MANUAL

RESI®
...feel the spirit of a new generation

RESI-KNX-MBMASTER

Great care has been taken in the creation of the text, illustrations and program examples in this manual. The editors and publishers accept no responsibility for any inadvertent omission of entries or for typographical or other errors herein. Nor can they be held responsible or liable for consequences arising from any errors herein.
This manual is subject to copyright law. All rights are reserved.
This manual may not be copied in part or whole in any form including electronic media without the written consent of RESI. Neither may it be transferred in any other language suitable for machines or data processing facilities. Also rights for reproduction through lecture, radio or television transmission are reserved.

This documentation and the accompanying software are copyrighted by RESI.

RESI Informatik & Automation GmbH	Date:	**21.11.2018**	Customer:		Pages
	Version:	**01.11**	Title:	**RESI-KNX-MBMASTER Manual**	**34**
	Edited by:	**DI HC Sigl**			
	Reviewed by:	**DI HC Sigl**	Project:		
	Reviewed by:	**-**			

1 History

Date	Editor	Description
10.09.18	DI HC Sigl	Initial release
09.10.18	DI HC Sigl	Revision of configuration size
21.11.18	DI HC Sigl	Adding different baud rates to DIP switch ON, ON

2 Content

3 IMPORTANT SECURITY NOTES

Danger to life through electrical current!
Only skilled personal trained in electro-engineering should perform the described steps in the following chapters. Please observe the country specific rules and standards. Do not perform any electrical work while the device is connected to power.

Pay attention to the following rules:

1. Disconnect the system from power
2. Secure the system against automatic power on
3. Check that the system is de-energized
4. Cover other energized parts of the system

IMPORTANT HINT: Before you start with the installation and the initial setup of the device, you have to read this document and the attached installation guide and the actual manual for the device very carefully. You have to follow all the herein given information very accurate!

- □ Only authorized and qualified personnel are allowed to install and setup the device!
- □ The connection of the device must be done in de-energized state!
- □ Do not perform any electrical work while the device is connected to power!
- □ Disable and secure the system against any automatic restart or power on procedure!
- □ The device must be operated with the defined voltage level!
- □ Supply voltage jitters must not exceed the technical specifications and tolerances given in the technical manuals for the product. If you do not obey this issue, the proper performance of the device cannot be guaranteed. This can lead to fail functions of the device and in worst case to a complete breakdown of the device!
- □ You have to obey the current EMC regulations for wiring!
- □ All signal, control and supply voltage cables must be wired in a way, that no inductive or capacitive interference or any other severe electrical noise disturbance may interfere with the device. Wrong wiring can lead to a malfunction of the device!
- □ For signal or sensor cables you have to use shielded cables, to avoid damages through induction!
- □ You have to obey and to apply the current safety regulations given by the ÖVE, VDE, the countries, their control authorities, the TÜV or the local energy supply company!
- □ Obey country-specific laws and standards!
- □ The device must be used for the intended purpose of the manufacturer!
- □ No warranties or liabilities will be accepted for defects and damages resulting from improper or incorrect usage of the device!
- □ Subsequent damages, which results from faults of this device, are excluded from warranty and liability!
- □ Only the technical data, wiring diagrams and operation instructions, which are part to the product shipment are valid!
- □ The information on our homepage, in our datasheets, in our manuals, in our catalogues or published by our partners can deviate from the product documentation and is not necessarily always actual, due to constant improvement of our products for technical progress!
- □ In case of modification of our devices made by the user, all warranty and liability claims are lost!
- □ The installation has to fulfil the technical conditions and specifications (e.g. operating temperatures, power supply, …) given in the devices documentation!
- □ Operating our device close to equipment, which do not comply with EMC directives, can influence the functionality of our device, leading to malfunction or in worst case to a breakdown of our device!

- ❑ Our devices must not be used for monitoring applications, which solely serve the purpose of protecting persons against hazards or injury, or as an emergency stop switch for systems or machinery, or for any other similar safety-relevant purposes!
- ❑ Dimensions of the enclosures or enclosures accessories may show slight tolerances on the specifications provided in these instructions!
- ❑ Modifications of this documentation is not allowed!
- ❑ In case of a complaint, only complete devices returned in original packing will be accepted!

4 General Information

Our RESI-KNX-MBMASTER gateway connects a KNX bus system to MODBUS/RTU slave devices, which have a serial RS232 or RS485 interface. The gateway offers an integrated KNX bus coupler with 2-wire connection. The time-critical KNX communication is handled directly in the gateway. On the MODBUS/RTU side, our module is a MODBUS/RTU master. This protocol is also processed completely in the gateway. Thus, our gateway works completely autonomous after the configuration.

The mapping between the MODBUS/RTU registers of the connected MODBUS/RTU slave devices and the KNX group addresses is created with our free MODBUSConfigurator software and loaded into the module. If the MODBUS/RTU master function is activated, the module starts polling the connected MODBUS/RTU slaves cyclically. The read-in values are then passed on to the KNX bus. If a KNX telegram is received, the incoming data is converted and transmitted to the connected MODBUS/RTU slave devices using appropriate MODBUS / RTU master write protocols.

- Simple integration of a KNX system into any plant
- MODBUS/RTU master protocol
- KNX and MODBUS/RTU master interface are galvanically isolated
- Supports all 32768 KNX group addresses
- Max. 128 configuration entries for KNX groups
- Max. 128 configuration entries for MODBUS/RTU master protocols
- Supports all DPT data types
- Integrated KNX bus coupler
- MODBUS/RTU master interface: RS232 or RS485, 300 to 57600 bps, 8 data bits, no or even parity, 1 stop bit
- Supply with 12-48V DC voltage
- Power consumption <0.5W
- Mounting on a DIN EN50022 rail

Type	Description	Voltage	Power	Weight
RESI-KNX-MBMASTER	KNX on MODBUS / RTU Master Gateway with RS232 and RS485 interface for all 32768 KNX groups and max. 128 configuration entries for KNX groups or max. 128 configuration entries for MODBUS/RTU master protocols	12…48 V=	<0.5W	55 g

Technical data			
Power supply			
Supply voltage	12…48 V= +/-10%	Storage temperature	-20...85 °C
Power LED indicator	Yes	Operation temperature	0...60°C
Power consumption	<0.5W	Humidity	25...90 % rH not condensing
		Protection class	IP20 (EN 60529)
		Dimensions LxWxH	17,5mm x 90mm x 58mm
		Weight	55g
		Mounting	on DIN EN50022 rail
MODBUS/RTU-KNX mapping			
Maximum entries	150		
MODBUS/RTU protocol			
Protocol	MODBUS/RTU master		
Type	RS232 or RS485		
Baud rate	300 to 57600/8/N or E/1		
Cable connection	Via clamps		
LED indicator	Yes		
Galvanic insulation to the KNX interface	Yes		
ASCII protocol interface			
Protocol	ASCII plain text		
Type	RS232 or RS485		
Baud rate	300 to 57600/8/N or E/1		
Cable connection	Via clamps		
LED indicator	Yes		
Galvanic insulation to the KNX interface	Yes		
KNX bus interface			
Protocol	KNX		
Baud rate	9600Bits/s		
Cable connection	Via clamps		
Galvanic insulation to serial interface	Yes		
LED indicator	Yes		
Clamps			
Clamp wire cross section	Max. 1,5 mm²		
Tightening torque	Max. 0.5Nm	**CE conformity**	Yes

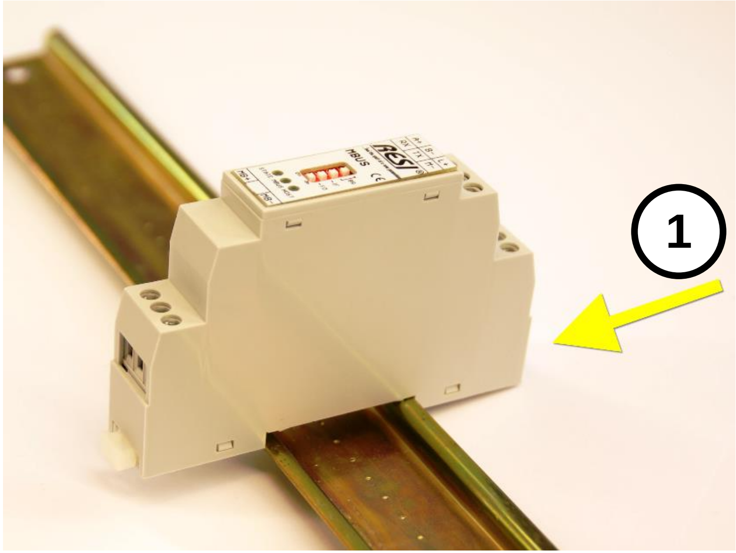
IT Accessories	
MODBUSConfigurator	Use our free software to configure and test the mapping between KNX and MODBUS register.

5 Mounting and Connections

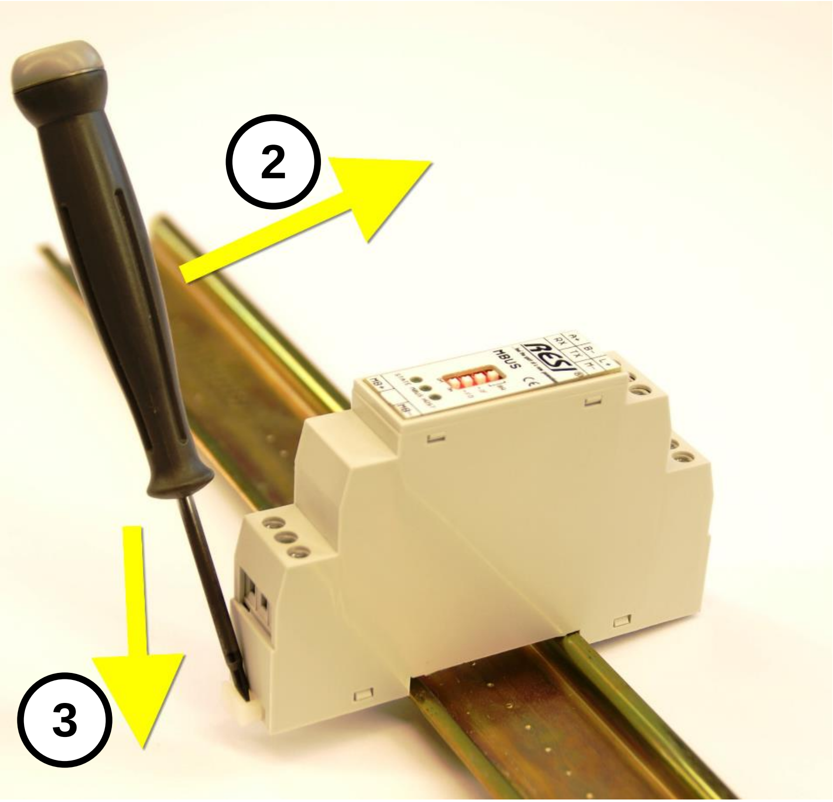
5.1 Assembling

Our RESI-KNX-MBMASTER gateway is designed for mounting on a 35mm DIN-EN50022 rail. Please note, that there are symbol photos used in the mounting pictures below.

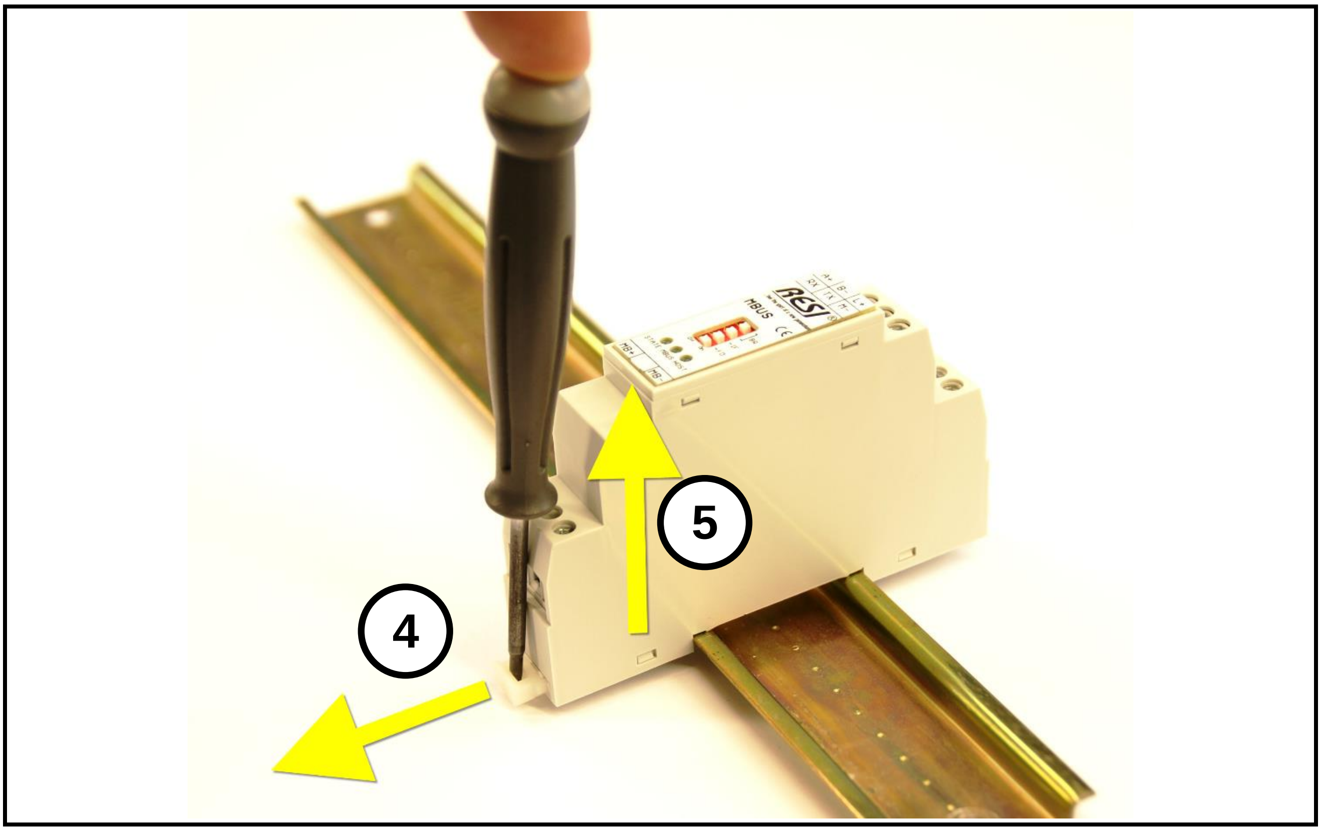
At first, put the gateway with the top side on the DIN rail (1).



Then open the clamp lever on the bottom side with a screw driver (2) and press the device on the DIN rail (3). Release the clamp lever. The module is now placed correctly on the DIN rail.



To dismount the module from the DIN rail first open the clamp lever with a screwdriver on the bottom side (4). Hold the clamp lever opened while you lift the module from the DIN rail (5). Then remove the gateway from the bar with while pulling it on the top side.



