Manual

Protocols

Modbus

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 801 515 D, Rev. 11.00 (1930)	German original document
HI 801 522 E, Rev. 11.00.00 (1933)	English translation of the German original document

Table of Contents

1 Introduction 5

1.1 Structure and Use of This Manual 5

1.2 Target Audience 6

1.3 Writing Conventions 6

1.3.1 Safety Notices 6

1.3.2 Operating Tips 7

1.4 Safety Lifecycle Services 7

2 Safety 8

2.1 Intended Use 8

2.2 Residual Risk 8

2.3 Safety Precautions 8

2.4 Emergency Information 8

2.5 Automation Security for HIMA Systems 8

3 Modbus 9

4 Modbus Master 10

4.1 Modbus Example 11

4.1.1 Configuring the Modbus TCP Slave 12

4.1.2 Configuring the Modbus TCP Master 13

4.2 Example of Alternative Register/Bit Addressing 15

4.3 Menu Functions of the Modbus Master 16

4.3.1 Edit 16

4.3.2 Properties 16

4.4 Modbus Function Codes of the Master 18

4.4.1 Modbus Standard Function Codes 18

4.4.2 HIMA-Specific Function Codes 19

4.4.3 Format of Request and Response Header 20

4.4.4 Read Request Telegrams 21

4.4.5 Read/Write Request Telegram 22

4.4.6 Write Request Telegram 22

4.5 Ethernet Slaves (TCP/UDP Slaves) 24

4.5.1 System Variables for TCP/UDP Slaves 25

4.5.2 TCP/UDP Slave Properties 26

4.6 Modbus Gateway (TCP/UDP Gateway) 27

4.6.1 Gateway Properties 29

4.6.2 System Variables for the Gateway Slave 29

4.6.3 Gateway Slave Properties 29

4.7 Serial Modbus 30

4.7.1 Serial Modbus Properties 31

4.7.2 System Variables for the Modbus Slave 31

4.7.3 Modbus Slave Properties 32

4.8 Control Panel (Modbus Master) 32

4.8.1 Context Menu (Modbus Master) 32

4.8.2 View Box (Modbus Master) 33

4.9 Control Panel (Modbus Master->Slave) 33

4.10 FBx LED Function in the Modbus Master 34
4.11 Function of the FAULT LED in the Modbus Master (HIMax only) 34
5 Modbus Slave 35
5.1 Equipment and System Requirements 35
5.2 Modbus Slave (Properties) 36
5.3 Configuring the Modbus TCP Slave 36
5.4 Configuring the Redundant Modbus TCP Slave 37
5.5 Rules for Redundant Modbus TCP Slaves 37
5.5.1 Process Data Volume Redundant Modbus Slaves 37
5.5.2 Slots Allowed for the Redundant Modbus Slave HIMax COM Modules 37
5.5.3 Redundant Modbus Slave COM Modules in Different Base Plates 38
5.6 Menu Functions of the HIMA Modbus Slave Set 38
5.6.1 Modbus Slave Set Properties 38
5.6.2 Register Variables 40
5.6.3 Bit Variables 41
5.7 Modbus Slave Set System Variables 41
5.7.1 Redundant Modbus Slave and Modbus Slave 42
5.7.2 System Variables 44
5.8 Modbus Function Codes of the Modbus Slaves 45
5.8.1 Modbus Function Codes 45
5.9 HIMA-Specific Function Codes 47
5.9.1 Format of Request and Response Header 48
5.10 Addressing Modbus using Bit and Register 49
5.10.1 Register Area 49
5.10.2 Bit Area 50
5.11 Offsets for Alternative Modbus Addressing 51
5.11.1 Access to the Register Variables in the Bit Area of the Modbus Slave 52
5.11.2 Access to the Bit Variables in the Register Area of the Modbus Slave 53
5.12 Control Panel (Modbus Slave) 54
5.12.1 Context Menu (Modbus Slave) 54
5.12.2 View Box (Modbus Slave) 54
5.12.3 View Box (Master Data) 54
5.13 FBx LED Function in the Modbus Slave 55
5.14 Function of the FAULT LED in the Modbus Slave (HIMax only) 55
6 General 56
6.1 Maximum Communication Time Slice 56
6.1.1 Determining the Maximum Duration of the Communication Time Slice 56
6.2 Load Limitation 56
Appendix 59
Glossary 59
Index of Figures 60
Index of Tables 60

1 Introduction

The Modbus manual describes the properties of the Modbus protocol and its configuration in SILworX for the safety-related HIMA controller systems.

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

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

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

1.1 Structure and Use of This Manual

The manual contains the following chapters:

- Introduction
- Safety
- Product description
- Modbus master
- Modbus slave
- General

Additionally, the following documents must be taken into account:

Name	Content	Document no.
HIMax system manual	Hardware description HIMax system	HI 801 001 E
HIMax safety manual	Safety function HIMax systems	HI 801 003 E
HIMatrix safety manual	Safety function HIMatrix systems	HI 800 023 E
HIMatrix compact system manual	Hardware description HIMatrix compact system	HI 800 141 E
HIMatrix modular system manual	Hardware description HIMatrix modular F 60 system	HI 800 191 E
HIQuad X system manual	Hardware description HIQuad X system	HI 803 211 E
HIQuad X safety manual	Safety function HIQuad X system	HI 803 209 E
Automation security manual	Description of automation security aspects related to the HIMA systems	HI 801 373 E
SILworX first steps manual	Introduction to SILworX.	HI 801 103 E

Table 1: Additional Applicable Manuals

All the current manuals can be obtained upon request by sending an e-mail to: documentation@hima.com. The documentation is available for registered HIMA customers in the download area https://www.hima.com/en/downloads/.

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.

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.

The product is operated with SELV or PELV. 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

To use the HIMA controllers, all pertinent requirements must be met, see manuals in Table 1.

2.2 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.

2.3 Safety Precautions

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

2.4 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.

2.5 Automation Security for HIMA Systems

The objectives of Automation Security are data confidentiality, integrity and availability. Targeted attacks are to be expected in Automation Security. In particular, interfaces such as described in this manual, are potential target of cyber attacks.

⚠ WARNING

Physical injury possible due to unauthorized manipulation of the controller!

Protect the controller against unauthorized access!

The user is responsible for implementing the necessary measures in a way suitable for the plant!

Careful planning should identify the measures to implement. The required measures are to be implemented after the risk analysis is completed. Such measures can include:

- Meaningful allocation of user groups.
- Maintained network maps help to ensure that secure networks are separated from public networks and, if required, only a well-defined connection exists (e.g., firewall or DMZ).
- Use of appropriate passwords.

A periodical review of the security measures is recommended, e.g., every year.

For further details, refer to the HIMA automation security manual (HI 801 373 E).

3 Modbus

For nearly all process control and visualization systems, Modbus coupling of the HIMA systems can be performed either directly via the RS485 interfaces or via the Ethernet interfaces of the controllers. HIMA systems can operate both as master and slave.

The Modbus functionality makes it particularly easy to connect to control panels or other controllers. Given its intensive use in projects worldwide, Modbus has been proven through a variety of applications.

Modbus master (see Chapter 4)
Redundancy of the Modbus master must be configured in the user program such that the user program monitors the redundant transport paths and assigns the redundantly transmitted process data to the corresponding transport path.

Modbus slave (see Chapter 5)
The Modbus slave can be configured redundantly.

i HIMA recommends using the Modbus slave V2 protocol for new projects. This is not based on a new Modbus specification, but is an enhanced HIMA variant concerning the internal processing of the protocol data on HIMA controllers. The standard Modbus function codes remain the same, the HIMA specific Modbus function codes no longer exist. For further information, refer to the Modbus slave V2 manual (HI 801 475 E).

4 Modbus Master

Data transmission between Modbus master and Modbus slaves can be configured via the serial interface (RS485) and via TCP/UDP (Ethernet). The Modbus master can be also used as a gateway (Modbus of TCP/UDP -> RS485.

Equipment and System Requirements

Element	Description
HIMA controller	HIMax with X-COM 01 module HIQuad X with F-COM 01 module HIMatrix
CPU module	The Ethernet interfaces on the processor module may not be used for Modbus TCP.
COM module	Ethernet 10/100BaseT D-sub connectors FB1 and FB2 If Modbus RTU is used, each of the serial fieldbus interfaces (FB1 and FB2) used on the COM module must be equipped with an optional HIMA RS485 submodule. For further details on the interface assignment, refer to the communication manual (HI 801 101 E).
Activation	Each of the two Modbus master functions must be activated individually; refer to the communication manual for details (HI 801 101 E). Modbus master RTU (RS485) and Modbus master TCP. The Modbus master RTU license is required for the Modbus gateway.

Table 2: Equipment and System Requirements for the Modbus Master

Modbus Master Properties

Property	Description
Modbus Master	One Modbus master can be configured for each COM module or each HIMatrix controller. The Modbus master can simultaneously: – Exchange data with TCP/UDP slaves. – Exchange data with serial slaves. – Be used as gateway from Modbus TCP to Modbus RTU.
Max. number of Modbus slaves	One Modbus master can operate up to 247 slaves. – 121 Modbus slaves per serial interface (FB1, FB2). – 64 TCP slaves via TCP/IP connection. – 247 UDP slaves through the UDP/IP connection. The maximum number of UDP slaves is limited since the slaves must be managed on the master side.
Max. number of request telegrams	Up to 988 request telegrams can be configured for each Modbus master.
Max. process data length for each request telegram	The maximum process data length depends on the request telegrams, see Chapter 4.4.
Max. size of send data Max. size of receive data	i The status bytes of the master and the status bytes of each slave assigned to it must be subtracted from the max. size of send data. Refer to the communication manual (HI 801 101 E).

<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>Display format of the Modbus data</td><td>The HIMA controllers use the big endian format for data.
Example: 32-bit data (e.g., DWORD, DINT):
<table>
<tr><td>32-bit data (hex)</td><td colspan="4">0x12345678</td></tr>
<tr><td>Byte number (from the left)</td><td>0</td><td>1</td><td>2</td><td>3</td></tr>
<tr><td>Big endian
(HIMax, HIQuad X, HIMatrix)</td><td>12</td><td>34</td><td>56</td><td>78</td></tr>
<tr><td>Middle endian (H51q)</td><td>56</td><td>78</td><td>12</td><td>34</td></tr>
<tr><td>Little endian</td><td>78</td><td>56</td><td>34</td><td>12</td></tr>
</table></td></tr>
</table>

Table 3: Properties of the Modbus Master

According to the standard, a total of three repeaters may be used so that a maximum of 121 slaves are possible per serial interface on a master.

4.1 Modbus Example

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

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

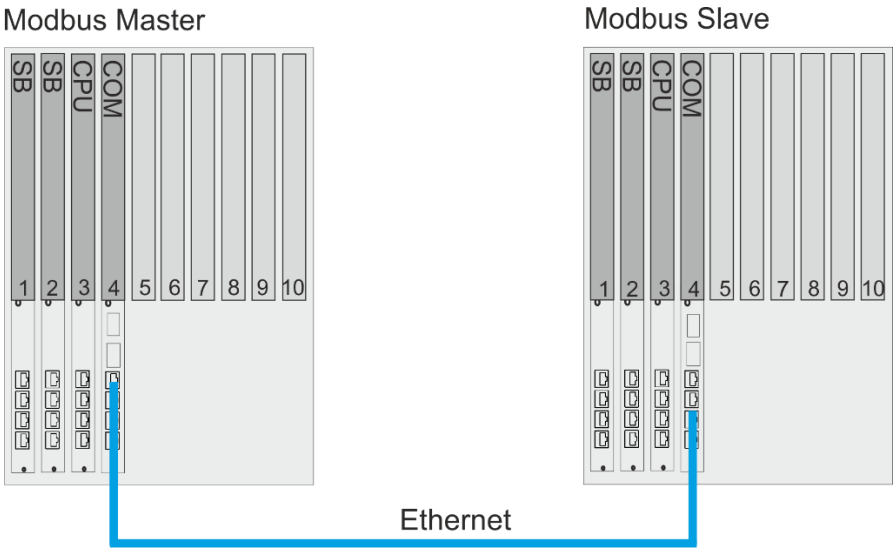


Figure 1: Communication via Modbus TCP/IP

For this example, the following global variables must be created in SILworX:

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

4.1.1 Configuring the Modbus TCP Slave

To create a new Modbus slave

1. In the structure tree, open **Configuration, Resource, Protocols.**
2. Right-click **Protocols** and select **New, Modbus Slave Set** from the context menu to add a new Modbus slave set.
3. Select **Edit** from the context menu of the Modbus slave set, open the **Modbus Slave Set Properties**, and retain the default values.
4. Select the **Modbus slave** tab and perform the following actions:
   - Select **COM Module**.
   - Activate **Enable TCP**.
   - The remaining parameters retain the default values.

