MANUAL

...feel the spirit of a new generation

RESI-IO-MODBUS

Our IO modules with
MODBUS/RTU slave interface
or ASCII slave interface

RESI Informatik & Automation GmbH	Date:	**22.07.2016**	Client:		Pages
	Version:	**1.60**	Title:	**RESI-IO Manual**	**370**
	Edited by:	**DI HC Sigl**			
	Reviewed by:	**DI HC Sigl**	Project:		
	Reviewed by:	**-**			

1 History

Date	Editor	Description
30.05.15	DI HC SIGL, MSc	First version with RESI-2RI-xxx
17.09.15	DI HC SIGL, MSc	Adding RESI-1RO-xxx module
20.10.15	DI HC SIGL, MSc	Adding RESI-1LED-xxx module
03.12.15	DI HC SIGL, MSc	Adding RESI-4LED-xxx module
10.12.15	DI HC SIGL, MSc	Adding RESI-16DI8RO-xxx module
22.02.16	DI HC SIGL, MSc	Adding RESI-2RTD-xxx module
12.04.16	DI HC SIGL, MSc	Adding RESI-32DI-xxx, RESI-14RI-xxx module
22.07.16	DI HC SIGL, MSc	Adding RESI-8CO-xxx module

2 Content

3 IMPORTANT SECURITY NOTES

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

Pay attention to the following rules:

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

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

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

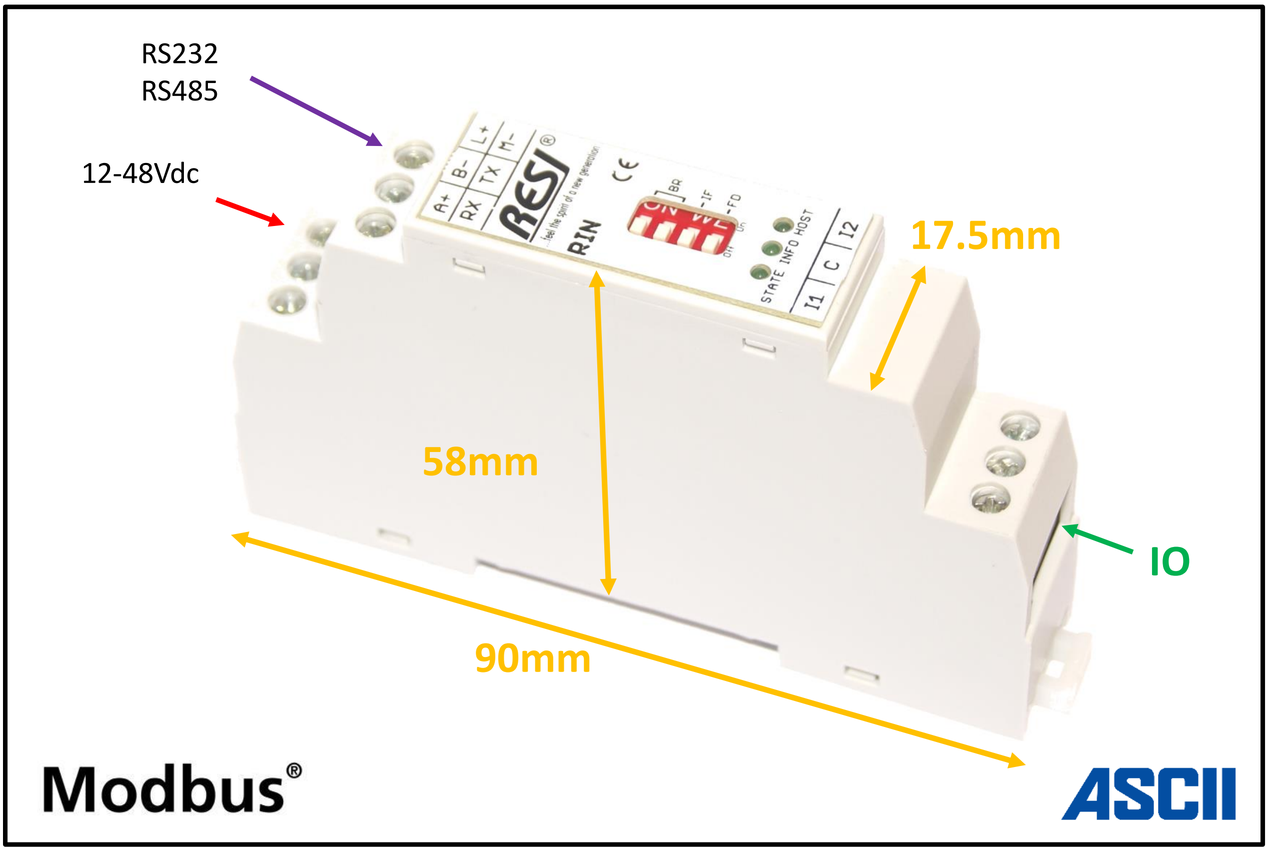
4 General Information

In this manual you will find a description to our IO modules, all equipped with a MODBUS/RTU slave interface or ASCII protocol. We offer in general two different series of products

- Ultra slim IO modules for applications in remote areas with a few IOs
- Compact IO modules with many IOs for signal interfacing in central locations

Here you find an overview about our ultra slim IO modules:

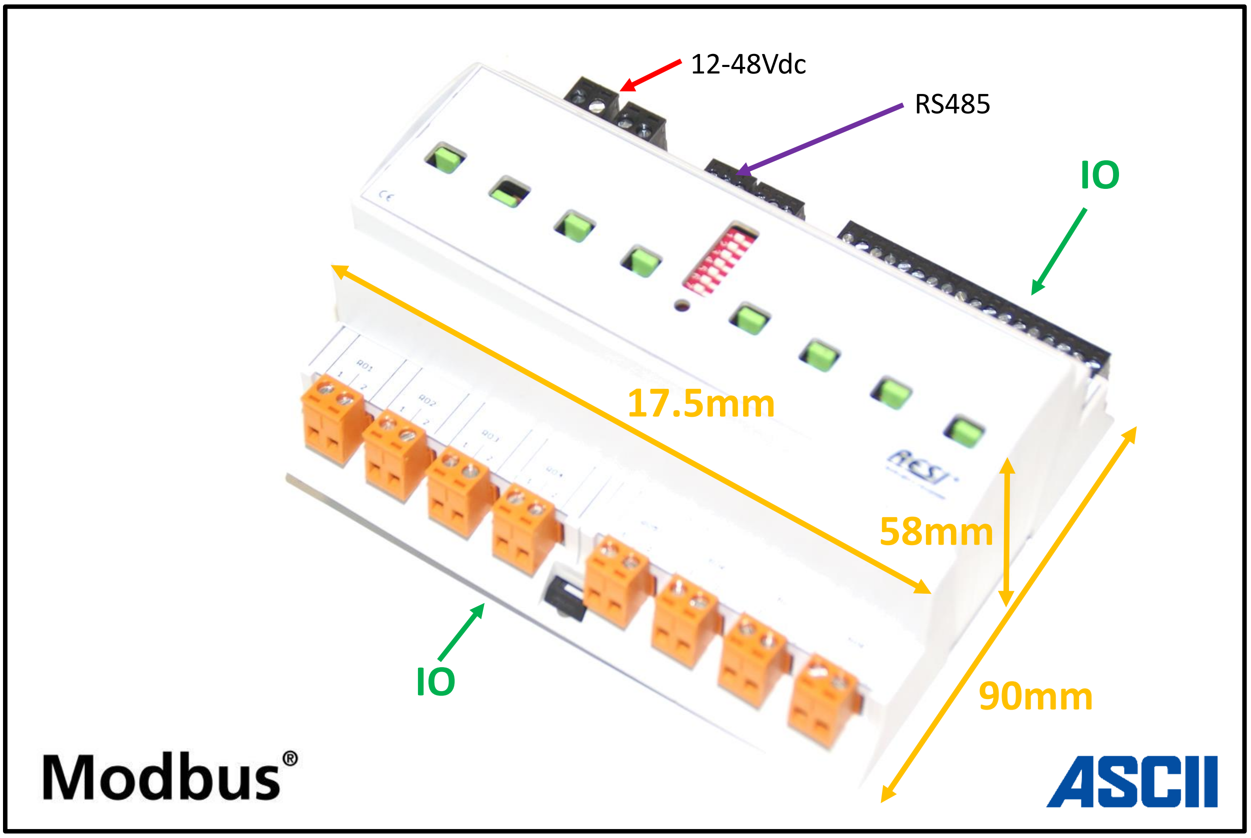
- Extreme small and slim IO modules: Only 17.5x90x58mm in size!
- Integrated RS232 and RS485 serial interface
- RESI-xx-MODBUS: MODBUS/RTU slave
- RESI-xx-ASCII: MODBUS/RTU slave and ASCII slave
- Configuration and testing with our free software MODBUSConfigurator
- DIP switch for setup of baud rate, interface type and unit id
- Baud rates: 9600, 19200, 38400 or 57600baud, 8 data bits, 1 stop bit
- Parity: none, even or odd parity
- LED indication for HOST communication, INFO and STATE
- Ideal for standard switchboard or distribution cabinets with 45mm cut-out
- Wide range power supply: 12-48Vdc
- Mounting onto an EN 50022 DIN rail



PRODUCT	HIGHLIGHTS	CLAMPS
RESI-2RI-MODBUS RESI-2RI-ASCII	2 digital inputs for 10-250Vac/dc Galvanic insulation	I1,I2: digital inputs #1 and #2 C: common ground
RESI-1CO-MODBUS RESI-1CO-ASCII	1 relay output with 3 clamps Relay: 250Vac, max. 8A or 30Vdc, max. 8A Galvanic insulation	NO,C,NC: connectors for the relay
RESI-4AI-MODBUS RESI-4AI-ASCII	4 analogue inputs for 0..10V or -10..+10V Resolution: 16 bits Galvanic insulation	AI1,AI2,AI3,AI4: analogue inputs #1 to #4 C: Common Ground
RESI-4AO-MODBUS RESI-4AO-ASCII	4 analogue outputs for 0..10V or -10..+10V Resolution: 12 bits Galvanic insulation	AO1,AO2,AO3,AO4: analogue outputs #1 to #4 C: Common Ground
RESI-2AI2AO-MODBUS RESI-2AI2AO-ASCII	2 analogue inputs for 0..10V and 2 analogue outputs for 0..10V Resolution: 12 bits Galvanic insulation	AI1,AI2: analogue inputs #1 and #2 AO1,AO2: analogue outputs #1 and #2 C: Common Ground
RESI-2RTD-MODBUS RESI-2RTD-ASCII	2 temperature sensor inputs for PT10, PT50, PT100, PT1000 sensors 2 wire or 3 wire connection of sensor Precision: <0.1°C Galvanic insulation	T1A;T1B,T1C: temperature sensor input #1 T2A;T2B,T2C: temperature sensor input #2
RESI-1LED-MODBUS RESI-1LED-ASCII	RGB-LED stripes dimming module Galvanic insulation	IN+,IN-: Supply of the LED stripe 10-60Vdc O+: Common anode of the LED stripe O1,O2,O3: output signals for RGB dimming

Here you find an overview about our powerful compact IO modules:

- Extreme powerful IO modules, Size: 140x90x58mm
- Integrated RS485 serial interface via removable plugs
- RESI-xx-MODBUS: MODBUS/RTU slave
- RESI-xx-ASCII: MODBUS/RTU slave and ASCII slave
- Configuration and testing with our free software MODBUSConfigurator
- DIP switch for setup of unit address, baud rate and parity
- Baud rates: 4800, 9600, 19200, 38400, 57600, 115200, 230400,256000baud, 8 data bits, 1 stop bit
- Parity: none or even parity
- LED indication RED/WHITE for HOST communication
- All inputs and outputs are plugged with color coded plugs
- Ideal for standard switchboard or distribution cabinets with 45mm cut-out
- Wide range power supply: 12-48Vdc via two removable plugs
- Mounting onto an EN 50022 DIN rail or wall mounting



PRODUCT	HIGHLIGHTS	SONSTIGES
RESI-16DI8RO-MODBUS RESI-16DI8RO-ASCII	16 digital inputs for 12-48Vdc 8 bistable relay outputs for light applications Max. 230Vac, max. 16A Galvanic insulation	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 (230 VAC) 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
RESI-32DI-MODBUS RESI-32DI-ASCII	32 digital inputs for 12-48Vdc Galvanic insulation	
RESI-14RI-MODBUS RESI-14RI-ASCII	14 digital inputs for 10-250Vac/dc Galvanic insulation	
RESI-8CO-MODBUS RESI-8CO-ASCII	8 relay outputs with 3 changeover contacts Relay: 250Vac, max. 8A or 30Vdc, max. 8A Galvanic insulation	For each relay NO,C,NC: Connectors for relay
RESI-12AI-MODBUS RESI-12AI-ASCII	12 analogue inputs for 0..10V or -10..+10V Resolution: 16 bits Galvanic insulation	
RESI-12AO-MODBUS RESI-12AO-ASCII	12 analogue outputs for 0..10V or -10..+10V Resolution: 12 bits Galvanic insulation	
RESI-8RTD-MODBUS RESI-8RTD-ASCII	8 temperature inputs for PT10, PT50, PT100, PT1000 sensors 2- wire, 3 wire or 4 wire connection of sensor Precision: <0.1°C Galvanic insulation	
RESI-4LED-MODBUS RESI-4LED-ASCII	Dimming module for 4 RGB LED stripes Galvanic insulation	

5 RESI-2RI-MODBUS, RESI-2RI-ASCII

5.1 Product description

This IO module offers the following features:

- 2 digital inputs for 10…250Vac/dc
- Galvanic insulated RS232/RS485 interface for communication with a host system
- RESI-2RI-MODBUS: MODBUS/RTU slave protocol
- RESI-2RI-ASCII: MODBUS/RTU slave and text based ASCII protocol
- Power supply 12-48Vdc
- DIP switch for setting the baud rates, the type of interface and the bus number
- LED indicator for the communication
- Mounting onto EN50022 DIN rail

Illustration: Our IO module

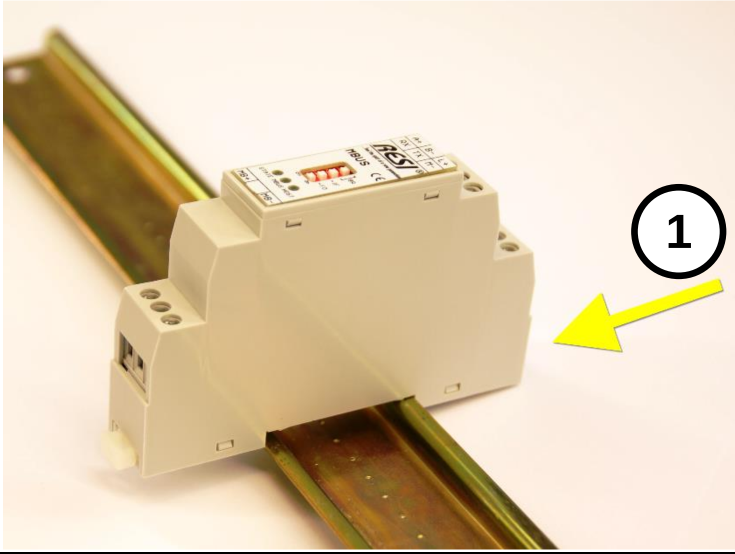
5.2 Technical data

Technical Data			
Power supply			
Supply voltage	12-48V= +/-10%	Storage temperature	-20...85 °C
Power LED	Yes	Operating Temperature	0...60°C
Power consumption	<0.5W	Humidity	25...90 % rH non-condensing
		Protection Class	IP20 (EN 60529)
		Dimensions LxWxH	17,5mm x 90mm x 58mm
		Weight	55g
		Mounting	on DIN EN50022 rail
ASCII/Modbus Interface			
Protocol	ASCII or Modbus/RTU		
Type	RS232 or RS485		
Baud rates	9600 to 57600/8/N or E/1		
Cable Connection	Via clamps		
LED indicator	Yes		
Galvanic insulation	Yes		
Digital inputs			
Number of channels	2		
Signal type	10…250Vac/dc		
Cable connection	Via clamps		
Galvanic insulation to serial interface	Yes		
LED indicator	Yes		
Clamps			
Clamp wire cross section	Max. 1,5 mm²		
Tightening torque	Max. 0.5Nm	**CE conformity**	Yes

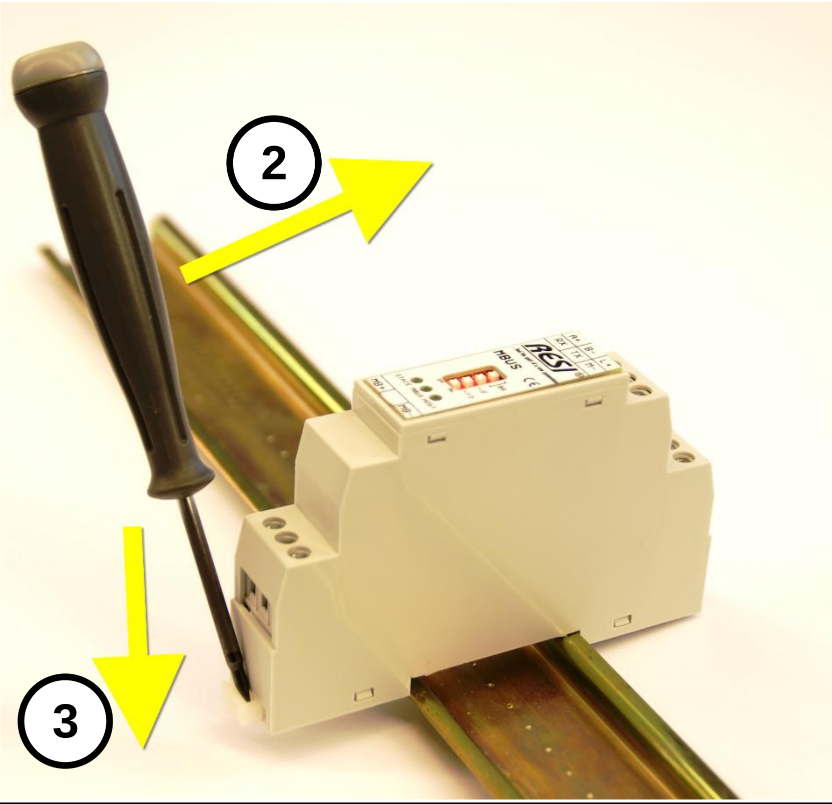
5.3 Assembling

Our IO modules are designed for mounting on a 35mm DIN-EN50022 rail.

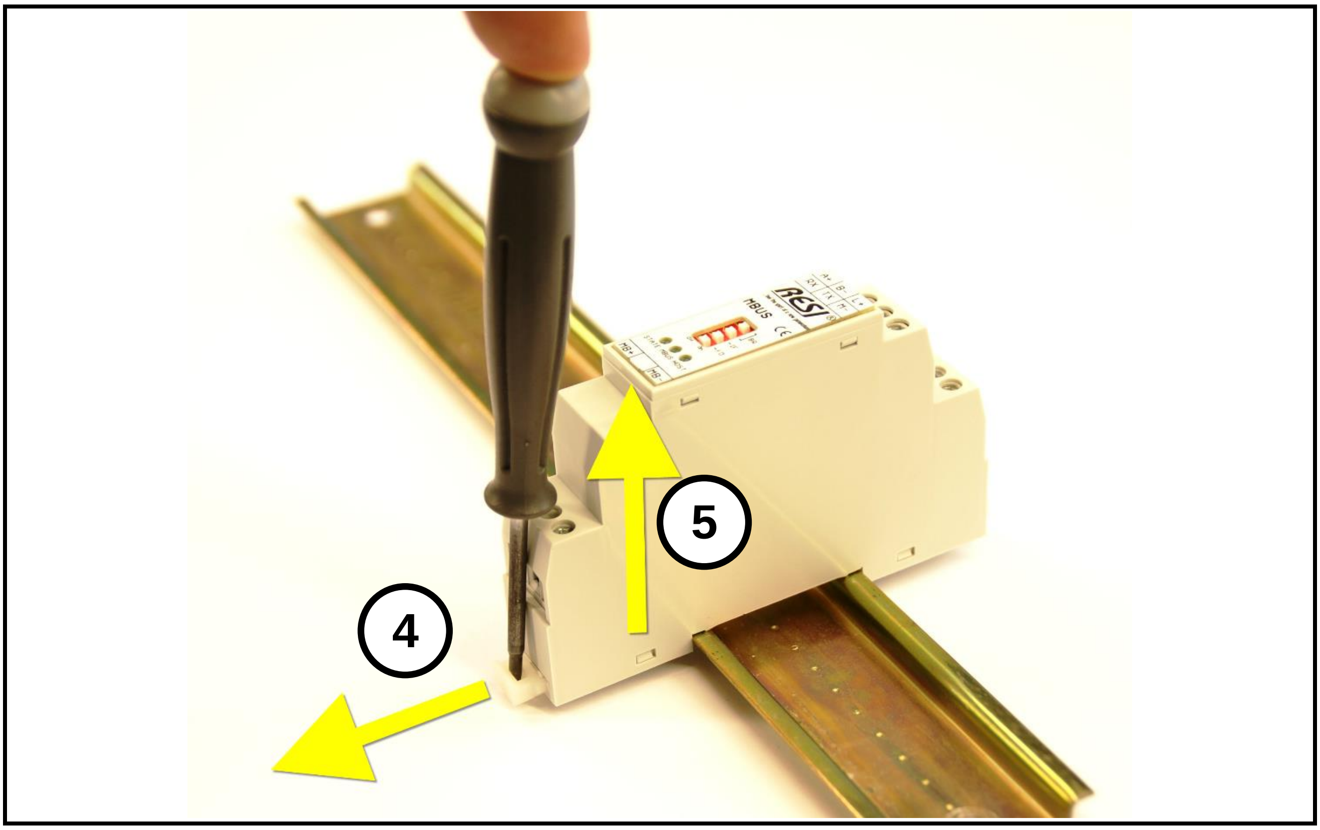
At first, put the modules with the top side on the DIN rail (1).



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.



5.4 Connection diagram

In the below drawing you will find the correct cabling of the IO module.

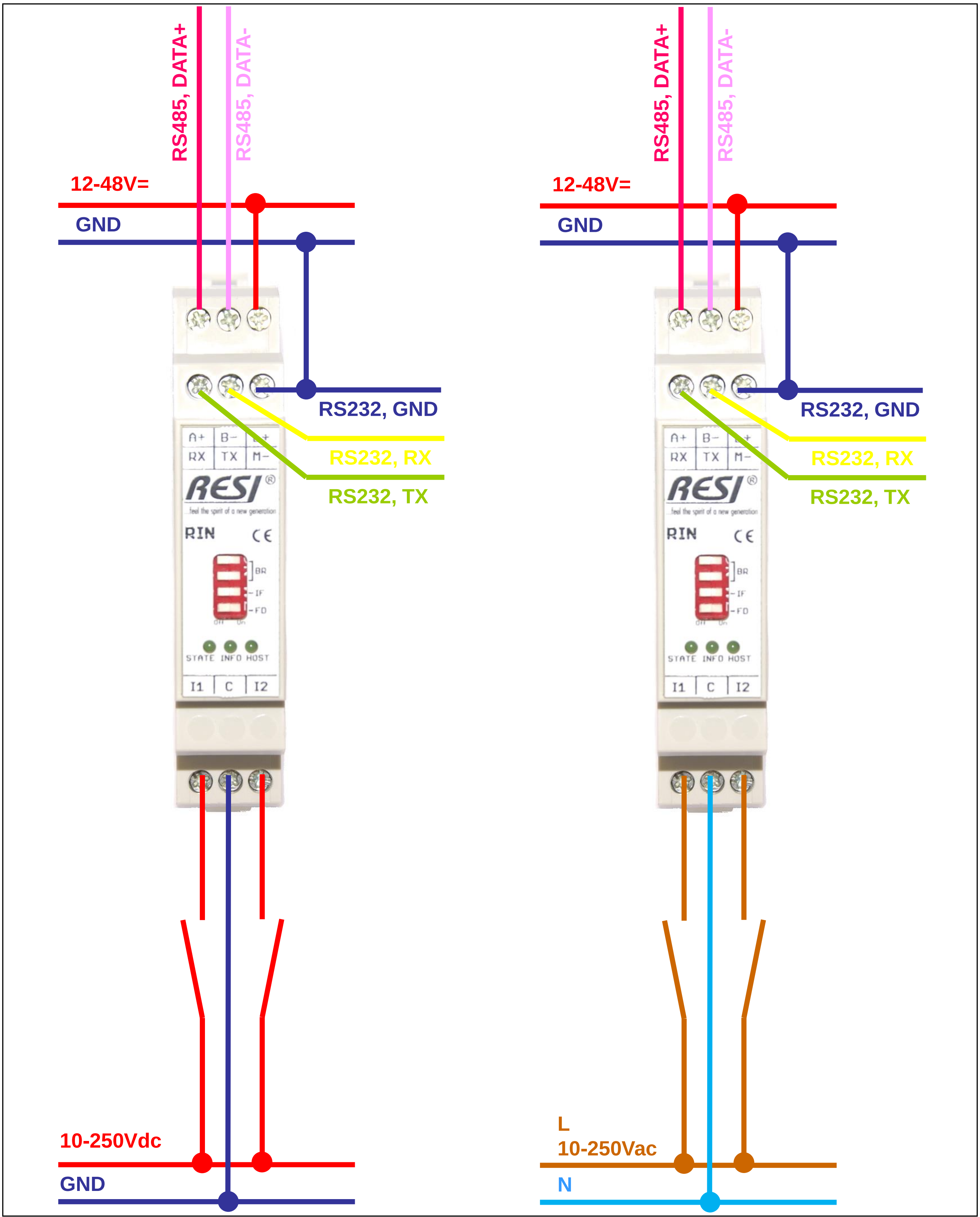


Illustration: Cabling of the IO module

5.5 Clamps

The IO module offers the following clamps:

CLAMP	DESCRIPTION
L+ M-	Power supply: L+: 12-48 V= M-: Ground
RS485 A+ B- M-	RS485 ASCII or MODBUS/RTU interface A+: RS485 DATA+ signal B-: RS485 DATA- signal M-: RS485 ground signal
RS232 TX+ RX- M-	RS232 ASCII or MODBUS/RTU interface TX+: RS232 Transmit signal RX-: RS232 Receive signal M-: RS232 Ground signal
DI C=Common I1 I2	2 digital inputs for 10..250Vac/dc signals C: Common contact for both inputs I1: First digital input 0=Open or GND, 1=10..250Vac/dc related to clamp C I2: Second digital input 0=Open or GND, 1=10..250Vac/dc related to clamp C

Table: Description of the clamps on the IO module

5.6 DIP switch setting and LED indicators

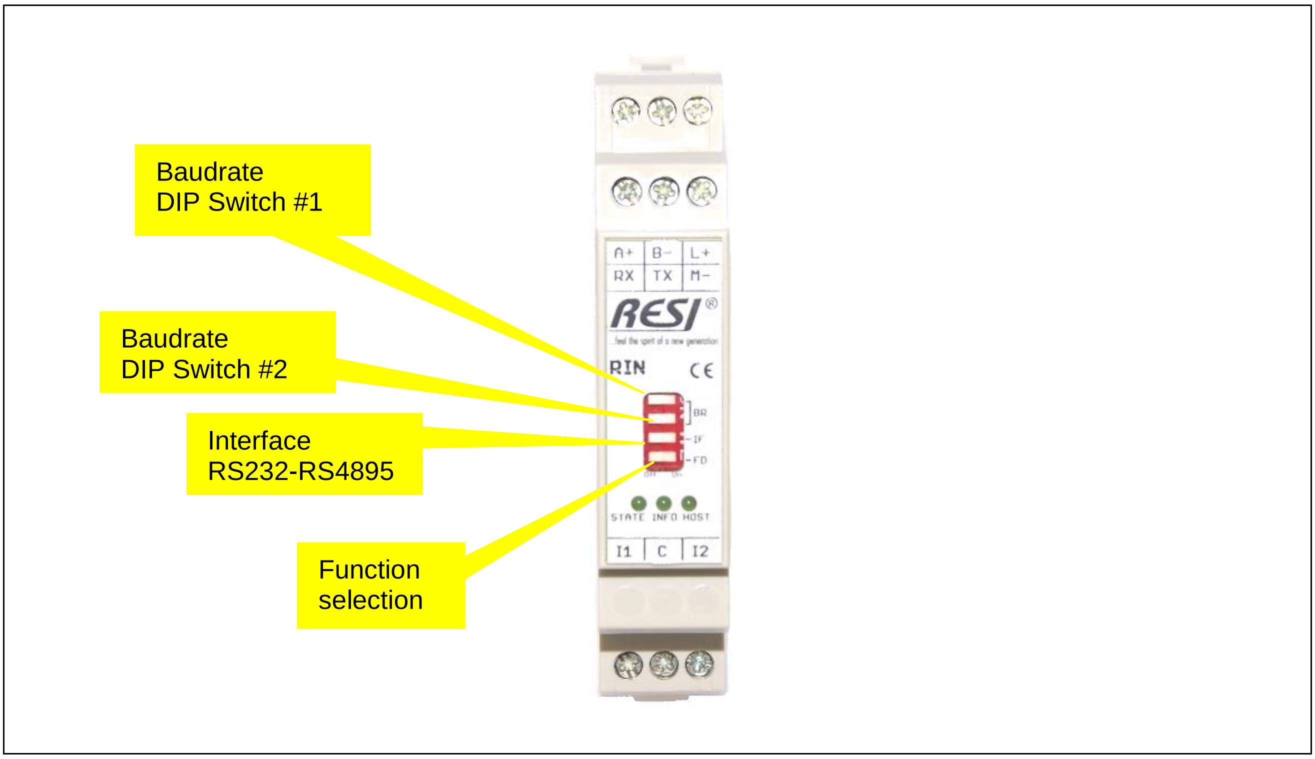


Illustration: Description of the DIP switch settings and LED indicators

DIP Switch	Description
Baudrate BR	Use DIP switches 1+2 to select the baud rate: OFF OFF: 9600Bd ON OFF: 19200Bd OFF ON: 38400Bd ON ON: 57600Bd HINT: The correct parity (NONE, EVEN, ODD) is defined by the PC software, not with the DIP switches.

Interface IF	Selects the physical type of the serial interface for the ASCII or MODBUS/RTU protocol: OFF=RS232 ON=RS485
Function definition FD	Select s special function in the module: OFF=The module uses the configured unit ID from the FLASH memory ON=The module uses always the unit ID 255
HINT	After a change of the DIP switches, the module reboots. No power off / power on cycle is necessary. After the reset all three LEDs are shortly on to represent the RESTART sequence.

Table: Description of the DIP switches of the IO module

LED	Description
STATE	Status LED: If the module is ok, this LED flashes slowly. If there is an error detected by the module, this LED flashes fast.
INFO	If one of the two digital inputs is high, this LED is on. If both digital inputs are low, this LED is off.
HOST	HOST-LED, Flashes, if the host is communicating with the module.

Table: Description of the LED indicator on the IO module

5.7 Dimensions of the module

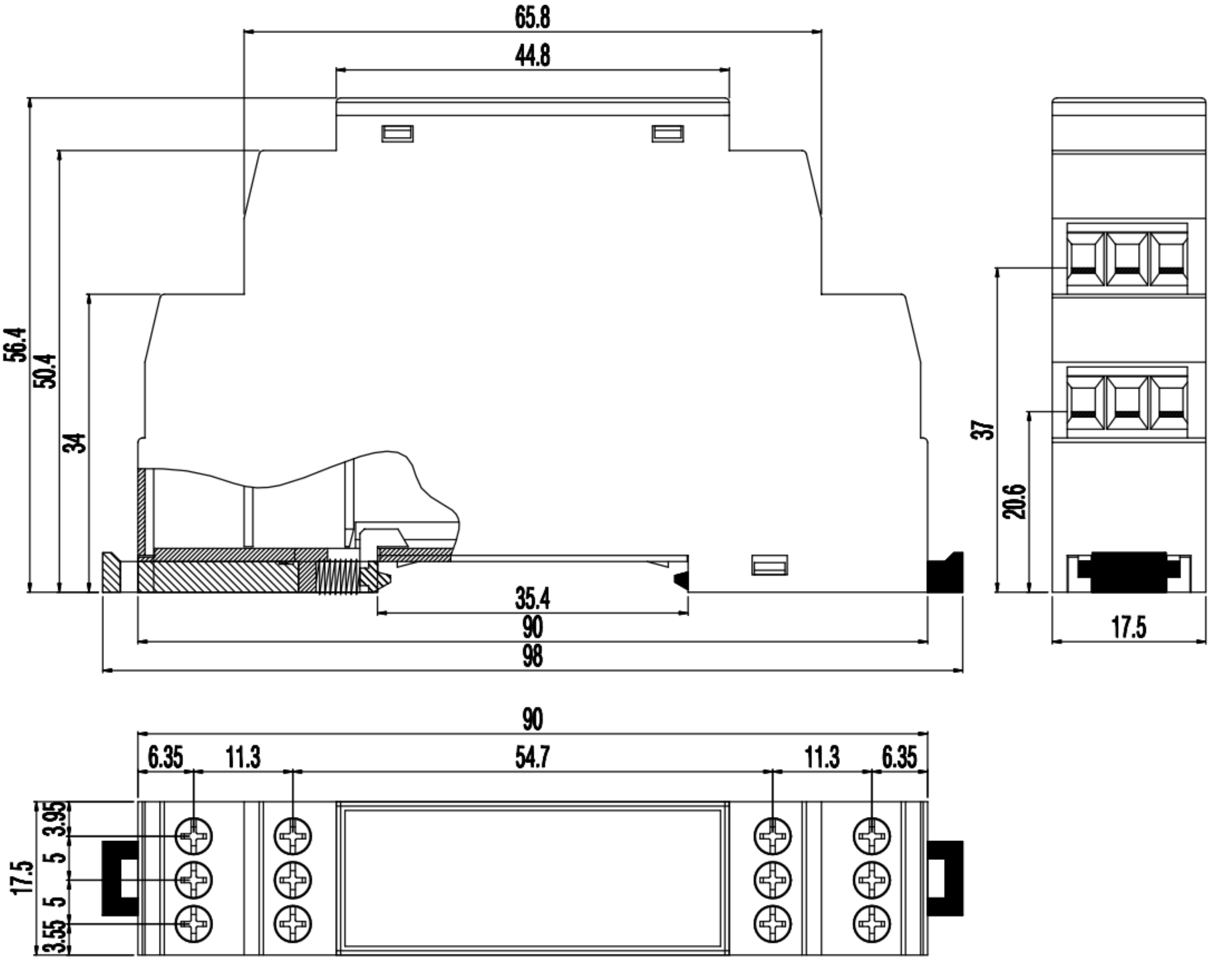


Illustration: dimension illustration in mm

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

Table: Data of enclosure

5.8 3D Drawing

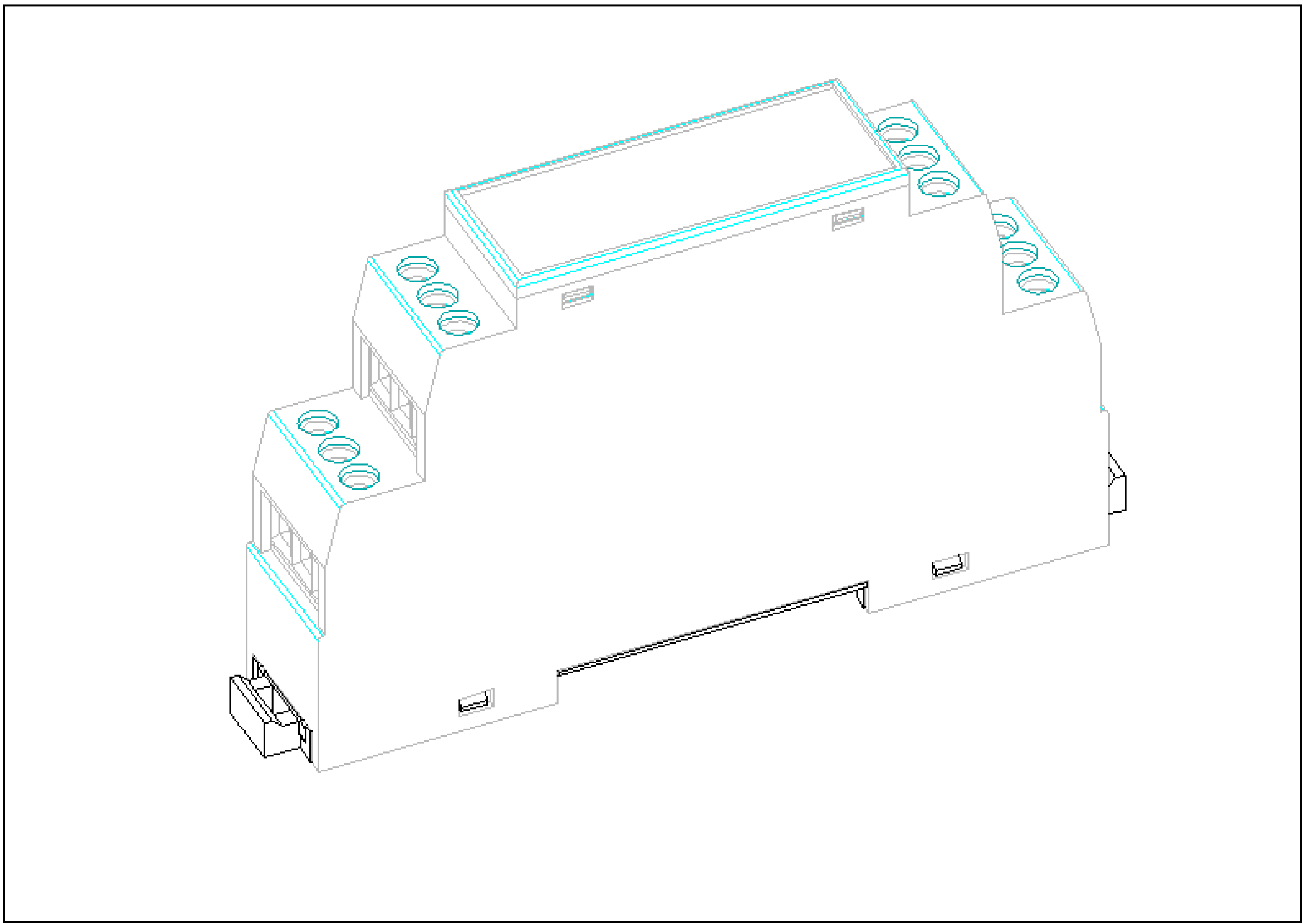


Illustration: 3D drawing of the enclosure

5.9 Power supply cabling of the module

In the image below you will see the correct cabling of the power supply of the module.

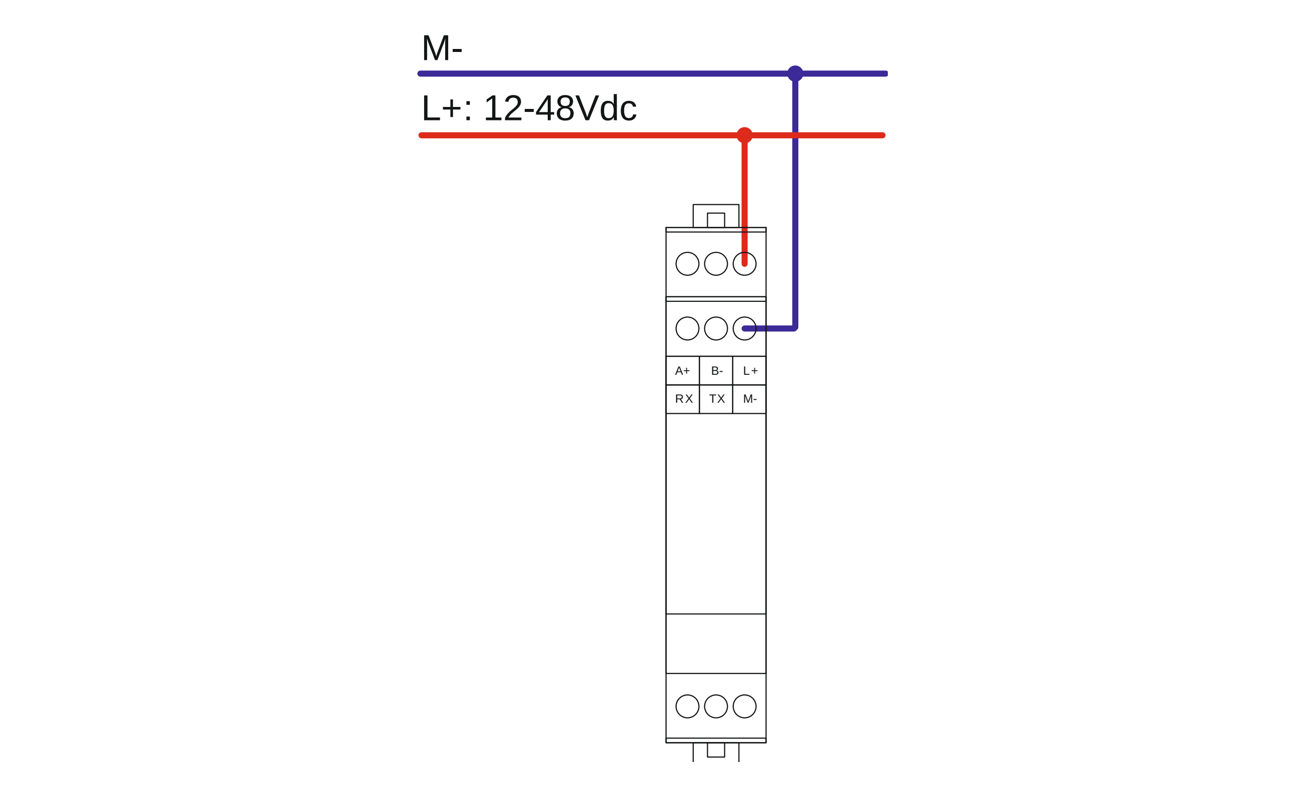


Illustration: Cabling of the power supply of the IO module

5.10 RS485 cabling of the IO module

In the image below you see the correct cabling of the RS485 interface of the IO module.

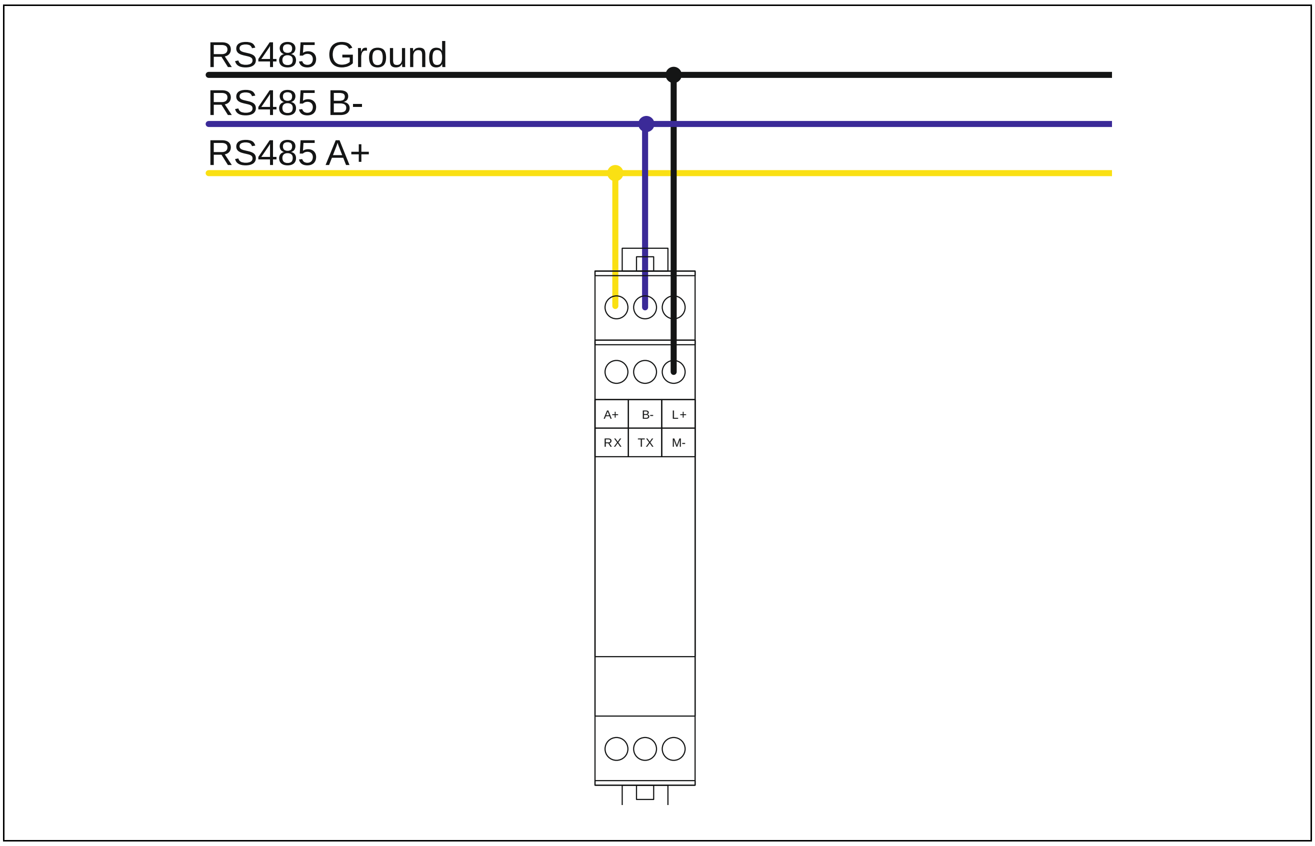


Illustration: RS485 bus cabling of the IO module

5.11 RS232 cabling of the IO module

In the image below you see the correct cabling of the RS232 interface of the IO module.

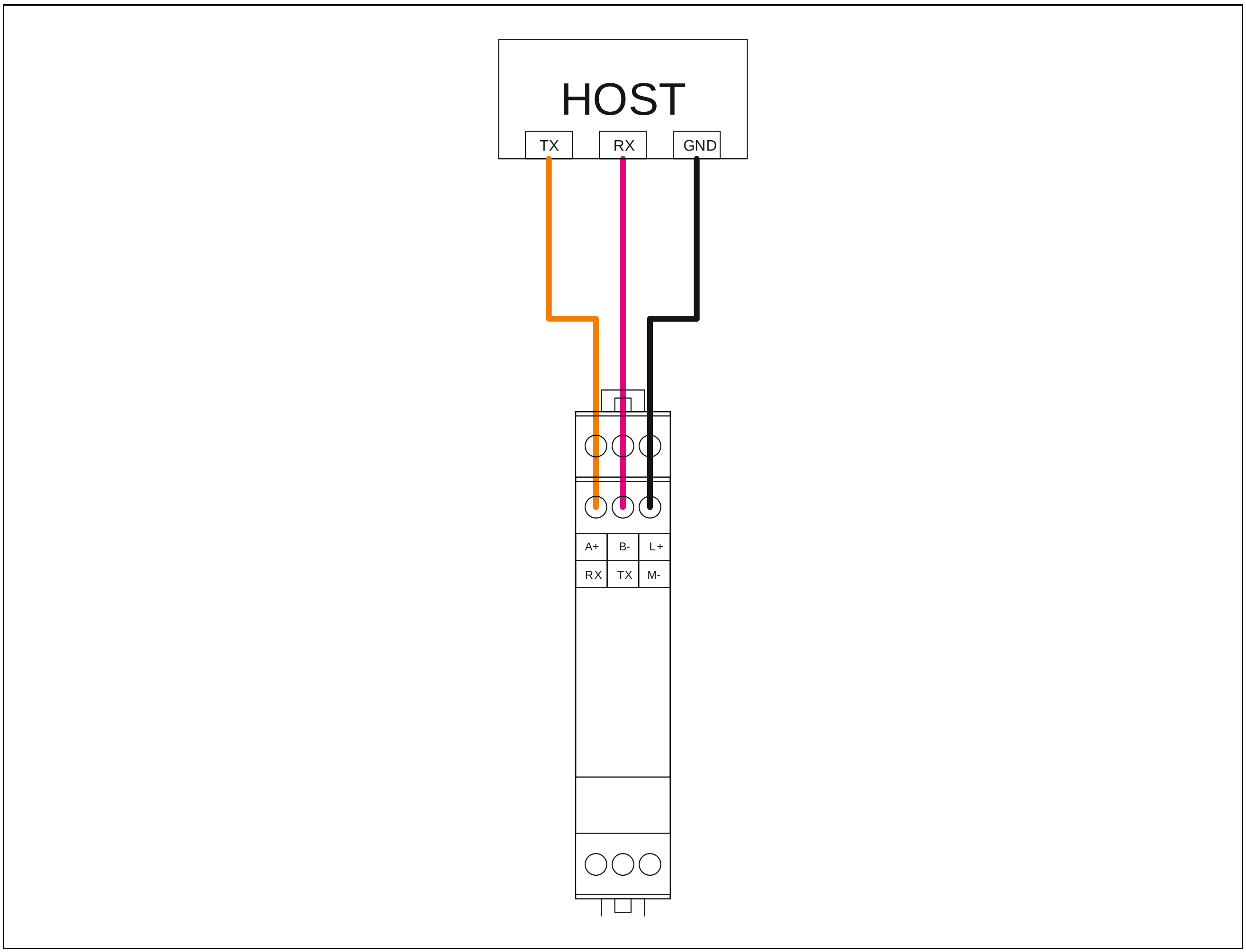


Illustration: RS232 bus cabling of the IO module

5.12 Cabling of the digital inputs of the module

In the image below you see the correct cabling of the two digital inputs for DC signals of the IO module. The clamp C is the common clamp for both digital signals. It's the common internal connection point for both inputs. The IO module consumes approx. 1.9mA current per digital input to avoid signal interferences with other cables.

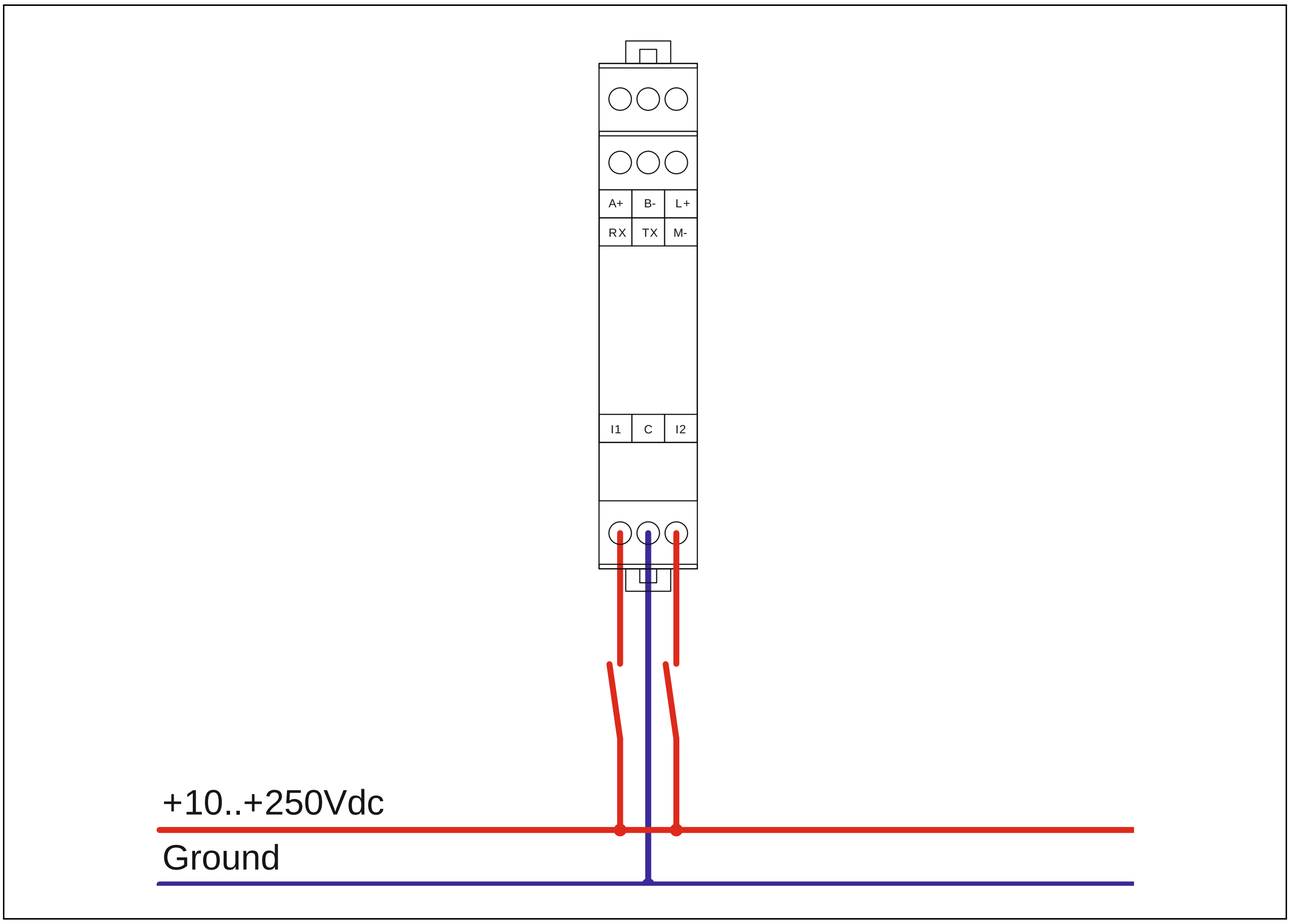


Illustration: Cabling of the digital inputs of the IO module for DC signals

In the image below you see the correct connection of the two digital inputs for AC signals. The clamp C is the common neutral wire. I1 and I2 are the two phase inputs.