5.2 Clamps and LEDs

	RESI-KNX-MBMASTER
L+ M-	Power supply L+: 12…48 V= M-: Ground
A B	RS485 Modbus/RTU master or ASCII text interface A: DATA+ B: DATA-
RX TX M-	RS232 Modbus/RTU master or ASCII text interface RX: serial receive TX: serial transmit M-: Ground for RS232
K+ K-	Interface to KNX bus system K+: KNX+ bus wire (red) K-: KNX- bus wire (black)
STATE	State-LED, flashes slowly, when gateway is ok and the KNX is connected. Flashes fast, if the gateway or the KNX connection has an error
KNX	KNX activity LED, this LED is on while the gateway send or receives KNX telegrams
HOST	HOST-LED, flashes, when host sends/receive telegrams

Table: Description of connectors and LEDs of the RESI-KNX-MBMASTER gateway

5.3 DIP switch settings

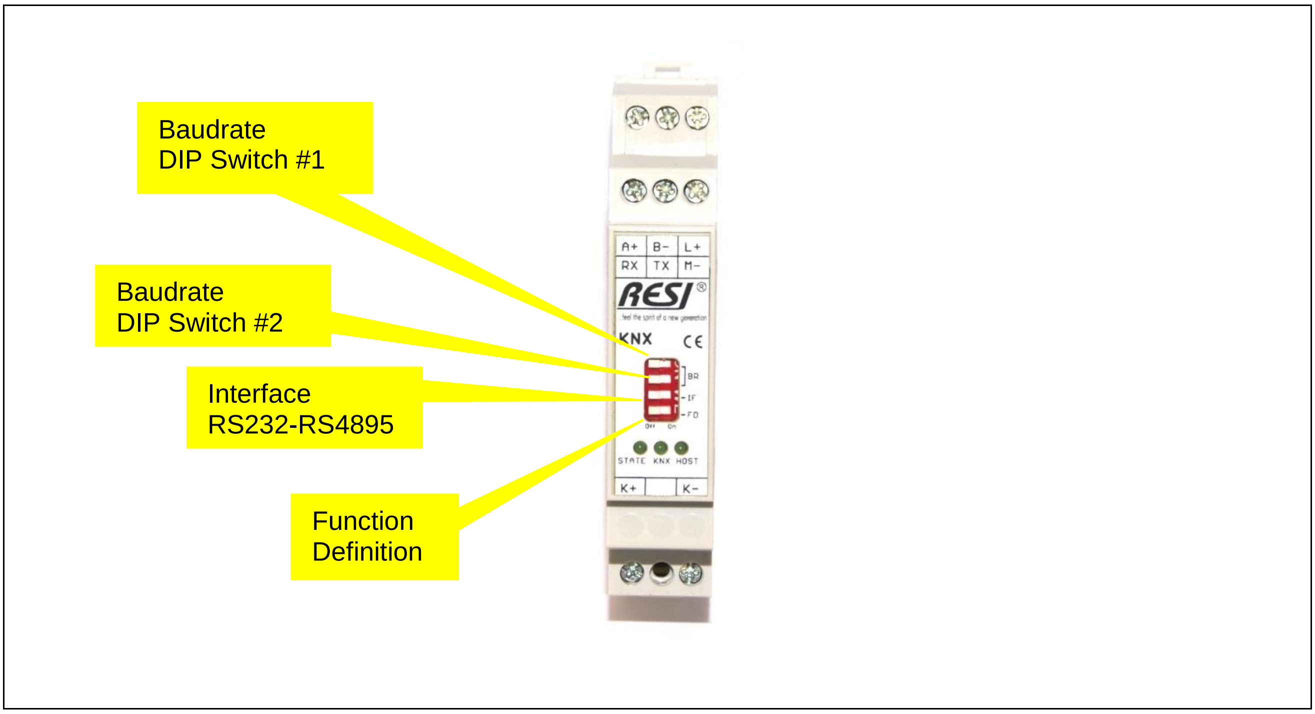


Illustration: Description of the DIP switch settings and LED status displays

DIP Switch	RESI-KNX-MBMASTER
Baudrate BR	Use DIP Switches 1+2 to select baud rate: OFF OFF: 9600Bd ON OFF: 19200Bd OFF ON: 38400Bd ON ON: Baudrate from FLASH, usually 57600Bd HINT: The correct parity (NONE, EVEN or ODD) is configured with the PC software, not via DIP switches!

Interface
IF
	Select serial interface for the host communcation
	OFF=RS232
	ON=RS485

Function Definition FD	Selects a special function: OFF=CONFIGURATION MODE: no MODBUS/RTU master functions are executed, module can be configured with PC software ON = MBRTU MASTER MODE: The module works as MODBUS/RTU master, no PC configuration possible