To configure the bit input variables of the Modbus slave

i The Boolean variables that the master addresses bit by bit are entered in the Bit Variables tab (function code 1, 2, 5, 15).

1. Select **Edit**, **Bit Variables** from the context menu of the Modbus slave.
2. Drag the following global variables from the **Object Panel** onto the **Bit Inputs** area.

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

3. Right-click a free space in the **Register Inputs** area to open the context menu, and select **New Offsets** to renumber the variable offsets.

To configure the register input variables of the Modbus slave

i The variables that the master addresses 16 register by register are entered in the Register Variables tab (function code 3, 4, 6, 16, 23).

1. Select **Edit**, **Register Variables** from the context menu of the Modbus slave.
2. Drag the following variables from the **Object Panel** onto the **Register Inputs** area.

Register address	Register variable	Type
0	Master->Slave_WORD_00	WORD
1	Master->Slave_WORD_01	WORD

3. Right-click a free space in the **Register Inputs** area to open the context menu, and select **New Offsets** to renumber the variable offsets.

To configure the register output variables of the Modbus slave

1. Select **Edit**, **Register Variables** from the context menu of the Modbus slave.
2. Drag the following variables from the **Object Panel** onto the **Register Outputs** area.

Register address	Register variable	Type
0	Slave->Master_WORD_00	WORD
1	Slave->Master_WORD_01	WORD

3. Right-click a free space in the **Register Outputs** area to open the context menu and then click **New Offsets** to renumber the variable offsets.

To check the Modbus TCP slave configuration

1. Select **Verification** from the context menu of the Modbus TCP master.
2. Thoroughly verify the messages displayed in the logbook and correct potential errors.

4.1.2 Configuring the Modbus TCP Master

To create the Modbus master

1. In the structure tree, open **Configuration, Resource, Protocols.**
2. Select **New**, **Modbus Master** from the context menu of protocols to add a new Modbus master.
3. Select **Properties**, **General** from the context menu of the Modbus master.
4. Select **COM Module**.
   The remaining parameters retain the default values.

To create the connection to the Modbus TCP slave in the Modbus master

1. In the structure tree, open **Resource**, **Protocols**, **Modbus Master**, **Ethernet Slaves.**
2. Right-click **Ethernet Slaves** and select **New** from the context menu.
3. From the list, select **TCP/UDP Slaves** and click **OK** to confirm.
4. Configure the TCP/UDP slave in the Modbus master:
   - Click **Edit** to assign the system variables, see Chapter 4.6.2.
   - **Click** Properties **to configure the properties, see Chapter** 4.6.3**.**
     Enter the **IP Address** of the TCP/UDP slave in the slave's properties.

The remaining parameters retain the default values.

To configure the request telegram for writing the bit output variables

1. Right-click **TCP/UDP Slave** and select **New** from the context menu.
2. From the list, select the request telegram **Write Multiple Coils (15)** request telegram.
3. Right-click **Write Multiple Coils (15)** and select **Properties** from the context menu.
   - Enter **0 in the start address of the write area**
4. Right-click **Write Multiple Coils (15)** and select **Edit** from context menu.
5. Drag the following variables from the **Object Panel** onto the **Output Variables** tab.

Offset	Bit variable	Type
0	Master->Slave_BOOL_00	BOOL
1	Master->Slave_BOOL_01	BOOL
2	Master->Slave_BOOL_02	BOOL

6. Right-click a free space in the **Output Variables** area to open the context menu and click **New Offsets** to renumber the variable offsets.

To configure the request telegram for writing the register output variables

1. Right-click **TCP/UDP Slave** and select **New** from the context menu.
2. From the list, select **Write Multiple Registers (16)** request telegram.
3. Right-click **Write Multiple Register (16)** and select **Properties** from the context menu.
   - Enter **0 in the start address of the write area**
4. Right-click **Write Multiple Registers (16)** and select **Edit** from the context menu.
5. Drag the following variables from the **Object Panel** onto the **Output Variables** tab.

Offset	Register variable	Type
0	Master->Slave_WORD_00	WORD
1	Master->Slave_WORD_01	WORD

6. Right-click a free space in the **Output Variables** area to open the context menu and click **New Offsets** to renumber the variable offsets.

To define the request telegram for reading the input variables in the Modbus master

1. Right-click **TCP/UDP Slave** and select **New** from the context menu.
2. From the list, select the **Read Holding Registers (03)** request telegram.
3. Right-click **Read Holding Registers (03)** and select **Properties** from the context menu.
   - **Enter** 0 **in t**he start address of the read area.
4. Right-click **Read Holding Registers (03)** and select **Edit** from the context menu.
5. Select the following variables in the **Object Panel** and drag them onto the **Input Variables** tab.

Offset	Register variable	Type
0	Slave->Master_WORD_00	WORD
1	Slave->Master_WORD_01	WORD

6. Right-click a free space in the **Input Variables** area to open the context menu and click **New Offsets** to renumber the variable offsets.

To check the Modbus TCP master configuration

1. Open the context menu for the Modbus TCP master and click **Verification**.
2. Thoroughly verify the messages displayed in the logbook and correct potential errors.

To create the code for the controllers

1. Start the code generator for the master and slave resource.
2. Make sure that the code was generated without error.
3. Load the codes into the master and slave controllers respectively.

4.2 Example of Alternative Register/Bit Addressing

In this example, the configuration defined in Chapter 4.1 is extended by 16 Boolean variables in the Register Area. The 16 Boolean variables are read with the request telegram **Write Multiple Coils (15)**, see also Chapter 5.11.

To configure input variables in the Modbus slave

1. Select **Edit**, **Register Variables** from the context menu of the Modbus slave.
2. Drag the 16 new Boolean variables from the **Object Panel** onto the **Register Inputs** area.

Register address	Register variable	Type
0	Master->Slave_WORD_00	WORD
1	Master->Slave_WORD_01	WORD
2	Master->Slave_BOOL_03 ..._18	BOOL

Add one again

3. Right-click a free space in the **Register Inputs** area to open the context menu, and select **New Offsets** to renumber the variable offsets.

To configure the alternative register/bit addressing in the Modbus slave

1. Select **Edit, Offsets** from the context menu of the Modbus slave and activate **Use Alternative Register/Bit Addressing.**
2. In this example, use the following offsets for the alternative areas:

Register Area Offset Bits Input	1000
Register Area Offset Bits Output	1000
Bit Area Offset Register Input	8000
Bit Area Offset Register Output	8000

i To use the Modbus request telegram **Write Multiple Coils (15)** to access the Boolean variables in the **Register Variables** area, the variables must be mirrored in the **Bit Variables** area.

To configure the request telegram for writing the output variables (BOOL) in the Modbus master

1. Right-click **TCP/UDP Slave** and select **New** from the context menu.
2. From the list, select the request telegram **Write Multiple Coils (15)** request telegram.
3. Right-click **Write Multiple Coils (15)** and select **Properties** from the context menu.
   - Enter **8032 in the start address of the write area**.
4. Right-click **Write Multiple Coils (15)** and select **Edit** from context menu.
5. Drag the following variables from the **Object Panel** onto the **Output Variables** tab.

Offset	Mirrored register variable	Type
0…15	Master->Slave_BOOL_03..._18	BOOL

6. Right-click a free space in the **Output Variables** area to open the context menu and click **New Offsets** to renumber the variable offsets.

4.3 Menu Functions of the Modbus Master

4.3.1 Edit

The **Edit** dialog box for the Modbus master contains the **System Variables** tab:

The **System Variables** tab provides the system variables that allow the Modbus master state to be evaluated in the user program and the Modbus master to be controlled.

Element	Description
Slave Connection Error Count	Number of faulty connections with slaves that are in the Activated state. Deactivated Modbus slaves are not taken into account.
Modbus Master Activation Control	Stop or start the Modbus master from within the user program. 0: Activate 1: Deactivate (Edge triggered! The Modbus master can be activated using the PADT even if the *Modbus master activation control* is 1).
Modbus Master Bus Error	Bus error on RS485, e.g., telegram error (code unknown etc.), length error.
Modbus Master State	It indicates the current protocol state: 1: OPERATE 0: OFFLINE
Reset All Slave Errors	A change from FALSE to TRUE resets all slave and bus errors.

Table 4: System Variables for the Modbus Master

4.3.2 Properties

The **Properties** context menu function for the Modbus master is used to open the *Properties* dialog box.

The dialog box contains the **General** and **CPU/COM** tabs:

4.3.2.1 General

The **General** tab contains the name and a description for the Modbus master. This tab is also used to set the parameters for specifying whether the Modbus master should also operate as a TCP and/or a UDP gateway.

Element	Description
Type	Modbus master.
Name	Name for the Modbus master.
Description	Description for the Modbus master.
Module	Selection of the COM module within which the protocol is processed.
Activate Max. µP Budget	Activated: Use the µP budget limit from the *Max. µP Budget in [%]* field. Deactivated: Do not use the µP budget limit for this protocol.
Max. µP Budget in [%]	Maximum µP budget of the module that can be used for processing the protocols. Range of values: 1…100% Default value: 30%

Element	Description
Behavior on CPU/COM Connection Loss	If the connection of the processor module to the communication module is lost, the input variables are either initialized or are still used unchanged in the process module, depending on this parameter. For instance, if the communication module is removed when communication is running. Apply Initial Data — Input variables are reset to their initial values. Retain Last Value — The input variables retain the last value.
Enable TCP Gateway	If the TCP Modbus gateway is enabled, at least one Modbus RS485 interface must be configured.
TCP Server Port	Standard: 502 Additional TCP ports may also be configured. Observe the port assignment provided by the *Internet Corporation for Assigned Names and Numbers* (ICANN).
Maximum Number of TCP Connections Operating as a Server	Maximum number of TCP connections opened simultaneously and operating as server. Range of values: 1…64 Default value: 20
Enable UDP Gateway	If the UDP Modbus gateway is enabled, at least one Modbus RS485 interface must be configured.
UDP Port	Standard: 502 Additional UDP ports may also be configured. Observe the port assignment provided by the Internet Corporation for Assigned Names and Numbers (ICANN).
Maximum Length of the Queue	Length of the gateway queue for other masters' request telegrams that have not been answered yet. This option is only taken into account if a gateway has been activated. Range of values: 1…20 Default value: 3

Table 5: General Properties of the Modbus Master

4.3.2.2 CPU/COM

The default values for the parameters provide the fastest possible exchange of Modbus data between the COM module and the processor module within the HIMA controller.

These parameters should only be changed if it is necessary to reduce the COM and CPU loads for an application, and the process allows this change.

i Only experienced programmers should modify the parameters.

Increasing the COM and CPU refresh rate means that the effective refresh rate of the Modbus data is also increased. The system time requirements must be verified.

Element	Description
Process Data Refresh Rate [ms]	Refresh rate in milliseconds at which the COM and CPU exchange protocol data. If the *Process Data Refresh Rate* is zero or lower than the cycle time for the controller, data is exchanged as fast as possible. Range of values: 0…(2^{31}–1) Default value: 0
Allow Multiple Fragments per Cycle	Activated: Transfer of all protocol data from the CPU to the COM within a CPU cycle. Deactivated: Transfers all of the protocol data from the CPU to the COM, distributed over multiple CPU cycles, each with 1100 bytes per data direction. This may also cause the cycle time of the controller to be reduced. Default value: Activated

Table 6: COM/CPU Parameters

4.4 Modbus Function Codes of the Master

The Modbus function blocks can be used to read and write to individual variables or several consecutive variables.

