high availability. Mode 3 The processor waits until the execution time not needed by the user programs has expired, thus increasing the cycle. Default value: Mode 1	Application-specific
Max. Com.Time Slice [ms]	N	Highest value in ms for the time slice used for communication during a resource cycle, see the communication manual (HI 801 101 E). Range of values: 2…5000 ms Default value: 60 ms	---

System parameter	S [1)]	Description		Setting for safe operation
Optimized Use of Com. Time Slice	N	The system parameter reduces the response times for communications via processor module(s). i This can affect the temporal utilization of *Max.Com. Time Slice ASYNC [ms]* and the system parameter *Max. Duration of Configuration Connections [ms]* such that these two times can be subject to more demands (e.g., during reload).		---
Max. Duration of Configuration Connections [ms]	N	This defines how much time within a CPU cycle is available for configuration connections. Range of values: 2...3500 ms Default value: 20 ms For further details, refer to the following chapters.		Application-specific
Maximum System Bus Latency [µs]	N	Maximum delay of a message between an I/O module and a processor module. 100...50 000 µs, Default value: *System Defaults* i A license is required for setting the maximum system bus latency to a value ≠ *System Defaults*.		Application-specific
Allow Online Settings	Y	TRUE:	**All** the switches/parameters listed under FALSE can be changed online using the PADT. This is only valid if the system variable *Read-only in RUN* has the value FALSE. Default value: TRUE.	HIMA recommends using the FALSE setting.
		FALSE:	The following parameters **cannot** be changed online: ▪ *System ID* ▪ *Autostart* ▪ *Global Forcing Allowed* ▪ *Global MultiForcing Allowed* ▪ *Global Force Timeout Reaction* ▪ *Load Allowed* ▪ *Reload Allowed* ▪ *Start Allowed* The following parameters can be changed online if *Reload Allowed* is TRUE. ▪ *Watchdog Time (for the resource)* ▪ *Safety Time* ▪ *Target Cycle Time* ▪ *Target Cycle Time Mode*	
		Allow Online Settings can only be TRUE when the controller is stopped or by performing a reload.		

System parameter	S [1)]	Description		Setting for safe operation
Autostart	Y	TRUE:	If the processor module is connected to the supply voltage, the user programs start automatically. Default value: TRUE.	Application-specific
		FALSE:	The user program does not start automatically after connecting the supply voltage.	
		Observe the settings in the resource program properties!		
Start Allowed	Y	TRUE:	Cold start or warm start permitted with the PADT in RUN or STOP. Default value: TRUE.	Application-specific
		FALSE:	Start not allowed.	
Load Allowed	Y	TRUE:	Configuration download is allowed. Default value: TRUE.	Application-specific
		FALSE:	Start not allowed.	
Reload Allowed	Y	TRUE:	Configuration reload is allowed. Default value: TRUE.	Application-specific
		FALSE:	Configuration reload is not allowed. A running reload process is not aborted when switching to FALSE.	
Global Forcing Allowed	Y	TRUE:	Global forcing is permitted for this resource. Default value: TRUE.	Application-specific
		FALSE:	Global forcing is not permitted for this resource.	
Global Force Timeout Reaction	N	Specifies how the resource should behave when the global force timeout has expired: ▪ Stop Forcing Only. ▪ Stop Forcing and Stop Resource. Default value: Stop Forcing Only.		Application-specific
Global MultiForcing Allowed	Y	TRUE:	Users with MultiForcing access can write force data (force values and individual force switches) for global variables in a resource if the required higher-order conditions have been met and the force permissions have been granted.	Application-specific
		FALSE:	Users with MultiForcing access cannot force global variables. Default value: FALSE (can be changed online)	

System parameter	S [1)]	Description		Setting for safe operation
Minimum Configuration Version	N	With this setting, it is possible to generate code that is compatible with previous or newer HIMax operating system versions in accordance with the project requirements. Default value: SILworX V11 for new projects.		Application-specific
		SILworX V2	The code is generated like in SILworX V2 for HIMax prior to V3.	
		SILworX V3	The code is generated like in SILworX V3 for HIMax V3.	
		SILworX V4	The code is generated like in SILworX V4 for HIMax V4.	
		SILworX V5	The code is generated like in SILworX V5 for HIMax V5.	
		SILworX V6	The code is generated like in SILworX V6.48 for HIMax V6.	
		SILworX V6b	The code is generated like in SILworX V6.114 for HIMax V6.	
		SILworX V7	The code is generated like in SILworX V7 for HIMax V7.	
		SILworX V8	The code is generated like in SILworX V8 for HIMax V8.	
		SILworX V9	The code is generated like in SILworX V9 for HIMax V9.	
		SILworX V10	The code is generated like in SILworX V10 for HIMax V10.	
		SILworX V11	The code is generated like in SILworX V11 for HIMax V11.	
Fast Start-Up	N	Not applicable to HIMax.		---
[1)] The operating system handles the system parameter in a safety-related manner, yes (Y) or no (N).				

Table 9: Resource System Parameters

5.2.4.1 Notices on the *Minimum Configuration Version* Parameter

The following notices apply to the *Minimum Configuration Version p*arameter:

- In a new project, the latest *Minimum Configuration Version* is selected. Verify that this setting is in accordance with the operating system version in use.
- In a project converted from a previous SILworX version, the value for *Minimum Configuration Version* remains the value set in the previous version. This ensures that the configuration CRC resulting from the code generation is the same as in the previous version and the generated configuration is compatible with the operating systems of the modules.
  For this reason, the value of Minimum Configuration Version should only be changed in connection with other changes to the affected resource in converted projects.
- If features only available in higher configuration versions are used in the project, SILworX automatically generates a configuration version higher than the preset *Minimum Configuration Version*. This is indicated by SILworX at the end of the code generation. The modules reject loading higher configuration versions that do not match their operating system.
  To remove such incompatibilities, it can be helpful to compare the information provided by the version comparison with the overview of the module data.
- If X-CPU 31 processor modules are used, *Minimum Configuration Version* must be set to SILworX V6 or higher.

5.2.4.2 Use of the Parameters *Target Cycle Time* and *Target Cycle Time Mode*

Using the settings for the *Target Cycle Time Mode* system parameter, the cycle time can be maintained as constant as possible at the value of *Target Cycle Time [ms]*. To do this, the system parameter must be set to a value > 0.

In doing so, HIMax limits reload and synchronization on the redundant modules to ensure that the target cycle time is maintained.

The following table describes the settings for the *Target Cycle Time Mode* system parameter.

Setting	Description
Fixed	If a CPU cycle is shorter than the defined target cycle time, the CPU cycle is extended to the target cycle time. If the CPU cycle takes longer than the target cycle time, the CPU resumes the cycle without delay. i A reload or synchronization process is rejected if the reserve time is not sufficient (target cycle time minus actual cycle time).
Fixed-tolerant	Similar to *Fixed*, but with the following differences: 1. To ensure that the synchronization process can be performed successfully, the target cycle time may be violated for a CPU cycle. 2. To ensure that the reload can be performed successfully, the target cycle time may be violated for 1 to n CPU cycles (where n is the number of changed user programs). The default setting is *Fixed-tolerant*! i After the first reload activation cycle, the values of watchdog time, target cycle time and target cycle time mode apply in accordance with the new configuration. A maximum of every fifth cycle can be extended during the reload. One single cycle may be extended during synchronization.
Dynamic	The CPU processes each CPU cycle as fast as possible. This corresponds to a target cycle time of 0 ms. i A reload or synchronization process is rejected if the reserve time is not sufficient (target cycle time minus actual cycle time). A maximum of every fifth cycle can be extended during the reload. One single cycle may be extended during synchronization.
Dynamic-tolerant	Similar to *Dynamic*, but with the following differences: 1. If necessary, the target cycle time is automatically increased for one CPU cycle to ensure that the synchronization process can be performed successfully. 2. To ensure that the reload can be performed successfully, the target cycle time may be automatically increased for 1 to n CPU cycles (where n is the number of changed user programs). i After the first reload activation cycle, the values of watchdog time, target cycle time and target cycle time mode apply in accordance with the new configuration. A reload or synchronization process is rejected if the reserve time is not sufficient (target cycle time minus actual cycle time).

Table 10: Settings for Target Cycle Time Mode

5.2.4.3 Maximum Communication Time Slice

The maximum communication time slice is the time period in milliseconds (ms) per CPU cycle assigned to the processor module for processing the communication tasks. Even if the protocol processing could not be completed within one communication time slice, the processor module still executes the safety-relevant monitoring for all the protocols within one CPU cycle.

i If not all upcoming communication tasks can be processed within one CPU cycle, the whole communication data is transferred over multiple CPU cycles. The number of communication time slices is then greater than 1.

For calculating the maximum response time, the number of communication time slices must be equal to 1.

5.2.4.4 Determining the Maximum Duration of the Communication Time Slice

For a first estimate of the maximum duration of the communication time slice, the sum of the following times must be entered in the *Max. Com. Time Slice [ms]* system parameter located in the properties of the resource.

- For each communication module (X-COM 01): 3 ms.
- For each redundant safe**ethernet** connection: 1 ms.
- For non-redundant safe**ethernet** connection: 0.5 ms.
- For each kilobyte user data of non-safety-related protocols, e.g., Modbus: 1 ms.

HIMA recommends comparing the value estimated for *Max. Com. Time Slice [ms]* with the value displayed in the Control Panel and, if necessary, correcting it in the properties of the resource. This can be done during an FAT (factory acceptance test) or SAT (site acceptance test).

To determine the actual duration of the maximum communication time slice

1. Operate the HIMax system under full load (FAT, SAT):
   All communication protocols are in operation (safe**ethernet** and standard protocols).
2. Open the **Control Panel** and select the **Com. Time Slice** structure tree folder.
3. Read the vale displayed for *Maximum Com. Time Slice Duration per Cycle [ms].*
4. Read the value displayed for *Maximum Number of Required Com. Time Slice Cycles*.

The duration of the communication time slice must be set so that, when using the communication time slice, the CPU cycle cannot exceed the watchdog time specified by the process.

5.2.4.5 Calculating the *Maximum Duration of Configuration Connections [ms]* T_{Config}

The *Max. Duration of Configuration Connections [ms]* system parameter corresponds to the time budget ($\mathbf{t}_{Config}$) required for the system-internal communication connections (tasks):

- PADT online connections (e.g., download/reload, OS update, online test, diagnostics).
- Remote I/O status connections (start, stop and diagnostics).
- Configuration of modules (e.g., loading of replaced modules).

If these tasks cannot be completed within one CPU cycle, the remaining tasks are processed in the next CPU cycle. This can cause unexpected delays for these tasks.

i HIMA recommends dimensioning $\mathbf{t}_{Config}$ in such a way that all tasks can be processed in a single CPU cycle.

$\mathbf{t}_{Config}$ for HIMax CPU operating systems ≤ V3 is fixed and set by SILworX to 6 ms. The time required to process the mentioned tasks may, however, exceed the default value in a CPU cycle.

$\mathbf{t}_{Config}$ for HIMax CPU operating systems ≥ V4 is calculated as follows:

X-CPU 01: $\mathbf{t}_{Config} = (\mathbf{n}_{Com} + \mathbf{n}_{PADT} + \mathbf{n}_{RIO}) * 0.25\ ms + 4\ ms + 4 * (\mathbf{t}_{Latency} * 2 + 0.31\ ms)$

X-CPU 31: $\mathbf{t}_{Config} = (\mathbf{n}_{Com} + \mathbf{n}_{PADT}) * 1\ ms + \mathbf{n}_{RIO} * 0.25\ ms + 4\ ms + 4 * (\mathbf{t}_{Latency} * 2 + 0.8\ ms)$

t_{Config}:	System parameter *Max. Duration of Configuration Connections [ms].*
n_{COM}:	Number of modules with Ethernet interfaces (X-SB, X-CPU, X-COM).
n_{PADT}:	5, maximum number of PADT connections.
n_{RIO}:	Number of configured remote I/Os.
$t_{Latency}$:	Use the active maximum system bus latency, see the following descriptions. If the value of the maximum system bus latency is expressed in µs, it must be divided by 1000 before the calculation to obtain the value in ms.

Depending on which system bus structure was selected for the HIMax system, the following value must be used for the system bus latency:

Network structure:	If 100…50 µs was manually entered for *Maximum System Bus Latency [µs]*, then this value must be used in the equation as $\mathbf{t}$*Latency*.
Line structure:	If *Maximum System Bus Latency [µs]* is set to System Defaults, the standard value of the maximum system bus latency specified for $\mathbf{t}$*Latency* in the following table should be used in the equation. As an alternative to the value indicated in the table, the maximum value can first be used: 550.4 µs for X-CPU 01 and 1166.4 µs for X-CPU 31.

When generating the code or converting the project, a warning message is displayed in the PADT logbook if the value defined for $\mathbf{t}_{Config}$ is less than the value resulting from the previous equation.

i Setting the value for $\mathbf{t}_{Config}$ too low can significantly impair the performance of PADT online connections (tasks) and cause the connection to remote I/Os to be aborted.

HIMA recommends comparing the value calculated for $\mathbf{t}_{Config}$ with the value displayed in the Control Panel and, if necessary, correcting it in the properties of the resource. This can be done during a SAT (site acceptance test).

For test purposes, $\mathbf{t}_{Config}$ can also be set online in the Control Panel.

The value set for $\mathbf{t}_{Config}$ must be taken into account for dimensioning the required watchdog time. For details, refer to the section on safety-relevant time parameters.

Maximum rack distance	Maximum system bus latency in µs				Example: the system consists of the mentioned racks
	X-CPU 01		X-CPU 31		
	Min	Max 1)	Min	Max 1)	
0	49.1	-	665.2	-	Only rack 0
1	105.5	155.5	721.6	771.6	Racks 0 and 1
2	161.9	211.9	778.0	828.0	Racks 0, 1, 3
3	218.4	268.4	834.4	884.4	Racks 0, 1, 3, 5
4	274.8	324.8	890.8	940.8	Racks 0, 1, 3, 5, 7
5	331.2	381.2	947.2	997.2	Racks 0, 1, 3, 5, 7, 9
6	387.6	437.6	1003.6	1053.6	Racks 0, 1, 3, 5, 7, 9, 11
7	444.0	494.0	1060.9	1110.9	Racks 0, 1, 3, 5 , 7, 9, 11, 13,
8	500.4	550.4	1116.4	1166.4	Racks 1, 0, 2, 4, 6, 8, 10, 12, 14
1) Maximum system bus latency including the maximum additional latency caused by the network infrastructure					

Table 11: Default Values for Maximum System Bus Latency

5.2.5 Rack System Variables

These system variables are used to change the behavior of the controller while it is operating in specific states. These variables can be set in the *System* tab located in the rack detail view of the SILworX Hardware Editor.

System variables	S [1)]	Function	Setting for safe operation
Force Deactivation	Y	Prevents the forcing process from starting and terminates a running forcing process. Default setting: FALSE.	Application-specific
Spare 0…Spare 16	Y	No function!	---
MultiForcing Denied	Y	MultiForcing can be enabled and disabled using the *MultiForcing Denied* system variable so that the associated functions can be controlled by the user program. For MultiForcing, the system variable must be set to FALSE. Default setting: FALSE.	Application-specific
Emergency Stop 1...Emergency Stop 4	Y	Shuts down the controller if faults are detected by the user program. Default setting: FALSE.	Application-specific
Read-only in RUN	Y	After the controller is started, the access permissions are downgraded to *Read-Only*. Exceptions are forcing and reload. Default setting: FALSE.	Application-specific
Reload Deactivation	Y	Locks the execution of reload. Default setting: FALSE.	Application-specific
1) The operating system handles the system parameter in a safety-related manner, yes (Y) or no (N).			

Table 12: System Variables of Racks

Global variables can be connected to these system variables; the value of the global variables is modified using a physical input or the user program logic.

5.2.5.1 Input variables for reading out parameters.

The input variables can be read out and assigned to a global variable in the *System* tab located in the rack detailed view of the SILworX Hardware Editor.

The input variables are selected in the *Input Variables* column through a tick.

System variables	S [1)]	Description	Data type
Number of Field Errors	N	Number of current field faults.	UDINT
Number of Field Errors - Historic Count	N	Counted number of field faults (counter resettable).	UDINT
Number of Field Warnings	N	Number of current field warnings.	UDINT
Number of Field Warnings - Historic Count	N	Counted number of field warnings (counter resettable).	UDINT
Number of Communication Errors	N	Number of current communication errors.	UDINT
Number of Communication Errors - Historic Count	N	Counted number of communication errors (counter resettable).	UDINT
Number of Communication Warnings	N	Number of current communication warnings.	UDINT
Number of Communication Warnings - Historic Count	N	Counted number of communication warnings (counter resettable).	UDINT
Number of System Errors	N	Number of current system errors.	UDINT
Number of System Errors - Historic Count	N	Counted number of system errors (counter resettable).	UDINT
Number of System Warnings	N	Number of current system warnings.	UDINT
Number of System Warnings - Historic Count	N	Counted number of system warnings (counter resettable).	UDINT
Autostart	Y	TRUE: When the processor module is connected to the supply voltage, it automatically starts the user program. FALSE: When the supply voltage is connected, the processor module enters the STOP state.	BOOL
OS Major [1]…[4] OS Minor [1]…[4]	Y	Version of the operating system for every processor module. The number of redundant processor modules and the values depend on the controller type.	UINT
CRC	Y	Resource configuration checksum.	UDINT
Date/time [ms portion] Date/time [s portion]	N	System date and system time in milliseconds and seconds since 1970-01-01: 0…999 ms, 0…4 294 967 295 s.	UDINT
Force Deactivation	Y	TRUE: Forcing is deactivated. FALSE: Forcing is possible.	BOOL
Forcing Active	Y	TRUE: Global or local forcing is active. FALSE: Global and local forcing are not active.	BOOL
Force Switch State	N	Information about the selected force switch. 0xFFFF FFFE: No force switch set. 0xFFFF FFFF: At least one force switch set.	UDINT
Global Forcing Started	Y	TRUE: Global forcing is active. FALSE: Global forcing is not active.	BOOL
Spare 0…15		Reserved, do **not** use!	UDINT
Leer 17		Reserved, do **not** use!	BOOL
Last Field Warning [ms] Last Field Warning [s]	N	Date and time of the last field warning in milliseconds and seconds since 1970-01-01: 0…999 ms, 0…4 294 967 295 s.	UDINT

System variables	S [1)]	Description	Data type
Last Communication Warning [ms] Last Communication Warning [s]	N	Date and time of the last communication warning in milliseconds and seconds since 1970-01-01: 0…999 ms, 0…4 294 967 295 s.	UDINT
Last System Warning [ms] Last System Warning [s]	N	Date and time of the last system warning in milliseconds and seconds since 1970-01-01: 0…999 ms, 0…4 294 967 295 s.	UDINT
Last Field Error [ms]Last Field Error [s]	N	Date and time of the last field error in milliseconds and seconds since 1970-01-01: 0…999 ms, 0…4 294 967 295 s.	UDINT
Last Communication Error [ms] Last Communication Error [s]	N	Date and time of the last communication error in milliseconds and seconds since 1970-01-01: 0…999 ms, 0…4 294 967 295 s.	UDINT
Last System Error [ms] Last System Error [s]	N	Date and time of the last system error in milliseconds and seconds since 1970-01-01: 0…999 ms, 0…4 294 967 295 s.	UDINT
Fan State	N	Depends on the controller type; see the documentation. 0xFF: Not available.	BYTE
Mono Startup Release	Y	Enable for non-redundant operation. The system variable exists depending on the controller family TRUE: A single processor module in rack 0, slot 10/12 (H51X) may also start with one system bus only. A single processor module in rack 0, slot 16/18 (H41X) may also start with one system bus only. FALSE: Both system buses are also necessary for a single processor module.	BOOL
MultiForcing Denied	Y	TRUE: MultiForcing is denied. FALSE: MultiForcing is possible.	BOOL
Allow Online Settings	Y	Main enable switch of the processor module. TRUE: The subordinate enable switches may be changed. FALSE: The subordinate enable switches may not be changed.	BOOL
Read-only in RUN	Y	TRUE: The following operator functions are locked: Stop, Start, Download. FALSE: The following operator functions are unlocked: Stop, Start, Download.	BOOL
Redundancy Info	Y	Bit-coded redundancy state of the processor modules. The variable exists depending on the controller family.	
Reload Allowed	Y	TRUE: Reload is allowed to load a controller. FALSE: Reload is not allowed to load a controller.	BOOL
Reload Deactivation	Y	TRUE: Reload is locked. FALSE: Reload is not locked.	BOOL
Reload Cycle	Y	TRUE: The current cycle is the first cycle after a reload. FALSE: Otherwise.	BOOL
Responsible Module Essential	Y	Essential state of redundant processor modules. The variable exists depending on the controller family.	BYTE
Safety Time [ms]	Y	Safety time set for the resource in ms.	UDINT

System variables	S [1)]	Description	Data type
Start Allowed	Y	TRUE: The processor module may be started through the PADT. FALSE: The processor module may not be started through the PADT.	BOOL
Start Cycle	Y	TRUE: The current cycle is the first cycle after the start. FALSE: Otherwise	BOOL
Power Supply State [1]…[4]	N	Bit-coded state of the power supply of processor modules 1…4. The following properties differ depending on the controller family: - Possible number of processor modules. - State bits suitable for safety functions!	BYTE
System ID [SRS]	Y	System ID of the controller, 1…65535.	UINT
Systemtick HIGH Systemtick LOW	Y	Revolving millisecond counter (64-bit).	UDINT
Temperature State [1]…[4]	N	Bit-coded temperature state of processor modules 1 … 2: Bit 0: Temperature threshold 1 exceeded. Bit 1: Temperature threshold 2 exceeded. Bit 2: Incorrect temperature value.	BYTE
Remaining Global Force Duration [ms]	Y	Time in ms until the time limit set for global forcing expires.	DINT
Watchdog Time [ms]	Y	Maximum permissible duration of a RUN cycle in ms (dependent on the controller family).	UDINT
Cycle Time, last [ms]	Y	Current cycle time.	UDINT
Cycle Time, max [ms]	N	Maximum cycle time in milliseconds.	UDINT
Cycle Time, min [ms]	N	Minimum cycle time in milliseconds.	UDINT
Cycle Time, average [ms]	N	Average cycle time in milliseconds.	UDINT

1) The operating system handles the system variable in a safety-related manner, yes (Y) or no (N).

Table 13: Input Variables

The following system variables taking from Table 13 are arrays. Their index is the processor module number:

- OS major, OS minor.
- Redundancy info (bit bar).
- Power supply state.
- Temperature state.

The X-CPU 01 processor module index used in these fields is mapped onto the slots of the processor modules in the racks as specified below:

1. In rack 0, the index must be counted in ascending order starting with slot 3.
2. In rack 1, the index must be counted in descending order down to slot 3.

The following assignment results:

	Slots			
	3	4	5	6
Rack 1	4	3		
Rack 0	1	2	3	4

Table 14: Assigning the Index to X-CPU 01 Processor Module Slots

Processor modules X-CPU 01 with indexes 3 and 4 can either be located in rack 0 or rack 1!

For processor modules X-CPU 31:

- The index of processor module on slot 1 is 1.
- The index of processor module on slot 2 is 2.

5.2.5.2 Locking and Unlocking the Resource

Locking the controller locks all functions and prevents users from accessing them during operation. This also protects against unauthorized manipulations to the user program.

Unlocking the controller deactivates any locks previously set, e.g., to perform work on the controller.