Table: Description of DIP switch functions

5.4 Wiring diagram

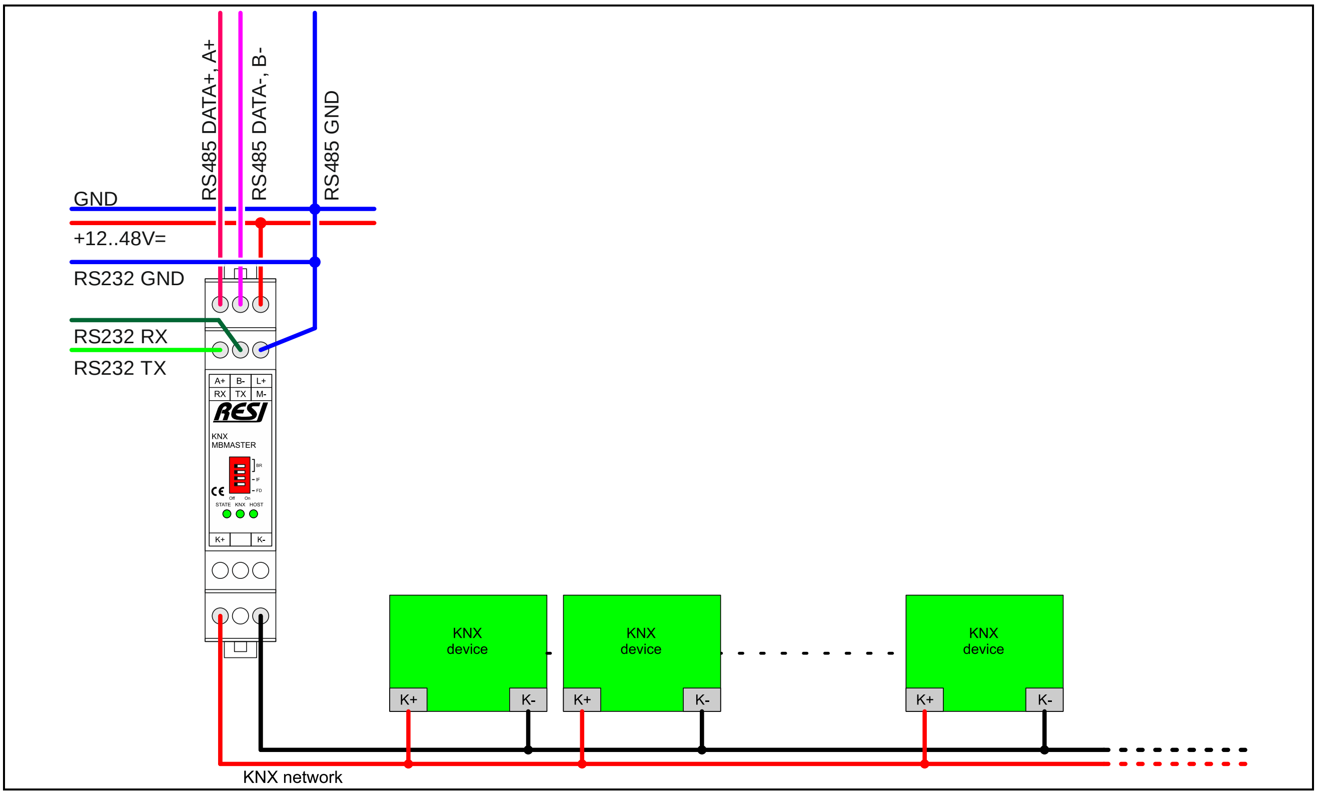


Illustration: wiring diagram of gateway on KNX

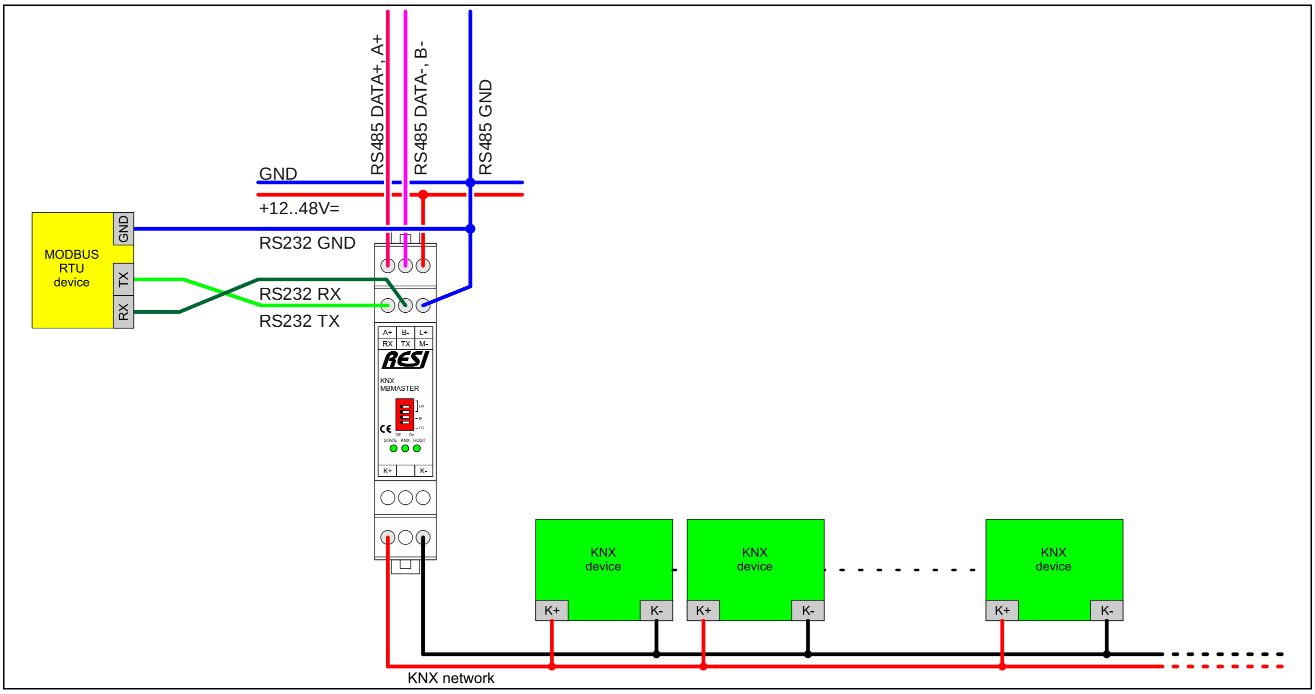


Illustration: wiring diagram of gateway with MODBUS/RTU slave on RS232

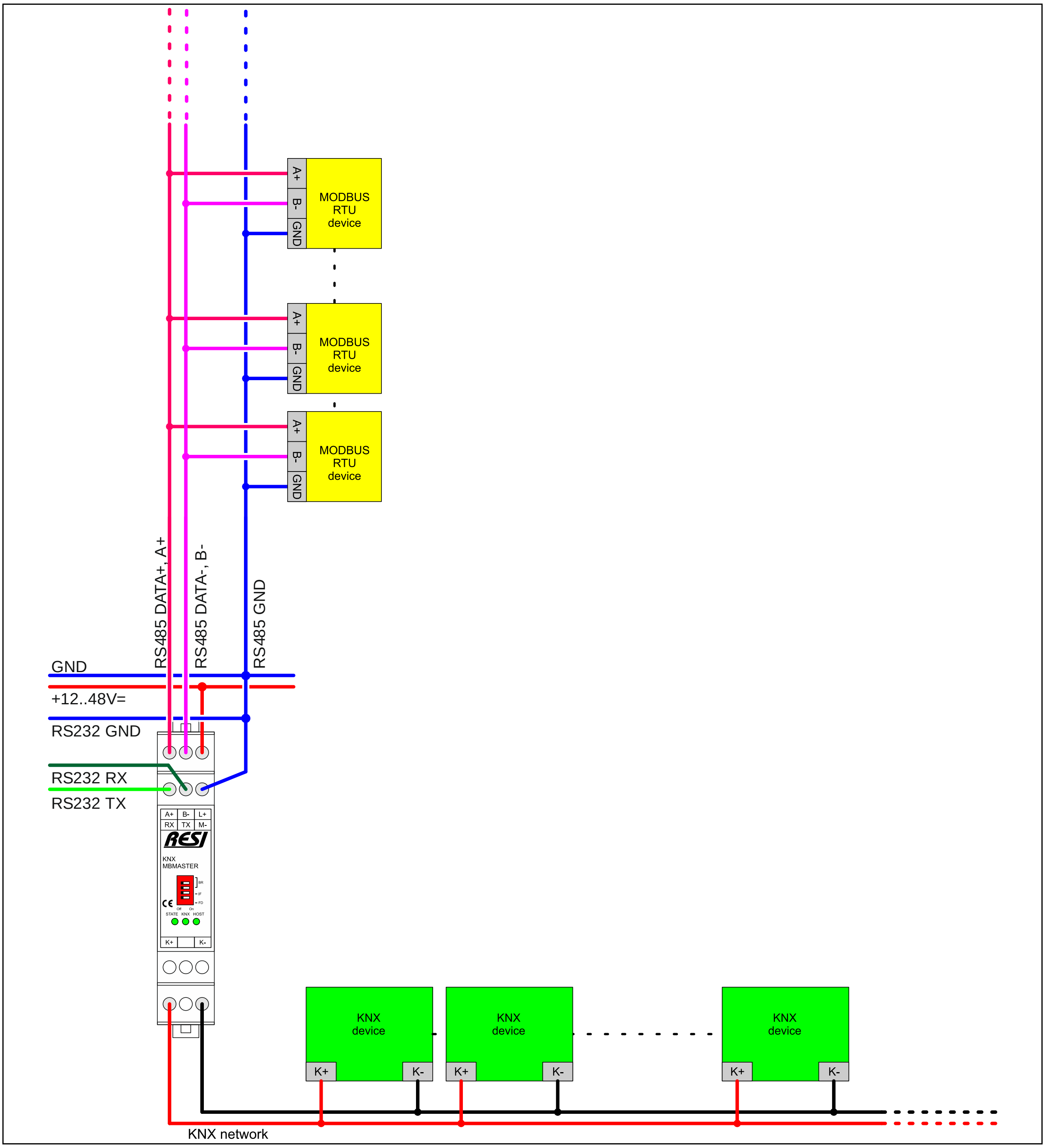


Illustration: wiring diagram of gateway with MODBUS/RTU slaves on RS485

6 Configuration with RESI MODBUSConfigurator software

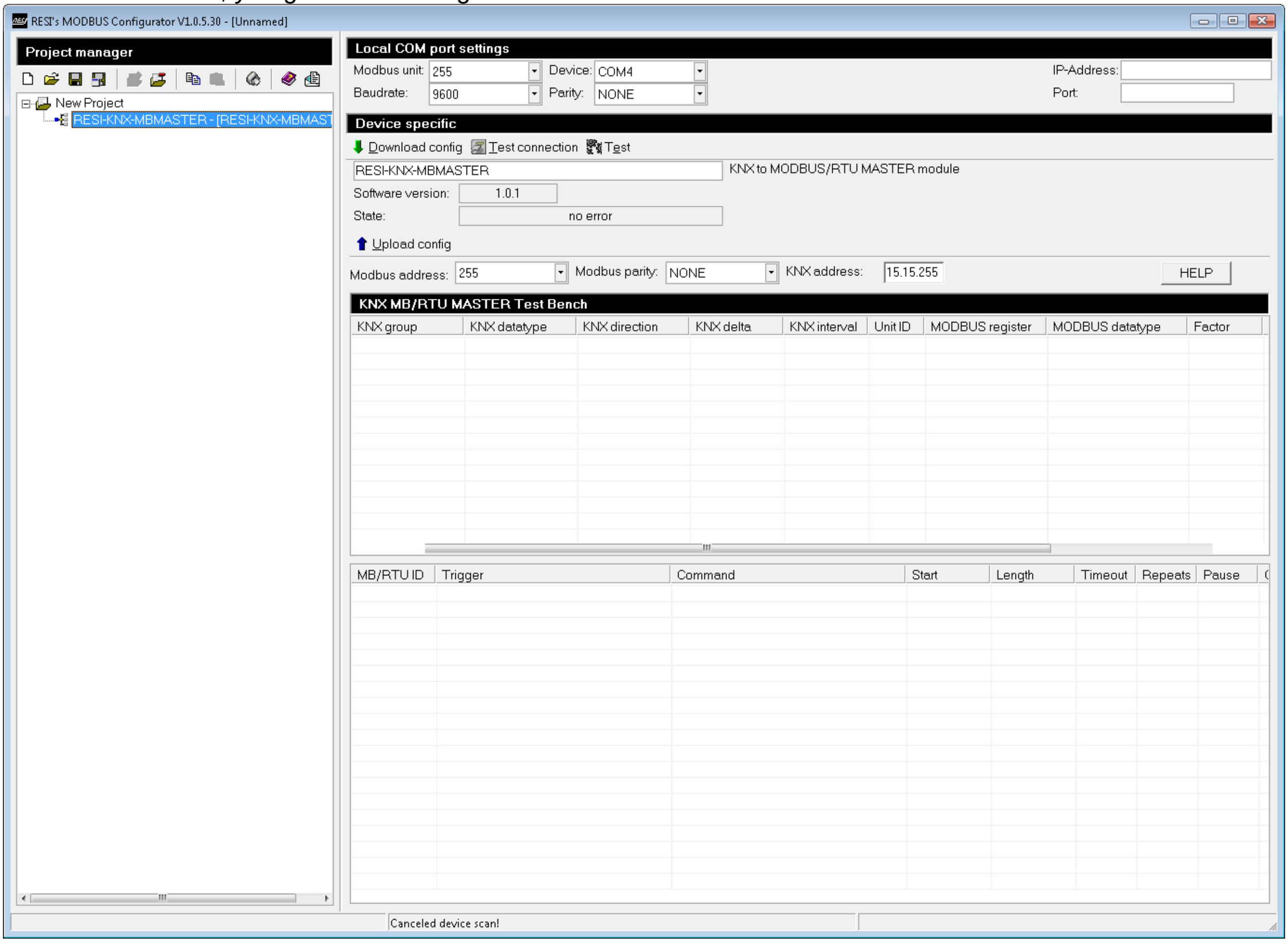
Download our free software from our homepage www.RESI.cc and install it on your computer.

6.1 Establish a connection

Establish a connection between the module and our software tool RESI MODBUSConfigurator.

ATTENTION: The Gateway can only communicate with the PC Software if the MODBUS/RTU Master functionality is disabled. This is achieved by setting the DIP Switch FD to OFF! The gateway will then stop polling the connected MODBUS/RTU slave devices immediately and is ready to communicate with the PC software!

If this is successful, you get the following screen:



RESI's MODBUS Configurator V1.0.5.30 - [Unnamed]

Project manager

New Project
RESI-KNX-MBMASTER - [RESI-KNX-MBMAST

Local COM port settings

Modbus unit: 255
Device: COM4
Baudrate: 9600
Parity: NONE
IP-Address:
Port:

Device specific

Download config | Test connection | Test

RESI-KNX-MBMASTER KNX to MODBUS/RTU MASTER module

Software version: 1.0.1

State: no error

Upload config

Modbus address: 255
Modbus parity: NONE
KNX address: 15.15.255
HELP

KNX MB/RTU MASTER Test Bench

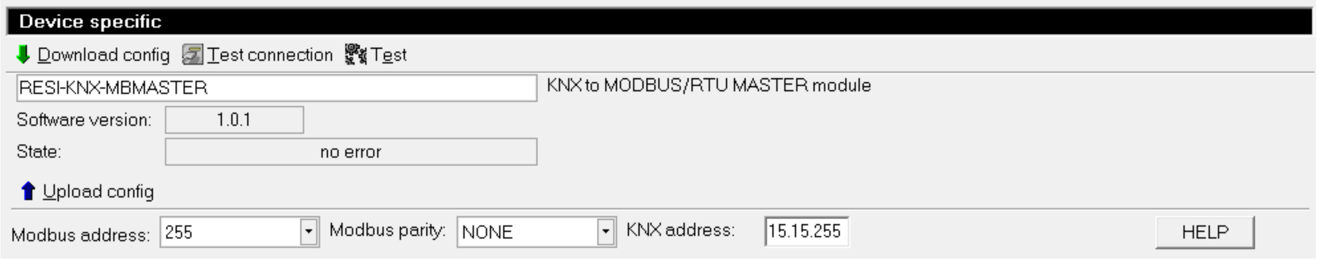
KNX group	KNX datatype	KNX direction	KNX delta	KNX interval	Unit ID	MODBUS register	MODBUS datatype	Factor

MB/RTU ID	Trigger	Command	Start	Length	Timeout	Repeats	Pause	(

Canceled device scan!

6.2 Basic functions

As you can see, you can do individual setup for the gateway in the above area of the screen:



Device specific

Download config | Test connection | Test

RESI-KNX-MBMASTER — KNX to MODBUS/RTU MASTER module

Software version: 1.0.1

State: no error

Upload config

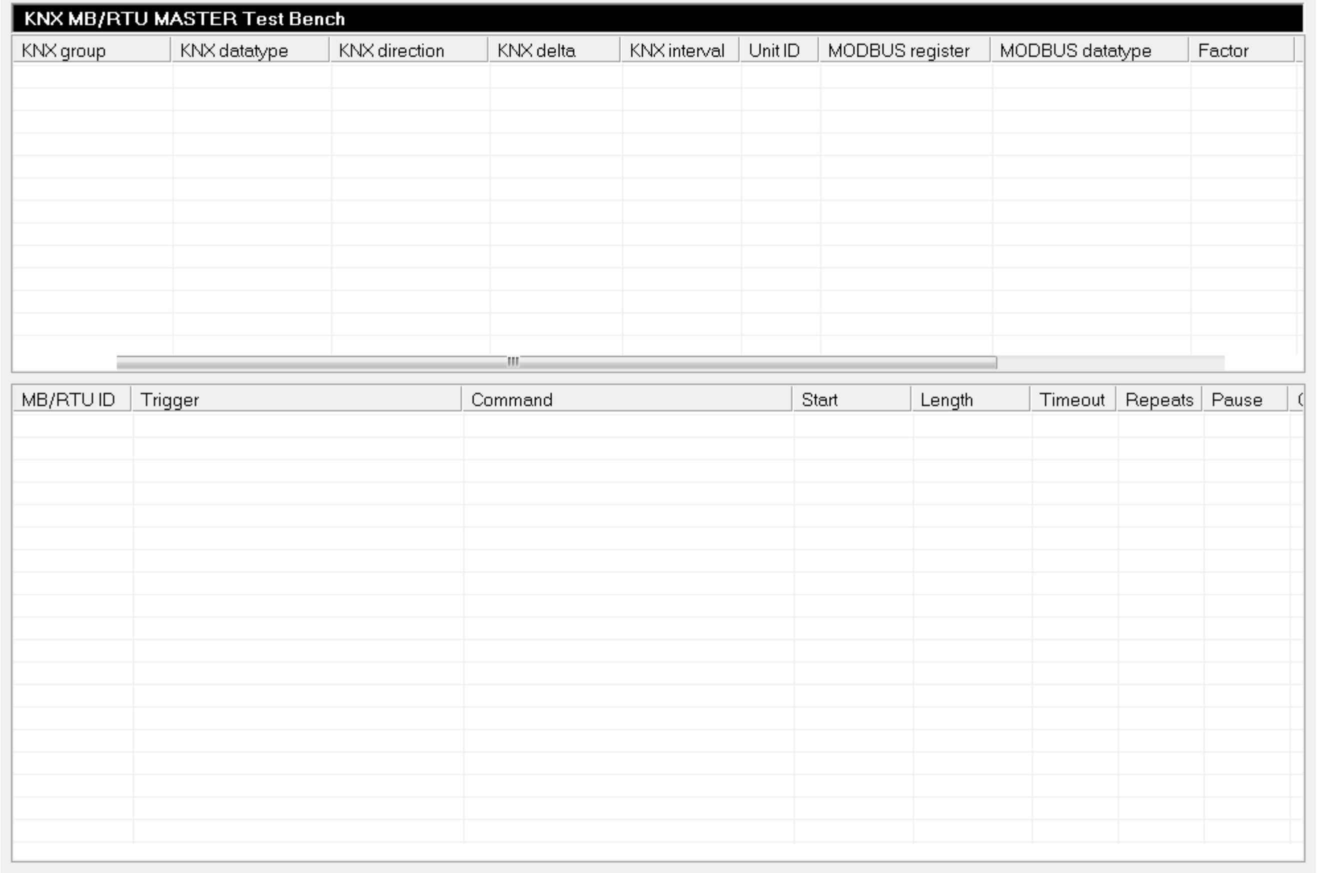
Modbus address: 255 | Modbus parity: NONE | KNX address: 15.15.255 | HELP

IMPORTANT HINT: The gateway can only communicate with the PC software if the module is in configuration mode and does not work as a MODBUS/RTU master. This mode switching MUST be made on the module. The DIP Switch FD serves this purpose. This switch must be set to OFF during configuration phase!

- Button “Download config”: If you change the MODBUS/RTU slave address, the MODBUS parity or KNX address or if you change the KNX mapping, you have to download the new configuration to the gateway to activate the changes.
- Button “Upload config”: With this button you can upload the complete mapping of the converter into the software. But remember, the comments are not stored into the gateway, so this information is lost, if you upload the mapping from a gateway!
- Button “Test connection”: This button tests, if the software can communicate with the gateway or not.
- Button “Test”: This button activates/deactivates a test function. This button is useless for this gateway because the PC software cannot access the gateway during an active MODBUS/RTU master communication!

6.3 The configuration table

The section "KNX MB/RTU MASTER Test Bench" shows two tables for configuring the assignment of the data between the connected MODBUS/RTU slave devices and the KNX bus system:



KNX MB/RTU MASTER Test Bench

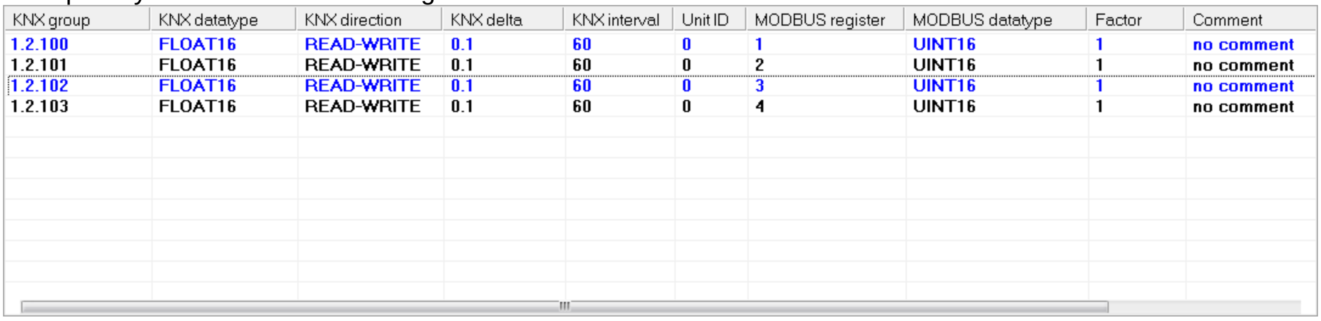
KNX group	KNX datatype	KNX direction	KNX delta	KNX interval	Unit ID	MODBUS register	MODBUS datatype	Factor

MB/RTU ID	Trigger	Command	Start	Length	Timeout	Repeats	Pause

This is done in two tables: The upper table defines the mapping between individual MODBUS registers of MODBUS slave devices on the KNX. The table below defines the behavior of the MODBUS/RTU master in the gateway and thus the type of data query and data transfer on the MODBUS side.

