RESI-xxx-SIO
RESI-xxx-ETH

Our series of intelligent IO modules based on MODBUS protocol and ASCII text protocol for building automation and industrial automation.

Content

1 Our portfolio

We offer the following IO products:

1.1 ULTRA SLIM serial IO modules

Our ultra slim IO modules are extreme small modules and offer various IOs, always in combination with a RS232 and RS485 interface.
The dimension of those IO modules is very slim:

- Only 17.5x90x58mm (WxHxD) in size

Those IO modules offer the following protocols:

- a MODBUS/RTU slave protocol
- a simple ASCII text protocol

The modules support the following baud rates:

- from 300bd up to 256000bd
- none, even and odd parity
- one or two stop bits.

All our modules are designed for use with 12 to 48Vdc power supplies, so they offer a broad range of applications. The serial interface is always galvanically insulated from the IOs on the module. The modules are designed for mounting on a DIN EN50022 rail.

Figure: Sample of an ultra slim IO module with serial interface

1.2 ULTRA SLIM Ethernet IO modules

Our ultra slim IO modules are extreme small modules and offer various IOs, always in combination with an Ethernet interface.
The dimension of those IO modules is very slim:

- Only 35.8x90x58mm (WxHxD) in size

Those IO modules offer various protocols:

- MODBUS/TCP server protocol
- MODBUS/RTU slave protocol via Ethernet
- simple ASCII text protocol via socket

The Ethernet interface offers

- RJ45 interface mit 10MBit/100MBit
- support of AUTO - MDIX

All our modules are designed for use with 12 to 48Vdc power supplies, so they offer a broad range of applications. The serial interface is always galvanically insulated from the IOs on the module. The modules are designed for mounting on a DIN EN50022 rail.

Figure: Sample of an ultra slim IO module with Ethernet interface

1.3 BIGIO serial IO modules

Our BIGIO modules are extreme compact modules with many IOs, always in combination with a serial RS485 interface. We offer two different housings depending on the amount of IOs implemented in the IO module.
The dimension of the XT8 IO modules is:

- 142,3x110x62mm (WxHxD) in size

The dimension of the XT12 IO modules is:

- 213x110x62mm (WxHxD) in size

Those IO modules offer various protocols:

- MODBUS/TCP server protocol
- MODBUS/RTU slave protocol via Ethernet
- simple ASCII text protocol via socket

The Ethernet interface offers

- RJ45 interface mit 10MBit/100MBit
- support of AUTO - MDIX

All our modules are designed for use with 12 to 48Vdc power supplies, so they offer a broad range of applications. The serial interface is always galvanically insulated from the IOs on the module. The modules are designed for mounting on a DIN EN50022 rail. but the modules offer also a wall mounting option.

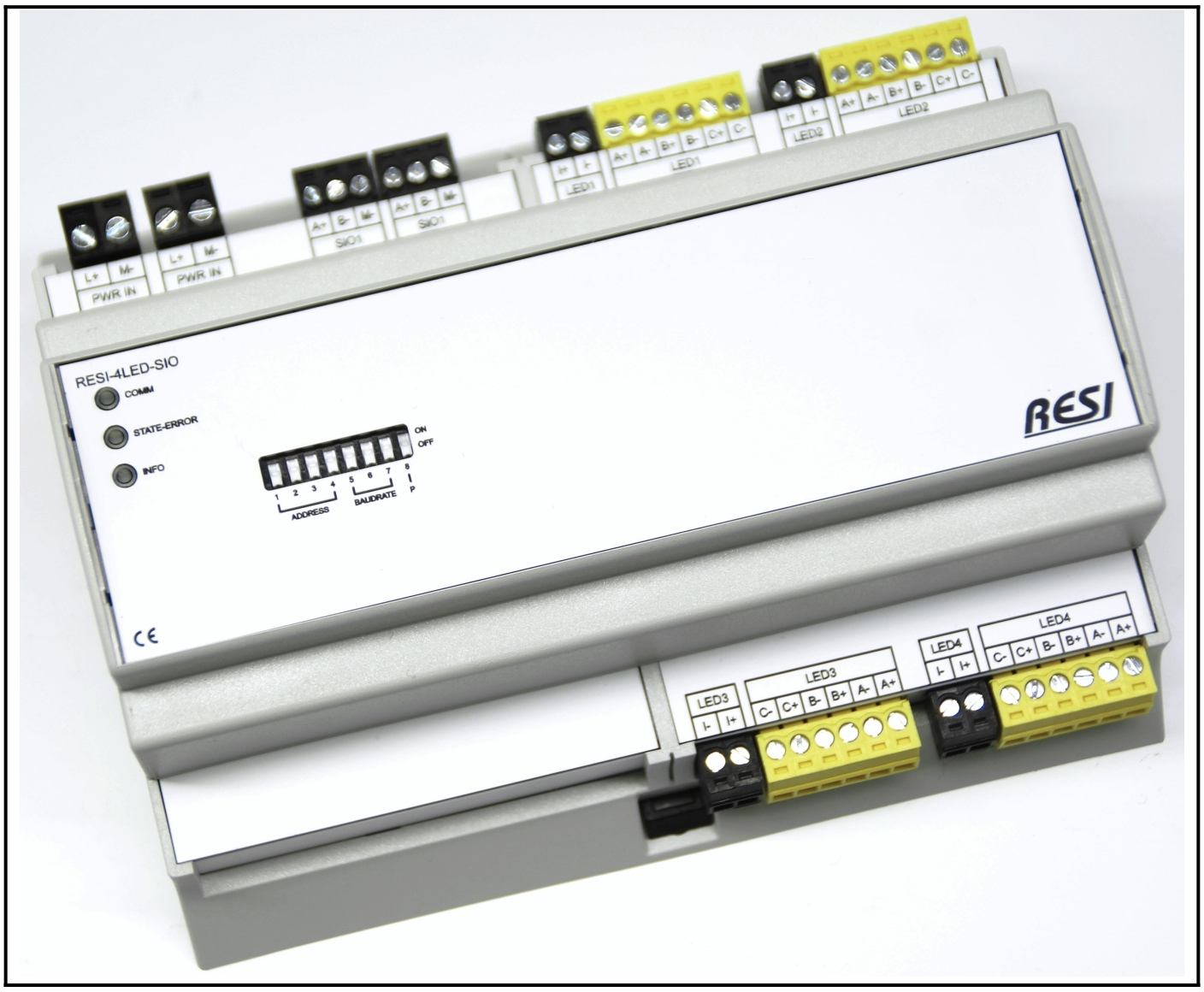


Figure: Sample of a XT8 IO module with serial interface

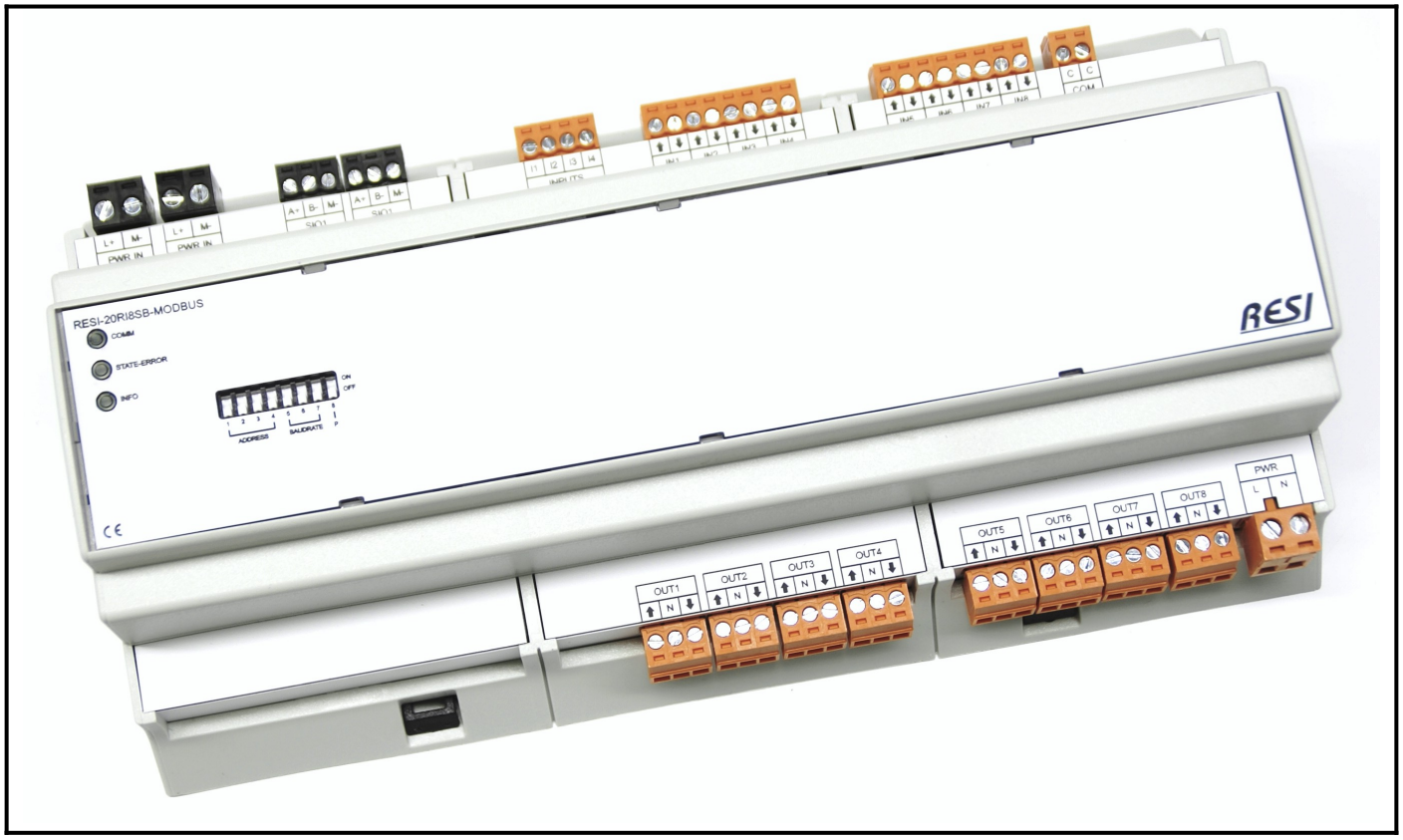


Figure: Sample of a XT12 IO module with serial interface

2 Declaration of conformity

2.1 CE

All products have passed the CE tests for environmental specifications when shielded cables are used for external wiring. We recommend the use of shielded cables.

2.2 Safety instructions

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 data sheets, in our manuals, in our catalogs 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 fulfill 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!

3 Our Portfolio

Here you find a list of all available IO modules:

3.1 Digital IO modules

PRODUCT	DESCRIPTION	HOUSING TYPE
RESI-2RI-SIO	2xdigital inputs for 10-250Vac/dc signals	ULTRA SLIM 17.5mm
RESI-2RI-ETH	2xdigital inputs for 10-250Vac/dc signals	ULTRA SLIM 35.8mm
RESI-4DI-SIO	4xdigital inputs for 12-48Vdc signals	ULTRA SLIM 17.5mm
RESI-4DI-ETH	4xdigital inputs for 12-48Vdc signals	ULTRA SLIM 35.8mm
RESI-14RI-SIO	14xdigital inputs for 10-250Vac/dc signals	BIGIO XT8 142.3mm
RESI-48RI-SIO	48xdigital inputs for 10-250Vac/dc signals	BIGIO XT12 213mm
RESI-32DI-SIO	32xdigital inputs for 12-48Vdc signals	BIGIO XT8 142.3mm
RESI-64DI-SIO	64xdigital inputs for 12-48Vdc signals	BIGIO XT12 213mm
RESI-1RO-SIO	1xrelay output with max. 230Vac,30Vdc, 8A and NO+NC contacts	ULTRA SLIM 17.5mm
RESI-1RO-ETH	1xrelay output with max. 230Vac,30Vdc, 8A and NO+NC contacts	ULTRA SLIM 35.8mm
RESI-2RO-SIO	2xrelay output with max. 230Vac,30Vdc, 8A and NO contacts	ULTRA SLIM 17.5mm
RESI-2RO-ETH	2xrelay output with max. 230Vac,30Vdc, 8A and NO contacts	ULTRA SLIM 35.8mm
RESI-2SSR-1A-SIO	2xsolid state relay outputs with max. 600Vac, 600Vdc, 1A and NO contacts	ULTRA SLIM 17.5mm
RESI-2SSR-1A-ETH	2xsolid state relay outputs with max. 600Vac, 600Vdc, 1A and NO contacts	ULTRA SLIM 35.8mm
RESI-2SSR-6A-SIO	2xsolid state relay outputs with max. 60Vac, 60Vdc, 6A and NO contacts	ULTRA SLIM 17.5mm
RESI-2SSR-6A-ETH	2xsolid state relay outputs with max. 60Vac, 60Vdc, 6A and NO contacts	ULTRA SLIM 35.8mm
RESI-4DO-SIO	4xdigital outputs with max. 2-32Vdc, 300mA	ULTRA SLIM 17.5mm
RESI-4DO-ETH	4xdigital outputs with max. 2-32Vdc, 300mA	ULTRA SLIM 35.8mm
RESI-8CO-SIO	8xrelay output with max. 230Vac,30Vdc, 8A and NO+NC contacts	BIGIO XT8 142.3mm
RESI-8COBI-SIO	8xbistable relay output with max. 230Vac,30Vdc, 8A and NO+NC contacts	BIGIO XT8 142.3mm
RESI-10SSR-1A-SIO	10xsolid state relay outputs with max. 600Vac, 600Vdc, 1A and NO contacts	BIGIO XT8 142.3mm
RESI-10SSR-6A-SIO	10xsolid state relay outputs with max. 60Vac, 60Vdc, 6A and NO contacts	BIGIO XT8 142.3mm
RESI-30DO-SIO	30xdigital outputs with max. 2-32Vdc, 300mA	BIGIO XT8 142.3mm
RESI-60DO-SIO	60xdigital outputs with max. 2-32Vdc, 300mA	BIGIO XT12 213mm

PRODUCT	DESCRIPTION	HOUSING TYPE
RESI-S16DI8RO-SIO	16xdigital inputs for 12-48Vdc signals 8xbistable power relais max. 250Vac, 16A, 200µF	BIGIO XT8 142.3mm
RESI-S8RO-SIO	8xbistable power relais max. 250Vac, 16A, 200µF	BIGIO XT8 142.3mm
RESI-16RI8RO-SIO	16xdigital inputs for 10-250Vac/dc signals 8xbistable power relais max. 250Vac, 16A, 200µF	BIGIO XT12 213mm
RESI-8RO-SIO	8xbistable power relais max. 250Vac, 16A, 200µF	BIGIO XT12 213mm
RESI-10RI4SB-SIO	10xdigital inputs for 10-250Vac/dc signals 8xrelais max. 250Vac, 6A, $AgSNO_2$ contacts	BIGIO XT8 142.3mm
RESI-4SB-SIO	8xrelais max. 250Vac, 6A, $AgSNO_2$ contacts	BIGIO XT8 142.3mm
RESI-20RI8SB-SIO	20xdigital inputs for 10-250Vac/dc signals 16xrelais max. 250Vac, 6A, $AgSNO_2$ contacts	BIGIO XT12 213mm
RESI-8SB-SIO	16xrelais max. 250Vac, 6A, $AgSNO_2$ contacts	BIGIO XT12 213mm

3.2 Analog IO modules

PRODUCT	DESCRIPTION	HOUSING TYPE
RESI-4AIU-SIO	4xanalog inputs for -10..+10Vdc signals, 16 bit, ±0.1%	ULTRA SLIM 17.5mm
RESI-4AIU-ETH	4xanalog inputs for -10..+10Vdc signals, 16 bit, ±0.1%	ULTRA SLIM 35.8mm
RESI-12AIU-SIO	12xanalog inputs for -10..+10Vdc signals, 16 bit, ±0.1%	BIGIO XT8 142.3mm
RESI-4AOU-SIO	4xanalog outputs for -10..+10Vdc signals, 16 bit, ±0.1%	ULTRA SLIM 17.5mm
RESI-4AOU-ETH	4xanalog outputs for -10..+10Vdc signals, 16 bit, ±0.1%	ULTRA SLIM 35.8mm
RESI-12AOU-SIO	12xanalog outputs for -10..+10Vdc signals, 16 bit, ±0.1%	BIGIO XT8 142.3mm
RESI-2AIU2AOU-SIO	2xanalog inputs for 0..+10Vdc signals, 12 bit, ±0.5% 2xanalog outputs for 0..+10Vdc signals, 12 bit, ±0.5%	ULTRA SLIM 17.5mm
RESI-2AIU2AOU-ETH	2xanalog inputs for 0..+10Vdc signals, 12 bit, ±0.5% 2xanalog outputs for 0..+10Vdc signals, 12 bit, ±0.5%	ULTRA SLIM 35.8mm

3.3 Temperature IO modules

PRODUCT	DESCRIPTION	HOUSING TYPE
RESI-2RTD-SIO	2xinputs for RTD temperature sensors (PT100, PT1000, NI1000, NI120,…) ±0.1% 2-wire, 3-wire and 4 wire connection	ULTRA SLIM 17.5mm
RESI-2RTD-ETH	2xinputs for RTD temperature sensors (PT100, PT1000, NI1000, NI120,…) ±0.1% 2-wire, 3-wire and 4 wire connection	ULTRA SLIM 35.8mm
RESI-8RTD-SIO	2xinputs for RTD temperature sensors (PT100, PT1000, NI1000, NI120,…) ±0.1% 2-wire, 3-wire and 4 wire connection	BIGIO XT8 142.3mm
RESI-8RTD2-SIO	2xinputs for RTD temperature sensors (PT100, PT1000, NI1000, NI120,…) ±0.1% 2-wire connection	BIGIO XT8 142.3mm

3.4 Special IO modules

PRODUCT	DESCRIPTION	HOUSING TYPE
RESI-1LED-SIO	1x3 PWM outputs for LED stripes <60Vdc, <5A per PWM channel	ULTRA SLIM 17.5mm
RESI-1LED-ETH	1x3 PWM outputs for LED stripes <60Vdc, <5A per PWM channel	ULTRA SLIM 35.8mm
RESI-4LED-SIO	4x3 PWM outputs for LED stripes <60Vdc, <5A per PWM channel	BIGIO XT8 142.3mm
RESI-1S0-SIO	1xS0 impulse input for smart meter with S0 interface	ULTRA SLIM 17.5mm
RESI-1S0-ETH	1xS0 impulse input for smart meter with S0 interface	ULTRA SLIM 35.8mm
RESI-2S0-SIO	2xS0 impulse input for smart meter with S0 interface	ULTRA SLIM 17.5mm
RESI-2S0-ETH	2xS0 impulse input for smart meter with S0 interface	ULTRA SLIM 35.8mm
RESI-1EGYDCS-SIO	1xDC metering with external shunt, DC voltage: 0..100Vdc, max. 255A shunt	ULTRA SLIM 17.5mm
RESI-1EGYDCS-ETH	1xDC metering with external shunt, DC voltage: 0..100Vdc, max. 255A shunt	ULTRA SLIM 35.8mm
RESI-1EGYDC-SIO	1xDC metering with external hall sensor, DC voltage: 0..100Vdc, max. 80A	ULTRA SLIM 17.5mm
RESI-1EGYDC-ETH	1xDC metering with external hall sensor, DC voltage: 0..100Vdc, max. 80A	ULTRA SLIM 35.8mm

3.5 MBUS IO modules

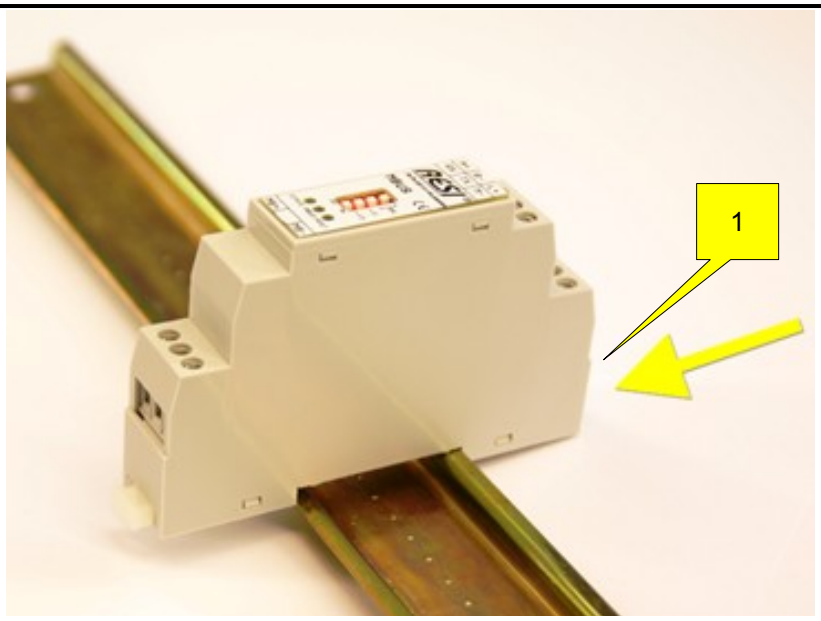
PRODUCT	DESCRIPTION	HOUSING TYPE
RESI-MBUS2-SIO	MBUS master to read data from 2 smart meter with MBUS interface	ULTRA SLIM 17.5mm
RESI-MBUS8SIO	MBUS master to read data from 8 smart meter with MBUS interface	ULTRA SLIM 17.5mm
RESI-MBUS24-SIO	MBUS master to read data from 24 smart meter with MBUS interface	ULTRA SLIM 17.5mm
RESI-MBUS48-SIO	MBUS master to read data from 48 smart meter with MBUS interface	ULTRA SLIM 17.5mm
RESI-MBUS2-ETH	MBUS master to read data from 2 smart meter with MBUS interface	ULTRA SLIM 35.8mm
RESI-MBUS8-ETH	MBUS master to read data from 8 smart meter with MBUS interface	ULTRA SLIM 35.8mm
RESI-MBUS24-ETH	MBUS master to read data from 24 smart meter with MBUS interface	ULTRA SLIM 35.8mm
RESI-MBUS48-ETH	MBUS master to read data from 48 smart meter with MBUS interface	ULTRA SLIM 35.8mm

4 Mounting

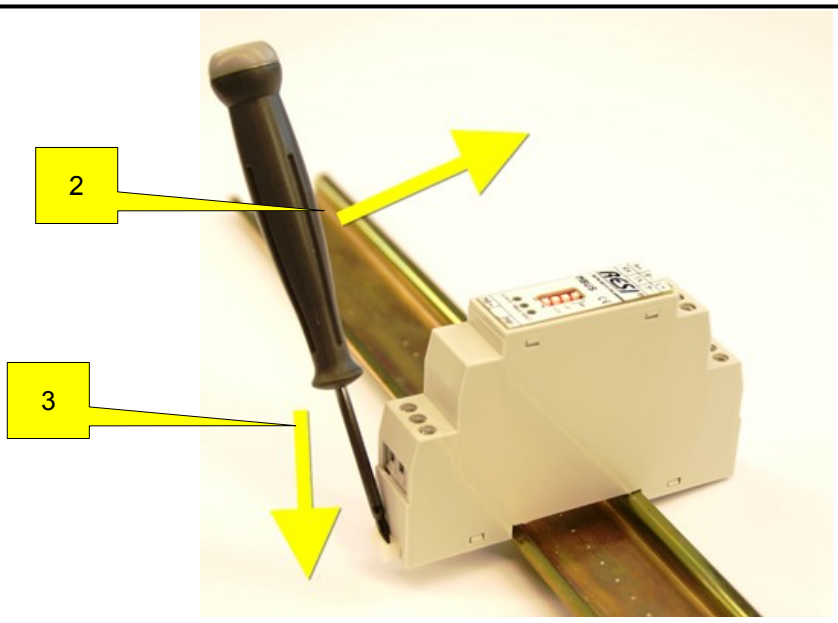
Our ULTRA SLIM IO module offer a 4 pin DIP switch for initial setup of the serial connection or the Ethernet connection. Our BIGIO modules offer a 8 pin DIP switch for initial setup.

4.1 Mounting for ULTRA SLIM IOs

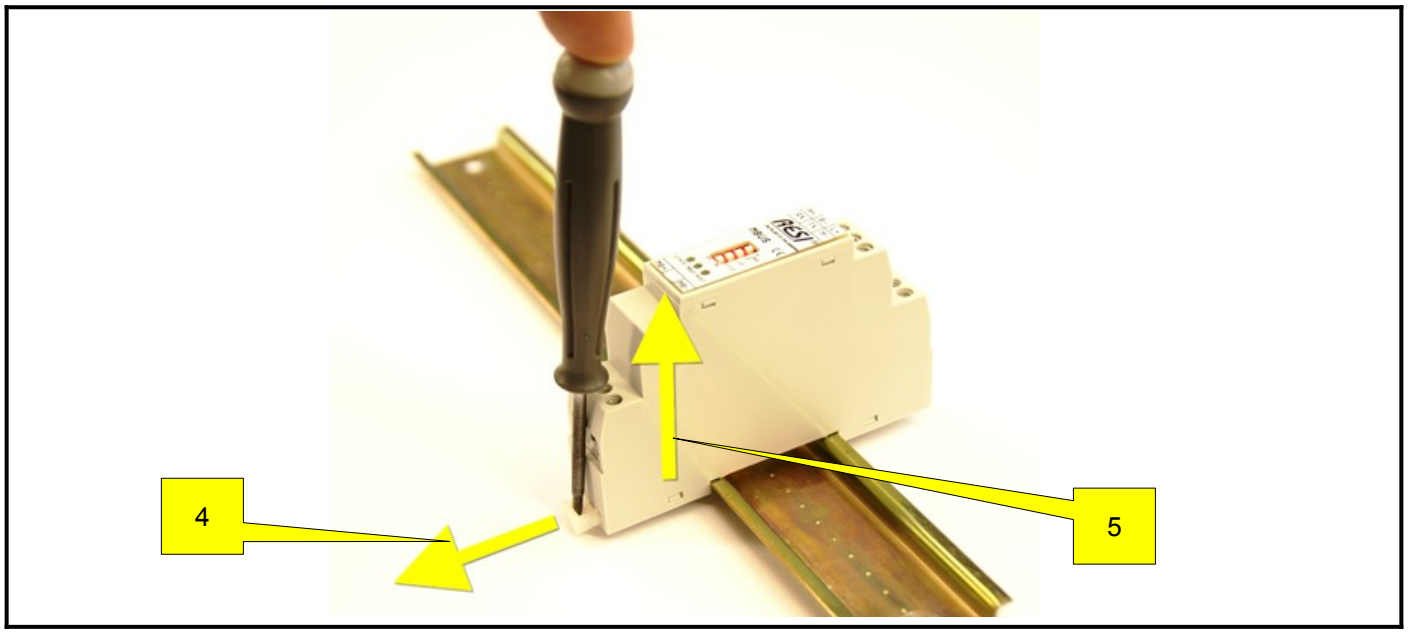
The Our IO modules are designed for mounting on a 35mm DIN-EN50022 rail.
At first, put the modules with the top side on the DIN rail (1).



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 module from the bar with while pulling it on the top side.



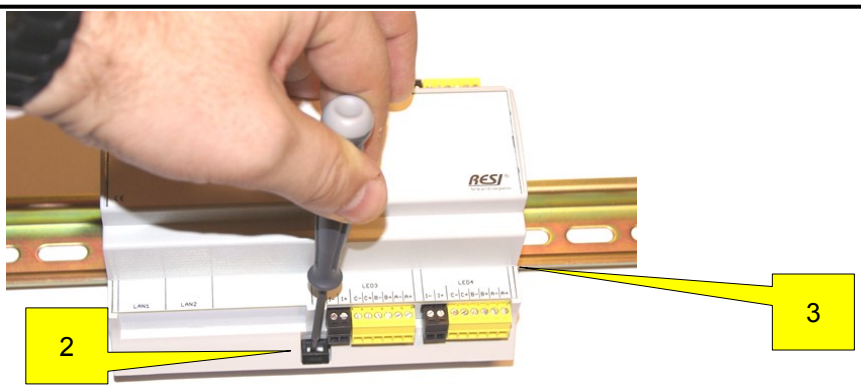
4.2 Mounting for BIG IOs XT8 or XT12

Our BIG IO modules are designed for mounting onto a 35mm DIN-EN50022 rail or for wall mounting. Please not, that in the following mounting description we use only symbolic photos of our IO modules.

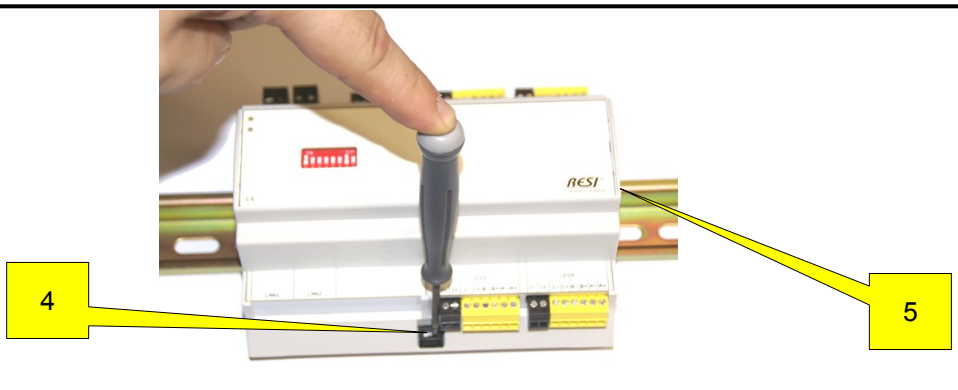
4.2.1 Mounting of a DIN EN50022 rail

First snap in the top part of the module into the DIN rail (1). The bottom part of the module is not snapped into the DIN rail at this moment.

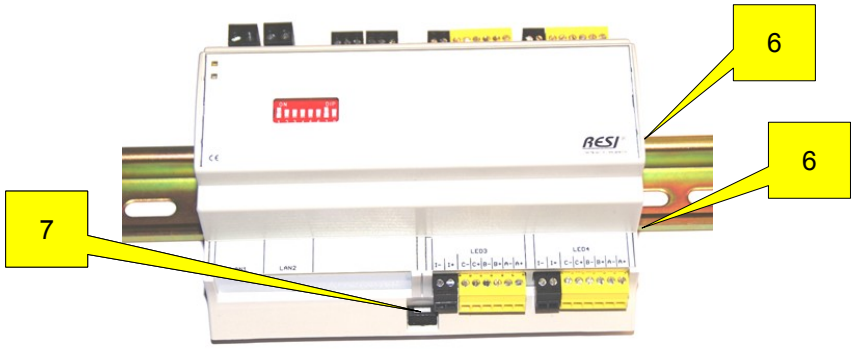
Then open the black hook with a screw driver (2). Now press the module with the opened hook onto the DIN rail until both sides of the module snap into the DIN rail (3). Release the screw driver now. The hook snaps into the DIN rail and the module is now mounted correctly onto the DIN rail.



To remove the module from the DIN rail, you must open the hook with a screwdriver first. (4). Afterwards tilt the bottom side of the module upwards with the open hook (5). Now remove the module slightly from the DIN rail with the top side, to completely hang out the module from the DIN rail.

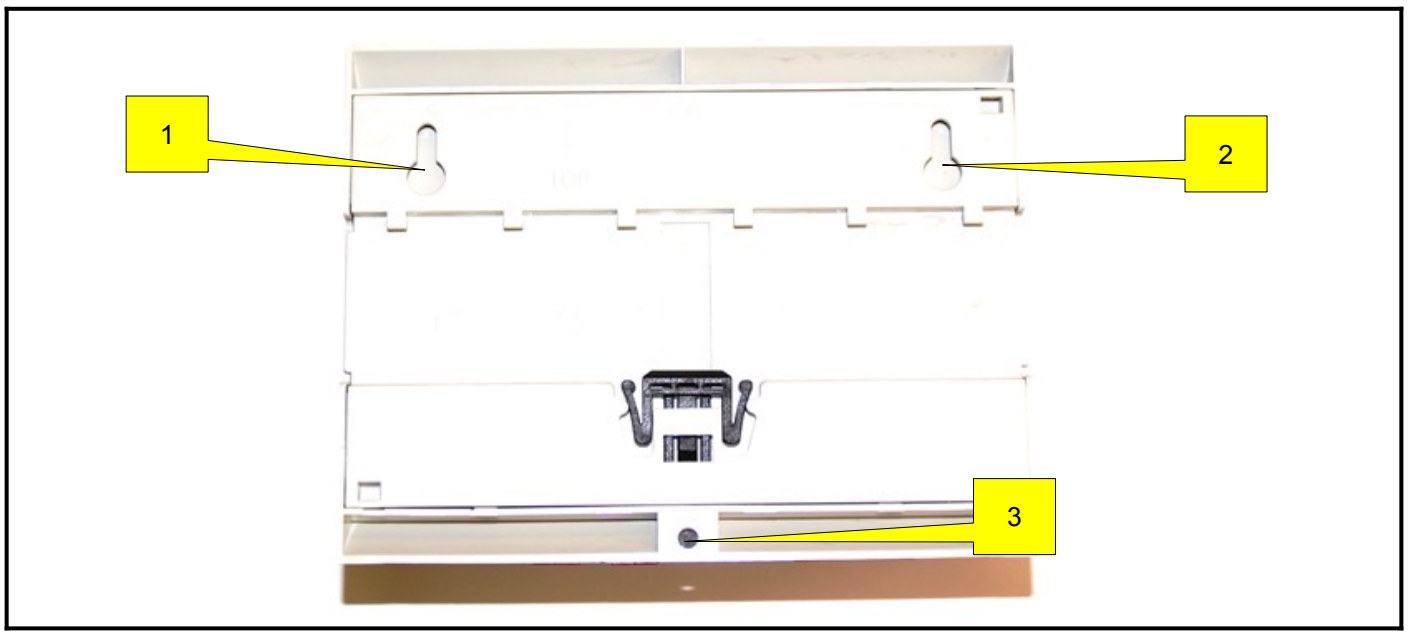


The module is correctly mounted, if the module has snapped into the DIN rail on both sides of the housing (6) and if the hook has snapped in too (7).

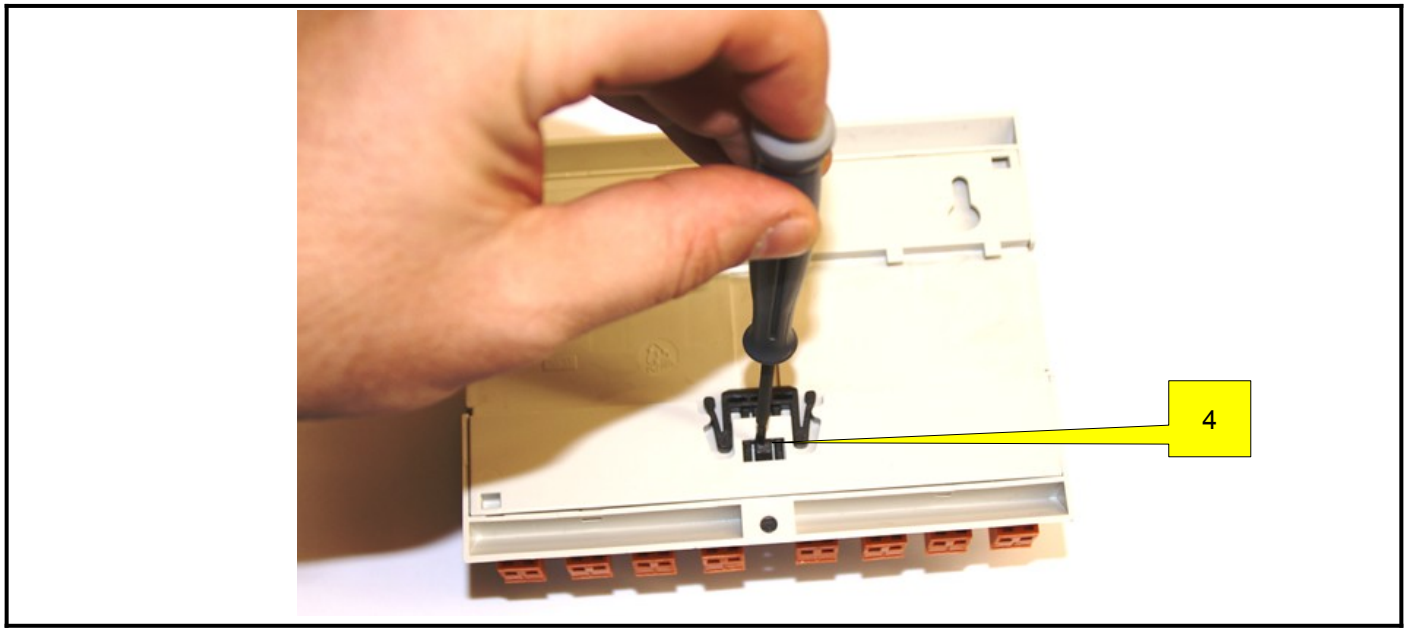


4.2.2 Mounting onto a wall

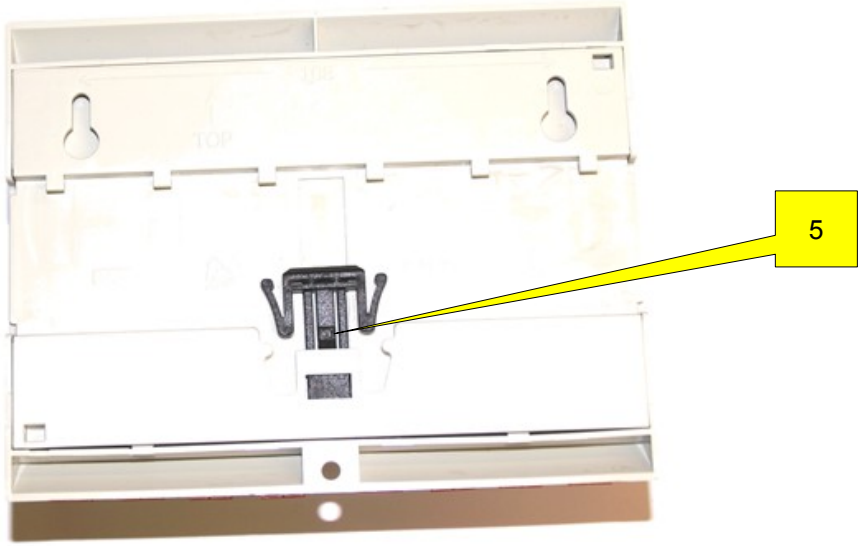
Our modules can also be mounted onto a wall. Turn over the module as shown in the picture below:

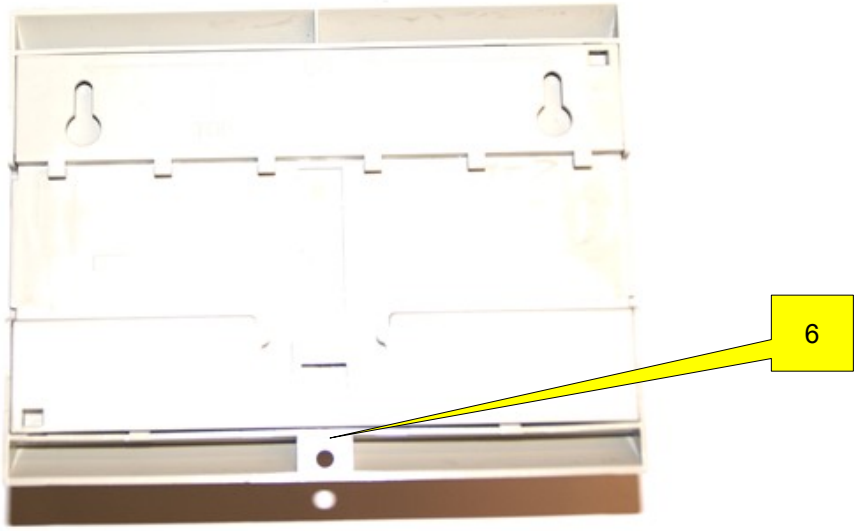


You will notice, that there are two holes for wall hooks or screws on the top side of the housing. (1) and (2). On the bottom side you will notice a small hole for a screw to fix the housing on the wall from the front (3). But first we have to remove the hook, which blocks the screw hole in the housing.



Press carefully the screwdriver onto the hook to open the lock (4) and pull back the hook to the inner side of the housing bottom to remove the hook. If the hook is not snapped into the housing, you can remove the hook by hand (5) and the screw hole for fixing the housing with a screen from the front side of the housing (6).





Now fix two wall hooks or screws into the wall. Use a center to center distance of 108mm between those two screws or hooks. The screw head must be bigger than 4mm but also smaller than 8mm to fix the housing onto the wall like a picture frame. If the housing in mounted onto the wall, you can fix the housing with a secure screw through the hole in the bottom housing from the front. But your screw must be smaller than 4mm to fit into this hole and the screw head must be bigger than 4mm to press the housing onto the wall.

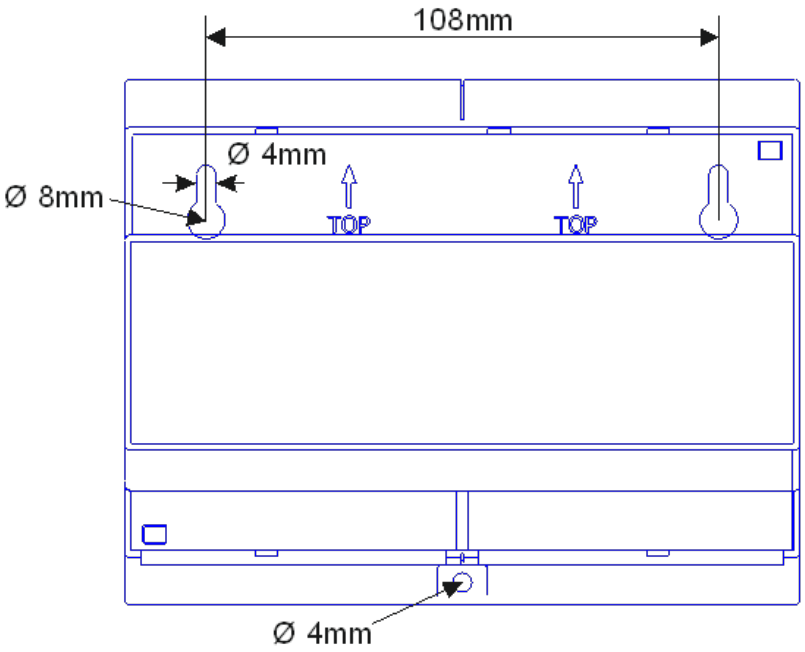


Illustration: Bottom view of the module with holes for XT8 wall mounting

5 General technical data

In this section you will find all technical data which is common to all IO modules. In the specific sections of the individual IO modules you will find only the differences and extensions to this standard description.

5.1 Basic technical data

Power supply	
Supply voltage	12-48 V = +/- 10%
Voltage LED indicator	Yes
Power consumption	see individual technical data for specific IO module

Serial interface	
(only for serial ULTRA SLIM IOs and BIG IOs)	
Protocols	MODBUS/RTU slave or ASCII text protocol
Type	RS232 or RS485 for ULTRA SLIM IOs
	RS485 for BIG IOs
Baud rate	300 to 256000bd
Data bits	8 bits
Parity	none, even or odd
Stop bits	1 or 2 bits
LED indicator	Yes

Ethernet interface	
(only for Ethernet ULTRA SLIM IOs)	
Protocols	MODBUS/TCP Server
	ASCII Text socket
	MODBUS/RTU over Ethernet
Type	Ethernet
Cable connection	via RJ 45 socket
LED indicator	Yes

General	
Storage temperature	-20...85 °C
Operating temperature	0...60 °C
Humidity	25...90% RH non-condensing
Protection class	IP20 (EN 60529)
Dimensions LxWxH	see section Dimension
Weight	see individual technical data for specific IO module
Installation	on DIN EN50022 rail for ULTRA SLIM IOs
	on DIN EN50022 rail and on wall for BIG IOs

Approvals	
CE conformity	Yes

5.2 Serial ULTRA SLIM IOS: basic terminals

The serial ULTRA SLIM IOs come in a housing with integrated clamps. All IO modules offer the following terminals:

L+, M-	Power supply:	
	L +:	12-48 V =
	M-:	mass
A, B, M-	RS485 ASCII or MODBUS/RTU interface	
	A +:	RS485 DATA + signal
	B-:	RS485 DATA signal
	M-:	RS485 ground signal
TX, RX, M-	RS232 ASCII or MODBUS/RTU interface	
	TX +:	RS232 transmit signal
	RX-:	RS232 receive Signal
	M-:	RS232 ground signal
Terminal type USLIM	Cable cross section:	max. 2.5 mm², max. 14AWG
	Screw:	M3
	Tightening torque:	max. 0.5Nm, max. 4.5 Lb-in

5.3 Ethernet ULTRA SLIM IOS: basic terminals

The Ethernet ULTRA SLIM IOs come in a housing with integrated clamps. All IO modules offer the following terminals:

L+, M-	Power supply:	
	L +:	12-48 V =
	M-:	mass
ETHERNET	RJ45 connector	
	Ethernet connection	10M/100Mbit adaptive
	supports AUTO–MDIX	
Terminal type USLIM	Cable cross section:	max. 2.5 mm², max. 14AWG
	Screw:	M3
	Tightening torque:	max. 0.5Nm, max. 4.5 Lb-in

5.4 Serial BIG IOS: basic terminals

The serial BIG IOs come in a housing with removable clamps. All IO modules offer the following terminals:

L+, M-	Power supply via two separated plug-in 2-pin terminal blocks.	
	For daisy chain IN and OUT power supply of many modules	
	Pin 1:	L+: 12-48 V=
	Pin 2:	M-: Ground
	Terminal type:	RM5
SIO1	RS485 ASCII or MODBUS/RTU serial interface IN	
	Pin 1:	A+: RS485 DATA+ signal
	Pin 2:	B-: RS485 DATA- signal
	Pin 3:	GND: RS485 ground signal
	Terminal type:	RM3.5
SIO2	RS485 ASCII or MODBUS/RTU serial interface OUT	
	Pin 1:	A+: RS485 DATA+ signal
	Pin 2:	B-: RS485 DATA- signal
	Pin 3:	GND: RS485 ground signal
	Terminal type:	RM3.5
Terminal type RM5	Cable cross section:	max. 2.5 mm², max. 14AWG
	Screw:	M3
	Tightening torque:	max. 0.5Nm, max. 4.43 Lb-in
Terminal type RM3.5	Cable cross section:	max. 1.5 mm², max. 16AWG
	Screw:	M2
	Tightening torque:	max. 0.2Nm, max. 1.77 Lb-in

6 Power supply

All of our IO modules support 12-48Vdc external power supply (±10%). The power cables should be selected according to the length of the power lines and the number of modules connected. When implementing a network with long cables, the use of thicker wire is more suitable due to the limitation of DC voltage drop. Furthermore, long wires can also cause interference with communication wires. All modules use onboard switching regulators to sustain efficiency over the 12..48Vdc input range. So the actual drawn current can be assumed to be inversely proportional to the DC voltage.

6.1 Power supply for serial ULTRA SLIM IO modules

The following drawings show the correct power supply for all of our serial SLIMIO products:

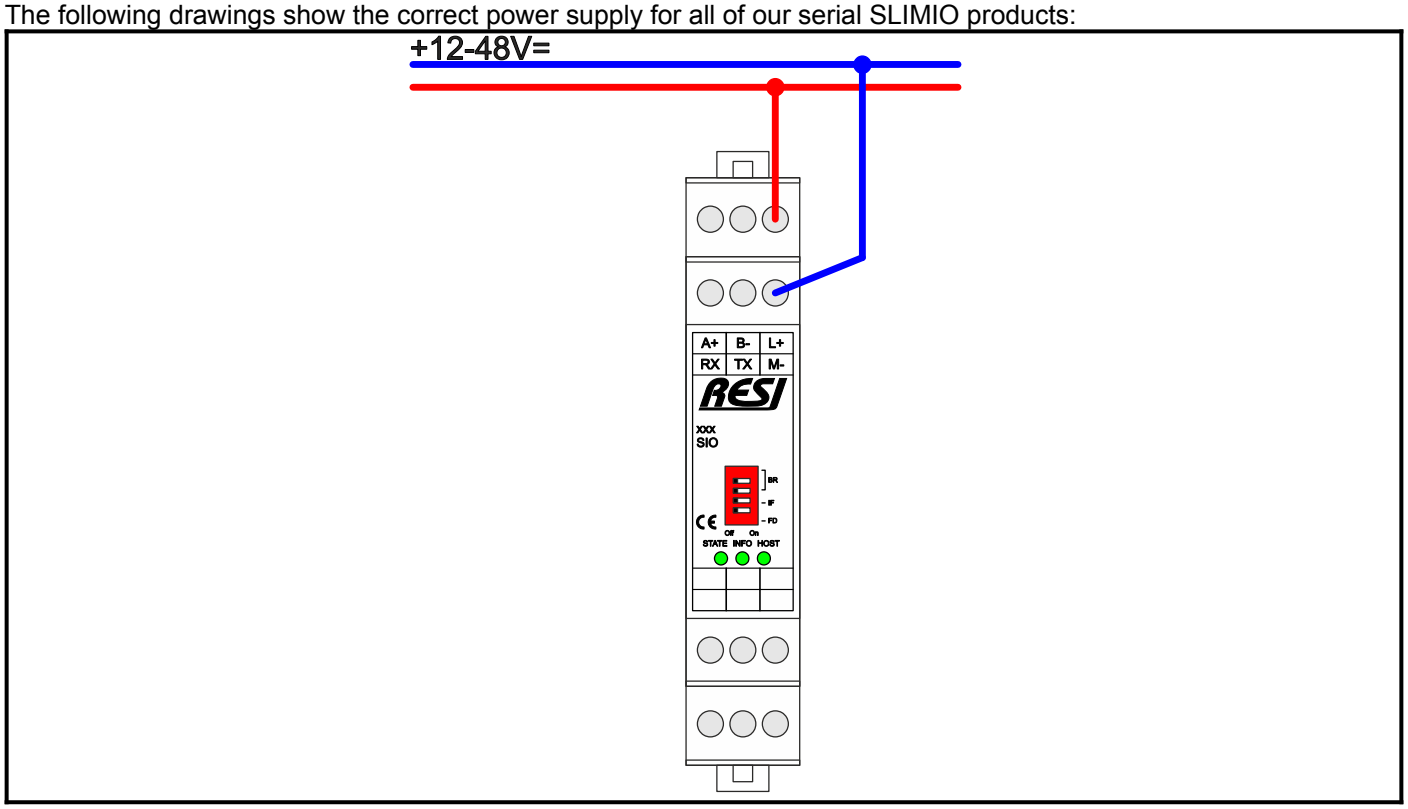


Figure: Power supply for our serial ULTRA SLIM IO modules

6.2 Power supply for Ethernet ULTRA SLIM IO modules

The following drawings show the correct power supply for all of our Ethernet SLIMIO products:

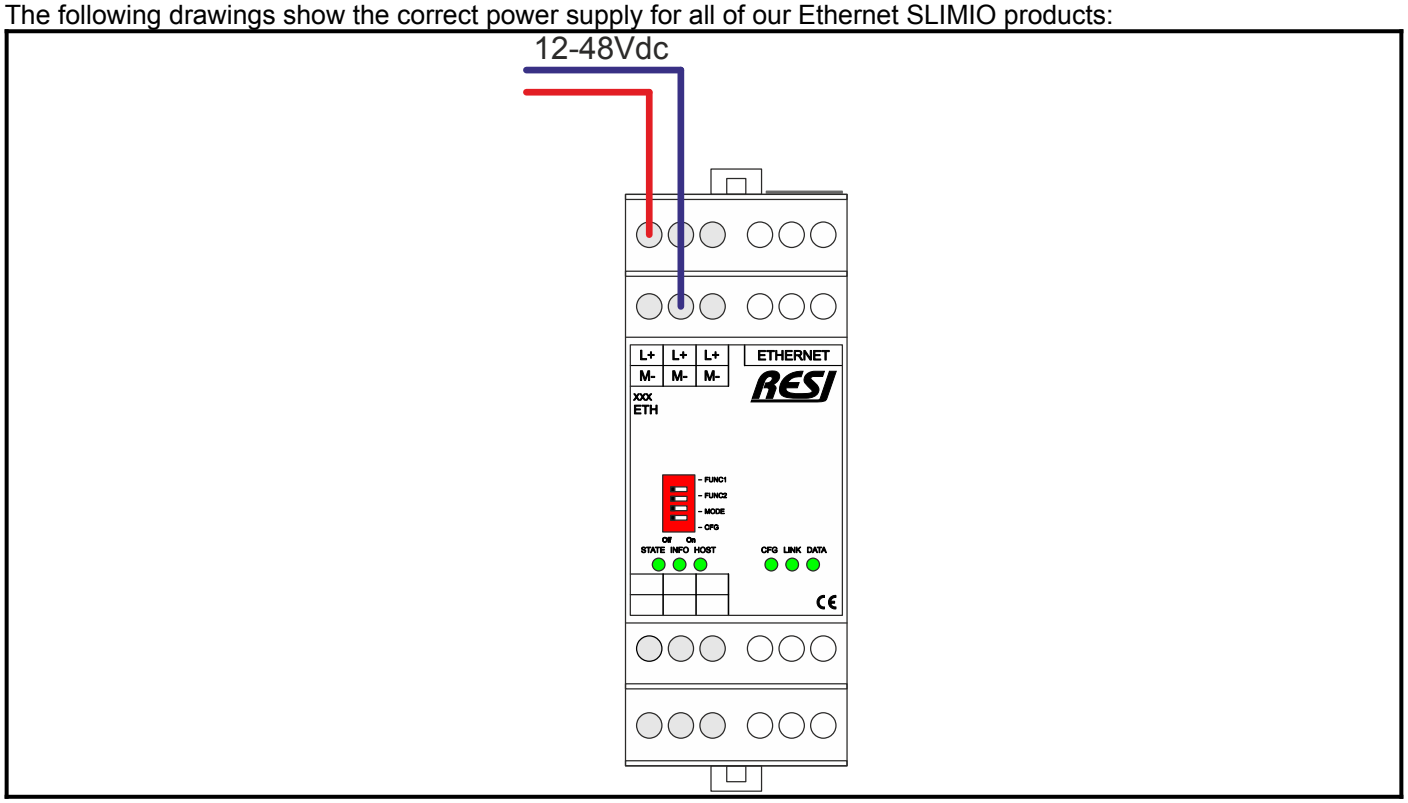


Figure: Power supply for our Ethernet ULTRA SLIM IO modules

6.3 Power supply for BIGIO XT8 modules

The following drawings show the correct power supply for all of our BIGIO XT8 products:

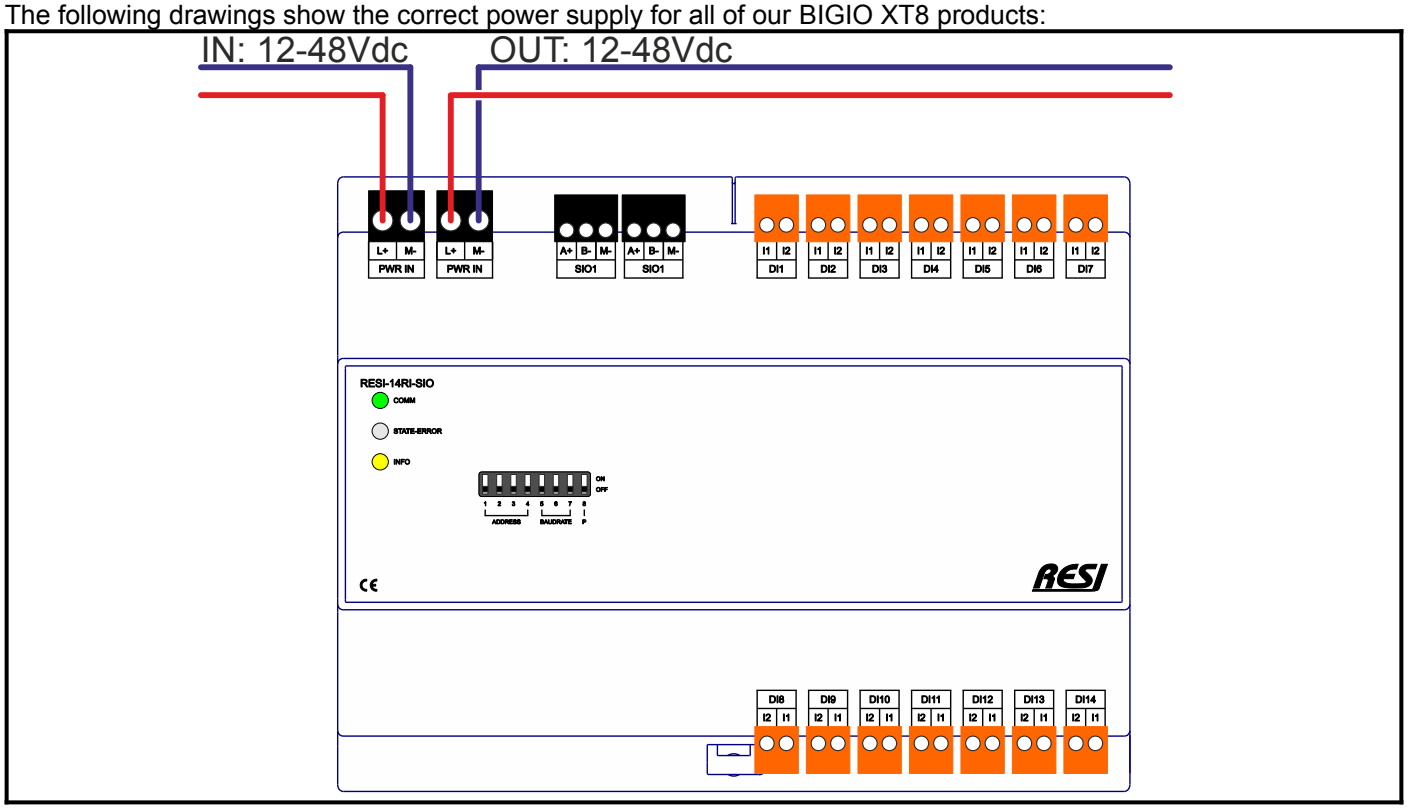


Figure: Power supply for our BIGIO XT8 modules

6.4 Power supply for BIGIO XT12 modules

The following drawings show the correct power supply for all of our BIGIO XT12 products:

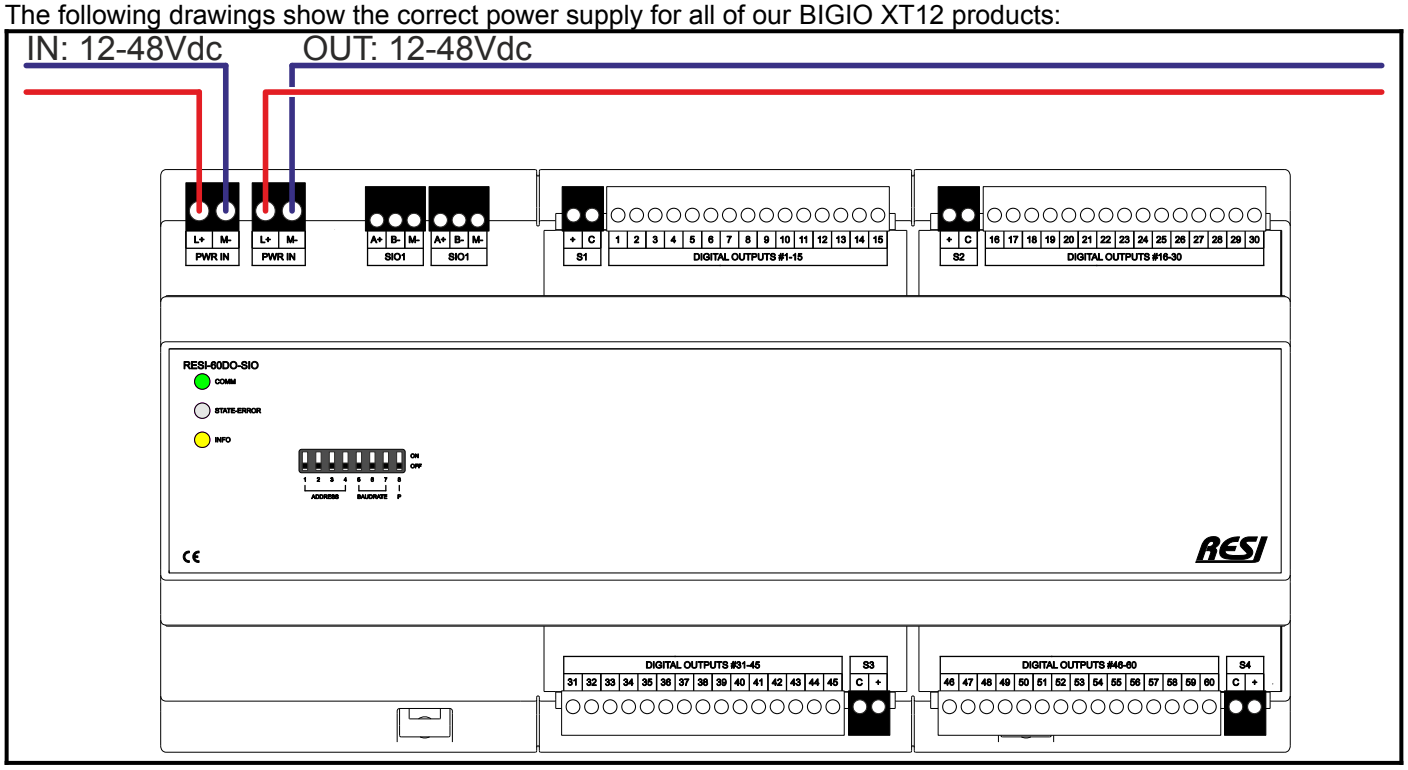


Figure: Power supply for our BIGIO XT12 modules

7 Serial connection

Our serial ULTRASLIM IO modules offer a RS232 or RS485 interface. Our serial BIGIO modules offer only a RS485 interface. The following drawings show the correct connection of the serial bus.

7.1 Serial connection for ULTRA SLIM IO modules

The following drawings show the correct serial connection of the RS232 or the RS485 for all of our serial SLIMIO products:

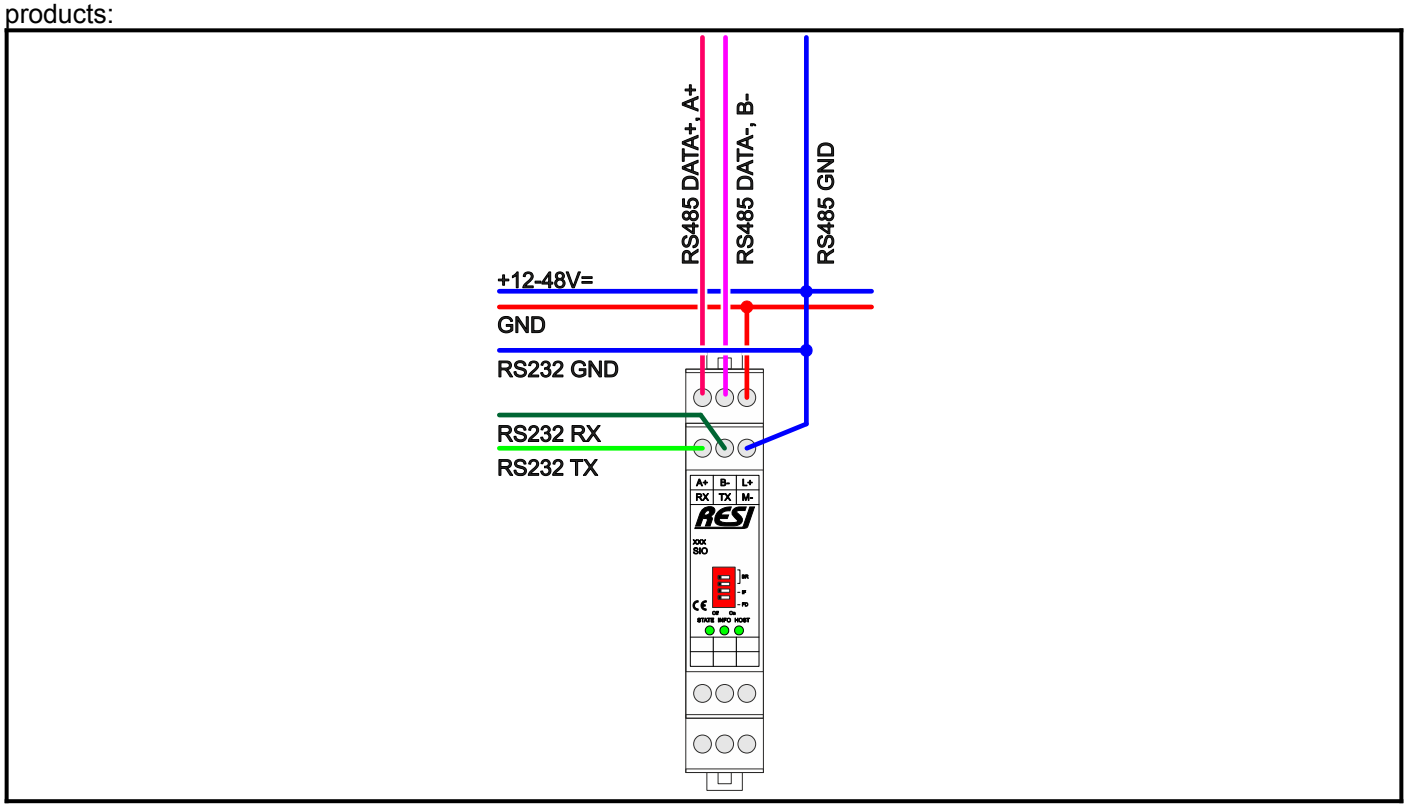


Figure: Serial connection for RS232 or RS485 for our serial ULTRA SLIM IO modules

7.2 Serial connection for BIGIO XT8 modules

The following drawings show the correct serial connection of the RS485 for all of our serial BIGIO products:

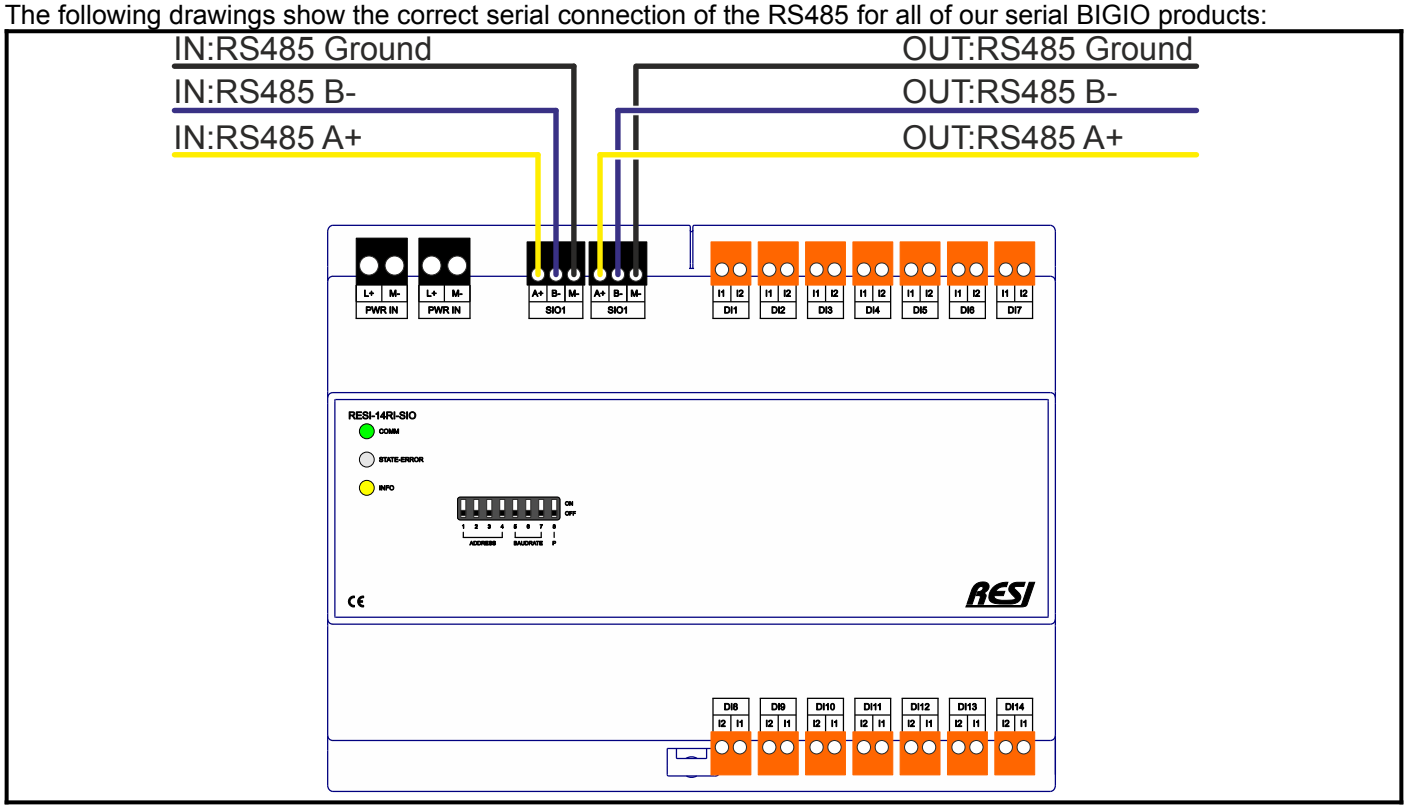


Figure: Serial connection for RS485 for our serial BIG IO modules

7.3 Serial connection for BIGIO XT12 modules

The following drawings show the correct serial connection of the RS485 for all of our serial BIGIO products:

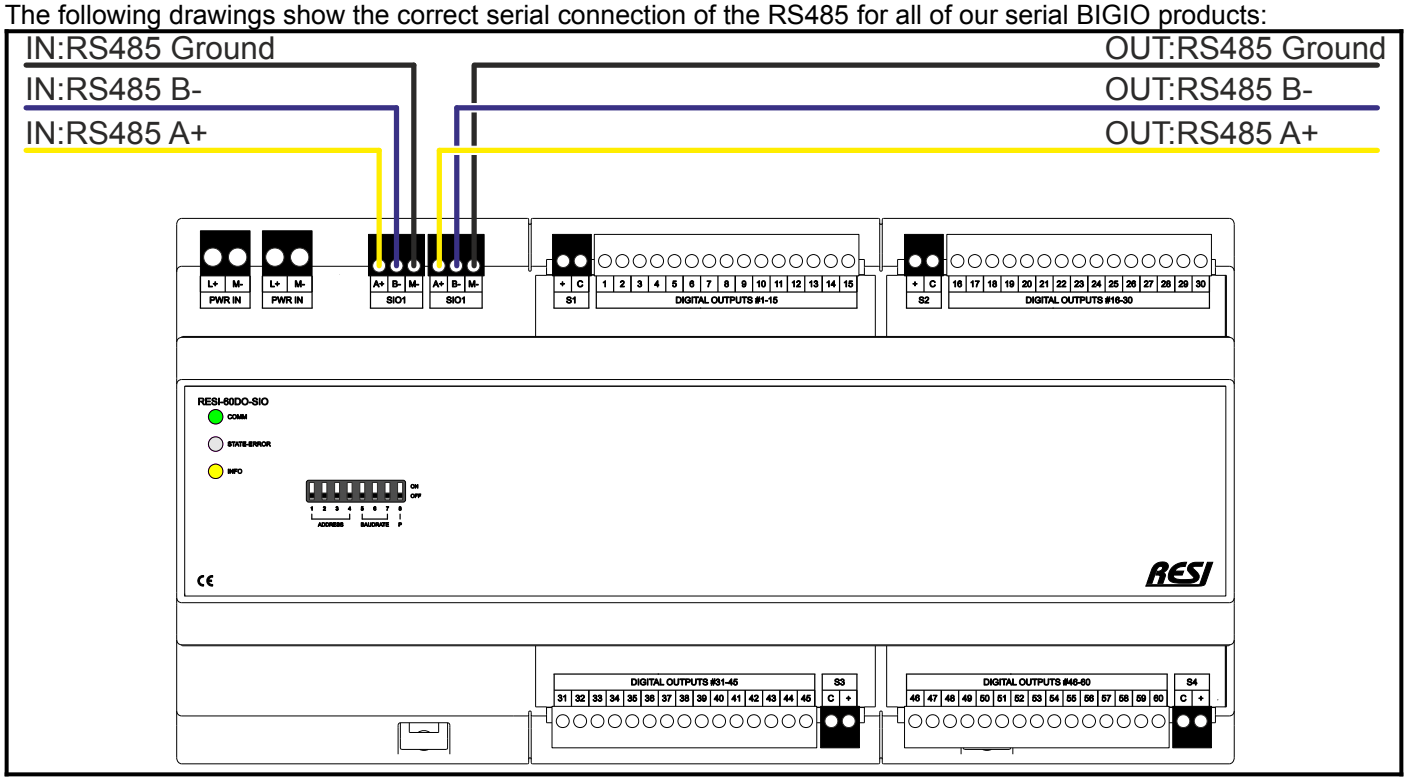


Figure: Serial connection for RS485 for our serial BIG IO modules

7.4 RESI-xxx-SIO SERIAL PROTOCOL

As mentioned our modules support either MODBUS/RTU or a simple ASCII text protocol.

7.4.1 MODBUS/RTU protocol

All of our serial IO modules communicate with a host system with the MODBUS/RTU slave protocol (RS232 or RS485 variants) or with the MODBUS/TCP server protocol (Ethernet version).

For communication via ASCII texts, ASCII messages with a special start character # (0x23, 35dec) and special end characters (0x0d, 13dec or CARRIAGE RETURN) are sent from the host to the module. The module also sends its responses with this special start and end character. See the ASCII command description below. In ASCII mode you can communicate with or without a bus number.

The following MODBUS functions are available for communication via MODBUS/RTU or MODBUS/TCP:

READ COIL STATUS (function code: 1)
READ INPUT STATUS (function code: 2)
READ HOLDING REGISTER (function code: 3)
READ INPUT REGISTER (function code: 4)
FORCE SINGLE COIL (function code: 5)
PRESET SINGLE REGISTER (function code: 6)
FORCE MULTIPLE COILS (function code: 15)
PRESET MULTIPLE REGISTERS (function code: 16)

Note:
The functions READ HOLDING REGISTER and PRESET MULTIPLE REGISTERS are limited to max. 125 registers limited per request! The functions READ INPUT STATUS, READ COIL STATUS and FORCE MULTIPLE COILS are limited to 2000 coils or inputs (bits) per data frame.

7.4.1.1 HOWTO map values to MODBUS registers

MODBUS is an international standard for communication between host systems like PLCs, DDCs or Industrial PCs and peripheral components or sensors.

More details about the MODBUS standard and the MODBUS protocol can be found here:
*http://en.wikipedia.org/wiki/**Modbus***
http://www.modbus.org/

You can find a documentation about this in the internet called “PI_MBUS_300.pdf”, which describes the MODBUS protocol pretty good.

There are three different MODBUS protocol versions available:
MODBUS/TCP: Used for communication with TCP/IP systems
MODBUS/RTU: A binary version of the MODBUS protocol
MODBUS/ASCII: An ASCII text based version of the protocol

To communicate, our RESI-xxx-SIO converters have either a RS232 interface to communicate 1 to 1, which means one MODBUS/RTU master (your host system) can talk to exact one MODBUS/RTU slave, or a RS485 to offer a one to many communication. Here one MODBUS/RTU master can communicate with a maximum of 255 MODBUS/RTU slaves. In older host systems the limit is 32 slaves. This depends on the capabilities of the RS485 driver IC in the host system. Our converters are able to use 256 communication partners on a RS485 line.

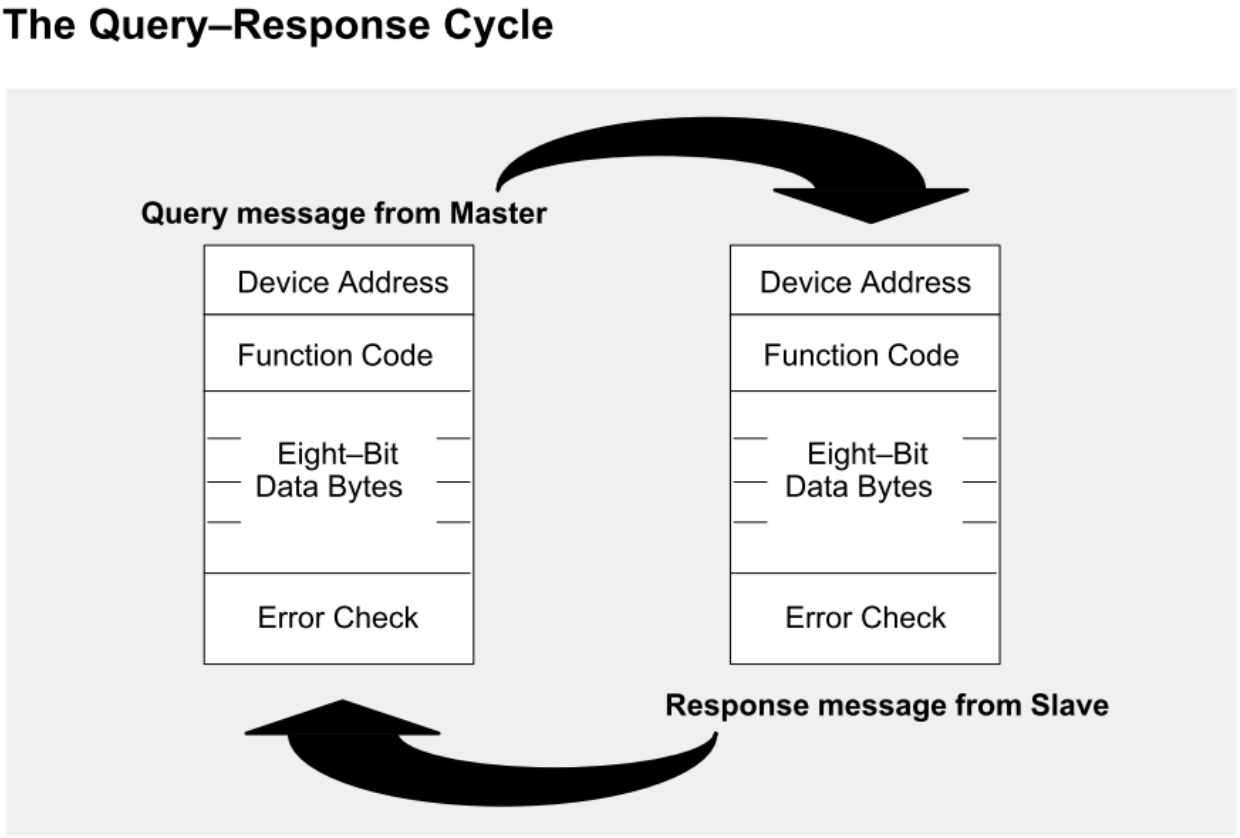
Our RESI-xxx-ETH converters can communicate with MODBUS/TCP protocol. A MODBUS/TCP system consists out of one TCP server which is in fact our gateway and at least one to n MODBUS/TCP clients. This will be your host. Our converters can connect only to one TCP client at a time.

To communicate the converters use an Ethernet interface.

MODBUS unit:
The MODBUS protocol demands a unique address of a MODBUS slave to address this special slave. This address is called MODBUS unit. The range of this address is from 0 to 255. Usually 0 is not used in applications. We use 0 for broadcast functions.

7.4.1.2 MODBUS query response cycle

MODBUS is a master slave protocol. This means, the master (your host system) has to send a protocol to a specific MODBUS slave (one of our converters), then this specific slave answers to the master, and then the master asks the next slave. The address of the slave is the so-called device address or unit address, which we mentioned before. See the below graphic, how a basic MODBUS request and response cycle looks like.



7.4.1.3 MODBUS/RTU telegram structure

A MODBUS/RTU protocol frame consists out of the following fields:

START: There is no specific start character, so a pause of four character timings depending on the baud rate of your communication must be established. This means at least for four characters, that there must be no communication on the serial line!

ADDRESS: This is the unit address of the slave, the master wants to talk to. It's a number between 0 and 255.

FUNCTION: This defines the type of data communication, the master wants to handle with the slave. Refer to the next pages for a detailed description of the functions.

DATA: This is a block of individual data bytes.

CRC CHECK: This is the checksum, to let the master and slave check, if the received protocol is correct and without communication errors.

END: Same as the start condition. Again there must not be communicated for at least 4 character times on the serial line.

IMPORTANT HINT: If there is more than one MODBUS slave on a serial line, the pausing of the START and END sequence are essential to re synchronize the slaves in case of data loss. If the host doesn't keep this gaps, communication with the slaves can be corrupted or impossible!

START	ADDRESS	FUNCTION	DATA	CRC CHECK	END
T1–T2–T3–T4	8 BITS	8 BITS	*n* x 8 BITS	16 BITS	T1–T2–T3–T4

7.4.1.4 MODBUS commands

The MODBUS standard defines many available commands . But not all systems handle the complete spectrum of telegrams. Our converter handles only all telegrams necessary for using holding and INPUT registers.

We support
03 READ HOLDING REGISTER
04 READ INPUT REGISTER
06 PRESET SINGLE REGISTER
16 PRESET MULTIPLE REGISTER

IMPORTANT HINT: All other protocols are ignored by our converters.

So what are HOLDING REGISTERs ?
According to the MODBUS standard, a MODBUS/RTU slave can hold up to 65535 HOLDING registers. Each holding register is a 16 bit register, capable for integer values between 0 and 65535 or in hexadecimal from 0x0000 to 0xFFFF. A MODBUS/RTU master system can read and write the contents of those registers.

IMPORTANT HINT:
A MODBUS/RTU master can read and write into this registers with a 16 bit index, called the
starting address. The problem is the definition of the starting address. A 16 bit value can store the values from 0 to 65535. But according the MODBUS standard the registers are numbered from 1 to 65536. So, if the MODBUS standard talks about register 1, an index of 0 must be used as start address in the telegram. You have to check carefully, how this index is interpreted by the manufacturers' documentation.

Code	Name
01	Read Coil Status
02	Read Input Status
03	Read Holding Registers
04	Read Input Registers
05	Force Single Coil
06	Preset Single Register
07	Read Exception Status
08	Diagnostics
09	Program 484
10	Poll 484
11	Fetch Comm. Event Ctr.
12	Fetch Comm. Event Log
13	Program Controller
14	Poll Controller
15	Force Multiple Coils
16	Preset Multiple Registers
17	Report Slave ID
18	Program 884/M84
19	Reset Comm. Link
20	Read General Reference
21	Write General Reference

Whenever you get a description of registers for a MODBUS device, the first question to solve is: How is the enumeration of the registers done?! Does the author use base=0, then he talks about the real start index of the telegram. Does the author mean base=1, conforming to naming conventions of the MODBUS consortium, then you have to subtract 1 before using this address in your telegrams.

IMPORTANT HINT:
If we display a holding register address like 4x00009 in our tool, we assume base=1 conforming to the standard. So your host system has to send the start index 00008 decimal to read out the correct register.

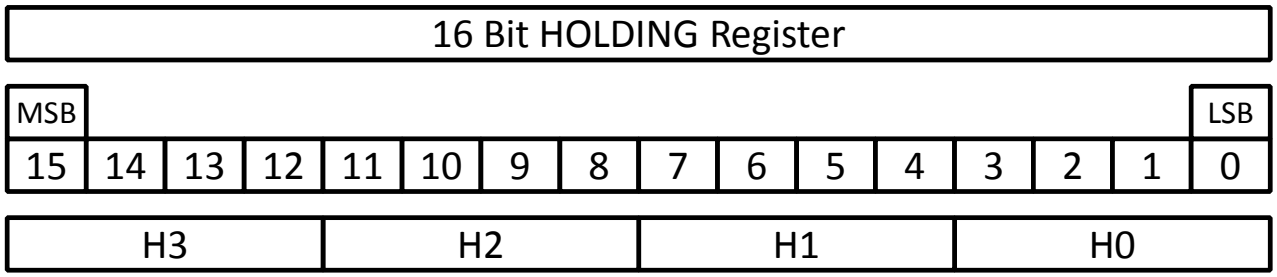
Start Index (Base=0)	MODBUS Register (Base=1)	Description
0	1	The first holding register
1	2	The second holding register
2	3	The third holding register
…	…	
65534	65535	The penultimate holding register
65535	65536	The last holding register

7.4.1.5 MODBUS 16 bit holding register structure

Here we give a brief introduction, how to build the contents of a MODBUS holding register, and how a hexadecimal writing of a 16 bit register looks like. We assume, that the user is familiar to hexadecimal and binary number systems and also how a computer stores data into its internal memory.

For more details consult the internet:
http://en.wikipedia.org/wiki/Hexadecimal
http://en.wikipedia.org/wiki/Binary_number

Usually a hexadecimal digit describes 4 bits. So we can group the 16 bits into 4 hexadecimal digits named H3,H2,H1,H0. This means eg. the hexadecimal number 0xABCD stands for H3=A, H2=B, H1=C, H0=D.



0xA=1010 binary, 10 dec, 0xB=1011,11 dec, 0xC=1100,12 dec and 0xD=1101, 13 dec. So the resulting binary number is 1010101111001101b or 43981 decimal.

See this graphical explanation, how the number is stored:

MSB															LSB
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	1	0	1	1	1	1	0	0	1	1	0	1

A	B	C	D

7.4.1.6 MODBUS big vs. least significant byte order

Now the first problem for a host system arises:
If we take the 16 bit number 0xABCD, we have to use 2 bytes to store this value internally. There are two concurrent versions of how to store this value in the RAM:

INTEL byte order, Little endian systems store the least significant byte first. So a memory map for 0xABCD look like:

Memory address 0	CD
Memory address 1	AB

MOTOROLA byte order, Big endian systems store the most significant byte first. So a memory map for 0xABCD look like:

Memory address 0	AB
Memory address 1	CD

Consult the internet for more details about this storage system.

http://en.wikipedia.org/wiki/Endianness

7.4.1.7 MODBUS storing larger data into 16 bit registers

After years, the market found out, that the capabilities of storing only 16 bit numbers into one holding register is not enough for many applications. The most common solution to store more than 16 bit values into holding registers is to use more than one register to hold the value. For storing e.g. a 32 bit value, we use two consecutive 16 bit holding registers, for storing a 32 bit float value we also use also two consecutive 16 bit registers!

We want to store the 32 bit integer value 0x12345678 into two consecutive holding registers starting at 4x00020. The memory map of the holding registers look like:

		16 bit value
Start Index 19	Holding Register 4x00020	0x1234
Start Index 20	Holding Register 4x00021	0x5678

But again, we can also store the reverse word order into two consecutive registers. Then the result looks like this:

		16 bit value
Start Index 19	Holding Register 4x00020	0x5678
Start Index 20	Holding Register 4x00021	0x1234

So none of the above mentioned orders is better than the other. It depends only on the programmer, how the 32 bit value is treated.

Be aware, that both systems (host and converter) have to treat the 32 bit value in the same way. Otherwise you will read out wrong data! We will discuss this issue later in combination with 32 bit float numbers.

Our converter uses the second described way to store 32 bit values. We follow the little endian strategy of INTEL systems and store 0x5678 into the first HOLDING register, and then we store 0x1234 in the consecutive register.

7.4.1.8 MODBUS datatypes in our converters

Our converter supports the following data types for storing MBUS values into MODBUS registers.

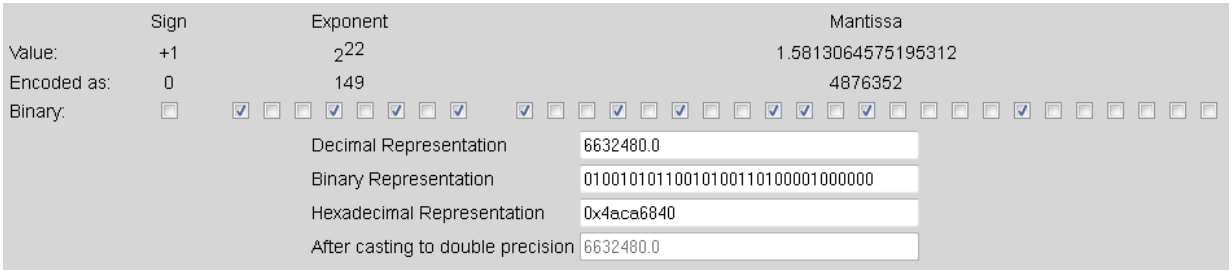
16 bit signed binary: This is an integer number between -32767..0..+32768 or 0x0000 to 0xFFFF hex. This number needs exactly one HOLDING register.

32 bit singed binary: This is an integer number between -2,147,483,647..0..+2,147,483,648 or 0x00000000 to 0xFFFFFFFF hex. This number needs two consecutive holding registers. We store the least significant word first. The serial number 2544082 is in hex 0x26D1D2. This leads to the following HOLDING register layout:

		16 bit value
Start Index 0	Holding Register 4x00001	0xD1D2 or 53714 dec
Start Index 1	Holding Register 4x00002	0x0026 or 38 dec

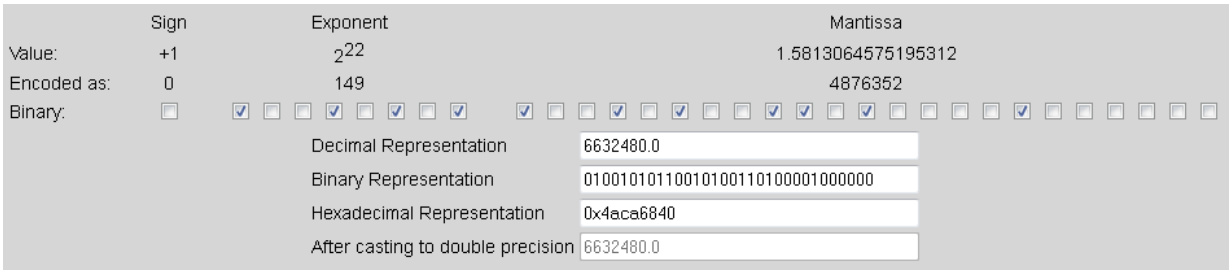
32 bit IEEE floating point: This is a float number using 32 bit. As before, this float needs two consecutive holding registers. We store the least significant word first. The energy value of 6632480,00 is defined in 32 bit hex with 0x4ACA6840. This leads to the following HOLDING register layout. For more details search in the internet or consult http://en.wikipedia.org/wiki/IEEE_floating_point or try out some float values and their hexadecimal representation under http://www.h-schmidt.net/FloatConverter/IEEE754.html

		16 bit value
Start Index 0	Holding Register 4x00001	0x6840
Start Index 1	Holding Register 4x00002	0x4ACA



32 bit IEEE floating point inverse: This is a float number using 32 bit. Again this float needs two consecutive holding registers. We store the least significant word first. The energy value of 6632480,00 is in 32 bit hex 0x4ACA6840. This means the following HOLDING register layout. For more details search in the internet or consult http://en.wikipedia.org/wiki/IEEE_floating_point or try out some float values and their hexadecimal representation under http://www.h-schmidt.net/FloatConverter/IEEE754.html

		16 bit value
Start Index 0	Holding Register 4x00001	0x4ACA
Start Index 1	Holding Register 4x00002	0x6840



IMPORTANT HINT:
32 bit floats are very tricky! Eg. The value 3,5351799 is represented internally as 0x40624063. But the reverse word order (if the host reads out the wrong register indexes or the host corrupts the word order) 0x40634062 leads to the float number 3,5508046. So this error in your software is very hard to find! Be very cautious, which register indexes you read and how the word order of the two registers are interpreted.

32 bit date&time: This is a compressed format using 32 bit. Again the least significant word is stored into the first register. The structure of the 32 bits are:
Bits 0..7: minute
Bits 8..15: hour
Bits 16..20: day
Bits 21..24: month
Bits 25..31: year
The current date & time “07.04.00 01:13” is represented hexadecimal with 0x0087010d (8847628dec) and stored as followed:

		16 bit value
Start Index 0	Holding Register 4x00001	0x010D
Start Index 1	Holding Register 4x00002	0x0087

7.4.1.9 MODBUS datatype storage and common pitfalls

In general MODBUS uses 16 bit wide registers. So if you use only datatypes, which needs also only one register, the mapping is easy. But as soon as you use datatypes, e.g. UINT32, which need two or more MODBUS registers, you can map the values in different ways.

We do a simple sample. We want to store the 32 bit unsigned integer value in hexadecimal 0x12345678 in MODBUS holding registers starting with index 4x00010. The mapping can be done in two different ways:

MODBUS Register	Storage of UINT32 datatype
4x00010 I:9	The high word of the 32 bit value 0x12345678 is stored in the first 16 bit wide MODBUS register. This means the value 0x1234 is stored here.
4x00011 I:10	The low word of the 32 bit value 0x12345678 is stored in the second 16 bit wide MODBUS register. This means the value 0x5678 is stored here.