The system variables *Read-Only in Run*, *Reload Deactivation*, *Force Deactivation* and *MultiForcing Denied* are used to lock the controller, see Table 12.

If all of the above system variables are TRUE, no access to the controller is possible. In this case, the controller can only enter the STOP state by setting the mode switch to the *Init* position, thus restarting all processor modules. Only then can a new user program be loaded. The example describes a simple case, in which a key-operated switch is used to lock or unlock all interventions to the resource.

To make a controller lockable

1. Define global variables of type BOOL and set initial values to FALSE.
2. Assign the global variable as output variables to the above system variables.
3. Assign the global variable to the channel value of a digital input.
4. Connect a key switch to the digital input.
5. Compile the program, load it into the controller, and start it.

► The owner of a corresponding key-operated switch is able to lock and unlock the controller. If the corresponding digital input module fails, the controller is unlocked.

This simple example can be modified using multiple global variables, digital inputs and key switches. This allows permissions for forcing, MultiForcing, reload and other operating functions to be distributed on different keys and persons.

5.2.6 User Program System Parameters

The following user parameters can be set in the *Properties* dialog box of the user programs:

System parameters	S [1)]	Description		Setting for safe operation
Name	N	Name of the user program. The name must be unique within the resource.		Any
Program ID	Y	ID for identifying the program when displayed in SILworX. Range of values: 0…4 294 967 295 Default value: 0 If *Code Generation Compatibility* is set to *SILworX V2*, only the value 1 is permitted.		Application-specific
Priority	Y	Priority of the user program. Range of values: 0…31 Default value: 0 (highest priority) This setting is only required if several user programs are used!		Application-specific
Program's Maximum Number of CPU Cycles	Y	Maximum number of CPU cycles that a user program cycle may take. Range of values: 1…4 294 967 295 Default value: 1 This setting is only required if several user programs are used!		Application-specific
Max. Duration per Cycle [µs]	N	Maximum time in each processor module cycle for executing the user program. Range of values: 0…4 294 967 295 Standard value: 0 (no limitation) The safety-related response is ensured through the watchdog. This setting is only required if several user programs are used!		Application-specific
Watchdog Time [ms] (calculated)	---	Monitoring time of the user program, calculated from the product of the watchdog time of the resource and the configured maximum number of CPU cycles. Not changeable!		
Classification	N	Classification of the user program in *Safety-related* or *Standard*; the setting is for documentation only and has no effects on the program's performance. Default value: Safety-related		Application-specific
Allow Online Settings	Y	If *Allow Online Settings* is deactivated, the settings of the reamaining program switches cannot be changed online (from within the Control Panel). Only applies if the *Allow Online Settings* switch for the resource is set to TRUE! The default setting is TRUE.		
Autostart	Y	Enabled type of Autostart: Cold Start, Warm Start, Off. The default setting is warm start.		Application-specific
Start Allowed	Y	TRUE:	The PADT may be used to start the user program. The default setting is TRUE.	Application-specific
		FALSE:	The PADT may not be used to start the user program.	

System parameters	S [1]	Description		Setting for safe operation
Test Mode Allowed	Y	TRUE:	The test mode is permitted for the user program.	Application-specific [2]
		FALSE:	The test mode is not permitted for the user program. The default setting is FALSE.	
Reload Allowed	Y	TRUE:	The user program reload is permitted. The default setting is TRUE.	Application-specific
		FALSE:	The user program reload is not permitted.	
		Observe the settings in the resource properties!		
Local Forcing Allowed	Y	TRUE:	Forcing is permitted at program level.	FALSE is recommended
		FALSE:	Forcing is not permitted at program level. The default setting is FALSE.	
Local Force Timeout Reaction	Y	Behavior of the user program after the forcing time has expired: ▪ Stop Forcing Only. ▪ Stop Program. The default setting is *Stop Forcing Only*.		
Code Generation Compatibility	-	Code generation is compatible with previous versions of SILworX.		Application-specific
		SILworX V2	Code generation is compatible with SILworX V2.	
		SILworX V3	Code generation is compatible with SILworX V3.	
		SILworX V4 – V6b	Code generation is compatible with SILworX V4 up to SILworX V6b.	
		SILworX V7 and higher	Code generation is compatible with SILworX V7.	
		The default setting for all new projects is *SILworX V7 and higher*.		

[1] The operating system handles the system parameter in a safety-related manner, yes (Y) or no (N)

[2] Once the test mode has stopped, a cold start must be performed prior to starting a safety-related operation!

Table 15: System Parameters of the User Program

5.2.7 Notes on the *Code Generation Compatibility* Parameter

Observe the following points in conjunction with the *Code Generation Compatibility* parameter:

- In a new project, SILworX selects the current setting for the *Code Generation Compatibility* parameter. This ensures that the current, enhanced features are activated and the current module and operating system versions are supported. Verify that this setting is in accordance with the hardware in use.
- In a previous project converted to the current SILworX version, the value for *Code Generation Compatibility* remains the value set in the previous version. This ensures that the configuration CRC resulting from the code generation is the same as in the previous version and the configuration is still compatible with the operating systems of the modules.
  The value of *Code Generation Compatibility* must only be changed for converted projects if additional functions of a controller should be used.
- If a *Minimum Configuration Version* of *SILworX V4* and higher is set in the resource properties, the *Code Generation Compatibility* parameter must be set to *SILworX V7 and Higher* in every user program.

5.2.8 Local User Program System Variables

The local system variables provide information at runtime about the operating conditions of the user program. Not all local system variables may be used for programming safety-related responses.

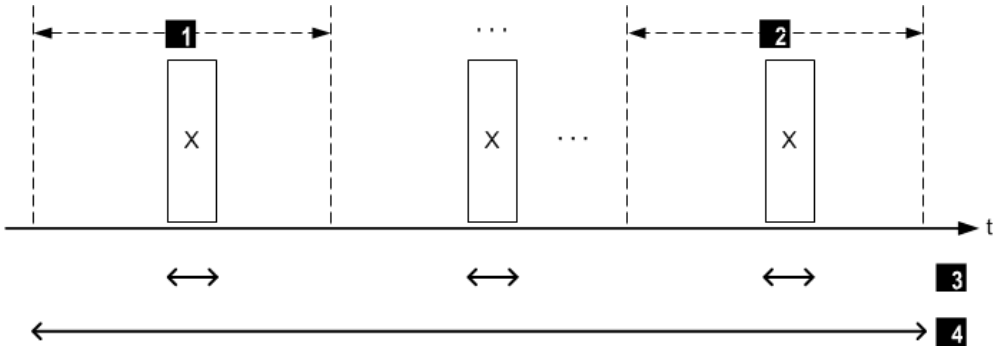
Variable	S [1)]	Description	Data type
Program_CRC	X	User program checksum used to detect potential corruptions	LWORD
Program_CycleDuration		Duration of all processor module cycles expressed in µs and required for executing a user program cycle, see Figure 13. The value is determined based on the duration of the previous user program cycle. In the first cycle after starting up the user program *Program_CycleDuration* is set to 0.	UDINT
Program_ExecutionCycles		Number of processor module cycles necessary to completely execute a user program cycle. The number is determined based on the number of processor module cycles required in the previous user program cycle.	UDINT
Program_ExecutionDuration		Time in µs necessary for executing a user program cycle, see Figure 13. The value is determined based on the processing time required in the previous cycle. In the first cycle after starting up the user program *Program_ExecutionDuration* is set to 0.	UDINT
Program_ForceSwitch	X	TRUE The conditions for local forcing are met. FALSE The conditions for local forcing are not met.	BOOL
Program_ID		Program ID assigned by the user. Program_ID identifies the user program in display functions and in the Control Panel.	UDINT
Program_ReloadCycle	X	TRUE During the first cycle after a reload Exception: Reload only changes user program parameters FALSE In all other cycles, even during a reload, which only changes user program parameters	BOOL
Program_StartCycle	X	TRUE During the first cycle after starting up the user program FALSE In all other cycles	BOOL

[1)] Only system variables that are marked in the S column may be used for safety functions! The other system variables must **not** be used for programming safety functions!

Table 16: Local User Program System Variables

Program_CycleDuration and *Program_ExecutionDuration*

The following figure shows the progression of a cycle of user program X lasting over multiple processor module cycles. The first processor module cycle considered includes the beginning of the user program cycle, and the last processor module cycle considered includes the end of the user program cycle.



1 First processor module cycle considered.
2 Last processor module cycle considered.
3 *Program_ExecutionDuration.*
4 *Program_CycleDuration.*

Figure 13: *Program_CycleDuration* and *Program_ExecutionDuration*

Program_CycleDuration indicates the duration of all the processor module cycles needed for a user program cycle; in Figure 13, it corresponds to the large double arrow 4.

Program_ExecutionDuration is the processing time portion of user program X in all the processor module cycles considered. In Figure 13, *Program_ExecutionDuration* corresponds to the sum of all small double arrows 3.

Program_ExecutionCycles is the number of cycles from the first processor module cycle 1 to the last 2.

5.2.9 Assignment to I/O Channels

In the Hardware Editor of SILworX, a global variable can be assigned to an I/O channel. In the detail view of an I/O module, drag a global variable from the Object Panel to the channel list of the I/O module.

In doing so, the channel's value and status information are available in the user program.

5.2.9.1 Use of Digital Inputs

To use the value of a digital input in the user program

1. Define a global variable of type BOOL.
2. When defining the global variable, enter the initial value as safe value.
3. Assign the global variable to the channel value of the input.

► The global variable provides the safe value to the user program.

For digital proximity switch input modules internally operating in analog mode, the raw value can also be used and the safe value can be calculated in the user program. For further details, see below.

To get additional options for diagnosing the external wiring and programming fault responses in the user program, assign global variables to *Channel OK* and to further diagnostic statuses. For further details on the individual diagnostic statuses, such as short-circuits and open-circuits, refer to the module-specific manual.

5.2.9.2 Use of Analog Inputs

Analog input channels convert the measured input currents into a value of type DINT (double integer). This value is then provided to the user program as a raw value. Here, 1 mA corresponds to a value of 10 000 and the range of values is 0…240 000.

As an easier alternative, the process value of the REAL data type can often be used instead of the raw value. HIMax calculates the process value based on the raw value and the scale value on the parameters 4 and 20 mA. Refer to the module-specific manual for details.

The values of analog inputs can be used in the user program in two ways:

- Using the process value
  If an analog input is configured correctly, its process value includes the safe fault response.
- Using the raw value
  The raw value is the measured value without the safe fault response. The safe fault response must be programmed in line with the project requirements.

To use the process value

1. Define a global variable of type REAL.
2. When defining the global variable, enter the initial value as safe value.
3. Assign the global variable to the process value of the input.
4. Specify the measuring range of the channel by giving a REAL value for 4 mA and for 20 mA.

► The global variable provides the safe value to the user program.

To use the raw value

1. Define a global variable of type DINT.
2. In the user program, define a global variable of the type needed.
3. Program a suitable conversion function to convert the raw value into a type used in the user program. In doing so, take the measuring range of the data type into account.
4. In the user program, program a safety-related fault response using the statuses *Channel OK, SC, OC* (and others, if necessary).

► The user program can safely process the measured value.

If the value 0 for a channel is within the valid measuring range, the user program must at least evaluate the parameter *Channel OK* in addition to the process value.

To get additional options for diagnosing the external wiring and programming fault responses in the user program, assign global variables to *Channel OK*, *Submodule OK*, *Module OK* and to further diagnostic statuses. For further details on the individual diagnostic statuses, such as short-circuits and open-circuits, refer to the module-specific manual.

5.2.9.3 Use of Safety-Related Counter Inputs

The counter reading or the rotational speed/frequency can be used as an integer value or as a scaled floating-point value.

To use the integer value

1. Define a global variable of type UDINT.
2. When defining the global variable, enter the initial value as safe value.
3. Assign the global variable to the integer value of the input.

► The global variable provides the safe value to the user program.

To use the scaled floating point value

1. Define a global variable of type REAL.
2. When defining the global variable, enter the initial value as safe value.
3. Assign the global variable to the scaled floating point value of the input.
4. Specify the scaling value of the channel by giving a REAL value.

► The global variable provides the safe value to the user program.

5.2.9.4 Use of Digital Outputs

To write a value in the user program to a digital output

1. Define a global variable of type BOOL.
2. When defining the global variable, enter the initial value as safe value.
3. Assign the global variable to the channel value of the output.

► The global variable provides the safe value to the digital output.

To get additional options for diagnosing the external wiring and programming fault responses in the user program, assign global variables to *Channel OK* and to further diagnostic statuses. For further details on the individual diagnostic statuses, such as short-circuits and open-circuits, refer to the module-specific manual.

5.2.9.5 Use of Analog Outputs

To write a value in the user program to an analog output

1. Define a global variable of type REAL.
2. When defining the global variable, enter the initial value as safe value.
3. Assign the global variable to the channel value of the output.
4. In connection with the output channel parameters *4 mA* and *20 mA*, set the REAL values in accordance with the range used for global variables.

► The global variable provides the safe value to the analog output.

i If output channels are not (or no longer) used, the parameters *4 mA* and *20 mA* must be set to the default settings 4.0 and 20.0, respectively.

To get additional options for diagnosing the external wiring and programming fault responses in the user program, assign global variables to *Channel OK* and to further diagnostic statuses. For further details on the individual diagnostic statuses, such as short-circuits and open-circuits, refer to the module-specific manual.

5.2.10 Assignment to Communication Connections

Communication connections can be used to send or receive the values of global variables. To do this, open the editor for the communication protocol in use and drag the global variable from the Object Panel onto the workspace.

For further details on the communication protocols, refer to the communication manual (HI 801 101 E). For further details on how to use the editor for the communication protocols, refer to the SILworX online help.

5.2.11 Configuring the Sequence of Events Recording

Definition of 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 **Alarms&Events** branch, if not existing.
3. Define events in the A&E Editor.
   - Drag global variables onto the event window for Boolean or scalar events.
   - Define the details of the events, see Table 17 and Table 18.

► The events are defined.

For further information, refer to the SILworX online help.

The parameters of the Boolean events must be entered in a table with the following columns:

Column	Description	Range of values
Name	Name for the event definition; must be unique within the resource.	Text, max. 63 characters
Global Variable	Name of the assigned global variable (added using drag&drop).	
Data type	Data type of the global variable; cannot be changed.	BOOL
Event Source	CPU event: The processor module creates the timestamp. It creates all the events in each of its cycles. I/O event: A suitable I/O module (e.g., AI 32 02) creates the timestamp. Auto Event: The timestamp is created by a suitable I/O module, if assigned, otherwise by the processor module. Default value: Auto	CPU, I/O, Auto
Alarm when FALSE	Activated: If the global variable value changes from TRUE to FALSE, an event is triggered. Deactivated: If the global variable value changes from FALSE to TRUE, an event is triggered. Default value: Deactivated	Checkbox activated, deactivated
Alarm Text	Text specifying the alarm state.	Text
Alarm Priority	Priority of the alarm state. Default value: 500	0...1000
Alarm Acknowledgment Required	Activated: The alarm state must be confirmed by the user (acknowledgement). Deactivated: The alarm state need not be confirmed by the user. Default value: Deactivated	Checkbox activated, deactivated
Return to Normal Text	Text specifying the alarm state.	Text
Return to Normal Severity	Priority of the normal state.	0...1000
Return to Normal Ack Required	The normal state must be confirmed by the user (acknowledgement). Default value: Deactivated	Checkbox activated, deactivated

Table 17: Parameters for Boolean Events

The parameters of the scalar events must be entered in a table with the following columns:

Column	Description	Range of values
Name	Name for the event definition; must be unique within the resource.	Text, max. 63 characters
Global Variable	Name of the assigned global variable (added using drag&drop).	
Data type	Data type of the global variable; cannot be changed.	Depending on the global variable type.
Event Source	CPU event: The processor module creates the timestamp. It creates all the events in each of its cycles. I/O event: A suitable I/O module (e.g., AI 32 02) creates the timestamp. Auto event: The timestamp is created by a suitable I/O module, if assigned, otherwise by the processor module. Default value: Auto	CPU, I/O, Auto
HH Alarm Text	Text specifying the alarm state of the highest limit (HH).	Text
HH Alarm Value	Highest limit (HH) triggering an event. Condition: (HH Alarm Value - Hysteresis) > H Alarm Value or HH Alarm Value = H Alarm Value	Depending on the global variable type.
HH Alarm Priority	Priority of the highest limit (HH); default value: 500	0...1000
HH Alarm Acknowledgment Required	Activated: The user must confirm that the highest limit (HH) has been exceeded (acknowledgment). Deactivated: The user need not confirm that the highest limit (HH) has been exceeded. Default value: Deactivated	Checkbox activated, deactivated
H Alarm Text	Text specifying the alarm state of the high limit (H).	Text
H Alarm Value	High limit (H) triggering 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 high limit (H); default value: 500	0...1000
H Alarm Acknowledgment Required	Activated: The user must confirm that the high limit (H) has been exceeded (acknowledgment). Deactivated: The user need not confirm that the high limit (H) has been exceeded. Default value: Deactivated	Checkbox activated, deactivated
Return to Normal Text	Text specifying the normal state.	Text
Return to Normal Severity	Priority of the normal state; default value: 500	0...1000
Return to Normal Ack Required	The normal state must be confirmed by the user (acknowledgement); default value: Deactivated	Checkbox activated, deactivated
L Alarm Text	Text specifying the alarm state of the low limit (L).	Text
L Alarm Value	Low limit (L) triggering 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 low limit (L); default value: 500	0...1000
L Alarm Acknowledgment Required	Activated: The user must confirm that the low limit (L) has been exceeded (acknowledgment). Deactivated: The user need not confirm that the low limit (L) has been exceeded. Default value: Deactivated	Checkbox activated, deactivated
LL Alarm Text	Text specifying the alarm state of the lowest limit (LL).	Text
LL Alarm Value	Lowest limit (LL) triggering an event. Condition: (LL Alarm Value + Hysteresis) < (L Alarm Value) or LL Alarm Value = L Alarm Value	Depending on the global variable type.

Column	Description	Range of values
LL Alarm Priority	Priority of the lowest limit (LL); default value: 500	0...1000
LL Alarm Acknowledgment Required	Activated: The user must confirm that the lowest limit (LL) has been exceeded (acknowledgment). Deactivated: The user need not confirm that the lowest limit (LL) has been exceeded. Default value: Deactivated	Checkbox activated, deactivated
Alarm Hysteresis	The hysteresis avoids a continuous creation of many events if the process value often oscillates around a limit.	Depending on the global variable type.

Table 18: Parameters for Scalar Events

NOTICE

Faulty event recording due to improper parameter settings possible!

Setting the parameters *L Alarm Value* and *H Alarm Value* to the same value can cause an unexpected behavior of the event recording since no normal range exists in such a case.

For this reason, make sure that *L Alarm Value* and *H Alarm Value* are set to different values.

5.2.11.1 Status of LL, L, N, H, HH in X-AI 32 01 and X-AI 32 02

If scalar events have been defined for the thresholds of a channel in an analog input module (X-AI 32 01 or X-AI 32 02), the state variables -> *State LL*, -> *State L*, -> *State N*, -> *State H*, -> *State HH* are available.

For safety-related applications, these state variables must be connected to *Channel OK***!**

5.3 Forcing

Forcing is the procedure of manually writing to variables with values that do not result from the process, but are defined by the user, while the controller is processing the user program.

There are different types of globally forcible data sources in a system:

- All input and status information from modules (e.g., I/O modules) and communication protocols.
- All global variables that have not been written, but have been read (VAR_EXTERNAL).
- All global variables that have been written to by a user program (VAR_EXTERNAL).

In addition to the globally forcible data sources in a system, there are also different types of locally (in the user program) forcible data sources:

- All user program variables that have not been written, but have been read (VAR).
- All variables from a user program that have been written (VAR).

i When a variable is forced, forcing always applies to its data source! A forced variable does not depend on the process since its value is defined by the users.

5.3.1 Use of Forcing

Forcing supports users during the following tasks:

- Testing of the user program for cases that do not, or only infrequently occur during normal operation and are therefore only testable up to a certain extent.
- Simulation of sensor values, e.g., of unconnected sensors.
- Service and repair work.
- General troubleshooting.

⚠ WARNING

Physical injury due to forced values is possible!

- **Only force values after consent of the person responsible for the plant and the test authority during commissioning.**
- **Only remove existing forcing restrictions with the consent of the person responsible for the plant and the test authority during commissioning.**

When forcing values, the person in charge must take further technical and organizational measures to ensure that the process is sufficiently monitored in terms of safety. HIMA recommends setting a time limit for the forcing procedure, refer to Chapter 5.3.3 for details.

⚠ WARNING

Failure of safety-related operation possible due to forced values!

- **Forced value may lead to unexpected output values.**
- **Forcing prolongs the cycle time. This can cause the watchdog time to be exceeded.**

Forcing can operate at two levels:

- Global forcing: Global variables are forced for all applications.
- Local forcing: Local variables are forced within a user program.

5.3.2 Assigning a Data Source Changed through Reload

Assigning variables to a new data source by performing a reload may have unexpected results in conjunction with the following input types:

- Hardware.
- Communication protocols.
- System variables.

The following changes resulting from a reload lead to changed force states:

1. A global variable A is assigned to a forced data source and is thus forced itself.
2. The assignment of global variable A is removed by performing a reload. The data source maintains the property *Forced*. Global variable A is no longer forced.
3. The forced data source is assigned another global variable (global variable B).
4. During the next reload, global variable B will be forced, even if unintentionally.

Consequence

To prevent this effect, stop forcing a variable before changing the data source. To this end, deactivate the individual force switch.

The *Inputs* tab in the Force Editor displays which channels are being forced.

i Global variables having the user program as data source retain the *forced* setting even when the assignment is changed. If the new data source is the user program again.

5.3.3 Time Limits

Different time limits can be set for global or local forcing. Once the defined time has expired, the controller stops forcing values.

The behavior of the HIMax system upon expiration of the time limit can be configured:

- For global forcing, the following settings can be selected:
  - *Stop Resource.*
  - *Stop Forcing Only,* i.e., the resource continues to operate.
- For local forcing, the following settings can be selected:
  - *Stop Program.*
  - *Stop Forcing Only,* i.e., the user program continues to run.

Forcing can also be used without time limit. In this case, the forcing procedure must be stopped manually.

The person responsible for forcing must clarify what effects stopping forcing have on the entire system!

5.3.4 Restricting the Use of Forcing

The user can limit the use of forcing; disturbed operation which may be caused by forcing, is to be avoided. The following measures can be implemented in the configuration:

- Configuring different user profiles with or without forcing permissions.
- Explicitly allowing forcing for a resource (PES).
- Setting up MultiForcing user accounts in PES User Management.
- Explicitly allowing local forcing for a user program.
- Forcing can be stopped immediately via the *Force Deactivation* system variable using the key switch.
- Additionally, the *MultiForcing Denied* system variable can be used to disable MultiForcing.

5.3.5 Force Editor

SILworX Force Editor lists all the variables, grouped in global and local variables.

For each variable, the following can be set:

- Forcing values.
- One individual force switch to prepare for forcing variables.

Forcing can be started and stopped for both local and global variables.

Forcing can be started for a predefined time limit or for an indefinite time period. After forcing has started, all variables with an active force switch are set to their force values.

When forcing is stopped, manually or because the time limit has expired, the variables will again receive their values from the process.

If forcing is started again, the configured force values will replace the values from the process!