6.3.1 Table KNX – MODBUS Register

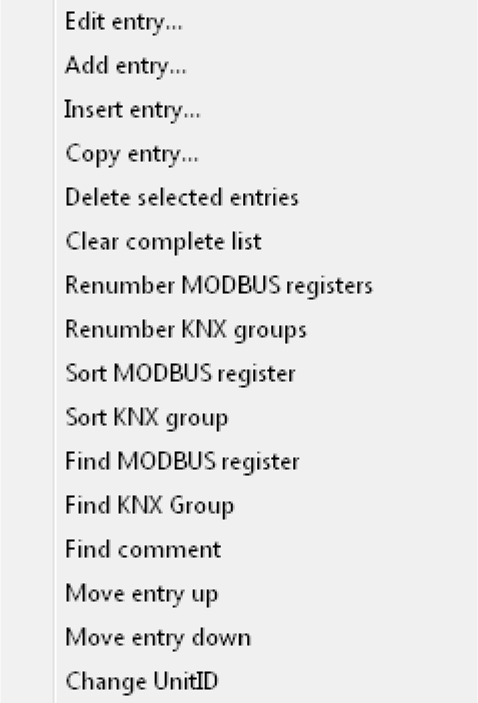
A map entry consists of the following entries:



KNX group	KNX datatype	KNX direction	KNX delta	KNX interval	Unit ID	MODBUS register	MODBUS datatype	Factor	Comment
1.2.100	FLOAT16	READ-WRITE	0.1	60	0	1	UINT16	1	no comment
1.2.101	FLOAT16	READ-WRITE	0.1	60	0	2	UINT16	1	no comment
1.2.102	FLOAT16	READ-WRITE	0.1	60	0	3	UINT16	1	no comment
1.2.103	FLOAT16	READ-WRITE	0.1	60	0	4	UINT16	1	no comment

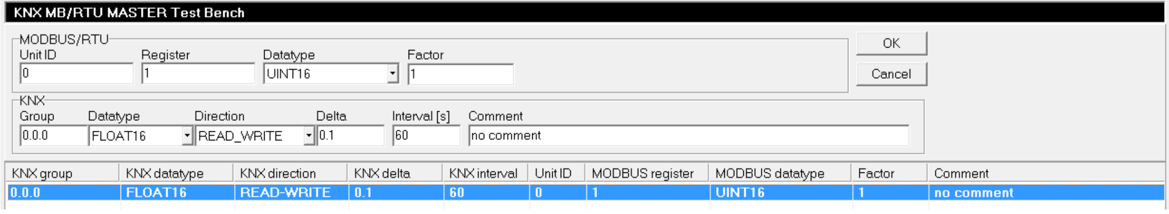
- **KNX group**: This entry defines the KNX group address that is used to send or receive KNX telegrams.
- **KNX datatype**: This entry defines the KNX data type that is used when sending or receiving KNX data with the set KNX group address.
- **KNX direction**: This entry defines the communication direction on the KNX bus: You can send KNX data to the KNX bus (WRITE), receive (READ) or both (READ-WRITE).
- **KNX delta**: This entry defines the deviation of the current value read by the MODBUS from the last value stored in the gateway. If the actual deviation is greater than this floating-point number, the value is sent again on the KNX. If this value is 0, this function will not be used.
- **KNX interval**: This entry defines an interval in seconds. When this interval expires, the value last received by the MODBUS is sent again on the KNX. If this interval equals 0, this function is deactivated.
- **Unit ID**: This entry defines for which MODBUS/RTU slave devices this KNX mapping applies at all. If the unit ID of the MODBUS/RTU slave device queried corresponds to the unit ID entered here, the received MODBUS value is output on the KNX depending on the settings of KNXdelta and KNXinterval. A unit ID of 0 means that every received MODBUS telegram is transmitted to the KNX.
- **MODBUS register**: The number of the register in the MODBUS/RTU slave device, which is used for this mapping. The number of used MODBUS registers results from the set MODBUS datatype.
- **MODBUS datatype**: The data type for the MODBUS register. This entry defines how the converter performs the conversion between KNX data and MODBUS data and how many MODBUS registers are used to calculate the KNX value.
- **Factor**: This entry defines a factor by which incoming MODBUS data is multiplied before it is sent on the KNX. When receiving KNX telegrams, the KNX data is divided by this factor before these values are passed on to the MODBUS/RTU slave device. The value zero defines that this factor is unused.
- **Comment**: This entry defines a custom text as an explanation for this mapping. This is intended only for the documentation and is not stored in the gateway, but only if you save the PC project. When uploading a configuration from a gateway into the software, this comment is not included and will be replaced by a standard text.

The context menu offers the following options:



6.3.1.1 Action KNX-MODBUS „Edit entry…“

This action activates the input window above the two tables for the selected entry. Here you can edit all values of the entry. The OK button accepts the changed data, the Cancel button discards all changes made.



MODBUS/RTU

Unit ID	Register	Datatype	Factor
0	1	UINT16	1

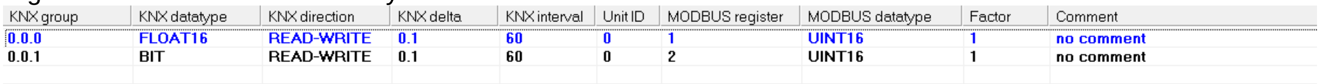
KNX

Group	Datatype	Direction	Delta	Interval [s]	Comment
0.0.0	FLOAT16	READ_WRITE	0.1	60	no comment

KNX group	KNX datatype	KNX direction	KNX delta	KNX interval	Unit ID	MODBUS register	MODBUS datatype	Factor	Comment
0.0.0	FLOAT16	READ-WRITE	0.1	60	0	1	UINT16	1	no comment

6.3.1.2 Action KNX-MODBUS „Add entry…“

This action adds a new record at the end of the configuration table. The KNX group address and the MODBUS register number are automatically increased.



KNX group	KNX datatype	KNX direction	KNX delta	KNX interval	Unit ID	MODBUS register	MODBUS datatype	Factor	Comment
0.0.0	FLOAT16	READ-WRITE	0.1	60	0	1	UINT16	1	no comment
0.0.1	BIT	READ-WRITE	0.1	60	0	2	UINT16	1	no comment

6.3.1.3 Action KNX-MODBUS „Insert entry…“

This action inserts a new record in front of the currently selected row in the table.

6.3.1.4 Action KNX-MODBUS „Copy entry…“

This action copies all selected lines and appends the new configuration lines at the end of the table.

6.3.1.5 Action KNX-MODBUS „Delete selected entries…“

This action deletes all selected lines from the configuration table.

6.3.1.6 Action KNX-MODBUS „Clear complete list“

This action deletes the entire configuration table if the query is answered with YES.

6.3.1.7 Action KNX-MODBUS „Renumber MODBUS registers“

This action renumbers the MODBUS registers in all marked lines. The starting value is the MODBUS register in the first marked line.

6.3.1.8 Action KNX-MODBUS „Renumber KNX groups“

This action renumbers the KNX groups in all marked lines. The start value is the configured KNX group in the first marked line.

6.3.1.9 Action KNX-MODBUS „Sort MODBUS registers“

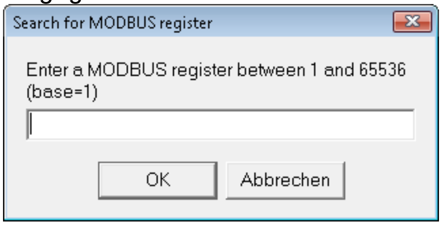
This action sorts the entire table in ascending order after the MODBUS register numbers.

6.3.1.10 Action KNX-MODBUS „Sort KNX group“

This action sorts the entire table after a query in ascending order of the KNX groups.

6.3.1.11 Action KNX-MODBUS „Find MODBUS register“

Diese Aktion markiert in der Tabelle alle jene Zeilen, in der die MODBUS Registernummer mit der im Dialog eingegebenen Nummer übereinstimmt.

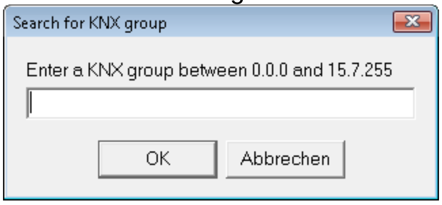


Search for MODBUS register

Enter a MODBUS register between 1 and 65536 (base=1)

6.3.1.12 Action KNX-MODBUS „Find KNX Group“

This action marks in the table all those lines in which the MODBUS register number matches the number entered in the dialog.



Search for KNX group

Enter a KNX group between 0.0.0 and 15.7.255

6.3.1.13 Action KNX-MODBUS „Find comment“

This action marks in the table all those lines in which a part of the comment matches the text entered in the dialog.

6.3.1.14 Action KNX-MODBUS „Move entry up“

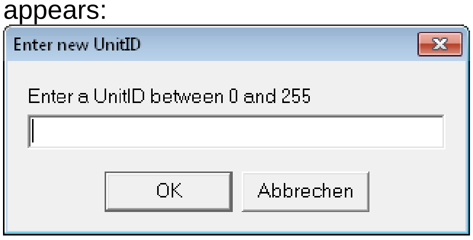
This action moves the current entry one line up.

6.3.1.15 Action KNX-MODBUS „Move entry down“

This action moves the current entry down one line.

6.3.1.16 Action KNX-MODBUS „Change UnitID“

This action changes all MODBUS Unit IDs in the selected rows. First, an input window for the new UnitID appears:

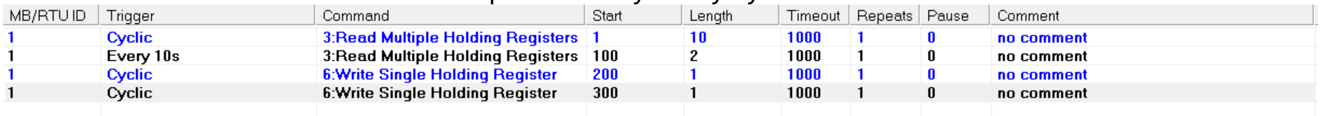


Enter new UnitID

Enter a UnitID between 0 and 255

6.3.2 Table MODBUS/RTU Master Telegrams

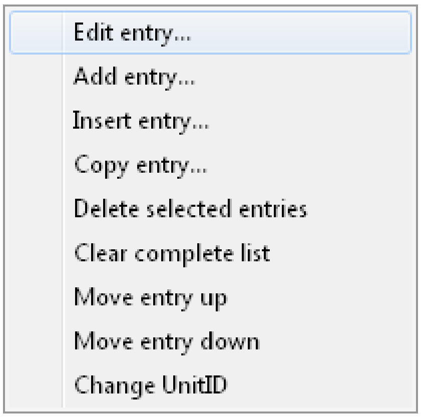
Here all telegrams used by the MODBUS/RTU master are configured to communicate with the connected MODBUS/RTU slave devices. This list is processed cyclically by the MODBUS/RTU master:



MB/RTU ID	Trigger	Command	Start	Length	Timeout	Repeats	Pause	Comment
1	Cyclic	3:Read Multiple Holding Registers	1	10	1000	1	0	no comment
1	Every 10s	3:Read Multiple Holding Registers	100	2	1000	1	0	no comment
1	Cyclic	6:Write Single Holding Register	200	1	1000	1	0	no comment
1	Cyclic	6:Write Single Holding Register	300	1	1000	1	0	no comment

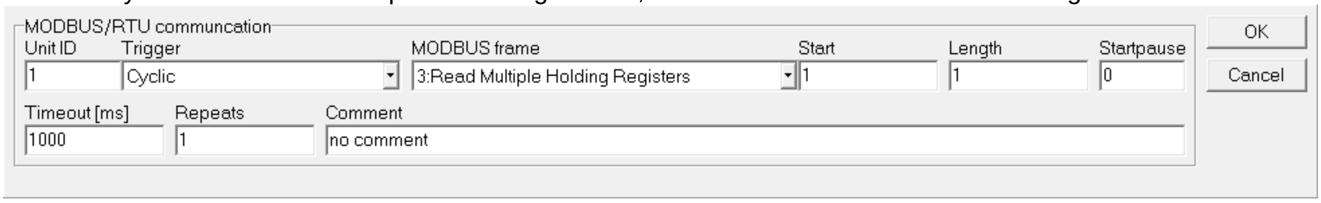
- **MB/RTU ID:** This entry defines for which MODBUS/RTU slave device this telegram is defined.
- **Trigger:** This entry defines when this telegram is generated in the MODBUS/RTU master. The following triggers are available:
  - **Cyclic:** In each cycle, this telegram is processed
  - **Every 1s:** This telegram is processed in a 1 second cycle
  - **Every 10s:** This telegram is processed in a 10 second cycle
  - **Every 60s:** This telegram is processed in a 60 second cycle
  - **On KNX value received:** This telegram is only processed if data has been received on the KNX bus for this telegram.
- **Command:** This entry defines which type of MODBUS/RTU master telegram should be generated. The following telegrams are available:
  - **1: Read Coils:** Query several bits
  - **2: Read Discrete Inputs:** Query several bits
  - **3: Read Multiple Holding Registers:** Querying multiple registers
  - **4: Read Input Registers:** Querying multiple registers
  - **5: Write Single Coil:** Write a single bit
  - **6: Write Single Holding Register:** Write a single register
  - **15: Write Multiple Coils:** Write several bits at once
  - **16: Write Multiple Holding Registers:** Write multiple registers at once
  - **23: Read/Write Multiple Registers:** Read and write several registers at once
  - **99: Pause in ms:** Take a break without communicating
- **Start:** This entry defines the start index from where registers or bits are read or written. The start index for MODBUS is between 1 and 65536.
- **Length:** This entry defines the number of registers that are written or read per telegram. Here a length between 1 and 250 is permissible.
- **Timeout:** This entry defines the maximum response time for the MODBUS slave for all MODBUS telegrams. The information is given in milliseconds. Only with function "99: pause in ms" will this value be used as pause time, before proceeding with the next MODBUS request.
- **Repeats:** This entry defines how often the MODBUS telegram should be repeated if the slave does not answer or returns an erroneous answer.
- **Pause:** This entry defines how long to wait before sending the MODBUS telegram. The time is given in milliseconds.
- **Comment**: This entry defines a custom text as an explanation for this mapping. This is intended only for the documentation and is not stored in the gateway, but only if you save the PC project. When uploading a configuration from a gateway into the software, this comment is not included and will be replaced by a standard text.

The context menu offers the following options:



6.3.2.1 Action MODBUS/RTU Master Telegram „Edit entry…“

This action activates the input window above the two tables for the selected entry. Here you can edit all values of the entry. The OK button accepts the changed data, the Cancel button discards all changes made.



MODBUS/RTU communcation

Unit ID: 1
Trigger: Cyclic
MODBUS frame: 3:Read Multiple Holding Registers
Start: 1
Length: 1
Startpause: 0
Timeout [ms]: 1000
Repeats: 1
Comment: no comment

6.3.2.2 Action MODBUS/RTU Master Telegram „Insert entry…“

This action inserts a new record in front of the currently selected row in the table.

6.3.2.3 Action MODBUS/RTU Master Telegram „Copy entry…“

This action copies all selected lines and appends the new configuration lines at the end of the table.

6.3.2.4 Action MODBUS/RTU Master Telegram „Delete selected entries…“

This action deletes all selected lines from the configuration table.

6.3.2.5 Action MODBUS/RTU Master Telegram „Clear complete list“

This action deletes the entire configuration table if the query is answered with YES.

6.3.2.6 Action MODBUS/RTU Master Telegram „Move entry up“

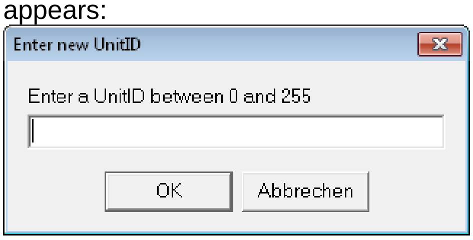
This action moves the current line up one position.

6.3.2.7 Action MODBUS/RTU Master Telegram „Move entry down“

This action moves the current line one position down.