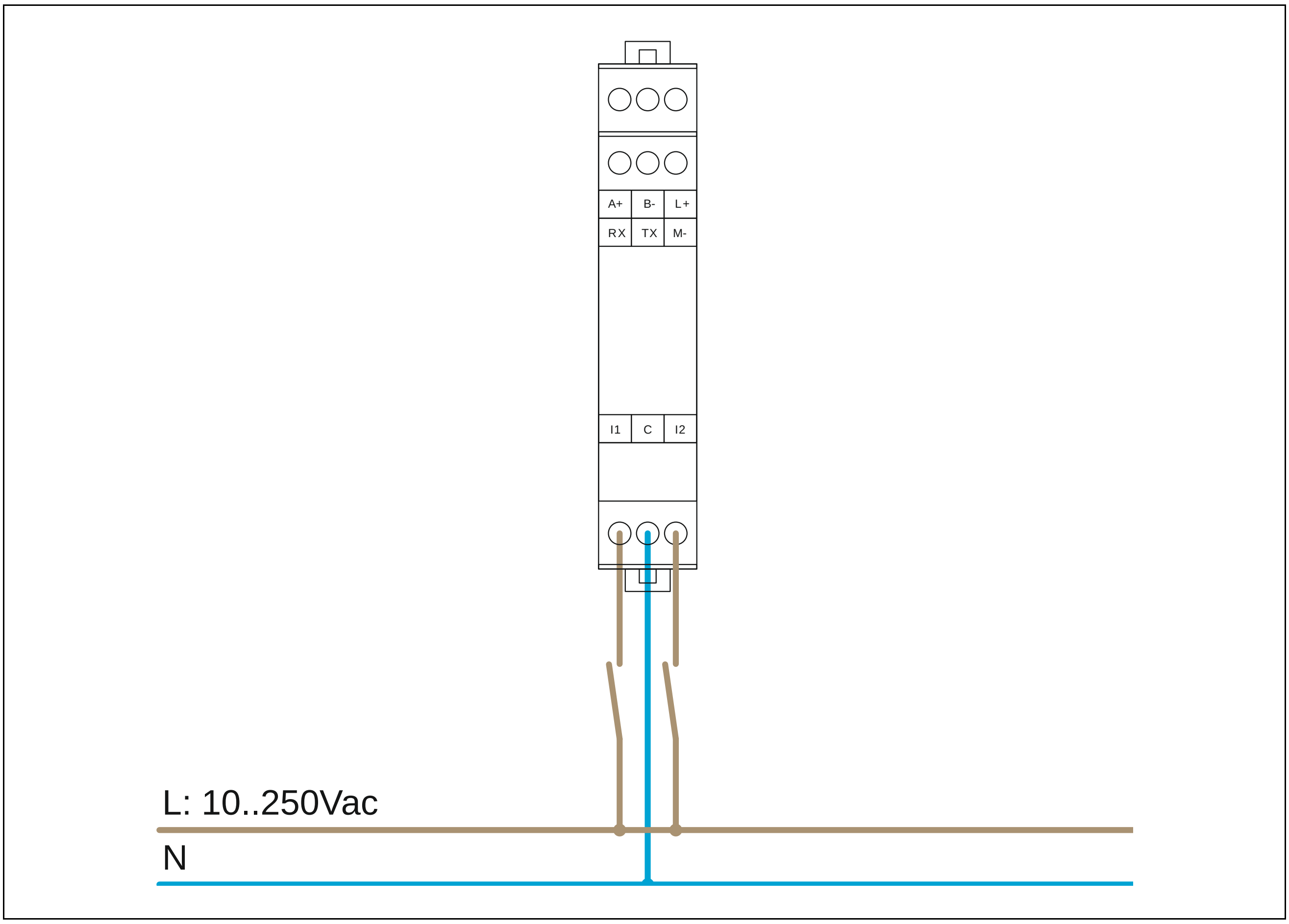


Illustration: Cabling of the digital inputs of the IO module for AC signals

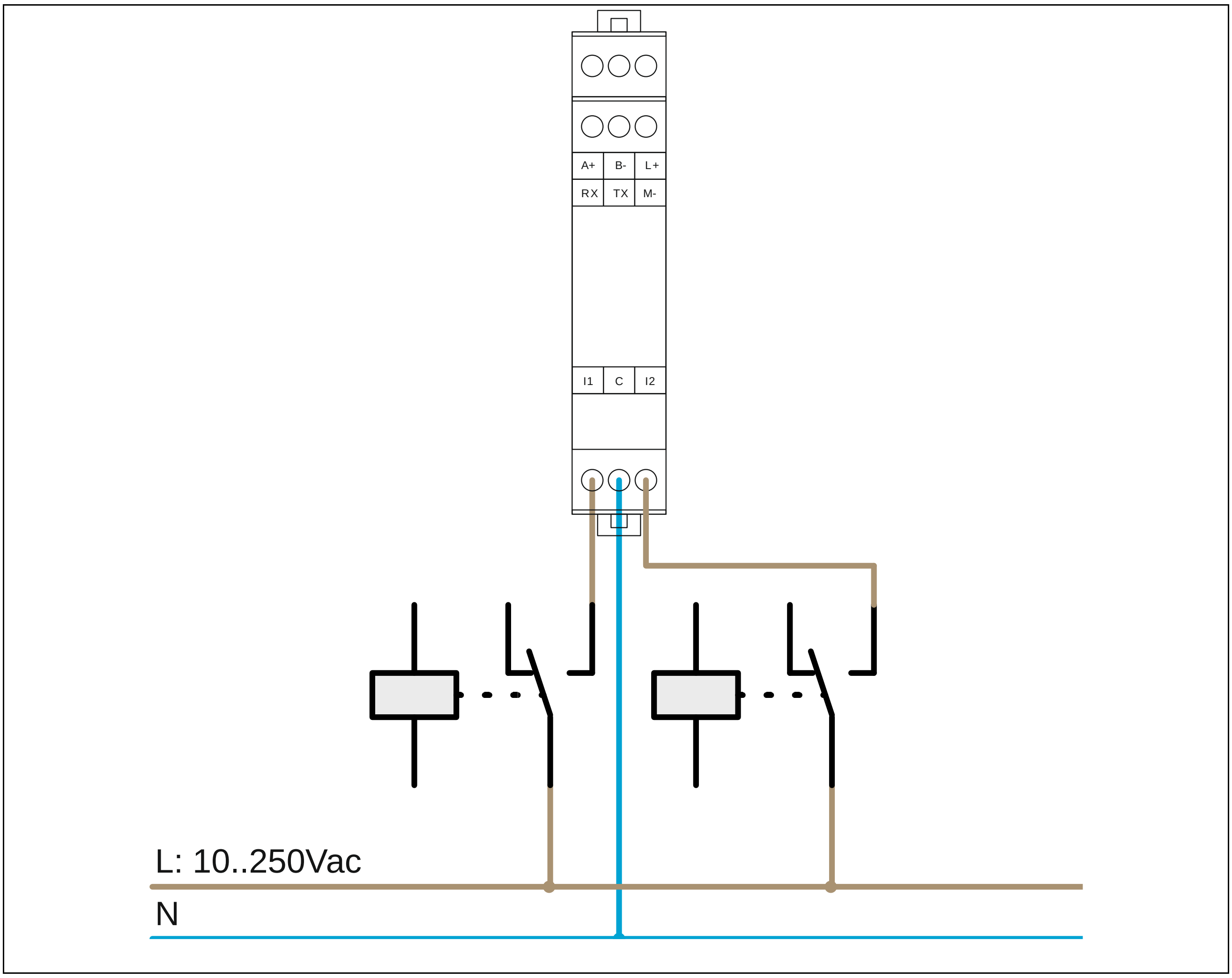


Illustration: Cabling of the two digital inputs of the IO module with two signaling relays

5.13 Functional description

This IO module communicates with a host system with the MODBUS/RTU slave protocol. All versions of the Io module with RESI-xxx-ASCII offer an additional protocol: An ASCII slave protocol with simple text string. The communication runs over a RS485 interface (half duplex) or over a RS232 interface (full duplex).

For the communication with ASCII texts, the host sends ASCII messages with a special start character # (0x23, 35dec) and a special end character (0x0d,13dec or CARRIAGE RETURN) to the module. The module uses also this special start and end characters to answer to the host request. Consult the below noted detailed command descriptions. In the ASCII protocol mode, the host can send messages with or without a bus number.

For communication with the MODBUS/RTU slave, the module offers the following MODBUS functions:

- 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)

HINT:
The functions READ HOLDING REGISTER and PRESET MULTIPLE REGISTERS are restricted to max. 125 register per request! The functions READ INPUT STATUS, READ COIL STATUS and FORCE MULTIPLE COILS are restricted to 2000 coils or inputs (bits) per request.

5.14 ASCII protocol description

5.14.1 Overview

The IO module communicates with simple ASCII commands. The following special characters are used in this documentation:

stand for the **Hashtag** ASCII character 35dec or 0x23
: stand for the **colon** ASCII character 58dec or 0x3A
= stand for the **equal sign** with the ASCII code 61ec or 0x3D
- stand for the **minus sign** with the ASCII code 45dec or 0x2D
, stands for the **comma** with the ASCII code 44dec or 0x2C
<CR> stand for **CARRIAGE RETURN** ASCII character 13dec or 0x0D. In the following text we use the representation CR.
<SP> stands for the **SPACE**. This is the space character with the ASCII code 32dec or 0x20. In the following text we will use the representation☐.
<ADR> will be used as the current **bus address** of the module. The bus number can be transmitted decimal or hexadecimal and is separated with a comma (ASCII code 44dec or 0x2C) from the rest of the command. Hexadecimal numbers always start with 0x. Its only allowed to use the ASCII characters ‚0‘-‚9‘ 48dec to 57dec, 0x30-0x39 and ‚A‘ to ‚F‘, 65dec to 70dec, 0x41-0x46. All modules react to the broadcast bus address 0 and to its own bus number. With a DIP switch, the user can easily change between the internal stored bus number in the FLASH and the fix bus number 255. Consult the DIP switch description for more details.

5.14.2 Communication sequence

In general the IO module sends no ASCII characters without a request from a host. So the host is the master of the communication and the IO module is always answering to host requests as a slave. If only one IO module is connected to a host (e.g. Via RS232), you can dump the bus number in the protocol. If you use a RS485 interface, more than one module can be connected to the host. Therefore a bus number in the request frame of the host is always necessary.

The structure of the command look like this:

The host sends a command or a command with parameters without a bus address to the module:
#<Command><CR> or
#<Command>:<Parameter><CR>

The module answers, if it feels addressed, with the following answer telegram:
#<Answer><CR>

If the bus number is used, the module answers with:
#<ADR>,<Command><CR> or
#<ADR>,<Command>:<Parameter><CR>

The module answers with:
#<ADR>,<Answer><CR>

The bus address lies in the range of 1dec to 255dec or 0x00 to 0xFF hexadecimal. The setup is done with our free configuration software MODBUSConfigurator. Each module reacts also to the broadcast bus address 0.

For each command, we define two different writings. A long version and a short version, to avoid unnecessary traffic on the bus. For example to request the software version of the module you can use the command VERSION or the short command VER.

5.14.3 Request VERSION

This command returns the current software version of the module.

Host long version:
#VERSION<CR> or
#<ADR>,VERSION<CR>

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

Answer:
#VERSION:<HIGH>.<MED>.<LOW><CR> or
#<ADR>,VERSION:<HIGH>,<MED>,<LOW><CR>

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

Samples:
→ **#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$_{CR}$

5.14.4 Request module TYPE

This command returns the current type of the module.

Host long version:
#TYPE<CR> or
#<ADR>,TYPE<CR>

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

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

<TYP> defines the current type of the module. Currently RESI-2RI-ASCII

Samples:
→ #TYPE$_{CR}$
← #TYPE:RESI-2RI-ASCII$_{CR}$

→ #255,TYP$_{CR}$
← #255,TYPE:RESI-2RI-ASCII$_{CR}$

5.14.5 Table of all ASCII commands

Here you will find a possible ASCII commands of the module. We use here only the version with bus number. That you can avoid the bus number, we have discussed earlier in this document. If an argument as the extension Dec, it will be returned as a decimal number, If an argument has the extension Hex, then this argument is returned as a hexadecimal number. Many command returns the argument in decimal and hexadecimal representation. So the host can select, what kind of number conversion, it will handle in its software.

Direction	**ASCII command**
Host	**#<BusAdr>,VER$_{CR}$** **#<BusAdr>,VERSION$_{CR}$**
Answer	**#<BusAdr>,VERSION:<VersionHi>.< VersionMed>.<VersionLo>$_{CR}$**
	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)
Host	**#<BusAdr>,TYP$_{CR}$** **#<BusAdr>,TYPE$_{CR}$**
Answer	**#<BusAdr>,TYPE:RESI-2RI-ASCII$_{CR}$**
	Returns the current type of the module
Host	**#<BusAdr>,OWN$_{CR}$** **#<BusAdr>,OWNER$_{CR}$**
Answer	**#<BusAdr>,OWNER:RESI$_{CR}$**
	Returns the owner of the module
Host	**#<BusAdr>,CRE$_{CR}$** **#<BusAdr>,CREATOR$_{CR}$**
Answer	**#<BusAdr>,CREATOR:DI HC SIGL,MSC$_{CR}$**
	Returns the creator of the module

Direction	ASCII command
Host	**#<BusAdr>,COPY$_{CR}$** **#<BusAdr>,COPYRIGHT$_{CR}$**
Answer	**#<BusAdr>,COPYRIGHT:2015 BY RESI AND DI HC SIGL,MSC WWW.RESI.CC$_{CR}$**
	Returns a copyright note about the module
Host	**#<BusAdr>,GDIP$_{CR}$** **#<BusAdr>,GET□DIP$_{CR}$**
Answer	**#<BusAdr>,GDIP:<DIPSwitchDec>,<DIPSwitchHex>$_{CR}$**
	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)
Host	**#<BusAdr>,GDIS$_{CR}$** **#<BusAdr>,GET□DIS$_{CR}$**
Answer	**#<BusAdr>,GDIS:<DISDec>,<DISHex>$_{CR}$**
	Returns the current state of both digital inputs of the module as decimal number and as hexadecimal number DISDec DISHex The current state of all digital inputs: Bit 0: State I1 (=0:OFF, =1:ON) Bit 1: State I2 (=0:OFF, =1:ON)
Host	**#<BusAdr>,GDI1$_{CR}$** **#<BusAdr>,GET□DI1$_{CR}$**
Answer	**#<BusAdr>,GDI1:<DI1Dec>,<DI1Hex>$_{CR}$**
Host	**#<BusAdr>,GDI2$_{CR}$** **#<BusAdr>,GET□DI2$_{CR}$**
Answer	**#<BusAdr>,GDI2:<DI2Dec>,<DI2Hex>$_{CR}$**
	Returns the current state of digital input Ix as decimal number and as hexadecimal number DIxDec DIxHex The current state of digital input Ix: =0: Digital input is OFF =1: Digital input is ON
Host	**#<BusAdr>,GNDI1$_{CR}$** **#<BusAdr>,GET□NEG□DI1$_{CR}$**
Answer	**#<BusAdr>,GNDI1:<NDI1Dec>,<NDI1Hex>$_{CR}$**
Host	**#<BusAdr>,GNDI2$_{CR}$** **#<BusAdr>,GET□NEG□DI2$_{CR}$**
Answer	**#<BusAdr>,GNDI2:<NDI2Dec>,<NDI2Hex>$_{CR}$**
	Returns the current state of the negated digital input Ix as decimal number and as hexadecimal number NDIxDec NDIxHex The current negated state of digital input Ix: =0: Digital input is ON =1: Digital input is OFF

Direction	ASCII command
Host	**#<BusAdr>,GNDIS$_{CR}$** **#<BusAdr>,GET□NEG□DIS$_{CR}$**
Answer	**#<BusAdr>,GNDIS:<NDISDec>,<NDISHex>$_{CR}$**
	Returns the current negated states of both digital inputs as decimal number and as hexadecimal number NDISDec NDISHex — The current negated state of all digital inputs: Bit 0: Negated state of I1 (=0:ON, =1:OFF) Bit 1: Negated state of I2 (=0:ON, =1:OFF)
Host	**#<BusAdr>,ORDIS$_{CR}$** **#<BusAdr>,OR□DIS$_{CR}$**
Answer	**#<BusAdr>,ORDIS:<ORDISDec>,<ORDISHex>$_{CR}$**
	Returns the result of a bitwise OR operation between the both digital inputs I1 and I2 as decimal number and as hexadecimal number ORDISDec ORDISHex — Bitwise OR operation between I2 and I1: I1 I2 OR OFF OFF =0 OFF ON =1 ON OFF =1 ON ON =1
Host	**#<BusAdr>,XORDIS$_{CR}$** **#<BusAdr>,XOR□DIS$_{CR}$**
Answer	**#<BusAdr>,XORDIS:<XORDISDec>,<XORDISHex>$_{CR}$**
	Returns the result of a bitwise EXCLUSIVE OR operation between the both digital inputs I1 and I2 as decimal number and as hexadecimal number XORDISDec XORDISHex — Bitwise XOR operation between I2 and I1: I1 I2 EXKLUSIVE OR OFF OFF =0 OFF ON =1 ON OFF =1 ON ON =0
Host	**#<BusAdr>,ANDDIS$_{CR}$** **#<BusAdr>,AND□DIS$_{CR}$**
Answer	**#<BusAdr>,ANDDIS:<ANDDISDec>,<ANDDISHex>$_{CR}$**
	Returns the result of a bitwise AND operation between the both digital inputs I1 and I2 as decimal number and as hexadecimal number ANDDISDec ANDDISHex — Bitwise AND operation between I2 and I1: I1 I2 AND OFF OFF =0 OFF ON =0 ON OFF =0 ON ON =1

Direction	ASCII command
Host	**#<BusAdr>,RDI1$_{CR}$** **#<BusAdr>,RISE□DI1$_{CR}$**
Answer	**#<BusAdr>,RDI1:<RDI1Dec>,<RDI1Hex>$_{CR}$**
Host	**#<BusAdr>,RDI2$_{CR}$** **#<BusAdr>,RISE□DI2$_{CR}$**
Answer	**#<BusAdr>,RDI2:<RDI2Dec>,<RDI2Hex>$_{CR}$**
	Returns the current edge counter for the digital input Ix. This counter counts the rising edges of Ix since reset of the module as decimal number and as hexadecimal number RDIxDec RDIxHex The current amount of rising edges on digital input Ix
Host	**#<BusAdr>,FDI1$_{CR}$** **#<BusAdr>,FALL□DI1$_{CR}$**
Answer	**#<BusAdr>,FDI1:<FDI1Dec>,<FDI1Hex>$_{CR}$**
Host	**#<BusAdr>,FDI2$_{CR}$** **#<BusAdr>,FALL□DI2$_{CR}$**
Answer	**#<BusAdr>,FDI2:<FDI2Dec>,<FDI2Hex>$_{CR}$**
	Returns the current edge counter for the digital input Ix. This counter counts the falling edges of Ix since reset of the module as decimal number and as hexadecimal number FDIxDec FDIxHex The current amount of falling edges on digital input Ix
Host	**#<BusAdr>,RC$_{CR}$** **#<BusAdr>,RESET□COUNTERS$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
	Clears all edge counters in the module.

Direction	**ASCII command**
Host	**#<BusAdr>,SMBADR:<MBUnit>$_{CR}$** **#<BusAdr>,SET□MODBUS□ADDRESS:<MBUnit>$_{CR}$**
Answer	**#<BusAdr>,OK $_{CR}$**
	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.
Host	**#<BusAdr>,GMBADR$_{CR}$** **#<BusAdr>,GET□MODBUS□ADDRESS$_{CR}$**
Answer	**#<BusAdr>,GMBADR:<MBUnitDec>,<MBFLASHDec>,<MBUnitHex>,<MBFLASHHex>$_{CR}$**
	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. MBUnitDec MBUnitHex — The current used MODBUS/RTU unit or ASCII address for communication MBFLASHDec MBFLASHHex — The internal stored MODBUS/RTU unit address or ASCII address from the FLASH memory, if the DIP switch is 0.
Host	**#<BusAdr>,RST$_{CR}$** **#<BusAdr>,RESET$_{CR}$**
Answer	**None**
	Executes a software reset (Reboot) of the module.

5.15 MODBUS – register description

5.15.1 Table of inputs and coils

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 0x00001 to 0x65536. Inputs are usually noted with 1x00001 to 1x65536. 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: 0x00100 for the coil 100, 1x00100 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.

Register	Description
0x00001 1x00001 I:0 R/O DI1	Current state of digital input I1 =0:DI is OFF, =1:DI is ON
0x00002 1x00002 I:1 R/O DI2	Current state of digital input I1 =0:DI is OFF, =1:DI is ON
0x00003 1x00003 I:2 R/O NDI1	Current negated state of digital input I1 =0:DI is ON, =1:DI is OFF
0x00004 1x00004 I:3 R/O NDI2	Current negated state of digital input I2 =0:DI is ON, =1:DI is OFF
0x00005 1x00005 I:4 R/O ORDIS	Bitwise OR between both digital inputs I1 OR I2 I1 I2 I1 OR I2 OFF OFF =0 OFF ON =1 ON OFF =1 ON ON =1
0x00006 1x00006 I:5 R/O XORDIS	Bitwise EXCUSIVE OR between both digital inputs I1 XOR I2 I1 I2 I1 EXKLUSIVE OR I2 OFF OFF =0 OFF ON =1 ON OFF =1 ON ON =0
0x00007 1x00007 I:6 R/O ANDDIS	Bitwise AND between both digital inputs I1 AND I2 I1 I2 I1 AND I2 OFF OFF =0 OFF ON =0 ON OFF =0 ON ON =1

Register	Description
0x00100 1x00100 I:99 R/W RESET COUNTER	Reset, resetting of the internal edge counters to 0. If read, this bit is always 0.

5.15.2 Table of holding/input registers

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.

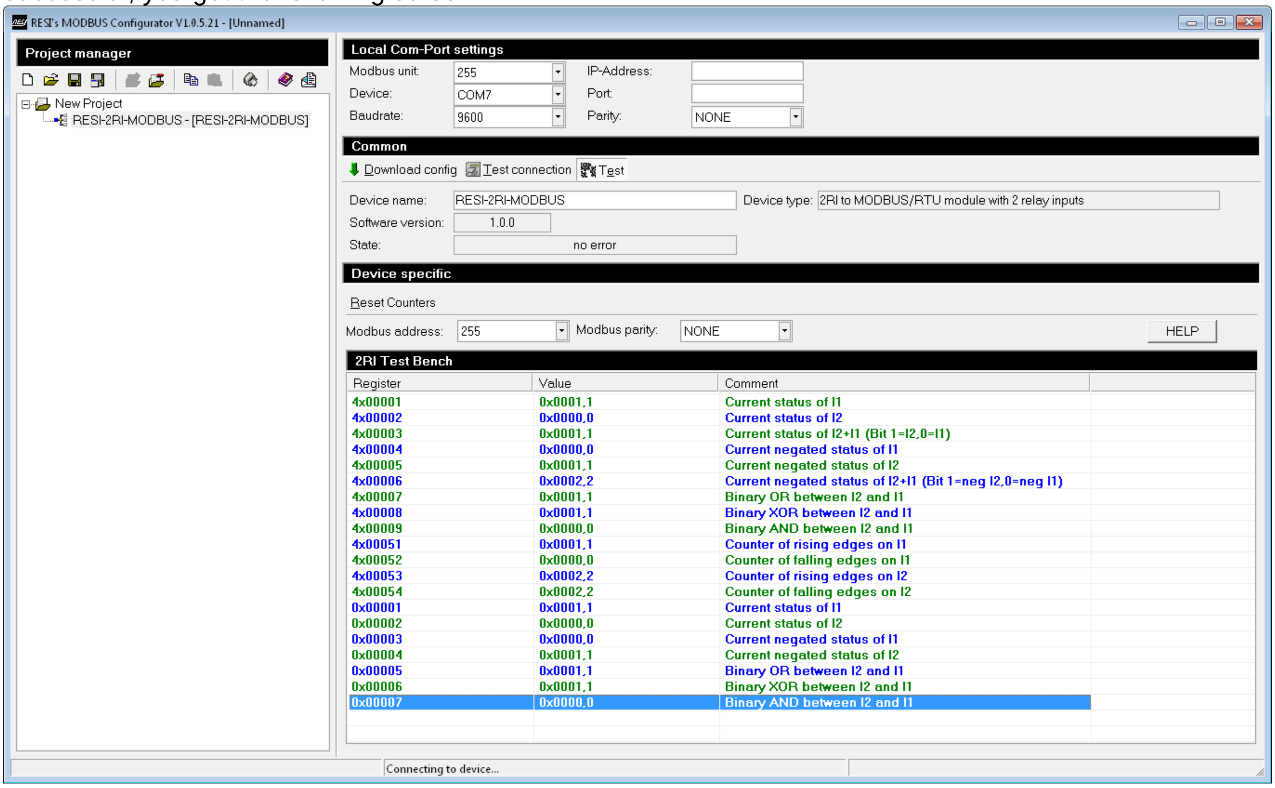
The MODBUS convention defines 65535 possible holding register with the notation 4x00001 to 4x65536. Input register are usually noted with 3x00001 to 3x65536. Please refer the software MODBUS POLL as a sample for this notation. Internally in the MODBUS/RTU frames an index notation is used, which starts 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.

Register	Description
4x00001 3x00001 I:0 R/O DI1	Current state of digital input I1 =0:I1 is OFF, =1:I1 is ON
4x00002 3x00002 I:1 R/O DI2	Current state of digital input I2 =0:I2 is OFF, =1:I2 is ON
4x00003 3x00003 I:2 R/O DIS	Current state of all digital inputs Bit 0: =0:I1 is OFF, =1:I1 is ON Bit 1: =0:I2 is OFF, =1:I2 is ON
4x00004 3x00004 I:3 R/O NDI1	Current negated state of digital input I1 =0:I1 is ON, =1:I1 is OFF
4x00005 3x00005 I:4 R/O NDI2	Current negated state of digital input I2 =0:I2 is ON, =1:I2 is OFF
4x00006 3x00006 I:5 R/O NDIS	Current negated state of all digital inputs Bit 0: =0:I1 is ON, =1:I1 is OFF Bit 1: =0:I2 is ON, =1:I2 is OFF
4x00007 3x00007 I:6 R/O ORDIS	Bitwise OR between the current states of both digital inputs I1 I2 I1 OR I2 OFF OFF =0 OFF ON =1 ON OFF =1 ON ON =1
4x00008 3x00008 I:7 R/O XORDIS	Bitwise EXCLUSIVE OR between the current states of both digital inputs I1 I2 I1 XOR I2 OFF OFF =0 OFF ON =1 ON OFF =1 ON ON =0

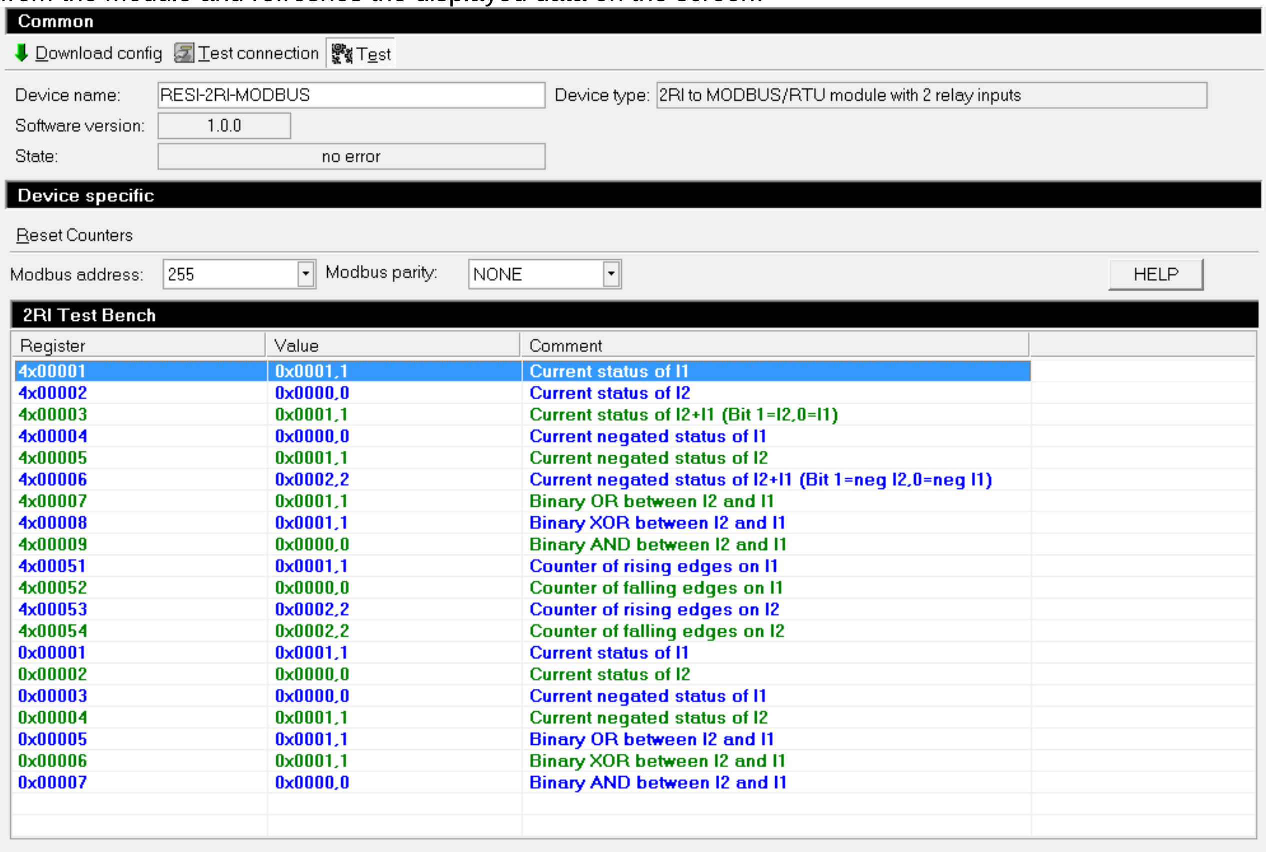
Register	Description
4x00009 3x00009 I:8 R/O ANDDIS	Bitwise AND between the current states of both digital inputs I1 I2 I1 ANS I2 OFF OFF =0 OFF ON =0 ON OFF =0 ON ON =1
4x0051 3x0051 I:50 R/O RISE DI1	Edge counter for rising edges on the digital input I1. If the module detects a rising edge on the input I1, this counter increments with 1. After a reset or a power loss of the module all internal edge counters are 0. With the function RESET COUNTER the host can cleat this counter to 0.
4x0052 3x0052 I:51 R/O FALL DI1	Edge counter for falling edges on the digital input I1. If the module detects a rising edge on the input I1, this counter increments with 1. After a reset or a power loss of the module all internal edge counters are 0. With the function RESET COUNTER the host can cleat this counter to 0.
4x0053 3x0053 I:52 R/O RISE DI2	Edge counter for rising edges on the digital input I2. If the module detects a rising edge on the input I2, this counter increments with 1. After a reset or a power loss of the module all internal edge counters are 0. With the function RESET COUNTER the host can cleat this counter to 0.
4x0054 3x0054 I:53 R/O FALL DI2	Edge counter for falling edges on the digital input I2. If the module detects a rising edge on the input I2, this counter increments with 1. After a reset or a power loss of the module all internal edge counters are 0. With the function RESET COUNTER the host can cleat this counter to 0.
4x0100 3x0100 I:99 R/W RESET COUNTER	If the host writes to this register, all edge counters are cleared to 0. If the host reads this register, always 0 is returned.
4x6001 3x6001 I:6000 W/O RESET SYSTEM	If the host writes to this register, the module executes a soft reset (reboot).
4x65222 3x65222 I:65221 R/W MODBUS UNIT ADDRESS	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.

5.16 Module test with RESI MODBUSConfigurator software

Establish a connection between the module and our software tool RESI MODBUSConfigurator. If this is successful, you get the following screen:



You can enable/disable the testing mode with the button „TEST". Every 5 seconds the software reads new data from the module and refreshes the displayed data on the screen:



For the 2RI module is also a button named „Reset Counters"in the software tool. This will reset all edge counters to 0.

6 RESI-1RO-MODBUS, RESI-1RO-ASCII

6.1 Product description

This IO module offers the following features:

- 1 relay output with normally open and normally closed contacts for 250Vac or 30vdc, max. 8A Signals
- Galvanic insulated RS232/RS485 interface for communication with a host system
- RESI-1RO-MODBUS: MODBUS/RTU slave protocol
- RESI-1RO-ASCII: MODBUS/RTU slave and text based ASCII protocol
- Power supply 12-48Vdc
- DIP switch for setting the baud rates, the type of interface and the bus number
- LED indicator for the communication
- Mounting onto EN50022 DIN rail

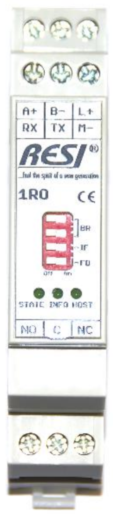


Illustration: Our IO module

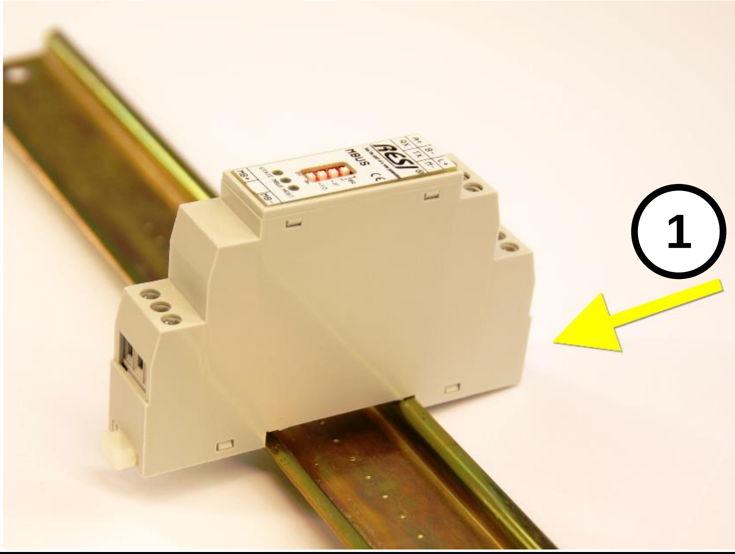
6.2 Technical data

Technical Data			
Power supply			
Supply voltage	12-48V= +/-10%	Storage temperature	-20...80 °C
Power LED	Yes	Operating Temperature	0...60°C
Power consumption	<0.8W	Humidity	25...90 % rH non-condensing
		Protection Class	IP20 (EN 60529)
		Dimensions LxWxH	17.5mm x90mm x58mm
		Weight	60g
		Mounting	on DIN EN50022 rail
ASCII/Modbus Interface			
Protocol	ASCII or Modbus/RTU		
Type	RS232 or RS485		
Baud rates	9600 to 57600/8/N or E/1		
Cable Connection	Via clamps		
LED indicator	Yes		
Galvanic insulation	Yes		
Relay output			
Number of channels	1		
Signal	250Vac or 30Vdc max. 8A		
Relay type	Changeover relays		
Contact material	Au-flashed AgNi		
Maximum contact rating	with 250Vac: 2000VA with 30Vdc: 240W		
Maximum contact voltage	250Vac or 125Vdc with 0.2A		
Cable connection	Via clamps		
Galvanic insulation to serial interface	Yes		
LED Indicator	Yes		
Clamps			
Clamp wire cross section	Max. 1,5 mm²		
Tightening torque	Max. 0.5Nm	**CE conformity**	Yes

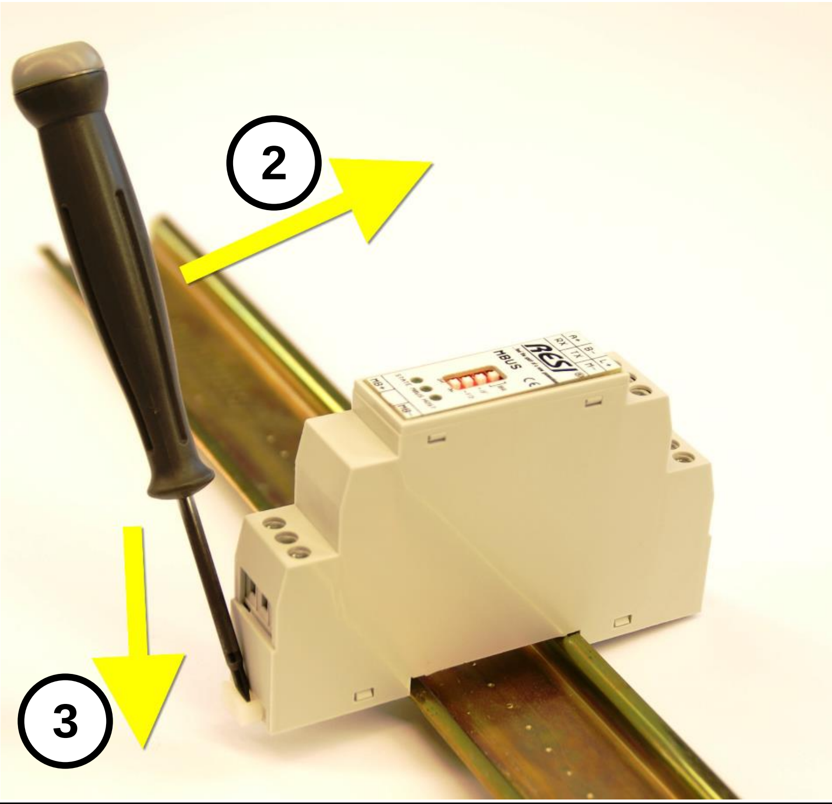
6.3 Assembling

Our IO modules are designed for mounting on a 35mm DIN-EN50022 rail.

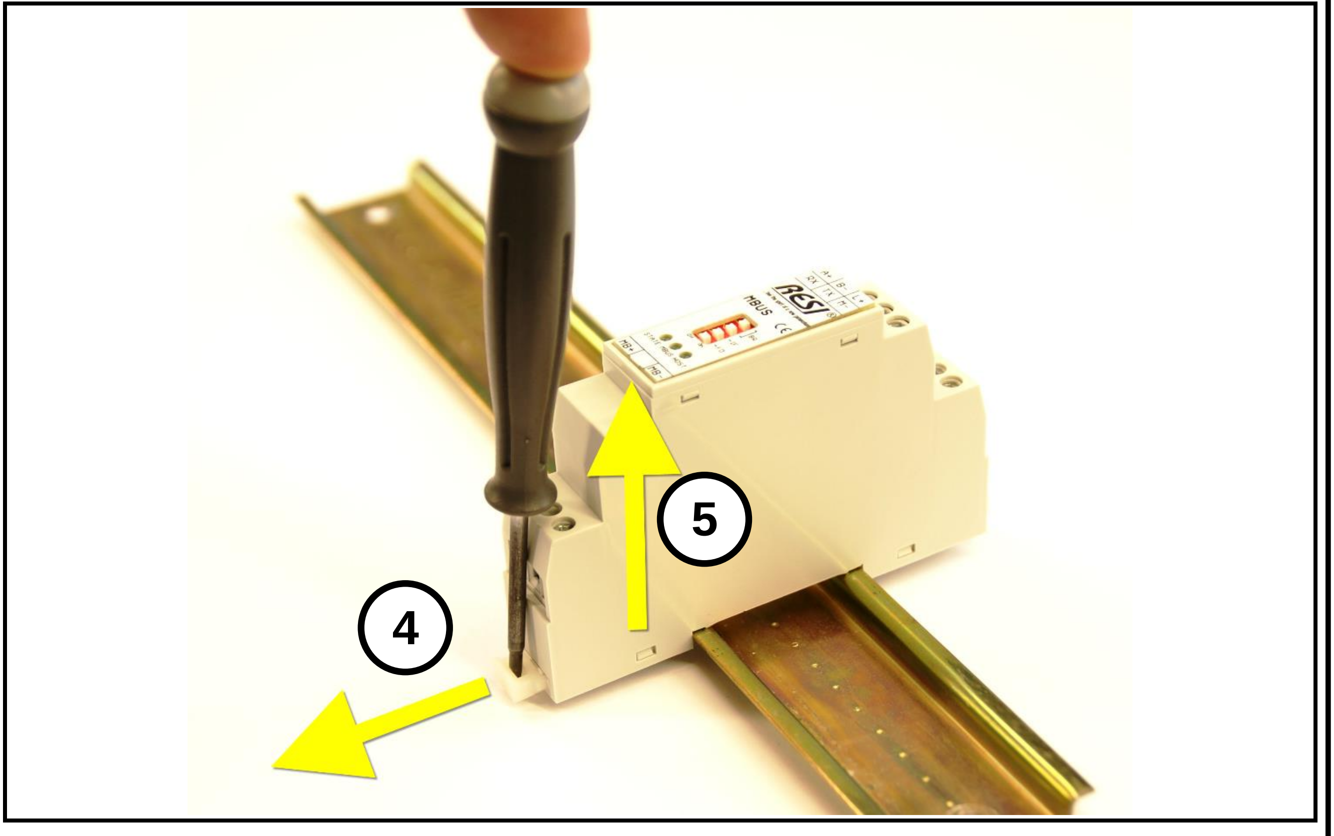
At first, put the modules with the top side on the DIN rail (1).



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.



6.4 Connection diagram

In the below drawing you will find the correct cabling of the IO module.

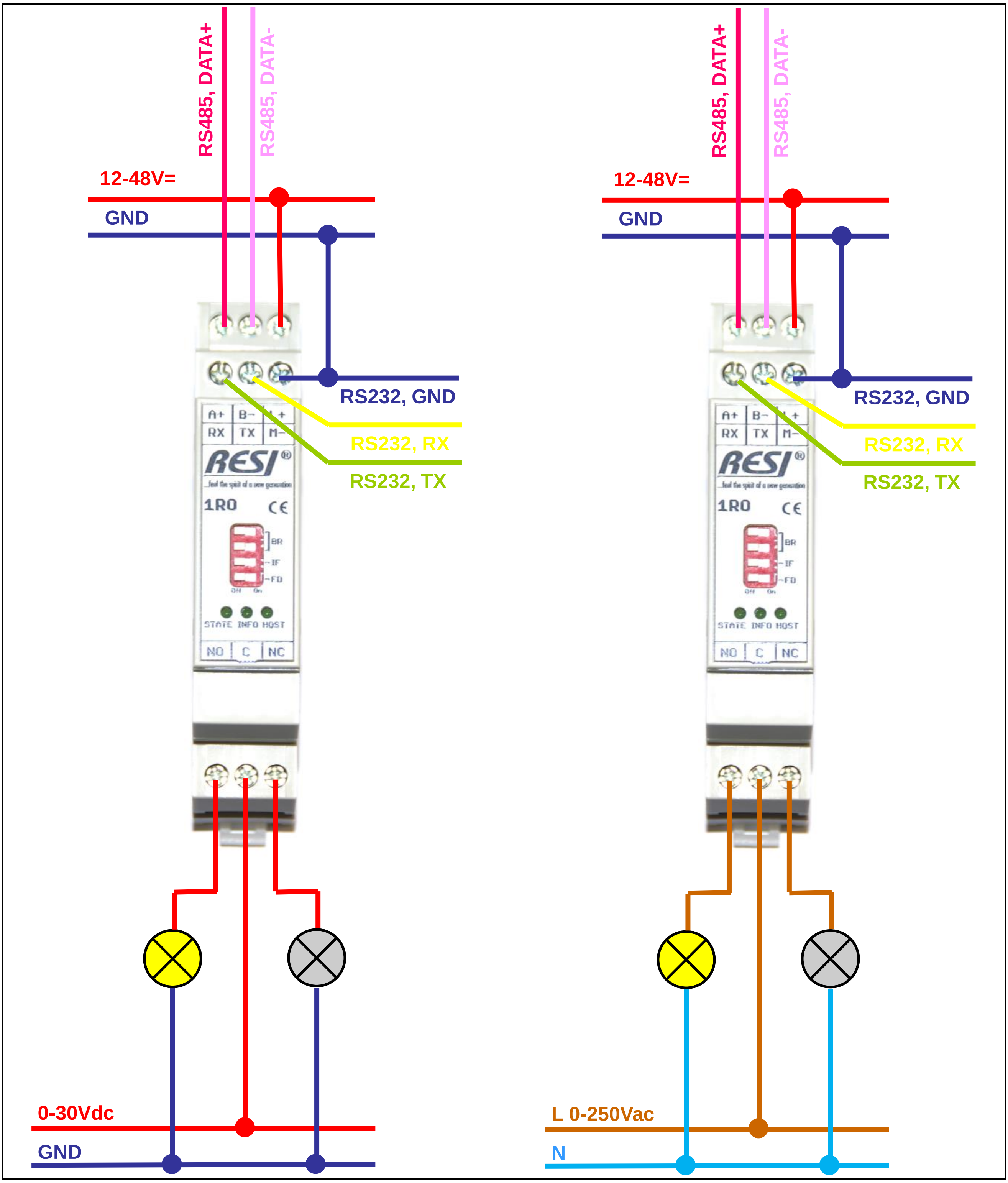


Illustration: Cabling of the IO module

6.5 Clamps

The IO module offers the following clamps:

CLAMP	DESCRIPTION
L+ M-	Power supply: L+: 12-48 V= M-: Ground
RS485 A+ B- M-	RS485 ASCII or MODBUS/RTU interface A+: RS485 DATA+ signal B-: RS485 DATA- signal M-: RS485 ground signal
RS232 TX+ RX- M-	RS232 ASCII or MODBUS/RTU interface TX+: RS232 Transmit signal RX-: RS232 Receive signal M-: RS232 Ground signal
Relays C=Common NO=Normally Open NC=Normally Closed	1 relay output with two contacts C: Common contact for NO and NC contact NO: Normally open contact related to clamp C NC: Normally closed contact related to clamp C

Table: Description of the clamps on the IO module

6.6 DIP switch setting and LED indicators

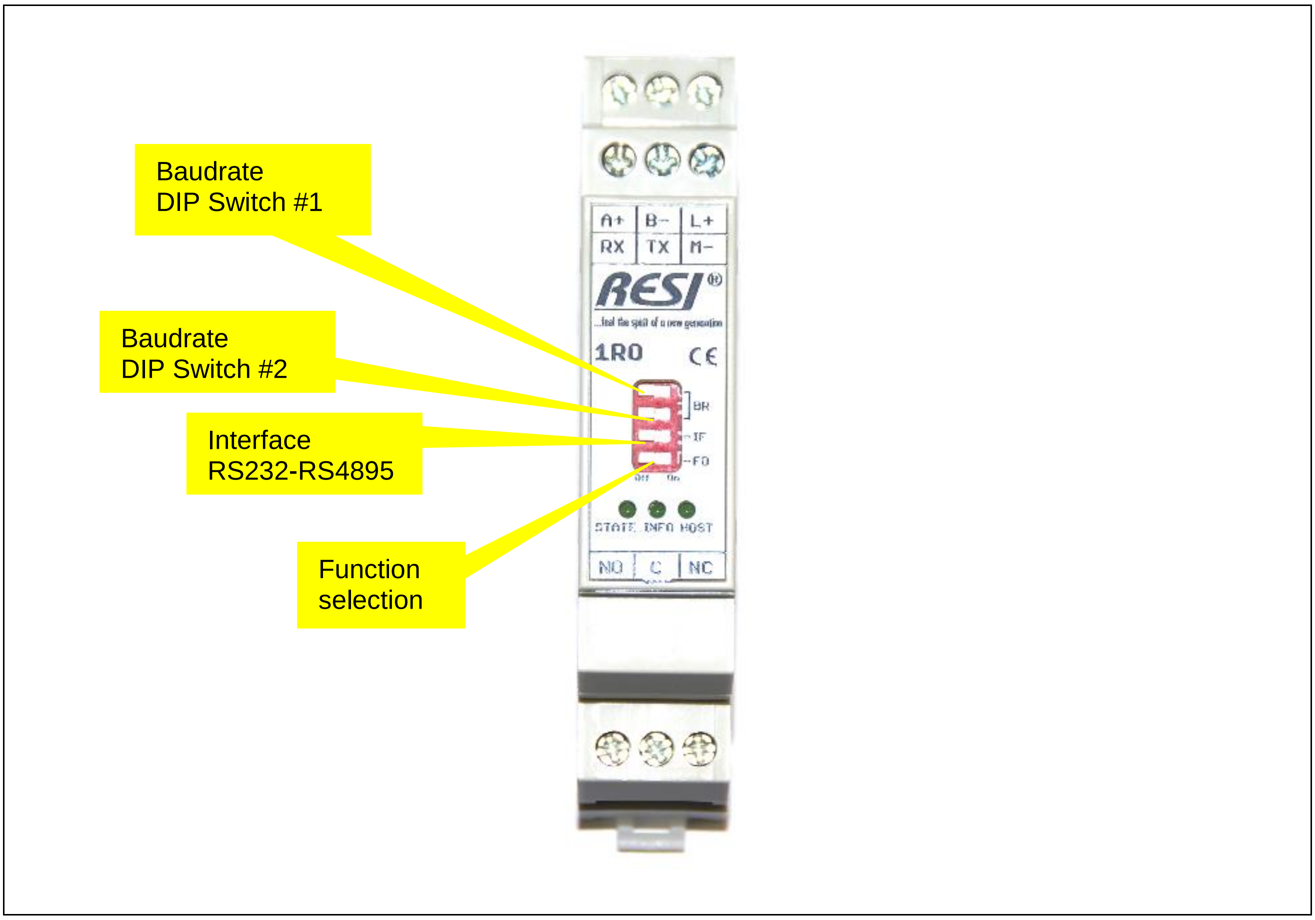


Illustration: Description of the DIP switch settings and LED indicators

DIP Switch	Description
Baudrate BR	Use DIP switches 1+2 to select the baud rate: OFF OFF: 9600Bd ON OFF: 19200Bd OFF ON: 38400Bd ON ON: 57600Bd HINT: The correct parity (NONE, EVEN, ODD) is defined by the PC software, not with the DIP switches.
Interface IF	Selects the physical type of the serial interface for the ASCII or MODBUS/RTU protocol: OFF=RS232 ON=RS485
Function definition FD	Select s special function in the module: OFF=The module uses the configured unit ID from the FLASH memory ON=The module uses always the unit ID 255
HINT	After a change of the DIP switches, the module reboots. No power off / power on cycle is necessary. After the reset all three LEDs are shortly on to represent the RESTART sequence.

Table: Description of the DIP switches of the IO module

LED	Description
STATE	Status LED: If the module is ok, this LED flashes slowly. If there is an error detected by the module, this LED flashes fast.
INFO	If the relay is on, this LED is on.
HOST	HOST-LED, Flashes, if the host is communicating with the module.