For further details on the Force Editor and forcing, refer to the SILworX online help.

Copying the current data from the Force Editor to the clipboard only gathers the data visible in the Force Editor. Data that are not visible are not periodically refreshed and can thus have an obsolete value! Press `Ctrl+A` to select and copy all the data, even if they are not in the visible area.

5.3.6 Automatic Forcing Reset

The operating system resets forcing in the following cases:

- When the resource is restarted, e.g., after connecting the supply voltage.
- When the resource is stopped.
- When a new configuration is loaded by performing a download.
- When a user program is stopped: Reset of local forcing for this user program.

During a reload, local and global force values, individual force switches, force times and force timeout responses are still valid.

⚠ WARNING

Failure of safety-related operation possible due to forced values!

- **Global force values and individual force switches can be set when a resource is stopped. The configured values become valid after restarting the resource and forcing.**
- **Local force values and individual force switches can be set when the user program is stopped. The configured values become valid after restarting the user program and forcing.**

5.3.7 Forcing and Scalar Events

When forcing a global variable used to create scalar events, observe the following points:

- The events are created in accordance with the force value.
- The values of these variable-dependent status variables are not tracked to the force value!

In such cases, the corresponding status variables must also be forced!

5.3.8 MultiForcing

Users with MultiForcing access can write force data (force values and individual force switches) for global variables in a resource if the required higher-order conditions have been met and the force permissions have been granted. To all other functions of a resource, users have Read-Only access. Starting, stopping or resetting a force process is not possible.

The use of MultiForcing is limited to a maximum of 5 users at a time. The users can be working from separate locations and also independently of each other in terms of time. The separation of the tasks performed by the individual users must be ensured by the operator through organizational measures.

⚠ WARNING

Behavior that cannot be controlled by the user, is possible!

The operator must ensure that different Force Users do not force the same variables simultaneously and that there can be no overlaps in timing. If several Force Users write to the same variables, those force values and force switches will prevail which were written last by the firmware. Because force data are transferred in several blocks , it would otherwise be possible for the settings of different Force Users to take effect on one single controller. This behavior cannot be controlled by the user.

⚠ WARNING

Existing force data is not deactivated, if *MultiForcing Denied* = TRUE!

If *MultiForcing Denied* is TRUE, users with MultiForcing access cannot modify force values or the force switches. Existing force data is not deactivated, if *MultiForcing Denied* = TRUE! Global Forcing, if allowed, is then only possible for a single user with at least Operator permissions.

5.3.8.1 Objectives of MultiForcing

For commissioning, normative and functional loop tests are prescribed as part of the site acceptance test, whereby a loop represents the path from the sensor to the actuator. MultiForcing makes it possible to distribute the resulting tasks to up to 5 PADTs thus processing them efficiently.

Based on loop tests, the nominal operating range is checked as well as the responses in the event of open-circuits and short-circuits. Because numerous loops must be tested frequently, the duration of site acceptance testing is a significant cost factor. MultiForcing can help to optimize these tasks.

- The behavior of actuators and linked information (e.g., end position feedback) is tested through forcing. The output signals are forced directly. This tests the wiring and the external circuit.
- In a system which is only partially functional, sensors are tested through forcing in such a way that the tests have no effect on the actuators. This approach can also be used for troubleshooting in connection with sensors.

5.3.8.2 Global MultiForcing

Global MultiForcing is the simultaneous writing of force data (force values and force switches) for global variables by more than one user (Force Users).

A Force User is a person who is logged into a controller with either MultiForcing, Operator, Write or Administrator permissions. Every Force User is able to read and also at least write force data. A maximum of 5 Force Users can be logged into each controller. The number of current Force Users is displayed in the SILworX status bar.

Force values and force switches set by a Force User with MultiForcing access may only take effect if the user is logged into the controller with at least Operator permissions. Only this user can start or stop forcing.

i To perform Global MultiForcing, Global Forcing must be allowed as well! The settings are displayed online.

5.4 Cycle Sequence

In a simplified overview, a processor module cycle (CPU cycle) of only one user program runs through the following phases:

1. Processing of the input data.
2. Processing of the user program.
3. Provision of the output data.

These phases do not include special tasks such as reload, which might be executed within a CPU cycle.

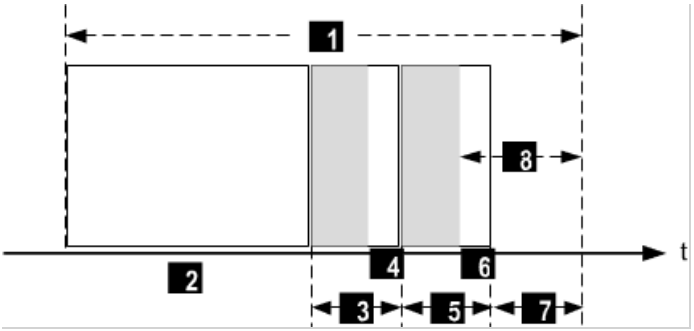
Global variables, results from function blocks, and other data are processed in the first phase and represent the input data for the second phase. The first phase need not start at the beginning of the cycle, but may be delayed. For this reason, inaccurate cycle times, potentially exceeding the watchdog time, may result if timer function blocks are used to determine the cycle time in the user program.

In the third phase, the user program results are forwarded for being processed in the following cycles and supplied to the output channels.

5.4.1 Watchdog Time Reserve and Target Cycle Time Reserve

The SILworX Control Panel displays the available reserve time, *Watchdog Time Reserve [ms]* and *Target Cycle Time Reserve [ms]*.

The reserve times indicate the time within the cycle that the controller has to execute additional tasks such as reload when the times configured in the cycle have elapsed.



1. Watchdog time or target cycle time.
2. Time required by the programs.
3. Value configured for *Max. Duration of Configuration Connections [ms].*
4. Unneeded portion of *Max. Duration of Configuration Connections [ms].*
5. Value configured for *Max.Com. Time Slice ASYNC [ms].*
6. Unneeded portion of *Max.Com. Time Slice ASYNC [ms].*
7. Reserve time displayed for operating systems V6 and higher.
8. Reserve time displayed for operating systems prior to V6.

Figure 14: Reserve Times within the Cycle

Note about Figure 14: The scale and order of the times do not necessarily correspond to the actual conditions!

The configured times are included in the calculations of the reserve times as $t_{CycleAdjust}$. The following condition applies:

Operating system version	Value of $t_{CycleAdjust}$
V6 and higher	Values configured for *Max. Duration of Configuration Connections [ms]* + $t_{IO\ phase\ timeout}$ (see below) + *Max. Com. Time Slice ASYNC [ms]*
Prior to V6	Value configured for *Max. Duration of Configuration Connections [ms]* + $t_{IO\ phase\ timeout}$ (see below)

Table 19: $t_{CycleAdjust}$ Depending on the CPU Operating System Version

The value of $t_{IO\ phase\ timeout}$ can be determined with the following table as a function of the parameter *Maximum System Bus Latency [µs]* and of the processor module type:

Maximum System Bus Latency [µs]	Processor module type	$t_{IO\ phase\ timeout}$ =
≥ 100	X-CPU 01	2* *Maximum System Bus Latency [µs]* + 0.169 µs
	X-CPU 31	2* *Maximum System Bus Latency [µs]* + 0.533 µs
≠ *System Defaults*	X-CPU 01	1.232 ms + 5.5 ms = 6.732 ms
	X-CPU 31	1.844 ms + 5.5 ms = 7.344 ms

Table 20: Determining the $t_{IO\ phase\ timeout}$

The reload uses the times as follows:

- The reload of a user program, the cycle of which takes **multiple** processor module cycles, uses the displayed reserve time (**7** or **8**).

  Such a reload is rejected if the reserve time is < 6 ms!
- The reload of a user program where the cycle only takes **one** processor module cycle uses the following times.
  - The displayed reserve time (**7** or **8**).
  - The portion of configured times ($t_{CycleAdjust}$), that was not completely used (**4** and, with V6 and higher, **6**).

  For this reason, a reload is also possible if the displayed reserve time is 0.

The following activities use the displayed reserve time to run faster:

- Activities at a user program start when the PES is in RUN.
- Project configuration storage in the non-volatile memory during a reload.

Watchdog Time Reserve

The *Watchdog Time Reserve [ms]* is calculated as follows:

$t_{Reserve} \leq t_{WDT} - t_T - t_{VarAdjust} - t_{CycleAdjust}$

Where:

$t_{Reserve}$	*Watchdog Time Reserve [ms]*
t_{WDT}	Configured watchdog time
t_T	Cycle time actually needed
$t_{VarAdjust}$	10 ms for HIMax
$t_{CycleAdjust}$	See also Table 19

Target Cycle Time Reserve

If a target cycle time is set to a value > 0, a value > 0 is displayed for the *Target Cycle Time Reserve* in the SILworX Control Panel. The *Target Cycle Time Reserve [ms]* is calculated as follows:

$t_{Reserve} \leq t_{Target} - t_T - t_{VarAdjust} - t_{CycleAdjust}$

Where:

$t_{Reserve}$	*Target Cycle Time Reserve [ms]*
t_{Target}	Configured target cycle time
t_T	Cycle time actually needed
$t_{VarAdjust}$	2 ms
$t_{CycleAdjust}$	See also Table 19

i A certain reserve time must exist for the target cycle time, otherwise HIMax may reject the synchronization of a processor module. Synchronization is also rejected if *Target Cycle Time Mode* is set to *Fixed-tolerant* or *Dynamic-tolerant*.

The reserve time value for the target cycle time depends on the settings of the following system parameters (see the previous formulas and tables):

- *Max. Duration of Configuration Connections [ms]*
- *Max. Com. Time Slice ASYNC [ms]*
- *Maximum System Bus Latency [µs]*

Workaround: Set the system parameters such that the reserve time for the target cycle time is sufficiently large.

5.4.2 Multitasking

Multitasking refers to the capability of the HIMax system to process up to 32 user programs within the processor module.

This allows the project's sub-functions to be separated from one another. The individual user programs can be started and stopped independently from one another. SILworX displays the states of the individual user programs in the Control Panel and enables their operation.

Using multitasking, the second phase changes so that a CPU cycle performs the following tasks:

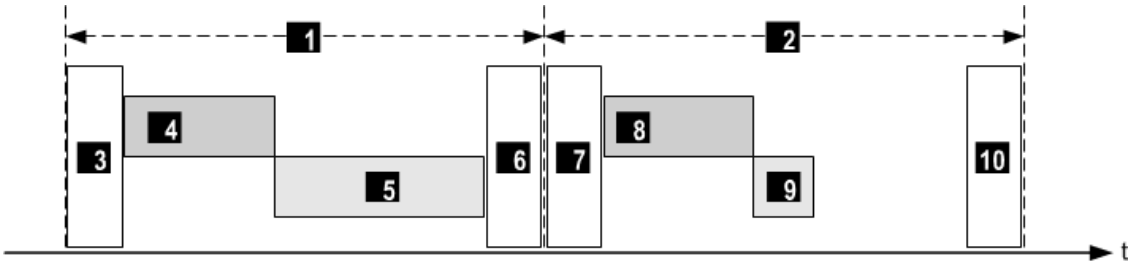
1. Processing of the input data.
2. Processing of all the user programs.
3. Provision of the output data.

In the second phase, the HIMax can run up to 32 user programs. Two scenarios are possible for each user program:

- An entire user program cycle can be run within a single CPU cycle.
- A user program cycle requires multiple CPU cycles to be completed.

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, but after the user program has been completely executed. This data can thus first be used as input values at the next start of another user program.

The example in Figure 15 shows both scenarios in a project containing two user programs, UP 1 and UP 2.



1 First CPU cycle considered.
2 Second CPU cycle considered.
3 Input processing in the first CPU cycle.
4 First cycle considered for UP1.
5 First portion of the cycle considered for UP 2.
6 Output processing in the first CPU cycle.
7 Input processing in the second CPU cycle.
8 Second cycle considered for UP1.
9 Second portion of the cycle considered for UP 2.
10 Output processing in the second CPU cycle.

Figure 15: CPU Cycle Sequence with Multitasking

Each UP 1 cycle is completely processed during each CPU cycle. UP 1 processes an input change registered by the system at the beginning of the CPU cycle 1 and delivers a response at the end of the cycle.

One UP 2 cycle requires two CPU cycles to be processed. UP 2 needs CPU cycle 1 to process an input change registered by the system at the beginning of CPU cycle 2. For this reason, the response to this input change is only available at the end of CPU cycle 2.
The response time of UP 2 is twice as long as that of UP 1.

Upon completion of the first part 5 of the UP 2 cycle under consideration, UP 2 processing is **completely** aborted and only resumed when 9 starts. During its cycle, UP 2 processes the data provided by the system during 3. The results of UP 2 are available to the system during 10 (e.g., for process output). The data that the system exchanges with the user program is always consistent.

The program execution order can be controlled by assigning a priority, which indicates how important the corresponding user program is compared to the others (see multitasking mode 2).

To specify the user program execution order, use the following parameters in the resources and programs or in the Multitasking Editor:

Parameters	Description	Configurable within
Watchdog time	Watchdog time of the resource.	Resource, Multitasking Editor
Target Cycle Time [ms]	Required or maximum cycle time.	Resource, Multitasking Editor
Multitasking Mode	Use of the execution duration not required by the user program, e. g., the difference between actual execution duration in one CPU cycle and the defined *Max. Duration for Each Cycle [µs]*.	Resource, Multitasking Editor
	Mode 1 The duration of a CPU cycle is based on the required execution time for all user programs.	
	Mode 2 The processor provides the execution time portion not needed by lower priority user programs to higher priority user programs. Operation mode for high availability.	
	Mode 3 The processor waits until the execution time not needed by the user programs has expired, thus increasing the cycle.	
Target Cycle Time Mode	Use of *Target Cycle Time [ms].*	Resource, Multitasking Editor
Program ID	ID for identifying the program when displayed in SILworX.	User Program
Priority	Importance of a user program; highest priority: 0.	User Program
Program's Maximum Number of CPU Cycles	Maximum number of CPU cycles required to process one user program cycle.	User Program
Max. Duration for Each Cycle [µs]	Time permitted for executing the user program within a CPU cycle.	User Program

Table 21: Parameters Configurable for Multitasking

Observe the following rules when setting the parameters:

- If *Max. Duration for Each Cycle [µs]* is set to 0, the execution time of the user program is not limited, e.g., it is always processed completely. Therefore, the *Program's Maximum Number of CPU Cycles* may only be set to 1 in this case.
- The sum of the *Max. Duration for Each Cycle [µs]* parameters in all user programs must not exceed the resource watchdog time. Make sure that sufficient reserve is planned for processing the remaining system tasks, see Chapter 5.4.1.
- The sum of the *Max. Duration for Each Cycle [µs]* parameters in all user programs must be large enough to ensure that sufficient reserve is available to maintain the target cycle time.
- The *Program ID*s of all user programs must be unique.

During verification and code generation, SILworX monitors that these rules are observed. These rules must also be observed when changing the parameters online or through reload. Additionally, take the following system behavior into account when *Program's Maximum Number of CPU Cycles* is modified through an online change or reload:

- If *Program's Maximum Number of CPU Cycles* is **reduced** compared to the current value, the change only applies when the user program processing has stopped. The new value only applies for the execution of the next user program cycle. During the execution of the user program cycle, the value of *Program's Maximum Number of CPU Cycles* is not reduced. This ensures that a user program processed over multiple CPU cycles does not enter the ERROR STOP state because *Program's Maximum Number of CPU Cycles* has been overrun.

- If *Program's Maximum Number of CPU Cycles* is **increased** compared to the current value, the change applies immediately. After a reload, the change takes effect as soon as the reload configuration becomes active in the resource. The value of *Program's Maximum Number of CPU Cycles* can also be reduced during the execution of the user program cycle.

SILworX uses these parameters to calculate the user program watchdog time:

Watchdog time of the user program = *Watchdog Time* * *Program's Maximum Number of CPU Cycles*

i The sequence control for executing the user programs is run in cycles of 250 µs. For this reason, the values set for *Max. Duration For Each Cycle [µs]* can be exceeded or under-run by up to 250 µs.

The individual user programs operate in an interference-free manner and independently from one another. However, reciprocal influence can be caused by:

- Use of the same global variables in several user programs.
- Unpredictably long runtimes can occur in individual user programs if no limit is configured with *Max Duration for Each Cycle*.
- The distribution of user program cycles over processor module cycles strongly affects the user program response time and the response time of the variables written to by the user program!
- A user program evaluates global variables written to by another user program at least one processor module cycle later. Depending on the value set in the programs for *Program's Maximum Number of CPU Cycles*, the reading process may be prolonged by many processor module cycles. The response to changes performed to such global variables is thus delayed!

i HIMA recommends setting the *Max. Duration for Each Cycle [µs]* parameter to an appropriate value ≠ 0. This ensures that a user program with an excessively long runtime is stopped during the current CPU cycle and resumed in the next CPU cycle without affecting the other user programs.

Otherwise, an unusually long runtime for one or several user programs can cause the target cycle time, or even the resource watchdog time, to be exceeded, thus leading to an error stop of the controller.

The operating system defines in which order the user programs are executed in accordance with the following scheme:

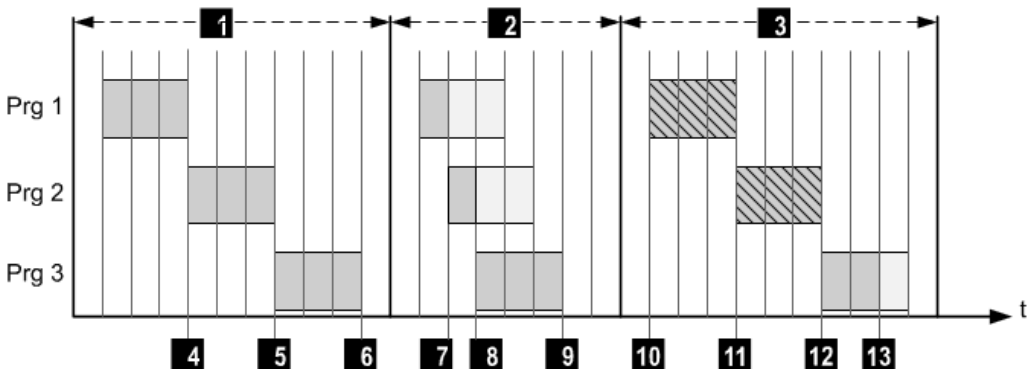
- User programs with lower priority are executed before user programs with higher priority.
- If the user programs have the same priority, the system processes them in ascending order of the *Program ID*s.

This order is also followed when starting and stopping the user program during the start and stop of the PES, respectively.

5.4.3 Multitasking Mode

There are 3 operation modes for multitasking: Multitasking Mode 1...Multitasking Mode 3. These modes differ in how the time that is not needed for executing the CPU cycle of the user programs is used. One mode can be selected for every resource.

1. **Multitasking Mode 1** uses the unneeded time to reduce the CPU cycle. If the user program is completely processed, processing of the next user program begins immediately. In total, this results in a shorter cycle.
   Example: 3 user programs (*Prg 1*, *Prg 2* and *Prg 3*) that allow a user program cycle to take up to 3 CPU cycles.



1 First CPU cycle considered. For additional numbers, see the text.
2 Second CPU cycle considered.
3 Third CPU cycle considered.

Figure 16: Multitasking Mode 1

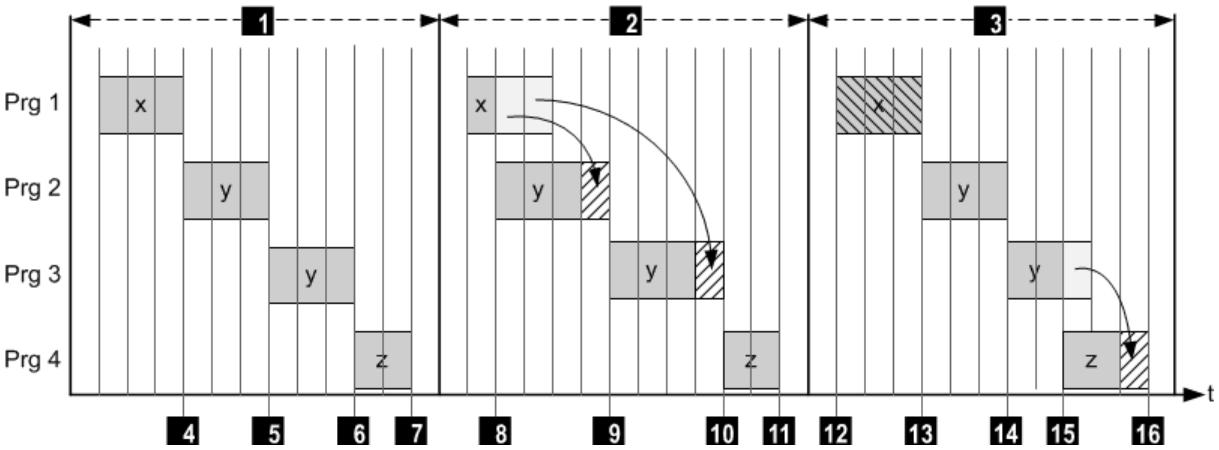
Sequence of the three cycles considered:

4 *Max. Duration for Each Cycle [µs]* of *Prg 1* expired, *Prg 2* starts.
5 *Max. Duration for Each Cycle [µs]* of *Prg 2* expired, *Prg 3* starts.
6 *Max. Duration per Cycle [µs]* of *Prg 3* expired, end of the first CPU cycle.
7 *User Program Cycle* of *Prg 1* complete, *Prg 2* resumed.
8 User program cycle of *Prg 2* complete, *Prg 3* resumed.
9 *Max. Duration per Cycle [µs]* of *Prg 3* expired, end of the second CPU cycle.
10 The next user program cycle of *Prg 1* starts.
11 *Max. Duration for Each Cycle [µs]* of *Prg 1* expired. Next user program cycle of *Prg 2* starts.
12 *Max. Duration for Each Cycle [µs]* of *Prg 2* expired, *Prg 3* starts.
13 User program cycle of *Prg 3* complete.

2. In **Multitasking Mode 2,** the unneeded duration of lower-priority user programs is distributed among higher-priority user programs. In addition to the specified *Max. Duration for Each Cycle [µs],* these user programs can use the portions of unneeded duration. This procedure ensures high availability.

   Four user programs are used in the example: *Prg 1*…*Prg 4*. The following priorities are allocated to the user programs:

   - *Prg 1* has the lowest priority, priority *x.*
   - *Prg 2* and *Prg 3* have a medium priority, priority *y.*
   - *Prg 4* has the highest priority, priority *z.*



1 First CPU cycle considered.
2 Second CPU cycle considered.
3 Third CPU cycle considered.

For additional numbers, see the text.