6.3.2.8 Action MODBUS/RTU Master Telegram „Change UnitID“

This action changes all MODBUS Unit IDs in the selected rows. First, an input window for the new UnitID appears:



Enter new UnitID

Enter a UnitID between 0 and 255

6.3.3 MODBUS Data types

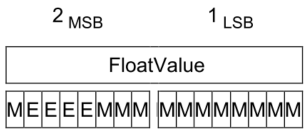
Here is an overview of the MODBUS/RTU master data types:

On the one hand, this data type defines how many MODBUS holding registers are actually used for this entry (eg: the data type UINT16 requires a holding register, whereas the data type FLOAT32 requires two consecutive MODBUS holding registers), and on the other hand defines this data type, how the KNX data are stored in these registers (eg: the data type FLOAT32 stores the higher-order 16-bit word in the first MODBUS holding register and the low-order 16-bit word in the second holding register, the data type FLOAT32R reverses this storage: here the low-order 16-bit word of the 32-bit value in the first holding register and the higher-order 16-bit word in the second holding register).

MODBUS DATATYPE	**SIZE**	**WORD ORDER**	**DESCRIPTION**
ERR	none	none	Defines an invalid configuration entry and is ignored by the gateway
UINT16	16 bits 1 register	none	Defines a 16 bit unsigned integer value in the range of 0 to 65535 or 0x0000 to 0xFFFF
SINT16	16 bits 1 register	none	Defines a 16 bit signed integer value in the range of -32768 to +32767 or 0x8000 to 0x7FFF
UINT32	32 bits 2 register	0:High Word 1:Low Word	Defines a 32 bit unsigned integer value in the range of 0 to 4.294.967.295 or 0x00000000 to 0xFFFFFFFF
SINT32	32 bits 2 register	0:High Word 1:Low Word	Defines a 32 bit signed integer value in the range of −2.147.483.648 to +2.147.483.647or 0x80000000 to 0x7FFFFFFF
UINT32R	32 bits 2 register	0:Low Word 1:High Word	Defines a 32 bit unsigned integer value in the range of 0 to 4.294.967.295 or 0x00000000 to 0xFFFFFFFF with reverse word order
SINT32R	32 bits 2 register	0:Low Word 1:High Word	Defines a 32 bit signed integer value in the range of −2.147.483.648 to +2.147.483.647or 0x80000000 to 0x7FFFFFFF with reverse word order
FLOAT32	32 bits 2 register	0:High Word 1:Low Word	Defines a 32 bit float value in the range of $\pm 1.4 \cdot 10^{-45}$ to $\pm 3.403 \cdot 10^{38}$. A mantissa of 23 bits and an exponent of 8 bits are used. The value can store 7 to 8 digits after the comma.
FLOAT32R	32 bits 2 register	0:Low Word 1:High Word	Defines a 32 bit float value in the range of $\pm 1.4 \cdot 10^{-45}$ to $\pm 3.403 \cdot 10^{38}$. A mantissa of 23 bits and an exponent of 8 bits are used. The value can store 7 to 8 digits after the comma. The two 16 bit words are stored in reverse order.
DOUBLE64	64 bits 4 register	0:Highest Word 1:Higer Word 2:Lower Word 3:Lowest Word	Defines a 64 bit float value in the range of $\pm 4.24 \cdot 10^{-324}$ to $\pm 1{,}798 \cdot 10^{308}$. A mantissa of 52 bits and an exponent of 11 bits are used. The value can store 15 to 16 digits after the comma.
DOUBLE64R	64 bits 4 register	0:Lowest Word 1:Lower Word 2:Higher Word 3:Highest Word	Defines a 64 bit float value in the range of $\pm 4.24 \cdot 10^{-324}$ to $\pm 1{,}798 \cdot 10^{308}$. A mantissa of 52 bits and an exponent of 11 bits are used. The value can store 15 to 16 digits after the comma. The four 16 bit words are stored in reverse order.
GENERIC	64 bits 4 register	0: FIRST and SECOND byte 1: THIRD and FOURTH byte 2: FIFTH and SIXTH byte 3: SEVENTH and EIGHT byte	Currently unused
STRING	64 bits 4 register	0: FIRST and SECOND byte 1: THIRD and FOURTH byte 2: FIFTH and SIXTH byte 3: SEVENTH and EIGHT byte	Currently unused

6.3.4 KNX Data types

Here is an overview of the KNX data types:

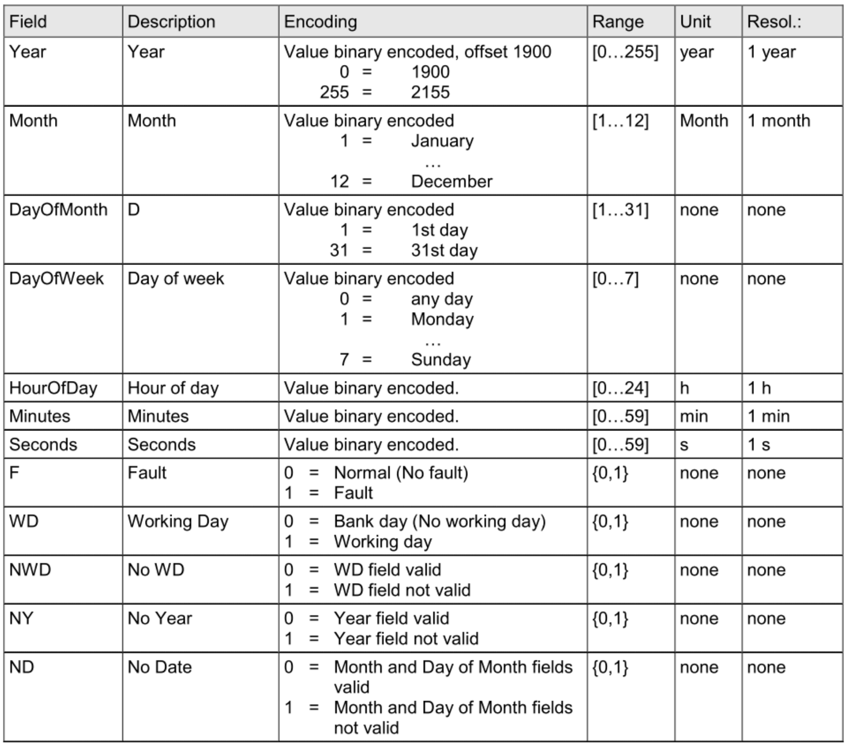
KNX DATATYPE	SIZE	DESCRIPTION
ERR	none	Defines an invalid configuration entry and is ignored by the gateway
BIT	1 bit	Defines a bit value in the range from 0 to 1 or 0x0 to 0x1. Often interpreted as OFF and ON.
TWOBITS	2 bits	Defines an integer value consisting out of two bits in the range from 0 to 3 or 0x0 to 0x3
FOURBITS	4 bits	Defines an integer value consisting out of four bits in the range from 0 to 15 or 0x0 to 0xF.
SIXBITS	6 bits	Defines an integer value consisting out of six bits in the range from 0 to 63 or 0x00 to 0x3F.
CHARACTER	8 bits	Defines one text character consisting out of eight bits in the range from 0 to 255 or 0x00 to 0xFF. Please refer to the KNX documentation, how the encoding of the text character is done by the KNX standard. The encoding can be done for ASCII characters or for ISO 8859.1 characters.
UINT8	8 bits	Defines a 8 bit unsigned integer value in the range of 0 to 255 or 0x00 to 0xFF
SINT8	8 bits	Defines a 8 bit signed integer value in the range of -128 to +127 or 0x80 to 0x7F
UINT16	16 bits	Defines a 16 bit unsigned integer value in the range of 0 to 65535 or 0x0000 to 0xFFFF
SINT16	16 bits	Defines a 16 bit signed integer value in the range of -32768 to +32767 or 0x8000 to 0x7FFF
UINT32	32 bits	Defines a 32 bit unsigned integer value in the range of 0 to 4.294.967.295 or 0x00000000 to 0xFFFFFFFF
SINT32	32 bits	Defines a 32 bit signed integer value in the range of −2.147.483.648 to +2.147.483.647or 0x80000000 to 0x7FFFFFFF
FLOAT16	16 bits	Defines a 16 bit float value with a 4 bit exponent and a 12 bit mantissa.  2 MSB 1 LSB FloatValue M E E E E M M M M M M M M M M M FloatValue = (0,01*M)*2$^{(E)}$ E = [0 … 15] M = [-2 048 … 2 047], two's complement notation For all Datapoint Types 9.xxx, the encoded value 7FFFh shall always be used to denote invalid data. [-671 088,64 … 670 760,96]

<table>
<tr><td>FLOAT32</td><td>32 bits</td><td>
Defines a 32 bit float value in the range of ±1.4·10^{-45} to ±3.403·10^{38}. A mantissa of 23 bits, and an exponent of 8 bits are used. The value can store 7 to 8 digits after the comma.
<table>
<tr><td>4 MSB</td><td>3</td><td>2</td><td>1 LSB</td></tr>
<tr><td>S Exponent</td><td colspan="3">Fraction</td></tr>
<tr><td>F F F F F F F F</td><td>F F F F F F F F</td><td>F F F F F F F F</td><td>F F F F F F F F</td></tr>
</table>
The values are encoded in the IEEE floating point format according IEEE 754 single precision format.

NOTE 7 This specifies that the exponent is biased. This allows negative exponent values.

S (Sign) = {0,1}
Exponent = [0 … 255]
Fraction = [0 … 8 388 607]

The resolution is given by the use of the IEEE 754 format and varies with the used exponent.
</td></tr>
<tr><td>TIME</td><td>24 bits</td><td>
Defines a 24 bit value encoded a time information in the following way:
<table>
<tr><td>3 MSB</td><td>2</td><td>1 LSB</td></tr>
<tr><td>0 0 0 Day</td><td>0 0 0 0 Month</td><td>0 Year</td></tr>
<tr><td>r r r U U U U U</td><td>r r r r U U U U</td><td>r U U U U U U U</td></tr>
</table>
<table>
<tr><th>Field:</th><th>Encoding:</th><th>Range:</th><th>Unit:</th><th>Resol.:</th></tr>
<tr><td>Day</td><td>1 = Monday
…
7 = Sunday
0 = no day</td><td>[0…7]</td><td>none</td><td>none</td></tr>
<tr><td>Hour</td><td>binary encoded</td><td>[0…23]</td><td>hours</td><td>h</td></tr>
<tr><td>Minutes</td><td>binary encoded</td><td>[0…59]</td><td>minutes</td><td>min</td></tr>
<tr><td>Seconds</td><td>binary encoded</td><td>[0…59]</td><td>seconds</td><td>s</td></tr>
</table>
</td></tr>
<tr><td>DATE</td><td>24 bits</td><td>
Defines a 24 bit value encoded a date information in the following way:
<table>
<tr><td>3 MSB</td><td>2</td><td>1 LSB</td></tr>
<tr><td>Day Hour</td><td>0 0 Minutes</td><td>0 0 Seconds</td></tr>
<tr><td>N N N U U U U U</td><td>r r U U U U U U</td><td>r r U U U U U U</td></tr>
</table>
<table>
<tr><th>Field:</th><th>Range:</th><th>Unit:</th><th>Resol.:</th></tr>
<tr><td>Day</td><td>[1…31]</td><td>Day of month</td><td>1 day</td></tr>
<tr><td>Month</td><td>[1…12]</td><td>Month</td><td>1 month</td></tr>
<tr><td>Year</td><td>[0…99]</td><td>Year</td><td>1 year</td></tr>
</table>
</td></tr>
<tr><td>DATETIME</td><td>64 bits</td><td>
Defines a 64 bit value encoded a date and time information in the following way:
<table>
<tr><td>8 MSB</td><td>7</td><td>6</td><td>5</td></tr>
<tr><td>Year</td><td>0 0 0 0 Month</td><td>0 0 0 DayOfMonth</td><td>DayOf-Week HourOfDay</td></tr>
<tr><td>U U U U U U U U</td><td>r r r r U U U U</td><td>r r r U U U U U</td><td>U U U U U U U U</td></tr>
<tr><td>4</td><td>3</td><td>2</td><td>1 LSB</td></tr>
<tr><td>0 0 Minutes</td><td>0 0 Seconds</td><td>F WD NWD NY ND NDoW NT SUTI</td><td>CLQ 0 0 0 0 0 0 0</td></tr>
<tr><td>r r U U U U U U</td><td>r r U U U U U U</td><td>B B B B B B B B</td><td>B r r r r r r r</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td></td><td>

<table>
<tr><th>Field</th><th>Description</th><th>Encoding</th><th>Range</th><th>Unit</th><th>Resol.:</th></tr>
<tr><td>Year</td><td>Year</td><td>Value binary encoded, offset 1900
0 = 1900
255 = 2155</td><td>[0…255]</td><td>year</td><td>1 year</td></tr>
<tr><td>Month</td><td>Month</td><td>Value binary encoded
1 = January
…
12 = December</td><td>[1…12]</td><td>Month</td><td>1 month</td></tr>
<tr><td>DayOfMonth</td><td>D</td><td>Value binary encoded
1 = 1st day
31 = 31st day</td><td>[1…31]</td><td>none</td><td>none</td></tr>
<tr><td>DayOfWeek</td><td>Day of week</td><td>Value binary encoded
0 = any day
1 = Monday
…
7 = Sunday</td><td>[0…7]</td><td>none</td><td>none</td></tr>
<tr><td>HourOfDay</td><td>Hour of day</td><td>Value binary encoded.</td><td>[0…24]</td><td>h</td><td>1 h</td></tr>
<tr><td>Minutes</td><td>Minutes</td><td>Value binary encoded.</td><td>[0…59]</td><td>min</td><td>1 min</td></tr>
<tr><td>Seconds</td><td>Seconds</td><td>Value binary encoded.</td><td>[0…59]</td><td>s</td><td>1 s</td></tr>
<tr><td>F</td><td>Fault</td><td>0 = Normal (No fault)
1 = Fault</td><td>{0,1}</td><td>none</td><td>none</td></tr>
<tr><td>WD</td><td>Working Day</td><td>0 = Bank day (No working day)
1 = Working day</td><td>{0,1}</td><td>none</td><td>none</td></tr>
<tr><td>NWD</td><td>No WD</td><td>0 = WD field valid
1 = WD field not valid</td><td>{0,1}</td><td>none</td><td>none</td></tr>
<tr><td>NY</td><td>No Year</td><td>0 = Year field valid
1 = Year field not valid</td><td>{0,1}</td><td>none</td><td>none</td></tr>
<tr><td>ND</td><td>No Date</td><td>0 = Month and Day of Month fields valid
1 = Month and Day of Month fields not valid</td><td>{0,1}</td><td>none</td><td>none</td></tr>
</table>
</td></tr>
<tr><td>STRING</td><td>max. 14 bytes</td><td>Defines up to 14 bytes of text data

14 MSB
…
1 LSB
Character 1
…
Character 14
AAAAAAAA
AAAAAAAA

These Datapoint Types are used to transmit strings of textual characters. The length is fixed to 14 octets. The contents are filled starting from the most significant octet. Each octet shall be encoded as specified for the chosen character set, as defined in clause 0. If the string to be transmitted is smaller then 14 octets, unused trailing octets in the character string shall be set to NULL (00h).
Example: ‘KNX is OK’ is encoded as follows :
4B 4E 58 20 69 73 20 4F 4B 00 00 00 00 00</td></tr>
<tr><td>GENERIC</td><td>64 bits</td><td>Defines a 64 bit value which represents up to 8 bytes from the data section of a generic KNX telegram. Due to the fact, that a generic KNX frame can hold up to 14 bytes, the field factor defines the start index for the 8 bytes in the range from 0 to 13. The system stores the first byte in the first 16 bit MODBUS register in the low 8 bits. The next byte is stored in the same register, but in the upper half of the word. The third byte is stored in the next Modbus register in the low half, and so on.</td></tr>
</table>

7 HOWTO change baud rate for DIP switch setting ON,ON

The converter supports different baud rate settings depending on the position of the first two DIP switches (Baudrate BR):

Use DIP Switches 1+2 to select baud rate:

OFF	OFF:	9600Bd
ON	OFF:	19200Bd
OFF	ON:	38400Bd
ON	ON:	Baudrate from FLASH, usually 57600Bd

HINT: The correct parity (NONE, EVEN or ODD) is configured with the PC software, not via DIP switches!