4.4.1 Modbus Standard Function Codes

The Modbus master supports the following Modbus standard function codes:

Element	Code	Type	Description
Read Coils	01	BOOL	Read multiple variables (BOOL) from the slave. Max. length of the process data: 250 bytes (2000 coils).
READ DISCRETE INPUTS	02	BOOL	Read multiple variables (BOOL) from the slave. Max. length of the process data: 250 bytes (2000 coils).
Read Holding Registers	03	WORD	Read multiple variables of any type from the slave. Max. length of the process data: 250 bytes
Read Input Registers	04	WORD	Read multiple variables of any type from the slave. Max. length of the process data: 250 bytes
Write Single Coil	05	BOOL	Write one single signal (BOOL) in the slave. Max. length of the process data: 1 byte (1 coil)
Write Single Register	06	WORD	Write one single signal (WORD) in the slave. Max. length of the process data: 2 bytes
Write Multiple Coils	15	BOOL	Write multiple variables (BOOL) in the slave. Max. length of the process data: 246 bytes (1968 coils).
Write Multiple Registers	16	WORD	Write multiple variables of any type in the slave. Max. length of the process data: 246 bytes.
READ WRITE HOLDING REGISTERS	23	WORD	Write and read multiple variables of any type in and from the slave. Max. length of the process data: 242 bytes (Modbus master's request telegram). 250 bytes (slave's response telegram).

Table 7: Modbus Function Codes

i For further details on Modbus, refer to the *Modbus Application Protocol Specification* www.modbus.org)

4.4.2 HIMA-Specific Function Codes

HIMA specific function codes correspond to the standard Modbus function codes. The two differences are the maximum permissible process data length of 1100 bytes and the format of the request and response header.

Element	Code	Type	Description
Read Coils Extended	100 (0x64)	BOOL	Corresponds to function code 01. Read multiple variables (BOOL) from the slave's import or export[1)] area. Maximum length of the process data: 1100 bytes.
Read Discrete Inputs Extended	101 (0x65)	BOOL	Corresponds to function code 02 Read multiple variables (BOOL) from the slave's export area. Maximum length of the process data: 1100 bytes.
Read Holding Registers Extended	102 (0x66)	WORD	Corresponds to function code 03. Read multiple variables of any type from the slave's import or export[1)] area. Maximum length of the process data: 1100 bytes.
Read Input Registers Extended	103 (0x67)	WORD	Corresponds to function code 04. Read multiple variables of any type from the slave's export area. Maximum length of the process data: 1100 bytes.
Write Multiple Coils Extended	104 (0x68)	BOOL	Corresponds to function code 15. Write multiple variables (BOOL) to the slave's import area. Maximum length of the process data: 1100 bytes.
Write Multiple Registers Extended	105 (0x69)	WORD	Corresponds to function code 16. Write multiple variables of any type to the slave's import area. Maximum length of the process data: 1100 bytes.
Read/Write Multiple Registers Extended	106 (0x6A)	WORD	Corresponds to function code 23. Write and read multiple variables of any type to and from the slave's import or export area. Maximum length of the process data: 1100 bytes (Modbus master's request telegram). 1100 bytes (slave's response telegram).
[1)] The export area can only be selected in HIMA slaves.			

Table 8: HIMA-Specific Function Codes

4.4.3 Format of Request and Response Header

The request and response header of the HIMA-specific Modbus function codes are structured as follows:

Code	Request	Response
100 (0x64)	1 byte function code 0x64 2 bytes start address 2 bytes number of coils 1…8800 (0x2260)	1 byte function code 0x64 2 bytes number of bytes = N N bytes coil data (8 coils are packed in one byte)
101 (0x65)	1 byte function code 0x65 2 bytes start address 2 bytes number of discrete inputs 1…8800 (0x2260)	1 byte function code 0x65 2 bytes number of bytes = N N bytes discrete inputs data (8 discrete inputs are packed in one byte)
102 (0x66)	1 byte function code 0x66 2 bytes start address 2 bytes number of registers 1…550 (0x226)	1 byte function code 0x66 2 bytes number of bytes = N N bytes register data
103 (0x67)	1 byte function code 0x67 2 bytes start address 2 bytes number of registers 1…550 (0x226)	1 byte function code 0x67 2 bytes number of bytes = N N bytes register data
104 (0x68)	1 byte function code 0x68 2 bytes start address 2 bytes number of coils 1…8800 (0x2260) 2 bytes number of bytes = N N bytes coil data	1 byte function code 0x68 2 bytes start address 2 bytes number of coils 1…8800 (0x2260)
105 (0x69)	1 byte function code 0x69 2 bytes start address 2 bytes number of registers 1…550 (0x226) 2 bytes number of bytes = N N bytes register data	1 byte function code 0x69 2 bytes start address 2 bytes number of registers 1…550 (0x226)
106 (0x6A)	1 byte function code 0x6a 2 bytes read start address 2 bytes number of read registers 1...550 (0x226) 2 bytes write start address 2 bytes number of write registers 1…550 (0x226) 2 bytes number of write bytes = N N bytes register data	1 byte function code 0x6a 2 bytes number of bytes = N N bytes register data
[1)] The export area can only be selected in HIMA slaves.		

Table 9: Request and Response Header

4.4.4 Read Request Telegrams

The read function codes can be used to read variables from the slave.

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

To read variables, the Modbus master sends a *Read Request Telegram* to the Modbus slave.

The Modbus slave responds to the Modbus master by sending back a response telegram with the required variables.

The following read request telegrams are available:

4.4.4.1 Read Coils (01) and Extended (100)

Read multiple variables (BOOL) from the slave.

Element	Description
Type	Modbus Function *Read Coils*.
Name	Any unique name for the Modbus function
Description	Description for the Modbus function.
Start address of the read area	0…65535

Table 10: Request Telegram Read Coils

4.4.4.2 Read Discrete Inputs (02) and Extended (101)

Read multiple variables (BOOL) from the slave.

Element	Description
Type	Modbus Function *Read Discrete Inputs.*
Name	Any unique name for the Modbus function.
Description	Description for the Modbus function.
Start address of the read area	0…65535

Table 11: Request Telegram Read Discrete Inputs

4.4.4.3 Read Holding Register (03) and Extended (102)

Read multiple variables of any type from the slave.

Element	Description
Type	Modbus Function *Read Holding Registers*
Name	Any unique name for the Modbus function.
Description	Description for the Modbus function.
Start address of the read area	0…65535

Table 12: Request Telegram Read Holding Registers

4.4.4.4 Read Input Registers (04) and Extended (103)

Read multiple variables of any type from the slave.

Element	Description
Type	Modbus Function *Read Input Registers*
Name	Any unique name for the Modbus function.
Description	Description for the Modbus function.
Start address of the read area	0…65535

Table 13: Request Telegram Read Input Registers

4.4.5 Read/Write Request Telegram

For reading and writing variables, the Modbus master sends a *Read and Write Request Telegram* to the Modbus slave.

First, the Modbus master writes the write variables to the defined import area of the Modbus slave.

Afterwards, the Modbus master reads the read signals from the defined export area of the Modbus slave.

The following read and write request telegrams are available:

i In the read and write request telegram, the write and read functions are independent of one another.

However, the r*ead and write request telegram* is often used such that the variables written by the Modbus master are read back. This ensures that the transferred variables were written correctly.

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

Write and read multiple variables of any type in and from the slave's import area.

Element	Description
Type	Modbus function *Read Write Holding Registers*.
Name	Any unique name for the Modbus function
Description	Description for the Modbus function.
Start address of the read area	0…65535
Start address of the write area	0…65535

Table 14: Read Write Holding Register

4.4.6 Write Request Telegram

If the write function codes are used, variables may only be written in a slave's import area.

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

To write variables, the Modbus master sends a *Write Request Telegram* to the Modbus slave.

The Modbus slave writes the received variables to its import area.

The variables that a Modbus master writes to a Modbus slave must be entered in the *Variable Connections* dialog box for a *write request telegram*.

The following write request telegrams are available:

4.4.6.1 Write Multiple Coils (15) and Extended (104)

Write multiple variables (BOOL) to the slave's import area.

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

Table15: The Write Multiple Coils Request Telegram

4.4.6.2 Write Multiple Registers (16) and Extended (105)

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

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

Table 16: The Write Multiple Registers Request Telegram

4.4.6.3 Write Single Coil (05)

Write one single variable (BOOL) to the slave's import area.

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

Table 17: Write Single Coil (05) Request Telegram

4.4.6.4 Write Single Register (06)

Write one single variable (WORD) to the slave's import area.

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

Table 18: Write Single Register Request Telegram

4.5 Ethernet Slaves (TCP/UDP Slaves)

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

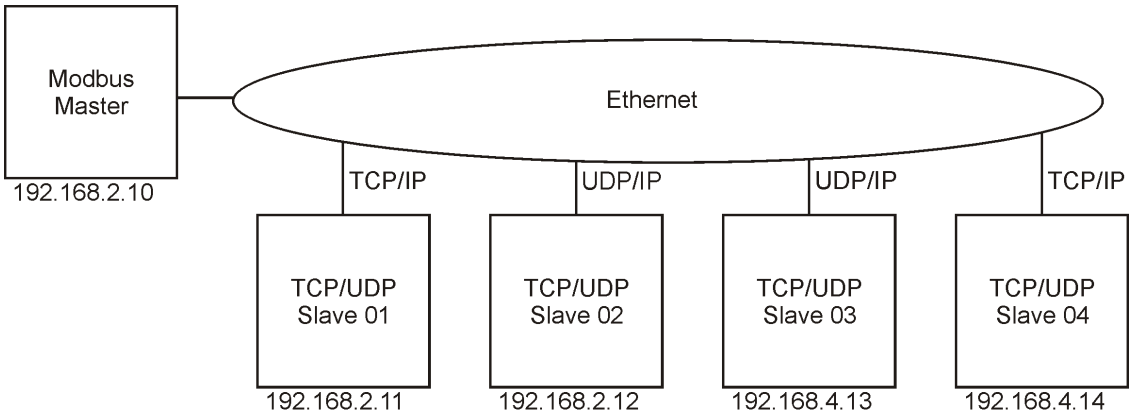


Figure 2: Modbus Network

To create a new connection to a TCP/UDP slave in the Modbus master

1. In the structure tree, open **Resource, Protocols, Modbus Master, Ethernet Slaves**.
2. Right-click **Ethernet Slaves** and select **New** from the context menu.
3. From the list, select **TCP/UDP Slaves** and click **OK** to confirm.
4. To configure the TCP/UDP slave in the Modbus master
   - Click **Edit** to assign the system variables, see Chapter 4.5.1.
   - Click **Properties** to configure the properties, see Chapter 4.5.2.
   -

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

With its telegrams to the Modbus TCP slave, the HIMA Modbus TCP master sends the IP address and a Modbus slave address (Unit Identifier), which is always FF (255).

4.5.1 System Variables for TCP/UDP Slaves

The *System Variables* tab provides the system variables that allow the TCP/UDP slave state to be evaluated in the user program and the TCP/UDP slave to be controlled.

The following status variables can be used to evaluate the TCP/UDP slave status from within the user program:

Element	Description
Modbus Slave Activation Control	The user program activates or deactivates the TCP/UDP slave using this function. 0: Activate 1: Deactivate (Edge triggered! The Modbus slave can still be activated using the PADT, even if *Modbus Slave Activation Control* is set to 1.
Modbus Slave Error	Error code The error codes 0x01…0x0b correspond to the Exception Codes of the Modbus protocol specification. 0x00: No error. Exception Codes: 0x01: Invalid function code. 0x02: Invalid addressing. 0x03: Invalid data. 0x04: Not used. 0x05: Not used. 0x06: Device busy (only gateway). 0x08: Not used. 0x0a: Not used. 0x0b: No response from slave (only gateway) HIMA-specific codes: 0x10: Defective frame received. 0x11: False frame transaction ID received. 0x12: Unexpected response received. 0x13: Response about invalid connection received. 0x14: False response to a write request. 0xff: Slave timeout.
Modbus Slave State	Connection status of the TCP/UDP slave: 0: Deactivated 1: Not connected. 2: Connected

Table 19: System Variables for TCP/UDP Slaves

4.5.2 TCP/UDP Slave Properties

To configure the connection to the TCP/UDP slave, the following parameters must be set in the Modbus master.