Table: Description of the LED indicator on the IO module

6.7 Dimensions of the module

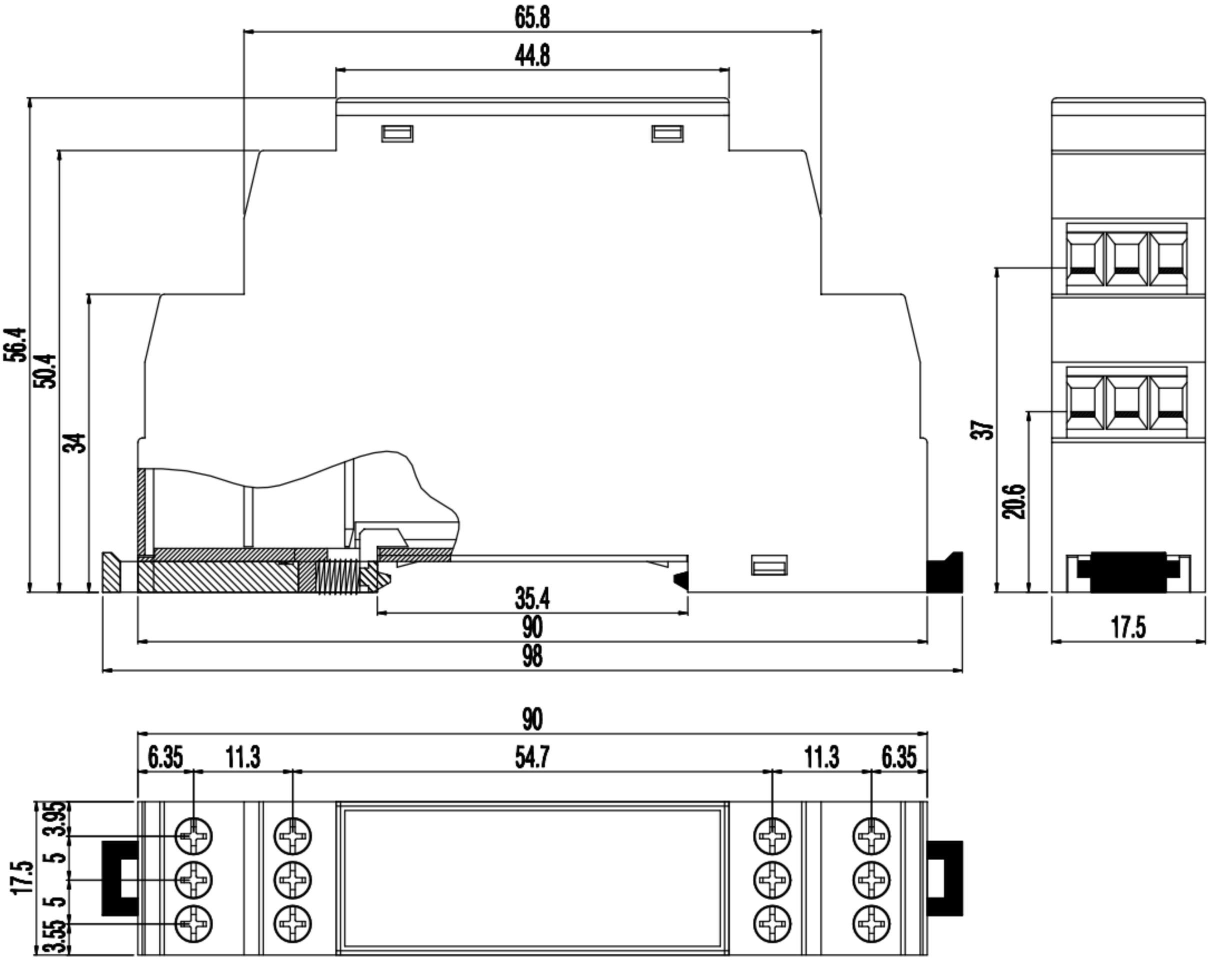


Illustration: dimension illustration in mm

Dimensions	
Enclosure dimensions L x W x H (mm)	17,5 x 90 x 58
Weight	60 g
Color	Grey RAL7035
Material	PA - UL 94 V0
Protection class	IP20 based on DIN 40050/EN 60529

Table: Data of enclosure

6.8 3D Drawing

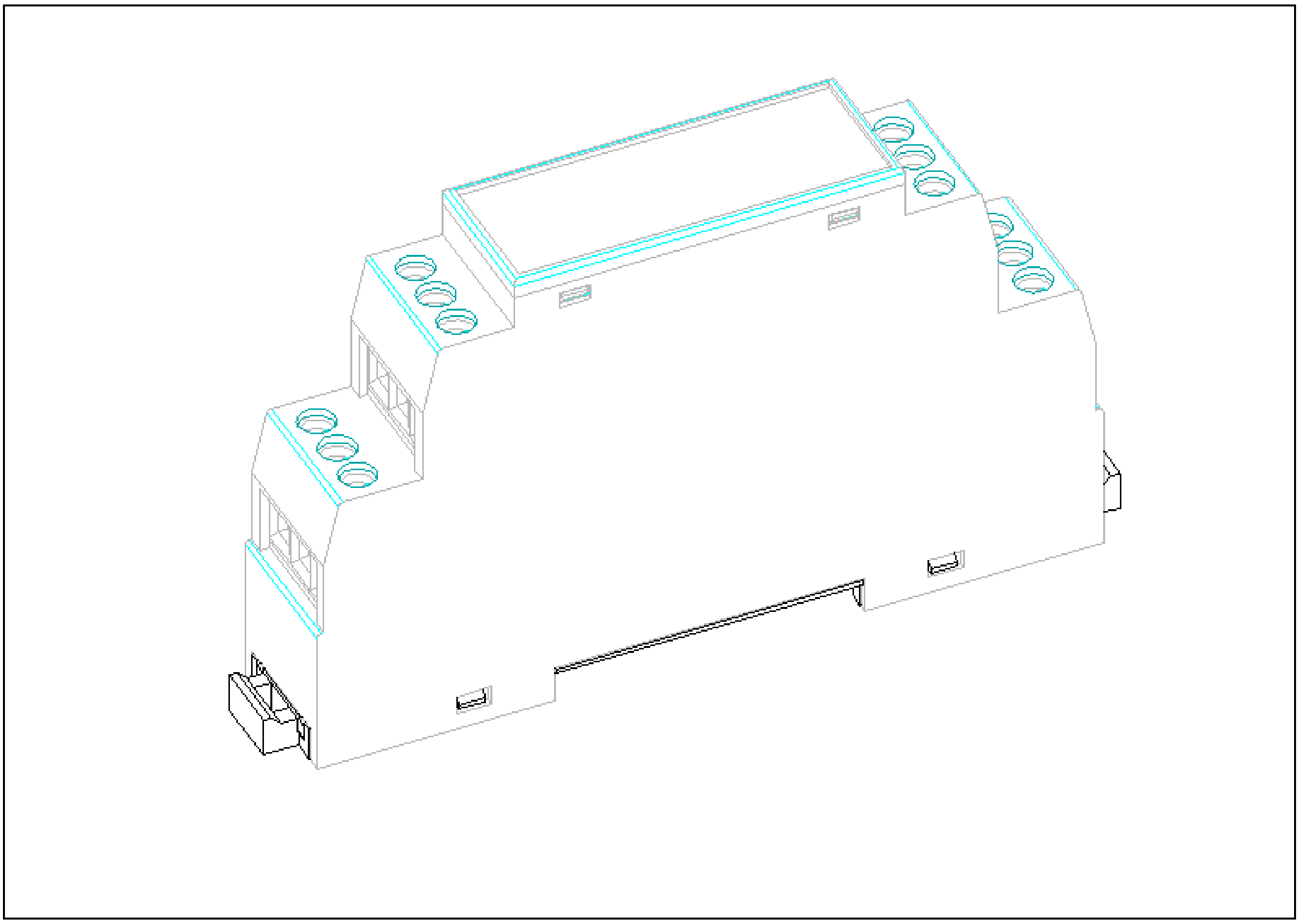


Illustration: 3D drawing of the enclosure

6.9 Power supply cabling of the module

In the image below you will see the correct cabling of the power supply of the module.

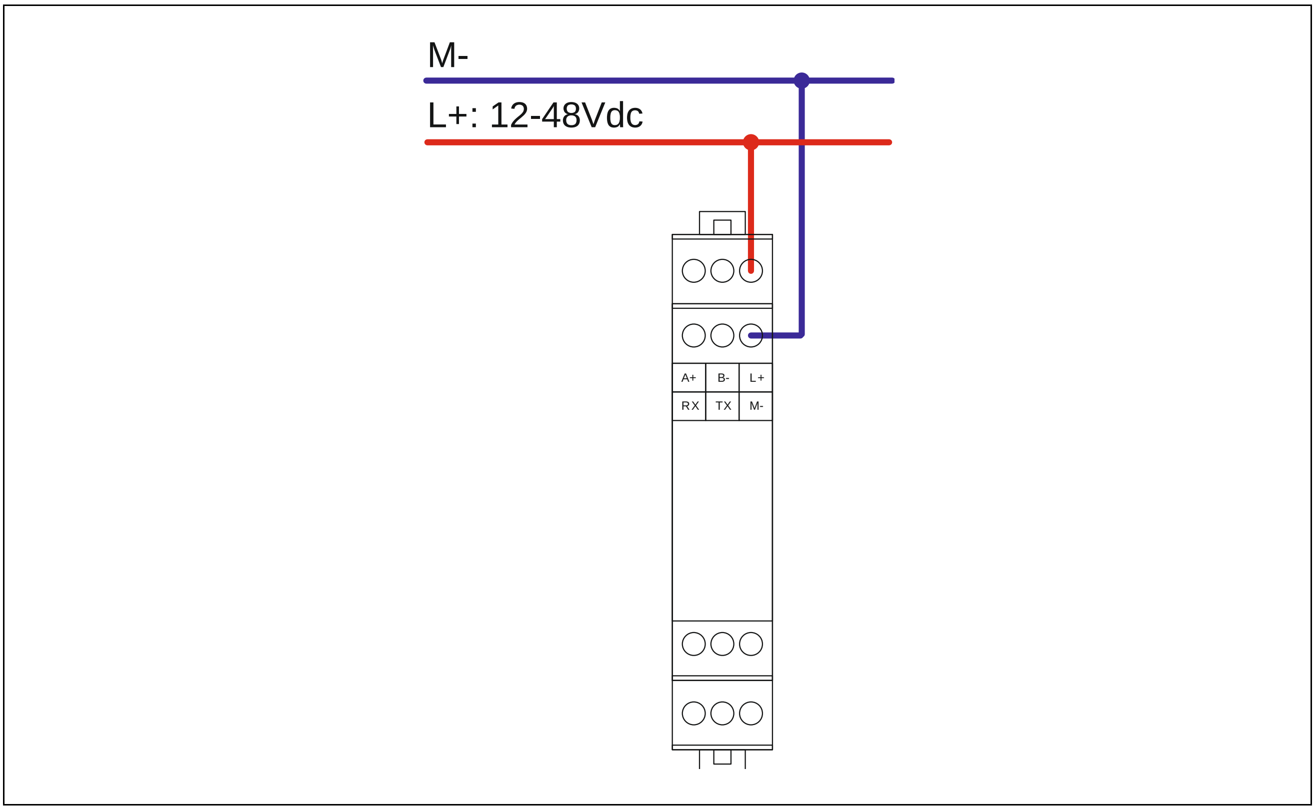


Illustration: Cabling of the power supply of the IO module

6.10 RS485 cabling of the IO module

In the image below you see the correct cabling of the RS485 interface of the IO module.

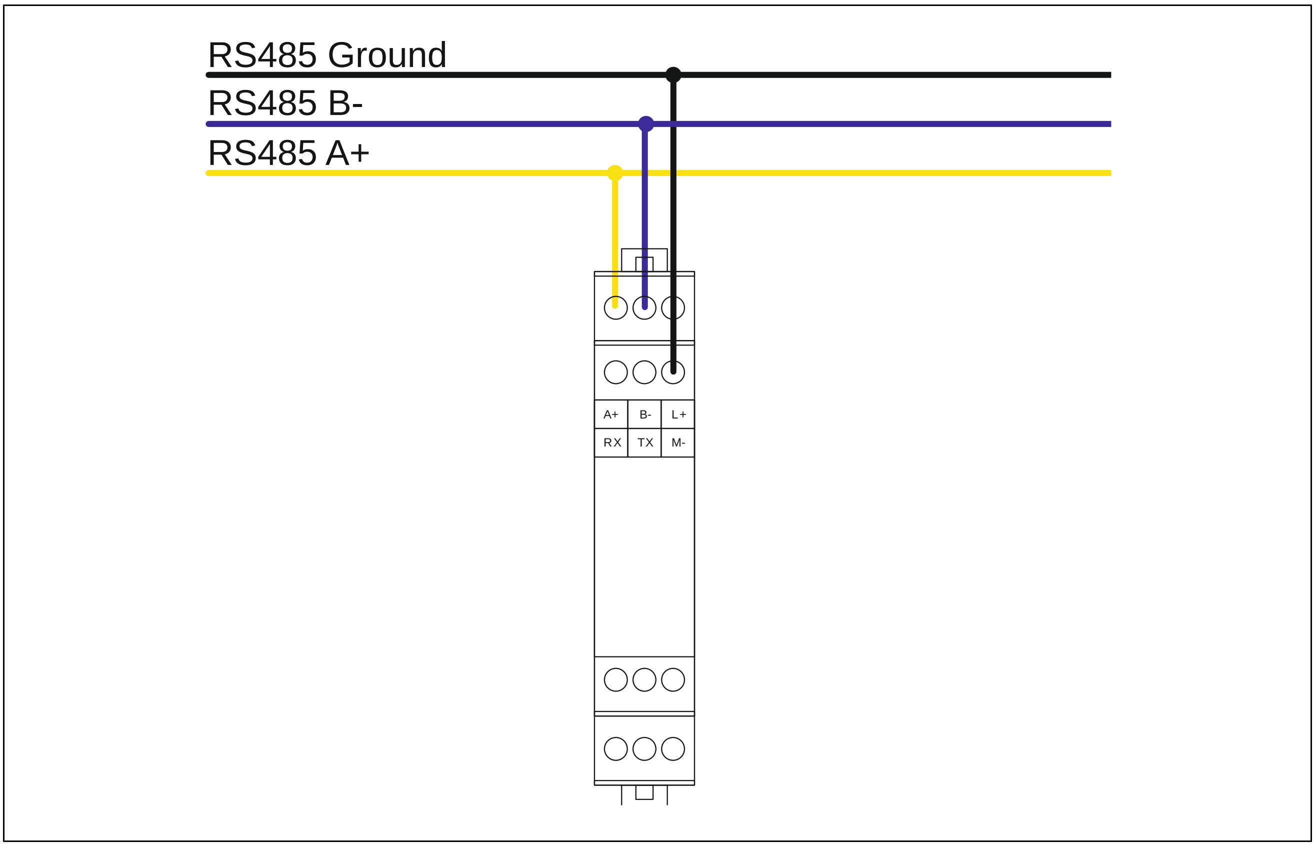


Illustration: RS485 bus cabling of the IO module

6.11 RS232 cabling of the IO module

In the image below you see the correct cabling of the RS232 interface of the IO module.

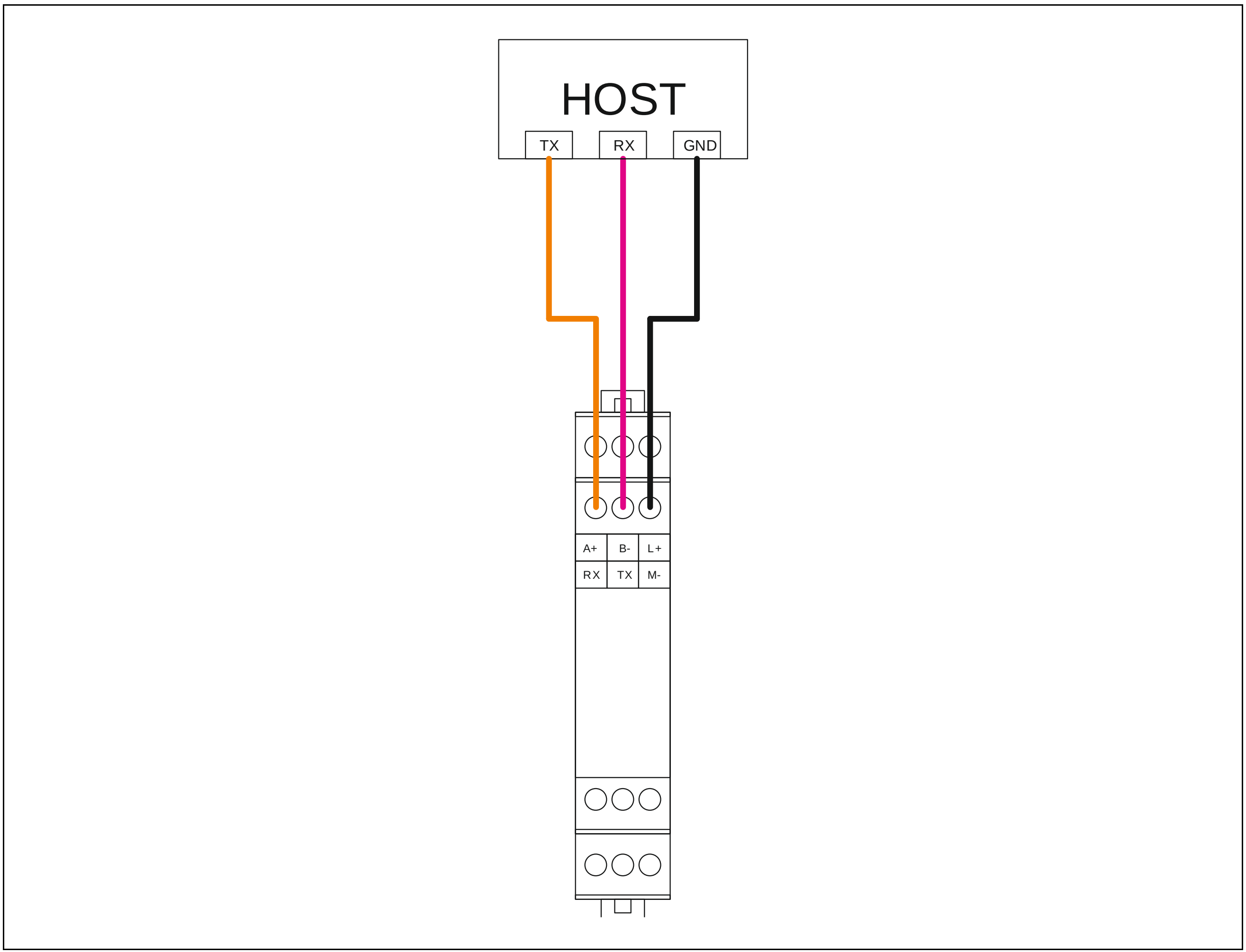


Illustration: RS232 bus cabling of the IO module

6.12 Cabling of the relay output of the module

In the image below you see the correct cabling of the relay output with the two contacts normally open (NO), normally closed (NC) of the IO module. The clamp C is the common clamp for both relay output contacts. To symbolize the state of the two relay contacts, we use two LEDs.

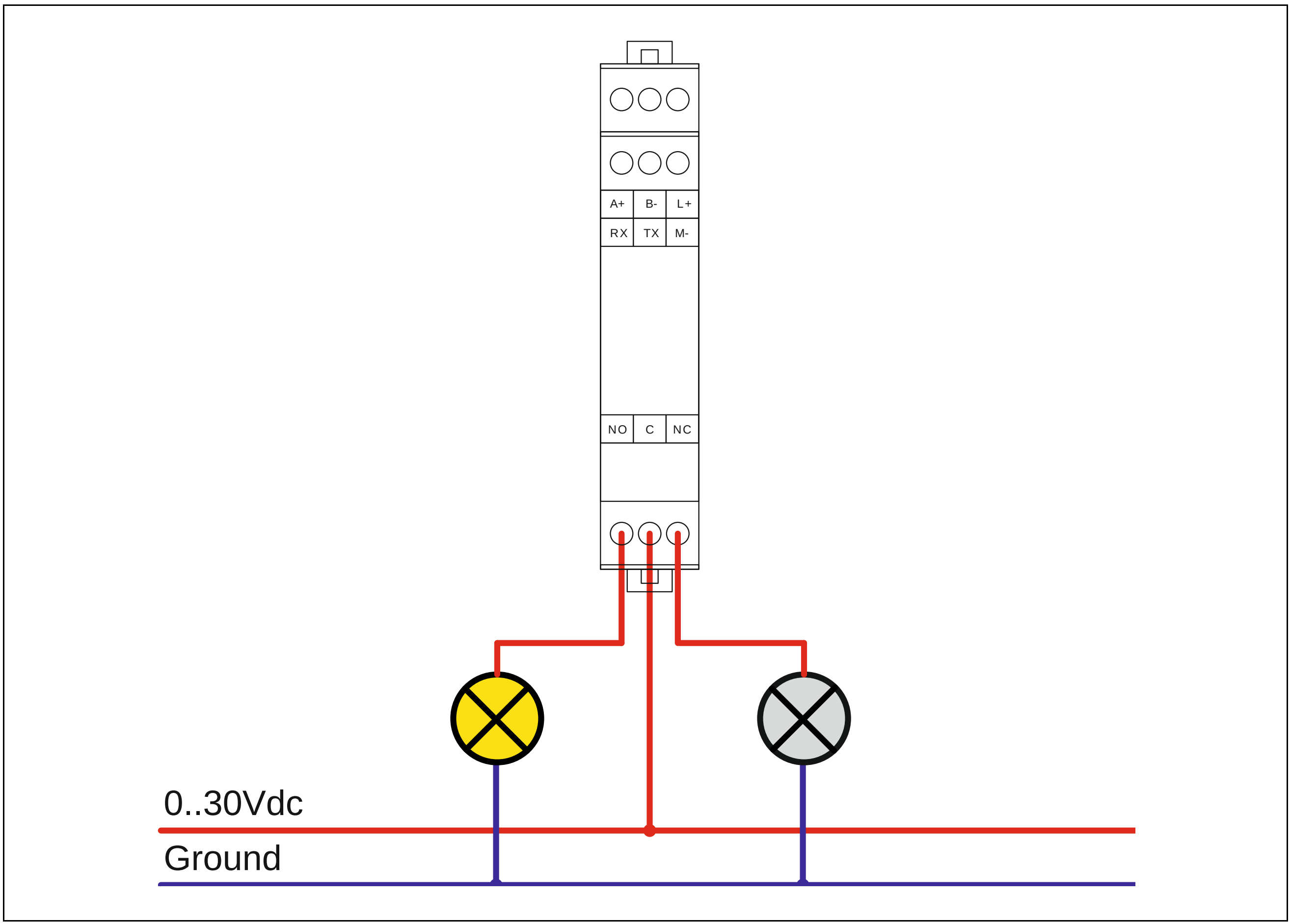


Illustration: Cabling of the relay output of the IO module for DC signals: State of the relay contacts, when the relay is under power (=ON).

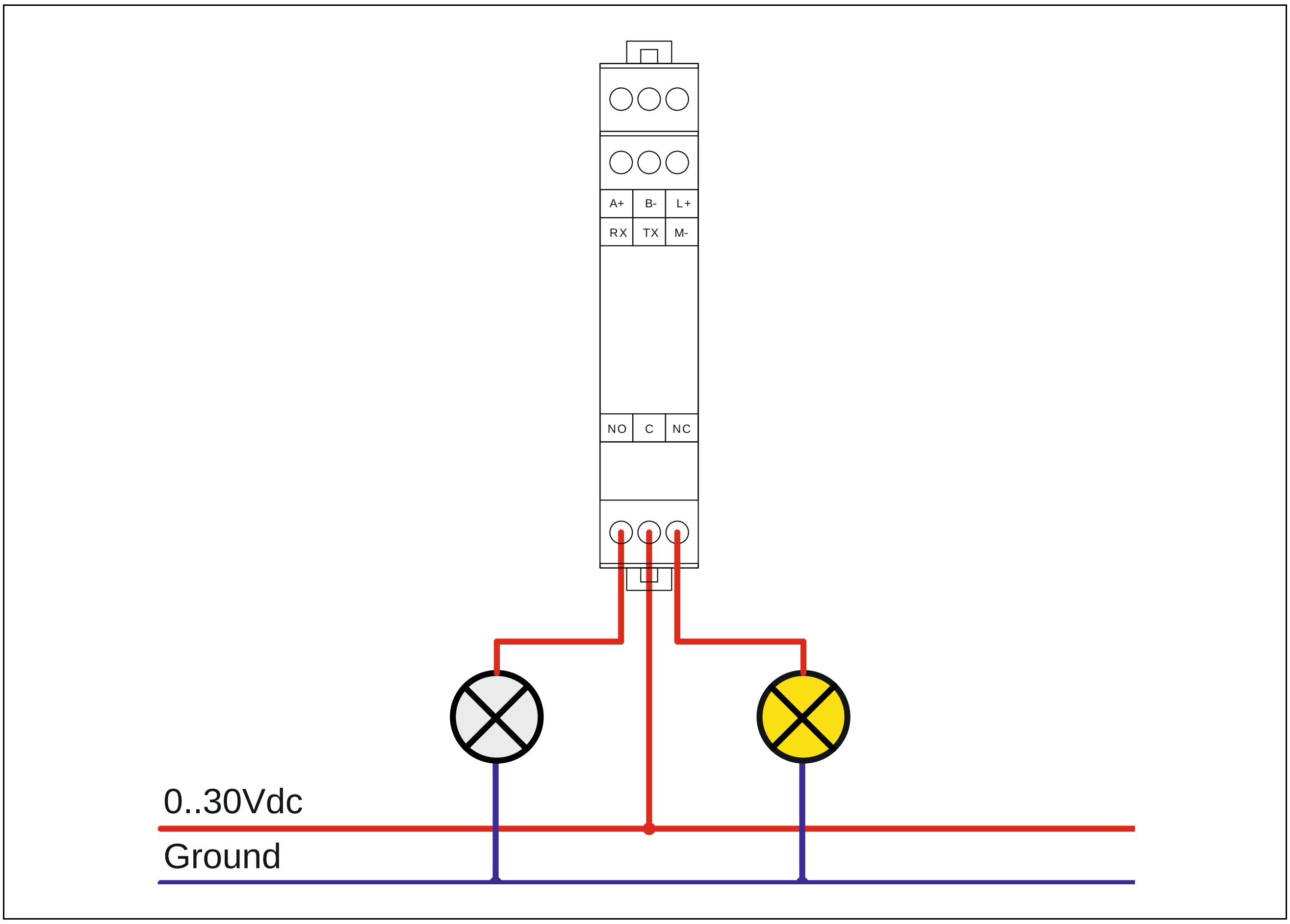


Illustration: Cabling of the relay output of the IO module for DC signals: State of the relay contacts, when the relay is in power-off state (=OFF).

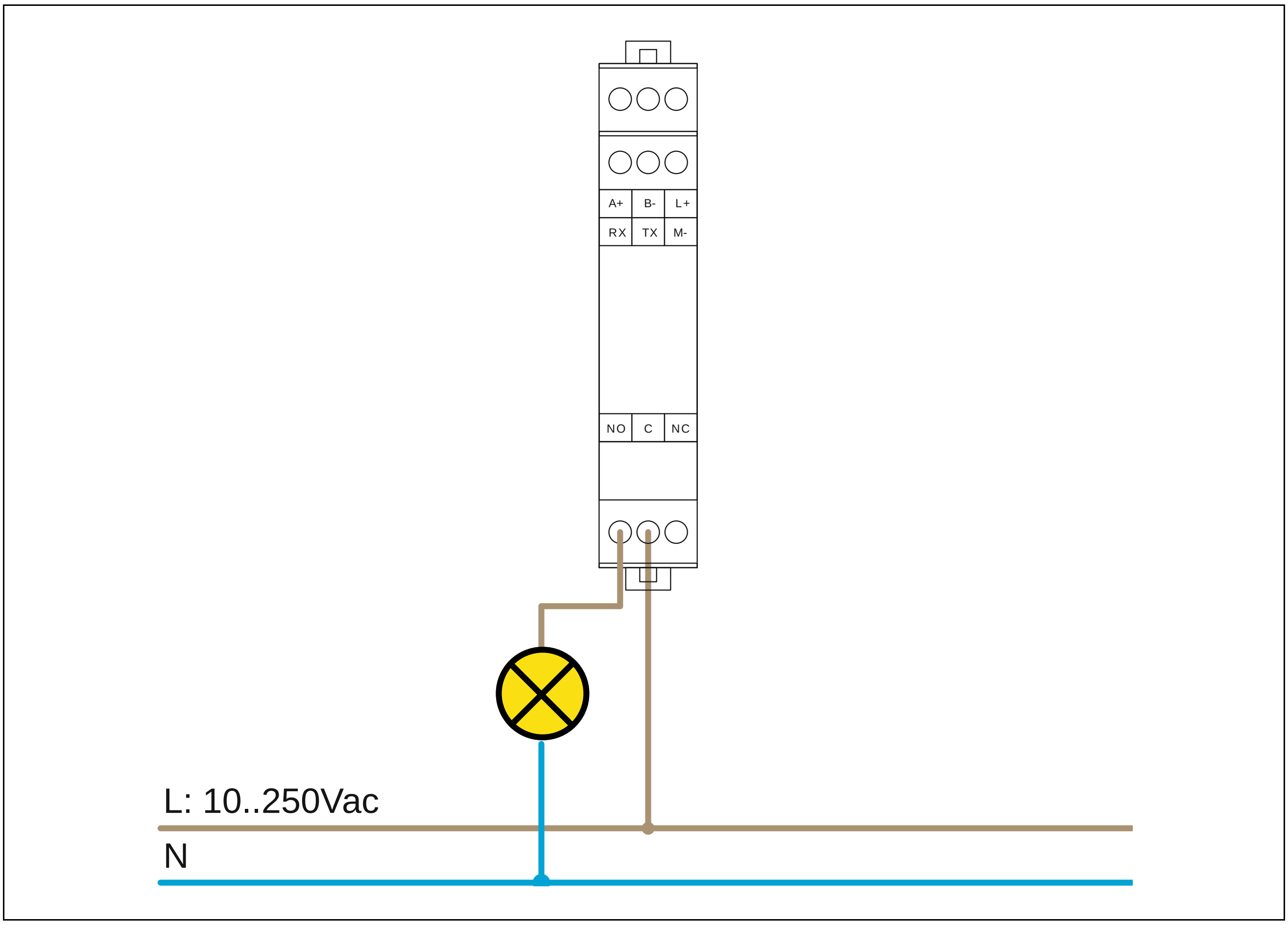


Illustration: Cabling of the relay output for AC signals with a classic closing contact (normally open).

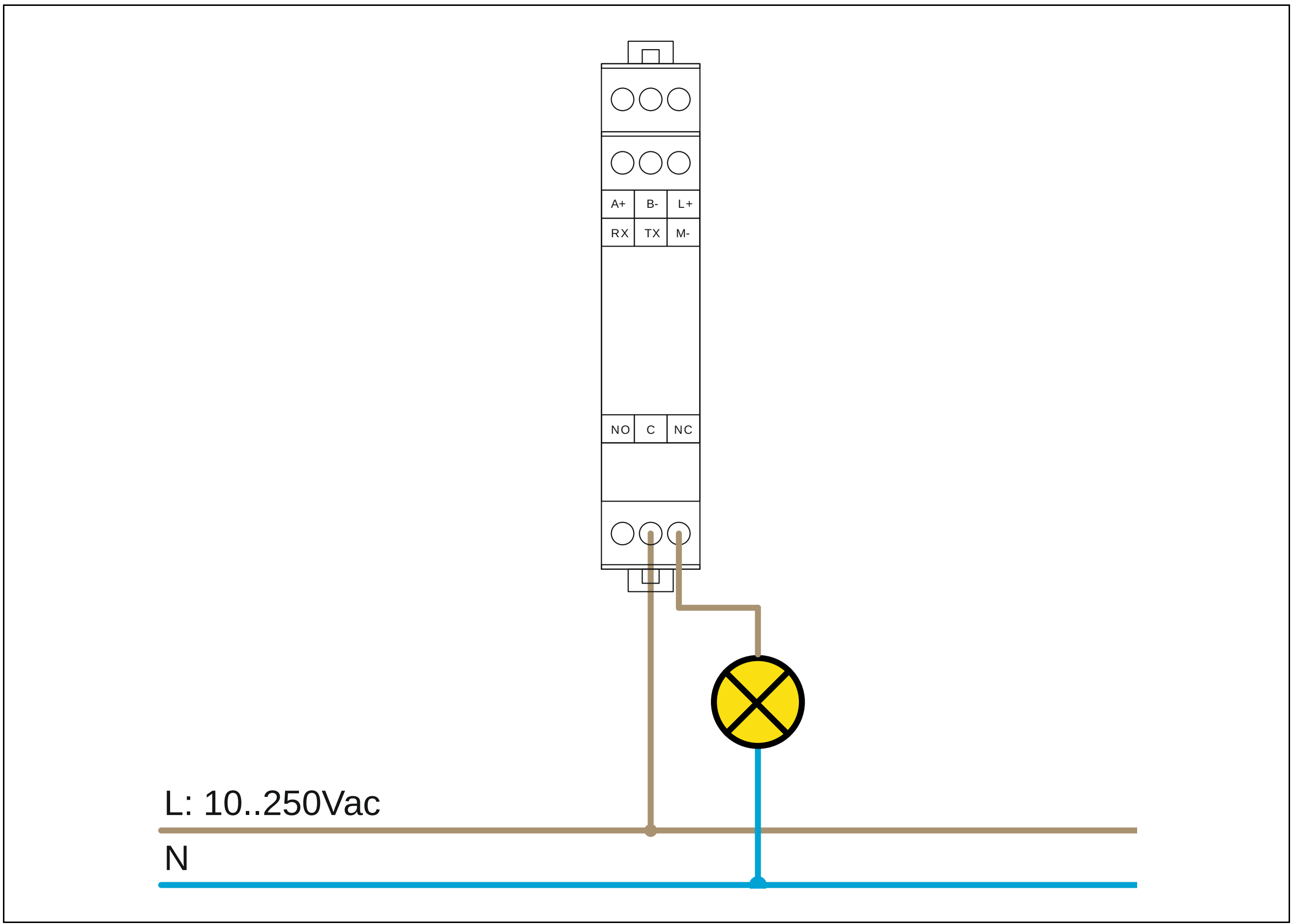


Illustration: Cabling of the relay output for AC signals with a classic break contact (normally close).

6.13 Functional description

This IO module communicates with a host system with the MODBUS/RTU slave protocol. All versions of the Io module with RESI-xxx-ASCII offer an additional protocol: An ASCII slave protocol with simple text string. The communication runs over a RS485 interface (half duplex) or over a RS232 interface (full duplex).

For the communication with ASCII texts, the host sends ASCII messages with a special start character # (0x23, 35dec) and a special end character (0x0d,13dec or CARRIAGE RETURN) to the module. The module uses also this special start and end characters to answer to the host request. Consult the below noted detailed command descriptions. In the ASCII protocol mode, the host can send messages with or without a bus number.

For communication with the MODBUS/RTU slave, the module offers the following MODBUS functions:

- 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)

HINT:
The functions READ HOLDING REGISTER and PRESET MULTIPLE REGISTERS are restricted to max. 125 register per request! The functions READ INPUT STATUS, READ COIL STATUS and FORCE MULTIPLE COILS are restricted to 2000 coils or inputs (bits) per request.

6.14 ASCII protocol description

6.14.1 Overview

The IO module communicates with simple ASCII commands. The following special characters are used in this documentation:

stand for the **Hashtag** ASCII character 35dec or 0x23
: stand for the **colon** ASCII character 58dec or 0x3A
= stand for the **equal sign** with the ASCII code 61ec or 0x3D
- stand for the **minus sign** with the ASCII code 45dec or 0x2D
, stands for the **comma** with the ASCII code 44dec or 0x2C
<CR> stand for **CARRIAGE RETURN** ASCII character 13dec or 0x0D. In the following text we use the representation CR.
<SP> stands for the **SPACE**. This is the space character with the ASCII code 32dec or 0x20. In the following text we will use the representation☐.
<ADR> will be used as the current **bus address** of the module. The bus number can be transmitted decimal or hexadecimal and is separated with a comma (ASCII code 44dec or 0x2C) from the rest of the command. Hexadecimal numbers always start with 0x. Its only allowed to use the ASCII characters ‚0‘-‚9‘ 48dec to 57dec, 0x30-0x39 and ‚A‘ to ‚F‘, 65dec to 70dec, 0x41-0x46. All modules react to the broadcast bus address 0 and to its own bus number. With a DIP switch, the user can easily change between the internal stored bus number in the FLASH and the fix bus number 255. Consult the DIP switch description for more details.

6.14.2 Communication sequence

In general the IO module sends no ASCII characters without a request from a host. So the host is the master of the communication and the IO module is always answering to host requests as a slave. If only one IO module is connected to a host (eg. Via RS232), you can dump the bus number in the protocol. If you use a RS485 interface, more than one module can be connected to the host. Therefore a bus number in the request frame of the host is always necessary.

The structure of the command look like this:

The host sends a command or a command with parameters without a bus address to the module:
#<Command><CR> or
#<Command>:<Parameter><CR>

The module answers, if it feels addressed, with the following answer telegram:
#<Answer><CR>

If the bus number is used, the module answers with:
#<ADR>,<Command><CR> or
#<ADR>,<Command>:<Parameter><CR>

The module answers with:
#<ADR>,<Answer><CR>

The bus address lies in the range of 1dec to 255dec or 0x00 to 0xFF hexadecimal. The setup is done with our free configuration software MODBUSConfigurator. Each module reacts also to the broadcast bus address 0.

For each command, we define two different writings. A long version and a short version, to avoid unnecessary traffic on the bus. For example to request the software version of the module you can use the command VERSION or the short command VER.

6.14.3 Request VERSION

This command returns the current software version of the module.

Host long version:
#VERSION<CR> or
#<ADR>,VERSION<CR>

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

Answer:
#VERSION:<HIGH>.<MED>.<LOW><CR> or
#<ADR>,VERSION:<HIGH>,<MED>,<LOW><CR>

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

Samples:
→ **#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$_{CR}$

6.14.4 Request module TYPE

This command returns the current type of the module.

Host long version:
#TYPE<CR> or
#<ADR>,TYPE<CR>

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

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

<TYP> defines the current type of the module. Currently RESI-1RO-ASCII

Samples:
→ #TYPE$_{CR}$
← #TYPE:RESI-1RO-ASCII$_{CR}$

→ #255,TYP$_{CR}$
← #255,TYPE:RESI-1RO-ASCII$_{CR}$

6.14.5 Table of all ASCII commands

Here you will find a possible ASCII commands of the module. We use here only the version with bus number. That you can avoid the bus number, we have discussed earlier in this document. If an argument as the extension Dec, it will be returned as a decimal number, If an argument has the extension Hex, then this argument is returned as a hexadecimal number. Many command returns the argument in decimal and hexadecimal representation. So the host can select, what kind of number conversion, it will handle in its software.

Direction	ASCII command
Host	**#<BusAdr>,VER$_{CR}$** **#<BusAdr>,VERSION$_{CR}$**
Answer	**#<BusAdr>,VERSION:<VersionHi>.< VersionMed>.<VersionLo>$_{CR}$**
	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)
Host	**#<BusAdr>,TYP$_{CR}$** **#<BusAdr>,TYPE$_{CR}$**
Answer	**#<BusAdr>,TYPE:RESI-1RO-ASCII$_{CR}$**
	Returns the current type of the module
Host	**#<BusAdr>,OWN$_{CR}$** **#<BusAdr>,OWNER$_{CR}$**
Answer	**#<BusAdr>,OWNER:RESI$_{CR}$**
	Returns the owner of the module
Host	**#<BusAdr>,CRE$_{CR}$** **#<BusAdr>,CREATOR$_{CR}$**
Answer	**#<BusAdr>,CREATOR:DI HC SIGL,MSC$_{CR}$**
	Returns the creator of the module

Direction	ASCII command
Host	**#<BusAdr>,COPY$_{CR}$** **#<BusAdr>,COPYRIGHT$_{CR}$**
Answer	**#<BusAdr>,COPYRIGHT:2015 BY RESI AND DI HC SIGL,MSC WWW.RESI.CC$_{CR}$**
	Returns a copyright note about the module
Host	**#<BusAdr>,GDIP$_{CR}$** **#<BusAdr>,GET☐DIP$_{CR}$**
Answer	**#<BusAdr>,GDIP:<DIPSwitchDec>,<DIPSwitchHex>$_{CR}$**
	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)
Host	**#<BusAdr>,GDOS$_{CR}$** **#<BusAdr>,GET☐DOS$_{CR}$**
Answer	**#<BusAdr>,GDOS:<DOSDec>,<DOSHex>$_{CR}$**
	Returns the current state of all digital outputs as decimal and hexadecimal number DOSDec DOSHex — The current state of all digital outputs: Bit 0: State of digital output (=0:OFF, =1:ON) Bit 1: State of the relay output (=0:OFF, =1:ON) The relay output state is the current state of the relay including offdelay timers or blinking timers in the module.
Host	**#<BusAdr>,GDO1$_{CR}$** **#<BusAdr>,GET☐DO1$_{CR}$**
Answer	**#<BusAdr>,GDO1:<DO1Dec>,<DO1Hex>$_{CR}$**
	Returns the current state of the digital output DO of the module as decimal and hexadecimal number DO1Dec DO1Hex — The current state of the digital output 1: =0: digital output is OFF =1: digital output is ON
Host	**#<BusAdr>,GNDOS$_{CR}$** **#<BusAdr>,GET☐NEG☐DOS$_{CR}$**
Answer	**#<BusAdr>,GNDOS:<NDOSDec>,<NDOSHex>$_{CR}$**
	Returns the negated current state of all digital outputs of the module as decimal and hexadecimal number NDOSDec NDOSHex — The negated current states of all digital outputs: Bit 0: negated state of digital output (=0:ON, =1:OFF) Bit 1: negated state of the true relay output (=0:OFF, =1:ON)
Host	**#<BusAdr>,GNDO1$_{CR}$** **#<BusAdr>,GET☐NEG☐DO1$_{CR}$**
Answer	**#<BusAdr>,GNDO1:<NDO1Dec>,<NDO1Hex>$_{CR}$**
	Returns the current negated state of the digital output DO as decimal and hexadecimal number NDO1Dec NDO1Hex — The current negated state of the digital output 1: =0: digital output is OFF =1: digital output is ON

Direction	ASCII command
Host	**#<BusAdr>,GTDO1$_{CR}$** **#<BusAdr>,GET☐TRUE☐DO1$_{CR}$**
Answer	**#<BusAdr>,GTDO1:<TDO1Dec>,<TDO1Hex>$_{CR}$**
	Returns the current state of the true digital output (the real state of the relay) as decimal and hexadecimal number. This function incudes all offdelay timers or internal timings of the relay. TDO1Dec TDO1Hex The true state of the relay output 1: =0: relay is OFF =1: relay is ON
Host	**#<BusAdr>,GOFFDLY$_{CR}$** **#<BusAdr>,GET☐OFFDELAY$_{CR}$**
Answer	**#<BusAdr>,GOFFDLY:<OFFDLYDec>,<OFFDLYHex>$_{CR}$**
	Returns the current value of the offdelay timer. The value is a time in milliseconds. If this value is not zero, the relay is on, If this value is 0, the relay is off. OFFDLYDec OFFDLYHex The remaining time of the offdelay timer in milliseconds.
Host	**#<BusAdr>,SDO1:<ONOFF>$_{CR}$** **#<BusAdr>,SET☐DO1:<ONOFF>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
	Set digital output DO to the new value of <ONOFF>. ONOFF The new value of the digital output DO1: =0: Digital output 1 is OFF =1: Digital output 1 is ON !=0: Digital output 1 is ON
Host	**#<BusAdr>,SOFFDLY:<OFFDLY>$_{CR}$** **#<BusAdr>,SET☐OFFDELAY:<OFFDLY>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
	Sets the internal offdelay timer to the timer value <OFFDLY> in milliseconds. OFFDLY The new time value for the offdelay timer function in milliseconds. This function offers a watchdog feature: If you execute this command (e.g.: with the value 5000), the module activates the output relay for 5 seconds. If the module again receives this command within this 5 seconds, the internal timer is reset to the given time and the relay is still on. If no command is received within 5 seconds, the module switches off the relay.

Direction	ASCII command
Host	**#<BusAdr>,SMBADR:<MBUnit>$_{CR}$** **#<BusAdr>,SET□MODBUS□ADDRESS:<MBUnit>$_{CR}$**
Answer	**#<BusAdr>,OK $_{CR}$**
	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.
Host	**#<BusAdr>,GMBADR$_{CR}$** **#<BusAdr>,GET□MODBUS□ADDRESS$_{CR}$**
Answer	**#<BusAdr>,GMBADR:<MBUnitDec>,<MBFLASHDec>,<MBUnitHex>,<MBFLASHHex>$_{CR}$**
	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. MBUnitDec MBUnitHex: The current used MODBUS/RTU unit or ASCII address for communication MBFLASHDec MBFLASHHex: The internal stored MODBUS/RTU unit address or ASCII address from the FLASH memory, if the DIP switch is 0.
Host	**#<BusAdr>,RST$_{CR}$** **#<BusAdr>,RESET$_{CR}$**
Answer	**None**
	Executes a software reset (Reboot) of the module.

6.15 MODBUS – register description

6.15.1 Table of inputs and coils

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 0x00001 to 0x65536. Inputs are usually noted with 1x00001 to 1x65536. 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: 0x00100 for the coil 100, 1x00100 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.

Register	Description
0x00001 1x00001 I:0 R/W DO1	Current state of the digital output DO =0:DO is OFF, =1:DO is ON If you write to this register, you will change the current state of the digital output to the new value =0:Switch off DO, =1:Switch on DO
0x00002 1x00002 I:1 R/O NDO1	Current negated state of the digital output DO =0:DO is ON, =1:DO is OFF
0x00003 1x00003 I:2 R/O TDO1	Current state of the true relay output =0:relay is OFF, =1:relay is ON
0x00004 1x00004 I:3 R/O NTDO1	Current negated state of the true relay output =0:relay is ON, =1:relay is OFF
0x00005 1x00005 I:4 R/O OFFDLY	Current state of the Offdelay timer =0:Offdelay timer is OFF, =1:Offdelay timer is ON

6.15.2 Table of holding/input registers

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.

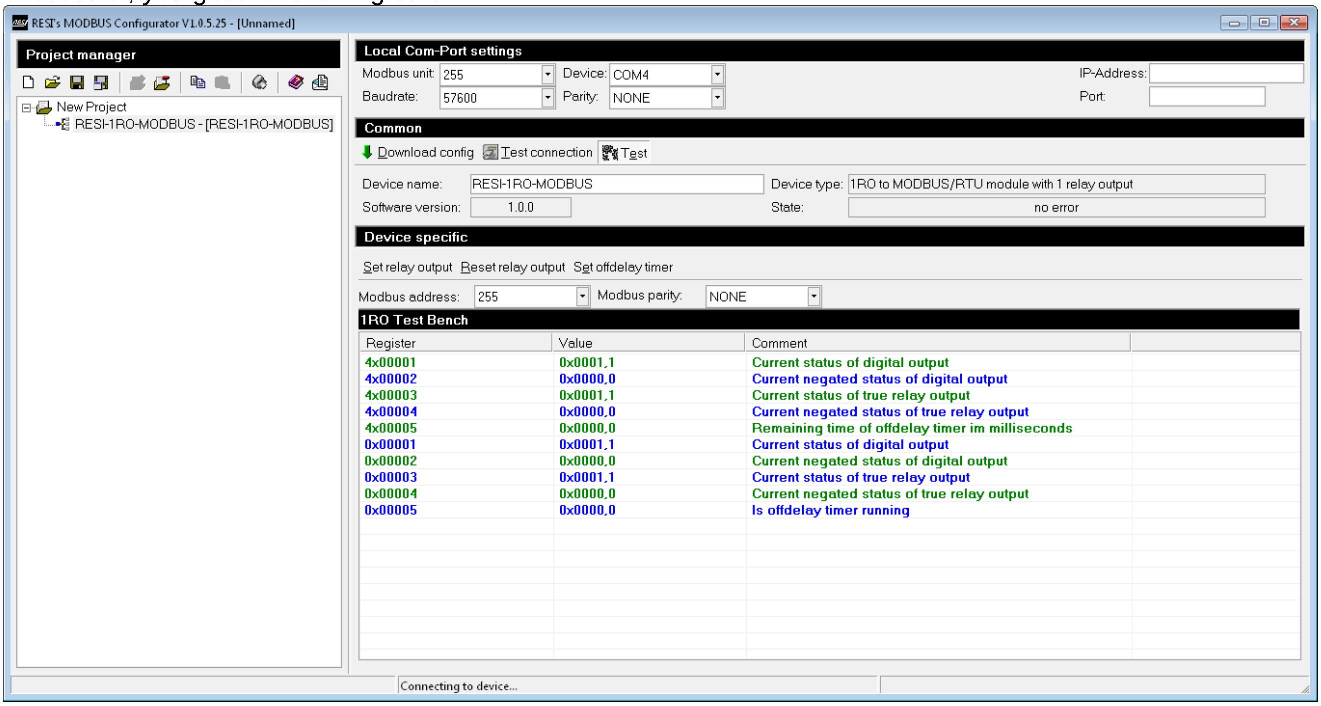
The MODBUS convention defines 65535 possible holding register with the notation 4x00001 to 4x65536. Input register are usually noted with 3x00001 to 3x65536. Please refer the software MODBUS POLL as a sample for this notation. Internally in the MODBUS/RTU frames an index notation is used, which starts 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.

Register	Description
4x00001 3x00001 I:0 R/W DO1	Current state of the digital output DO1 =0:DO is OFF, =1:DO is ON If you write to this register, you will change the current state of the digital output to the new value =0:Switch off DO, =1:Switch on DO
4x00002 3x00002 I:1 R/O NDO1	Current negated state of the digital output DO =0:DO is ON, =1:DO is OFF
4x00003 3x00003 I:2 R/O TDO1	Current state of the true relay output =0:relay is OFF, =1:relay is ON
4x00004 3x00004 I:3 R/O NTDO1	Current negated state of the true relay output =0:relay is ON, =1:relay is OFF
4x00005 3x00005 I:4 R/W OFFDLY	Current remaining time of the offdelay timer function in milliseconds. Remaining time of the offdelay timer If you write to this register, you will trigger the offdelay timer with the give value in milliseconds. A timeout between 0 and 65.535 seconds is configurable.

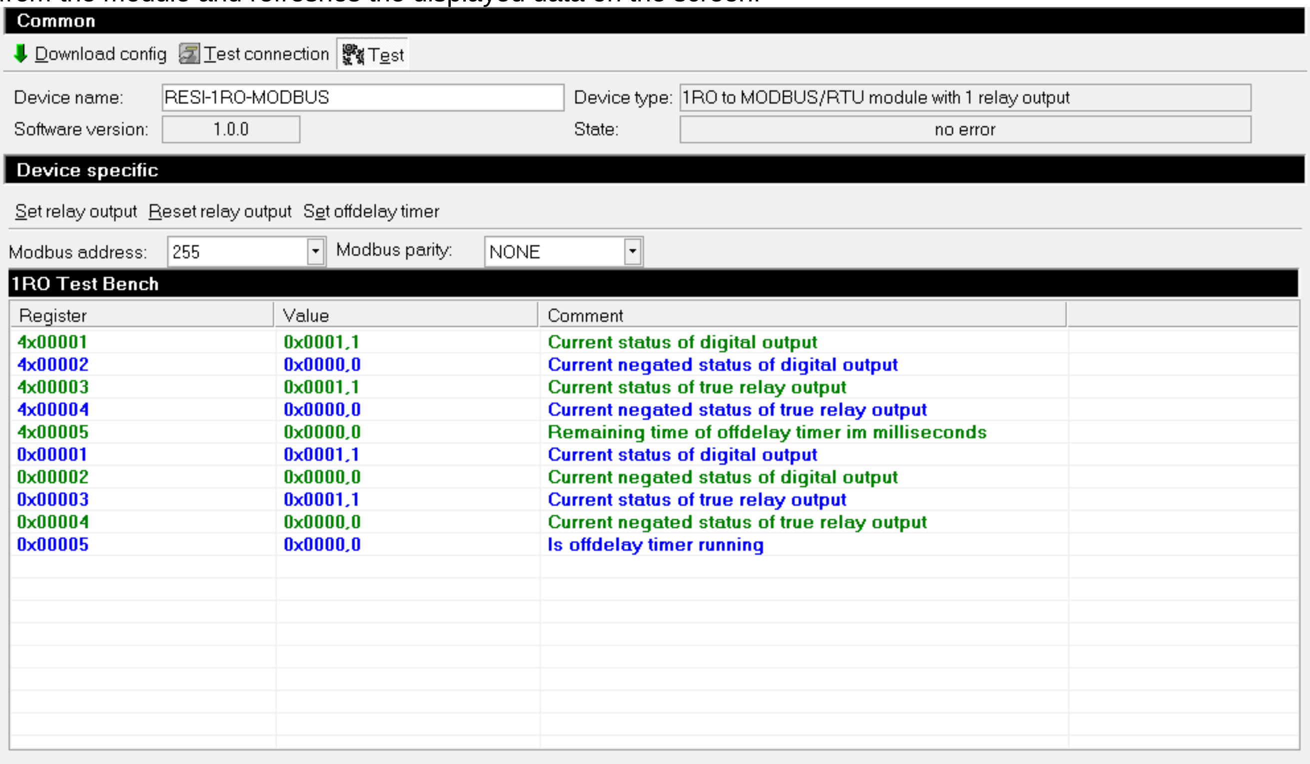
Register	Description
4x6001 3x6001 I:6000 W/O RESET SYSTEM	If the host writes to this register, the module executes a soft reset (reboot).
4x65222 3x65222 I:65221 R/W MODBUS UNIT ADDRESS	If the host reads this register, the current programmed unit ID is returned. All values above unit ID 255 define also 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.