Figure 17: Multitasking Mode 2

Cycle sequence:

4 *Max. Duration for Each Cycle [µs]* of *Prg 1* expired, Prg 2 starts.
5 *Max. Duration for Each Cycle [µs]* of *Prg 2* expired, *Prg 3* starts.
6 *Max. Duration for Each Cycle [µs]* of *Prg 3* expired, *Prg 4* starts.
7 *Max. Duration for Each Cycle [µs]* of *Prg 4* expired, first CPU cycle complete.
8 User program cycle of *Prg 1* complete, *Prg 2* resumed. The remaining duration is distributed to the *Max. Duration for Each Cycle [µs]* of *Prg 2* and *Prg 3* (medium priority *y*) (arrows).
9 *Max. Duration for Each Cycle [µs]* of *Prg 2*+ proportional remaining duration of *Prg 1* expired, *Prg 3* resumed.
10 *Max. Duration per Cycle [µs]* of *Prg 3* and proportional remaining duration of *Prg 1* expired, *Prg 4* starts.
11 *Max. Duration for Each Cycle [µs]* of *Prg 4* expired, second CPU cycle complete.
12 The next user program cycle of *Prg 1* starts.
13 *Max. Duration for Each Cycle [µs]* of *Prg 1* expired. *Prg 2* resumed.
14 *Prg 2 Max. Duration for Each Cycle [µs]* complete, *Prg 3* resumed.
15 User Program Cycle of *Prg 3* complete, *Prg 4* resumed. The remaining duration is added to *Prg 4* (highest priority *z*).
16 *Max. Duration for Each Cycle [µs]* of *Prg 4* + remaining duration of *Prg 3* expired, third CPU cycle complete.

i The execution time not used by user programs that were not run, is not available as residual time for other user programs. User programs are not run if they are in one of the following states:

- STOP
- Error
- TEST_MODE

As a consequence, the number of CPU cycles required to process another user program cycle could increase.

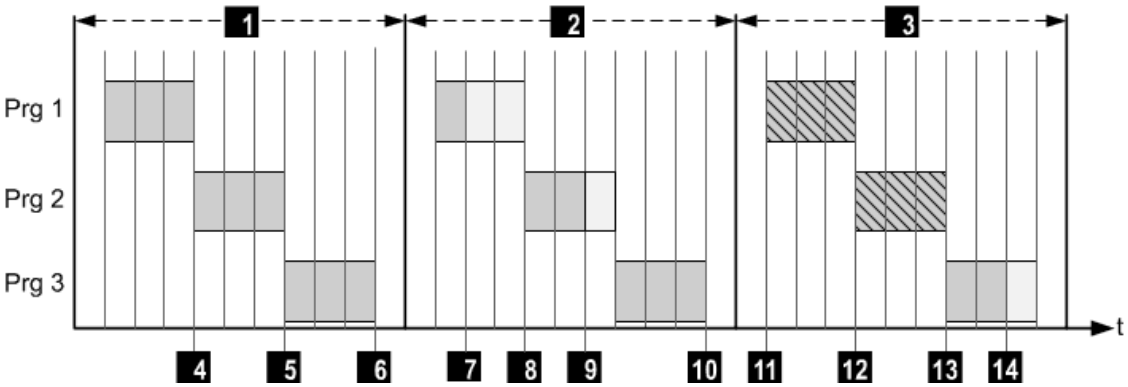
In such a case, if the value set for *Maximum Cycle Count* is too low, the maximum time for processing a user program can be exceeded and result in ERROR_STOP!
Maximum processing time = *Max. Duration for Each Cycle [µs] * Maximum Number of Cycles*

Use Multitasking Mode 3 to verify the parameter setting!

3. **Multitasking Mode 3** does not use the unneeded duration for running the user programs, rather, it waits until the *Max. Duration for Each Cycle [µs]* of the user program is reached and then starts processing the next user program. This behavior results in CPU cycles of the same duration.

   Multitasking Mode 3 serves for verifying if Multitasking Mode 2 can ensure proper program execution, even in the worst case scenario.

   The example examines user programs named *Prg 1*, *Prg 2* and *Prg 3*:



1 First CPU cycle considered.
2 Second CPU cycle considered.
3 Third CPU cycle considered.

For additional numbers, see the text.

Figure 18: Multitasking Mode 3

Cycle sequence:

4 *Max. Duration for Each Cycle [µs]* of *Prg 1* expired, *Prg 2* starts.
5 *Max. Duration for Each Cycle [µs]* of *Prg 2* expired, *Prg 3* starts.
6 *Max. Duration for Each Cycle [µs]* of *Prg 3* expired, first CPU cycle complete. *Prg 1* resumed.
7 User program cycle of *Prg 1* complete. Waiting for the remaining duration.
8 *Max. Duration for Each Cycle [µs]* of *Prg 1* expired. *Prg 2* resumed.
9 User program cycle of *Prg 2* complete. Waiting for the remaining duration.
10 *Max. Duration for Each Cycle [µs]* of *Prg 3* expired. Second CPU cycle complete.
11 The next user program cycle of *Prg 1* starts.
12 *Max. Duration for Each Cycle [µs]* of *Prog 1* expired. Next user program cycle of *Prog 2* starts.
13 *Max. Duration for Each Cycle [µs]* of *Prog 2* expired. *Prg 3* resumed.
14 User program cycle *Prg 3* complete. Standby time until expiration *Max. Duration for Each Cycle [µs*] of *Prg 3*. Third CPU cycle complete.

i In the examples illustrating the multitasking modes, input and output processing are represented as empty spaces at the beginning and the end of each CPU cycle.

The multitasking mode can be set using the resource parameter *Multitasking Mode*, see Table 9.

5.4.4 Typical Response Time

A safety-related response of the HIMax system must occur within a period of time that is shorter than the process safety time determined by the process. The process safety time must be determined in line with a hazard and risk analysis performed by the operator. The process safety time is usually documented in the safety requirements specification

The response time considered here only refers to the HIMax resource!

Delays such as sensor and actuator switching times, which must be considered for the response time of the entire safety function (SIF), but take effect outside the HIMax system, are not taken into account here.

The typical *Response Time* is composed of the following times:

- Time requirements in the input module:
  - I/O cycle time 2 ms.
  - Configured delay (TON, TOFF).
  - For details on additional hardware delays (e.g., setting time in connection with analog inputs), refer to the corresponding manual.
- Time requirements in the processor module:

  Max. 2 * cycle time
- Time requirements in the output module:

  2 * I/O cycle time

The cycle of a processor module includes the following activities in chronological order.

Phase	Activity	Time requirements depending on
1	Self-Tests	
2	Processing of configuration connections	System bus structure, number of remote I/Os
3	Reading the inputs through communication	Number and data volume of communication connections
4	Reading the physical inputs	Number of physical inputs
5	Processing of the user program	User program structure and size
6	Writing to the physical outputs	Number of physical outputs
7	Writing to the outputs through communication	Number and data volume of communication connections
8	(Watchdog reserve time: Residual time until the watchdog time has expired)	(no time requirements, reserve for reload, synchronization)

Table 22: Structure of a Processor Module Cycle

The time required for phase 2 is determined by the value set for *Max. Duration of Configuration Connections [ms]*, see Chapter 5.2.4.5.

The time required for phases 3 and 7 is determined by the value set for *Max.Com. Time Slice ASYNC [ms]*.

Additional requirements for the typical response time are:

- The system operates without interference.
- No noise blanking is effective.

Refer to the HIMax safety manual (HI 801 003 E) for details that must be taken into account for a HIMax resource.

Depending on the conditions, the response time can either be twice the CPU cycle time or the safety time.

Double CPU cycle time as response time

For systems operating in low demand mode, **twice** the value of the **CPU cycle time** can be taken as **response time,** if all the following conditions are met.

Low demand mode means that a system's safety function is performed on demand and the frequency of demands is not greater than once per year.

The following conditions must be met:

- The system is free from faults occurring at a regular basis and able to trigger noise blanking (noise blanking can delay the response to a detected fault up to the safety time). To this end, an effective diagnostic function must be implemented and repair must be ensured within the repair time used in the PFD calculation.
- The HIMax system is free from frequent load peaks, which are triggered by:
  - User programs:e.g., by using enable inputs (EN) in function blocks or program loops.
  - Communication (static communication load).
- The HIMax is a delay-free system:
  - Delays configured in the user program, e.g., the timer function blocks TON.
  - The system bus latency is set to the default value 0.
  - The program logic execution order is determined such that each input can be read within a cycle before the outputs are written to.
- Load peaks triggered by user interventions are very rare:
  - Reload.
  - Synchronization of processor modules.

- There is just one user program (no multitasking).

Note: If the actual timing behavior is not known or frequent load peaks (e.g. due to reloads, user programs or communication) cannot be reliably precluded, the **watchdog time** is to be considered as the maximum cycle time!

Otherwise, the actual cycle time monitored in the Control Panel over a long period of time (reset the statistics prior to starting monitoring!) can be adopted.

In either case, the requirement for a fault-free system remains!

Safety time as response time

In systems operating in high demand or continuous mode, or for which one of the previously mentioned conditions is not met, the maximum response time may not exceed the safety time.

High demand mode means that a system's safety function is performed on demand and the frequency of demands is greater than once per year.

For high demand mode, further considerations are necessary. Those considerations must be performed together with the consulted test laboratory.

Conclusion

The mentioned boundary conditions determine which of the following times can be used in the calculation of the response time:

- The **cycle time actually monitored**.
- The **watchdog time**.
- The **safety time**.

When in doubt, a conservative approach should be chosen.

The decision must be justified to the authority responsible for the acceptance test.

5.5 Loading User Programs

SILworX can be used to load the project configuration with the user programs into the controller. Two load variants exist:

- Download
  Load of a new project configuration with interruption of safety-related operation.
- Reload
  Load of a changed project configuration without interruption of safety-related operation.

i HIMA recommends performing a data backup of the project configuration, e.g., on a removable medium, after loading a user program into the controller.

This will ensure that the project data matching the configuration in the controller remains available even if the PADT fails.

HIMA recommends performing a data backup on a regular basis, independently from loading the user program.

5.5.1 Download

Requirements for download:

- The controller is in the STOP state.
- The resource enable switch is set to Load Allowed.

After the download, start the user program using SILworX to launch safety-related operation.

Use the download function to load a new program into the controller or if one of the conditions mentioned in the next section precludes the use of the reload function.

5.5.2 Reload

Requirements for reload:

- The controller is in the RUN state.
- The enable switch Reload Allowed is set to TRUE.
- The system variable *Reload Deactivation* is set to FALSE.

i

Reload can be performed with one or several processor modules.

During a reload, the PADT cannot be used to access the controller!

Exceptions:

Abort of the reload is possible as well as the change of the watchdog time and the target cycle time to enable reload.

If a user program already running in a controller is modified, a reload can be performed to load the modified version into the controller. While the previous version of the user program is still running, the new version is stored in the controller memory, checked and provided with the variable values. Once the preparation steps are completed, the controller adopts the new user program version and continues safety-related operation seamlessly.

During a reload, the global and local variables obtain the values of the corresponding variables from the previous project version. The names of local variables contain the POU instance names.

If names are changed and loaded into the controller by performing a reload, this procedure has the following consequences:

- Renaming a variable has the same effect as deleting the variable and creating a new one, i.e., it results in an initialization process. This also applies to retain variables. The variables lose their current value.
- Renaming a function block instance results in initializing all the variables, including retain variables, and all the function block instances.
- Renaming a program results in initializing all its variables and function block instances.

Similarly, moving parts of the program logic is like deleting them from a user program and adding them to another location.

This behavior may have unintended effects on one or multiple user programs and therefore on the plant to be controlled!

The following factors restrict the potential that a changed program is loaded into the controller:

- The restrictions described in Chapter 5.5.2.1.
- The time required to perform a reload.

Since time is required to process the additional reload tasks, the cycle takes longer. To prevent that the watchdog triggers and the controller enters the ERROR STOP state, both SILworX and the controller verify the additional time required to perform a reload. If the required time is too long, a reload is rejected.

i When configuring the watchdog and target cycle times, plan sufficient time reserve for the reload.

HIMA recommends following the procedure for determining the watchdog time described in the safety manual (HI 801 003 E).

The watchdog and target cycle times can be increased just for the duration of the reload, refer to the SILworX online help for more details. This may be necessary if the defined time reserve is too short and the reload process is blocked in the Cleanup phase.

The online function can only be used to the increase the watchdog and target cycle times, and not to reduce them below the values configured in the project.

i **Observe the following points when reloading sequence chains:**

The reload information for sequence chains does not take the current sequence status into account. The sequence can therefore be changed and set to an undefined state by performing a reload. The user is responsible for this action.

Examples:

- Deleting the active step. As a result, no chain step has the *active* state.
- Renaming the initial step while another step is active.
  As a result, a sequence has two active steps!

i **Observe the following points when reloading actions:**

During the reload, actions are loaded with their complete data. All potential consequences must be carefully analyzed prior to performing a reload.

Examples:

- If a timer action qualifier is deleted due to the reload, the timer expires immediately. Depending on the remaining settings, the *Q* output can therefore be set to TRUE.
- If the status action qualifier (e.g., the *S* action qualifier) is deleted for a set element, the element remains set.
- Removing a *P0* action qualifier set to TRUE actuates the trigger function.

5.5.2.1 Conditions for Using the Reload Function

A reload can be performed to load the following project changes 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 in accordance with Table 23.

Changes to	Type of change			
	Add	Delete	Change of the initial value	Assignment of other variables
Assigning global variables to				
User programs	•	•	•	•
System Variables	•	•	•	•
I/O channels	•	•	•	•
Communication protocols [3]	•	•	•	•
safe**ethernet** [1]	•	•	•	•
Rack with system bus and I/O modules	•	•	n. a.	n. a.
Modules (I/O, system bus, and processor modules)	•	•*	n. a.	n. a.
Communication protocols [3]	•	•	n. a.	n. a.
User programs	•	•**	n. a.	n. a.
Event definitions [2]	•	•	n. a.	• (Event states)

Changes to	Changes
Rack names	• [3]
Module names	•, System bus modules and communication modules: • [3]
System ID, rack ID	-
safe**ethernet** target addresses (IP addresses)	• [1]
User accounts and licenses	•
Limits and hysteresis for scalar event definitions	•
Processor modules: ▪ IP configuration ▪ Routings ▪ Switch configuration ▪ Adding, changing, deleting communication protocols	• [3]
System bus modules: ▪ IP configuration ▪ Routings ▪ Power supply and temperature monitoring ▪ Setting *Minimum Configuration Version* ▪ Module name	• [3]
Communication modules	Refer to the communication manual (HI 801 101 E)

- • Reload possible
- \- Reload not possible
- * Reload possible, except for system bus modules in which the *Responsible* attribute is activated
- ** Reload possible, but the controller must still contain at least one user program

n.a.: not applicable

[1] For details on how to load changes via reload in connection with safe**ethernet**, refer to the communication manual (HI 801 101 E)

[2] The event source of an event definition may not be modified by performing a reload, i.e., the identification number cannot be used again through a reload.

[3] By performing a cold reload, i.e., restarting the module

Table 23: Reload after Changes

A reload may only be performed in accordance with the conditions mentioned in the previous section. In all the other cases, stop the controller and perform a download.

5.5.2.2 Cold Reload

In certain cases, a reload cannot be executed for an individual module:

- The conditions designated with 3) in Table 23.
- Configuration changes within communication and system bus modules.
- The communication module is operating with standard protocols that are not capable of reload.

For these cases, a cold reload must be performed. The concerned module must be in STOP (cold) while the reload process is running and must be restarted upon completion of the process.

- Communication modules and system bus modules are stopped and restarted by the reload process.
- When processor modules are being reloaded, a message appears, asking the user to stop or start the module.

Either way, the user maintains the control of the reload process and is informed about its progress. If necessary, the user can abort the reload process.

Prior to performing a cold reload, consider the following points:

- Which modules might be stopped?
- Are redundant modules configured for the modules potentially concerned, or which of the functions they perform can be temporarily renounced?

TIP Proceed as described below to avoid a cold reload in situations in which global variable assignments are added:

- When creating the configuration, already assign unused global variables to communication protocols.
- Assign safe value as initial value to unused global variables.

In doing so, there is no need later to add variables, but only to change their names, which allows a reload to be performed.

5.5.2.3 Restrictions for Reload

The reload of a user program may become impossible if the following conditions apply:

- The number of function and function block instances (POUs) in SILworX is limited to 21 845. Loading of a user program containing more instances cannot be performed through a reload.
- Large arrays with user-defined structure elements can no longer be loaded through the reload when the structure is changed.

  Example: Adding/deleting a structure element in/from a structure data type.

In both cases, SILworX indicates that the maximum number of transfer operations (65 536) was exceeded.

Workaround

To reduce the number of transfer operations, the following is possible:

- Avoid excessively structured programs, e.g., reduce the number of POUs.
- Avoid large structure data types and large arrays of structures.
- A simple workaround is to avoid using VAR in POUs with many instances.

  Replacing all POU VAR with VAR_OUTPUT reduces the number of transfer operations, but also has the following effects:
  - Number of POU outputs is increased.
  - Forcing the variables is not possible.

5.6 User Management

The user management is used for project-specific management of user groups, user accounts, access permissions for the PADT and access permissions for the controllers (PES).

A user group consists of one or more user accounts. Opening a project in SILworX may require a login with the data of a user account. In addition, the user group can be allowed to access resources.

Exactly one user management can be created in a project. The user management applies for this project and is saved in encrypted form in the project. The user management name is assigned by SILworX and is language-dependent.

A user group is a collection of users with identical access permissions for operation of the PADT or for access to the controllers.

PADT User Management	The access to the SILworX project are managed in the PADT user management.
PES User Management	The PES-related access permissions are managed in the PES user management. The PES user management is based on the data in the PADT user management.

⚠ WARNING

If no user management scheme is configured, any kinds of manipulations of the project or the resource are possible!

Safe operation of a system calls for a project to employ a user management with several different user groups and different access modes. There must be a number of user accounts defined in the user groups.
Otherwise, anyone can open the project in SILworX and log themselves in to resources as the default user.

If no user management is available, a project is opened without a login dialog. Otherwise, a dialog box for a PADT user query is displayed and there is a prompt to enter a user name and a password.

If the project was created with an earlier version of SILworX and a project conversion is required, the project is only converted if the logged-in user has at least Write access. Otherwise, the project is not converted and is not opened.

When the user management is created, an initial security administrator is also automatically added to the PADT user management and the parameters are configured. After this, changes can only be made to the user management by this or another security administrator.

5.6.1 Standard Access Permissions

As long as no user management has been set up for a project, no *User Management* element is displayed in the structure tree. The factory-specified access permissions apply for both the project opened in SILworX and the resources.

	PADT	Resource (controller)
Number of users	1	1
User ID	Security administrator	Administrator
Password	None	None
Access permission	Security administrator	Administrator

Table 24: Factory Access Permissions for PADT and PES

i The factory-specified access permissions are disabled when a user management is created in the project.

5.6.1.1 The Security Administrators User Group

Security administrators have all access permissions. SILworX automatically creates a user group with the *Security Administrator* access mode as soon as a user management is created. By default this user group contains one (1) user account, and additional user accounts can be added.

This ensures that there is always at least one security administrator present in the user administration.

The user account created by default and all other user accounts added to this user group later on have the *Security Administrator* access mode.

This access mode is only valid for editing the project in SILworX. The security administrators user group can also have a different access mode with reduced privileges in the *PES User Management*.

i Security administrators have full access to all user accounts. For example, they can change the passwords of these user accounts without knowing the user account's current password. This can become necessary if users forget their password.

5.6.2 Access Modes and Permissions

Depending on the defined access mode and the licenses available, a user will be able to use more or fewer permissions and functions in SILworX or with a resource.

Depending on the user currently logged in, the access permissions can also be restricted via licenses. To do this, the *Maintenance* license option can be used to downgrade the operative permissions for every user to *Read*.

5.6.2.1 PADT User Management Access Modes

The following table summarizes the access modes which can be used in the PADT user management.

Access Mode	Description
Security administrator	The *Security Administrator* access mode allows the user to operate all functions in SILworX which are available with the existing licenses.
Read and Write	The *Read and Write* access mode allows users to execute all functions in SILworX that are are covered by the available licenses and not explicitly bound to the *Security Administrator* privilege.
Read	Users with this access mode can access functions in reading mode only. This means that Archiving is not permitted, for instance. Some types of information are not displayed (e.g., details concerning user management).

Table 25: Access Modes for the PADT User Management

5.6.2.2 PES User Management Access Modes

The following table summarizes the access modes which can be used in the PES user management. A maximum of 5 PES users can simultaneously be logged into a controller.

Access Mode	Description
Administrator	**Activity**: Putting a new system into operation. Within the scope of application of the resource, administrators have all of the permissions which are available with the existing licenses. - Actions required to put modules or systems into operation. - Setting up a replaced controller. - Start-up of safety devices.
Read and Write	**Activity**: Modifying an existing system. The *Read and Write* access mode allows users to execute all functions on a resource that are required to make modifications and are covered by the available licenses. Users can change the online settings for which modification has been enabled, but not the setting of the *Allow Online Settings* parameter itself.
Read and Operator	**Activity**: Maintaining a running system in operation. The *Read and Operator* access mode allows users to carry out all actions which are required to maintain a running system. No modifications to the programmed functions are possible. The permissions here are downgraded once again as compared to *Read and Write* and are by and large restricted to the following activities: - All actions which are necessary for diagnostic purposes. - Actions which are required to start and to force a system.
MultiForcing	**Activity**: Forcing by several users simultaneously. Users with *MultiForcing* access can write force data (force values and individual force switches) for global variables in a resource if the required higher-order conditions have been met and the force permissions have been granted. The use of preconfigured watchpages is also possible. To all other functions of a resource, users have Read-Only access. Starting, stopping or resetting a force process is not possible. To modify watchpages and save them in the project, users require the corresponding access permissions in the PADT (*Read and Write* or *Security Administrator*).
Read	**Activity**: Reading the diagnostics. Users with this access mode can access functions in reading mode only. The *Read* access mode is assigned automatically during a system login to a resource when there is already another user logged in with Write access permissions.
No Access	**Activity**: None, login is locked. This is the default setting for all user groups for which no access mode has been explicitly set in the assignment table. A blank table cell is displayed for *No Access*.

Table 26: Access Modes for the PES User Management

5.6.3 Creating the PADT User Management

Refer to the SILworX online help for furhter details on how to create the PADT user management.

5.6.4 Creating the PES User Management

The PES user management automatically displays the user groups created in the PADT user management. Access permissions for the created user groups can be assigned to the resources in the PES user management. Refer to the SILworX online help for details.

User accounts for up to 10 user groups can be managed in the PES user management of each controller. The user data is available for login once the project has been loaded in the controller through a download. The user accounts are stored in the controller and still apply after switching off the operating voltage. The user accounts of a controller also apply to the connected remote I/Os.

Users log in to a controller using the user group name and password. The values for PES user group name and password are not those currently entered in the project, but those transferred to the PES during the last loading process!

6 Diagnostics

The diagnostic LEDs are used to give a first quick overview of the system state.
The diagnostic history in SILworX provides detailed information.

6.1 Light Emitting Diodes (LEDs)

LEDs on the front plate indicate the module state. All LEDs should be considered together. A single LED is not sufficient to assess the module state.

A description of the LEDs is provided in the module-specific manuals.

After connecting the supply voltage, an LED test is performed and all the LEDs are lit for at least 2 s. The color of two-color LEDs changes once during the test.

Definition of blinking frequencies

The following table defines the blinking frequencies:

Definition	Blinking frequencies
Blinking1	Long (600 ms) on, long (600 ms) off.
Blinking2	Short (200 ms) on, short (200 ms) off, short (200 ms) on, long (600 ms) off.
Blinking-x	Ethernet communication: Blinking synchronously with data transmission.

Table 27: Blinking Frequencies of the LEDs

Some LEDs can report warnings (On) and faults or errors (Blinking1). The indication of errors or faults has priority over the indication of warnings. Warnings cannot be reported if errors or faults are being signaled.

6.2 Diagnostic History

Each HIMax module keeps a chronological history of the faults and events that have occurred. The history is organized as a ring buffer.