Element	Description
Type	TCP/UDP slave.
Name	Any unique name for the TCP/UDP slave.
Description	Any unique description for the TCP/UDP slave.
Master-Slave Data Exchange [ms]	Time interval for exchanging data with this slave 1 to (2^{31}-1). If the slave could not be reached after achieving the *Maximum Number of Resends,* the value for *Master-Slave Data Exchange* is set four times higher.
TCP Connection only if Required	If the type of transmission protocol is TCP, this parameter can be used to define if the connection to the slave should be automatically closed whenever data is exchanged: TRUE: Close the connection. FALSE: Do not close connection Default value: FALSE
Receive Timeout [ms]	Receive timeout [ms] for this slave. Once this time period has expired, a resend is attempted.
IP Address	IP address of the TCP/UDP slave.
Port	Standard: 502 Additional TCP/UDP ports may also be configured. Observe the port assignment provided by the Internet Corporation for Assigned Names and Numbers (ICANN).
Type of Communication IP Protocol	TCP or UDP. Default value: TCP
Maximum Number of Resends	Maximal number of resends attempted if the slave does not respond. The number of resends can be freely set. (0…65535). With TCP/IP, the value is always set to 0 and cannot be changed. HIMA recommends setting a value between 0 and 8.

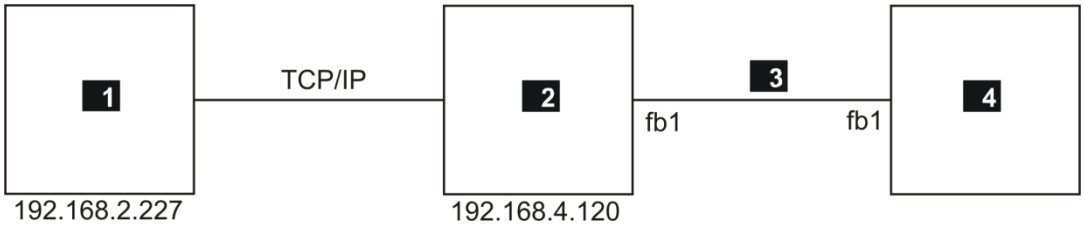
Table 20: Configuration Parameters

4.6 Modbus Gateway (TCP/UDP Gateway)

The Modbus master RTU license is required to enable the Modbus master to operate as Modbus gateway. In this mode, master's requests that the gateway receives via Ethernet are forwarded to the RS485 slaves connected to the gateway. Accordingly, the slave's responses are forwarded to the Modbus master through the gateway.

Up to 121 serial Modbus slaves can be addressed per serial interface.

The slave's address range is 1...247. Even if only the Modbus gateway is used, a Modbus master license is required for Modbus master 2 (Modbus gateway).



1 Modbus master 1
2 Modbus master 2 (Modbus gateway)
3 Serial Modbus
4 Serial slave

Figure 3: Modbus Gateway

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

Modbus Master 1

To create the connection to the Modbus gateway in Modbus master 1

1. In the structure tree, open **Resource, Protocols, Modbus Master**.
2. Right-click **Modbus Master** and select **New** from the context menu.
3. From the list, select Modbus Gateway and click **OK** to confirm.
4. Configure the Modbus gateway in Modbus Master 1:
   - Click **Properties** to configure the properties, see Chapter 4.7.3.
   - Enter the **IP Address** of Modbus master 2 (Modbus gateway) in the properties.

To create the connection to the gateway slave in Modbus master 1

In Modbus master 1, the serial slave must be created as gateway slave.

1. In the structure tree, open **Resource, Protocols, Modbus Master, Modbus Gateway**.
2. Right-click **Modbus Gateway** and select **New** from the context menu.
3. From the list, select **Gateway Slave** and click **OK** to confirm.
4. To configure the gateway slave in Modbus master 1:
   - Click **Edit** to assign the system variables, see Chapter 4.6.2.
   - Click **Properties** to configure the properties, see Chapter 4.6.3.
   - Enter the **Serial Address** of the gateway slave in the slave's properties.

To define input and output variables to the serial slave in Modbus master 1

1. Right-click **Gateway Slave** and select **New** from the context menu.
2. Select the required **request telegrams** from the list.
3. Right-click the individual **request telegram** and select **Edit** from the context menu. Enter the input or output variables in the *Process Variables* tab.

Modbus Master 2 (Modbus Gateway)

The gateway function must be activated in the properties of Modbus master 02. In doing so, the gateway slave configured in master 01 is connected to the serial slaves.

To activate the gateway function in Modbus master 2

1. In the structure tree, open **Resource, Protocols, Modbus Master**.
2. Right-click **Modbus Master** and select **Properties** from the context menu.
3. Activate the **Enable TCP Gateway** parameter to allow the Modbus master to additionally operate as TCP gateway.
4. Activate the **Enable UDP Gateway** parameter to allow the Modbus master to additionally operate as UDP gateway.

To configure the serial Modbus in Modbus master 2

1. In the structure tree, open **Resource, Protocols, Modbus Master**.
2. Right-click **Modbus Master** and select **New** from the context menu.
3. From the list, select **Serial Modbus** and click **OK** to confirm.
4. Configure the serial Modbus:
   - Click **Properties** and enter the interface, the baud rate, etc.

To configure the connection to the serial slave in Modbus master 2

1. In the structure tree, open **Resource, Protocols, Modbus Master, Serial Modbus**.
2. Right-click **Serial Modbus** and select **New** from the context menu.
3. From the list, select **Modbus Slave** and click **OK** to confirm.
4. Configure the Modbus slave**:**
   - Select **Properties** and enter the **Slave Address** of the serial slave.

Serial Slave

To configure the serial Modbus slave

1. **In the structure tree, open Resource, Protocols, Modbus Slave**
2. Right-click **Modbus Slave** and select **Edit** from the context menu.
3. Configure the Modbus slave**:**
   - Select **Properties** and enter **Slave Address** of the serial slave.

4.6.1 Gateway Properties

The Modbus gateway allows the Modbus master to communicate with its Modbus slave.

To configure the connection to the Modbus gateway, the following parameters must be set in the Modbus master.

Element	Description
Type	Modbus gateway.
Name	Any unique name for the gateway.
Description	Any unique description for the TCP/UDP slave.
Communication IP Protocol	TCP or UDP Default value: TCP
IP Address	Gateway's IP address that the Modbus master should use to communicate with its Modbus slave. Default value: (0.0.0.0)
Port	Default value: 502

Table 21: Connection Parameters for the Modbus Gateway

4.6.2 System Variables for the Gateway Slave

The editor contains three status variables:

Element	Description
Modbus Slave Activation Control	Using this function, the user program can activate or deactivate the gateway slave. 0: Activate 1: Deactivate (Edge triggered! The Modbus slave can still be activated using the PADT, even if *Modbus Slave Activation Control* is set to 1).
Modbus Slave Error	Parameters: the same as for TCP/UDP slave, see Chapter4.5.1.
Modbus Slave State	Connection status of the gateway slave: 0: Deactivated 1: Not connected 2: Connected

Table 22: Status Variables for the Gateway Slave

4.6.3 Gateway Slave Properties

To configure the connection to the gateway slave, the following parameters must be set in the Modbus master.

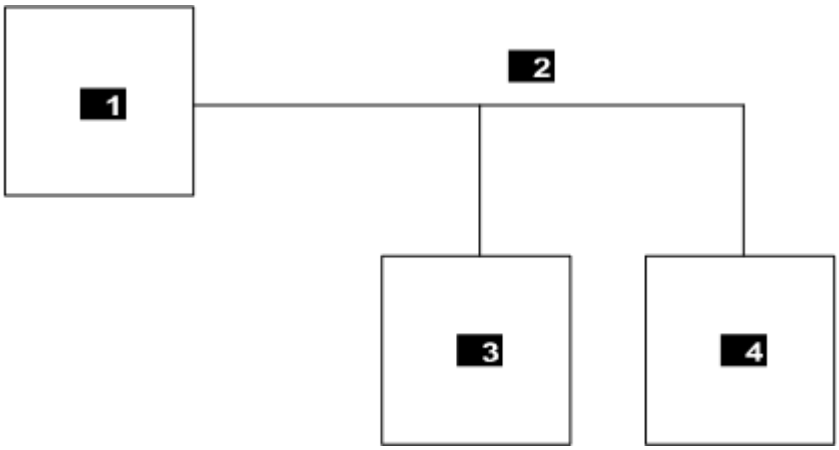
Element	Description
Type	Gateway slave.
Name	Any unique name for the gateway slave.
Description	Any unique description for the gateway slave.
Slave Address	1…247
The remaining parameters are the same as for TCP/UDP slave, see Chapter4.5.2.	

Table 23: Connection Parameters for the Gateway Slave

4.7 Serial Modbus

The Modbus master can communicate with up to 247 serial slaves. According to the standard, a total of three repeaters may be used so that a maximum of 121 bus subscribers are possible per serial interface on a master.

i For further details on the pin assignment of the X-COM module's D-sub connectors (fb1, fb2), refer to the communication manual (HI 801 101 E).

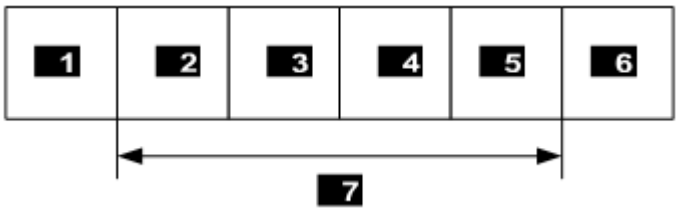


1 Modbus master 01 (fb1, fb2 or fb3)
2 Serial Modbus
3 Serial slave 1 (address 11)
4 Serial slave 2 (address 12)

Figure 4: Serial Modbus

The Modbus master supports data transmission in RTU (Remote Terminal Unit) format.

The RTU telegram frame starts and ends with the idle characters set by the user (default value: 5 idle characters).



1 Beginning of the frame (5 char)
2 Address (8-bit)
3 Function (8-bit)
4 Data (N * 8-bit)
5 CRC checksum (16-bit)
6 End of the frame (5 char)
7 Modbus telegram

Figure 5: Modbus Telegram

To create a serial Modbus in the Modbus master

1. In the structure tree, open **Resource, Protocols, Modbus Master, Serial Modbus**.
2. Right-click **Serial Modbus** and select **New** from the context menu.
3. From the list, select **Modbus Slave** and click **OK** to confirm.
4. To configure the Modbus slave in the Modbus master:
   - Click **Edit** to assign the system variables, see Chapter 4.7.2.
   - Click **Properties** to configure the properties, see Chapter 4.7.3.

4.7.1 Serial Modbus Properties

To configure the serial Modbus master, the following parameters must be set.

Element	Description
Type	Serial Modbus
Name	The serial Modbus name may be selected by the user.
Description	Any unique description for the serial Modbus.
Interface	Fieldbus interface that should be used for the Modbus master (fb1, fb2).
Baud Rate [bps]	Possible value for transfer rate for RS485: 300 Bit/s 600 Bit/s 1200 Bit/s 2400 Bit/s 4800 Bit/s 9600 Bit/s 19200 Bit/s 38400 bit/s (maximum baud rate for HIMax prior to V4). 57600 bit/s (maximum baud rate for HIMax V4 and higher). 62500 bit/s (HIMatrix/ HIQuad X). 76800 bit/s (HIMatrix/ HIQuad X). 115000 Bit/s (HIMatrix/ HIQuad X)
Parity	None odd Even Default value: Even
Stop Bits	Standard (adapts the number of stop bits to the parity: with parity = 1 stop bit, no parity = 2 stop bits). One stop bit Two stop bits Default value: Default
Number of Idle Chars	Number of idle characters at the start and the end of a RTU telegram frame. Range of values: 0…65535 Default value: 5 characters

Table 24: Parameters for the Serial Modbus Master

4.7.2 System Variables for the Modbus Slave

The editor Edit contains three status variables (system variables).

Element	Description
Modbus Slave Activation Control	Activate or deactivate the Modbus slave in the user program. 0: Activate 1: Deactivate (Edge triggered! The Modbus slave can still be activated using the PADT, even if *Modbus Slave Activation Control* is set to 1).
Modbus Slave Error	Parameters: the same as for TCP/UDP slave, see Chapter 4.5.2.
Modbus Slave State	Connection status of the Modbus slave: 0: Deactivated 1: Not connected 2: Connected

Table 25: System Variables in the Modbus Slave

4.7.3 Modbus Slave Properties

To configure the connection to the serial slave, the following parameters must be set in the Modbus master.

Element	Description
Type	Modbus slave.
Name	The Modbus slave name may be selected by the user
Description	Any unique description for the Modbus slave
Slave Address	1…247
The remaining parameters are the same as for TCP/UDP slave, see Chapter4.5.2.	