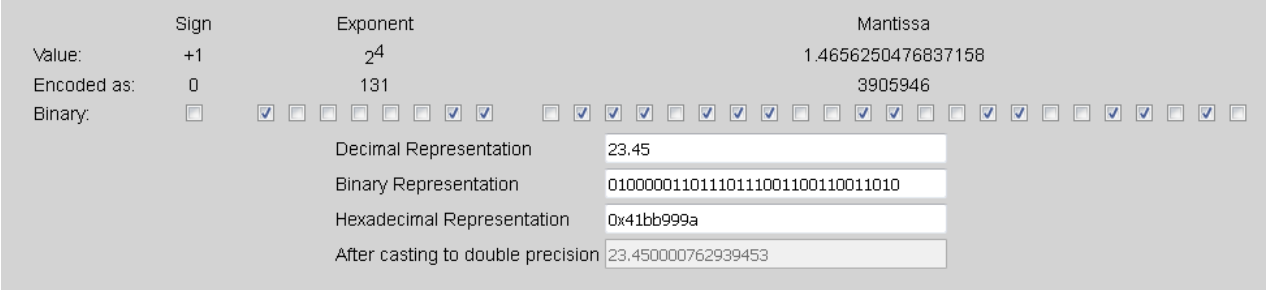
But it is only one possibility, that we store the high word in the first MODBUS register. With the same right, we can define to store the low word in the first register, and the high word in the second.

The result will look like this:

MODBUS Register	Storage of UINT32R datatype
4x00010 I:9	The low word of the 32 bit value 0x12345678 is stored in the first 16 bit wide MODBUS register. This means the value 0x5678 is stored here.
4x00011 I:10	The high word of the 32 bit value 0x12345678 is stored in the second 16 bit wide MODBUS register. This means the value 0x1234 is stored here.

More complicated is the storage of a FLOAT32 value into two consecutive holding registers. We use a standard room temperature e.g. 23,45 °C as a value, we want to store into two registers.

First we have to translate this value into a valid IEE754 float value. Therefore we use a perfect site in the internet (http://www.h-schmidt.net/FloatConverter/IEEE754.html):



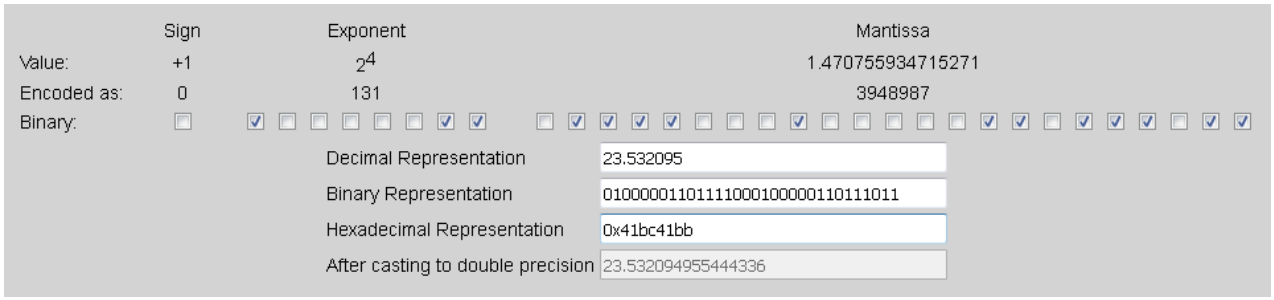
We enter the value 23.45 and we get a 32 bit hexadecimal representation of the float value. It is the number 0x41BB999A. Now we store this value in the same way, we have stored the UINT32 value into two registers:

MODBUS Register	Storage of FLOAT32 datatype
4x00010 I:9	The high word of the 32 bit float value 0x41BB999A is stored in the first 16 bit wide MODBUS register. This means the value 0x41BB is stored here.
4x00011 I:10	The low word of the 32 bit float value 0x41BB999A is stored in the second 16 bit wide MODBUS register. This means the value 0x999A is stored here.

But we can also use the reverse notation:

MODBUS Register	Storage of FLOAT32R datatype
4x00010 I:9	The low word of the 32 bit float value 0x41BB999A is stored in the first 16 bit wide MODBUS register. This means the value 0x999A is stored here.
4x00011 I:10	The high word of the 32 bit float value 0x41BB999A is stored in the second 16 bit wide MODBUS register. This means the value 0x41BB is stored here.

Now we show a common pitfall in writing and reading more than one MODBUS register and rebuilding a value. We use a different float value. In hexadecimal it is 0x41BC41BB. Again we use the online converter:

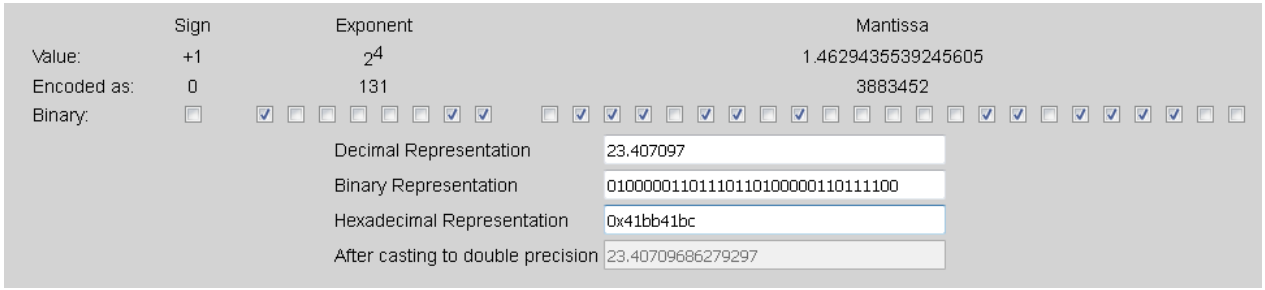


You notice, the float value is 23.532095.

Now we store it with HIGH word first into two registers:

MODBUS Register	**Storage of FLOAT32 datatype**
4x00010 I:9 HIGH WORD	The high word of the 32 bit float value 0x41BC41BB is stored in the first 16 bit wide MODBUS register. This means the value 0x41BC is stored here.
4x00011 I:10 LOW WORD	The low word of the 32 bit float value 0x41BC41BB is stored in the second 16 bit wide MODBUS register. This means the value 0x41BB is stored here.

But now we make a very big mistake, we read the two registers and restore the hexadecimal value in our host software in the reverse word order. First low word, then high word. The result is the 32 bit value 0x41BB41BC instead the correct value 0x41BC41BB. Then we convert this into an IEE754 float value.



The result is 23.407097. This is not far away from the original number of 23.532095! So this massive software error can be undiscovered for a long time. Only if the reverse float value generates numbers which are physically not possible for the measured signal, this error is discovered!

7.4.1.10 MODBUS data type table

The following table shows, how more complex data types are stored in successive 16 bit holding or input registers within the MODBUS registers.:

MODBUS DATATYPE	SIZE	WORD ORDER	DESCRIPTION
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:Higher 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.

7.4.1.11 MODBUS table

COILS (1x) & INPUTS (2x)
The module holds internally a list of 1 bit coil and input register. Those registers can be read by the host with the function READ COIL STATUS (function code: 1). If the register can also be modified by the host, the host can use the functions FORCE SINGLE COIL (function code: 5) and FORCE MULTIPLE COILS (function code: 15).

In addition the SAME registers are also readable over the function READ INPUT STATUS (function code: 2). This is for host systems, which do not support all MODBUS/RTU functions properly.

The MODBUS convention defines 65535 possible coils with the notation 1x00001 to 1x65536. Inputs are usually noted with 2x00001 to 2x65536. Please refer the software MODBUS POLL as a sample for this notation. Internally in the MODBUS/RTU frames an index notation is used, which starts with 0 and ends with 65535. So we decided to note in the following document a register with: 1x00100 for the coil 100, 2x00100 as a hint, that you can read this register also as the input 100, and in addition also the real index of the protocol index 99 with the notation I:99.

HOLDING REGISTER (3x) & INPUT REGISTER (4x)
The module holds internally a list of 16 bit wide holding register. Those registers can be read by the host with the function READ HOLDING REGISTER (function code: 3). If the register can also be modified by the host, the host can use the functions PRESET SINGLE REGISTER (function code: 6) and PRESET MULTIPLE REGISTERS (function code: 16).

In addition the SAME holding registers are also readable over the function READ INPUT REGISTER (function code: 4). This is for host systems, which do not support all MODBUS/RTU functions properly.

The MODBUS convention defines 65535 possible holding register with the notation 4x00001 to 4x65536. Input register are usually noted with 3x00001 to 3x65536. Please refer the software MODBUS POLL as a sample for this notation. Internally in the MODBUS/RTU frames an index notation is used, which starts with 0 and ends with 65535. So we decided to note in the following document a register with: 4x00100 for the holding register 100, 3x00100 as a hint, that you can read this register also as the input register 100, and in addition also the real index of the protocol index 99 with the notation I:99.

SOFTWARE RESET						
RESET	1x06001 2x06001 I:6000	0,0x00 B:00		N/A:NO CHANGE	BIT R/W	NO
Performs a software reset, whenever 1 is written to this register. If the host writes to this register 1, the module executes a soft reset (reboot).						
RESET	3x06001 4x06001 I:6000	0,0x0000 B:00 00		N/A:NO CHANGE	UINT16 R/W	NO
Performs a software reset, whenever 1 is written to this register. If the host writes to this register 1, the module executes a soft reset (reboot).						
CONVERTER STATUS						
CONVERTER STATUS	3x06002 4x06002 I:6001	0,0x0000 B:00 00			UINT16 R/O	
Current status of the converter						
DIP SWITCH						
DIP SWITCH	3x10010 4x10010 I:10009	65,0x0041 B:00 41			UINT16 R/O	
Returns the current setting of the Dip switches. For ULTRA SLIM IOs: The current value of the DIP switches: Bit 0: DIP Switch 1 (=0:OFF, =1:ON) Bit 1: DIP Switch 2 (=0:OFF, =1:ON) Bit 2: DIP Switch 3 (=0:OFF, =1:ON) Bit 3: DIP Switch 4 (=0:OFF, =1:ON) For BIG IOs: The current value of the DIP switches: Bit 0: DIP Switch 1 (=0:OFF, =1:ON) Bit 1: DIP Switch 2 (=0:OFF, =1:ON) Bit 2: DIP Switch 3 (=0:OFF, =1:ON) Bit 3: DIP Switch 4 (=0:OFF, =1:ON) Bit 4: DIP Switch 5 (=0:OFF, =1:ON) Bit 5: DIP Switch 6 (=0:OFF, =1:ON) Bit 6: DIP Switch 7 (=0:OFF, =1:ON) Bit 7: DIP Switch 8 (=0:OFF, =1:ON)						
PRODUCT DATA						
HW_GROUP	3x65201 4x65201 I:65200	4096,0x1000 B:10 00			UINT16 R/O	
This is the group of hardware of the current product						
SW_GROUP	3x65202 4x65202 I:65201	37,0x0025 B:00 25			UINT16 R/O	
This is the group of software of the current product						
SW_VERSION	3x65203 4x65203 I:65202	4352,0x1100 B:11 00			UINT16 R/O	
		SW VERSION:1.1.0				
This is the current software version of the firmware						
SW_AUTHOR	3x65204 4x65204 I:65203	18771,0x4953 B:49 53			UINT16 R/O	
This is the current software author of the firmware						

MODBUS SETTINGS						
UNIT_ID	3x65222 4x65222 I:65221	1,0x0001 B:00 01		N/A:NO CHANGE	UINT16 R/W	NO
		UNIT ID:1				
f the host reads this register, the current programmed unit ID is returned. All values above unit ID 255 define also the unit ID 255. If the host write a new value into this register, the new value will be stored in the FLASH as the new unit ID. The new unit ID is activated after a power off/power on cycle or a software reboot of the module. The host can execute a reboot in writing to the register RESET SYSTEM. NOTE:DIP switch 4 must set to OFF to activate this unit ID, otherwise the unit ID is 255.						
HINT:This settings will be active after you repower or reset your device !!						
BAUD_RATE	3x65223 4x65223 I:65222	57600,0x0000E100 B:00 00 E1 00	38400	38400	UINT32 R/W	NO
		57600Bd		ENTER BAUD RATE		
This is the current configured baud rate in the FLASH For ULTRA SLIM IOs RESI-xxx-SIO: This baudrate is only used, if DIP switch mode DIP1=ON+DIP2=ON (BR) (default is 57600bd) For BIG IOs RESI-xxx-SIO: This baudrate is only used, if DIP switch mode DIP7=ON (PARAMETER) (default is 57600bd)						
Valid baud rates are: 300bd, 600bd, 900bd, 1200bd, 2400bd, 4800bd, 9600bd, 19200bd, 38400bd, 57600bd, 115200bd, 128000bd 230400bd, 250000bd, 256000bd						
HINT:This settings will be active after you repower or reset your device !!						
PARITY	3x65225 4x65225 I:65224	0,0x0000 B:00 00		N/A:NO CHANGE	UINT16 R/W	NO
		NO PARITY		SELECT PARITY		
If the register is read out, the currently set parity of the serial interface is returned. Writing a value to this register will change the new parity in FLASH. This will only take effect after a restart of the module. This can be triggered by writing to the RESET SYSTEM register. Parity values are 0: no parity 1: even parity 2: odd parity						
STOP BITS	3x65226 4x65226 I:65225	1,0x0001 B:00 01		N/A:NO CHANGE	UINT16 R/W	NO
		ONE STOPBIT		SELECT STOPBITS		
f the register is read out, the currently set number of stop bits of the serial interface is returned. Writing a value to this register will change the new number of stop bits in the FLASH. This will only take effect after a restart of the module. This can be triggered by writing to the RESET SYSTEM register. Values for stop bits are 1: one stop bit 2: two stop bits						

7.4.2 ASCII protocol

All of our IO modules communicate with very simple ASCII commands. The following special characters are used in this description:

stands for the **hash sign** ASCII character 35dec or 0x23
: stands for the **colon** ASCII characters 58dec or 0x3A
= stands for the **equal sign** with the ASCII code 61ec or 0x3D
- stands for the **minus sign** with the ASCII code 45dec or 0x2D
, stands for the **comma** with the ASCII code 44dec or 0x2C
<CR> or $_{CR}$ stands for the **CARRIAGE RETURN** ASCII character 13dec or 0x0D. This is shown as CR in the following.
<SP> or ☐ stands for **SPACE**. This is the space in ASCII code 32dec or 0x20. The space is shown as , hereinafter.
In the following **<ADR>** is used for the **bus address**. This can be transmitted in decimal or hexadecimal and is separated from the following command with a comma (ASCII characters 44dec or 0x2C). Hexadecimal numbers always start with 0x. Only the ASCII characters '0' - '9' 48dec to 57dec, 0x30-0x39 and 'A' to 'F', 65dec to 70dec, 0x41-0x46 may be used. Each module always responds to broadcast address 0 and its own bus address. An external DIP switch can be used to quickly switch between the fixed bus address 255 and the programmed bus address. See the DIP switch description.

7.4.2.1 COMMUNICATION SEQUENCE

In principle, the IO module does not send any characters by itself. Communication always starts from the host. If only one IO module is used on a bus line (e.g. with RS232 interface), there is no need for a bus address in the protocol. In RS485 mode, however, several modules can be connected on an RS485 line. Then a bus address is absolutely necessary for communication.

The command structure looks like this:

The host sends a command or a command with parameters without a bus address:
#<command><CR> or
#<command>:<parameter><CR>

The module responds when it feels addressed with the telegram:
#<respond><CR>

The host sends the following to the module with the bus address:
#<ADR>,<command><CR> or
#<ADR>,<command>:<parameter><CR>

The module then replies with:
#<ADR>,<reply><CR>

The bus address is in the range from 1dec to 255dec or 0x00 to 0xFF hexadecimal. The setting is made using our free configuration software MODBUSConfigurator or our free LIBRE OFFICE® based configurator.

There are two spellings for each command. A long version and a short version, so that you have to send less. For example, you can query the software version with the VERSION command or with the VER command.

7.4.2.2 Example: Query VERSION

This command provides the current type of the module.

Long host version:
#VERSION<CR> or
#<ADR>,VERSION<CR>

Short host version:
#VER<CR> oder
#<ADR>,VER<CR>

Reply:
#VERSION:<HIGH>.<MED>.<LOW><CR> oder
#<ADR>,VERSION:<HIGH>,<MED>,<LOW><CR>

<HIGH>.<MED>.<LOW> represents the current software version, e.g. 3.0.0

Examples:
#VERSION$_{CR}$
#VERSION:3.0.0$_{CR}$

With broadcast address in decimal and long version:
#0,VERSION$_{CR}$
#0,VERSION:3.0.0$_{CR}$

With broadcast address in hexadecimal and short version:
#0x00,VER$_{CR}$
#0x00,VERSION:3.0.0$_{CR}$

With bus address 255 in decimal
#255,VER$_{CR}$
#255,VERSION:3.0.0$_{CR}$

With bus address 255 in hexadecimal
#0xFF,VERSION$_{CR}$
#0xFF,VERSION:3.0.0$_{CR}$

With bus address 43 in decimal
#43,VER$_{CR}$
#43,VERSION:3.0.0$_{CR}$

With bus address 43 in hexadecimal
#0x2B,VER$_{CR}$
#0x2B,VERSION:3.0.0$_{CR}$

7.4.2.3 Example: Query module TYPE

This command provides the current type of the module.

Long host version:
#TYPE<CR> or
#<ADR>,TYPE<CR>

Host short version:
#TYP<CR> or
#<ADR>,TYP<CR>

Respond:
#TYPE:<TYP><CR> or
#<ADR>,TYPE:<TYP><CR>

<TYP> represents the current type of the module. A RESI-2RI-SIO is shown as an example

Examples:
#TYPE$_{CR}$
#TYPE:RESI-2RI-SIO$_{CR}$

#255,TYP$_{CR}$
#255,TYPE:RESI-2RI-SIO$_{CR}$

7.4.2.4 Table of all ASCII commands

In this list you will find all possible ASCII commands. Only the version including the bus address is listed here. It has already been explained that this can also be omitted. If an argument has the addition dec, it is returned as a decimal number. If an argument has the addition hex, a hexadecimal number is returned. Many commands return both the decimal and the hexadecimal representation. The host can thus choose which number conversion he would like to carry out.

Please refer to the description of individual products for more details about the available ASCII commands.

ASCII COMMANDS				
GET VERSION	**ASCII READ COMMAND**	#VERSION<CR> #VER<CR> Result: #VERSION:<VersionHi>,<VersionMed>,<VersionLo><CR>	ASCII	
	TX	#VERSION<CR>		
	RX	#1,VERSION:1.1.0<CR>		
		Current SW version:1.1.0		
Returns the version number of the module VersionHi: Version number high (1..255) VersionMed: Version number medium (1..255) VersionLo: Version number low (1..255)				
GET TYPE	**ASCII READ COMMAND**	#TYPE<CR> #TYP<CR> Result: #TYPE:<Type><CR>	ASCII	
	TX	#TYPE<CR>		
	RX	TYPE:RESI-S8RO-SIO<CR>		
		Current module type:RESI-S8RO-SIO		
Returns the current module type				
GET OWNER	**ASCII READ COMMAND**	#OWNER<CR> #OWN<CR> Result: #OWNER:<Owner><CR>	ASCII	
	TX	#OWNER<CR>		
	RX	OWNER:RESI<CR>		
		Current owner:RESI		
Returns the current owner of the module				
GET CREATOR	**ASCII READ COMMAND**	#CREATOR<CR> #CRE<CR> Result: #CREATOR:<Creator><CR>	ASCII	
	TX	#CREATOR<CR>		
	RX	#1,CREATOR:DI HC SIGL,MSC<CR>		
		Current creator:DI HC SIGL,MSC		
Returns the current creator of the module				
GET COPYRIGHT	**ASCII READ COMMAND**	#COPYRIGHT<CR> #COPY<CR> Result: #COPYRIGHT:<Copyright><CR>	ASCII	
	TX	#COPYRIGHT<CR>		
	RX	#1,COPYRIGHT:2015-20 BY RESI AND DI HC SIGL,MSC WWW.RESI.CC<CR>		
		Current copyright:2015-20 BY RESI AND DI HC SIGL,MSC WWW.RESI.CC		
Returns the current copyright of the module				
GET DIP SWITCH	**ASCII READ COMMAND**	#GET DIP<CR> #GDIP<CR> Result: #GDIP:<DIPSwitchDec>,<DIPSwitchHex><CR>	ASCII	
	TX	#GET DIP<CR>		
	RX	#1,GDIP:65,0x41<CR>		
		Current DIP SWITCH settings:01000001		
Returns the current setting of the Dip switches as decimal number and as hexadecimal number. DIPSwitchDec DIPSwitchHex The current value of the DIP switches: Bit 0: DIP Switch 1 (=0:OFF, =1:ON) Bit 1: DIP Switch 2 (=0:OFF, =1:ON) Bit 2: DIP Switch 3 (=0:OFF, =1:ON) Bit 3: DIP Switch 4 (=0:OFF, =1:ON) Bit 4: DIP Switch 5, if available (=0:OFF, =1:ON) Bit 5: DIP Switch 6, if available (=0:OFF, =1:ON) Bit 6: DIP Switch 7, if available (=0:OFF, =1:ON) Bit 7: DIP Switch 8, if available (=0:OFF, =1:ON)				

MODBUS INTERFACE				
SET MODBUS ADDRESS	**ASCII WRITE COMMAND**	#SET MODBUS ADDRESS:<UNITID><CR> #SETMBADR:<UNITID><CR> Result: #OK<CR>	ASCII	NO
	UNITID	1		
	TX	#SET MODBUS ADDRESS:1<CR>		
	RX	N/A		
Redefines the unit ID of the module. This change will affect the MODBUS/RTU communication immediately. As a Unit IO you can use the values 0dec to 255dec. HINT: The new settings are activated after a system reboot or power off on cycle!				
SET MODBUS BAUDRATE	**ASCII WRITE COMMAND**	#SET MODBUS BAUDRATE:<BAUD><CR> #SETMBBAUD:<BAUD><CR> Result: #OK<CR>	ASCII	NO
	BAUD	57600:57600BD		
	TX	#SET MODBUS BAUDRATE:57600<CR>		
	RX	N/A		
Sets a new baud rate in the FLASH For ULTRA SLIM IOs RESI-xxx-SIO: This baudrate is only used, if DIP switch mode DIP1=ON+DIP2=ON (BR) (default is 57600bd) For BIG IOs RESI-xxx-SIO: This baudrate is only used, if DIP switch mode DIP7=ON (PARAMETER) (default is 57600bd) The following baudrates are allowed: 300bd, 600bd, 900bd, 1200bd, 2400bd, 4800bd, 9600bd, 19200bd, 38400bd, 57600bd, 115200bd, 128000bd 230400bd, 250000bd, 256000bd HINT: The new setup parameters will be active after a restart of the module.				
SET MODBUS PARITY	**ASCII WRITE COMMAND**	#SET MODBUS PARITY:<PARITY><CR> #SETMBPAR:<PARITY><CR> Result: #OK<CR>	ASCII	NO
	PARITY	NONE:NO PARITY		
	TX	#SET MODBUS PARITY:NONE<CR>		
	RX	N/A		
Sets a new parity for the serial interface. MBParity: NONE: no parity EVEN: even parity ODD: odd parity HINT: The new setup parameters will be active after a restart of the module.				
SET MODBUS STOPS	**ASCII WRITE COMMAND**	#SET MODBUS STOP:<STOPBIT><CR> #SETMBSTOP:<STOPBIT><CR> Result: #OK<CR>	ASCII	NO
	STOPBIT	ONE:ONE STOPBIT		
	TX	#SET MODBUS STOP:ONE<CR>		
	RX	N/A		
Sets a new amount of stop bits for the serial interface. MBStops ONE: one stop bit TWO: two stop bits HINT: The new setup parameters will be active after a restart of the module.				
SET MODBUS PARAMS	**ASCII WRITE COMMAND**	#SET MODBUS PARAMS:<UNITID>,<BAUD>,<PARITY>,<STOPBIT><CR> #SETMBPARAMS:<UNITID>,<BAUD>,<PARITY>,<STOPBIT><CR> Result: #OK<CR>	ASCII	YES
	UNITID	1		
	BAUD	57600:57600BD		
	PARITY	NONE:NO PARITY		
	STOPBIT	ONE:ONE STOPBIT		
	TX	#SET MODBUS PARAMS:1,57600,NONE,ONE<CR>		
	RX	N/A		
Sets all parameters for serrial interface				

GET MODBUS ADDRESS	**ASCII READ COMMAND**	#GET MODBUS ADDRESS<CR> #GMBADR<CR> Result: #GMBADR:<MBUnitDec>,<MBFLASHDec>,<MBUnitHex>,<MBFLASHHex><CR>	ASCII	
	TX	#GET MODBUS ADDRESS<CR>		
	RX	#1,GMBADR:1,1,0x1,0x1<CR>		
		Current MODBUS unit ID for DIP4=OFF:1,1,0x1,0x1		
Shows the current used MODBUS/RTU or ASCII unit address and shows also the stored unit address in the FLASH memory, which is only used if the DIP switch for the bus address is set to 0.MBUnitDecMBUnitHexThe current used MODBUS/RTU unit or ASCII address for communicationMBFLASHDecMBFLASHHexThe internal stored MODBUS/RTU unit address or ASCII address from the FLASH memory, if the DIP switch DIP3 is OFF.				
GET MODBUS BAUDRATE	**ASCII READ COMMAND**	#GET MODBUS BAUDRATE<CR> #GMBBAUD<CR> Result: #GMBBAUD:<BaudRate><CR>	ASCII	
	TX	#GET MODBUS BAUDRATE<CR>		
	RX	#1,GMBBAUD:57600<CR>		
		Current baudrate for DIP1+2=ON:57600		
This is the current configured baud rate in the FLASH For ULTRA SLIM IOs RESI-xxx-SIO: This baudrate is only used, if DIP switch mode DIP1=ON+DIP2=ON (BR) (default is 57600bd) For BIG IOs RESI-xxx-SIO: This baudrate is only used, if DIP switch mode DIP7=ON (PARAMETER) (default is 57600bd) The following baudrates are allowed: 300bd, 600bd, 900bd, 1200bd, 2400bd, 4800bd, 9600bd, 19200bd, 38400bd, 57600bd, 115200bd, 128000bd 230400bd, 250000bd, 256000bd				
GET MODBUS PARITY	**ASCII READ COMMAND**	#GET MODBUS PARITY<CR> #GMBPAR<CR> Result: #GMBPAR:<MBParity><CR>	ASCII	
	TX	#GET MODBUS PARITY<CR>		
	RX	#1,GMBPAR:NONE<CR>		
		Current parity:NONE		
Shows the current configured parity of the serial interface. MBParity NONE: no parity EVEN: even parity ODD: odd parity				
GET MODBUS STOP	**ASCII READ COMMAND**	#GET MODBUS STOP<CR> #GMBSTOP<CR> Result: #GMBSTOP:<MBStop><CR>	ASCII	
	TX	#GET MODBUS STOP<CR>		
	RX	#1,GMBSTOP:ONE<CR>		
		Current stopbit(s):ONE		
Shows the current configured parity of the serial interface. MBParity NONE: no parity EVEN: even parity ODD: odd parity				
GET MODBUS PARAMS	**ASCII READ COMMAND**	#GET MODBUS PARAMS<CR> #GMBPARAMS<CR> Result: #GMBPARAMS:<MBUnitDec>,<MBFLASHDec>,<MBUnitHex>,<MBFLASHHex>, <MBBaudrateDec>,<MBBaudrateHex>,<MBParity>,<MBStops><CR>	ASCII	
	TX	#GET MODBUS PARAMS<CR>		
	RX	#1,GMBPARAMS:1,0x1,1,0x1,57600,0xE100,NONE,ONE<CR>		
		Current MODBUS unit ID used:1		
		Current MODBUS unit ID in FLASH:1		
		Current baudrate in FLASH:57600		
		Current parity in FLASH:NONE		
		Current stopbit(s) in FLASH:ONE		
Returns the complete settings for serial interface				

SYSTEM COMMANDS				
RESET	**ASCII WRITE COMMAND**	#RESET<CR> #RST<CR> Result: #OK<CR>	ASCII	NO
	TX	#RESET<CR>		
	RX	N/A		
Executes a software reset (Reboot) of the module.				
FACTORY RESET	**ASCII WRITE COMMAND**	#FACTORY RESET<CR> #FRST<CR> Result: #OK<CR>	ASCII	NO
	TX	#FACTORY RESET<CR>		
	RX	N/A		

7.5 RESI-xxx-SIO SERIAL PARAMETERS

Normally you select the serial parameters via DIP switch for fast use of the modules. But in special cases you will need a different setup for the serial interface. Please find all information here, how you can change the serial setup via MODBUS/RTU or ASCII commands.

HINT: This commands are only valid for the ULTRA SLIM IOs with serial RS232 or RS485 interface and for the BIG IOs with RS485 interface.

7.5.1 ULTRA SLIM IOs: Howto change the UnitID of the IO module

When DIP switch #4:FD is set to ON, the module always communicates with the Unit ID 255. When you switch this DIP switch to OFF, the module will use the internal Unit ID from the FLASH memory.

You can set this Unit ID either with this MODBUS register:

UNIT_ID	3x65222 4x65222 I:65221	1,0x0001 B:00 01		N/A:NO CHANGE	UINT16 R/W	NO
		UNIT ID:1				
If the host reads this register, the current programmed unit ID is returned. All values above unit ID 255 define also the unit ID 255. If the host write a new value into this register, the new value will be stored in the FLASH as the new unit ID. The new unit ID is activated after a power off/power on cycle or a software reboot of the module. The host can execute a reboot in writing to the register RESET SYSTEM. NOTE:DIP switch 4 must set to OFF to activate this unit ID, otherwise the unit ID is 255.						

or you use this ASCII command:

SET MODBUS ADDRESS	**ASCII WRITE COMMAND**	#SET MODBUS ADDRESS:<UNITID><CR> #SETMBADR:<UNITID><CR> Result: #OK<CR>	ASCII	NO
	UNITID	1		
	TX	#SET MODBUS ADDRESS:1<CR>		
	RX	N/A		
Redefines the unit ID of the module. This change will affect the MODBUS/RTU communication immediately. As a Unit IO you can use the values 0dec to 255dec. HINT: The new settings are activated after a system reboot or power off on cycle!				

After you changed the Unit ID you have to restart the module to make the changes effective. You can also use the MODBUS register for resetting the module:

SOFTWARE RESET						
RESET	1x06001 2x06001 I:6000	0,0x00 B:00		N/A:NO CHANGE	BIT R/W	NO
Performs a software reset, whenever 1 is written to this register. If the host writes to this register 1, the module executes a soft reset (reboot).						
RESET	3x06001 4x06001 I:6000	0,0x0000 B:00 00		N/A:NO CHANGE	UINT16 R/W	NO
Performs a software reset, whenever 1 is written to this register. If the host writes to this register 1, the module executes a soft reset (reboot).						

Or you use the ASCII command:

SYSTEM COMMANDS				
RESET	**ASCII WRITE COMMAND**	#RESET<CR> #RST<CR> Result: #OK<CR>	ASCII	NO
	TX	#RESET<CR>		
	RX	N/A		
Executes a software reset (Reboot) of the module.				

Remember, only if the DIP Switch #4 FD=OFF, you can use your new UnitID. Otherwise the module communicates with UnitID 255.

7.5.2 ULTRA SLIM IOs: Howto change the parity+stopbits of the IO module

Usually the IO module communicates with no parity and one stopbit. But you can change this behaviour:
You can set the parity and the stop bits with this MODBUS register:

PARITY	3x65225 4x65225 I:65224	0,0x0000 B:00 00		N/A:NO CHANGE	UINT16 R/W	NO
		NO PARITY		SELECT PARITY		
If the register is read out, the currently set parity of the serial interface is returned. Writing a value to this register will change the new parity in FLASH. This will only take effect after a restart of the module. This can be triggered by writing to the RESET SYSTEM register. Parity values are 0: no parity 1: even parity 2: odd parity						
STOP BITS	3x65226 4x65226 I:65225	1,0x0001 B:00 01		N/A:NO CHANGE	UINT16 R/W	NO
		TWO STOPBITS		SELECT STOPBITS		
f the register is read out, the currently set number of stop bits of the serial interface is returned. Writing a value to this register will change the new number of stop bits in the FLASH. This will only take effect after a restart of the module. This can be triggered by writing to the RESET SYSTEM register. Values for stop bits are 1: one stop bit 2: two stop bits						

or you use this ASCII command:

SET MODBUS PARITY	**ASCII WRITE COMMAND**	#SET MODBUS PARITY:<PARITY><CR> #SETMBPAR:<PARITY><CR> Result: #OK<CR>	ASCII	NO
	PARITY	NONE:NO PARITY		
	TX	#SET MODBUS PARITY:NONE<CR>		
	RX	N/A		
Sets a new parity for the serial interface. MBParity: NONE: no parity EVEN: even parity ODD: odd parity HINT: The new setup parameters will be active after a restart of the module.				
SET MODBUS STOPS	**ASCII WRITE COMMAND**	#SET MODBUS STOP:<STOPBIT><CR> #SETMBSTOP:<STOPBIT><CR> Result: #OK<CR>	ASCII	NO
	STOPBIT	ONE:ONE STOPBIT		
	TX	#SET MODBUS STOP:ONE<CR>		
	RX	N/A		
Sets a new amount of stop bits for the serial interface. MBStops ONE: one stop bit TWO: two stop bits HINT: The new setup parameters will be active after a restart of the module.				

After you changed the two settings you have to restart the module to make the changes effective. You can also use the MODBUS register for resetting the module:

SOFTWARE RESET						
RESET	1x06001 2x06001 I:6000	0,0x00 B:00		N/A:NO CHANGE	BIT R/W	NO
Performs a software reset, whenever 1 is written to this register. If the host writes to this register 1, the module executes a soft reset (reboot).						
RESET	3x06001 4x06001 I:6000	0,0x0000 B:00 00		N/A:NO CHANGE	UINT16 R/W	NO
Performs a software reset, whenever 1 is written to this register. If the host writes to this register 1, the module executes a soft reset (reboot).						

Or you use the ASCII command:

SYSTEM COMMANDS				
RESET	**ASCII WRITE COMMAND**	#RESET<CR> #RST<CR> Result: #OK<CR>	ASCII	NO
	TX	#RESET<CR>		
	RX	N/A		
Executes a software reset (Reboot) of the module.				

Remember, now the Module uses ALWAYS the new parity and stop bit setting in all communication modes!

7.5.3 ULTRA SLIM IOs: Howto change the baud rate of the IO module

Usually the IO module communicates with baud rates selected by the two DIP switches #1+#2. This will be:

#1	#2	
OFF	OFF	9600 baud
ON	OFF	19200 baud
OFF	ON	38400 baud
ON	ON	57600 baud or the new defined BAUDRATE from FLASH

But you can change the baud rate used with DIP switch setting ON,ON:
You can set the baud rate with this MODBUS register:

BAUD_RATE	3x65223 4x65223 I:65222	57600,0x0000E100 B:00 00 E1 00	38400	38400	UINT32 R/W	NO
		57600Bd		ENTER BAUD RATE		
This is the current configured baud rate in the FLASH For ULTRA SLIM IOs RESI-xxx-SIO: This baudrate is only used, if DIP switch mode DIP1=ON+DIP2=ON (BR) (default is 57600bd) For BIG IOs RESI-xxx-SIO: This baudrate is only used, if DIP switch mode DIP7=ON (PARAMETER) (default is 57600bd)						
Valid baud rates are: 300bd, 600bd, 900bd, 1200bd, 2400bd, 4800bd, 9600bd, 19200bd, 38400bd, 57600bd, 115200bd, 128000bd 230400bd, 250000bd, 256000bd						

or you use this ASCII command:

SET MODBUS BAUDRATE	**ASCII WRITE COMMAND**	#SET MODBUS BAUDRATE:<BAUD><CR> #SETMBBAUD:<BAUD><CR> Result: #OK<CR>	ASCII	NO
	BAUD	57600:57600BD		
	TX	#SET MODBUS BAUDRATE:57600<CR>		
	RX	N/A		
Sets a new baud rate in the FLASH For ULTRA SLIM IOs RESI-xxx-SIO: This baudrate is only used, if DIP switch mode DIP1=ON+DIP2=ON (BR) (default is 57600bd) For BIG IOs RESI-xxx-SIO: This baudrate is only used, if DIP switch mode DIP7=ON (PARAMETER) (default is 57600bd) The following baudrates are allowed: 300bd, 600bd, 900bd, 1200bd, 2400bd, 4800bd, 9600bd, 19200bd, 38400bd, 57600bd, 115200bd, 128000bd 230400bd, 250000bd, 256000bd HINT: The new setup parameters will be active after a restart of the module.				

After you changed the two settings you have to restart the module to make the changes effective. You can also use the MODBUS register for resetting the module:

SOFTWARE RESET						
RESET	1x06001 2x06001 I:6000	0,0x00 B:00		N/A:NO CHANGE	BIT R/W	NO
Performs a software reset, whenever 1 is written to this register. If the host writes to this register 1, the module executes a soft reset (reboot).						
RESET	3x06001 4x06001 I:6000	0,0x0000 B:00 00		N/A:NO CHANGE	UINT16 R/W	NO
Performs a software reset, whenever 1 is written to this register. If the host writes to this register 1, the module executes a soft reset (reboot).						

Or you use the ASCII command:

SYSTEM COMMANDS				
RESET	**ASCII WRITE COMMAND**	#RESET<CR> #RST<CR> Result: #OK<CR>	ASCII	NO
	TX	#RESET<CR>		
	RX	N/A		
Executes a software reset (Reboot) of the module.				

Remember, now the Module uses ALWAYS the new bau drate, if yu set the two DIP switches #1+#2 to ON,ON!

7.5.4 BIG IOs: Howto change the UnitID of the IO module

When DIP switches #1-#4:ADDRESS are all set to OFF, the module always communicates with the Unit ID from the FLASH. Otherwise the module uses the UnitIDs 1..15.

You can set this Unit ID either with this MODBUS register:

UNIT_ID	3x65222 4x65222 I:65221	1,0x0001 B:00 01		N/A:NO CHANGE	UINT16 R/W	NO
		UNIT ID:1				
f the host reads this register, the current programmed unit ID is returned. All values above unit ID 255 define also the unit ID 255. If the host write a new value into this register, the new value will be stored in the FLASH as the new unit ID. The new unit ID is activated after a power off/power on cycle or a software reboot of the module. The host can execute a reboot in writing to the register RESET SYSTEM. NOTE:DIP switch 4 must set to OFF to activate this unit ID, otherwise the unit ID is 255.						

or you use this ASCII command:

SET MODBUS ADDRESS	**ASCII WRITE COMMAND**	#SET MODBUS ADDRESS:<UNITID><CR> #SETMBADR:<UNITID><CR> Result: #OK<CR>	ASCII	NO
	UNITID	1		
	TX	#SET MODBUS ADDRESS:1<CR>		
	RX	N/A		
Redefines the unit ID of the module. This change will affect the MODBUS/RTU communication immediately. As a Unit IO you can use the values 0dec to 255dec. HINT: The new settings are activated after a system reboot or power off on cycle!				

After you changed the Unit ID you have to restart the module to make the changes effective. You can also use the MODBUS register for resetting the module:

SOFTWARE RESET						
RESET	1x06001 2x06001 I:6000	0,0x00 B:00		N/A:NO CHANGE	BIT R/W	NO
Performs a software reset, whenever 1 is written to this register. If the host writes to this register 1, the module executes a soft reset (reboot).						
RESET	3x06001 4x06001 I:6000	0,0x0000 B:00 00		N/A:NO CHANGE	UINT16 R/W	NO
Performs a software reset, whenever 1 is written to this register. If the host writes to this register 1, the module executes a soft reset (reboot).						

Or you use the ASCII command:

SYSTEM COMMANDS				
RESET	**ASCII WRITE COMMAND**	#RESET<CR> #RST<CR> Result: #OK<CR>	ASCII	NO
	TX	#RESET<CR>		
	RX	N/A		
Executes a software reset (Reboot) of the module.				

Remember, only if the DIP Switches #1-#4:ADDRESS are all OFF, you can use your new UnitID. Otherwise the module communicates with the selected UnitID.

7.5.5 BIG IOs: Howto change the parity+stopbits of the IO module

Usually the IO module communicates with no parity and one stopbit. What kind of parity and stop bit setting the IO module is using, is defined by DIP switch #8: PARAMETER.
If this DIP switch is set to OFF, the IO module ALWAYS uses no parity and one stopbit!
If this DIP switch is set to ON, the IO module will use the settings from the FLASH memory!

You can set the parity and the stop bits with this MODBUS register:

PARITY	3x65225 4x65225 I:65224	0,0x0000 B:00 00		N/A:NO CHANGE	UINT16 R/W	NO
		NO PARITY		SELECT PARITY		
If the register is read out, the currently set parity of the serial interface is returned. Writing a value to this register will change the new parity in FLASH. This will only take effect after a restart of the module. This can be triggered by writing to the RESET SYSTEM register. Parity values are 0: no parity 1: even parity 2: odd parity						
STOP BITS	3x65226 4x65226 I:65225	1,0x0001 B:00 01		N/A:NO CHANGE	UINT16 R/W	NO
		TWO STOPBITS		SELECT STOPBITS		
If the register is read out, the currently set number of stop bits of the serial interface is returned. Writing a value to this register will change the new number of stop bits in the FLASH. This will only take effect after a restart of the module. This can be triggered by writing to the RESET SYSTEM register. Values for stop bits are 1: one stop bit 2: two stop bits						

or you use this ASCII command:

SET MODBUS PARITY	**ASCII** **WRITE** **COMMAND**	#SET MODBUS PARITY:<PARITY><CR> #SETMBPAR:<PARITY><CR> Result: #OK<CR>	ASCII	NO
	PARITY	NONE:NO PARITY		
	TX	#SET MODBUS PARITY:NONE<CR>		
	RX	N/A		
Sets a new parity for the serial interface. MBParity: NONE: no parity EVEN: even parity ODD: odd parity HINT: The new setup parameters will be active after a restart of the module.				
SET MODBUS STOPS	**ASCII** **WRITE** **COMMAND**	#SET MODBUS STOP:<STOPBIT><CR> #SETMBSTOP:<STOPBIT><CR> Result: #OK<CR>	ASCII	NO
	STOPBIT	ONE:ONE STOPBIT		
	TX	#SET MODBUS STOP:ONE<CR>		
	RX	N/A		
Sets a new amount of stop bits for the serial interface. MBStops ONE: one stop bit TWO: two stop bits HINT: The new setup parameters will be active after a restart of the module.				

After you changed the two settings you have to restart the module to make the changes effective. You can also use the MODBUS register for resetting the module:

SOFTWARE RESET						
RESET	1x06001 2x06001 I:6000	0,0x00 B:00		N/A:NO CHANGE	BIT R/W	NO
Performs a software reset, whenever 1 is written to this register. If the host writes to this register 1, the module executes a soft reset (reboot).						
RESET	3x06001 4x06001 I:6000	0,0x0000 B:00 00		N/A:NO CHANGE	UINT16 R/W	NO
Performs a software reset, whenever 1 is written to this register. If the host writes to this register 1, the module executes a soft reset (reboot).						

Or you use the ASCII command:

SYSTEM COMMANDS				
RESET	**ASCII** **WRITE** **COMMAND**	#RESET<CR> #RST<CR> Result: #OK<CR>	ASCII	NO
	TX	#RESET<CR>		
	RX	N/A		
Executes a software reset (Reboot) of the module.				

Remember, now the Module uses the new parity and stop bit setting only, if DIP switch #8=ON!

7.5.6 BIG IOs: Howto change the baud rate of the IO module

Usually the IO module communicates with baud rates selected by the three DIP switches #5+#6+#7. This will be:

DIP #7:BR2	DIP #6:BR1	DIP #5:BR0	MODBUS/RTU or ASCII baud rate
OFF	OFF	OFF	4800bd
OFF	OFF	ON	9600bd
OFF	ON	OFF	19200bd
OFF	ON	ON	38400bd
ON	OFF	OFF	57600bd
ON	OFF	ON	115200bd
ON	ON	OFF	230400bd
ON	ON	ON	256000bd

This baud rates and the parity NONE and ONE stop bit are used, if the DIP switch #8 is set to OFF.
But you can change the baud rate used with DIP switch #8 PARAMETER setting to ON. Remember, that you will also use the configured parity and stop bits from the FLASH memory!

You can set the baud rate with this MODBUS register:

BAUD_RATE	3x65223 4x65223 I:65222	57600,0x0000E100 B:00 00 E1 00	38400	38400	UINT32 R/W	NO
		57600Bd		ENTER BAUD RATE		
This is the current configured baud rate in the FLASH For ULTRA SLIM IOs RESI-xxx-SIO: This baudrate is only used, if DIP switch mode DIP1=ON+DIP2=ON (BR) (default is 57600bd) For BIG IOs RESI-xxx-SIO: This baudrate is only used, if DIP switch mode DIP7=ON (PARAMETER) (default is 57600bd)						
Valid baud rates are: 300bd, 600bd, 900bd, 1200bd, 2400bd, 4800bd, 9600bd, 19200bd, 38400bd, 57600bd, 115200bd, 128000bd 230400bd, 250000bd, 256000bd						

or you use this ASCII command:

SET MODBUS BAUDRATE	**ASCII WRITE COMMAND**	#SET MODBUS BAUDRATE:<BAUD><CR> #SETMBBAUD:<BAUD><CR> Result: #OK<CR>	ASCII	NO
	BAUD	57600:57600BD		
	TX	#SET MODBUS BAUDRATE:57600<CR>		
	RX	N/A		
Sets a new baud rate in the FLASH For ULTRA SLIM IOs RESI-xxx-SIO: This baudrate is only used, if DIP switch mode DIP1=ON+DIP2=ON (BR) (default is 57600bd) For BIG IOs RESI-xxx-SIO: This baudrate is only used, if DIP switch mode DIP7=ON (PARAMETER) (default is 57600bd) The following baudrates are allowed: 300bd, 600bd, 900bd, 1200bd, 2400bd, 4800bd, 9600bd, 19200bd, 38400bd, 57600bd, 115200bd, 128000bd 230400bd, 250000bd, 256000bd HINT: The new setup parameters will be active after a restart of the module.				

After you changed the two settings you have to restart the module to make the changes effective. You can also use the MODBUS register for resetting the module:

SOFTWARE RESET						
RESET	1x06001 2x06001 I:6000	0,0x00 B:00		N/A:NO CHANGE	BIT R/W	NO
Performs a software reset, whenever 1 is written to this register. If the host writes to this register 1, the module executes a soft reset (reboot).						
RESET	3x06001 4x06001 I:6000	0,0x0000 B:00 00		N/A:NO CHANGE	UINT16 R/W	NO
Performs a software reset, whenever 1 is written to this register. If the host writes to this register 1, the module executes a soft reset (reboot).						

Or you use the ASCII command:

SYSTEM COMMANDS				
RESET	**ASCII WRITE COMMAND**	#RESET<CR> #RST<CR> Result: #OK<CR>	ASCII	NO
	TX	#RESET<CR>		
	RX	N/A		
Executes a software reset (Reboot) of the module.				

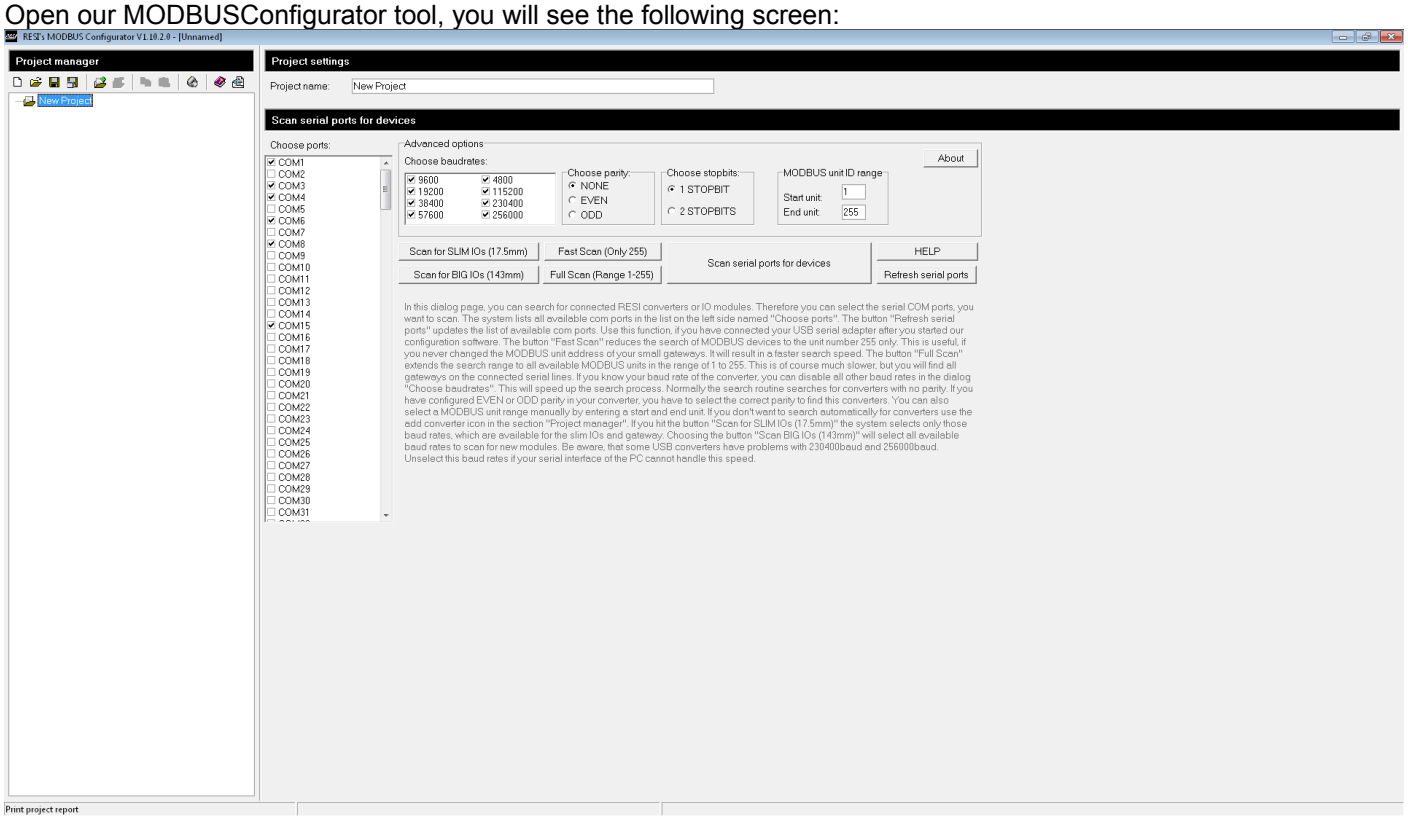
Remember, now the Module uses ALWAYS the new baud rate, the parity and the stop bits, if you set the DIP switch #8 to ON! If you set the DIP switch #8 to OFF, you will use the baud rate defined by DIP switch #5-#7 and the parity is always NONE and the stop bit is always ONE.

7.6 RESI's MODBUS Configurator

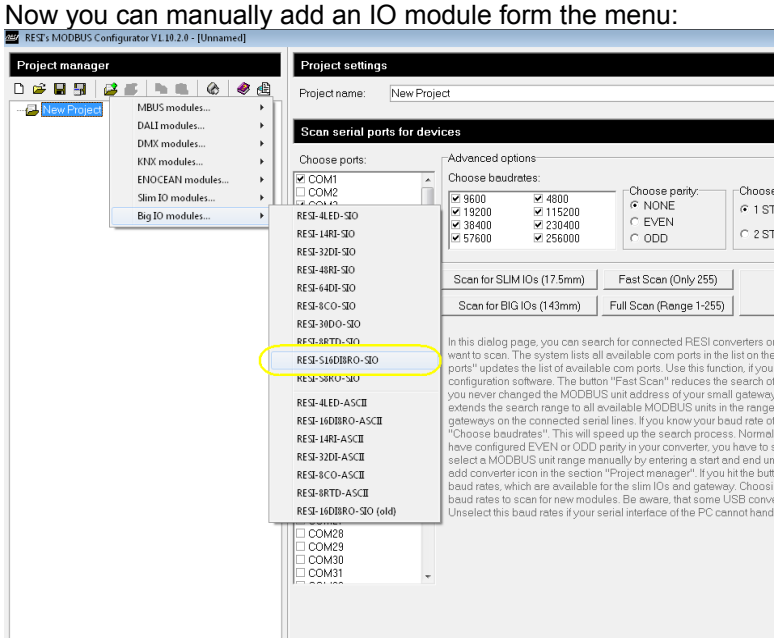
Almost all of our products can be used together with our MODBUSConfigurator software tool. You can configure and test the modules.

7.6.1 HOWTO manually establish a serial connection to the module

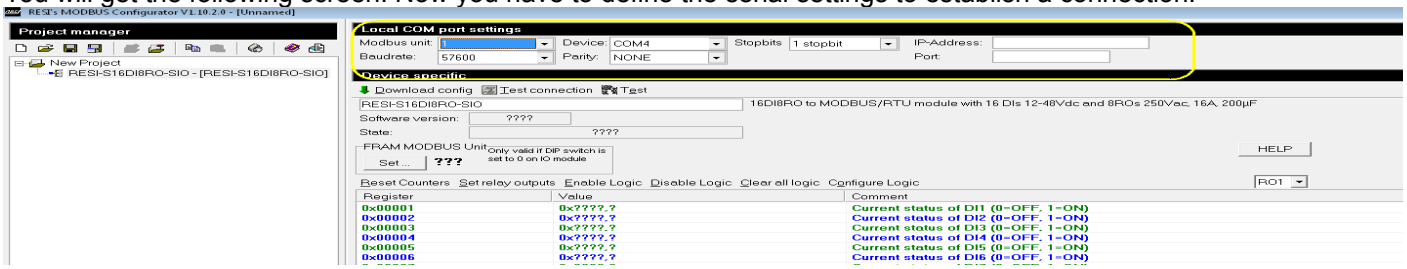
Open our MODBUSConfigurator tool, you will see the following screen:



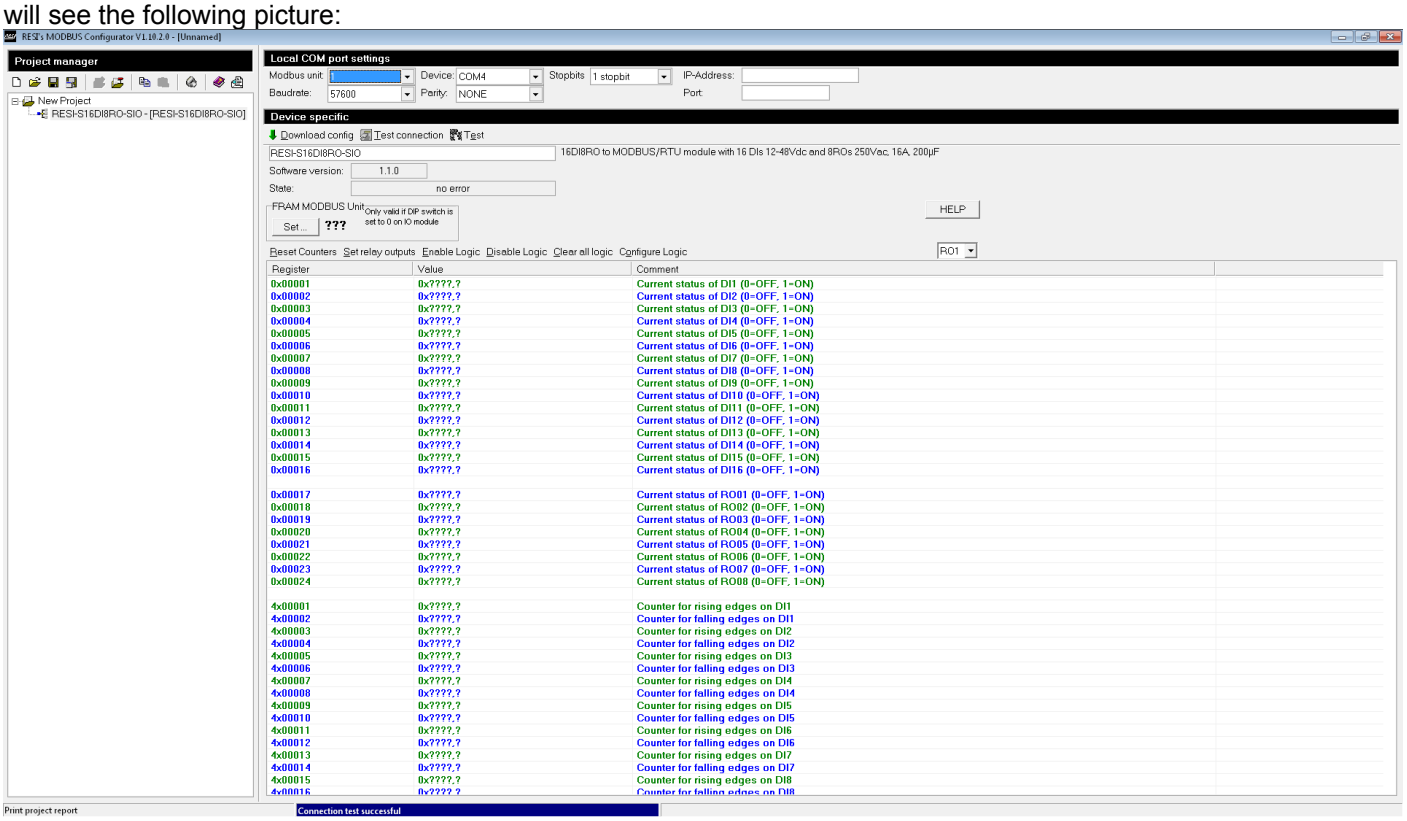
Now you can manually add an IO module form the menu:



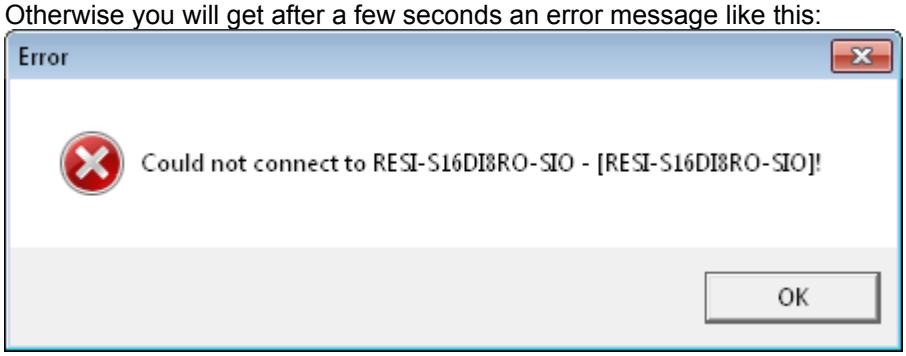
You will get the following screen. Now you have to define the serial settings to establish a connection:



Test the connection by pressing the button "Test connection". If you have successfully established the connection, you will see the following picture:

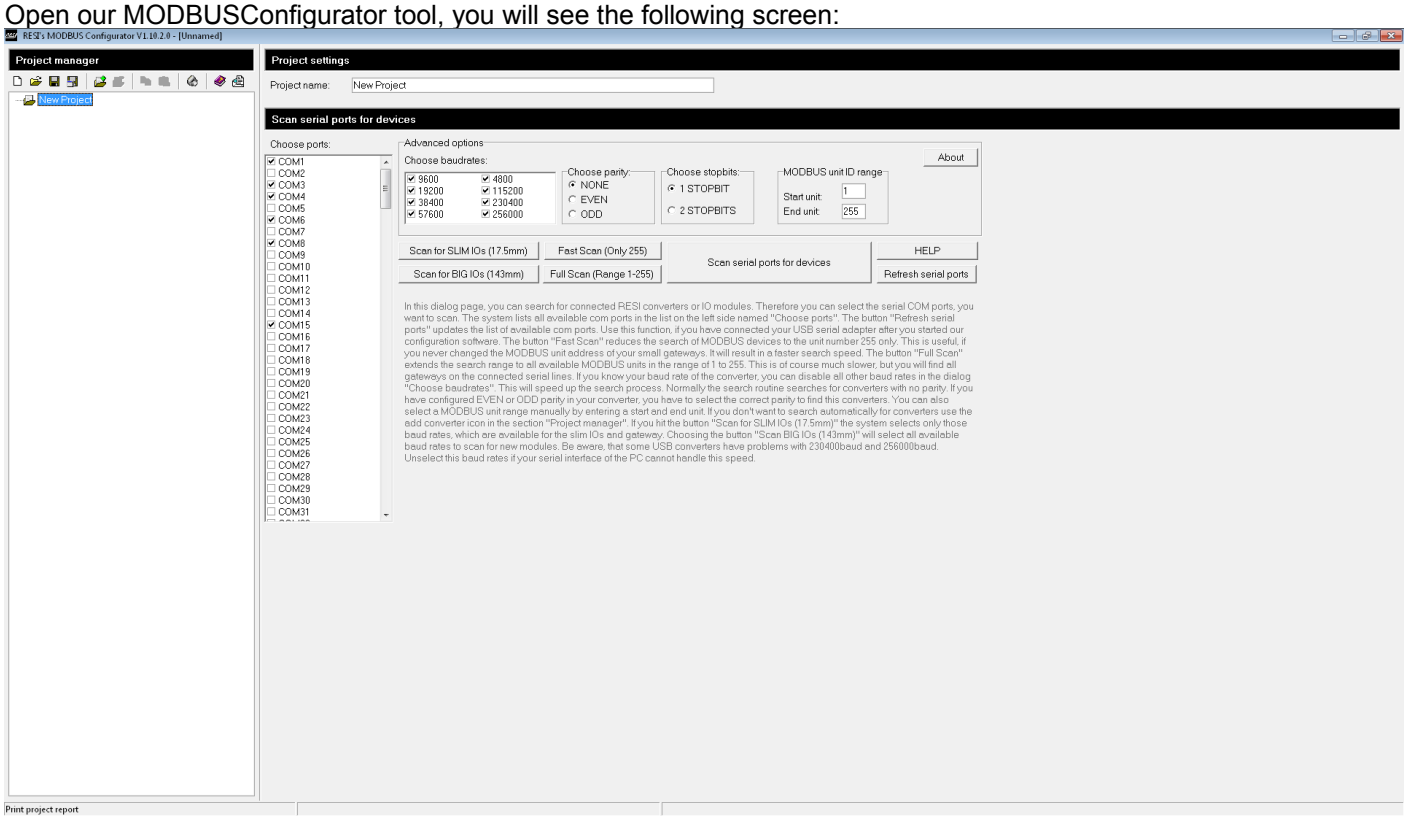


Otherwise you will get after a few seconds an error message like this:

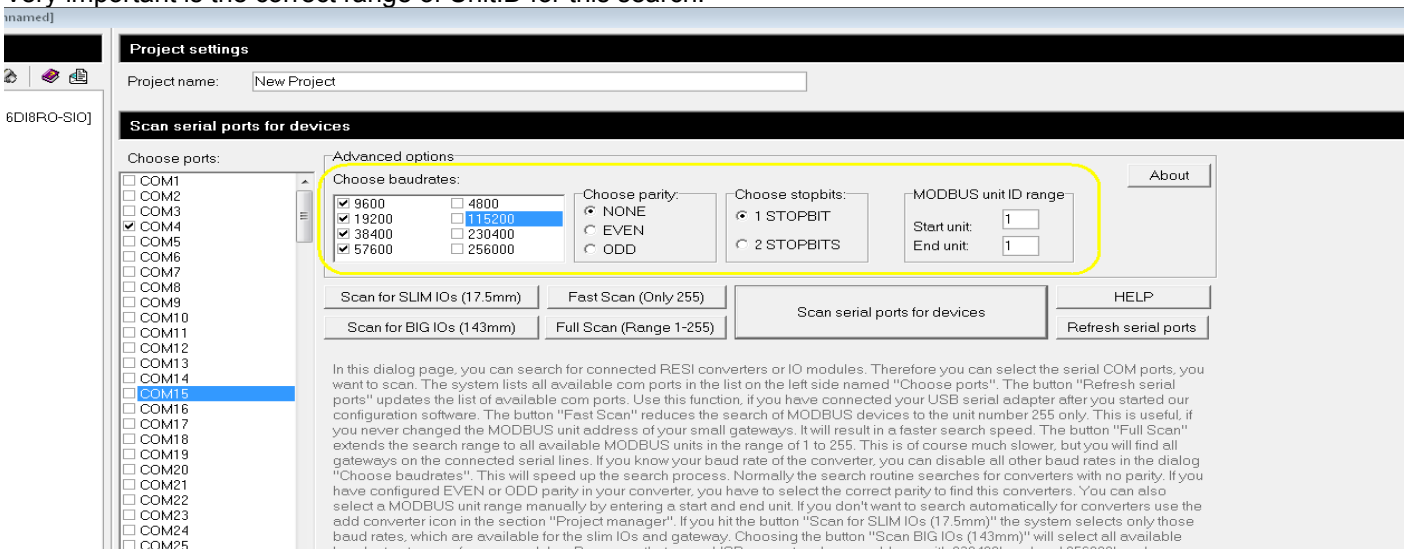


7.6.2 HOWTO search for serial modules

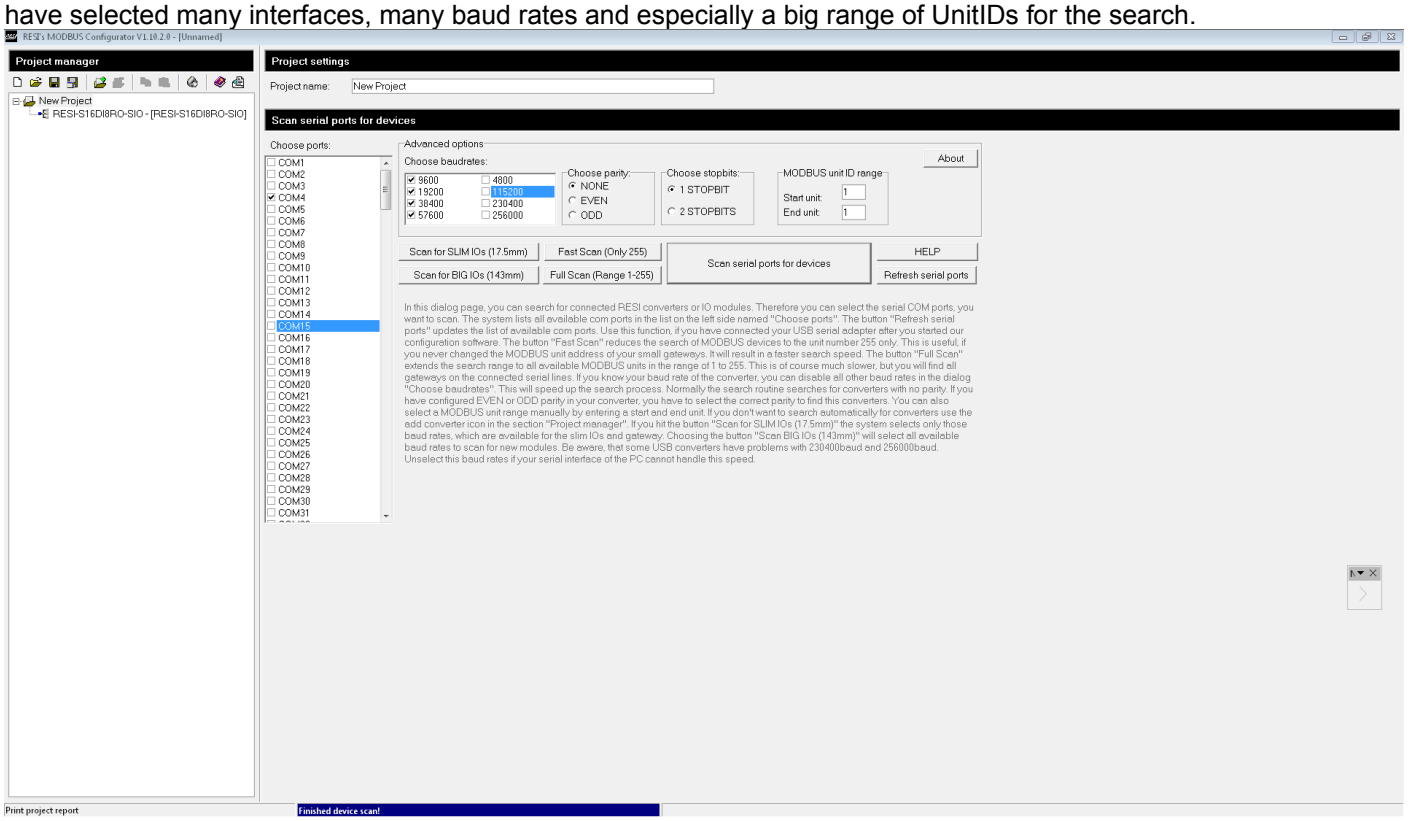
Open our MODBUSConfigurator tool, you will see the following screen:



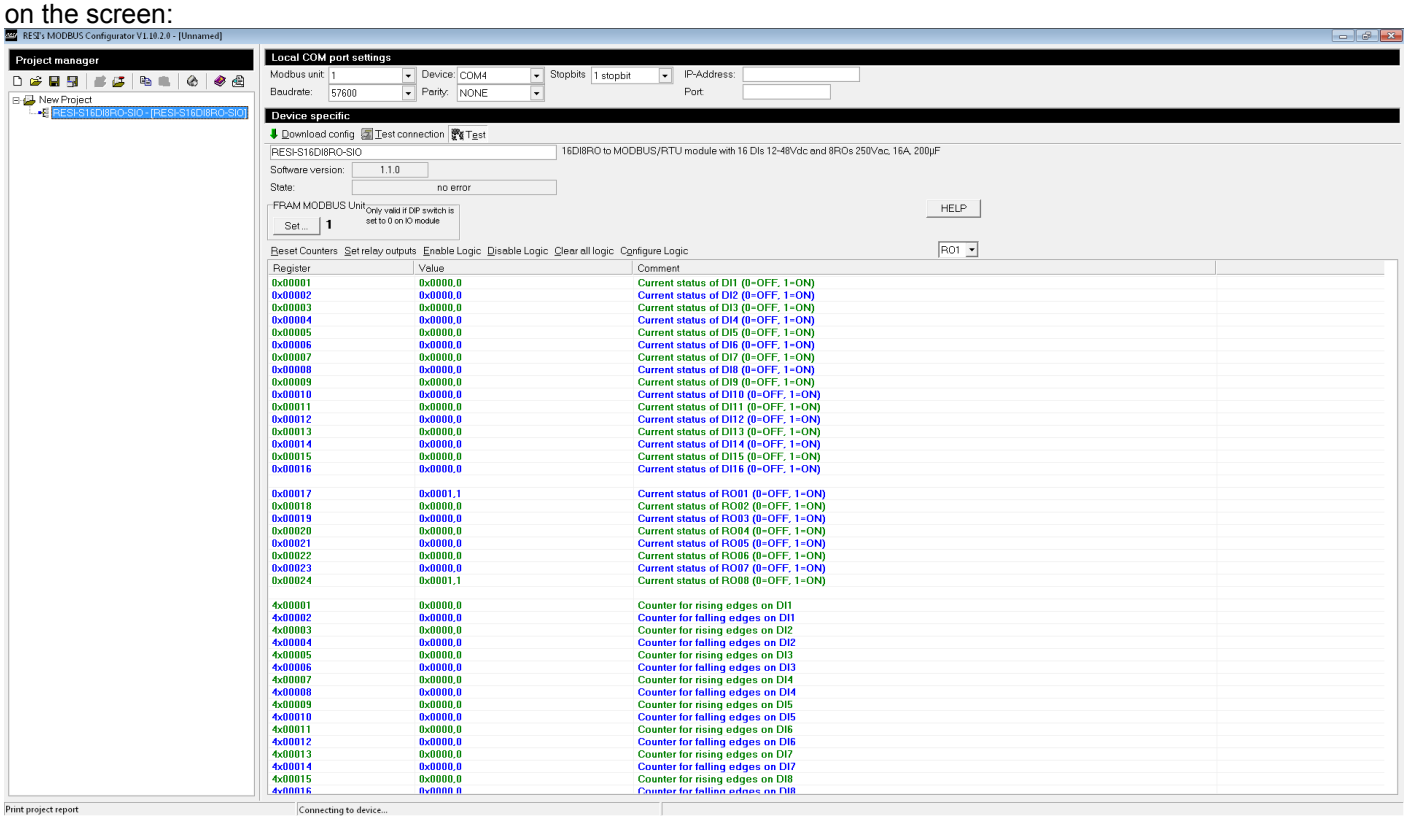
Select the correct serial interfaces, the correct baud rates, parity and stop bits for an automatic search for IO modules. Very important is the correct range of UnitID for this search:



Then click on the button "Scan serial ports for devices" to start the automatic search. This can last for minutes, if you have selected many interfaces, many baud rates and especially a big range of UnitIDs for the search.



Now select the module from the project tree and activate the test mode, you will see all actual values of your module on the screen:



You can also use now commands from then command bar to control the connected IO module.

8 Ethernet connection

Our Ethernet ULTRA SLIM IO modules offer an Ethernet interface.

8.1 Ethernet connection for ULTRA SLIM IO modules

The following drawings show the correct Ethernet connection for all of our SLIMIO products:

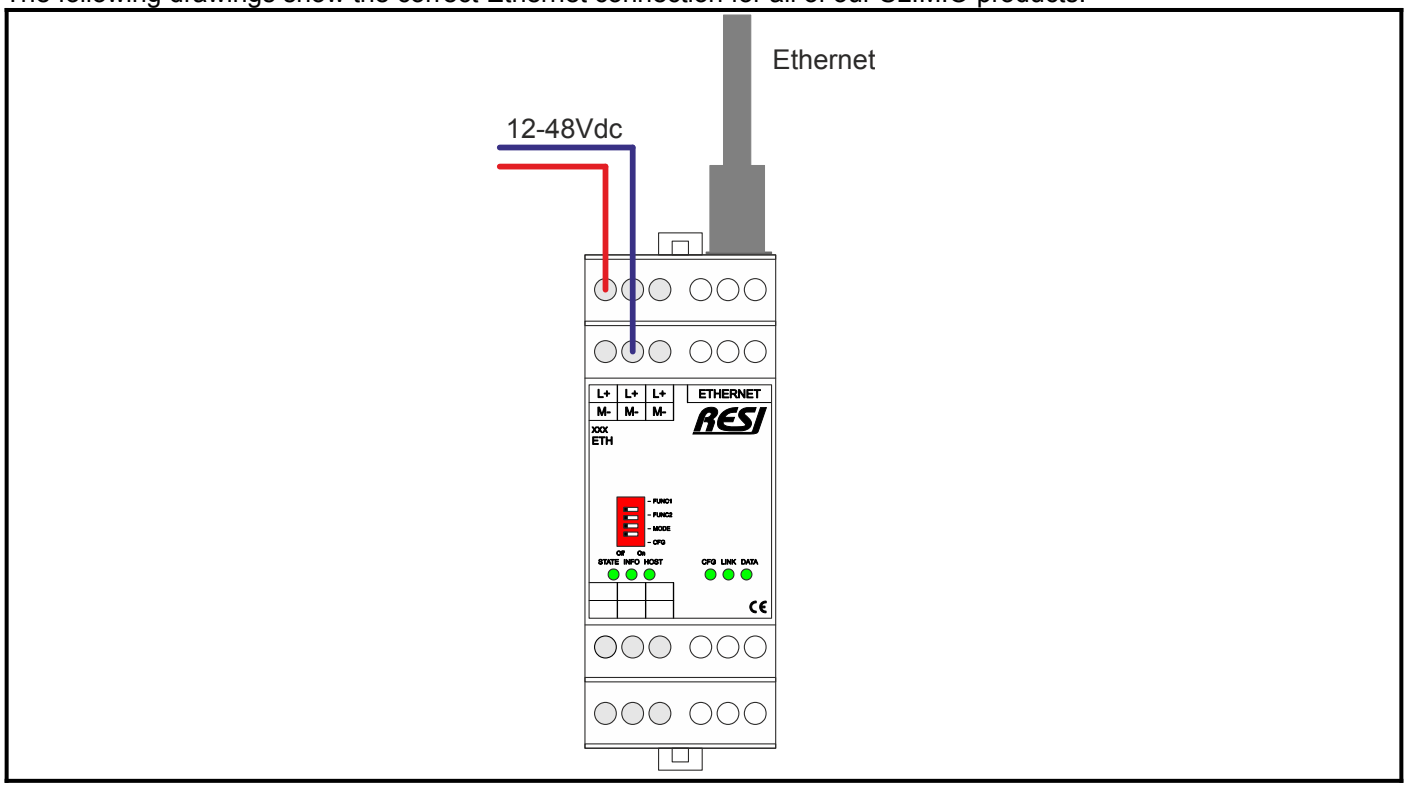


Figure: Ethernet connection for our ULTRA SLIM IO modules

8.2 RESI-xxx-ETH OPERATING MODES

The gateway basically supports two different operating modes:

- **TRANSPARENT MODE:** Bidirectional, transparent gateway between Ethernet socket data and IO module. All data arriving at the Ethernet socket are processed directly by the IO module. All data from the IO module is forwarded directly to the Ethernet socket. This mode is required for the ASCII protocol.

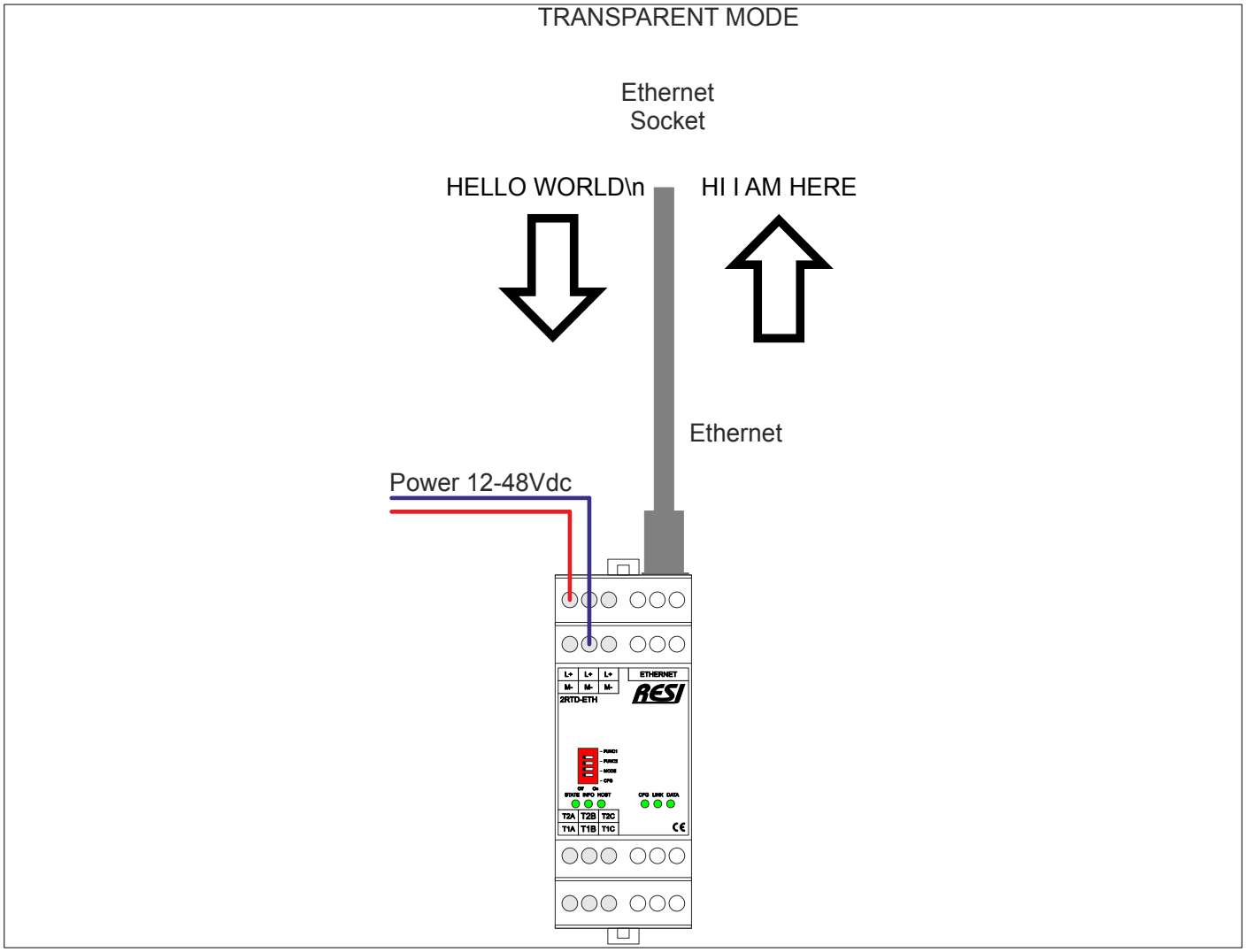


Figure: TRANSPARENT MODE for the RESI-xxx-ETH module

In this mode you can also use the **MODBUS/RTU protocol via Ethernet** to communicate with the IO module.

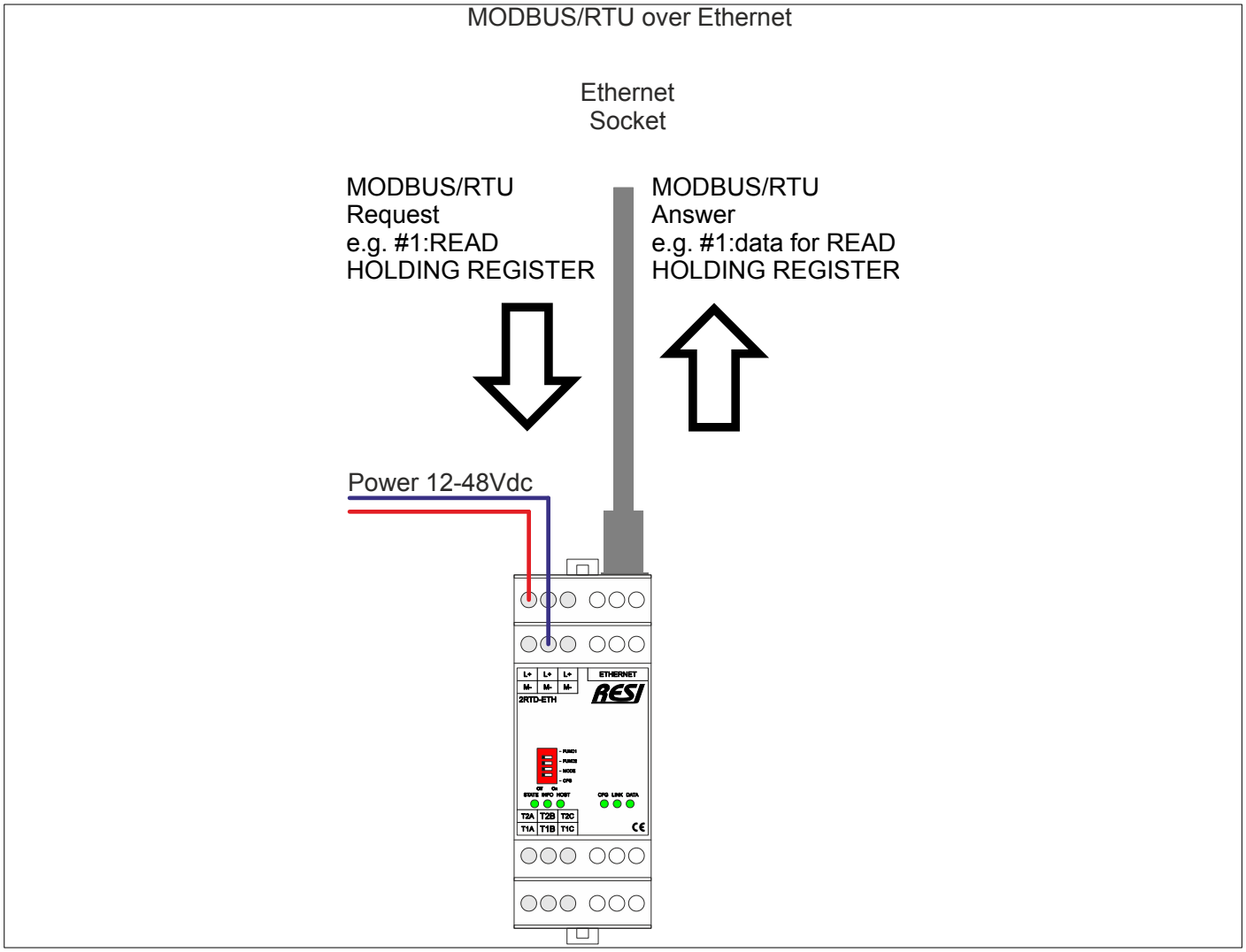


Figure: MODBUS/RTU via ETHERNET MODE for the RESI-xxx-ETH module

• **MODBUS/TCP server**: The module is a MODBUS/TCP server. A host with MODBUS/TCP master protocol can communicate directly with the IO module connected via Ethernet.

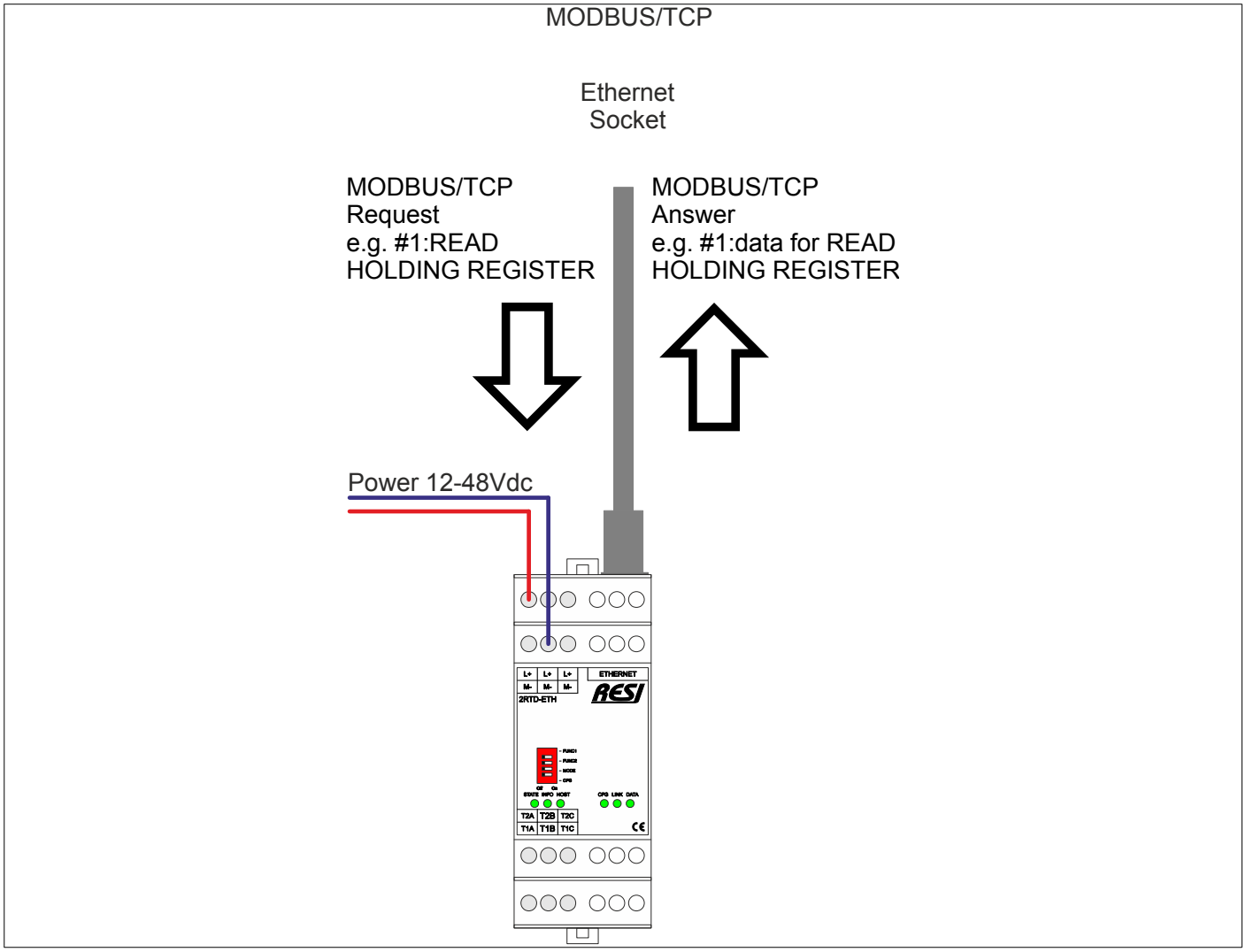


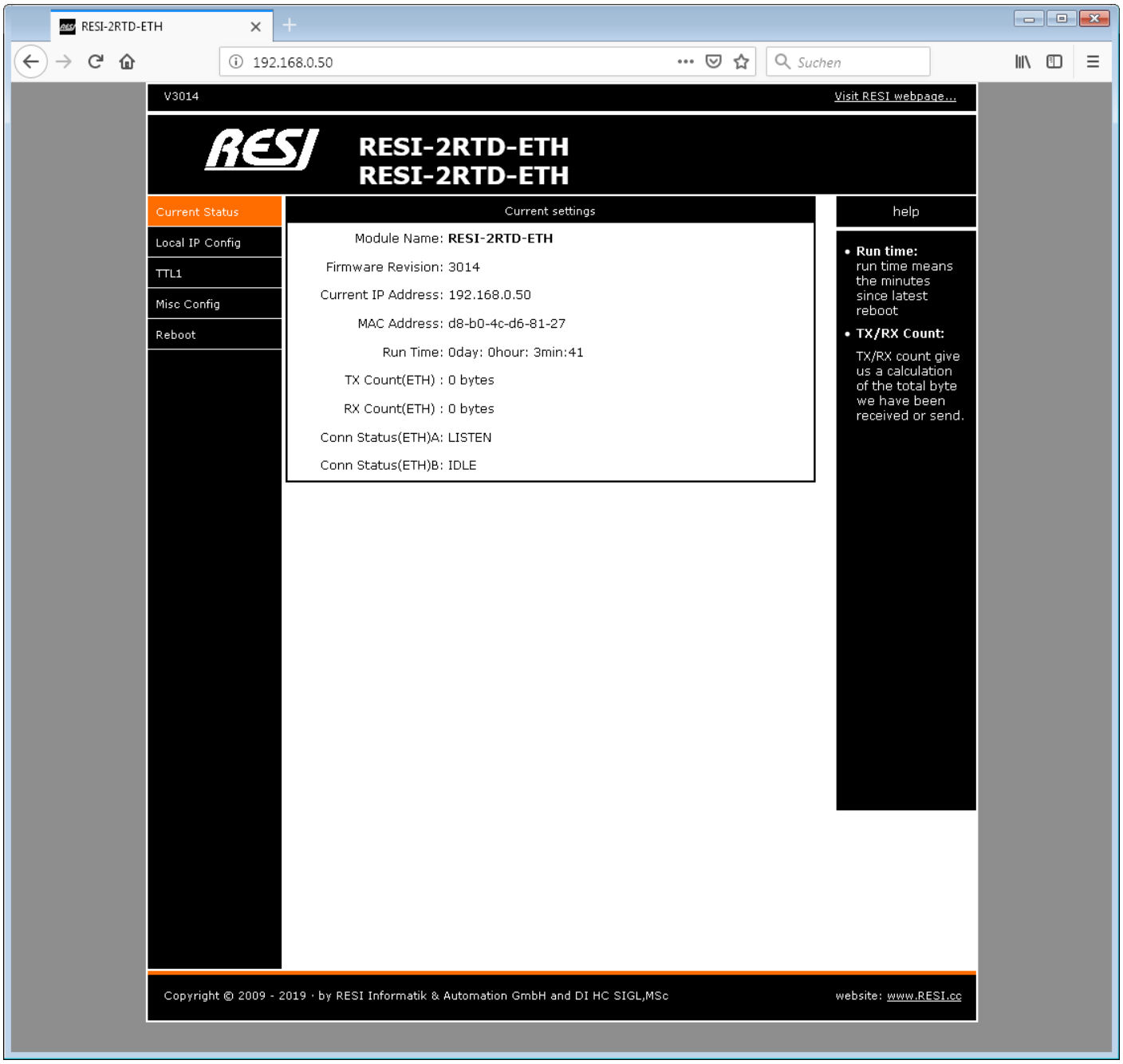
Figure: MODBUS/TCP SERVER MODE for the RESI-xxx-ETH gateway

8.3 RESI-xxx-ETH WEB CONFIGURATION

All of our RESI-xxx-ETH gateways have an integrated web server that configures basic access to the Ethernet interface and reads out MODBUS/TCP. To do this, open an Internet Explorer and enter the configured IP address of the selected gateway.