The diagnostic history is composed of short-term and long-term diagnostics.

- Short-term diagnostics

  If the maximum number of entries has been reached, each new entry deletes the oldest entry.
- Long term diagnostics:

  The long term diagnosis essentially stores actions and configuration changes performed by the user.
  If the maximum number of entries has been reached, each new entry deletes the oldest entry if this is older than three days.
  The new entry is rejected if the existing entries are not older than three days. The rejection is marked by a special entry.

The number of events that can be stored depends on the type of module.

Module type	Max. number of events long term diagnosis	Max. number of events short-term diagnostics
X-CPU 01 and X-CPU 31	2500	1500
X-COM 01	300	700
I/O Modules	400	500
X-SB 01	400	500

Table 28: Maximum Number of Entries Stored in the Diagnostic History per Module Type

i The diagnostic entries can be lost if a power outage occurs just before they could be saved into non-volatile memory.

SILworX can be used to read the histories of the individual modules and represent them so that the information required to analyze a problem is available:

Example:

- Mixing the histories from various sources
- Filtering by time period.
- Printing out the edited history
- Saving the edited history

For additional functions, see the SILworX online help.

6.2.1 I/O Module Diagnostic Message

A diagnostic message for I/O modules is structured as follows:

IO ERROR >> slot S I/O module type: MMMM status[Mod: mm OUT: AAAA IN: EEEE] channel[OUT:aaaa IN:eeee] <<

The following table describes the data fields in the message.

Data field	Format	Description
S	Decimal	Slot number of the I/O module.
MMMM	Hexadecimal	Module type.
mm	Hexadecimal	Status of the module.
AAAA	Hexadecimal	Code for faults in the module's outputs.
EEEE	Hexadecimal	Code for faults in the module's inputs.
aaaa	Hexadecimal	Code for channel faults in the output channels.
eeee	Hexadecimal	Code for channel faults in the input channels.

Table 29: Data Field of Diagnostic Message

For further details on the error codes, refer to the corresponding manuals. If multiple channels are faulty, the data field aaaa / eeee contains an OR gate with 0x8000, e.g., the most significant bit is set to 1 in addition to the error code.

The module type can be determined in the Hardware Editor.

6.3 Online Diagnosis

The online view in the SILworX Hardware Editor is used to diagnose failures in the HIMax modules. Failed modules are signalized by a color change:

- Red indicates severe failures, e.g., that the module is not inserted.
- Yellow indicates less severe failures, e.g., that the temperature limit has been exceeded.

Point to a module to display a tooltip providing the following state information on the module:

<table>
<tr><th>Information</th><th>Representation</th><th>Range of values</th><th>Description</th></tr>
<tr><td>SRS</td><td>Decimal numbers</td><td>System: 0…65535
Rack: 0…16
Slot: 1…18</td><td>Module identification: System, Rack, Slot</td></tr>
<tr><td>Name</td><td>Text</td><td></td><td>Designation of the information, here always: Online MODULE Information.</td></tr>
<tr><td>Module State</td><td>Text</td><td>RUN, STOP, NOT CONNECTED, Unknown, ...</td><td>State in which the module is operating.</td></tr>
<tr><td>Plugged Module Type</td><td>Text</td><td>Permissible module types</td><td>Type of module which is plugged into the rack.</td></tr>
<tr><td>Configured Module Type</td><td>Text</td><td>Permissible module types</td><td>Type of module which is configured and loaded in the controller.</td></tr>
<tr><td>Module Type in Project</td><td>Text</td><td>Permissible module types</td><td>Type of module which is being projected as the digital depiction.</td></tr>
<tr><td>Connection Status of the Protocol</td><td>Hexadecimal value</td><td>16#00…0F</td><td>Status of the connection between each of the processor modules (max. 4). Each of the bits 0...3 shows the connection to the processor module with the corresponding index.
Bit x = 0: Not connected.
Bit x = 1: Connected.</td></tr>
<tr><td>Interface Send Status</td><td>Hexadecimal value</td><td>16#0000…FFFF</td><td>Every two bits represent the state of one interface identified with an index 0...16. Bits 0 and 1 apply to interface 0, and so on.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>00</td><td>No message has been received or sent yet, unknown status.</td></tr>
<tr><td>01</td><td>OK, no faults.</td></tr>
<tr><td>10</td><td>Last data received or sent was defective.</td></tr>
<tr><td>11</td><td>No faults during last reception/transmission, one fault occurred before.</td></tr>
</table></td></tr>
<tr><td>Interface Receive Status</td><td>Hexadecimal value</td><td>16#0000…FFFF</td><td>Every two bits represent the state of one interface identified with an index 0...16. Bits 0 and 1 apply to interface 0, and so on.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>00</td><td>No message has been received or sent yet, unknown status.</td></tr>
<tr><td>01</td><td>OK, no faults.</td></tr>
<tr><td>10</td><td>Last data received or sent was defective.</td></tr>
<tr><td>11</td><td>No faults during last reception/transmission, one fault occurred before.</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Information</th><th>Representation</th><th>Range of values</th><th>Description</th></tr>
<tr><td>Module Error Status</td><td>Hexadecimal value</td><td>16#00…3F</td><td>Bit-coded module status:
<table>
<tr><th>Bit</th><th>Description for value = 1</th></tr>
<tr><td>0</td><td>Warning related to external communication.</td></tr>
<tr><td>1</td><td>Warning related to field connection.</td></tr>
<tr><td>2</td><td>System warning.</td></tr>
<tr><td>3</td><td>External communication error.</td></tr>
<tr><td>4</td><td>Field connection error.</td></tr>
<tr><td>5</td><td>System error.</td></tr>
<tr><td>6</td><td rowspan="2">Not used.</td></tr>
<tr><td>7</td></tr>
</table></td></tr>
<tr><td>Connection status of system bus A</td><td>Hexadecimal value</td><td>16#0…3</td><td>Status of the interface to system buses A:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>The interface is OK.</td></tr>
<tr><td>1</td><td>The interface detected an error during last reception, now it is OK.</td></tr>
<tr><td>2</td><td>An error occurred on the interface.</td></tr>
<tr><td>3</td><td>The interface is switched off.</td></tr>
</table></td></tr>
<tr><td>Connection status of system bus A</td><td>Hexadecimal value</td><td>16#0…3</td><td>Status of the interface to system buses A:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>The interface is OK.</td></tr>
<tr><td>1</td><td>The interface detected an error during last reception, now it is OK.</td></tr>
<tr><td>2</td><td>An error occurred on the interface.</td></tr>
<tr><td>3</td><td>The interface is switched off.</td></tr>
</table></td></tr>
</table>

Table 30: Diagnostic Information Displayed in the Online View for the Hardware Editor

Due to the timing behavior of the operating system, it is possible that the safety parameters are not displayed in the diagnostics. To allow the safety parameters to be displayed in the diagnostics, proceed as follows when opening the diagnostics:

1. Open the Control Panel and wait for all the fields to be refreshed.
2. In the Hardware Editor, open the diagnostics using the context menu of the online view and not of the detail view!

Prior to opening the diagnostics, do not open the detail view and open as few online views as possible (e.g., Force Editor, online test)!

7 Product Data, Dimensioning

This chapter specifies the environmental requirements and the dimensions to be set in the SILworX programming tool.

7.1 Environmental Requirements

Exposing the HIMax system to environmental conditions other than those indicated can cause it to malfunction. Additionally, the instructions provided in the module-specific manuals must be observed.

General	
Protection class	Protection class II in accordance with IEC/EN 61131-2
Ambient temperature	0…+60 °C
Transport and storage temperature	-40... +70 °C
Pollution	Pollution degree II in accordance with IEC/EN 60664-1
Installation height	< 2000 m
Enclosure	Standard: IP20 If required by the relevant application standards (e.g., EN 60204), the HIMax system must be installed in an enclosure with the specified degree of protection (e.g., IP54).
Power Supply Input Voltage	24 VDC, -15...+20 %, $r_p \leq 5$ % SELV, PELV

Table 31: Environmental Requirements

Exposing the HIMax system to environmental conditions other than those indicated can cause the HIMax system to malfunction.

7.2 Dimensioning

For details, refer to the component-specific manuals and communication manual (HI 801 101 E).

For each resource	Value from ... to
Number of racks	1…16
Number of I/O modules	
X-CPU 01	0…200
X-CPU 31	0…64
Number of I/O elements (sensors, actuators)	Depending on the module type; in this case, for modules with 32 inputs or outputs:
X-CPU 01	0…6400
X-CPU 31	0…2048
Maximum length of a system cable to FTA	30 m
Number of processor modules	
X-CPU 01	1…4
X-CPU 31	1…2
Total program and data memory for all user programs	
X-CPU 01	10 MB less 4 kB for CRCs
X-CPU 31	5 MB less 4 kB for CRCs
Memory for retain variable	32 kB
Memory available for global process data	512 kB
Number of system bus modules, per rack	1…2

For each resource	Value from ... to
Maximum length of system buses, when the default latency setting are used	1500 m
Using fiber optic cables (see Chapter 3.2)	19.6 km
With higher latency values, longer lengths are possible, see Chapter 3.2.3	
Number of communication modules	
X-CPU 01	0…20
X-CPU 31	0…4
Number of safe**ethernet** connections	0…255
Number of safe**ethernet** connections between two resources of the following types: ▪ HIMax ▪ HIMatrix F30 03, F35 03 or F60 with CPU 03 with CPU operating system V10 and higher, and COM operating system V15 and higher	0…64
safe**ethernet** buffer size for each connection	
▪ Connection to another HIMax controller. ▪ With HIMatrix controllers F30 03, F35 03, F60 CPU 03	1100 bytes for each direction
Connection to other HIMatrix controller	900 bytes for each direction
Connection buffer size to an X-OPC Server	128 kBytes
Number of PES user groups	1…10
Number of user programs	1…32
Number of event definitions	0…20 000
Size of the non-volatile event buffer	5000 Events

Table 32: Dimensioning of a HIMax Controller

Detailed specifications are provided in the manuals for the individual components and in the communication manual (HI 801 101 E).

8 Lifecycle

This chapter describes the following lifecycle phases:

- Installation.
- Start-up.
- Maintenance and repairs.

Instructions for a correct decommissioning and disposal of the products are provided in the manuals for the individual components.

8.1 Installation

This chapter describes how to install and connect the HIMax controllers.

8.1.1 Mechanical Structure

To ensure faultless operation, choose a suitable mounting location for the HIMax system in accordance with the conditions of use, see Chapter 2.1.

Observe the instructions for installing base plates and other components specified in the corresponding manuals.

8.1.2 Connecting the Field Level to the I/O Modules

HIMax is a flexible system designed for continuous operation. It allows the I/O modules to be connected to the field level:

- Directly, via the connector boards.
- Indirectly, via the FTAs.

The following section describes the four recommended wiring variants.

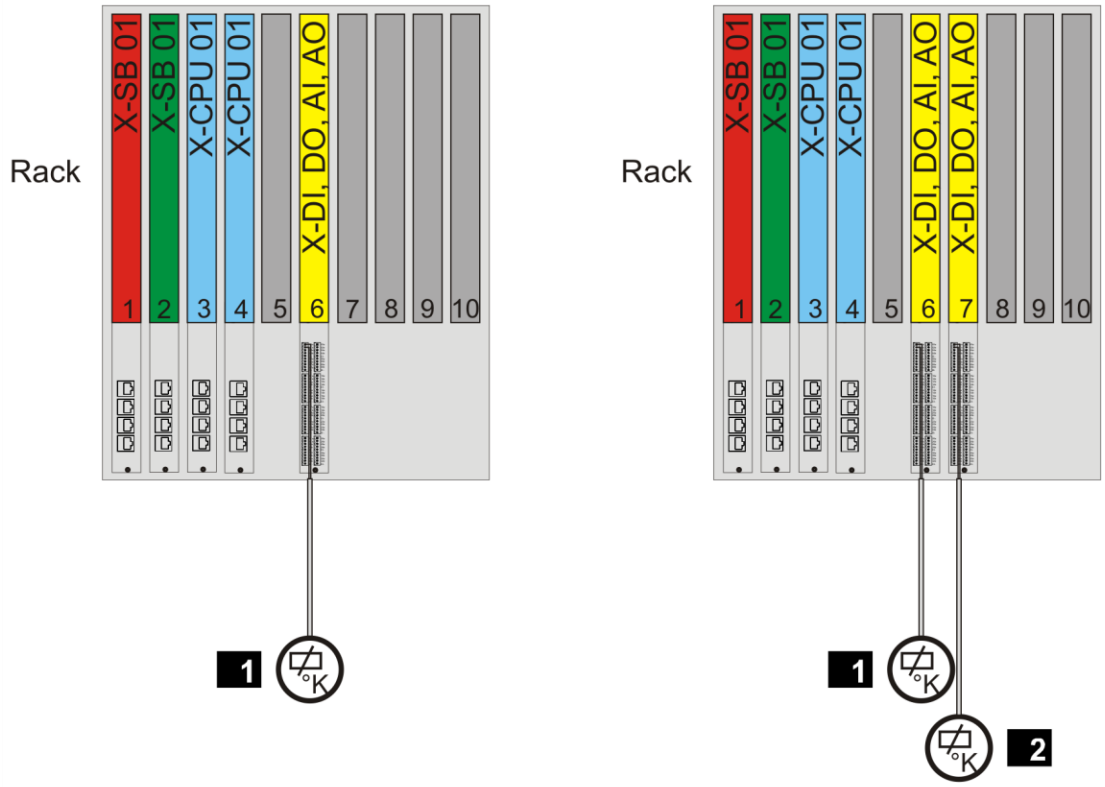
1. Connection to mono connector boards with screw terminals.
2. Connection to redundant connector boards with screw terminals.
3. Connection to mono connector boards via FTA and system cables.
4. Connection to redundant connector boards via FTA and system cables.

Additional wiring options require higher planning efforts and are not described in the manuals. If required, HIMA recommends contacting HIMA's experts in the Global Project Management and Engineering department.

8.1.2.1 Wiring 1

Connection of sensors or actuators to a mono connector board with screw terminals for individual I/O modules.

- Connect each sensor or actuator to a channel of an individual, non-redundant I/O module.
- Connect two or more redundant sensors or actuators to individual channels of two or more redundant I/O modules. The number of redundant sensors or actuators must be identical to the number of redundant modules (e.g., two sensors/two modules).



1 Sensor or actuator.

2 Redundant sensor or actuator.

Figure 19: Wiring 1 - Mono Connector Board with Screw Terminals

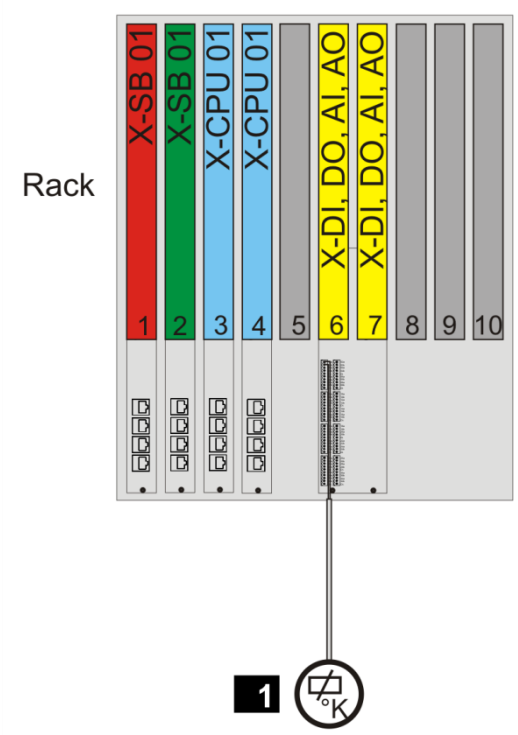
For wiring 1, connector boards of type 01, e.g., X-CB 008 01, are required in the base plate.

8.1.2.2 Wiring 2

Connection of sensors or actuators to a redundant connector board with screw terminals. The connector board distributes the signals from one sensor to two redundant modules or merges the signals from two redundant modules to one actuator.

For this wiring, a redundant system bus and redundant power supply must be ensured.

- Connect each sensor or actuator to a channel of a redundant connector board, on which the I/O modules are mounted adjacently.



1 Sensor or actuator.

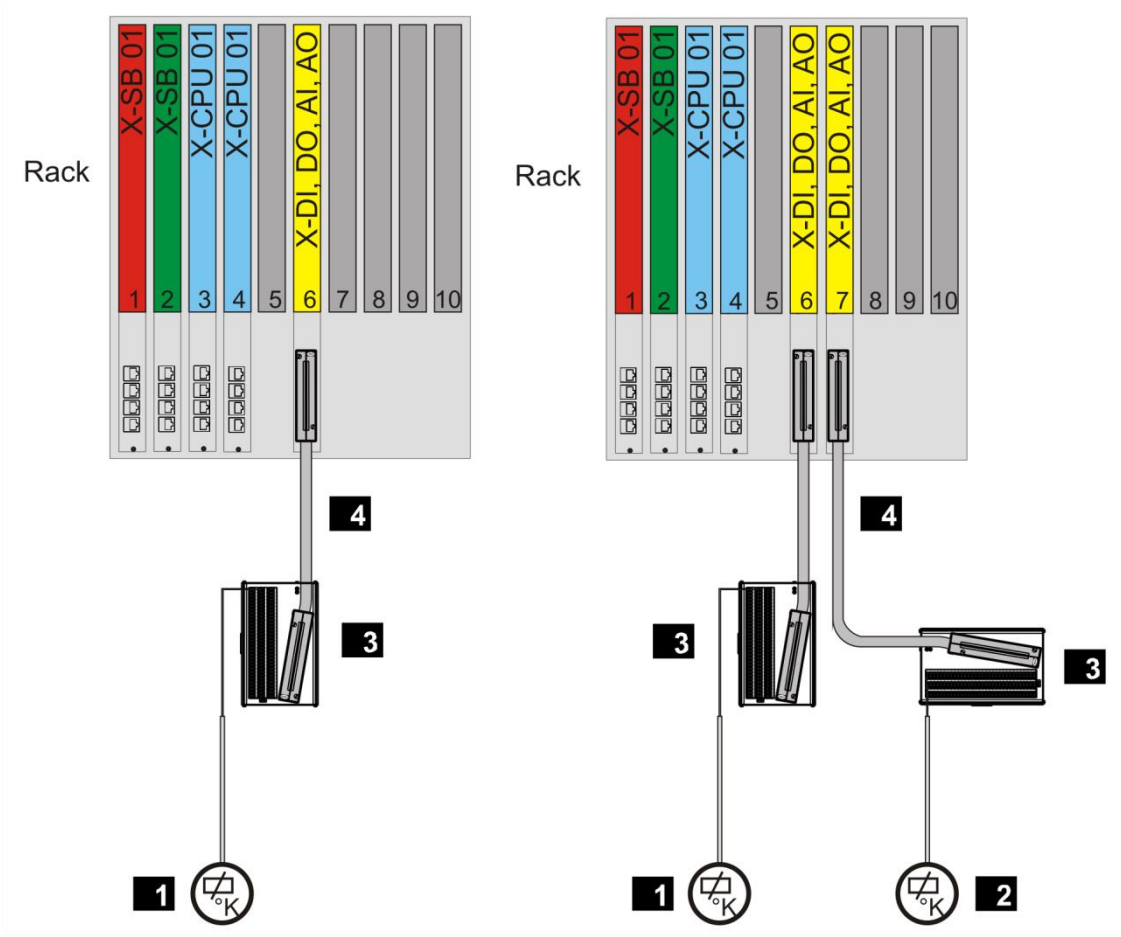
Figure 20: Wiring 2 – Redundant Connector Board with Screw Terminals

For wiring 2, connector boards of type 02, e.g., X-CB 008 02, are required in the base plate.

8.1.2.3 Wiring 3

Connection of sensors or actuators to a mono connector board with cable plug via FTA.

- Connect each sensor or actuator to a channel of an FTA.
- Connect two or more redundant sensors or actuators to individual channels of two or more redundant FTAs. Connect each FTA to a mono connector board via a system cable. The number of redundant sensors or actuators must be identical to the number of redundant modules (e.g., 2 sensors/2 modules).



1 Sensor or actuator.
2 Redundant sensor or actuator.
3 FTA.
4 System cable.

Figure 21: Wiring 3 – Mono Connector Board with System Cable

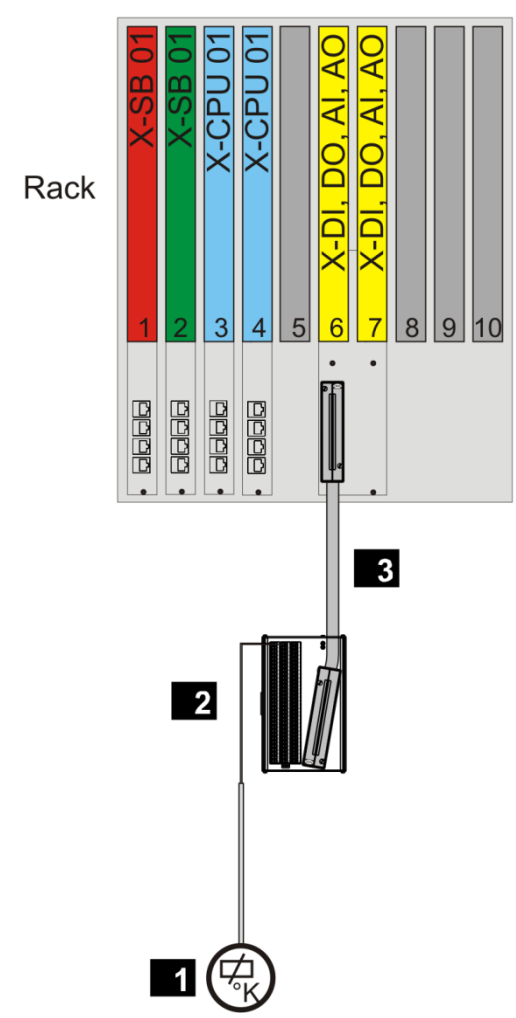
For wiring 3, connector boards of type 03, e.g., X-CB 008 03, are required in the base plate.

8.1.2.4 Wiring 4

Connection of sensors or actuators to a redundant connector board with cable plug via FTA and system cable. The connector board distributes the signal from one sensor to two redundant modules or merges the signals from two redundant modules to one actuator.

For this wiring, a redundant system bus and redundant power supply must be ensured.

Connect each sensor or actuator to a channel of a redundant connector board via an FTA. In doing so, insert the I/O modules into adjacent slots.



1 Sensor or actuator.
2 FTA.
3 System cable.

Figure 22: Wiring 4 – Redundant Connector Board with System Cable

For wiring 4, connector boards of type 04 (e.g., X-CB 008 04) are required on the base plate.

8.1.3 Grounding

Observe the requirements specified in the low voltage directives SELV (Safety Extra Low Voltage) or PELV (Protective Extra Low Voltage).
Functional ground is prescribed to improve the electromagnetic compatibility (EMC). Implement the functional ground in the control cabinet so that it meets the requirements for protective ground.

All HIMax systems can be operated with grounded L- or ungrounded.

8.1.3.1 Ungrounded Operation

In ungrounded operation, a single ground fault does not affect the safety and availability of the controller.

If several undetected ground faults occur, faulty control signals can be triggered. For this reason, HIMA recommends using ground fault monitoring for ungrounded operation. Some application standards, e.g., DIN EN 50156-1:2005, prescribe the use of ground fault monitoring. Only use ground fault monitoring devices approved by HIMA.