Table 26: Connection Parameters for the Modbus Master

i The receive timeout in the serial Modbus slave depends on the configured transfer rate.

If the baud rate is 19200 [bps] or higher, the default value may be used as receive timeout. If the baud rate is lower than 19200 [bit/s], the value for Receive Timeout must be increased.

4.8 Control Panel (Modbus Master)

The Control Panel can be used to verify and control the settings for the Modbus master. It also displays details on the master's current state (e.g., master state, etc.).

To open the Control Panel for monitoring the Modbus master

1. Right-click the **Hardware** structure tree element and select **Online** from the context menu.
2. In the **System Login** window, enter the access data to open the online view for the hardware.
3. Double-click **COM Module** and select **Modbus Master** in the structure tree.

4.8.1 Context Menu (Modbus Master)

The following commands can be chosen from the context menu of the selected Modbus master:

Offline

This command is used to stop the Modbus master.

Operate

This command is used to start the Modbus master.

Reset Statistics

This command is used to reset the statistical data (e.g., number of bus errors, min. or max. cycle time) to zero.

4.8.2 View Box (Modbus Master)

The view box displays the following values of the selected Modbus master.

Element	Description
Name	Modbus master name
Master State	It indicates the current protocol state: Operate OFFLINE
Number of Bus Errors	Counter for bus errors.
Disturbed Connections	Counter for disturbed connections.
µP Load (planned)	See Chapter 4.3.2.
µP Load (actual)	

Table 27: View Box of the Modbus Master

4.9 Control Panel (Modbus Master->Slave)

The Control Panel is used to verify and activate/deactivate the settings for the communication partner. Details on the current status (e.g., slave state, etc.) of the communication partner are displayed.

To open the Control Panel for monitoring the Modbus connection

1. Right-click the **Hardware** structure tree element and select **Online** from the context menu.
2. In the **System Login** window, enter the access data to open the online view for the hardware.
3. Double-click **COM Module** and select the **Modbus Master**, **Slave** in the structure tree.

4.10 FBx LED Function in the Modbus Master

The FBx LED of the corresponding fieldbus interface indicates the state of the Modbus protocol. The states of the FBx LED are specified in the following table:

LED	Color	Status	Description
FBx	Yellow	Blinking	The Modbus master protocol is active and is exchanging data with the Modbus slave.
		Off	The Modbus master protocol is not active! This means that the controller is in STOP or no Modbus master is configured.

Table 28: The FBx LED

4.11 Function of the FAULT LED in the Modbus Master (HIMax only)

The FAULT LED of the corresponding fieldbus interface indicates a failure of the Modbus protocol. The states of the FAULT LED are specified in the following table:

LED	Color	Status	Description
Fault	Red	Blinking1[1)]	The following events result in a malfunction. ▪ Incorrect response or error message from the slave received. ▪ Timeout for one or more slaves. ▪ The protocol load attains the maximum of the configured computing time budget. The protocol processing is constrained by the load limitation. This can lead to considerable impairments of the protocol function. Either the protocol settings or the settings of the computing time budget must be adjusted. If no faults occur for a period longer than 5 seconds, the state changes to "Protocol not disturbed".
		Off	The Modbus master protocol is not disturbed.
[1)] Blinking frequency: Long (600 ms) on, long (600 ms) off.			

Table 29: The FAULT LED

5 Modbus Slave

The Modbus slave can simultaneously use the serial interface (RS485) and the TCP/UDP (Ethernet) to operate various Modbus masters.

i HIMA recommends using the Modbus slave V2 protocol for new projects. This is not based on a new Modbus specification, but is an enhanced HIMA variant concerning the internal processing of the protocol data on HIMA controllers. The standard Modbus function codes remain the same, the HIMA specific Modbus function codes no longer exist. For further information, refer to the Modbus slave V2 manual (HI 801 475 E).

5.1 Equipment and System Requirements

Element	Description
HIMA controller	HIMax with X-COM 01 module HIQuad X with F-COM 01 module HIMatrix
CPU module	The Ethernet interfaces on the processor module may not be used for Modbus TCP.
COM module	Ethernet 10/100BaseT D-sub connectors FB1 and FB2 If Modbus RTU is used, each of the serial fieldbus interfaces (FB1 and FB2) used on the COM module must be equipped with a HIMA RS485 submodule. For further details on the interface assignment, refer to the communication manual (HI 801 101 E).
Activation	Each of the two Modbus slave functions must be enabled individually; refer to the communication manual (HI 801 101 E) for details. Modbus slave RTU (RS485) Modbus slave TCP 2 licenses are required for the redundant Modbus slave, one for each communication module. A redundant Modbus slave configuration is only supported for HIMax and HIQuad X controllers.

Table 30: Equipment and System Requirements for the Modbus Slave

5.2 Modbus Slave (Properties)

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Modbus Slave</td><td>One TCP Modbus slave and one RTU Modbus slave (RS485) can be configured for each COM module.</td></tr>
<tr><td>Redundancy</td><td>Number of redundant Modbus slave communication module pairs that can be operated in a HIMA system.
HIMax: Max. 10
HIQuad X: Max. 5
As long as the Modbus slave communication module pair operates redundantly, both communication modules are used to exchange the same input and output data with the Modbus master, see Chapter 5.5.</td></tr>
<tr><td>Number of Master Accesses</td><td>RTU: Due to the RS485 transmission technology, only one Modbus master can have access to an installed RS485 interface.
TCP: A maximum of 20 Modbus masters can access the slave.
UDP: Unlimited number of Modbus masters can access the slave.</td></tr>
<tr><td>Max. Size of Send Data</td><td rowspan="2">Refer to the communication manual (HI 801 101 E).</td></tr>
<tr><td>Max. Size of Receive Data</td></tr>
<tr><td>Display format of the Modbus data</td><td>The HIMA controllers use the big endian format for data. Example: 32-bit data (e.g., DWORD, DINT):
<table>
<tr><td>32-bit data (hex)</td><td colspan="4">0x12345678</td></tr>
<tr><td>Byte number (from the left)</td><td>0</td><td>1</td><td>2</td><td>3</td></tr>
<tr><td>Big endian (HIMax, HIQuad X, HIMatrix)</td><td>12</td><td>34</td><td>56</td><td>78</td></tr>
<tr><td>Middle endian (H51q)</td><td>56</td><td>78</td><td>12</td><td>34</td></tr>
<tr><td>Little endian</td><td>78</td><td>56</td><td>34</td><td>12</td></tr>
</table></td></tr>
</table>

Table 31: Properties of the Modbus Slave

5.3 Configuring the Modbus TCP Slave

To create a new Modbus slave

1. In the structure tree, open **Configuration, Resource, Protocols.**
2. Right-click **Protocols** and select **New, Modbus Slave Set** from the context menu to add a new Modbus slave set.
3. Select **Edit** from the context menu of the Modbus slave set, open the **Modbus Slave Set Properties**, and retain the default values.
4. Select the **Modbus slave** tab and perform the following actions:
   - Select **COM Module**.
   - Activate **Enable TCP**.
   - The remaining parameters retain the default values.

i If the Ethernet interfaces are used as transmission channel, the HIMA controller and the communication partner must be located in the same subnet, or, if a router is used, they must have the corresponding routing settings.

Chapter 4.1 provides an example of how to configure the connection between an HIMA Modbus TCP slave and an HIMA Modbus TCP master.

5.4 Configuring the Redundant Modbus TCP Slave

To create a redundant Modbus slave

1. In the structure tree, open **Configuration, Resource, Protocols, Modbus Slave Set**.
2. Select **Edit** from the context menu of the Modbus slave set, open the **Modbus Slave Set Properties**, and configure the following settings:
   - **Activate Set Redundancy Operation.**
   - ☑ The **Modbus Slave Redundant** tab is automatically added.
3. Select the **Modbus Slave Redundant** tab and perform the following actions:
   - Select **COM Module**.
   - Activate **Enable TCP**.

The remaining parameters retain the default values.

i The send and receive variables assigned in the Modbus slave set are valid for both Modbus slaves.

5.5 Rules for Redundant Modbus TCP Slaves

To redundantly operate the HIMax or HIQuad X Modbus slave communication modules, HIMA recommends configuring the system redundantly, refer to the system manual (HI 801 001 E) for more details.

Already at the occurrence of the first fault in the HIMax or HIQuad X system, it would otherwise no longer be ensured that the Modbus slave communication module pairs behave consistently towards their external partner (Modbus master).

5.5.1 Process Data Volume Redundant Modbus Slaves

The process data volume that the user can configure, is reduced by half. In HIMax from 128 kB to 64 kB and in HIQuad X from 64 kB to 32 kB for each direction (input and output). Refer to the information on the system quantity structure for protocols in the communication manual (HI 801 101 E).

5.5.2 Slots Allowed for the Redundant Modbus Slave HIMax COM Modules

The system bus segments (1…3) on the base plate must be taken into account to minimize the risk of collisions on the HIMax system bus. For this reason, the redundant Modbus slave communication modules should only be inserted in the same segment of a base plate in the following slots:

Segment	Slot
1	3…6 (as long as no processor module has been planned).
2	7…14
3	15…18

Table 32: Slots Allowed for the Redundant Modbus Slave HIMax COM Modules

5.5.3 Redundant Modbus Slave COM Modules in Different Base Plates

A maximum of two redundant Modbus slave communication module pairs may be operated if their redundant Modbus slave communication modules are located in different base plates (0…15).

In this case, these redundant Modbus slave communication modules may only be located in adjacent racks.

Furthermore, 8 additional Modbus slave communication module pairs may be operated on the same HIMax system in accordance with the rules specified in Chapter 5.5.2.

NOTICE

System malfunction possible!

Only use slots for redundant Modbus slave communication modules in compliance with the stated rules!

Between an X-COM module and the X-CPU modules, a maximum additional delay of 50 µs is allowed (due to cable length, switches, etc).

Recommendation: Operate the X-COM modules as close as possible to the X-CPU modules (e.g., rack 0, rack 1).

5.6 Menu Functions of the HIMA Modbus Slave Set

The **Edit** context menu function for the Modbus slave set is used to open the *Modbus Slave Set* dialog box. The dialog box contains the following tabs:

5.6.1 Modbus Slave Set Properties

The following parameters for the Modbus slave are set in the **Modbus Slave Set Properties** tab.

Element	Description
Name	Name of the Modbus slave set
Activate Max. µP Budget	Activated: Use the µP budget limit from the *Max. µP Budget in [%]* field. Deactivated: Do not use the µP budget limit for this protocol.
Max. µP Budget in [%]	Maximum µP load of the COM module that can be used for processing the protocols. Range of values: 1…100% Default value: 30%
Set Redundancy Operation	Activated: Redundancy operation The *Modbus Slave Redundant* tab is activated. Deactivated: Mono operation The *Modbus Slave Redundant* tab is deactivated. Default value: deactivated
Maximum Response Delay [ms]	Time period after the reception of a request within which the Modbus slave may respond. Range of values: 0…(2^{31}-1) [ms] Default value: 5000 ms (0 = No limitation)
Area for Reading Function Codes 1, 3, 100, 102	This parameter defines from which data field the data should be read for function code 1, 3, 100, 102. Range of values: ▪ Import area ▪ Export Area (compatible with H51q)

Element	Description
Area for Reading Function Code 23, 106	The user can specify the Modbus slave's area from which the function code 23 should read. Import area: The master has read/Write access to the slave's import area. Export area: The master reads from the slave's export area and writes to the slave's import area. Notice: Writing and reading is performed within a single CPU cycle. This means that the read data was provided during the last CPU cycle.
COM: Values at Connection Loss to Master	If the connection of the communication module to the Modbus master is lost, the input variables are forwarded to the processor module as initialized or unchanged, depending on this parameter. Apply Initial Data: Input variables are reset to their initial values. Retain Last Value: The input variables retain the last value.
CPU: Values at Connection Loss to Master	If the connection of the processor module to the communication module is lost, the input variables are either initialized or are still used unchanged in the process module, depending on this parameter. For instance, if the communication module is removed when communication is running. The Same Behavior as the COM to the Master: See the settings of the *COM: Values at Connection Loss to Master* parameter. Retain Last Value: The input variables retain the last value. Default value: The Same Behavior as COM to Master
Use the Alternative Register / Bit Addressing	Activated: Use the alternative addressing. Deactivated: Do not use the alternative addressing Default value: Deactivated, see Chapter 5.11.
Register Area Offset Bits (Input)	Range of values: 0…65535 Default value: 0
Register Area Offset Bits (Output)	Range of values: 0…65535 Default value: 0
Bit Area Offset Register (Input)	Range of values: 0…65535 Default value: 0
Bit Area Offset Bits (Output)	Range of values: 0…65535 Default value: 0
Refresh Rate [ms]	Refresh rate in milliseconds at which the COM and CPU exchange protocol data. If the *Refresh Rate* is zero or less than the cycle time for the controller, data is exchanged as fast as possible. Range of values: 0…(2^{31}–1) Default value: 0
Allow Multiple Fragments per Cycle	Activated: Transfer of all protocol data from the CPU to the COM within a CPU cycle. Deactivated: Transfers all of the protocol data from the CPU to the COM, distributed over multiple CPU cycles, each with 1100 bytes per data direction. This may also cause the cycle time of the controller to be reduced. Default value: Activated

Table 33: The Modbus Slave Set Properties Tab

5.6.2 Register Variables

The variables that the master addresses register by register (16 bits) are entered in the **Register Variables** tab (function code 3, 4, 6, 16, 23, 102, 103, 105, 106).