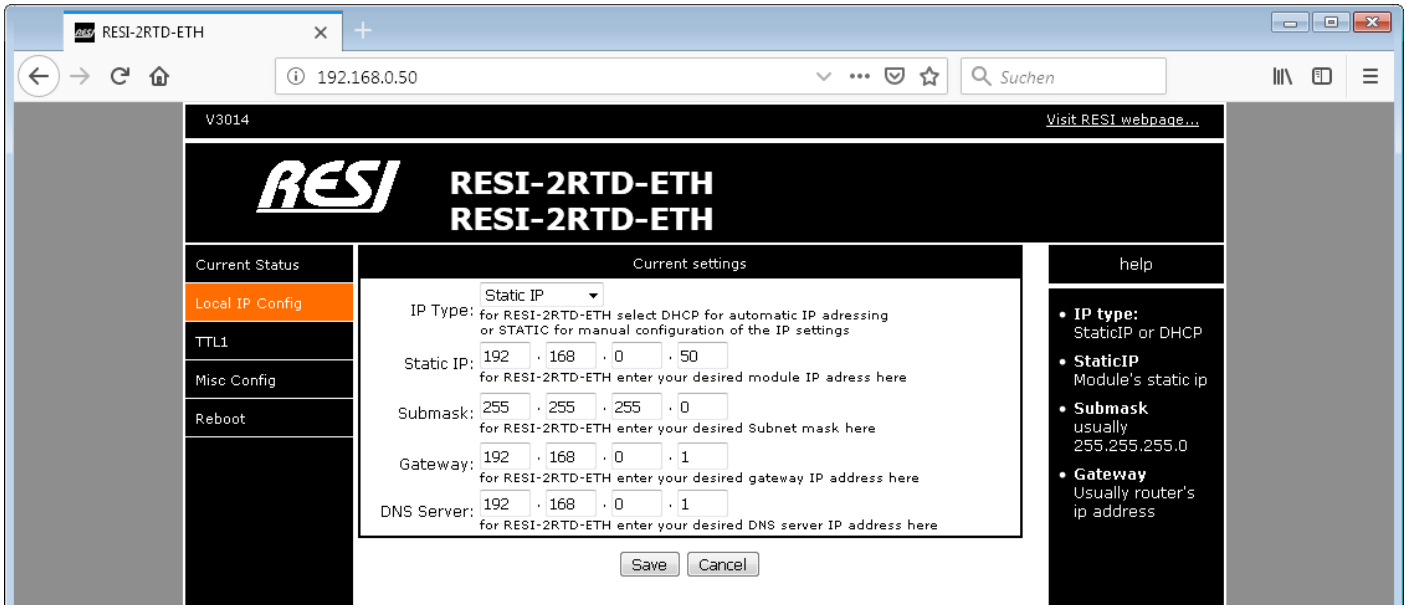
HINT: Please check the individual section of each IO module for the standard IP settings of our specific module.
The default user name is RESI and the default password is RESI.

You should see the following page:



8.3.1 How to set up the IP address

Select the "Local IP Config" page. Use the following mask to edit the IP settings:

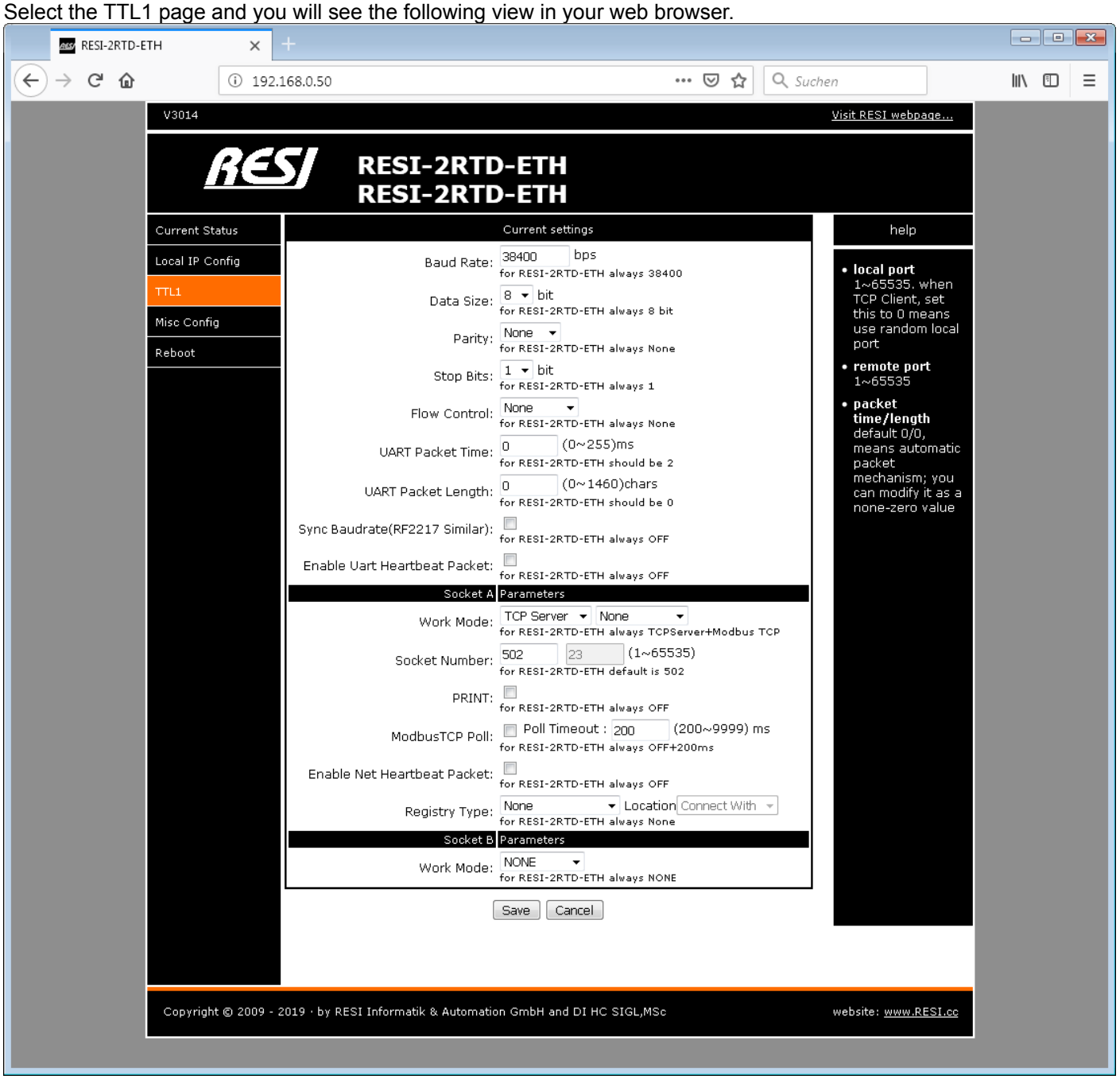


- **IP type**: Selection between STATIC IP for a static IP address or DHCP mode for an automatic assignment of the IP address.
- **Static IP**: Choose your desired IP address in IPv4 format
- **Submask**: Select your desired subnet mask in IPv4 format
- **Gateway**: Select your desired gateway IP address in IPv4 format
- **DNS server**: Select your desired DNS server IP address in IPv4 format

Click SAVE to save your data. But don't forget to restart the device for the new IP settings to take effect. If you have problems, set the CFG DIP switch to ON and restart the device. Wait for more than 30 seconds. The gateway resets to the factory settings with the IP standard settings defined above. Don't forget to set the CFG DIP switch back to OFF afterwards.

8.3.2 How to change the socket number

Select the TTL1 page and you will see the following view in your web browser.



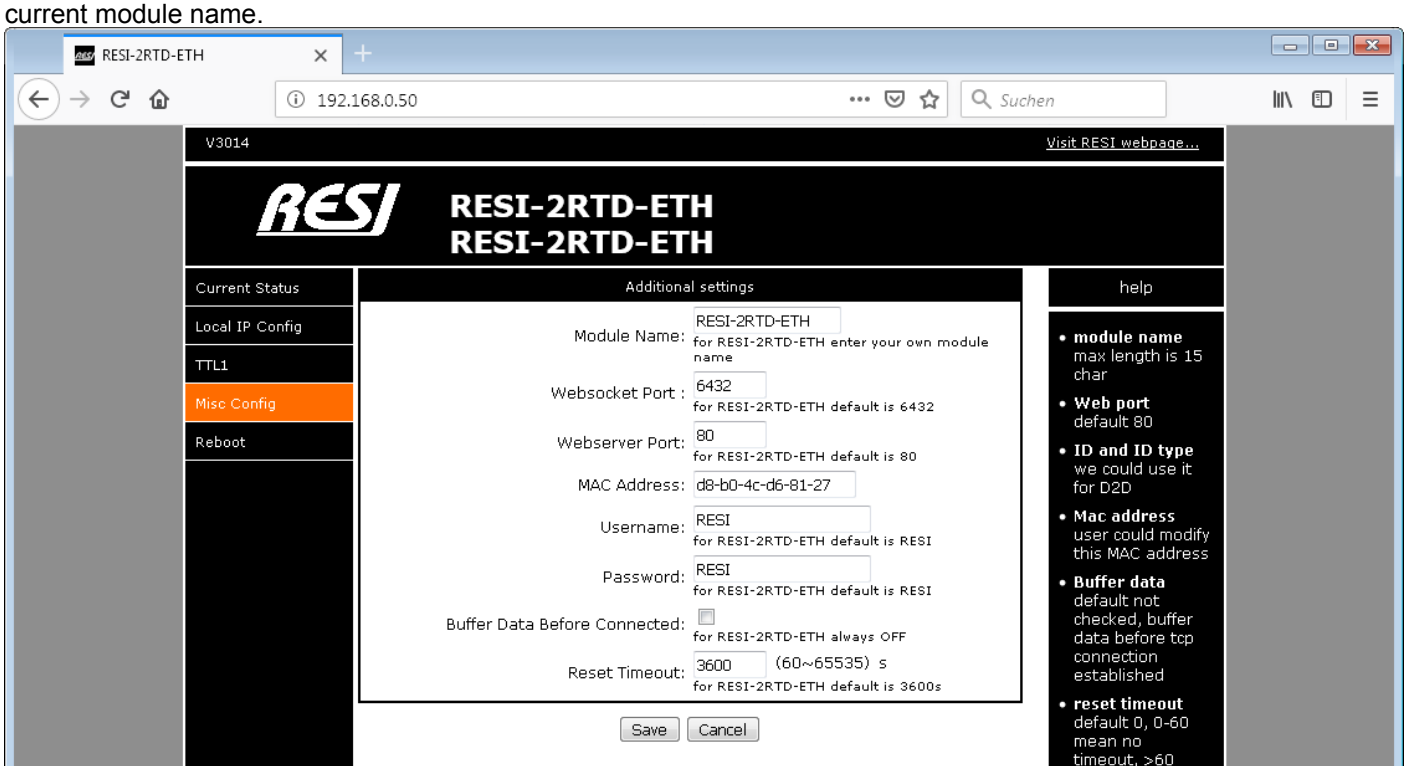
NOTE: Do not change the TTL communication parameters (e.g. baud rate, ...). You can lose the connection to the gateway!

- **Work mode**: Here you can select TCP Server/none if you want to communicate in transparent mode. All incoming data on the socket are forwarded directly to the IO module. If you want to use the internal MODBUS / TCP to MODBUS/RTU converter, you have to select TCP Server/ModbusTCP. If you select TCP-Server/none, you can also communicate with the MODBUS/RTU protocol over Ethernet or use the ASCII protocol.
- **Socket number**: Here you can select the desired socket number that you want to use for the Ethernet connection. The default value for our converters is 1024, for MODBUS/TCP 502.

Please leave the rest of the parameters unchanged. They are only suitable for experts!

8.3.3 How to change username and password

If you select the Misc config page, you will see the currently configured user name and password. You will also see the current module name.

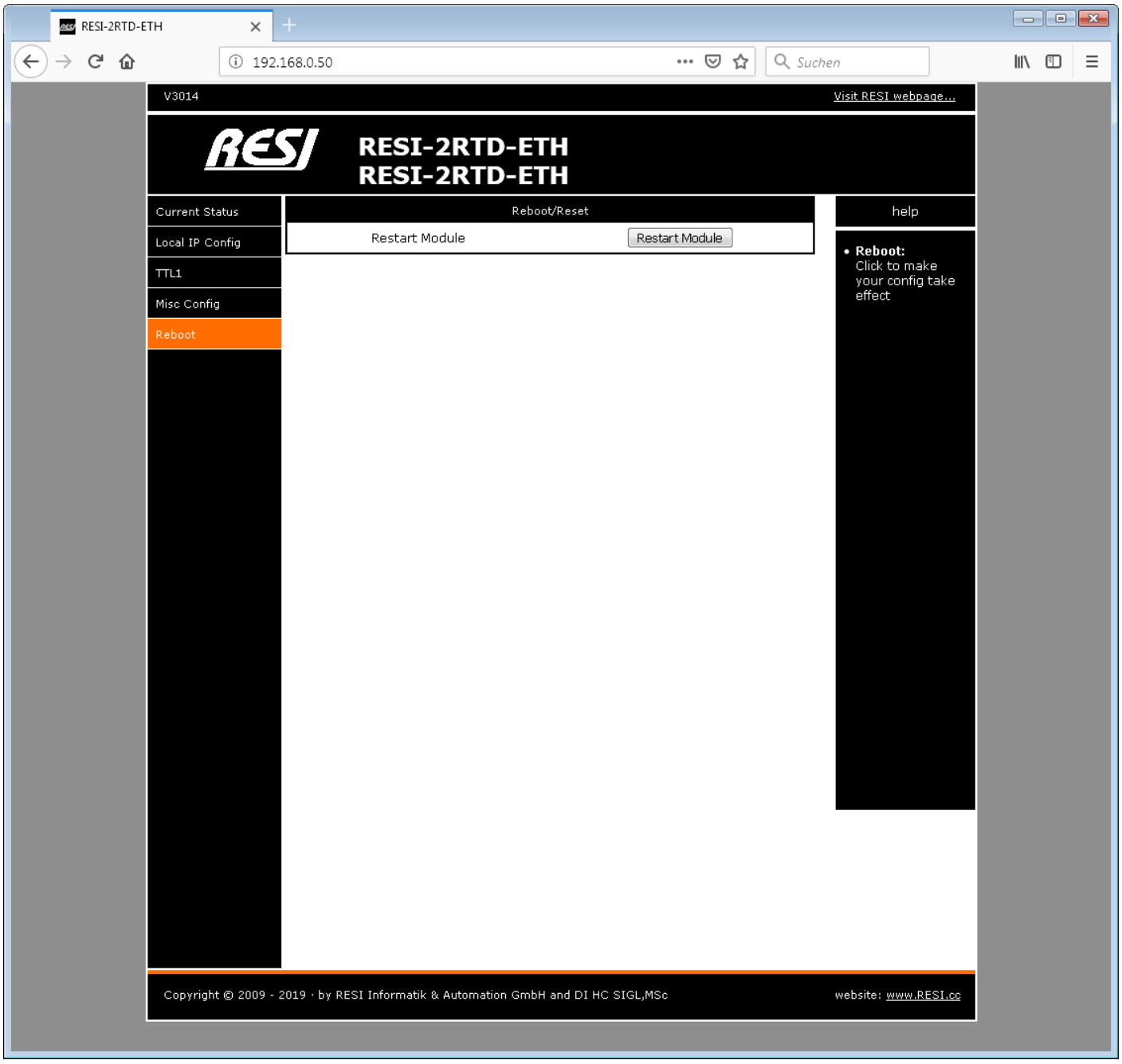


- Module name: Here you can enter a new module name. It is used for better identification if you have more than one gateway in your network.
- Username: Here you can enter a new user name for accessing the web configuration.
- Password: Here you can enter a new password for accessing the web configuration.

Don't forget to save the new settings with the SAVE button!
Please leave the rest of the parameters unchanged. These are only for experts!

8.3.4 How to restart the gateway via Ethernet

First select the Reboot page. Then select the Restart Module button to restart the software.

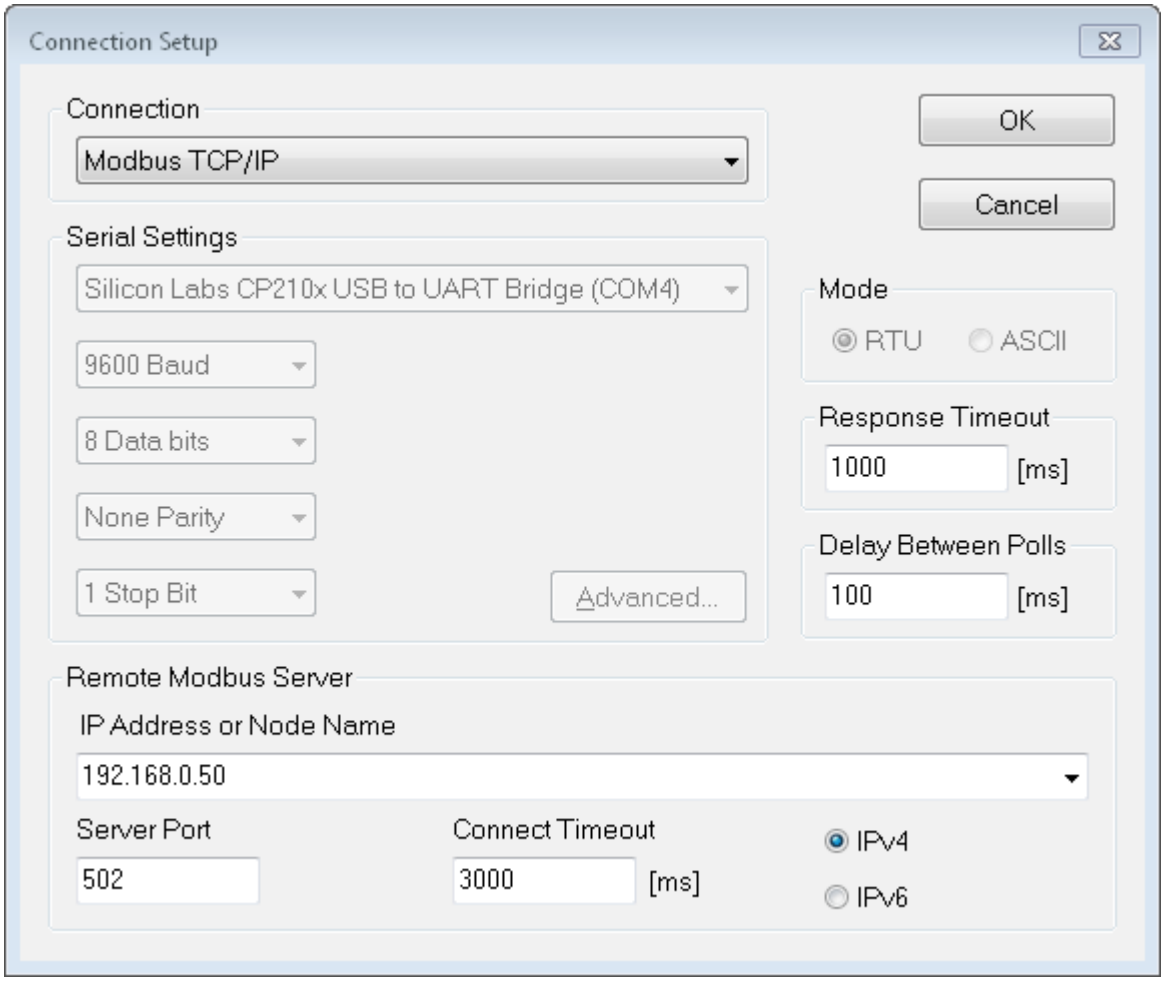


8.3.5 How to select the MODBUS / TCP server mode

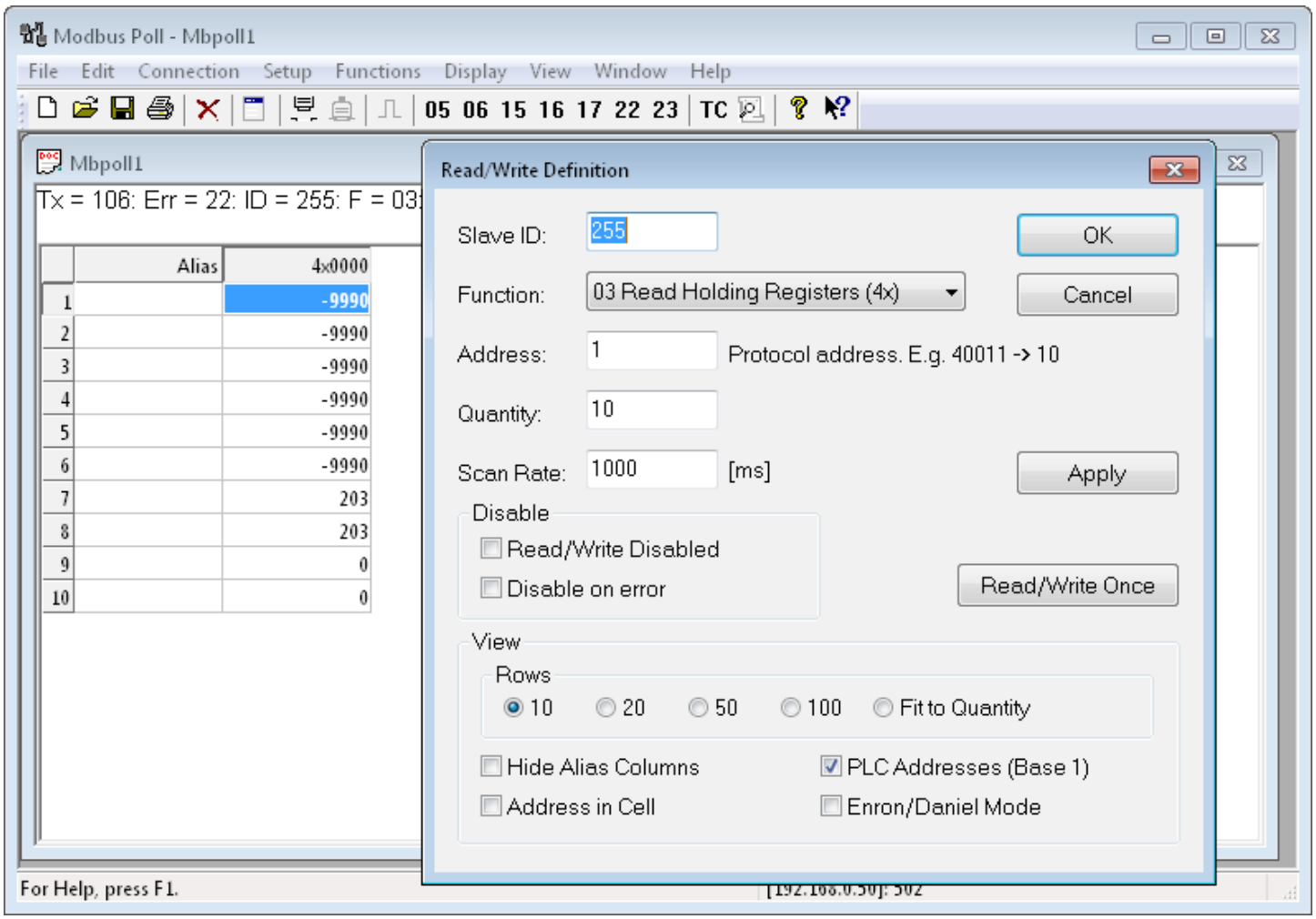
A gateway can be switched to one of the following states very quickly:

1. Activate DIP switch 4: CFG
2. Wait for about 30 seconds. The gateway LEDs flash very quickly
3. Deactivate all DIP switches

Now you have reset the factory settings to the IP standard settings and selected the MODBUS/TCP server mode.
To test your MODBUS/TCP server functionality, use the MODBUS Poll software shown here:



Now select the area of the MODBUS-holding register you want to display. Select the function Setup/Read-Write Definition .. and configure the following parameters. After you click OK, the updated values are displayed.

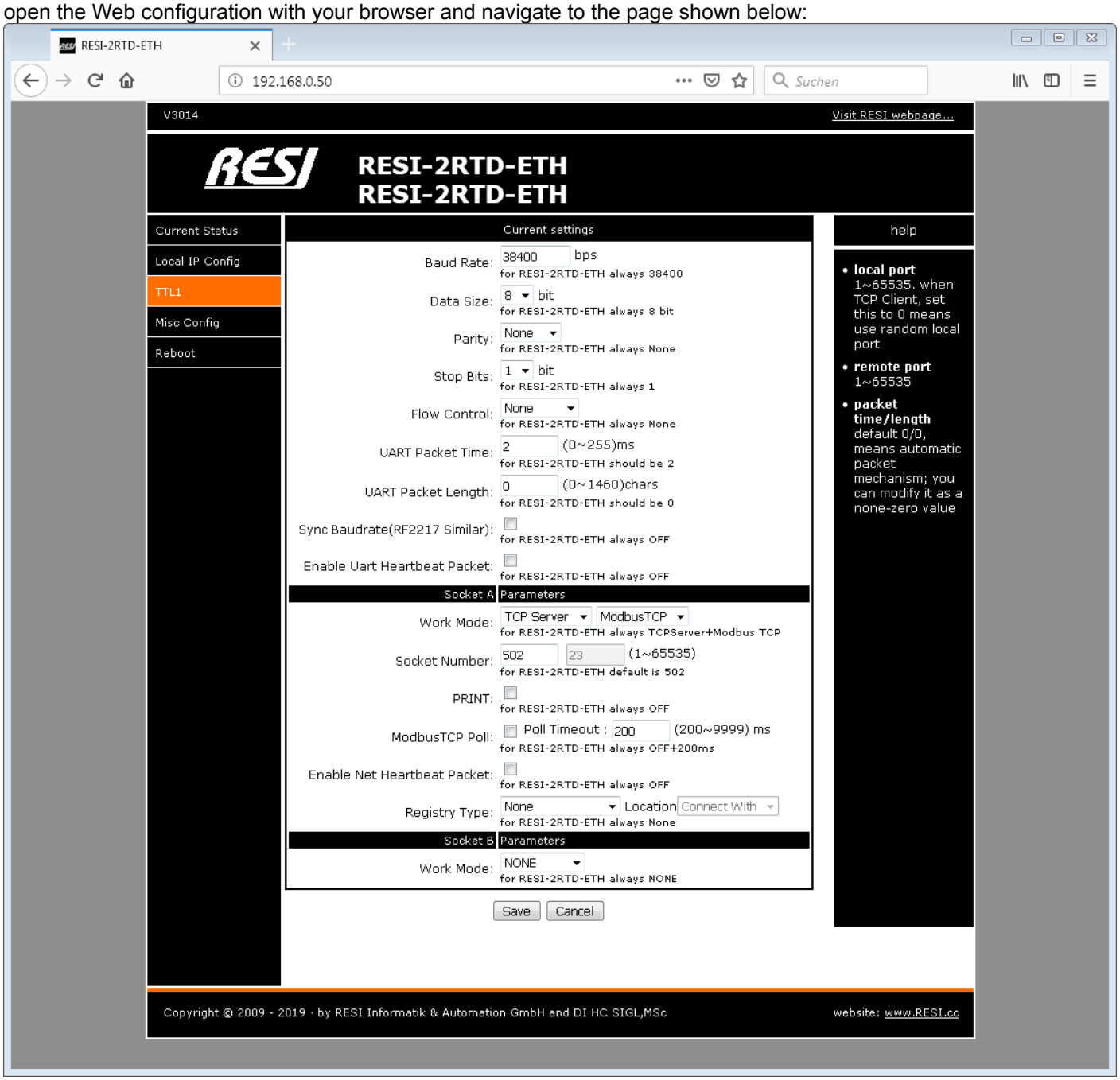


8.3.6 How to select the TRANSPARENT or MODBUS/RTU via ETHERNET mode

A gateway can be switched to one of the following states very quickly:

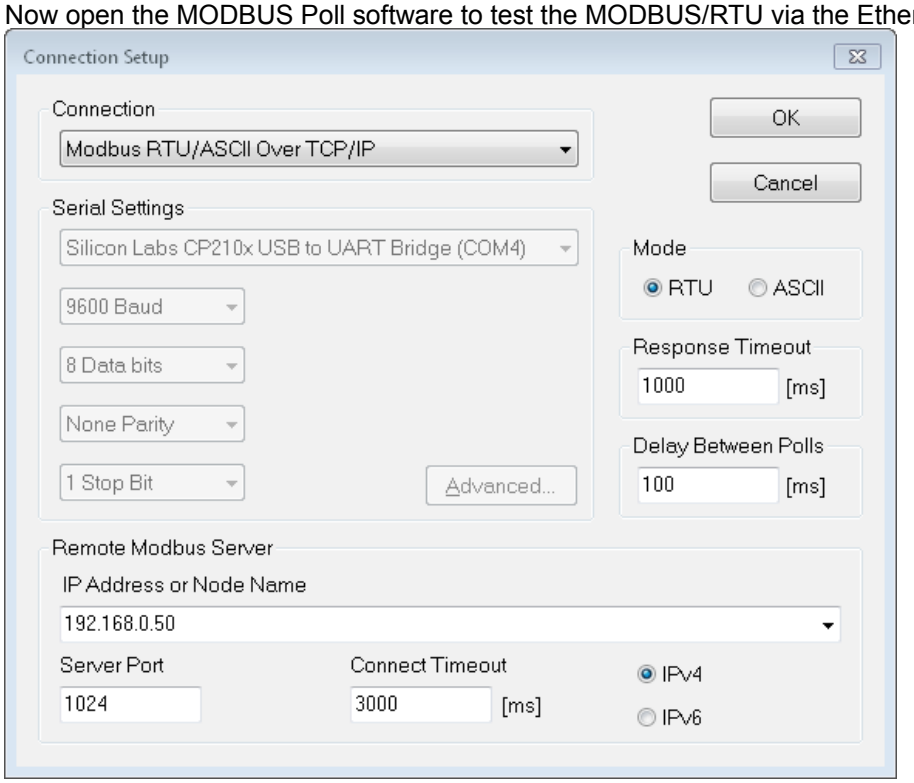
1. Activate DIP switch 4: CFG
2. Wait for about 30 seconds. The gateway LEDs flash very quickly
3. Deactivate all DIP switches

Now you have reset the factory settings to the IP standard settings and selected the MODBUS/TCP server mode. Now open the Web configuration with your browser and navigate to the page shown below:

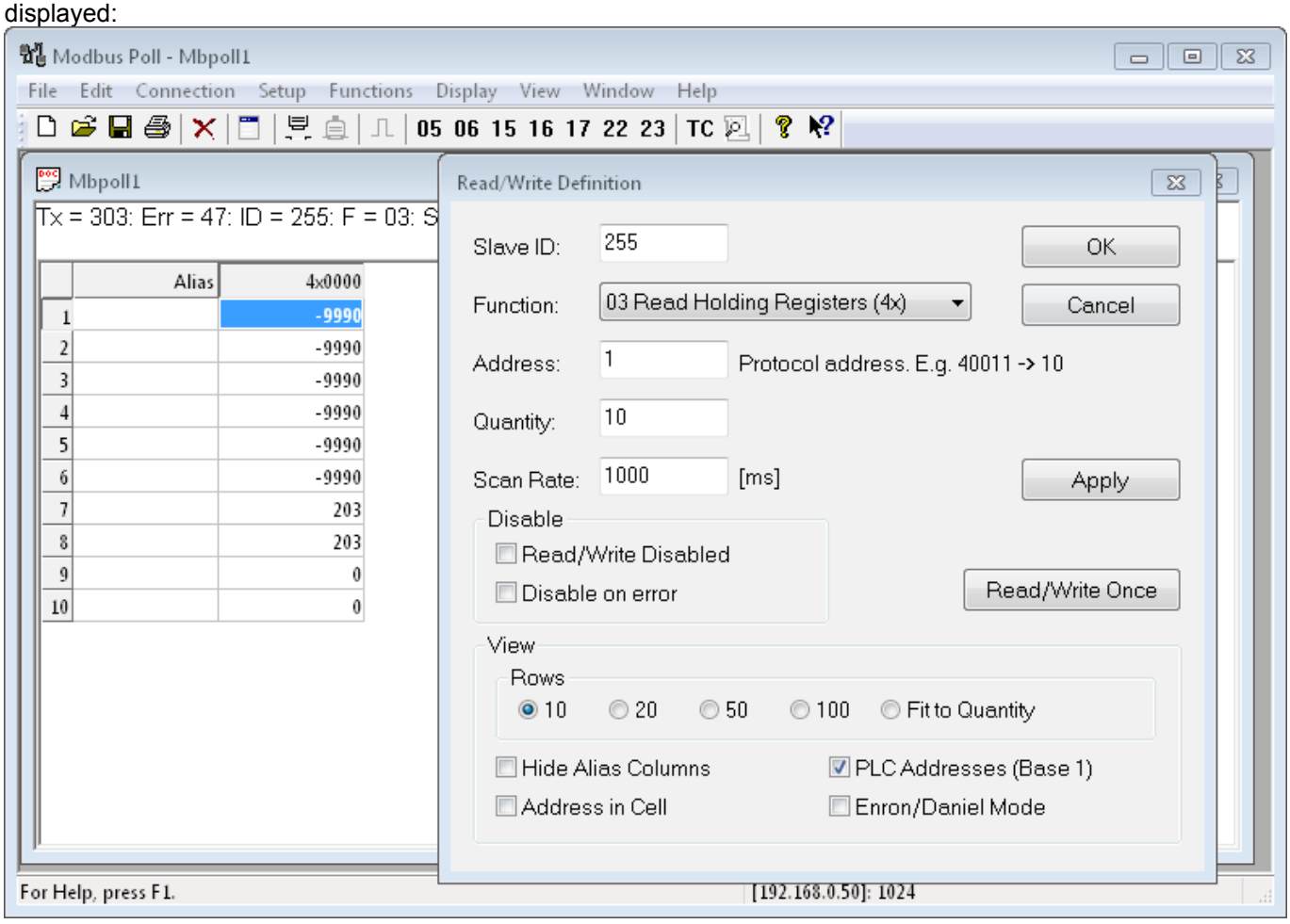


Now change the **Work Mode** from **MODBUS/TCP** to **None** and adapt the socket number to your needs. (e.g. 1024). Click SAVE and restart the module with the RESTART button. Now the module works in TRANSPARENT mode.

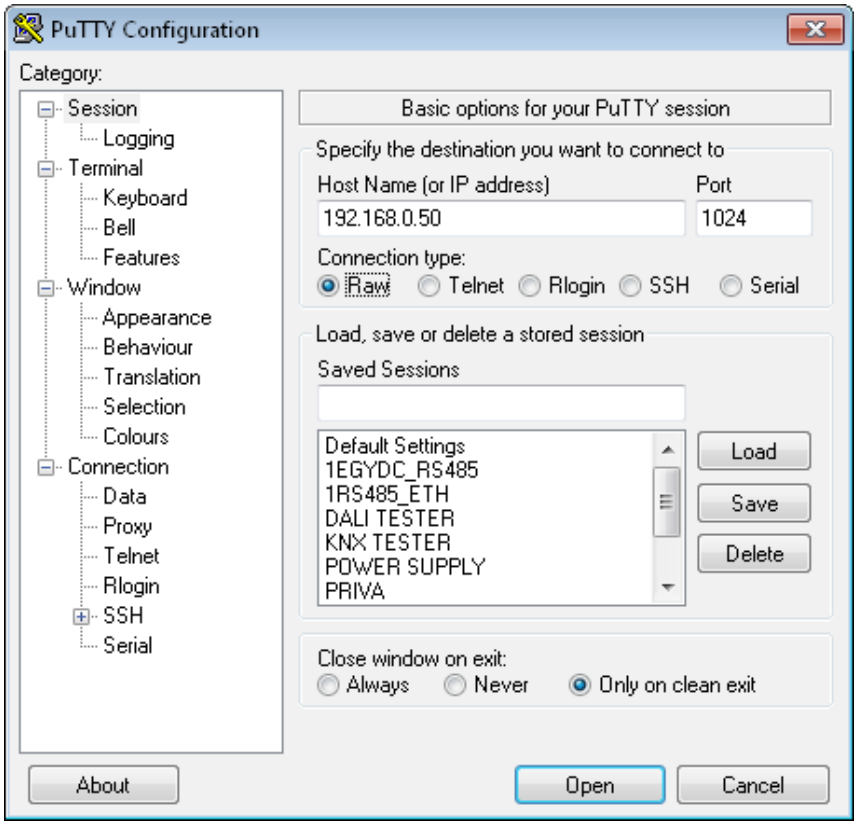
Now open the MODBUS Poll software to test the MODBUS/RTU via the Ethernet mode:



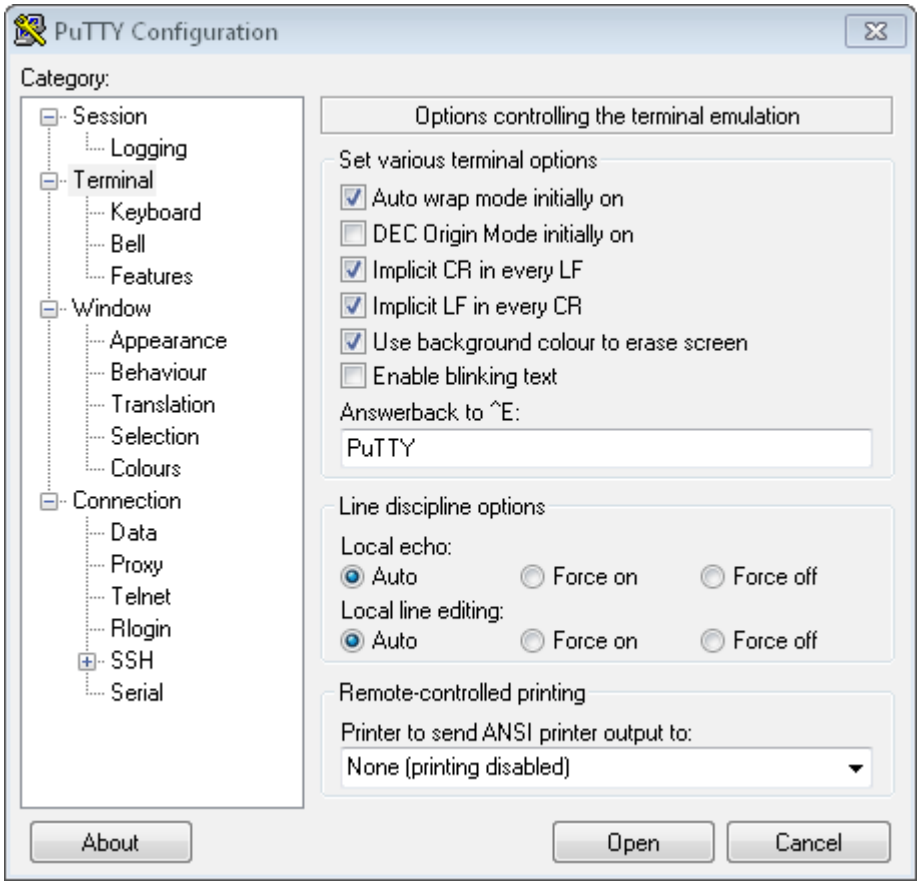
After you have established a connection, set the MODBUS read parameters to your needs. Select the function Setup / Read-Write Definition .. and configure the following parameters. If successful, the following values should be displayed:



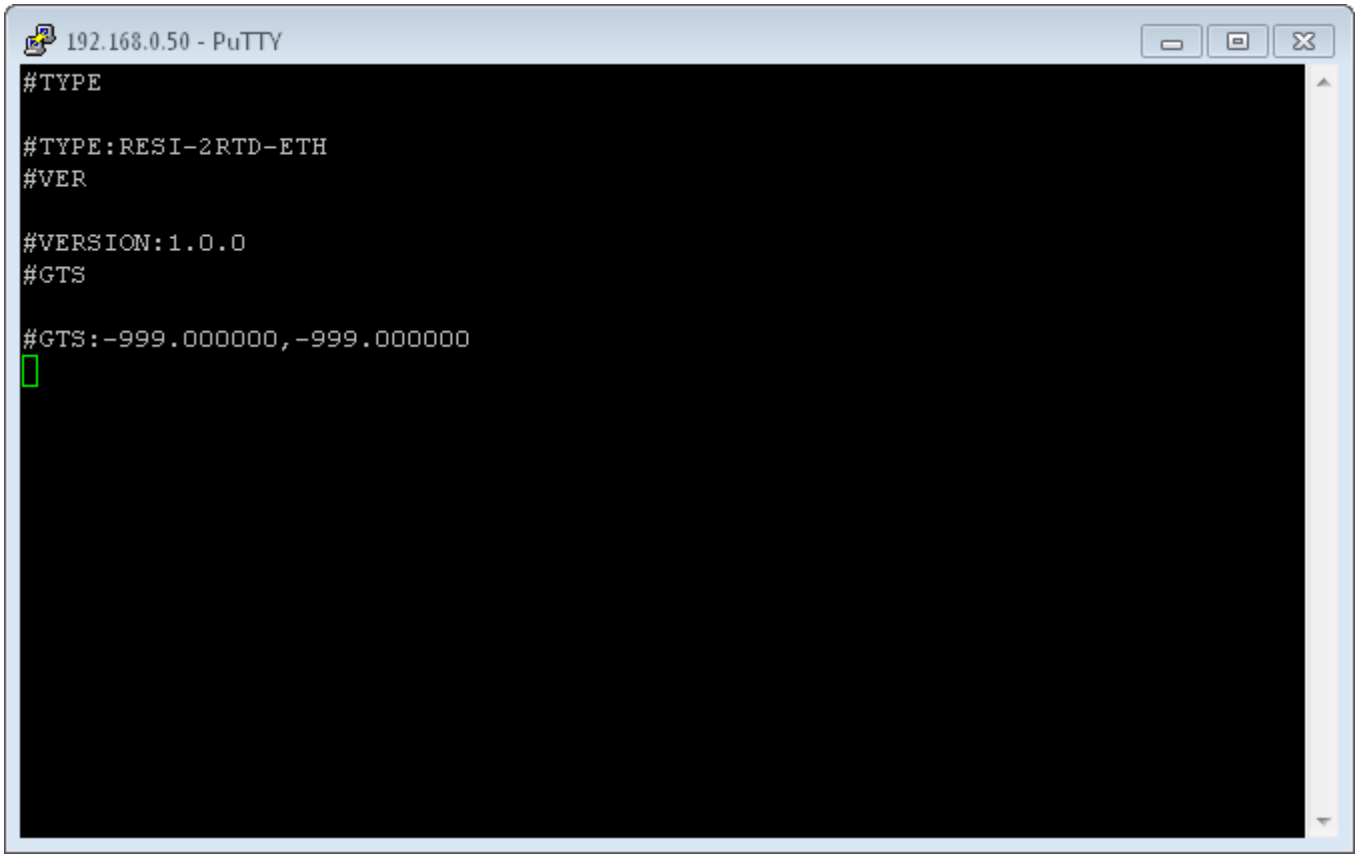
To test the ASCII protocol, use the freeware tool putty to establish a socket connection to the module. Configure your IP settings as follows:



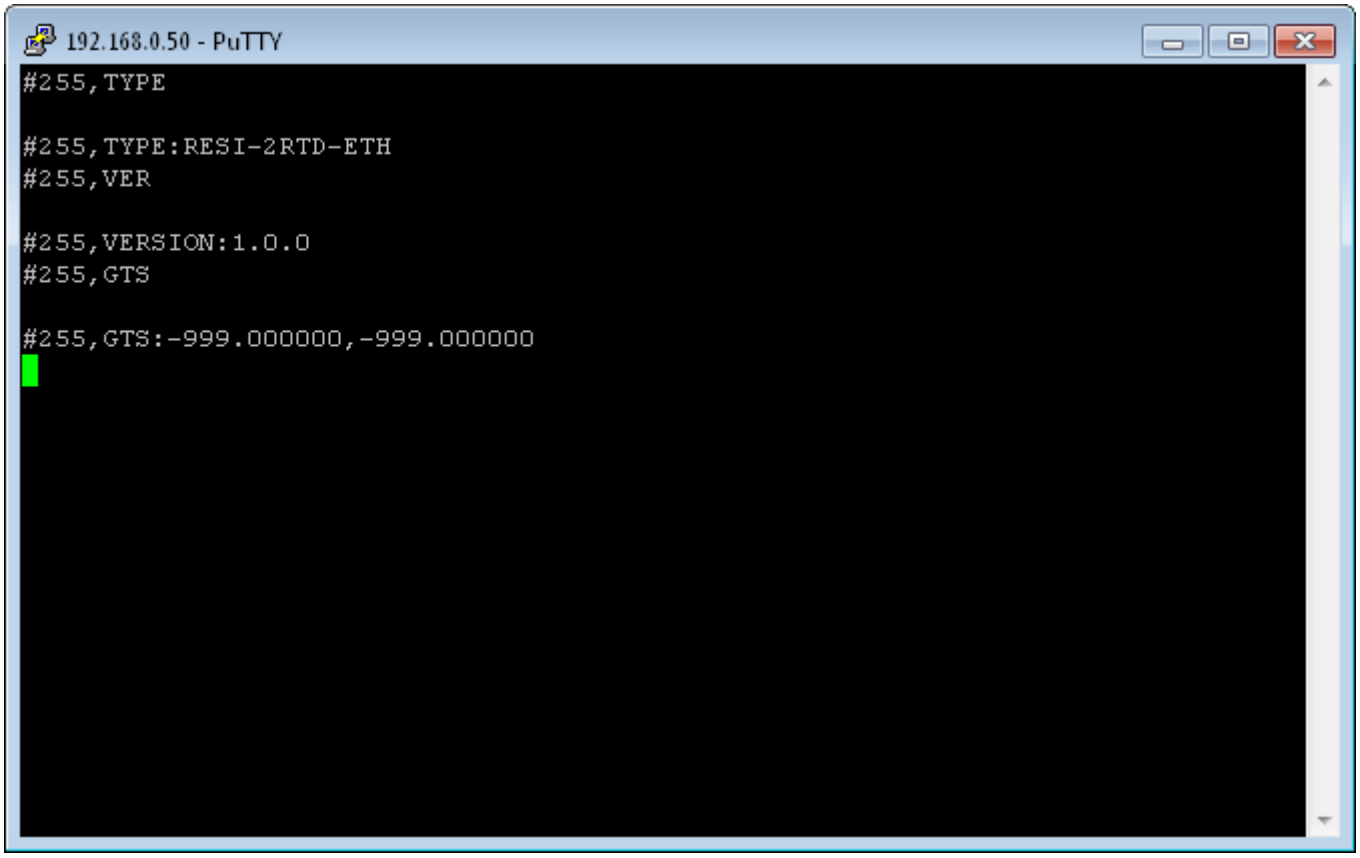
Then we have to change the behavior of the PUTTY terminal emulation. Click Terminal in the tree on the left and change the settings to the parameters shown below:



Click Open to establish a socket connection to the module. Enter the first command #TYPE <CR>. The IO module responds with the current module type.



You can also use the UnitID of the IO module in this protocol:

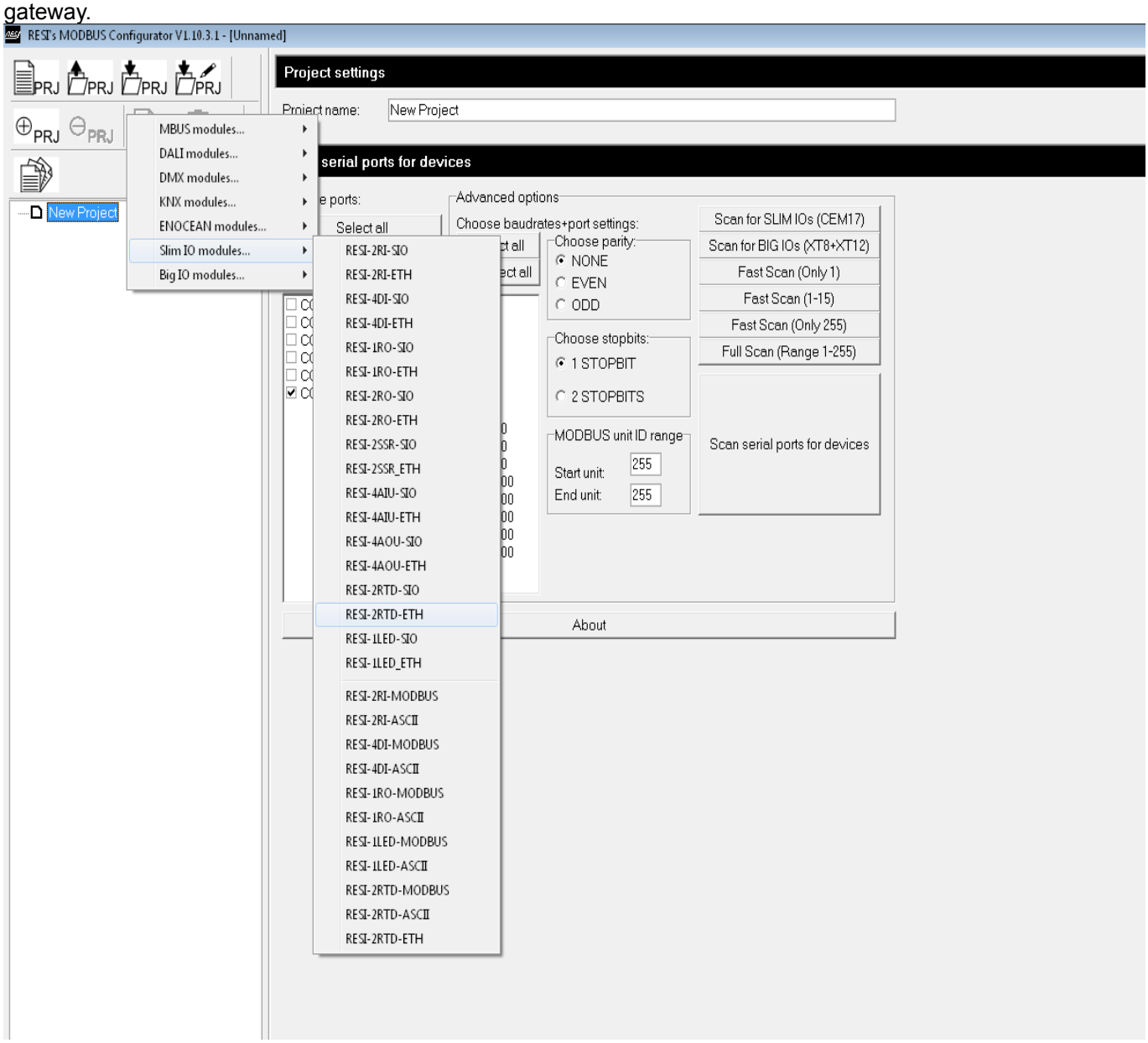


8.4 HOWTO connect to an Ethernet gateway

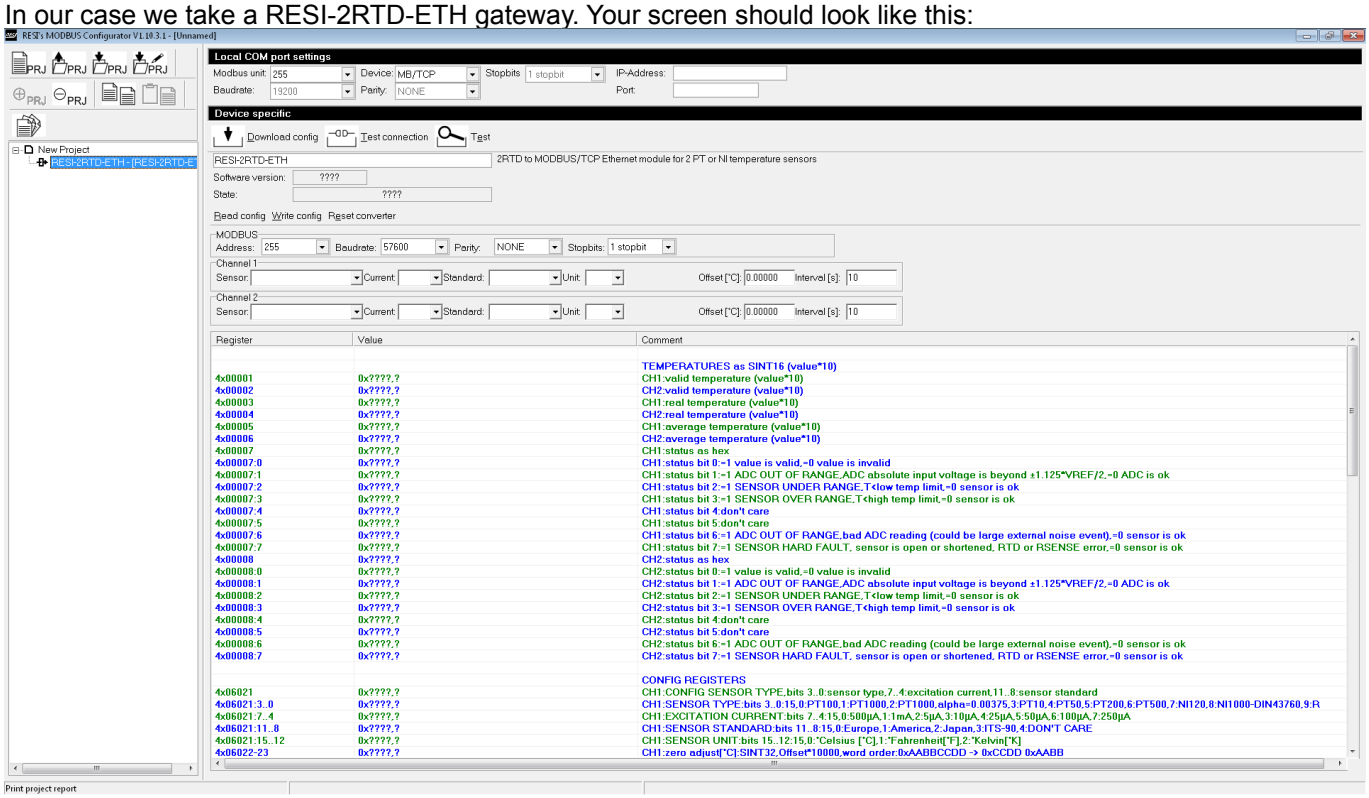
Follow these steps for communicating with an Ethernet gateway.

8.4.1 Example: Add RESI-2RTD-ETH to project tree

First, start the MODBUSConfigurator software. Click on the project tree title “New Project” and add a desired Ethernet gateway.

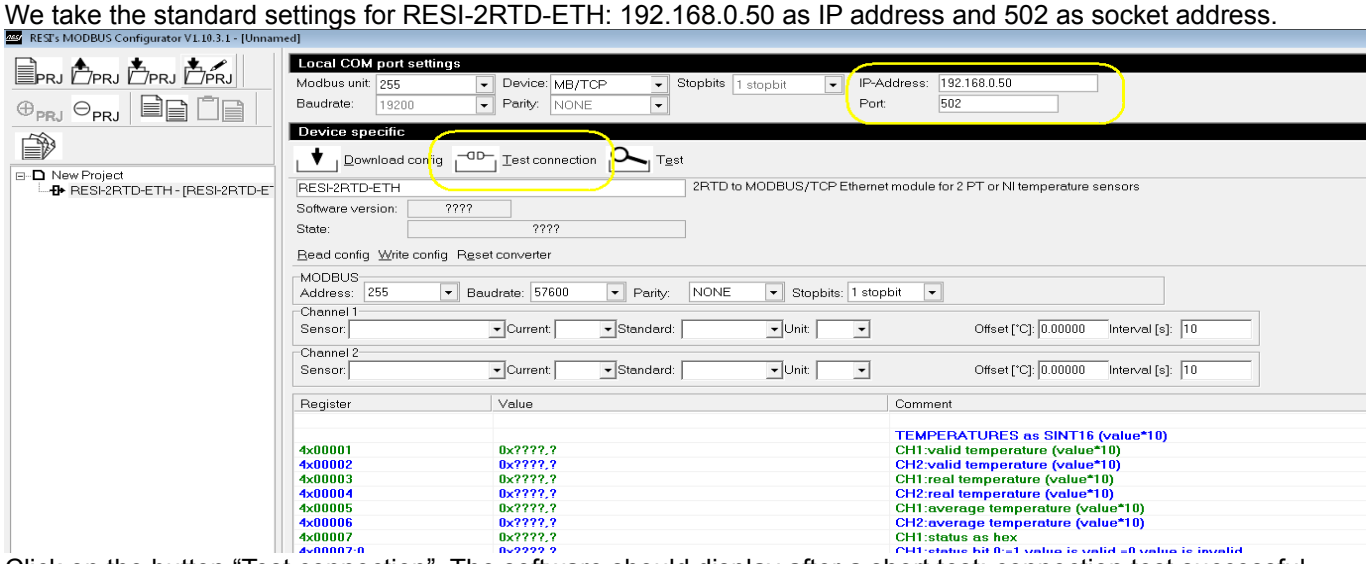


In our case we take a RESI-2RTD-ETH gateway. Your screen should look like this:



8.4.2 Enter IP address & socket port

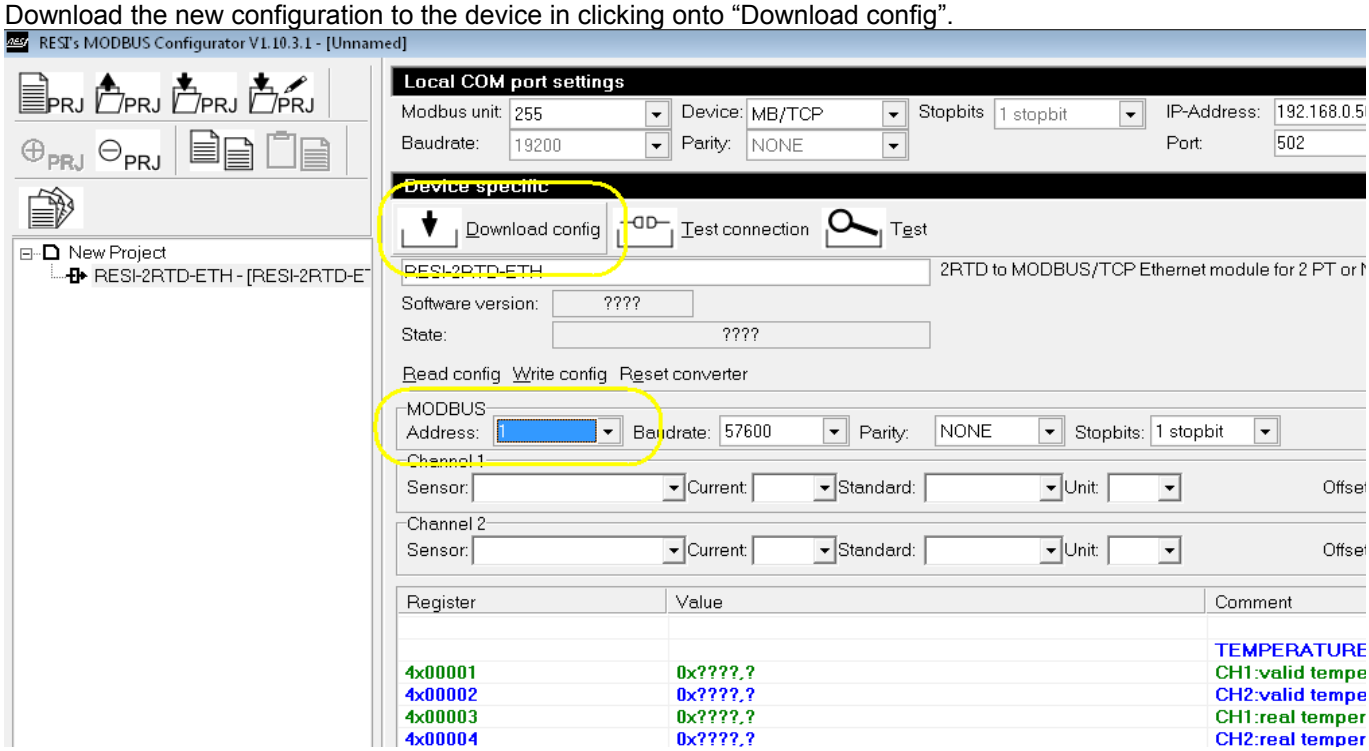
You will notice, that the software automatically suggest as a device “MB/TCP”. Now you have to enter the via web configuration defined IP address and socket number for the communication via MODBUS/TCP protocol.
We take the standard settings for RESI-2RTD-ETH: 192.168.0.50 as IP address and 502 as socket address.



Click on the button “Test connection”. The software should display after a short test: connection test successful.

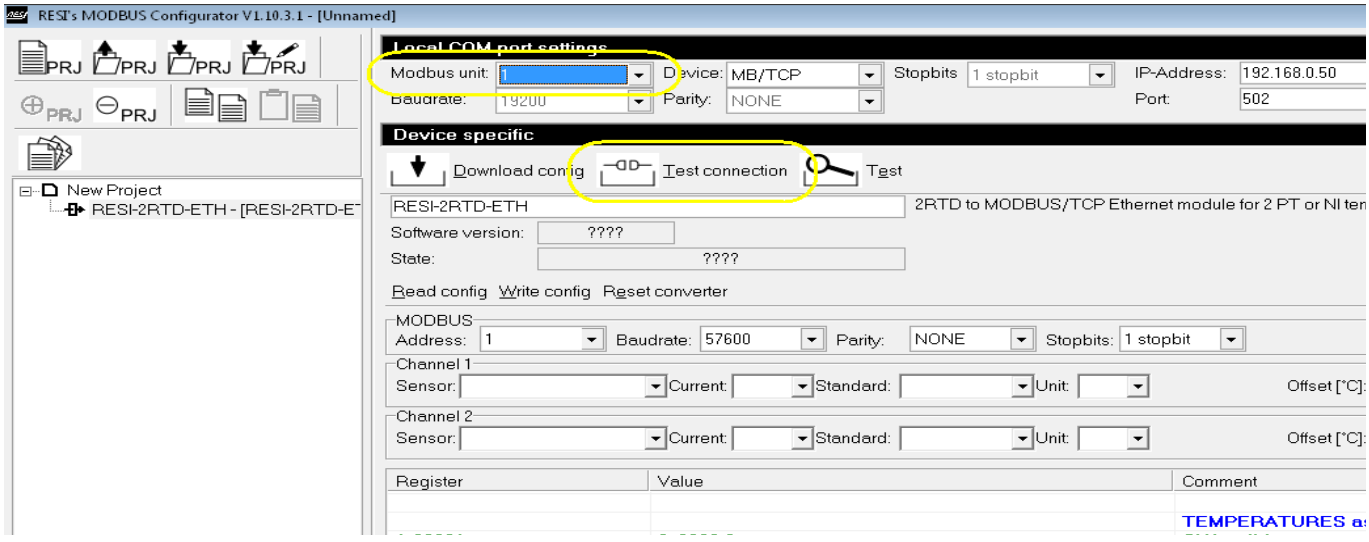
8.4.3 Change MODBUS unit ID to your needs

Now you can change the MODBUS address to your needs. We use 1 as a Unit ID for MODBUS/TCP communication. Download the new configuration to the device in clicking onto “Download config”.



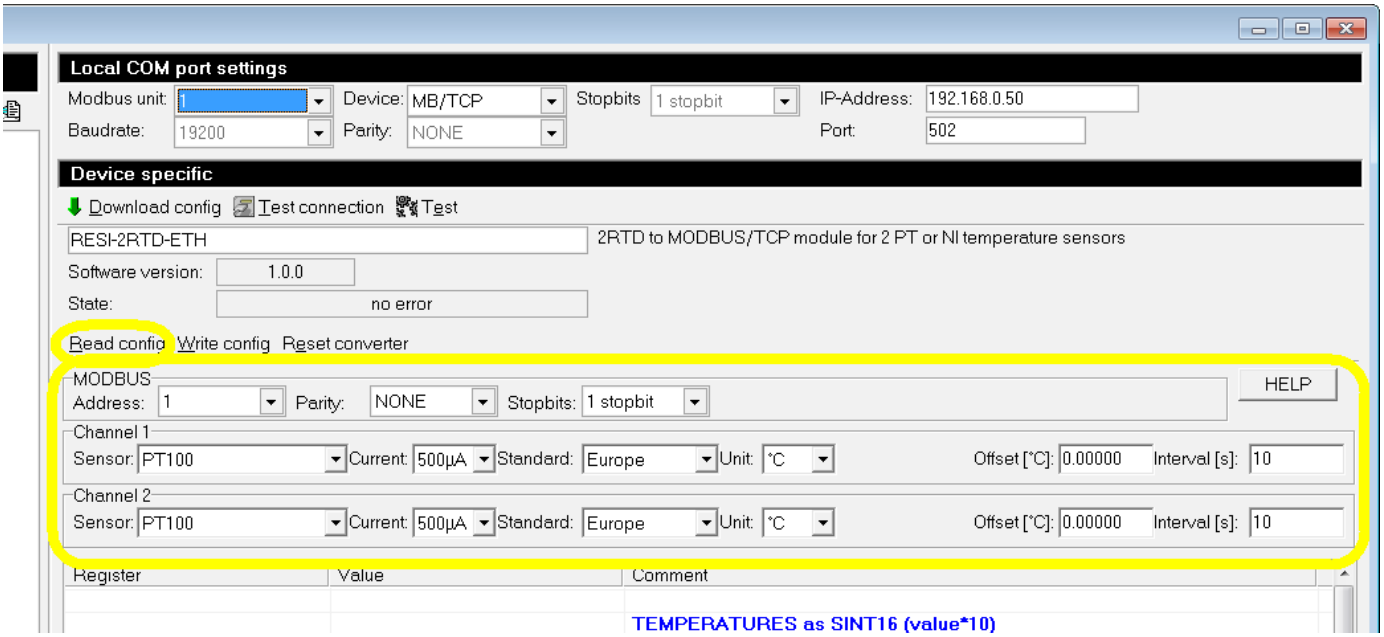
8.4.4 After Download config, change local COM port settings

The next step is to change the MODBUS Unit ID in the Local COM port settings to 1. Check the new settings with the function “Test connection”.

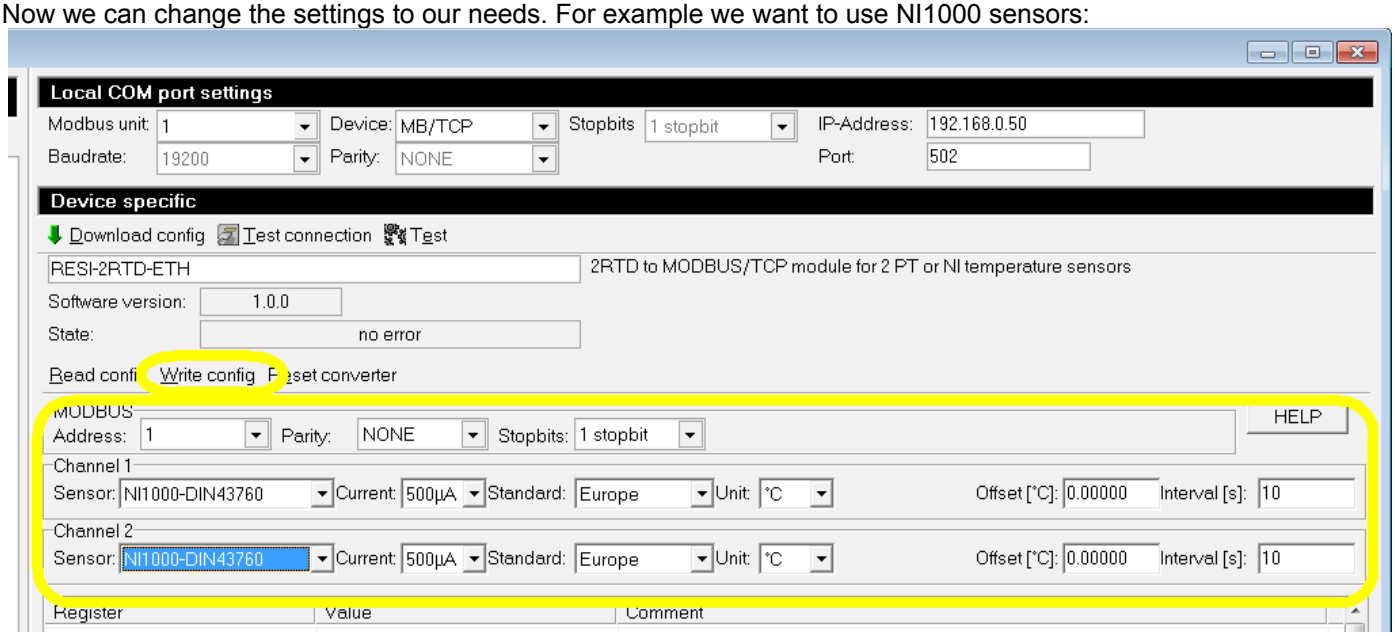


8.4.5 Read sensor configration

Now we read out the current sensor configuration of the IO module. Click on Read config. Your display should look like this:

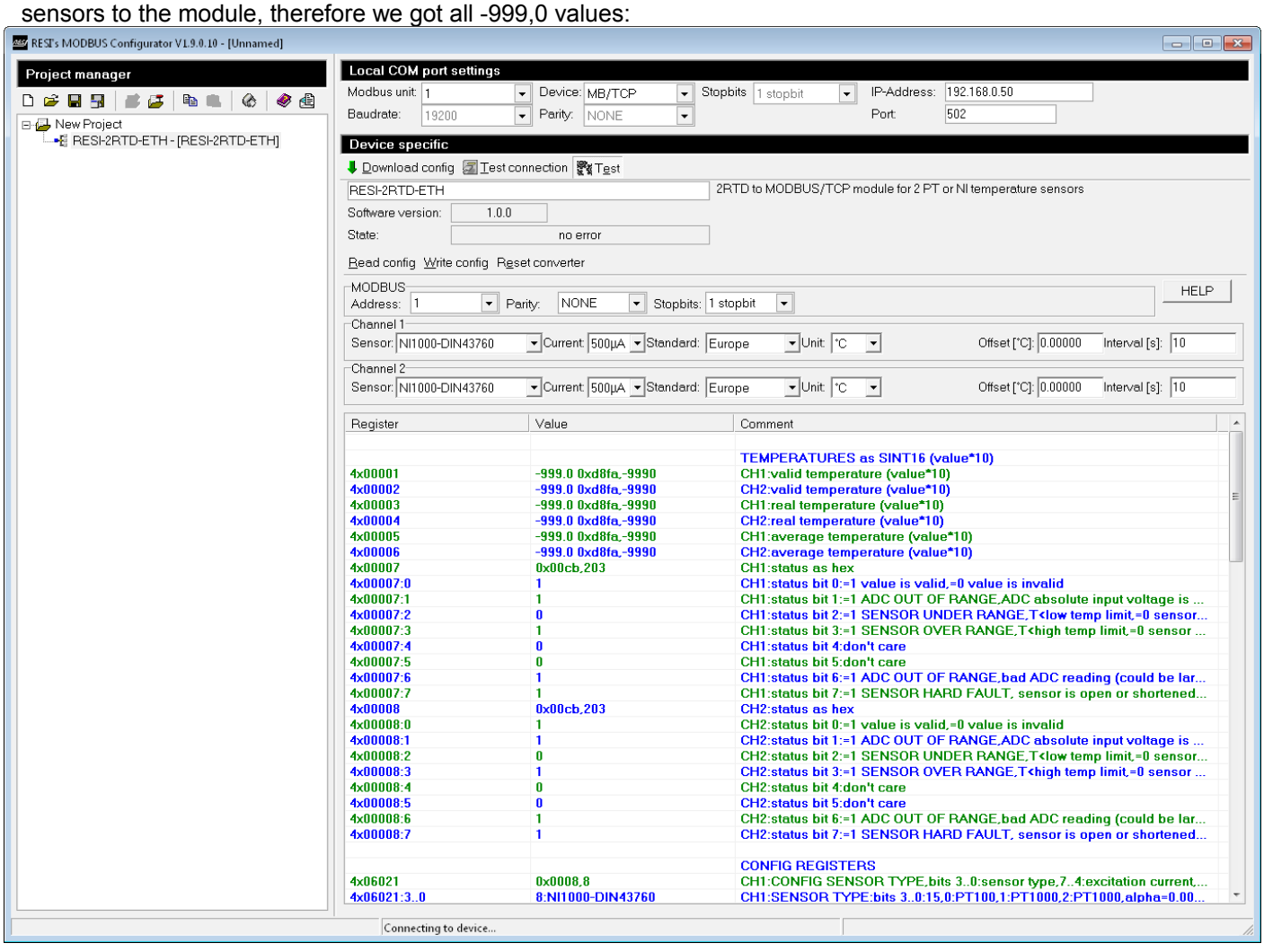


Now we can change the settings to our needs. For example we want to use NI1000 sensors:



8.4.6 Test the configuration

After a successful download we activate the test function. (Don't forget to select the correct Unit ID in the Local COM port settings. Otherwise you will get no connection). You should get the following result (We have not connected any sensors to the module, therefore we got all -999,0 values:



9 DIP switch settings

Our ULTRA SLIM IO module offer a 4 pin DIP switch for initial setup of the serial connection or the Ethernet connection. Our BIGIO modules offer a 8 pin DIP switch for initial setup.

9.1 DIP switch for serial ULTRA SLIM IOs

The following drawings show the DIP switches for all of our serial SLIMIO products:

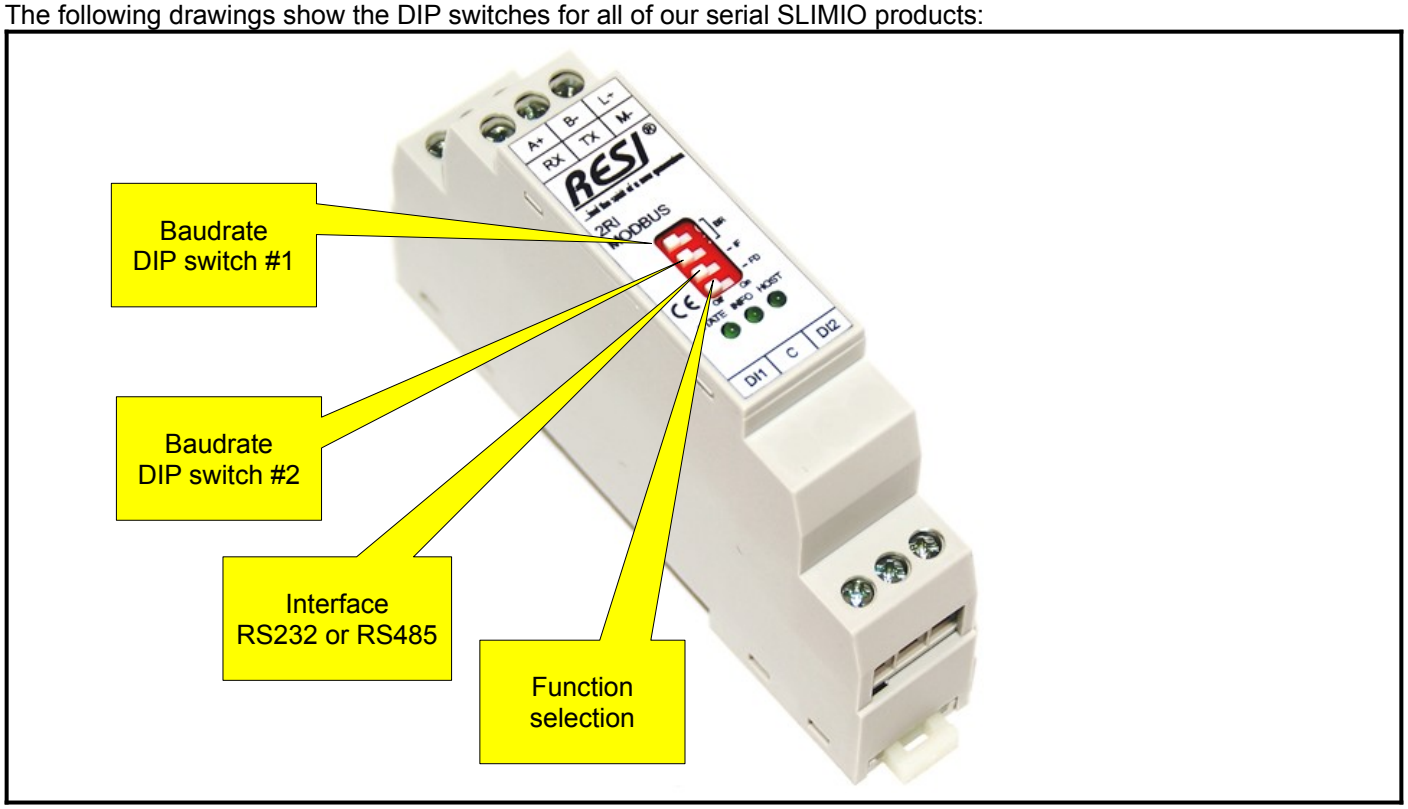


Figure: DIP switches for our serial ULTRA SLIM IO modules

Baud rate BR

Use DIP switch 1 + 2 to select the baud rate:

OFF	OFF:	9600Bd
ON	OFF:	19200Bd
OFF	ON:	38400Bd
ON	ON:	from FLASH (normally 57600Bd)

NOTE: The correct parity (NONE, EVEN, ODD) and the stop bits are set via the configuration software, not via the DIP switches! Likewise, the baud rate for the DIP switch position BR=ON,ON is set via the configuration software.

Interface IF

Selects the physical type of the serial interface for the ASCII or MODBUS/RTU protocol:

OFF=RS232

ON=RS485

Function selection FD

Selects a special function:

OFF=The unit ID from the FLASH is used

ON=Unit ID 255 is always used

NOTE

After changing the DIP switch, the device will be booted automatically So no voltage off/voltage one cycle is necessary. After restarting, all LEDs flash briefly to represent the restart sequence.

9.2 DIP switch for Ethernet ULTRA SLIM IOs

The following drawings show the DIP switches for all of our Ethernet SLIMIO products:

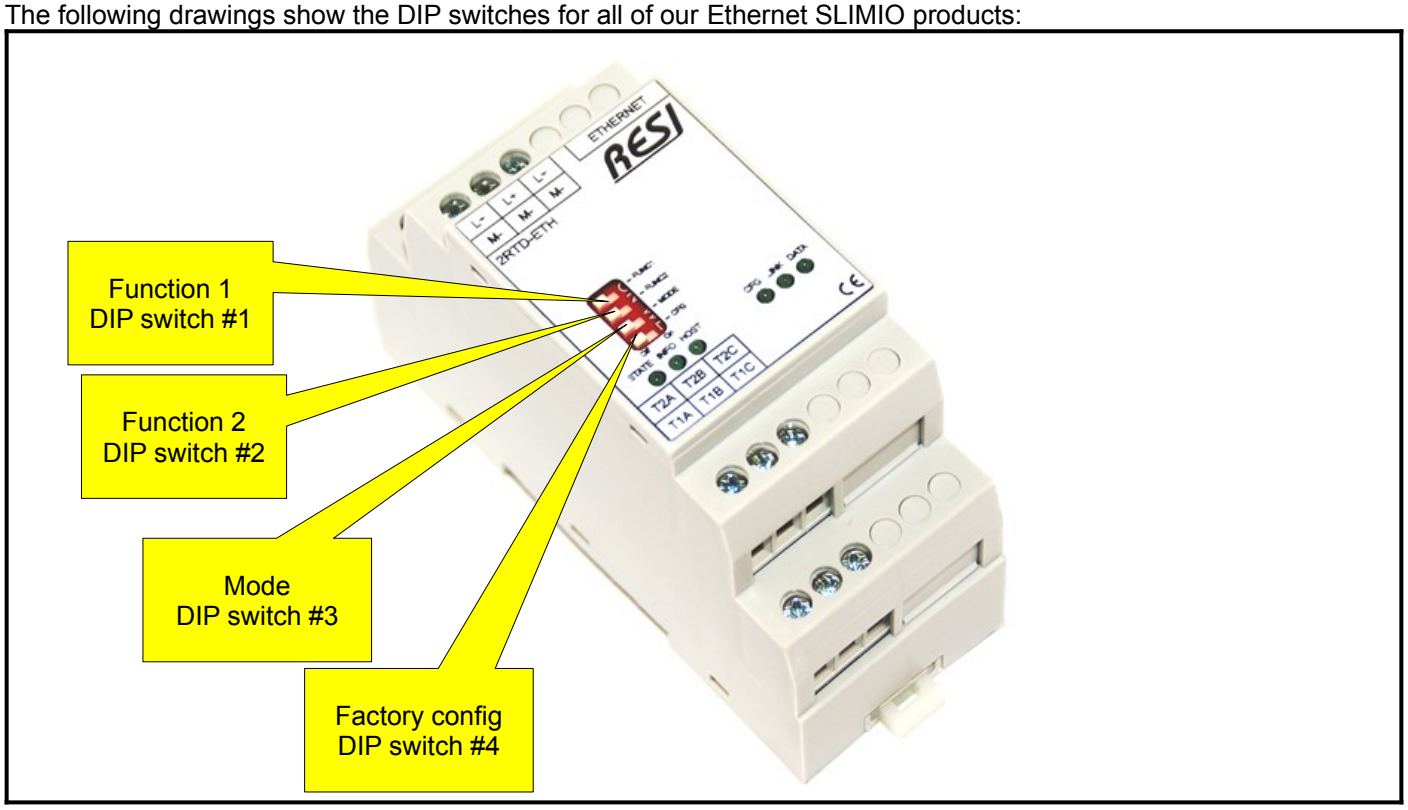


Figure: DIP switches for our Ethernet ULTRA SLIM IO modules

FUNC1	ON: When the module is restarted, the module changes to STATIC IP configuration with the standard IP settings
	OFF: The current IP settings are used
FUNC2	ON: When the module is restarted, the module changes to DHCP IP configuration.
	OFF: The current IP settings are used
MODE	While resetting to factory settings (CFG=ON):
	OFF: Socket mode is set to MODBUS/TCP Socket
	ON: Socket mode is set to MODBUS/RTU or ASCII over Ethernet
	In normal operation:
	OFF: The configured UnitID is used
	ON: UnitID 255 is always used!
CFG	ON: When the module restarts, the module restores the factory settings.
	Wait for about 30 seconds until the STATE + CFG LED
	blink quickly. Then set all DIP switches to OFF.
	The module restarts automatically and is ready for use.
	OFF: Normal start of the module
NOTE	After changing a DIP switch, the module restarts immediately.
	After restarting, all LEDs are briefly switched on to visually indicate the restart of the device.

9.3 DIP switch for serial BIG IOs

The following drawings show the DIP switches for all of our serial BIGIO products:

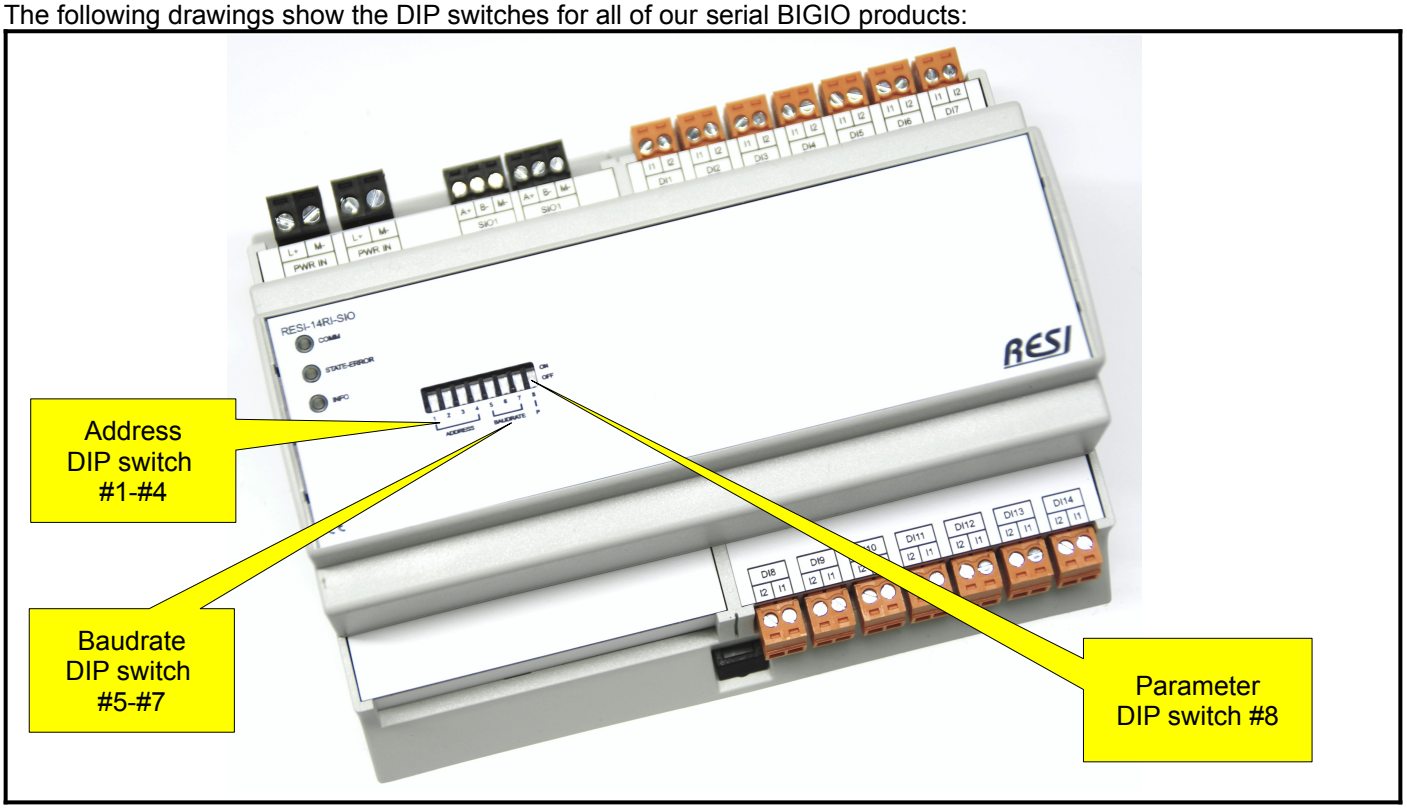


Figure: DIP switches for our serial BIG IO modules

The 8 pin DIP switch has the following mapping:

DIP SWITCH
1=ADR0
2=ADR1
3=ADR2
4=ADR3
5=BR0
6=BR1
7=BR2
8=PARAMETER

ADDRESS This four DIP switches ADR3-ADR0 create the MODBUS/RTU unit number or ASCII bus address in the range of 0 to 15. You can use the following settings:

ADR3	ADR2	ADR1	ADR0	MODBUS/RTU unit number or ASCII bus number
OFF	OFF	OFF	OFF	Internal MODBUS/RTU unit number is used from the FLASH memory in the range of 0 to 255.
OFF	OFF	OFF	ON	1
OFF	OFF	ON	OFF	2
OFF	OFF	ON	ON	3
OFF	ON	OFF	OFF	4
OFF	ON	OFF	ON	5
OFF	ON	ON	OFF	6
OFF	ON	ON	ON	7
ON	OFF	OFF	OFF	8
ON	OFF	OFF	ON	9
ON	OFF	ON	OFF	10
ON	OFF	ON	ON	11
ON	ON	OFF	OFF	12
ON	ON	OFF	ON	13
ON	ON	ON	OFF	14
ON	ON	ON	ON	15

BAUD RATE Those three DIP switches BR2-BR0 define the MODBUS/RTU or ASCII baud rate for the communication:

BR2	BR1	BR0	MODBUS/RTU or ASCII baud rate
OFF	OFF	OFF	4800bd
OFF	OFF	ON	9600bd
OFF	ON	OFF	19200bd
OFF	ON	ON	38400bd
ON	OFF	OFF	57600bd
ON	OFF	ON	115200bd
ON	ON	OFF	230400bd
ON	ON	ON	256000bd

PARAMETER This DIP switch selects between the configuration via DIP switch or via FLASH parameter for the serial setup.

=0: The selected UnitID, baud rate from the DIP switch settings are used.

The parity is NONE and the one stop bit is used

=1: The selected UnitID from the DIP switches is used, but the serial parameters are taken from the FLASH parameters.

Baud rate can be selected between 300 to 256000 Baud.

Parity can be NONE, EVEN or ODD.

Stop bits can be ONE or TWO.

NOTE After changing the DIP switch, the device will be booted automatically So no voltage off/ voltage one cycle is necessary. After restarting, all LEDs flash briefly to represent the restart sequence.

9.4 DIP switches for BIG IOs RESI-S16DI8RO-SIO,RESI-S8R-SIO

The following drawings show the DIP switches for our serial BIGIO products RESI-S16DI8RO-SIO and RESI-S8RO-SIO:

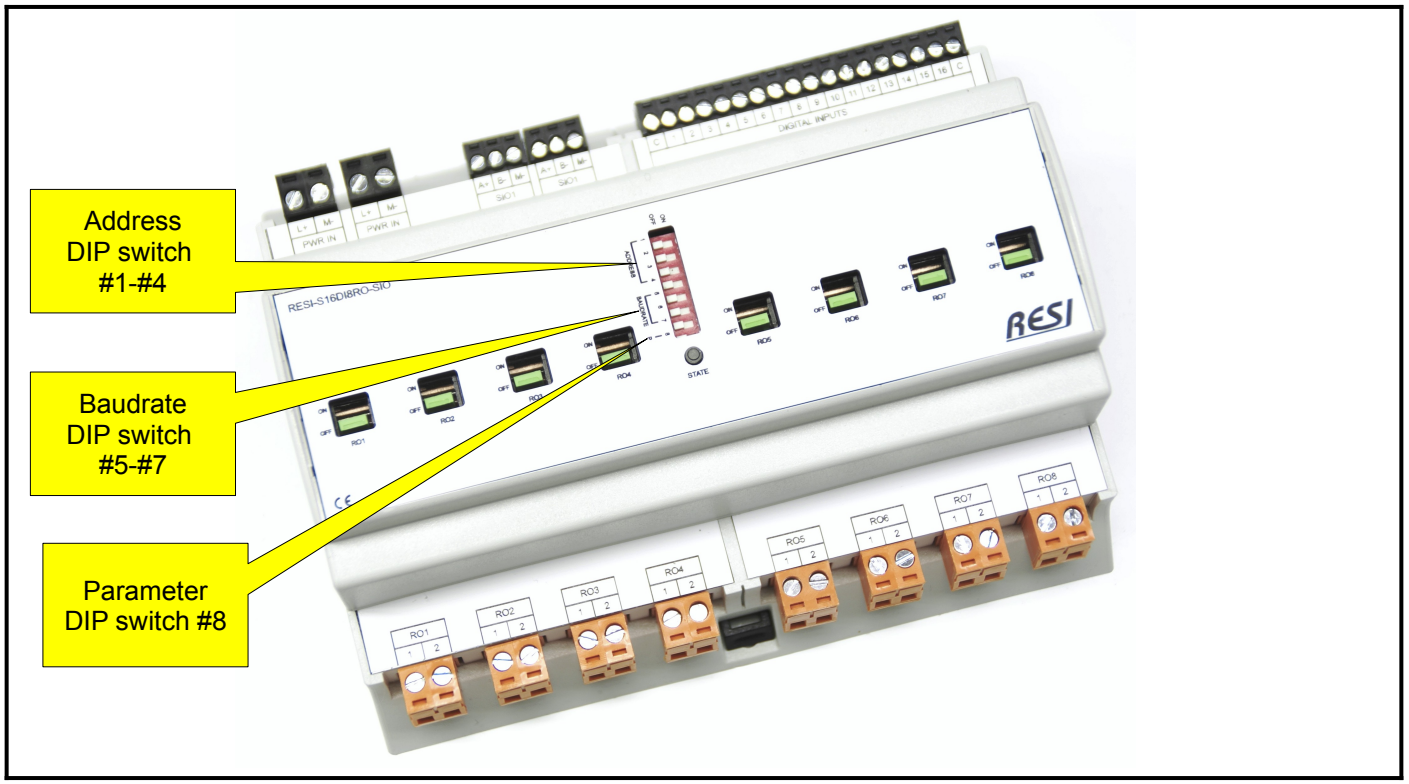


Figure: DIP switches for our BIG IO modules RESI-S16DI8RO-SIO, RESI-S8RO-SIO

The 8 pin DIP switch has the following mapping:

DIP SWITCH
1=ADR0
2=ADR1
3=ADR2
4=ADR3
5=BR0
6=BR1
7=BR2
8=PARAMETER

ADDRESS This four DIP switches ADR3-ADR0 create the MODBUS/RTU unit number or ASCII bus address in the range of 0 to 15. You can use the following settings:

ADR3	ADR2	ADR1	ADR0	MODBUS/RTU unit number or ASCII bus number
OFF	OFF	OFF	OFF	Internal MODBUS/RTU unit number is used from the FLASH memory in the range of 0 to 255.
OFF	OFF	OFF	ON	1
OFF	OFF	ON	OFF	2
OFF	OFF	ON	ON	3
OFF	ON	OFF	OFF	4
OFF	ON	OFF	ON	5
OFF	ON	ON	OFF	6
OFF	ON	ON	ON	7
ON	OFF	OFF	OFF	8
ON	OFF	OFF	ON	9
ON	OFF	ON	OFF	10
ON	OFF	ON	ON	11
ON	ON	OFF	OFF	12
ON	ON	OFF	ON	13
ON	ON	ON	OFF	14
ON	ON	ON	ON	15

BAUD RATE Those three DIP switches BR2-BR0 define the MODBUS/RTU or ASCII baud rate for the communication:

BR2	BR1	BR0	MODBUS/RTU or ASCII baud rate
OFF	OFF	OFF	4800bd
OFF	OFF	ON	9600bd
OFF	ON	OFF	19200bd
OFF	ON	ON	38400bd
ON	OFF	OFF	57600bd
ON	OFF	ON	115200bd
ON	ON	OFF	230400bd
ON	ON	ON	256000bd

PARAMETER This DIP switch selects between the configuration via DIP switch or via FLASH parameter for the serial setup.

=0: The selected UnitID, baud rate from the DIP switch settings are used.

The parity is NONE and the one stop bit is used

=1: The selected UnitID from the DIP switches is used, but the serial parameters are taken from the FLASH parameters.

Baud rate can be selected between 300 to 256000 Baud.

Parity can be NONE, EVEN or ODD.