8.1.3.2 Grounded Operation

Requirements for grounded operation are proper ground conditions and, whenever possible, separate ground connection, in which no parasitic currents may flow. Only the negative pole L- may be grounded. The positive pole L+ must not be grounded since a potential ground fault on the sensor wire would bridge the affected sensor.

L- can only be grounded at one point within the system. L- is usually grounded directly behind the power supply unit (e.g., on the busbar). Grounding should be easy to access and well separate. The grounding resistance must be ≤ 2 Ω.

8.1.3.3 CE-Compliant Structure of the Control Cabinet

In accordance with the EU Council Directive 89/336/EEC, converted in the EMC law for the Federal Republic of Germany, from 1st January 1996, all electrical equipment within the European Union must be labeled with the CE conformity marking for electromagnetic compatibility (EMC).

All modules of the HIMA HIMax system family are labeled with the CE conformity marking.

To prevent EMC issues when installing controllers in control cabinets and support frames, ensure proper and interference-free electrical installation in the vicinity of the controllers, e.g., do not lay power lines together with 24 VDC supply lines.

8.1.3.4 Grounding the HIMA Controllers

While also taking the EMC aspects into account, implement the following grounding measures to ensure the safe function of HIMA controllers.

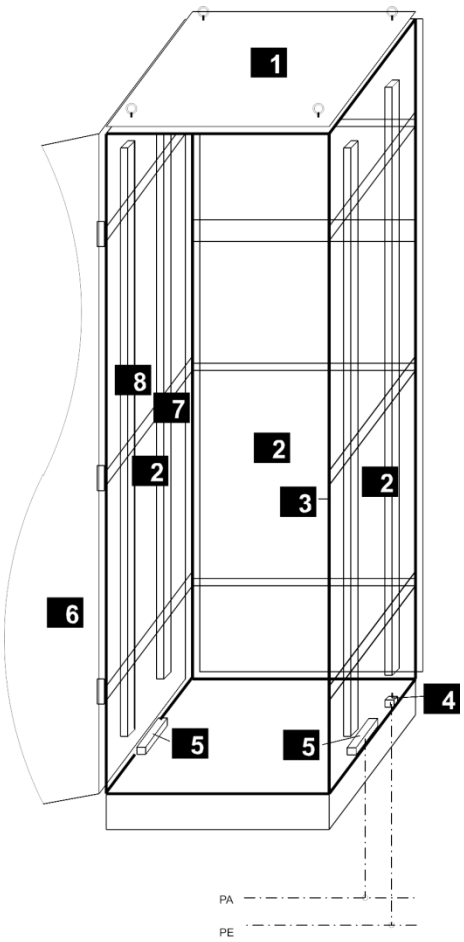
All tangible surfaces of the HIMax components (e.g., base plate), except for pluggable modules, are electrically conductive (ESD protection, ESD = electrostatic discharge). Use cage nuts with claw fasteners to ensure the safe electrical connection of components such as base plates and the control cabinet. The claw fasteners penetrate the components' surface and ensure safe contact. Stainless steel screws and flat washers are used to prevent electrical corrosion.

8.1.3.5 Mounting the HIMax on a Support Frame

The roof sheeting is secured to the cabinet frame with four lifting eyes (see Figure 23). The cabinet frame is electrically connected to the side panels, backplane and floor panel through grounding claw fasteners.

Two M 2500 busbars **5** are installed in the control cabinet as standard equipment and connected to the cabinet frame through 25 mm² round cables. After removing this connection, the busbars **5** can be used for a ground-isolated potential (e.g., for connecting the field cable shielding).

An M8 bolt is located on the cabinet frame to allow customers to connect the protective ground cable **4**.



1 Shielding on the roof sheet connected to the cabinet frame with standard fasteners.

2 Shielding and grounding on the side panels, backplane, floor panels and base plate connected to the cabinet frame with standard fasteners.

3 The cabinet frame serves as the reference ground for the cabinet.

4 Central grounding point for grounding the cabinet frame (M8 bolts).

5 M 2500 busbars isolated from the cabinet ground and mounted on the cabinet frame. These serve as the intake for the equipotential bounding from the external supply and the I/O cables from the field.

6 Shielding and grounding of moveable cabinet parts connected to the cabinet frame with ground straps.

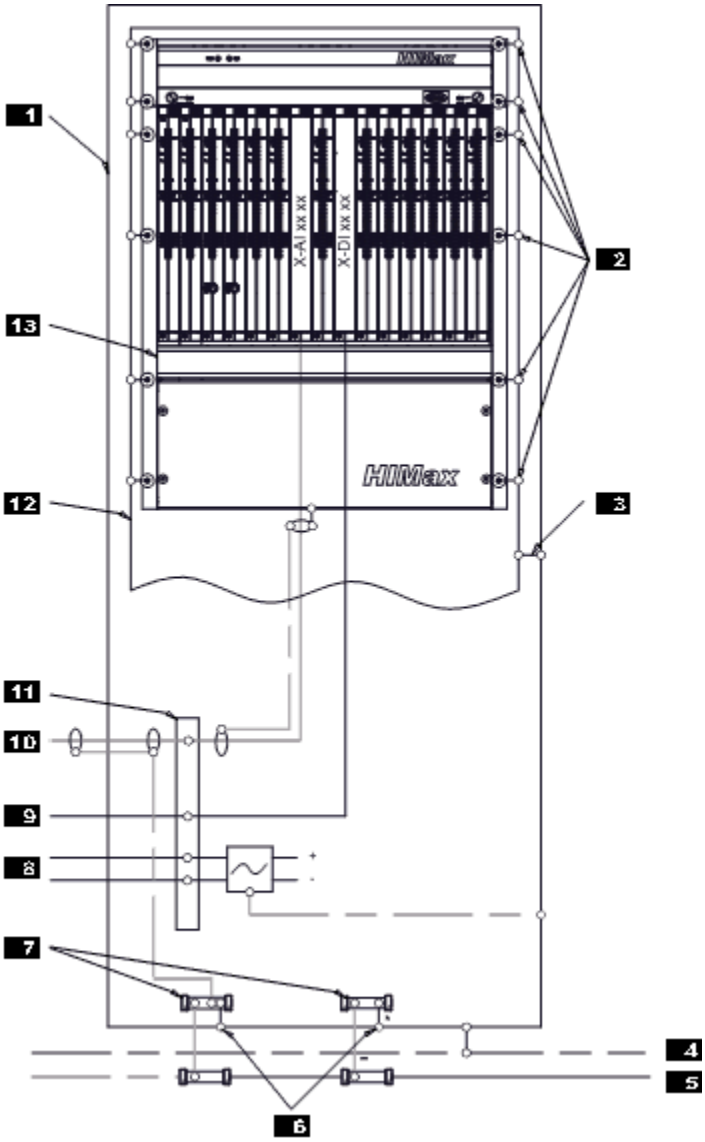
7 Standard fasteners used to ground mechanic parts such as the chassis. The parts are connected to one another and to the cabinet frame. 25 mm^2 ground straps are used to ground the mounting plate.

8 Potential bounding via mounting rails or cable shield rails. Standard case: equipotential bounding via reference ground. The rails must be electrically connected to the chassis or to the mounting plate.

Figure 23: Grounding Connections in the Control Cabinet

When installing devices with a voltage of ≥ 60 VDC or ≥ 42 VAC, a 25 mm grounding strap must be used.

Figure 24 shows the concept of grounding and shielding the 19" control cabinet.

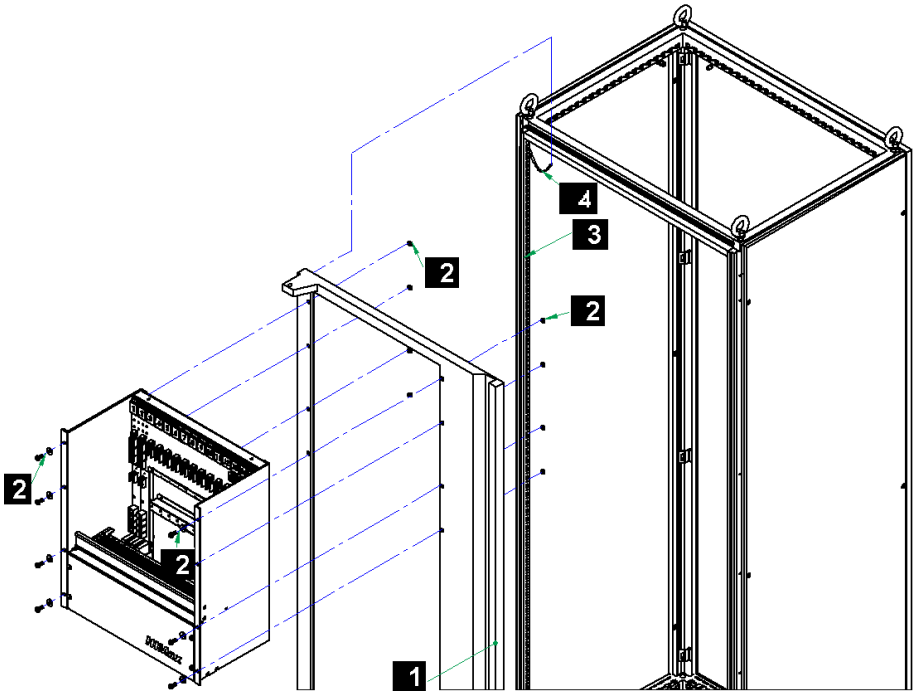


1. Cabinet frame.
2. Base plate attached using cage nuts with claw fasteners.
3. Connection of the pivoting frame to the cabinet frame using 25 mm² grounding straps.
4. PE = protective ground.
5. EB = equipotential bonding.
6. Standard connection to HIMA control cabinets.
7. M 2500 busbar.
8. 24 VDC supply.
9. Digital signals, terminals on field terminal assembly (FTA).
10. Analog signals, terminals on field terminal assembly (FTA).
11. Terminals.
12. Pivoting or fixed frame.
13. Base plate.

Figure 24: Grounding and Shielding the 19" Control Cabinet

8.1.3.6 Mounting the HIMax on a Pivoting Frame

The components of the cabinet frame 3 are welded together and function as electrically conductive construction element. Short grounding straps with cross-sections of 16 mm² or 25 mm² are used to conductively connect the pivoting frames, door and, optionally, mounting plates to the cabinet frame.



1 Pivoting frame.
2 Screws and washers.
3 Cabinet frame
4 25 mm grounding strap.

Figure 25: Grounding Connectors for Base Plate

8.1.3.7 Grounding Connectors

The following table provides an overview of the dimensions of grounding connectors:

Place of installation	Cross-section	Length
Door	16 mm²	300 mm
Pivoting frame (in Figure 25)	25 mm²	300 mm
M 2500 busbar (connection with GN/YE round cables)	25 mm²	300 mm

Table 33: Grounding Connectors

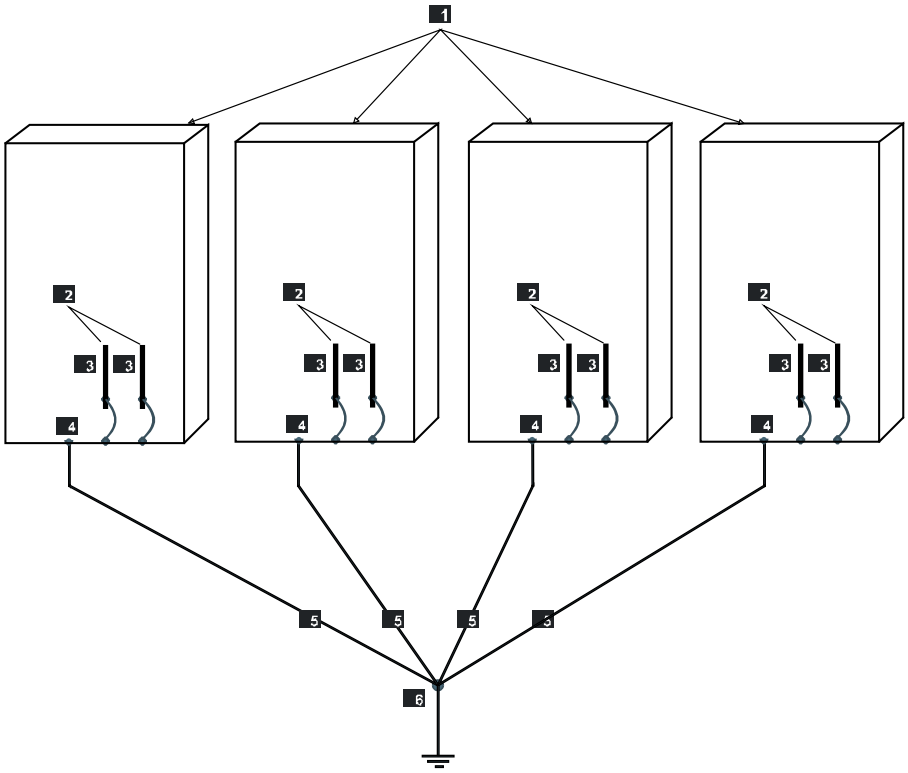
The following elements are relevant for the grounding procedure:

- Fastener terminals used on the side panels, backplane and floor panel
- Central ground point (position 4 in Figure 23)
- Lifting eyes
  The roof sheeting is secured to the cabinet frame with four lifting eyes. Contact disks are used to perform the electrical connection.

Ensure proper installation of grounding connectors!

8.1.3.8 Connecting the Ground Terminals of Several Control Cabinets

Central ground should be as free of interference voltage as possible. If this cannot be ensured, provide the controller with its own ground.



1 Cabinet frame.
2 M 2500 busbar.
3 EB = equipotential bonding.
4 PE = protective ground.
5 At least 16 mm² cross-section.
6 Central ground.

Figure 26: Ground Terminals of Various Control Cabinets

8.1.4 Electrical Connections

8.1.4.1 Shielding within the Input and Output Areas

Lay field cables for sensors and actuators separately from the power supply lines and at a sufficient distance from electromagnetically active devices (electric motors, transformers).

To avoid interferences, ensure that the field cables are provided with continuous shielding. To this end, connect the shielding on both ends of the field cables. This applies, in particular, to field cables of analog inputs and proximity switches.

If high compensation currents are expected, the shielding must be applied on at least one end. Further measures, e.g., galvanic separation, must be implemented to avoid compensation currents.

For details on shielding and grounding requirements, refer to the module-specific manuals.

8.1.4.2 Lightning Protection for Data Lines in HIMA Communication Systems

To minimize problems due to lightning, the following measures can be implemented:

- Completely shield the field wiring of the HIMA communication systems.
- Properly ground the system.

Install lightning protection devices in places outside of buildings and exposed to lightning.

8.1.4.3 Cable Colors

The cable colors used in the HIMax devices comply with international standards.

Notwithstanding the HIMA standard, other cable colors can be used for wiring due to national standard requirements. In such a case, document and verify the deviations.

8.1.4.4 Connecting the Supply Voltage

The supply voltage infeed lines must be connected to the clamp terminals of the rack (L1+, L2+, L1-, L1-).

Attach the supply voltage infeed lines of the system fan to the screw terminals.

When tightening the screw, make sure that the maximum locked-rotor torque specified in the manual for X-BASE PLATE (HI 801 025 E) is not exceeded to ensure compliance with the UL requirements.

8.1.4.5 Connecting Field Devices and Shielding

In connection with I/O modules, either attach the supply lines for the field devices to the screw terminal connector block on the connector boards or on the FTA. In doing so, observe the locked-rotor torque of the screws specified in the module-specific manuals to comply with the UL requirements.

To connect the field devices via FTAs, use the system cables intended for this use. Use the system cables to connect the FTAs and the corresponding connector boards.

If shielded cables are used, the shield must be applied to both ends.

i The correct wiring depends on the application. Observe the following points when wiring the module:

- Proper wiring.
- Cable/wire bending radius.
- Strain relief.
- Cable/wire load capacity.

8.1.4.6 Connecting the Racks

To establish a (redundant) connection between the system buses of two racks

1. Plug an RJ-45 connector of a patch cable into the UP socket located on the connector board of the left system bus module within the first rack.
2. Plug the second RJ-45 connector of the same patch cable into the DOWN socket located on the connector board of the left system bus module within the second rack.
   ☑ A non-redundant connection is established.
3. Plug an RJ-45 connector of a second patch cable into the UP socket located on the connector board of the right system bus module within the first rack.
4. Plug the second RJ-45 connector of the same patch cable into the DOWN socket located on the connector board of the right system bus module within the second rack.

► The two racks are redundantly connected.

i Patch cables colored or marked in a different way help to avoid mixing up cables, e.g., red patch cables for system bus A, green patch cables for system bus B

8.1.5 Mounting a Connector Board

Tools and utilities:

- Screwdriver, cross PH 1 or slotted 0.8 x 4.0 mm.
- Matching connector board.

To install the connector board

1. Insert the connector board into the guiding rail with the groove facing upwards (see following drawing). Fit the groove into the guiding rail pin.
2. Place the connector board on the cable shield rail.
3. Secure the captive screws to the base plate. First screw in the lower screws than the upper ones.

To remove the connector board

1. Release the captive screws from the base plate.
2. Carefully lift the lower section of the connector board from the cable shield rail.
3. Remove the connector board from the guiding rail.

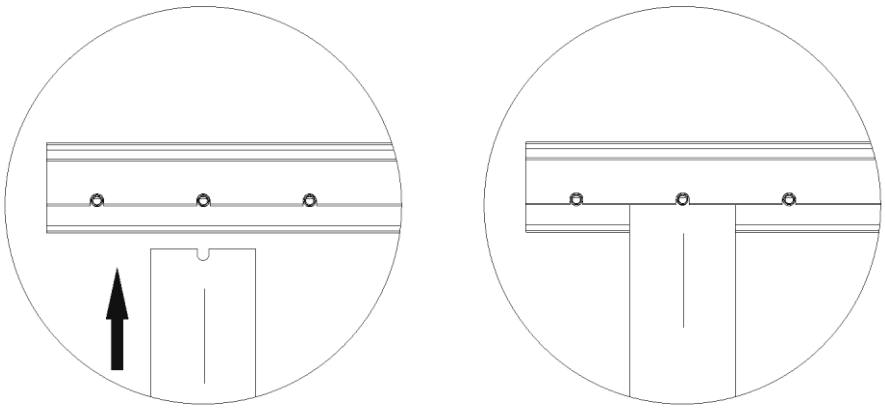


Figure 27: Example of how to Insert the Mono Connector Board

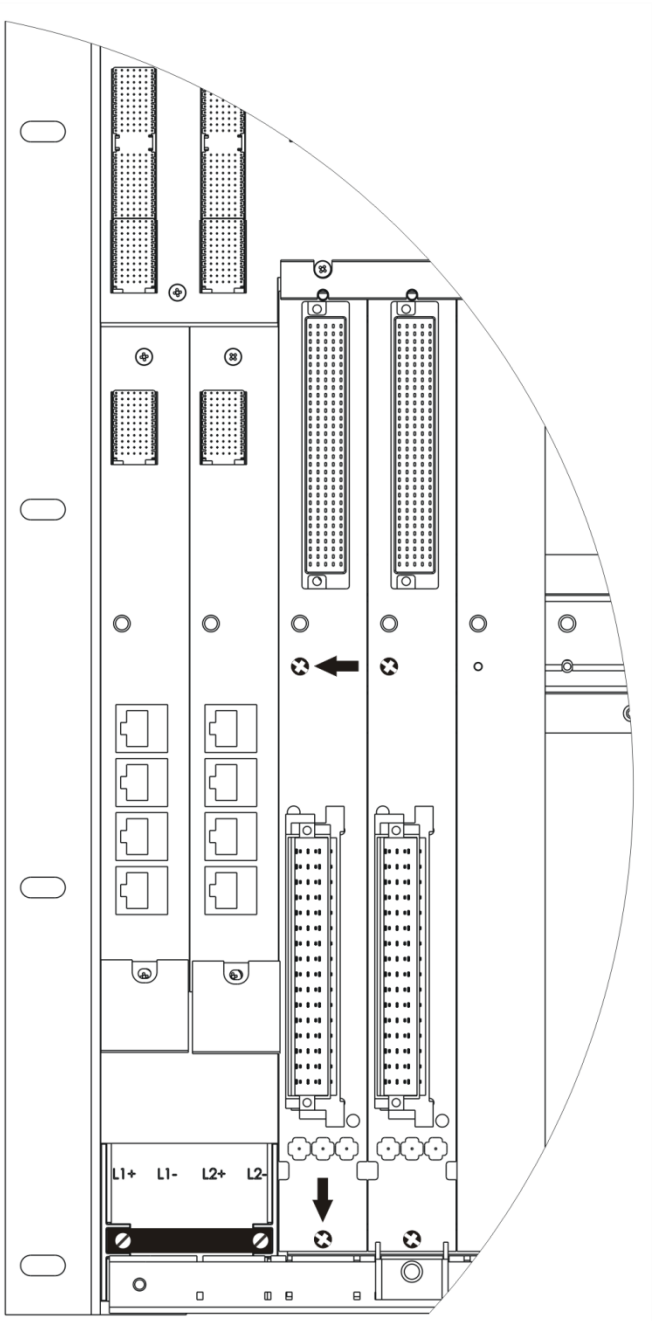


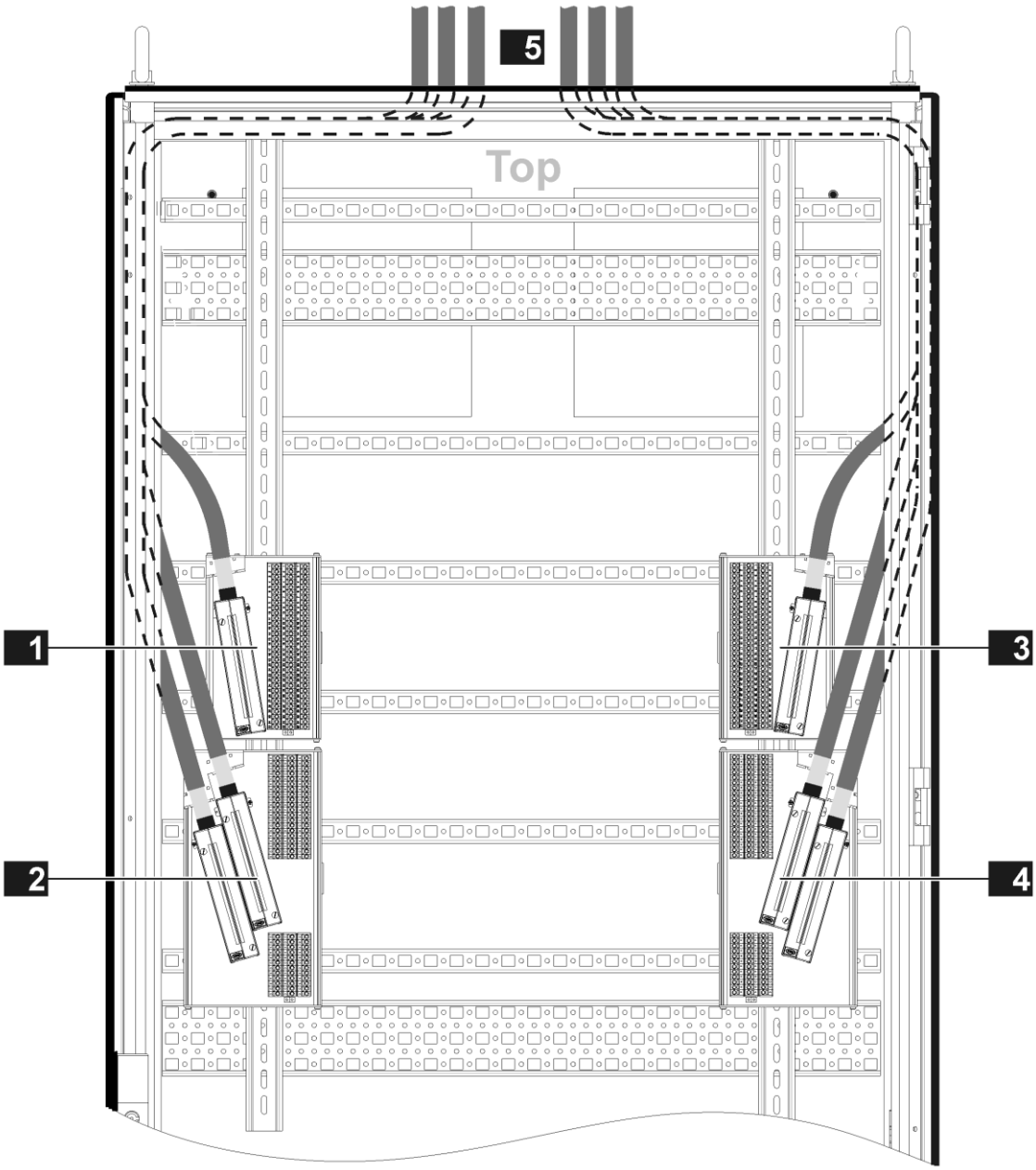
Figure 28: Example of how to Secure the Mono Connector Board with Captive Screws

i These instructions also apply for redundant connector boards. The number of used slots varies in accordance with the connector board type. The number of captive screws depends on the connector board type.