The input and output variables are dragged from the Object Panel onto the *Register Inputs* or *Register Outputs* fields.

- All the variables that the Modbus slave receives from the Modbus master are entered in the *Register Inputs* field.
- All the variables that the Modbus slave sends to the Modbus master are entered in the *Register Outputs* field.

To generate new offsets

1. Right-click the **Register Inputs** or **Register Outputs** field and select **New Offsets** from the context menu to open the **New Offsets** dialog box.
2. Enter the required settings to renumber the offsets of the variables, see Table 34.

The dialog box is used for the automatic offset assignment of variables. The PADT allocates a new offset sorting depending on the order of the variables in the selected list.

Element	Description
Start Register	Start address of the first register.
Mode	<table><tr><th>Element</th><th>▪ Description</th></tr><tr><td>Compact (use all bytes)</td><td>▪ Packed BOOLs, on bit limits. ▪ Variables with a size larger than 1 byte, on whole bytes.</td></tr><tr><td>Register Alignment (except for bits)</td><td>▪ Packed BOOLs, on bit limits. ▪ Otherwise on integer register addresses. The PADT sets variables of size 2…8 bits to register bit 0.</td></tr><tr><td>Register Alignment (except for bits and up to 1 byte)</td><td>▪ Packed BOOLs, on bit limits. ▪ Variables with a size of 2…8 bits, on whole byte addresses. ▪ Variables larger than 1 byte, on integer register addresses.</td></tr></table>
Automatically close the dialog upon success.	Activated: Automatically close the dialog upon success. Deactivated: Click **OK** to close the dialog box.

Table 34: The New Offsets Dialog Box

5.6.3 Bit Variables

The variables that master 1 addresses bit by bit are entered in the **Bit Variables** tab (function code 1, 2, 5, 15, 100, 101, 104).

The input and output variables are dragged from the Object Panel onto the *Bit Inputs* or *Bit Outputs* fields.

- All the variables that the Modbus slave receives from the Modbus master are entered in the *Bit Inputs* field.
- All the variables that the Modbus slave sends to the Modbus master are entered in the *Bit Outputs* field.

To generate new offsets

1. Right-click the **Bit Inputs** or **Bit Outputs** field and select **New Offsets** from the context menu to open the **New Offsets** dialog box.
2. Enter the required settings to renumber the offsets of the variables, see Table 35.

The dialog box is used for the automatic offset assignment of variables. The PADT allocates a new offset sorting depending on the order of the variables in the selected list.

Element	Description
Start Bit	Start address of the first bit.
Mode	<table><tr><th>Element</th><th>▪ Description</th></tr><tr><td>Compact (use all bytes)</td><td>▪ Packed BOOLs, on bit limits. ▪ Variables with a size larger than 1 byte, on whole bytes.</td></tr><tr><td>Register Alignment (except for bits)</td><td>▪ Packed BOOLs, on bit limits. ▪ Otherwise on integer register addresses. The PADT sets variables of size 2…8 bits to register bit 0.</td></tr><tr><td>Register Alignment (except for bits and up to 1 byte)</td><td>▪ Packed BOOLs, on bit limits. ▪ Variables with a size of 2…8 bits, on whole byte addresses. ▪ Variables larger than 1 byte, on integer register addresses.</td></tr></table>
Automatically close the dialog upon success.	Activated: Automatically close the dialog upon success. Deactivated: Click **OK** to close the dialog box.

Table 35: The New Offsets Dialog Box

5.7 Modbus Slave Set System Variables

The **Modbus Slave Set System Variables** tab contains the following system variables.

Element	Description
Redundant State	This parameter describes the redundancy state of the redundant Modbus slave communication module pair. 0: The redundant Modbus slave COM modules are active. 1: The first Modbus slave COM module is inactive. 2: The redundant Modbus slave COM module inactive. 3: Both Modbus slave COM modules are inactive.

Table 36: The Modbus Slave Set Tab

5.7.1 Redundant Modbus Slave and Modbus Slave

The *Modbus Slave* tab contains the 2 tabs *Properties* and *System Variables*.

The *Modbus Slave Redundant* tab is available in the *Modbus Slave Set* dialog box if the *Activate Set Redundancy Operation* parameter was activated in the *Modbus Slave Set Properties* tab.

i Refer to the communication manual (HI 801 101 E) for a description of the pin assignment of the D-sub connectors (fb1, fb2).

5.7.1.1 Properties

Element	Description
Module	Selection of the COM module within which the protocol is processed.
Master Monitoring Time [ms]	Time monitoring of the Modbus master(s). If the master monitoring time is not 0, a check is run to determine whether a valid request concerning the Modbus slave data areas has been received from any Modbus master within this time span. If this is the case, the *Master Connection State* system variable is set to TRUE. If the connection of the communication module to the Modbus master is lost, the input variables are either initialized or are forwarded unchanged to the processor module, depending on the parameter *COM: Values at Connection Loss to Master*. Notice: If multiple master of the same type are used (Eth or RS485), the master monitoring function can not distinguish between masters. It may therefore occur, that a master fails without being noticed. See Chapter 5.6.1. Range of values: 1…(231^{-1}) [ms] Default value: 0 ms (no limitation)
Parameters for the Ethernet interface	
Enable the TCP/IP connection	Activated The TCP/IP connection is enabled. Deactivated The TCP/IP connection is disabled. Default value: deactivated
TCP Port	Default value: 502
Maximum Number of TCP connections	Maximum number of TCP connections opened simultaneously and operating as server. Range of values: 1…20 Default value: 20
UDP Enable	Activated The UDP/IP connection is enabled. Deactivated The UDP/IP connection is disabled. Default value: deactivated
UDP Port	Default value: 502

Parameters for the serial interface	
Name	Name of the serial interface
Interface	Selection of the fieldbus interfaces that are available and can be used for the Modbus slave (none, fb1 and/or fb2).
Slave Address	Slave bus address Range of values: 1…247
Baud Rate [bps]	Possible value for transfer rate for RS485: 300 Bit/s 600 Bit/s 1200 Bit/s 2400 Bit/s 4800 Bit/s 9600 Bit/s 19200 Bit/s 38400 bit/s (maximum baud rate for HIMax prior to V4). 57600 bit/s (maximum baud rate for HIMax V4 and higher). 62500 bit/s (HIMatrix/ HIQuad X). 76800 bit/s (HIMatrix/ HIQuad X). 115000 Bit/s (HIMatrix/ HIQuad X)
Parity	Range of values: ▪ None ▪ odd ▪ Even Default value: Even
Stop Bits	Range of values: Standard (adapts the number of stop bits to the parity: with parity = 1 stop bit, no parity = 2 stop bits). One stop bit Two stop bits Default value: Default
Number of Idle Chars	Number of idle characters at the start and the end of a RTU telegram frame. Range of values: 1…65535 Default value: 5 characters

Table 37: The TCP and UDP Ports Tab for the Modbus Slave

5.7.2 System Variables

The **System Variables** tab provides the system variables that allow the Modbus slave state to be evaluated in the user program and the Modbus slave to be controlled.

Element	Description
Average Buffer Fill Level for Requests	Average number of concurrent master requests
Valid Master Requests	Number of valid master requests since the last reset of all counters or last HIMax controller's start-up.
Master Requests	Total number of master requests since the last reset of all counters or last HIMax controller's start-up.
Master Monitoring Time [ms]	Refer to Chapter 5.7.1 for details on the time monitoring of the Modbus master(s).
Master Connection State	FALSE: Not connected TRUE: Connected
Maximum Buffer Fill Level for Requests	Maximum number of concurrent master requests.
Reset All Counters	This system variable is used to reset all counters in the user program. A change from 0 to 1 triggers the reset function. Values greater than 1 are treated as 1.
Invalid Master Requests	Number of invalid master requests since the last reset of all counters or last HIMax controller's start-up. An invalid request is a request upon which the Modbus slave sends an error code to the Modbus master. Incorrect transmissions that were already detected and filtered out at the driver level (framing errors, CRC errors, length errors) are not included here, but they are reported through the diagnostics.
Discarded Requests	Number of discarded master requests since the last reset of all counters or since start-up.
Response Timeout	Number of *response timeouts* since the last reset of all counters or since start-up. The *response timeout* is the maximum time within which the sending station must receive the message acknowledgment.

Table 38: The System Variables Tab for the Modbus Slave

5.8 Modbus Function Codes of the Modbus Slaves

5.8.1 Modbus Function Codes

The Modbus slave supports the following Modbus function codes:

Element	Code	Type	Description
Read Coils	01	BOOL	Read multiple variables (BOOL) from the slave's import or export[1)] area. Max. length of the process data: 250 bytes (2000 coils).
READ DISCRETE INPUT	02	BOOL	Read multiple variables (BOOL) from the slave's export area. Max. length of the process data: 250 bytes (2000 coils).
READ HOLDING REGISTER	03	WORD	Read multiple variables of any type from the slave's import or export[1)] area. Max. length of the process data: 250 bytes
READ INPUT REGISTER	04	WORD	Read multiple variables of any type from the slave's export area. Max. length of the process data: 250 bytes
Write Single Coil	05	BOOL	Write one single signal (BOOL) to the slave's import area. Max. length of the process data: 1 byte (1 coil).
Write Single Register	06	WORD	Write one single signal (WORD) to the slave's import area. Max. length of the process data: 2 bytes
Diagnosis	08	x	Only sub-code 0: Loop-back function of the slave.
Write Multiple Coils	15	BOOL	Write multiple variables (BOOL) to the slave's import area. Max. length of the process data: 246 bytes (1968 coils).
WRITE MULTIPLE REGISTER	16	WORD	Write multiple variables of any type to the slave's import area. Max. length of the process data: 246 bytes
READ WRITE MULTIPLE REGISTER	23	WORD	Write and read multiple variables of any type to and from the slave's import or export area. Max. length of the process data: 242 bytes (Modbus master's request telegram). 250 bytes (slave's response telegram).
Read Device Identification	43	x	Transmit the slave identification data to the master.
1) The export area must be set as required by the master.			

Table 39: Modbus Function Codes of the Modbus Slave

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

Error codes:

- If the master sends a telegram with unknown function codes, the controller responds with error code 1 (invalid code).
- If the master's telegram does not match the Modbus slave configuration (e.g., the request telegram does not end exactly at the variable limit), the slave returns error code 2 (invalid data).
- If the master sends a telegram with incorrect values (e.g., length fields), the slave responds with error code 3 (invalid value).

Communication only takes place if the COM module is in RUN. If the COM module is in any of the other operating states, no requests from the master is answered.

Note for Modbus Function: Read Device Identification (43)

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

Basic:
0x00 VendorName “HIMA Paul Hildebrandt GmbH”
0x01 ProductCode “<Module serial number>”
0x02 MajorMinorRevision “<COM Vx.y CRC>”

Regular:
0x03 VendorUrl “http://www.hima.com”
0x04 ProductName “HIMax”
0x05 ModelName “HIMax”
0x06 UserApplicationName “--------[S.R.S]”

Extended:
0x80 blank “--------”
0x81 blank “--------”
0x82 blank “--------”
0x83 blank “--------”
0x84 blank “--------”
0x85 blank “--------“
0x86 CRC of the file modbus.config “<0x234adcef>“
(configuration file for the Modbus slave protocol in the CPU file system. To compare with the information provided in SILworX, Online/Version Comparison).