Stop bits can be ONE or TWO.

NOTE After changing the DIP switch, the device will be booted automatically So no voltage off/ voltage one cycle is necessary. After restarting, all LEDs flash briefly to represent the restart sequence.

10 LED indicators

Our serial ULTRA SLIM IO modules offer 3 LED indicators for status display, our Ethernet ULTRA SLIM IO modules offer 6 LED indicators for status display. Our BIGIO modules offer 4 LED indicators for status display.

10.1 LED indicators for serial ULTRA SLIM IOs

The following drawings show the LED indicators for all of our serial SLIMIO products:

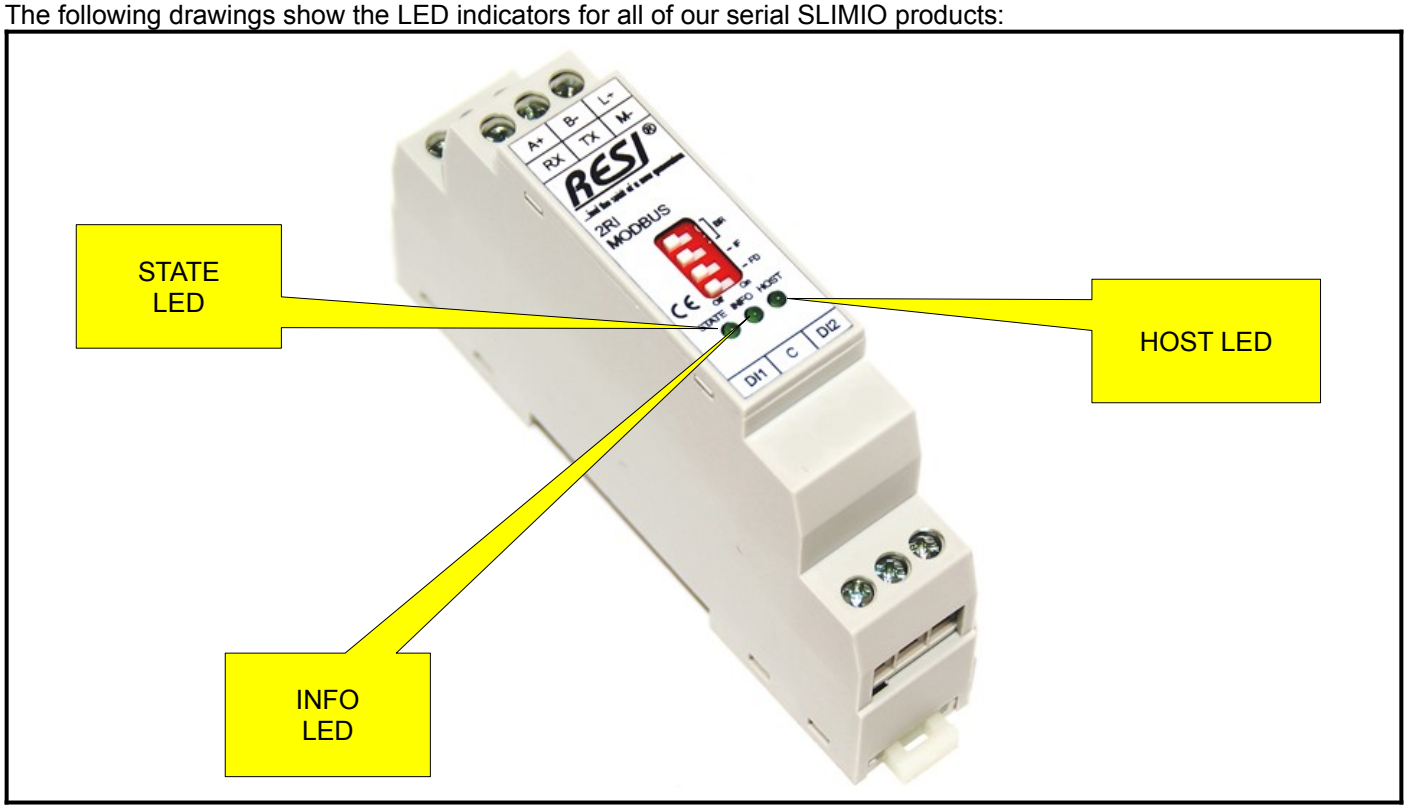


Figure: LED indicators for our serial ULTRA SLIM IO modules

STATE State LED, flashes slowly (approx. 1s) if the module is OK.
Flashes quickly when the module has an internal error

INFO This LED shows more information about the local IOs. The functionality depends on the used IO module. Please refer to the detailed description for each IO module.

HOST Shows whether serial data is currently being sent or received via the RS232 or RS485 interface

10.2 LED indicators for Ethernet ULTRA SLIM IOs

The following drawings show the LED indicators for all of our Ethernet SLIMIO products:

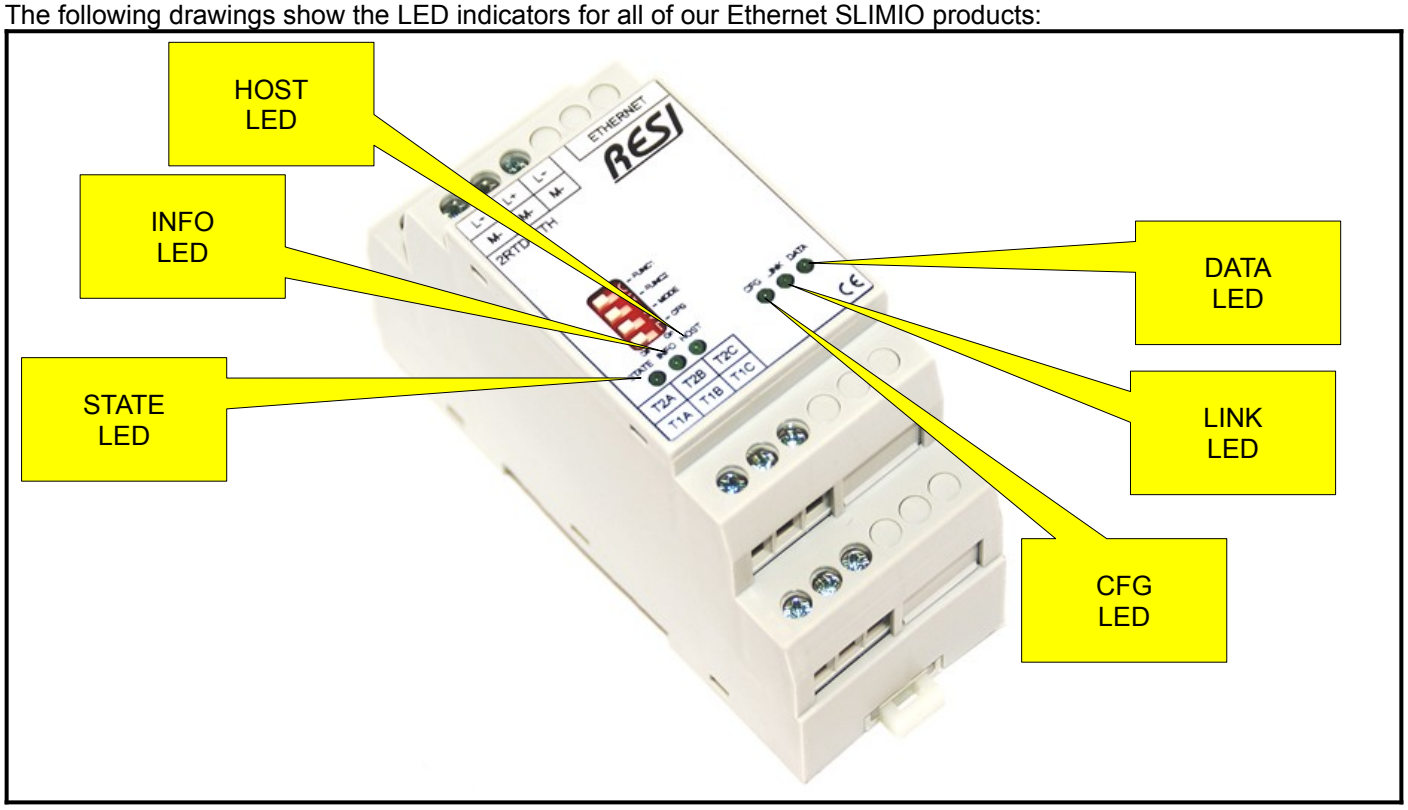


Figure: LED indicators for our Ethernet ULTRA SLIM IO modules

STATE State LED, flashes slowly (approx. 1s) if the module is OK.
Flashes quickly when the module has an internal error

INFO This LED shows more information about the local IOs. The functionality depends on the used IO module. Please refer to the detailed description for each IO module.

HOST Shows whether serial data is currently being sent or received via the internal serial interface to the Ethernet controller

CFG Factory setting LED:

In working mode, this LED flashes in the same rhythm as the STATE LED.

If the DIP switch CFG is ON when restarting, the STATE LED is always on and the CFG LED flashes slowly. When this process is complete, both LEDs will flash very fast.

Then the CFG LED must be set to OFF again!

LINK This LED is on when the Ethernet interface is electrically connected correctly with the network

DATA This LED shows the data flow on the Ethernet interface

10.3 LED indicators for serial BIG IOs

The following drawings show the LED indicators for all of our serial BIGIO products:

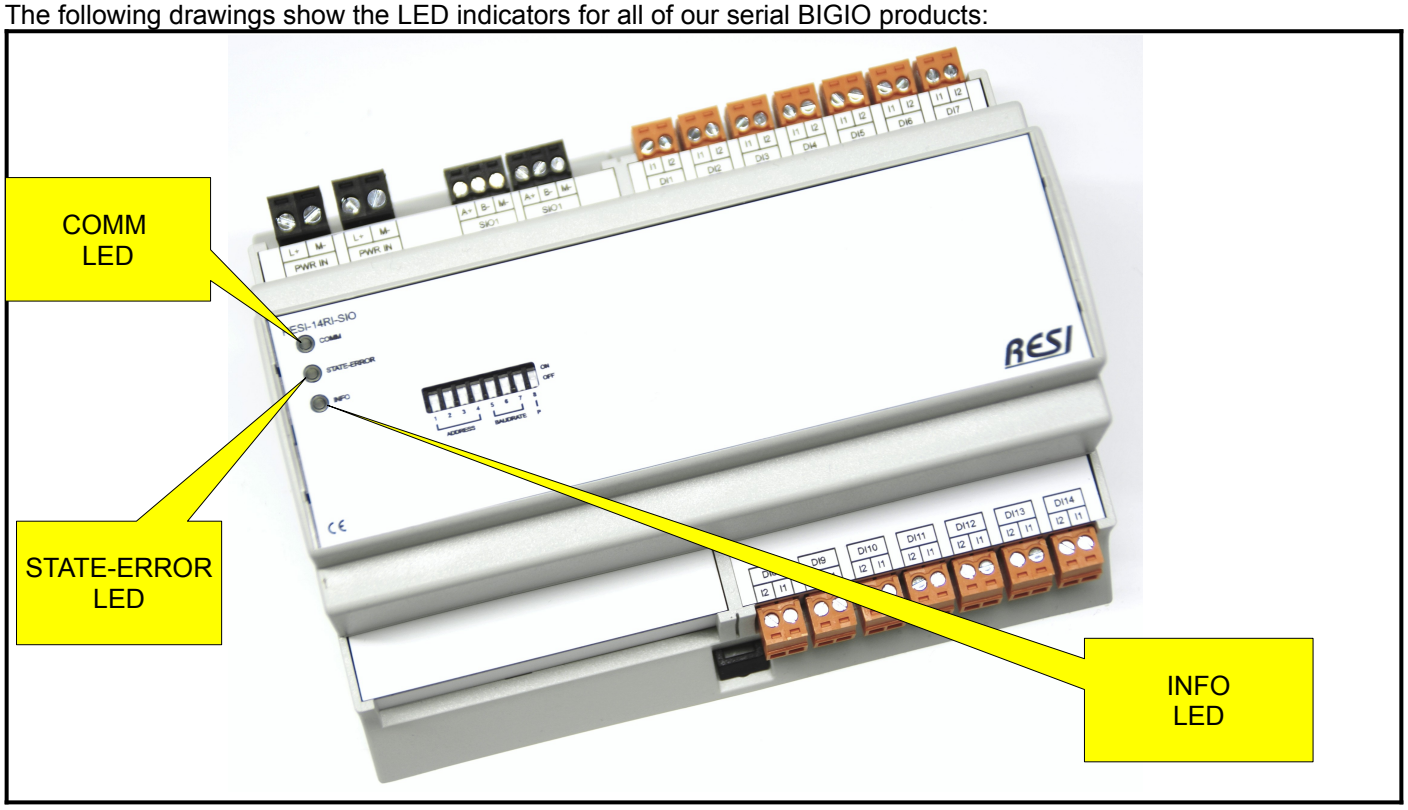


Figure: LED indicators for our serial BIG IO modules

COMM Shows whether serial data is currently being sent or received via the RS485 interface

STATE-ERROR State LED, flashes slowly (approx. 1s) in WHITE if the module is OK.
Flashes quickly in RED when the module has an internal error

INFO This LED shows more information about the local IOs. The functionality depends on the used IO module. Please refer to the detailed description for each IO module.

10.4 LED indicators for BIG IOs RESI-S16DI8RO-SIO,RESI-S8R-SIO

The following drawings show the LED indicators for our serial BIGIO products RESI-S16DI8RO-SIO and RESI-S8RO-SIO:

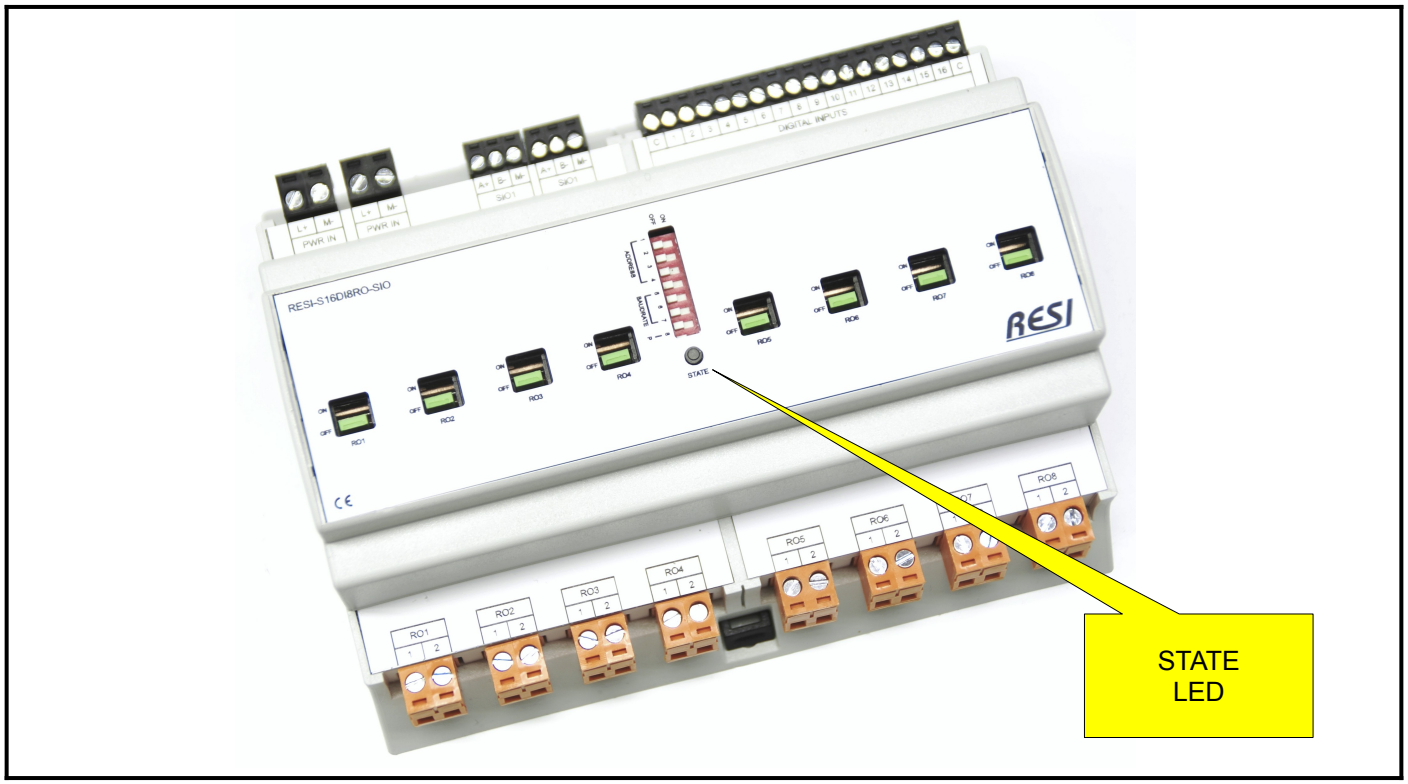


Figure: LED indicators for our BIG IO modules RESI-S16DI8RO-SIO, RESI-S8RO-SIO

STATE State LED, flashes slowly (approx. 1s) in WHITE if the module is OK.
Flashes quickly in RED when serial data is currently being sent or received via the RS485 interface

11 DIMENSIONS

11.1 ULTRA SLIM IOs: RESI-xxx-SIO

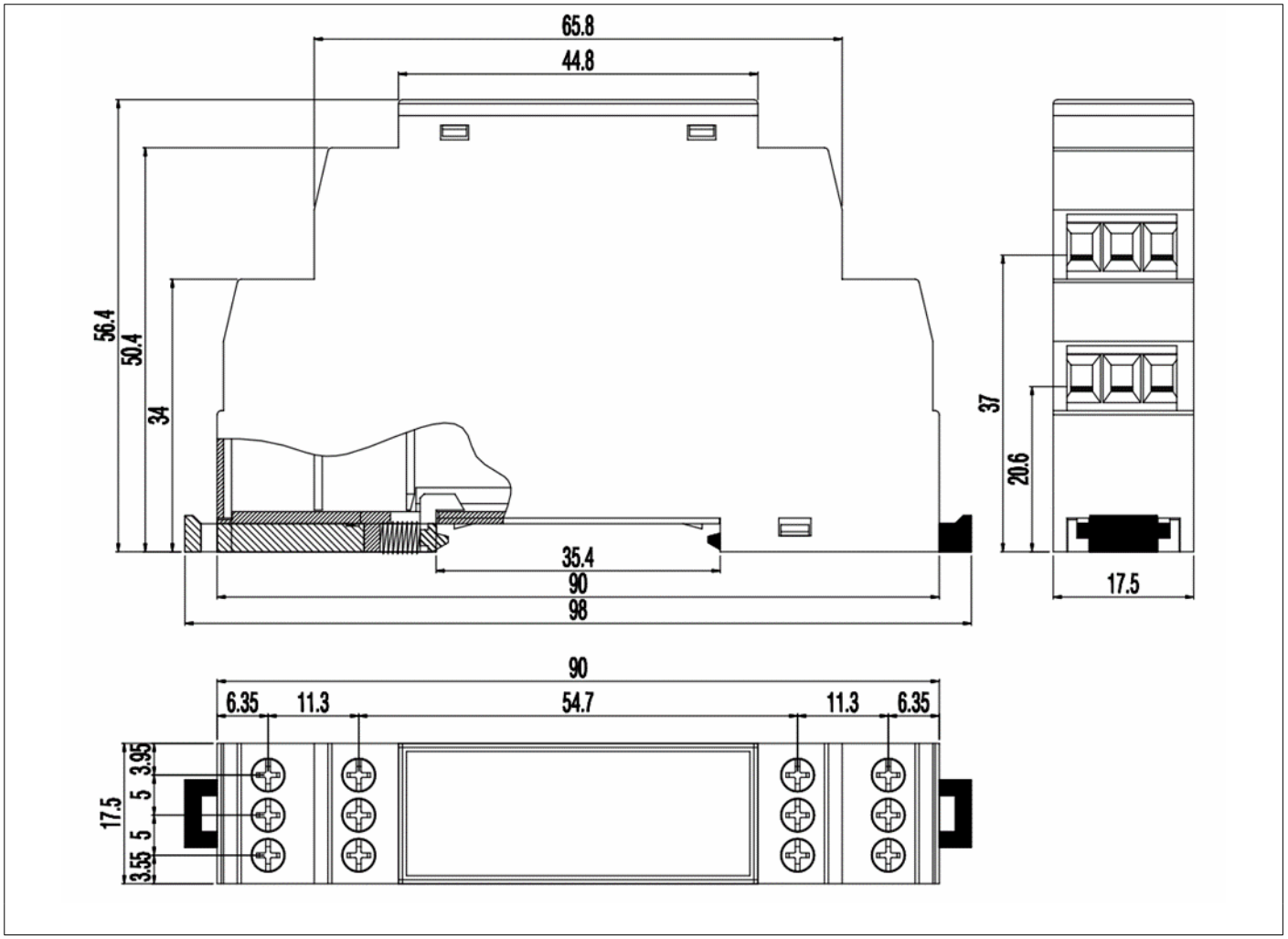


Figure: Dimensions of the housing for our serial ULTRA SLIM IO modules in mm

Dimensions

Housing illustration	17.5 x 90 x 58
Color	gray RAL 7035
Material	PA-UL 94 V0
Protection class	IP20 based on DIN 40050 / EB 60529

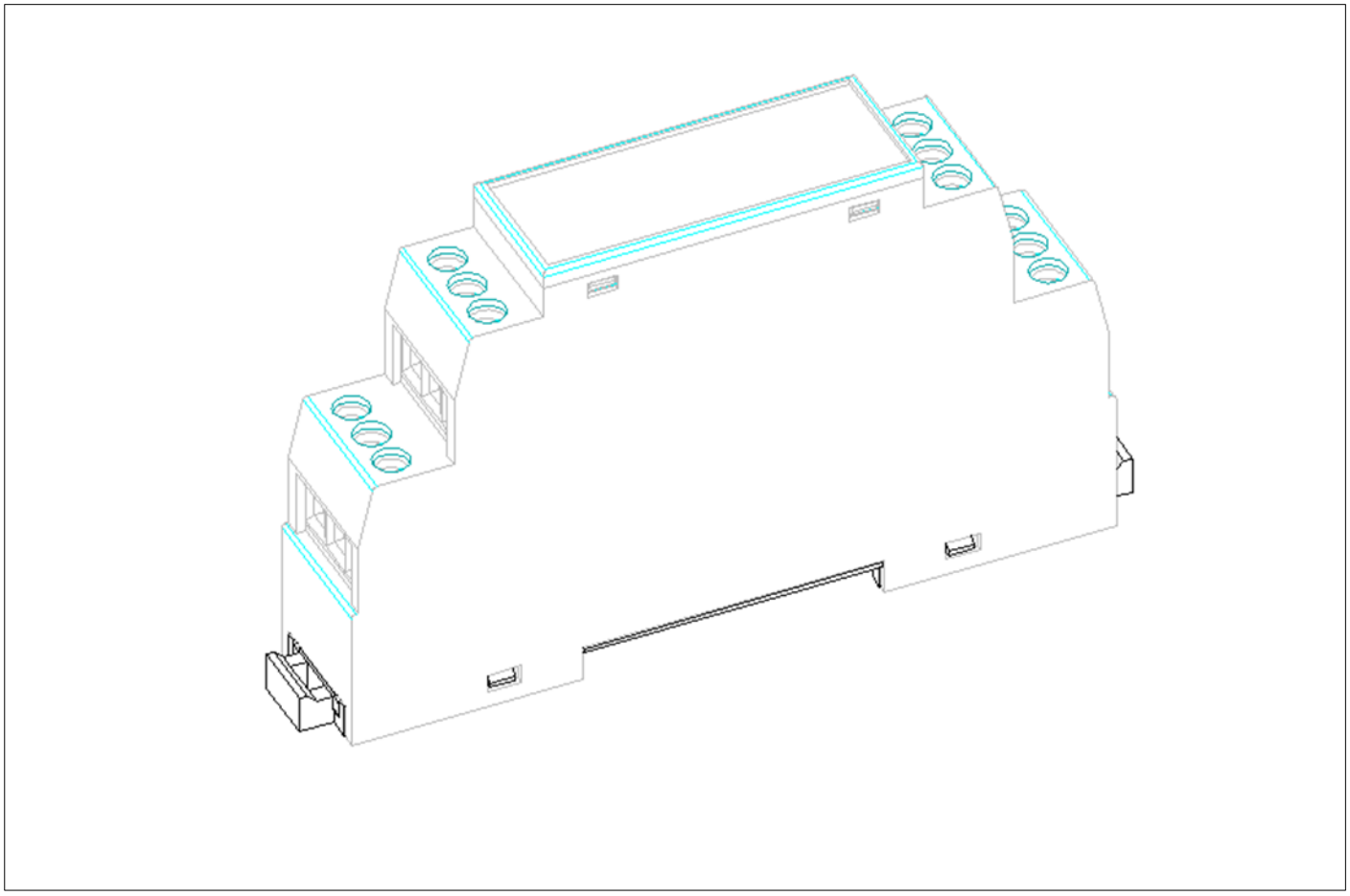


Figure: For our serial ULTRA SLIM IO modules: Housing illustration in 3D

11.2 ULTRA SLIM IOs: RESI-xxx-ETH

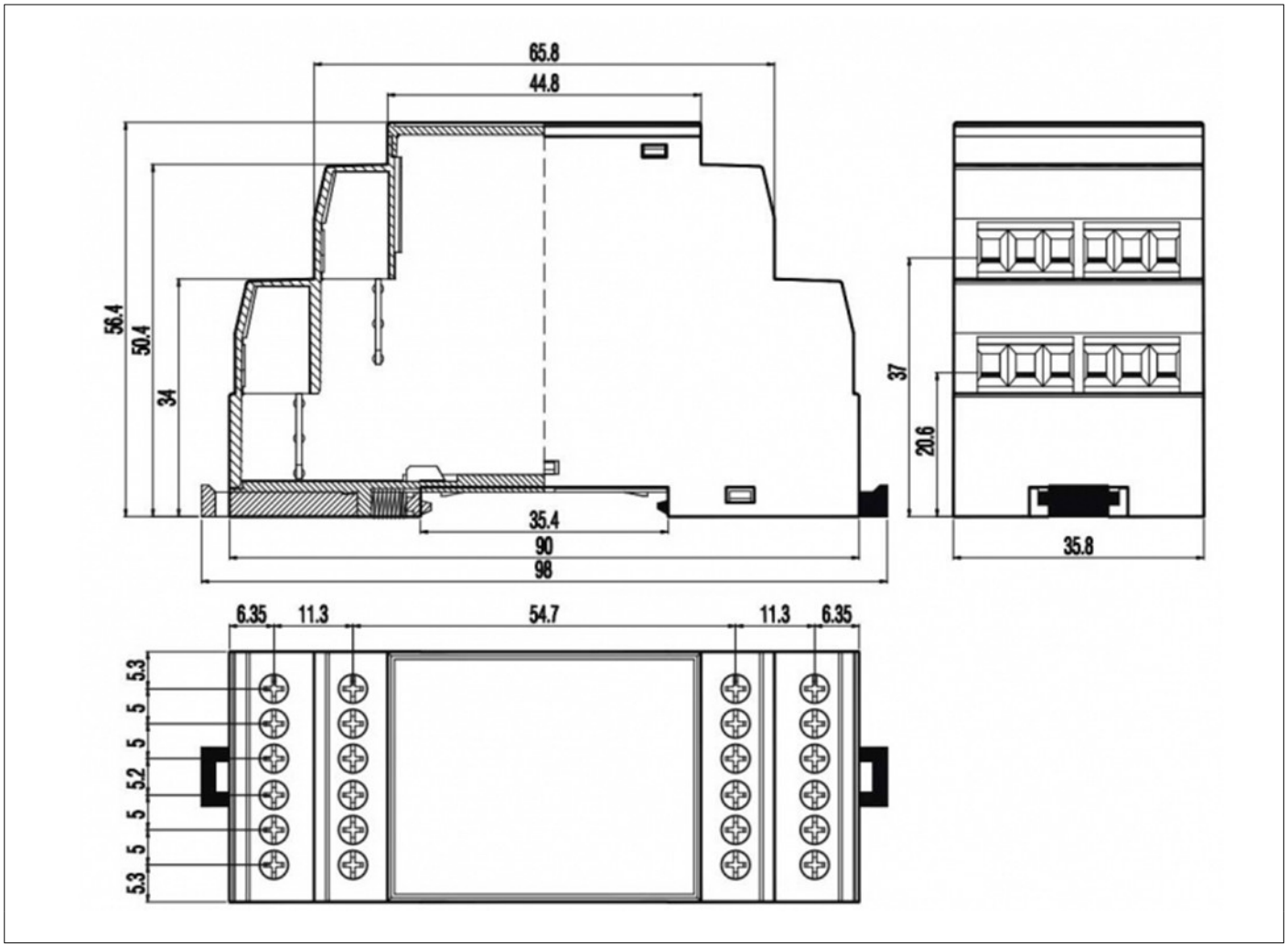


Figure: Dimensions of the housing for our Ethernet ULTRA SLIM IO modules in mm

Dimensions

Housing illustration	35.8 x 90 x 58
Color	gray RAL 7035
Material	PA-UL 94 V0
Protection class	IP20 based on DIN 40050 / EB 60529

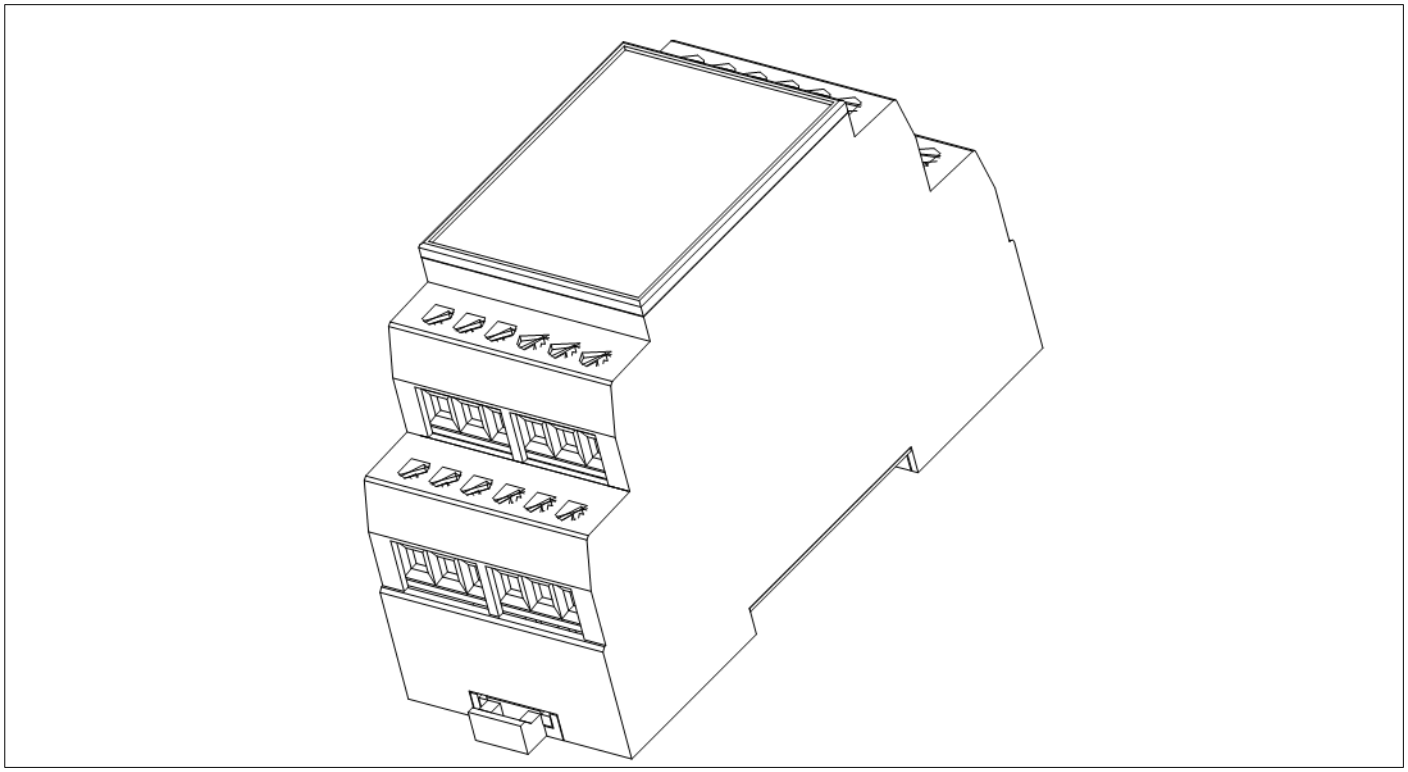


Figure: For our Ethernet ULTRA SLIM IO modules: Housing illustration in 3D

11.3 BIG IOs: RESI-xxx-SIO XT8

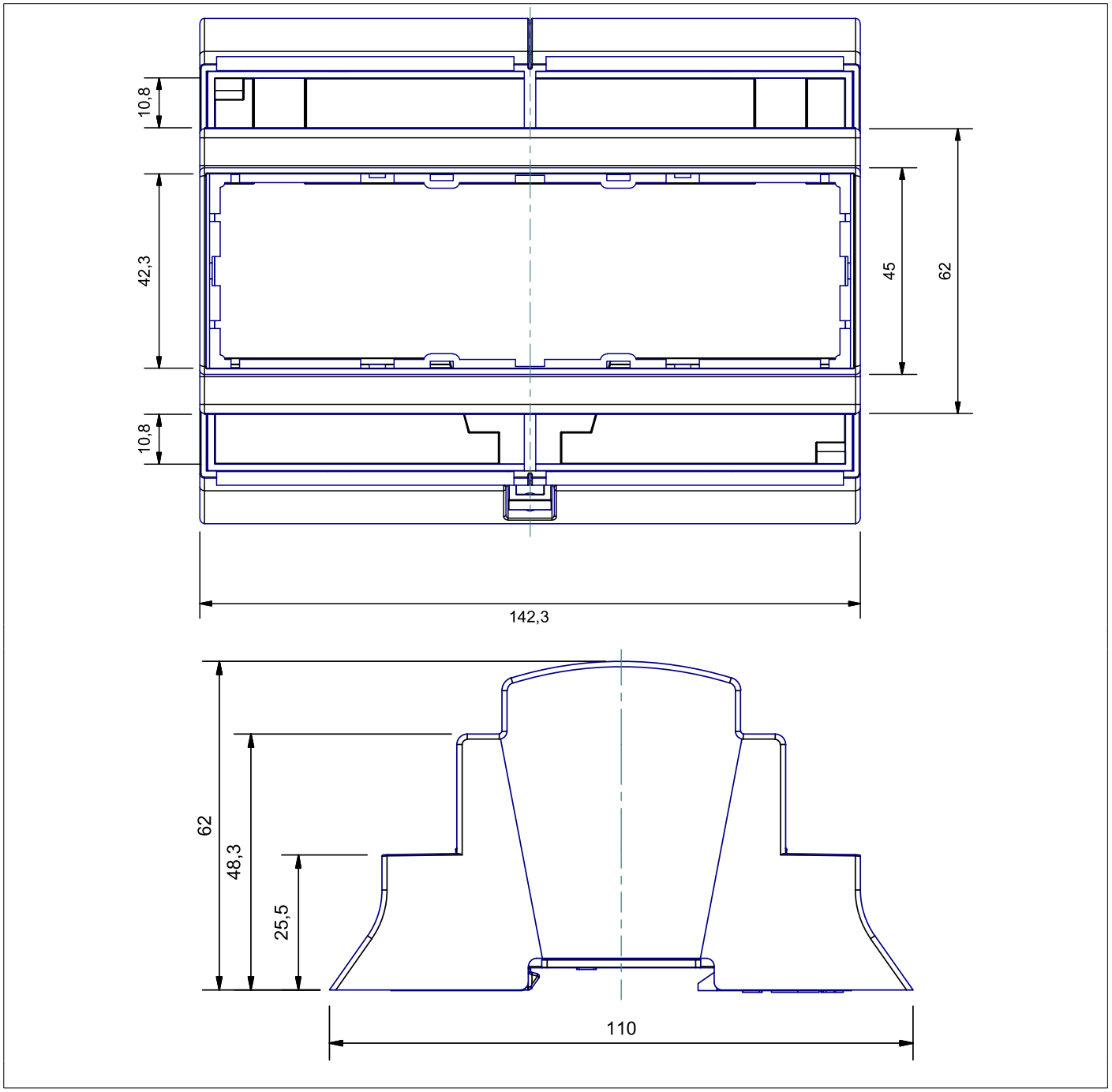


Figure: Dimensions of the housing for our serial BIG IOs XT8 modules in mm

Dimensions

Housing illustration	142.3 x 110 x 62
Color	grey RAL 7035
Material	Self-extinguishing Blend PC/ABS UL94-VO
Protection class	IP20 based on DIN 40050 / EB 60529

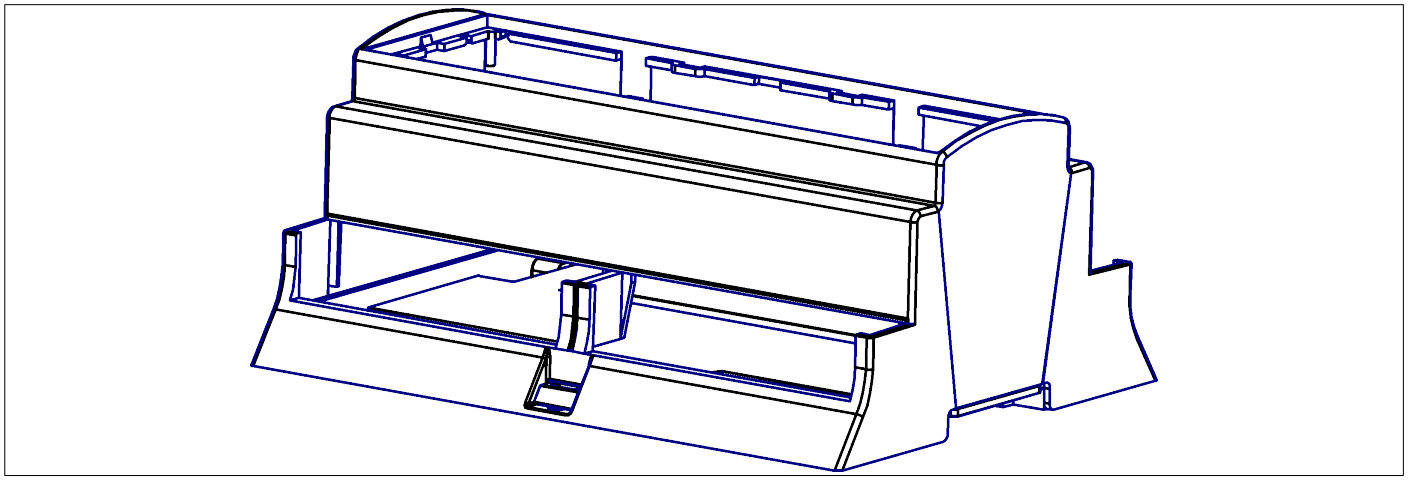


Figure: For our serial BIG IOs XT8 modules: Housing illustration in 3D

11.4 BIG IOs: RESI-xxx-SIO XT12

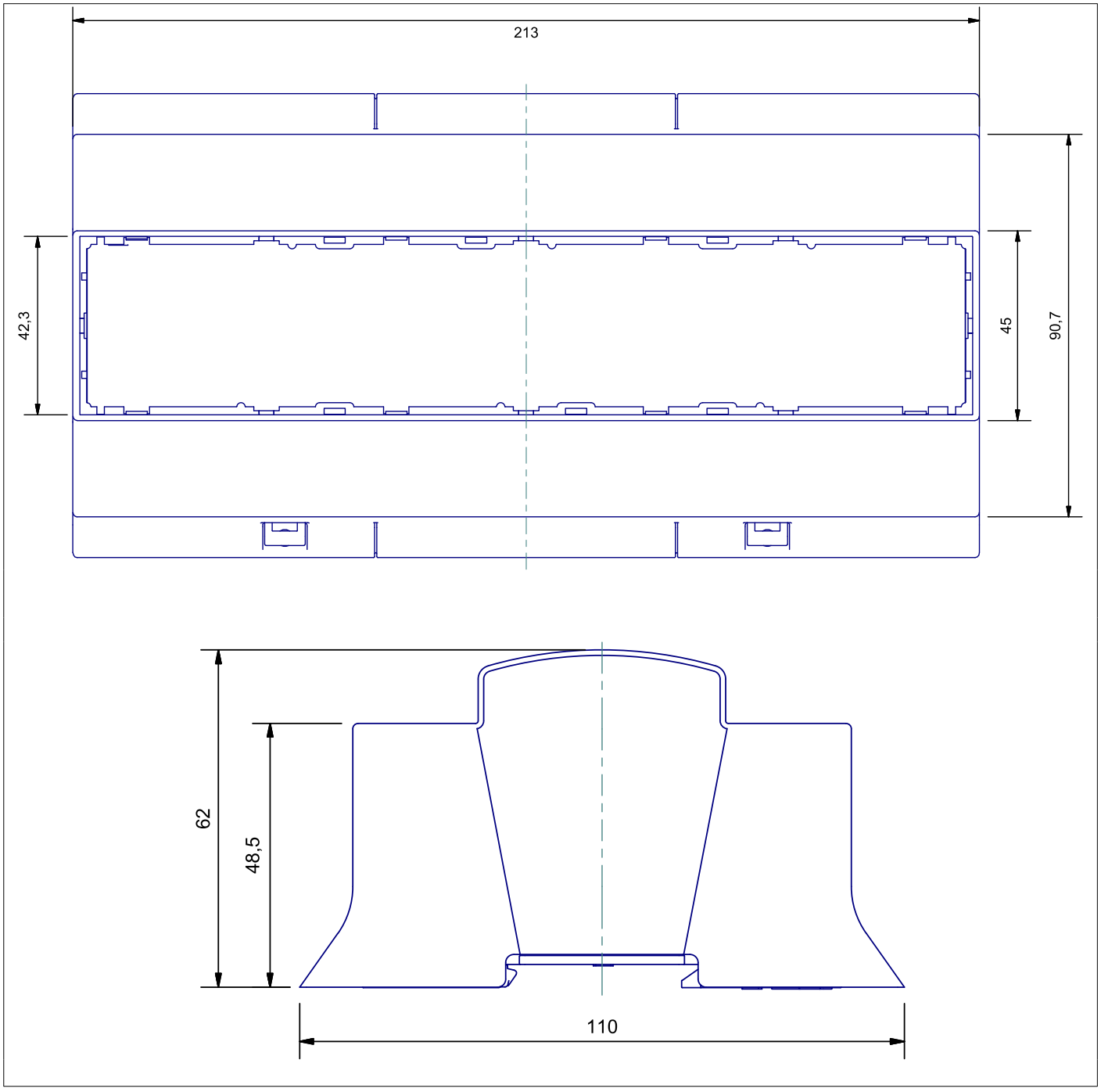


Figure: Dimensions of the housing for our serial BIG IOs XT12 modules in mm

Dimensions

Housing illustration	213 x 110 x 62
Color	grey RAL 7035
Material	Self-extinguishing Blend PC/ABS UL94-VO
Protection class	IP20 based on DIN 40050 / EB 60529

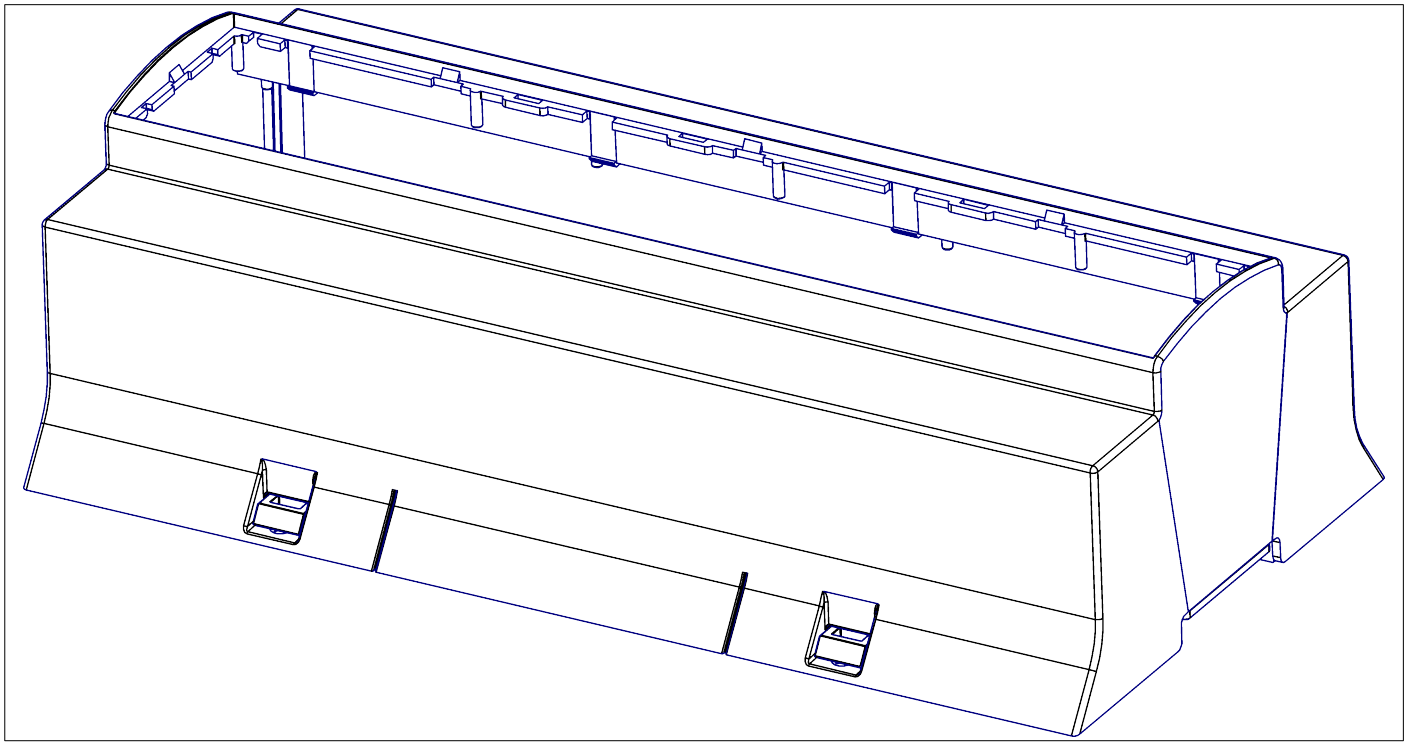
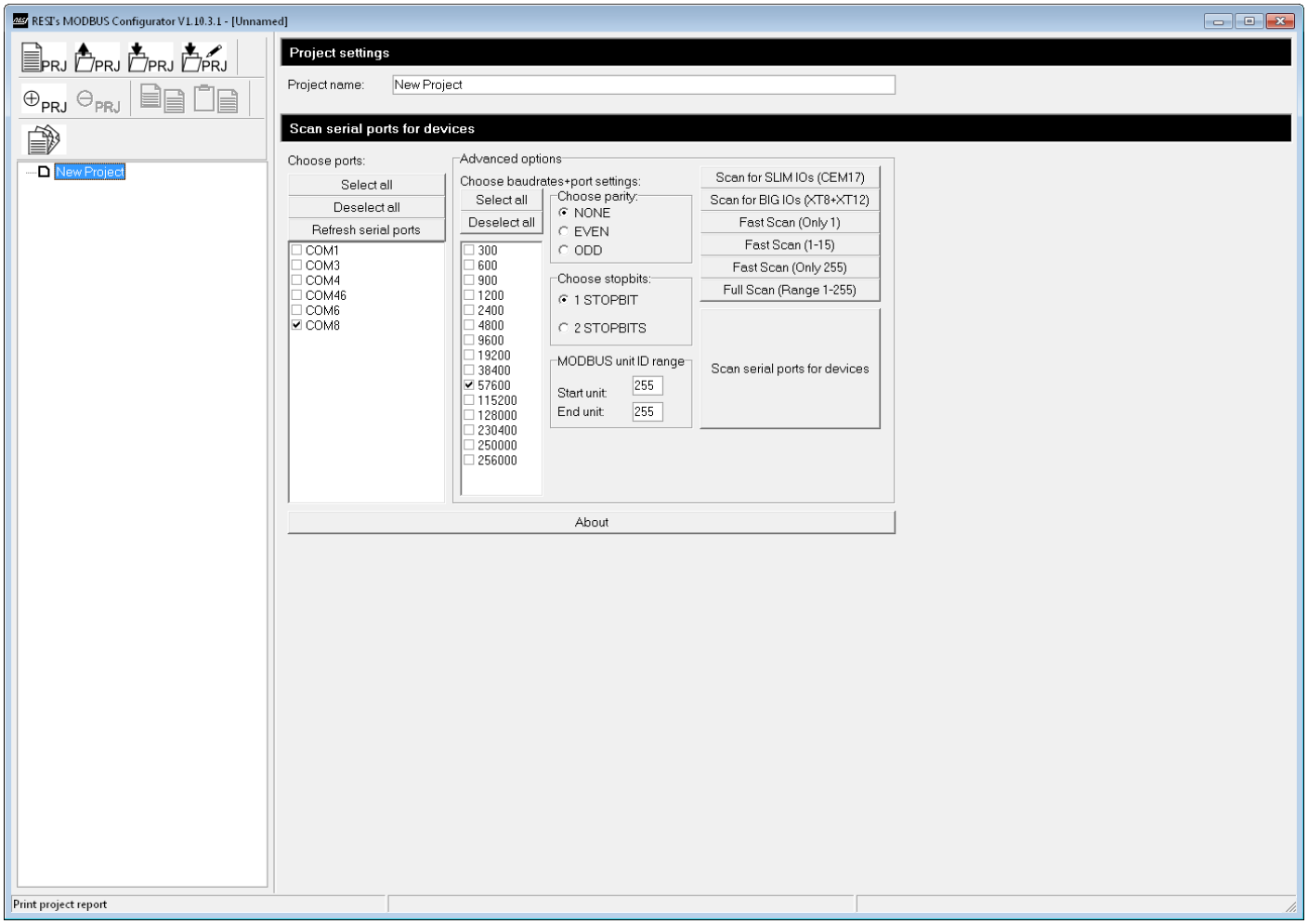


Figure: For our serial BIG IOs XT12 modules: Housing illustration in 3D

12 MODBUSConfigurator software

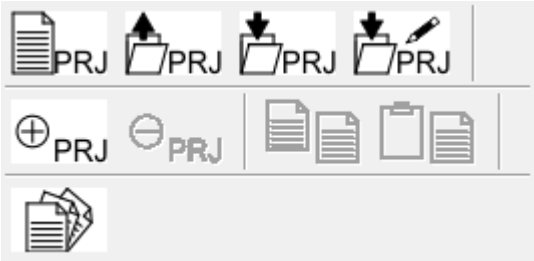
12.1 General information

We offer a free configuration & test tool named MODBUSConfigurator. This tool offers the possibility to configure and test almost all of our products. When you start the software you will see the following picture.



12.2 Main menu icons

The main menu icons provide the purpose to start a new project or open an existing project or save the current project to a new project file. You can also add some new items to a gateway like meters or DALI lamps or you can add a new gateway to an existing project. of course you can delete and configure gateway from a project. You can copy and paste objects within the existing project and you can generate a project report for documentation.



First row:

- Create a new empty project
- Open an existing project
- Save current project
- Save current project with new name

Second row:

- Add a gateway, IO module or object to the project tree
- Delete selected item from the project tree
- Copy selected item into internal clipboard
- Paste internal clipboard to project tree

Third row:

- Generate project report

12.3 Project settings

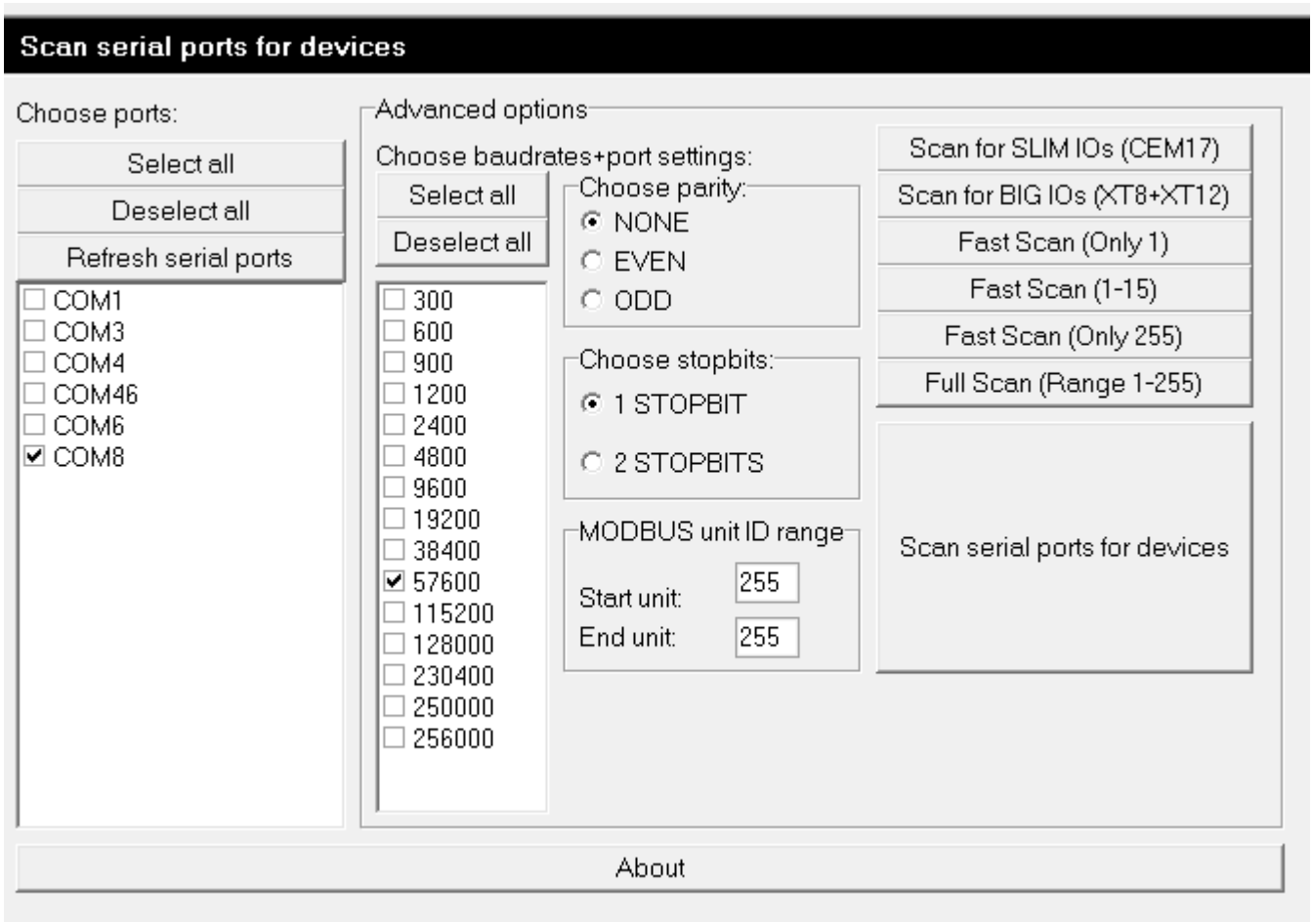
In this section you can define your special project name:

12.4 Scan for serial devices

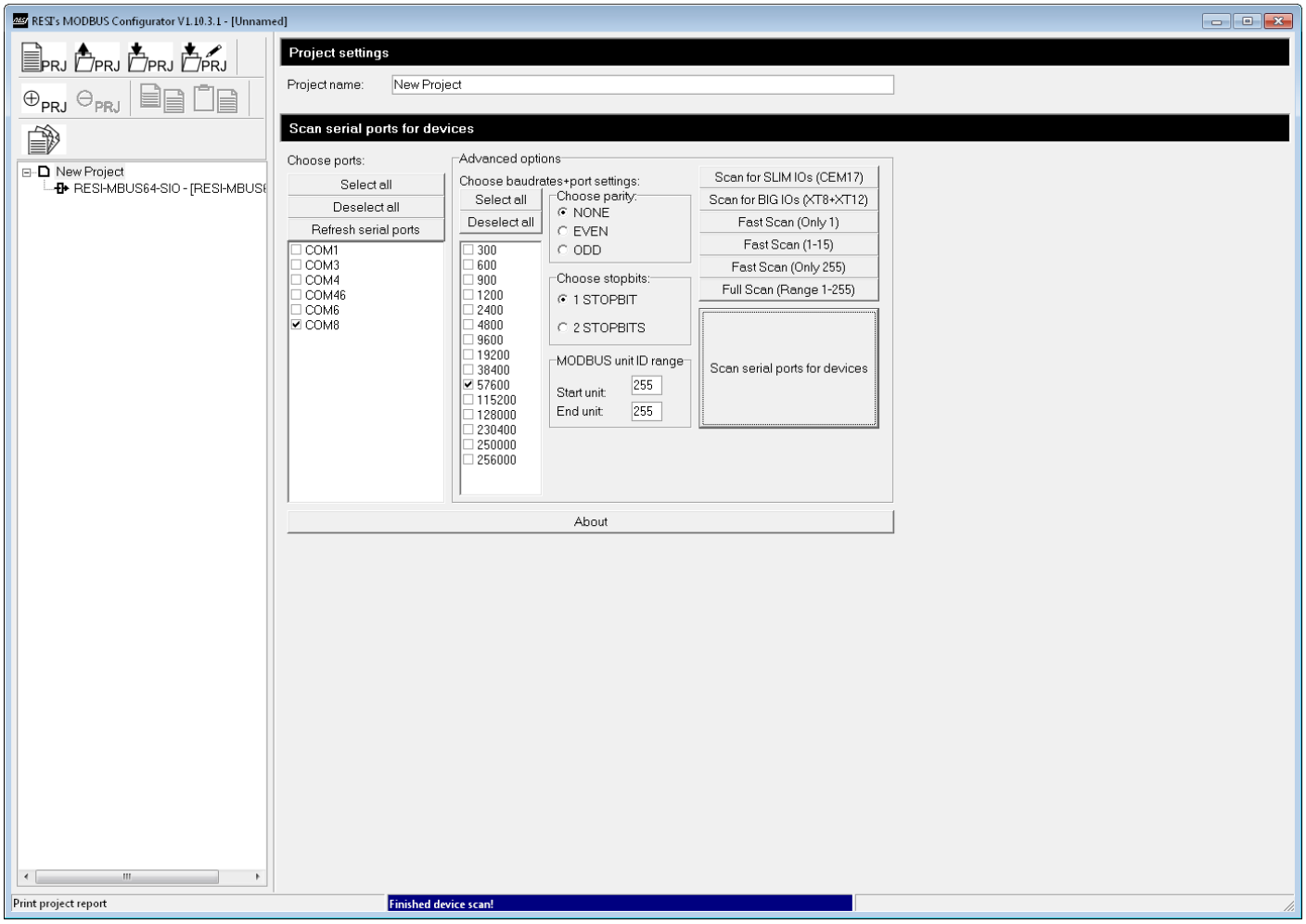
In this section you can configure an automatic search process to find all connected devices. Therefore you see on the left side the current available serial interfaces currently connected to your computer.

With the button select all you can select all of the available serial interfaces. With the button Select all you can select all serial interfaces for the automatic search process. The button Deselect all will deselect all serial interfaces for this automatic search. The button Refresh serial ports will scan again all connected serial interfaces of your computer and refresh the displayed list of serial interfaces.

In the area on the right side you will find settings for baud rates, parity and stop bits. Also you can select the range of MODBUS unit IDs which are used for this automatic search process. The automatic search for connected serial devices starts by pressing the Scan serial ports for devices button. The remaining buttons offer a quick selection for certain scenarios. The About button opens a dialog with information about the program version.

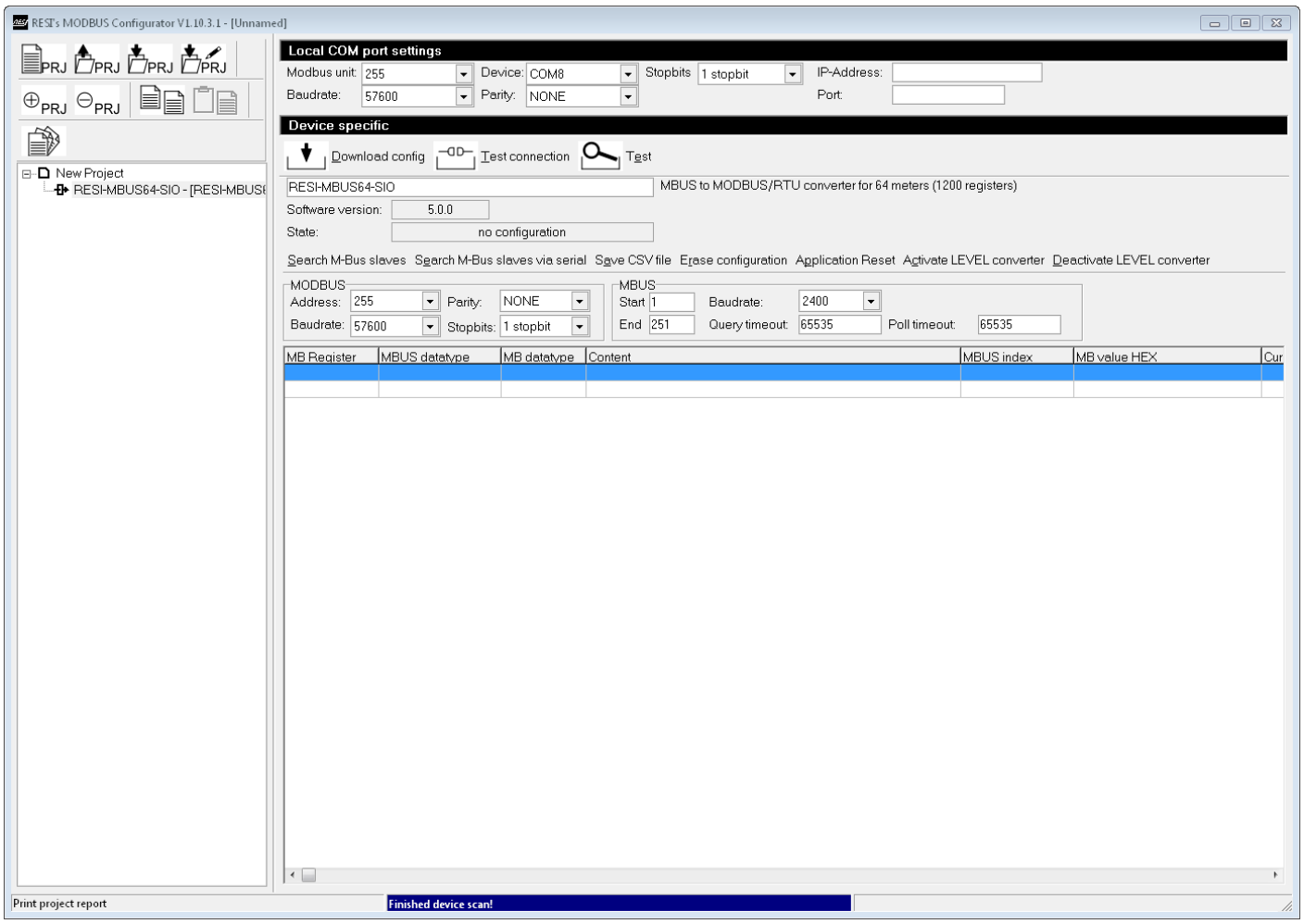


If your automatic scan was successful, the project tree will show the connected IO modules or gateways:



12.5 Configure and test a device

When you click in the project tree onto a device you can select a device. On the right side you will see all technical parameters to set up and test a specific device.



12.5.1 Local COM port settings

In this section you will see the current configured serial or IP settings to communicate with the selected device. Also you can change the MODBUS unit ID which is used for the communication protocol.

Select serial communication:

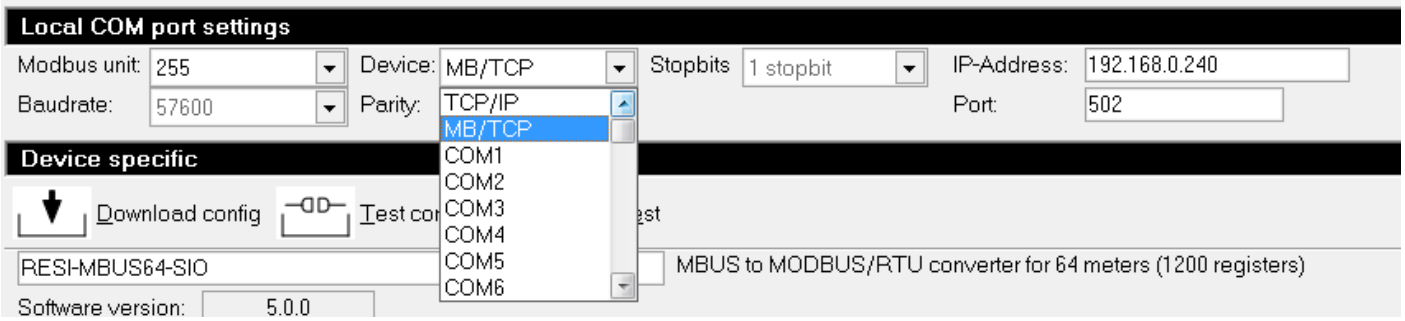
As long as you select a serial device COMx from the drop-down list Device you will use a serial interface to communicate with the connected module. Choose your desired parameters for baud rate, parity and stopbits.

Select Ethernet communication:

If you open the drop down list, you will notice two other options

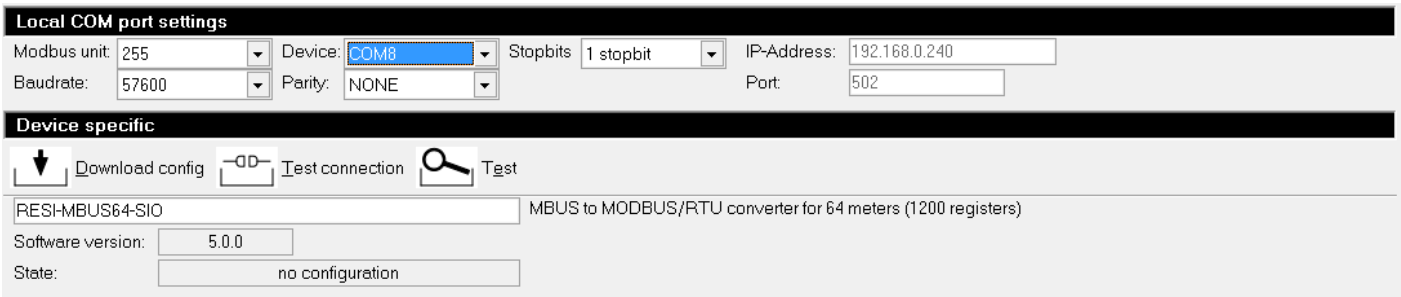
- TCP/IP: Use serial communication via TCP/IP
- MB/TCP: use MODBUS/TCP protocol via TCP/IP

First, select one of the two options, then you can enter a IP address and a socket number for the communication via Ethernet.



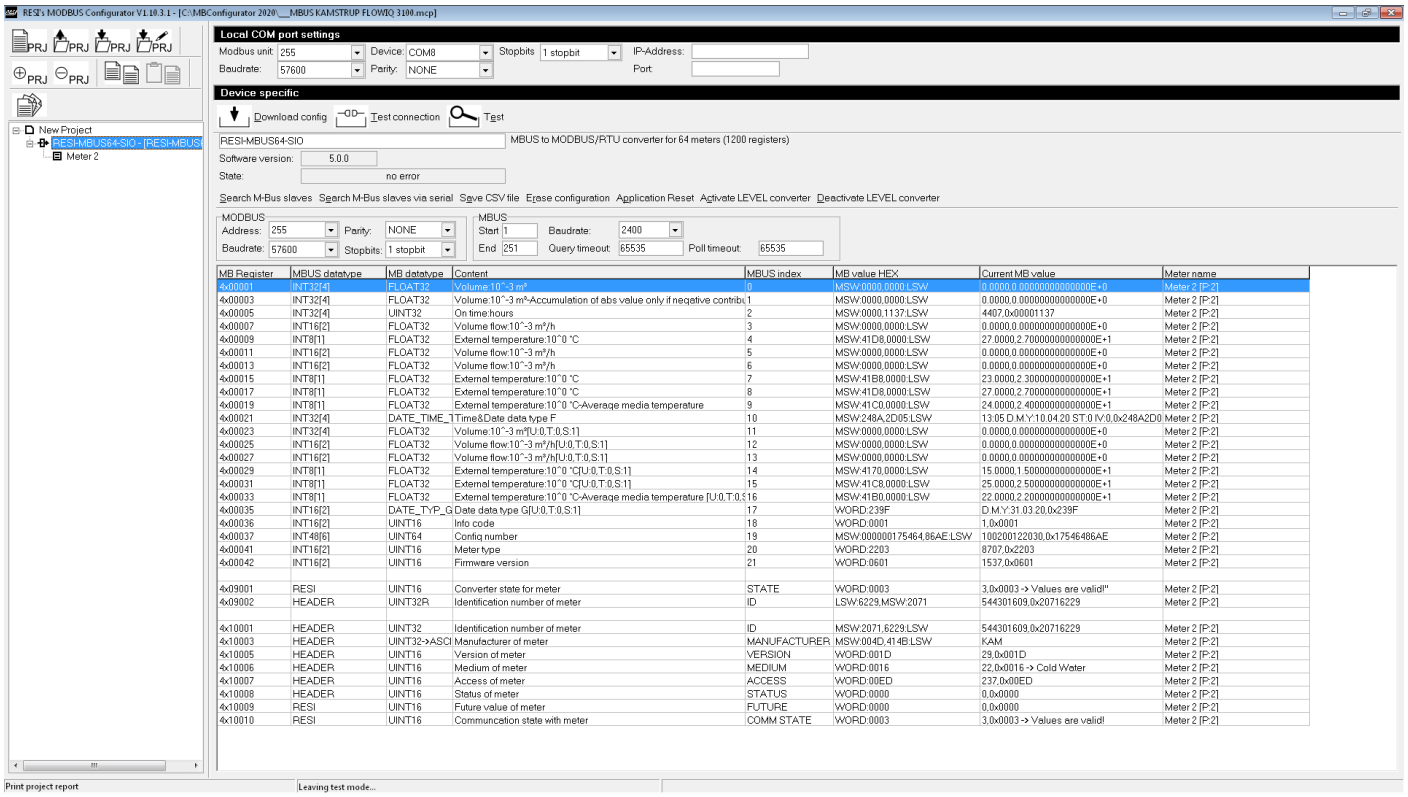
Check connection settings:

After you have defined your communication settings, you can test the communication by pressing the button test connection. If the connection is not successful established, an error dialog will pop up. If the communication is ok, the fields Software version and State will show more information about the device.



12.5.2 Device specific area

In this section you will find specific information about the connected device. In this sample we have connected a MBUS devices with one meter.



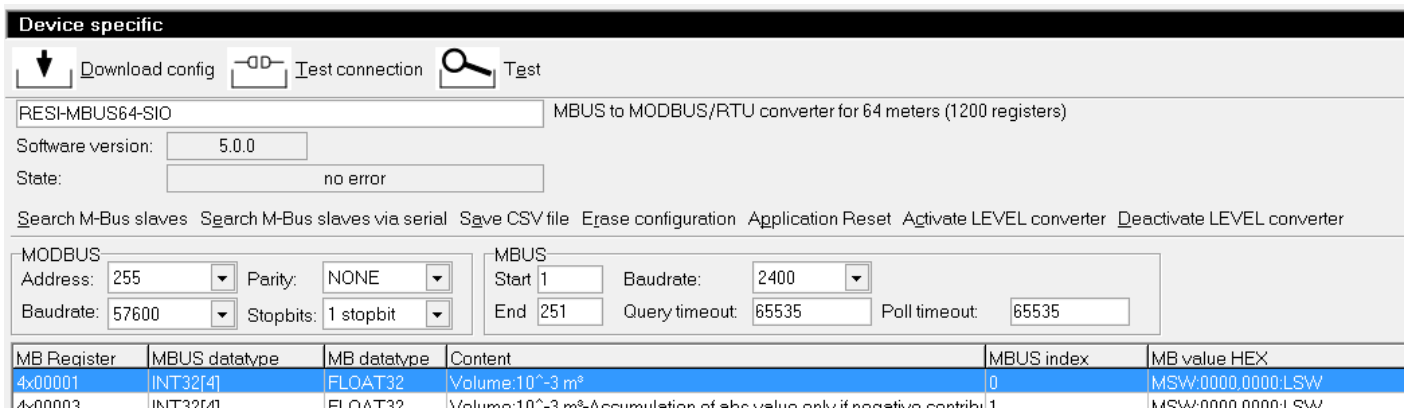
For all devices you have two options:

Download config

With this button you can download your new settings which you have selected in the device specific area into the connected module. After that you may have changed the basic configuration settings. So don't forget to change the Local COM port settings to establish communication to the module again.

Test

This button activates a cyclic test option, which will show values from the connected device. IN this case it will show the current meter values of the connected MBUS meter on your MBUS gateway.



Device specific commands:

You will also find a command area with buttons for device specific commands. In the case of our MBUS gateways you will find the functions:

- Search M-Bus slaves
- Search M-Bus svales via serial
- Save CSV file
- etc.

Please refer to the detailed documentation for each module, what the specific commands are for and how you can use them.

General device settings:

Below of the device specific command area is an area with general settings for the selected device. IN our sample case it will be:

- MODBUS address
- Baudrate
- Parity
- Stopbits
- Primary MBUS start ID
- Primary MBUS end ID
- etc.

This settings can be downloaded into the device with the button Download. Some of your modules can also upload this settings fro the device. Then they have an additional button in the device specific command area.

Value grid:

Under the specific device settings most of our module show a gird with more configuration possibilities or a grid with MODBUS registers. Again the configuration grid will be downloaded with the button Download int the device. The MODBUS values will be cyclic updated by activating the Test button.

For more details refer to the specific devices, what information the MODBUSConfigurator software will offer.

13 RESI-14RI-SIO

13.1 General information

This series of IO modules offer the following features:

- 14 digital inputs for 24-250Vac/dc signals
- Each digital input is galvanic insulated to all other digital inputs
- Each digital input is cabled via extra 2 pin removable terminal
- Galvanic insulated RS485 interface for communication with a host system

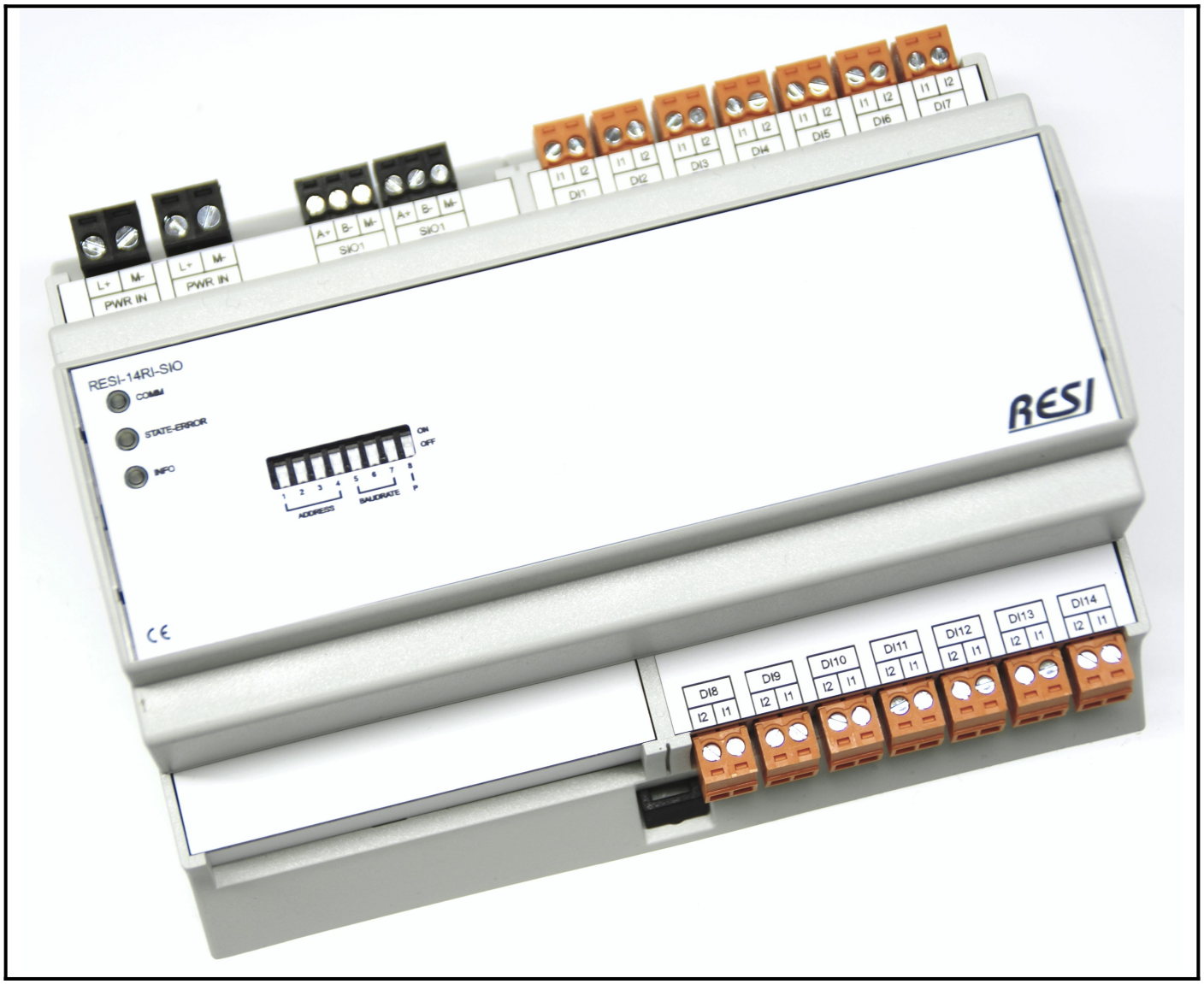


Figure: Our RESI-14RI-SIO module

13.2 Technical specification

Beside the basic technical data, which fulfil all of our IO modules, this IO modules meet the following technical specifications:

Power consumption	<0.5W
Product housing	BIG IO XT8
Product weight	255g
Digital inputs	
Total amount of inputs	14
Sampling rate	Every 5ms
DC rating	
Input voltage range	24-250V= +/-10%
Input current	per channel
	approx. 1.0mA@20V=
	approx. 1.6mA@24V=
	approx. 1.9mA@32V=
	approx. 2.1mA@250V=
Input power consumption	max. 0.6W/channel
Logic levels	0: <3V=
	1: >20V=
AC rating	
Input voltage range	24-250V= +/-10%
Input current	per channel
	approx. 1.2mA@20V~
	approx. 1.4mA@24V~
	approx. 1.8mA@48V~
	approx. 2.0mA@110V~
	approx. 2.1mA@230V~
	approx. 2.1mA@250V~
Input power consumption	max. 0.6W/channel
Logic levels	0: <3V~
	1: >20V~
Cable connection	Via 14 2-pin plug-in terminal block
Terminal type	RM3.5
Galvanic insulation	Yes, to each other digital input and to IO module
Default serial settings	
Baud rate	via DIP switch
Parity	none
Stop bit(s)	one
UnitID	255

13.3 Additional terminals & LED states

DIGITAL INPUTS	14 digital inputs for 24-250Vac/dc signals	
	Eight 2 pin plug-in terminal block	
	Terminal type:	RM3.5
	I1:	Digital input +: AC/DC signal
	I2:	Digital input -: Ground or neutral wire
		0=open or short cut
		1=AC or DC voltage between 20 and 250V
Pin layout	Pin 1:	I1
	Pin 2:	I2

INFO This LED is on, if at least one of the digital inputs is high (1).

This LED is off, if all digital inputs are low (0).

13.4 Connection diagram

13.4.1 Cabling of the digital inputs with DC signals

In the below drawing you see the cabling of the 14 digital inputs of the module with DC signals.

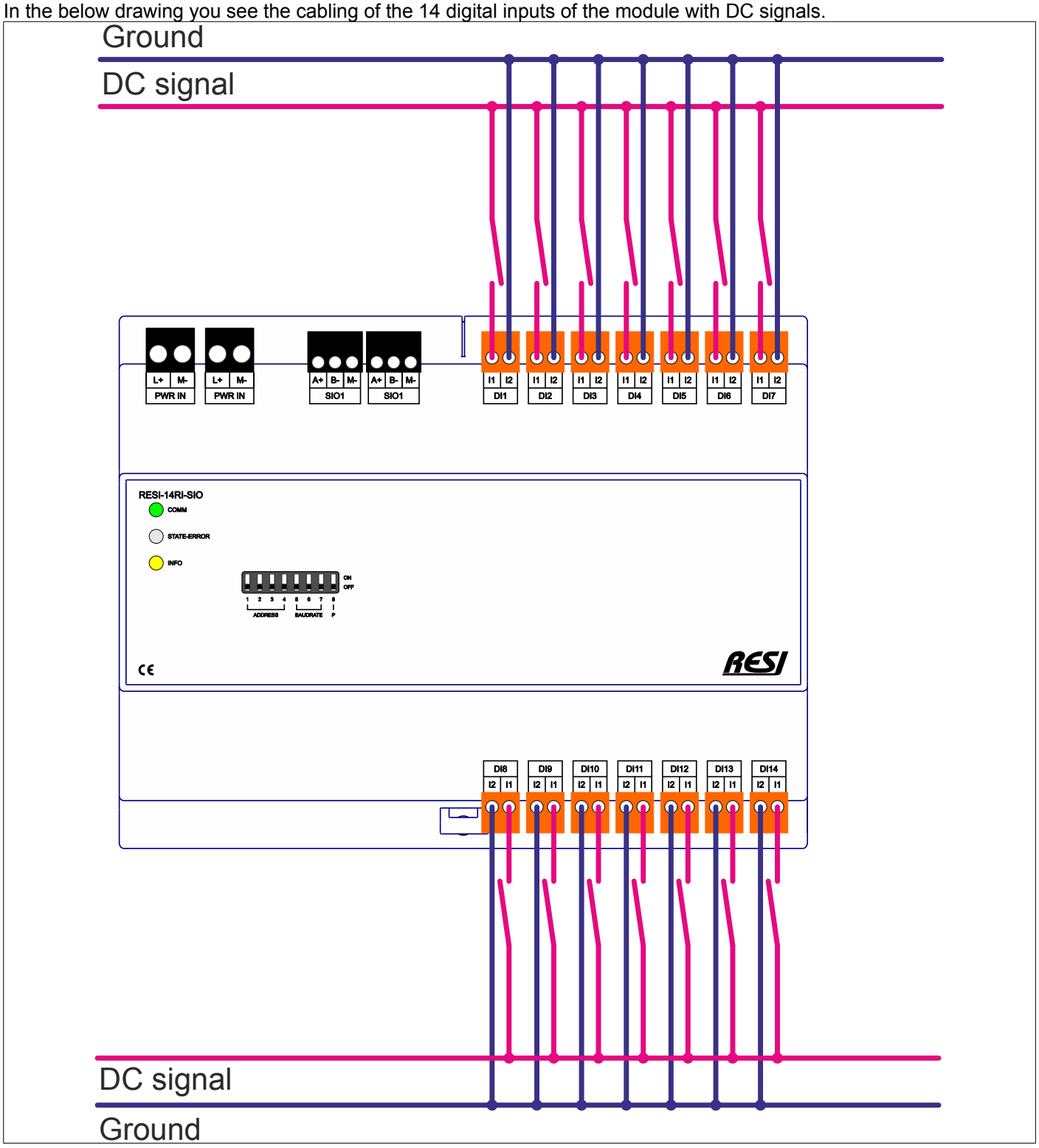


Figure: Cabling of the digital inputs of the IO module with DC signals

Don’t forget, that you can use signals from different DC power supplies for each input, because all digital inputs are galvanically insulated to each other. Also you can mix AC and DC input signals on one module!

13.4.2 Cabling of the digital inputs with AC signals

In the below drawing you see the cabling of the 14 digital inputs of the module with AC signals.

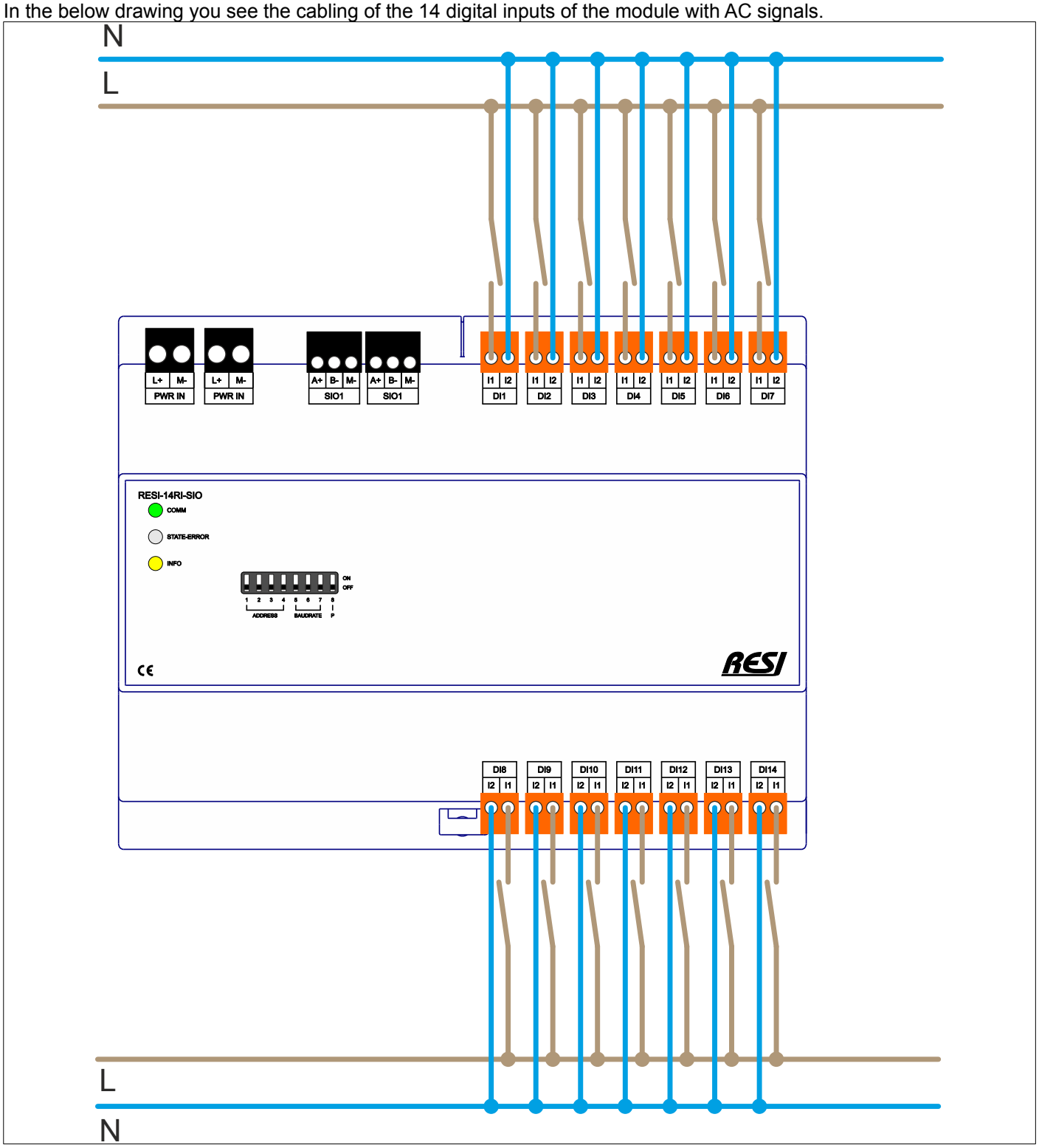


Figure: Cabling of the digital inputs of the IO module with AC signals

Don’t forget, that you can use signals from different AC power supplies for each input, because all digital inputs are galvanically insulated to each other. Also you can mix AC and DC input signals on one module!

13.5 Additional MODBUS register & coils

Here you will find only the additional MODBUS registers and coils especially for this IO module. Please refer to the description of of the standard MODBUS mapping for more details about the available basic MODBUS registers and coils.

Please refer to the external document for detailed documentation of the current MODBUS register mapping for this IO module:

RESI-L-14RI-SIO-MODBUS+ASCII-ENxx.pdf

13.6 Additional ASCII commands

Here you will find only the additional ASCII commands especially for this IO module. Please refer to the description of of the standard commands for more details about the available basic ASCII commands.

Please refer to the external document for detailed documentation of the current ASCII commands for this IO module:

RESI-L-14RI-SIO-MODBUS+ASCII-ENxx.pdf

14 RESI-S16DI8RO-SIO, RESI-S8RO-SIO

14.1 General information

This series of IO modules offer the following features:

- Only RESI-S16DI8RO-SIO: 16 digital inputs for 12-48Vdc signals
- 8 bistable relay outputs with special power relays
- Maximum switching power: max. 250Vac, max. 16A, max 200µF
- Internal FRAM memory to save the last relay position
- Automatic recovery of the correct relay position after power loss
- Remanent counter for each output counting the switching cycles of the relays
- Only RESI-S16DI8RO-SIO:
  - Stand-alone operation mode: Internal logic functions between the digital inputs and the relay outputs
  - Configure simple logic functions like switch light on/off, central light on, central light off, stairway light with off delay timer, etc. with pushbuttons
- Galvanic insulated RS485 interface for communication with a host system

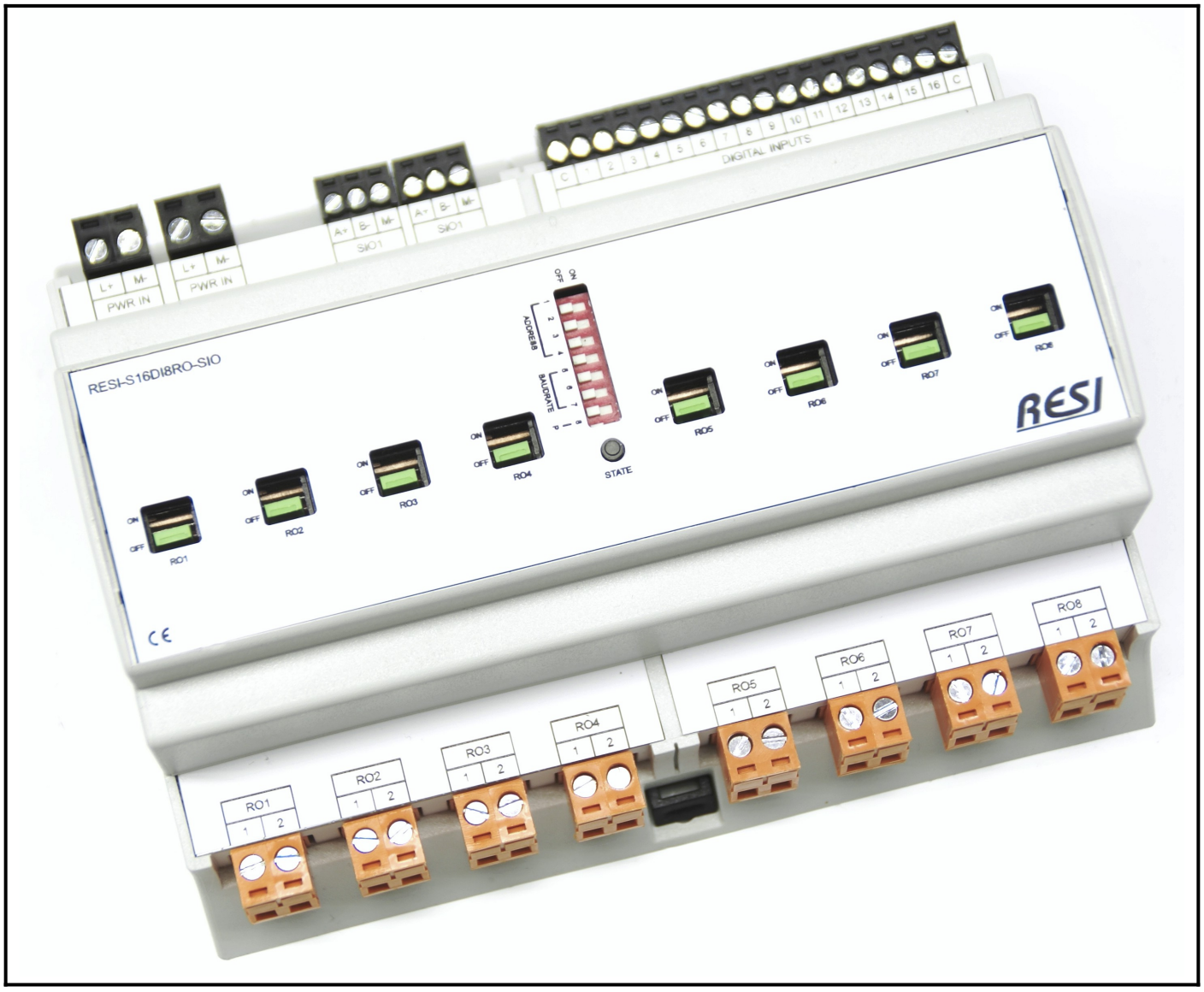


Figure: Our RESI-S16DI8RO-SIO module

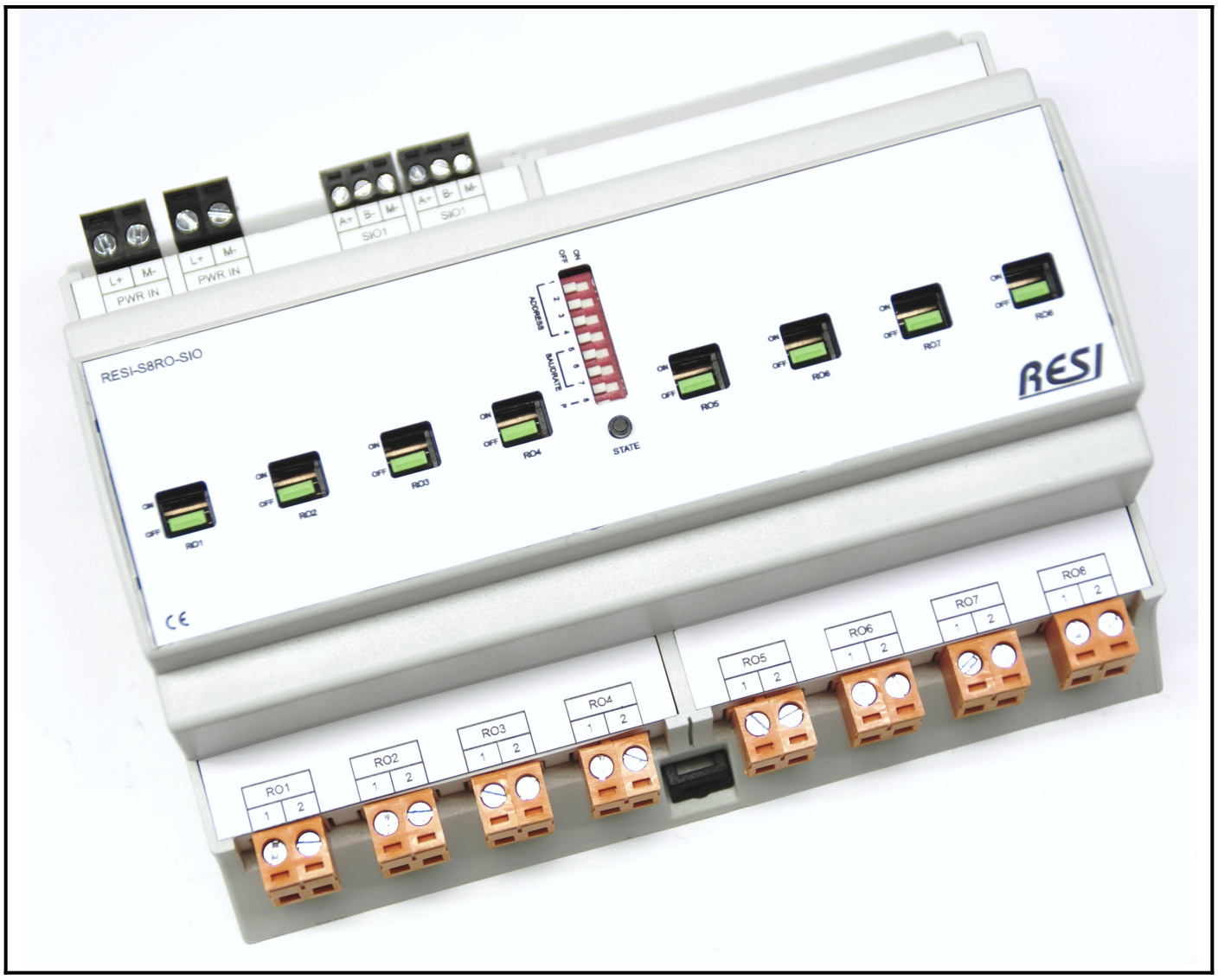


Figure: Our RESI-S8RO-SIO module

14.2 Internal logic functions

The IO module offers internal logic functions, which are handled by the module autonomous. All parameters for this logic functions are stored in the internal permanent memory FRAM. After a power loss all this configuration is not deleted and the module executes the logic functions again.
This internal logic functions can operate side by side with control commands via MODBUS/RTU or ASCII.