8.1.6 Mounting FTAs in the Control Cabinet

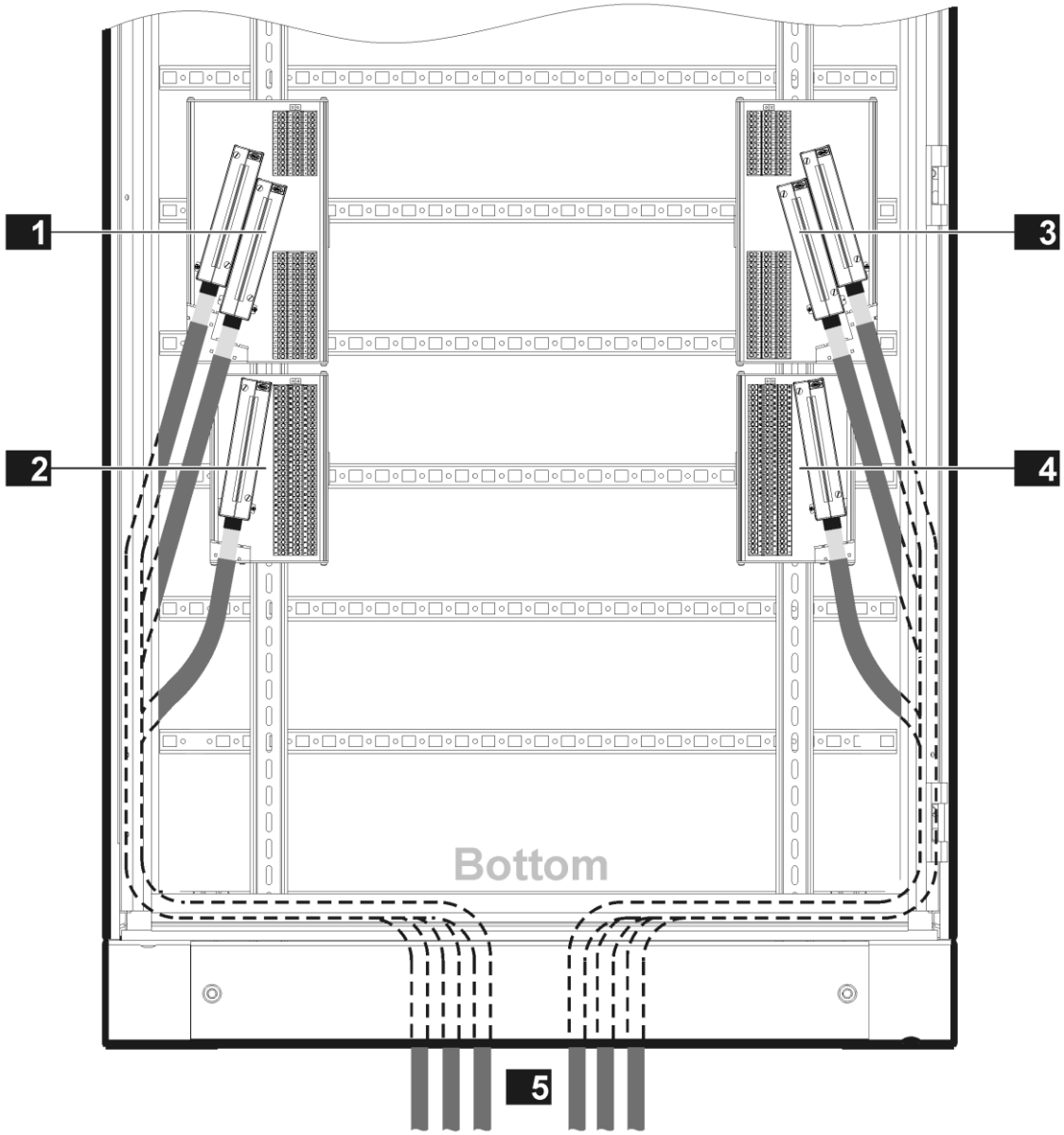
FTAs are available in a right and a left variant. The left FTAs are connected on the left side with system cables, the right FTAs on the right side. When mounting the FTAs vertically, distinguish whether they are connected from the top or the bottom and consider how they are arranged in the control cabinet.

i When mounting the system cables, ensure that the system cables are not subject to permanent tensile strain.



1 Right FTA
2 Right redundant FTA
3 Left FTA
4 Left redundant FTA
5 System cable entry

Figure 29: FTA Assembly in Control Cabinet with System Cable Entry from the Top



1 Left redundant FTA
2 Left FTA
3 Right redundant FTA
4 Right FTA
5 System cable entry

Figure 30: FTA Assembly in Control Cabinet with System Cable Entry from the Bottom

8.1.7 Considerations about Heat

The high integration level of electronic components causes heat loss, which also depends on the external load of the HIMax modules. For this reason, the installation and ventilation within the control cabinet are relevant in connection with the structure. Low ambient temperature increases the product life and the reliability of the electronic components within the system.

i The environmental requirements must be taken into account when mounting the devices.

8.1.7.1 Heat Dissipation

A closed enclosure or a closed cabinet must be designed such that the heat generated inside can be dissipated through the surface.

Choose the mounting type and position such that heat dissipation is ensured.

The power dissipation of the installed equipment is decisive for determining the fan components. Uniform distribution of the heat load and unhindered natural convection are assumed.

8.1.7.2 Definitions

Magnitude	Description	Unit
P_V	Power dissipation (heat capacity) of the electronic components within the device	W
A	Effective enclosure surface (see below)	m²
B	Enclosure width	m
H	Enclosure height	m
T	Enclosure depth	m
K	Coefficient of heat transfer of the enclosure	W/m² K
	Example: Steel plate	Approx. 5.5 W/m² K

Table 34: Definitions for Calculating the Power Dissipation

8.1.7.3 Installation Type

The effective enclosure surface area A as a function of the mounting or installation type is determined as follows:

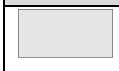
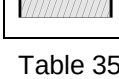
Enclosure installation type in accordance with VDE 0660, Part 5		Calculation of the enclosure surface A
	Individual enclosure, free-standing on all sides	A = 1.8 x H x (W + D) + 1.4 x W x D
	Individual enclosure for wall mounting	A = 1.4 x W x (H + D) + 1.8 x H x D
	Final enclosure, free-standing	A = 1.4 x D x (W + H) + 1.8 x W x H
	Final enclosure for wall mounting	A = 1.4 x H x (W + D) + 1.4 x W x D
	Central enclosure, free-standing	A = 1.8 x W + H+ 1.4 x W x D + H + D
	Central enclosure, for wall mounting	A = 1.4 x W x (H + D) + H x D
	Central enclosure, for wall mounting, with covered roof area	A = 1.4 x W + H+ 0.7 x W x D + H + D

Table 35: Installation Types

8.1.7.4 Natural Convection

When natural convection is applied, the lost heat is dissipated through the enclosure walls. Requirement: The ambient temperature must be lower than the temperature within the enclosure.

The maximum temperature increase $(\Delta T)_{max}$ of all electronic devices within the enclosure is calculated as follows:

$$(\Delta T)_{max} = \frac{P_V}{k * A}$$

The power dissipation P_V can be calculated based on the specifications for the electric power rating of the controller and its inputs and outputs.

8.1.7.5 Note on the Standard

The temperature within an enclosure can also be calculated in accordance with VDE 0660, Part 507 (HD 528 S2).

i Considerations about heat must take every component within a cabinet or enclosure into account, also components that are not directly part of the HIMax system!

8.1.7.6 Temperature Monitoring

The controllers are intended for operation up to a maximum ambient temperature of 60 °C. The temperature states of the individual modules are centrally evaluated by the processor module.

A temperature sensor located on a specific temperature-relevant position independently detects and continuously monitors the temperature state on the corresponding module.

In SILworX, the *Temperature State [X]* system variable can be used to evaluate the temperature state of each processor module X = 1…4. Refer to Chapter 5.2.5 for details on the assignment of indexes to slot numbers.

The *Temperature State[X]* system variable indicates the environment temperatures measured in the following temperature ranges:

Temperature range	Temperature state	System variable value *Temperature State[X]* [BYTE]
< 40 °C	Normal	0x00
≥ 40 °C	Threshold 1 exceeded	0x01
> 60 °C	Threshold 2 exceeded	0x03
Back to 60°C...40°C	Threshold 1 exceeded	0x01
Back to < 40 °C	Normal	0x00

Table 36: Temperature States

If the ambient temperature falls below or exceeds a specific threshold, the temperature state changes.

Table 36 applies when the HIMax with the X-FAN system fan is running in normal operation. Depending on the module slot in the rack and the own power dissipation of a module, the temperature state can already change at ambient temperatures below the specified temperature range limits.

During abnormal operation, e.g., without fans, the temperature state can already change at ambient temperatures below the specified temperature range limits.

The transition to the state *Threshold 1 exceeded* or *Threshold 2 exceeded* does **not** indicate an impairment of the system safety.

The temperature threshold that, if exceeded, causes a message to be issued, can be defined for each rack. In the SILworX Hardware Editor, use the detail view for the rack to configure this setting.

8.2 Start-Up

Only power up the system after the hardware is completely mounted and all the cables are connected. First start up the control cabinet, then the PES itself.

NOTICE

System damage possible!

Possible system damage caused by safety-related automation systems improperly connected or programmed.

Check all connections and test the entire system before start-up!

8.2.1 Starting up the Control Cabinet

Prior to connecting the supply voltage, check if all cables are properly connected, thus ensuring that no risk exists for controller and system.

8.2.1.1 Test of All Inputs and Outputs

Impermissible parasitic voltage (in particular with 230 VAC against ground or L-) can be measured using a universal measuring instrument.

HIMA recommends testing every individual terminal for impermissible parasitic voltage.

When testing external cables for isolation resistance, potential short-circuits or open-circuits, no cable ends must be connected to prevent potential damage or destruction of modules caused by high voltage.

To check for ground faults, unplug the voltage connection plugs from the power distributor and disconnect the supply voltage for sensors and the negative pole of actuators.

If the negative pole is grounded during operation, the ground connection must be interrupted for the duration of the ground fault check. The same applies to the ground connection of ground fault measuring facility, which may be connected to the system.

A megohmmeter or a special measuring facility must be used to check each connection against ground.

8.2.1.2 Voltage Connection

Requirement: The HIMax modules must be inserted and the corresponding cables must be connected. Proper polarity, voltage and ripple must be checked prior to connecting the 24 VDC supply voltage.

8.2.2 Starting Up Controllers with X-CPU 01 Modules

Requirements for start-up:

- The hardware is installed.
- The racks are not interconnected.
- The mode switches of all the processor modules are set to *Init*.
- All remaining modules are in STOP.
- The PADT network connection is configured such that the modules of the HIMax base plate can be reached. If required, enter a routing for the interface card in use.
- A suitable project is available with configured rack ID, IP address and system ID.

To start up the controller with X-CPU 01

1. Switch on the supply voltage.
2. Set the SRS, IP address, *Responsible* attribute and system bus mode for the system bus module in rack 0, slot 1:
   - Establish a direct physical connection between PADT and system bus module.

i The Ethernet interface *PADT* of the system bus module cannot perform an auto crossover.
Therefore use a crossover cable to connect to the system bus module.

   - In the structure tree, select **Hardware** within the resource element, and click **Online** on the Action Bar.
     The *Online Hardware* tab and the *System Login* window appear.
   - Click the **To Module Login** button.
   - In the *Online Hardware,* log in to the system bus module (double-click the system bus module, the module login window appears).
     Use the MAC address (see the label on the module) to read the IP address and the SRS (**Browse** button in the login window) and complete the login procedure.
   - Use the **Change** button on the *Search via MAC* dialog box to display the *Write via MAC* window. This window can be used to set the SRS, IP address, *Responsible* attribute and the system bus mode on the system bus module.

     For system bus A, the system bus module in slot 1, rack 0, is always set to *Responsible.* For system bus B, users can choose between the system bus module in slot 2, rack 0, and the system bus module in slot 2, rack 1.
3. Repeat step 2 for all the system bus modules on all existing racks.
4. Connect the system buses in all the racks with one another. To this end, establish an Ethernet connection between:
   - Rack 0, slot 1, port *DOWN* and rack 1, slot 1, port *UP*.
   - Rack 0, slot 2, port *DOWN* and rack 1, slot 2, port *UP*.

   If rack 2 is used, establish an Ethernet connection between:
   - Rack 0, slot 1, port *UP* and rack 2, slot 1, port *DOWN*.
   - Rack 0, slot 2, port *UP* and rack 2, slot 2, port *DOWN*.

   The connection to additional racks is established accordingly.

   ☑ The *UP* and *DOWN* LEDs as well as the *Red.* LEDs on the associated system bus modules are lit.
5. Prepare the processor module in slot 3, rack 0:
   - Log in to the processor module: Double click the processor module symbol in the online image.

i If a valid configuration is loaded into a processor module and the conditions for system operation are met, all settings such as SRS and IP address from the valid configuration become operative. This is particularly important during the initial operation of a processor module that was previously used.

HIMA recommends resetting to the factory settings (master reset) when using processor modules with an unknown history.

- Reset the processor module to the factory settings (master reset).
- Set the SRS on the processor module.
  With a mono system (i.e., the system is only equipped with one processor module or one system bus module), set mono operation. To do so, click **Set Mono/Redundancy Operation** in the *Online->Start-up* menu.
  This setting only takes effect if a mono project is loaded. Otherwise, the system automatically resets the switch.
- Set the mode switch of the processor module to *Stop*.
- ☑ After some time, the processor module indicates having achieved system operation: The *Stop* LED is lit or blinking, the *Init* LED is not lit.

6. Log in to the system.
7. Set the mode switches of all remaining processor modules, one after another, to *Stop.*
   ☑ The processor modules participate in system operation (redundancy). The *Stop* LEDs are lit or blinking, the *Init* LEDs are not lit.
8. Load the existing configuration to the processor modules (menu functions: **Online -> Resource Download**).
   ☑ All processor modules enter the STOP/VALID CONFIGURATION state.
9. Set the mode switches of all the processor modules to *Run*.
10. Perform a resource cold start.

► The system, i.e., all modules, are in RUN (or in RUN / UP STOP, if the user program was not started).

For further details on how to start up the system, refer to the first steps manual (HI 801 103 E).

8.2.2.1 Faults

- A processor module does not start redundant operation or quits it, in case of malfunction.
- The system enters the STOP / INVALID CONFIGURATION state if the project in SILworX does not match the hardware.

8.2.3 Starting Up Controllers with X-CPU 31 Modules

Processor modules of X-CPU 31 type can only be used in rack 0. They simultaneously operate as processor and system bus modules. They must therefore be inserted in slots 1 and 2.

Create the preconditions:

- The supply voltage is switched off.
- The racks are not interconnected.
- X-CPU 31 processor modules are inserted in slots 1 and 2. The communication modules and I/O modules may also be inserted.
- The mode switches of the processor modules are set to *Init*.
- All remaining modules are in STOP.
- The PADT network connection is configured such that the modules of the HIMax base plate can be reached. If required, enter a routing for the interface card in use.
- A suitable SILworX project is available with configured rack IDs, IP addresses, and system ID.

To start up the controller with X-CPU 31

1. Connect the supply voltage.
2. Set the SRS, IP address, *Responsible* attribute and the system bus mode of the left processor module X-CPU 31:
   - Establish a direct physical connection between PADT and processor module.
   - In the structure tree, select **Hardware** within the resource element, and click **Online** on the Action Bar.
     The *Online Hardware* tab and the *System Login* window appear.
   - Click the **To Module Login** button.
   - In the *Online Hardware,* log in to the processor module (double-click the processor module, the module login window appears).
     Use the MAC address (see the label on the module) to read the IP address and the SRS (**Browse** button in the login window) and complete the login procedure.
   - Use the **Change** button on the *Search via MAC* dialog box to display the *Write via MAC* window. This window can be used to set the SRS, IP address, *Responsible* attribute and the system bus mode on the processor module.

     Both X-CPU 31 processor modules are always set to *Responsible*.
3. Repeat step 2 for the right X-CPU 31 processor module and for all system bus modules on all existing racks. The system bus modules cannot be set to *Responsible*!
4. Connect the system buses in all the racks with one another. To this end, establish an Ethernet connection between:
   - Rack 0, slot 1, port *DOWN* and rack 1, slot 1, port *UP*.
   - Rack 0, slot 2, port *DOWN* and rack 1, slot 2, port *UP*.

   If rack 2 is used, establish an Ethernet connection between:
   - Rack 0, slot 1, port *UP* and rack 2, slot 1, port *DOWN*.
   - Rack 0, slot 2, port *UP* and rack 2, slot 2, port *DOWN*.

   The connection to additional racks is established accordingly.

   ☑ The *UP* and *DOWN* LEDs as well as the *Red.* LEDs on the associated processor and system bus modules are lit.
5. Prepare the processor module in slot 1, rack 0:
   - Log in to the processor module: Double click the processor module symbol in the online image.

i If a valid configuration is loaded into a processor module and the conditions for system operation are met, all settings such as SRS and IP addresses from the valid configuration become operative. This is particularly important during the initial operation of a processor module that was previously used.

HIMA recommends resetting to the factory settings (master reset) when using processor modules with an unknown history.

   - Reset the processor module to the factory settings (master reset).
   - If the system is only equipped with one processor module (mono system), set mono operation. To do so, click **Set Mono/Redundancy Operation** in the *Online->Start-up* menu.
     This setting only takes effect if a mono project is loaded. Otherwise, the system automatically resets the switch.
6. Set the mode switch of the left processor module to *Stop* and wait until the processor module indicates to be running in system operation.

   ☑ The *Stop* LED is lit or blinking, the *Init* LED is off.
7. Log in to the system.
8. Set the mode switch of the right processor module to *Stop*.

   ☑ The right processor module starts operating redundantly. The *Stop* LED is lit and the *Init* LED is off.

9. Load the existing configuration to the processor modules by performing a **download** (menu functions: **Online -> Resource Download**)
   ☑ The processor modules enter the STOP/VALID CONFIGURATION state.
10. The mode switches on all the processor modules are set to *Run*.
11. Perform a resource cold start.

► The system, i.e., all modules, are in RUN (or in RUN / UP STOP, if the user program was not started).

For further details on how to start up the system, refer to the first steps manual (HI 801 103 E).

8.2.3.1 Faults

- A processor module does not start redundant operation or quits it, in case of malfunction.
- The system enters the STOP / INVALID CONFIGURATION state if the project in SILworX does not match the hardware.

8.2.4 Assigning the Rack ID

The identification number must be assigned to the base plates or modified if already existing, when assembling or extending the hardware.

The rack ID is stored in the connector board of the system bus module and must be modified using the system bus module. The system bus module distributes the rack IDs among the remaining modules on the rack.

Whether the rack and its modules are uniquely identified depends on the rack ID. This, in turn, determines the identification of the inputs and outputs.

Always set the rack ID using a direct connection of the PADT to the corresponding system bus module to avoid that the rack ID of other system bus modules can be accidentally modified.

Observe this procedure since the rack ID is a safety-critical parameter

To assign the rack ID

1. Create the preconditions:
   ☑ Ensure that all modules on the rack are in STOP (to prevent the modules from exchanging old rack IDs).
   ☑ Ensure that no connection exists between the PADT and processor module.
   ☑ Establish a direct connection between PADT and system bus module.
2. Change the rack ID:
   - Change the rack ID of one system bus module through the direct connection.
   - Also change the rack ID of a second system bus module (if existing) through the direct connection.

► The new rack ID is valid. The configuration is consistent.

NOTICE

Controller malfunction due to inconsistent rack IDs!

Since the rack ID is a safety-critical parameter, it may only be changed using the described procedure!

8.2.5 Switching Between Line Structure and Network Structure

The HIMax system can only switch between line structure and network structure by switching the system bus modules.

8.2.5.1 Switching to Network Structure

Requirements for switching the system bus mode to network structure:

- The racks are connected with a line structure.
- All racks are connected redundantly.
- The system is free of faults and properly configured.
- The processor modules are in the STOP state.
- The PADT is connected to the system through rack 0. A system login has been performed.

To switch to network structure

1. First switch system bus A. To do so, perform steps 2 and 3 for the **left** system bus module or the X-CPU 31 processor module of each rack:
2. Set the mode of the system bus module, which is farthest from rack 0, to **Network**. Farthest means that the connection to this rack passes through most of the other racks or Ethernet segments.
3. Perform step 2 for all racks, one after the other and starting with the farthest one up to rack ID 0.
4. After switching the system bus module or the X-CPU 31 processor module in rack 0, system bus A reconnects itself. This can take a few minutes.
5. If the mode of system bus A is set to Network and connected, switch system bus B. To do so, perform steps 2 and 3 for the **right** system bus module of each rack (or X-CPU 31 in rack 0).

► The HIMax system operates with a network structure. The racks can be reconnected with the given structure.

8.2.5.2 Switching to Line Structure

Requirements for switching the system bus mode to line structure:

- The racks are structured as a network.
- The system is free of faults and properly configured.
- The processor modules are in the STOP state.
- The PADT is connected to the system through rack 0. A system login has been performed.

To switch to line structure

1. First switch to system bus A. To do so, perform steps 2 and 3 for the **left** system bus module of each rack (or X-CPU 31 in rack 0):
2. Set the mode of the system bus module, which is farthest from rack 0, to **Line**. Farthest means that the connection to this rack passes through most of the other racks or Ethernet segments.
3. Perform step 2 for all racks, one after the other and starting with the farthest one up to rack ID 0.
4. After switching system bus A, readjust the system bus wiring in a line structure. Connect the racks such that the sequence of the rack IDs corresponds to a proper line structure.
5. After system bus A wiring was successfully restructured, switch and restructure system bus B. To do so, perform steps 2 and 3 for the **right** system bus module of each rack (or X-CPU 31 in rack 0).