The following ReadDevice ID codes are supported:
(1) Read Basic device identification (stream access)
(2) Read regular device identification (stream access)
(3) Read extended device identification (stream access)
(4) Read one specific identification object (individual access)

For further details on Modbus, refer to the *Modbus Application Protocol Specification* www.modbus.org)

5.9 HIMA-Specific Function Codes

HIMA specific function codes correspond to the standard Modbus function codes. The only differences are the maximum permissible process data length of 1100 bytes and the format of the request and response header:

Element	Code	Type	Description
Read Coils Extended	100 (0x64)	BOOL	Corresponds to function code 01. Read multiple variables (BOOL) from the slave's import or export[1)] area. Maximum length of the process data: 1100 bytes.
Read Discrete Inputs Extended	101 (0x65)	BOOL	Corresponds to function code 02 Read multiple variables (BOOL) from the slave's export area. Maximum length of the process data: 1100 bytes.
Read Holding Registers Extended	102 (0x66)	WORD	Corresponds to function code 03. Read multiple variables of any type from the slave's import or export[1)] area. Maximum length of the process data: 1100 bytes.
Read Input Registers Extended	103 (0x67)	WORD	Corresponds to function code 04. Read multiple variables of any type from the slave's export area. Maximum length of the process data: 1100 bytes.
Write Multiple Coils Extended	104 (0x68)	BOOL	Corresponds to function code 15. Write multiple variables (BOOL) to the slave's import area. Maximum length of the process data: 1100 bytes.
Write Multiple Registers Extended	105 (0x69)	WORD	Corresponds to function code 16. Write multiple variables of any type to the slave's import area. Maximum length of the process data: 1100 bytes.
Read/Write Multiple Registers Extended	106 (0x6A)	WORD	Corresponds to function code 23. Write and read multiple variables of any type to and from the slave's import or export area. Maximum length of the process data: 1100 bytes (Modbus master's request telegram). 1100 bytes (slave's response telegram).

Table 40: HIMA-Specific Function Codes

5.9.1 Format of Request and Response Header

The request and response header of the HIMA-specific Modbus function codes are structured as follows:

Code	Request	Response
100 (0x64)	1 byte function code 0x64 2 bytes start address 2 bytes number of coils 1…8800 (0x2260)	1 byte function code 0x64 2 bytes number of bytes = N N bytes coil data (8 coils are packed in one byte)
101 (0x65)	1 byte function code 0x65 2 bytes start address 2 bytes number of coils 1…8800 (0x226)	1 byte function code 0x65 2 bytes number of bytes = N N bytes coil data (8 coils are packed in one byte)
102 (0x66)	1 byte function code 0x66 2 bytes start address 2 bytes number of registers 1…550 (0x226)	1 byte function code 0x66 2 bytes number of bytes = N N bytes register data
103 (0x67)	1 byte function code 0x67 2 bytes start address 2 bytes number of registers 1…550 (0x226)	1 byte function code 0x67 2 bytes number of bytes = N N bytes register data
104 (0x68)	1 byte function code 0x68 2 bytes start address 2 bytes number of coils 1…8800 (0x2260) 2 bytes number of bytes = N N bytes coil data	1 byte function code 0x66 2 bytes start address 2 bytes number of coils 1…8800 (0x2260)
105 (0x69)	1 byte function code 0x69 2 bytes start address 2 bytes number of registers 1…550 (0x226) 2 bytes number of bytes = N N bytes register data	1 byte function code 0x69 2 bytes start address 2 bytes number of registers 1…550 (0x226)
106 (0x6A)	1 byte function code 0x6a 2 bytes read start address 2 bytes number of read registers 1...550 (0x226) 2 bytes write start address 2 bytes number of write registers 1…550 (0x226) 2 bytes number of write bytes = N N bytes register data	1 byte function code 0x6a 2 bytes number of bytes = N N bytes register data

Table 41: Request and Response Header

5.10 Addressing Modbus using Bit and Register

The addressing mode adheres to the Modbus addressing standard and only knows the two data lengths bit (1 bit) and register (16 bits) that are used to transfer all the data types allowed.

There are two areas within the Modbus slave: a **Register Area** (inputs and outputs) and a **Bit Area** (inputs and outputs). Both areas are separated from one another and can accept all permitted data types. The difference between these areas resides in the Modbus function codes permitted for accessing them.

i The Modbus addressing using bit and register does not guarantee the variable integrity, meaning that any arbitrary portion of a variable can be read or written with this access mode. Variables of type BOOL are stored in a packed format, i.e., each variable of type BOOL is stored as a bit within a byte.

5.10.1 Register Area

The variables in the **Register Area** are created in the **Register Variables** tab. For further details on how to assign send/receive variables, refer to Chapter 5.6.2.

i To access variables in the **Register Area** with the Modbus function codes 1, 2, 5, 15, the variables must be mirrored in the Bit Area, see Chapter 5.11.1.

The variables in the **Register Area** can only be accessed using the Modbus function codes 3, 4, 6, 16, 23. To do this, enter the start address of the first variable in the properties of the function code.

Example: Accessing the variables in the **Register Area** of the Modbus slave

Register Variables	Register.Bit	Bit
00_Register_Area_WORD	0.0	0
01_Register_Area_SINT	1.8	16
02_Register_Area_SINT	1.0	24
03_Register_Area_REAL	2.0	32
04_Register_Area_BOOL	4.8	64
05_Register_Area_BOOL	4.9	65
06_Register_Area_BOOL	4.10	66
07_Register_Area_BOOL	4.11	67
08_Register_Area_BOOL	4.12	68
09_Register_Area_BOOL	4.13	69
10_Register_Area_BOOL	4.14	70
11_Register_Area_BOOL	4.15	71
12_Register_Area_BOOL	4.0	72
13_Register_Area_BOOL	4.1	73
14_Register_Area_BOOL	4.2	74
15_Register_Area_BOOL	4.3	75
16_Register_Area_BOOL	4.4	76
17_Register_Area_BOOL	4.5	77
18_Register_Area_BOOL	4.6	78
19_Register_Area_BOOL	4.7	79

Table 42: Register Variables in the Register Area of the Modbus Slave

5.10.1.1 Modbus Master Configuration of the Request Telegram

To read the variables 01_Register_Area_SINT to 03_Register_Area_REAL in the Modbus master

1. Right-click **TCP/UDP Slave** and select **New** from the context menu.
2. Select **Read Holding Registers (3)** from the list.
3. Right-click **Read Holding Registers (3)** and select **Properties.**
   - **Enter** 1 **in t**he start address of the read area.
4. Right-click **Read Holding Registers (3)** and select **Edit.**
5. Drag the following variables from the **Object Panel** onto the **Input Variables** tab.

Register Variables	Offset
01_Register_Area_SINT	0
02_Register_Area_SINT	1
03_Register_Area_REAL	2

6. Right-click a free space in the **Output Variables** area to open the context menu and click **New Offsets** to renumber the variable offsets.

5.10.2 Bit Area

The variables in the **Bit Area** are created in the **Bit Variables** tab. For further details on how to assign send/receive variables, refer to Chapter 5.6.3.

i To access variables in the Register Area with the Modbus function codes 3, 4, 6, 16 and 23, the variables must be mirrored in the Bit Area, see Chapter 5.11.2.

The variables in the **Bit Area** can only be accessed using the Modbus function codes 1, 2, 5, 15. To do this, enter the start address of the first variable in the properties of the function code.

Example: Accessing the variables in the **Bit Area** of the Modbus slave

Bit Variables	Bit	Register.Bit
00_BIT_Area_WORD	0	0.0
01_BIT_Area_SINT	16	1.8
02_BIT_Area_SINT	24	1.0
03_BIT_Area_REAL	32	2.0
04_BIT_Area_BOOL	64	4.8
05_BIT_Area_BOOL	65	4.9
06_BIT_Area_BOOL	66	4.10
07_BIT_Area_BOOL	67	4.11
08_BIT_Area_BOOL	68	4.12
09_BIT_Area_BOOL	69	4.13
10_BIT_Area_BOOL	70	4.14
11_BIT_Area_BOOL	71	4.15
12_BIT_Area_BOOL	72	4.0
13_BIT_Area_BOOL	73	4.1
14_BIT_Area_BOOL	74	4.2
15_BIT_Area_BOOL	75	4.3
16_BIT_Area_BOOL	76	4.4
17_BIT_Area_BOOL	77	4.5
18_BIT_Area_BOOL	78	4.6
19_BIT_Area_BOOL	79	4.7

Table 43: Bit Variables in the Bit Area of the Modbus Slave

5.10.2.1 Modbus Master Configuration of the Request Telegram

To read the variables 04_BIT_Area_BOOL to 06_Area_BOOL in the Modbus master

1. Right-click **TCP/UDP Slave** and select **New** from the context menu.
2. Select **Read Coils (1)** from the list.
3. Right-click **Read Coils (1)** and select **Edit** from the context menu.
   - **Enter** 64 **in t**he start address of the read area.
4. Right-click **Read Coils (1)** and select **Edit** from the context menu.
5. Drag the following variables from the **Object Panel** onto the **Input Variables** tab.

Bit Variables	Offset
04_BIT_Area_BOOL	0
05_BIT_Area_BOOL	1
06_BIT_Area_BOOL	2

6. Right-click a free space in the **Output Variables** area to open the context menu and click **New Offsets** to renumber the variable offsets.

5.11 Offsets for Alternative Modbus Addressing

To access the variables in the **Bit Area** using the Modbus function codes (of type Register), the variables must be mirrored in the Register Area, and to access the variables in the **Register Area** using the Modbus function codes (of type Bit), the variables must be mirrored in the **Bit Area**.

The offsets for the mirrored variables are entered in the **Modbus Slave Set Properties** tab.

To mirror the variables in the bit area and register area

1. In the **Modbus Slave Set Properties** tab of the Modbus slave activate **Use Alternative Register/Bit Addressing**.
   ☑ This action mirrors the variables in the corresponding area.
2. Enter the offset for the mirrored variables in the bit area and register area.

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

Element	Description / Range of values
Alternative register / bit addressing	Activated — Use the alternative addressing. Deactivated — Do not use the alternative addressing Default value: deactivated
Register Area Offset Bits (Input)	0…65535
Register Area Offset Bits (Output)	0…65535
Bit Area Offset Register (Input)	0…65535
Bit Area Offset Bits (Output)	0…65535

Table 44: The Modbus Slave Set Properties Tab

5.11.1 Access to the Register Variables in the Bit Area of the Modbus Slave

To access the **Register Variables** with the Modbus function codes (of type Bit) 1, 2, 5, 15, the register variables must be mirrored in the **Bit Area**.
The offsets for the mirrored **register variables** must be entered in the **Properties/Offsets** tab.

Example:

Bit area offset /register Inputs 8000
Bit area offset / register outputs 8000
The variables mirrored from the Register Area to the **Bit Area** are located here starting with Bit Address 8000.

Mirrored register variables	Bit
00_Register_Area_WORD	8000
01_Register_Area_SINT	8016
02_Register_Area_SINT	8024
03_Register_Area_REAL	8032
04_Register_Area_BOOL	8064
05_Register_Area_BOOL	8065
06_Register_Area_BOOL	8066
07_Register_Area_BOOL	8067
08_Register_Area_BOOL	8068
09_Register_Area_BOOL	8069
10_Register_Area_BOOL	8070
11_Register_Area_BOOL	8071
12_Register_Area_BOOL	8072
13_Register_Area_BOOL	8073
14_Register_Area_BOOL	8074
15_Register_Area_BOOL	8075
16_Register_Area_BOOL	8076
17_Register_Area_BOOL	8077
18_Register_Area_BOOL	8078
19_Register_Area_BOOL	8079

Table 45: Variables Mirrored from the Register Area to the Bit Area

5.11.1.1 Modbus Master Configuration of the Request Telegram

To read the variables 04_Register_Area_BOOL to 06_Register_Area_BOOL in the Modbus master

1. Right-click **TCP/UDP Slave** and select **New** from the context menu.
2. Select **Read Coils (1)** from the list.
3. Right-click **Read Coils (1)** and select **Edit** from the context menu.
   - **Enter** 8064 **in t**he start address of the read area.
4. Right-click **Read Coils (1)** and select **Edit** from the context menu.
5. Drag the following variables from the **Object Panel** onto the **Input Variables** tab.