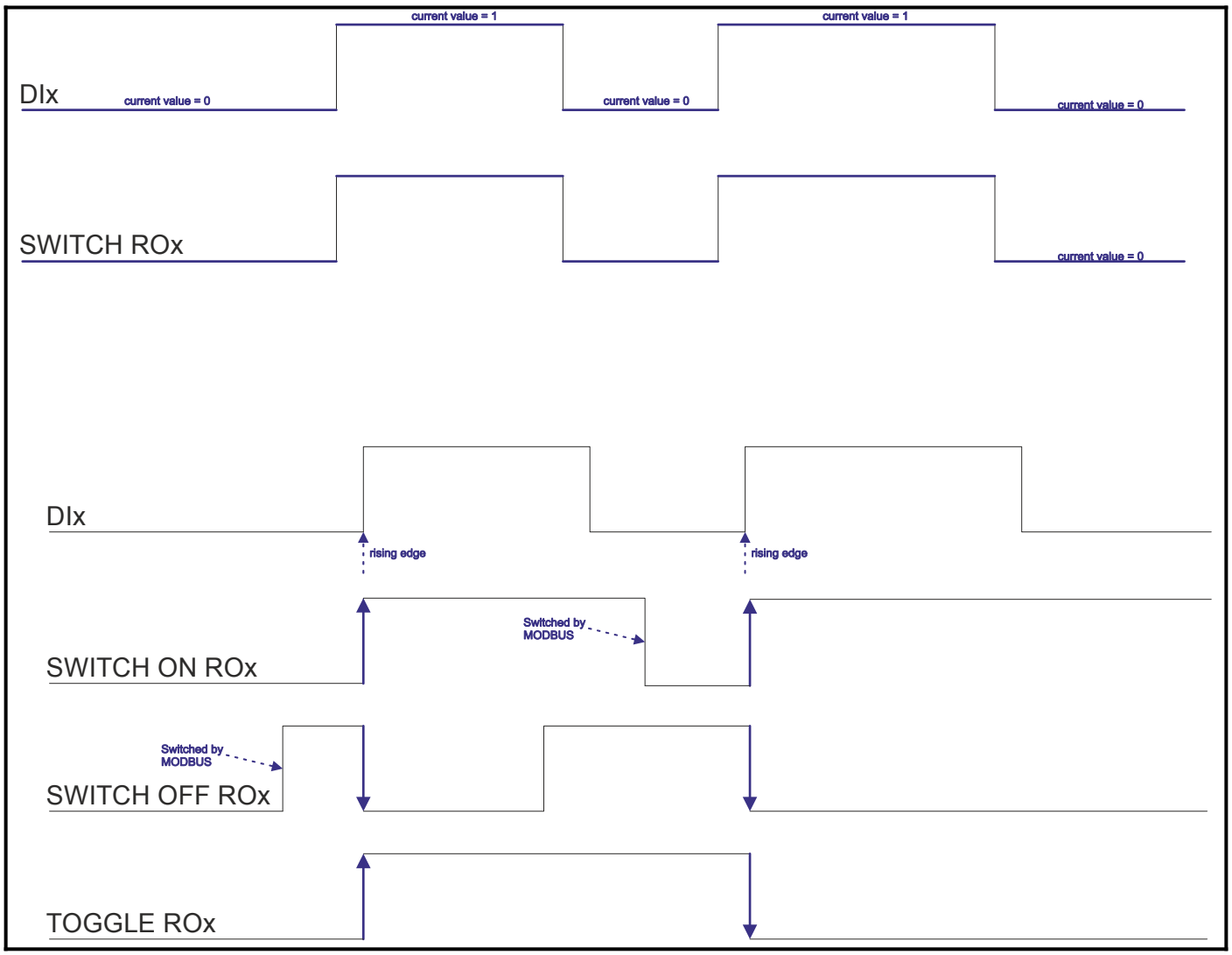


Figure: Internal logic functions

14.2.1 Switch on or off the internal logic processing

There is a general switch to enable or disable the execution of the internal logic operations. Therefore on the MODBUS/RTU interface you will find the register ENABLE LOGIC FUNCTIONS (4x21001). On the ASCII protocol the command SET SPECIAL MODE and GET SPECIAL MODE controls this feature.

Only if this register contains 1, the internal logic is executed by the module. Of course you will need a correct configuration for a desired logic function, if the module should react to a digital input.

- Activate logic function: Write to the MODBUS register ENABLE LOGIC FUNCTIONS the value 1 or execute the ASCII command SET SPECIAL MODE:1
- Deactivate logic function: Write to the MODBUS Register ENABLE LOGIC FUNCTIONS the value 0 or execute the ASCII command SET SPECIAL MODE:0
- Request the current execution status of logic function: Read out the current value in the MODBUS register ENABLE LOGIC FUNCTIONS. If this value is 1, the module executes the internal logic functions. If this value is 0, no logic functions are executed. Or you request the current status with the ASCII command GET SPECIAL MODE. If the answer is GSMODE:1,0x1, the internal logic is executed by the module. If the answer is GSMODE:0,0x0, no logic execution is active.

14.2.2 Reset internal logic

Sometimes it is very convenient to delete the complete configuration of the internal logic functions. This is handled by the ASCII command RESET SPECIAL MODE. On the MODBUS side you have to write the value 1 to the register CLEAR ALL LOGIC FUNCTIONS (4x21002). The module deletes the complete internal configuration permanently in the FRAM memory and no logic functions are executed.

14.2.3 Logic function SWITCH

This is the simplest logic function. You can map for each relay output a digital input. If this digital input is high (1), the corresponding output relay will be switched on. If this digital input is low (0), the mapped output relay will be switched off.

Example: Switch the output relay RO1 on and off with the digital input DI1

Over the ASCII interface you have to send the following commands:

PC->IO: #SET SWITCH1:0x0001

IO->PC: #OK

PC->IO: #SET SPECIAL MODE:1

IO->PC: #OK

Via the MODBUS interface you have to set the following registers:

PC->IO: Write value 0x0001 in MODBUS register SWITCH RO1 (4x20001)

PC->IO: Write value 0x0001 in MODBUS register ENABLE LOGIC FUNCTIONS (4x21001)

The other relay outputs are not affected by this configuration.

Example: Switch the output relay DO1 with digital input DI1 on and off, with DI2 the relay RO2, with DI3 the relay RO3 and so on.

Over the ASCII interface you have to send the following commands:

PC->IO: #SET SWITCH1:0x0001

IO->PC: #OK

PC->IO: #SET SWITCH2:0x0002
IO->PC: #OK
PC->IO: #SET SWITCH3:0x0004
IO->PC: #OK
PC->IO: #SET SWITCH4:0x0008
IO->PC: #OK
PC->IO: #SET SWITCH5:0x0010
IO->PC: #OK
PC->IO: #SET SWITCH6:0x0020
IO->PC: #OK
PC->IO: #SET SWITCH7:0x0040
IO->PC: #OK
PC->IO: #SET SWITCH8:0x0080
IO->PC: #OK
PC->IO: #SET SPECIAL MODE:1
IO->PC: #OK

Via the MODBUS interface you have to set the following registers:
PC->IO: Write value 0x0001 to MODBUS register SWITCH RO1 (4x20001)
PC->IO: Write value 0x0002 to MODBUS register SWITCH RO2 (4x20002)
PC->IO: Write value 0x0004 to MODBUS register SWITCH RO3 (4x20003)
PC->IO: Write value 0x0008 to MODBUS register SWITCH RO4 (4x20004)
PC->IO: Write value 0x0010 to MODBUS register SWITCH RO5 (4x20005)
PC->IO: Write value 0x0020 to MODBUS register SWITCH RO6 (4x20006)
PC->IO: Write value 0x0040 to MODBUS register SWITCH RO7 (4x20007)
PC->IO: Write value 0x0080 to MODBUS register SWITCH RO8 (4x20008)
PC->IO: Write value 0x0001 to MODBUS register ENABLE LOGIC FUNCTIONS (4x21001)

Now you can switch on or off all 8 relay outputs RO1 to RO8 with the first 8 digital inputs DI1 to DI8.

14.2.4 Logic function SWITCH ON

This logic function checks the status of the mapped digital inputs and sets the corresponding relay output to a defined state. In case of the function SWITCH ON to 1, if the module detects a rising edge on one of the mapped digital inputs.

Example: The relay output RO1 is switched on by one of the four digital inputs DI1, DI2, DI3 and DI4

Over the ASCII interface you have to send the following commands:

PC->IO: #SET SWITCH ON1:0x000F

IO->PC: #OK

PC->IO: #SET SPECIAL MODE:1

IO->PC: #OK

Via the MODBUS interface you have to set the following registers:

PC->IO: Write value 0x000F to MODBUS register SWITCH ON RO1 (4x20017)

PC->IO: Write value 0x0001 to MODBUS register ENABLE LOGIC FUNCTIONS (4x21001)

The other relay outputs are not affected by this configuration.

Example: Central light on with digital input DI16

Over the ASCII interface you have to send the following commands:

PC->IO: #SET SWITCH ON1:0x8000

IO->PC: #OK

PC->IO: #SET SWITCH ON2:0x8000

IO->PC: #OK

PC->IO: #SET SWITCH ON3:0x8000

IO->PC: #OK

PC->IO: #SET SWITCH ON4:0x8000

IO->PC: #OK

PC->IO: #SET SWITCH ON5:0x8000

IO->PC: #OK

PC->IO: #SET SWITCH ON6:0x8000

IO->PC: #OK

PC->IO: #SET SWITCH ON7:0x8000

IO->PC: #OK

PC->IO: #SET SWITCH ON8:0x8000

IO->PC: #OK

PC->IO: #SET SPECIAL MODE:1

IO->PC: #OK

Via the MODBUS interface you have to set the following registers:

PC->IO: Write value 0x8000 to MODBUS register SWITCH ON RO1 (4x20017)

PC->IO: Write value 0x8000 to MODBUS register SWITCH ON RO2 (4x20018)

PC->IO: Write value 0x8000 to MODBUS register SWITCH ON RO3 (4x20019)

PC->IO: Write value 0x8000 to MODBUS register SWITCH ON RO4 (4x20020)

PC->IO: Write value 0x8000 to MODBUS register SWITCH ON RO5 (4x20021)

PC->IO: Write value 0x8000 to MODBUS register SWITCH ON RO6 (4x20022)

PC->IO: Write value 0x8000 to MODBUS register SWITCH ON RO7 (4x20023)

PC->IO: Write value 0x8000 to MODBUS register SWITCH ON RO8 (4x20024)

PC->IO: Write value 0x8000 to MODBUS register ENABLE LOGIC FUNCTIONS (4x21001)

If you connect a push button switch to the digital input 16 and press this button, all eight relay outputs are switched on immediately. If you don't press the button, you can switch each of the eight relay on or off via MODBUS or ASCII protocol

14.2.5 Logic function SWITCH OFF

This logic function checks the status of the mapped digital inputs and sets the corresponding relay output to a defined state. In case of the function SWITCH OFF to 0, if the module detects a rising edge on one of the mapped digital inputs.

Example: Switch off relay output RO2 with one of the three digital inputs DI1, DI3, DI6

Over the ASCII interface you have to send the following commands:

Bit 0 stands for DI1 -> 1

Bit 2 stands for DI3 -> 4

Bit 5 stands for DI6 -> 32

Results in 1+4+32 -> 37

PC->IO: #SET SWITCH OFF2:37

IO->PC: #OK

PC->IO: #SET SPECIAL MODE:1

IO->PC: #OK

Via the MODBUS interface you have to set the following registers:

PC->IO: Write value 37 to MODBUS register SWITCH OFF RO2 (4x20026)

PC->IO: Write value 0x0001 to MODBUS register ENABLE LOGIC FUNCTIONS (4x21001)

The other relay outputs are not affected by this configuration.

Example: Central light off with DI15

Over the ASCII interface you have to send the following commands:

PC->IO: #SET SWITCH OFF1:0x4000

IO->PC: #OK

PC->IO: #SET SWITCH OFF2:0x4000

IO->PC: #OK

PC->IO: #SET SWITCH OFF3:0x4000

IO->PC: #OK

PC->IO: #SET SWITCH OFF4:0x4000

IO->PC: #OK

PC->IO: #SET SWITCH OFF5:0x4000

IO->PC: #OK

PC->IO: #SET SWITCH OFF6:0x4000

IO->PC: #OK

PC->IO: #SET SWITCH OFF7:0x4000

IO->PC: #OK

PC->IO: #SET SWITCH OFF8:0x4000

IO->PC: #OK

PC->IO: #SET SPECIAL MODE:1

IO->PC: #OK

Via the MODBUS interface you have to set the following registers:

PC->IO: Write value 0x4000 to MODBUS register SWITCH OFF RO1 (4x20025)

PC->IO: Write value 0x4000 to MODBUS register SWITCH OFF RO2 (4x20026)

PC->IO: Write value 0x4000 to MODBUS register SWITCH OFF RO3 (4x20027)

PC->IO: Write value 0x4000 to MODBUS register SWITCH OFF RO4 (4x20028)

PC->IO: Write value 0x4000 to MODBUS register SWITCH OFF RO5 (4x20029)

PC->IO: Write value 0x4000 to MODBUS register SWITCH OFF RO6 (4x20030)

PC->IO: Write value 0x4000 to MODBUS register SWITCH OFF RO7 (4x20031)

PC->IO: Write value 0x4000 to MODBUS register SWITCH OFF RO8 (4x20032)

PC->IO: Write value 0x0001 to MODBUS register ENABLE LOGIC FUNCTIONS (4x21001)

The other relay outputs are not affected by this configuration. If you connect a pushbutton switch to digital input DI15, all eight relay outputs are switched immediately to 0, if the button is pressed. If the button is released, you can switch on or off each output relay via the MODBUS or ASCII protocol.

14.2.6 Logic function TOGGLE

This logic function checks the status of the mapped digital inputs and sets the corresponding relay output to a defined state. In case of the function TOGGLE, the module inverts the current state of the relay output, if the module detects a rising edge on one of the mapped digital inputs.

Example: Toggle switch: With one of the two digital inputs DI1, DI2 we want to invert the relay output RO4.

Over the ASCII interface you have to send the following commands:

Bit 0 stands for DI1 -> 1

Bit 1 stands for DI2 -> 2

Results in 1+2 -> 3

PC->IO: #SET TOGGLE4:3

IO->PC: #OK

PC->IO: #SET SPECIAL MODE:1

IO->PC: #OK

Via the MODBUS interface you have to set the following registers:

PC->IO: Write value 3 to MODBUS register TOGGLE RO4 (4x20012)

PC->IO: Write value 0x0001 to MODBUS register ENABLE LOGIC FUNCTIONS (4x21001)

The other relay outputs are not affected by this configuration. If you connect two push buttons to the digital inputs DI1 and DI2 and press one of them, the current status of the relay output RO4 is inverted.

14.2.7 Logic function PULSE

This logic function checks the status of the mapped digital inputs and sets the corresponding relay output to a defined state. In case of the function PULSE, the module starts an off delay timer with the time span of PULSE TIME on, if the module detects a rising edge on one of the mapped digital inputs.

Example: Stairway lighting: With one of the two digital inputs DI1, DI2 we want to switch on the output relay RO1 for 30 seconds.

Over the ASCII interface you have to send the following commands:

Bit 0 stands for DI1 -> 1

Bit 1 stands for DI2 -> 2

Results in 1+2 -> 3

PC->IO: #SET PULSE4:3

IO->PC: #OK

The time is defined in 1/10s. So the value 300 defines a time of 30 seconds.

PC->IO: #SET PULSE TIME4:300

IO->PC: #OK

PC->IO: #SET SPECIAL MODE:1

IO->PC: #OK

Via the MODBUS interface you have to set the following registers:

PC->IO: Write value 3 to MODBUS register PULSE RO1 (4x20033)

PC->IO: Write value 300 as a 32 bit value to the two registers PULSE TIME RO1 4x20065-4x20066.

The number 0x12345678 will be divided into two 16 bit values and stored in this way:

4x200065:0x1234 and 4x20066:0x5678

300 as hexadecimal number is 0x0000012C.

PC->IO: Write value 0x0000 to MODBUS register PULSE TIME RO1 (4x20065)

PC->IO: Write value 0x012C to MODBUS register PULSE TIME RO1 (4x20066)

or:

PC->IO: Write value 300 as a 32 bit value to the two registers PULSE TIME RO1 4x20081-4x20082

The number 0x12345678 will be divided into two 16 bit values and stored in this way:

4x200081:0x5678 and4x20066:0x1234

300 as hexadecimal number is 0x0000012C.

PC->IO: Write value 0x012C to MODBUS register PULSE TIME RO1 (4x20081)

PC->IO: Write value 0x0000 to MODBUS register PULSE TIME RO1 (4x20082)

PC->IO: Write value 0x0001 to MODBUS register ENABLE LOGIC FUNCTIONS (4x21001)

The other relay outputs are not affected by this configuration. If you connect two push buttons to the digital inputs DI1 and DI2 and you press one of the two buttons, the relay output RO4 will be on for 30 seconds. After this time span the relay output will be switched off automatically. If you press one of the two buttons again, if the output relay is on, the time span of 30 seconds starts again.

14.3 Technical specification

Beside the basic technical data, which fulfil all of our IO modules, this IO modules meet the following technical specifications:

Power consumption	
RESI-S16DI8RO-SIO	<2.0W
RESI-S8RO-SIO	<2.0W

Product housing	
RESI-S16DI8RO-SIO	BIG IO XT8
RESI-S8RO-SIO	BIG IO XT8

Product weight	
RESI-S16DI8RO-SIO	565g
RESI-S8RO-SIO	555g

Digital inputs	
only RESI-S16DI8RO-SIO	
Total amount of inputs	16
Sampling rate	Every 5ms
Input voltage range	12-48V= +/-10%
Input current	approx. 1mA per channel
Logic levels	0: <3V=
	1: >5V=
Cable connection	Via 18-pin plug-in terminal block
Terminal type	RM3.5
Galvanic insulation	No

Relay outputs	
Number of outputs	8 bistable relays
	for socket-outlets and light applications
Relay type	Bistable with manual operation
Incandescent electric lamp load	Max 4.800 W
Capacitive load	Max. 200µF
Maximum voltage	250Vac
Maximum current	16A
Mechanical lifetime	10^6 cycles of operation
Contact material	$AgSnO_2$
Insulation	Creepage and clearance distance 8mm
Cable connection	Via 8 2-pin plug-in terminal blocks
Terminal type	RM5
Galvanic insulation	Yes, with the relay

Output power per channel:	
Incandescent lamp	4.800 W
Fluorescent lamp not compensated	5.000 W
Fluorescent lamp parallel compensated	2.500 W / 200 µF
Fluorescent lamp duo-combination	2 x 5.000 W
Halogen lamp (230VAC)	5.000 W
Low voltage halogen lamp with transformer	2.000 VA
Mercury arc sodium discharge lamp not compensated	5.000 W
Mercury arc sodium discharge lamp parallel compensated	5.000 W / 200 µF
Dulux lamp not compensated	4.000 W
Dulux lamp parallel compensated	3.000 W / 200 µF

Default serial settings	
Baud rate	via DIP switch
Parity	none
Stop bit(s)	one
UnitID	255

14.4 Additional terminals & LED states

DIGITAL INPUTS	16 digital inputs for 12-48Vdc signals	
	One 18 pin plug-in terminal block	
	Terminal type: RM3.5	
	C:	Ground of the module
	DI1-DI16:	Digital inputs
		0=open or GND,
		1=+12Vdc..+48Vdc
Pin layout	Pin 1:	C=GND
	Pin 2:	1=DI1
	Pin 3:	2=DI2
	Pin 4:	3=DI3
	Pin 5:	4=DI4
	Pin 6:	5=DI5
	Pin 7:	6=DI6
	Pin 8:	7=DI7
	Pin 9:	8=DI8
	Pin 10:	9=DI9
	Pin 11:	10=DI10
	Pin 12:	11=DI11
	Pin 13:	12=DI12
	Pin 14:	13=DI13
	Pin 15:	14=DI14
	Pin 16:	15=DI15
	Pin 17:	16=DI16
	Pin 18:	C=GND

RELAY OUTPUTS	8 bistable relays for max 250Vac signals	
	Eight 2 pin plug-in terminal blocks for Form A relay	
	Terminal type:	RM5
	1:	Switching contact of the relay +
	2:	Switching contact of the relay -
Pin layout	Pin 1:	1=Switching contact of the relay +
	Pin 2:	2=Switching contact of the relay -

14.5 Connection diagram

14.5.1 Cabling of the digital inputs

Only for RESI-S16DI8RO-SIO: In the below drawing you see the cabling of the 16 digital inputs of the module. Both terminals C are internally connected to the ground signal.

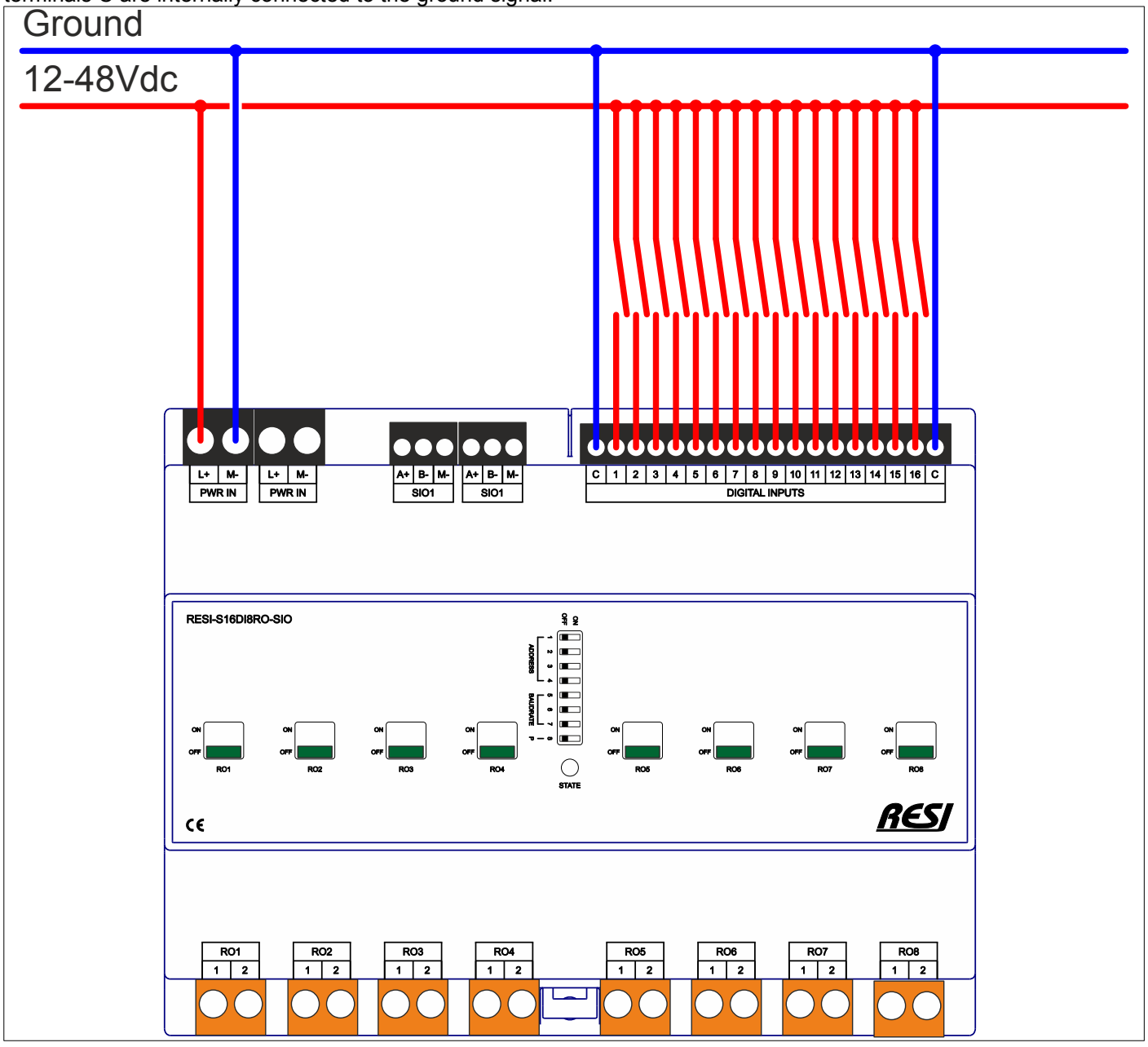


Figure: Connecting the digital inputs to the IO module

14.5.2 Cabling of the bistable relay outputs

In the below drawing the cabling of the bistable relay outputs is shown.

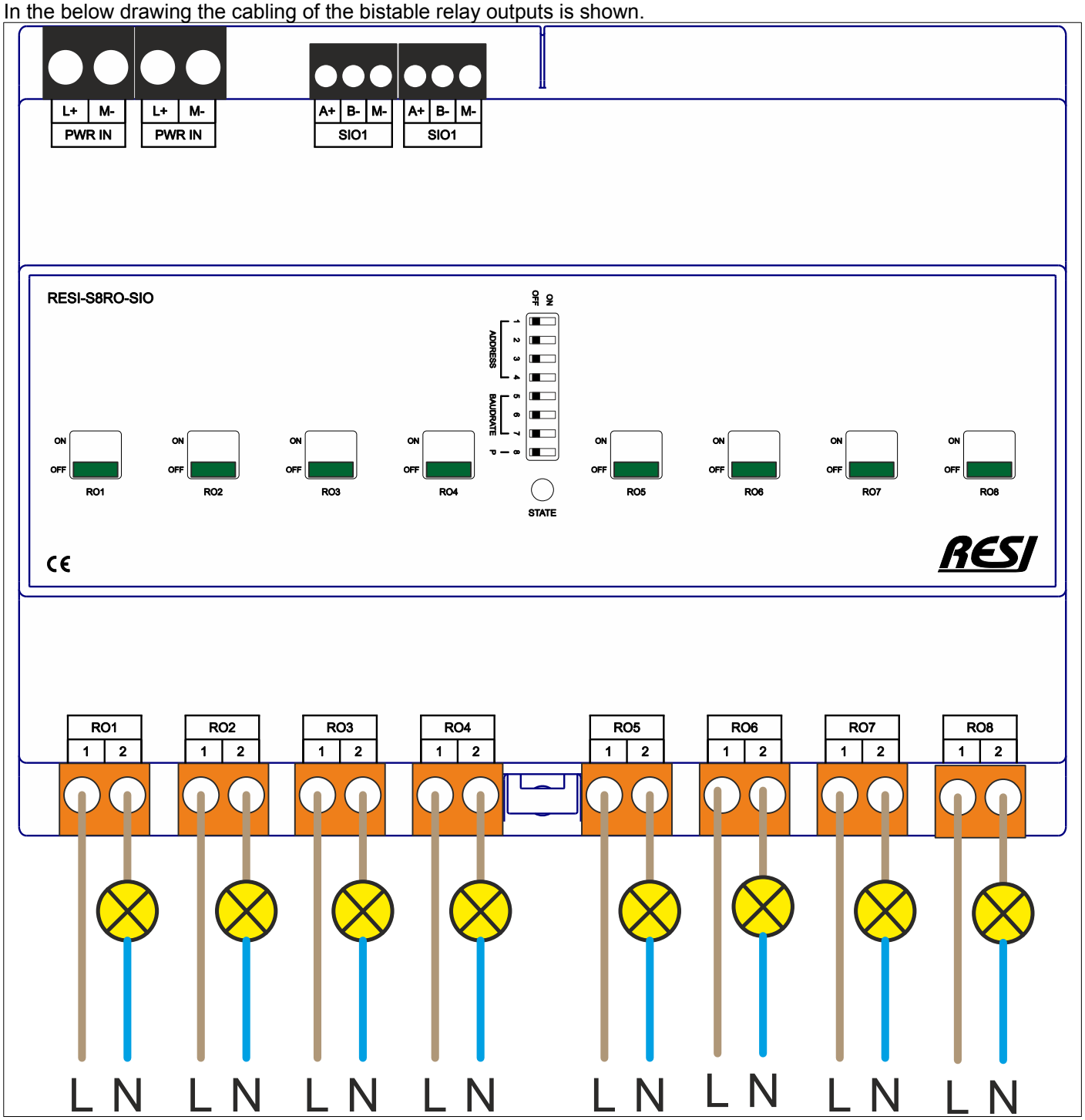


Figure: Connecting the bistable relay outputs to the IO module

14.6 Additional MODBUS register & coils

Here you will find only the additional MODBUS registers and coils especially for this IO module. Please refer to the description of of the standard MODBUS mapping for more details about the available basic MODBUS registers and coils.

Please refer to the external document for detailed documentation of the current MODBUS register mapping for this IO module:

RESI-L-S16DI8RO,S8RO-SIO-MODBUS+ASCII-ENxx.pdf

14.7 Additional ASCII commands

Here you will find only the additional ASCII commands especially for this IO module. Please refer to the description of of the standard commands for more details about the available basic ASCII commands.

Please refer to the external document for detailed documentation of the current ASCII commands for this IO module:

RESI-L-S16DI8RO,S8RO-SIO-MODBUS+ASCII-ENxx.pdf

15 RESI-4AIU-SIO, RESI-4AIU-ETH

15.1 General information

This series of IO modules offer the following features:

- 4 high precision analog inputs for -10Vdc .. + 10Vdc signals (-10.24Vdc to + 10.24Vdc)
- ADC resolution 16 bit, accuracy +/- 0.1V
- RESI-xxx-SIO: Galvanic isolated RS232 and RS485 interface for communication with a host system
- RESI-xxx-ETH: Galvanic isolated Ethernet interface for communication with a host system

Figure: Our serial IO module

Figure: Our Ethernet IO module

15.2 Technical specification

Beside the basic technical data, which fulfill all of our IO modules, this IO modules meet the following technical specifications:

Power consumption	
RESI-4AIU-SIO	<0.7W
RESI-4AIU-ETH	<1.1W

Product housing	
RESI-4AIU-SIO	CEM17
RESI-4AIU-ETH	CEM35

Product weight	
RESI-4AIU-SIO	65g
RESI-4AIU-ETH	90g

Analog inputs	
Number	4
Update speed	Every 100ms
Range	-10V .. + 10V
ADC resolution	16 bit
Input voltage range	-10.24V .. + 10.24V
Accuracy	+/- 0.1V
Cable connection	via terminals
Galvanic isolation	Yes

Default serial settings	
baud rate	via DIP switch
Parity	none
Stopbits	one
UnitID	255

Default Ethernet settings	
IP address	192.168.0.51
IP mask	255.255.255.0
gateway	192.168.0.1
UnitID	255

User	RESI
password	RESI

15.3 Additional terminals & LED states

ANALOG INPUTS	4 analog inputs for -10V..0V..+10V signals	
	Two 3 pin terminal blocks	
	Terminal type:	USLIM
	C:	Ground for all analog inputs
	AI1-AI4:	Analog inputs
Pin layout	AI1:	Signal input for analog input #1
	AI2:	Signal input for analog input #2
	AI3:	Signal input for analog input #3
	AI4:	Signal input for analog input #4
	C:	Signal ground for analog inputs #1-#4
		Both signal grounds are internally bridged
INFO	If everything is OK, this LED is on. If there is an internal error at the analog inputs,	
	this LED flashes quickly.	

15.4 RESI-4AIU-SIO: Connection diagram

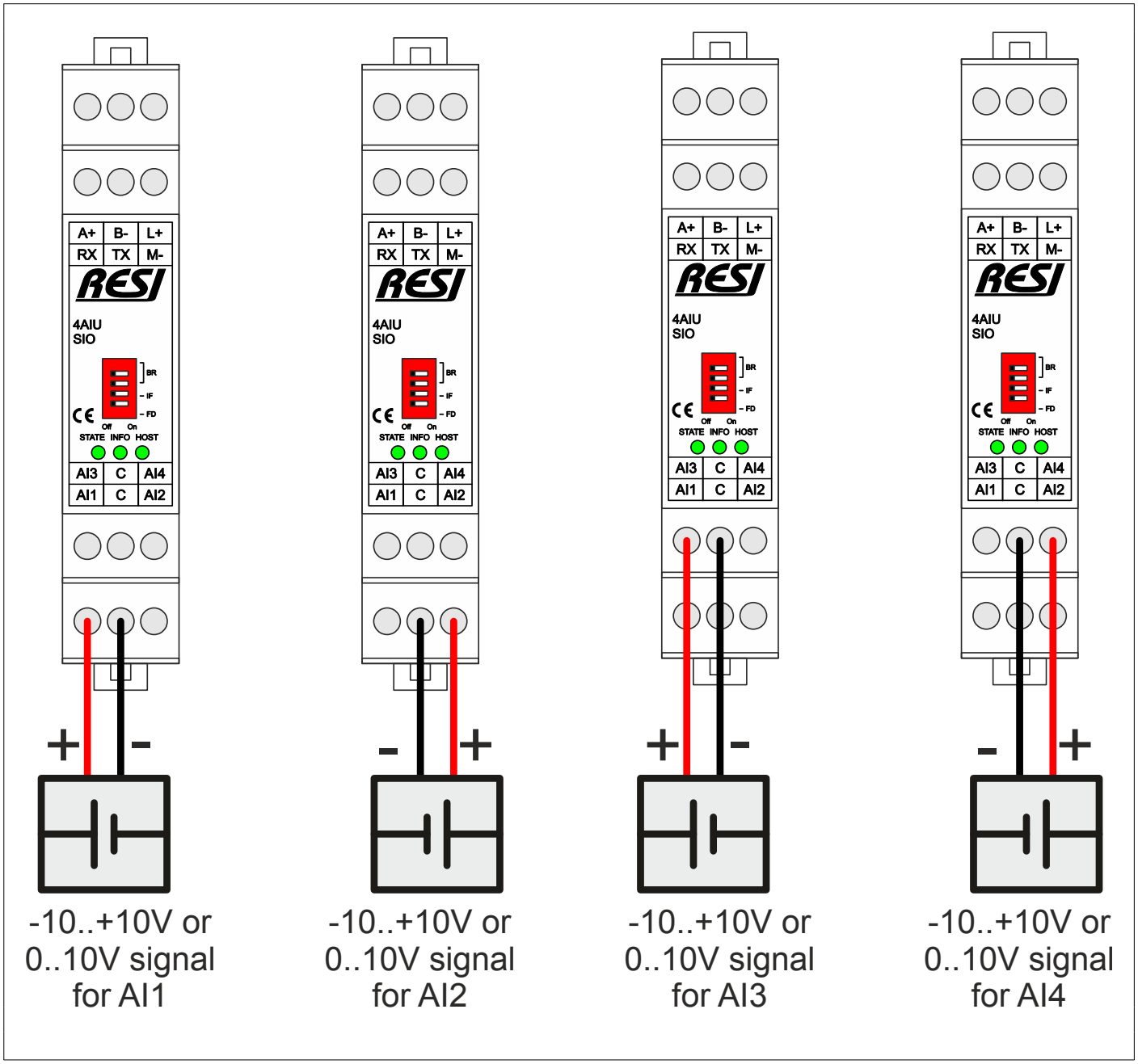


Figure: Connecting the analog inputs to the IO module

15.5 RESI-4AIU-ETH: Connection diagram

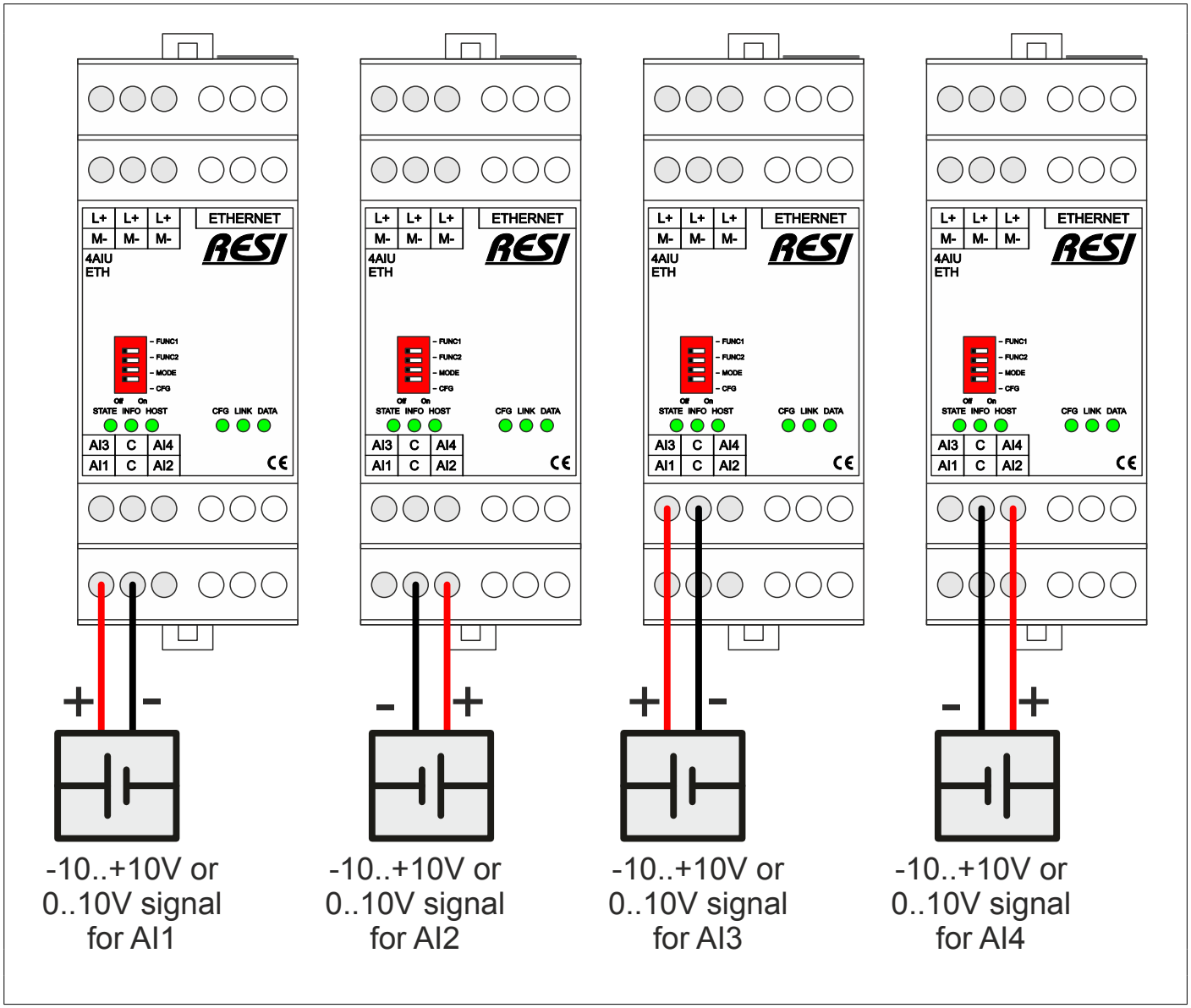


Figure: Connecting the analog inputs to the IO module

15.6 Additional MODBUS register & coils

Here you will find only the additional MODBUS registers and coils especially for this IO module. Please refer to the description of of the standard MODBUS mapping for more details about the available basic MODBUS registers and coils.

Please refer to the external document for detailed documentation of the current MODBUS register mapping for this IO module:

RESI-L-4AIU-SIO-ETH-MODBUS+ASCII-ENxx.pdf

15.7 Additional ASCII commands

Here you will find only the additional ASCII commands especially for this IO module. Please refer to the description of of the standard commands for more details about the available basic ASCII commands.

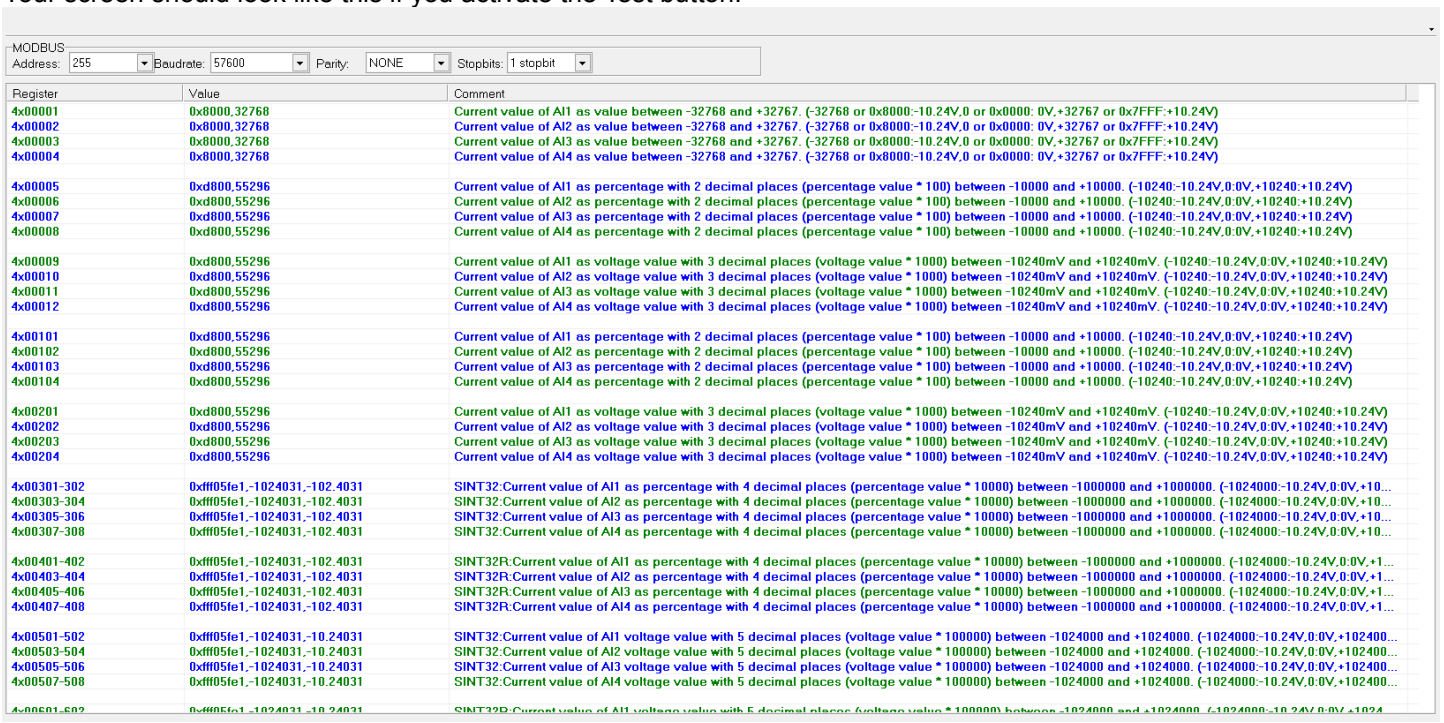
Please refer to the external document for detailed documentation of the current ASCII commands for this IO module:

RESI-L-4AIU-SIO-ETH-MODBUS+ASCII-ENxx.pdf

15.8 Additional MODBUSConverter software information

Click on the add to project button to open a dialog with all available IO modules. Then select the section SLIM-IO modules… and select RESI-4AIU-SIO or RESI-4AIU-ETH to add this device to your project. or you search the connected module automatically.

Your screen should look like this if you activate the Test button.



16 RESI-MBUSx-SIO, RESI-MBUSx-ETH

16.1 General information

This series of IO modules offer the following features:

- MBUS master interface to collect data from up to 64 smart meter with MBUS protocol
- Automatic conversion of MBUS data from MBUS data types to MODBUS register data types
- Status readout for each MBUS device
- Integrated MBUS power supply
- free PC based configuration tool for MBUS to MODBUS mapping of meter data
- RESI-xxx-SIO: Galvanic isolated RS232 and RS485 interface for communication with a host system
- RESI-xxx-ETH: Galvanic isolated Ethernet interface for communication with a host system

We offer the following different MBUS models:

- **RESI-MBUS2-SIO**: MBUS master gateway for readout of MBUS data from 2 MBUS smart meter, up to 40 MODBUS holding registers/20 mappings from MBUS to MODBUS, serial RS232 and RS485 interface
  → Former product RESI-MBUST-MODBUS
- **RESI-MBUS8-SIO**: MBUS master gateway for readout of MBUS data from 8 MBUS smart meter, up to 400 MODBUS holding registers/200 mappings from MBUS to MODBUS, serial RS232 and RS485 interface
  → Former product RESI-MBUS-MODBUS
- **RESI-MBUS24-SIO**: MBUS master gateway for readout of MBUS data from 24 MBUS smart meter, up to 1000 MODBUS holding registers/500 mappings from MBUS to MODBUS, serial RS232 and RS485 interface
  → Former product RESI-MBUS2-MODBUS
- **RESI-MBUS48-SIO**: MBUS master gateway for readout of MBUS data from 48 MBUS smart meter, up to 1200 MODBUS holding registers/600 mappings from MBUS to MODBUS, serial RS232 and RS485 interface
  → Former product RESI-MBUS3-MODBUS
- **RESI-MBUS64-SIO**: MBUS master gateway for readout of MBUS data from 64 MBUS smart meter, up to 1200 MODBUS holding registers/600 mappings from MBUS to MODBUS, serial RS232 and RS485 interface
  → New product

- **RESI-MBUS2-ETH**: MBUS master gateway for readout of MBUS data from 2 MBUS smart meter, up to 40 MODBUS holding registers/20 mappings from MBUS to MODBUS, Ethernet interface
  → Former product RESI-MBUST-ETH
- **RESI-MBUS8-ETH**: MBUS master gateway for readout of MBUS data from 8 MBUS smart meter, up to 400 MODBUS holding registers/200 mappings from MBUS to MODBUS, Ethernet interface
  → Former product RESI-MBUS-ETH
- **RESI-MBUS24-ETH**: MBUS master gateway for readout of MBUS data from 24 MBUS smart meter, up to 1000 MODBUS holding registers/500 mappings from MBUS to MODBUS, Ethernet interface
  → Former product RESI-MBUS2-ETH
- **RESI-MBUS48-ETH**: MBUS master gateway for readout of MBUS data from 48 MBUS smart meter, up to 1200 MODBUS holding registers/600 mappings from MBUS to MODBUS, Ethernet interface
  → Former product RESI-MBUS3-ETH
- **RESI-MBUS64-ETH**: MBUS master gateway for readout of MBUS data from 64 MBUS smart meter, up to 1200 MODBUS holding registers/600 mappings from MBUS to MODBUS, Ethernet interface
  → New product

The amount of meters are defined by the standard unit load of 1.5mA per meter. Please note that many meters need more current from the MBUS power supply, so the total number of meters may not recht the maximum of the used module.

Figure: Our serial IO module

Figure: Our Ethernet IO module

16.2 Technical specification

Beside the basic technical data, which fulfil all of our IO modules, this IO modules meet the following technical specifications:

Power consumption	
RESI-MBUSx-SIO	<0.6W no connected MBUS meter
	<2.0W shortcut on MBUS line
	<8.0W MBUS line overload for short time
RESI-MBUSx-ETH	<1.0W no connected MBUS meter
	<2.4W shortcut on MBUS line
	<8.4W MBUS line overload for short time

Product housing	
RESI-MBUSx-SIO	CEM17
RESI-MBUSx-ETH	CEM35

Product weight	
RESI-MBUSx-SIO	55g
RESI-MBUSx-ETH	90g

MBUS power supply	
Nominal output voltage	~34,2V
Maximum output current	~174mA shortcut on MBUS line
	~155mA MBUS line overload for short time

MBUS cabling	
Nominal cable for MBUS bus	JYStY 2x0.8mm² or
	JYStY 0x1.5mm²
Nominal cable resistance	75 Ohm/km
Nominal cable capacity	50nF/km
Maximum cable length	max. 7000m
Maximum cable capacity:	max. 180nF

HINT: The real cable length is determined how many MBUS meters you will connect to the segment and how the segment is designed (star, tree, line) and how fast you will communicate over the bus line. Please refer to the internet for more details how to build a correct MBUS meter network!

Default serial settings	
baud rate	via DIP switch
Parity	none
Stopbits	one
UnitID	255

Default Ethernet settings	
IP address	
RESI-MBUS2-ETH	192.168.0.210
RESI-MBUS8-ETH	192.168.0.211
RESI-MBUS24-ETH	192.168.0.212
RESI-MBUS48-ETH	192.168.0.213
RESI-MBUS64-ETH	192.168.0.214
IP mask	255.255.255.0
gateway	192.168.0.1
UnitID	255

User	RESI
password	RESI

16.3 Additional terminals & LED states

MBUS system	MBUS master for connection of 2/8/24/48/64 smart meters with MBUS interface	
	One 3 pin terminal blocks	
	Terminal type:	USLIM
	MB+:	Positive signal of MBUS bus system
	MB-:	Negative signal of MBUS bus system
	HINT: Swapping the two wires of the bus is also permitted and	
	does not generate any errors	
Pin layout	MB+:	Positive signal of MBUS bus system
	N/C:	not connected
	MB-:	Negative signal of MBUS bus system
STATE	If no configuration is downloaded into the module, this LED blinks very quickly (~100ms)	
	If the configuration or the module has an error this LED blinks very fast (~50ms)	
	If everything is ok this LED blinks very slow (~1s)	
MBUS	If any data is send or received by the MBUS interface, this LED flashes	

16.4 RESI-MBUSx-SIO: Connection diagram

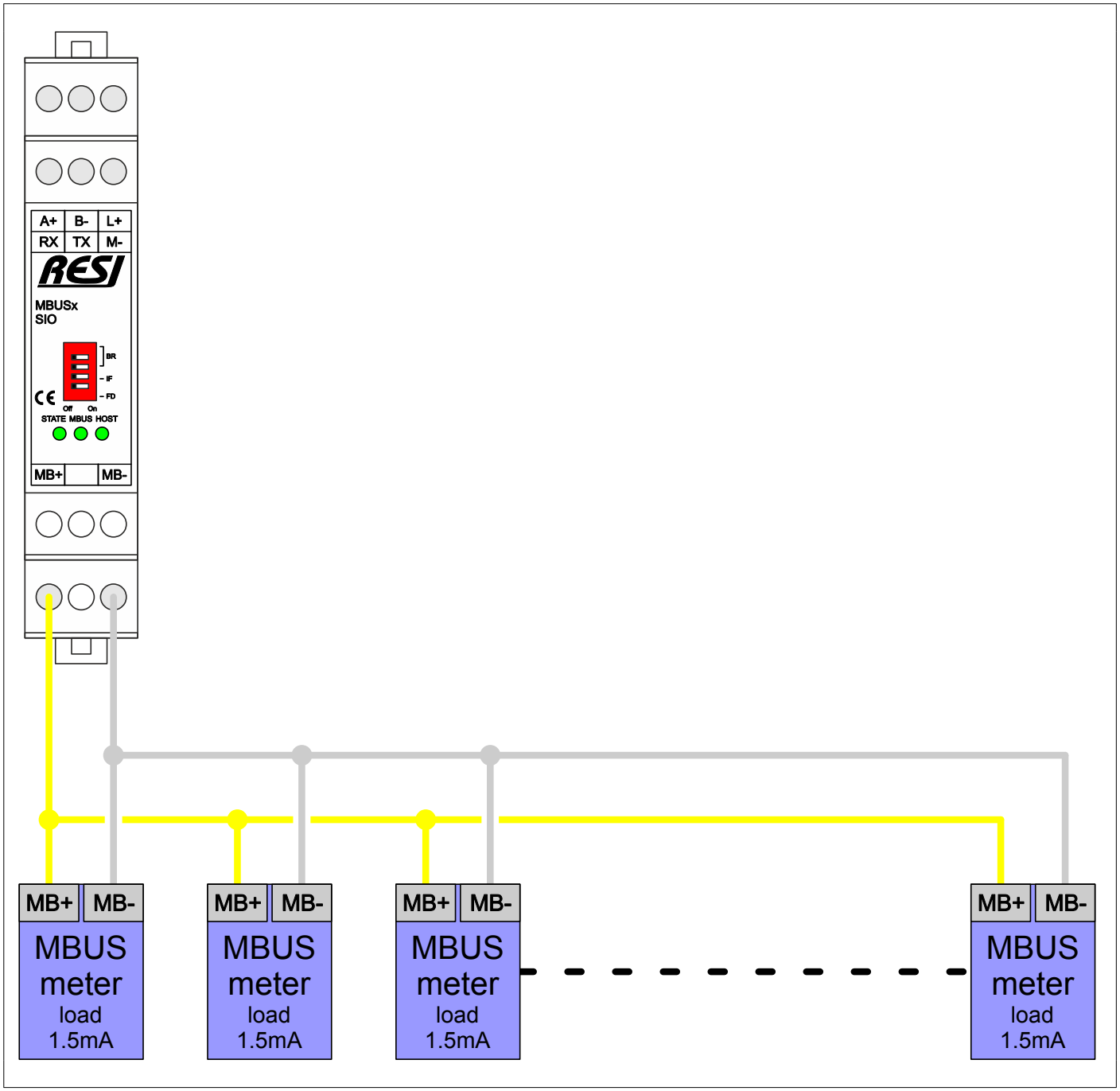


Figure: Connecting the MBUS bus system to the serial MBUSx converter

16.5 RESI-MBUSx-ETH: Connection diagram

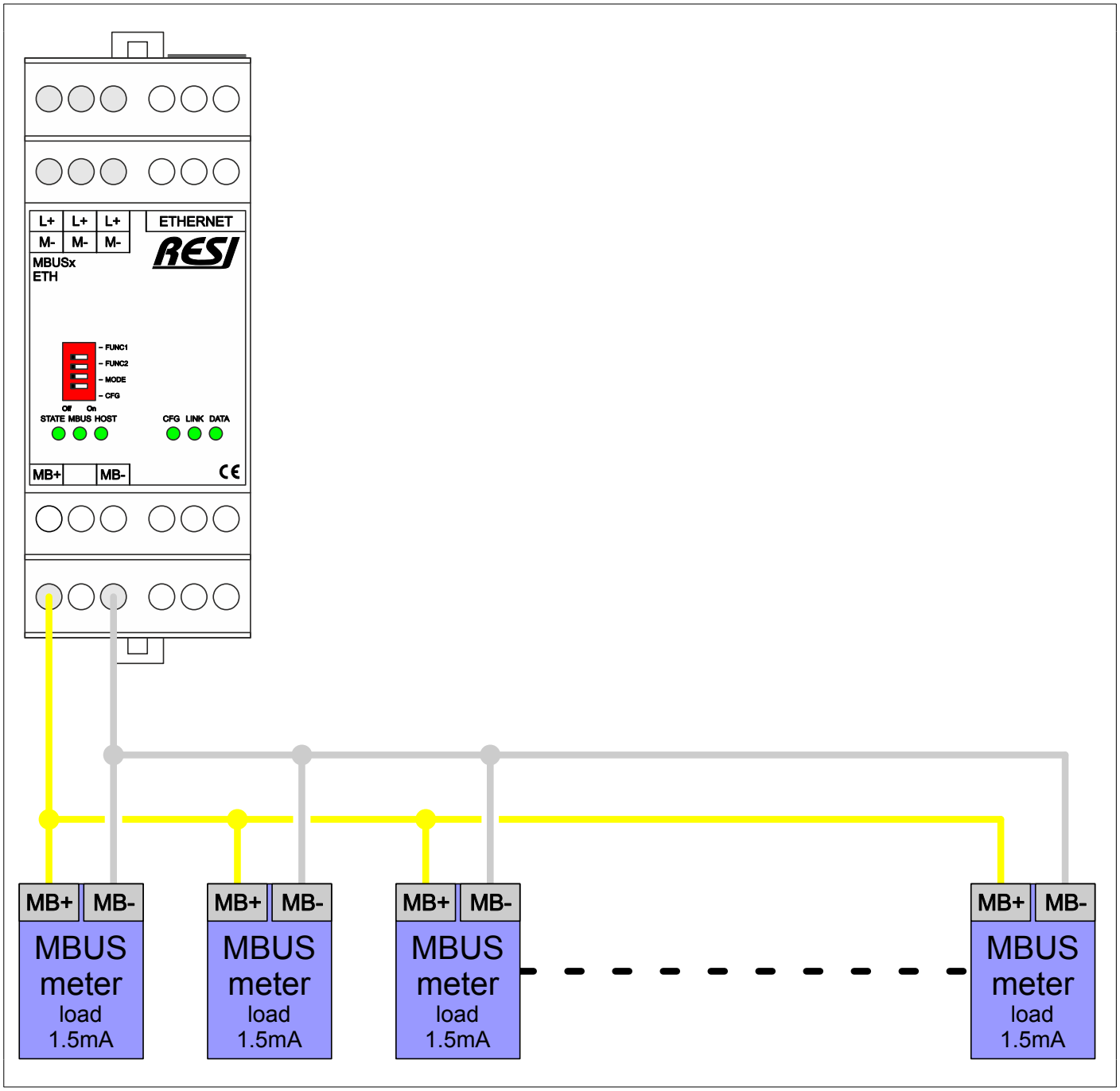


Figure: Connecting the MBUS bus system to the Ethernet MBUSx converter

16.6 MBUS bus topology

The MBUS bus topology is free. You can use star, line or tree bus topology. Only a ring topology is forbidden! The MBUS cable is a two wire cable, which connects the MBUS master (our gateway) with every connected MBUS slave (the meter). The M-Bus is polarity independent and needs no line termination resistors at the end of the cables. Any cable type may be used as long as the cable is suitable for >36V/500mA. Shielding is not necessary and not recommended since the capacity of the cable should be minimized.

In most cases a standard telephone cable is used which is a twisted pair wire with a diameter of 0.8mm each (2x0.8mm). This type of cable should be used for the main wiring. For the wiring to the meters from the main wiring (last one or two meters to the meter) a cable with smaller diameter may be used.

The maximum distance between a slave and the master is around 3km to 10km, depending on the individual network configuration. This distance applies for the standard configuration having Baud rates between 300 and 2400 Baud, and a maximum of 64 slaves. The maximum distance can be increased by limiting the Baud rate and using fewer slaves, but the bus voltage at no point in a segment fall below 24V, because of the remote powering of the slaves. In the standard configuration the total cable length should not exceed 3000 m, in order to meet the requirement of a maximum cable capacitance of 180nF.

Please refer to the internet for more details about the MBUS bus cabling and the theoretical and practical cable length.

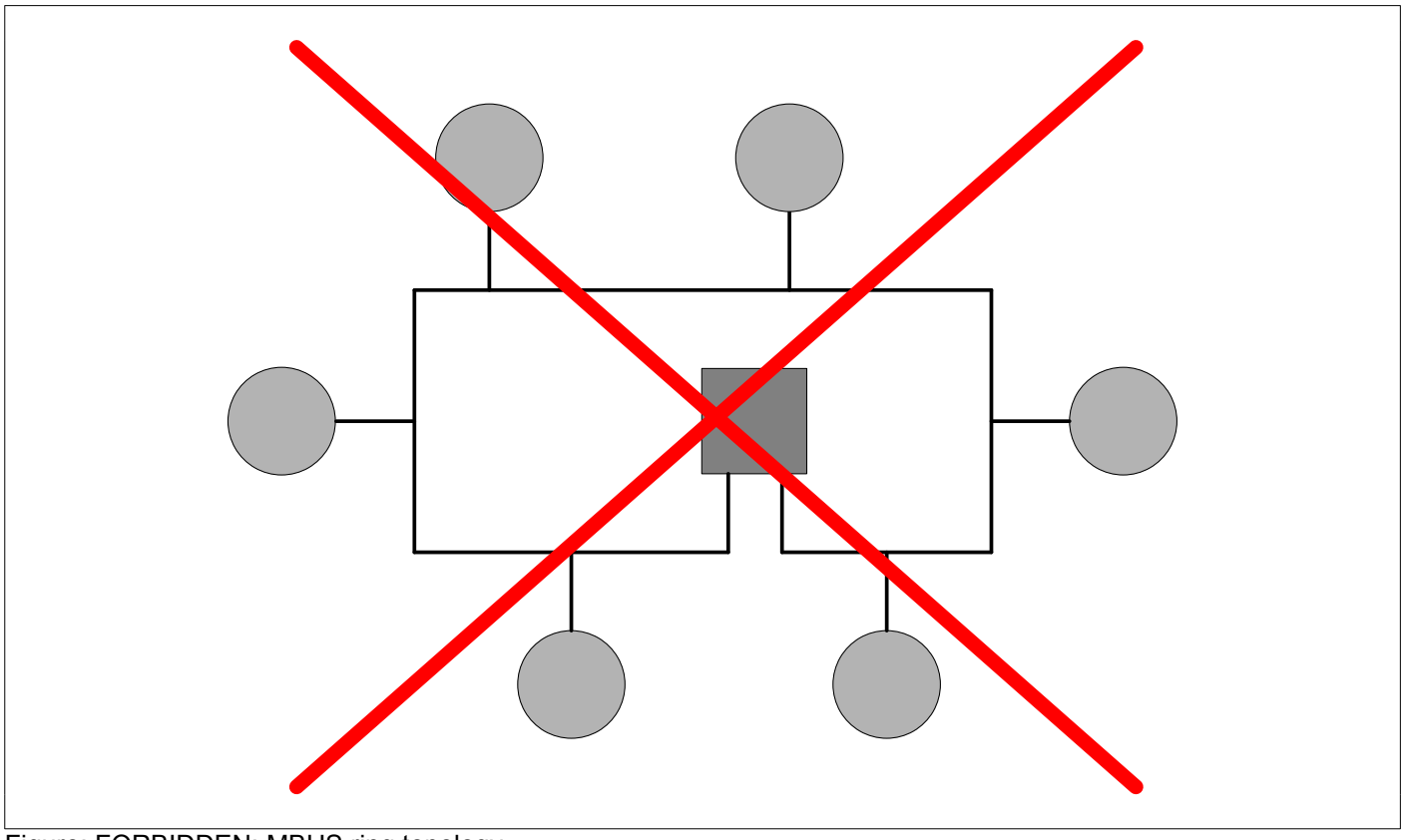


Figure: FORBIDDEN: MBUS ring topology

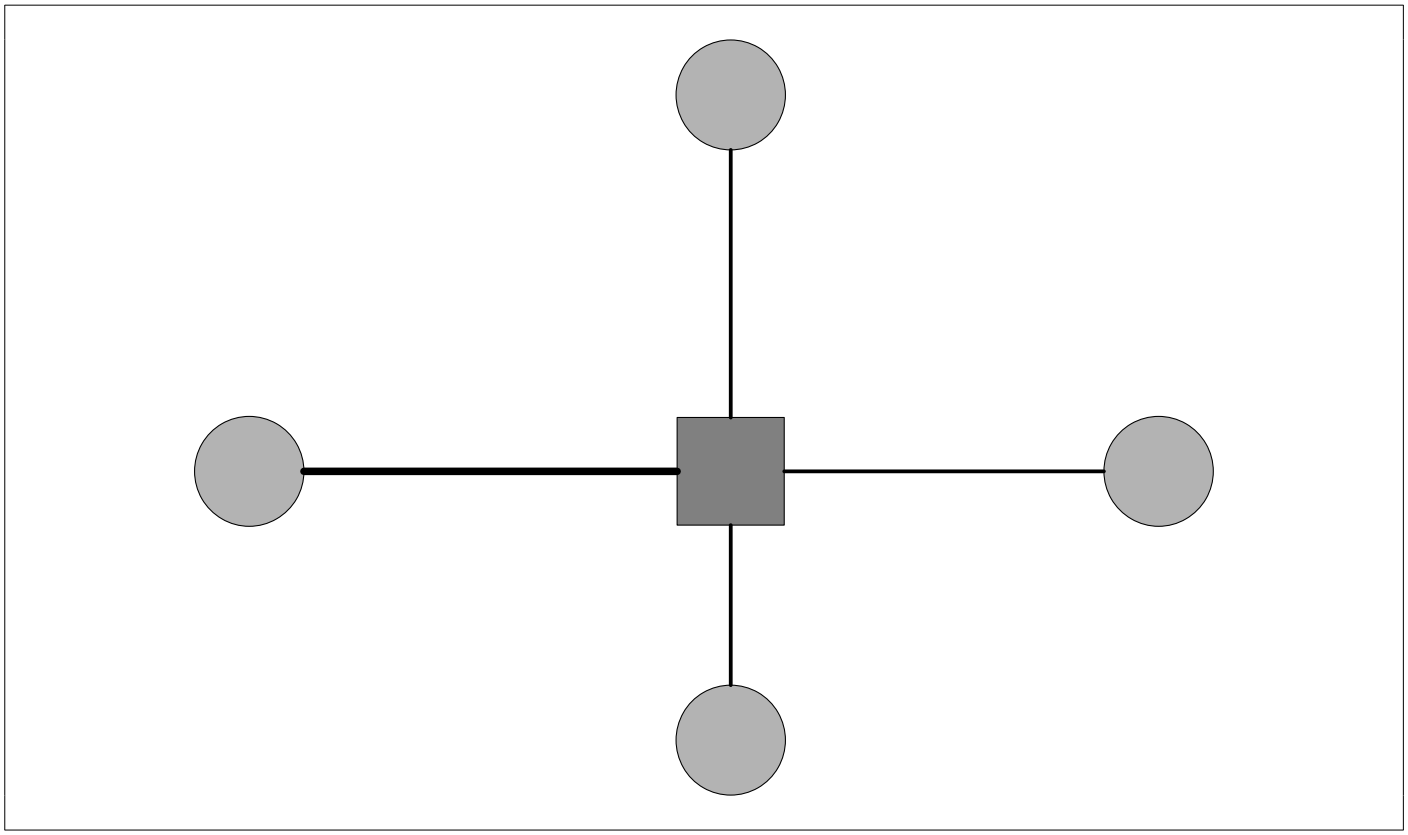


Figure: MBUS star topology

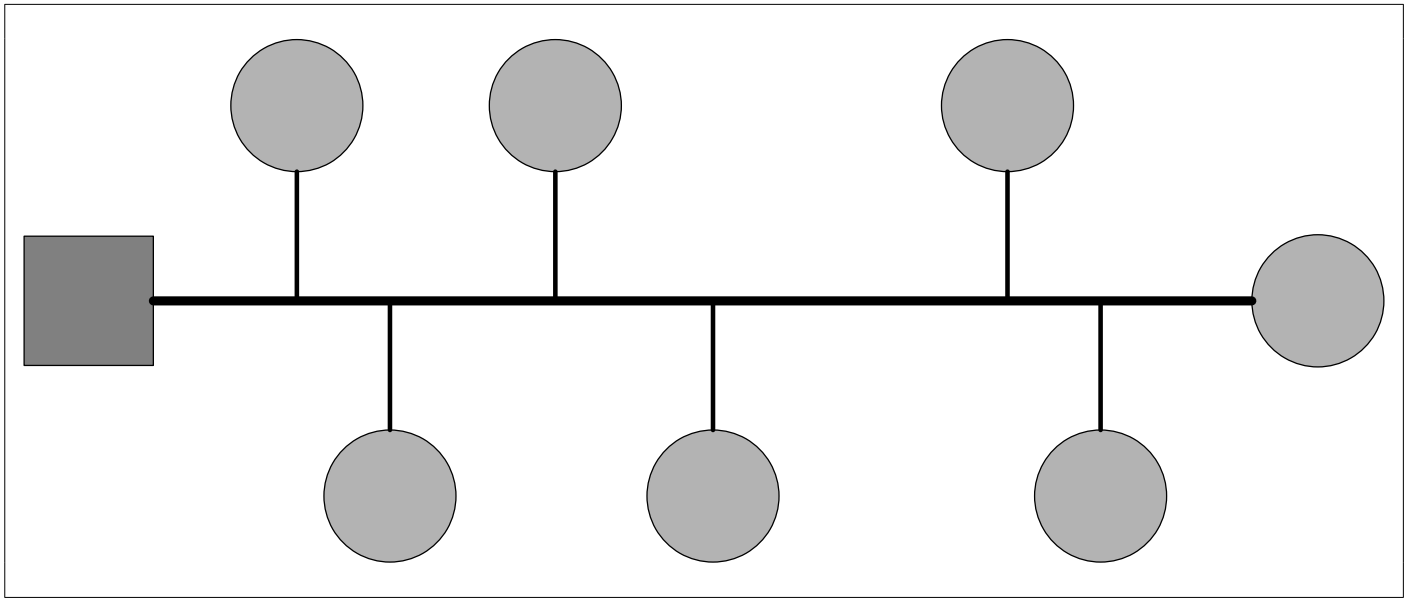


Figure: MBUS line topology

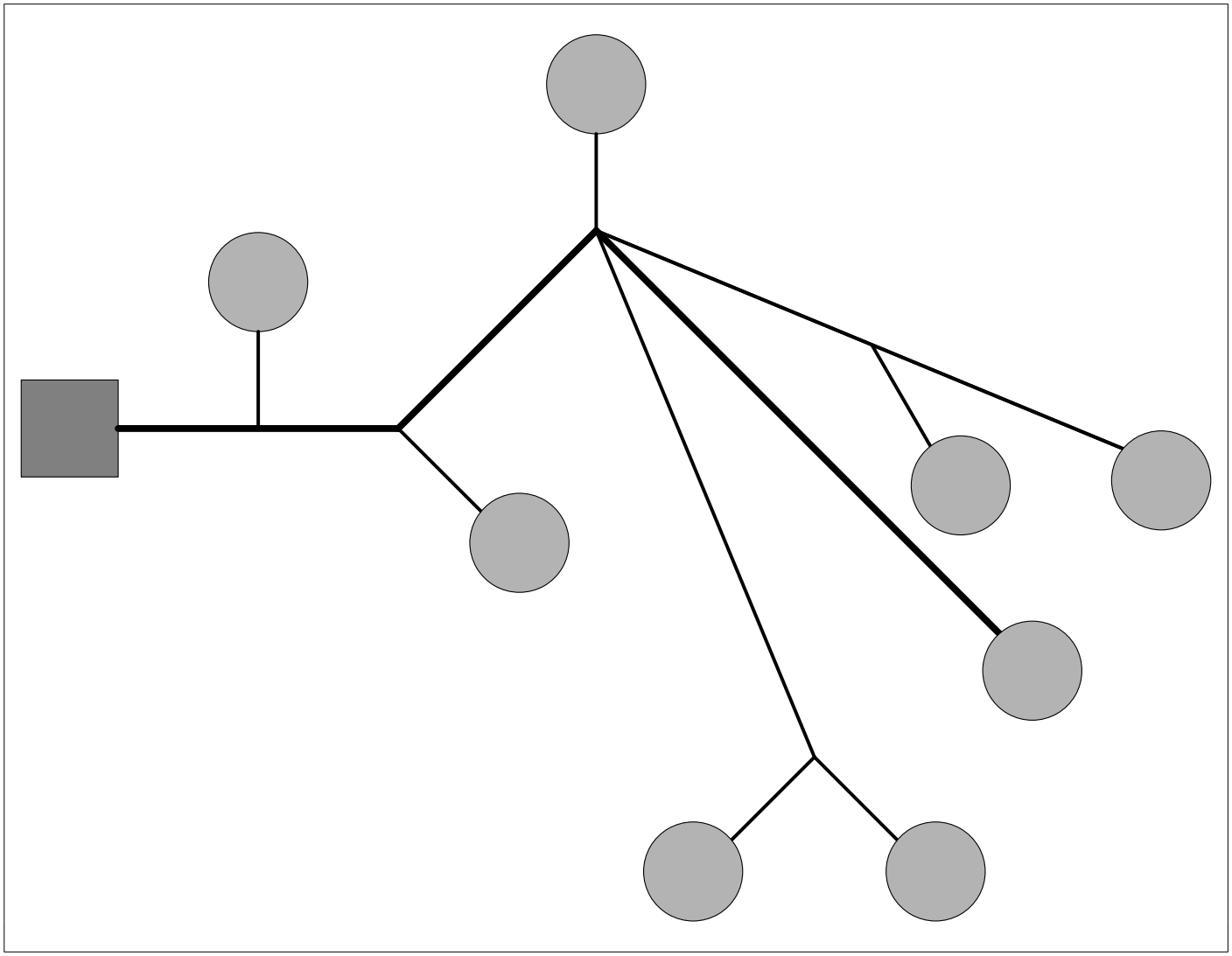


Figure: MBUS tree topology

16.7 MBUS bus recomendations

This are some recommendations for MBUS bus lines from literature out of the internet.
Don't forget: This is only a helpful hint. RESI or partners of RESI do not guarantee, that your bus system works in any case, if you follow this hints! You are responsible to plan and design your individual MBUS bus system correctly.

The resistive cable length defines the maximum length of a cable segment with in the bus structure. The cable segment length is the distance from the M-Bus Master to the M-Bus device furthest away. The capacitive cable length defines the maximum bus cable length in total.

16.7.1 Small inhouse installations

Description: small and medium-sizes residential buildings
- resistive cable length: max. 350m
- capacitive cable length: max. 1km
- cable cross section: min. 0.5mm²

Usage:
- max. 64 devices with max 9600 baud

16.7.2 large inhouse installations

Description: medium-sizes and large residential buildings
- resistive cable length: max. 350m
- capacitive cable length: max. 3km
- cable cross section: min. 0.5mm²

Usage:
- max. 64 devices with max 2400 baud

16.7.3 Small wide area installation

Description: small to medium-sized residential areas
- resistive cable length: max. 1km
- capacitive cable length: max. 4km
- cable cross section: min. 0.5mm²

Usage:
- max. 64 devices with max 2400 baud

16.7.4 Big wide area installation

Description: medium-sized to large residential areas
- resistive cable length: max. 3km
- capacitive cable length: max. 5km
- cable cross section: min. 1.5mm²

Usage:
- max. 64 devices with max 2400 baud

16.7.5 Provider network installation

Description: energy provider driven networks

- resistive cable length: max. 5km
- capacitive cable length: max. 7km
- cable cross section: min. 1.5mm²

Usage:

- max. 16 devices with max 300 baud

16.7.6 Maximum segment installation

Description:

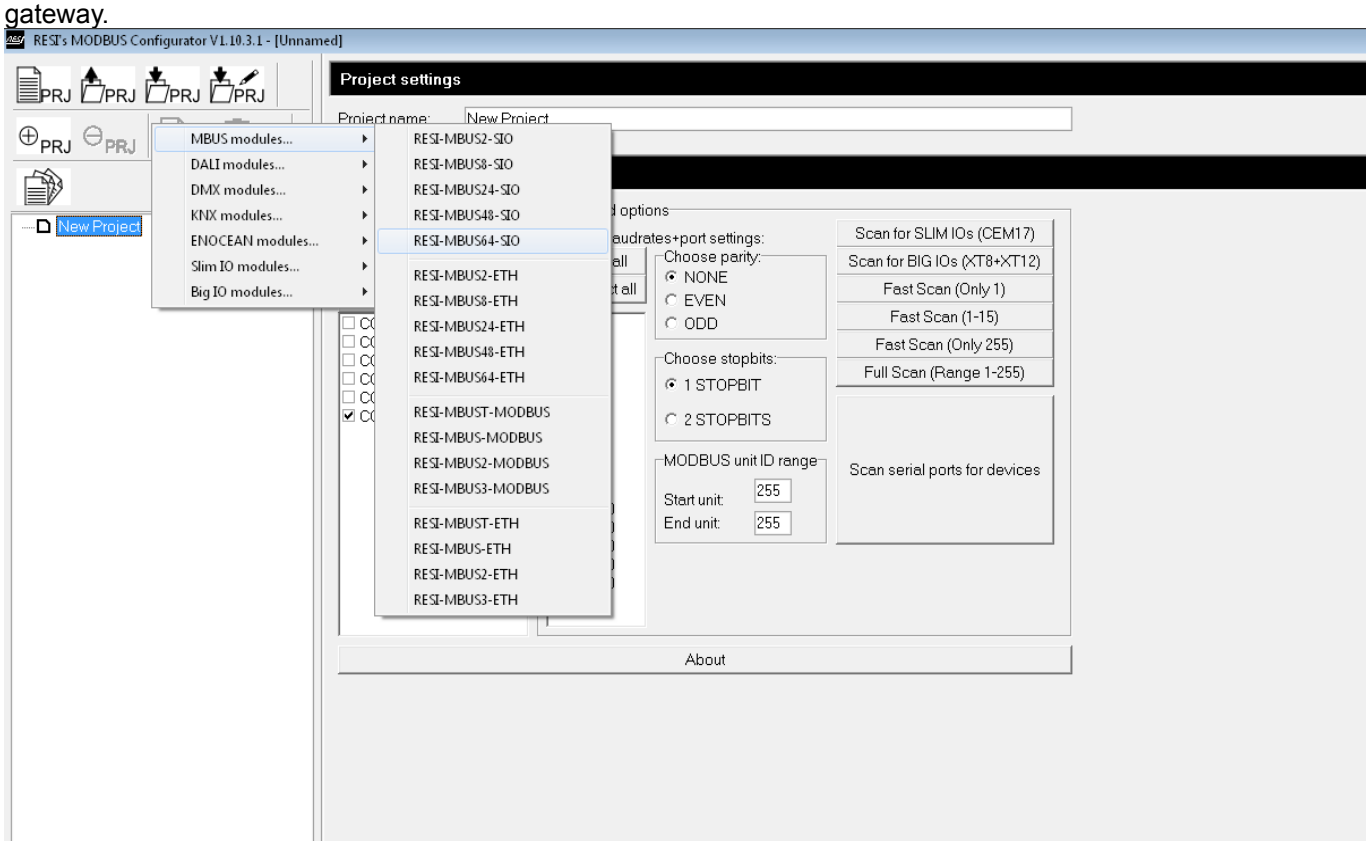
- linear topology
- cable length: max. 10km
- cable cross section: min. 1.5mm²

Usage:

- max. 1 device with max 300 baud

16.8 Add RESI-MBUSx-xxx device to project tree

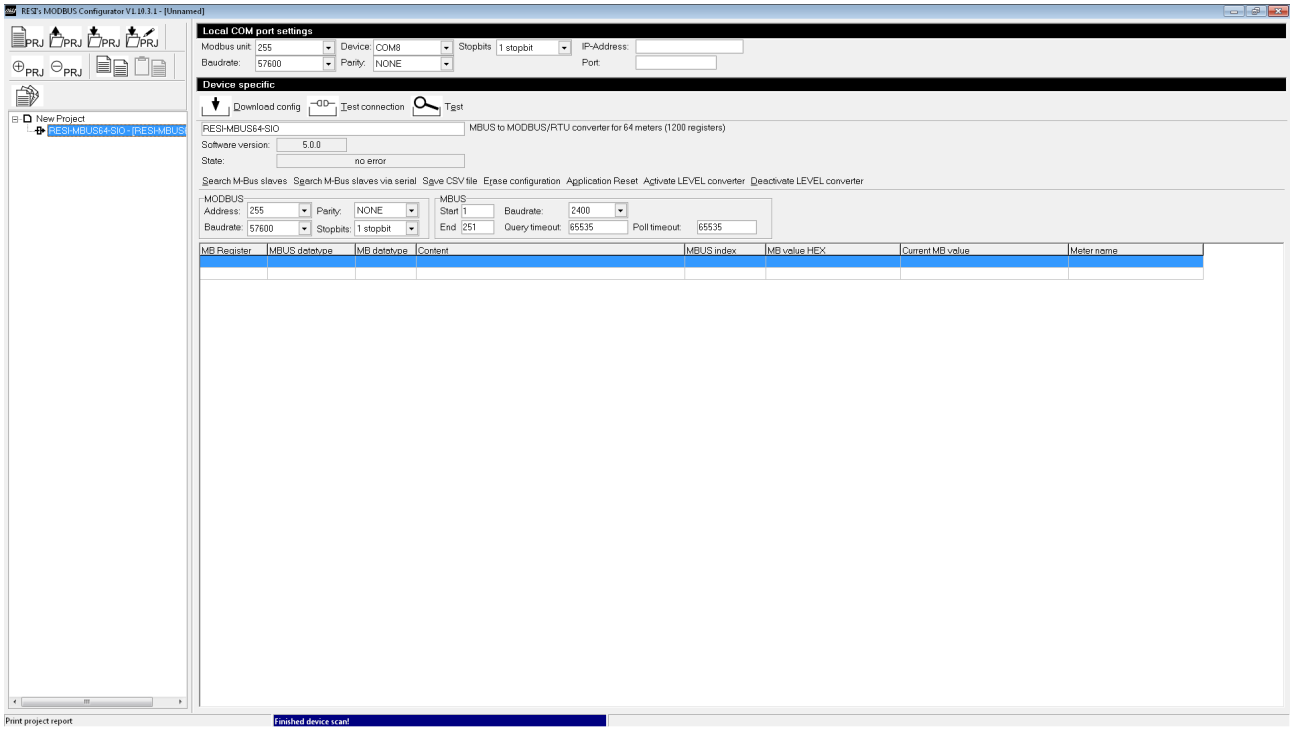
First, start the MODBUSConfigurator software. Click on the project tree title “New Project” and add a desired Ethernet gateway.



You will notice, that you find the new MBUS converters on top of the list. But for compatibility reasons, you can also add and use older modules or older projects with the old MBUS modules form us.

After you have selected your device, don't forget to set the serial or Ethernet parameters correctly for communication with your module. Or you use the automatic search function of our software.

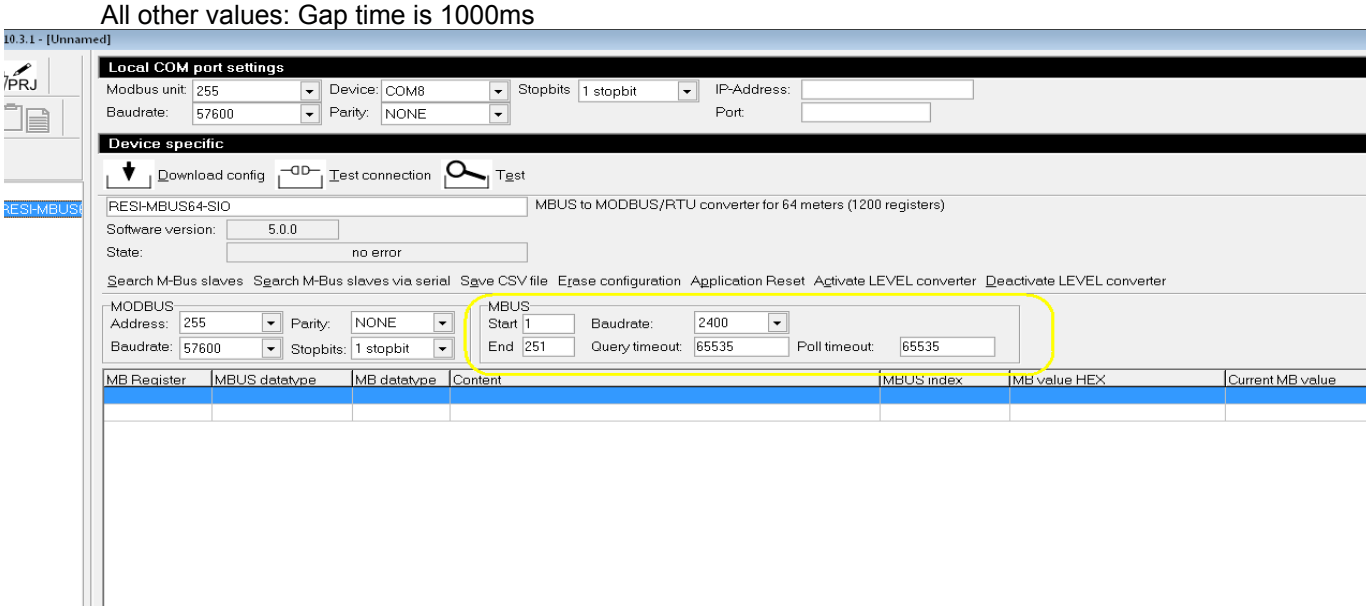
Your screen should look like this:



16.9 HOWTO setup MBUS communication parameters

First of all, you have to select the correct MBUS communication speed suitable for your meters. In the area MBUS you will find the following setup parameters:

- **Baudrate**: This is the current used MBUS baud rate on the MBUS. Usually you will see the currently configured baud rate of your converter.
- **Start, End**: This two fields define the primary address range, which will be used for an automatic search for connected MBUS slaves via primary addressing mode. You can enter a valid MBUS primary address in the range from 1 to 251. If you have connected only one meter, you can also use the primary broadcast address 254 for communication with this meter.
- **Query timeout**: This field defines the timeout between two query cycles in the gateway. Usually the gateway communicates with all configured meters sequentially. After finishing the data readout for the last meter, the gateway pauses for this defined interval in seconds. This values are used:
  Value 65535 or values 0..5 defines ~5s pause.
  Values 6 to 65534: defines 6 to 65534 seconds of pause, before the next polling cycle will start.
- **Poll timeout**: This field defines a general pause after the readout of a configured meter before the readout of the next meter starts. In the past we discovered that there are many meters out in the market, which need a special treatment in the timing. e.g. very old KAMSTRUP meters need often two readout cycles with a gap of at least 10-15 seconds. This is non standard to the MBUS. Or other meters have problems with secondary addressing, if there is a too small gap between the readout. So we introduced this new parameter: This timeout defines the pause after finishing reading of a meter and starting reading the next meter. In the previous firmware versions this timeout was fixed to 250ms gap, which was ok for 99% of the meter readout on the markets. But some meter fail to process this little gap. The values is interpreted as follows:
  Value 1..30: Gap time 1 seconds to 30 seconds
  Value 101..400: Gaptime=(Value-100)*0.1s → 0.1s .. 30s e.g. 105 → 0.5s
  Value 65535: Gap time is 1 second
  Value 65534: Gap time is 250ms
  Value 65533: Gap time is 500ms
  Value 65532: Gap time is7250ms
  All other values: Gap time is 1000ms



Change the MBUS baudrate and/or the poll & query timeouts
Follow this steps to change to communication baud rate of the MBUS bus system:

1. Select a new baud rate from the drop down list Baudrate in the MBUS area.
2. Change the query timeout to your needs
3. Change the poll timeout to your needs
4. Use the Download config button to download the new settings into your device
5. Now your device will use the new settings of the baud rate and the timeouts on the MBUS side

You can achieve the same with writing the new MBUS baud rate and the timeouts to certain MODBUS registers. Please refer to the section of the MODBUS register description, how this function will work.