To set a different baud rate in FLASH you have two possibilities:

7.1 Change baud rate with ASCII command

Open an ASCII terminal program like hterm. Open a connection with 57600baud to the converter.

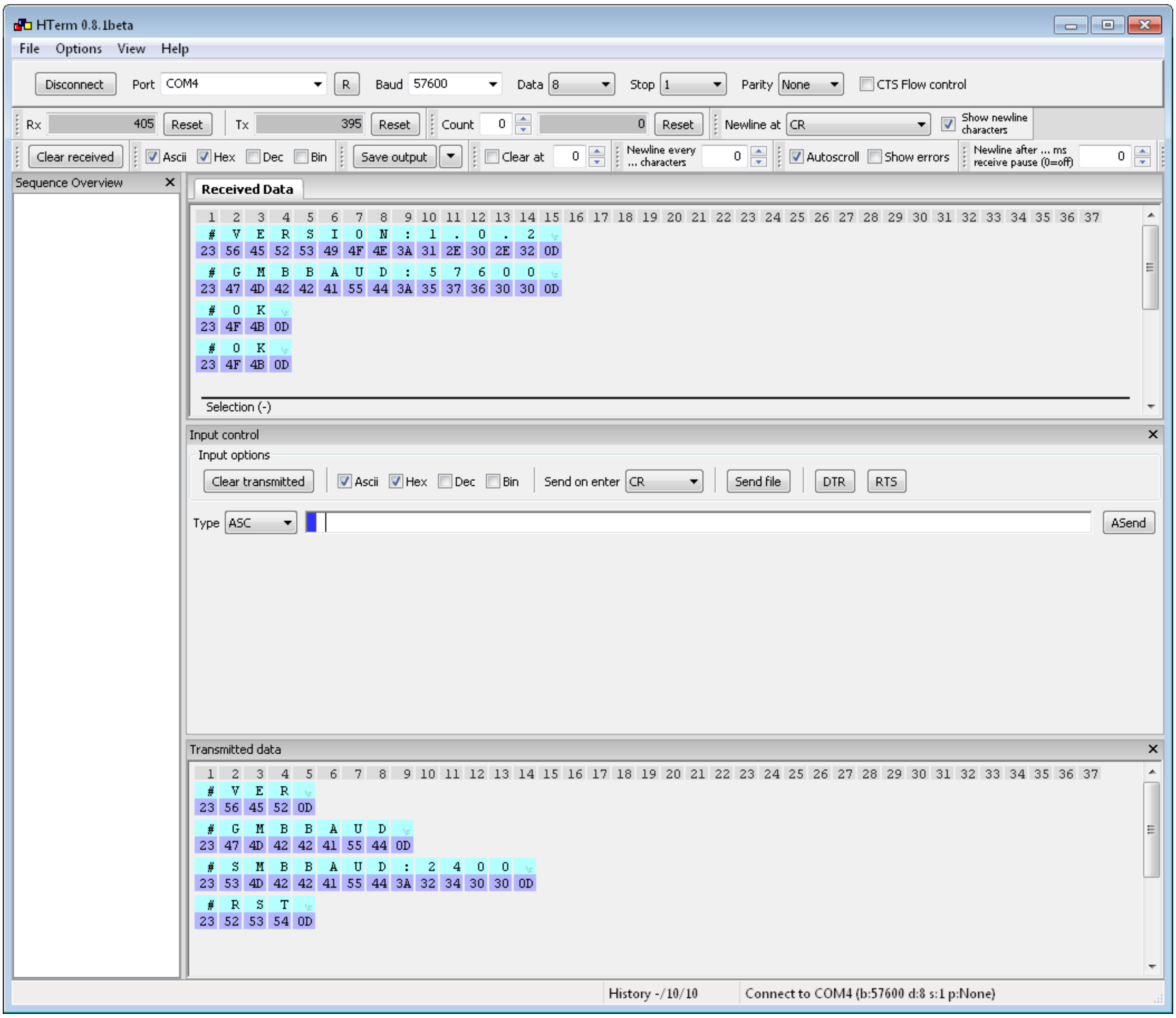
1. Check the connection using the following ASCII command
   (<CR> stands for carriage return character 0x0d):
   ```
   PC->CONVERTER:          #VER<CR>
   CONVERTER -> PC:        #VERSION:1.0.2<CR>
   ```

2. Check the current setting of the baud rate with the command:
   ```
   PC->CONVERTER:          #GMBBAUD<CR>
   CONVERTER -> PC:        #GMBBAUD:57600<CR>
   ```

3. Set the new desired baud rate with the command:
   ```
   PC->CONVERTER:          #SMBBAUD:2400<CR>
   CONVERTER->PC:          #OK<CR>
   ```
   HINT: The new setting will be activated after a power on of the module or a software reset!
   Valid baud rates are: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600

4. Restart the converter with the new settings with the command:
   ```
   PC->CONVERTER:          #RST<CR>
   CONVERTER->PC:          #OK<CR>
   ```

The new settings will be used from now!



Port COM4 · Baud 57600 · Data 8 · Stop 1 · Parity None · ☐ CTS Flow control

Rx 405 · Tx 395 · Count 0 · Newline at CR · ☑ Show newline characters

☑ Ascii ☑ Hex ☐ Dec ☐ Bin · ☐ Clear at 0 · Newline every ... characters 0 · ☑ Autoscroll ☐ Show errors · Newline after ... ms receive pause (0=off) 0

Received Data:

```
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
 #  V  E  R  S  I  O  N  :  1  .  0  .  2
23 56 45 52 53 49 4F 4E 3A 31 2E 30 2E 32 0D
 #  G  M  B  B  A  U  D  :  5  7  6  0  0
23 47 4D 42 42 41 55 44 3A 35 37 36 30 30 0D
 #  O  K
23 4F 4B 0D
 #  O  K
23 4F 4B 0D
```

Selection (-)

Input options: ☑ Ascii ☑ Hex ☐ Dec ☐ Bin · Send on enter CR · Type ASC

Transmitted data:

```
 1  2  3  4  5  6  7  8  9 10 11 12 13 14
 #  V  E  R
23 56 45 52 0D
 #  G  M  B  B  A  U  D
23 47 4D 42 42 41 55 44 0D
 #  S  M  B  B  A  U  D  :  2  4  0  0
23 53 4D 42 42 41 55 44 3A 32 34 30 30 0D
 #  R  S  T
23 52 53 54 0D
```

History -/10/10 · Connect to COM4 (b:57600 d:8 s:1 p:None)

7.2 Change baud rate with MODBUS/RTU commands

Use a Modbus test software (e.g. ModbusPoll) and establish a connection to the converter.
Use the following registers to change the baud rate settings:

Register	Description
4x6001 3x6001 I:6000 W/O RESET SYSTEM	If the host writes to this register, the module executes a soft reset (reboot).
4x65223 3x65223 I:65222 R/W MODBUS BAUDRATE	If the host reads this register, the current stored baud rate for the DIP Switch settings (Baudrate BR) ON,ON is returned as Baudrate/100 value: =3 300Bd =6 600Bd =12 1200Bd =24 2400Bd =48 4800Bd =96 9600Bd =192 91200Bd =384 38400Bd =576 57600Bd Writing these values to this register will change the internal baud rate for the DIP setting ON,ON to the new desired Baudrate. But this Baudrate will be activated after a power off-power on cycle or after a software reset.

So follow these steps to change the Baudrate for DIP setting ON,ON:

1. Establish MODBUS/RTU connection to the module
2. Read out current value of holding register 4x65223, 3x65223, I:65222.
3. Write new baud rate value to holding register 4x65223, 3x65223, I:65222.
4. Restart module via software reset while writing the value 1 to register 4x06001, 3x06001, I:6000.

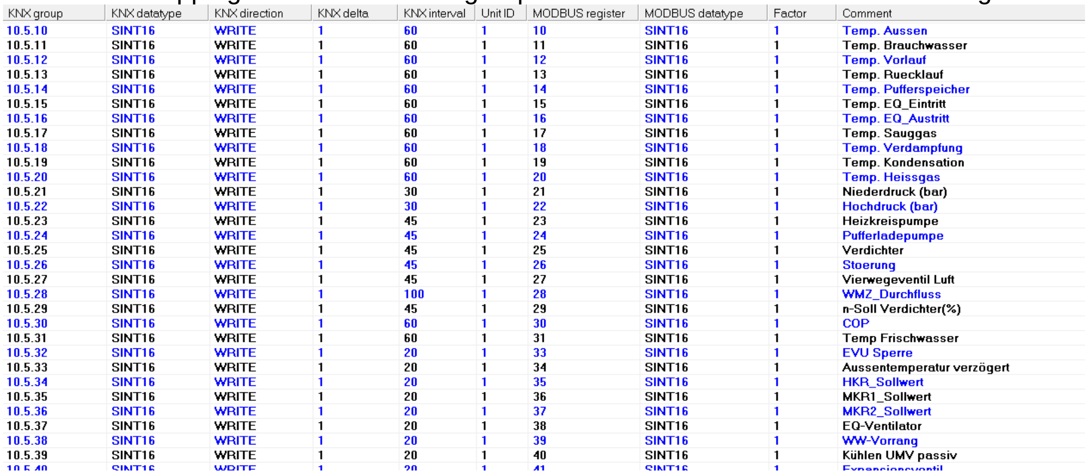
8 Sample configurations

Here are some sample configurations to explain how our gateway works and which configuration principle is used.

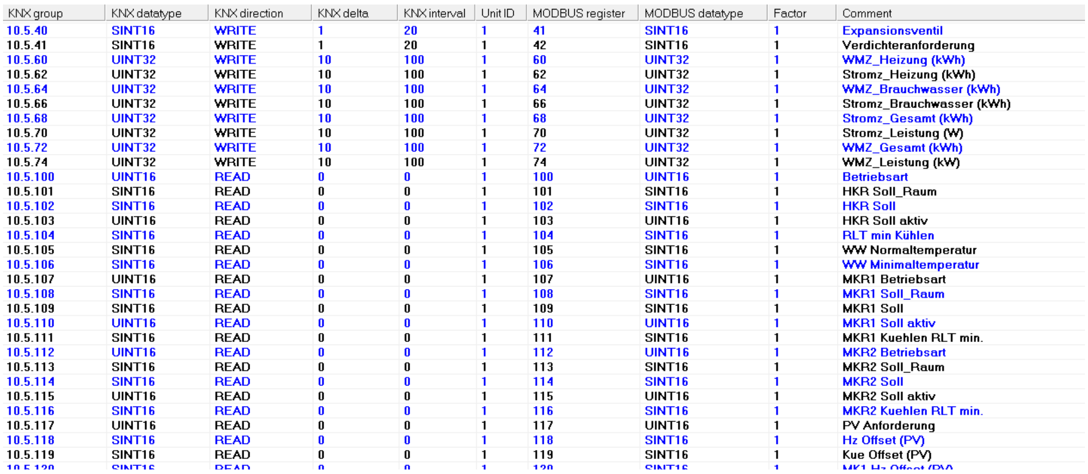
8.1 HELIOTHERM connection

The following connection was configured for a HELIOTHERM heat pump:

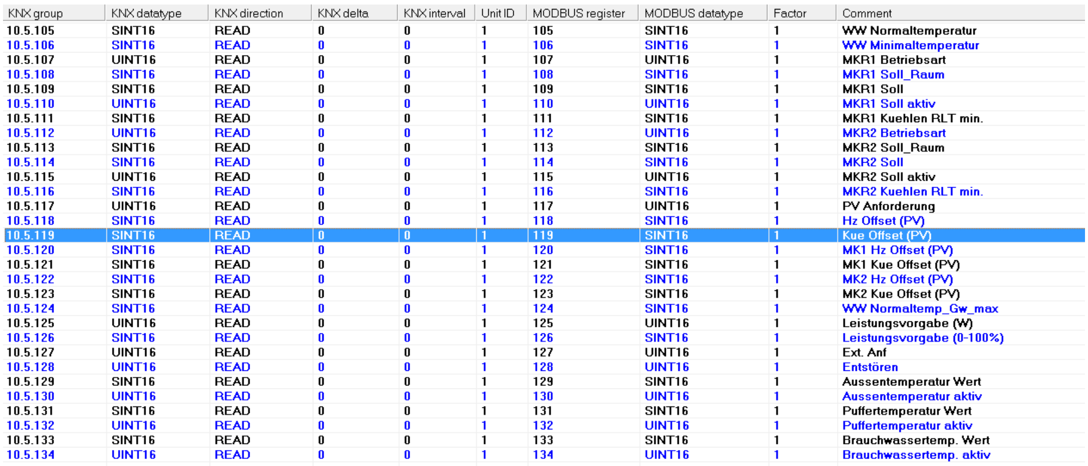
Here is the mapping between the KNX group addresses and the MODBUS RTU registers:



KNX group	KNX datatype	KNX direction	KNX delta	KNX interval	Unit ID	MODBUS register	MODBUS datatype	Factor	Comment
10.5.10	SINT16	WRITE	1	60	1	10	SINT16	1	Temp. Aussen
10.5.11	SINT16	WRITE	1	60	1	11	SINT16	1	Temp. Brauchwasser
10.5.12	SINT16	WRITE	1	60	1	12	SINT16	1	Temp. Vorlauf
10.5.13	SINT16	WRITE	1	60	1	13	SINT16	1	Temp. Ruecklauf
10.5.14	SINT16	WRITE	1	60	1	14	SINT16	1	Temp. Pufferspeicher
10.5.15	SINT16	WRITE	1	60	1	15	SINT16	1	Temp. EQ_Eintritt
10.5.16	SINT16	WRITE	1	60	1	16	SINT16	1	Temp. EQ_Austritt
10.5.17	SINT16	WRITE	1	60	1	17	SINT16	1	Temp. Sauggas
10.5.18	SINT16	WRITE	1	60	1	18	SINT16	1	Temp. Verdampfung
10.5.19	SINT16	WRITE	1	60	1	19	SINT16	1	Temp. Kondensation
10.5.20	SINT16	WRITE	1	60	1	20	SINT16	1	Temp. Heissgas
10.5.21	SINT16	WRITE	1	30	1	21	SINT16	1	Niederdruck (bar)
10.5.22	SINT16	WRITE	1	30	1	22	SINT16	1	Hochdruck (bar)
10.5.23	SINT16	WRITE	1	45	1	23	SINT16	1	Heizkreispumpe
10.5.24	SINT16	WRITE	1	45	1	24	SINT16	1	Pufferladepumpe
10.5.25	SINT16	WRITE	1	45	1	25	SINT16	1	Verdichter
10.5.26	SINT16	WRITE	1	45	1	26	SINT16	1	Stoerung
10.5.27	SINT16	WRITE	1	45	1	27	SINT16	1	Vierwegeventil Luft
10.5.28	SINT16	WRITE	1	100	1	28	SINT16	1	WMZ_Durchfluss
10.5.29	SINT16	WRITE	1	45	1	29	SINT16	1	n-Soll Verdichter(%)
10.5.30	SINT16	WRITE	1	60	1	30	SINT16	1	COP
10.5.31	SINT16	WRITE	1	60	1	31	SINT16	1	Temp Frischwasser
10.5.32	SINT16	WRITE	1	20	1	33	SINT16	1	EVU Sperre
10.5.33	SINT16	WRITE	1	20	1	34	SINT16	1	Aussentemperatur verzögert
10.5.34	SINT16	WRITE	1	20	1	35	SINT16	1	HKR_Sollwert
10.5.35	SINT16	WRITE	1	20	1	36	SINT16	1	MKR1_Sollwert
10.5.36	SINT16	WRITE	1	20	1	37	SINT16	1	MKR2_Sollwert
10.5.37	SINT16	WRITE	1	20	1	38	SINT16	1	EQ-Ventilator
10.5.38	SINT16	WRITE	1	20	1	39	SINT16	1	WW-Vorrang
10.5.39	SINT16	WRITE	1	20	1	40	SINT16	1	Kühlen UMV passiv
10.5.40	SINT16	WRITE	1	20	1	41	SINT16	1	Expansionsventil

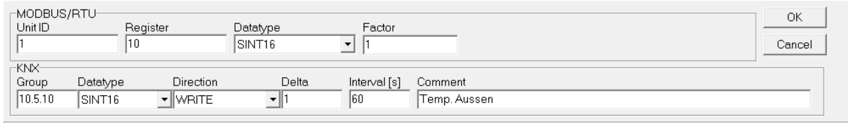


KNX group	KNX datatype	KNX direction	KNX delta	KNX interval	Unit ID	MODBUS register	MODBUS datatype	Factor	Comment
10.5.40	SINT16	WRITE	1	20	1	41	SINT16	1	Expansionsventil
10.5.41	SINT16	WRITE	1	20	1	42	SINT16	1	Verdichteranforderung
10.5.60	UINT32	WRITE	10	100	1	60	UINT32	1	WMZ_Heizung (kWh)
10.5.62	UINT32	WRITE	10	100	1	62	UINT32	1	Stromz_Heizung (kWh)
10.5.64	UINT32	WRITE	10	100	1	64	UINT32	1	WMZ_Brauchwasser (kWh)
10.5.66	UINT32	WRITE	10	100	1	66	UINT32	1	Stromz_Brauchwasser (kWh)
10.5.68	UINT32	WRITE	10	100	1	68	UINT32	1	Stromz_Gesamt (kWh)
10.5.70	UINT32	WRITE	10	100	1	70	UINT32	1	Stromz_Leistung (W)
10.5.72	UINT32	WRITE	10	100	1	72	UINT32	1	WMZ_Gesamt (kWh)
10.5.74	UINT32	WRITE	10	100	1	74	UINT32	1	WMZ_Leistung (kW)
10.5.100	UINT16	READ	0	0	1	100	UINT16	1	Betriebsart
10.5.101	SINT16	READ	0	0	1	101	SINT16	1	HKR Soll_Raum
10.5.102	SINT16	READ	0	0	1	102	SINT16	1	HKR Soll
10.5.103	UINT16	READ	0	0	1	103	UINT16	1	HKR Soll aktiv
10.5.104	SINT16	READ	0	0	1	104	SINT16	1	RLT min Kühlen
10.5.105	SINT16	READ	0	0	1	105	SINT16	1	WW Normaltemperatur
10.5.106	SINT16	READ	0	0	1	106	SINT16	1	WW Minimaltemperatur
10.5.107	UINT16	READ	0	0	1	107	UINT16	1	MKR1 Betriebsart
10.5.108	SINT16	READ	0	0	1	108	SINT16	1	MKR1 Soll_Raum
10.5.109	SINT16	READ	0	0	1	109	SINT16	1	MKR1 Soll
10.5.110	UINT16	READ	0	0	1	110	UINT16	1	MKR1 Soll aktiv
10.5.111	SINT16	READ	0	0	1	111	SINT16	1	MKR1 Kuehlen RLT min.
10.5.112	UINT16	READ	0	0	1	112	UINT16	1	MKR2 Betriebsart
10.5.113	SINT16	READ	0	0	1	113	SINT16	1	MKR2 Soll_Raum
10.5.114	SINT16	READ	0	0	1	114	SINT16	1	MKR2 Soll
10.5.115	UINT16	READ	0	0	1	115	UINT16	1	MKR2 Soll aktiv
10.5.116	SINT16	READ	0	0	1	116	SINT16	1	MKR2 Kuehlen RLT min.
10.5.117	UINT16	READ	0	0	1	117	UINT16	1	PV Anforderung
10.5.118	SINT16	READ	0	0	1	118	SINT16	1	Hz Offset (PV)
10.5.119	SINT16	READ	0	0	1	119	SINT16	1	Kue Offset (PV)
10.5.120	SINT16	READ	0	0	1	120	SINT16	1	MK1 Hz Offset (PV)



KNX group	KNX datatype	KNX direction	KNX delta	KNX interval	Unit ID	MODBUS register	MODBUS datatype	Factor	Comment
10.5.105	SINT16	READ	0	0	1	105	SINT16	1	WW Normaltemperatur
10.5.106	SINT16	READ	0	0	1	106	SINT16	1	WW Minimaltemperatur
10.5.107	UINT16	READ	0	0	1	107	UINT16	1	MKR1 Betriebsart
10.5.108	SINT16	READ	0	0	1	108	SINT16	1	MKR1 Soll_Raum
10.5.109	SINT16	READ	0	0	1	109	SINT16	1	MKR1 Soll
10.5.110	UINT16	READ	0	0	1	110	UINT16	1	MKR1 Soll aktiv
10.5.111	SINT16	READ	0	0	1	111	SINT16	1	MKR1 Kuehlen RLT min.
10.5.112	UINT16	READ	0	0	1	112	UINT16	1	MKR2 Betriebsart
10.5.113	SINT16	READ	0	0	1	113	SINT16	1	MKR2 Soll_Raum
10.5.114	SINT16	READ	0	0	1	114	SINT16	1	MKR2 Soll
10.5.115	UINT16	READ	0	0	1	115	UINT16	1	MKR2 Soll aktiv
10.5.116	SINT16	READ	0	0	1	116	SINT16	1	MKR2 Kuehlen RLT min.
10.5.117	UINT16	READ	0	0	1	117	UINT16	1	PV Anforderung
10.5.118	SINT16	READ	0	0	1	118	SINT16	1	Hz Offset (PV)
10.5.119	SINT16	READ	0	0	1	119	SINT16	1	Kue Offset (PV)
10.5.120	SINT16	READ	0	0	1	120	SINT16	1	MK1 Hz Offset (PV)
10.5.121	SINT16	READ	0	0	1	121	SINT16	1	MK1 Kue Offset (PV)
10.5.122	SINT16	READ	0	0	1	122	SINT16	1	MK2 Hz Offset (PV)
10.5.123	SINT16	READ	0	0	1	123	SINT16	1	MK2 Kue Offset (PV)
10.5.124	SINT16	READ	0	0	1	124	SINT16	1	WW Normaltemp_Gw_max
10.5.125	UINT16	READ	0	0	1	125	UINT16	1	Leistungsvorgabe (W)
10.5.126	SINT16	READ	0	0	1	126	SINT16	1	Leistungsvorgabe (0-100%)
10.5.127	UINT16	READ	0	0	1	127	UINT16	1	Ext. Anf
10.5.128	UINT16	READ	0	0	1	128	UINT16	1	Entstören
10.5.129	SINT16	READ	0	0	1	129	SINT16	1	Aussentemperatur Wert
10.5.130	UINT16	READ	0	0	1	130	UINT16	1	Aussentemperatur aktiv
10.5.131	SINT16	READ	0	0	1	131	SINT16	1	Puffertemperatur Wert
10.5.132	UINT16	READ	0	0	1	132	UINT16	1	Puffertemperatur aktiv
10.5.133	SINT16	READ	0	0	1	133	SINT16	1	Brauchwassertemp. Wert
10.5.134	UINT16	READ	0	0	1	134	UINT16	1	Brauchwassertemp. aktiv

A detail for the KNX data point 10.5.10:



MODBUS/RTU

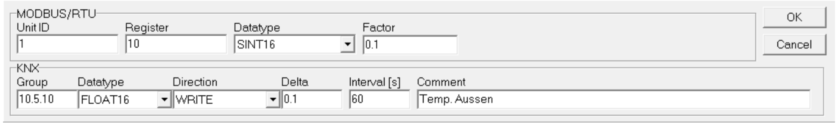
Unit ID	Register	Datatype	Factor
1	10	SINT16	1

KNX

Group	Datatype	Direction	Delta	Interval [s]	Comment
10.5.10	SINT16	WRITE	1	60	Temp. Aussen

The following has been configured:

- From the heat pump, a SINT16 holding register is read from the MDOBUS/RTU address (base=1) 10 and forwarded directly to the KNX with the group address 10.5.10.
- The cyclically read MODBUS value is transmitted to the KNX under the following conditions: 1. If 60 seconds have passed since the last KNX transmission or the value last read by MODBUS differs by +/-1 from the old value stored in the gateway.
- The parameter Factor 1 defines that the value is multiplied by 1. For example, the outside temperature is 25.7°C and in the MODBUS holding register the outside temperature is multiplied by a factor of 10, ie 257. This value is now output directly to the KNX. If you want to display the actual temperature on the KNX, you have to configure the following: Factor 0.1. Then 257 in the MODBUS register on the KNX becomes 25.7. For this purpose, the KNX data type SINT16 is no longer suitable, but the standard KNX data type FLOAT16. This data type can be processed by most KNX devices.



MODBUS/RTU

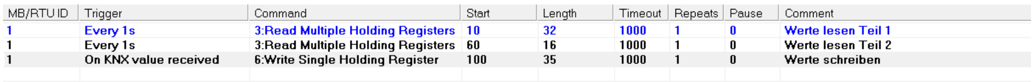
Unit ID	Register	Datatype	Factor
1	10	SINT16	0.1

KNX

Group	Datatype	Direction	Delta	Interval [s]	Comment
10.5.10	FLOAT16	WRITE	0.1	60	Temp. Aussen

IMPORTANT NOTE: The Gateway can only buffer 32 KNX telegrams internally. Thus, after a restart of the gateway, if all MODBUS values are refreshed, not all MODBUS values are output on the KNX immediately. However, this is negligible, as these values are cyclically repeated anyway due to the delta and interval settings.

Here is the mapping of the MODBUS RTU communication telegrams:



MB/RTU ID	Trigger	Command	Start	Length	Timeout	Repeats	Pause	Comment
1	Every 1s	3:Read Multiple Holding Registers	10	32	1000	1	0	Werte lesen Teil 1
1	Every 1s	3:Read Multiple Holding Registers	60	16	1000	1	0	Werte lesen Teil 2
1	On KNX value received	6:Write Single Holding Register	100	35	1000	1	0	Werte schreiben

The first entry defines that the MODBUS master reads every second 32 holding registers from the MODBUS/RTU slave with the UnitID 1 starting at the holding register (base = 1) 10. These 32 holding registers are assigned to the corresponding KNX data points via the above configuration.

The second entry defines that the MODBUS master reads every second 16 holding registers from the MODBUS/RTU slave with the UnitID 1 starting at the holding register (base = 1) 60. These 32 holding registers are assigned to the corresponding KNX data points via the above configuration.

The third entry defines what happens when a KNX group address is received whose configured MODBUS register falls within the range of holding registers (base=1) 100 to 134 (35 registers). Then a "Write Single Holding Register" protocol with the current value of the received KNX group address is sent to the MODBUS/RTU device with the UnitID 1.

9 Functional Description

9.1 MODBUS data types, memory and common pitfalls

In general, the MODBUS uses only 16 bit wide registers. As long as you only use data types that fit in a 16-bit register, the assignment is easy. But as soon as you use data types that occupy two or more MODBUS registers, e.g. UINT32, you can put these numbers in a variety of ways in the MODBUS register.

We show a simple example. We want to store the 32-bit unsigned integer value as a hexadecimal number 0x12345678 in the MODBUS Holding Register, starting from the start index 4x00010. The assignment can now be made in two different ways:

MODBUS Register	Storing an UINT32 data type
4x00010 I:9	The upper word of the 32-bit value 0x12345678 is stored in the first 16-bit MODBUS register. This means that the value 0x1234 is stored here.
4x00011 I:10	The lower word of the 32-bit value 0x12345678 is stored in the second 16-bit MODBUS register. This means that the value 0x5678 is stored here.

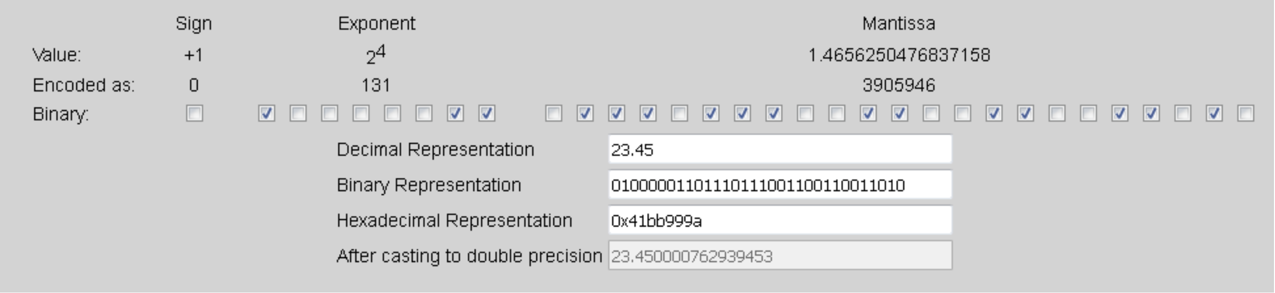
But this is only one way to store the higher word in the first MODBUS register. With the same right, we can define to store the lower word in the first MODBUS register, followed by the higher word.