Mirrored Register Variables	Offset
04_Register_Area_BOOL	0
05_Register_Area_BOOL	1
06_Register_Area_BOOL	2

6. Right-click a free space in the **Output Variables** area to open the context menu and click **New Offsets** to renumber the variable offsets.

5.11.2 Access to the Bit Variables in the Register Area of the Modbus Slave

To access the **Bit Variables** with the Modbus function codes (of type Register) 3, 4, 6, 16, 23, the bit variables must be mirrored in the **Register Area**.
The offsets for the mirrored bit variables must be entered in the **Properties/Offsets** tab.

Example:

Register area offset / bit inputs: 1000
Register area offset / bit outputs: 1000
The variables mirrored from the bit area to the register area are located here starting with register address 1000.

Mirrored bit variables	Register.Bit
00_BIT_Area_WORD	1000.0
01_BIT_Area_SINT	1001.8
02_BIT_Area_SINT	1001.0
03_BIT_Area_REAL	1002.0
04_BIT_Area_BOOL	1004.8
05_BIT_Area_BOOL	1004.9
06_BIT_Area_BOOL	1004.10
07_BIT_Area_BOOL	1004.11
08_BIT_Area_BOOL	1004.12
09_BIT_Area_BOOL	1004.13
10_BIT_Area_BOOL	1004.14
11_BIT_Area_BOOL	1004.15
12_BIT_Area_BOOL	1004.0
13_BIT_Area_BOOL	1004.1
14_BIT_Area_BOOL	1004.2
15_BIT_Area_BOOL	1004.3
16_BIT_Area_BOOL	1004.4
17_BIT_Area_BOOL	1004.5
18_BIT_Area_BOOL	1004.6
19_BIT_Area_BOOL	1004.7

Table 46: Variables Mirrored from the Bit Area to the Register Area

5.11.2.1 Modbus Master Configuration of the Request Telegram

To read the variables 01_BIT_Area_SINT to 03_BIT_Area_REAL in the Modbus master

1. Right-click **TCP/UDP Slave** and select **New** from the context menu.
2. Select **Read Holding Registers (3)** from the list.
3. Right-click **Read Holding Registers (3)** and select **Properties.**
   - **Enter 1001 in the** start address of the read area.
4. Right-click **Read Holding Registers (3)** and select **Edit.**
5. Drag the following variables from the **Object Panel** onto the **Input Variables** tab.

Mirrored Bit Variables	Offset
01_BIT_Area_SINT	0
02_BIT_Area_SINT	1
03_BIT_Area_REAL	2

6. Right-click a free space in the **Output Variables** area to open the context menu and click **New Offsets** to renumber the variable offsets.

5.12 Control Panel (Modbus Slave)

The Control Panel can be used to verify and control the settings for the Modbus slave. Details on the slave's current status (e.g., master state, etc.) are displayed.

To open the Control Panel for monitoring the Modbus slave

1. Right-click the **Hardware** structure tree element and select **Online** from the context menu.
2. In the **System Login** window, enter the access data to open the online view for the hardware.
3. Double-click **COM Module** and select **Modbus Slave** in the structure tree.

5.12.1 Context Menu (Modbus Slave)

The following command is available from the context menu of the selected Modbus slave:

Reset Statistics

This command is used to reset the statistical data (e.g., min. or max. cycle time) to zero.

5.12.2 View Box (Modbus Slave)

The view box displays the following values of the selected Modbus slave.

Element	Description
Name	Modbus slave name.
Planned µP Budget	See Chapter 5.6.
Current µP Budget	
SRS of Redundant Module	SRS of the redundant COM module.
Response Time [ms]	Time period after the reception of a request within which the Modbus slave may respond.

Table 47: The Modbus Slave View Box

5.12.3 View Box (Master Data)

The Master Data view box displays the following values.

Element	Description
Name	Name of master data.
Requests	Total number of master requests since the last counter reset.
Valid Requests	Number of valid master requests since the last counter reset.
Invalid Requests	Number of invalid master requests since the last counter reset. The invalid requests only include requests that were acknowledged by the master. Requests with CRC error that were improperly received are automatically rejected.
Master Timeout [ms]	Period within which the slave must receive at least one request from the master. If the slave does not receive any request within the this period, *Master Connection Status* is set to *Not connected*.
Connection State	0 = Not monitored. 1 = Connected. 2 = Connected
Response Timeout	Number of *response timeouts* since the last reset of all counters or since start-up. The *response timeout* is the maximum time within which the sending station must receive the message acknowledgment.

Element	Description
Discarded Requests	Number of discarded master requests since the last reset of all counters or since start-up.
Maximum Buffer Fill Level for Requests	Maximum number of concurrent master requests.
Average Buffer Fill Level for Requests	Average number of concurrent master requests

Table 48: The Master Data View Box

5.13 FBx LED Function in the Modbus Slave

The FBx LED of the corresponding fieldbus interface indicates the state of the Modbus protocol. The states of the FBx LED are specified in the following table:

LED	Color	Status	Description
FBx	Yellow	Blinking	The Modbus slave protocol is active and is exchanging data with the Modbus master.
		Off	The Modbus slave protocol is not active! This means that the controller is in STOP or no Modbus master is configured.

Table 49: The FBx LED

5.14 Function of the FAULT LED in the Modbus Slave (HIMax only)

The FAULT LED of the corresponding fieldbus interface indicates a failure of the Modbus protocol. The states of the FAULT LED are specified in the following table:

LED	Color	Status	Description
Fault	Red	Blinking1[1)]	The Modbus slave protocol is disturbed. The following events result in a malfunction: ▪ Unknown function code received. ▪ Request with incorrect addressing received. ▪ The protocol load attains the maximum of the configured computing time budget. The protocol processing is constrained by the load limitation. This can lead to considerable impairments of the protocol function. Either the protocol settings or the settings of the computing time budget must be adjusted. If no faults occur for a period longer than 5 seconds, the state changes to *Protocol not disturbed*.
		Off	PROFIBUS DP slave protocol is not disturbed.
[1)] Blinking frequency: Long (600 ms) on, long (600 ms) off.			

Table 50: The FAULT LED

6 General

This chapter describes parameters that are relevant for all communication protocols.

6.1 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 CPU still executes the safety-relevant monitoring for all 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.

6.1.1 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 COM module: 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 HIMA 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.

6.2 Load Limitation

A computing time budget expressed in % *(µP budget)* can be defined for each communication protocol. It allows the available computing time to be distributed among the configured protocols. The sum of the computing time budgets configured for all communication protocols on a CPU or COM module may not exceed 100%.

The defined computing time budgets of the individual communication protocols are monitored. If a communication protocol has already achieved or exceeded its budget and no reserve computing time is available, the communication protocol cannot be processed.

If sufficient additional computing time is available, it is used to process the communication protocol that has already achieved or exceeded its budget. It can therefore happen that a communication protocol uses more computing time budget than has been allocated to it.

It is possible that more than 100% computing time budget is displayed online. This is not a fault; the computing time budget exceeding 100% indicates the additional computing time used.

i The additional computing time budget is not a guarantee for a certain communication protocol and can be revoked from the system at any time.

Appendix

Glossary

Term	Description
ARP	Address resolution protocol, network protocol for assigning the network addresses to hardware addresses
Bit variable	Variable that is addressed bit by bit.
CENELEC	Comité Européen de Normalisation Électrotechnique (European Commitee for Electrotechnical Standardization)
COM	Communication module
Connector board	Connector board for the HIMax module.
CPU	Processor module
CRC	Cyclic redundancy check
Data view	The global variables for output and output data are assigned to a data view to allow access to Modbus sources.
EN	European standard
Export area	The export area is the process data volume that is written to by the system (a user program, hardware input or another protocol) and is read by the Modbus master.
FB	Fieldbus
FBD	Function block diagrams
ICMP	Internet control message protocol, network protocol for status or error messages.
IEC	International electrotechnical commission.
Import area	Process data volume that is written to by the Modbus master and can be used as input data for the system (in a user program, hardware output or another protocol).
Interference-free	Supposing that two input circuits are connected to the same source (e.g., a transmitter). An input circuit is termed "interference-free" if it does not distort the signals of the other input circuit.
KE	Communication end point
MAC address	Media access control address, hardware address of one network connection.
NSIP	Not safety-related protocol.
PADT	Programming and debugging tool (acc. to IEC 61131-3), PC with SILworX.
PE	Protective ground.
PELV	Protective extra low voltage.
PES	Programmable electronic system.
R	Read.
R/W	Read/Write.
Rack ID	Base rack identification (number).
Register variable	Variable that is addressed word by word.
SB	System bus.
SFF	Safe failure fraction, i.e., portion of faults that can be safely controlled.
SIF	Safety-instrumented function.
SIL	Safety integrity level (in accordance with IEC 61508).
SILworX	Programming tool for HIMax, HIQuad X und HIMatrix.
SIP	Safety-instrumented protocol.
SNTP	Simple network time protocol (RFC 1769).
SRS	System.Rack.Slot
SW	Software
TMO	Timeout
W	Write
WD	Watchdog
WDT	Watchdog time

Index of Figures

Figure 1: Communication via Modbus TCP/IP 11
Figure 2: Modbus Network 24
Figure 3: Modbus Gateway 27
Figure 4: Serial Modbus 30
Figure 5: Modbus Telegram 30

Index of Tables

Table 1: Additional Applicable Manuals 5
Table 2: Equipment and System Requirements for the Modbus Master 10
Table 3: Properties of the Modbus Master 11
Table 4: System Variables for the Modbus Master 16
Table 5: General Properties of the Modbus Master 17
Table 6: COM/CPU Parameters 18
Table 7: Modbus Function Codes 18
Table 8: HIMA-Specific Function Codes 19
Table 9: Request and Response Header 20
Table 10: Request Telegram Read Coils 21
Table 11: Request Telegram Read Discrete Inputs 21
Table 12: Request Telegram Read Holding Registers 21
Table 13: Request Telegram Read Input Registers 21
Table 14: Read Write Holding Register 22
Table15: The Write Multiple Coils Request Telegram 22
Table 16: The Write Multiple Registers Request Telegram 23
Table 17: Write Single Coil (05) Request Telegram 23
Table 18: Write Single Register Request Telegram 23
Table 19: System Variables for TCP/UDP Slaves 25
Table 20: Configuration Parameters 26
Table 21: Connection Parameters for the Modbus Gateway 29
Table 22: Status Variables for the Gateway Slave 29
Table 23: Connection Parameters for the Gateway Slave 29
Table 24: Parameters for the Serial Modbus Master 31
Table 25: System Variables in the Modbus Slave 31
Table 26: Connection Parameters for the Modbus Master 32
Table 27: View Box of the Modbus Master 33
Table 28: The FBx LED 34
Table 29: The FAULT LED 34
Table 30: Equipment and System Requirements for the Modbus Slave 35
Table 31: Properties of the Modbus Slave 36

Table 32:	Slots Allowed for the Redundant Modbus Slave HIMax COM Modules	37
Table 33:	The Modbus Slave Set Properties Tab	39
Table 34:	The New Offsets Dialog Box	40
Table 35:	The New Offsets Dialog Box	41
Table 36:	The Modbus Slave Set Tab	41
Table 37:	The TCP and UDP Ports Tab for the Modbus Slave	43
Table 38:	The System Variables Tab for the Modbus Slave	44
Table 39:	Modbus Function Codes of the Modbus Slave	45
Table 40:	HIMA-Specific Function Codes	47
Table 41:	Request and Response Header	48
Table 42:	Register Variables in the Register Area of the Modbus Slave	49
Table 43:	Bit Variables in the Bit Area of the Modbus Slave	50
Table 44:	The Modbus Slave Set Properties Tab	51
Table 45:	Variables Mirrored from the Register Area to the Bit Area	52
Table 46:	Variables Mirrored from the Bit Area to the Register Area	53
Table 47:	The Modbus Slave View Box	54
Table 48:	The Master Data View Box	55
Table 49:	The FBx LED	55
Table 50:	The FAULT LED	55

MANUAL
Modbus

HI 801 522 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 HIMA solutions online:

www.hima.com/en/

HIMA SMART SAFETY.

www.hima.com