Here you will find a basic diagram, how the MBUS master request cycle is handled by our gateways. The two parameters can be configured like this:

- **Query timeout**: This field defines the timeout between two query cycles in the gateway. Usually the gateway communicates with all configured meters sequentially. After finishing the data readout for the last meter, the gateway pauses for this defined interval in seconds. This values are used:
  Value 65535 or values 0..5 defines ~5s pause.
  Values 6 to 65534: defines 6 to 65534 seconds of pause, before the next polling cycle will start.
- **Poll timeout**: This field defines a general pause after the readout of a configured meter before the readout of the next meter starts. In the past we discovered that there are many meters out in the market, which need a special treatment in the timing. e.g. very old KAMSTRUP meters need often two readout cycles with a gap of at least 10-15 seconds. This is non standard to the MBUS. Or other meters have problems with secondary addressing, if there is a too small gap between the readout. So we introduced this new parameter: This timeout defines the pause after finishing reading of a meter and starting reading the next meter. In the previous firmware versions this timeout was fixed to 250ms gap, which was ok for 99% of the meter readout on the markets. But some meter fail to process this little gap. The values is interpreted as follows:
  Value 1..30: Gap time 1 seconds to 30 seconds
  Value 101..400: Gaptime=(Value-100)*0.1s → 0.1s .. 30s e.g. 105 → 0.5s
  Value 65535: Gap time is 1 second
  Value 65534: Gap time is 250ms
  Value 65533: Gap time is 500ms
  Value 65532: Gap time is7250ms
  All other values: Gap time is 1000ms

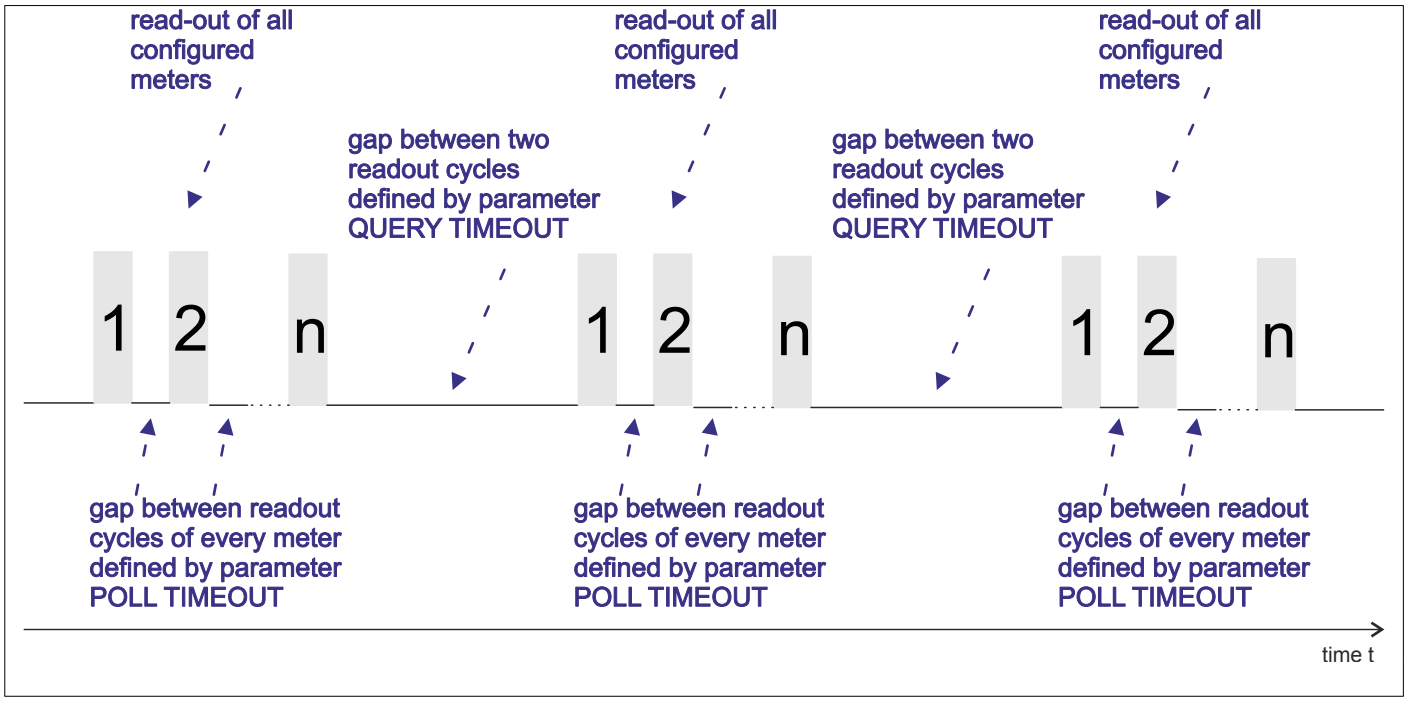


Figure: Basic timing of MBUS master read-out for MBUS slaves

Now we go more into detail, how the MBUS gateway will handle the request process of one meter. Forst we define the parameters:

- **Poll repeats 1**: This field defines the amount of telegram repetitions for the addressing command to a meter, before the gateway declares the communication as not possible and resumes with the next meter.
  Value 65535 or 0: use 3 repeats as standard
  Value 1..n: Use n repeats
- **Poll repeats 2**: This field defines the amount of telegram repetitions for the data readout command to a meter, before the gateway declares the communication as not possible and resumes with the next meter.
  Value 65535 or 0: use 5 repeats as standard
  Value 1..n: Use n repeats
- **Poll pre delay 1**: This field defines the first pause time in Milliseconds before starting to send the first addressing command telegram to a meter.
  Value 65535: use 250ms as standard pause time
  Value 0..65534: Use x ms as pause time

- **Poll pre delay 2**: This field defines the first pause time in Milliseconds before starting to send the first data request telegram to a meter.
  Value 65535: use 100ms as standard pause time
  Value 0..65534: Use x ms as pause time
- **Poll post delay 1**: This field defines a pause time in Milliseconds. If the gateway do not receive a correct answer to an addressing command telegram and the addressing command is repeated, then this pause time is inserted, before resending the addressing telegram to the meter.
  Value 65535: use 0ms as standard pause time
  Value 0..65534: Use x ms as pause time
- **Poll post delay 2**: This field defines a pause time in Milliseconds. If the gateway do not receive a correct answer to a readout data telegram and the readout data command is repeated, then this pause time is inserted, before resending the readout data telegram to the meter.
  Value 65535: use 100ms as standard pause time
  Value 0..65534: Use x ms as pause time

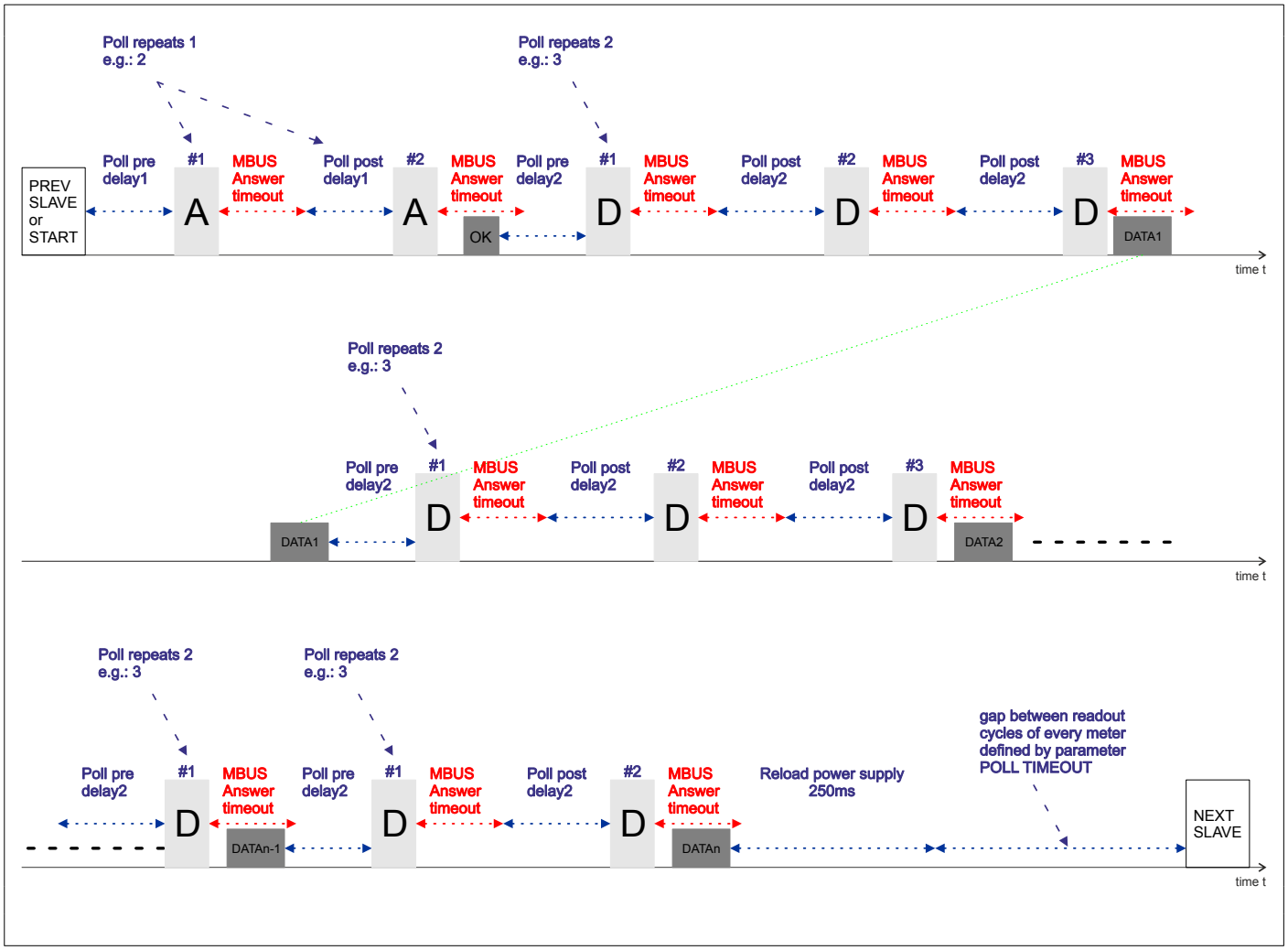


Figure: Basic timing of MBUS master read-out for MBUS slaves

16.10 HOWTO find connected MBUS meters

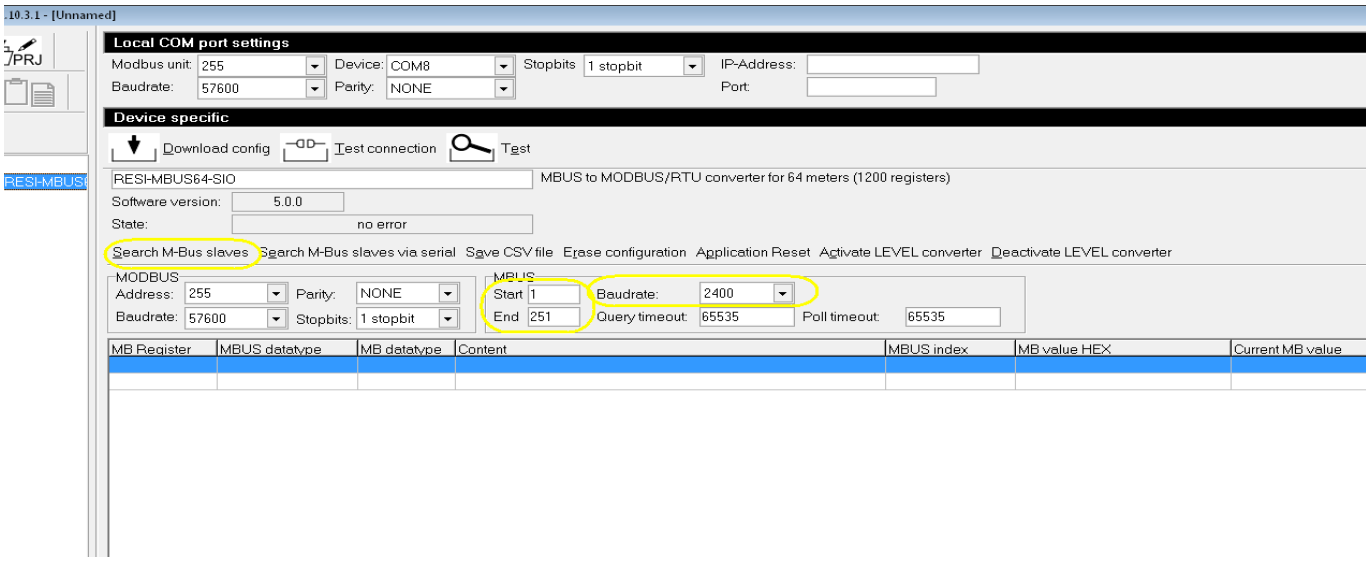
There are two ways for searching for connected MBUS meters.

- **Search M-Bus slaves with primary address**: With this function the MBUS network is scanned for new meters only by addressing the meters with the primary address. The address range is defined with the parameters Start and End in the MBUS area. Every found meter, which is not part of the configuration, will be added automatically to the project.
- **Search M-Bus slaves with secondary address**: With this function the MBUS network is scanned for new meters using secondary addressing mode with the unique serial number of the meters. Every found meter, which is not part of the configuration, will be added automatically to the project.

16.10.1 Search for new meters – primary addressing mode

HINT: Don't forget to setup the MBUS bus baud rate for your search before. If you have to change it, select a new one from the drop down list and don't forget to download the bus speed into your gateway!

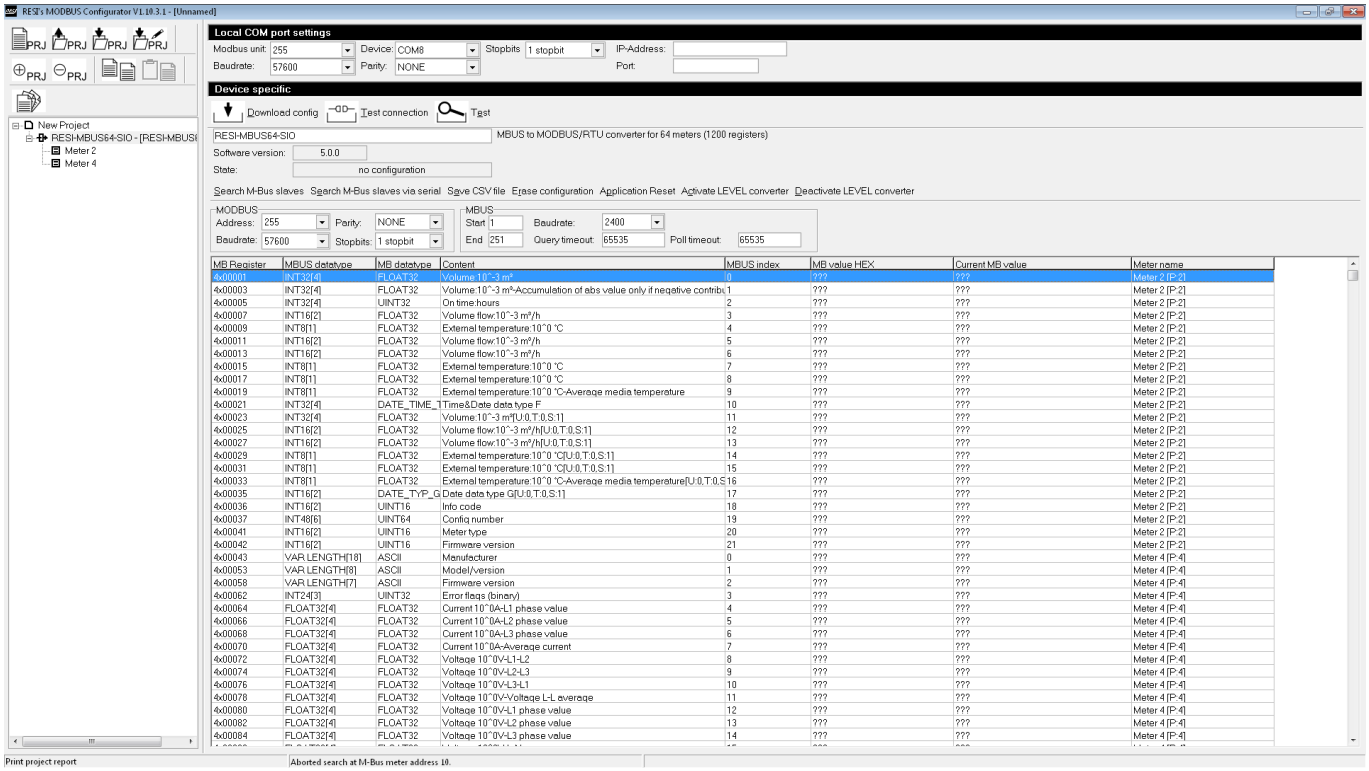
First setup the address range for your search with defining Start end End parameter in the area MBUS. In our example we use the full range 1 to 251:



Click then on the button Search M-Bus slaves to start the automatic search. Be aware, that you will not find a connected meter if it has a different baud rate configured or if it has no primary address programmed or if there are two meters with the same primary address on the bus!

HINT: You can interrupt the automatic search process by pressing the ESC button. After a few seconds the search will be interrupted.

In our test case we have connected two meters to our test system with the primary address 2 and 4. The result will look like this:



You notice, that now two meters are sown in the project tree. One with the number 2 and one with the number 4. Also the software has build an automatic mapping table between the MBUS data points and the MODBUS registers of the meter. This table is shown below the current settings. lets take a closer look into this table:

MB Register	MBUS datatype	MB datatype	Content	MBUS index	MB value HEX	Current MB value	Meter name
4x00001	INT32[4]	FLOAT32	Volume:10^-3 m³	0	???	???	Meter 2 [P:2]
4x00003	INT32[4]	FLOAT32	Volume:10^-3 m³-Accumulation of abs value only if negative contribu	1	???	???	Meter 2 [P:2]
4x00005	INT32[4]	UINT32	On time:hours	2	???	???	Meter 2 [P:2]
4x00007	INT16[2]	FLOAT32	Volume flow:10^-3 m³/h	3	???	???	Meter 2 [P:2]
4x00009	INT8[1]	FLOAT32	External temperature:10^0 °C	4	???	???	Meter 2 [P:2]
4x00011	INT16[2]	FLOAT32	Volume flow:10^-3 m³/h	5	???	???	Meter 2 [P:2]
4x00013	INT16[2]	FLOAT32	Volume flow:10^-3 m³/h	6	???	???	Meter 2 [P:2]
4x00015	INT8[1]	FLOAT32	External temperature:10^0 °C	7	???	???	Meter 2 [P:2]
4x00017	INT8[1]	FLOAT32	External temperature:10^0 °C	8	???	???	Meter 2 [P:2]
4x00019	INT8[1]	FLOAT32	External temperature:10^0 °C-Average media temperature	9	???	???	Meter 2 [P:2]
4x00021	INT32[4]	DATE_TIME_T	Time&Date data type F	10	???	???	Meter 2 [P:2]
4x00023	INT32[4]	FLOAT32	Volume:10^-3 m³[U:0,T:0,S:1]	11	???	???	Meter 2 [P:2]
4x00025	INT16[2]	FLOAT32	Volume flow:10^-3 m³/h[U:0,T:0,S:1]	12	???	???	Meter 2 [P:2]
4x00027	INT16[2]	FLOAT32	Volume flow:10^-3 m³/h[U:0,T:0,S:1]	13	???	???	Meter 2 [P:2]
4x00029	INT8[1]	FLOAT32	External temperature:10^0 °C[U:0,T:0,S:1]	14	???	???	Meter 2 [P:2]
4x00031	INT8[1]	FLOAT32	External temperature:10^0 °C[U:0,T:0,S:1]	15	???	???	Meter 2 [P:2]
4x00033	INT8[1]	FLOAT32	External temperature:10^0 °C-Average media temperature[U:0,T:0,S	16	???	???	Meter 2 [P:2]
4x00035	INT16[2]	DATE_TYP_G	Date data type G[U:0,T:0,S:1]	17	???	???	Meter 2 [P:2]
4x00036	INT16[2]	UINT16	Info code	18	???	???	Meter 2 [P:2]
4x00037	INT48[6]	UINT64	Config number	19	???	???	Meter 2 [P:2]
4x00041	INT16[2]	UINT16	Meter type	20	???	???	Meter 2 [P:2]
4x00042	INT16[2]	UINT16	Firmware version	21	???	???	Meter 2 [P:2]
4x00043	VAR LENGTH[18]	ASCII	Manufacturer	0	???	???	Meter 4 [P:4]
4x00053	VAR LENGTH[8]	ASCII	Model/version	1	???	???	Meter 4 [P:4]
4x00058	VAR LENGTH[7]	ASCII	Firmware version	2	???	???	Meter 4 [P:4]
4x00062	INT24[3]	UINT32	Error flags (binary)	3	???	???	Meter 4 [P:4]
4x00064	FLOAT32[4]	FLOAT32	Current 10^0A-L1 phase value	4	???	???	Meter 4 [P:4]
4x00066	FLOAT32[4]	FLOAT32	Current 10^0A-L2 phase value	5	???	???	Meter 4 [P:4]
4x00068	FLOAT32[4]	FLOAT32	Current 10^0A-L3 phase value	6	???	???	Meter 4 [P:4]
4x00070	FLOAT32[4]	FLOAT32	Current 10^0A-Average current	7	???	???	Meter 4 [P:4]
4x00072	FLOAT32[4]	FLOAT32	Voltage 10^0V-L1-L2	8	???	???	Meter 4 [P:4]
4x00074	FLOAT32[4]	FLOAT32	Voltage 10^0V-L2-L3	9	???	???	Meter 4 [P:4]
4x00076	FLOAT32[4]	FLOAT32	Voltage 10^0V-L3-L1	10	???	???	Meter 4 [P:4]
4x00078	FLOAT32[4]	FLOAT32	Voltage 10^0V-Voltage L-L average	11	???	???	Meter 4 [P:4]
4x00080	FLOAT32[4]	FLOAT32	Voltage 10^0V-L1 phase value	12	???	???	Meter 4 [P:4]
4x00082	FLOAT32[4]	FLOAT32	Voltage 10^0V-L2 phase value	13	???	???	Meter 4 [P:4]
4x00084	FLOAT32[4]	FLOAT32	Voltage 10^0V-L3 phase value	14	???	???	Meter 4 [P:4]

You see, that the first meter is mapped to the MODBUS registers 4x00001 to 4x00041. The second meter is mapped to the MODBUS registers 4x00042 to 4x000232, because the meter offers has much more MBUS data points.

MB Register	MBUS datatype	MB datatype	Content	MBUS index	MB value HEX	Current MB value	Meter name
4x00203	INT24[3]	UINT32	Nominal frequency	61	???	???	Meter 4 [P:4]
4x00205	FLOAT32[4]	FLOAT32	Energy:10^0 Wh	62	???	???	Meter 4 [P:4]
4x00207	FLOAT32[4]	FLOAT32	Energy:10^0 Wh-Export energy value	63	???	???	Meter 4 [P:4]
4x00209	FLOAT32[4]	FLOAT32	Energy:10^0 Wh[U:1,T:0,S:0]	64	???	???	Meter 4 [P:4]
4x00211	FLOAT32[4]	FLOAT32	Energy:10^0 Wh-Export energy value[U:1,T:0,S:0]	65	???	???	Meter 4 [P:4]
4x00213	FLOAT32[4]	FLOAT32	Energy:10^0 Wh-parital energy value	66	???	???	Meter 4 [P:4]
4x00215	FLOAT32[4]	FLOAT32	Energy:10^0 Wh-parital energy value[U:1,T:0,S:0]	67	???	???	Meter 4 [P:4]
4x00217	FLOAT32[4]	FLOAT32	Energy:10^0 Wh-L1 phase value	68	???	???	Meter 4 [P:4]
4x00219	FLOAT32[4]	FLOAT32	Energy:10^0 Wh-L2 phase value	69	???	???	Meter 4 [P:4]
4x00221	FLOAT32[4]	FLOAT32	Energy:10^0 Wh-L3 phase value	70	???	???	Meter 4 [P:4]
4x00223	FLOAT32[4]	FLOAT32	Cumulation counter	71	???	???	Meter 4 [P:4]
4x00225	FLOAT32[4]	FLOAT32	Energy:10^0 Wh[U:0,T:1,S:0]	72	???	???	Meter 4 [P:4]
4x00227	FLOAT32[4]	FLOAT32	Energy:10^0 Wh[U:0,T:2,S:0]	73	???	???	Meter 4 [P:4]
4x00229	FLOAT32[4]	FLOAT32	Energy:10^0 Wh[U:0,T:3,S:0]	74	???	???	Meter 4 [P:4]
4x00231	FLOAT32[4]	FLOAT32	Energy:10^0 Wh[U:0,T:4,S:0]	75	???	???	Meter 4 [P:4]
4x09001	RESI	UINT16	Converter state for meter	STATE	???	???	Meter 2 [P:2]
4x09002	HEADER	UINT32R	Identification number of meter	ID	???	???	Meter 2 [P:2]
4x09004	RESI	UINT16	Converter state for meter	STATE	???	???	Meter 4 [P:4]
4x09005	HEADER	UINT32R	Identification number of meter	ID	???	???	Meter 4 [P:4]
4x10001	HEADER	UINT32	Identification number of meter	ID	???	???	Meter 2 [P:2]
4x10003	HEADER	UINT32->ASCI	Manufacturer of meter	MANUFACTURER	???	???	Meter 2 [P:2]
4x10005	HEADER	UINT16	Version of meter	VERSION	???	???	Meter 2 [P:2]
4x10006	HEADER	UINT16	Medium of meter	MEDIUM	???	???	Meter 2 [P:2]
4x10007	HEADER	UINT16	Access of meter	ACCESS	???	???	Meter 2 [P:2]
4x10008	HEADER	UINT16	Status of meter	STATUS	???	???	Meter 2 [P:2]
4x10009	RESI	UINT16	Future value of meter	FUTURE	???	???	Meter 2 [P:2]
4x10010	RESI	UINT16	Communcation state with meter	COMM STATE	???	???	Meter 2 [P:2]
4x10011	HEADER	UINT32	Identification number of meter	ID	???	???	Meter 4 [P:4]
4x10013	HEADER	UINT32->ASCI	Manufacturer of meter	MANUFACTURER	???	???	Meter 4 [P:4]
4x10015	HEADER	UINT16	Version of meter	VERSION	???	???	Meter 4 [P:4]
4x10016	HEADER	UINT16	Medium of meter	MEDIUM	???	???	Meter 4 [P:4]
4x10017	HEADER	UINT16	Access of meter	ACCESS	???	???	Meter 4 [P:4]
4x10018	HEADER	UINT16	Status of meter	STATUS	???	???	Meter 4 [P:4]
4x10019	RESI	UINT16	Future value of meter	FUTURE	???	???	Meter 4 [P:4]
4x10020	RESI	UINT16	Communcation state with meter	COMM STATE	???	???	Meter 4 [P:4]

You can download the configuration and press the Test button. After a few seconds you will see the table filled with online values from the connected meter.

16.10.2 Status information for every meter

Behind the mapping from the MBUS data points to the MODBUS data points, you will see two areas of status information.

MB Register	MBUS datatype	MB datatype	Content	MBUS index	MB value HEX	Current MB value	Meter name
4x00203	INT24[3]	UINT32	Nominal frequency	61	MSW:0000,0032:LSW	50,0x00000032	Meter 4 [P:4]
4x00205	FLOAT32[4]	FLOAT32	Energy:10^0 Wh	62	MSW:3CCC,CCCD:LSW	0.0250,2.50000003725290E-2	Meter 4 [P:4]
4x00207	FLOAT32[4]	FLOAT32	Energy:10^0 Wh-Export energy value	63	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 4 [P:4]
4x00209	FLOAT32[4]	FLOAT32	Energy:10^0 Wh[U:1,T:0,S:0]	64	MSW:3C75,C28F:LSW	0.0150,1.49999996647239E-2	Meter 4 [P:4]
4x00211	FLOAT32[4]	FLOAT32	Energy:10^0 Wh-Export energy value[U:1,T:0,S:0]	65	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 4 [P:4]
4x00213	FLOAT32[4]	FLOAT32	Energy:10^0 Wh-parital energy value	66	MSW:3CCC,CCCD:LSW	0.0250,2.50000003725290E-2	Meter 4 [P:4]
4x00215	FLOAT32[4]	FLOAT32	Energy:10^0 Wh-parital energy value[U:1,T:0,S:0]	67	MSW:3C75,C28F:LSW	0.0150,1.49999996647239E-2	Meter 4 [P:4]
4x00217	FLOAT32[4]	FLOAT32	Energy:10^0 Wh-L1 phase value	68	MSW:3CCC,CCCD:LSW	0.0250,2.50000003725290E-2	Meter 4 [P:4]
4x00219	FLOAT32[4]	FLOAT32	Energy:10^0 Wh-L2 phase value	69	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 4 [P:4]
4x00221	FLOAT32[4]	FLOAT32	Energy:10^0 Wh-L3 phase value	70	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 4 [P:4]
4x00223	FLOAT32[4]	FLOAT32	Cumulation counter	71	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 4 [P:4]
4x00225	FLOAT32[4]	FLOAT32	Energy:10^0 Wh[U:0,T:1,S:0]	72	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 4 [P:4]
4x00227	FLOAT32[4]	FLOAT32	Energy:10^0 Wh[U:0,T:2,S:0]	73	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 4 [P:4]
4x00229	FLOAT32[4]	FLOAT32	Energy:10^0 Wh[U:0,T:3,S:0]	74	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 4 [P:4]
4x00231	FLOAT32[4]	FLOAT32	Energy:10^0 Wh[U:0,T:4,S:0]	75	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 4 [P:4]
4x09001	RESI	UINT16	Converter state for meter	STATE	WORD:0003	3,0x0003 -> Values are valid!	Meter 2 [P:2]
4x09002	HEADER	UINT32R	Identification number of meter	ID	LSW:6229,MSW:2071	544301609,0x20716229	Meter 2 [P:2]
4x09004	RESI	UINT16	Converter state for meter	STATE	WORD:0003	3,0x0003 -> Values are valid!	Meter 4 [P:4]
4x09005	HEADER	UINT32R	Identification number of meter	ID	LSW:4163,MSW:0636	104218979,0x06364163	Meter 4 [P:4]
4x10001	HEADER	UINT32	Identification number of meter	ID	MSW:2071,6229:LSW	544301609,0x20716229	Meter 2 [P:2]
4x10003	HEADER	UINT32->ASCI	Manufacturer of meter	MANUFACTURER	MSW:004D,414B:LSW	KAM	Meter 2 [P:2]
4x10005	HEADER	UINT16	Version of meter	VERSION	WORD:001D	29,0x001D	Meter 2 [P:2]
4x10006	HEADER	UINT16	Medium of meter	MEDIUM	WORD:0016	22,0x0016 -> Cold Water	Meter 2 [P:2]
4x10007	HEADER	UINT16	Access of meter	ACCESS	WORD:0072	114,0x0072	Meter 2 [P:2]
4x10008	HEADER	UINT16	Status of meter	STATUS	WORD:0000	0,0x0000	Meter 2 [P:2]
4x10009	RESI	UINT16	Future value of meter	FUTURE	WORD:0000	0,0x0000	Meter 2 [P:2]
4x10010	RESI	UINT16	Communcation state with meter	COMM STATE	WORD:0003	3,0x0003 -> Values are valid!	Meter 2 [P:2]
4x10011	HEADER	UINT32	Identification number of meter	ID	MSW:0636,4163:LSW	104218979,0x06364163	Meter 4 [P:4]
4x10013	HEADER	UINT32->ASCI	Manufacturer of meter	MANUFACTURER	MSW:0043,4553:LSW	SEC	Meter 4 [P:4]
4x10015	HEADER	UINT16	Version of meter	VERSION	WORD:0018	24,0x0018	Meter 4 [P:4]
4x10016	HEADER	UINT16	Medium of meter	MEDIUM	WORD:0002	2,0x0002 -> Electricity	Meter 4 [P:4]
4x10017	HEADER	UINT16	Access of meter	ACCESS	WORD:0014	20,0x0014	Meter 4 [P:4]
4x10018	HEADER	UINT16	Status of meter	STATUS	WORD:0000	0,0x0000	Meter 4 [P:4]
4x10019	RESI	UINT16	Future value of meter	FUTURE	WORD:0000	0,0x0000	Meter 4 [P:4]
4x10020	RESI	UINT16	Communcation state with meter	COMM STATE	WORD:0003	3,0x0003 -> Values are valid!	Meter 4 [P:4]

Area 1 is compatible to our old MBUS converter modules, but it is located in a different area of the MBUS registers starting at 4x09001. For every configured meter two MODBUS entries are generated. One holds the communication state of the MBUS gateway with the meter with the following states:

- **0 - Meter isn't configured!**: This value shows, that this meter slot is currently not configured in the MBUS gateway.
- **1 - Meter isn't normalized!**: This value shows, that the configured meter doesn't answer to the addressing command. Either via primary addressing or via secondary addressing mode. This depends, how the meter was configured.
- **2 - Meter isn't read!**: This value shows, that the configured meter has answered to the addressing command but there are problems by reading all data from the meter. So the meter data is not valid any more.

- **3 - Values are valid!**: This value shows, that the configured meter has answered to the addressing command and has answered correctly to the readout commands and the reading of all data from the meter was successful. So the meter data in the MODBUS register is valid.

The other entry holds the serial number of the configured meter in two consecutive holding registers.

Area 2 is new to the new series of gateways and represent the information of the MBUS fixed data header.

Ident. Nr.	Manufr.	Version	Medium	Access No.	Status	Signature
4 Byte	2 Byte	1 Byte	1 Byte	1 Byte	1 Byte	2 Byte

This header is sent by many answer frames of the MBUS meter to the master. Due to the fact, that is is not part of the variable data block of the meter, our old converters could not map this information to registers. Our new series map this information to the following register set starting at 4x10001. For each meter there are eight MODBUS entires:

ENTRY 1: Identification number of the meter
Register <METERBASE>+0, <METERBASE>+1
Each meter offers a unique ID. In the MBUS protocol there are four bytes reserved for this number. In our gateway we need a UINT32 to represent this 4 bytes of the ID.

ENTRY 2: Manufacturer of the meter
Register <METERBASE>+2, <METERBASE>+3
Each meter offers a manufacturer ID, represented in two bytes. But in this two bytes there are three ASCII digits encoded. Our gateway decode this ASCII digits and stores this digits into a UINT32 using ASCII encoding with 0x00 at the end representing a standard null terminated ASCII string of three letters.

ENTRY 3: Version of the meter
Register <METERBASE>+4
In this fixed data header, there is also a version number encoded into one byte. It represents the version of the meter. Our gateway stores this byte into a UINT16 holding register for easy readout.

ENTRY 4: Medium of the meter
Register <METERBASE>+5
In this fixed data header, there is also a medium number encoded into one byte. it defines what type of medium the meter is measuring. Our gateway stores this byte into a UINT16 holding register for easy readout.

The following medium types are defined by the standard for meters with fixed+variable data structure:

- 0x00: OTHER
- 0x01: OIL
- 0x02: Electricity
- 0x03: Gas
- 0x04: Heat-Volume measured at return temperature outlet
- 0x05: Steam
- 0x06: Hot Water
- 0x07: Water
- 0x08: H.C.A.=Heat Cost Allocator
- 0x09: Compressed Air
- 0x0A: Cooling load meter Volume measured at return temperature outlet
- 0x0B: Cooling load meter Volume measured at flow temperature inlet
- 0x0C: Heat Volume measured at flow temperature inlet
- 0x0D: Heat/Cooling load meter
- 0x0E: Bus/System
- 0x0F: Unknown Medium
- 0x16: Cold Water
- 0x17: Dual Water
- 0x18: Pressure
- 0x19: A/D Converter

For meters with fixed data structure only, the 16 bit value must be interpreted in another way. Refer to the MBUS standard for this definition.

ENTRY 5: Access counter of the meter
Register <METERBASE>+6
In this fixed data header, there is also an access counter encoded into one byte. It will be incremented by every access of the meter data. So each readout of the meter will increment this access counter by 1 in the range from 0 to 255. Our gateway stores this byte into a UINT16 holding register for easy readout.

ENTRY 6: Status of the meter
Register <METERBASE>+7
In this fixed data header, there is also a status field encoded into one byte. It shows the current meter status. Our gateway stores this byte into a UINT16 holding register for easy readout.

The byte has the following meaning:

- Bit 1+Bit 0: =00 (0) NO ERROR
- Bit 1+Bit 0: =10 (1) APPLICATION NOT READY
- Bit 1+Bit 0: =01 (2) APPLICATION ERROR
- Bit 1+Bit 0: =11 (3) RESERVED
- Bit 2: =1: POWER LOW, =0: POWER OK
- Bit 3: =1: PERMANENT ERROR, =0: NO PERMANENT ERROR
- Bit 4: =1: TEMPORARY ERROR, =0: NO TEMPORARY ERROR
- Bit 5: =1: MANUFACTURER SPECIFIC ERROR 1, =0: NO MANUFACTURER SPECIFIC ERROR 1
- Bit 6: =1: MANUFACTURER SPECIFIC ERROR 2, =0: NO MANUFACTURER SPECIFIC ERROR 2
- Bit 7: =1: MANUFACTURER SPECIFIC ERROR 3, =0: NO MANUFACTURER SPECIFIC ERROR 3

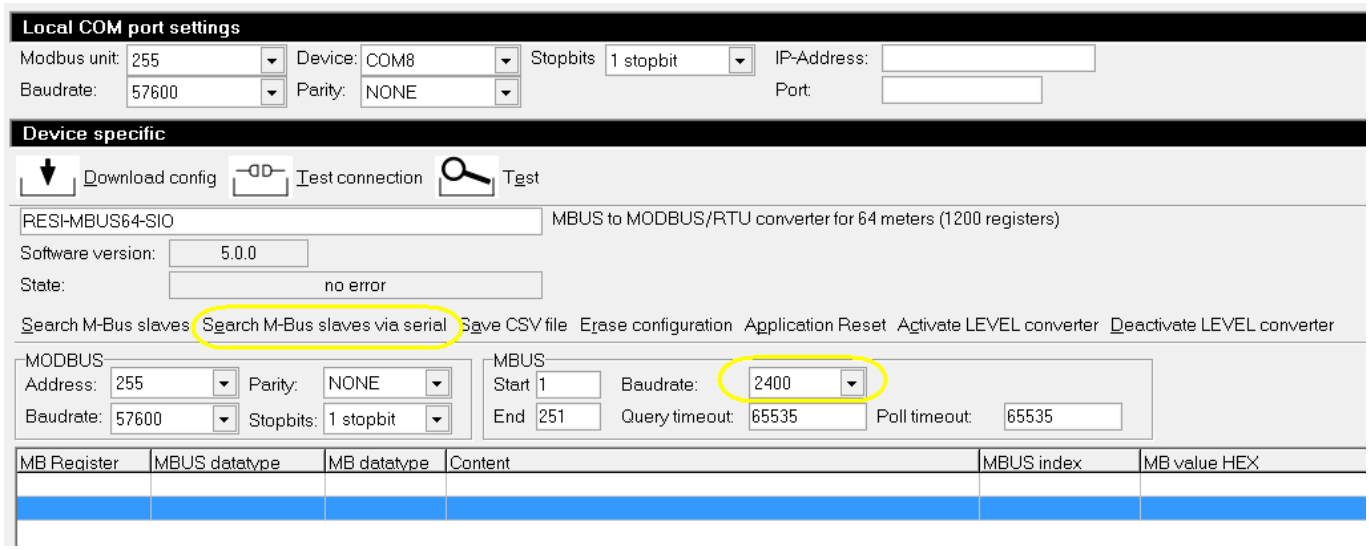
ENTRY 7: Future value of the meter
Register <METERBASE>+8
This UINT16 holding register is reserved for future use.

ENTRY 8: Communication state with meter
Register <METERBASE>+9
This UINT16 holding register hold the current state of the communication between the MBUS gateway and the meter with the following states:

- **0 - Meter isn't configured!**: This value shows, that this meter slot is currently not configured in the MBUS gateway.
- **1 - Meter isn't normalized!**: This value shows, that the configured meter doesn't answer to the addressing command. Either via primary addressing or via secondary addressing mode. This depends, how the meter was configured.
- **2 - Meter isn't read!**: This value shows, that the configured meter has answered to the addressing command but there are problems by reading all data from the meter. So the meter data is not valid any more.
- **3 - Values are valid!**: This value shows, that the configured meter has answered to the addressing command and has answered correctly to the readout commands and the reading of all data from the meter was successful. So the meter data in the MODBUS register is valid.

16.10.3 Search for new meters – secondary addressing mode

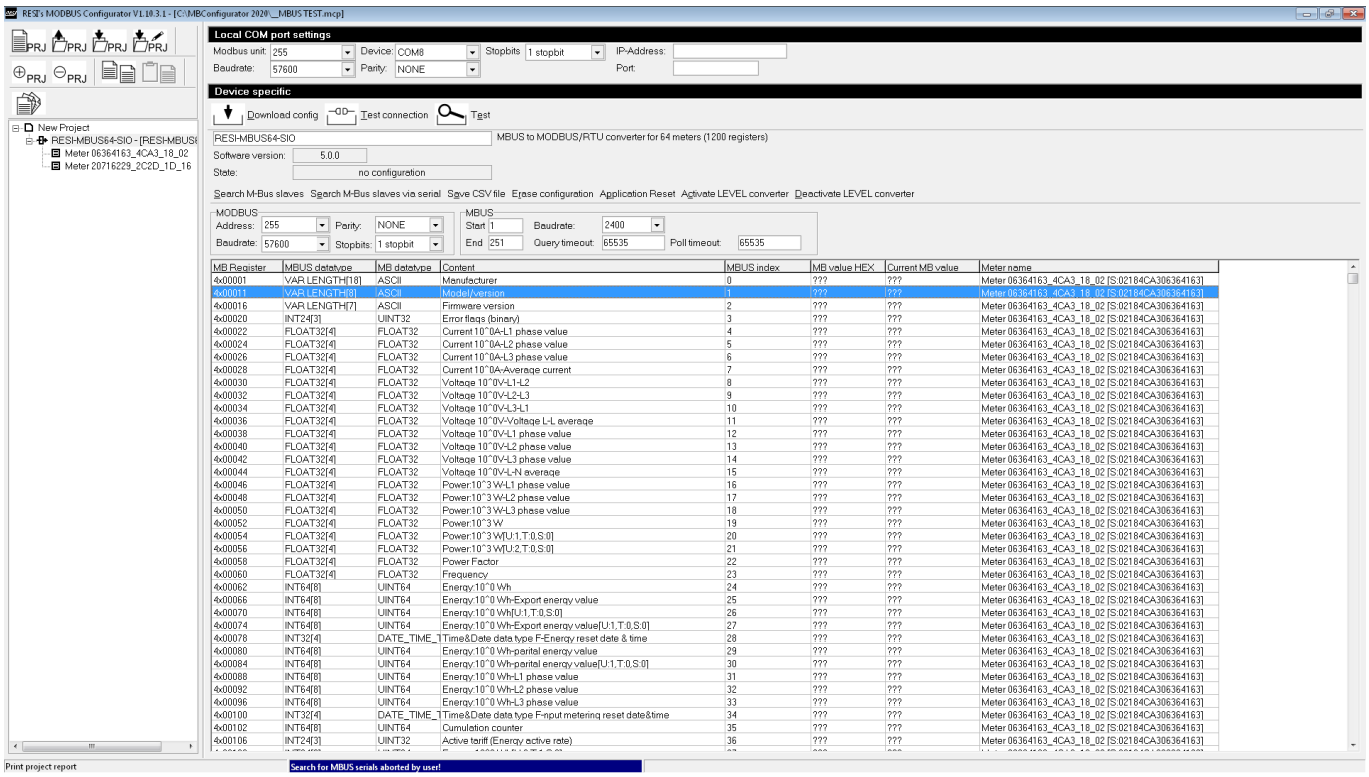
HINT: Don't forget to setup the MBUS bus baud rate for your search before. If you have to change it, select a new one from the drop down list and don't forget to download the bus speed into your gateway!



Click on the button Search M-Bus slaves via serial to start the automatic search with secondary addressing mode for the meters. Now the software does a pattern search for new connected MBUS meters. The benefit with this method is, that you don't have to program any primary addresses into your meters.

HINT: You can interrupt the automatic search process by pressing the ESC button. After a few seconds the search will be interrupted.

In our test case the screen will look like this:

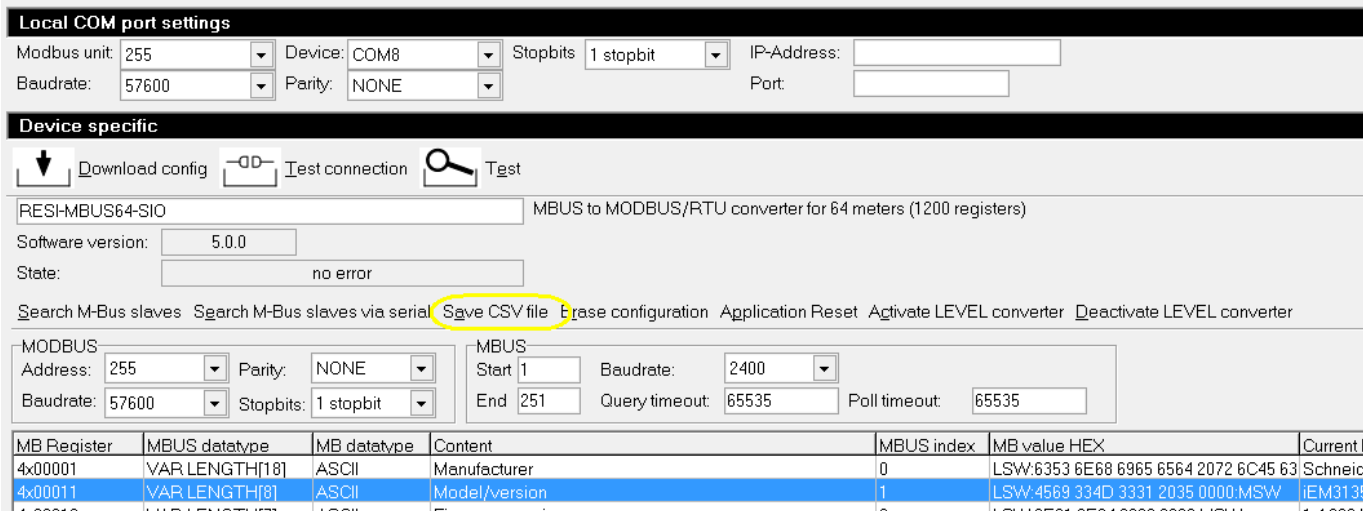


Again you can now download the configuration and start a quick test by activating the test mode with the button Test. Now you will see values in the data grid after a few seconds:

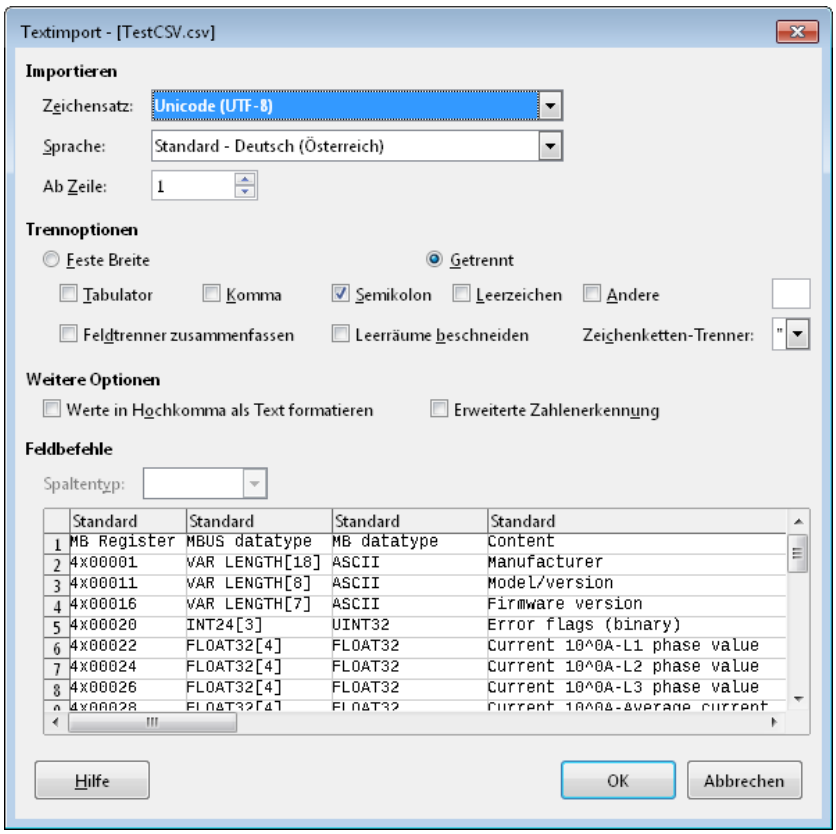
MB Register	MBUS datatype	MB datatype	Content	MBUS index	MB value HEX	Current MB value	Meter name
4x00001	VAR LENGTH[18]	ASCII	Manufacturer	0	LSW:6353 6E68 6965 6564 2072 6C45 63	Schneider Electric,LSB:53 63 68 6E 65 69 64 65 72 20 45 6C 65 63 74 72 69 63 00:MSB	Meter 06364163_
4x00011	VAR LENGTH[8]	ASCII	Model/version	1	LSW:4569 334D 3331 2035 0000:MSW	iEM3135,LSB:69 45 4D 33 31 33 35 20 00:MSB	Meter 06364163_
4x00016	VAR LENGTH[7]	ASCII	Firmware version	2	LSW:2E31 2E34 3030 0032:MSW	1.4.002,LSB:31 2E 34 2E 30 30 32 00:MSB	Meter 06364163_
4x00020	INT24[3]	UINT32	Error flags (binary)	3	MSW:0000,0000:LSW	0,0x00000000	Meter 06364163_
4x00022	FLOAT32[4]	FLOAT32	Current 10^0A-L1 phase value	4	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 06364163_
4x00024	FLOAT32[4]	FLOAT32	Current 10^0A-L2 phase value	5	MSW:FFC0,0000:LSW	NAN,NAN	Meter 06364163_
4x00026	FLOAT32[4]	FLOAT32	Current 10^0A-L3 phase value	6	MSW:FFC0,0000:LSW	NAN,NAN	Meter 06364163_
4x00028	FLOAT32[4]	FLOAT32	Current 10^0A-Average current	7	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 06364163_
4x00030	FLOAT32[4]	FLOAT32	Voltage 10^0V-L1-L2	8	MSW:FFC0,0000:LSW	NAN,NAN	Meter 06364163_
4x00032	FLOAT32[4]	FLOAT32	Voltage 10^0V-L2-L3	9	MSW:FFC0,0000:LSW	NAN,NAN	Meter 06364163_
4x00034	FLOAT32[4]	FLOAT32	Voltage 10^0V-L3-L1	10	MSW:FFC0,0000:LSW	NAN,NAN	Meter 06364163_
4x00036	FLOAT32[4]	FLOAT32	Voltage 10^0V-Voltage L-L average	11	MSW:FFC0,0000:LSW	NAN,NAN	Meter 06364163_
4x00038	FLOAT32[4]	FLOAT32	Voltage 10^0V-L1 phase value	12	MSW:4366,04AC:LSW	230.0182,2.30018249511719E+2	Meter 06364163_
4x00040	FLOAT32[4]	FLOAT32	Voltage 10^0V-L2 phase value	13	MSW:FFC0,0000:LSW	NAN,NAN	Meter 06364163_
4x00042	FLOAT32[4]	FLOAT32	Voltage 10^0V-L3 phase value	14	MSW:FFC0,0000:LSW	NAN,NAN	Meter 06364163_
4x00044	FLOAT32[4]	FLOAT32	Voltage 10^0V-L-N average	15	MSW:4366,04AC:LSW	230.0182,2.30018249511719E+2	Meter 06364163_
4x00046	FLOAT32[4]	FLOAT32	Power:10^3 W-L1 phase value	16	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 06364163_
4x00048	FLOAT32[4]	FLOAT32	Power:10^3 W-L2 phase value	17	MSW:FFC0,0000:LSW	NAN,NAN	Meter 06364163_
4x00050	FLOAT32[4]	FLOAT32	Power:10^3 W-L3 phase value	18	MSW:FFC0,0000:LSW	NAN,NAN	Meter 06364163_
4x00052	FLOAT32[4]	FLOAT32	Power:10^3 W	19	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 06364163_
4x00054	FLOAT32[4]	FLOAT32	Power:10^3 W[U:1,T:0,S:0]	20	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 06
4x00056	FLOAT32[4]	FLOAT32	Power:10^3 W[U:2,T:0,S:0]	21	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 06
4x00058	FLOAT32[4]	FLOAT32	Power Factor	22	MSW:FFC0,0000:LSW	NAN,NAN	Meter 06
4x00060	FLOAT32[4]	FLOAT32	Frequency	23	MSW:4247,FDFD:LSW	49.9980,4.99980354309082E+1	Meter 06364163_
4x00062	INT64[8]	UINT64	Energy:10^0 Wh	24	MSW:0000000000000,0019:LSW	25,0x00000019	Meter 06364163_
4x00066	INT64[8]	UINT64	Energy:10^0 Wh-Export energy value	25	MSW:0000000000000,0000:LSW	0,0x00000000	Meter 06364163_
4x00070	INT64[8]	UINT64	Energy:10^0 Wh[U:1,T:0,S:0]	26	MSW:0000000000000,000F:LSW	15,0x0000000F	Meter 06364163_
4x00074	INT64[8]	UINT64	Energy:10^0 Wh-Export energy value[U:1,T:0,S:0]	27	MSW:0000000000000,0000:LSW	0,0x00000000	Meter 06364163_
4x00078	INT32[4]	DATE_TIME_T	Time&Date data type F-Energy reset date & time	28	MSW:0101,2080:LSW	00:00 D.M.Y:01.01.00 ST:0 IV:1,0x01012080	Meter 06364163_
4x00080	INT64[8]	UINT64	Energy:10^0 Wh-parital energy value	29	MSW:0000000000000,0019:LSW	25,0x00000019	Meter 06364163_
4x00084	INT64[8]	UINT64	Energy:10^0 Wh-parital energy value[U:1,T:0,S:0]	30	MSW:0000000000000,000F:LSW	15,0x0000000F	Meter 06364163_
4x00088	INT64[8]	UINT64	Energy:10^0 Wh-L1 phase value	31	MSW:0000000000000,0019:LSW	25,0x00000019	Meter 06364163_
4x00092	INT64[8]	UINT64	Energy:10^0 Wh-L2 phase value	32	MSW:0000000000000,0000:LSW	0,0x00000000	Meter 06364163_
4x00096	INT64[8]	UINT64	Energy:10^0 Wh-L3 phase value	33	MSW:0000000000000,0000:LSW	0,0x00000000	Meter 06364163_
4x00100	INT32[4]	DATE_TIME_T	Time&Date data type F-nput metering reset date&time	34	MSW:0101,2080:LSW	00:00 D.M.Y:01.01.00 ST:0 IV:1,0x01012080	Meter 06364163_
4x00102	INT64[8]	UINT64	Cumulation counter	35	MSW:0000000000000,0000:LSW	0,0x00000000	Meter 06364163_

16.10.4 Save to CSV file

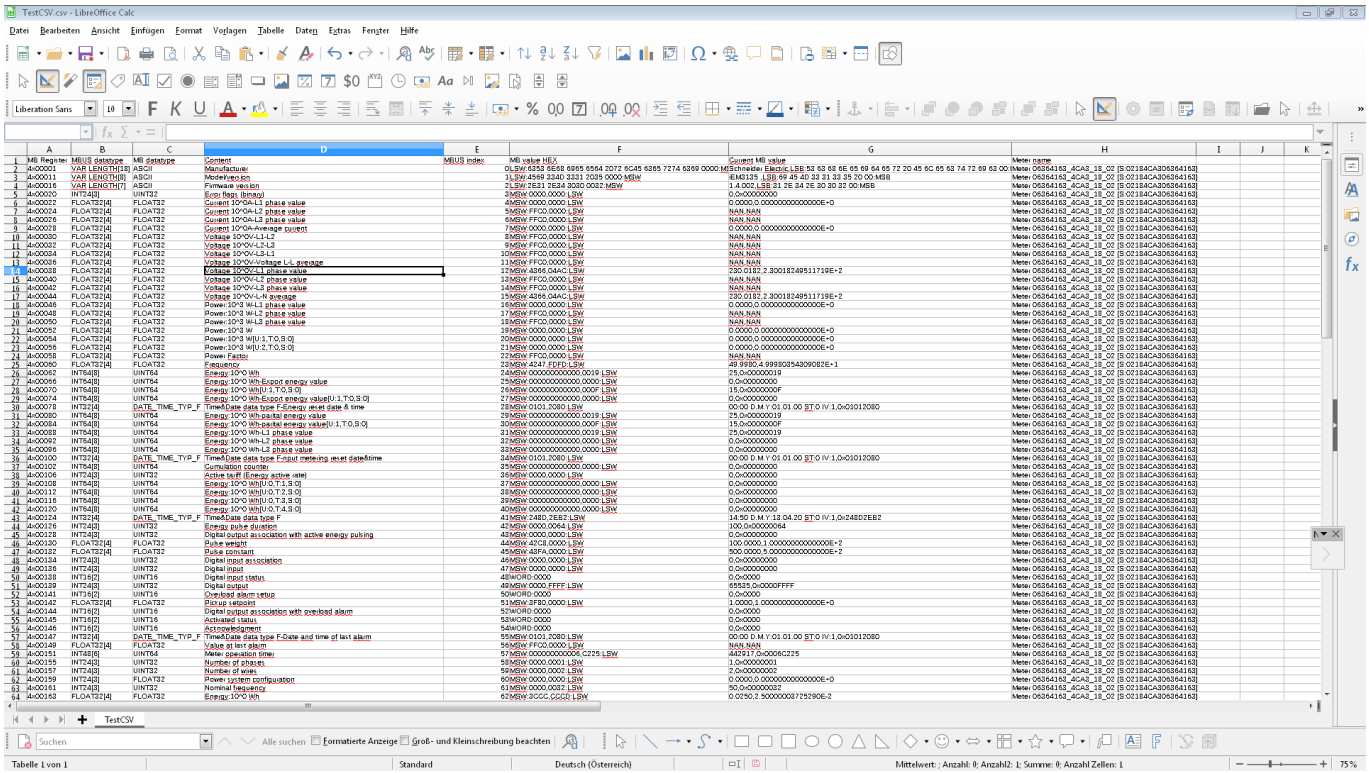
With the action Save to CSV file you can store the current data of the data grind into a CSV file for processing in Libre Office® or Microsoft Office® calculation software.



Click on the button Save CSV file. A dialog for entering the name of the CSV file will be opened. After you defined the name, the CSV file is on your file system. Take a calculation software to open the CSV file (in our case Libre office), select Semicolon as a separator and open the CSV file.

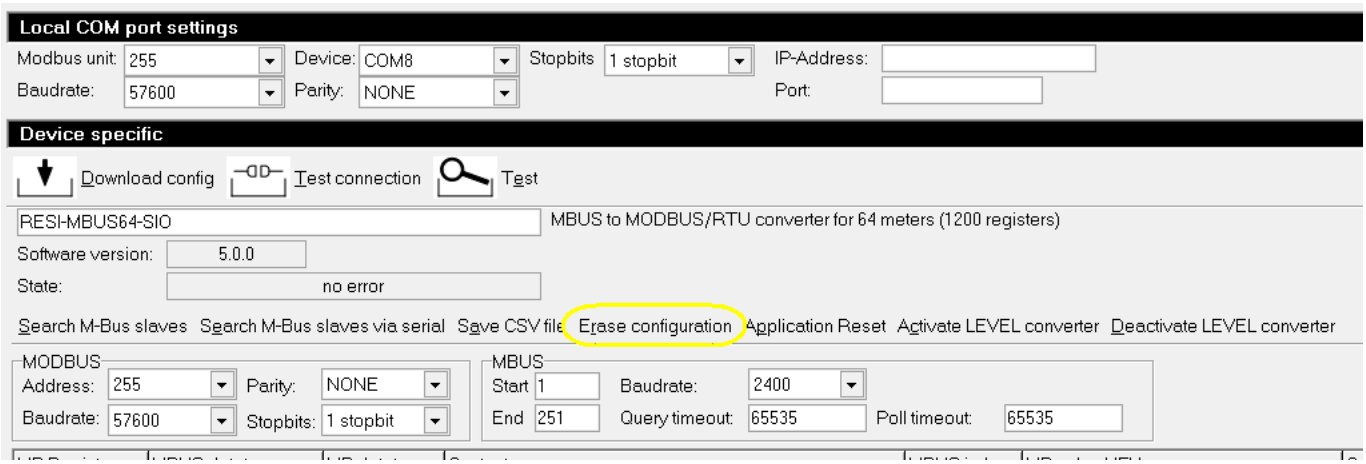


You will see the complete data grid in your calc software for your own purposes:



16.10.5 Erase configuration

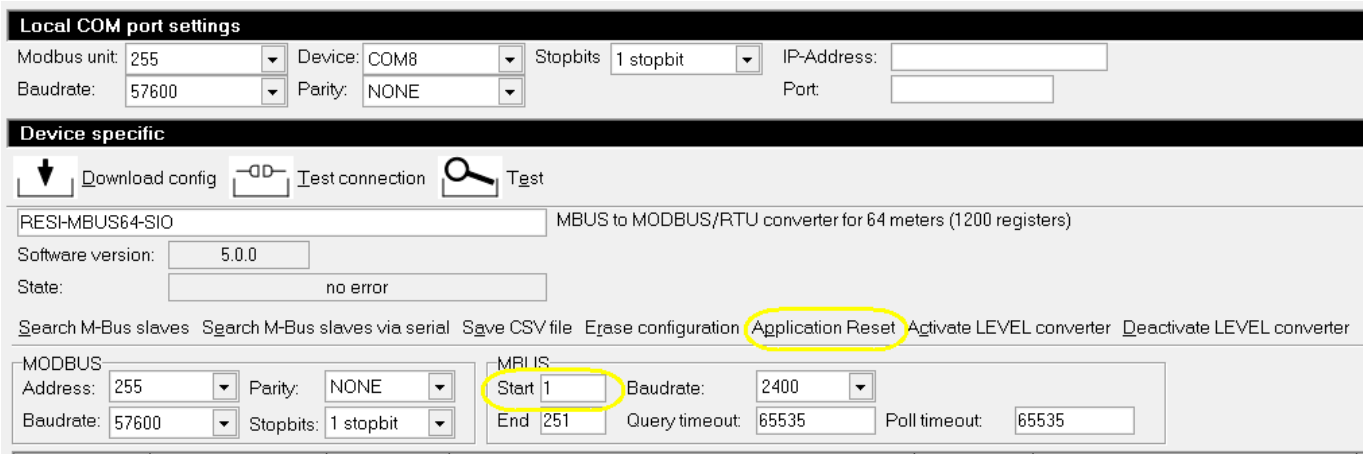
With the action Erase configuration you can delete the complete configuration of the gateway and restore factory settings for all parameters.



Click on the button Erase configuration. A question will pop up. If you answer with YES, the gateway will be restored to factory defaults and the meter configuration will be erased.

16.10.6 Application reset

With the action Application reset you can send the special MBUS command "Application reset" to a defined MBUS meter.



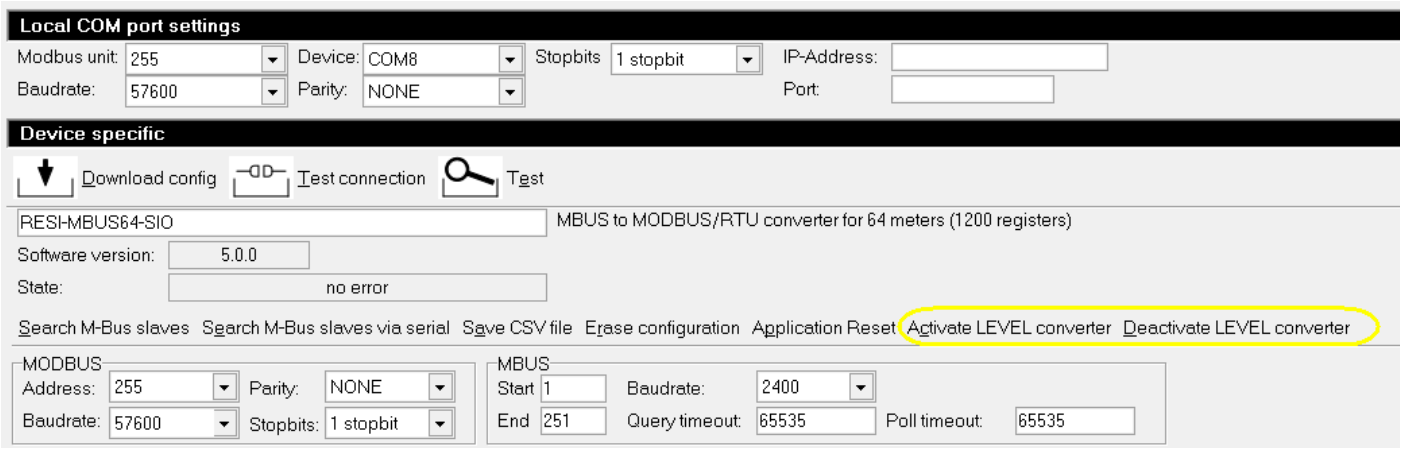
Select the desired primary address for this action with the filed Start in the MBUS area. Then click on the button Application reset. A question will pop up. If you answer with YES, the gateway will send the special MBUS command Application reset to the selected meter.

This is helpful, because some of the meters have trouble to resynchronize to the start of data readout when do a lot of connection /disconnection or other electrical stuff on the MBUS line. There it helps to send this command before trying to search for the connected meter.

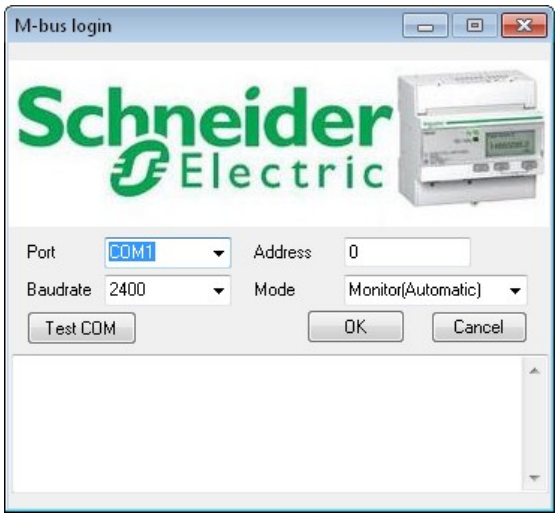
16.10.7 Activate/Deactivate LEVEL converter

With the two actions Activate/deactivate LEVEL converter you cans witch the MBUS gateway to a transparent mode, where every incoming MBUS data is directly send to the host and every incoming characters from the host are send to the MBUS line directly. Also a baud rate conversion will be done. The serial line will use the settings for the serial interface and the MBUS line will use the settings for the MBUS interface.

The integrated LEVEL converter is designed to configure meters with individual software form manufacturers over a standard level converter. Usually you have to have another MBUS level converter module either from RESI or from other suppliers like RELAY ® to configure your meters. Now you can do this over our gateway.

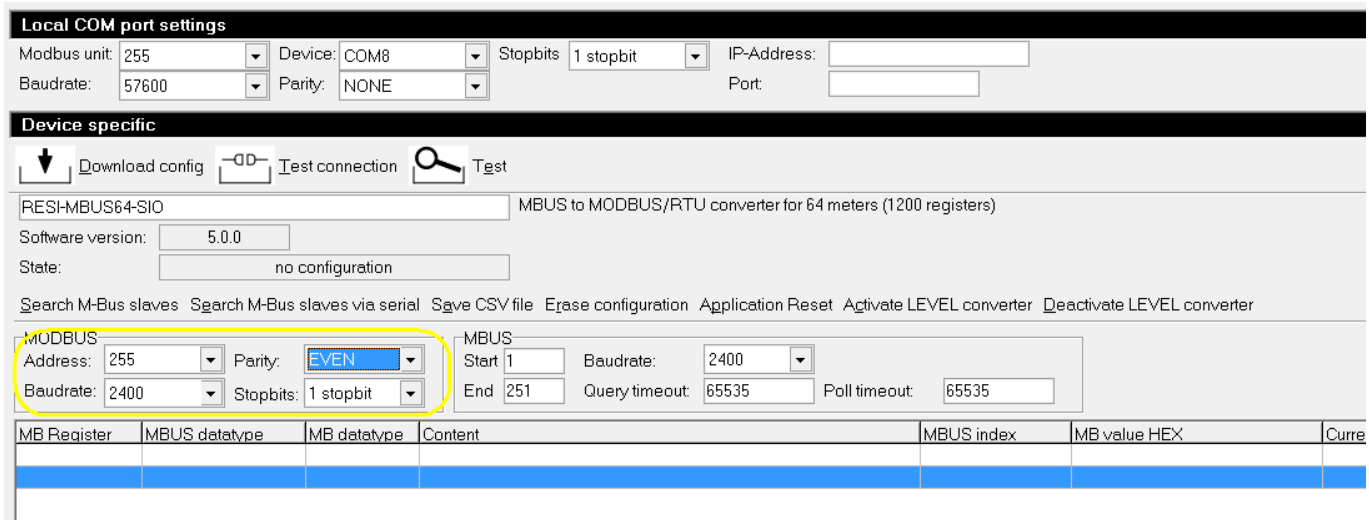


In our test szenario, we want to connect to a Schneider Electric meter with the original Schneider Electric configuration software. So when we start the software, we get the following screen:

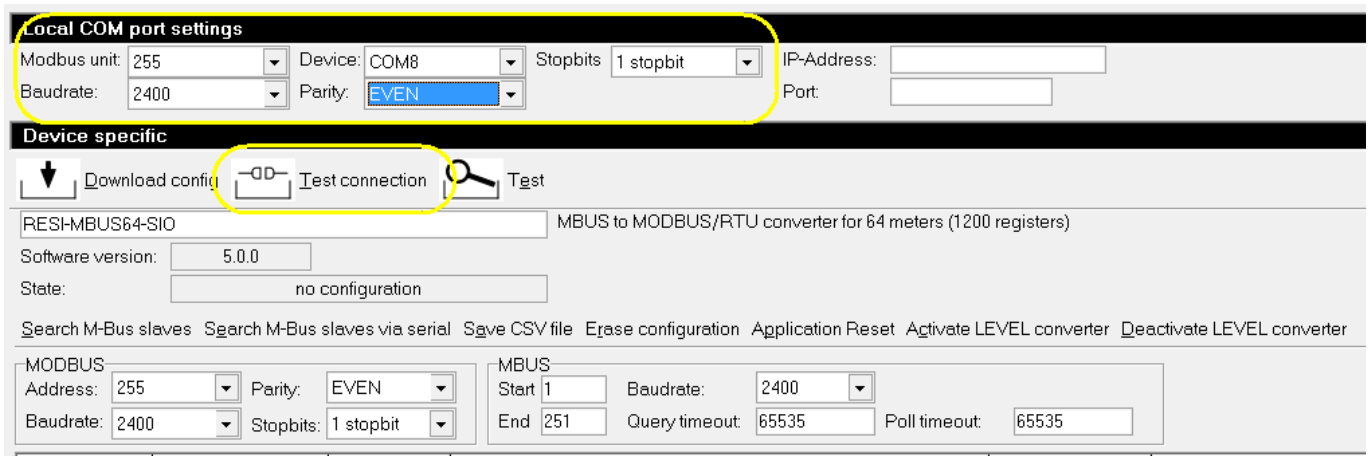


So first of all we have to change the speed settings for our gateway to parameters which are suitable to most of the MBUS tools on the market. Since the MBUS standard defines 2400bd, even parity and one stop bit as common on the MBUS side and many MBUS gateways are simple electrical converters, the tools assume a gateway with 2400bd, EVEN parity and one stop bit.

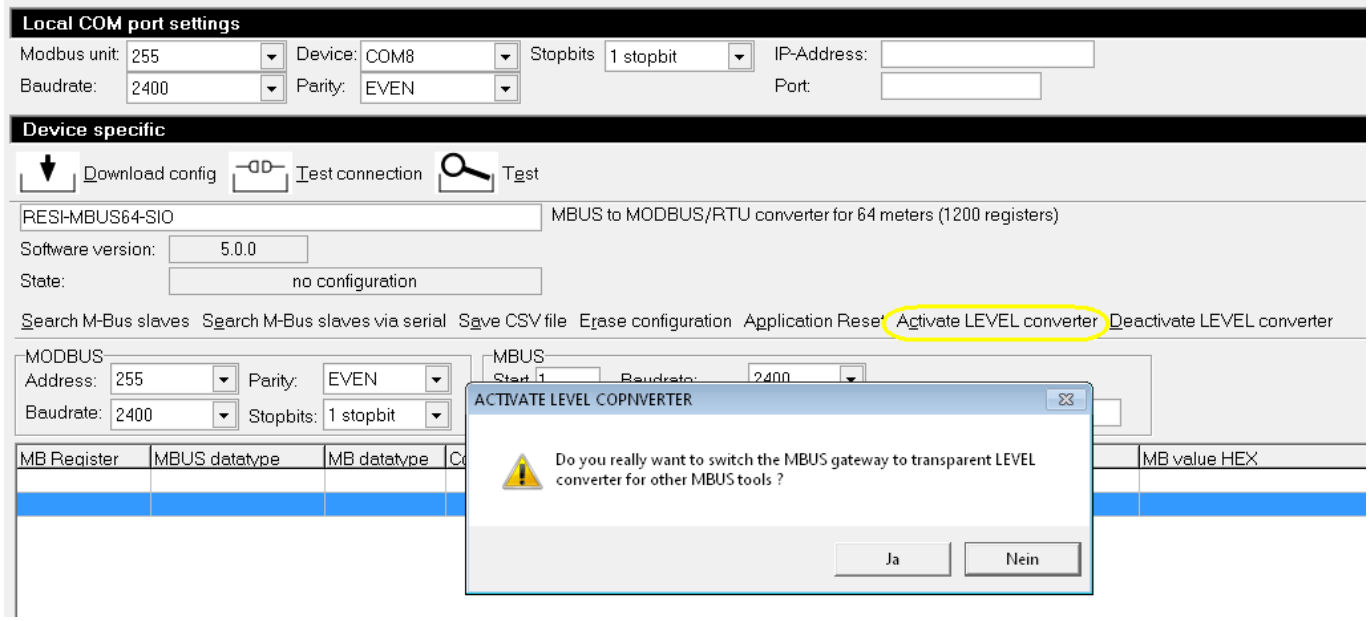
Select 2400bd, even parity and 1 stopbit in the area MODBUS and download this configuration with the button Download config.



The adopt your local COM settings for this new settings in the converter. Check the connection with the button Test connection.

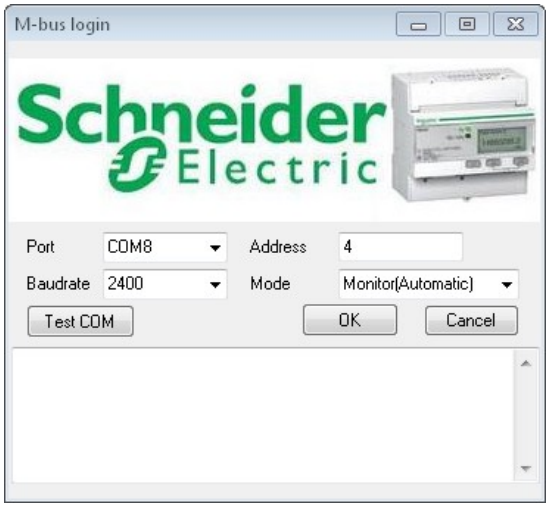


After that activate the integrated LEVEL converter by pressing the button Activate LEVEL converter.

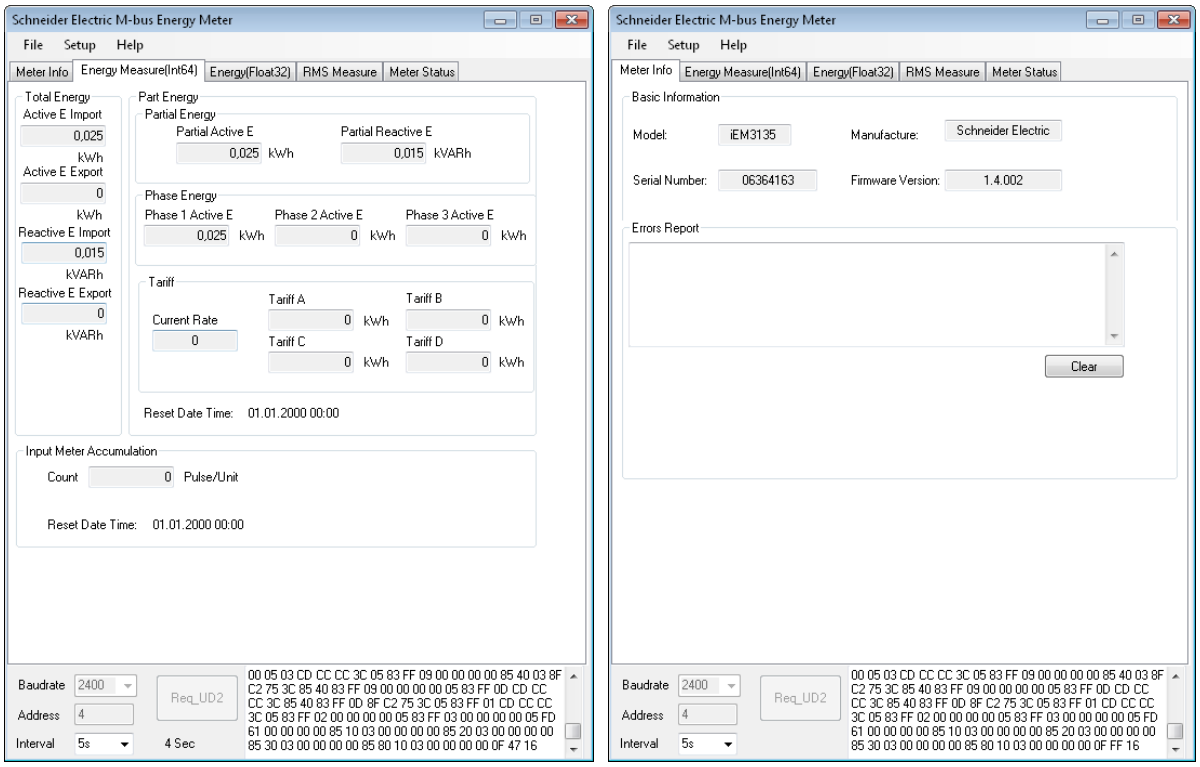


if everything is ok, the converter LED state will flash extraordinary fast to show, that the LEVEL converter is active.

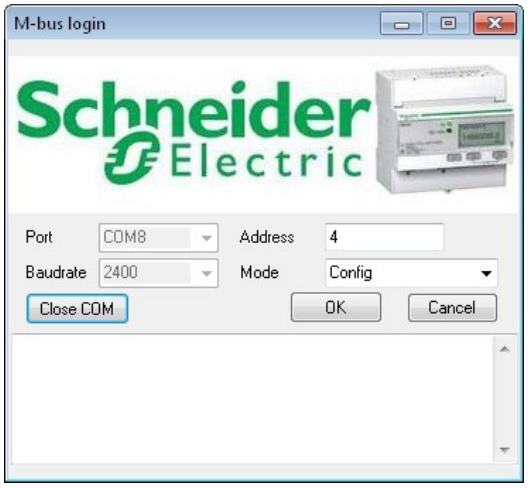
Now we activate the SCHNEIDER software by selecting the correct COM port and the correct primary address and the desired mode:



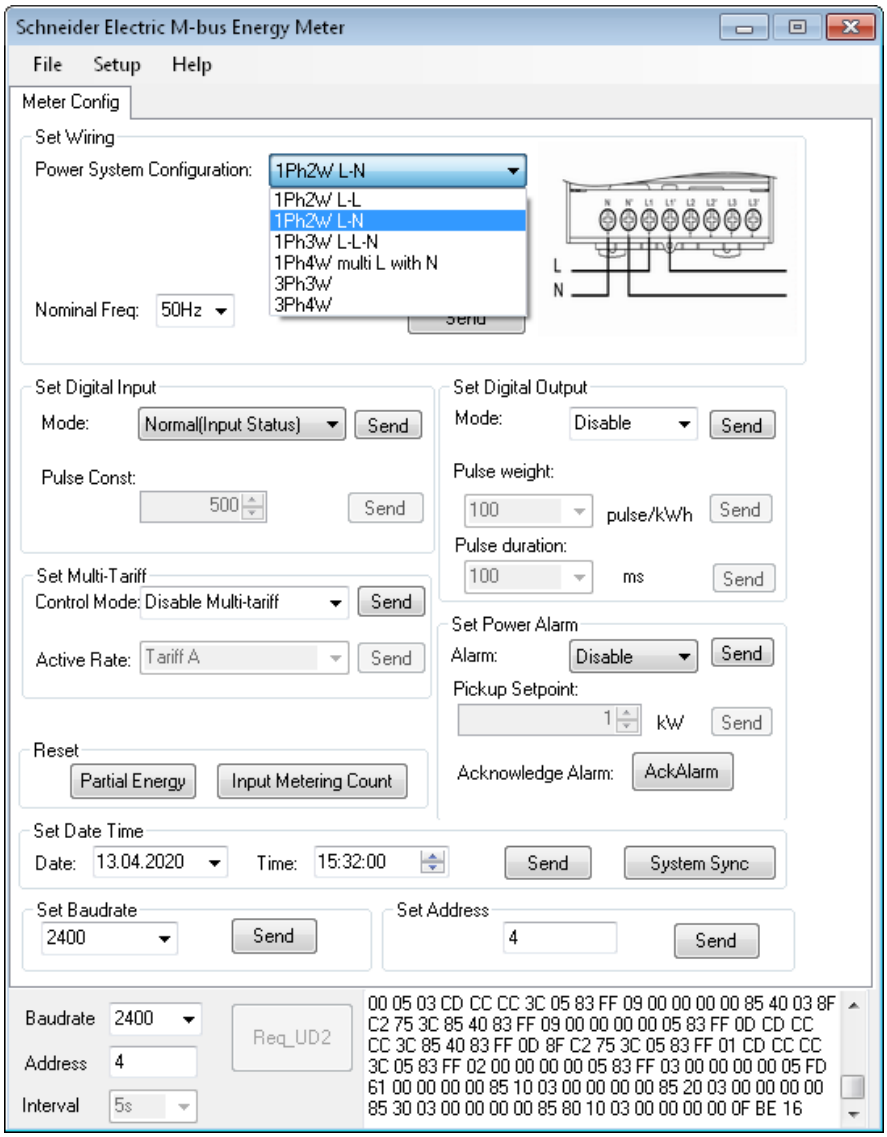
Press the button Test COM first, then press the OK button. The software will now scan automatically the meter and show the result on the screen:



If you start the MBUS configuration software in mode CONFIG



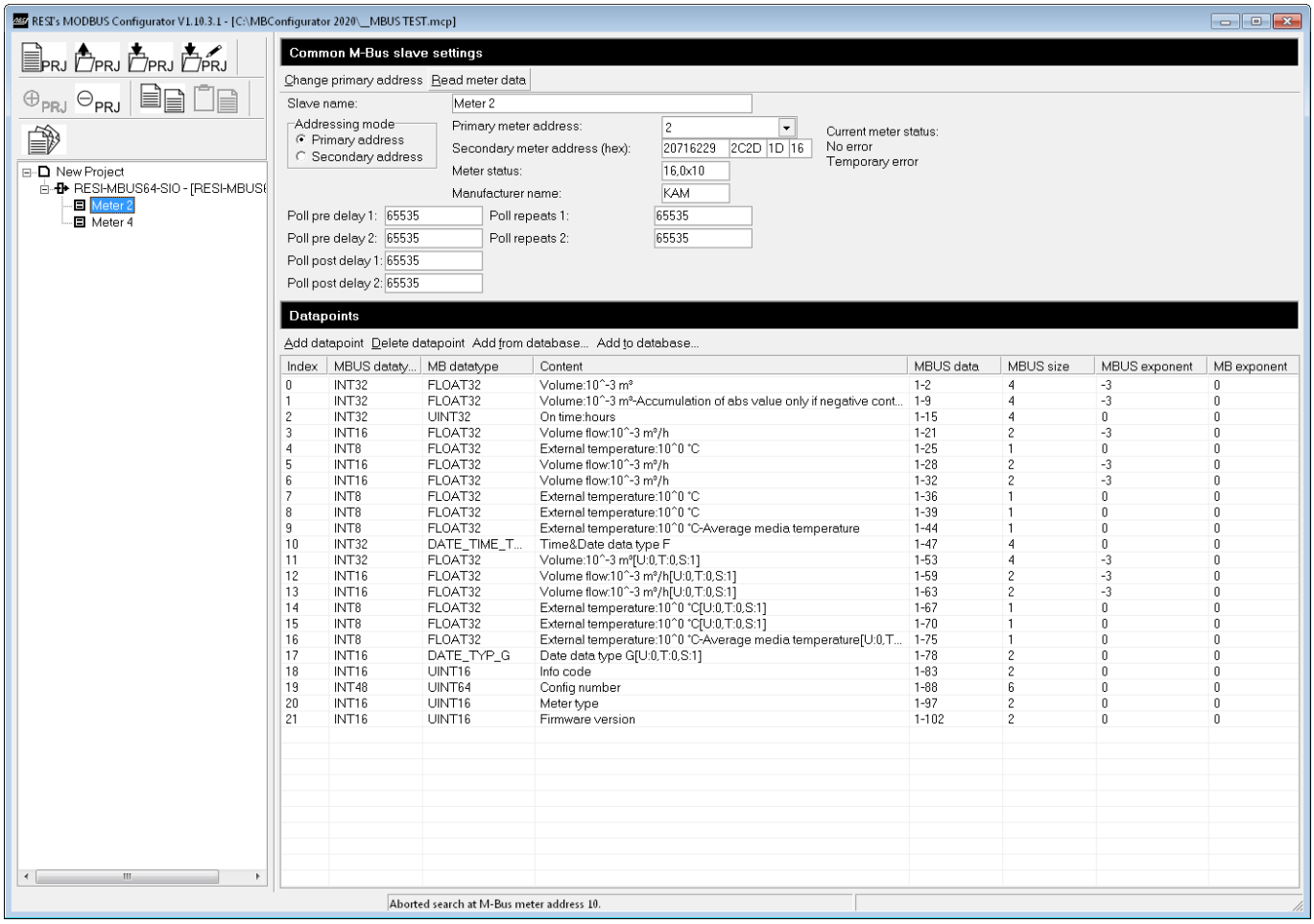
Click again first on Test COM button then on OK button. You will get the following screen:



After you have exited the SCHNEIDER software you can deactivate the LEVEL function either by disconnecting/reconnecting the power supply from the gateway (hard reset) or by pressing the button Deactivate LEVEL converter in the software. This will do a software reset and the STATE LED will flash normally again.

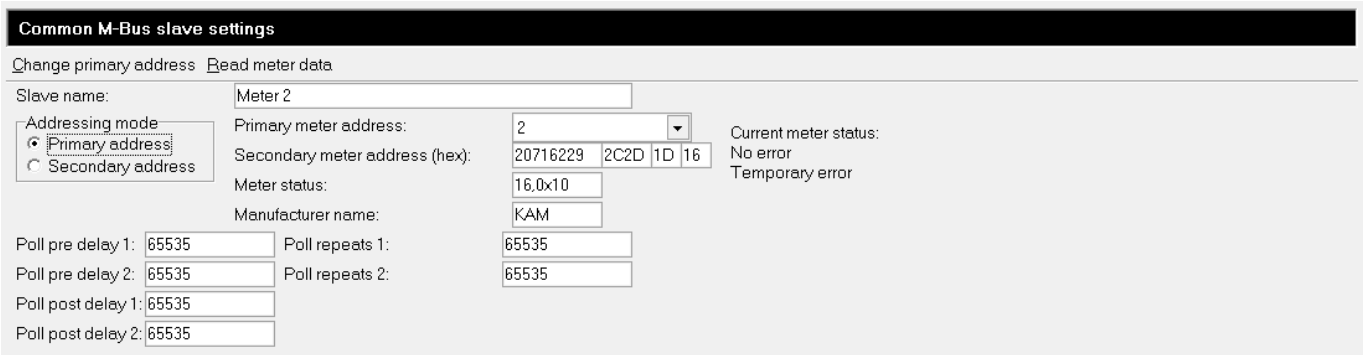
16.10.8 MBUS meter configuration

In the project tree you will find under the MBUS gateway for every configured meter a unique node. Click on this node. You will get the following result:



16.10.8.1 WHAT is displayed in the Common M-Bus slave settings

In this area you will find the following information:



- **Button Change primary address**: With this function you can program a new primary address in the selected meter, as long as the meter supports the standard MBUS command for setting a new primary address.
- **Button Read meter data**: With this function you can read out all MBUS datapoints from the connected meter again in the below data grid. This is useful, if you have erroneous deleted some datapoints of the meter and you want to restore the original datapoints of the meter.
- **Slave name**: Here you can define the name of the meter for the tree view and the documentation.
- **Addressing mode radio button**: This radio button selects the addressing mode for this meter. Either primary addressing mode in combination with the selected primary address in the field Primary meter address or Secondary addressing mode in combination with the first of the four fields in the row Secondary meter address. this is the field serial number of the meter.
- **Primary meter address**: This drop down defines the primary address for the meter either for readout or for programming a new primary address. Use 1 to 251 for slave address or if you have connected only one meter use 254 (Broadcast address), if you don't know the correct primary address.
- **Secondary meter address (hex)**: This four fields represents the following information:
  - **Serial number**: The first field is the current serial number of the meter. Or you enter a desired serial number for secondary addressing mode for a specific meter.
  - **Manufacturer ID**: The second field represents the two bytes of the manufacturer ID from the fixed data structure at the beginning of a variable data frame of the meter. The manufacturer is defined by three ASCII uppercase characters encoded with the following formula (In our example 2C2D stands for KAM=KAMSTRUP):
    IEC 870 Man.ID=[ASCII(1st letter)-64]•32•32+[ASCII(2nd letter)-64]•32+[ASCII(3rd letter)-64]
  - **Version**: The third field represents one byte from the fixed data structure at the beginning of a variable data frame of the meter defining the version of the meter.
  - **Medium**: The fourth field represents one byte from the fixed data structure at the beginning of a variable data frame of the meter defining the medium of the meter.
  - **Meter status**: This field represents one byte from the fixed data structure at the beginning of a variable data frame of the meter defining the status of the meter. Beside this field you will see under the caption current meter status the interpretation of the bits of this status byte as text.
  - **Manufacturer name**: This field shows the three ASCII letters from the two byte manufacturer ID from the fixed data structure at the beginning of a variable data frame of the meter. In our case KAM for KAMSTRUP.
  - **Poll pre delay 1**: This is a pause time in ms, before the gateway will send a primary or secondary address telegram to the meter to initiate the data readout process with this meter.
  - **Poll pre delay 2**: This is a pause time in ms, before the gateway will send a request for data telegram to the meter to readout more data from this meter.
  - **Poll post delay 1**: This is a pause time in ms, after the gateway will send a primary or secondary address telegram to the meter to initiate the data readout process with this meter.
  - **Poll post delay 2**: This is a pause time in ms, after the gateway will send a request for data telegram to the meter to readout more data from this meter.
  - **Poll repeats 1**: This is a repeat count, how often the MBUS gateway will send a primary or secondary address telegram to the meter, in the case the meter do not answer correctly.
  - **Poll repeats 2**: This is a repeat count, how often the MBUS gateway will send a request for data telegram to the meter, in the case the meter do not answer correctly.

All this setup parameters for the meter will be downloaded with the button Download configuration.

16.10.8.2 HOWTO set up individual poll parameters for one meter

In the basic setup of the gateway you will find the two parameters Query timeout and Poll timeout for general timing of the sequential process of requesting data from the connected meters. The two parameters can be configured like this:

- **Query timeout**: This field defines the timeout between two query cycles in the gateway. Usually the gateway communicates with all configured meters sequentially. After finishing the data readout for the last meter, the gateway pauses for this defined interval in seconds. This values are used:
  Value 65535 or values 0..5 defines ~5s pause.
  Values 6 to 65534: defines 6 to 65534 seconds of pause, before the next polling cycle will start.
- **Poll timeout**: This field defines a general pause after the readout of a configured meter before the readout of the next meter starts. In the past we discovered that there are many meters out in the market, which need a special treatment in the timing. e.g. very old KAMSTRUP meters need often two readout cycles with a gap of at least 10-15 seconds. This is non standard to the MBUS. Or other meters have problems with secondary addressing, if there is a too small gap between the readout. So we introduced this new parameter: This timeout defines the pause after finishing reading of a meter and starting reading the next meter. In the previous firmware versions this timeout was fixed to 250ms gap, which was ok for 99% of the meter readout on the markets. But some meter fail to process this little gap. The values is interpreted as follows:
  Value 1..30: Gap time 1 seconds to 30 seconds
  Value 101..400: Gaptime=(Value-100)*0.1s → 0.1s .. 30s e.g. 105 → 0.5s
  Value 65535: Gap time is 1 second
  Value 65534: Gap time is 250ms
  Value 65533: Gap time is 500ms
  Value 65532: Gap time is7250ms
  All other values: Gap time is 1000ms

Here you will find a basic diagram, how the MBUS master request cycle is handled by our gateways.