► The HIMax system operates with a line structure.

8.3 Maintenance and Repairs

HIMA recommends replacing the fans of the controllers at regular intervals.

i For a safety-related application, the controller must be subject to a proof test at regular intervals. For further details, refer to the safety manual (HI 801 003 E).

NOTICE

Malfunction due to electrostatic discharge!

Damage to the controller or electronic devices connected to it!

Only qualified personnel may perform maintenance actions to supply, signal and data lines. Implement ESD protection measures. Personnel must be electrostatically discharged prior to any contact with the supply or signal lines!

8.3.1 Interferences

Interferences in the processor modules cause the redundant processor module to assume the control task. If no redundant processor module is configured, the entire controller is shut down.

The *Error* LED on the processor module indicates interference.

For possible reasons for the *Error* LED to light, refer to the manual of the processor module used.

All the modules automatically detect interference during operation and indicate it through the *Error* LED on the module front plate.
SILworX can be used to diagnose faults (except for communication faults) even if the controller is in the STOP state.

Some of the module self-tests run in the background. If a module has failed the background test, an entry appears in the diagnostic history. If the module does not pass the background test within 24 h after the diagnostic entry, it switches off as faulty.

Prior to replacing an I/O module, check if disturbances exist on the external line and if the corresponding sensor or actuator is operating properly.

Once a failure has been removed (e.g., because the external wires have been repaired or a module has been replaced), the HIMax system autonomously enters the fault-free state and switches off the corresponding LEDs. No user acknowledgment is required.

If a restart inhibition is required for an application, it must be programmed in the user program.

8.3.2 Connecting the Power Supply after a Service Interruption

After connecting to the power supply, the HIMax system modules start in random order. This applies for the HIMax modules as well as for the connected remote I/Os.

8.3.3 Connecting the Redundant Power Supply

Because of potential high currents, act with particular caution when connecting a redundant power supply during operation.

⚠ WARNING

Physical injury due to overheating possible when connecting a redundant power source!

Check proper polarity, prior to connecting a redundant power supply unit during operation!

8.3.4 Loading Operating Systems

Refer to the release notes for the corresponding operating system version for details on how to load the operating system.

All HIMax system modules contain processors and operating systems controlling the module. The operating systems are delivered with the modules. As part of product maintenance, HIMA is continuously developing and improving the operating systems to be loaded into the modules using SILworX.

The following table lists the modules and associated operating system versions (for example: **HIMAXIO_HA1_OS_Vx.xx.ldb**) according to the hardware revision index (HW-Rev.).

I/O Modules	HW-Rev. xx [1)]	Operating system file name beginning with:	Connect the PADT to:
X-AI 16 51	All	**HIMaxIO_HA2 …**	Communication module, processor module, system bus module. Refer to Chapter 5.1.1 concerning the use of the Ethernet interfaces.
X-AI 32 01	01, 10, 11, 12, 13	**HIMaxIO_HA1 …**	
	14	**HIMaxIO_HA3 …**	
X-AI 32 02	All	**HIMaxIO_HA3 …**	
X-AI 32 51	All	**HIMaxIO_HA2 …**	
X-AO 16 01	All	**HIMaxIO_HA1 …**	
X-AO 16 51	All	**HIMaxIO_HA2 …**	
X-CI 24 01	All	**HIMaxIO_HA3 …**	
X-CI 24 51	All	**HIMaxIO_HA2 …**	
X-DI 16 01	All	**HIMaxIO_HA1 …**	
X-DI 32 01	01, 02, 10, 11	**HIMaxIO_HA1 …**	
	12	**HIMaxIO_HA3 …**	
X-DI 32 02	01, 02, 10, 11	**HIMaxIO_HA1 …**	
	12	**HIMaxIO_HA3 …**	
X-DI 32 03	All	**HIMaxIO_HA1 …**	
X-DI 32 04	All	**HIMaxIO_HA3 …**	
X-DI 32 05	All	**HIMaxIO_HA3 …**	
X-DI 32 51	All	**HIMaxIO_HA2 …**	
X-DI 32 52	All	**HIMaxIO_HA2 …**	
X-DI 64 01	10	**HIMaxIO_HA1 …**	
	11	**HIMaxIO_HA3 …**	
X-DI 64 51	All	**HIMaxIO_HA2 …**	
X-DO 12 01	All	**HIMaxIO_HA1 …**	
X-DO 12 51	All	**HIMaxIO_HA2 …**	
X-DO 12 02	All	**HIMaxIO_HA1 …**	
X-DO 24 01	01, 02, 10, 11, 12, 13	**HIMaxIO_HA1 …**	
	14	**HIMaxIO_HA3 …**	
X-DO 24 02	All	**HIMaxIO_HA1 …**	

I/O Modules	HW-Rev. xx [1])	Operating system file name beginning with:	Connect the PADT to:
X-DO 32 01	10, 11	**HIMaxIO_HA1 …**	Communication Module Processor Module System bus module, See Chapter 5.1.1
	12	**HIMaxIO_HA3 …**	
X-DO 32 51	All	**HIMaxIO_HA2 …**	
X-HART 32 01	All	**HIMaxIO_HA3 …**	
X-MIO 7/6 01	All	**HIMaxIO_HA3 …**	
Processor modules	HW-Rev. xx [1])	Operating system file name beginning with:	Connect the PADT to:
X-CPU 01	All	**HIMaxCPU_ …**	Communication Module Processor Module System bus module, See Chapter 5.1.1
X-CPU 31	All	**HIMaxCPU3x_ …**	
Notes: ▪ The simultaneous use of processor modules with differing operating system versions is only allowed for the duration of the upgrade! ▪ If safe**ethernet** is used, the processor modules must be upgraded one after the other, without performing any actions in between! Only operating system versions ≥ V6 may be loaded into X-CPU 31 processor modules!			
System bus modules	HW-Rev. xx [1])	Operating system file name beginning with:	Connect the PADT to:
X-SB 01	All	**HIMaxSB_HA2 …**	Communication Module Processor Module System bus module, See Chapter 5.1.1
Communication module	HW-Rev. xx [1])	Operating system file name beginning with:	Connect the PADT to:
X-COM 01	All	**HIMaxCOM_HA2 …**	Processor Module
[1]) The hardware revision index of each module can be read out in the SILworX **Module Data Overview.**			

Table : Operating Systems to be Loaded into the Modules

i

Prior to upgrading the operating systems, the HIMax system must be in a fault-free state!

No further actions may be performed on the system during the upgrade process!

Interruption of operation possible during the load procedure!

Ensure the operation of a functional, redundant module! The redundant module maintains operation during the load procedure.

8.3.4.1 Upgrading and Downgrading Operating Systems

In rare cases, it can make sense to load the previous operating system version into the module.

If a controller has run for a long time without modification and a single module must be replaced, it may be better to load the existing operating system version into the new module. The existing operating system version may be better suited for use with that of the remaining modules.

Loading a previous operating system version into a module used can cause the data in the non-volatile memory, e.g., the diagnostic history, to be deleted.

Processor modules with operating system version < V6.30 must be subject to an interim upgrade to V6.30, V7 or V8 prior to upgrading them to V10.X!

Loading an operating system version prior to V3 must be avoided.

8.3.5 Repair

NOTICE

Malfunction of the controller due to insufficient repair!

Only HIMA is authorized to repair a safety-related HIMax system or the modules contained in it.

In case of unauthorized intervention in the device, functional safety cannot be ensured and the warranty or certification lapses.

8.4 Special Operating States

The HIMax system can achieve the operating states specified here if the components have failed or the user has intentionally stopped the operating states.

In these cases, the modules' operating systems ensure that the HIMax system enters a predefined state.

8.4.1 Mono operation

Mono operation must have been configured to start a HIMax system with only one system bus A and one X-CPU 01 in rack 0, slot 3 or X-CPU 31 in slot 1.

To set mono operation

1. Meet the requirements:
   - Only one system bus is active.
   - The mono resource configuration is loaded.
   - PES user is logged in with write permission.
   - *Allow Online Settings* is ON.
2. Select the **Set Mono/Redundancy Operation** menu function from **Online->Start-up**.
   ☑ The *Set Mono/Redundancy Operation* dialog box appears.
3. In the dialog box, change the setting from *Redundancy* to *Mono* and click **OK** to confirm.

► The HIMax system is configured for mono operation and can start operation with only one system bus module and one processor module.

The setting is stored in the non-volatile memory.

If mono operation is configured and only one system bus can be accessed, the processor module in slot 3 of rack 0 starts operating alone. This also applies if additional processor modules with more recent retain data are used!

The X-CPU 31 module in slot 1 will also start mono operation, if a second X-CPU 31 in slot 2 exists, but is not connected to it.

Loading a redundant resource configuration into a system configured for mono operation results in a setting adjustment supporting redundant operation.

When starting up a mono system, HIMA recommends temporarily inserting a system bus module into rack 0, slot 2 and setting it to *Responsible*. In doing so, the following becomes possible:

- Easy replacement of the system bus module if faults occur, even if slot 1 is faulty.
- Extension to redundant operation while the system is running.

8.4.2 Start with only one system bus or processor module set to *Responsible.*

A redundant system does not start up after the supply voltage is switched on unless both responsible system bus modules or processor modules (X-CPU 31) can be reached and are functional.

This limitation of the availability must be rectified before starting, for example, by replacing the defective module.

Start Emergency Mono System Operation serves to start a system with only one responsible module in exceptional cases. Starting the mono system operation by setting the *Mono Startup* switch or using *Start Emergency Mono System Operation* for a processor module is only permitted if there is no processor module present in system operation and if no other processor module apart from the selected one is ready to commence system operation. Otherwise, there is no guarantee that the selected processor module will start the system operation first.

For details, refer to the SILworX online help.

8.4.3 X-CPU 01 Processor Modules Distributed on Rack 0 and Rack 1

A highly redundant system structure is as follows:

- The X-CPU 01 processor modules are distributed among rack 0 and rack 1.

  This configuration is only possible with X-CPU 01, not with X-CPU 31!
- The responsible system bus modules are distributed among rack 0, slot 1, and rack 1, slot 2.
- The type of wiring of the two system buses between rack 0 and rack 1 prevents a fault or interruption from simultaneously occurring on both system buses within the watchdog time.

In this configuration, simultaneous disconnection of the **two** systems between rack 0 and rack 1 could result in two HIMax subsystems. Each subsystem would be able to operate and control processes independently from the other subsystem. This could result in hazardous states within the process.

To avoid this scenario, the operating systems behave as follows:

- If system bus A is disconnected before system bus B, rack 0 enters the STOP state. Rack 1 maintains the RUN state.
- If system bus B is disconnected before system bus A, rack 1 enters the STOP state. Rack 0 maintains the RUN state.

8.4.3.1 Shutting Down a Rack with Processor Modules

After shutting down a rack with processor module, the remaining processor module must be able to continue system operation of the entire system alone, i.e., it must have an intact connection to all other modules within the HIMax system.

A rack with processor module may be shut down if the following conditions are met:

- Both racks are in proper working order:
  - The X-CPU 01 processor modules are distributed among rack 0 and rack 1 and are in the RUN state.
  - Rack 0 or rack 1 can be shut down if none of the system bus modules set to *Responsible* is also configured as *Essential*.
- One of the two racks is in proper working order:
  - The X-CPU 01 processor module in the rack that is still operating is in the RUN state.
  - The rack with system bus module set to *Responsible* but not to *Essential* may be shut down.
  - The rack with system bus module set to *Responsible* and to *Essential* must not be shut down. A problem exists on system bus A or system bus B.
    The rack may only be shut down if the problem has been fixed and the system bus module is no longer set to *Essential*.

8.4.4 Processor Modules with Differing Project Configurations

Particular care must be paid when processor modules previously in use in other systems are to be used.

When a system containing processor modules with differing project configurations is restarted, users must ensure that the first processor module to be started is that with a project configuration suitable for the system.

To this end, processor modules, which may contain a differing project configuration, should only be added after the first processor module in the rack has started up. They synchronize themselves and adopt the configuration of the first processor module.

Alternatively: The mode switch in the processor modules that may contain another project configuration should be set to *Init*. Once the first processor module has started, the mode switch in the remaining processor modules must be set to *Run*.

8.4.5 Autostart When the System is Stopped

If the *Autostart* system parameter is activated and the system is in STOP, the following actions cause the system to unintentionally enter the RUN state:

- Removing and reinserting both system bus modules with the *Responsible* attribute.
- Switching the supply voltage off and on again.

The transition to the RUN state can be prevented by resetting the *Autostart* system parameter online prior to performing the action.

8.4.6 Communication Between two X-CPU 31 Processor Modules

If communication between two redundant X-CPU 31 processor modules is disturbed, the processor module not showing *Essential* can be replaced.

If communication is still disturbed, the processor modules showing *Essential* can specifically be removed from system operation through a targeted PADT command. The new processor module adopts system operation and becomes *Essential*.

Prior to stopping the processor module, check whether the other processor module can maintain system operation alone. This is the case when the new processor module has intact connections to all the other undisturbed modules and to the safe**ethernet** communication partners.

9 Documentation

This chapter provides a table with the list of the available HIMax documents.

9.1 HIMax System Documentation

The following documents are available:

Document	Content	Document no.
System manual	Hardware description of the HIMax system	HI 801 001 E
Safety Manual	Safety functions of the HIMax system	HI 801 003 E
X-BASE PLATE	Base plate for X-CPU 01	HI 801 025 E
X-BASE PLATE	Base plate for X-CPU 31	HI 801 371 E
X-FAN	System Fans	HI 801 033 E
X-CPU 01	Processor module, SIL 3	HI 801 009 E
X-CPU 31	Processor module with system bus, SIL 3	HI 801 355 E
X-COM 01	Communication Module	HI 801 011 E
X-SB 01	System bus module, SIL 3	HI 801 007 E
X-AI 16 51	Analog input module, 16 channels, SIL 1	HI 801 179 E
X-AI 32 01	Analog input module, 32 channels, SIL 3	HI 801 021 E
X-AI 32 02 SOE	Analog input module, 32 channels, SOE, SIL 3	HI 801 055 E
X-AI 32 51	Analog input module, 32 channels	HI 801 181 E
X-AO 16 01	Analog output module, 16 channels, SIL 3	HI 801 111 E
X-CI 24 01	Counter input module, 24 channels, SIL 3	HI 801 113 E
X-CI 24 51	Counter input module, 24 channels	HI 801 189 E
X-DI 16 01	Digital input module, 16 channels, SIL 3	HI 801 057 E
X-DI 32 01	Digital input module, 32 channels, SIL 3	HI 801 015 E
X-DI 32 02	Digital input module, 32 channels for proximity switches, SIL 3	HI 801 017 E
X-DI 32 03	Digital input module, 32 channels, SIL 3	HI 801 059 E
X-DI 32 04 SOE	Digital input module, 32 channels, SOE, SIL 3	HI 801 051 E
X-DI 32 05 SOE	Digital input module, 32 channels for proximity switches, SOE, SIL 3	HI 801 053 E
X-DI 32 51	Digital input module, 32 channels, SIL 3	HI 801 173 E
X-DI 32 52	Digital input module, 32 channels for proximity switches, SIL 3	HI 801 175 E
X-DI 64 01	Digital input module, 32 channels, SIL 3	HI 801 093 E
X-DI 64 02	Digital input module, 32 channels, SIL 3	HI 801 177 E
X-DO 12 01	Digital relay output module, 12 channels, SIL 3	HI 801 023 E
X-DO 12 02	Digital output module, 12 channels, SIL 3	HI 801 099 E
X-DO 12 51	Digital relay output module, 12 channels	HI 801 185 E
X-DO 24 01	Digital output module, 24 channels, SIL 3	HI 801 019 E
X-DO 24 02	Digital output module, 24 channels, SIL 3	HI 801 095 E
X-DO 32 01	Digital output module, 32 channels, SIL 3	HI 801 097 E
X-DO 32 51	Digital output module, 32 channels	HI 801 183 E
X-HART 32 01	HART module	HI 801 015 E
X-MIO 6/7 01	Overspeed trip module	HI 801 305 E
X-FTA 001 01	FTAs for the various modules	HI 801 115 E
X-FTA 001 02		HI 801 131 E
X-FTA 002 01		HI 801 117 E
X-FTA 002 02		HI 801 119 E
X-FTA 003 02		HI 801 121 E

Document	Content	Document no.
X-FTA 005 02		HI 801 125 E
X-FTA 006 01		HI 801 127 E
X-FTA 006 02		HI 801 129 E
X-FTA 007 02		HI 801 133 E
X-FTA 008 02		HI 801 135 E
X-FTA 009 02		HI 801 137 E
SILworX first steps manual	Introduction to SILworX	HI 801 103 E
SILworX online help (OLH)	Instructions on how to use SILworX	
Communication manual	Description of safe**ethernet** communication and list of the available protocols.	HI 801 101 E

Table 37: Overview of the HIMax Documents

The current manuals can be obtained upon request by sending an e-mail to: documentation@hima.com. For registered HIMA customers, the product documentation is available at https://www.hima.com/en/downloads/.

Appendix

Application Examples

This chapter provides examples of how to set up the HIMax systems. I/O modules and communication modules were not taken into account. They are plugged in to the remaining slots, if required.

If required, base plates with 15 or 18 slots can also be used instead of base plates with 10 slots as presented in the examples.

Small System

This redundant system is composed of one base plate and two processor modules. The base plate has rack ID 0.

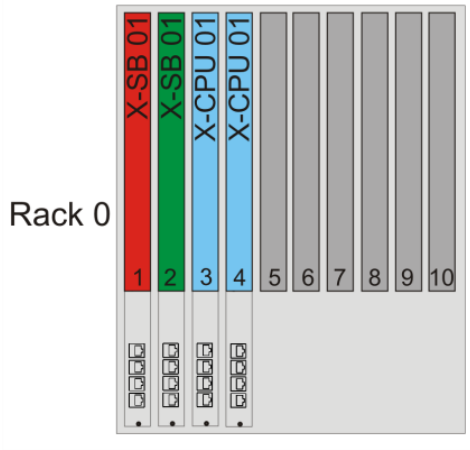


Figure 31: Small HIMax System: One Base Plate, Two Processor Modules

Minimum System

This system without redundancy represents the absolute modicum: One base plate 0, one processor module, one system bus module. Only one system bus A is used.

To ensure proper ventilation, a blank module must be used in slot 2. Slot 2 may not be used for communication or I/O modules.

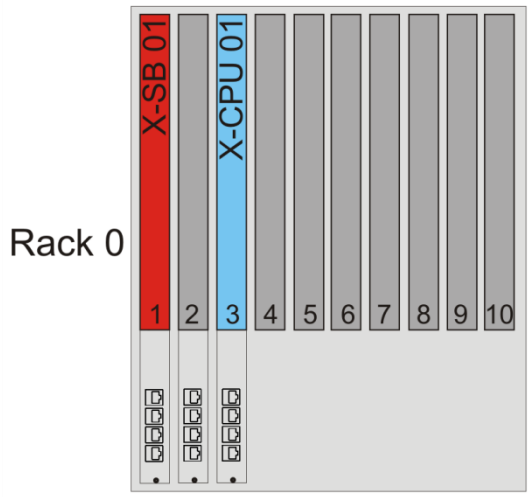


Figure 32: Minimum System without Redundancy

HIMA recommends using both system bus modules.

Distributed Redundancy

This system contains four redundant processor modules distributed on base plate 0 and base plate 1.

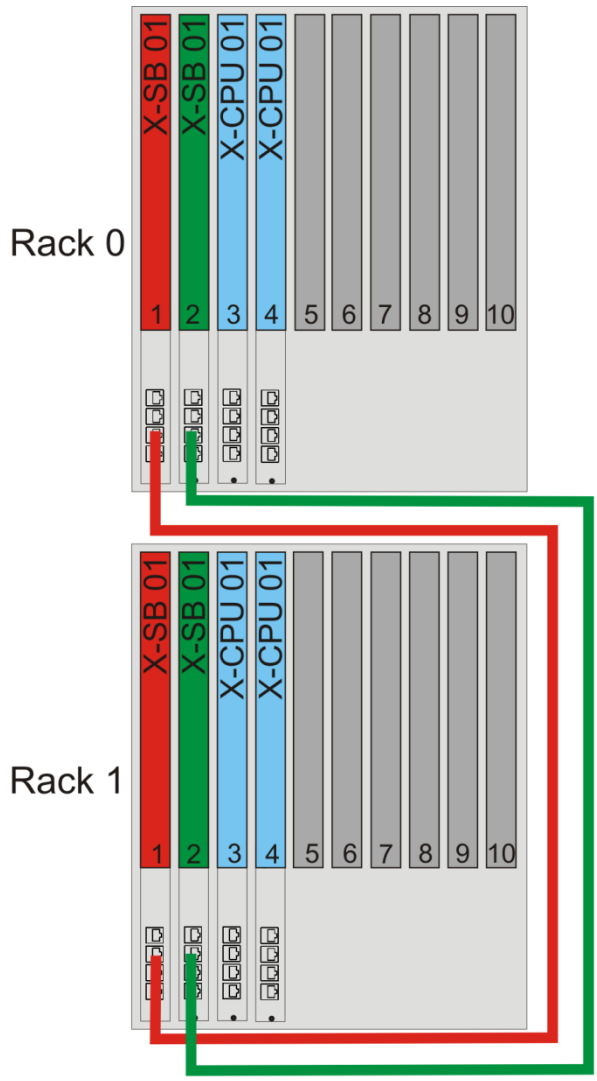


Figure 33: HIMax System with Distributed Redundancy

Glossary

Term	Description
AI	Analog input
AO	Analog output
ARP	Address resolution protocol, network protocol for assigning the network addresses to hardware addresses
COM	Communication module
CRC	Cyclic redundancy check
DI	Digital input
DO	Digital output
EMC	Electromagnetic compatibility
EN	European standard
ESD	Electrostatic discharge
FB	Fieldbus
FBD	Function block diagrams
HW	Hardware
ICMP	Internet control message protocol, network protocol for status or error messages
IEC	International electrotechnical commission
Interference-free	Inputs are designed for interference-free operation and can be used in circuits with safety functions
MAC	Media access control address, hardware address of one network connection
PADT	Programming and debugging tool (in accordance with IEC 61131-3), PC with SILworX
PELV	Protective extra low voltage
PES	Programmable electronic system
R	Read, the variable is read out
R/W	Read/Write, column title for system variable type
Rack ID	Base plate identification (number)
r_P	Peak value of a total AC component
SB	System bus (module)
SC/OC	Short-circuit/open-circuit
SELV	Safety extra low voltage
SFF	Safe failure fraction, portion of faults that can be safely controlled
SIL	Safety integrity level (in accordance with IEC 61508)
SILworX	Programming tool
SNTP	Simple network time protocol (RFC 1769)
SRS	System.Rack.Slot, addressing of a module
SW	Software
TMO	Timeout
W	Write, the variable receives a value, e.g., from the user program
WD	Watchdog, device for monitoring the system's correct operation Signal for fault-free process
WDT	Watchdog time

Index of Figures

Index of Tables

Index

MANUAL

HIMax system manual

HI 801 001 E

For further information, please contact:

HIMA Paul Hildebrandt GmbH
Albert-Bassermann-Str. 28
68782 Brühl, Germany

Phone +49 6202 709-0
Fax +49 6202 709-107
E-mail info@hima.com

Learn more about HIMax online:

https://www.hima.com/en/products-services/himax/

www.hima.com