6.16 Module test with RESI MODBUSConfigurator software

Establish a connection between the module and our software tool RESI MODBUSConfigurator. If this is successful, you get the following screen:



You can enable/disable the testing mode with the button „TEST“. Every 5 seconds the software reads new data from the module and refreshes the displayed data on the screen:



Some additional buttons offer the following features for the 1RO modules:
„Set relay output“ to switch on the relay output
„Reset relay output“to switch off the relay output
„Set offdelay timer“ to define a new watchdog time for the offdelay timer function

7 RESI-2RTD-MODBUS, RESI-2RTD-ASCII

7.1 Product description

This IO module offers the following features:

- 2 sensor inputs for temperature sensors
- Measurement accuracy +/-0.1%
- Measurement resolution +/-0.001%
- Measurement range -200°C…+850°C
- Various sensor types are applicable: PT100, PT1000, PT10, PT50, PT200, PT500, NI120
- Various standards for linearization are selectable: Europa, Amerika, Japan, ITS-90
- Output of the temperatures in °Celsius [°C], °Fahrenheit [°F] or °Kelvin [°K]
- Different measurement currents are selectable: 5µA, 10µA, 25µA, 50µA, 100µA, 250µA,500µA, 1mA
- Various sensor connection types : 2 wire, 3 wire or 4 wire sensors connectable
- Internal calculation of an average temperature per channel
- Galvanic insulated RS232/RS485 interface for communication with a host system
- RESI-2RTD-MODBUS: MODBUS/RTU slave protocol
- RESI-2RTD-ASCII: MODBUS/RTU slave and text based ASCII protocol
- Power supply 12-48Vdc
- DIP switch for setting the baud rates, the type of interface and the bus number
- LED indicator for the communication
- Mounting onto EN50022 DIN rail

Illustration: Our IO module

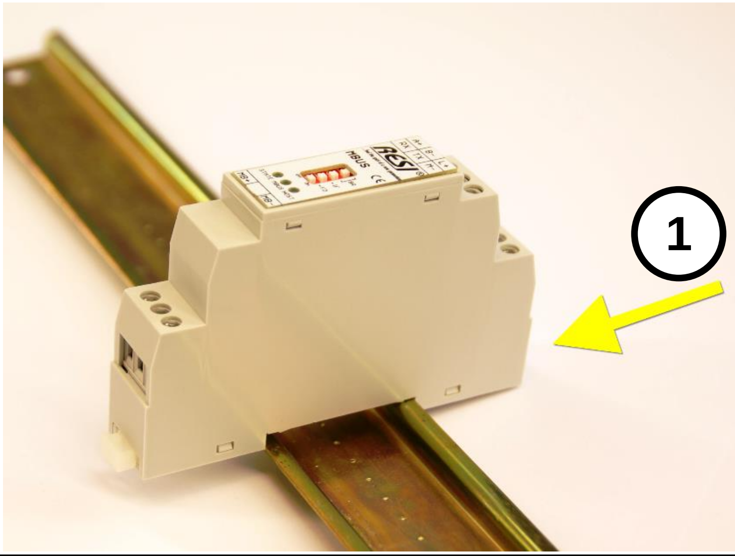
7.2 Technical data

Technical Data			
Power supply			
Supply voltage	12-48V= +/-10%	Storage temperature	-20...80 °C
Power LED	Yes	Operating Temperature	0...60°C
Power consumption	<0.7W	Humidity	25...90 % rH non-condensing
		Protection Class	IP20 (EN 60529)
		Dimensions LxWxH	17.5mm x90mm x58mm
		Weight	65g
		Mounting	on DIN EN50022 rail
ASCII/Modbus Interface			
Protocol	ASCII or Modbus/RTU		
Type	RS232 or RS485		
Baud rates	9600 to 57600/8/N or E/1		
Cable Connection	Via clamps		
LED indicator	Yes		
Galvanic insulation	Yes		
Temperature inputs			
Number of inputs	2		
Signal type	temperature measurement		
Type of measurement	Measurement of resistance		
ADC	24 bit sigma delta ADC		
Accuracy	+/-0.1°C for PT-100, PT-200,PT-500, PT-1000, NI-120 +/-3°C for PT-10, PT-50		
Resolution	+/-0.001°C		
Reference stability	10ppm/°C		
Sensor types	PT-100, PT-1000, PT-1000 α=0.00375, PT-10, PT-50, PT-200, PT-500, NI-120		
Linearization standards	Europa Amerika Japan ITS-90		
Excitation current for measurement	5µA, 10µA, 25µA, 50µA, 100µA, 250µA, 500µA, 1mA		
Cable connection	Via clamps		
Galvanic insulation to serial interface	Yes		
LED Indicator	Yes		
Clamps			
Clamp wire cross section	Max. 1,5 mm²		
Tightening torque	Max. 0.5Nm	**CE conformity**	Yes

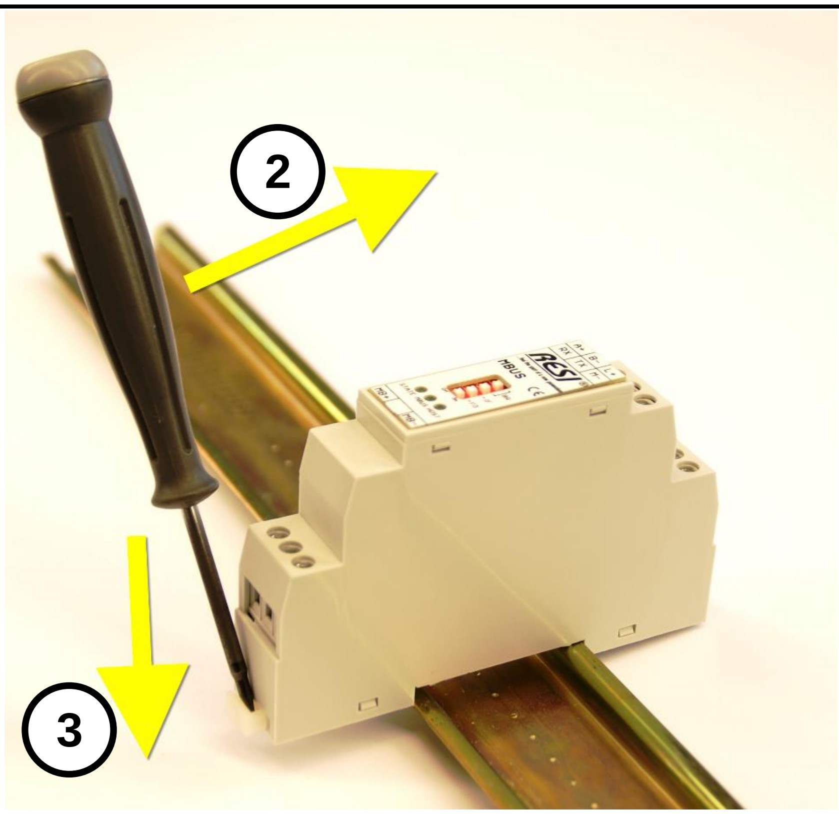
7.3 Assembling

Our IO modules are designed for mounting on a 35mm DIN-EN50022 rail.

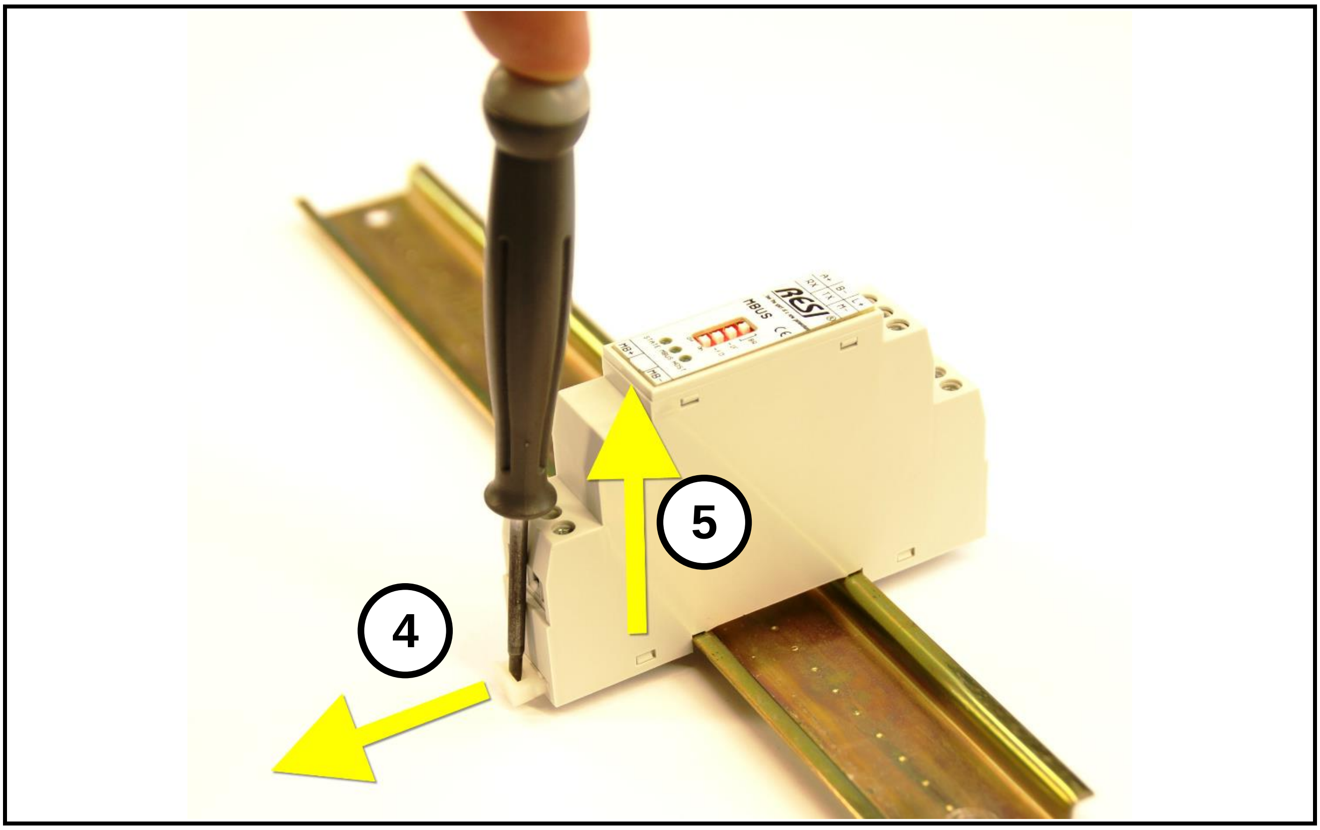
At first, put the modules with the top side on the DIN rail (1).



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.



7.4 Connection diagram

In the below drawing you will find the correct cabling of the IO module.

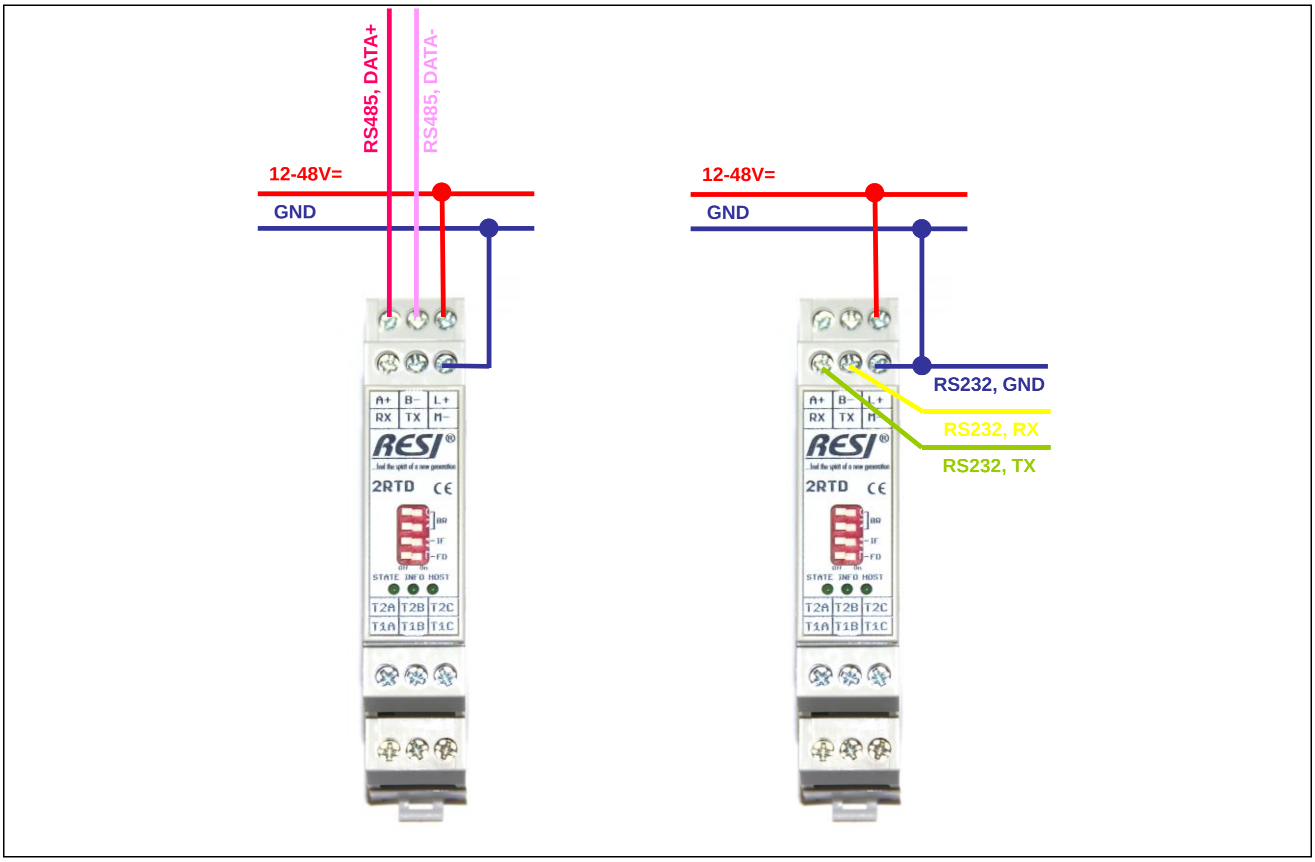


Illustration: Cabling of the serial bus interface of the IO module

7.5 Clamps

The IO module offers the following clamps:

CLAMP	DESCRIPTION
L+ M-	Power supply: L+: 12-48 V= M-: Ground
RS485 A+ B- M-	RS485 ASCII or MODBUS/RTU interface A+: RS485 DATA+ signal B-: RS485 DATA- signal M-: RS485 ground signal
RS232 TX+ RX- M-	RS232 ASCII or MODBUS/RTU interface TX+: RS232 Transmit signal RX-: RS232 Receive signal M-: RS232 Ground signal
Sensor 1 T1A T1B T1C	Sensor input 1 for 2 wire, 3 wire or 4 wire temperature measurement **2 wire sensor connection:** Sensor is cabled between T1C and T1B T1A: bridged with T1B T1B: bridged with T1A and sensor wire 2 (right cable of sensor) T1C: sensor wire 1 (left cable of sensor) **3 wire sensor connection:** Sensor is cabled between T1C, T1B and T1A T1A: Sensor cable 3 (right cable of sensor, 2nd cable) T1B: Sensor cable 2 (right cable of sensor, 1st cable) T1C: Sensor cable 1 (left cable of sensor) **4 wire sensor connection:** Sensor is cabled between T1C, T1B and T1A T1A: Sensor cable 4 (right cable of sensor, 2nd cable) T1B: Sensor cable 3 (right cable of sensor, 1st cable) T1C: Sensor cable 1+2 (both cables on the left side of sensor)
Sensor 2 T2A T2B T2C	Sensor input 2 for 2 wire, 3 wire or 4 wire temperature measurement **2 wire sensor connection:** Sensor is cabled between T1C and T1B T1A: bridged with T1B T1B: bridged with T1A and sensor wire 2 (right cable of sensor) T1C: sensor wire 1 (left cable of sensor) **3 wire sensor connection:** Sensor is cabled between T1C, T1B and T1A T1A: Sensor cable 3 (right cable of sensor, 2nd cable) T1B: Sensor cable 2 (right cable of sensor, 1st cable) T1C: Sensor cable 1 (left cable of sensor) **4 wire sensor connection:** Sensor is cabled between T1C, T1B and T1A T1A: Sensor cable 4 (right cable of sensor, 2nd cable) T1B: Sensor cable 3 (right cable of sensor, 1st cable) T1C: Sensor cable 1+2 (both cables on the left side of sensor)

Table: Description of the clamps on the IO module

7.6 DIP switch setting and LED indicators

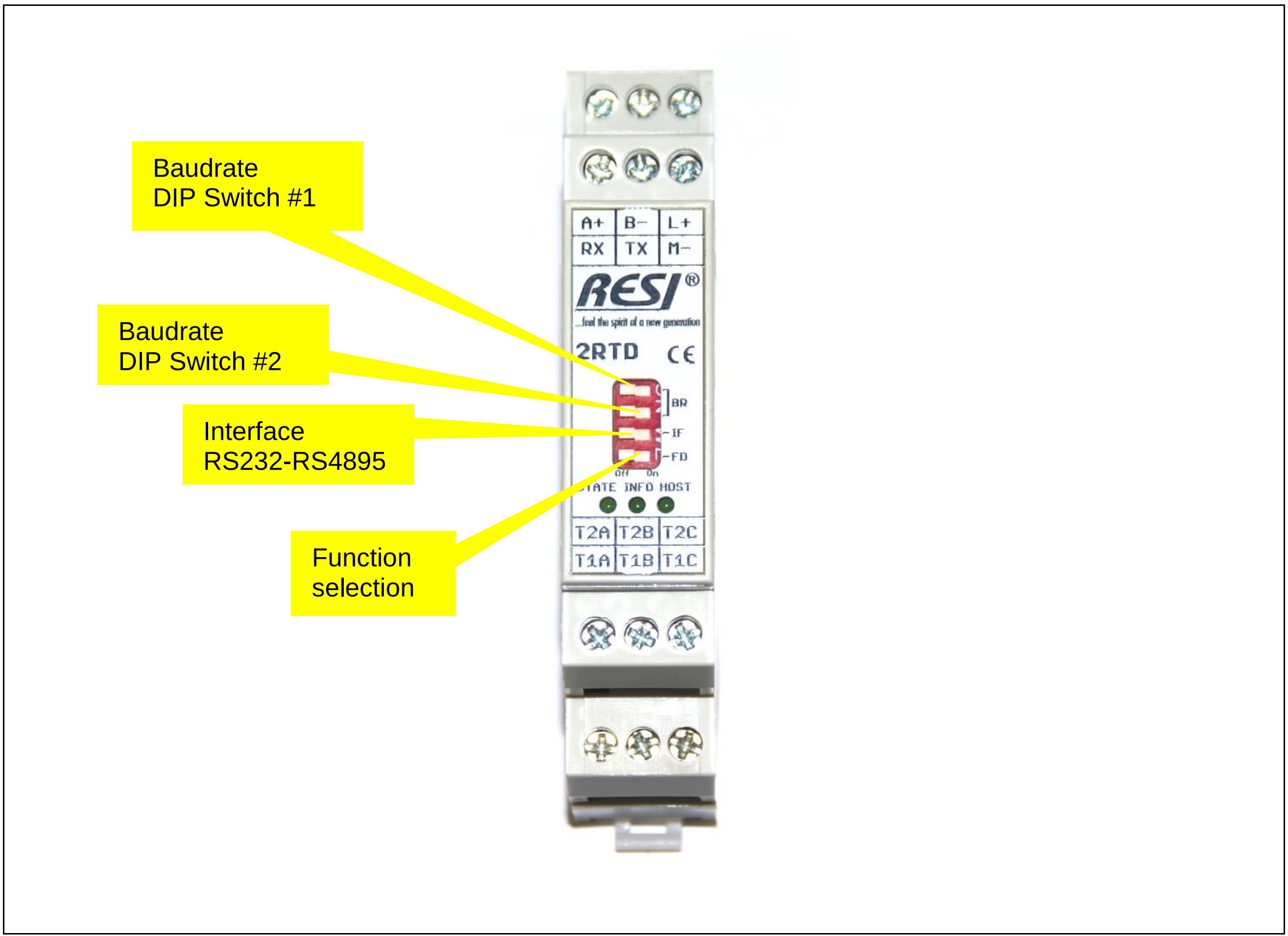


Illustration: Description of the DIP switch settings and LED indicators

DIP Switch	Description
Baudrate BR	Use DIP switches 1+2 to select the baud rate: OFF OFF: 9600Bd ON OFF: 19200Bd OFF ON: 38400Bd ON ON: 57600Bd HINT: The correct parity (NONE, EVEN, ODD) is defined by the PC software, not with the DIP switches.
Interface IF	Selects the physical type of the serial interface for the ASCII or MODBUS/RTU protocol: OFF=RS232 ON=RS485
Function definition FD	Select s special function in the module: OFF=The module uses the configured unit ID from the FLASH memory ON=The module uses always the unit ID 255
HINT	After a change of the DIP switches, the module reboots. No power off / power on cycle is necessary. After the reset all three LEDs are shortly on to represent the RESTART sequence.

Table: Description of the DIP switches of the IO module

LED	Description
STATE	Status LED: If the module is ok, this LED flashes slowly. If there is an error detected by the module, this LED flashes fast.
INFO	If everything is ok this LED is on. If there is an internal error with the temperature measurement, this LED flashes fast.
HOST	HOST-LED, Flashes, if the host is communicating with the module.

Table: Description of the LED indicator on the IO module

7.7 Dimensions of the module

In the below drawing you will see the dimensions of the module.

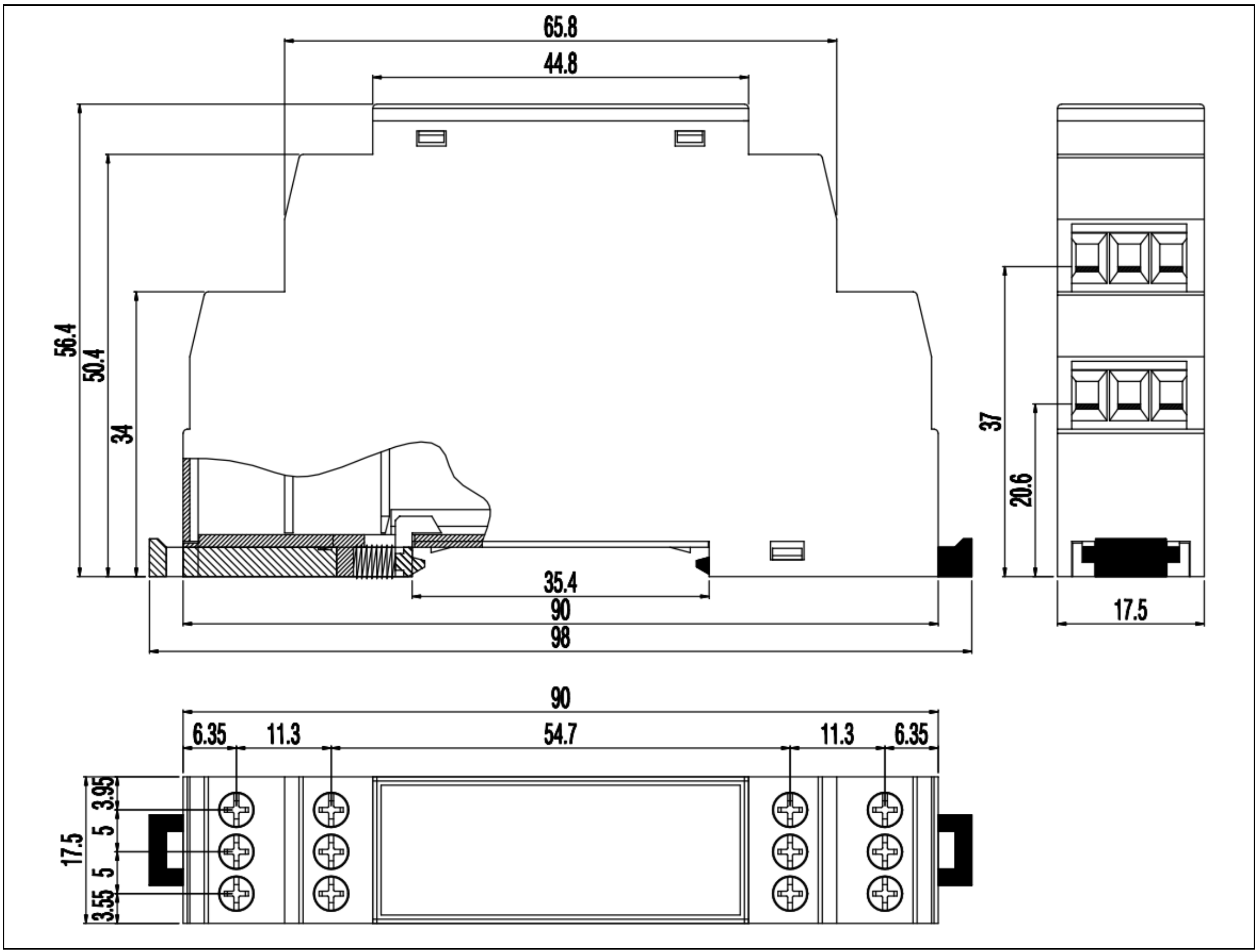


Illustration: dimension illustration in mm

Dimensions	
Enclosure dimensions L x W x H (mm)	17,5 x 90 x 58
Weight	65 g
Color	Grey RAL7035
Material	PA - UL 94 V0
Protection class	IP20 based on DIN 40050/EN 60529

Table: Data of enclosure

7.8 3D Drawing

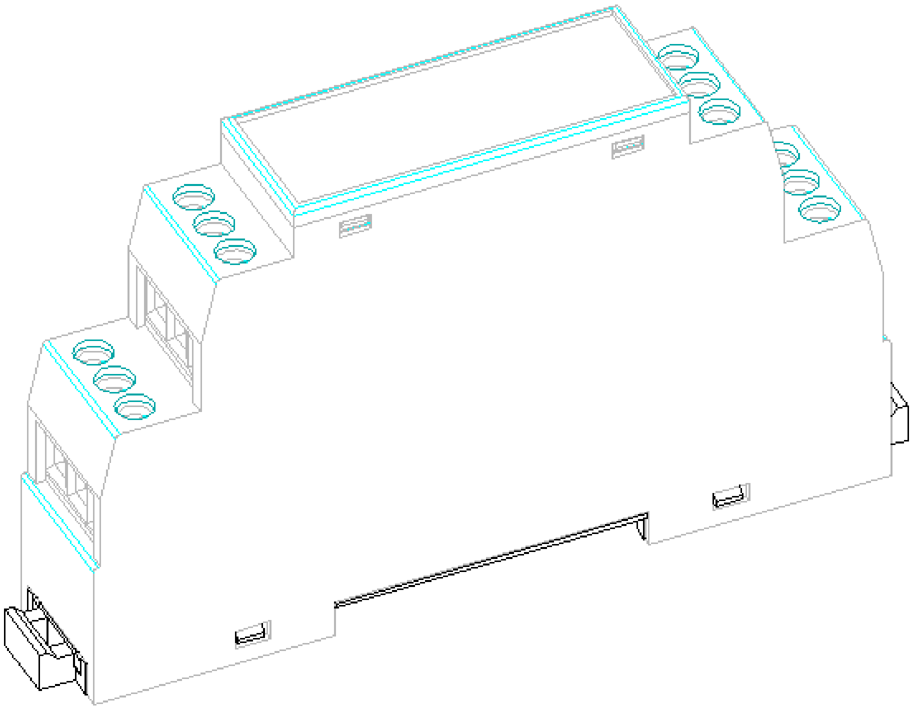


Illustration: 3D drawing of the enclosure

7.9 Power supply cabling of the module

In the image below you will see the correct cabling of the power supply of the module.

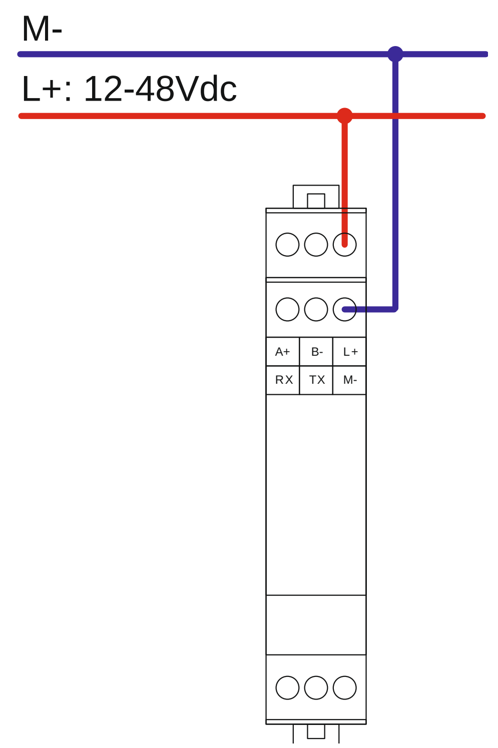


Illustration: Cabling of the power supply of the IO module

7.10 RS485 cabling of the IO module

In the image below you see the correct cabling of the RS485 interface of the IO module.

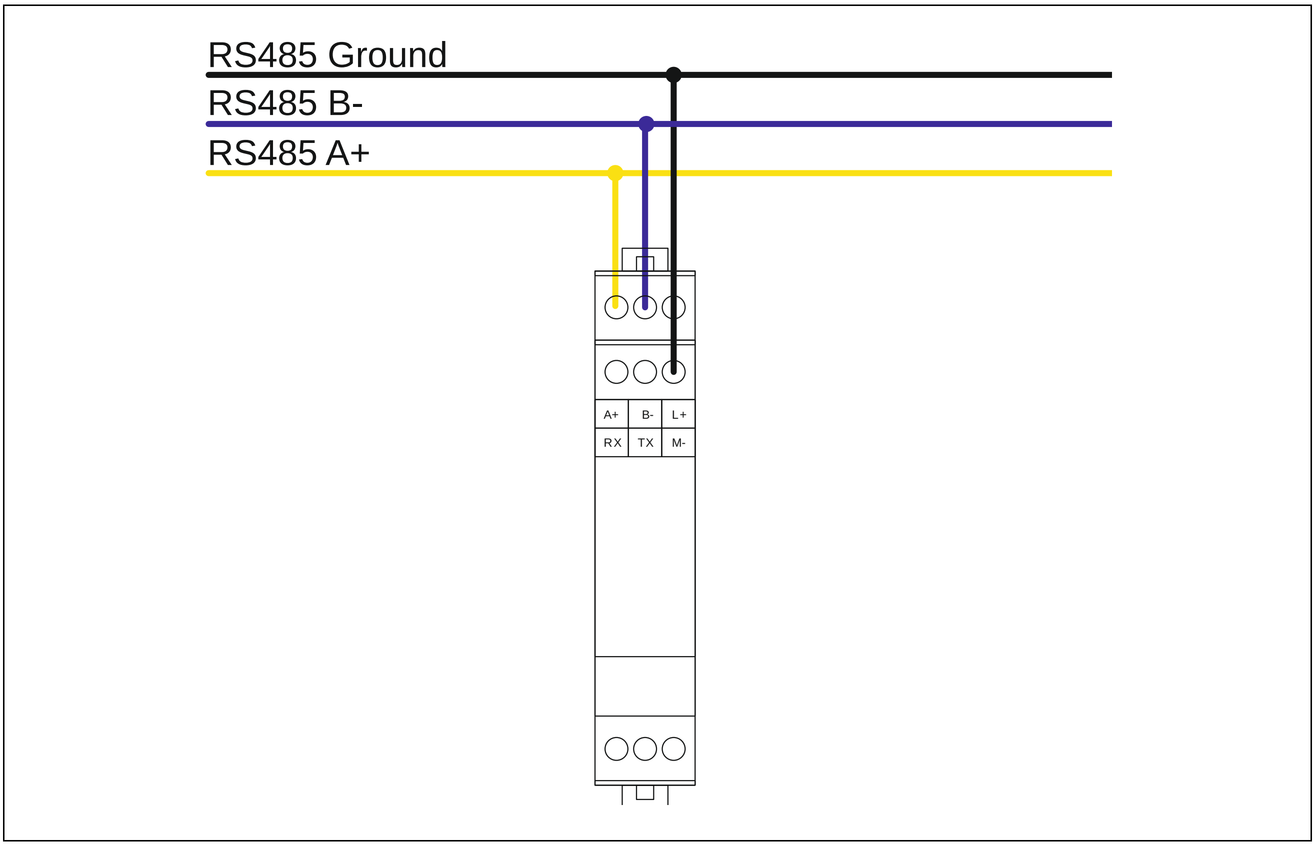


Illustration: RS485 bus cabling of the IO module

7.11 RS232 cabling of the IO module

In the image below you see the correct cabling of the RS232 interface of the IO module.

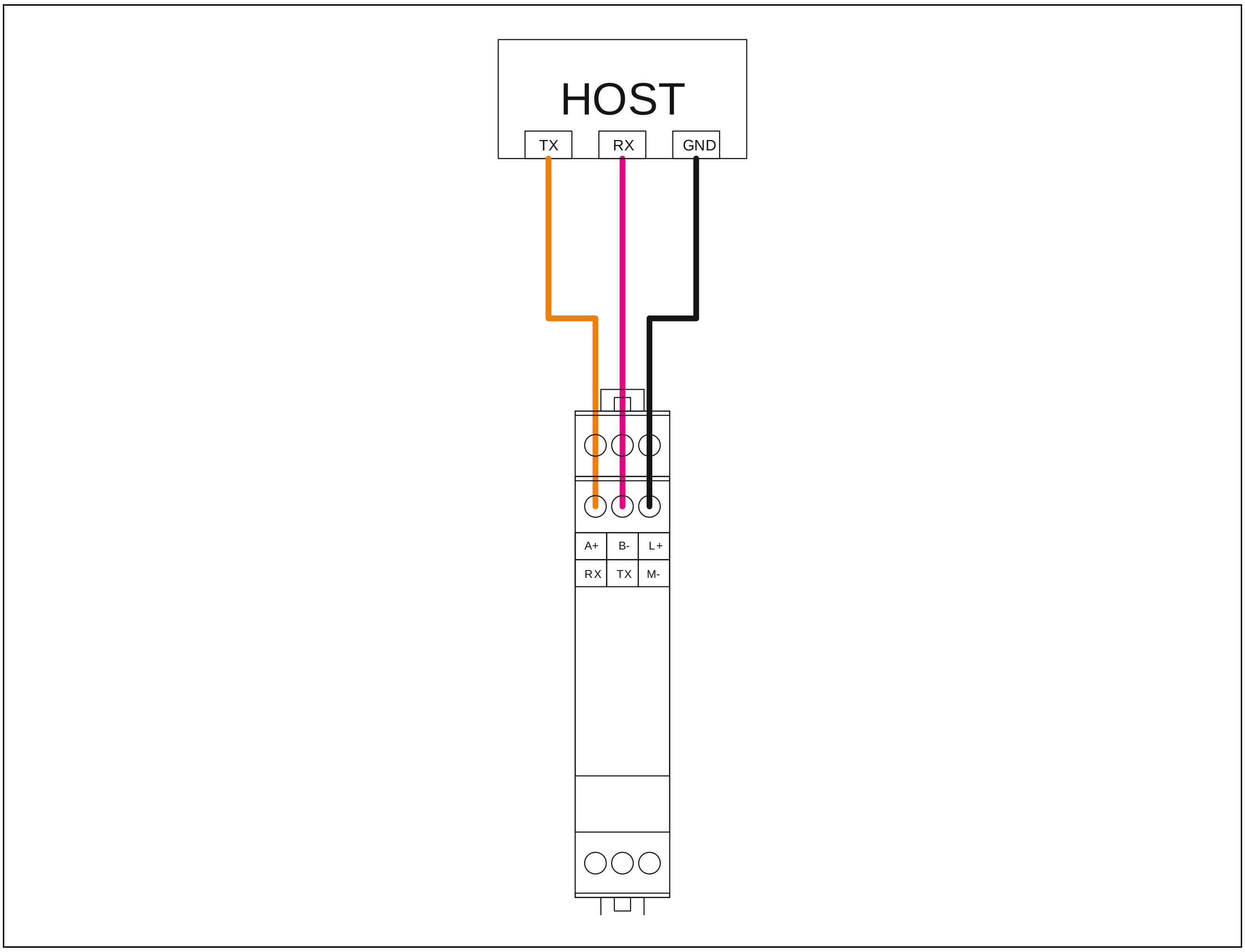


Illustration: RS232 bus cabling of the IO module

7.12 Cabling of temperature sensors

A typical temperature sensor with different connection cables is shown in the figure below:

- 2 wire: A red and white cable
- 3 wire: Two red and one white cable
- 4 wire: Two read and two white cable

The sensor element is always mounted between the red and white cables!

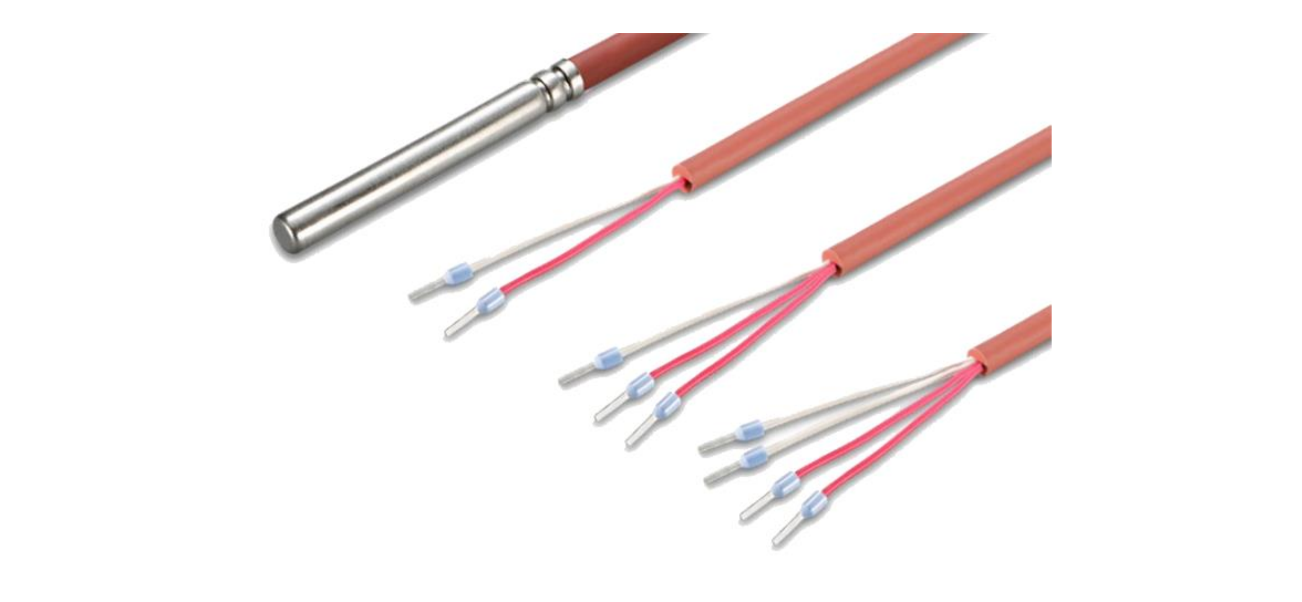


Illustration: typical temperature sensor with different connection cables

7.12.1 Wiring of 2 wire sensors

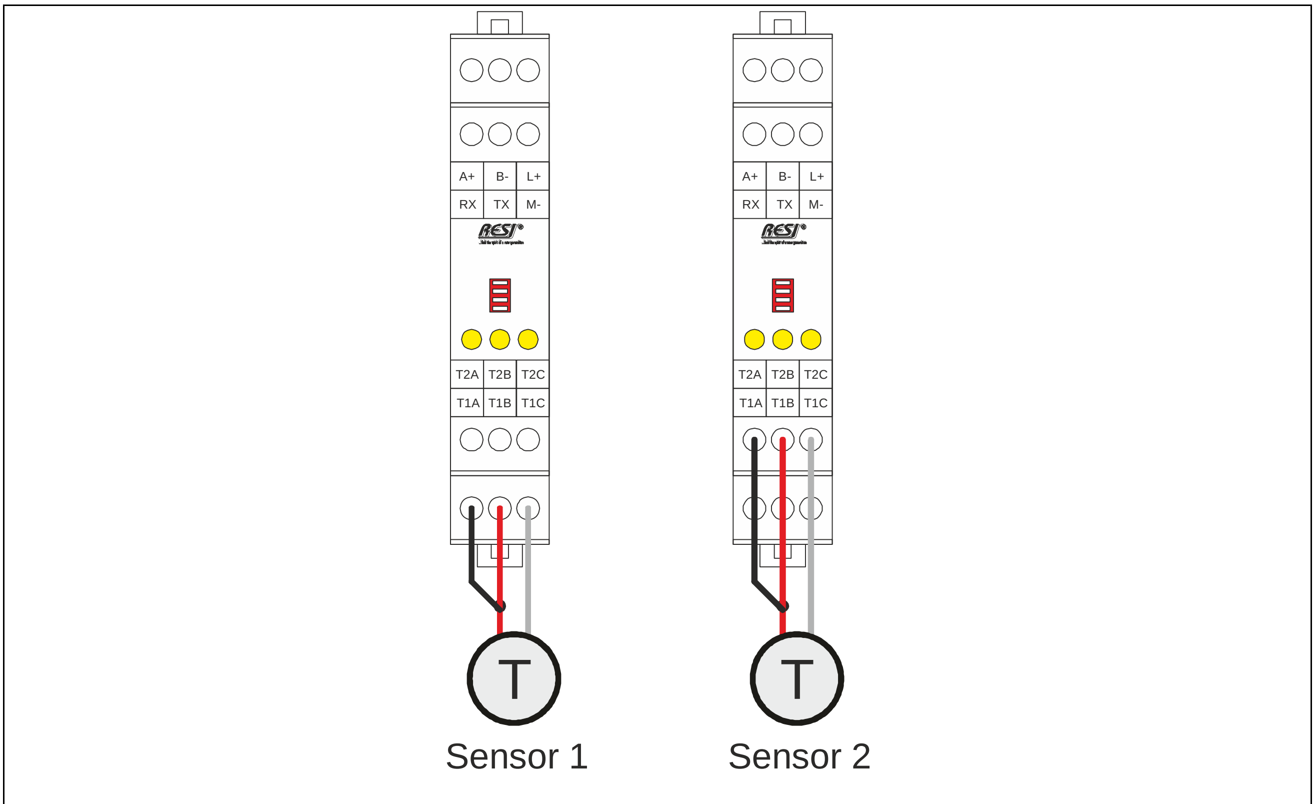


Illustration: Wiring of two 2 wire sensors to our module

IMPORTANT: Due to the reason, that our module is doing always a 3 wire measurement, you have to set always a bridge cable between the clamps TxA and TxB!

7.12.2 Wiring of 3 wire sensors

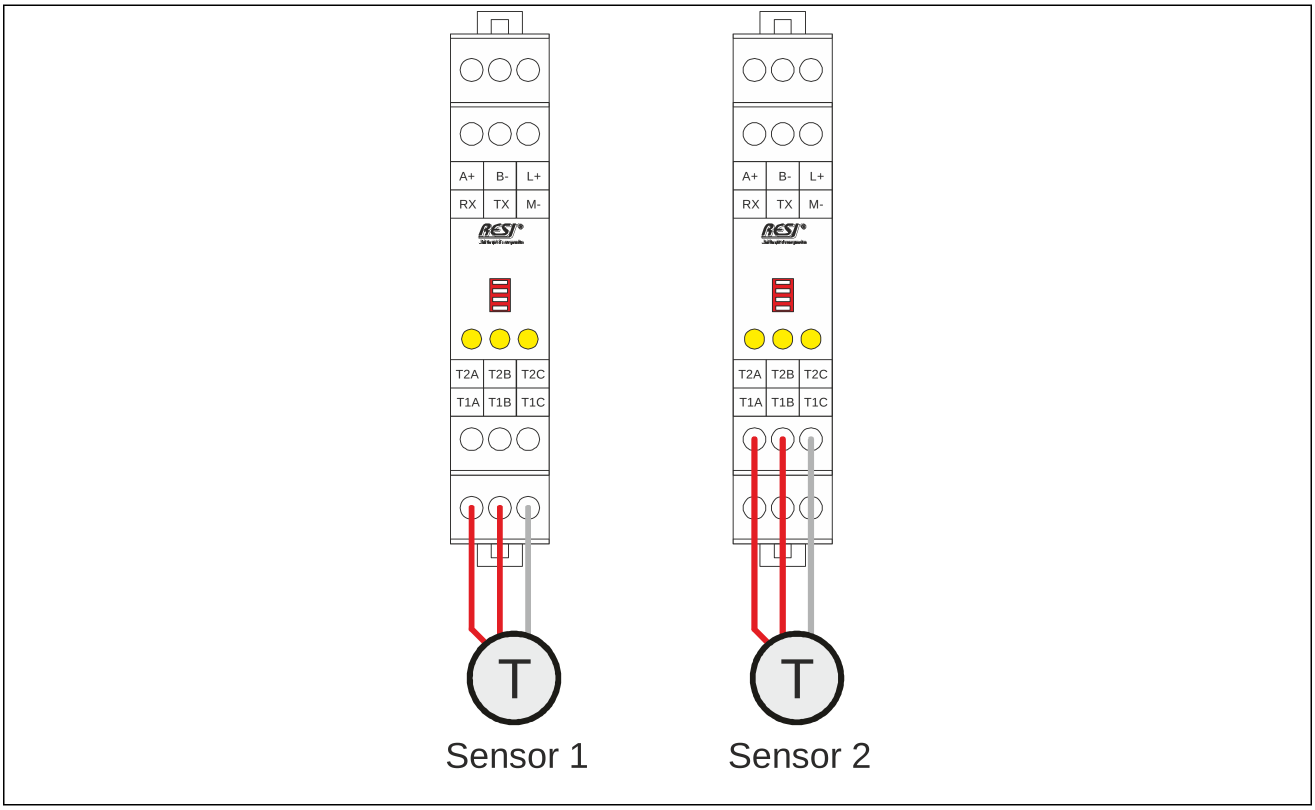


Illustration: Wiring of two 3 wire sensors to our module

7.12.3 Wiring of 4 wire sensors

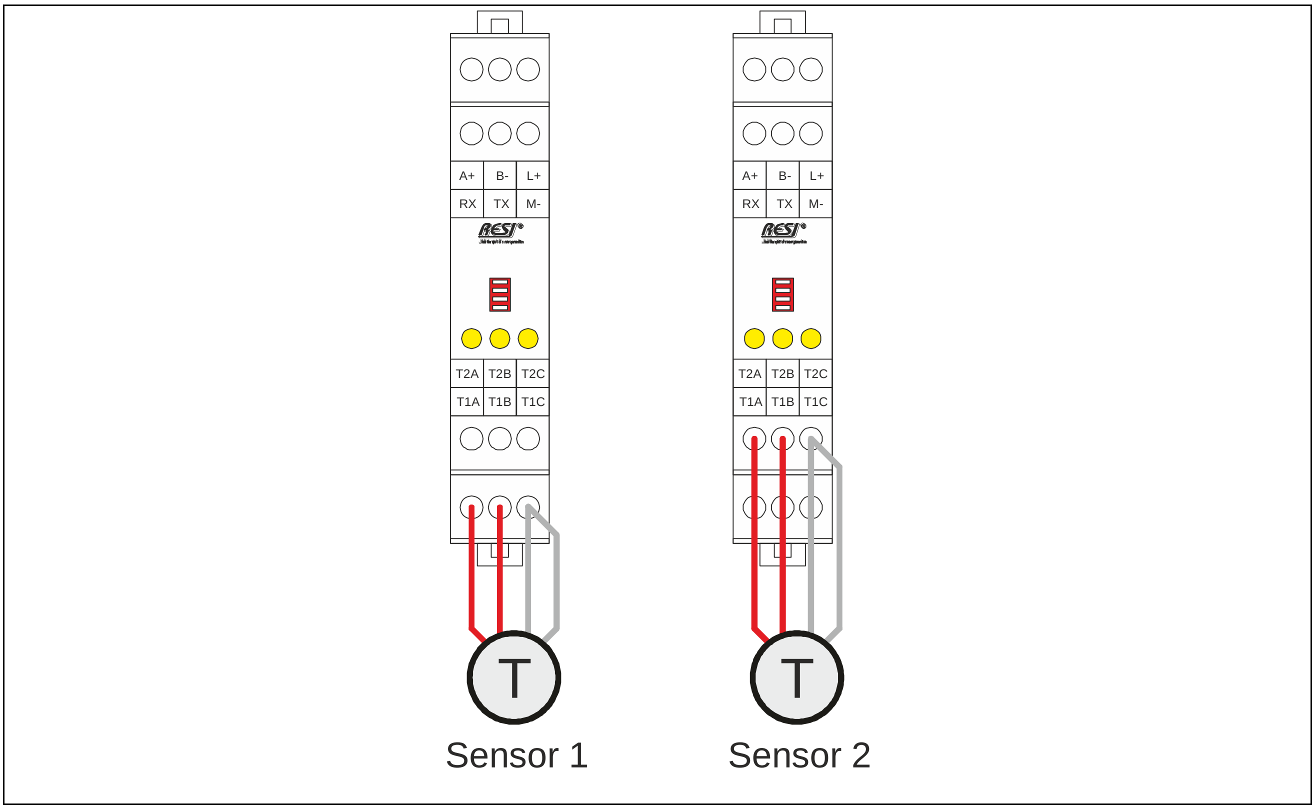


Illustration: Wiring of two 4 wire sensors to our module

7.13 Useable sensor types and measurement accuracy

This section describes the suitable sensors and explains the measurement accuracy of the sensor inputs of the module.

HINT: Use our free software RESI MODBUSConfigurator to configure and test our 2RTD module. You can also use your own software to handle the complete configuration while writing to MODBUS/RTU registers or with ASCII text commands.

7.13.1 Useable sensor types

The following types of sensors can be used per input:

Platin sensors:

- PT-100 sensors: Measurement range from 1.95Ω to 34.5Ω, -200°C to +850°C
- PT-1000 sensors: Measurement range from 195Ω to 3450Ω, -200°C to +850°C
- PT-1000 sensors with an α=0.00375: Measurement range from 195Ω to 3450Ω, -200°C to +850°C
- PT-10 sensors: Measurement range from 1.95Ω to 34.5Ω, -200°C to +850°C
- PT-50 sensors: Measurement range from 9.75Ω to 172.5Ω, -200°C to +850°C
- PT-200 sensors: Measurement range from 39Ω to 690Ω, -200°C to +850°C
- PT-500 sensors: Measurement range from 97.5Ω to 1725Ω, -200°C to +850°C

Nickel sensors:

- NI-120 sensors: Measurement range from 66.6Ω to 380.3Ω, -80°C to +260°C

Each of the two sensor inputs of the module can measure a different sensor type!

You can use all sensor accuracy classes (class AA, A, B, C). Please consult the DIN EN 60751:2009-05 for an exact definition of the sensor accuracy. Don't forget, that the whole measurement error for the temperature measurement consists always out of the error of the sensor element itself, the error of the used cabling and the measurement errors of the measurement electronic.

Out resistance measurement electronic uses an internal 2kΩ sense resistor. With an excitation current of 500µA the voltage drop on this resistor is 1V. This is the ideal range, to achieve the highest measurement accuracy. Use sensor type PT100, PT200, PT500, PT-1000 or NI-120 to achieve the best accuracy of our module with +/-0.1°C.

For PT10 and PT50 sensors this internal sense resistor is too big. So the reachable accuracy lies only about +/-3°C.

7.13.2 Configurable excitation current

For each input you can define an individual excitation current for the measurement:

- 5µA
- 10µA
- 25µA
- 50µA
- 100µA
- 250µA
- 500µA
- 1mA

The electronic executes an internal reference measurement on an Rsense resistor with 2kΩ (Accuracy +/-0.05%). Please adjust the excitation current for each channel in a way, that the resulting maximum voltage drop on this internal Rsense resistor <=1.0V.

U=R*I -> U=2kΩ*500µA -> 1V

This results in a maximum excitation current of 500µA with this module. If the excitation current exceeds this voltage range, the module signals this error with „ADC-Out-of-Range“ in the status flags of each channel.

The ideal excitation current of the module is 500µA! With smaller excitation currents the measurement will be more and more inaccurate!

7.13.3 Selectable linearization standard

A PLATIN resistor (PT sensor) is defined with a standardized characteristic. This is the Callendar-Van Dusen equation:

This is defined as follows:

RT = R0 • (1 + a • T + b • T2 + (T – 100°C) • c • T3) for T < 0°C,
RT = R0 • (1 + a • T + b • T2) for T > 0°C

The equation is used with different coefficients depending of the selected linearization standard to calculate a temperature from the measured resistor.

STANDARD	ALPHA (α)	a	b	c
Europe DIN EN 60751 IEC 751 JIS C1604-1997	α=0x00385	$3.908300*10^{-03}$	$–5.775000*10^{-07}$	$–4.183000*10^{-12}$
America SAMA Standard	α=0x003911	$3.969200*10^{-03}$	$–5.849500*10^{-07}$	$–4.232500*10^{-12}$
Japan JIS C1604-1987	α=0x003916	$3.973900*10^{-03}$	$–5.870000*10^{-07}$	$–4.400000*10^{-12}$
ITS-90	α=0x003926	$3.984800*10^{-03}$	$–5.870000*10^{-07}$	$–4.400000*10^{-12}$
RTD-1000-375	α=0x00375	$3.810200*10^{-03}$	$–6.018880*10^{-07}$	$–6.000000*10^{-12}$
NI-120	N/A	N/A	N/A	N/A

7.13.4 Sensor evaluation and accuracy

Our module computes the final temperature value °Celsius [°C] and delivers this temperature on various MODBUS registers in various number formats and via various ASCII commands to the host.