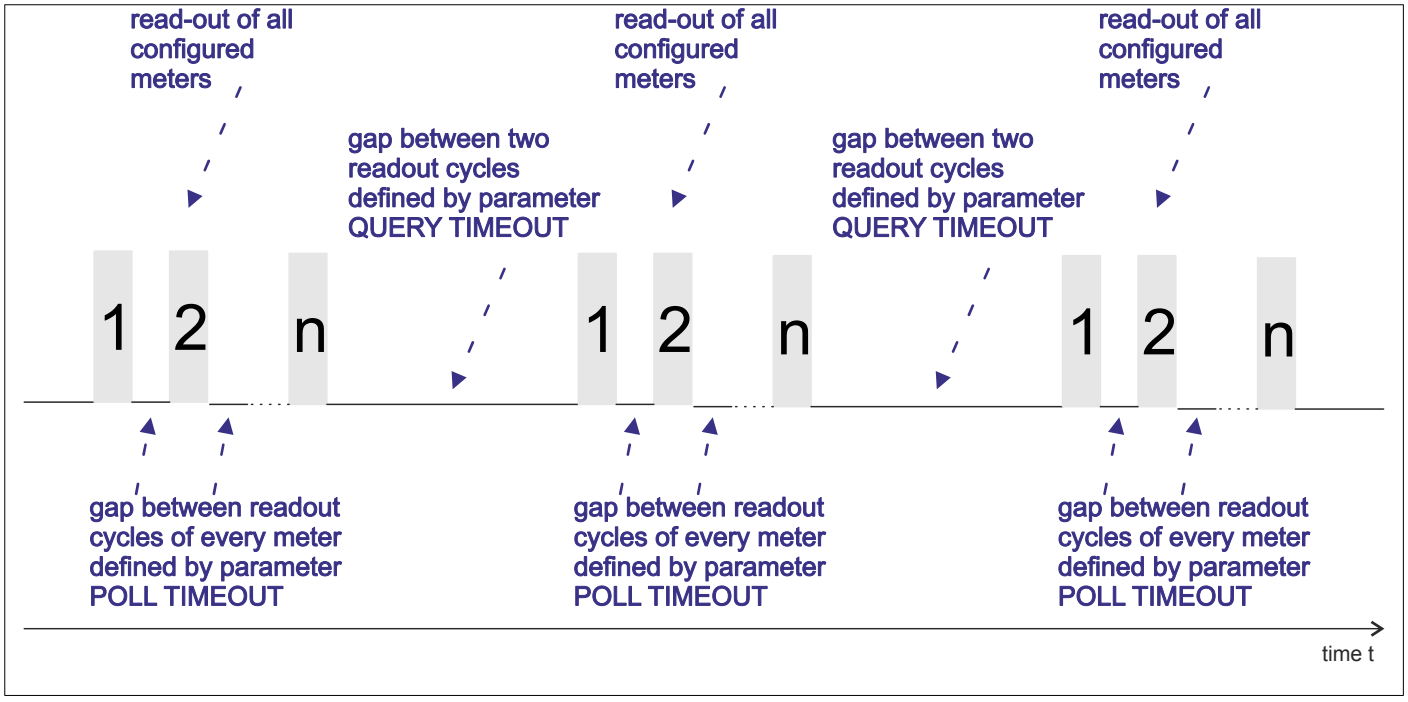


Figure: Basic timing of MBUS master read-out for MBUS slaves

Now we go more into detail, how the MBUS gateway will handle the request process of one meter. Forst we define the parameters:

- **Poll repeats 1**: This field defines the amount of telegram repetitions for the addressing command to a meter, before the gateway declares the communication as not possible and resumes with the next meter.
  Value 65535 or 0: use 3 repeats as standard
  Value 1..n: Use n repeats
- **Poll repeats 2**: This field defines the amount of telegram repetitions for the data readout command to a meter, before the gateway declares the communication as not possible and resumes with the next meter.
  Value 65535 or 0: use 5 repeats as standard
  Value 1..n: Use n repeats
- **Poll pre delay 1**: This field defines the first pause time in Milliseconds before starting to send the first addressing command telegram to a meter.
  Value 65535: use 250ms as standard pause time
  Value 0..65534: Use x ms as pause time
- **Poll pre delay 2**: This field defines the first pause time in Milliseconds before starting to send the first data request telegram to a meter.
  Value 65535: use 100ms as standard pause time
  Value 0..65534: Use x ms as pause time
- **Poll post delay 1**: This field defines a pause time in Milliseconds. If the gateway do not receive a correct answer to an addressing command telegram and the addressing command is repeated, then this pause time is inserted, before resending the addressing telegram to the meter.
  Value 65535: use 0ms as standard pause time
  Value 0..65534: Use x ms as pause time
- **Poll post delay 2**: This field defines a pause time in Milliseconds. If the gateway do not receive a correct answer to a readout data telegram and the readout data command is repeated, then this pause time is inserted, before resending the readout data telegram to the meter.
  Value 65535: use 100ms as standard pause time
  Value 0..65534: Use x ms as pause time

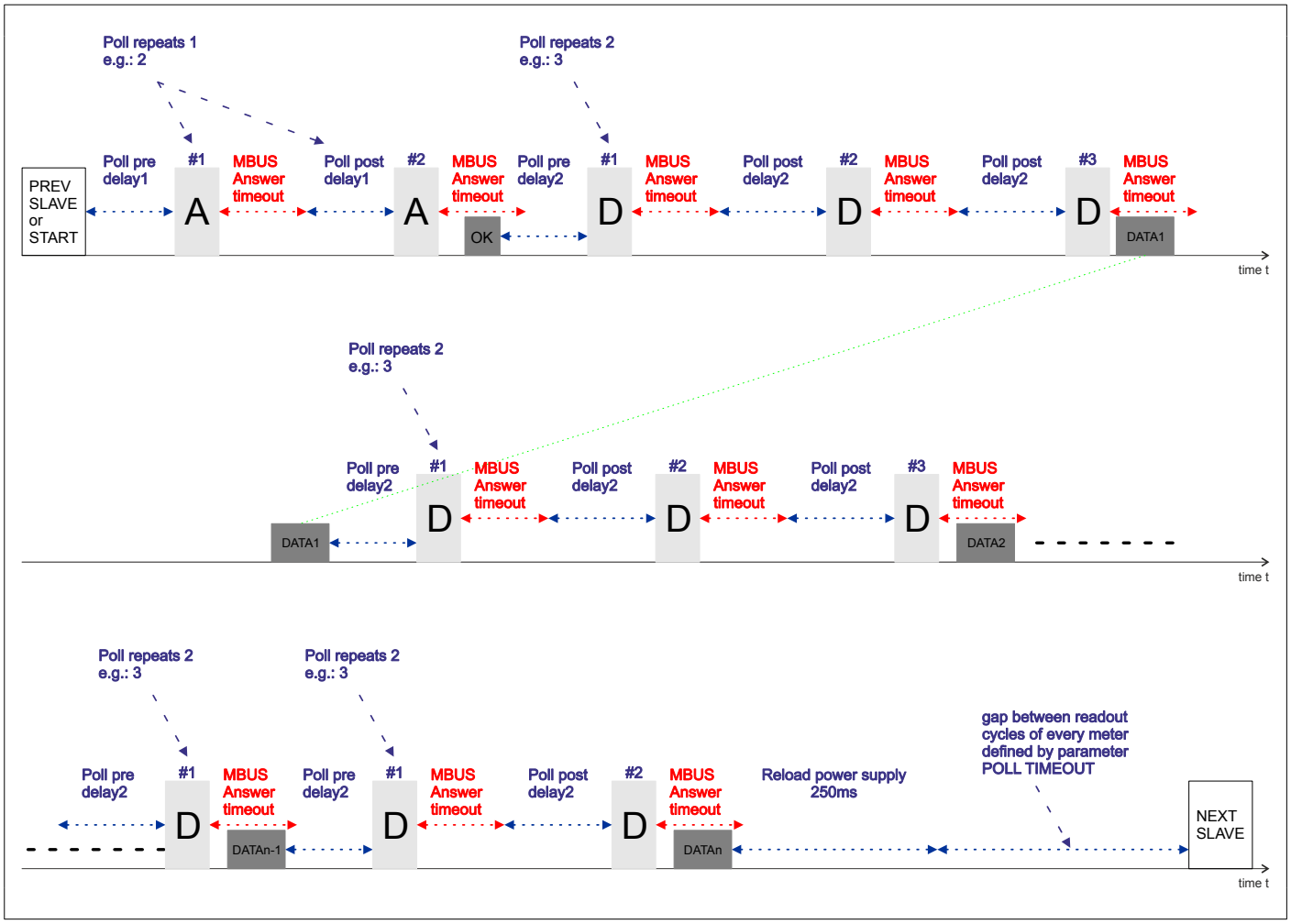
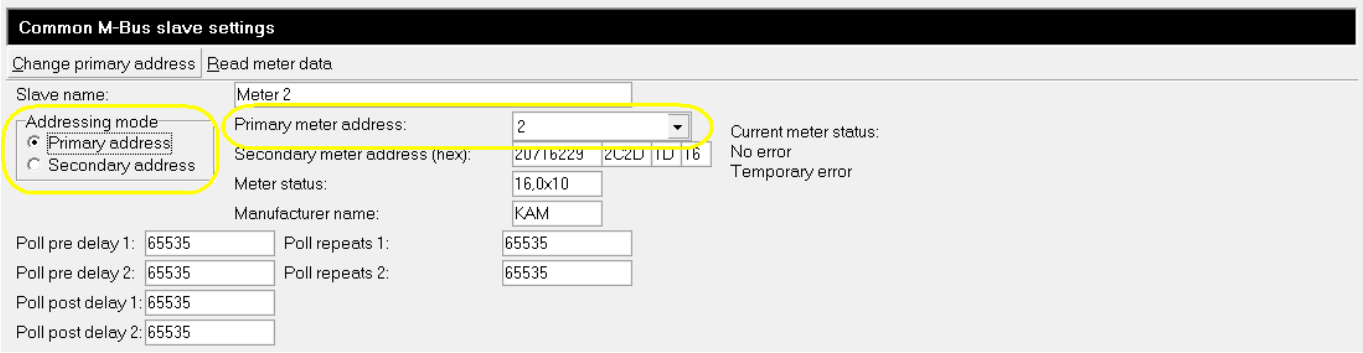


Figure: Basic timing of MBUS master read-out for MBUS slaves

16.10.8.3 HOWTO select primary addressing mode

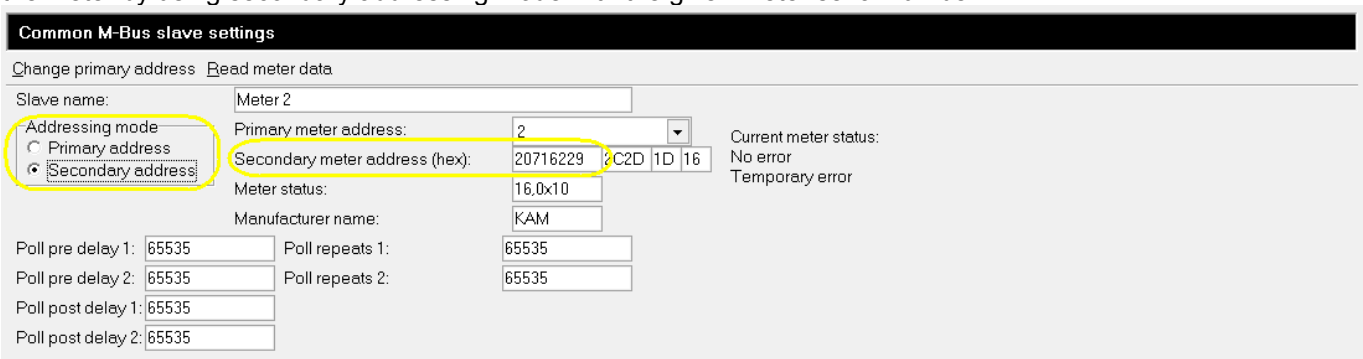
To select primary addressing mode, you have to define a primary address for the meter in the range from 1 to 151. and you have to select in the Addressing mode radio button the mode Primary address. After you have successfully downloaded the configuration into the gateway, this meter will be addressed by the meter by using primary addressing mode with the given primary address.

Don't forget, that the meter will only answer to the request, if the meter is programmed for the defined primary address, the meter uses the same MBUS baud rate and there is not another meter on the MBUS with the same primary address.



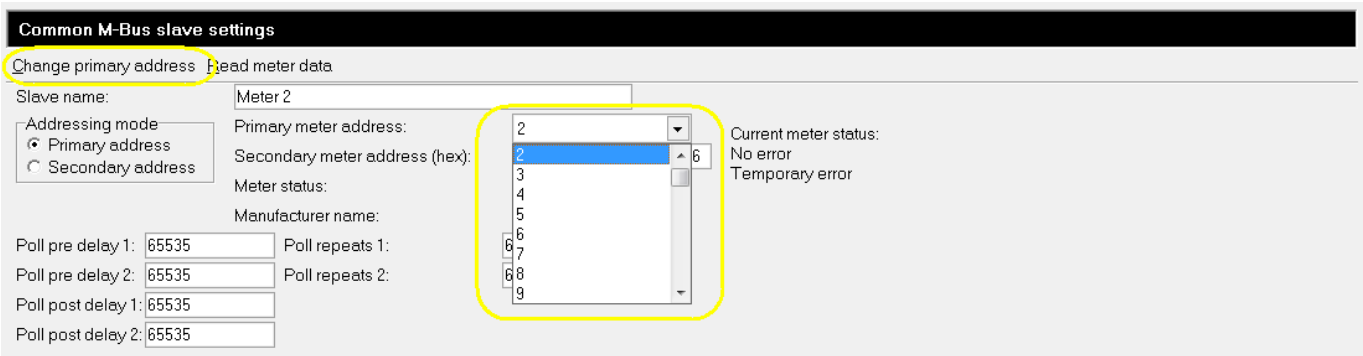
16.10.8.4 HOWTO select secondary addressing mode

To select secondary addressing mode, you have to define the unique meter ID (serial number) for the meter in the field Secondary meter address (hex). Then you have to select in the Addressing mode radio button the mode Secondary address. After you have successfully downloaded the configuration into the gateway, this meter will be addressed by the meter by using secondary addressing mode with the given Meter serial number.



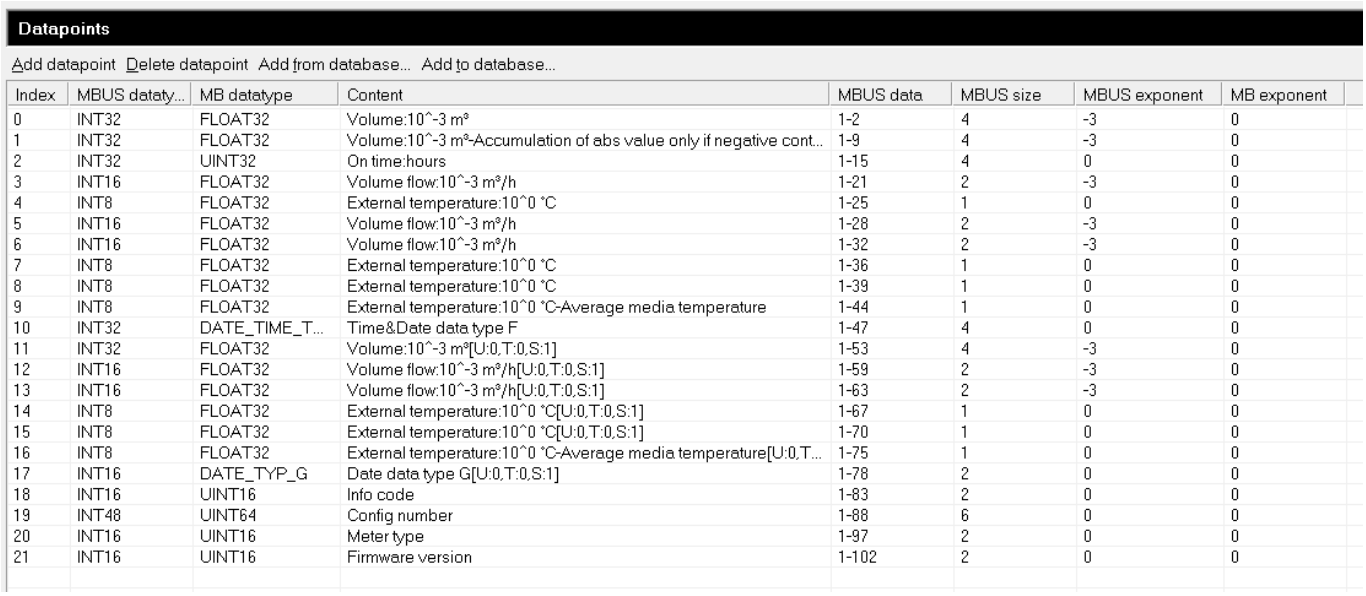
16.10.8.5 HOWTO change the primary MBUS address in meter

When you want to change the primary address of the meter, first you have to select a new primary address form the drop down list Primary meter address. Take a unique address between 1 and 251 from the list and make sure, that you don't have another meter on the network with the same address you want to use in the future. Then click on the button Change primary address. Don't forget to change the Slave name. The standard is, that the slave name contains the primary address ant the end of the name.



16.10.8.6 WHAT is displayed in the Datapoints data grid

In this area you will find the following information:



Datapoints

Add datapoint Delete datapoint Add from database... Add to database...

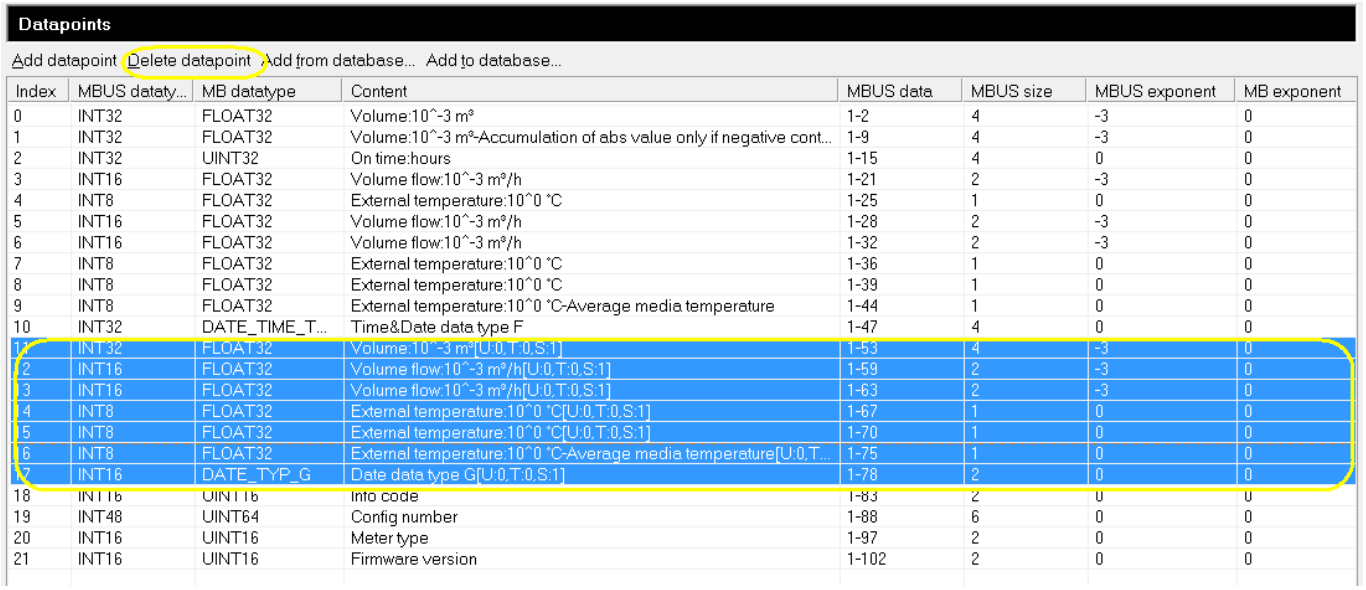
Index	MBUS dataty...	MB datatype	Content	MBUS data	MBUS size	MBUS exponent	MB exponent
0	INT32	FLOAT32	Volume:10^-3 m³	1-2	4	-3	0
1	INT32	FLOAT32	Volume:10^-3 m³-Accumulation of abs value only if negative cont...	1-9	4	-3	0
2	INT32	UINT32	On time:hours	1-15	4	0	0
3	INT16	FLOAT32	Volume flow:10^-3 m³/h	1-21	2	-3	0
4	INT8	FLOAT32	External temperature:10^0 °C	1-25	1	0	0
5	INT16	FLOAT32	Volume flow:10^-3 m³/h	1-28	2	-3	0
6	INT16	FLOAT32	Volume flow:10^-3 m³/h	1-32	2	-3	0
7	INT8	FLOAT32	External temperature:10^0 °C	1-36	1	0	0
8	INT8	FLOAT32	External temperature:10^0 °C	1-39	1	0	0
9	INT8	FLOAT32	External temperature:10^0 °C-Average media temperature	1-44	1	0	0
10	INT32	DATE_TIME_T...	Time&Date data type F	1-47	4	0	0
11	INT32	FLOAT32	Volume:10^-3 m³[U:0,T:0,S:1]	1-53	4	-3	0
12	INT16	FLOAT32	Volume flow:10^-3 m³/h[U:0,T:0,S:1]	1-59	2	-3	0
13	INT16	FLOAT32	Volume flow:10^-3 m³/h[U:0,T:0,S:1]	1-63	2	-3	0
14	INT8	FLOAT32	External temperature:10^0 °C[U:0,T:0,S:1]	1-67	1	0	0
15	INT8	FLOAT32	External temperature:10^0 °C[U:0,T:0,S:1]	1-70	1	0	0
16	INT8	FLOAT32	External temperature:10^0 °C-Average media temperature[U:0,T...	1-75	1	0	0
17	INT16	DATE_TYP_G	Date data type G[U:0,T:0,S:1]	1-78	2	0	0
18	INT16	UINT16	Info code	1-83	2	0	0
19	INT48	UINT64	Config number	1-88	6	0	0
20	INT16	UINT16	Meter type	1-97	2	0	0
21	INT16	UINT16	Firmware version	1-102	2	0	0

In this grid you will find all datapoints regarding the selected meter. The grid has the following columns:

- **Index**: This is a running index starting with 0 to see how many datapoints you have defined. This is important, because the amount of datapoint mappings between MBUS and MODBUS is limited like the amount of MODBUS registers. e.g. The RESI-MBUS64-SIO can handle 1200 MODBUS registers but only 600 MBUS datapoints in total.
- **MBUS datatype**: Here you will see the used data type in the MBUS frame.
- **MB datatype**: Here you will find the MODBUS data type to map the MBUS data type to MODBUS register.
- **Content**: here you will see the name of the datapoint. This name will be build automatically with the additional information in the MBUS data (DIF+VIF fields). But it can be changed manually to user data.
- **MBUS data**: Here you can see in which record and on which offset within this record the MBUS data was found. The writing is <record>-<offset> in Bytes. This describes the location in the variable data structure of the MBUS data frame.
- **MBUS size**: This column shows the current size of the MBUS data in bytes.
- **MBUS exponent**: This column shows the exponent of the MBUS value, how it is defined in the MBUS data due to the DIF+VIF fields.
- **MB exponent**: This column shows the user defined exponent to shift the value in MODBUS registers.

16.10.8.7 HOWTO delete datapoints for a meter configuration

Since every MBUS datapoint needs mapping space in the MODBUS registers and the MODBUS registers are limited in the gateway, it makes sense to configure only those datapoints, which are necessary for your application. IN our example we don't want to read the storage values defined by storage number S:1. So we select all lines with this items (Use the pressed Control key and the mouse to do a multiselect on the grid), and the we delet the selected datapoints from the list by pressing the button Delete datapoint.

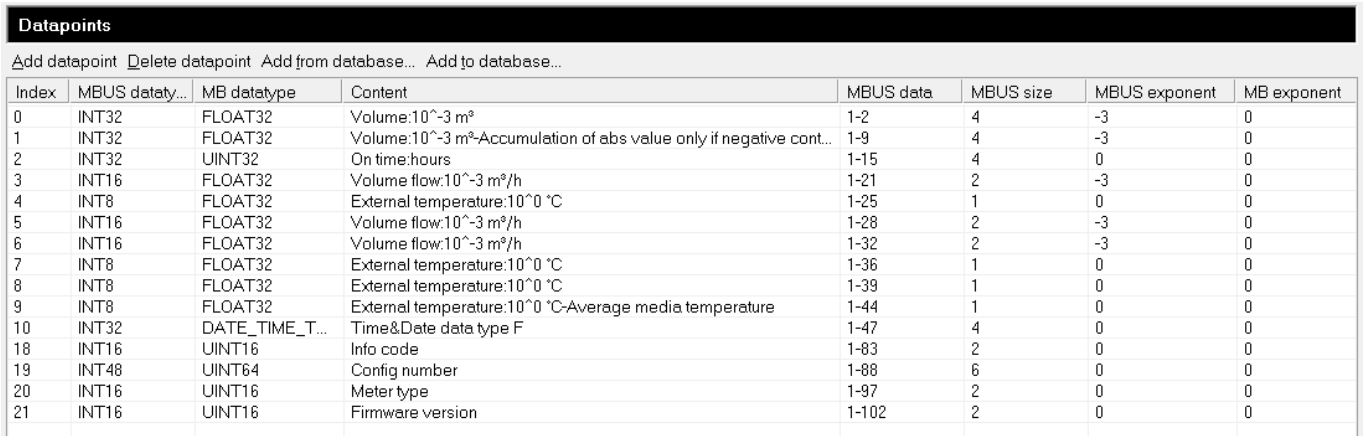


Datapoints

Add datapoint | Delete datapoint | Add from database... | Add to database...

Index	MBUS dataty...	MB datatype	Content	MBUS data	MBUS size	MBUS exponent	MB exponent
0	INT32	FLOAT32	Volume:10^-3 m³	1-2	4	-3	0
1	INT32	FLOAT32	Volume:10^-3 m³-Accumulation of abs value only if negative cont...	1-9	4	-3	0
2	INT32	UINT32	On time:hours	1-15	4	0	0
3	INT16	FLOAT32	Volume flow:10^-3 m³/h	1-21	2	-3	0
4	INT8	FLOAT32	External temperature:10^0 °C	1-25	1	0	0
5	INT16	FLOAT32	Volume flow:10^-3 m³/h	1-28	2	-3	0
6	INT16	FLOAT32	Volume flow:10^-3 m³/h	1-32	2	-3	0
7	INT8	FLOAT32	External temperature:10^0 °C	1-36	1	0	0
8	INT8	FLOAT32	External temperature:10^0 °C	1-39	1	0	0
9	INT8	FLOAT32	External temperature:10^0 °C-Average media temperature	1-44	1	0	0
10	INT32	DATE_TIME_T...	Time&Date data type F	1-47	4	0	0
11	INT32	FLOAT32	Volume:10^-3 m³[U:0,T:0,S:1]	1-53	4	-3	0
12	INT16	FLOAT32	Volume flow:10^-3 m³/h[U:0,T:0,S:1]	1-59	2	-3	0
13	INT16	FLOAT32	Volume flow:10^-3 m³/h[U:0,T:0,S:1]	1-63	2	-3	0
14	INT8	FLOAT32	External temperature:10^0 °C[U:0,T:0,S:1]	1-67	1	0	0
15	INT8	FLOAT32	External temperature:10^0 °C[U:0,T:0,S:1]	1-70	1	0	0
16	INT8	FLOAT32	External temperature:10^0 °C-Average media temperature[U:0,T...	1-75	1	0	0
17	INT16	DATE_TYP_G	Date data type G[U:0,T:0,S:1]	1-78	2	0	0
18	INT16	UINT16	Info code	1-83	2	0	0
19	INT48	UINT64	Config number	1-88	6	0	0
20	INT16	UINT16	Meter type	1-97	2	0	0
21	INT16	UINT16	Firmware version	1-102	2	0	0

Your new list will look like this. If you download this configuration, only the desired datapoints are mapped to the MODBUS registers. The gateway requests only as much MBUS frames as necessary for mapping all values to the MODBUS registers.



Datapoints

Add datapoint | Delete datapoint | Add from database... | Add to database...

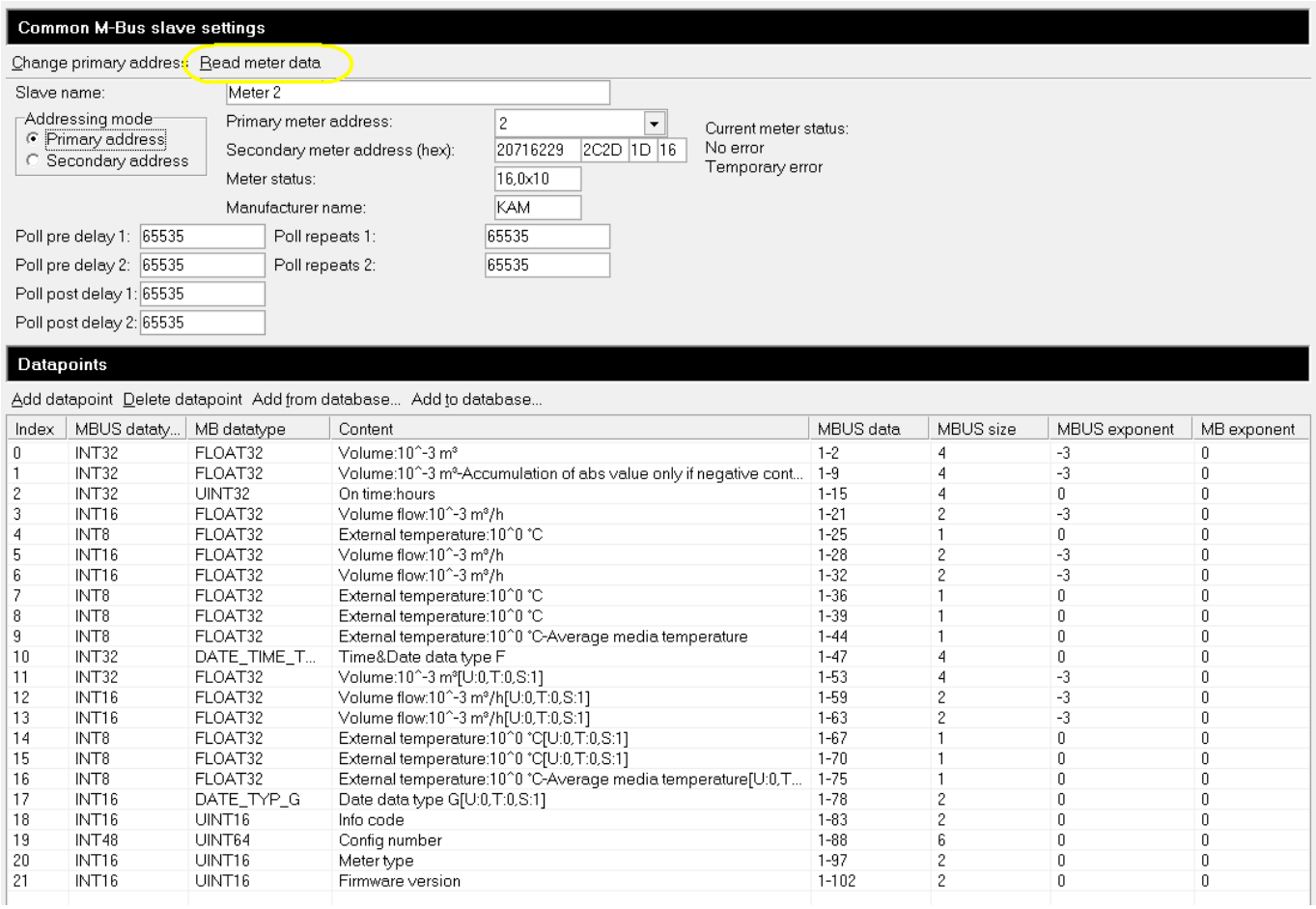
Index	MBUS dataty...	MB datatype	Content	MBUS data	MBUS size	MBUS exponent	MB exponent
0	INT32	FLOAT32	Volume:10^-3 m³	1-2	4	-3	0
1	INT32	FLOAT32	Volume:10^-3 m³-Accumulation of abs value only if negative cont...	1-9	4	-3	0
2	INT32	UINT32	On time:hours	1-15	4	0	0
3	INT16	FLOAT32	Volume flow:10^-3 m³/h	1-21	2	-3	0
4	INT8	FLOAT32	External temperature:10^0 °C	1-25	1	0	0
5	INT16	FLOAT32	Volume flow:10^-3 m³/h	1-28	2	-3	0
6	INT16	FLOAT32	Volume flow:10^-3 m³/h	1-32	2	-3	0
7	INT8	FLOAT32	External temperature:10^0 °C	1-36	1	0	0
8	INT8	FLOAT32	External temperature:10^0 °C	1-39	1	0	0
9	INT8	FLOAT32	External temperature:10^0 °C-Average media temperature	1-44	1	0	0
10	INT32	DATE_TIME_T...	Time&Date data type F	1-47	4	0	0
18	INT16	UINT16	Info code	1-83	2	0	0
19	INT48	UINT64	Config number	1-88	6	0	0
20	INT16	UINT16	Meter type	1-97	2	0	0
21	INT16	UINT16	Firmware version	1-102	2	0	0

But be aware, that you have changed your MODBUS register list also with this action:

MB Register	MBUS datatype	MB datatype	Content	MBUS index	MB value HEX	Cu
4x00001	INT32[4]	FLOAT32	Volume:10^-3 m³	0	???	??
4x00003	INT32[4]	FLOAT32	Volume:10^-3 m³-Accumulation of abs value only if negative contribu	1	???	??
4x00005	INT32[4]	UINT32	On time:hours	2	???	??
4x00007	INT16[2]	FLOAT32	Volume flow:10^-3 m³/h	3	???	??
4x00009	INT8[1]	FLOAT32	External temperature:10^0 °C	4	???	??
4x00011	INT16[2]	FLOAT32	Volume flow:10^-3 m³/h	5	???	??
4x00013	INT16[2]	FLOAT32	Volume flow:10^-3 m³/h	6	???	??
4x00015	INT8[1]	FLOAT32	External temperature:10^0 °C	7	???	??
4x00017	INT8[1]	FLOAT32	External temperature:10^0 °C	8	???	??
4x00019	INT8[1]	FLOAT32	External temperature:10^0 °C-Average media temperature	9	???	??
4x00021	INT32[4]	DATE_TIME_T	Time&Date data type F	10	???	??
4x00023	INT16[2]	UINT16	Info code	18	???	??
4x00024	INT48[6]	UINT64	Config number	19	???	??
4x00028	INT16[2]	UINT16	Meter type	20	???	??
4x00029	INT16[2]	UINT16	Firmware version	21	???	??
4x00030	VAR LENGTH[18]	ASCII	Manufacturer	0	???	??
4x00040	VAR LENGTH[8]	ASCII	Model/version	1	???	??
4x00045	VAR LENGTH[7]	ASCII	Firmware version	2	???	??
4x00049	INT24[3]	UINT32	Error flags (binary)	3	???	??
4x00051	FLOAT32[4]	FLOAT32	Current 10^0A-L1 phase value	4	???	??
4x00053	FLOAT32[4]	FLOAT32	Current 10^0A-L2 phase value	5	???	??
4x00055	FLOAT32[4]	FLOAT32	Current 10^0A-L3 phase value	6	???	??
4x00057	FLOAT32[4]	FLOAT32	Current 10^0A-Average current	7	???	??
4x00059	FLOAT32[4]	FLOAT32	Voltage 10^0V-L1-L2	8	???	??
4x00061	FLOAT32[4]	FLOAT32	Voltage 10^0V-L2-L3	9	???	??
4x00063	FLOAT32[4]	FLOAT32	Voltage 10^0V-L3-L1	10	???	??
4x00065	FLOAT32[4]	FLOAT32	Voltage 10^0V-Voltage L-L average	11	???	??
4x00067	FLOAT32[4]	FLOAT32	Voltage 10^0V-L1 phase value	12	???	??
4x00069	FLOAT32[4]	FLOAT32	Voltage 10^0V-L2 phase value	13	???	??
4x00071	FLOAT32[4]	FLOAT32	Voltage 10^0V-L3 phase value	14	???	??

16.10.8.8 HOWTO refresh datapoints for a meter configuration

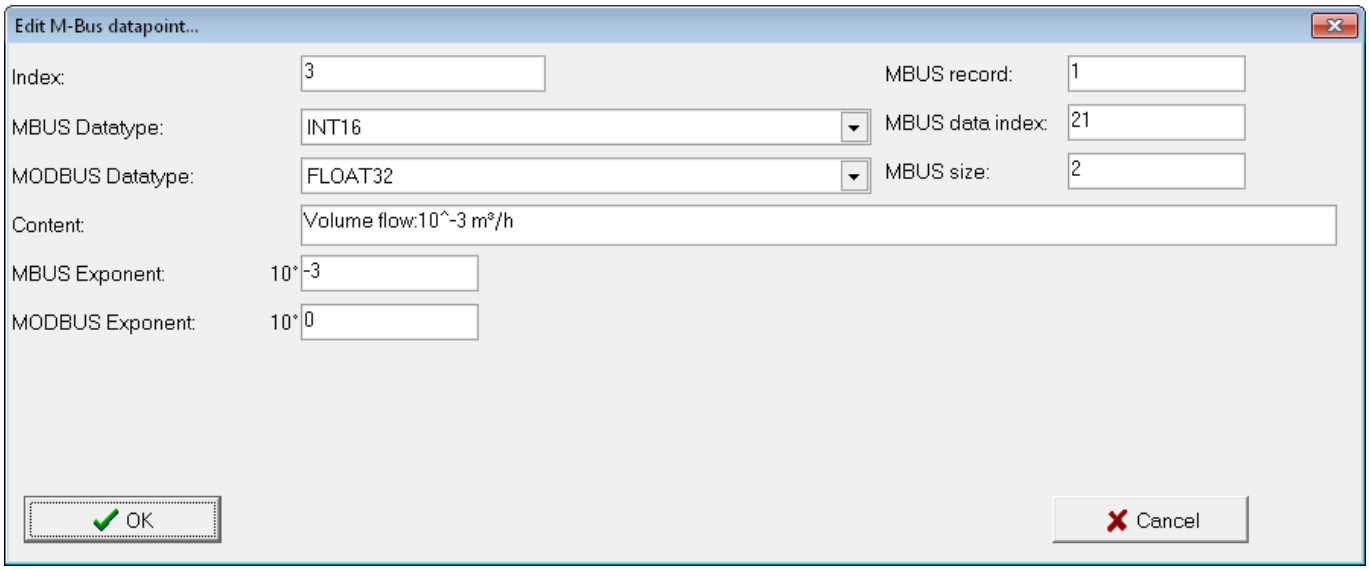
So if you have deleted some datapoints for one meter and you want to restore the original mapping from the meter, you can simple press the button "Read meter data". It will scan all MBUS datapoints of the selected meter again and refresh the list:



Index	MBUS dataty...	MB datatype	Content	MBUS data	MBUS size	MBUS exponent	MB exponent
0	INT32	FLOAT32	Volume:10^-3 m³	1-2	4	-3	0
1	INT32	FLOAT32	Volume:10^-3 m³-Accumulation of abs value only if negative cont...	1-9	4	-3	0
2	INT32	UINT32	On time:hours	1-15	4	0	0
3	INT16	FLOAT32	Volume flow:10^-3 m³/h	1-21	2	-3	0
4	INT8	FLOAT32	External temperature:10^0 °C	1-25	1	0	0
5	INT16	FLOAT32	Volume flow:10^-3 m³/h	1-28	2	-3	0
6	INT16	FLOAT32	Volume flow:10^-3 m³/h	1-32	2	-3	0
7	INT8	FLOAT32	External temperature:10^0 °C	1-36	1	0	0
8	INT8	FLOAT32	External temperature:10^0 °C	1-39	1	0	0
9	INT8	FLOAT32	External temperature:10^0 °C-Average media temperature	1-44	1	0	0
10	INT32	DATE_TIME_T...	Time&Date data type F	1-47	4	0	0
11	INT32	FLOAT32	Volume:10^-3 m³[U:0,T:0,S:1]	1-53	4	-3	0
12	INT16	FLOAT32	Volume flow:10^-3 m³/h[U:0,T:0,S:1]	1-59	2	-3	0
13	INT16	FLOAT32	Volume flow:10^-3 m³/h[U:0,T:0,S:1]	1-63	2	-3	0
14	INT8	FLOAT32	External temperature:10^0 °C[U:0,T:0,S:1]	1-67	1	0	0
15	INT8	FLOAT32	External temperature:10^0 °C[U:0,T:0,S:1]	1-70	1	0	0
16	INT8	FLOAT32	External temperature:10^0 °C-Average media temperature[U:0,T...	1-75	1	0	0
17	INT16	DATE_TYP_G	Date data type G[U:0,T:0,S:1]	1-78	2	0	0
18	INT16	UINT16	Info code	1-83	2	0	0
19	INT48	UINT64	Config number	1-88	6	0	0
20	INT16	UINT16	Meter type	1-97	2	0	0
21	INT16	UINT16	Firmware version	1-102	2	0	0

16.10.8.9 HOWTO modify MBUS datapoint mapping manually

The MODBUSConfigurator software will try to map the MBUS data types automatically to correct MODBUS data types and MODBUS registers. But you can also modify this mapping. Double click onto an item in the data grid, you will see the following dialog:

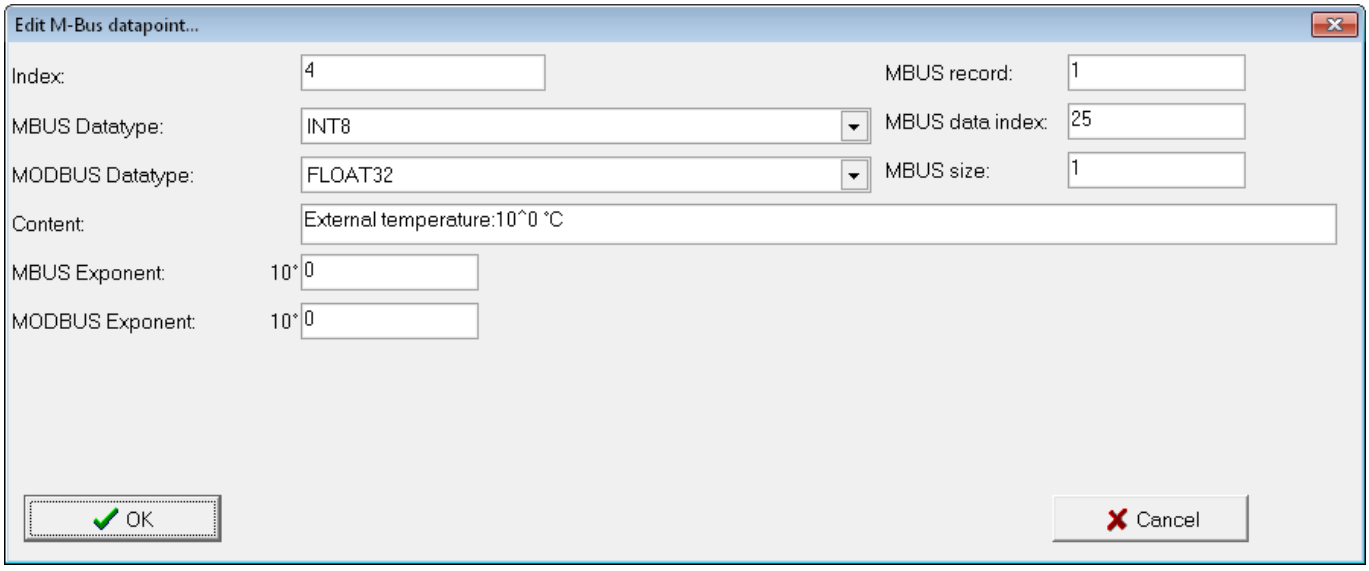


Basically it is the data grid line in an editable version. You can change the content description here. Or you can change the MODBUS data type here. If you really add MBUS data frames manually you can also edit the MBUS data type, the MBUS exponent, the MBUS record number, the MBUS data index and the MBUS size in here to define the exact location of the MBUS data frame.

Usually you will change the MODBUS exponent and or the MODBUS data type. Lets do a sample configuration change:

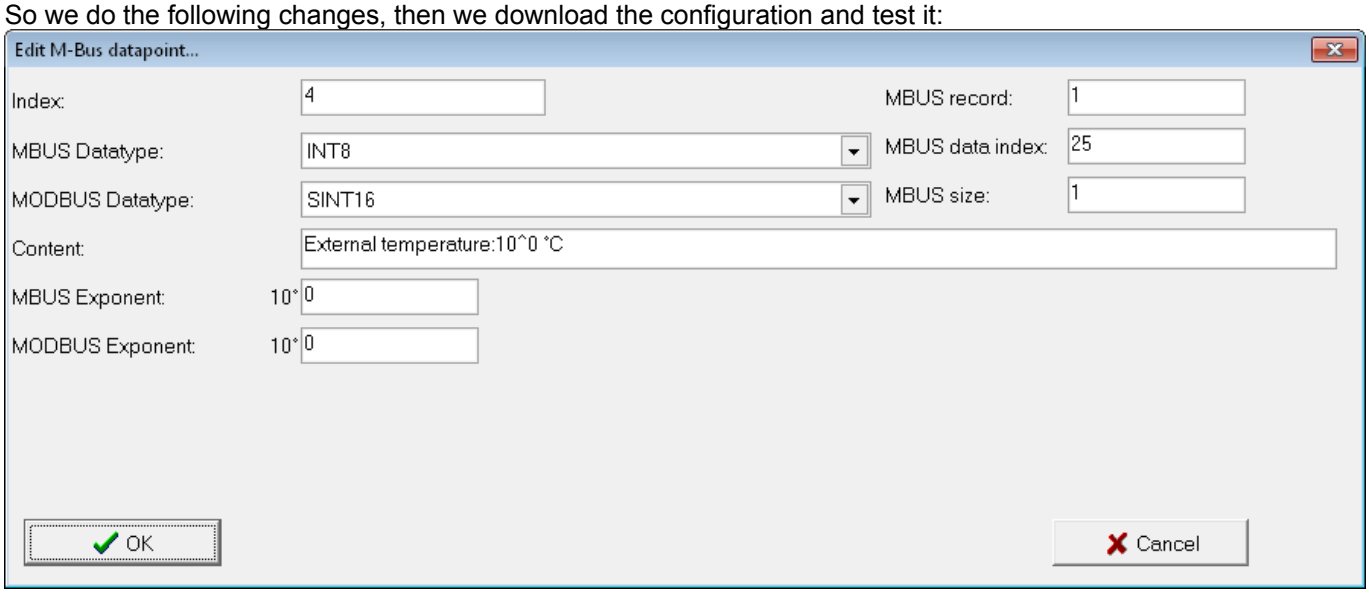
MB Register	MBUS datatype	MB datatype	Content	MBUS index	MB value HEX	Current MB value	Meter name
4x00001	INT32[4]	FLOAT32	Volume:10^-3 m³	0	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 2 [P:2]
4x00003	INT32[4]	FLOAT32	Volume:10^-3 m³-Accumulation of abs value only if negative contribu	1	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 2 [P:2]
4x00005	INT32[4]	UINT32	On time:hours	2	MSW:0000,1183:LSW	4483,0x00001183	Meter 2 [P:2]
4x00007	INT16[2]	FLOAT32	Volume flow:10^-3 m³/h	3	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 2 [P:2]
4x00009	INT8[1]	FLOAT32	External temperature:10^0 °C	4	MSW:41E0,0000:LSW	28.0000,2.80000000000000E+1	Meter 2 [P:2]
4x00011	INT16[2]	FLOAT32	Volume flow:10^-3 m³/h	5	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 2 [P:2]
4x00013	INT16[2]	FLOAT32	Volume flow:10^-3 m³/h	6	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 2 [P:2]
4x00015	INT8[1]	FLOAT32	External temperature:10^0 °C	7	MSW:41B8,0000:LSW	23.0000,2.30000000000000E+1	Meter 2 [P:2]

As you can see from the live data, the external temperature is currently 28°C. Our automatic mapping algorithm maps the MBUS data type INT8 (8 bit SIGNED INTEGER) to a FLOAT32 using two consecutive MODBUS registers, because we try to show on the MODBUS side the correct value with the correct exponent. But in this special case a standard Holding register will be enough.



So we change the configuration from FLOAT32 to SINT16 to map the value into a single holding register. This saves register space and it also increases the conversion accuracy to 100%, because INT8 to SINT16 is as loss free conversation in comparison to INT8 to FLOAT32 is not a loss free conversation, because the FLOAT32 format is too inaccurate to show in all cases the real INT8 value.

So we do the following changes, then we download the configuration and test it:

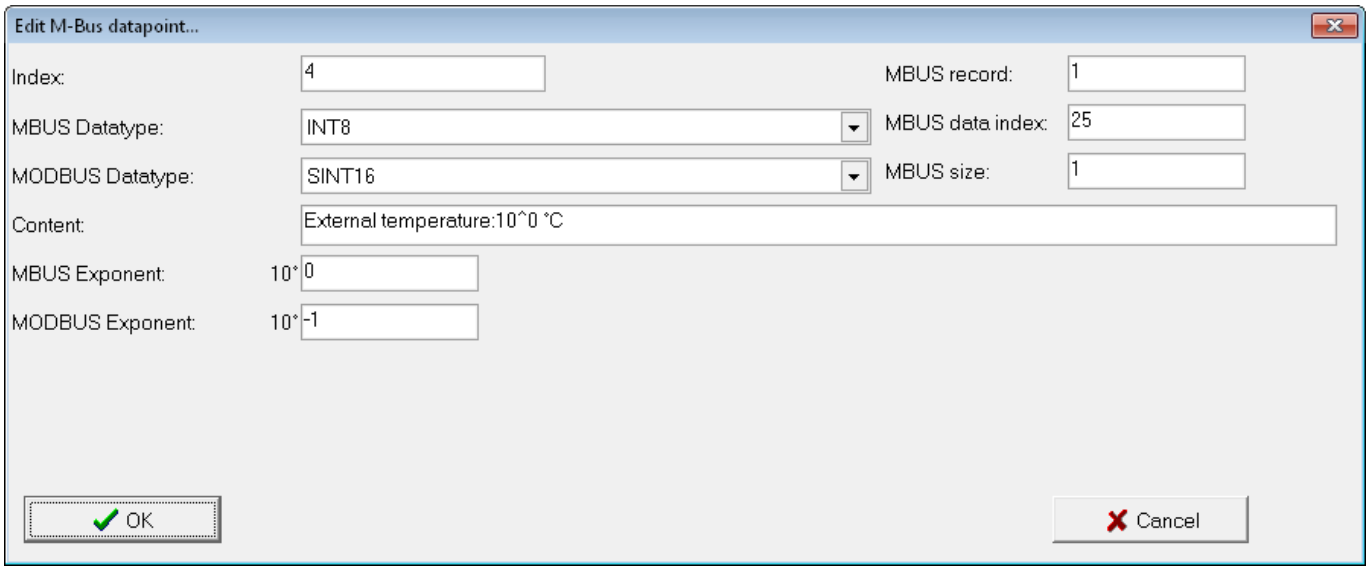


The result will be like this:

MB Register	MBUS datatype	MB datatype	Content	MBUS index	MB value HEX	Current MB value	Meter name
4x00001	INT32[4]	FLOAT32	Volume:10^-3 m³	0	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 2 [P:2]
4x00003	INT32[4]	FLOAT32	Volume:10^-3 m³-Accumulation of abs value only if negative contribu	1	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 2 [P:2]
4x00005	INT32[4]	UINT32	On time:hours	2	MSW:0000,1183:LSW	4483,0x00001183	Meter 2 [P:2]
4x00007	INT16[2]	FLOAT32	Volume flow:10^-3 m³/h	3	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 2 [P:2]
4x00009	INT8[1]	SINT16	External temperature:10^0 °C	4	WORD:0018	28,0x001C	Meter 2 [P:2]
4x00010	INT16[2]	FLOAT32	Volume flow:10^-3 m³/h	5	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 2 [P:2]
4x00012	INT16[2]	FLOAT32	Volume flow:10^-3 m³/h	6	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 2 [P:2]
4x00014	INT8[1]	FLOAT32	External temperature:10^0 °C	7	MSW:41B8,0000:LSW	23.0000,2.30000000000000E+1	Meter 2 [P:2]

Please note also, that the next MBUS datapoint starts not longer in the register 4x00011, It starts now in the register 4x00010. So we saved really one register.

Now we define, that your host can handle only temperatures with one comma. This means the 28°C should be stored as 280 in the holding register. For that we change the MODBUS exponent field to -1 to shift the result by 10:



Now we get this result in test mode:

MB Register	MBUS datatype	MB datatype	Content	MBUS index	MB value HEX	Current MB value	Meter name
4x00001	INT32[4]	FLOAT32	Volume:10^-3 m³	0	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 2 [P:2]
4x00003	INT32[4]	FLOAT32	Volume:10^-3 m³-Accumulation of abs value only if negative contribu	1	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 2 [P:2]
4x00005	INT32[4]	UINT32	On time:hours	2	MSW:0000,1184:LSW	4484,0x00001184	Meter 2 [P:2]
4x00007	INT16[2]	FLOAT32	Volume flow:10^-3 m³/h	3	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 2 [P:2]
4x00009	INT8[1]	SINT16	External temperature:10^0 °C->*10^-1	4	WORD:0018	280,0x0118	Meter 2 [P:2]
4x00010	INT16[2]	FLOAT32	Volume flow:10^-3 m³/h	5	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 2 [P:2]
4x00012	INT16[2]	FLOAT32	Volume flow:10^-3 m³/h	6	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 2 [P:2]
4x00014	INT8[1]	FLOAT32	External temperature:10^0 °C	7	MSW:41B8,0000:LSW	23.0000,2.30000000000000E+1	Meter 2 [P:2]

16.11 HOWTO save datapoints to user specific meter database

Our software offers the possibility to save a current meter setup to a user database for future use. Therefore select the desired meter in the project tree and click on the button Add to database...

Common M-Bus slave settings

Change primary address Read meter data

Slave name: Meter 06364163_4CA3_18_02

Addressing mode
- Primary address
- Secondary address

Primary meter address: 253

Secondary meter address (hex): 06364163 4CA3 18 02

Meter status: 0,0x00

Manufacturer name: SEC

Current meter status: No error

Poll pre delay 1: 65535 Poll repeats 1: 65535

Poll pre delay 2: 65535 Poll repeats 2: 65535

Poll post delay 1: 65535

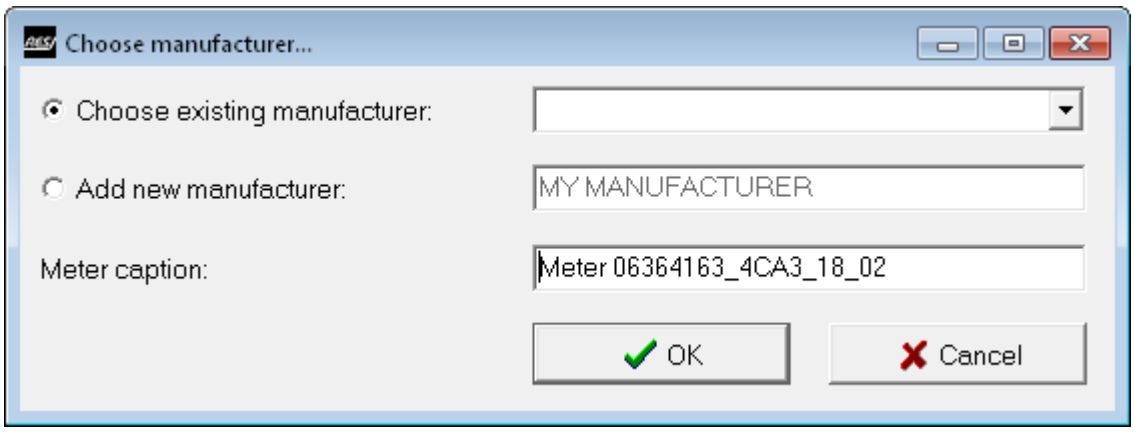
Poll post delay 2: 65535

Datapoints

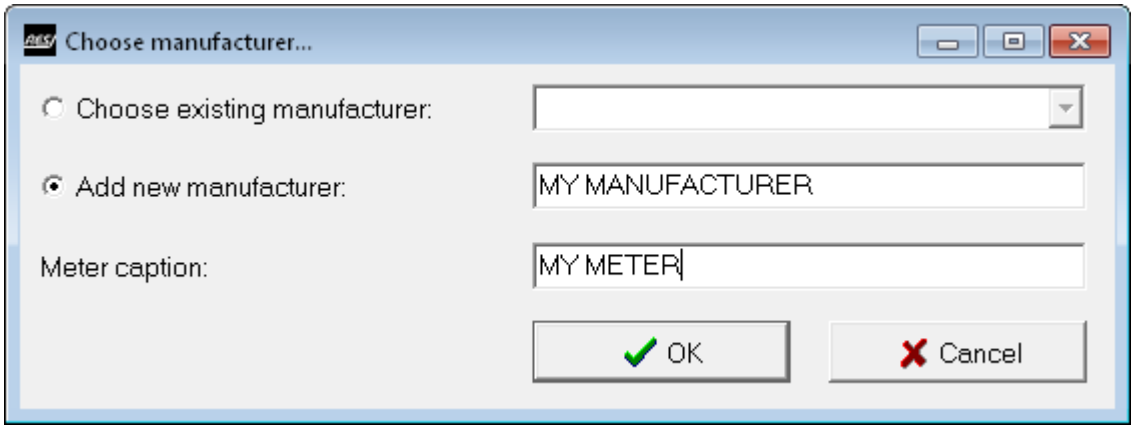
Add datapoint Delete datapoint Add from database... Add to database...

Index	MBUS dataty...	MB datatype	Content	MBUS data	MBUS size	MBUS exponent	MB exponent
0	LVAR:ASCII	ASCII	Manufacturer	1-4	18	0	0
1	LVAR:ASCII	ASCII	Model/version	1-26	8	0	0
2	LVAR:ASCII	ASCII	Firmware version	1-38	7	0	0
3	INT24	UINT32	Error flags (binary)	1-48	3	0	0
4	FLOAT32	FLOAT32	Current 10^0A-L1 phase value	1-56	4	0	0
5	FLOAT32	FLOAT32	Current 10^0A-L2 phase value	1-65	4	0	0
6	FLOAT32	FLOAT32	Current 10^0A-L3 phase value	1-74	4	0	0
7	FLOAT32	FLOAT32	Current 10^0A-Average current	1-83	4	0	0
8	FLOAT32	FLOAT32	Voltage 10^0V-L1-L2	1-92	4	0	0
9	FLOAT32	FLOAT32	Voltage 10^0V-L2-L3	1-101	4	0	0
10	FLOAT32	FLOAT32	Voltage 10^0V-L3-L1	1-110	4	0	0
11	FLOAT32	FLOAT32	Voltage 10^0V-Voltage L-L average	1-119	4	0	0
12	FLOAT32	FLOAT32	Voltage 10^0V-L1 phase value	1-128	4	0	0
13	FLOAT32	FLOAT32	Voltage 10^0V-L2 phase value	1-137	4	0	0
14	FLOAT32	FLOAT32	Voltage 10^0V-L3 phase value	1-146	4	0	0
15	FLOAT32	FLOAT32	Voltage 10^0V-L-N average	1-155	4	0	0
16	FLOAT32	FLOAT32	Power:10^3 W-L1 phase value	1-163	4	3	0
17	FLOAT32	FLOAT32	Power:10^3 W-L2 phase value	1-171	4	3	0
18	FLOAT32	FLOAT32	Power:10^3 W-L3 phase value	1-179	4	3	0
19	FLOAT32	FLOAT32	Power:10^3 W	1-185	4	3	0
20	FLOAT32	FLOAT32	Power:10^3 W[U:1,T:0,S:0]	1-192	4	3	0
21	FLOAT32	FLOAT32	Power:10^3 W[U:2,T:0,S:0]	1-200	4	3	0
22	FLOAT32	FLOAT32	Power Factor	1-207	4	0	0
23	FLOAT32	FLOAT32	Frequency	1-214	4	0	0
24	INT64	UINT64	Energy:10^0 Wh	1-220	8	0	0

You will see the following dialog:

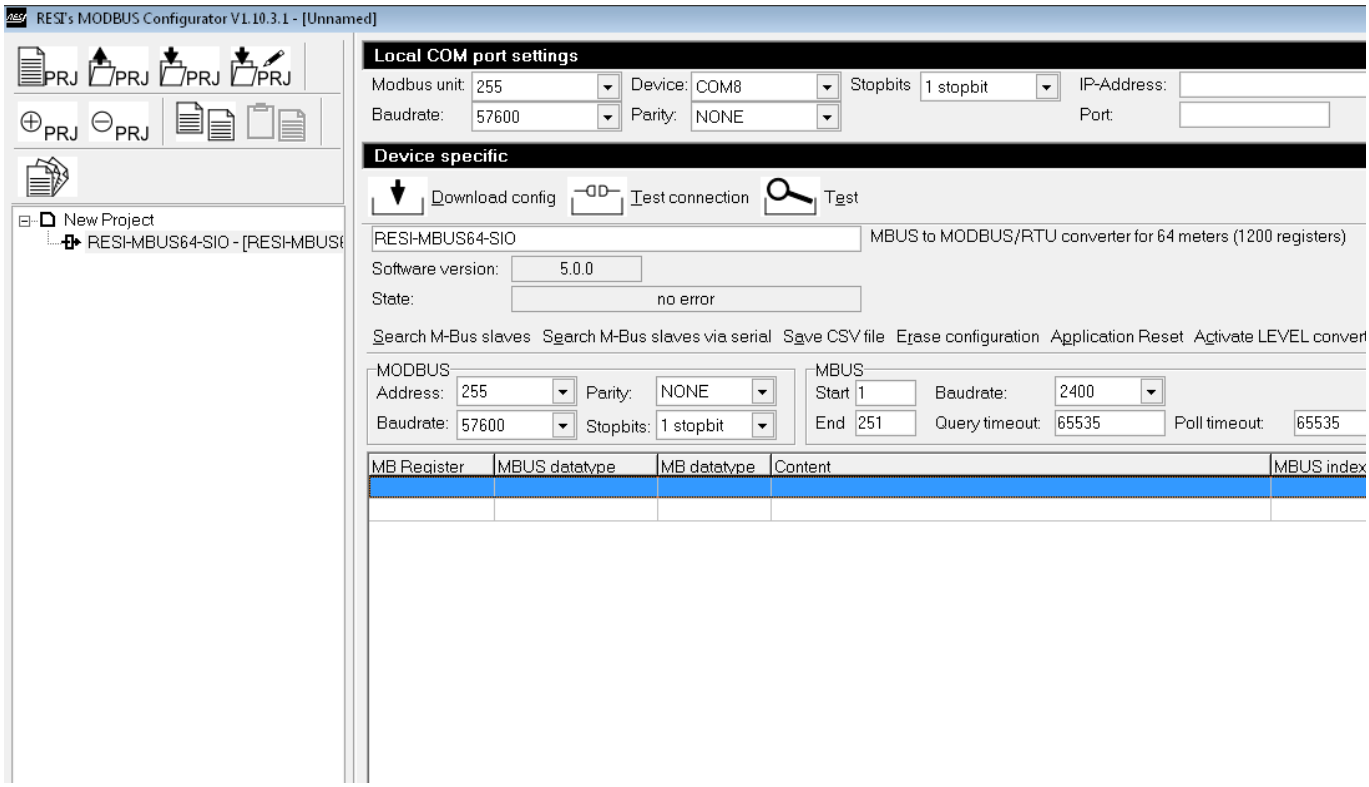


Either choose an existing manufacturer from the drop down list or set the radio button to the Add new manufacturer section and enter a new manufacturer name. In our example we choose the name MY MANUFACTURER and the meter name MY METER and we press the OK button. All your defined datapoints for this meter are stored in the user specific database and the meter is added to the user specific database for meter templates.



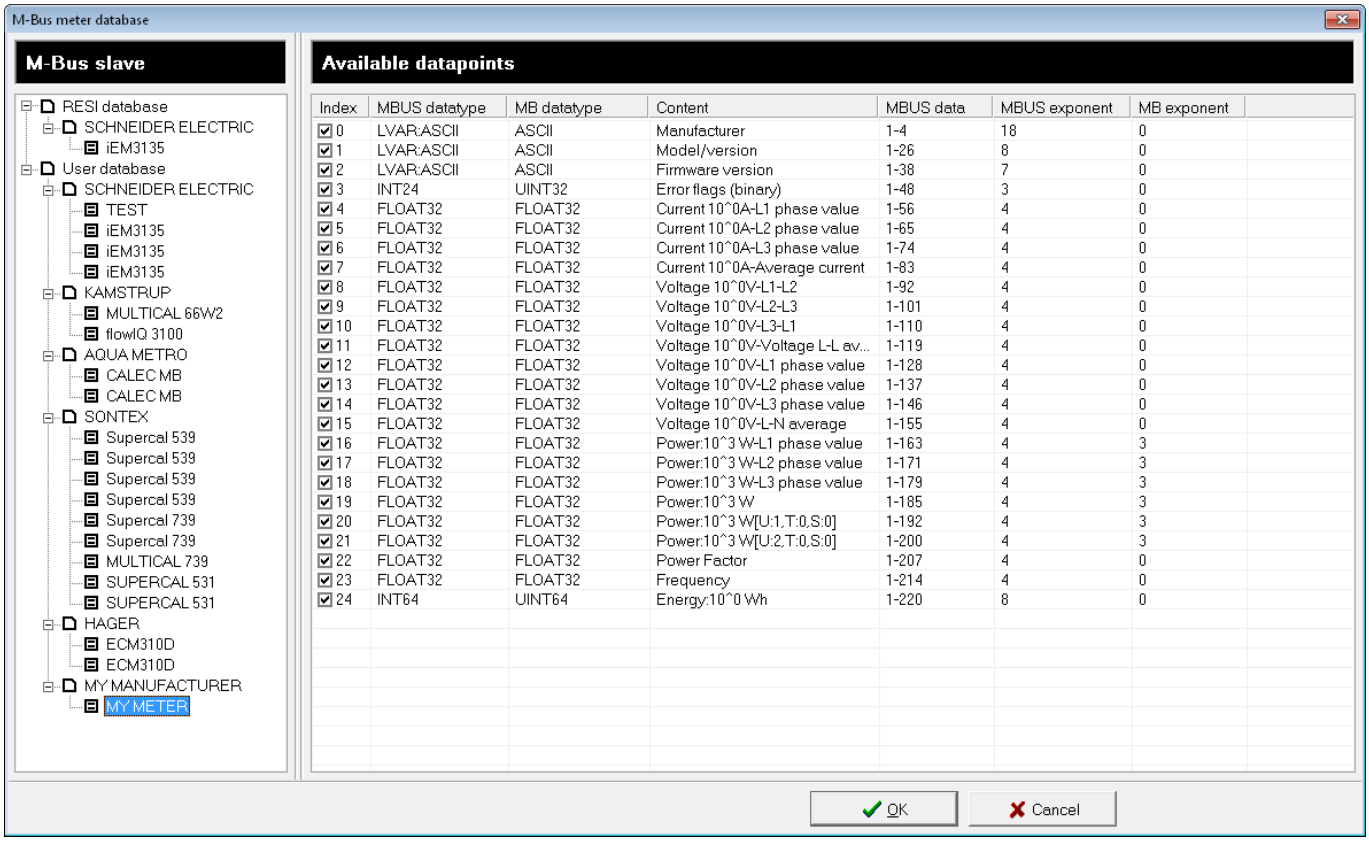
16.12 HOWTO add a complete meter from the database

You can add meter mappings manually to your gateway from previous saved own meters or form our general meter database. First you need a MBUS gateway in your project. Click on the project tree to select the MBUS gateway:



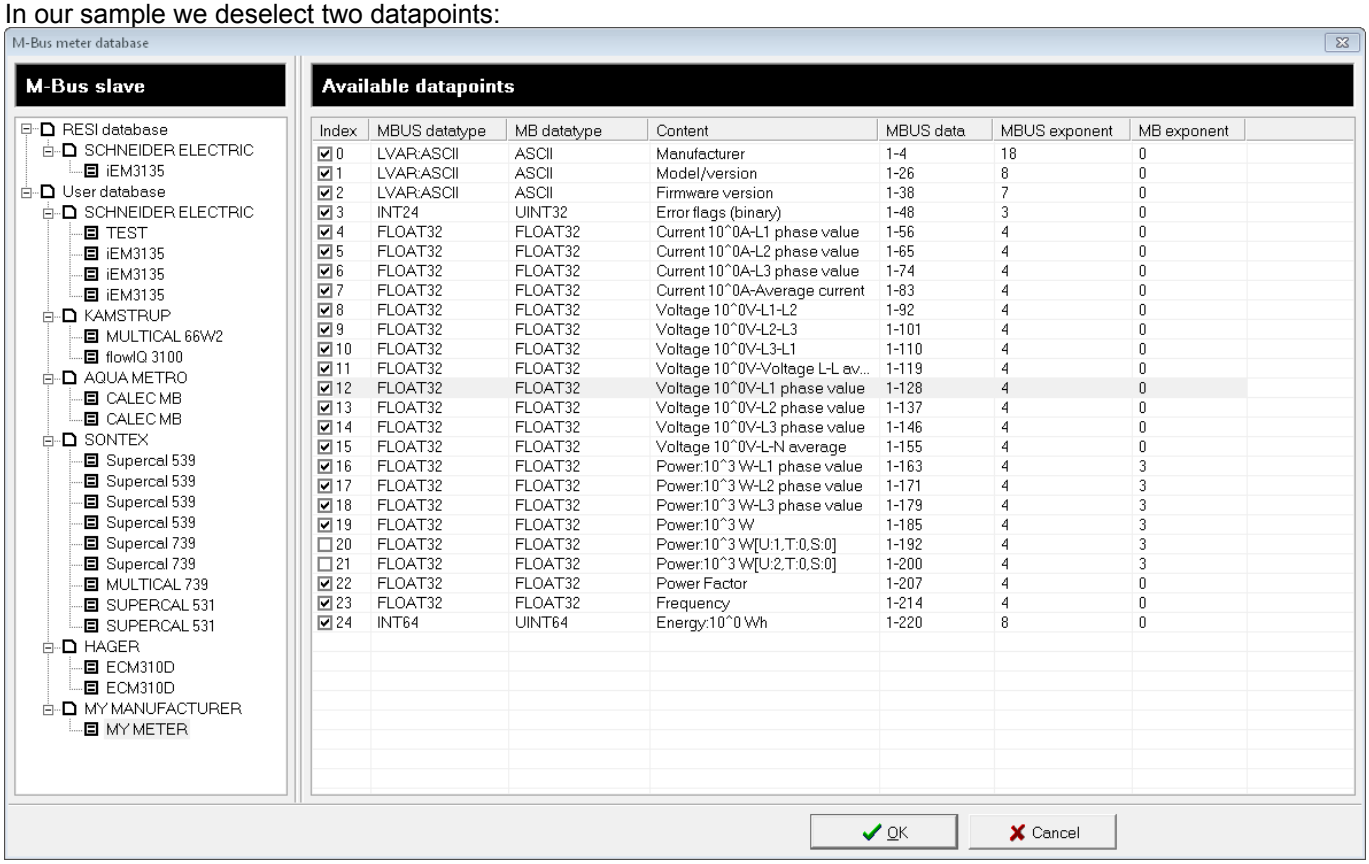
Now click on the button Add to project: 

. A dialog will open and show all meters from the general meter database and all your user defined meter templates:

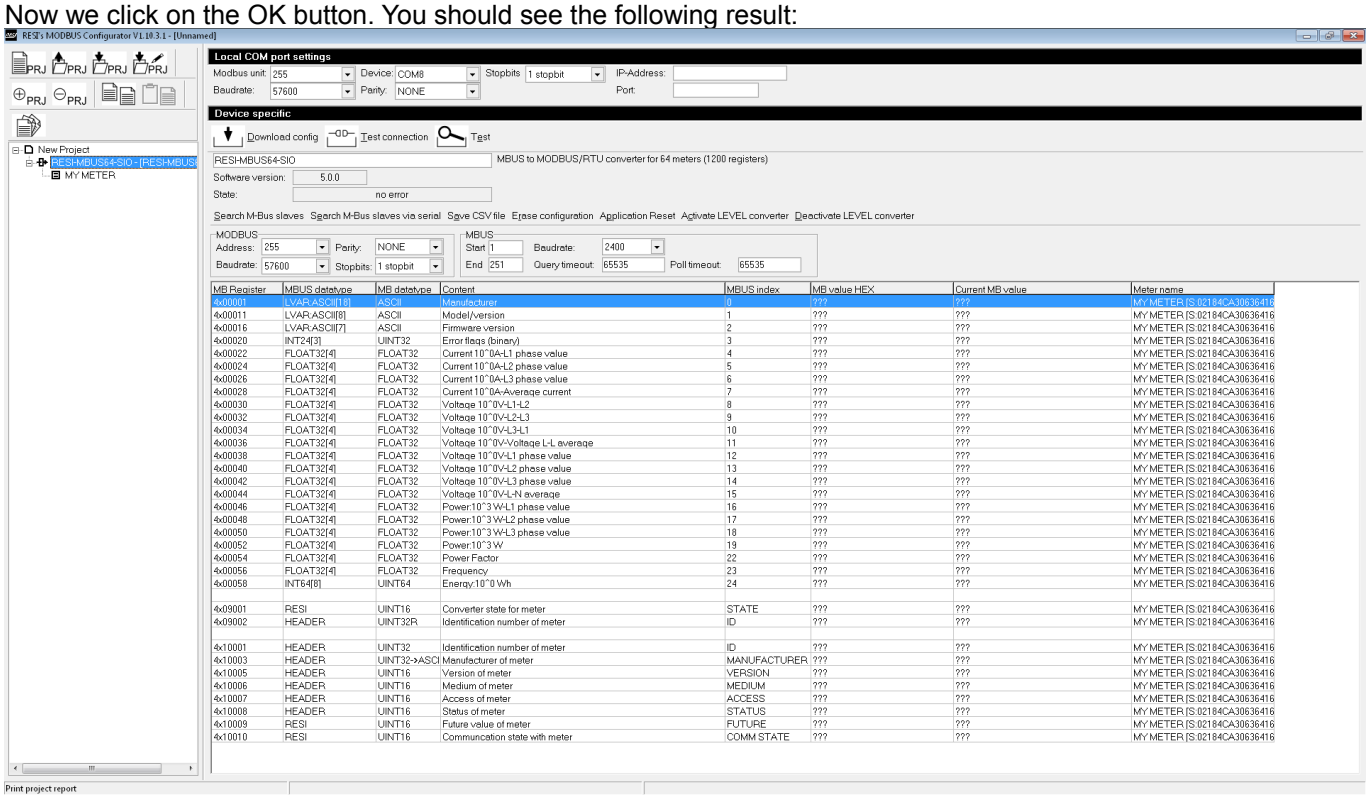


Select the meter MY MANUFACTURER/MY METER like shown above. Note the checkbox beside the datapoint index: Only the datapoints selected in this list are added to the gateway. You can change the selection status by clickin onto the checkbox for each datapoint. If you do a right click in the area of the data grind with the MBUS datapoints you will see a drop down menu with the two options Select all and Deselect all for fast selection/deselection of all datapoints.

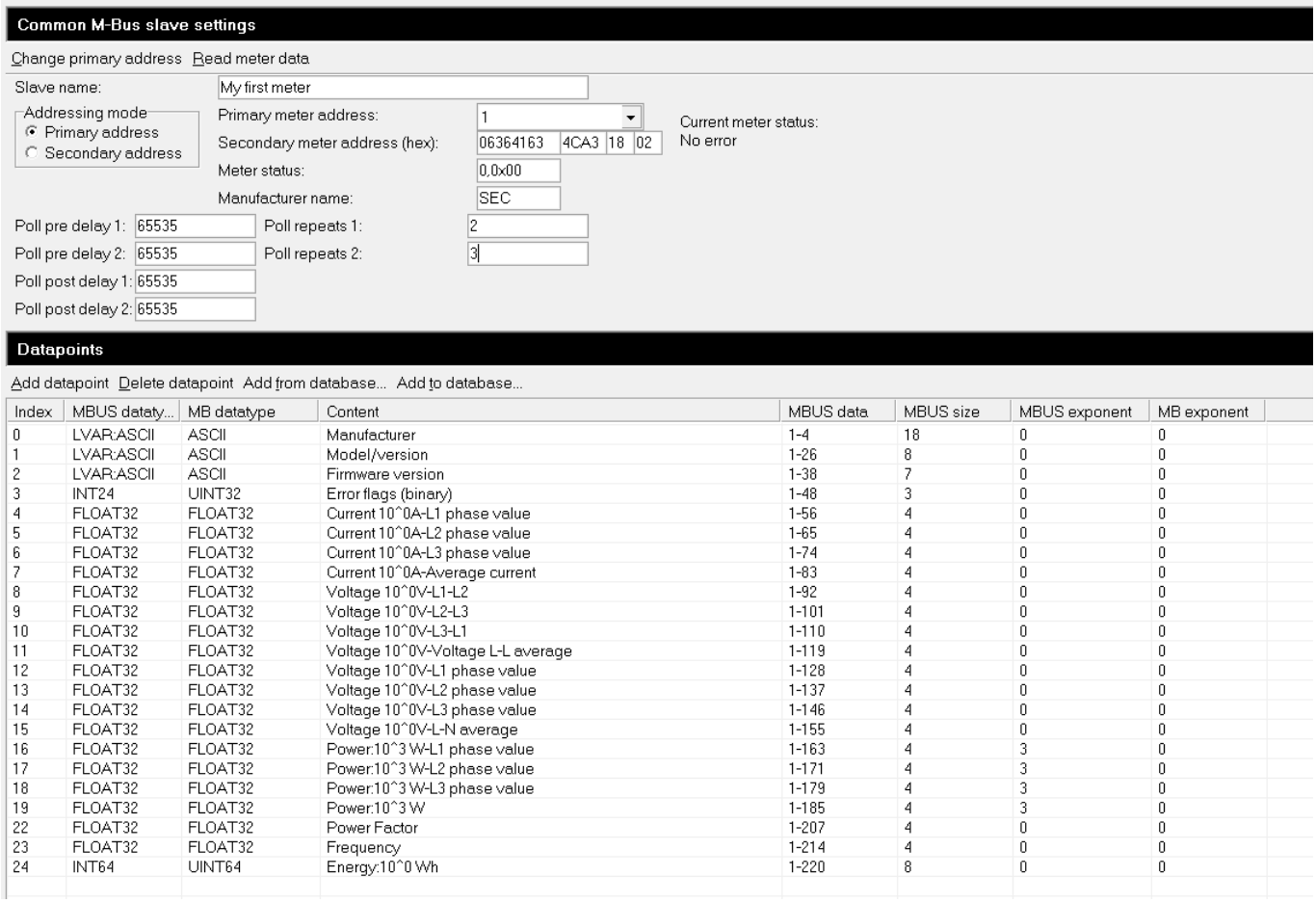
In our sample we deselect two datapoints:



Now we click on the OK button. You should see the following result:



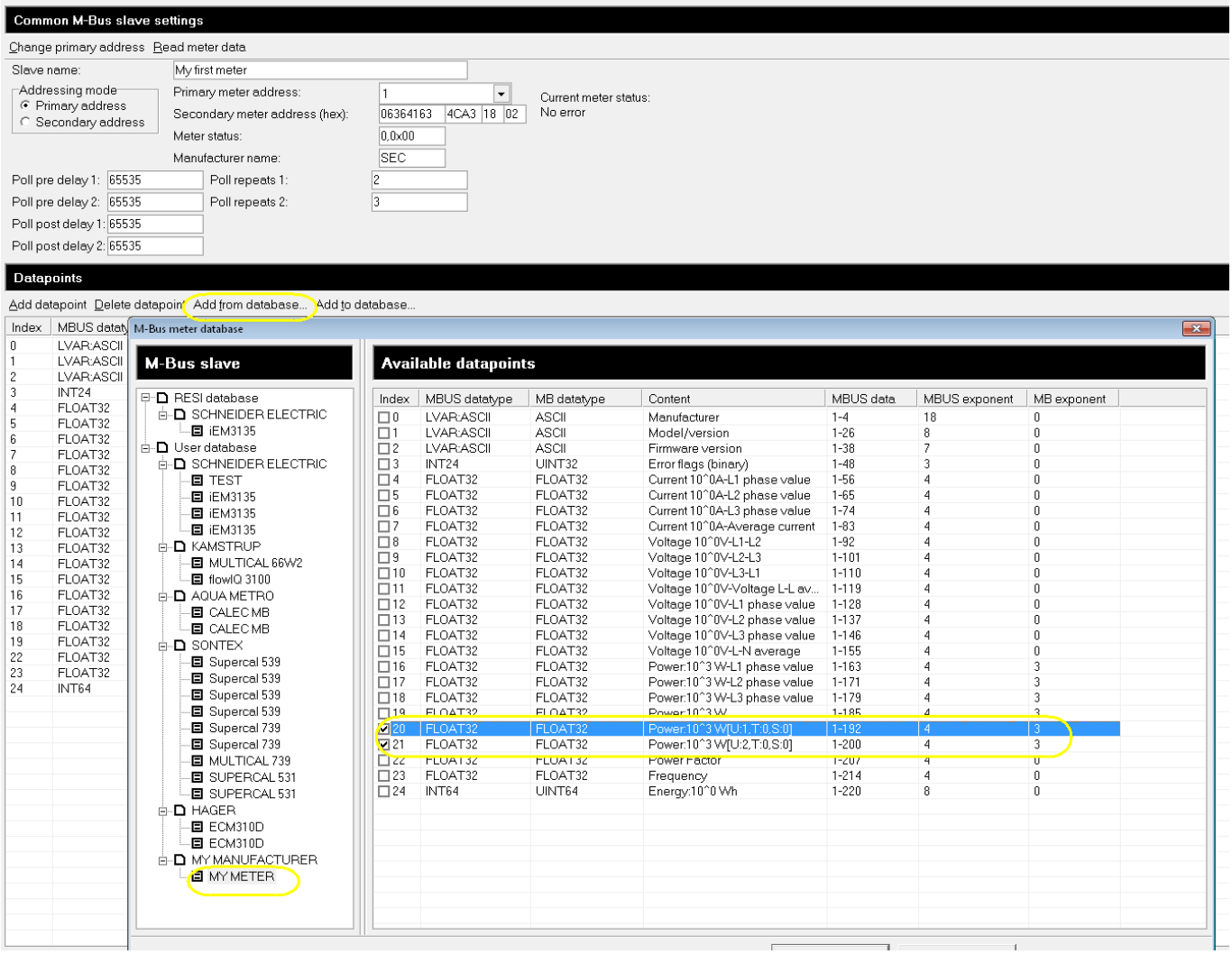
Click in the project tree on the meter MY METER to change the individual parameters for the selected meter. Adopt the addressing mode, the meter name and the other parameters of the meter, so that your gateway can communicate with this meter.



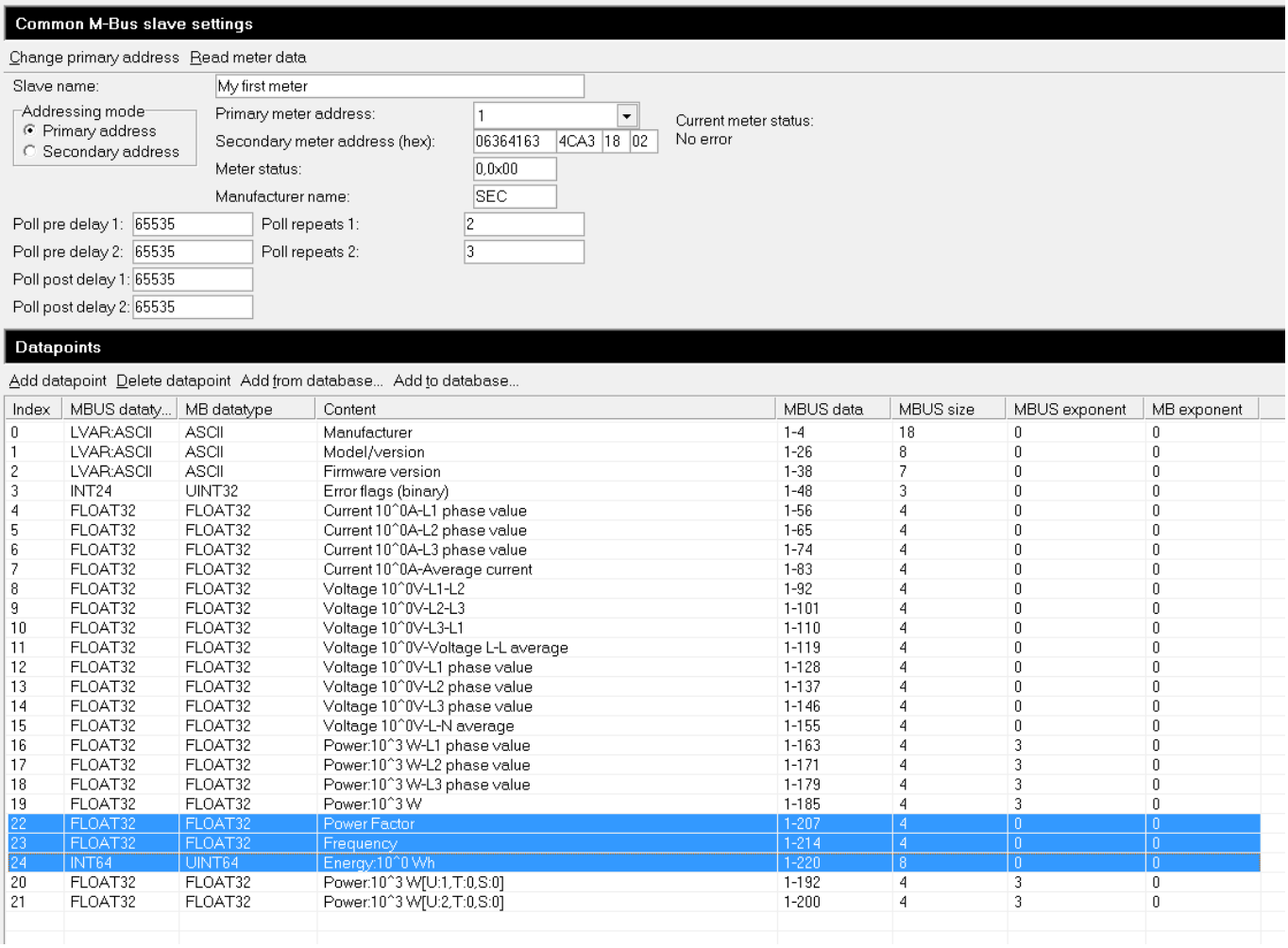
Index	MBUS datatyp...	MB datatype	Content	MBUS data	MBUS size	MBUS exponent	MB exponent
0	LVAR:ASCII	ASCII	Manufacturer	1-4	18	0	0
1	LVAR:ASCII	ASCII	Model/version	1-26	8	0	0
2	LVAR:ASCII	ASCII	Firmware version	1-38	7	0	0
3	INT24	UINT32	Error flags (binary)	1-48	3	0	0
4	FLOAT32	FLOAT32	Current 10^0A-L1 phase value	1-56	4	0	0
5	FLOAT32	FLOAT32	Current 10^0A-L2 phase value	1-65	4	0	0
6	FLOAT32	FLOAT32	Current 10^0A-L3 phase value	1-74	4	0	0
7	FLOAT32	FLOAT32	Current 10^0A-Average current	1-83	4	0	0
8	FLOAT32	FLOAT32	Voltage 10^0V-L1-L2	1-92	4	0	0
9	FLOAT32	FLOAT32	Voltage 10^0V-L2-L3	1-101	4	0	0
10	FLOAT32	FLOAT32	Voltage 10^0V-L3-L1	1-110	4	0	0
11	FLOAT32	FLOAT32	Voltage 10^0V-Voltage L-L average	1-119	4	0	0
12	FLOAT32	FLOAT32	Voltage 10^0V-L1 phase value	1-128	4	0	0
13	FLOAT32	FLOAT32	Voltage 10^0V-L2 phase value	1-137	4	0	0
14	FLOAT32	FLOAT32	Voltage 10^0V-L3 phase value	1-146	4	0	0
15	FLOAT32	FLOAT32	Voltage 10^0V-L-N average	1-155	4	0	0
16	FLOAT32	FLOAT32	Power:10^3 W-L1 phase value	1-163	4	3	0
17	FLOAT32	FLOAT32	Power:10^3 W-L2 phase value	1-171	4	3	0
18	FLOAT32	FLOAT32	Power:10^3 W-L3 phase value	1-179	4	3	0
19	FLOAT32	FLOAT32	Power:10^3 W	1-185	4	3	0
22	FLOAT32	FLOAT32	Power Factor	1-207	4	0	0
23	FLOAT32	FLOAT32	Frequency	1-214	4	0	0
24	INT64	UINT64	Energy:10^0 Wh	1-220	8	0	0

16.13 HOWTO add meter datapoints to an existing meter

You can add individual datapoints to an existing meter in your configuration. First select the meter in your project tree, where you want to add a datapoint. Then click on the button Add from database and select a meter template with the desired datapoints. In our case we select the meter MY METER. Then we deselect all datapoint by doing a right click on the data grid and choose the menu Deselect all. Then we select the two datapoints and click the OK button.



You will get the following result:



Index	MBUS dataty...	MB datatype	Content	MBUS data	MBUS size	MBUS exponent	MB exponent
0	LVAR:ASCII	ASCII	Manufacturer	1-4	18	0	0
1	LVAR:ASCII	ASCII	Model/version	1-26	8	0	0
2	LVAR:ASCII	ASCII	Firmware version	1-38	7	0	0
3	INT24	UINT32	Error flags (binary)	1-48	3	0	0
4	FLOAT32	FLOAT32	Current 10^0A-L1 phase value	1-56	4	0	0
5	FLOAT32	FLOAT32	Current 10^0A-L2 phase value	1-65	4	0	0
6	FLOAT32	FLOAT32	Current 10^0A-L3 phase value	1-74	4	0	0
7	FLOAT32	FLOAT32	Current 10^0A-Average current	1-83	4	0	0
8	FLOAT32	FLOAT32	Voltage 10^0V-L1-L2	1-92	4	0	0
9	FLOAT32	FLOAT32	Voltage 10^0V-L2-L3	1-101	4	0	0
10	FLOAT32	FLOAT32	Voltage 10^0V-L3-L1	1-110	4	0	0
11	FLOAT32	FLOAT32	Voltage 10^0V-Voltage L-L average	1-119	4	0	0
12	FLOAT32	FLOAT32	Voltage 10^0V-L1 phase value	1-128	4	0	0
13	FLOAT32	FLOAT32	Voltage 10^0V-L2 phase value	1-137	4	0	0
14	FLOAT32	FLOAT32	Voltage 10^0V-L3 phase value	1-146	4	0	0
15	FLOAT32	FLOAT32	Voltage 10^0V-L-N average	1-155	4	0	0
16	FLOAT32	FLOAT32	Power:10^3 W-L1 phase value	1-163	4	3	0
17	FLOAT32	FLOAT32	Power:10^3 W-L2 phase value	1-171	4	3	0
18	FLOAT32	FLOAT32	Power:10^3 W-L3 phase value	1-179	4	3	0
19	FLOAT32	FLOAT32	Power:10^3 W	1-185	4	3	0
22	FLOAT32	FLOAT32	Power Factor	1-207	4	0	0
23	FLOAT32	FLOAT32	Frequency	1-214	4	0	0
24	INT64	UINT64	Energy:10^0 Wh	1-220	8	0	0
20	FLOAT32	FLOAT32	Power:10^3 W[U:1,T:0,S:0]	1-192	4	3	0
21	FLOAT32	FLOAT32	Power:10^3 W[U:2,T:0,S:0]	1-200	4	3	0

Then we select the datapoints 22-24 and delete then from the meter setup, by clicking on the button Delete datapoints. Now our new setup is finished and can be downloaded into the gateway.