The result then looks like this:

MODBUS Register	Storing an UINT32 data type
4x00010 I:9	The lower word of the 32-bit value 0x12345678 is stored in the first 16-bit MODBUS register. This means that the value 0x5678 is stored here.
4x00011 I:10	The upper word of the 32-bit value 0x12345678 is stored in the second 16-bit MODBUS register. This means that the value 0x1234 is stored here.

More complex is depositing a FLOAT32 value into two consecutive holding registers. We use a room temperature, e.g. 23.45 ° C as a value that we want to put in two registers.

First, we need to convert the value to a valid IEE754 floating-point number. For this we use a perfect website (http://www.h-schmidt.net/FloatConverter/IEEE754.html):



	Sign	Exponent	Mantissa
Value:	+1	2^4	1.4656250476837158
Encoded as:	0	131	3905946
Binary:	☐	☑ ☐ ☐ ☐ ☐ ☐ ☑ ☑	☐ ☑ ☑ ☑ ☐ ☑ ☑ ☑ ☐ ☐ ☑ ☑ ☐ ☐ ☑ ☑ ☐ ☐ ☑ ☑ ☐ ☑ ☐

Decimal Representation: 23.45
Binary Representation: 01000001101110111001100110011010
Hexadecimal Representation: 0x41bb999a
After casting to double precision: 23.450000762939453

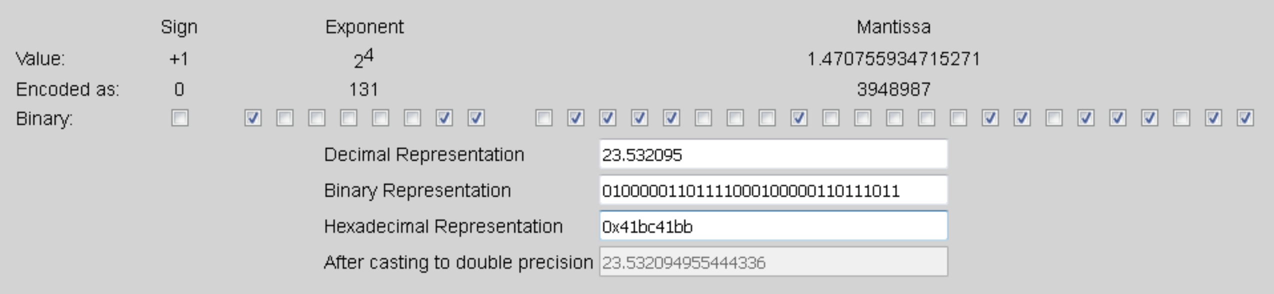
We enter the value 23.45 and get a representation of the floating point number as a 32 bit hexadecimal number. It is the number 0x41BB999A. Now we store this value in the same way as we saved the UINT32 value:

MODBUS Register	Storing of FLOAT32 data types
4x00010 I:9	The upper word of the 32-bit value 0x41BB999A is stored in the first 16-bit MODBUS register. This means that the value 0x41BB is stored here.
4x00011 I:10	The lower word of the 32-bit value 0x41BB999A is stored in the first 16-bit MODBUS register. This means that the value 0x999A is stored here.

But we can also use the reverse notation:

MODBUS Register	Storing of FLOAT32 data types
4x00010 I:9	The lower word of the 32-bit value 0x41BB999A is stored in the first 16-bit MODBUS register. This means that the value 0x999A is stored here.
4x00011 I:10	The upper word of the 32-bit value 0x41BB999A is stored in the first 16-bit MODBUS register. This means that the value 0x41BB is stored here.

Now we show a widespread trap concerning the storing and reading of more than one MODBUS register. We use another floating-point value. As a hexadecimal number, it is the number 0x41BC41BB. Again we use the online converter:



	Sign	Exponent	Mantissa
Value:	+1	2^4	1.470755934715271
Encoded as:	0	131	3948987
Binary:	☐	☑ ☐ ☐ ☐ ☐ ☐ ☑ ☑	☐ ☑ ☑ ☑ ☑ ☐ ☐ ☐ ☑ ☐ ☐ ☐ ☐ ☐ ☑ ☑ ☐ ☑ ☑ ☑ ☐ ☑ ☑

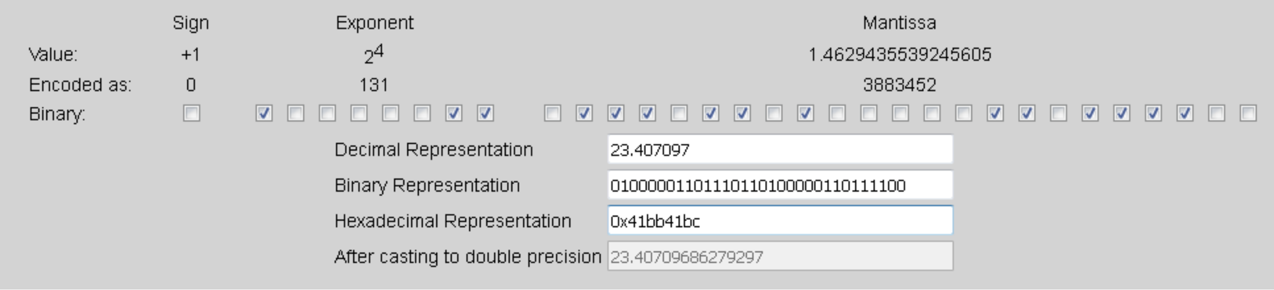
Decimal Representation	23.532095
Binary Representation	01000001101111000100000110111011
Hexadecimal Representation	0x41bc41bb
After casting to double precision	23.532094955444336

You notice, it's the floating-point number 23.532095.

Now we store this number with the upper word first in two registers:

MODBUS Register	Storing of FLOAT32 data types
4x00010 I:9 HIGH WORD	The upper word of the 32-bit value 0x41BC41BB is stored in the first 16-bit MODBUS register. This means that the value 0x41BC is stored here.
4x00011 I:10 LOW WORD	The lower word of the 32-bit value 0x41BC41BB is stored in the first 16-bit MODBUS register. This means that the value 0x41BB is stored here.

Now we make a very serious error in the reading procedure. We read both registers and build the hexadecimal value in our host software in the wrong word order. First the lower word, then the upper word. The result is also a 32 bit hexadecimal number: 0x41BB41BC instead of the correct number 0x41BC41BB. Now we convert this hexadecimal number into an IEE754 floating-point number:



	Sign	Exponent	Mantissa
Value:	+1	2^4	1.4629435539245605
Encoded as:	0	131	3883452
Binary:	☐	☑☐☐☐☐☐☑☑	☐☑☑☑☐☑☑☐☑☐☐☐☐☐☑☑☐☑☑☑☑☐☐

Decimal Representation	23.407097
Binary Representation	01000001101110110100000110111100
Hexadecimal Representation	0x41bb41bc
After casting to double precision	23.40709686279297

The result is 23.407097. That's not far from the original number 23.532095! Thus, this massive software bug can live undetected for a long time. Only if the exchanged value generates physically meaningless values, this error is recognized!

10 Specifications

10.1 Dimensions

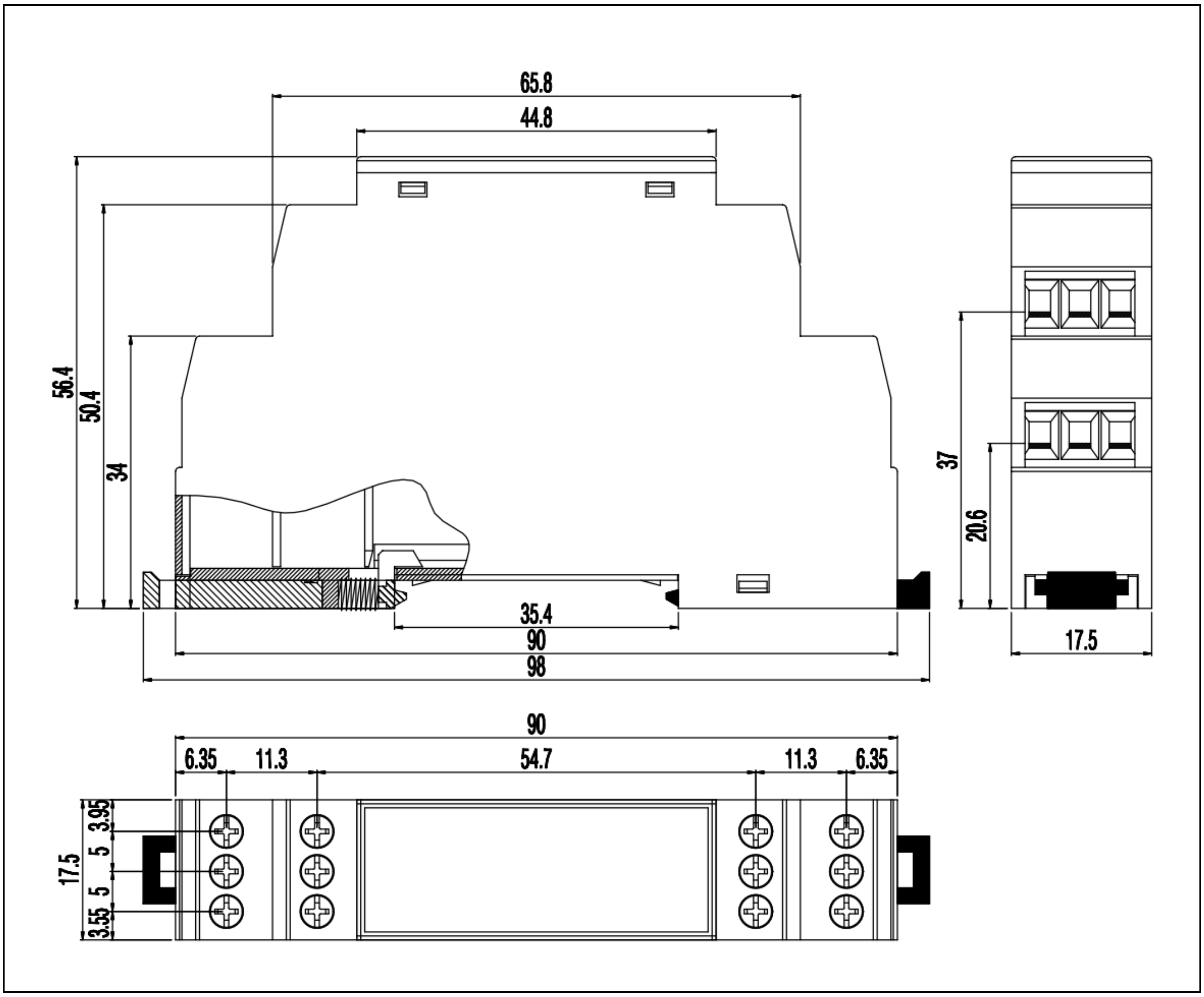


Illustration: dimension illustration in mm

Dimensions	
Enclosure dimensions L x W x H (mm)	17,5 x 90 x 58
Weight	55 g
Colour	Grey RAL7035
Material	PA - UL 94 V0
Protection class	IP20 based on DIN 40050/EN 60529

Table: Data of enclosure

10.2 3D Drawing

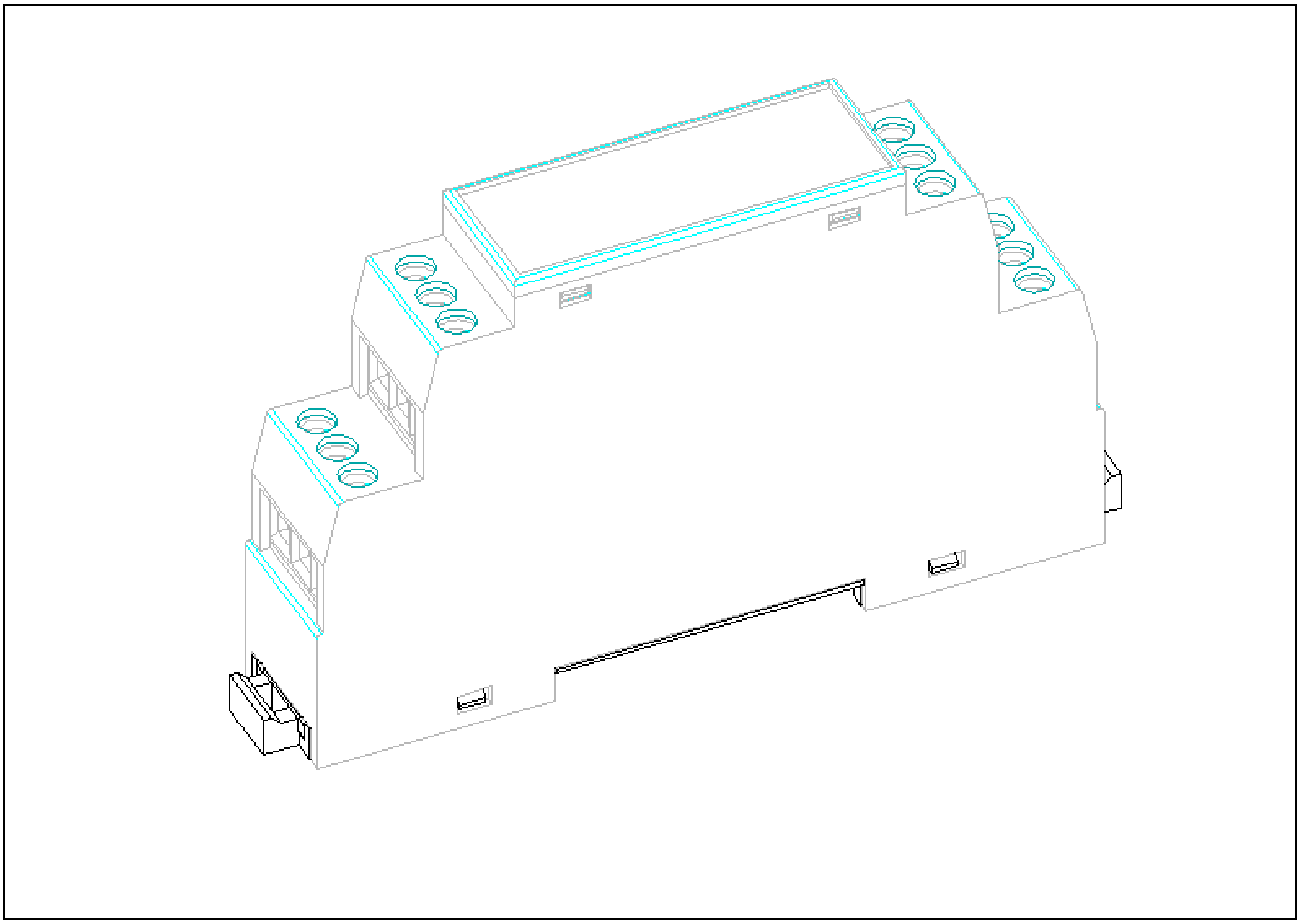


Illustration: Dimension illustration in 3D