In addition our module can convert the temperature also in °Fahrenheit [°F] with the formula:

T[°F]=T[°C]*1.8+32

Also this temperature value can be read out with MODBUS Register or ASCII text commands. An own conversion on the host from Celsius in Fahrenheit is not necessary.

Also our module converts the temperature data into °Kelvin [°K] with the formula:

T[°K]=T[°C] +273.15

Also this temperature value can be read out with MODBUS Register or ASCII text commands. An own conversion on the host from Celsius in kelvin is not necessary.

Our module uses a 24 bit sigma/delta ADC with a noise suppression for 50/60Hz internally. Our module achieves a very high measurement accuracy of +/-0.1°C and a measurement resolution of +/-0.001°C!

Our module measures every channel around 1 time per second. In addition our module computes an average temperature for each channel with a user selectable time range in seconds, to suppress short noise signals in standard applications.

A manual adjustable zero offset allows a zero point shift to compensate static effects of the cabling, especially useful for 2 wire sensors.

Our module offers a very complex internal hardware to evaluate if the measured temperature is valid or not. Therefore the module offers for each channel a status representing the result of the last converted temperature. This status uses 8 bits, which have the following meaning:

Bit	NAME	DESCRIPTION
0	VALID	**=1**: If the measurement result is valid, this bit is set and all other bits in the status are 0! **=0**: if the system detects a conversion error or problem, this bit is 0 and the measurement result must be discarded!
1	ADC OUT OF RANGE	**=1**: If the product of 2kΩ * excitation current >1V, this bit is 1 and the measurement result is invalid. The absolute input voltage of the ACD beyond ±1.125 • VREF/2 **=0**: Everything is ok
2	SENSOR UNDER RANGE	**=1**: The current measured temperature is beyond the lower limit for the selected sensor type. For PT: -200°C, for NI-120: -80°C **=0**: Everything is ok
3	SENSOR OVER RANGE	**=1**: The current measured temperature is above the upper limit for the selected sensor type. For PT: +850°C, for NI-120: +260°C **=0**: Everything is ok
4	NOT USED	Ignore this bit
5	NOT USED	Ignore this bit
6	HARD ADC OUT OF RANGE	**=1**: Erroneous readout of the ADC value. A possibility is an extreme high noise level on the signal. The sensor value will be discarded. A second option is an open wiring for the sensor. **=0**: Everything is ok
7	SENSOR HARD FAULT	**=1**: Sensor wiring is open or no sensor is cabled to the module. Sensor has a shortcut or the internal sense resistor has an error. **=0**: Everything is ok

7.14 Functional description

This IO module communicates with a host system with the MODBUS/RTU slave protocol. All versions of the Io module with RESI-xxx-ASCII offer an additional protocol: An ASCII slave protocol with simple text string. The communication runs over a RS485 interface (half duplex) or over a RS232 interface (full duplex).

For the communication with ASCII texts, the host sends ASCII messages with a special start character # (0x23, 35dec) and a special end character (0x0d,13dec or CARRIAGE RETURN) to the module. The module uses also this special start and end characters to answer to the host request. Consult the below noted detailed command descriptions. In the ASCII protocol mode, the host can send messages with or without a bus number.

For communication with the MODBUS/RTU slave, the module offers the following MODBUS functions:

- 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)

HINT:
The functions READ HOLDING REGISTER and PRESET MULTIPLE REGISTERS are restricted to max. 125 register per request! The functions READ INPUT STATUS, READ COIL STATUS and FORCE MULTIPLE COILS are restricted to 2000 coils or inputs (bits) per request.

7.15 ASCII protocol description

7.15.1 Overview

The IO module communicates with simple ASCII commands. The following special characters are used in this documentation:

stand for the **Hashtag** ASCII character 35dec or 0x23
: stand for the **colon** ASCII character 58dec or 0x3A
= stand for the **equal sign** with the ASCII code 61ec or 0x3D
- stand for the **minus sign** with the ASCII code 45dec or 0x2D
, stands for the **comma** with the ASCII code 44dec or 0x2C
<CR> stand for **CARRIAGE RETURN** ASCII character 13dec or 0x0D. In the following text we use the representation CR.
<SP> stands for the **SPACE**. This is the space character with the ASCII code 32dec or 0x20. In the following text we will use the representation☐.
<ADR> will be used as the current **bus address** of the module. The bus number can be transmitted decimal or hexadecimal and is separated with a comma (ASCII code 44dec or 0x2C) from the rest of the command. Hexadecimal numbers always start with 0x. Its only allowed to use the ASCII characters ‚0‘-‚9‘, 48dec to 57dec, 0x30-0x39 and ‚A‘ to ‚F‘, 65dec to 70dec, 0x41-0x46. All modules react to the broadcast bus address 0 and to its own bus number. With a DIP switch, the user can easily change between the internal stored bus number in the FLASH and the fix bus number 255. Consult the DIP switch description for more details.

7.15.2 Communication sequence

In general the IO module sends no ASCII characters without a request from a host. So the host is the master of the communication and the IO module is always answering to host requests as a slave. If only one IO module is connected to a host (eg. Via RS232), you can dump the bus number in the protocol. If you use a RS485 interface, more than one module can be connected to the host. Therefore a bus number in the request frame of the host is always necessary.

The structure of the command look like this:

The host sends a command or a command with parameters without a bus address to the module:
#<Command><CR> or
#<Command>:<Parameter><CR>

The module answers, if it feels addressed, with the following answer telegram:
#<Answer><CR>

If the bus number is used, the module answers with:
#<ADR>,<Command><CR> or
#<ADR>,<Command>:<Parameter><CR>

The module answers with:
#<ADR>,<Answer><CR>

The bus address lies in the range of 1dec to 255dec or 0x00 to 0xFF hexadecimal. The setup is done with our free configuration software MODBUSConfigurator. Each module reacts also to the broadcast bus address 0.

For each command, we define two different writings. A long version and a short version, to avoid unnecessary traffic on the bus. For example to request the software version of the module you can use the command VERSION or the short command VER.

7.15.3 Request VERSION

This command returns the current software version of the module.

Host long version:
#VERSION<CR> or
#<ADR>,VERSION<CR>

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

Answer:
#VERSION:<HIGH>.<MED>.<LOW><CR> or
#<ADR>,VERSION:<HIGH>,<MED>,<LOW><CR>

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

Samples:
→ **#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**$_{CR}$

7.15.4 Request module TYPE

This command returns the current type of the module.

Host long version:
#TYPE<CR> or
#<ADR>,TYPE<CR>

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

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

<TYP> defines the current type of the module. Currently RESI-2RTD-ASCII

Samples:
→ **#TYPE**$_{CR}$
← **#TYPE:RESI-2RTD-ASCII**$_{CR}$

→ **#255,TYP**$_{CR}$
← **#255,TYPE:RESI-2RTD-ASCII**$_{CR}$

7.15.5 Table of all ASCII commands

Here you will find a possible ASCII commands of the module. We use here only the version with bus number. That you can avoid the bus number, we have discussed earlier in this document. If an argument as the extension Dec, it will be returned as a decimal number, If an argument has the extension Hex, then this argument is returned as a hexadecimal number. Many command returns the argument in decimal and hexadecimal representation. So the host can select, what kind of number conversion, it will handle in its software.

Direction	ASCII command
Host	**#<BusAdr>,VER$_{CR}$** **#<BusAdr>,VERSION$_{CR}$**
Answer	**#<BusAdr>,VERSION:<VersionHi>.< VersionMed>.<VersionLo>$_{CR}$**
	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)
Host	**#<BusAdr>,TYP$_{CR}$** **#<BusAdr>,TYPE$_{CR}$**
Answer	**#<BusAdr>,TYPE:RESI-1RO-ASCII$_{CR}$**
	Returns the current type of the module
Host	**#<BusAdr>,OWN$_{CR}$** **#<BusAdr>,OWNER$_{CR}$**
Answer	**#<BusAdr>,OWNER:RESI$_{CR}$**
	Returns the owner of the module
Host	**#<BusAdr>,CRE$_{CR}$** **#<BusAdr>,CREATOR$_{CR}$**
Answer	**#<BusAdr>,CREATOR:DI HC SIGL,MSC$_{CR}$**
	Returns the creator of the module
Host	**#<BusAdr>,COPY$_{CR}$** **#<BusAdr>,COPYRIGHT$_{CR}$**
Answer	**#<BusAdr>,COPYRIGHT:2016 BY RESI AND DI HC SIGL,MSC WWW.RESI.CC$_{CR}$**
	Returns a copyright note about the module
Host	**#<BusAdr>,GDIP$_{CR}$** **#<BusAdr>,GET☐DIP$_{CR}$**
Answer	**#<BusAdr>,GDIP:<DIPSwitchDec>,<DIPSwitchHex>$_{CR}$**
	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)

Direction	ASCII command
Host	**#<BusAdr>,GTS$_{CR}$** **#<BusAdr>,GET□TEMPS$_{CR}$**
Answer	**#<BusAdr>,GTS:<SENSOR1Dbl>,<SENSOR2Dbl>$_{CR}$**
	Returns the last measured valid temperatures on both channels as a floating point number. SENSOR1Dbl — The last valid measured temperature value of sensor 1 as floating point number with a . as a decimal point character. SENSOR2Dbl — The last valid measured temperature value of sensor 2 as floating point number with a . as a decimal point character. The temperature value is returned in the actual configured unit in register CHx_UNIT (°Celsius, °Fahrenheit or °Kelvin).
Host	**#<BusAdr>,GT1$_{CR}$** **#<BusAdr>,GET□TEMP1$_{CR}$**
Answer	**#<BusAdr>,GT1:<SENSOR1Dbl>$_{CR}$**
	Returns the last measured valid temperature on channel sensor 1 as a floating point number. SENSOR1Dbl — The last valid measured temperature value of sensor 1 as floating point number with a . as a decimal point character. The temperature value is returned in the actual configured unit in register CH1_UNIT (°Celsius, °Fahrenheit or °Kelvin).
Host	**#<BusAdr>,GT2$_{CR}$** **#<BusAdr>,GET□TEMP2$_{CR}$**
Answer	**#<BusAdr>,GT2:<SENSOR2Dbl>$_{CR}$**
	Returns the last measured valid temperature on channel sensor 2 as a floating point number. SENSOR2Dbl — The last valid measured temperature value of sensor 2 as floating point number with a . as a decimal point character. The temperature value is returned in the actual configured unit in register CH2_UNIT (°Celsius, °Fahrenheit or °Kelvin).

Direction	ASCII command
Host	**#<BusAdr>,GRTS$_{CR}$** **#<BusAdr>,GET□REAL□TEMPS$_{CR}$**
Answer	**#<BusAdr>,GRTS:<REALTEMP1Dbl>,<REALTEMP2Dbl>$_{CR}$**
	Returns the last measured temperature values on both sensor inputs as floating point numbers. The measured values can be erroneous or invalid measurement results or valid measurement results. REALTEMP1Dbl: The last temperature measurement result from sensor 1 as floating point number with a . for the decimal point. REALTEMP2Dbl: The last temperature measurement result from sensor 2 as floating point number with a . for the decimal point. The temperature value is returned in the actual configured unit in register CHx_UNIT (°Celsius, °Fahrenheit or °Kelvin).
Host	**#<BusAdr>,GRT1$_{CR}$** **#<BusAdr>,GET□REAL□TEMP1$_{CR}$**
Answer	**#<BusAdr>,GRT1:<REALTEMP1Dbl>$_{CR}$**
	Returns the last measured temperature values on sensor input 1 as a floating point number. The measured value can be an erroneous or invalid measurement result or a valid measurement result. REALTEMP1Dbl: The last temperature measurement result from sensor 1 as floating point number with a . for the decimal point. The temperature value is returned in the actual configured unit in register CH1_UNIT (°Celsius, °Fahrenheit or °Kelvin).
Host	**#<BusAdr>,GRT2$_{CR}$** **#<BusAdr>,GET□REAL□TEMP2$_{CR}$**
Answer	**#<BusAdr>,GRT2:<REALTEMP2Dbl>$_{CR}$**
	Returns the last measured temperature values on sensor input 2 as a floating point number. The measured value can be an erroneous or invalid measurement result or a valid measurement result. REALTEMP2Dbl: The last temperature measurement result from sensor 2 as floating point number with a . for the decimal point. The temperature value is returned in the actual configured unit in register CH2_UNIT (°Celsius, °Fahrenheit or °Kelvin).

Direction	ASCII command
Host	**#<BusAdr>,GATS$_{CR}$** **#<BusAdr>,GET☐AVG☐TEMPS$_{CR}$**
Answer	**#<BusAdr>,GATS:<AVGTEMP1Dbl>,<AVGTEMP2Dbl>$_{CR}$**
	Returns the last calculated average temperatures for both sensor inputs as floating point numbers. AVGTEMP1Dbl — The last calculated average temperature result for sensor 1 as floating point number with a . for the decimal point. AVGTEMP2Dbl — The last calculated average temperature result for sensor 2 as floating point number with a . for the decimal point. The temperature value is returned in the actual configured unit in register CHx_UNIT (°Celsius, °Fahrenheit or °Kelvin).
Host	**#<BusAdr>,GAT1$_{CR}$** **#<BusAdr>,GET☐AVG☐TEMP1$_{CR}$**
Answer	**#<BusAdr>,GAT1:<AVGTEMP1Dbl>$_{CR}$**
	Returns the last calculated average temperature for sensor input 1 as a floating point number. AVGTEMP1Dbl — The last calculated average temperature result for sensor 1 as floating point number with a . for the decimal point. The temperature value is returned in the actual configured unit in register CH1_UNIT (°Celsius, °Fahrenheit or °Kelvin).
Host	**#<BusAdr>,GAT2$_{CR}$** **#<BusAdr>,GET☐AVG☐TEMP2$_{CR}$**
Answer	**#<BusAdr>,GAT2:<AVGTEMP2Dbl>$_{CR}$**
	Returns the last calculated average temperature for sensor input 2 as a floating point number. AVGTEMP2Dbl — The last calculated average temperature result for sensor 2 as floating point number with a . for the decimal point. The temperature value is returned in the actual configured unit in register CH2_UNIT (°Celsius, °Fahrenheit or °Kelvin).

Direction	ASCII command
Host	**#<BusAdr>,GAIS$_{CR}$** **#<BusAdr>,GET□AVG□INTERVALS$_{CR}$**
Answer	**#<BusAdr>,GAIS:<AVGINTERVAL1Dec>,<AVGINTERVAL1Hex>, <AVGINTERVAL2Dec>,<AVGINTERVAL2Hex>$_{CR}$**
	Returns the current configured time span for the average calculation in Seconds as decimal or hexadecimal value for both sensor channels. AVGINTERVAL1Dec AVGINTERVAL1Hex — The configured time span for the average calculation for sensor input 1 in Seconds AVGINTERVAL2Dec AVGINTERVAL2Hex — The configured time span for the average calculation for sensor input 2 in Seconds
Host	**#<BusAdr>,GAI1$_{CR}$** **#<BusAdr>,GET□AVG□INTERVAL1$_{CR}$**
Answer	**#<BusAdr>,GAI1:<AVGINTERVAL1Dec>,<AVGINTERVAL1Hex>$_{CR}$**
	Returns the current configured time span for the average calculation in Seconds as decimal or hexadecimal value for sensor channel 1. AVGINTERVAL1Dec AVGINTERVAL1Hex — The configured time span for the average calculation for sensor input 1 in Seconds
Host	**#<BusAdr>,GAI2$_{CR}$** **#<BusAdr>,GET□AVG□INTERVAL2$_{CR}$**
Answer	**#<BusAdr>,GAI2:<AVGINTERVAL2Dec>,<AVGINTERVAL2Hex> $_{CR}$**
	Returns the current configured time span for the average calculation in Seconds as decimal or hexadecimal value for sensor channel 2. AVGINTERVAL2Dec AVGINTERVAL2Hex — The configured time span for the average calculation for sensor input 2 in Seconds

Direction	**ASCII command**
Host	**#<BusAdr>,SAIS:<AVGINTERVAL1>,<AVGINTERVAL2>$_{CR}$** **#<BusAdr>,SET☐AVG☐INTERVALS:<AVGINTERVAL1>,<AVGINTERVAL2>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
	Defines a new time interval for the average calculation in Seconds for both channels. AVGINTERVAL1 — The new time span for the average calculation on sensor input 1 in Seconds. AVGINTERVAL2 — The new time span for the average calculation on sensor input 2 in Seconds. All values are stored in the internal FLASH memory. The new values will be valid after a REBOOT of the module!
Host	**#<BusAdr>,SAI1:<AVGINTERVAL1>$_{CR}$** **#<BusAdr>,SET☐AVG☐INTERVAL1:<AVGINTERVAL1>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
	Defines a new time interval for the average calculation in Seconds for channel 1. AVGINTERVAL1 — The new time span for the average calculation on sensor input 1 in Seconds. This value is stored in the internal FLASH memory. The new value will be valid after a REBOOT of the module!
Host	**#<BusAdr>,SAI2:<AVGINTERVAL2>$_{CR}$** **#<BusAdr>,SET☐AVG☐INTERVAL2:<AVGINTERVAL2>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
	Defines a new time interval for the average calculation in Seconds for channel 2. AVGINTERVAL2 — The new time span for the average calculation on sensor input 2 in Seconds. This value is stored in the internal FLASH memory. The new value will be valid after a REBOOT of the module!

Direction	**ASCII command**
Host	**#<BusAdr>,GOTS$_{CR}$** **#<BusAdr>,GET☐OFFSET☐TEMPS$_{CR}$**
Answer	**#<BusAdr>,GOTS:<OFSTEMP1Dbl>,<OFSTEMP2Dbl>$_{CR}$**
	Returns the current configured zero offset values for both sensor channels as a floating point number. OFSTEMP1Dbl — The configured zero offset value for sensor input 1 as floating point number with a . as a comma sign. OFSTEMP2Dbl — The configured zero offset value for sensor input 2 as floating point number with a . as a comma sign. The offset values are returned in the actual configured unit in register CHx_UNIT (°Celsius, °Fahrenheit or °Kelvin).
Host	**#<BusAdr>,GOT1$_{CR}$** **#<BusAdr>,GET☐OFFSET☐TEMP1$_{CR}$**
Answer	**#<BusAdr>,GOT1:<OFSTEMP1Dbl>$_{CR}$**
	Returns the current configured zero offset values for sensor channel 1 as a floating point number. OFSTEMP1Dbl — The configured zero offset value for sensor input 1 as floating point number with a . as a comma sign. The offset value is returned in the actual configured unit in register CH1_UNIT (°Celsius, °Fahrenheit or °Kelvin).
Host	**#<BusAdr>,GOT2$_{CR}$** **#<BusAdr>,GET☐OFFSET☐TEMP2$_{CR}$**
Answer	**#<BusAdr>,GOT2:<OFSTEMP2Dbl>$_{CR}$**
	Returns the current configured zero offset values for sensor channel 1 as a floating point number. OFSTEMP2Dbl — The configured zero offset value for sensor input 2 as floating point number with a . as a comma sign. The offset value is returned in the actual configured unit in register CH2_UNIT (°Celsius, °Fahrenheit or °Kelvin).

Direction	ASCII command
Host	**#<BusAdr>,SOTS:<OFSTEMP1>,<OFSTEMP2>$_{CR}$** **#<BusAdr>,SET□OFFSET□TEMPS:<OFSTEMP1>,<OFSTEMP2>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
	Defines a new zero offset value for both sensor inputs as a temperature value in the current configured temperature unit. OFSTEMP1 — the new zero offset as a floating point number for sensor channel 1 with a . as a comma sign. OFSTEMP2 — the new zero offset as a floating point number for sensor channel 2 with a . as a comma sign. All values are stored in the internal FLASH memory. The new values will be valid after a REBOOT of the module!
Host	**#<BusAdr>,SOT1:<OFSTEMP1>$_{CR}$** **#<BusAdr>,SET□OFFSET□TEMP1:<OFSTEMP1>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
	Defines a new zero offset value for sensor input 1 as a temperature value in the current configured temperature unit. OFSTEMP1 — the new zero offset as a floating point number for sensor channel 1 with a . as a comma sign. This value is stored in the internal FLASH memory. The new value will be valid after a REBOOT of the module!
Host	**#<BusAdr>,SOT2:<OFSTEMP2>$_{CR}$** **#<BusAdr>,SET□OFFSET□TEMP2:<OFSTEMP2>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
	Defines a new zero offset value for sensor input 2 as a temperature value in the current configured temperature unit. OFSTEMP2 — the new zero offset as a floating point number for sensor channel 2 with a . as a comma sign. This value is stored in the internal FLASH memory. The new value will be valid after a REBOOT of the module!

Direction	ASCII command
Host	**#<BusAdr>,GSCS$_{CR}$** **#<BusAdr>,GET□SENSOR□CONFIGS$_{CR}$**
Answer	**#<BusAdr>,CSCS:** **S1,<S1Type>,<S1Current>,<S1Linearisation>,<S1Unit>,** **S2,<S2Type>,<S2Current>,<S2Linearisation>,<S2Unit>$_{CR}$**
	Show the current configuration of both sensor inputs: S1Type: The current type of the sensor: PT100 Platin 100Ω PT1000 Platin 1000Ω PT1000_375 Platin 1000Ω α=0.00375 PT10 Platin 10Ω PT50 Platin 50Ω PT200 Platin 200Ω PT500 Platin 500Ω NI120 Nickel 120Ω S1Current: The actual measurement current for the sensor 500MYA 500µA 1MA 1mA 5MYA 5µA 10MYA 10µA 20MYA 20µA 50MYA 50µA 100MYA 100µA 250MYA 250µA S1Linearisation: The actual linearization method for the sensor EUROPE AMERICA JAPAN ITS90 DONT_CARE S1Unit: The actual temperature unit for the sensor CELSIUS FAHRENHEIT KELVIN S2Type: like S1Type S2Current: like S1Current S2Linearisation: like S1Linearisation S2Unit: like S1Unit
Host	**#<BusAdr>,GSC1$_{CR}$** **#<BusAdr>,GET□SENSOR□CONFIG1$_{CR}$**
Answer	**#<BusAdr>,CSC1:<SType>,<SCurrent>,<SLinearisation>,<SUnit>$_{CR}$**
	Shows the current configuration of sensor input 1: SType: like S1Type SCurrent: like S1Current SLinearisation: like S1Linearisation SUnit: like S1Unit
Host	**#<BusAdr>,GSC2$_{CR}$** **#<BusAdr>,GET□SENSOR□CONFIG2$_{CR}$**
Answer	**#<BusAdr>,CSC2:<SType>,<SCurrent>,<SLinearisation>,<SUnit>$_{CR}$**
	Shows the current configuration of sensor input 2: Parameters like GET SENSOR CONFIG 1

Direction	**ASCII command**
Host	**#<BusAdr>,SSCS:** **S1,<S1Type>,<S1Current>,<S1Linearisation>,<S1Unit>,** **S2,<S2Type>,<S2Current>,<S2Linearisation>,<S2Unit>$_{CR}$** **#<BusAdr>,SET□SENSOR□CONFIGS:** **S1,<S1Type>,<S1Current>,<S1Linearisation>,<S1Unit>,** **S2,<S2Type>,<S2Current>,<S2Linearisation>,<S2Unit>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
	Defines a new configuration for both sensor inputs. The changes are valid after a REBOOT of the module. The configuration data will be written to the internal FLASH memory. Parameter like command GET SENSOR CONFIGS
Host	**#<BusAdr>,SSC1:** **<SType>,<SCurrent>,<SLinearisation>,<SUnit>$_{CR}$** **#<BusAdr>,SET□SENSOR□CONFIG1:** **<SType>,<SCurrent>,<SLinearisation>,<SUnit>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
	Defines a new configuration for sensor input 1. The changes are valid after a REBOOT of the module. The configuration data will be written to the internal FLASH memory. Parameter like command GET SENSOR CONFIG1
Host	**#<BusAdr>,SSC2:** **<SType>,<SCurrent>,<SLinearisation>,<SUnit>$_{CR}$** **#<BusAdr>,SET□SENSOR□CONFIG2:** **<SType>,<SCurrent>,<SLinearisation>,<SUnit>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
	Defines a new configuration for sensor input 2. The changes are valid after a REBOOT of the module. The configuration data will be written to the internal FLASH memory. Parameter like command GET SENSOR CONFIG2

Direction	ASCII command
Host	**#<BusAdr>,GSS$_{CR}$** **#<BusAdr>,GET□SENSOR□STATUS$_{CR}$**
Answer	**#<BusAdr>,GSS:<S1StatusDec>,<S1StatusHex>,<S2StatusDec>,<S2StatusHex>$_{CR}$**
	Returns the current status for both sensor inputs: S1StatusDec S1StatusHex Status for the first sensor input S2StatusDec S2StatusHex Status for the second sensor input **Explanation of status bits:** **Bit 0:VALID** **=1**: If the measurement result is valid, this bit is set and all other bits in the status are 0! **=0**: if the system detects a conversion error or problem, this bit is 0 and the measurement result must be discarded! **Bit 1:ADC OUT OF RANGE** **=1**: If the product of 2kΩ * excitation current >1V, this bit is 1 and the measurement result is invalid. The absolute input voltage of the ACD beyond ±1.125 • VREF/2 **=0**: Everything is ok **Bit 2: SENSOR UNDER RANGE** **=1**: The current measured temperature is beyond the lower limit for the selected sensor type. For PT: -200°C, for NI-120: -80°C **=0**: Everything is ok **Bit 3: SENSOR OVER RANGE** **=1**: The current measured temperature is above the upper limit for the selected sensor type. For PT: +850°C, for NI-120: +260°C **=0**: Everything is ok **Bit 4: NOT USED** Ignore this bit **Bit 5: NOT USED** Ignore this bit **Bit 6: HARD ADC OUT OF RANGE** **=1**: Erroneous readout of the ADC value. A possibility is an extreme high noise level on the signal. The sensor value will be discarded. A second option is an open wiring for the sensor. **=0**: Everything is ok **Bit 7: SENSOR HARD FAULT** **=1**: Sensor wiring is open or no sensor is cabled to the module. Sensor has a shortcut or the internal sense resistor has an error. **=0**: Everything is ok **Bits 8..15: ALWAYS ZERO** Are always 0

Direction	**ASCII command**
Host	**#<BusAdr>,GSS1$_{CR}$** **#<BusAdr>,GET□SENSOR□STATUS1$_{CR}$**
Answer	**#<BusAdr>,GSS1:<SStatusDec>,<SStatusHex>$_{CR}$**

Returns the status for the first sensor input 1.

SStatusDec
SStatusHex — Status of the first sensor channel 1

Explanation of status bits:
Bit 0:VALID
=1: If the measurement result is valid, this bit is set and all other bits in the status are 0!

=0: if the system detects a conversion error or problem, this bit is 0 and the measurement result must be discarded!

Bit 1:ADC OUT OF RANGE
=1: If the product of 2kΩ * excitation current >1V, this bit is 1 and the measurement result is invalid.

The absolute input voltage of the ACD beyond ±1.125 • VREF/2

=0: Everything is ok

Bit 2: SENSOR UNDER RANGE
=1: The current measured temperature is beyond the lower limit for the selected sensor type. For PT: -200°C, for NI-120: -80°C

=0: Everything is ok

Bit 3: SENSOR OVER RANGE
=1: The current measured temperature is above the upper limit for the selected sensor type. For PT: +850°C, for NI-120: +260°C

=0: Everything is ok

Bit 4: NOT USED
Ignore this bit

Bit 5: NOT USED
Ignore this bit

Bit 6: HARD ADC OUT OF RANGE
=1: Erroneous readout of the ADC value. A possibility is an extreme high noise level on the signal. The sensor value will be discarded. A second option is an open wiring for the sensor.

=0: Everything is ok

Bit 7: SENSOR HARD FAULT
=1: Sensor wiring is open or no sensor is cabled to the module. Sensor has a shortcut or the internal sense resistor has an error.

=0: Everything is ok
Bits 8..15: ALWAYS ZERO
Are always 0

Direction	**ASCII command**
Host	**#<BusAdr>,GSS2$_{CR}$** **#<BusAdr>,GET□SENSOR□STATUS2$_{CR}$**
Answer	**#<BusAdr>,GSS2:<SStatusDec>,<SStatusHex>$_{CR}$**
	Returns the status for the second sensor input 2. SStatusDec SStatusHex Status of the second sensor channel 2 **Explanation of status bits:** **Bit 0:VALID** **=1**: If the measurement result is valid, this bit is set and all other bits in the status are 0! **=0**: if the system detects a conversion error or problem, this bit is 0 and the measurement result must be discarded! **Bit 1:ADC OUT OF RANGE** **=1**: If the product of 2kΩ * excitation current >1V, this bit is 1 and the measurement result is invalid. The absolute input voltage of the ACD beyond ±1.125 • VREF/2 **=0**: Everything is ok **Bit 2: SENSOR UNDER RANGE** **=1**: The current measured temperature is beyond the lower limit for the selected sensor type. For PT: -200°C, for NI-120: -80°C **=0**: Everything is ok **Bit 3: SENSOR OVER RANGE** **=1**: The current measured temperature is above the upper limit for the selected sensor type. For PT: +850°C, for NI-120: +260°C **=0**: Everything is ok **Bit 4: NOT USED** Ignore this bit **Bit 5: NOT USED** Ignore this bit **Bit 6: HARD ADC OUT OF RANGE** **=1**: Erroneous readout of the ADC value. A possibility is an extreme high noise level on the signal. The sensor value will be discarded. A second option is an open wiring for the sensor. **=0**: Everything is ok **Bit 7: SENSOR HARD FAULT** **=1**: Sensor wiring is open or no sensor is cabled to the module. Sensor has a shortcut or the internal sense resistor has an error. **=0**: Everything is ok **Bits 8..15: ALWAYS ZERO** Are always 0

Direction	**ASCII command**
Host	**#<BusAdr>,SMBADR:<MBUnit>$_{CR}$** **#<BusAdr>,SET□MODBUS□ADDRESS:<MBUnit>$_{CR}$**
Answer	**#<BusAdr>,OK $_{CR}$**
	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.
Host	**#<BusAdr>,GMBADR$_{CR}$** **#<BusAdr>,GET□MODBUS□ADDRESS$_{CR}$**
Answer	**#<BusAdr>,GMBADR:<MBUnitDec>,<MBFLASHDec>,<MBUnitHex>,<MBFLASHHex>$_{CR}$**
	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. MBUnitDec MBUnitHex — The current used MODBUS/RTU unit or ASCII address for communication MBFLASHDec MBFLASHHex — The internal stored MODBUS/RTU unit address or ASCII address from the FLASH memory, if the DIP switch is 0.
Host	**#<BusAdr>,RST$_{CR}$** **#<BusAdr>,RESET$_{CR}$**
Answer	**None**
	Executes a software reset (Reboot) of the module.

7.16 MODBUS – register description

7.16.1 Table of inputs and coils

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 0x00001 to 0x65536. Inputs are usually noted with 1x00001 to 1x65536. 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: 0x00100 for the coil 100, 1x00100 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.

Register	Description
NONE	Currently no inputs and coils are available for this module

7.16.2 Table of holding/input registers

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.

7.16.2.1 Temperatures in the format SINT16*10

All temperatures are converted into the current configured temperature unit for the corresponding sensor input.

Register	Description
	SINT16 Temperature values a integer values with temperature*10
4x00001 3x00001 I:0 R/O CH1: VALID_TEMP	Current valid temperature of the 1st channel. Data type: SINT16 Value: temperature*10 Unit: in the temperature unit set by CH1_UNIT This is the last valid measured temperature. If there was no valid measurement in the past, the value -999.0 -> -9990 in °C will be returned.
4x00002 3x00002 I:1 R/O CH2: VALID_TEMP	Current valid temperature of the 2nd channel. Data type: SINT16 Value: temperature*10 Unit: in the temperature unit set by CH2_UNIT This is the last valid measured temperature. If there was no valid measurement in the past, the value -999.0 -> -9990 in °C will be returned.
4x00003 3x00003 I:2 R/O CH1: REAL_TEMP	Last measured temperature value for the 1st channel. Data type: SINT16 Value: temperature*10 Unit: in the temperature unit set by CH1_UNIT This is the last measured temperature on the ADC. If the was an erroneous conversion, this value will not be stored into the register VALID_TEMP. If the measurement result is completely invalid this register returns the value -999.0 -> -9990 in °C.
4x00004 3x00004 I:3 R/O CH2: REAL_TEMP	Last measured temperature value for the 2nd channel. Data type: SINT16 Value: temperature*10 Unit: in the temperature unit set by CH2_UNIT This is the last measured temperature on the ADC. If the was an erroneous conversion, this value will not be stored into the register VALID_TEMP. If the measurement result is completely invalid this register returns the value -999.0 -> -9990 in °C.

Register	Description
4x00005 3x00005 I:4 R/O CH1: AVG_TEMP	Last average temperature calculated for sensor channel 1. Data type: SINT16 Value: temperature*10 Unit: in the temperature unit set by CH1_UNIT The module adds internally all values of the register VALID_TEMP_IN_C for a configured time span. After the time span has expired, the module calculates the average temperature and stores the result into this register. After a module reboot while the first time span is running this register delivers the value -999.0 -> -9990 in °C.
4x00006 3x00006 I:5 R/O CH2: AVG_TEMP	Last average temperature calculated for sensor channel 2. Data type: SINT16 Value: temperature*10 Unit: in the temperature unit set by CH2_UNIT The module adds internally all values of the register VALID_TEMP_IN_C for a configured time span. After the time span has expired, the module calculates the average temperature and stores the result into this register. After a module reboot while the first time span is running this register delivers the value -999.0 -> -9990 in °C.

7.16.2.2 Erklärung Statusbits

Register	Description
4x00007 3x00007 I:6 R/O CH1: STATUS	This registers delivers the current status of the last measurement of the 1st sensor channel. Data type: SINT16 Value: Each bit has an individual meaning **Bit 0:VALID** **=1**: If the measurement result is valid, this bit is set and all other bits in the status are 0! **=0**: if the system detects a conversion error or problem, this bit is 0 and the measurement result must be discarded! **Bit 1:ADC OUT OF RANGE** **=1**: If the product of 2kΩ * excitation current >1V, this bit is 1 and the measurement result is invalid. The absolute input voltage of the ACD beyond ±1.125 • VREF/2 **=0**: Everything is ok **Bit 2: SENSOR UNDER RANGE** **=1**: The current measured temperature is beyond the lower limit for the selected sensor type. For PT: -200°C, for NI-120: -80°C **=0**: Everything is ok **Bit 3: SENSOR OVER RANGE** **=1**: The current measured temperature is above the upper limit for the selected sensor type. For PT: +850°C, for NI-120: +260°C **=0**: Everything is ok **Bit 4: NOT USED** Ignore this bit **Bit 5: NOT USED** Ignore this bit **Bit 6: HARD ADC OUT OF RANGE** **=1**: Erroneous readout of the ADC value. A possibility is an extreme high noise level on the signal. The sensor value will be discarded. A second option is an open wiring for the sensor. **=0**: Everything is ok **Bit 7: SENSOR HARD FAULT** **=1**: Sensor wiring is open or no sensor is cabled to the module. Sensor has a shortcut or the internal sense resistor has an error. **=0**: Everything is ok **Bits 8..15: ALWAYS ZERO** Are always 0

Register	Description
4x00008 3x00008 I:7 R/O CH2: STATUS	This registers delivers the current status of the last measurement of the 2nd sensor channel. Data type: SINT16 Value: Each bit has an individual meaning **Bit 0:VALID** **=1**: If the measurement result is valid, this bit is set and all other bits in the status are 0! **=0**: if the system detects a conversion error or problem, this bit is 0 and the measurement result must be discarded! **Bit 1:ADC OUT OF RANGE** **=1**: If the product of 2kΩ * excitation current >1V, this bit is 1 and the measurement result is invalid. The absolute input voltage of the ACD beyond ±1.125 • VREF/2 **=0**: Everything is ok **Bit 2: SENSOR UNDER RANGE** **=1**: The current measured temperature is beyond the lower limit for the selected sensor type. For PT: -200°C, for NI-120: -80°C **=0**: Everything is ok **Bit 3: SENSOR OVER RANGE** **=1**: The current measured temperature is above the upper limit for the selected sensor type. For PT: +850°C, for NI-120: +260°C **=0**: Everything is ok **Bit 4: NOT USED** Ignore this bit **Bit 5: NOT USED** Ignore this bit **Bit 6: HARD ADC OUT OF RANGE** **=1**: Erroneous readout of the ADC value. A possibility is an extreme high noise level on the signal. The sensor value will be discarded. A second option is an open wiring for the sensor. **=0**: Everything is ok **Bit 7: SENSOR HARD FAULT** **=1**: Sensor wiring is open or no sensor is cabled to the module. Sensor has a shortcut or the internal sense resistor has an error. **=0**: Everything is ok **Bits 8..15: ALWAYS ZERO** Are always 0

7.16.2.3 Temperatures in the format SINT32*100000 0xAABBCCDD -> 0xAABB 0xCCDD

Register	Description
	SINT32 temperature values as integer numbers in format temperature*100000 **Word order: 0xAABBCCDD -> 1st word: 0xAABB 2nd word: 0xCCDD**
4x00101-102 3x00101-102 I:100-101 R/O CH1: VALID_TEMP	Current valid temperature of the 1st channel. Data type: SINT32 Value: temperature*100000 Unit: in the temperature unit set by CH1_UNIT This is the last valid measured temperature. If there was no valid measurement in the past, the value -999.0 -> -99900000 in °C will be returned.
4x00103-104 3x00103-104 I:102-103 R/O CH2: VALID_TEMP	Current valid temperature of the 2nd channel. Data type: SINT32 Value: temperature*100000 Unit: in the temperature unit set by CH2_UNIT This is the last valid measured temperature. If there was no valid measurement in the past, the value -999.0 -> -99900000 in °C will be returned.
4x00105-106 3x00105-106 I:104-105 R/O CH1: REAL_TEMP	Last measured temperature value for the 1st channel. Data type: SINT32 Value: temperature*100000 Unit: in the temperature unit set by CH1_UNIT This is the last measured temperature on the ADC. If the was an erroneous conversion, this value will not be stored into the register VALID_TEMP. If the measurement result is completely invalid this register returns the value -999.0 -> -99900000 in °C.
4x00107-108 3x00107-108 I:106-107 R/O CH2: REAL_TEMP	Last measured temperature value for the 2nd channel. Data type: SINT32 Value: temperature*100000 Unit: in the temperature unit set by CH2_UNIT This is the last measured temperature on the ADC. If the was an erroneous conversion, this value will not be stored into the register VALID_TEMP. If the measurement result is completely invalid this register returns the value -999.0 -> -99900000 in °C.
4x00109-110 3x00109-110 I:108-109 R/O CH1: AVG_TEMP	Last average temperature calculated for sensor channel 1. Data type: SINT32 Value: temperature*100000 Unit: in the temperature unit set by CH1_UNIT The module adds internally all values of the register VALID_TEMP_IN_C for a configured time span. After the time span has expired, the module calculates the average temperature and stores the result into this register. After a module reboot while the first time span is running this register delivers the value -999.0 -> -99900000 in °C.
4x00111-112 3x00111-112 I:110-111 R/O CH2: AVG_TEMP	Last average temperature calculated for sensor channel 2. Data type: SINT32 Value: temperature*100000 Unit: in the temperature unit set by CH2_UNIT The module adds internally all values of the register VALID_TEMP_IN_C for a configured time span. After the time span has expired, the module calculates the average temperature and stores the result into this register. After a module reboot while the first time span is running this register delivers the value -999.0 -> -99900000 in °C.

Register	Description
4x00113-114 3x00113-114 I:112-113 R/O CH1: STATUS	This registers delivers the current status of the last measurement of the 1st sensor channel. Data type: SINT32 Value: Each bit has an individual meaning For a detailed description of the meaning of each bit, please consult register 4x00007!
4x00115-116 3x00115-116 I:114-115 R/O CH2: STATUS	This registers delivers the current status of the last measurement of the 2nd sensor channel. Data type: SINT32 Value: Each bit has an individual meaning For a detailed description of the meaning of each bit, please consult register 4x00008!

7.16.2.4 Temperatures in the format SINT32I*100000 0xAABBCCDD -> 0xCCDD 0xAABB

Register	Description
	SINT32I temperature values as integer numbers in format temperature*100000 **Word order: 0xAABBCCDD -> 1st word: 0xCCDD 2nd word: 0xAABB**
4x00201-202 3x00201-202 I:200-201 R/O CH1: VALID_TEMP	Current valid temperature of the 1st channel. Data type: SINT32I Value: temperature*100000 Unit: in the temperature unit set by CH1_UNIT This is the last valid measured temperature. If there was no valid measurement in the past, the value -999.0 -> -99900000 in °C will be returned.
4x00203-204 3x00203-204 I:202-203 R/O CH2: VALID_TEMP	Current valid temperature of the 2nd channel. Data type: SINT32I Value: temperature*100000 Unit: in the temperature unit set by CH2_UNIT This is the last valid measured temperature. If there was no valid measurement in the past, the value -999.0 -> -99900000 in °C will be returned.
4x00205-206 3x00205-206 I:204-205 R/O CH1: REAL_TEMP	Last measured temperature value for the 1st channel. Data type: SINT32I Value: temperature*100000 Unit: in the temperature unit set by CH1_UNIT This is the last measured temperature on the ADC. If the was an erroneous conversion, this value will not be stored into the register VALID_TEMP. If the measurement result is completely invalid this register returns the value -999.0 -> -99900000 in °C.
4x00207-208 3x00207-208 I:206-207 R/O CH2: REAL_TEMP	Last measured temperature value for the 2nd channel. Data type: SINT32I Value: temperature*100000 Unit: in the temperature unit set by CH2_UNIT This is the last measured temperature on the ADC. If the was an erroneous conversion, this value will not be stored into the register VALID_TEMP. If the measurement result is completely invalid this register returns the value -999.0 -> -99900000 in °C.
4x00209-210 3x00209-210 I:208-209 R/O CH1: AVG_TEMP	Last average temperature calculated for sensor channel 1. Data type: SINT32I Value: temperature*100000 Unit: in the temperature unit set by CH1_UNIT The module adds internally all values of the register VALID_TEMP_IN_C for a configured time span. After the time span has expired, the module calculates the average temperature and stores the result into this register. After a module reboot while the first time span is running this register delivers the value -999.0 -> -99900000 in °C.
4x00211-212 3x00211-212 I:210-211 R/O CH2: AVG_TEMP	Last average temperature calculated for sensor channel 2. Data type: SINT32I Value: temperature*100000 Unit: in the temperature unit set by CH2_UNIT The module adds internally all values of the register VALID_TEMP_IN_C for a configured time span. After the time span has expired, the module calculates the average temperature and stores the result into this register. After a module reboot while the first time span is running this register delivers the value -999.0 -> -99900000 in °C.

Register	Description
4x00213-214 3x00213-214 I:212-213 R/O CH1: STATUS	This registers delivers the current status of the last measurement of the 1st sensor channel. Data type: SINT32I Value: Each bit has an individual meaning For a detailed description of the meaning of each bit, please consult register 4x00007!
4x00215-216 3x00215-216 I:214-215 R/O CH2: STATUS	This registers delivers the current status of the last measurement of the 2nd sensor channel. Data type: SINT32I Value: Each bit has an individual meaning For a detailed description of the meaning of each bit, please consult register 4x00008!

7.16.2.5 Temperatures in the format FLOAT32 0xAABBCCDD -> 0xAABB 0xCCDD

Register	Description
	FLOAT32 temperature values as floating point values **Word order: 0xAABBCCDD -> 1st word: 0xAABB 2nd word: 0xCCDD**
4x00301-302 3x00301-302 I:300-301 R/O CH1: VALID_TEMP	Current valid temperature of the 1st channel. Data type: FLOAT32 Value: temperature Unit: in the temperature unit set by CH1_UNIT This is the last valid measured temperature. If there was no valid measurement in the past, the value -999.0 in °C will be returned.
4x00303-304 3x00303-304 I:302-303 R/O CH2: VALID_TEMP	Current valid temperature of the 2nd channel. Data type: FLOAT32 Value: temperature Unit: in the temperature unit set by CH2_UNIT This is the last valid measured temperature. If there was no valid measurement in the past, the value -999.0 in °C will be returned.
4x00305-306 3x00305-306 I:304-305 R/O CH1: REAL_TEMP	Last measured temperature value for the 1st channel. Data type: FLOAT32 Value: temperature Unit: in the temperature unit set by CH1_UNIT This is the last measured temperature on the ADC. If the was an erroneous conversion, this value will not be stored into the register VALID_TEMP. If the measurement result is completely invalid this register returns the value -999.0 in °C.
4x00307-308 3x00307-308 I:306-307 R/O CH2: REAL_TEMP	Last measured temperature value for the 2nd channel. Data type: FLOAT32 Value: temperature Unit: in the temperature unit set by CH2_UNIT This is the last measured temperature on the ADC. If the was an erroneous conversion, this value will not be stored into the register VALID_TEMP. If the measurement result is completely invalid this register returns the value -999.0 in °C.
4x00309-310 3x00309-310 I:308-309 R/O CH1: AVG_TEMP	Last average temperature calculated for sensor channel 1. Data type: FLOAT32 Value: temperature Unit: in the temperature unit set by CH1_UNIT The module adds internally all values of the register VALID_TEMP_IN_C for a configured time span. After the time span has expired, the module calculates the average temperature and stores the result into this register. After a module reboot while the first time span is running this register delivers the value -999.0 in °C.
4x00311-312 3x00311-312 I:310-311 R/O CH2: AVG_TEMP	Last average temperature calculated for sensor channel 2. Data type: FLOAT32 Value: temperature Unit: in the temperature unit set by CH2_UNIT The module adds internally all values of the register VALID_TEMP_IN_C for a configured time span. After the time span has expired, the module calculates the average temperature and stores the result into this register. After a module reboot while the first time span is running this register delivers the value -999.0 in °C.

Register	Description
4x00313-314 3x00313-314 I:312-313 R/O CH1: STATUS	This registers delivers the current status of the last measurement of the 1st sensor channel. Data type: FLOAT32 Value: Each bit has an individual meaning For a detailed description of the meaning of each bit, please consult register 4x00007!
4x00315-316 3x00315-316 I:314-315 R/O CH2: STATUS	This registers delivers the current status of the last measurement of the 2nd sensor channel. Data type: FLOAT32 Value: Each bit has an individual meaning For a detailed description of the meaning of each bit, please consult register 4x00008!

7.16.2.6 Temperatures in the format FLOAT32I 0xAABBCCDD -> 0xCCDD 0xAABB

Register	Description
	FLOAT32I temperature values as floating point values **Word order: 0xAABBCCDD -> 1st word: 0xCCDD 2nd word: 0xAABB**
4x00321-322 3x00321-322 I:320-321 R/O CH1: VALID_TEMP	Current valid temperature of the 1st channel. Data type: FLOAT32I Value: temperature Unit: in the temperature unit set by CH1_UNIT This is the last valid measured temperature. If there was no valid measurement in the past, the value -999.0 in °C will be returned.
4x00323-324 3x00323-324 I:322-323 R/O CH2: VALID_TEMP	Current valid temperature of the 2nd channel. Data type: FLOAT32I Value: temperature Unit: in the temperature unit set by CH2_UNIT This is the last valid measured temperature. If there was no valid measurement in the past, the value -999.0 in °C will be returned.
4x00325-326 3x00325-326 I:324-325 R/O CH1: REAL_TEMP	Last measured temperature value for the 1st channel. Data type: FLOAT32I Value: temperature Unit: in the temperature unit set by CH1_UNIT This is the last measured temperature on the ADC. If the was an erroneous conversion, this value will not be stored into the register VALID_TEMP. If the measurement result is completely invalid this register returns the value -999.0 in °C.
4x00327-328 3x00327-328 I:326-327 R/O CH2: REAL_TEMP	Last measured temperature value for the 2nd channel. Data type: FLOAT32I Value: temperature Unit: in the temperature unit set by CH2_UNIT This is the last measured temperature on the ADC. If the was an erroneous conversion, this value will not be stored into the register VALID_TEMP. If the measurement result is completely invalid this register returns the value -999.0 in °C.
4x00329-330 3x00329-330 I:328-329 R/O CH1: AVG_TEMP	Last average temperature calculated for sensor channel 1. Data type: FLOAT32I Value: temperature Unit: in the temperature unit set by CH1_UNIT The module adds internally all values of the register VALID_TEMP_IN_C for a configured time span. After the time span has expired, the module calculates the average temperature and stores the result into this register. After a module reboot while the first time span is running this register delivers the value -999.0 in °C.
4x00331-332 3x00331-332 I:330-331 R/O CH2: AVG_TEMP	Last average temperature calculated for sensor channel 2. Data type: FLOAT32I Value: temperature Unit: in the temperature unit set by CH2_UNIT The module adds internally all values of the register VALID_TEMP_IN_C for a configured time span. After the time span has expired, the module calculates the average temperature and stores the result into this register. After a module reboot while the first time span is running this register delivers the value -999.0 in °C.

Register	Description
4x00333-334 3x00333-334 I:332-333 R/O CH1: STATUS	This registers delivers the current status of the last measurement of the 1st sensor channel. Data type: FLOAT32I Value: Each bit has an individual meaning For a detailed description of the meaning of each bit, please consult register 4x00007!
4x00335-336 3x00335-336 I:334-335 R/O CH2: STATUS	This registers delivers the current status of the last measurement of the 2nd sensor channel. Data type: FLOAT32I Value: Each bit has an individual meaning For a detailed description of the meaning of each bit, please consult register 4x00008!

7.16.2.7 Temperatures in the format DOUBLE64 0x1122334455667788 -> 0x7788 0x5566 0x3344 0x1122

Register	Description
	DOUBLE64 temperature values as floating point values **Word order: 0x1122334455667788** **-> 1st word: 0x7788 2nd word: 0x5566 3rd word: 0x3344 4th word: 0x1122**
4x00501-504 3x00501-504 I:500-503 R/O CH1: VALID_TEMP	Current valid temperature of the 1st channel. Data type: DOUBLE64 Value: temperature Unit: in the temperature unit set by CH1_UNIT This is the last valid measured temperature. If there was no valid measurement in the past, the value -999.0 in °C will be returned.
4x00505-508 3x00505-508 I:504-507 R/O CH2: VALID_TEMP	Current valid temperature of the 2nd channel. Data type: DOUBLE64 Value: temperature Unit: in the temperature unit set by CH2_UNIT This is the last valid measured temperature. If there was no valid measurement in the past, the value -999.0 in °C will be returned.
4x00509-512 3x00509-512 I:508-511 R/O CH1: REAL_TEMP	Last measured temperature value for the 1st channel. Data type: DOUBLE64 Value: temperature Unit: in the temperature unit set by CH1_UNIT This is the last measured temperature on the ADC. If the was an erroneous conversion, this value will not be stored into the register VALID_TEMP. If the measurement result is completely invalid this register returns the value -999.0 in °C.
4x00513-516 3x00513-516 I:512-515 R/O CH2: REAL_TEMP	Last measured temperature value for the 2nd channel. Data type: DOUBLE64 Value: temperature Unit: in the temperature unit set by CH2_UNIT This is the last measured temperature on the ADC. If the was an erroneous conversion, this value will not be stored into the register VALID_TEMP. If the measurement result is completely invalid this register returns the value -999.0 in °C.
4x00517-520 3x00517-520 I:516-519 R/O CH1: AVG_TEMP	Last average temperature calculated for sensor channel 1. Data type: DOUBLE64 Value: temperature Unit: in the temperature unit set by CH1_UNIT The module adds internally all values of the register VALID_TEMP_IN_C for a configured time span. After the time span has expired, the module calculates the average temperature and stores the result into this register. After a module reboot while the first time span is running this register delivers the value -999.0 in °C.
4x00521-524 3x00521-524 I:520-523 R/O CH2: AVG_TEMP	Last average temperature calculated for sensor channel 2. Data type: DOUBLE64 Value: temperature Unit: in the temperature unit set by CH2_UNIT The module adds internally all values of the register VALID_TEMP_IN_C for a configured time span. After the time span has expired, the module calculates the average temperature and stores the result into this register. After a module reboot while the first time span is running this register delivers the value -999.0 in °C.

Register	Description
4x00525-528 3x00525-528 I:524-527 R/O CH1: STATUS	This registers delivers the current status of the last measurement of the 1st sensor channel. Data type: DOUBLE64 Value: Each bit has an individual meaning For a detailed description of the meaning of each bit, please consult register 4x00007!
4x00529-532 3x00529-532 I:528-531 R/O CH2: STATUS	This registers delivers the current status of the last measurement of the 2nd sensor channel. Data type: DOUBLE64 Value: Each bit has an individual meaning For a detailed description of the meaning of each bit, please consult register 4x00008!