Index	MBUS dataty...	MB datatype	Content	MBUS data	MBUS size	MBUS exponent	MB exp
0	LVAR:ASCII	ASCII	Manufacturer	1-4	18	0	0
1	LVAR:ASCII	ASCII	Model/version	1-26	8	0	0
2	LVAR:ASCII	ASCII	Firmware version	1-38	7	0	0
3	INT24	UINT32	Error flags (binary)	1-48	3	0	0
4	FLOAT32	FLOAT32	Current 10^0A-L1 phase value	1-56	4	0	0
5	FLOAT32	FLOAT32	Current 10^0A-L2 phase value	1-65	4	0	0
6	FLOAT32	FLOAT32	Current 10^0A-L3 phase value	1-74	4	0	0
7	FLOAT32	FLOAT32	Current 10^0A-Average current	1-83	4	0	0
8	FLOAT32	FLOAT32	Voltage 10^0V-L1-L2	1-92	4	0	0
9	FLOAT32	FLOAT32	Voltage 10^0V-L2-L3	1-101	4	0	0
10	FLOAT32	FLOAT32	Voltage 10^0V-L3-L1	1-110	4	0	0
11	FLOAT32	FLOAT32	Voltage 10^0V-Voltage L-L average	1-119	4	0	0
12	FLOAT32	FLOAT32	Voltage 10^0V-L1 phase value	1-128	4	0	0
13	FLOAT32	FLOAT32	Voltage 10^0V-L2 phase value	1-137	4	0	0
14	FLOAT32	FLOAT32	Voltage 10^0V-L3 phase value	1-146	4	0	0
15	FLOAT32	FLOAT32	Voltage 10^0V-L-N average	1-155	4	0	0
16	FLOAT32	FLOAT32	Power:10^3 W-L1 phase value	1-163	4	3	0
17	FLOAT32	FLOAT32	Power:10^3 W-L2 phase value	1-171	4	3	0
18	FLOAT32	FLOAT32	Power:10^3 W-L3 phase value	1-179	4	3	0
19	FLOAT32	FLOAT32	Power:10^3 W	1-185	4	3	0
20	FLOAT32	FLOAT32	Power:10^3 W[U:1,T:0,S:0]	1-192	4	3	0
21	FLOAT32	FLOAT32	Power:10^3 W[U:2,T:0,S:0]	1-200	4	3	0

16.14 Table of MBUS data types

The following table shows, which MBUS data types are used and how they are processed by the gateway:

MBUS DATATYPE	SIZE	BYTE ORDER	DESCRIPTION
BCD2	8 bits 1 byte	Decimal digits HL → 0xHL	Defines an 8 bit unsigned integer value in the range of 0 to 99 stored as BCD number with encoding: Byte 0xHL: Bits 7-4: H: UPPER DIGIT as hex value 0x0 to 0x9 Bits 3-0: L: LOWER DIGIT as hex value 0x0 to 0x9 So hex value 0x12 means decimal value 1*10+2=12 in decimal
NOTE to BCD numbers: According to the MBUS standard not only positive BCD numbers are handled by our gateway. Due to the encoding of BCD numbers as 4 bit hexadecimal characters for each digit, only the hexadecimal numbers 0x0 to 0x9 are used for the decimal representation. Therefore the hexadecimal digits 0xA to 0xF are not used to represent a BCD number. If the leading digit of the BCD number encodes a 0xF, this stands for a negative sign. So the number 0x0123 means the decimal representation of +123, the number 0xF123 means the decimal representation of -123.			
BCD4	16 bits 2 byte	Decimal digits ABCD → 0xCD 0xAB	Defines a 16 bit unsigned integer value in the range of 0 to 9999 stored as BCD number with encoding: First byte 00xCD: Bits 7-4: C: DIGIT*10 as hex value 0x0 to 0x9 Bits 3-0: D: DIGIT*1 as hex value 0x0 to 0x9 Second byte 00xAB: Bits 7-4: A: DIGIT*1000 as hex value 0x0 to 0x9 Bits 3-0: B: DIGIT*100 as hex value 0x0 to 0x9 So hex value 0x1234 means decimal value 1*1000+2*100+3*10+4=1234 in decimal
BCD6	24 bits 3 byte	Decimal digits ABCDEF → 0xEF 0xCD 0xAB	Defines a 24 bit unsigned integer value in the range of 0 to 999999 stored as BCD number with encoding: First byte 00xEF: Bits 7-4: F: DIGIT*10 as hex value 0x0 to 0x9 Bits 3-0: E: DIGIT*1 as hex value 0x0 to 0x9 Second byte 00xCD: Bits 7-4: C: DIGIT*1000 as hex value 0x0 to 0x9 Bits 3-0: D: DIGIT*100 as hex value 0x0 to 0x9 Third byte 00xAB: Bits 7-4: A: DIGIT*100000 as hex value 0x0 to 0x9 Bits 3-0: B: DIGIT*10000 as hex value 0x0 to 0x9 So hex value 0x123456 means decimal value 1*100000+2*10000+3*1000+4*100+5*10+6=123456 in decimal
BCD8	32 bits 4 byte	Decimal digits ABCDEFGH → 0xGH 0xEF 0xCD 0xAB	Defines a 32 bit unsigned integer value in the range of 0 to 99999999 stored as BCD number with encoding: First byte 00xGH: Bits 7-4: G: DIGIT*10 as hex value 0x0 to 0x9 Bits 3-0: H: DIGIT*1 as hex value 0x0 to 0x9 Second byte 00xEF: Bits 7-4: E: DIGIT*1000 as hex value 0x0 to 0x9 Bits 3-0: F: DIGIT*100 as hex value 0x0 to 0x9 Third byte 00xCD: Bits 7-4: C: DIGIT*100000 as hex value 0x0 to 0x9 Bits 3-0: D: DIGIT*10000 as hex value 0x0 to 0x9 Fourth byte 00xAB: Bits 7-4: A: DIGIT*10000000 as hex value 0x0 to 0x9 Bits 3-0: B: DIGIT* 1000000 as hex value 0x0 to 0x9 So hex value 0x12345678 means decimal value 1*10000000+2*1000000+3*100000+4*10000+5*1000+6*100+7*10+8=12345678 in decimal

MBUS DATATYPE	SIZE	BYTE ORDER	DESCRIPTION
BCD12	48 bits 6 byte	Decimal digits ABCDEFGHIJKL → 0xKL 0xIJ 0xGH 0xEF 0xCD 0xAB	Defines a 48 bit unsigned integer value in the range of 0 to 999999999999 stored as BCD number with encoding: First byte 00xKL: Bits 7-4: K: DIGIT*10 as hex value 0x0 to 0x9 Bits 3-0: L: DIGIT*1 as hex value 0x0 to 0x9 Second byte 00xIJ: Bits 7-4: I: DIGIT*1000 as hex value 0x0 to 0x9 Bits 3-0: J: DIGIT*100 as hex value 0x0 to 0x9 Third byte 00xGH: Bits 7-4: G: DIGIT*100000 as hex value 0x0 to 0x9 Bits 3-0: H: DIGIT*10000 as hex value 0x0 to 0x9 Fourth byte 00xEF: Bits 7-4: E: DIGIT*10000000 as hex value 0x0 to 0x9 Bits 3-0: F: DIGIT*1000000 as hex value 0x0 to 0x9 Fifth byte 00xCD: Bits 7-4: C: DIGIT*1000000000 as hex value 0x0 to 0x9 Bits 3-0: D: DIGIT*100000000 as hex value 0x0 to 0x9 Sixth byte 00xAB: Bits 7-4: A: DIGIT*100000000000 as hex value 0x0 to 0x9 Bits 3-0: B: DIGIT*10000000000 as hex value 0x0 to 0x9 So hex value 0x123456789012 means decimal value 1*100000000000+2*10000000000+ 3*1000000000+4*100000000+ 5*10000000+6*1000000+7*100000+8*10000+ 9*1000+0*100+1*10+2*1 =123456789012 in decimal
SINT8	8 bits 1 byte	none	Defines a 8 bit signed integer value in the range of -128 to +127 or 0x80 to 0x7F First byte: Bit 7: Sign Bits 6-0: integer value
SINT16	16 bits 2 byte	0x1234 → 0x34 0x12	Defines a 16 bit signed integer value in the range of -32768 to +32767 or 0x8000 to 0x7FFF First byte: Bits 7-0 Second byte: Bit 15: Sign Bits 15-8
SINT24	24 bits 3 byte	0x123456 → 0x56 0x34 0x12	Defines a 24 bit signed integer value in the range of -8.388.608 to +8.388.608 or 0x80.0000 to 0x7F.FFFF First byte: Bits 7-0 Second byte: Bits 15-8 Third byte: Bit 23: Sign Bits 22-16
SINT32	32 bits 4 byte	0x12345678 → 0x78 0x56 0x34 0x12	Defines a 32 bit signed integer value in the range of -4.294.967.296 to 4.294.967.295 or 0x8000.0000 to 0x7FFF.FFFF First byte: Bits 7-0 Second byte: Bits 15-8 Third byte: Bits 23-16 Fourth byte: Bit 31: Sign Bits 30-24
SINT48	48 bits 6 byte	0x1234567890 → 0x90 0x78 0x56 0x34 0x12	Defines a 32 bit signed integer value in the range of -140.737.488.355.328 to +140.737.488.355.327 or 0x8000.0000.0000 to 0x7FFF.FFFF.FFFF First byte: Bits 7-0 Second byte: Bits 15-8 Third byte: Bits 23-16 Fourth byte: Bits 31-24 Fifth byte: Bits 39-32 Sixth byte: Bit 47: Sign Bits 46-40
SINT64	64 bits 8 byte	0x12345678 90ABCDEF → 0xEF 0xCD 0xAB 0x90 0x78 0x56 0x34 0x12	Defines a 32 bit signed integer value in the range of -9.223.372.036.854.775.808 to +9.223.372.036.854.775.807 or 0x8000.0000.0000.0000 to 0x7FFF.FFFF.FFFF.FFFF First byte: Bits 7-0 Second byte: Bits 15-8 Third byte: Bits 23-16 Fourth byte: Bits 31-24 Fifth byte: Bits 39-32 Sixth byte: Bits 47-40 Seventh byte: Bits 55-48 Eight byte: Bit 63: Sign Bits 62-56

MBUS DATATYPE	SIZE	BYTE ORDER	DESCRIPTION
FLOAT32	32 bits 4 byte	0x40490FDA → 0xDA 0x0F 0x49 0x40	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. Fraction F:=Bits 0..22 Exponent E:=Bits 30..23 Sigh S::= Bit 31 First byte: Fraction F Bits 7-0 Second byte: Fraction F Bits 15-8 Third byte: Exponent E Bit 23 Fraction F Bits 22-16 Fourth byte: Sign S Bit 31 Exponent Bits 30-24
DATE & TIME TYPE F	32 bits 4 byte	0x12345678 → 0x78 0x56 0x34 0x12	Defines a 32 bit value interpreted as date & time Minutes: Bits 5-0 → 0..59 Hour: Bits 12-8 → 0..23 Day: Bits 20-16 → 1..31 Month: Bits 27-23 → 1..12 Year: Bits 31-28,23-21 → 0..99 Invalid: Bit 7: =0 valid, =1: invalid Summertime Bit 15 =0 standard time, =1 summer time Reserved Bit 6 =0 Reserved Bit 13 =0 Reserved Bit 15 =0
DATE TYPE G	16 bits 2 byte	0x1234 → 0x34 0x12	Defines a 16 bit value interpreted as date Day: Bits 4-0 → 1..31 Month: Bits 11-8 → 1..12 Year: Bits 15-12,7-5 → 0..99
VARIABLE LENGTH	n*8 bits n bytes	Byte n-1 Byte n-2 … Byte 2 Byte 1 Byte 0	Defines a variable length field with n bytes of data. First byte: data[n-1] Second byte: data[n-2] … n-1. byte: data[1] n. byte (last byte): data[0] The length byte defines the representation of the variable length data field: LEN=0x00..0xBF: ASCII string LEN=0xC0..0xCF: positive BCD number with (LEN-0xC0)*2 digits LEN=0xD0..0xDF: negative BCD number with (LEN-0xD0)*2 digits LEN=0xE0..0xEF: integer number with (LEN-0xE0) bytes LEN=0xF0..0xFA: float number with (LEN-0xF0) bytes LEN=0xFB..0xFF: reserved

16.15 Table of MODBUS data types

The following table shows, which MODBUS data types are used and how they are processed by the gateway:

MODBUS DATATYPE	SIZE	WORD ORDER	DESCRIPTION
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:Higher 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.
ASCII	2*n*8 bits n register	0:Highest Word 1:Higher Word …. n-1:Lower Word n: Lowest Word	Defines a byte array with ASCII characters stored in 16 bit words. The ASCII string is terminated with a trailing 0x00 byte. To achieve word alignment a second 0x00 character can be stuffed at the end of the string. The low byte of the first word holds the first ASCII character, The high byte of the first word holds the second ASCII character and so on.
ASCIIR	2*n*8 bits n register	0:Lowest Word 1:Lower Word …. n-1:Higher Word n: HIghest Word	Defines a byte array with ASCII characters stored in 16 bit words. The ASCII string is terminated with a trailing 0x00 byte. To achieve word alignment a second 0x00 character can be stuffed at the end of the string. The low byte of the last word holds the first ASCII character, The high byte of the last word holds the second ASCII character and so on.
DATE TIME TYPE F	32 bits 2 register	0:High Word 1:Low Word	Defines a 32 bit value interpreted as date & time Minutes: Bits 5-0 → 0..59 Hour: Bits 12-8 → 0..23 Day: Bits 20-16 → 1..31 Month: Bits 27-23 → 1..12 Year: Bits 31-28,23-21 → 0..99 Invalid: Bit 7: =0 valid, =1: invalid Summertime Bit 15 =0 standard time, =1 summer time Reserved Bit 6 =0 Reserved Bit 13 =0 Reserved Bit 15 =0
DATE TIME TYPE FR	32 bits 2 register	0:Low Word 1:High Word	Defines a 32 bit value interpreted as date & time Minutes: Bits 5-0 → 0..59 Hour: Bits 12-8 → 0..23 Day: Bits 20-16 → 1..31 Month: Bits 27-23 → 1..12 Year: Bits 31-28,23-21 → 0..99 Invalid: Bit 7: =0 valid, =1: invalid Summertime Bit 15 =0 standard time, =1 summer time Reserved Bit 6 =0 Reserved Bit 13 =0 Reserved Bit 15 =0
DATE TYP G	16 bits 1 register	none	Defines a 16 bit value interpreted as date Day: Bits 4-0 → 1..31 Month: Bits 11-8 → 1..12 Year: Bits 15-12,7-5 → 0..99

MODBUS DATATYPE	SIZE	WORD ORDER	DESCRIPTION
BUFFER	2*n*8 bits n register	0:Highest Word 1:Higher Word …. n-1:Lower Word n: Lowest Word	Defines a byte array stored in 16 bit words. To achieve word alignment an additional 0x00 byte can be stuffed at the end of the byte array. The low byte of the first word holds the first byte, The high byte of the first word holds the second byte and so on.
BUFFERR	2*n*8 bits n register	0:Lowest Word 1:Lower Word …. n-1:Higher Word n: HIghest Word	Defines a byte array stored in 16 bit words. To achieve word alignment an additional 0x00 byte can be stuffed at the end of the byte array. The low byte of the last word holds the first byte, The high byte of the last word holds the second byte and so on.

16.16 HOW the MBUS to MODBUS mapping works

The following section describes, how the internal process of mapping MBUS to MODBUS datapoints in done by the gateway. For that we take a SCHNEIDER electrical meter (primary address 4) as a sample to describe the main principle of the conversion process.

In this case, the meter answers with only one data frame to the MBUS request of the gateway (All bytes in hexadecimal):

GATEWAY to METER: Send slave init to primary address 4
10 40 04 44 16
METER to GATEWAY: Send OK
E5
GATEWAY to METER: Send REQ_UD2 frame to metering
10 7B 04 7F 16
METER TO GATEWAY: Send variable data frame as answered
68 F4 F4 68 08 04 72 63 41 36 06 A3 4C 18 02 C8 00 00 00 0D FD 0A 12 63 69 72 74 63 65 6C 45 20 72 65 64 69 65 6E 68 63 53 0D FD 0C 08 20 35 33 31 33 4D 45 69 0D FD 0E 07 32 30 30 2E 34 2E 31 03 FD 17 00 00 00 05 FD DC FF 01 00 00 00 00 05 FD DC FF 02 00 00 C0 FF 05 FD DC FF 03 00 00 C0 FF 05 FD DC FF 00 00 00 00 00 05 FD C9 FF 05 00 00 C0 FF 05 FD C9 FF 06 00 00 C0 FF 05 FD C9 FF 07 00 00 C0 FF 05 FD C9 FF 08 00 00 C0 FF 05 FD C9 FF 01 8E 8B 64 43 05 FD C9 FF 02 00 00 C0 FF 05 FD C9 FF 03 00 00 C0 FF 05 FD C9 FF 04 8E 8B 64 43 05 AE FF 01 00 00 00 00 05 AE FF 02 00 00 C0 FF 05 AE FF 03 00 00 C0 FF 05 2E 00 00 00 00 85 40 2E 00 00 00 00 85 80 40 2E 00 00 00 00 05 FF 0A 00 00 C0 FF 05 FF 0B 29 F1 47 42 07 03 19 00 00 00 00 00 00 00 1F 1D 16

What data is in the MBUS frame? This is the interpretation of the complete received MBUS frame by our software:

```
================================================================================================================================
MBUS:FRAME TYPE:0x72:12 BYTE HEADER+VARIABLE DATA
================================================================================================================================
MBUS:HEADER12:ID:104218979,0x06364163
MBUS:HEADER12:MANUFACTURER:19619,0x4CA3,SEC
MBUS:HEADER12:VERSION:24,0x18
MBUS:HEADER12:MEDIUM:2,0x02
MBUS:HEADER12:MEDIUM:Electricity
MBUS:HEADER12:ACCESS:200,0xC8
MBUS:HEADER12:STATUS:0,0x00
MBUS:HEADER12:STATUS:NO ERROR
MBUS:HEADER12:SIGNATURE:0,0x0000

================================================================================================================================
MBUS:VARIABLE DATA
================================================================================================================================
--------------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:DATARECORD:1:DATABLOCK:0
--------------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:[0]DIF:0D
MBUS:VARIABLE DATA:DIF:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:DIF:BIT 6:LSB STORAGE NUMBER:0
MBUS:VARIABLE DATA:DIF:BIT 5-4:FUNCTION FIELD:0:00:Instantaneous value
MBUS:VARIABLE DATA:DIF:BIT 3-0:DATA FIELD:D:D:1101:variable length
MBUS:VARIABLE DATA:[1]VIF:FD
MBUS:VARIABLE DATA:VIF:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIF:BIT 6-0:UNIT+MULTIPLIER:7D:LINEAR VIF EXTENSION
MBUS:VARIABLE DATA:[2]VIFE:0A
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:0A:SECONDARY VIF (8.4.4) a.
SUB VIF:MANUFACTURER (as in fixed header)
MBUS:VARIABLE DATA:ASCII:18
MBUS:VARIABLE DATA:[3]-[21]DATABLOCK:LENGTH:12,18
MBUS:VARIABLE DATA:DATA BLOCK:DATA:[63][69][72][74][63][65][6C][45][20][72][65][64][69][65][6E][68][63][53]
MBUS:VARIABLE DATA:DATA BLOCK:DATA:ASCII:Schneider Electric
DIFFUNCTIONTEXT:INSTANTANEUS VALUE
DIFTEXT:LVAR:ASCII(18 bytes)
VIFTEXT:Manufacturer
--------------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:DATARECORD:1:DATABLOCK:1
--------------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:[22]DIF:0D
MBUS:VARIABLE DATA:DIF:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:DIF:BIT 6:LSB STORAGE NUMBER:0
MBUS:VARIABLE DATA:DIF:BIT 5-4:FUNCTION FIELD:0:00:Instantaneous value
MBUS:VARIABLE DATA:DIF:BIT 3-0:DATA FIELD:D:D:1101:variable length
MBUS:VARIABLE DATA:[23]VIF:FD
MBUS:VARIABLE DATA:VIF:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIF:BIT 6-0:UNIT+MULTIPLIER:7D:LINEAR VIF EXTENSION
MBUS:VARIABLE DATA:[24]VIFE:0C
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:0C:SECONDARY VIF (8.4.4) a.
SUB VIF:MODEL/VERSION
```

```
MBUS:VARIABLE DATA:ASCII:8
MBUS:VARIABLE DATA:[25]-[33]DATABLOCK:LENGTH:08,8
MBUS:VARIABLE DATA:DATA BLOCK:DATA:[20][35][33][31][33][4D][45][69]
MBUS:VARIABLE DATA:DATA BLOCK:DATA:ASCII:iEM3135
DIFFUNCTIONTEXT:INSTANTANEUS VALUE
DIFTEXT:LVAR:ASCII(8 bytes)
VIFTEXT:Model/version
-------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:DATARECORD:1:DATABLOCK:2
-------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:[34]DIF:0D
MBUS:VARIABLE DATA:DIF:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:DIF:BIT 6:LSB STORAGE NUMBER:0
MBUS:VARIABLE DATA:DIF:BIT 5-4:FUNCTION FIELD:0:00:Instantaneous value
MBUS:VARIABLE DATA:DIF:BIT 3-0:DATA FIELD:D:D:1101:variable length
MBUS:VARIABLE DATA:[35]VIF:FD
MBUS:VARIABLE DATA:VIF:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIF:BIT 6-0:UNIT+MULTIPLIER:7D:LINEAR VIF EXTENSION
MBUS:VARIABLE DATA:[36]VIFE:0E
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:0E:SECONDARY VIF (8.4.4) a.
SUB VIF:FIRMWARE VERSION
MBUS:VARIABLE DATA:ASCII:7
MBUS:VARIABLE DATA:[37]-[44]DATABLOCK:LENGTH:07,7
MBUS:VARIABLE DATA:DATA BLOCK:DATA:[32][30][30][2E][34][2E][31]
MBUS:VARIABLE DATA:DATA BLOCK:DATA:ASCII:1.4.002
DIFFUNCTIONTEXT:INSTANTANEUS VALUE
DIFTEXT:LVAR:ASCII(7 bytes)
VIFTEXT:Firmware version
-------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:DATARECORD:1:DATABLOCK:3
-------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:[45]DIF:03
MBUS:VARIABLE DATA:DIF:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:DIF:BIT 6:LSB STORAGE NUMBER:0
MBUS:VARIABLE DATA:DIF:BIT 5-4:FUNCTION FIELD:0:00:Instantaneous value
MBUS:VARIABLE DATA:DIF:BIT 3-0:DATA FIELD:3:3:0011:24 Bit Integer
MBUS:VARIABLE DATA:[46]VIF:FD
MBUS:VARIABLE DATA:VIF:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIF:BIT 6-0:UNIT+MULTIPLIER:7D:LINEAR VIF EXTENSION
MBUS:VARIABLE DATA:[47]VIFE:17
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:17:SECONDARY VIF (8.4.4) a.
SUB VIF:ERROR FLAGS (BINARY)
MBUS:FIX DATA:[48]-[50]DATABLOCK:LENGTH:03,3
MBUS:FIX DATA:DATA BLOCK:DATA:[00][00][00]
DIFFUNCTIONTEXT:INSTANTANEUS VALUE
DIFTEXT:24BIT INT(3 bytes)
VIFTEXT:Error flags (binary)
-------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:DATARECORD:1:DATABLOCK:4
-------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:[51]DIF:05
MBUS:VARIABLE DATA:DIF:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:DIF:BIT 6:LSB STORAGE NUMBER:0
MBUS:VARIABLE DATA:DIF:BIT 5-4:FUNCTION FIELD:0:00:Instantaneous value
MBUS:VARIABLE DATA:DIF:BIT 3-0:DATA FIELD:5:5:0101:32 Bit Real
MBUS:VARIABLE DATA:[52]VIF:FD
MBUS:VARIABLE DATA:VIF:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIF:BIT 6-0:UNIT+MULTIPLIER:7D:LINEAR VIF EXTENSION
MBUS:VARIABLE DATA:[53]VIFE:DC
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:5C:SECONDARY VIF (8.4.4) a.
SUB VIF:Current 10^0A
MBUS:VARIABLE DATA:[54]VIFE:FF:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:1:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:7F:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:[55]VIFE:01:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:0:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:01:MANUFACTURER SPECIFIC:SEC
VIF SEC:L1 phase value
MBUS:FIX DATA:[56]-[59]DATABLOCK:LENGTH:04,4
MBUS:FIX DATA:DATA BLOCK:DATA:[00][00][00][00]
DIFFUNCTIONTEXT:INSTANTANEUS VALUE
DIFTEXT:32BIT FLOAT(4 bytes)
VIFTEXT:Current 10^0A-L1 phase value
-------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:DATARECORD:1:DATABLOCK:5
-------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:[60]DIF:05
MBUS:VARIABLE DATA:DIF:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:DIF:BIT 6:LSB STORAGE NUMBER:0
MBUS:VARIABLE DATA:DIF:BIT 5-4:FUNCTION FIELD:0:00:Instantaneous value
MBUS:VARIABLE DATA:DIF:BIT 3-0:DATA FIELD:5:5:0101:32 Bit Real
MBUS:VARIABLE DATA:[61]VIF:FD
MBUS:VARIABLE DATA:VIF:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIF:BIT 6-0:UNIT+MULTIPLIER:7D:LINEAR VIF EXTENSION
MBUS:VARIABLE DATA:[62]VIFE:DC
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:5C:SECONDARY VIF (8.4.4) a.
SUB VIF:Current 10^0A
MBUS:VARIABLE DATA:[63]VIFE:FF:MANUFACTURER SPECIFIC VIFE
```

```
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:1:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:7F:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:[64]VIFE:02:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:0:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:02:MANUFACTURER SPECIFIC:SEC
VIF SEC:L2 phase value
MBUS:FIX DATA:[65]-[68]DATABLOCK:LENGTH:04,4
MBUS:FIX DATA:DATA BLOCK:DATA:[00][00][C0][FF]
DIFFUNCTIONTEXT:INSTANTANEUS VALUE
DIFTEXT:32BIT FLOAT(4 bytes)
VIFTEXT:Current 10^0A-L2 phase value
-----------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:DATARECORD:1:DATABLOCK:6
-----------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:[69]DIF:05
MBUS:VARIABLE DATA:DIF:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:DIF:BIT 6:LSB STORAGE NUMBER:0
MBUS:VARIABLE DATA:DIF:BIT 5-4:FUNCTION FIELD:0:00:Instantaneous value
MBUS:VARIABLE DATA:DIF:BIT 3-0:DATA FIELD:5:5:0101:32 Bit Real
MBUS:VARIABLE DATA:[70]VIF:FD
MBUS:VARIABLE DATA:VIF:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIF:BIT 6-0:UNIT+MULTIPLIER:7D:LINEAR VIF EXTENSION
MBUS:VARIABLE DATA:[71]VIFE:DC
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:5C:SECONDARY VIF (8.4.4) a.
SUB VIF:Current 10^0A
MBUS:VARIABLE DATA:[72]VIFE:FF:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:1:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:7F:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:[73]VIFE:03:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:0:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:03:MANUFACTURER SPECIFIC:SEC
VIF SEC:L3 phase value
MBUS:FIX DATA:[74]-[77]DATABLOCK:LENGTH:04,4
MBUS:FIX DATA:DATA BLOCK:DATA:[00][00][C0][FF]
DIFFUNCTIONTEXT:INSTANTANEUS VALUE
DIFTEXT:32BIT FLOAT(4 bytes)
VIFTEXT:Current 10^0A-L3 phase value
-----------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:DATARECORD:1:DATABLOCK:7
-----------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:[78]DIF:05
MBUS:VARIABLE DATA:DIF:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:DIF:BIT 6:LSB STORAGE NUMBER:0
MBUS:VARIABLE DATA:DIF:BIT 5-4:FUNCTION FIELD:0:00:Instantaneous value
MBUS:VARIABLE DATA:DIF:BIT 3-0:DATA FIELD:5:5:0101:32 Bit Real
MBUS:VARIABLE DATA:[79]VIF:FD
MBUS:VARIABLE DATA:VIF:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIF:BIT 6-0:UNIT+MULTIPLIER:7D:LINEAR VIF EXTENSION
MBUS:VARIABLE DATA:[80]VIFE:DC
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:5C:SECONDARY VIF (8.4.4) a.
SUB VIF:Current 10^0A
MBUS:VARIABLE DATA:[81]VIFE:FF:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:1:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:7F:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:[82]VIFE:00:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:0:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:00:MANUFACTURER SPECIFIC:SEC
VIF SEC:Average current
MBUS:FIX DATA:[83]-[86]DATABLOCK:LENGTH:04,4
MBUS:FIX DATA:DATA BLOCK:DATA:[00][00][00][00]
DIFFUNCTIONTEXT:INSTANTANEUS VALUE
DIFTEXT:32BIT FLOAT(4 bytes)
VIFTEXT:Current 10^0A-Average current
-----------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:DATARECORD:1:DATABLOCK:8
-----------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:[87]DIF:05
MBUS:VARIABLE DATA:DIF:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:DIF:BIT 6:LSB STORAGE NUMBER:0
MBUS:VARIABLE DATA:DIF:BIT 5-4:FUNCTION FIELD:0:00:Instantaneous value
MBUS:VARIABLE DATA:DIF:BIT 3-0:DATA FIELD:5:5:0101:32 Bit Real
MBUS:VARIABLE DATA:[88]VIF:FD
MBUS:VARIABLE DATA:VIF:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIF:BIT 6-0:UNIT+MULTIPLIER:7D:LINEAR VIF EXTENSION
MBUS:VARIABLE DATA:[89]VIFE:C9
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:49:SECONDARY VIF (8.4.4) a.
SUB VIF:Voltage 10^0V
MBUS:VARIABLE DATA:[90]VIFE:FF:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:1:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:7F:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:[91]VIFE:05:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:0:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:05:MANUFACTURER SPECIFIC:SEC
VIF SEC:L1-L2
MBUS:FIX DATA:[92]-[95]DATABLOCK:LENGTH:04,4
MBUS:FIX DATA:DATA BLOCK:DATA:[00][00][C0][FF]
DIFFUNCTIONTEXT:INSTANTANEUS VALUE
DIFTEXT:32BIT FLOAT(4 bytes)
VIFTEXT:Voltage 10^0V-L1-L2
```

```
-------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:DATARECORD:1:DATABLOCK:9
-------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:[96]DIF:05
MBUS:VARIABLE DATA:DIF:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:DIF:BIT 6:LSB STORAGE NUMBER:0
MBUS:VARIABLE DATA:DIF:BIT 5-4:FUNCTION FIELD:0:00:Instantaneous value
MBUS:VARIABLE DATA:DIF:BIT 3-0:DATA FIELD:5:5:0101:32 Bit Real
MBUS:VARIABLE DATA:[97]VIF:FD
MBUS:VARIABLE DATA:VIF:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIF:BIT 6-0:UNIT+MULTIPLIER:7D:LINEAR VIF EXTENSION
MBUS:VARIABLE DATA:[98]VIFE:C9
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:49:SECONDARY VIF (8.4.4) a.
SUB VIF:Voltage 10^0V
MBUS:VARIABLE DATA:[99]VIFE:FF:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:1:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:7F:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:[100]VIFE:06:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:0:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:06:MANUFACTURER SPECIFIC:SEC
VIF SEC:L2-L3
MBUS:FIX DATA:[101]-[104]DATABLOCK:LENGTH:04,4
MBUS:FIX DATA:DATA BLOCK:DATA:[00][00][C0][FF]
DIFFUNCTIONTEXT:INSTANTANEUS VALUE
DIFTEXT:32BIT FLOAT(4 bytes)
VIFTEXT:Voltage 10^0V-L2-L3
-------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:DATARECORD:1:DATABLOCK:10
-------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:[105]DIF:05
MBUS:VARIABLE DATA:DIF:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:DIF:BIT 6:LSB STORAGE NUMBER:0
MBUS:VARIABLE DATA:DIF:BIT 5-4:FUNCTION FIELD:0:00:Instantaneous value
MBUS:VARIABLE DATA:DIF:BIT 3-0:DATA FIELD:5:5:0101:32 Bit Real
MBUS:VARIABLE DATA:[106]VIF:FD
MBUS:VARIABLE DATA:VIF:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIF:BIT 6-0:UNIT+MULTIPLIER:7D:LINEAR VIF EXTENSION
MBUS:VARIABLE DATA:[107]VIFE:C9
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:49:SECONDARY VIF (8.4.4) a.
SUB VIF:Voltage 10^0V
MBUS:VARIABLE DATA:[108]VIFE:FF:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:1:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:7F:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:[109]VIFE:07:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:0:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:07:MANUFACTURER SPECIFIC:SEC
VIF SEC:L3-L1
MBUS:FIX DATA:[110]-[113]DATABLOCK:LENGTH:04,4
MBUS:FIX DATA:DATA BLOCK:DATA:[00][00][C0][FF]
DIFFUNCTIONTEXT:INSTANTANEUS VALUE
DIFTEXT:32BIT FLOAT(4 bytes)
VIFTEXT:Voltage 10^0V-L3-L1
-------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:DATARECORD:1:DATABLOCK:11
-------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:[114]DIF:05
MBUS:VARIABLE DATA:DIF:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:DIF:BIT 6:LSB STORAGE NUMBER:0
MBUS:VARIABLE DATA:DIF:BIT 5-4:FUNCTION FIELD:0:00:Instantaneous value
MBUS:VARIABLE DATA:DIF:BIT 3-0:DATA FIELD:5:5:0101:32 Bit Real
MBUS:VARIABLE DATA:[115]VIF:FD
MBUS:VARIABLE DATA:VIF:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIF:BIT 6-0:UNIT+MULTIPLIER:7D:LINEAR VIF EXTENSION
MBUS:VARIABLE DATA:[116]VIFE:C9
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:49:SECONDARY VIF (8.4.4) a.
SUB VIF:Voltage 10^0V
MBUS:VARIABLE DATA:[117]VIFE:FF:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:1:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:7F:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:[118]VIFE:08:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:0:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:08:MANUFACTURER SPECIFIC:SEC
VIF SEC:Voltage L-L average
MBUS:FIX DATA:[119]-[122]DATABLOCK:LENGTH:04,4
MBUS:FIX DATA:DATA BLOCK:DATA:[00][00][C0][FF]
DIFFUNCTIONTEXT:INSTANTANEUS VALUE
DIFTEXT:32BIT FLOAT(4 bytes)
VIFTEXT:Voltage 10^0V-Voltage L-L average
-------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:DATARECORD:1:DATABLOCK:12
-------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:[123]DIF:05
MBUS:VARIABLE DATA:DIF:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:DIF:BIT 6:LSB STORAGE NUMBER:0
MBUS:VARIABLE DATA:DIF:BIT 5-4:FUNCTION FIELD:0:00:Instantaneous value
MBUS:VARIABLE DATA:DIF:BIT 3-0:DATA FIELD:5:5:0101:32 Bit Real
MBUS:VARIABLE DATA:[124]VIF:FD
MBUS:VARIABLE DATA:VIF:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIF:BIT 6-0:UNIT+MULTIPLIER:7D:LINEAR VIF EXTENSION
```

```
MBUS:VARIABLE DATA:[125]VIFE:C9
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:49:SECONDARY VIF (8.4.4) a.
SUB VIF:Voltage 10^0V
MBUS:VARIABLE DATA:[126]VIFE:FF:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:1:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:7F:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:[127]VIFE:01:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:0:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:01:MANUFACTURER SPECIFIC:SEC
VIF SEC:L1 phase value
MBUS:FIX DATA:[128]-[131]DATABLOCK:LENGTH:04,4
MBUS:FIX DATA:DATA BLOCK:DATA:[8E][8B][64][43]
DIFFUNCTIONTEXT:INSTANTANEUS VALUE
DIFTEXT:32BIT FLOAT(4 bytes)
VIFTEXT:Voltage 10^0V-L1 phase value
-------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:DATARECORD:1:DATABLOCK:13
-------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:[132]DIF:05
MBUS:VARIABLE DATA:DIF:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:DIF:BIT 6:LSB STORAGE NUMBER:0
MBUS:VARIABLE DATA:DIF:BIT 5-4:FUNCTION FIELD:0:00:Instantaneous value
MBUS:VARIABLE DATA:DIF:BIT 3-0:DATA FIELD:5:5:0101:32 Bit Real
MBUS:VARIABLE DATA:[133]VIF:FD
MBUS:VARIABLE DATA:VIF:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIF:BIT 6-0:UNIT+MULTIPLIER:7D:LINEAR VIF EXTENSION
MBUS:VARIABLE DATA:[134]VIFE:C9
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:49:SECONDARY VIF (8.4.4) a.
SUB VIF:Voltage 10^0V
MBUS:VARIABLE DATA:[135]VIFE:FF:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:1:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:7F:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:[136]VIFE:02:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:0:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:02:MANUFACTURER SPECIFIC:SEC
VIF SEC:L2 phase value
MBUS:FIX DATA:[137]-[140]DATABLOCK:LENGTH:04,4
MBUS:FIX DATA:DATA BLOCK:DATA:[00][00][C0][FF]
DIFFUNCTIONTEXT:INSTANTANEUS VALUE
DIFTEXT:32BIT FLOAT(4 bytes)
VIFTEXT:Voltage 10^0V-L2 phase value
-------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:DATARECORD:1:DATABLOCK:14
-------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:[141]DIF:05
MBUS:VARIABLE DATA:DIF:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:DIF:BIT 6:LSB STORAGE NUMBER:0
MBUS:VARIABLE DATA:DIF:BIT 5-4:FUNCTION FIELD:0:00:Instantaneous value
MBUS:VARIABLE DATA:DIF:BIT 3-0:DATA FIELD:5:5:0101:32 Bit Real
MBUS:VARIABLE DATA:[142]VIF:FD
MBUS:VARIABLE DATA:VIF:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIF:BIT 6-0:UNIT+MULTIPLIER:7D:LINEAR VIF EXTENSION
MBUS:VARIABLE DATA:[143]VIFE:C9
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:49:SECONDARY VIF (8.4.4) a.
SUB VIF:Voltage 10^0V
MBUS:VARIABLE DATA:[144]VIFE:FF:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:1:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:7F:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:[145]VIFE:03:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:0:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:03:MANUFACTURER SPECIFIC:SEC
VIF SEC:L3 phase value
MBUS:FIX DATA:[146]-[149]DATABLOCK:LENGTH:04,4
MBUS:FIX DATA:DATA BLOCK:DATA:[00][00][C0][FF]
DIFFUNCTIONTEXT:INSTANTANEUS VALUE
DIFTEXT:32BIT FLOAT(4 bytes)
VIFTEXT:Voltage 10^0V-L3 phase value
-------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:DATARECORD:1:DATABLOCK:15
-------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:[150]DIF:05
MBUS:VARIABLE DATA:DIF:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:DIF:BIT 6:LSB STORAGE NUMBER:0
MBUS:VARIABLE DATA:DIF:BIT 5-4:FUNCTION FIELD:0:00:Instantaneous value
MBUS:VARIABLE DATA:DIF:BIT 3-0:DATA FIELD:5:5:0101:32 Bit Real
MBUS:VARIABLE DATA:[151]VIF:FD
MBUS:VARIABLE DATA:VIF:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIF:BIT 6-0:UNIT+MULTIPLIER:7D:LINEAR VIF EXTENSION
MBUS:VARIABLE DATA:[152]VIFE:C9
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:49:SECONDARY VIF (8.4.4) a.
SUB VIF:Voltage 10^0V
MBUS:VARIABLE DATA:[153]VIFE:FF:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:1:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:7F:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:[154]VIFE:04:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:0:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:04:MANUFACTURER SPECIFIC:SEC
VIF SEC:L-N average
```

```
MBUS:FIX DATA:[155]-[158]DATABLOCK:LENGTH:04,4
MBUS:FIX DATA:DATA BLOCK:DATA:[8E][8B][64][43]
DIFFUNCTIONTEXT:INSTANTANEUS VALUE
DIFTEXT:32BIT FLOAT(4 bytes)
VIFTEXT:Voltage 10^0V-L-N average
-------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:DATARECORD:1:DATABLOCK:16
-------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:[159]DIF:05
MBUS:VARIABLE DATA:DIF:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:DIF:BIT 6:LSB STORAGE NUMBER:0
MBUS:VARIABLE DATA:DIF:BIT 5-4:FUNCTION FIELD:0:00:Instantaneous value
MBUS:VARIABLE DATA:DIF:BIT 3-0:DATA FIELD:5:5:0101:32 Bit Real
MBUS:VARIABLE DATA:[160]VIF:AE
MBUS:VARIABLE DATA:VIF:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIF:BIT 6-0:UNIT+MULTIPLIER:2E:PRIMARY VIF 8.4.3
VIF POWER:10^3 W
MBUS:VARIABLE DATA:[161]VIFE:FF:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:1:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:7F:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:[162]VIFE:01:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:0:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:01:MANUFACTURER SPECIFIC:SEC
VIF SEC:L1 phase value
MBUS:FIX DATA:[163]-[166]DATABLOCK:LENGTH:04,4
MBUS:FIX DATA:DATA BLOCK:DATA:[00][00][00][00]
DIFFUNCTIONTEXT:INSTANTANEUS VALUE
DIFTEXT:32BIT FLOAT(4 bytes)
VIFTEXT:Power:10^3 W-L1 phase value
-------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:DATARECORD:1:DATABLOCK:17
-------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:[167]DIF:05
MBUS:VARIABLE DATA:DIF:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:DIF:BIT 6:LSB STORAGE NUMBER:0
MBUS:VARIABLE DATA:DIF:BIT 5-4:FUNCTION FIELD:0:00:Instantaneous value
MBUS:VARIABLE DATA:DIF:BIT 3-0:DATA FIELD:5:5:0101:32 Bit Real
MBUS:VARIABLE DATA:[168]VIF:AE
MBUS:VARIABLE DATA:VIF:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIF:BIT 6-0:UNIT+MULTIPLIER:2E:PRIMARY VIF 8.4.3
VIF POWER:10^3 W
MBUS:VARIABLE DATA:[169]VIFE:FF:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:1:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:7F:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:[170]VIFE:02:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:0:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:02:MANUFACTURER SPECIFIC:SEC
VIF SEC:L2 phase value
MBUS:FIX DATA:[171]-[174]DATABLOCK:LENGTH:04,4
MBUS:FIX DATA:DATA BLOCK:DATA:[00][00][C0][FF]
DIFFUNCTIONTEXT:INSTANTANEUS VALUE
DIFTEXT:32BIT FLOAT(4 bytes)
VIFTEXT:Power:10^3 W-L2 phase value
-------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:DATARECORD:1:DATABLOCK:18
-------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:[175]DIF:05
MBUS:VARIABLE DATA:DIF:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:DIF:BIT 6:LSB STORAGE NUMBER:0
MBUS:VARIABLE DATA:DIF:BIT 5-4:FUNCTION FIELD:0:00:Instantaneous value
MBUS:VARIABLE DATA:DIF:BIT 3-0:DATA FIELD:5:5:0101:32 Bit Real
MBUS:VARIABLE DATA:[176]VIF:AE
MBUS:VARIABLE DATA:VIF:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIF:BIT 6-0:UNIT+MULTIPLIER:2E:PRIMARY VIF 8.4.3
VIF POWER:10^3 W
MBUS:VARIABLE DATA:[177]VIFE:FF:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:1:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:7F:MANUFACTURER SPECIFIC VIFE
MBUS:VARIABLE DATA:[178]VIFE:03:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:0:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:03:MANUFACTURER SPECIFIC:SEC
VIF SEC:L3 phase value
MBUS:FIX DATA:[179]-[182]DATABLOCK:LENGTH:04,4
MBUS:FIX DATA:DATA BLOCK:DATA:[00][00][C0][FF]
DIFFUNCTIONTEXT:INSTANTANEUS VALUE
DIFTEXT:32BIT FLOAT(4 bytes)
VIFTEXT:Power:10^3 W-L3 phase value
-------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:DATARECORD:1:DATABLOCK:19
-------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:[183]DIF:05
MBUS:VARIABLE DATA:DIF:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:DIF:BIT 6:LSB STORAGE NUMBER:0
MBUS:VARIABLE DATA:DIF:BIT 5-4:FUNCTION FIELD:0:00:Instantaneous value
MBUS:VARIABLE DATA:DIF:BIT 3-0:DATA FIELD:5:5:0101:32 Bit Real
MBUS:VARIABLE DATA:[184]VIF:2E
MBUS:VARIABLE DATA:VIF:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:VIF:BIT 6-0:UNIT+MULTIPLIER:2E:PRIMARY VIF 8.4.3
VIF POWER:10^3 W
MBUS:FIX DATA:[185]-[188]DATABLOCK:LENGTH:04,4
MBUS:FIX DATA:DATA BLOCK:DATA:[00][00][00][00]
DIFFUNCTIONTEXT:INSTANTANEUS VALUE
```

```
DIFTEXT:32BIT FLOAT(4 bytes)
VIFTEXT:Power:10^3 W
-------------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:DATARECORD:1:DATABLOCK:20
-------------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:[189]DIF:85
MBUS:VARIABLE DATA:DIF:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:DIF:BIT 6:LSB STORAGE NUMBER:0
MBUS:VARIABLE DATA:DIF:BIT 5-4:FUNCTION FIELD:0:00:Instantaneous value
MBUS:VARIABLE DATA:DIF:BIT 3-0:DATA FIELD:5:5:0101:32 Bit Real
MBUS:VARIABLE DATA:[190]DIFE:40
MBUS:VARIABLE DATA:DIFE:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:DIFE:BIT 6:DEVICE UNIT:1
MBUS:VARIABLE DATA:DIFE:BIT 5-4:TARIFF:0
MBUS:VARIABLE DATA:DIFE:BIT 3-0:STORAGE NUMBER:0
MBUS:VARIABLE DATA:[191]VIF:2E
MBUS:VARIABLE DATA:VIF:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:VIF:BIT 6-0:UNIT+MULTIPLIER:2E:PRIMARY VIF 8.4.3
VIF POWER:10^3 W
MBUS:FIX DATA:[192]-[195]DATABLOCK:LENGTH:04,4
MBUS:FIX DATA:DATA BLOCK:DATA:[00][00][00][00]
DIFFUNCTIONTEXT:INSTANTANEUS VALUE
DIFTEXT:32BIT FLOAT(4 bytes)
VIFTEXT:Power:10^3 W
-------------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:DATARECORD:1:DATABLOCK:21
-------------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:[196]DIF:85
MBUS:VARIABLE DATA:DIF:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:DIF:BIT 6:LSB STORAGE NUMBER:0
MBUS:VARIABLE DATA:DIF:BIT 5-4:FUNCTION FIELD:0:00:Instantaneous value
MBUS:VARIABLE DATA:DIF:BIT 3-0:DATA FIELD:5:5:0101:32 Bit Real
MBUS:VARIABLE DATA:[197]DIFE:80
MBUS:VARIABLE DATA:DIFE:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:DIFE:BIT 6:DEVICE UNIT:0
MBUS:VARIABLE DATA:DIFE:BIT 5-4:TARIFF:0
MBUS:VARIABLE DATA:DIFE:BIT 3-0:STORAGE NUMBER:0
MBUS:VARIABLE DATA:[198]DIFE:40
MBUS:VARIABLE DATA:DIFE:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:DIFE:BIT 6:DEVICE UNIT:1
MBUS:VARIABLE DATA:DIFE:BIT 5-4:TARIFF:0
MBUS:VARIABLE DATA:DIFE:BIT 3-0:STORAGE NUMBER:0
MBUS:VARIABLE DATA:[199]VIF:2E
MBUS:VARIABLE DATA:VIF:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:VIF:BIT 6-0:UNIT+MULTIPLIER:2E:PRIMARY VIF 8.4.3
VIF POWER:10^3 W
MBUS:FIX DATA:[200]-[203]DATABLOCK:LENGTH:04,4
MBUS:FIX DATA:DATA BLOCK:DATA:[00][00][00][00]
DIFFUNCTIONTEXT:INSTANTANEUS VALUE
DIFTEXT:32BIT FLOAT(4 bytes)
VIFTEXT:Power:10^3 W
-------------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:DATARECORD:1:DATABLOCK:22
-------------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:[204]DIF:05
MBUS:VARIABLE DATA:DIF:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:DIF:BIT 6:LSB STORAGE NUMBER:0
MBUS:VARIABLE DATA:DIF:BIT 5-4:FUNCTION FIELD:0:00:Instantaneous value
MBUS:VARIABLE DATA:DIF:BIT 3-0:DATA FIELD:5:5:0101:32 Bit Real
MBUS:VARIABLE DATA:[205]VIF:FF
MBUS:VARIABLE DATA:VIF:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIF:BIT 6-0:UNIT+MULTIPLIER:7F:MANUFACTURER SPECIFIC VIF
MBUS:VARIABLE DATA:[206]VIFE:0A:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:0:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:0A:MANUFACTURER SPECIFIC:SEC
VIF SEC:Power Factor
MBUS:FIX DATA:[207]-[210]DATABLOCK:LENGTH:04,4
MBUS:FIX DATA:DATA BLOCK:DATA:[00][00][C0][FF]
DIFFUNCTIONTEXT:INSTANTANEUS VALUE
DIFTEXT:32BIT FLOAT(4 bytes)
VIFTEXT:Power Factor
-------------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:DATARECORD:1:DATABLOCK:23
-------------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:[211]DIF:05
MBUS:VARIABLE DATA:DIF:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:DIF:BIT 6:LSB STORAGE NUMBER:0
MBUS:VARIABLE DATA:DIF:BIT 5-4:FUNCTION FIELD:0:00:Instantaneous value
MBUS:VARIABLE DATA:DIF:BIT 3-0:DATA FIELD:5:5:0101:32 Bit Real
MBUS:VARIABLE DATA:[212]VIF:FF
MBUS:VARIABLE DATA:VIF:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIF:BIT 6-0:UNIT+MULTIPLIER:7F:MANUFACTURER SPECIFIC VIF
MBUS:VARIABLE DATA:[213]VIFE:0B:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:0:MANUFACTURER SPECIFIC:SEC
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:0B:MANUFACTURER SPECIFIC:SEC
VIF SEC:Frequency
MBUS:FIX DATA:[214]-[217]DATABLOCK:LENGTH:04,4
MBUS:FIX DATA:DATA BLOCK:DATA:[29][F1][47][42]
DIFFUNCTIONTEXT:INSTANTANEUS VALUE
DIFTEXT:32BIT FLOAT(4 bytes)
VIFTEXT:Frequency
-------------------------------------------------------------------------------------------------------------------------------
```

```
MBUS:VARIABLE DATA:DATARECORD:1:DATABLOCK:24
-------------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:[218]DIF:07
MBUS:VARIABLE DATA:DIF:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:DIF:BIT 6:LSB STORAGE NUMBER:0
MBUS:VARIABLE DATA:DIF:BIT 5-4:FUNCTION FIELD:0:00:Instantaneous value
MBUS:VARIABLE DATA:DIF:BIT 3-0:DATA FIELD:7:7:0111:64 Bit Integer
MBUS:VARIABLE DATA:[219]VIF:03
MBUS:VARIABLE DATA:VIF:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:VIF:BIT 6-0:UNIT+MULTIPLIER:03:PRIMARY VIF 8.4.3
VIF ENERGY:10^0 Wh
MBUS:FIX DATA:[220]-[227]DATABLOCK:LENGTH:08,8
MBUS:FIX DATA:DATA BLOCK:DATA:[19][00][00][00][00][00][00][00]
DIFFUNCTIONTEXT:INSTANTANEUS VALUE
DIFTEXT:64BIT INT(8 bytes)
VIFTEXT:Energy:10^0 Wh
-------------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:DATARECORD:1:DATABLOCK:25
-------------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:[228]DIF:1F
MBUS:VARIABLE DATA:DIF:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:DIF:BIT 6:LSB STORAGE NUMBER:0
MBUS:VARIABLE DATA:DIF:BIT 5-4:FUNCTION FIELD:1:01:Maximum value
MBUS:VARIABLE DATA:DIF:BIT 3-0:DATA FIELD:F:F:1111:Special Functions
MBUS:VARIABLE DATA:DIF:MORE RECORDS

===============================================================================================================================
MBUS:END VARIABLE DATA
===============================================================================================================================
END OF FRAME
===============================================================================================================================
```

You will notice, that the MBUS answer starts with a fixed header. This information is interpreted by our gateway and stored in a fixed MODBUS mapping structure starting at 4x10001. See the MODBUS register definition for more details.

4x10001	HEADER	UINT32	Identification number of meter	ID	MSW:0636,4163:LSW	104218979,0x06364163	Meter 4 [P:4]
4x10003	HEADER	UINT32->ASCI	Manufacturer of meter	MANUFACTURER	MSW:0043,4553:LSW	SEC	Meter 4 [P:4]
4x10005	HEADER	UINT16	Version of meter	VERSION	WORD:0018	24,0x0018	Meter 4 [P:4]
4x10006	HEADER	UINT16	Medium of meter	MEDIUM	WORD:0002	2,0x0002 -> Electricity	Meter 4 [P:4]
4x10007	HEADER	UINT16	Access of meter	ACCESS	WORD:0056	86,0x0056	Meter 4 [P:4]
4x10008	HEADER	UINT16	Status of meter	STATUS	WORD:0000	0,0x0000	Meter 4 [P:4]
4x10009	RESI	UINT16	Future value of meter	FUTURE	WORD:0000	0,0x0000	Meter 4 [P:4]
4x10010	RESI	UINT16	Communcation state with meter	COMM STATE	WORD:0003	3,0x0003 -> Values are valid!	Meter 4 [P:4]

After the fixed data header, there come one or more data records. The gateway will read this records sequentially and map one datapoint to another to MODBUS registers. So the configured mapping datapoints must be sequentially defined according to the data in the MBUS frame.

The first entry is:

```
-------------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:DATARECORD:1:DATABLOCK:0
-------------------------------------------------------------------------------------------------------------------------------
MBUS:VARIABLE DATA:[0]DIF:0D
MBUS:VARIABLE DATA:DIF:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:DIF:BIT 6:LSB STORAGE NUMBER:0
MBUS:VARIABLE DATA:DIF:BIT 5-4:FUNCTION FIELD:0:00:Instantaneous value
MBUS:VARIABLE DATA:DIF:BIT 3-0:DATA FIELD:D:D:1101:variable length
MBUS:VARIABLE DATA:[1]VIF:FD
MBUS:VARIABLE DATA:VIF:BIT 7:EXTENSION BIT:1
MBUS:VARIABLE DATA:VIF:BIT 6-0:UNIT+MULTIPLIER:7D:LINEAR VIF EXTENSION
MBUS:VARIABLE DATA:[2]VIFE:0A
MBUS:VARIABLE DATA:VIFE:BIT 7:EXTENSION BIT:0
MBUS:VARIABLE DATA:VIFE:BIT 6-0:UNIT+MULTIPLIER:0A:SECONDARY VIF (8.4.4) a.
SUB VIF:MANUFACTURER (as in fixed header)
MBUS:VARIABLE DATA:ASCII:18
MBUS:VARIABLE DATA:[3]-[21]DATABLOCK:LENGTH:12,18
MBUS:VARIABLE DATA:DATA BLOCK:DATA:[63][69][72][74][63][65][6C][45][20][72][65][64][69][65][6E][68][63][53]
MBUS:VARIABLE DATA:DATA BLOCK:DATA:ASCII:Schneider Electric
DIFFUNCTIONTEXT:INSTANTANEUS VALUE
DIFTEXT:LVAR:ASCII(18 bytes)
VIFTEXT:Manufacturer
```

The first entry in our configuration matches this MBUS data. It is an ASCII string which defines the manufacturer of the meter. The Name starts at byte index 4 in the first received data frame from the meter and it needs 18 bytes. So now the gateway knows exactly, that it has to copy the 18 bytes starting form index 4 to the first 9 MODBUS 16-Bit registers starting at 4x00001. But also the byte order is mirrored for the ASCII string and there is no trailing 0x00 character at the end of the string, so our software maps this string to 10 16-bit registers and adds the trailing 0x00 character.

Index	MBUS datat...	MB datatype	Content	MBUS data	MBUS size	MBUS exponent	MB exponent
0	LVAR:ASCII	ASCII	Manufacturer	1-4	18	0	0
1	LVAR:ASCII	ASCII	Model/version	1-26	8	0	0
2	LVAR:ASCII	ASCII	Firmware version	1-38	7	0	0
3	INT24	UINT32	Error flags (binary)	1-48	3	0	0
4	FLOAT32	FLOAT32	Current 10^0A-L1 phase value	1-56	4	0	0
5	FLOAT32	FLOAT32	Current 10^0A-L2 phase value	1-65	4	0	0
6	FLOAT32	FLOAT32	Current 10^0A-L3 phase value	1-74	4	0	0
7	FLOAT32	FLOAT32	Current 10^0A-Average current	1-83	4	0	0
8	FLOAT32	FLOAT32	Voltage 10^0V-L1-L2	1-92	4	0	0
9	FLOAT32	FLOAT32	Voltage 10^0V-L2-L3	1-101	4	0	0
10	FLOAT32	FLOAT32	Voltage 10^0V-L3-L1	1-110	4	0	0

16.16.1 HOW the exponents affect the result

In the MBUS protocol not only the MBUS data for a data point is transmitted, also also the meaning of the data and the dimension of the value is transmitted. For example we take a KAMSTRUP flowIQ meter. Our software generates the following mapping:

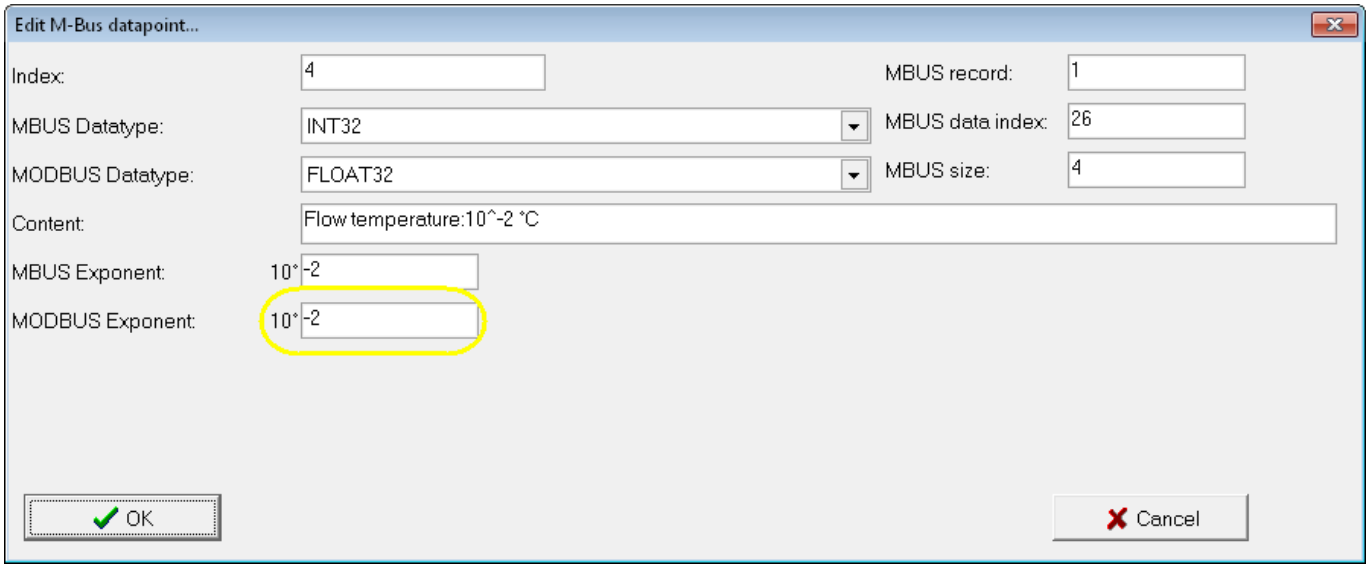
Index	MBUS dataty...	MB datatype	Content	MBUS data	MBUS size	MBUS exponent	MB exponent
0	BCD8	SINT32	Fabrication number	1-2	4	0	0
1	INT32	UINT32	Energy:10^4 Wh	1-8	4	4	0
2	INT32	FLOAT32	Volume:10^-1 m³	1-14	4	-1	0
3	INT32	UINT32	On time:hours	1-20	4	0	0
4	INT32	FLOAT32	Flow temperature:10^-2 °C	1-26	4	-2	0
5	INT32	FLOAT32	Return temperature:10^-2 °C	1-32	4	-2	0
6	INT32	FLOAT32	Temperature difference:10^-2 K	1-38	4	-2	0
7	INT32	FLOAT32	Power:10^2 W	1-44	4	2	0
8	INT32	FLOAT32	Power:10^2 W	1-50	4	2	0
9	INT32	FLOAT32	Volume flow:10^-3 m³/h	1-56	4	-3	0
10	INT32	FLOAT32	Volume flow:10^-3 m³/h	1-62	4	-3	0
11	INT32	FLOAT32	Volume:10^-1 m³[U:1,T:0,S:0]	1-69	4	-1	0
12	INT32	FLOAT32	Volume:10^0 m³[U:2,T:0,S:0]	1-77	4	0	0
13	INT32	DATE_TIME_T...	Time&Date data type F	1-83	4	0	0
14	INT32	UINT32	Energy:10^4 Wh[U:0,T:0,S:1]	1-89	4	4	0
15	INT32	FLOAT32	Volume:10^-1 m³[U:0,T:0,S:1]	1-95	4	-1	0
16	INT32	FLOAT32	Power:10^2 W[U:0,T:0,S:1]	1-101	4	2	0
17	INT32	FLOAT32	Volume flow:10^-3 m³/h[U:0,T:0,S:1]	1-107	4	-3	0
18	INT32	FLOAT32	Volume:10^-1 m³[U:1,T:0,S:1]	1-114	4	-1	0
19	INT32	FLOAT32	Volume:10^0 m³[U:2,T:0,S:1]	1-122	4	0	0
20	INT16	DATE_TYP_G	Date data type G[U:0,T:0,S:1]	1-128	2	0	0
21	SPCL FUNCT	BUFFER	Manufacturer specific data	1-131	44	0	0

You notice in the column MBUS exponent different exponents for different values and in the column Content those exponents are added with 10^xx.

When you download and test this configuration, we get the following online data:

MB Register	MBUS datatype	MB datatype	Content	MBUS index	MB value HEX	Current MB value	Me
4x00001	BCD8[4]	SINT32	Fabrication number	0	MSW:0026,D1D2:LSW	2544082,0x0026D1D2	Me
4x00003	INT32[4]	UINT32	Energy:10^4 Wh	1	MSW:000A,1ED0:LSW	663248,0x000A1ED0	Me
4x00005	INT32[4]	FLOAT32	Volume:10^-1 m³	2	MSW:45AC,6E66:LSW	5517.7998,5.51779980468750E+3	Me
4x00007	INT32[4]	UINT32	On time:hours	3	MSW:0000,B64B:LSW	46667,0x0000B64B	Me
4x00009	INT32[4]	FLOAT32	Flow temperature:10^-2 °C	4	MSW:41F6,147B:LSW	30.7600,3.07600002288818E+1	Me
4x00011	INT32[4]	FLOAT32	Return temperature:10^-2 °C	5	MSW:4081,EB85:LSW	4.0600,4.05999994277954E+0	Me
4x00013	INT32[4]	FLOAT32	Temperature difference:10^-2 K	6	MSW:41D5,999A:LSW	26.7000,2.67000007629395E+1	Me
4x00015	INT32[4]	FLOAT32	Power:10^2 W	7	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Me
4x00017	INT32[4]	FLOAT32	Power:10^2 W	8	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Me
4x00019	INT32[4]	FLOAT32	Volume flow:10^-3 m³/h	9	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Me
4x00021	INT32[4]	FLOAT32	Volume flow:10^-3 m³/h	10	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Me
4x00023	INT32[4]	FLOAT32	Volume:10^-1 m³[U:1,T:0,S:0]	11	MSW:44A1,0CCD:LSW	1288.4000,1.28840002441406E+3	Me
4x00025	INT32[4]	FLOAT32	Volume:10^0 m³[U:2,T:0,S:0]	12	MSW:46CB,1000:LSW	25992.0000,2.59920000000000E+4	Me
4x00027	INT32[4]	DATE_TIME_T	Time&Date data type F	13	MSW:1234,2602:LSW	06:02 D.M.Y:20.02.09 ST:0 IV:0,0x12342602	Me
4x00029	INT32[4]	UINT32	Energy:10^4 Wh[U:0,T:0,S:1]	14	MSW:000A,1ED0:LSW	663248,0x000A1ED0	Me
4x00031	INT32[4]	FLOAT32	Volume:10^-1 m³[U:0,T:0,S:1]	15	MSW:45AC,6E66:LSW	5517.7998,5.51779980468750E+3	Me
4x00033	INT32[4]	FLOAT32	Power:10^2 W[U:0,T:0,S:1]	16	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Me
4x00035	INT32[4]	FLOAT32	Volume flow:10^-3 m³/h[U:0,T:0,S:1]	17	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Me
4x00037	INT32[4]	FLOAT32	Volume:10^-1 m³[U:1,T:0,S:1]	18	MSW:474D,6680:LSW	52582.5000,5.25825000000000E+4	Me
4x00039	INT32[4]	FLOAT32	Volume:10^0 m³[U:2,T:0,S:1]	19	MSW:4900,6010:LSW	525825.0000,5.25825000000000E+5	Me
4x00041	INT16[2]	DATE_TYP_G	Date data type G[U:0,T:0,S:1]	20	WORD:1601	D.M.Y:01.06.08,0x1601	Me
4x00042	SPCL FUNCT[44]	BUFFER	Manufacturer specific data	21	LSW:010B 0070 0C01 0141 2300 4	LSB:0B 01 70 00 01 0C 41 01 00 23 1B 44 0(	Me
4x09001	RESI	UINT16	Converter state for meter	STATE	WORD:0003	3,0x0003 -> Values are valid!	Me
4x09002	HEADER	UINT32R	Identification number of meter	ID	LSW:4082,MSW:0254	39075970,0x02544082	Me
4x10001	HEADER	UINT32	Identification number of meter	ID	MSW:0254,4082:LSW	39075970,0x02544082	Me
4x10003	HEADER	UINT32->ASCI	Manufacturer of meter	MANUFACTURER	MSW:004D,414B:LSW	KAM	Me
4x10005	HEADER	UINT16	Version of meter	VERSION	WORD:0002	2,0x0002	Me
4x10006	HEADER	UINT16	Medium of meter	MEDIUM	WORD:0004	4,0x0004 -> Heat-Volume measured at retu	Me
4x10007	HEADER	UINT16	Access of meter	ACCESS	WORD:0012	18,0x0012	Me
4x10008	HEADER	UINT16	Status of meter	STATUS	WORD:0000	0,0x0000	Me
4x10009	RESI	UINT16	Future value of meter	FUTURE	WORD:0000	0,0x0000	Me
4x10010	RESI	UINT16	Communcation state with meter	COMM STATE	WORD:0003	3,0x0003 -> Values are valid!	Me

Look at registers 4x00009 to 4x00013, three temperature values. In the MBUS frame, the temperatures are transmitted as INT32 values with the exponent 10^-2. So in fact this are integer values with two commas: 2812 will mean 28,12°C. Our converter maps this values to FLOAT32 values and automatically shifts the MBUS exponents to display the value based to 10^0. But in some cases you don't want to shift. So we double click on the desired datapoint in the meter configuration and modify the MODBUS exponent in entering the number -2. This means, that we want to multiply the MBUS value by 10^-2.



The result in the MODBUS registers will be a temperature value multiplied by 100 to represent 1/100°C:

MB Register	MBUS datatype	MB datatype	Content	MBUS index	MB value HEX	Current MB value	Meter name
4x00001	BCD8[4]	SINT32	Fabrication number	0	MSW:0026,D1D2:LSW	2544082,0x0026D1D2	Meter 0254
4x00003	INT32[4]	UINT32	Energy:10^4 Wh	1	MSW:000A,1ED0:LSW	663248,0x000A1ED0	Meter 0254
4x00005	INT32[4]	FLOAT32	Volume:10^-1 m³	2	MSW:45AC,6E66:LSW	5517.7998,5.51779980468750E+3	Meter 0254
4x00007	INT32[4]	UINT32	On time:hours	3	MSW:0000,B64B:LSW	46667,0x0000B64B	Meter 0254
4x00009	INT32[4]	FLOAT32	Flow temperature:10^-2 °C->*10^-2	4	MSW:4540,3000:LSW	3075.0000,3.07500000000000E+3	Meter 0254
4x00011	INT32[4]	FLOAT32	Return temperature:10^-2 °C->*10^-2	5	MSW:43CA,8000:LSW	405.0000,4.05000000000000E+2	Meter 0254
4x00013	INT32[4]	FLOAT32	Temperature difference:10^-2 K->*10^-1	6	MSW:4385,8000:LSW	267.0000,2.67000000000000E+2	Meter 0254
4x00015	INT32[4]	FLOAT32	Power:10^2 W	7	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	Meter 0254

If we change now the MODBUS data type to SINT16 we use only one MODBUS register for every temperature, but because the original MBUS value is based to 10^-2, we have to set the MODBUS exponent to 0. This is different to a FLOAT32 or DOUBLE64 MODBUS register, where the gateway always normalize the MBUS value to 10^0.

Index	MBUS dataty...	MB datatype	Content	MBUS data	MBUS size	MBUS exponent	MB exponent
0	BCD8	SINT32	Fabrication number	1-2	4	0	0
1	INT32	UINT32	Energy:10^4 Wh	1-8	4	4	0
2	INT32	FLOAT32	Volume:10^-1 m³	1-14	4	-1	0
3	INT32	UINT32	On time:hours	1-20	4	0	0
4	INT32	SINT16	Flow temperature:10^-2 °C	1-26	4	-2	0
5	INT32	SINT16	Return temperature:10^-2 °C	1-32	4	-2	0
6	INT32	SINT16	Temperature difference:10^-2 K	1-38	4	-2	0
7	INT32	FLOAT32	Power:10^2 W	1-44	4	2	0
8	INT32	FLOAT32	Power:10^2 W	1-50	4	2	0
9	INT32	FLOAT32	Volume flow:10^-3 m³/h	1-56	4	-3	0
10	INT32	FLOAT32	Volume flow:10^-3 m³/h	1-62	4	-3	0
11	INT32	FLOAT32	Volume:10^-1 m³[U:1,T:0,S:0]	1-69	4	-1	0
12	INT32	FLOAT32	Volume:10^0 m³[U:2,T:0,S:0]	1-77	4	0	0
13	INT32	DATE_TIME_T...	Time&Date data type F	1-83	4	0	0
14	INT32	UINT32	Energy:10^4 Wh[U:0,T:0,S:1]	1-89	4	4	0
15	INT32	FLOAT32	Volume:10^-1 m³[U:0,T:0,S:1]	1-95	4	-1	0
16	INT32	FLOAT32	Power:10^2 W[U:0,T:0,S:1]	1-101	4	2	0
17	INT32	FLOAT32	Volume flow:10^-3 m³/h[U:0,T:0,S:1]	1-107	4	-3	0
18	INT32	FLOAT32	Volume:10^-1 m³[U:1,T:0,S:1]	1-114	4	-1	0
19	INT32	FLOAT32	Volume:10^0 m³[U:2,T:0,S:1]	1-122	4	0	0
20	INT16	DATE_TYP_G	Date data type G[U:0,T:0,S:1]	1-128	2	0	0
21	SPCL FUNCT	BUFFER	Manufacturer specific data	1-131	44	0	0

Download and test the new configuration, you will see the difference:

MB Register	MBUS datatype	MB datatype	Content	MBUS index	MB value HEX	Current MB value	M
4x00001	BCD8[4]	SINT32	Fabrication number	0	MSW:0026,D1D2:LSW	2544082,0x0026D1D2	M
4x00003	INT32[4]	UINT32	Energy:10^4 Wh	1	MSW:000A,1ED0:LSW	663248,0x000A1ED0	M
4x00005	INT32[4]	FLOAT32	Volume:10^-1 m³	2	MSW:45AC,6E66:LSW	5517.7998,5.51779980468750E+3	M
4x00007	INT32[4]	UINT32	On time:hours	3	MSW:0000,B64B:LSW	46667,0x0000B64B	M
4x00009	INT32[4]	SINT16	Flow temperature:10^-2 °C	4	WORD:0C04	3076,0x0C04	M
4x00010	INT32[4]	SINT16	Return temperature:10^-2 °C	5	WORD:0196	406,0x0196	M
4x00011	INT32[4]	SINT16	Temperature difference:10^-2 K	6	WORD:0A6E	2670,0x0A6E	M
4x00012	INT32[4]	FLOAT32	Power:10^2 W	7	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	M
4x00014	INT32[4]	FLOAT32	Power:10^2 W	8	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	M

Now we want to store the temperatures only with one comma in the 16 bit holding registers. Therefore we use the MODBUS exponent to divide the values by 10. So we enter an exponent of 1 for all tree temperatures:

Index	MBUS dataty...	MB datatype	Content	MBUS data	MBUS size	MBUS exponent	MB exponent
0	BCD8	SINT32	Fabrication number	1-2	4	0	0
1	INT32	UINT32	Energy:10^4 Wh	1-8	4	4	0
2	INT32	FLOAT32	Volume:10^-1 m³	1-14	4	-1	0
3	INT32	UINT32	On time:hours	1-20	4	0	0
4	INT32	SINT16	Flow temperature:10^-2 °C	1-26	4	-2	1
5	INT32	SINT16	Return temperature:10^-2 °C	1-32	4	-2	1
6	INT32	SINT16	Temperature difference:10^-2 K	1-38	4	-2	1
7	INT32	FLOAT32	Power:10^2 W	1-44	4	2	0
8	INT32	FLOAT32	Power:10^2 W	1-50	4	2	0
9	INT32	FLOAT32	Volume flow:10^-3 m³/h	1-56	4	-3	0
10	INT32	FLOAT32	Volume flow:10^-3 m³/h	1-62	4	-3	0
11	INT32	FLOAT32	Volume:10^-1 m³[U:1,T:0,S:0]	1-69	4	-1	0
12	INT32	FLOAT32	Volume:10^0 m³[U:2,T:0,S:0]	1-77	4	0	0
13	INT32	DATE_TIME_T...	Time&Date data type F	1-83	4	0	0
14	INT32	UINT32	Energy:10^4 Wh[U:0,T:0,S:1]	1-89	4	4	0
15	INT32	FLOAT32	Volume:10^-1 m³[U:0,T:0,S:1]	1-95	4	-1	0
16	INT32	FLOAT32	Power:10^2 W[U:0,T:0,S:1]	1-101	4	2	0
17	INT32	FLOAT32	Volume flow:10^-3 m³/h[U:0,T:0,S:1]	1-107	4	-3	0
18	INT32	FLOAT32	Volume:10^-1 m³[U:1,T:0,S:1]	1-114	4	-1	0
19	INT32	FLOAT32	Volume:10^0 m³[U:2,T:0,S:1]	1-122	4	0	0
20	INT16	DATE_TYP_G	Date data type G[U:0,T:0,S:1]	1-128	2	0	0
21	SPCL FUNCT	BUFFER	Manufacturer specific data	1-131	44	0	0

Download and test again, the result will look like this:

MB Register	MBUS datatype	MB datatype	Content	MBUS index	MB value HEX	Current MB value	M
4x00001	BCD8[4]	SINT32	Fabrication number	0	MSW:0026,D1D2:LSW	2544082,0x0026D1D2	M
4x00003	INT32[4]	UINT32	Energy:10^4 Wh	1	MSW:000A,1ED0:LSW	663248,0x000A1ED0	M
4x00005	INT32[4]	FLOAT32	Volume:10^-1 m³	2	MSW:45AC,6E66:LSW	5517.7998,5.51779980468750E+3	M
4x00007	INT32[4]	UINT32	On time:hours	3	MSW:0000,B64B:LSW	46667,0x0000B64B	M
4x00009	INT32[4]	SINT16	Flow temperature:10^-2 °C->/10^1	4	WORD:0133	307,0x0133	M
4x00010	INT32[4]	SINT16	Return temperature:10^-2 °C->/10^1	5	WORD:0028	40,0x0028	M
4x00011	INT32[4]	SINT16	Temperature difference:10^-2 K->/10^1	6	WORD:010B	267,0x010B	M
4x00012	INT32[4]	FLOAT32	Power:10^2 W	7	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	M
4x00014	INT32[4]	FLOAT32	Power:10^2 W	8	MSW:0000,0000:LSW	0.0000,0.00000000000000E+0	M

So in general note the following mapping rules:

1. Using FLOAT32, FLOAT32R, DOUBLE64, DOUBLE64R as a MODBUS register type for a MBUS value always forces a normalization of the original MBUS value to base 10^0 to represent a value according to the defined unit of the MBUS value, to which we are used (°C or Wh or m³, etc.)
2. You can now multiply or divide this normalized value by entering a MBUS exponent. A negative exponent will divide the value be the factor 10^exponent, a positive exponent will multiply the value by the factor 10^exponent before the data is written to the MODBUS register.
3. Using other MODBUS data types, the original MBUS data is taken without normalization. Then the MBUS exponent only informs you, what the basis for your value is.
4. But again you can multiply or divide the values by entering an exponent into the MODBUS exponent field manually, before the value is written to the MODBUS register. Enter a positive exponent to multiply the original value by 10^exponent, enter a negative exponent to divide the value by 10^exponent.

16.17 Additional MODBUS register & coils

Here you will find only the additional MODBUS registers and coils especially for this IO module. Please refer to the description of of the standard MODBUS mapping for more details about the available basic MODBUS registers and coils.

Please refer to the external document for detailed documentation of the current MODBUS register mapping for this IO module:

RESI-L-MBUSx-SIO-ETH-MODBUS+ASCII-ENxx.pdf

16.17.1 MODBUS register for meter data

This registers are compatible to our old versions of the product (RESI-MBUSx-MODBUS and RESI-MBUSx-ETH). For the mapped MBUS data the converter uses the MODBUS holding registers starting at 4x00001.

Register	Description
4x00001 3x00001 I:0 R/O MODBUS MAPPING	First holding register of MBUS data mapping for first configured MBUS meter
...	
4x00040 3x00040 I:39 R/O MODBUS MAPPING	Last holding register of MBUS data mapping for last configured MBUS meter for products RESI-MBUS2-SIO RESI-MBUS2-ETH
…	
4x00400 3x00400 I:399 R/O MODBUS MAPPING	Last holding register of MBUS data mapping for last configured MBUS meter for products RESI-MBUS8-SIO RESI-MBUS8-ETH
...	
4x01000 3x01000 I:999 R/O MODBUS MAPPING	Last holding register of MBUS data mapping for last configured MBUS meter for products RESI-MBUS24-SIO RESI-MBUS24-ETH
...	
4x01200 3x01200 I:1199 R/O MODBUS MAPPING	Last holding register of MBUS data mapping for last configured MBUS meter for products RESI-MBUS48-SIO RESI-MBUS48-ETH
...	
4x01200 3x01200 I:1199 R/O MODBUS MAPPING	Last holding register of MBUS data mapping for last configured MBUS meter for products RESI-MBUS64-SIO RESI-MBUS64-ETH

16.17.2 MODBUS status register for meters

This status registers are compatible to our old versions of the product (RESI-MBUSx-MODBUS and RESI-MBUSx-ETH), but the position is shifted to the MODBUS range starting at 4x09001.

Register	Description
4x09001 3x09001 I:9000 R/O STATE METER 1	Returns the current state of the communication with the meter #1 =0: Meter isn't configured =1: Meter isn't normalized =2: Meter isn't read =3: Values are valid
4x09002-3 3x09002-3 I:9001-2 R/O SERIAL NUMBER METER 1	UINT32R: Returns the current serial number of the meter #1 as a 32 bit unsigned integer value 1st.WORD: lower 16 bit of the serial number 2nd.WORD: higher 16 bit of the serial number
4x09004 3x09004 I:9003 R/O STATE METER 2	UINT16: Returns the current state of the communication with the meter #2 =0: Meter isn't configured =1: Meter isn't normalized =2: Meter isn't read =3: Values are valid
4x09005-6 3x09005-6 I:9004-5 R/O SERIAL NUMBER METER 2	UINT32R: Returns the current serial number of the meter #2 as a 32 bit unsigned integer value 1st.WORD: lower 16 bit of the serial number 2nd.WORD: higher 16 bit of the serial number
…	
4x09022 3x09022 I:9021 R/O STATE METER 8	UINT16: Returns the current state of the communication with the meter #8 =0: Meter isn't configured =1: Meter isn't normalized =2: Meter isn't read =3: Values are valid
4x09023-24 3x09024-24 I:9022-23 R/O SERIAL NUMBER METER 8	UINT32R: Returns the current serial number of the meter #8 as a 32 bit unsigned integer value 1st.WORD: lower 16 bit of the serial number 2nd.WORD: higher 16 bit of the serial number
...	

Register	Description
4x09070 3x09070 I:9069 R/O STATE METER 24	UINT16: Returns the current state of the communication with the meter #24 =0: Meter isn't configured =1: Meter isn't normalized =2: Meter isn't read =3: Values are valid
4x09071-72 3x09071-72 I:9070-71 R/O SERIAL NUMBER METER 24	UINT32R: Returns the current serial number of the meter #24 as a 32 bit unsigned integer value 1st.WORD: lower 16 bit of the serial number 2nd.WORD: higher 16 bit of the serial number
...	
4x09142 3x09142 I:9141 R/O STATE METER 48	UINT16: Returns the current state of the communication with the meter #48 =0: Meter isn't configured =1: Meter isn't normalized =2: Meter isn't read =3: Values are valid
4x09143-144 3x09143-144 I:9142-143 R/O SERIAL NUMBER METER 48	UINT32R: Returns the current serial number of the meter #48 as a 32 bit unsigned integer value 1st.WORD: lower 16 bit of the serial number 2nd.WORD: higher 16 bit of the serial number
…	
4x09190 3x09190 I:9189 R/O STATE METER 64	UINT16: Returns the current state of the communication with the meter #64 =0: Meter isn't configured =1: Meter isn't normalized =2: Meter isn't read =3: Values are valid
4x09191-92 3x09191-92 I:9190-91 R/O SERIAL NUMBER METER 64	UINT32R: Returns the current serial number of the meter #64 as a 32 bit unsigned integer value 1st.WORD: lower 16 bit of the serial number 2nd.WORD: higher 16 bit of the serial number

16.17.3 MODBUS extended status register for meters

This extended status registers are new to our new version of the product (RESI-MBUSx-SIO and RESI-MBUSx-ETH). For every meter there is a set of 10 MODBUS holding registers starting with 4x10001. Mainly this registers represent the information of the MBUS fixed data header:

Ident. Nr.	Manufr.	Version	Medium	Access No.	Status	Signature
4 Byte	2 Byte	1 Byte	1 Byte	1 Byte	1 Byte	2 Byte

This header is sent by many answer frames of the MBUS meter to the master. Due to the fact, that is is not part of the variable data block of the meter, our old converters could not map this information to registers. Our new series map this information to the following register set starting at 4x10001. For each meter there are eight MODBUS entries:

Register	**Description**
4x10001-2 3x10001-2 I:10000-1 R/O ID NUMBER METER 1	UINT32: Returns the current serial number of the meter #1 as a 32 bit unsigned integer value 1st.WORD: higher 16 bit of the serial number 2nd.WORD: lower 16 bit of the serial number Each meter offers a unique ID. In the MBUS protocol there are four bytes reserved for this number. In our gateway we need a UINT32 to represent this 4 bytes of the ID.
4x10003-4 3x10003-4 I:10002-3 R/O MANUFACTURER METER 1	UINT32->ASCII: Returns the current manufacturer of the meter #1 as a 32 bit unsigned integer value 1st.WORD: higher 16 bit of the manufacturer name as ASCII text 2nd.WORD: lower 16 bit of of the manufacturer name as ASCII text Each meter offers a manufacturer ID, represented in two bytes. But in this two bytes there are three ASCII digits encoded. Our gateway decode this ASCII digits and stores this digits into a UINT32 using ASCII encoding with 0x00 at the end representing a standard null terminated ASCII string of three letters. For example the manufacturer KAMSTRUM uses KAM with the bytes 0x4B 0x41 0x4D. This will be represented by 32 bit value : 0x004D414B, so the higher WORD will be 0x004D and the lower word will be 0x414B.
4x10005 3x10005 I:10004 R/O VERSION METER 1	UINT16: Returns the current version of the meter #1 In the fixed data header, there is also a version number encoded into one byte. It represents the version of the meter. Our gateway stores this byte into a UINT16 holding register for easy readout.
4x10006 3x10006 I:10005 R/O MEDIUM METER 1	UINT16: Returns the current medium of the meter #1 In the fixed data header, there is also a medium number encoded into one byte. it defines what type of medium the meter is measuring. Our gateway stores this byte into a UINT16 holding register for easy readout. The following medium types are defined by the standard for meters with fixed+variable data structure: 0x00: OTHER, 0x01: OIL, 0x02: Electricity, 0x03: Gas, 0x04: Heat-Volume measured at return temperature outlet, 0x05: Steam, 0x06: Hot Water, 0x07: Water, 0x08: H.C.A.=Heat Cost Allocator, 0x09: Compressed Air, 0x0A: Cooling load meter Volume measured at return temperature outlet, 0x0B: Cooling load meter Volume measured at flow temperature inlet, 0x0C: Heat Volume measured at flow temperature inlet, 0x0D: Heat/Cooling load meter, 0x0E: Bus/System, 0x0F: Unknown Medium, 0x16: Cold Water, 0x17: Dual Water, 0x18: Pressure, 0x19: A/D Converter For meters with fixed data structure only, the 16 bit value must be interpreted in another way. Refer to the MBUS standard for this definition.

Register	Description
4x10007 3x10007 I:10006 R/O ACCESS COUNTER METER 1	UINT16: Returns the current access counter of the meter #1 In the fixed data header, there is also an access counter encoded into one byte. It will be incremented by every access of the meter data. So each readout of the meter will increment this access counter by 1 in the range from 0 to 255. Our gateway stores this byte into a UINT16 holding register for easy readout.
4x10008 3x10008 I:10007 R/O STATUS METER 1	UINT16: Returns the current status of the meter #1 In the fixed data header, there is also a status field encoded into one byte. It shows the current meter status. Our gateway stores this byte into a UINT16 holding register for easy readout. The byte has the following meaning: Bit 1+Bit 0: =00 (0) NO ERROR Bit 1+Bit 0: =10 (1) APPLICATION NOT READY Bit 1+Bit 0: =01 (2) APPLICATION ERROR Bit 1+Bit 0: =11 (3) RESERVED Bit 2: =1: POWER LOW, =0: POWER OK Bit 3: =1: PERMANENT ERROR, =0: NO PERMANENT ERROR Bit 4: =1: TEMPORARY ERROR, =0: NO TEMPORARY ERROR Bit 5: =1: MANUFACTURER SPECIFIC ERROR 1, =0: NO MANUFACTURER SPECIFIC ERROR 1 Bit 6: =1: MANUFACTURER SPECIFIC ERROR 2, =0: NO MANUFACTURER SPECIFIC ERROR 2 Bit 7: =1: MANUFACTURER SPECIFIC ERROR 3, =0: NO MANUFACTURER SPECIFIC ERROR 3
4x10009 3x10009 I:10008 R/O FUTURE VALUE METER 1	UINT16: Returns a future value of the meter #1 This UINT16 holding register is reserved for future use.
4x10010 3x10010 I:10009 R/O COMMUNICATION STATE METER 1	UINT16: Returns the current state of the communication with the meter #1 This UINT16 holding register hold the current state of the communication between the MBUS gateway and the meter with the following states: **=0 - Meter isn't configured!**: This value shows, that this meter slot is currently not configured in the MBUS gateway **=1 - Meter isn't normalized!**: This value shows, that the configured meter doesn't answer to the addressing command. Either via primary addressing or via secondary addressing mode. This depends, how the meter was configured **=2 - Meter isn't read!**: This value shows, that the configured meter has answered to the addressing command but there are problems by reading all data from the meter. So the meter data is not valid any more **=3 - Values are valid!**: This value shows, that the configured meter has answered to the addressing command and has answered correctly to the readout commands and the reading of all data from the meter was successful. So the meter data in the MODBUS register is valid **=1000..65535 – meter readout is asynchron!**: If the mapping of the received MBUS frame differ to the mapping from the configuration, this value shows the position of the first asynchron received data: Value=1000+MBUS record*1000+MBUS byte Index within record. If MBUS record number >=64, the received value is always 1000+64*1000+MBUS byte index.

Register	Description
4x10011-12 3x10011-12 I:10010-11 R/O ID NUMBER METER 2	UINT32: Returns the current serial number of the meter #2 as a 32 bit unsigned integer value Refer to meter #1 description
4x10013-14 3x10013-14 I:10012-13 R/O MANUFACTURER METER 2	UINT32->ASCII: Returns the current manufacturer of the meter #2 as a 32 bit unsigned integer value Refer to meter #1 description
4x10015 3x10015 I:10014 R/O VERSION METER 2	UINT16: Returns the current version of the meter #2 Refer to meter #1 description
4x10016 3x10016 I:10015 R/O MEDIUM METER 2	UINT16: Returns the current medium of the meter #2 Refer to meter #1 description
4x10017 3x10017 I:10016 R/O ACCESS COUNTER METER 2	UINT16: Returns the current access counter of the meter #2 Refer to meter #1 description
4x10018 3x10018 I:10017 R/O STATUS METER 2	UINT16: Returns the current status of the meter #2 Refer to meter #1 description
4x10019 3x10019 I:10018 R/O FUTURE VALUE METER 2	UINT16: Returns a future value of the meter #2 Refer to meter #1 description
4x10020 3x10020 I:10019 R/O COMMUNICATION STATE METER 2	UINT16: Returns the current state of the communication with the meter #2 Refer to meter #1 description
...	

Register	Description
...	
4x10071-72 3x10071-72 I:10070-71 R/O ID NUMBER METER 8	UINT32: Returns the current serial number of the meter #8 as a 32 bit unsigned integer value Refer to meter #1 description
4x10073-74 3x10073-74 I:10072-73 R/O MANUFACTURER METER 8	UINT32->ASCII: Returns the current manufacturer of the meter #8 as a 32 bit unsigned integer value Refer to meter #1 description
4x10075 3x10075 I:10074 R/O VERSION METER 8	UINT16: Returns the current version of the meter #8 Refer to meter #1 description
4x10076 3x10076 I:10075 R/O MEDIUM METER 8	UINT16: Returns the current medium of the meter #8 Refer to meter #1 description
4x10077 3x10077 I:10076 R/O ACCESS COUNTER METER 8	UINT16: Returns the current access counter of the meter #8 Refer to meter #1 description
4x10078 3x10078 I:10077 R/O STATUS METER 8	UINT16: Returns the current status of the meter #8 Refer to meter #1 description
4x10079 3x10079 I:10078 R/O FUTURE VALUE METER 8	UINT16: Returns a future value of the meter #8 Refer to meter #1 description
4x10080 3x10080 I:10079 R/O COMMUNICATION STATE METER 8	UINT16: Returns the current state of the communication with the meter #8 Refer to meter #1 description
...	

Register	Description
...	
4x10231-232 3x10231-232 I:10230-231 R/O ID NUMBER METER 24	UINT32: Returns the current serial number of the meter #24 as a 32 bit unsigned integer value Refer to meter #1 description
4x10233-234 3x10233-234 I:10232-233 R/O MANUFACTURER METER 24	UINT32->ASCII: Returns the current manufacturer of the meter #24 as a 32 bit unsigned integer value Refer to meter #1 description
4x10235 3x10235 I:10234 R/O VERSION METER 24	UINT16: Returns the current version of the meter #24 Refer to meter #1 description
4x10236 3x10236 I:10235 R/O MEDIUM METER 24	UINT16: Returns the current medium of the meter #24 Refer to meter #1 description
4x10237 3x10237 I:10236 R/O ACCESS COUNTER METER 24	UINT16: Returns the current access counter of the meter #24 Refer to meter #1 description
4x10238 3x10238 I:10237 R/O STATUS METER 24	UINT16: Returns the current status of the meter #24 Refer to meter #1 description
4x10239 3x10239 I:10238 R/O FUTURE VALUE METER 24	UINT16: Returns a future value of the meter #24 Refer to meter #1 description
4x10240 3x10240 I:10239 R/O COMMUNICATION STATE METER 24	UINT16: Returns the current state of the communication with the meter #24 Refer to meter #1 description
...	

Register	Description
...	
4x10471-472 3x10471-472 I:10470-471 R/O ID NUMBER METER 48	UINT32: Returns the current serial number of the meter #48 as a 32 bit unsigned integer value Refer to meter #1 description
4x10473-474 3x10473-474 I:10472-473 R/O MANUFACTURER METER 48	UINT32->ASCII: Returns the current manufacturer of the meter #48 as a 32 bit unsigned integer value Refer to meter #1 description
4x10475 3x10475 I:10474 R/O VERSION METER 48	UINT16: Returns the current version of the meter #48 Refer to meter #1 description
4x10476 3x10476 I:10475 R/O MEDIUM METER 48	UINT16: Returns the current medium of the meter #48 Refer to meter #1 description
4x10477 3x10477 I:10476 R/O ACCESS COUNTER METER 48	UINT16: Returns the current access counter of the meter #48 Refer to meter #1 description
4x10478 3x10478 I:10477 R/O STATUS METER 48	UINT16: Returns the current status of the meter #48 Refer to meter #1 description
4x10479 3x10479 I:10478 R/O FUTURE VALUE METER 48	UINT16: Returns a future value of the meter #48 Refer to meter #1 description
4x10480 3x10480 I:10479 R/O COMMUNICATION STATE METER 48	UINT16: Returns the current state of the communication with the meter #48 Refer to meter #1 description
...	

Register	Description
...	
4x10631-632 3x10631-632 I:10630-631 R/O ID NUMBER METER 64	UINT32: Returns the current serial number of the meter #64 as a 32 bit unsigned integer value Refer to meter #1 description
4x10633-634 3x10633-634 I:10632-633 R/O MANUFACTURER METER 64	UINT32->ASCII: Returns the current manufacturer of the meter #64 as a 32 bit unsigned integer value Refer to meter #1 description
4x10635 3x10635 I:10634 R/O VERSION METER 64	UINT16: Returns the current version of the meter #64 Refer to meter #1 description
4x10636 3x10636 I:10635 R/O MEDIUM METER 64	UINT16: Returns the current medium of the meter #64 Refer to meter #1 description
4x10637 3x10637 I:10636 R/O ACCESS COUNTER METER 64	UINT16: Returns the current access counter of the meter #64 Refer to meter #1 description
4x10638 3x10638 I:10637 R/O STATUS METER 64	UINT16: Returns the current status of the meter #64 Refer to meter #1 description
4x10639 3x10639 I:10638 R/O FUTURE VALUE METER 64	UINT16: Returns a future value of the meter #64 Refer to meter #1 description
4x10640 3x10640 I:10639 R/O COMMUNICATION STATE METER 64	UINT16: Returns the current state of the communication with the meter #64 Refer to meter #1 description

16.17.4 MODBUS registers for special configuration

This registers hold special information for the converter:

Register	Description
4x65231 3x65231 I:65230 R/W MBUS BAUDRATE	UINT16: The baud rate for the MBUS interface. Parity is always EVEN, ONE stop bit is used. The following baud rates are available: 300,600,900,1200,2400,4800,9600,19200,38400,57600 All other values are interpreted as 2400 baud. HINT: After writing a new value to this register a reboot is necessary to activate the new settings
4x65232 3x65232 I:65231 R/W MBUS QUERY TIMEOUT	UINT16: The query timeout for the MBUS polling process. This value defines the timeout between two query cycles in the gateway. Usually the gateway communicates with all configured meters sequentially. After finishing the data readout for the last meter, the gateway pauses for this defined interval in seconds. This values are used: Value 65535 or values 0..5 defines ~5s pause. Values 6 to 65534: defines 6 to 65534 seconds of pause, before the next polling cycle will start. HINT: After writing a new value to this register a reboot is necessary to activate the new settings
4x65233 3x65233 I:65232 R/W MBUS POLL DELAY	UINT16: The poll delay for the MBUS polling process. This value defines a general pause after the readout of a configured meter before the readout of the next meter starts. In the past we discovered that there are many meters out in the market, which need a special treatment in the timing. e.g. very old KAMSTRUP meters need often two readout cycles with a gap of at least 10-15 seconds. This is non standard to the MBUS. Or other meters have problems with secondary addressing, if there is a too small gap between the readout. So we introduced this new parameter: This timeout defines the pause after finishing reading of a meter and starting reading the next meter. In the previous firmware versions this timeout was fixed to 250ms gap, which was ok for 99% of the meter readout on the markets. But some meter fail to process this little gap. The values is interpreted as follows: Value 1..30: Gap time 1 seconds to 30 seconds Value 101..400: Gaptime=(Value-100)*0.1s → 0.1s .. 30s e.g. 105 → 0.5s Value 65535: Gap time is 1 second Value 65534: Gap time is 250ms Value 65533: Gap time is 500ms Value 65532: Gap time is7250ms All other values: Gap time is 1000ms HINT: After writing a new value to this register a reboot is necessary to activate the new settings

16.18 Additional ASCII commands

Here you will find only the additional ASCII commands especially for this IO module. Please refer to the description of of the standard commands for more details about the available basic ASCII commands.

Please refer to the external document for detailed documentation of the current ASCII commands for this IO module:

RESI-L-MBUSx-SIO-ETH-MODBUS+ASCII-ENxx.pdf