7.16.2.8 Temperatures in the format DOUBLE64I 0x1122334455667788 -> 0x1122 0x3344 0x5566 0x7788

Register	Description
	DOUBLE64I temperature values as floating point values **Word order: 0x1122334455667788** **-> 1st word: 0x1122 2nd word: 0x3344 3rd word: 0x5566 4th word: 0x7788**
4x00701-704 3x00701-704 I:700-703 R/O CH1: VALID_TEMP	Current valid temperature of the 1st channel. Data type: DOUBLE64I Value: temperature Unit: in the temperature unit set by CH1_UNIT This is the last valid measured temperature. If there was no valid measurement in the past, the value -999.0 in °C will be returned.
4x00705-708 3x00705-708 I:704-707 R/O CH2: VALID_TEMP	Current valid temperature of the 2nd channel. Data type: DOUBLE64I Value: temperature Unit: in the temperature unit set by CH2_UNIT This is the last valid measured temperature. If there was no valid measurement in the past, the value -999.0 in °C will be returned.
4x00709-712 3x00709-712 I:708-711 R/O CH1: REAL_TEMP	Last measured temperature value for the 1st channel. Data type: DOUBLE64I Value: temperature Unit: in the temperature unit set by CH1_UNIT This is the last measured temperature on the ADC. If the was an erroneous conversion, this value will not be stored into the register VALID_TEMP. If the measurement result is completely invalid this register returns the value -999.0 in °C.
4x00713-716 3x00713-716 I:712-715 R/O CH2: REAL_TEMP	Last measured temperature value for the 2nd channel. Data type: DOUBLE64I Value: temperature Unit: in the temperature unit set by CH2_UNIT This is the last measured temperature on the ADC. If the was an erroneous conversion, this value will not be stored into the register VALID_TEMP. If the measurement result is completely invalid this register returns the value -999.0 in °C.
4x00717-720 3x00717-720 I:716-719 R/O CH1: AVG_TEMP	Last average temperature calculated for sensor channel 1. Data type: DOUBLE64I Value: temperature Unit: in the temperature unit set by CH1_UNIT The module adds internally all values of the register VALID_TEMP_IN_C for a configured time span. After the time span has expired, the module calculates the average temperature and stores the result into this register. After a module reboot while the first time span is running this register delivers the value -999.0 in °C.
4x00721-724 3x00721-724 I:720-723 R/O CH2: AVG_TEMP	Last average temperature calculated for sensor channel 2. Data type: DOUBLE64I Value: temperature Unit: in the temperature unit set by CH2_UNIT The module adds internally all values of the register VALID_TEMP_IN_C for a configured time span. After the time span has expired, the module calculates the average temperature and stores the result into this register. After a module reboot while the first time span is running this register delivers the value -999.0 in °C.

Register	Description
4x00725-728 3x00725-728 I:724-727 R/O CH1: STATUS	This registers delivers the current status of the last measurement of the 1st sensor channel. Data type: DOUBLE64I Value: Each bit has an individual meaning For a detailed description of the meaning of each bit, please consult register 4x00007!
4x00729-732 3x00569-732 I:728-731 R/O CH2: STATUS	This registers delivers the current status of the last measurement of the 2nd sensor channel. Data type: DOUBLE64I Value: Each bit has an individual meaning For a detailed description of the meaning of each bit, please consult register 4x00008!

7.16.2.9 Additional internal registers

Register	**Description**
	Additional internal registers
4x00901-904 3x00901-904 I:900-903 R/O CH1: AVG_SUM	Current sum of the average calculation for the 1st channel. Data type: DOUBLE64 Value: temperature Unit: in °Celsius [°C] Memory: 0x11223344556677 -> 0x6677 0x4455 0x2233 0x1122 This is current temporary temperature sum for the average calculation.
4x00905-908 3x00905-908 I:904-907 R/O CH2: AVG_SUM	Current sum of the average calculation for the 2nd channel. Data type: DOUBLE64 Value: temperature Unit: in °Celsius [°C] Memory: 0x11223344556677 -> 0x6677 0x4455 0x2233 0x1122 This is current temporary temperature sum for the average calculation.
4x00909-912 3x00909-912 I:908-911 R/O CH1: AVG_SUM	Current sum of the average calculation for the 1st channel. Data type: DOUBLE64 Value: temperature Unit: in °Celsius [°C] Memory : 0x11223344556677 -> 0x1122 0x3344 0x5566 0x7788 This is current temporary temperature sum for the average calculation.
4x00909-912 3x00909-912 I:908-911 R/O CH2: AVG_SUM	Current sum of the average calculation for the 2nd channel. Data type: DOUBLE64 Value: temperature Unit: in °Celsius [°C] Memory : 0x11223344556677 -> 0x1122 0x3344 0x5566 0x7788 This is current temporary temperature sum for the average calculation.

Register	Description
	Additional internal registers
4x00913-914 3x00913-914 I:912-913 R/O CH1: AVG_COUNT	Current count of summated temperature values for the average temperature calculation for the 1st channel Data type: UINT32 Value: Count Unit: in pieces Memory: 0xAABBCCDD -> 0xCCDD 0xAABB This is the counter register for the average value calculation. Each time the average sum is updated this counter will be incremented by 1. At the end of the time span, the module divides the sum by this counter value to calculate the average temperature value.
4x00915-916 3x00915-916 I:914-915 R/O CH2: AVG_COUNT	Current count of summated temperature values for the average temperature calculation for the 2nd channel Data type: UINT32 Value: Count Unit: in pieces Speicher: 0xAABBCCDD -> 0xCCDD 0xAABB This is the counter register for the average value calculation. Each time the average sum is updated this counter will be incremented by 1. At the end of the time span, the module divides the sum by this counter value to calculate the average temperature value.
4x00917-918 3x00917-918 I:916-917 R/O CH1: AVG_COUNT	Current count of summated temperature values for the average temperature calculation for the 1st channel Data type: UINT32 Value: Count Unit: in pieces Speicher: 0xAABBCCDD -> 0xAABB 0xCCDD This is the counter register for the average value calculation. Each time the average sum is updated this counter will be incremented by 1. At the end of the time span, the module divides the sum by this counter value to calculate the average temperature value.
4x00919-920 3x00919-920 I:918-919 R/O CH2: AVG_COUNT	Current count of summated temperature values for the average temperature calculation for the 2nd channel Data type: UINT32 Value: Count Unit: in pieces Speicher: 0xAABBCCDD -> 0xAABB 0xCCDD This is the counter register for the average value calculation. Each time the average sum is updated this counter will be incremented by 1. At the end of the time span, the module divides the sum by this counter value to calculate the average temperature value.

7.16.2.10 Additional system registers

Register	Description
4x06001 3x06001 I:6000 W/O RESET SYSTEM	If the host writes to this register, the module executes a soft reset (reboot).
4x65222 3x65222 I:65221 R/W MODBUS UNIT ADDRESS	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.

7.16.2.11 Configuration registers for sensor 1

Register	Description
4x06021 3x06021 I:06020 R/W CH1: SENSOR_TYPE	This register defines the type of the connected sensor to sensor channel 1. Data type: UINT16 Format **Bit 0..3:CH1_TYPE:Sensor type:** 15,0:PT100 1:PT1000 2:PT1000 α=0.00375 3:PT10 4:PT50 5:PT200 6:PT500 7:NI120 **Bit 4..7:CH1_CURRENT:Excitation current:** 15,0:500µA 1:1mA 2:5µA 3:10µA 4:25µA 5:50µA 6:100µA 7:250µA **Bit 8..11:CH1_LINEARISATION:Linearization standard:** 15,0:Europe 1:America 2:Japan 3:ITS-90 4:DON'T CARE **Bit 12..15:CH1_UNIT:Display unit:** 15,0:°Celsius [°C] 1:°Fahrenheit [°F] 2:°Kelvin [°K] This value will be stored into an internal FLASH memory. The new setting will be valid after a REBOOT of the module! IMPORTANT: The internal FLASH memory cannot be written indefinitely!
4x06022-23 3x06022-23 I:6021-22 R/W CH1: ZERO_OFFSET	In this register you can set up a zero offset value to compensate a long cable as a SINT32 value in the format 0xAABBCCDD -> 1st word:0xCCDD 2nd word:0xAABB. The value represents a temperature value as an integer value in the format °C *100000! The offset -1.23456 will be -123456. Therefore you can define an offset with five digits after the comma! This value will be stored into an internal FLASH memory. The new setting will be valid after a REBOOT of the module! IMPORTANT: The internal FLASH memory cannot be written indefinitely!
4x06024-25 3x06024-25 I:6023-24 R/W CH1: AVG_INTERVAL	This register contains the time span in Seconds as an UINT32 value in the format 0xAABBCCDD -> 1st word:0xCCDD 2nd word:0xAABB for the average calculation of the 1st sensor channel This value will be stored into an internal FLASH memory. The new setting will be valid after a REBOOT of the module! IMPORTANT: The internal FLASH memory cannot be written indefinitely!

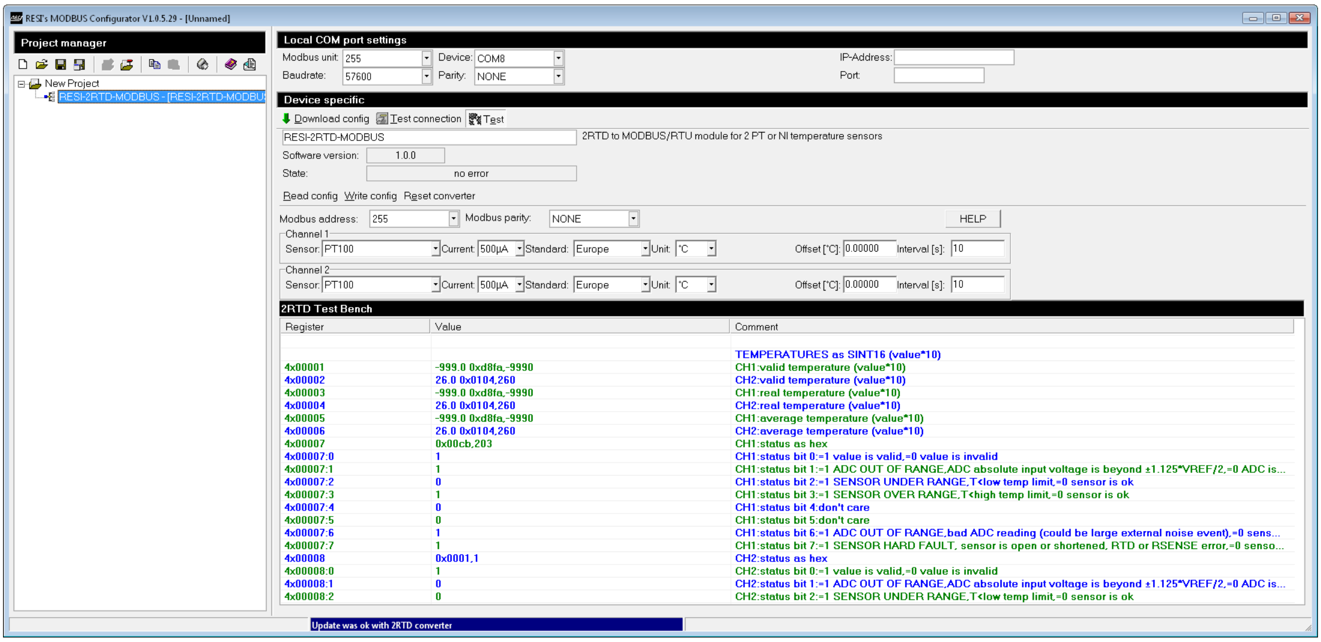
7.16.2.12 Configuration registers for sensor 2

Register	Description
4x06041 3x06041 I:06040 R/W CH2: SENSOR_TYPE	This register defines the type of the connected sensor to sensor channel 2. Data type: UINT16 Format **Bit 0..3:CH2_TYPE:Sensor type:** 15,0:PT100 1:PT1000 2:PT1000 α=0.00375 3:PT10 4:PT50 5:PT200 6:PT500 7:NI120 **Bit 4..7:CH2_CURRENT:Excitation current:** 15,0:500µA 1:1mA 2:5µA 3:10µA 4:25µA 5:50µA 6:100µA 7:250µA **Bit 8..11:CH2_LINEARISATION:Linearization standard:** 15,0:Europa 1:Amerika 2:Japan 3:ITS-90 4:DON'T CARE **Bit 12..15:CH2_UNIT:Display unit:** 15,0:°Celsius [°C] 1:°Fahrenheit [°F] 2:°Kelvin [°K] This value will be stored into an internal FLASH memory. The new setting will be valid after a REBOOT of the module! IMPORTANT: The internal FLASH memory cannot be written indefinitely!
4x06042-43 3x06042-43 I:6041-42 R/W CH2: ZERO_OFFSET	In this register you can set up a zero offset value to compensate a long cable as a SINT32 value in the format 0xAABBCCDD -> 1st word:0xCCDD 2nd word:0xAABB. The value represents a temperature value as an integer value in the format °C *100000! The offset -1.23456 will be -123456. Therefore you can define an offset with five digits after the comma! This value will be stored into an internal FLASH memory. The new setting will be valid after a REBOOT of the module! IMPORTANT: The internal FLASH memory cannot be written indefinitely!
4x06044-45 3x06044-45 I:6043-44 R/W CH2: AVG_INTERVAL	This register contains the time span in Seconds as an UINT32 value in the format 0xAABBCCDD -> 1st word:0xCCDD 2nd word:0xAABB for the average calculation of the 1st sensor channel This value will be stored into an internal FLASH memory. The new setting will be valid after a REBOOT of the module! IMPORTANT: The internal FLASH memory cannot be written indefinitely!

RESI®

7.17 Module test with RESI MODBUSConfigurator software

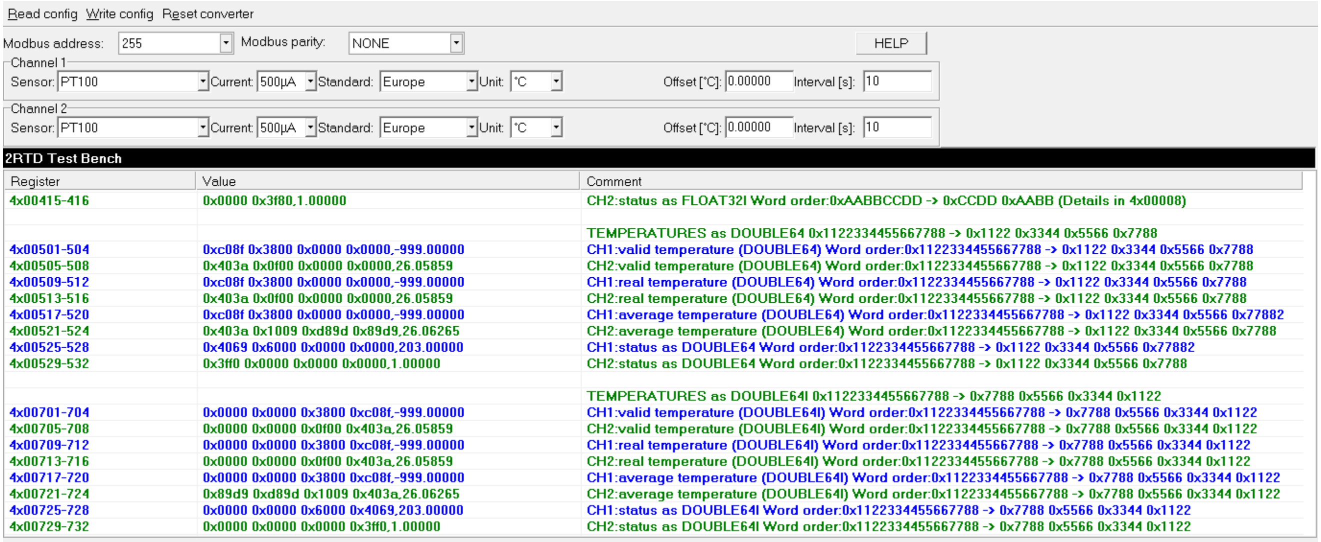
Establish a connection between the module and our software tool RESI MODBUSConfigurator. If this is successful, you get the following screen:



Establish the test mode while toggling the button “test”. The software refreshes the current values from the connected module of the list every 5 seconds.

Register	Value	Comment
		TEMPERATURES as SINT16 (value*10)
4x00001	-999.0 0xd8fa,-9990	CH1:valid temperature (value*10)
4x00002	26.0 0x0104,260	CH2:valid temperature (value*10)
4x00003	-999.0 0xd8fa,-9990	CH1:real temperature (value*10)
4x00004	26.0 0x0104,260	CH2:real temperature (value*10)
4x00005	-999.0 0xd8fa,-9990	CH1:average temperature (value*10)
4x00006	26.0 0x0104,260	CH2:average temperature (value*10)
4x00007	0x00cb,203	CH1:status as hex
4x00007:0	1	CH1:status bit 0:=1 value is valid,=0 value is invalid
4x00007:1	1	CH1:status bit 1:=1 ADC OUT OF RANGE,ADC absolute input voltage is beyond ±1.125*VREF/2,=0 ADC is...
4x00007:2	0	CH1:status bit 2:=1 SENSOR UNDER RANGE,T<low temp limit,=0 sensor is ok
4x00007:3	1	CH1:status bit 3:=1 SENSOR OVER RANGE,T<high temp limit,=0 sensor is ok
4x00007:4	0	CH1:status bit 4:don't care
4x00007:5	0	CH1:status bit 5:don't care
4x00007:6	1	CH1:status bit 6:=1 ADC OUT OF RANGE,bad ADC reading (could be large external noise event),=0 sens...
4x00007:7	1	CH1:status bit 7:=1 SENSOR HARD FAULT, sensor is open or shortened, RTD or RSENSE error,=0 senso...
4x00008	0x0001,1	CH2:status as hex
4x00008:0	1	CH2:status bit 0:=1 value is valid,=0 value is invalid
4x00008:1	0	CH2:status bit 1:=1 ADC OUT OF RANGE,ADC absolute input voltage is beyond ±1.125*VREF/2,=0 ADC is...
4x00008:2	0	CH2:status bit 2:=1 SENSOR UNDER RANGE,T<low temp limit,=0 sensor is ok
4x00008:3	0	CH2:status bit 3:=1 SENSOR OVER RANGE,T<high temp limit,=0 sensor is ok
4x00008:4	0	CH2:status bit 4:don't care
4x00008:5	0	CH2:status bit 5:don't care
4x00008:6	0	CH2:status bit 6:=1 ADC OUT OF RANGE,bad ADC reading (could be large external noise event),=0 sens...
4x00008:7	0	CH2:status bit 7:=1 SENSOR HARD FAULT, sensor is open or shortened, RTD or RSENSE error,=0 senso...
		CONFIG REGISTERS
4x06021	0xffff,65535	CH1:CONFIG SENSOR TYPE,bits 3..0:sensor type,7..4:excitation current,11..8:sensor standard
4x06021:3..0	15:PT100	CH1:SENSOR TYPE:bits 3..0:15,0:PT100,1:PT1000,2:PT1000,alpha=0.00375,3:PT10,4:PT50,5:PT200,6:PT5...
4x06021:7..4	15:500µA	CH1:EXCITATION CURRENT:bits 7..4:15,0:500µA,1:1mA,2:5µA,3:10µA,4:25µA,5:50µA,6:100µA,7:250µA
4x06021:11..8	15:Europe	CH1:SENSOR STANDARD:bits 11..8:15,0:Europe,1:America,2:Japan,3:ITS-90,4:DON'T CARE
4x06021:15..12	15:°Celsius [°C]	CH1:SENSOR UNIT:bits 15..12:15,0:°Celsius [°C],1:°Fahrenheit[°F],2:°Kelvin[°K]
4x06022-23	0xffffffff,-1,-0.00001	CH1:zero adjust[°C]:SINT32,Offset*10000,word order:0xAABBCCDD -> 0xCCDD 0xAABB
4x06024-25	0xffffffff,-1	CH1:AVG interval[s]:UINT32,word order:0xAABBCCDD -> 0xCCDD 0xAABB
4x06041	0xffff,65535	CH2:CONFIG SENSOR TYPE,Bits 3..0:sensor type,7..4:excitation current,11..8:sensor standard
4x06041:3..0	15:PT100	CH2:SENSOR TYPE:15,0:PT100,1:PT1000,2:PT1000,alpha=0.00375,3:PT10,4:PT50,5:PT200,6:PT500,7:NI120
4x06041:7..4	15:500µA	CH2:EXCITATION CURRENT:15,0:500µA,1:1mA,2:5µA,3:10µA,4:25µA,5:50µA,6:100µA,7:250µA
4x06041:11..8	15:Europe	CH2:SENSOR STANDARD:15,0:Europe,1:America,2:Japan,3:ITS-90,4:DON'T CARE
4x06041:15..12	15:°Celsius [°C]	CH2:SENSOR UNIT:bits 15..12:15,0:°Celsius [°C],1:°Fahrenheit[°F],2:°Kelvin[°K]
4x06042-43	0xffffffff,-1,-0.00001	CH2:zero adjust[°C]:SINT32,Offset*10000,word order:0xAABBCCDD -> 0xCCDD 0xAABB
4x06044-45	0xffffffff,-1	CH2:AVG interval[s]:UINT32,word order:0xAABBCCDD -> 0xCCDD 0xAABB
		TEMPERATURES as SINT32 (value*100000) 0xAABBCCDD -> 0xAABB 0xCCDD
4x00101-102	0xfa0ba5a0,-99900000,-999.00000	CH1:valid temperature as SINT32 (value*100000) Word order:0xAABBCCDD -> 0xAABB 0xCCDD
4x00103-104	0x0027c692,2606738,26.06738	CH2:valid temperature as SINT32 (value*100000) Word order:0xAABBCCDD -> 0xAABB 0xCCDD
4x00105-106	0xfa0ba5a0,-99900000,-999.00000	CH1:real temperature as SINT32 (value*100000) Word order:0xAABBCCDD -> 0xAABB 0xCCDD
4x00107-108	0x0027c692,2606738,26.06738	CH2:real temperature as SINT32 (value*100000) Word order:0xAABBCCDD -> 0xAABB 0xCCDD
4x00109-110	0xfa0ba5a0,-99900000,-999.00000	CH1:average temperature as SINT32 (value*100000) Word order:0xAABBCCDD -> 0xAABB 0xCCDD
4x00111-112	0x0027c5a9,2606505,26.06505	CH2:average temperature as SINT32 (value*100000) Word order:0xAABBCCDD -> 0xAABB 0xCCDD
4x00113-114	0x000000cb,203	CH1:status as hex Word order:0xAABBCCDD -> 0xAABB 0xCCDD (Details in 4x00007)
4x00115-116	0x00000001,1	CH2:status as hex Word order:0xAABBCCDD -> 0xAABB 0xCCDD (Details in 4x00008)
		TEMPERATURES as SINT32l (value*100000) 0xAABBCCDD -> 0xCCDD 0xAABB
4x00201-202	0xfa0ba5a0,-99900000,-999.00000	CH1:valid temperature as SINT32l (value*100000) Word order:0xAABBCCDD -> 0xCCDD 0xAABB
4x00203-204	0x0027c692,2606738,26.06738	CH2:valid temperature as SINT32l (value*100000) Word order:0xAABBCCDD -> 0xCCDD 0xAABB
4x00205-206	0xfa0ba5a0,-99900000,-999.00000	CH1:real temperature as SINT32l (value*100000) Word order:0xAABBCCDD -> 0xCCDD 0xAABB
4x00207-208	0x0027c692,2606738,26.06738	CH2:real temperature as SINT32l (value*100000) Word order:0xAABBCCDD -> 0xCCDD 0xAABB
4x00209-210	0xfa0ba5a0,-99900000,-999.00000	CH1:average temperature as SINT32l (value*100000) Word order:0xAABBCCDD -> 0xCCDD 0xAABB
4x00211-212	0x0027c5a9,2606505,26.06505	CH2:average temperature as SINT32l (value*100000) Word order:0xAABBCCDD -> 0xCCDD 0xAABB
4x00213-214	0x000000cb,203	CH1:status as hex Word order:0xAABBCCDD -> 0xCCDD 0xAABB (Details in 4x00007)
4x00215-216	0x00000001,1	CH2:status as hex Word order:0xAABBCCDD -> 0xCCDD 0xAABB (Details in 4x00008)
		TEMPERATURES as FLOAT32 0xAABBCCDD -> 0xAABB 0xCCDD
4x00301-302	0xc479 0xc000,-999.00000	CH1:valid temperature (FLOAT32) Word order:0xAABBCCDD -> 0xAABB 0xCCDD
4x00303-304	0x41d0 0x8a00,26.06738	CH2:valid temperature (FLOAT32) Word order:0xAABBCCDD -> 0xAABB 0xCCDD
4x00305-306	0xc479 0xc000,-999.00000	CH1:real temperature (FLOAT32) Word order:0xAABBCCDD -> 0xAABB 0xCCDD
4x00307-308	0x41d0 0x8a00,26.06738	CH2:real temperature (FLOAT32) Word order:0xAABBCCDD -> 0xAABB 0xCCDD
4x00309-310	0xc479 0xc000,-999.00000	CH1:average temperature (FLOAT32) Word order:0xAABBCCDD -> 0xAABB 0xCCDD
4x00311-312	0x41d0 0x853b,26.06505	CH2:average temperature (FLOAT32) Word order:0xAABBCCDD -> 0xAABB 0xCCDD
4x00313-314	0x434b 0x0000,203.00000	CH1:status as FLOAT32 Word order:0xAABBCCDD -> 0xAABB 0xCCDD (Details in 4x00007)
4x00315-316	0x3f80 0x0000,1.00000	CH2:status as FLOAT32 Word order:0xAABBCCDD -> 0xAABB 0xCCDD (Details in 4x00008)
		TEMPERATURES as FLOAT32l 0xAABBCCDD -> 0xCCDD 0xAABB
4x00401-402	0xc000 0xc479,-999.00000	CH1:valid temperature (FLOAT32l) Word order:0xAABBCCDD -> 0xCCDD 0xAABB
4x00403-404	0x8a00 0x41d0,26.06738	CH2:valid temperature (FLOAT32l) Word order:0xAABBCCDD -> 0xCCDD 0xAABB
4x00405-406	0xc000 0xc479,-999.00000	CH1:real temperature (FLOAT32l) Word order:0xAABBCCDD -> 0xCCDD 0xAABB
4x00407-408	0x8a00 0x41d0,26.06738	CH2:real temperature (FLOAT32l) Word order:0xAABBCCDD -> 0xCCDD 0xAABB
4x00409-410	0xc000 0xc479,-999.00000	CH1:average temperature (FLOAT32l) Word order:0xAABBCCDD -> 0xCCDD 0xAABB
4x00411-412	0x853b 0x41d0,26.06505	CH2:average temperature (FLOAT32l) Word order:0xAABBCCDD -> 0xCCDD 0xAABB
4x00413-414	0x0000 0x434b,203.00000	CH1:status as FLOAT32l Word order:0xAABBCCDD -> 0xCCDD 0xAABB (Details in 4x00007)
4x00415-416	0x0000 0x3f80,1.00000	CH2:status as FLOAT32l Word order:0xAABBCCDD -> 0xCCDD 0xAABB (Details in 4x00008)
		TEMPERATURES as DOUBLE64 0x1122334455667788 -> 0x1122 0x3344 0x5566 0x7788
4x00501-504	0xc08f 0x3800 0x0000 0x0000,-999.00000	CH1:valid temperature (DOUBLE64) Word order:0x1122334455667788 -> 0x1122 0x3344 0x5566 0x7788
4x00505-508	0x403a 0x1140 0x0000 0x0000,26.06738	CH2:valid temperature (DOUBLE64) Word order:0x1122334455667788 -> 0x1122 0x3344 0x5566 0x7788
4x00509-512	0xc08f 0x3800 0x0000 0x0000,-999.00000	CH1:real temperature (DOUBLE64) Word order:0x1122334455667788 -> 0x1122 0x3344 0x5566 0x7788
4x00513-516	0x403a 0x1140 0x0000 0x0000,26.06738	CH2:real temperature (DOUBLE64) Word order:0x1122334455667788 -> 0x1122 0x3344 0x5566 0x7788
4x00517-520	0xc08f 0x3800 0x0000 0x0000,-999.00000	CH1:average temperature (DOUBLE64) Word order:0x1122334455667788 -> 0x1122 0x3344 0x5566 0x77882
4x00521-524	0x403a 0x10a7 0x6276 0x2762,26.06505	CH2:average temperature (DOUBLE64) Word order:0x1122334455667788 -> 0x1122 0x3344 0x5566 0x7788
4x00525-528	0x4069 0x6000 0x0000 0x0000,203.00000	CH1:status as DOUBLE64 Word order:0x1122334455667788 -> 0x1122 0x3344 0x5566 0x77882
4x00529-532	0x3ff0 0x0000 0x0000 0x0000,1.00000	CH2:status as DOUBLE64 Word order:0x1122334455667788 -> 0x1122 0x3344 0x5566 0x7788
		TEMPERATURES as DOUBLE64l 0x1122334455667788 -> 0x7788 0x5566 0x3344 0x1122
4x00701-704	0x0000 0x0000 0x3800 0xc08f,-999.00000	CH1:valid temperature (DOUBLE64l) Word order:0x1122334455667788 -> 0x7788 0x5566 0x3344 0x1122
4x00705-708	0x0000 0x0000 0x1140 0x403a,26.06738	CH2:valid temperature (DOUBLE64l) Word order:0x1122334455667788 -> 0x7788 0x5566 0x3344 0x1122
4x00709-712	0x0000 0x0000 0x3800 0xc08f,-999.00000	CH1:real temperature (DOUBLE64l) Word order:0x1122334455667788 -> 0x7788 0x5566 0x3344 0x1122
4x00713-716	0x0000 0x0000 0x1140 0x403a,26.06738	CH2:real temperature (DOUBLE64l) Word order:0x1122334455667788 -> 0x7788 0x5566 0x3344 0x1122
4x00717-720	0x0000 0x0000 0x3800 0xc08f,-999.00000	CH1:average temperature (DOUBLE64l) Word order:0x1122334455667788 -> 0x7788 0x5566 0x3344 0x1122
4x00721-724	0x2762 0x6276 0x10a7 0x403a,26.06505	CH2:average temperature (DOUBLE64l) Word order:0x1122334455667788 -> 0x7788 0x5566 0x3344 0x1122
4x00725-728	0x0000 0x0000 0x6000 0x4069,203.00000	CH1:status as DOUBLE64l Word order:0x1122334455667788 -> 0x7788 0x5566 0x3344 0x1122
4x00729-732	0x0000 0x0000 0x0000 0x3ff0,1.00000	CH2:status as DOUBLE64l Word order:0x1122334455667788 -> 0x7788 0x5566 0x3344 0x1122

For the 2RTD modules the software offers a configuration area for defining the connected sensor type and various measurement options for each channel:
„Read config“: This function reads the current setup and configuration data for both sensors from the module and shows the setup in drop down lists.
„Write config“: This function write the modified setup data for the measurement from the PC software into the module and stores this data in the internal FLASH memory of the module.
„Reset converter“: This function reboots the connected module.



Register	Value	Comment
4x00415-416	0x0000 0x3f80,1.00000	CH2:status as FLOAT32I Word order:0xAABBCCDD -> 0xCCDD 0xAABB (Details in 4x00008)
		TEMPERATURES as DOUBLE64 0x1122334455667788 -> 0x1122 0x3344 0x5566 0x7788
4x00501-504	0xc08f 0x3800 0x0000 0x0000,-999.00000	CH1:valid temperature (DOUBLE64) Word order:0x1122334455667788 -> 0x1122 0x3344 0x5566 0x7788
4x00505-508	0x403a 0x0f00 0x0000 0x0000,26.05859	CH2:valid temperature (DOUBLE64) Word order:0x1122334455667788 -> 0x1122 0x3344 0x5566 0x7788
4x00509-512	0xc08f 0x3800 0x0000 0x0000,-999.00000	CH1:real temperature (DOUBLE64) Word order:0x1122334455667788 -> 0x1122 0x3344 0x5566 0x7788
4x00513-516	0x403a 0x0f00 0x0000 0x0000,26.05859	CH2:real temperature (DOUBLE64) Word order:0x1122334455667788 -> 0x1122 0x3344 0x5566 0x7788
4x00517-520	0xc08f 0x3800 0x0000 0x0000,-999.00000	CH1:average temperature (DOUBLE64) Word order:0x1122334455667788 -> 0x1122 0x3344 0x5566 0x77882
4x00521-524	0x403a 0x1009 0xd89d 0x89d9,26.06265	CH2:average temperature (DOUBLE64) Word order:0x1122334455667788 -> 0x1122 0x3344 0x5566 0x7788
4x00525-528	0x4069 0x6000 0x0000 0x0000,203.00000	CH1:status as DOUBLE64 Word order:0x1122334455667788 -> 0x1122 0x3344 0x5566 0x77882
4x00529-532	0x3ff0 0x0000 0x0000 0x0000,1.00000	CH2:status as DOUBLE64 Word order:0x1122334455667788 -> 0x1122 0x3344 0x5566 0x7788
		TEMPERATURES as DOUBLE64I 0x1122334455667788 -> 0x7788 0x5566 0x3344 0x1122
4x00701-704	0x0000 0x0000 0x3800 0xc08f,-999.00000	CH1:valid temperature (DOUBLE64I) Word order:0x1122334455667788 -> 0x7788 0x5566 0x3344 0x1122
4x00705-708	0x0000 0x0000 0x0f00 0x403a,26.05859	CH2:valid temperature (DOUBLE64I) Word order:0x1122334455667788 -> 0x7788 0x5566 0x3344 0x1122
4x00709-712	0x0000 0x0000 0x3800 0xc08f,-999.00000	CH1:real temperature (DOUBLE64I) Word order:0x1122334455667788 -> 0x7788 0x5566 0x3344 0x1122
4x00713-716	0x0000 0x0000 0x0f00 0x403a,26.05859	CH2:real temperature (DOUBLE64I) Word order:0x1122334455667788 -> 0x7788 0x5566 0x3344 0x1122
4x00717-720	0x0000 0x0000 0x3800 0xc08f,-999.00000	CH1:average temperature (DOUBLE64I) Word order:0x1122334455667788 -> 0x7788 0x5566 0x3344 0x1122
4x00721-724	0x89d9 0xd89d 0x1009 0x403a,26.06265	CH2:average temperature (DOUBLE64I) Word order:0x1122334455667788 -> 0x7788 0x5566 0x3344 0x1122
4x00725-728	0x0000 0x0000 0x6000 0x4069,203.00000	CH1:status as DOUBLE64I Word order:0x1122334455667788 -> 0x7788 0x5566 0x3344 0x1122
4x00729-732	0x0000 0x0000 0x0000 0x3ff0,1.00000	CH2:status as DOUBLE64I Word order:0x1122334455667788 -> 0x7788 0x5566 0x3344 0x1122

8 RESI-1LED-MODBUS, RESI-1LED-ASCII

8.1 Product description

This IO module offers the following features:

- 3 dimmable PWM output channels for LED stripes, 0..48Vdc, max. 5A each channel
- Six selectable modes: OFF, ON, FLASHING, FADING, RANDOM, SEQUENCE
- External power supply for LED stripes, 0..48Vdc, max. 15A
- Galvanic insulated RS232/RS485 interface for communication with a host system
- RESI-1LED-MODBUS: MODBUS/RTU slave protocol
- RESI-1LED-ASCII: MODBUS/RTU slave and text based ASCII protocol
- Power supply 12-48Vdc
- DIP switch for setting the baud rates, the type of interface and the bus number
- LED indicator for the communication
- Mounting onto EN50022 DIN rail

Illustration: Our IO module

8.1.1 The modes of the LED module

The LED module offers six modes. You can switch the mode by setting a special register via MODBUS/RTU or by executing the #SMODE ASCII command. Be aware that the converter does not save a mode in remanent memory. After reset the module starts always in mode ON!

8.1.2 LED mode OFF

In this mode all three outputs are switched to 0. It doesn't matter, what values are in the set point registers LO1 4x00001, LO2 4x00002 or LO3 4x00003. The registers for the actual output values CLO1 4x00008, CLO2 4x00009 and CLO3 4x00010 return always the value 0.

8.1.3 LED mode ON

In this mode all three outputs are switched immediately to the current values in the registers LO1 4x00001, LO2 4x00002 or LO3 4x00003. The registers for the actual output values CLO1 4x00008, CLO2 4x00009 and CLO3 4x00010 delivers always the same value as the registers LO1 4x00001, LO2 4x00002 or LO3 4x00003 to indicate, that the values are really outputted to the three PWM channels.

8.1.4 LED mode FLASH

In this mode all three outputs are switched as a recycler relay between the three current values in the registers LO1 4x00001, LO2 4x00002 or LO3 4x00003 and 0. While ON time, the module outputs the three registers LO1 4x00001, LO2 4x00002 or LO3 4x00003 to the real outputs for a timespan defined in the register MINIMUM TIME 4x00006 in 1/10s. In this time the registers for the actual output values CLO1 4x00008, CLO2 4x00009 and CLO3 4x00010 delivers always the same value as the registers LO1 4x00001, LO2 4x00002 or LO3 4x00003 to indicate, that the values are really outputted to the three PWM channels. Then the converter switches all three channels to 0 for the OFF time span. This time span is defined with the value of the MAXIMUM TIME register 4x00007 in 1/10s. In this time the registers for the actual output values CLO1 4x00008, CLO2 4x00009 and CLO3 4x00010 delivers always the value 0. This ON/OFF cycle is repeated endlessly.

Steps for FLASH:

Step 1: Output of the three set point values LO1, LO2, and LO3 to the real PWM outputs
Step 2: Wait for MINIMUM TIME in 1/10s
Step 3: output of the values 0, 0, 0 to the real PWM outputs
Step 4: Wait for MAXIMUM TIME in 1/10s
Step 5: continue with step 1

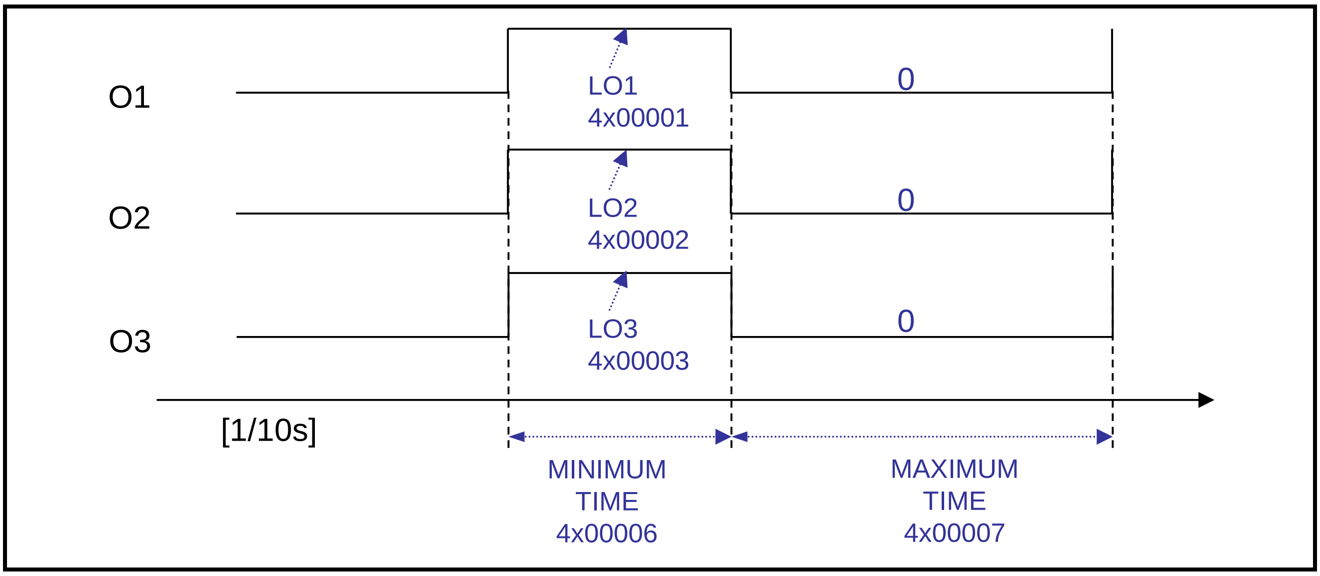


Illustration: timing of mode FLASH

8.1.5 LED mode FADE

In this mode the converter doesn't change the output values immediately. No, it uses a ramp to change slowly from the current value to the new value. This ramp is defined in the register FADE SPEED 4x00005. The setup is done in steps per 1/100s and is valid for all three channels. To set a new value write into the three registers LO1 4x00001, LO2 4x00002 or LO3 4x00003. The system fades from the current value to the new values. If you read the registers CLO1 4x00008, CLO2 4x00009 and CLO3 4x00010 while fading, you will get every value change from the old value to the new value. Also the register IS FADE ACTIVE 4x00014 will return a 1 while fading is running. When the module reaches the new values, reading of the registers CLO1 4x00008, CLO2 4x00009 and CLO3 4x00010 will return the same values as the registers LO1 4x00001, LO2 4x00002 and LO3 4x00003. Also the register value of IS FADE ACTIVE 4x00014 will be 0.

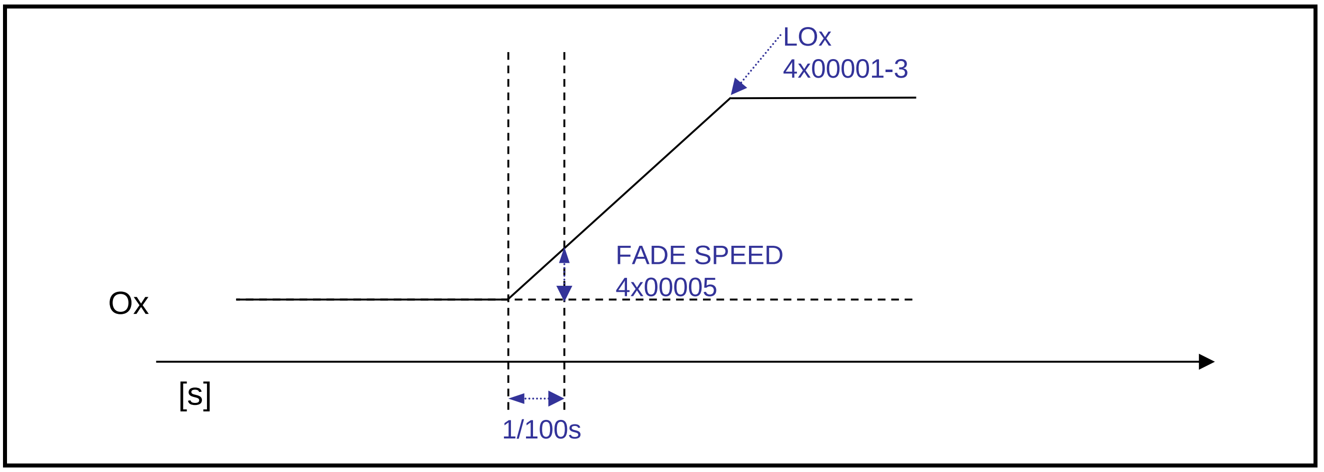


Illustration: timing of mode FADE

8.1.6 LED mode RANDOM

In this mode the converter generates random values for each output. For this you can setup a time interval. If this time interval expires the system dices new random values for the three outputs. The time interval is defined by the register MINIMUM TIME 4x00006 and the register MAXIMUM TIME 4x00007 in seconds. The system generates a random time interval between those two parameters. If the time expires, the system dices new random values for the three registers RLO1 4x00011, RLO2 4x00012 and RLO3 4x00013. Then the system fades the current values in the registers CLO1 4x00008, CLO2 4x00009 and CLO3 4x00010 to the new random values. This fade ramp is defined in the register FADE SPEED 4x00005. The setup is done in steps per 1/100s. If you read the registers CLO1 4x00008, CLO2 4x00009 and CLO3 4x00010 while fading, you will get every value change from the old value to the new value. Also the register IS FADE ACTIVE 4x00014 will return a 1 while fading is running. When the module reaches the new values, reading of the registers CLO1 4x00008, CLO2 4x00009 and CLO3 4x00010 will return the same values as the registers RLO1 4x00001, RLO2 4x00002 and RLO3 4x00003. Also the register value of IS FADE ACTIVE 4x00014 will be 0. The diced values in the registers RLO1 4x00011, RLO2 4x00012 and RLO3 4x00013 will be in the range of 0 to LO1 4x00001, LO2 4x00002 and LO3 4x00003.

Steps for RANDOM:

Step 1: Dice three random numbers in the range of 0..LOx and store the values in RLOx
Step 2: Dice a random wait period between MINIMUM TIME and MAXIMUM TIME in seconds
Step 3: Fade up or down from the actual output values CLOx to the new end values RLOx
Step 4: If the random wait period is over, continue with step 1

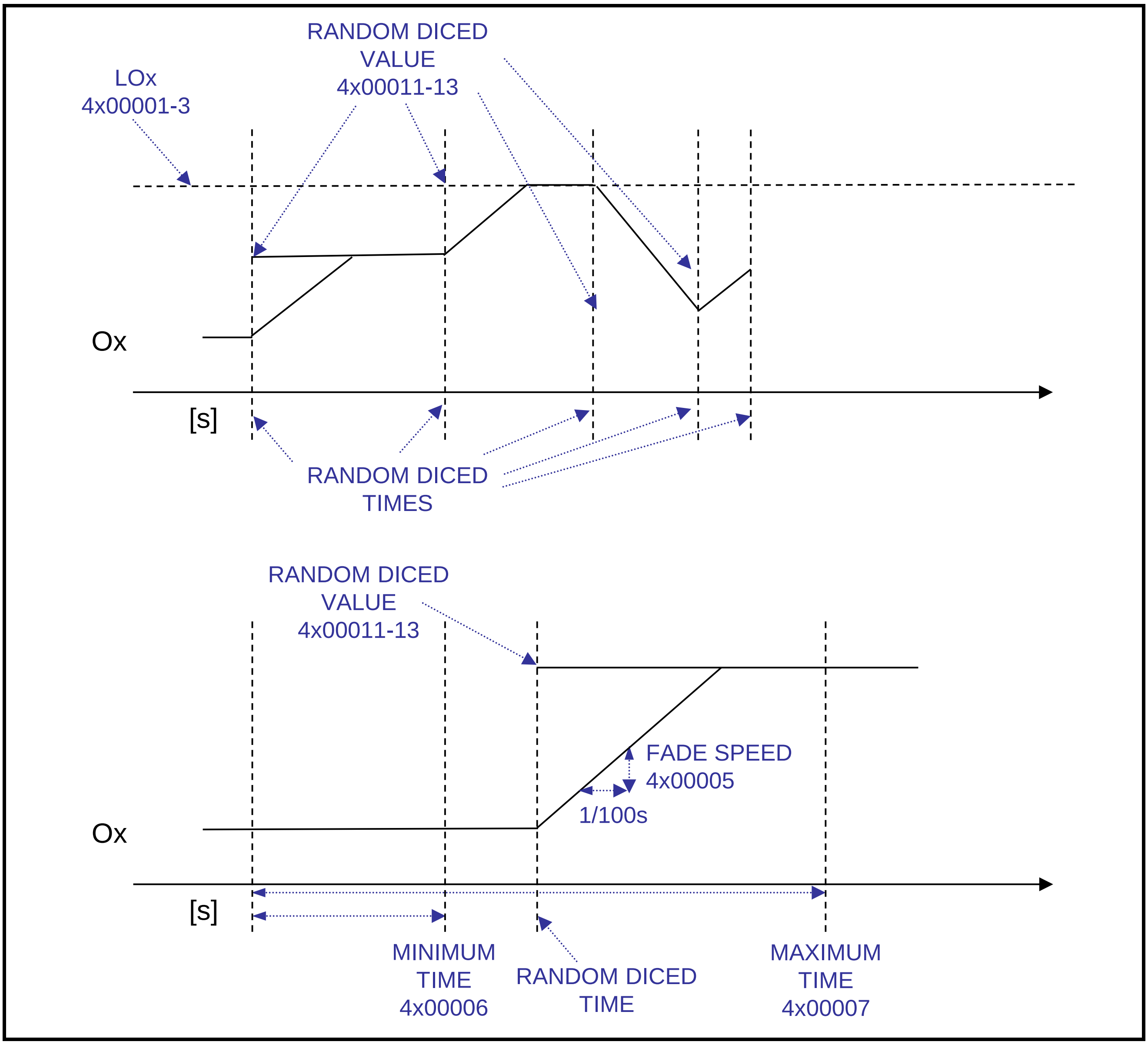


Illustration: timing of mode RANDOM

8.1.7 LED mode SEQUENCE

In this mode, the module creates a sequential flash light with the three PWM outputs. The outputs flashes between the three set points LO1, LO2 and LO3 and 0 in sequence. In the first ON phase the module sets the real output CLO1 to the set point LO1, the other two outputs are set to 0. This phase lasts for MINIMUM TIME in 1/10s. While this period of time, the current value register CLO1 delivers the same value as in LO1, and the other two current value registers CLO2 and CLO3 deliver the value 0. Then the module switches all three outputs to 0 for a time period defined with the register MAXIMUM TIME in 1/10s (OFF time period). While this period of time, all three output registers CLOx deliver the value 0. Now the system repeats the ON phase with the next set point register LO2. The two registers CLO1 and CLO3 are 0 in this phase. Next the OFF time period is executed. The last phase is the ON phase with the register LO3. The two registers CLO1 and CLO2 are 0 in this phase. The last OFF time period is executed. This three times ON/OFF cycle is repeated endlessly.

Steps for SEQUENCE:

Step 1: Output the three set points LO1, 0, 0 to the three PWM outputs
Step 2: wait for MINIMUM TIME in 1/10s
Step 3: Output the values 0, 0, 0 to the three PWM outputs
Step 4: wait for MAXIMUM TIME in 1/10s
Step 5: Output the three set points 0, LO2, 0 to the three PWM outputs
Step 6: wait for MINIMUM TIME in 1/10s
Step 7: Output the values 0, 0, 0 to the three PWM outputs
Step 8: wait for MAXIMUM TIME in 1/10s
Step 9: Output the three set points 0, 0, LO3 to the three PWM outputs
Step 10: wait for MINIMUM TIME in 1/10s
Step 11: Output the values 0, 0, 0 to the three PWM outputs
Step 12: wait for MAXIMUM TIME in 1/10s
Step 13: continue with step 1

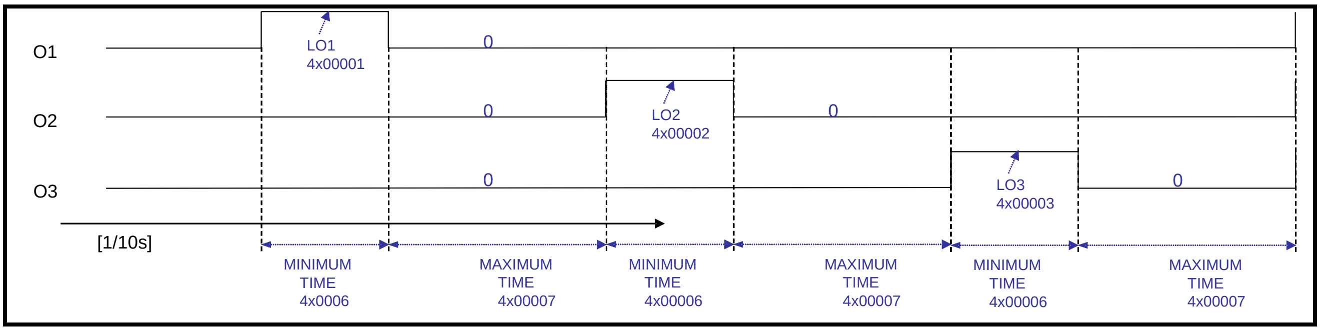


Illustration: timing diagram for mode SEQUENCE

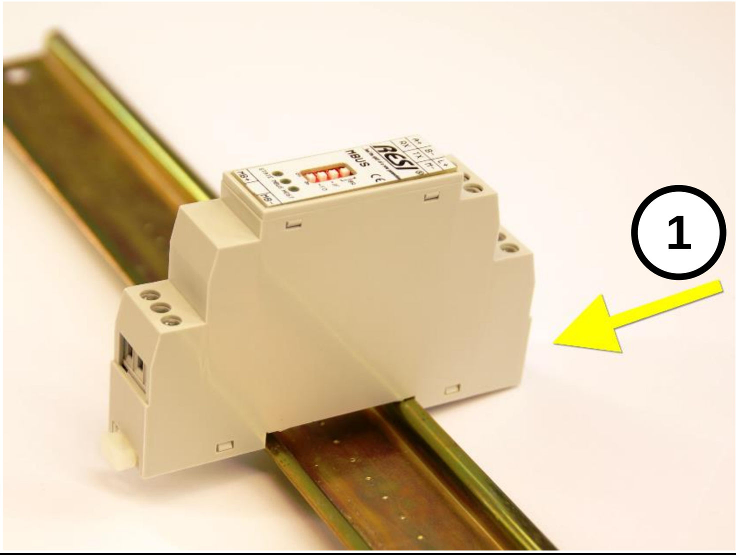
8.2 Technical data

Technical Data			
Power supply			
Supply voltage	12-48V= +/-10%	Storage temperature	-20...85 °C
Power LED	Yes	Operating Temperature	0...60°C
Power consumption	<0.6W	Humidity	25...90 % rH non-condensing
		Protection Class	IP20 (EN 60529)
		Dimensions LxWxH	17,5mm x 90mm x 58mm
		Weight	55g
		Mounting	on DIN EN50022 rail
ASCII/Modbus Interface			
Protocol	ASCII or Modbus/RTU		
Type	RS232 or RS485		
Baud rates	9600 to 57600/8/N or E/1		
Cable Connection	Via clamps		
LED indicator	Yes		
Galvanic insulation	Yes		
LED stripe output			
Number of outputs	3 dimmable outputs		
Signal	PWM with 400Hz		
LED stripes	RGB Dual white Mono color		
LED connection	Via common anode		
Output voltage	0..48Vdc		
Output current	Max. 5A per channel		
LED Power supply	0..48Vdc,max 15A 180W@12Vdc 360W@24vdc 720W@48Vdc		
Cable connection	Via clamps		
Galvanic insulation to serial interface	Yes		
LED indicator	Yes		
Clamps			
Clamp wire cross section	Max. 1,5 mm²		
Tightening torque	Max. 0.5Nm	**CE conformity**	Yes

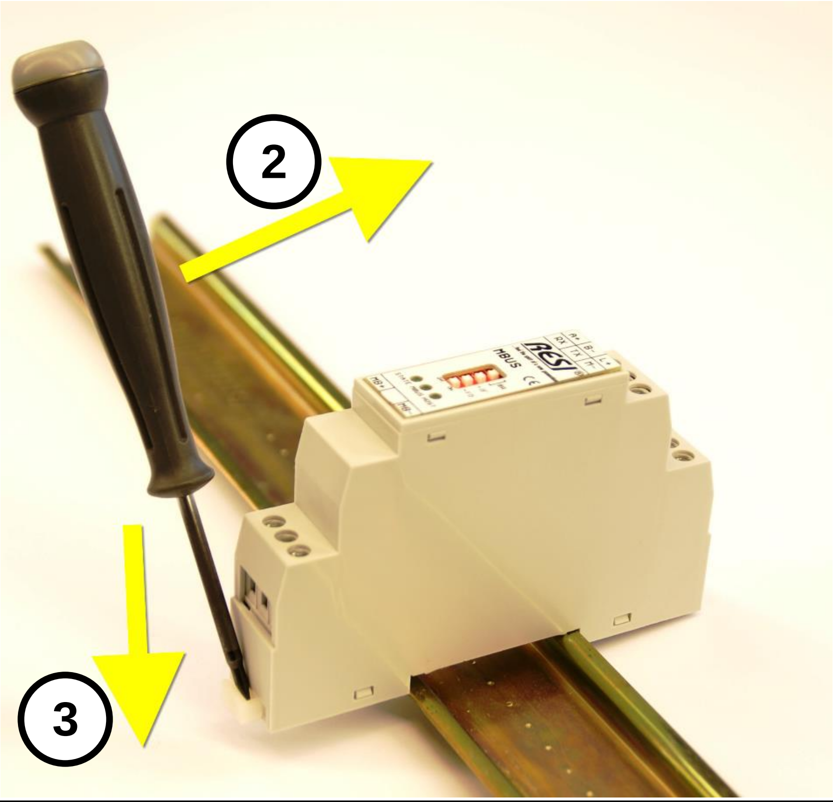
8.3 Assembling

Our IO modules are designed for mounting on a 35mm DIN-EN50022 rail.

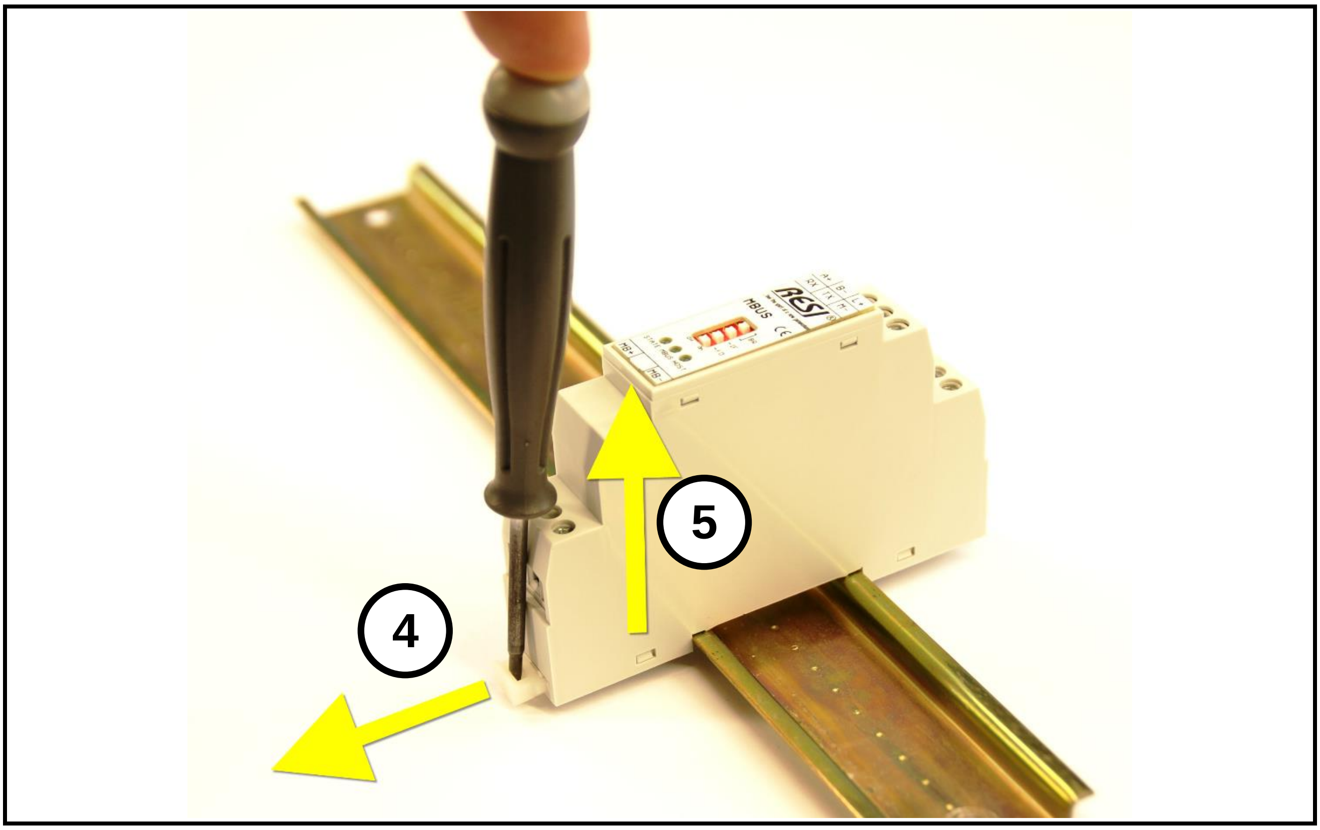
At first, put the modules with the top side on the DIN rail (1).



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.



8.4 Connection diagram

In the below drawing you will find the correct cabling of the IO module.

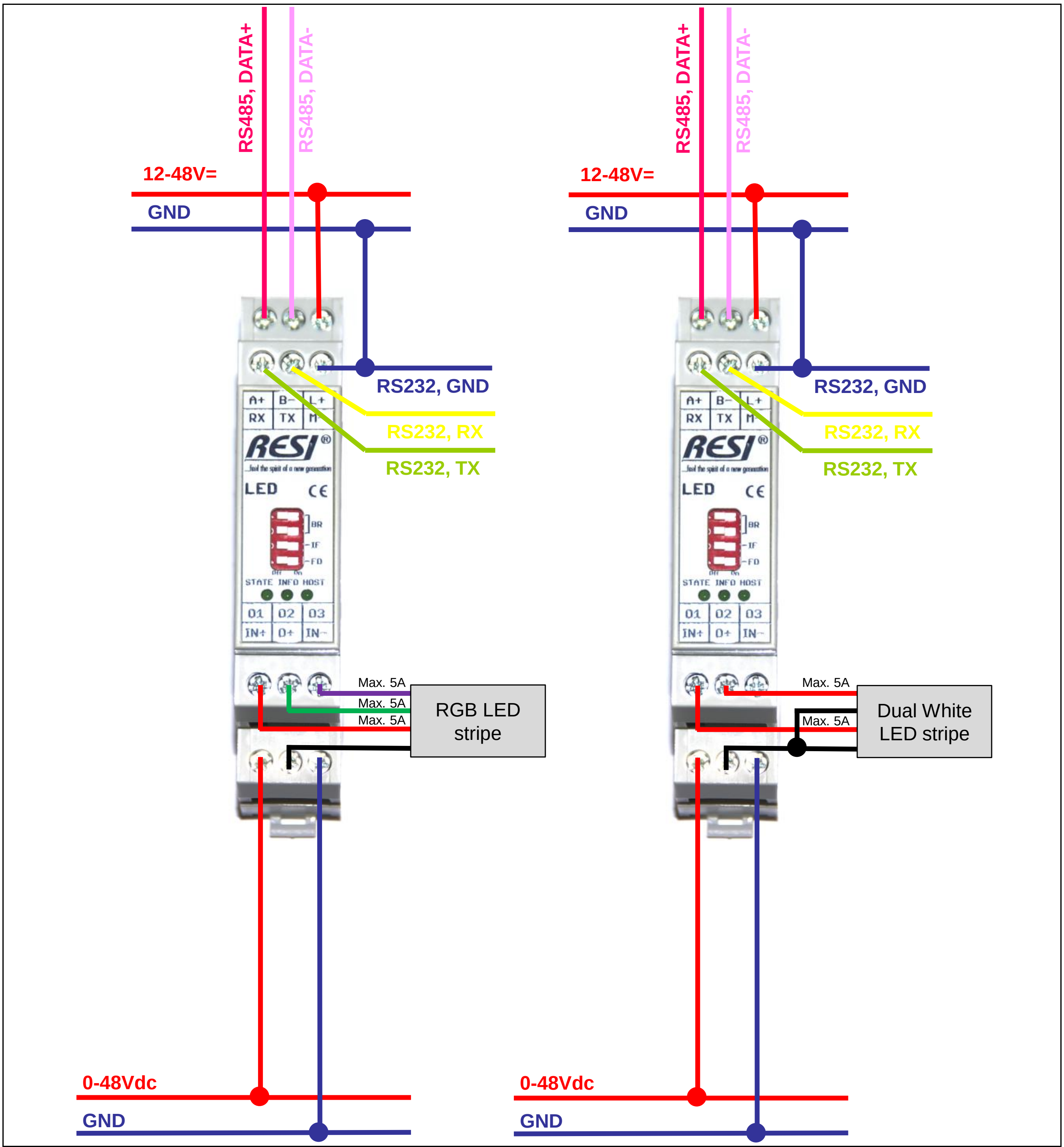


Illustration: Cabling of the IO module

8.5 Clamps

The IO module offers the following clamps:

CLAMP	DESCRIPTION
L+ M-	Power supply: L+: 12-48 V= M-: Ground
RS485 A+ B- M-	RS485 ASCII or MODBUS/RTU interface A+: RS485 DATA+ signal B-: RS485 DATA- signal M-: RS485 ground signal
RS232 TX+ RX- M-	RS232 ASCII or MODBUS/RTU interface TX+: RS232 Transmit signal RX-: RS232 Receive signal M-: RS232 Ground signal
LED stripes O1, O2, O3 IN+, O+, IN-	LED stripe connection: IN+, IN-: Power supply 0..48Vdc O1, O2, O3: Dimmable PWM output max. 5A O+: Common anode

Table: Description of the clamps on the IO module

8.6 DIP switch setting and LED indicators

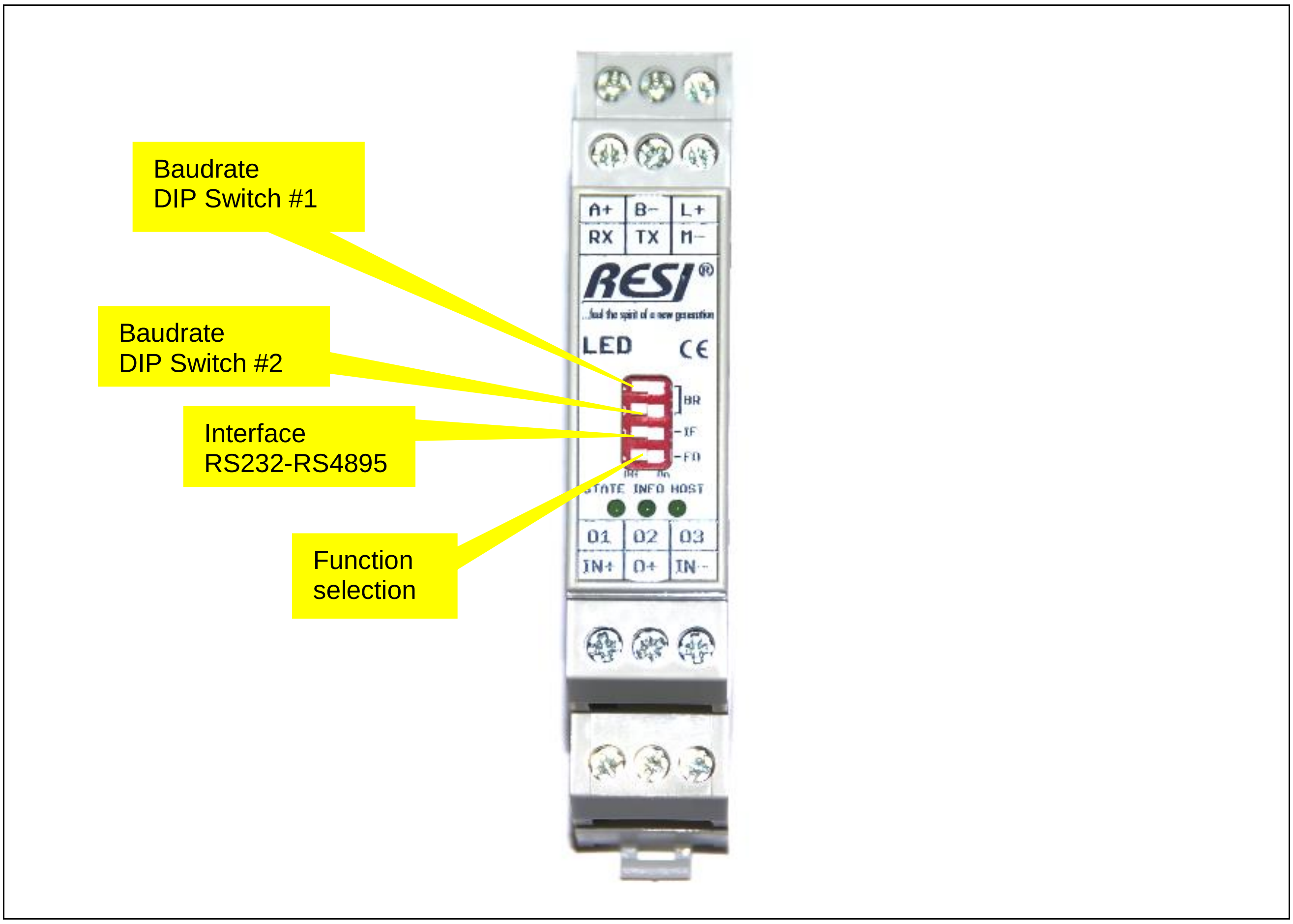


Illustration: Description of the DIP switch settings and LED indicators

DIP Switch	Description
Baudrate BR	Use DIP switches 1+2 to select the baud rate: OFF OFF: 9600Bd ON OFF: 19200Bd OFF ON: 38400Bd ON ON: 57600Bd HINT: The correct parity (NONE, EVEN, ODD) is defined by the PC software, not with the DIP switches.
Interface IF	Selects the physical type of the serial interface for the ASCII or MODBUS/RTU protocol: OFF=RS232 ON=RS485
Function definition FD	Select s special function in the module: OFF=The module uses the configured unit ID from the FLASH memory ON=The module uses always the unit ID 255
HINT	After a change of the DIP switches, the module reboots. No power off / power on cycle is necessary. After the reset all three LEDs are shortly on to represent the RESTART sequence.

Table: Description of the DIP switches of the IO module

LED	Description
STATE	Status LED: If the module is ok, this LED flashes slowly. If there is an error detected by the module, this LED flashes fast.
INFO	This LED shows the status of the three outputs. In mode OFF this LED is off. In mode ON the LED is on. The LED flashes if fading is active
HOST	HOST-LED, Flashes, if the host is communicating with the module.

Table: Description of the LED indicator on the IO module

8.7 Dimensions of the module

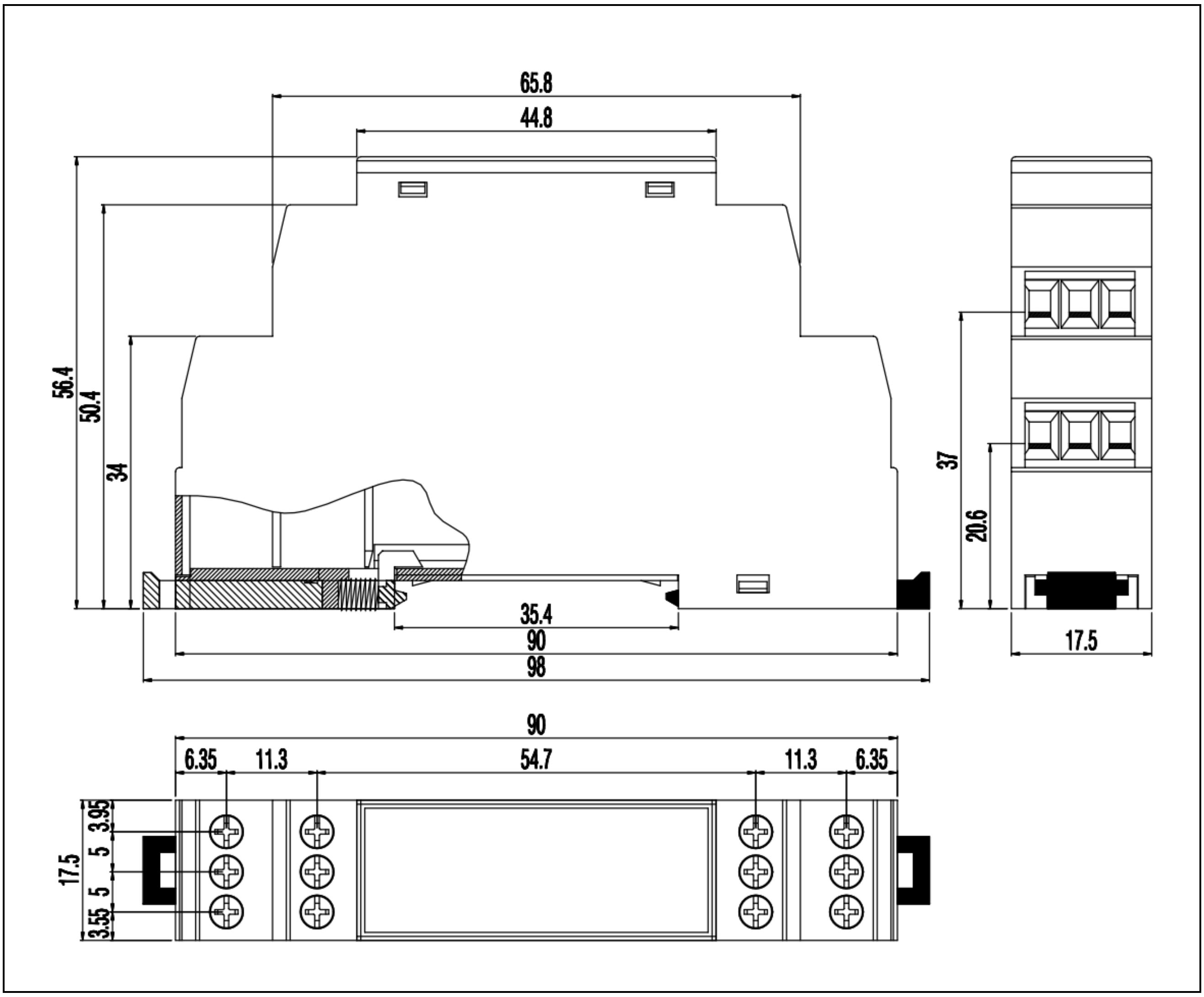


Illustration: dimension illustration in mm

Dimensions	
Enclosure dimensions L x W x H (mm)	17,5 x 90 x 58
Weight	60 g
Color	Grey RAL7035
Material	PA - UL 94 V0
Protection class	IP20 based on DIN 40050/EN 60529

Table: Data of enclosure

8.8 3D Drawing

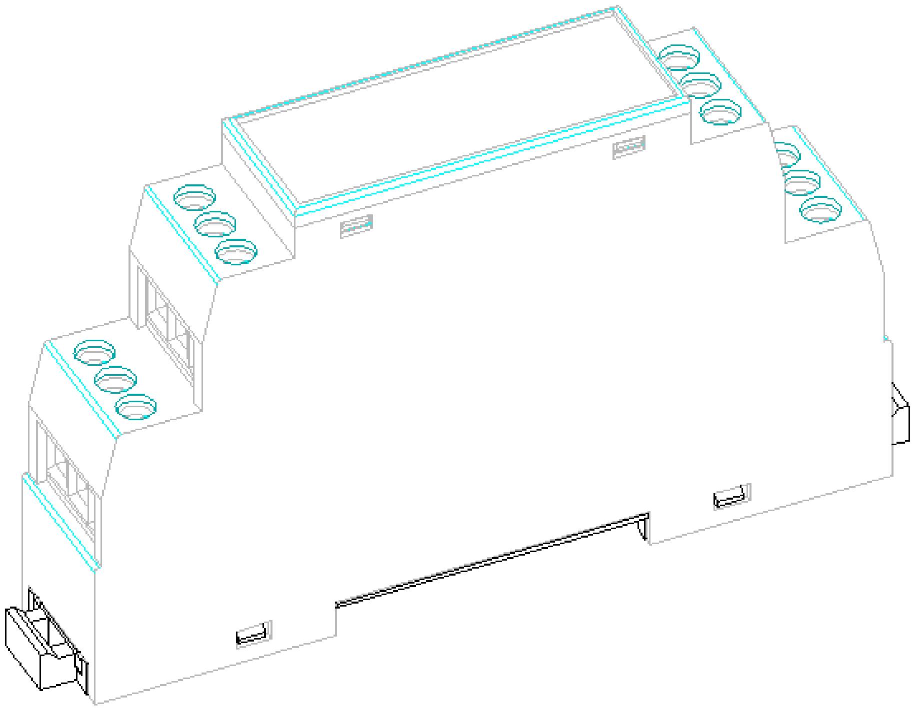


Illustration: 3D drawing of the enclosure

8.9 Power supply cabling of the module

In the image below you will see the correct cabling of the power supply of the module.

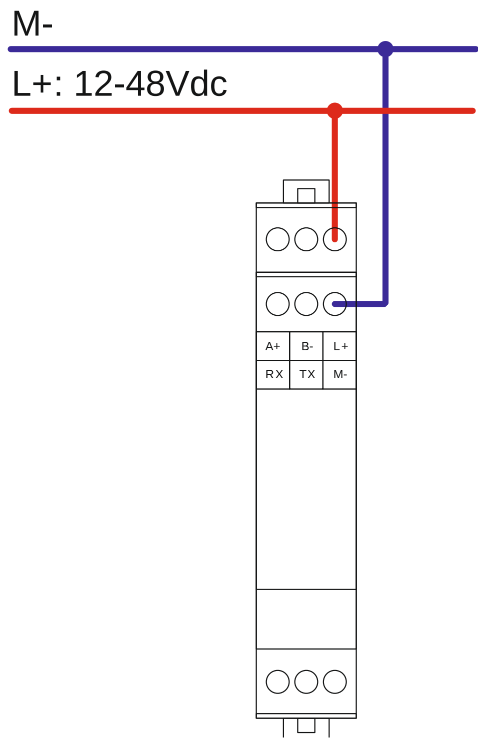


Illustration: Cabling of the power supply of the IO module

8.10 RS485 cabling of the IO module

In the image below you see the correct cabling of the RS485 interface of the IO module.

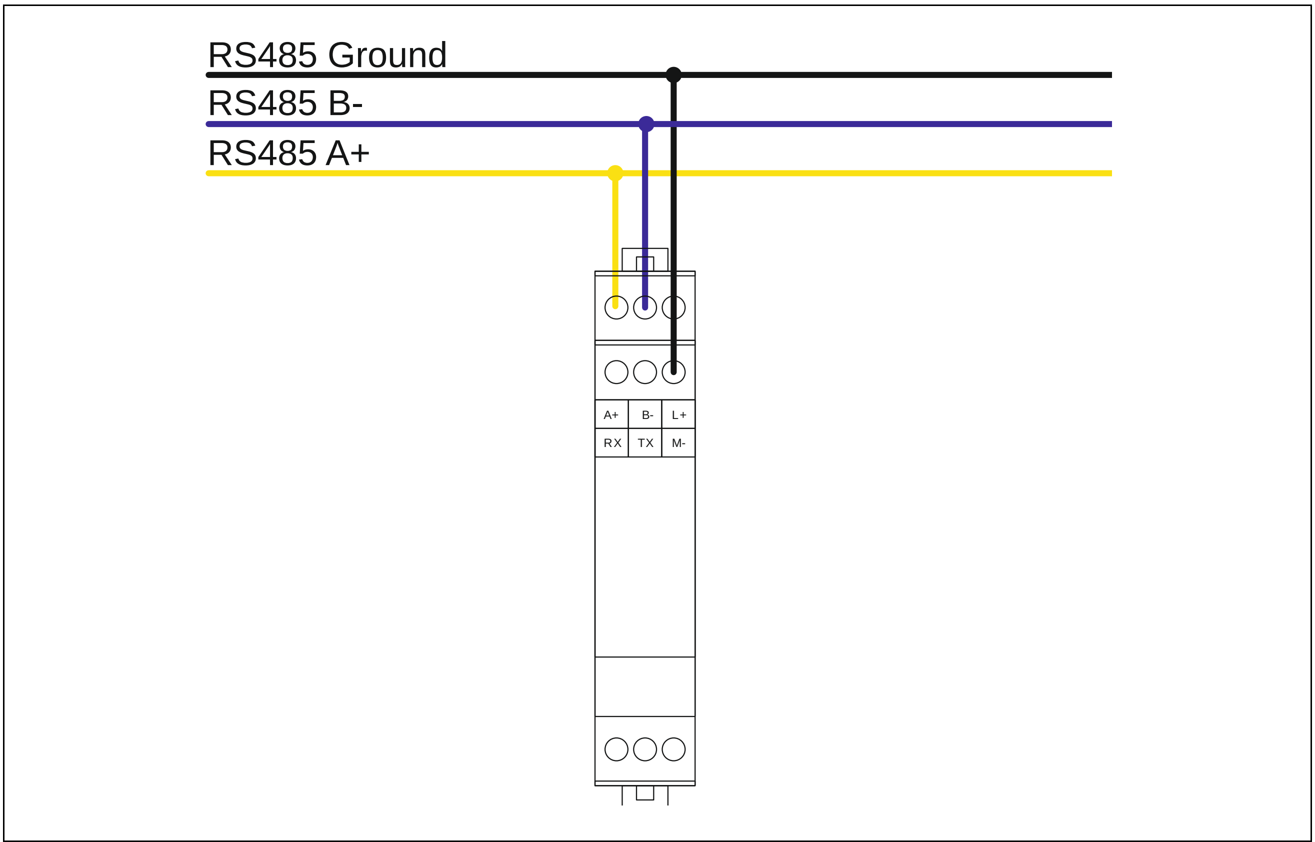


Illustration: RS485 bus cabling of the IO module

8.11 RS232 cabling of the IO module

In the image below you see the correct cabling of the RS232 interface of the IO module.

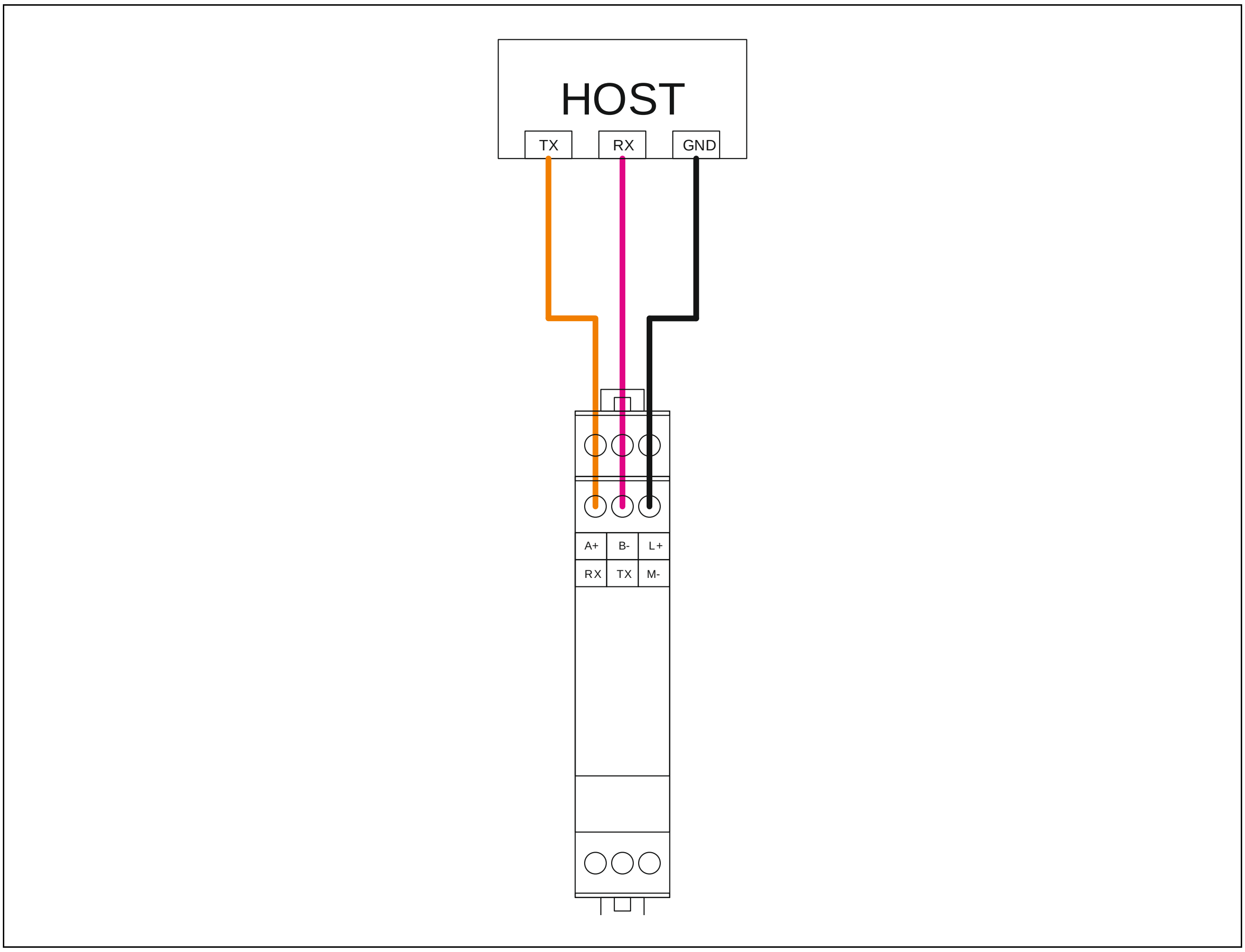


Illustration: RS232 bus cabling of the IO module

8.12 Cabling of the LED stripes on the module

In the below drawings you will find a detailed description, how to cable the different types of LED stripes to the module. We now inspect the different types of LED stripes in detail.

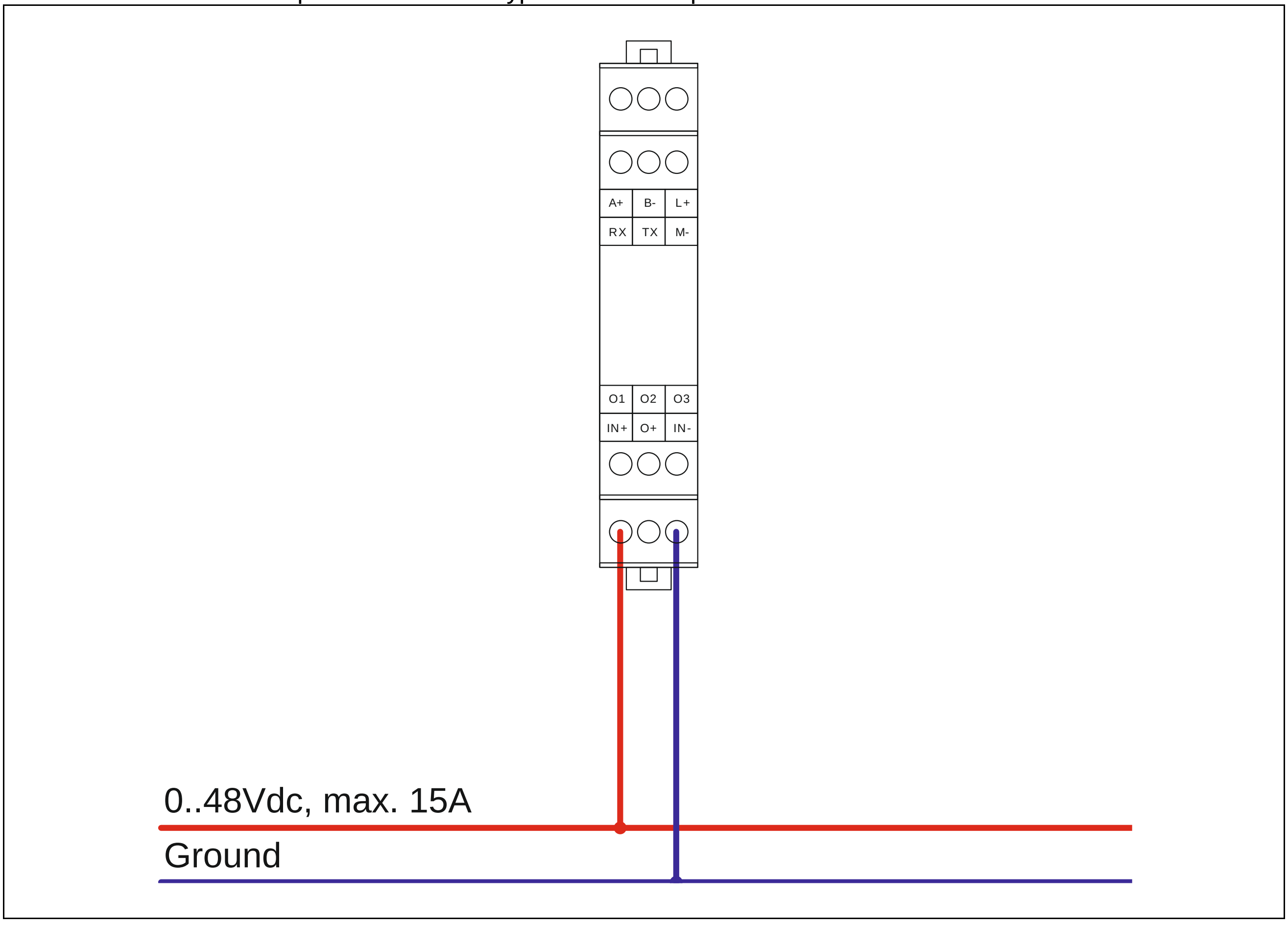


Illustration: Power supply for LED stripes

The power supply for the LED stripes must be cabled externally. The module offers the two clamps IN+ and IN- to connect the power supply. Depending on the type of LED stripe you want to use, you can connect various types of power supplies. It is very important, that the used power supply does not exceed the maximum continuous current rating of 15A! The result is the following mandatory limits for supplying LED stripes with different voltage levels:

- LED stripes with 12Vdc power supply: 12Vdc*15A -> max. 180W mains adapter
- LED stripes with 24Vdc power supply: 24Vdc*15A -> max. 360W mains adapter
- LED stripes with 48Vdc power supply: 48Vdc*15A -> max. 720W mains adapter

But be aware, that every output can only drive a maximum current of 5A!

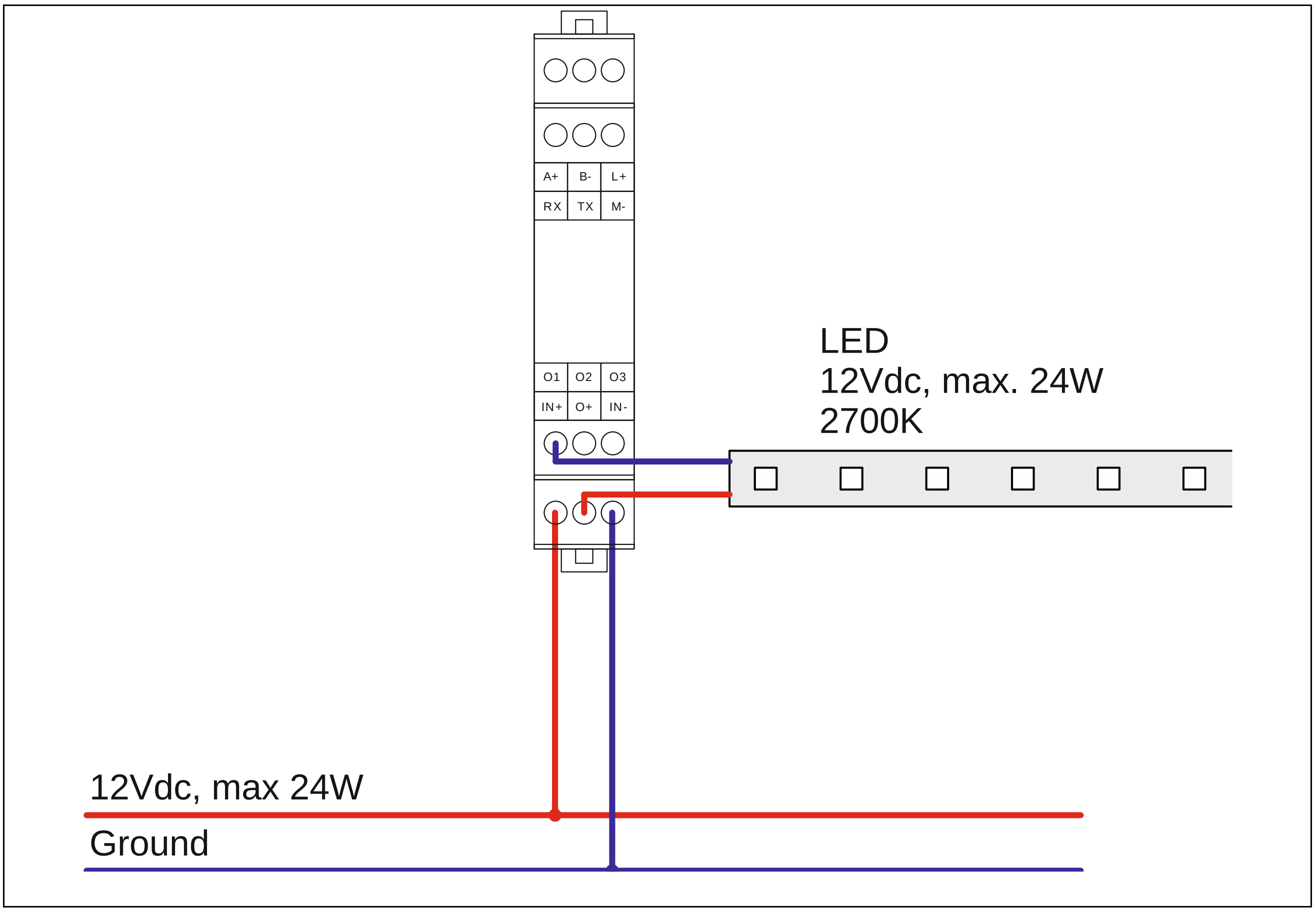


Illustration: Cabling of a 12Vdc LED stripe with 24W power consumption, luminous color 2700K. Due to the reason, that the LED strip consumes only 24W, we use also a 24W mains adapter. So there flows an input current of 2A. Via the output O1 flows also an output current of 2A. (<5A, so this is ok).

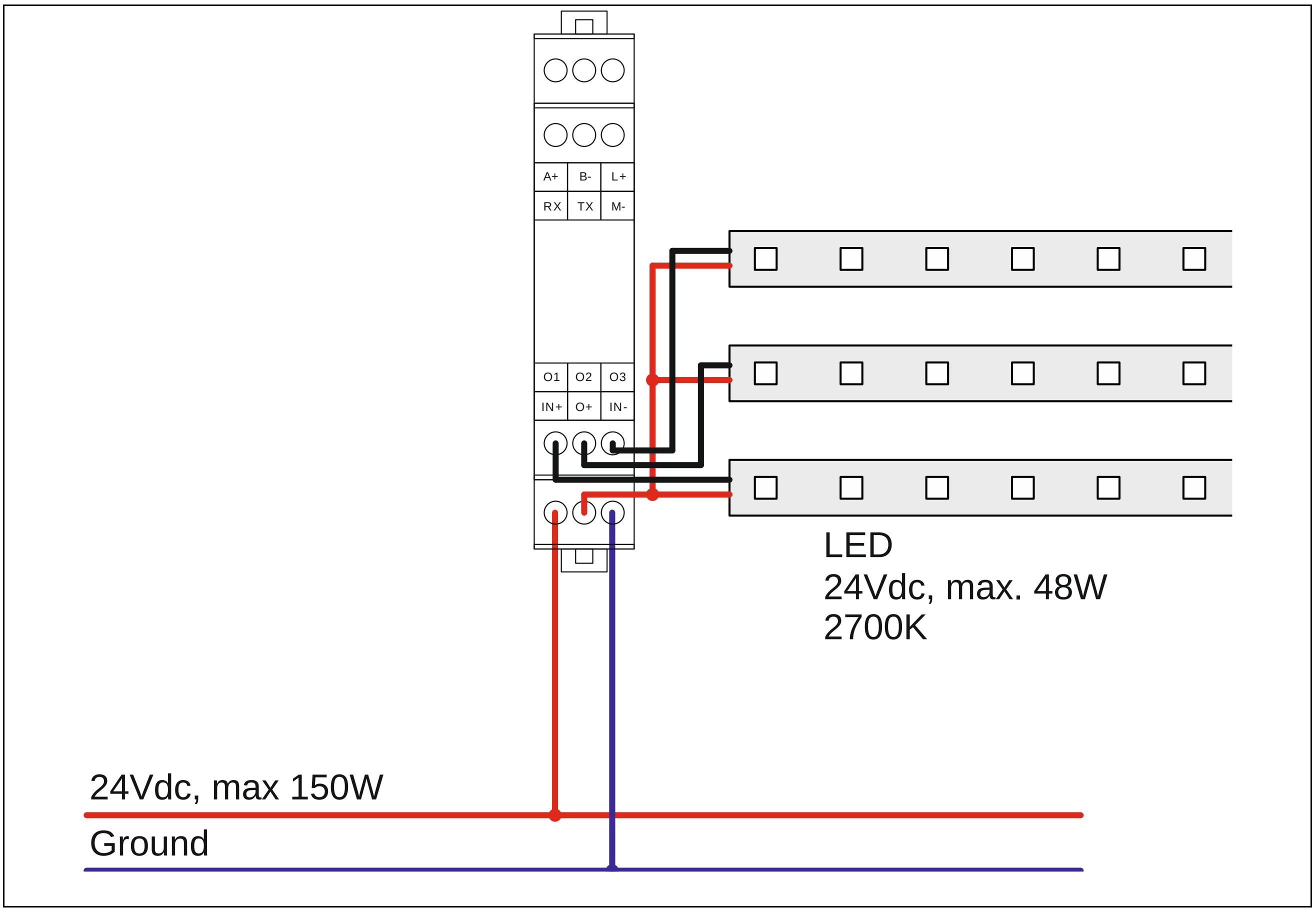


Illustration: Cabling of three 24Vdc LED stripes with 48W each stripe. Each of the three LED stripes can be dimmed individually. The sample uses all three LED outputs to create three individual dimmable groups of LED stripes. Each LED stripe consumes 48W power. So we use a power supply with 3x48W -> 150W. The input current on the clamps IN+, IN- is max. 6.25A. This is less than the allowed 15A and ok. While we use on each output a LED stripe with 48W power, the output current per channel is max. 2A. This is lower than the maximum rating of 5A per output und therefore ok too.

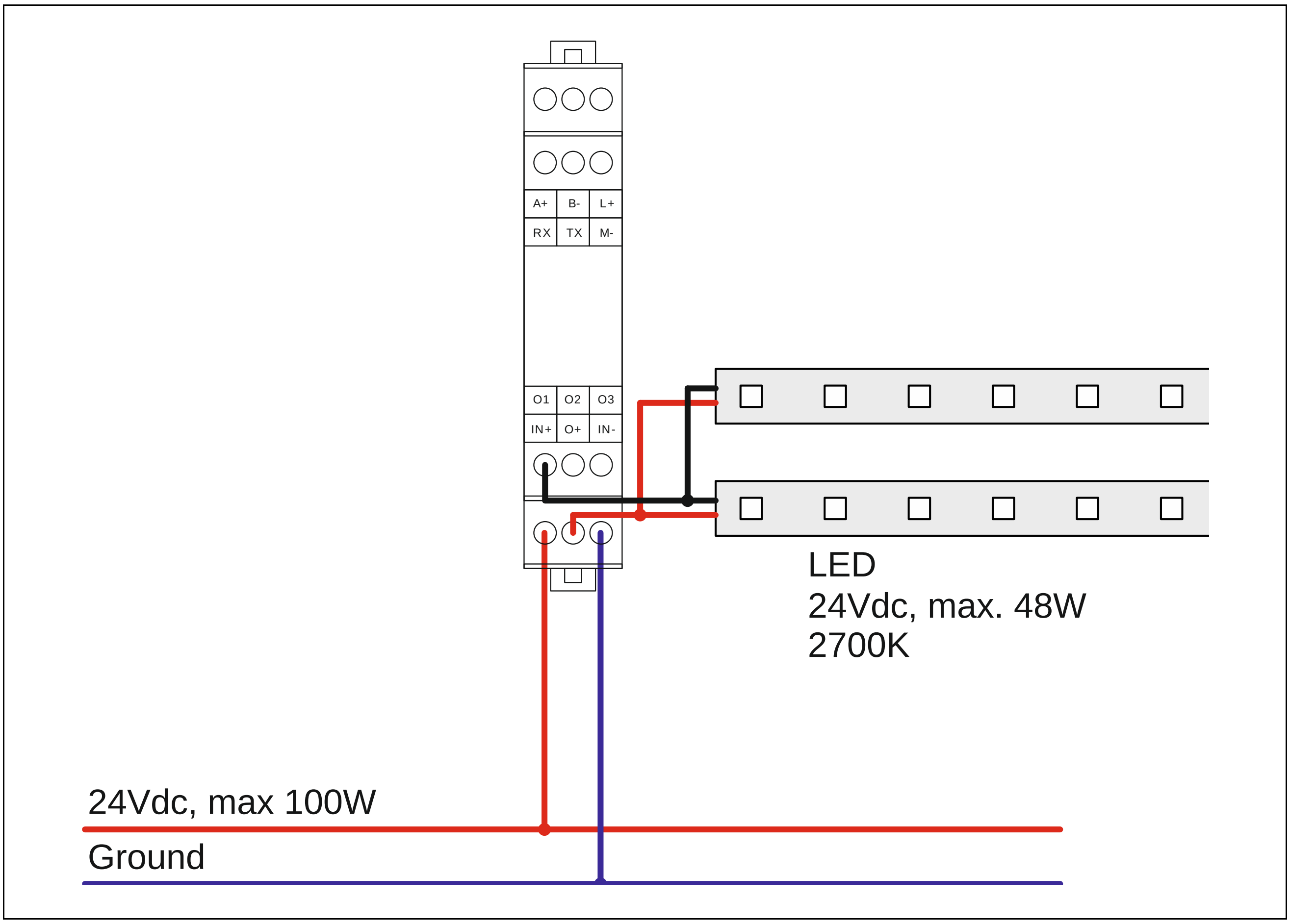


Illustration: Cabling of two 24Vdc LED stripes with 48W power consumption each. Both LED stripes are only together dimmable. Only output O1 is used for both LED stripes. We use a 100W power supply. The primary input current is 4.17A. This is smaller than the allowed 15A and therefore ok. We operate with two LED stripes on one output. This output must drive 96W power. We choose a 24Vdc LED stripe. So the output current is 4A. This is again smaller than the allowed 5A and ok.

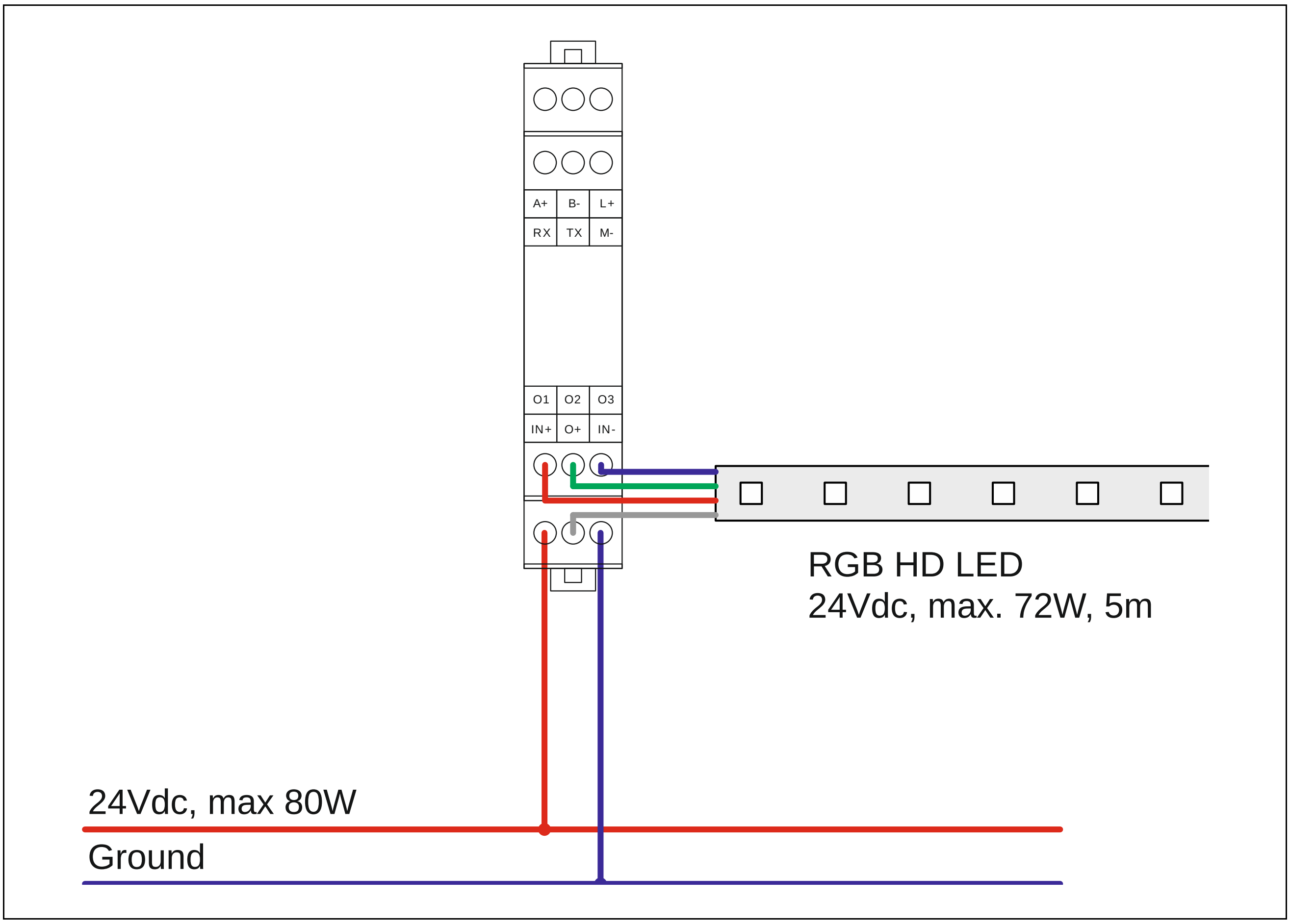


Illustration: In this sample we use RGB LED HD stripe. This stripe offers three dimmable channels for the three primary colors red, green and blue. The common anode is again connected to O+. The 80W power supply delivers a maximum current of 3,34A. So this current is far beyond the allowed 15A for the power input. The LED stripe consumes only 1/3rd of the total power of 72W on each channel. This is 24W, the current is 1A. Again the output current on all three outputs is much lower than the allowed 5A.

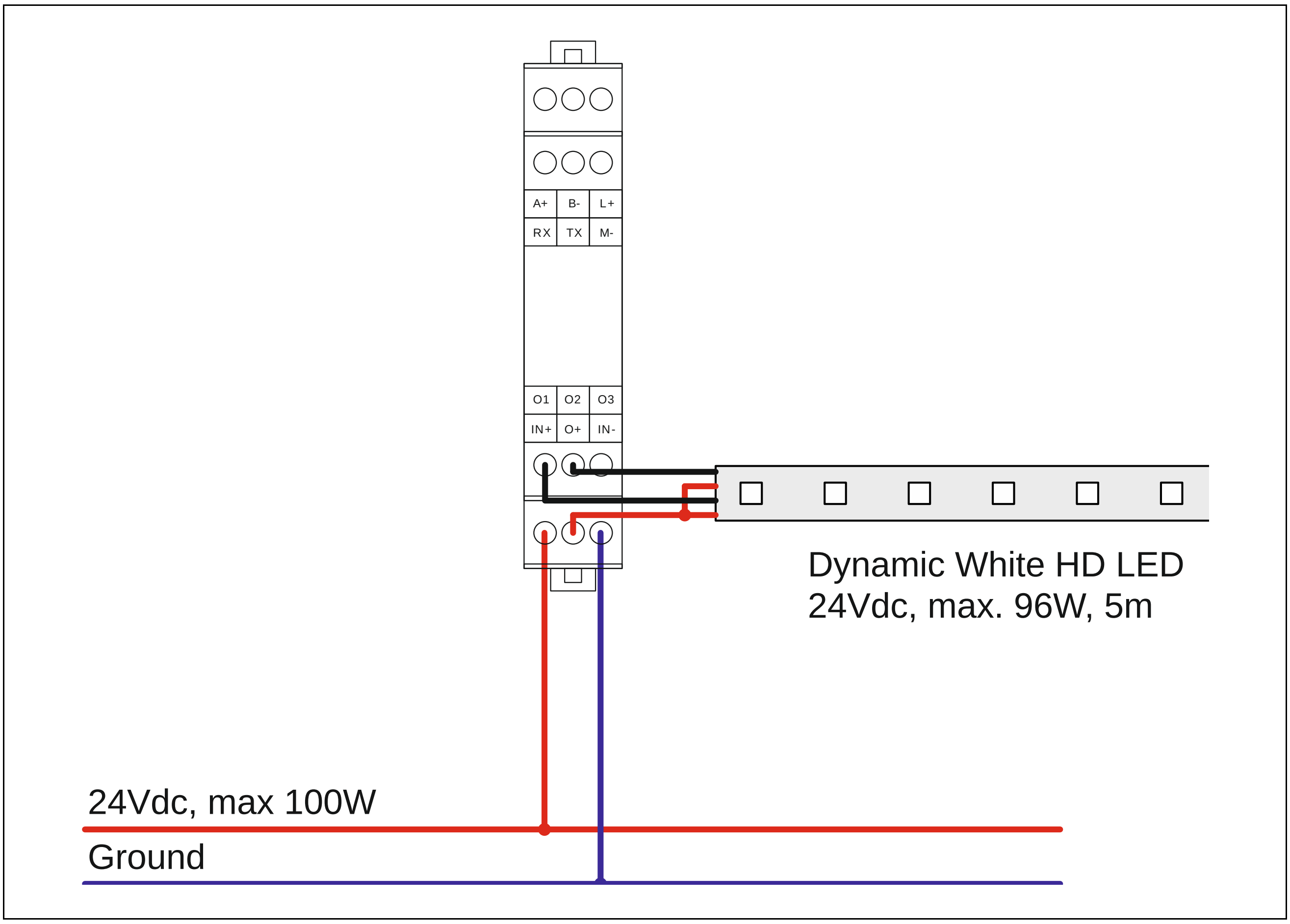


Illustration: Cabling of a dynamic white LED stripe. This type of LED stripe combines two LED types with different luminous colors in one LED stripe. This LED stripe can mix a spectrum of white colors, mostly from warm white to cold white. We have to wire the four cables of the LED stripe to our module as shown in the above drawing. Each output must drive 48W. Again we have a maximum output current of 2A per channel. This is far beyond the allowed 5A and ok. The 100W power supply delivers an input current of max. 4.16A. Also this current is far beyond the allowed 15A and ok.

8.13 Functional description

This IO module communicates with a host system with the MODBUS/RTU slave protocol. All versions of the Io module with RESI-xxx-ASCII offer an additional protocol: An ASCII slave protocol with simple text string. The communication runs over a RS485 interface (half duplex) or over a RS232 interface (full duplex).

For the communication with ASCII texts, the host sends ASCII messages with a special start character # (0x23, 35dec) and a special end character (0x0d,13dec or CARRIAGE RETURN) to the module. The module uses also this special start and end characters to answer to the host request. Consult the below noted detailed command descriptions. In the ASCII protocol mode, the host can send messages with or without a bus number.

For communication with the MODBUS/RTU slave, the module offers the following MODBUS functions:

- 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)

HINT:
The functions READ HOLDING REGISTER and PRESET MULTIPLE REGISTERS are restricted to max. 125 register per request! The functions READ INPUT STATUS, READ COIL STATUS and FORCE MULTIPLE COILS are restricted to 2000 coils or inputs (bits) per request.

8.14 ASCII protocol description

8.14.1 Overview

The IO module communicates with simple ASCII commands. The following special characters are used in this documentation:

stand for the **Hashtag** ASCII character 35dec or 0x23
: stand for the **colon** ASCII character 58dec or 0x3A
= stand for the **equal sign** with the ASCII code 61ec or 0x3D
- stand for the **minus sign** with the ASCII code 45dec or 0x2D
, stands for the **comma** with the ASCII code 44dec or 0x2C
<CR> stand for **CARRIAGE RETURN** ASCII character 13dec or 0x0D. In the following text we use the representation CR.
<SP> stands for the **SPACE**. This is the space character with the ASCII code 32dec or 0x20. In the following text we will use the representation☐.
<ADR> will be used as the current **bus address** of the module. The bus number can be transmitted decimal or hexadecimal and is separated with a comma (ASCII code 44dec or 0x2C) from the rest of the command. Hexadecimal numbers always start with 0x. Its only allowed to use the ASCII characters ‚0‘-‚9‘ 48dec to 57dec, 0x30-0x39 and ‚A‘ to ‚F‘, 65dec to 70dec, 0x41-0x46. All modules react to the broadcast bus address 0 and to its own bus number. With a DIP switch, the user can easily change between the internal stored bus number in the FLASH and the fix bus number 255. Consult the DIP switch description for more details.

8.14.2 Communication sequence

In general the IO module sends no ASCII characters without a request from a host. So the host is the master of the communication and the IO module is always answering to host requests as a slave. If only one IO module is connected to a host (e.g. Via RS232), you can dump the bus number in the protocol. If you use a RS485 interface, more than one module can be connected to the host. Therefore a bus number in the request frame of the host is always necessary.

The structure of the command look like this:

The host sends a command or a command with parameters without a bus address to the module:
#<Command><CR> or
#<Command>:<Parameter><CR>

The module answers, if it feels addressed, with the following answer telegram:
#<Answer><CR>

If the bus number is used, the module answers with:
#<ADR>,<Command><CR> or
#<ADR>,<Command>:<Parameter><CR>

The module answers with:
#<ADR>,<Answer><CR>

The bus address lies in the range of 1dec to 255dec or 0x00 to 0xFF hexadecimal. The setup is done with our free configuration software MODBUSConfigurator. Each module reacts also to the broadcast bus address 0.

For each command, we define two different writings. A long version and a short version, to avoid unnecessary traffic on the bus. For example to request the software version of the module you can use the command VERSION or the short command VER.

8.14.3 Request VERSION

This command returns the current software version of the module.

Host long version:
#VERSION<CR> or
#<ADR>,VERSION<CR>

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

Answer:
#VERSION:<HIGH>.<MED>.<LOW><CR> or
#<ADR>,VERSION:<HIGH>,<MED>,<LOW><CR>

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

Samples:
→ **#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$_{CR}$**

8.14.4 Request module TYPE

This command returns the current type of the module.

Host long version:
#TYPE<CR> or
#<ADR>,TYPE<CR>

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

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

<TYP> defines the current type of the module. Currently RESI-1LED-ASCII

Samples:
→ **#TYPE$_{CR}$**
← **#TYPE:RESI-1LED-ASCII$_{CR}$**

→ **#255,TYP$_{CR}$**
← **#255,TYPE:RESI-1LED-ASCII$_{CR}$**

8.14.5 Table of all ASCII commands

Here you will find a possible ASCII commands of the module. We use here only the version with bus number. That you can avoid the bus number, we have discussed earlier in this document. If an argument as the extension Dec, it will be returned as a decimal number, If an argument has the extension Hex, then this argument is returned as a hexadecimal number. Many command returns the argument in decimal and hexadecimal representation. So the host can select, what kind of number conversion, it will handle in its software.

Direction	**ASCII command**
Host	**#<BusAdr>,VER**$_{CR}$ **#<BusAdr>,VERSION**$_{CR}$
Answer	**#<BusAdr>,VERSION:<VersionHi>.< VersionMed>.<VersionLo>**$_{CR}$
	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)
Host	**#<BusAdr>,TYP**$_{CR}$ **#<BusAdr>,TYPE**$_{CR}$
Answer	**#<BusAdr>,TYPE:RESI-1LED-ASCII**$_{CR}$
	Returns the current type of the module
Host	**#<BusAdr>,OWN**$_{CR}$ **#<BusAdr>,OWNER**$_{CR}$
Answer	**#<BusAdr>,OWNER:RESI**$_{CR}$
	Returns the owner of the module
Host	**#<BusAdr>,CRE**$_{CR}$ **#<BusAdr>,CREATOR**$_{CR}$
Answer	**#<BusAdr>,CREATOR:DI HC SIGL,MSC**$_{CR}$
	Returns the creator of the module

Direction	ASCII command
Host	**#<BusAdr>,COPY$_{CR}$** **#<BusAdr>,COPYRIGHT$_{CR}$**
Answer	**#<BusAdr>,COPYRIGHT:2015-16 BY RESI AND DI HC SIGL,MSC WWW.RESI.CC$_{CR}$**
	Returns a copyright note about the module
Host	**#<BusAdr>,GDIP$_{CR}$** **#<BusAdr>,GET□DIP$_{CR}$**
Answer	**#<BusAdr>,GDIP:<DIPSwitchDec>,<DIPSwitchHex>$_{CR}$**
	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)
Host	**#<BusAdr>,GLOS$_{CR}$** **#<BusAdr>,GET□LOS$_{CR}$**
Answer	**#<BusAdr>,GLOS:<LO1Dec>,<LO2Dec>,<LO3Dec>,<LO1Hex>,<LO2Hex>,<LO3Hex>$_{CR}$**
	Returns the current value of all three PWM output O1, O2 and O3 as decimal and hexadecimal number LO1Dec LO1Hex The current set point of the dimmable output O1 in the range of 0 to 4095 or 0x000 to 0xFFF LO2Dec LO2Hex The current set point of the dimmable output O2 in the range of 0 to 4095 or 0x000 to 0xFFF LO3Dec LO3Hex The current set point of the dimmable output O3 in the range of 0 to 4095 or 0x000 to 0xFFF
Host	**#<BusAdr>,GLO1$_{CR}$** **#<BusAdr>,GET□LO1$_{CR}$**
Answer	**#<BusAdr>,GLO1:<LO1Dec>,<LO1Hex>$_{CR}$**
Host	**#<BusAdr>,GLO2$_{CR}$** **#<BusAdr>,GET□LO2$_{CR}$**
Answer	**#<BusAdr>,GLO2:<LO2Dec>,<LO2Hex>$_{CR}$**
Host	**#<BusAdr>,GLO3$_{CR}$** **#<BusAdr>,GET□LO3$_{CR}$**
Answer	**#<BusAdr>,GLO3:<LO3Dec>,<LO3Hex>$_{CR}$**
	Returns the current set point value of the PWM output Ox as decimal and hexadecimal number LOxDec LOxHex The current set point of the dimmable output Ox in the range of 0 to 4095 or 0x000 to 0xFFF
Host	**#<BusAdr>,SLO1:<LO1Value>$_{CR}$** **#<BusAdr>,SET□LO1:<LO1Value>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
Host	**#<BusAdr>,SLO2:<LO2Value>$_{CR}$** **#<BusAdr>,SET□LO2:<LO2Value>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
Host	**#<BusAdr>,SLO3:<LO3Value>$_{CR}$** **#<BusAdr>,SET□LO3:<LO3Value>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
	Stores the new value LOxValue into the set point register LOx. LOxValue the new set point value for the register LOx in the range of 0..4095 or 0x000 to 0xFFF

Direction	ASCII command
Host	**#<BusAdr>,SMODE:<MODE>$_{CR}$** **#<BusAdr>,SET□MODE:<MODE>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
	Sets the mode for the three LED outputs to the new mode MODE. MODE The new mode for the LED module =0: OFF: All three outputs are immediately switched to 0 =1: ON: All three outputs are dimmed to the values LO1, LO2, LO3 immediately =2: FLASH: All three outputs flashes in the rhythm of the parameterized minimum and maximum times with the three values LO1, LO2 and LO3. =3: FADE: All three outputs fade with the current speed FADE SPEED to the new values LO1, LO2, LO3. =4: RANDOM: All three outputs dices a random number for each channel in the range of 0 to LOx. Then the three outputs fade to the new values with the current FADE SPEED. After a random pause between the configured minimum and maximum time in seconds, this procedure will be repeated. =5: SEQUENCE: All three outputs flashes successively with the set points LO1, LO2, LO3. The three outputs are on for the time period MIN TIME in 1/10s. In between the three outputs are 0 for a time period MAXTIME in 1/10s.
Host	**#<BusAdr>,GMODE$_{CR}$** **#<BusAdr>,GET□MODE$_{CR}$**
Answer	**#<BusAdr>,GMODE:<MODEDec>,<MODEHex>$_{CR}$**
	Returns the current mode of the LED module. MODEDec MODEHex The current mode of the LED module. See MODE description in the command SET MODE
Host	**#<BusAdr>,SFADE:<FADE>$_{CR}$** **#<BusAdr>,SET□FADE:<FADE>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
	Sets the new fading speed for the fading in the two modes FADE and RANDOM FADE The new speed for fading in steps per 1/100s.
Host	**#<BusAdr>,GFADE$_{CR}$** **#<BusAdr>,GET□FADE$_{CR}$**
Answer	**#<BusAdr>,GFADE:<FADEDec>,<FADEHex>$_{CR}$**
	Returns the current fade speed of the LED module in steps per 1/100s. FADEDec FADEHex The current fade speed in steps per 1/100s.

Direction	**ASCII command**
Host	**#<BusAdr>,SMINT:<MINTIME>**$_{CR}$ **#<BusAdr>,SET☐MIN☐TIME:<MINTIME>**$_{CR}$
Answer	**#<BusAdr>,OK**$_{CR}$
	Sets the new minimum time for the LED module. This time is used in the three modes FLASH, RANDOM and SEQUENCE. MINTIME The new value for the minimum time. In the modes FLASH and SEQUENCE, this time defines the ON time span of the three outputs with the values LO1, LO2 and LO3. The OFF time span with the three values 0 is defined with the MAXTIME parameter. The parameter specifies a time in 1/10s. In the mode RANDOM this time defines the minimum time span between two random value changes. The parameter specifies a time span in seconds.
Host	**#<BusAdr>,GMINT**$_{CR}$ **#<BusAdr>,GET☐MIN☐TIME**$_{CR}$
Answer	**#<BusAdr>,GMINT:<MINTIMEDec>,<MINTIMEHex>**$_{CR}$
	Returns the current defined minimum time for the LED module. MINTIMEDec MINTIMEHex The current value for the minimum time. In the modes FLASH and SEQUENCE in 1/10s In the mode RANDOM in seconds.
Host	**#<BusAdr>,SMAXT:<MAXTIME>**$_{CR}$ **#<BusAdr>,SET☐MAX☐TIME:<MAXTIME>**$_{CR}$
Answer	**#<BusAdr>,OK**$_{CR}$
	Sets the new maximum time for the LED module. This time is used in the tree modes FLASH, RANDOM and SEQUENCE. MAXTIME The new value for the maximum time. In the modes FLASH and SEQUENCE, this time defines the OFF time span of the three outputs with the value 0. The ON time span with the three values LO1, LO2 and LO3 is defined with the MAXTIME parameter. The parameter specifies a time in 1/10s. In the mode RANDOM this time defines the maximum time span between two random value changes. The parameter specifies a time span in seconds.
Host	**#<BusAdr>,GMAXT**$_{CR}$ **#<BusAdr>,GET☐MAX☐TIME**$_{CR}$
Answer	**#<BusAdr>,GMAXT:<MAXTIMEDec>,<MAXTIMEHex>**$_{CR}$
	Returns the current defined maximum time for the LED module. MAXTIMEDec MAXTIMEHex The current value for the maximum time. In the modes FLASH and SEQUENCE in 1/10s In the mode RANDOM in seconds.

Direction	ASCII command
Host	**#<BusAdr>,STIMES:<MINTIME>,<MAXTIME>$_{CR}$** **#<BusAdr>,SET□TIMES:<MINTIME>,<MAXTIME>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
	Sets the new minimum and maximum times for the three modes FLASH, RANDOM and SEQUENCE. MINTIME The new value for the minimum time. In the modes FLASH and SEQUENCE, this time defines the ON time span of the three outputs with the values LO1, LO2 and LO3. The OFF time span with the three values 0 is defined with the MAXTIME parameter. The parameter specifies a time in 1/10s. In the mode RANDOM this time defines the minimum time span between two random value changes. The parameter specifies a time span in seconds. MAXTIME The new value for the maximum time. In the modes FLASH and SEQUENCE, this time defines the OFF time span of the three outputs with the value 0. The ON time span with the three values LO1, LO2 and LO3 is defined with the MAXTIME parameter. The parameter specifies a time in 1/10s. In the mode RANDOM this time defines the maximum time span between two random value changes. The parameter specifies a time span in seconds.
Host	**#<BusAdr>,GTIMES$_{CR}$** **#<BusAdr>,GET□TIMES$_{CR}$**
Answer	**#<BusAdr>,GTIMES:<MINTIMEDec>,<MAXTIMEDec>,<MINTIMEHex>,<MAXTIMEHex>$_{CR}$**
	Returns the current minimum and maximum times of the LED module. MINTIMEDec MINTIMEHex The current value for the minimum time. In the modes FLASH and SEQUENCE in 1/10s In the mode RANDOM in seconds. MAXTIMEDec MAXTIMEHex The current value for the maximum time. In the modes FLASH and SEQUENCE in 1/10s In the mode RANDOM in seconds.

Direction	**ASCII command**
Host	**#<BusAdr>,SALL:<MODE>,<LO1>,<LO2>,<LO3>,<MINTIME>,<MAXTIME>,<FADE>$_{CR}$** **#<BusAdr>,SET□ALL:<MODE>,<LO1>,<LO2>,<LO3>,<MINTIME>,<MAXTIME>,<FADE>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
	Sets all parameters for the LED module with one command. MODE: The new mode of the LED module =0: OFF: All three outputs are set to 0. =1: ON: All three outputs are set to the values LO1, LO2 and LO3 =2: FLASH: All three outputs flashes in the rhythm: ON with values LO1, LO2, LO3 for minimum time in 1/10s, OFF with value 0 for maximum time in 1/10s. =3: FADE: All three outputs fade to the new values LO1, LO2, LO3 with the selected FADE SPEED. =4: RANDOM: All three outputs dice a random value between 0 and LOx and dimm to the new values with the selected fade speed. After a random wait time between minimum time and maximum time, this procedure will be repeated. =5: SEQUENCE: All three outputs flashes successively with the set points LO1, LO2, LO3. The three outputs are on for the time period MIN TIME in 1/10s. In between the three outputs are 0 for a time period MAXTIME in 1/10s. LO1: The new value of the PWM output O1 in the range of 0..4095 or 0x000 to 0xFFF LO2: The new value of the PWM output O2 in the range of 0..4095 or 0x000 to 0xFFF LO3: The new value of the PWM output O3 in the range of 0..4095 or 0x000 to 0xFFF MINTIME: The new values for the minimum time. In the mode FLASH the parameter represents a time in 1/10s and defines the ON phase with the values LO1, LO2, LO3. The length of the OFF phase will be defined with the parameter MAXTIME. In the mode RANDOM this parameter represents a time in seconds and defines the minimum pause time between to dices on new values. MAXTIME: The new values for the maximum time. In the mode FLASH the parameter represents a time in 1/10s and defines the OF phase with the value 0. The length of the ON phase with the values LO1, LO2 and LO3 will be defined with the parameter MINTIME. In the mode RANDOM this parameter represents a time in seconds and defines the maximum pause time between to dices on new values. FADE: The new values for the fade speed in steps per 1/100s.

Direction	ASCII command
Host	**#<BusAdr>,GALL$_{CR}$** **#<BusAdr>,GET□ALL$_{CR}$**
Answer	**#<BusAdr>,GALL:<MODEDec>,<LO1Dec>,<LO2Dec>,<LO3Dec>,<MINTIMEDec>, <MAXTIMEDec>,<FADEDec>,<CLO1Dec>,<CLO2Dec>,<CLO3Dec>,<RLO1Dec>, <RLO2Dec>,<RLO3Dec>,<MODEHex>,<LO1Hex>,<LO2Hex>,<LO3Hex>,<MINTIMEHex>, <MAXTIMEHex>,<FADEHex>,<CLO1Hex>,<CLO2Hex>,<CLO3Hex>,<RLO1Hex>, <RLO2Hex>,<RLO3Hex>$_{CR}$**
	Returns all current values for all parameters of the LED module. MODEDec MODEHex — The current mode of the LED module =0: OFF: All three outputs are set to 0. =1: ON: All three outputs are set to the values LO1, LO2 and LO3 =2: FLASH: All three outputs flashes in the rhythm: ON with values LO1, LO2, LO3 for minimum time in 1/10s, OFF with value 0 for maximum time in 1/10s. =3: FADE: All three outputs fade to the new values LO1, LO2, LO3 with the selected FADE SPEED. =4: RANDOM: All three outputs dice a random value between 0 and LOx and dimm to the new values with the selected fade speed. After a random wait time between minimum time and maximum time, this procedure will be repeated. =5: SEQUENCE: All three outputs flashes successively with the set points LO1, LO2, LO3. The three outputs are on for the time period MIN TIME in 1/10s. In between the three outputs are 0 for a time period MAXTIME in 1/10s. LO1Dec LO1Hex — The current value of the output O1 in the range of 0 to 4095 or 0x000 to 0xFFF LO2Dec LO2Hex — The current value of the output O2 in the range of 0 to 4095 or 0x000 to 0xFFF LO3Dec LO3Hex — The current value of the output O3 in the range of 0 to 4095 or 0x000 to 0xFFF MINTIMEDec MINTIMEHex — The current value for the minimum time. In the mode FLASH in 1/10s, in the mode RANDOM in seconds. MAXTIMEDec MAXTIMEHex — The current value for the maximum time. In the mode FLASH in 1/10s, in the mode RANDOM in seconds. FADEDec FADEHex — The current value for the fade speed in steps per 1/100s. CLO1Dec CLO1Hex — The real value of the output O1 in the range of 0 to 4095 or 0x000 to 0xFFF, including all fading and all modes. CLO2Dec CLO2Hex — The real value of the output O2 in the range of 0 to 4095 or 0x000 to 0xFFF, including all fading and all modes. CLO3Dec CLO3Hex — The real value of the output O3 in the range of 0 to 4095 or 0x000 to 0xFFF, including all fading and all modes. RLO1Dec RLO1Hex — The last diced value in mode RANDOM for the output O1 in the range of 0 to 4095 or 0x000 to 0xFFF. RLO2Dec RLO2Hex — The last diced value in mode RANDOM for the output O2 in the range of 0 to 4095 or 0x000 to 0xFFF. RLO3Dec RLO3Hex — The last diced value in mode RANDOM for the output O3 in the range of 0 to 4095 or 0x000 to 0xFFF.

Direction	**ASCII command**
Host	**#<BusAdr>,GCLOS**$_{CR}$ **#<BusAdr>,GET☐CURRENT☐LOS**$_{CR}$
Answer	**#<BusAdr>,GCLOS: <CLO1Dec>,<CLO2Dec>,<CLO3Dec>, <CLO1Hex>,<CLO2Hex>,<CLO3Hex>**$_{CR}$
	Returns all current values of all three outputs O1, O2 and O3 on the LED module. CLO1Dec CLO1Hex The real value of the output O1 in the range of 0 to 4095 or 0x000 to 0xFFF, including all fading and all modes. CLO2Dec CLO2Hex The real value of the output O2 in the range of 0 to 4095 or 0x000 to 0xFFF, including all fading and all modes. CLO3Dec CLO3Hex The real value of the output O3 in the range of 0 to 4095 or 0x000 to 0xFFF, including all fading and all modes.
Host	**#<BusAdr>,GRLOS**$_{CR}$ **#<BusAdr>,GET☐RANDOM☐LOS**$_{CR}$
Answer	**#<BusAdr>,GRLOS: <RLO1Dec>,<RLO2Dec>,<RLO3Dec>, <RLO1Hex>,<RLO2Hex>,<RLO3Hex>**$_{CR}$
	Returns the last diced values for the three outputs O1, O2 and O3 in mode RANDOM. RLO1Dec RLO1Hex The last diced value in mode RANDOM for the output O1 in the range of 0 to 4095 or 0x000 to 0xFFF. RLO2Dec RLO2Hex The last diced value in mode RANDOM for the output O2 in the range of 0 to 4095 or 0x000 to 0xFFF. RLO3Dec RLO3Hex The last diced value in mode RANDOM for the output O3 in the range of 0 to 4095 or 0x000 to 0xFFF.
Host	**#<BusAdr>,GCLO1**$_{CR}$ **#<BusAdr>,GET☐CURRENT☐LO1**$_{CR}$
Answer	**#<BusAdr>,GCLO1:<CLO1Dec>,<CLO1Hex>**$_{CR}$
Host	**#<BusAdr>,GCLO2**$_{CR}$ **#<BusAdr>,GET☐CURRENT☐LO2**$_{CR}$
Answer	**#<BusAdr>,GCLO2:<CLO2Dec>,<CLO2Hex>**$_{CR}$
Host	**#<BusAdr>,GCLO3**$_{CR}$ **#<BusAdr>,GET☐CURRENT☐LO3**$_{CR}$
Answer	**#<BusAdr>,GCLO3:<CLO3Dec>,<CLO3Hex>**$_{CR}$
	Return the real value for the output Ox. CLOxDec CLOxHex The real value of the output Ox in the range of 0 to 4095 or 0x000 to 0xFFF, including all fading and all modes.
Host	**#<BusAdr>,GRLO1**$_{CR}$ **#<BusAdr>,GET☐RANDOM☐LO1**$_{CR}$
Answer	**#<BusAdr>,GRLO1:<RLO1Dec>,<RLO1Hex>**$_{CR}$
Host	**#<BusAdr>,GRLO2**$_{CR}$ **#<BusAdr>,GET☐RANDOM☐LO2**$_{CR}$
Answer	**#<BusAdr>,GRLO2:<RLO2Dec>,<RLO2Hex>**$_{CR}$
Host	**#<BusAdr>,GRLO3**$_{CR}$ **#<BusAdr>,GET☐RANDOM☐LO3**$_{CR}$
Answer	**#<BusAdr>,GRLO3:<RLO3Dec>,<RLO3Hex>**$_{CR}$
	Returns the last dices output value for output Ox in the mode RANDOM. RLOxDec RLOxHex The last diced value in mode RANDOM for the output Ox in the range of 0 to 4095 or 0x000 to 0xFFF.

Direction	**ASCII command**
Host	**#<BusAdr>,SMBADR:<MBUnit>**$_{CR}$ **#<BusAdr>,SET□MODBUS□ADDRESS:<MBUnit>**$_{CR}$
Answer	**#<BusAdr>,OK** $_{CR}$
	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.
Host	**#<BusAdr>,GMBADR**$_{CR}$ **#<BusAdr>,GET□MODBUS□ADDRESS**$_{CR}$
Answer	**#<BusAdr>,GMBADR:<MBUnitDec>,<MBFLASHDec>,<MBUnitHex>,<MBFLASHHex>**$_{CR}$
	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. MBUnitDec MBUnitHex — The current used MODBUS/RTU unit or ASCII address for communication MBFLASHDec MBFLASHHex — The internal stored MODBUS/RTU unit address or ASCII address from the FLASH memory, if the DIP switch is 0.
Host	**#<BusAdr>,RST**$_{CR}$ **#<BusAdr>,RESET**$_{CR}$
Answer	**None**
	Executes a software reset (Reboot) of the module.

8.15 MODBUS – register description

8.15.1 Table of inputs and coils

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 0x00001 to 0x65536. Inputs are usually noted with 1x00001 to 1x65536. 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: 0x00100 for the coil 100, 1x00100 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.

Register	Description
0x00001 1x00001 I:0 R/O ISFADING	Is the module currently fading from one value to another value =0:no fading is active, =1:Fading is running

8.15.2 Table of holding/input registers

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.

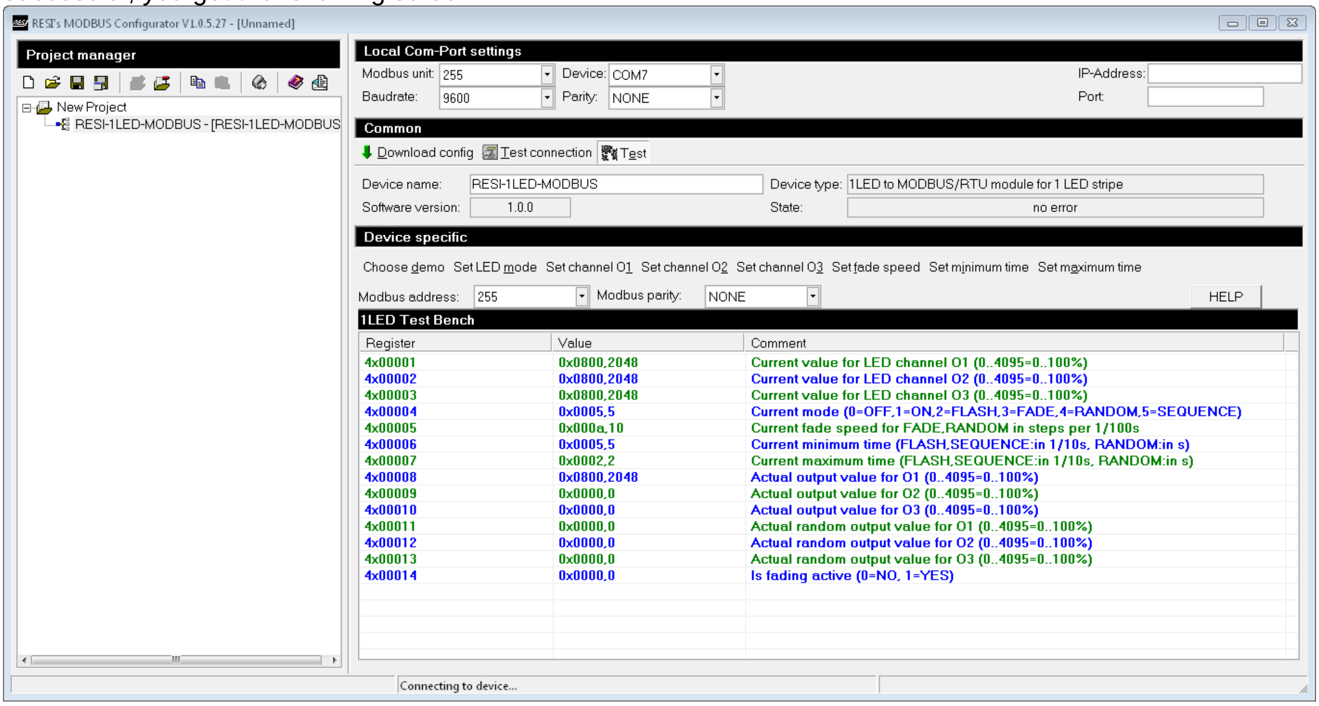
Register	Description
4x00001 3x00001 I:0 R/W LO1	Dimming value for the LED PWM output O1. 0..4095 or 0x000...0xFFF for 0% to 100% brightness If you write to this register, you define a new value for the output O1
4x00002 3x00002 I:1 R/W LO2	Dimming value for the LED PWM output O2. 0..4095 or 0x000...0xFFF for 0% to 100% brightness If you write to this register, you define a new value for the output O2
4x00003 3x00003 I:2 R/W LO3	Dimming value for the LED PWM output O3. 0..4095 or 0x000...0xFFF for 0% to 100% brightness If you write to this register, you define a new value for the output O3
4x00004 3x00004 I:3 R/W MODE	Current mode for the LED module =0:OFF: The three output values LO1, LO2, LO3 are ignored and all three outputs O1, O2, O3 are set to 0. =1:ON: The three outputs O1, O2 and O3 are set immediately to the three output values LO1, LO2 and LO3. =2:FLASH: The three outputs o1, O2, O3 are set to the output values LO1, Lo2 and LO3 for the configured time span MINTIME in 1/10s. Afterwards all three outputs are set to 0 for the configured time span MAXTIME in 1/10s. This cycle will be repeated as long as this mode is active. =3: FADE: Whenever you write a new value into one of the three output registers LO1, LO2 or LO3, the module fades the current output registers CLO1, CLO2 and CLO3 to the new values. The fade speed is defined in steps per 17100s in the register FADE SPEED. =4:RANDOM: After a random time span between MINTIME and MAXTIME in seconds, the system dices three new output values for the registers RLO1, RLO2 and RLO3. For the diced values those rules are applied: New diced value for RLOx is between 0 and Lox Then the system fades with the configured FADESPEED to the new values RLO1, ROL2 und ROL3. This procedure will be repeated. =5: SEQUENCE: All three outputs flashes successively with the set points LO1, LO2, LO3. The three outputs are on for the time period MIN TIME in 1/10s. In between the three outputs are 0 for a time period MAXTIME in 1/10s. If you write to this register, you set up a new mode for the module.

Register	Description
4x00005 3x00005 I:4 R/W FADESPEED	Current dimming or fading speed for the outputs in mode FADE and RANDOM in steps per 1/100s. The smallest value is 1. Every 1/100s the system add/subtracts this FADESPEED value from the three outputs CLO1, CLO2 and CLO3. So the value 1 in FADESPEED means, that if CLO1 hast the start value 0 and the new value LO1 is 4095, the fade up process will last for 40.95 seconds. This is the slowest fading speed of the module. A value of 4095 or more defines the fastest fade speed. After 1/100s the new value will be valid. If you write on this register, you will redefine the FADESPEED
4x00006 3x00006 I:5 R/W MINTIME	For the mode FLASH and SEQUENCE this value means the ON time of the flashing or sequence cycle in 1/10s. For the mode RANDOM this value defines the minimum time span for a new dicing cycle. The dicing cycle dices new random values for the registers RLO1, RLO2 and RLO3 and fades to this new brightness values. The time span is defined in seconds. If you write to this register, you will redefine this value.
4x00007 3x00007 I:6 R/W MAXTIME	For the mode FLASH and SEQUENCE this value means the OFF time of the flashing or sequence cycle in 1/10s. For the mode RANDOM this value defines the maximum time span for a new dicing cycle. The dicing cycle dices new random values for the registers RLO1, RLO2 and RLO3 and fades to this new brightness values. The time span is defined in seconds. If you write to this register, you will redefine this value.
4x00008 3x00008 I:7 R/O CLO1	The real value on the LED PWM output O1 including diming and the current mode. 0..4095 or 0x000...0xFFF for 0% to 100% brightness
4x00009 3x00009 I:8 R/O CLO2	The real value on the LED PWM output O2 including diming and the current mode. 0..4095 or 0x000...0xFFF for 0% to 100% brightness
4x00010 3x00010 I:9 R/O CLO3	The real value on the LED PWM output O3 including diming and the current mode. 0..4095 or 0x000...0xFFF for 0% to 100% brightness
4x00011 3x00011 I:10 R/O RLO1	The last diced random number for the LED PWM output O1 in mode RANDOM. 0..4095 or 0x000...0xFFF for 0% to 100% brightness
4x00012 3x00012 I:11 R/O RLO2	The last diced random number for the LED PWM output O2 in mode RANDOM. 0..4095 or 0x000...0xFFF for 0% to 100% brightness
4x00013 3x00013 I:12 R/O RLO3	The last diced random number for the LED PWM output O3 in mode RANDOM. 0..4095 or 0x000...0xFFF for 0% to 100% brightness
4x00014 3x00014 I:13 R/O ISFADING	Is the module currently fading from one value to another value =0:no fading is active, =1:fading is running

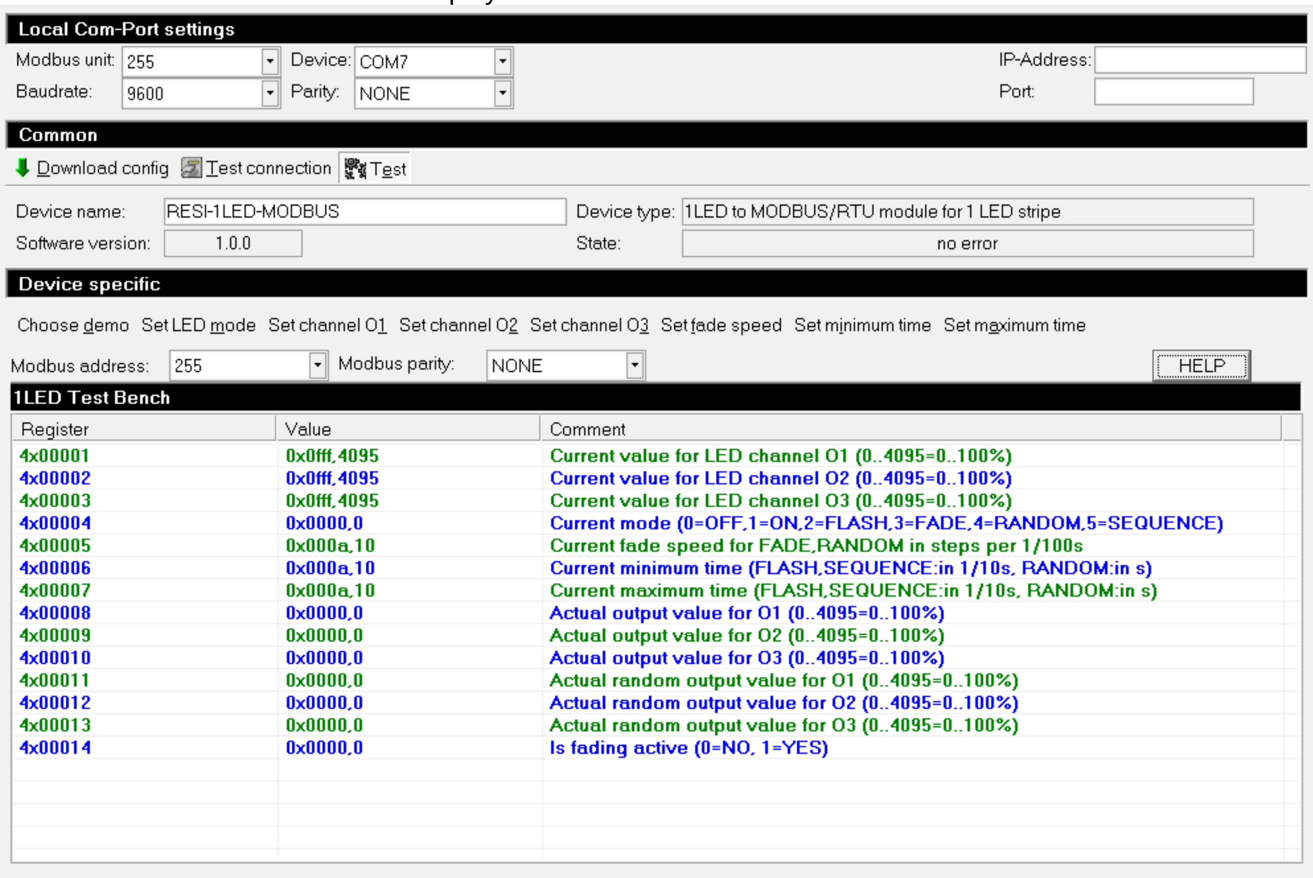
Register	Description
4x6001 3x6001 I:6000 W/O RESET SYSTEM	If the host writes to this register, the module executes a soft reset (reboot).
4x65222 3x65222 I:65221 R/W MODBUS UNIT ADDRESS	If the host reads this register, the current programmed unit ID is returned. All values above unit ID 255 define also 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.

8.16 Module test with RESI MODBUSConfigurator software

Establish a connection between the module and our software tool RESI MODBUSConfigurator. If this is successful, you get the following screen:



You can enable/disable the testing mode with the button „TEST". Every 5 seconds the software reads new data from the module and refreshes the displayed data on the screen:



For the 1LED modules the software offers the additional buttons:
„Choose demo“ to select a predefined demo mode of the module. The following menu is displayed:

- Demo OFF
- Demo O1 100%
- Demo O2 100%
- Demo O3 100%
- Demo O1+O2+O3 100%
- Demo O1+O2+O3 0%
- Demo fade O1 100% fast
- Demo fade O1 0% fast
- Demo fade O2 100% normal
- Demo fade O2 0% normal
- Demo fade O3 100% slow
- Demo fade O3 0% slow
- Demo flash O1 1/10s
- Demo flash O2 1/2s
- Demo flash O3 5s
- Demo flash O1+O2+O3 1s on, 4s off
- Demo random O1+O2+O3 100% 3s fast
- Demo random O1+O2+O3 50% 5s normal
- Demo random O1+O2+O3 100% 10s-30s slow
- Demo random O1 100% 2s very fast
- Demo sequence 100% slow
- Demo sequence 50% fast

Choose a demo mode from the list, explore the behavior on the connected LED stripe and check, what registers are changed by the selected demo mode. This mode is for better understanding, what for possibilities our LED module offers.

„Set LED mode“ to select a new mode of the LED module. The following choice is displayed:

- OFF
- ON
- FLASH
- FADE
- RANDOM
- SEQUENCE

To understand those modes, read the description for the MODBUS register MODE.

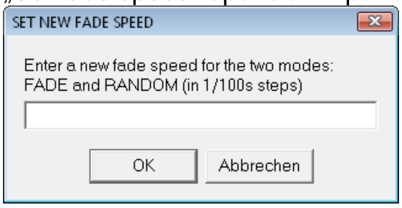
„Set channel O1“ to set a new value into the register LO1. The system shows the following options:

- Set O1 to 0 (0%)
- Set O1 to 512 (12,5%)
- Set O1 to 1024 (25%)
- Set O1 to 2048 (50%)
- Set O1 to 3072 (75%)
- Set O1 to 4095 (100%)
- Set O1 to xx%

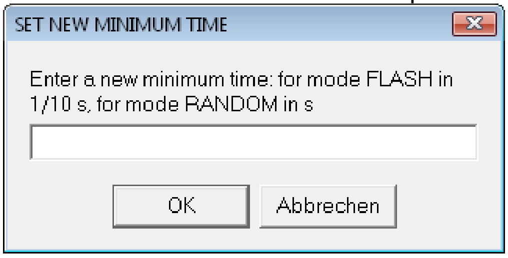
If you select the last menu item, an input box will be opened, where you can enter a new value between 0 and 4095.

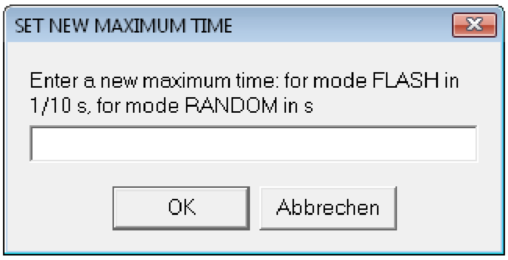
„Set channel O2“ and „Set channel O3“ offer the same functionality as the „Set channel O1“ button, but for the other two channels.

„Set fade speed“ opens an input box for entering a new dimming speed for the modes FADE and RANDOM:



„Set minimum time“ and „Set maximum time“ define the new values for both registers minimum and maximum time. For the mode FLASH the ON and OFF times are defined in 1/10s. In the mode RANDOM you can define the minimum and maximum time span between two dices of random values. The time values are in seconds:





9 RESI-4LED-MODBUS, RESI-4LED-ASCII

9.1 Product description

This IO module offers the following features:

- 12 dimmable PWM output channels for LED stripes, 0..48Vdc, max. 5A each channel, organized in 4 groups with three channels A,B and C each
- Each LED group offers six selectable modes: OFF, ON, FLASHING, FADING, RANDOM, SEQUENCE
- External power supply for LED stripes, 0..48Vdc, max. 15A
- Galvanic insulated RS485 interface for communication with a host system
- RESI-4LED-MODBUS: MODBUS/RTU slave protocol
- RESI-4LED-ASCII: MODBUS/RTU slave and text based ASCII protocol
- Power supply 12-48Vdc
- DIP switch for setting the baud rates, the type of interface and the bus number
- LED indicator for the communication
- Mounting onto EN50022 DIN rail or wall mounting

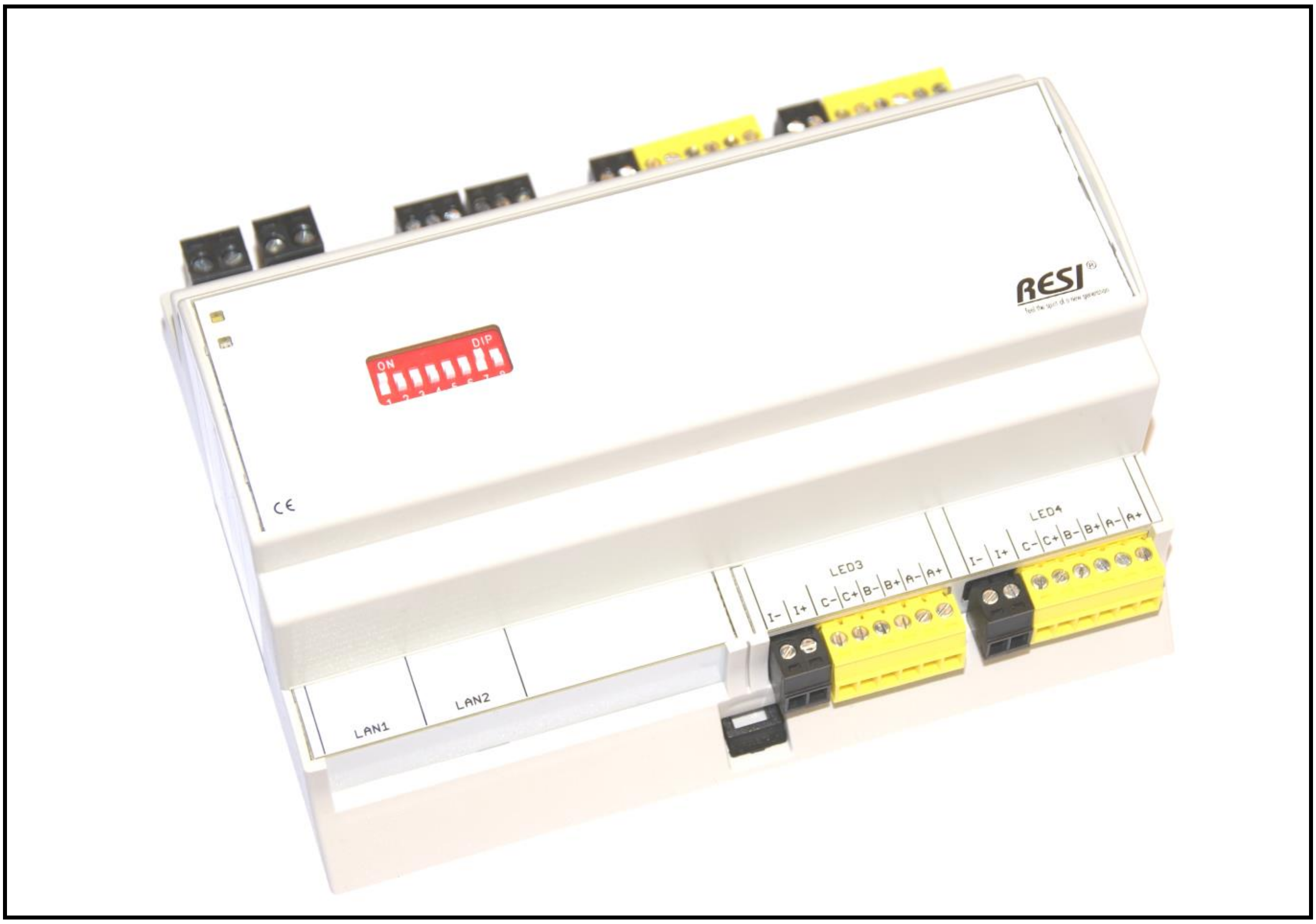


Illustration: Our IO module

9.1.1 The modes of the LED module

The LED module offers six modes. Each of the four LED groups has its own mode. You can switch the mode by setting a special register via MODBUS/RTU or by executing the #SMODEx ASCII command. Be aware that the converter does not save a mode in remanent memory. After reset the module starts always in mode ON for all four LED groups!

9.1.2 LED mode OFF

In this mode all three outputs of a LED group are switched to 0. It doesn't matter, what values are actual in the three set point registers LOx. The registers for the three actual output values CLOx return always the value 0.

9.1.3 LED mode ON

In this mode all three outputs are switched immediately to the current values in the three registers Lox of the corresponding LED group. The three registers for the actual output values CLOx of the affected LED group deliver always the same value as the three registers LOx to indicate, that the values are really outputted to the three PWM channels of the LED group.

9.1.4 LED mode FLASH

In this mode all three outputs are switched as a recycler relay between the three current values in the registers LOx and 0 of the LED group. While ON time span, the module outputs the three values of the set point registers Lox to the real outputs for a timespan defined in the register MINIMUM TIMEx in 1/10s. In this time the registers for the actual output values CLOx of the affected LED group deliver always the same value as in the registers LOx to indicate, that the values are really outputted to the three PWM channels of the affected LED group. Then the converter switches all three channels to 0 for the OFF time span. This time span is defined with the value of the MAXIMUM TIMEx register in 1/10s. In this time span the registers for the actual output values CLOx of the affected LED group deliver always the value 0. This ON/OFF cycle is repeated endlessly.

Steps for FLASH:

Step 1: Output of the three set point values LOx A, LOx B, LOx C to the real PWM outputs
Step 2: Wait for MINIMUM TIMEx in 1/10s
Step 3: output of the values 0, 0, 0 to the real PWM outputs
Step 4: Wait for MAXIMUM TIMEx in 1/10s
Step 5: continue with step 1

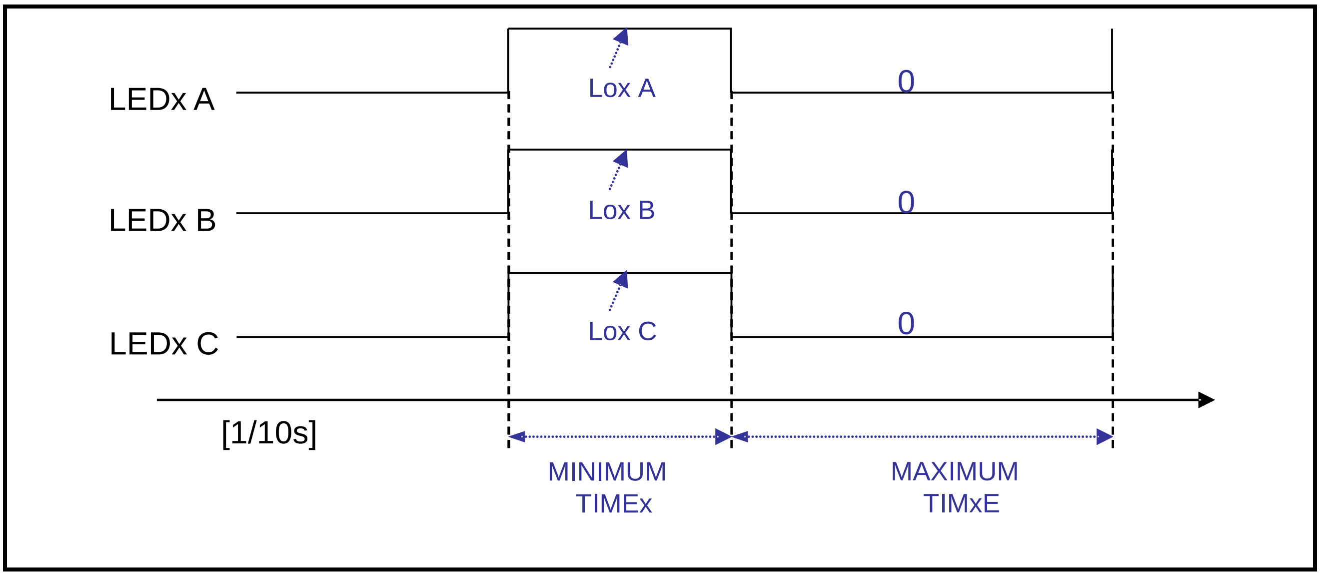


Illustration: timing diagram of mode FLASH

9.1.5 LED Modus FADE

In this mode the converter doesn't change the three PWM outputs of a LED group not immediately, if the set point registers Lox are changed. No, it uses a ramp to change slowly from the current value to the new value. This ramp is defined in the register FADE SPEEDx. The setup is done in steps per 1/100s and is valid for all three channels of the affected LED group. To set a new value write into one of the three set point registers LOx. The LED group fades each output channel from the current value to the new set point value. If you read the registers CLOx of the LED group while fading, you will get every value change from the old value to the new value for each channel. Also the register IS FADE ACTIVEx will return a 1 while fading is running at least on one of the three channels of a LED group. When the LED group reaches the new set point values, reading of the registers CLOx will return the same values as in the registers LOx for the affected LED group. Also the register value of IS FADE ACTIVEx will be 0.

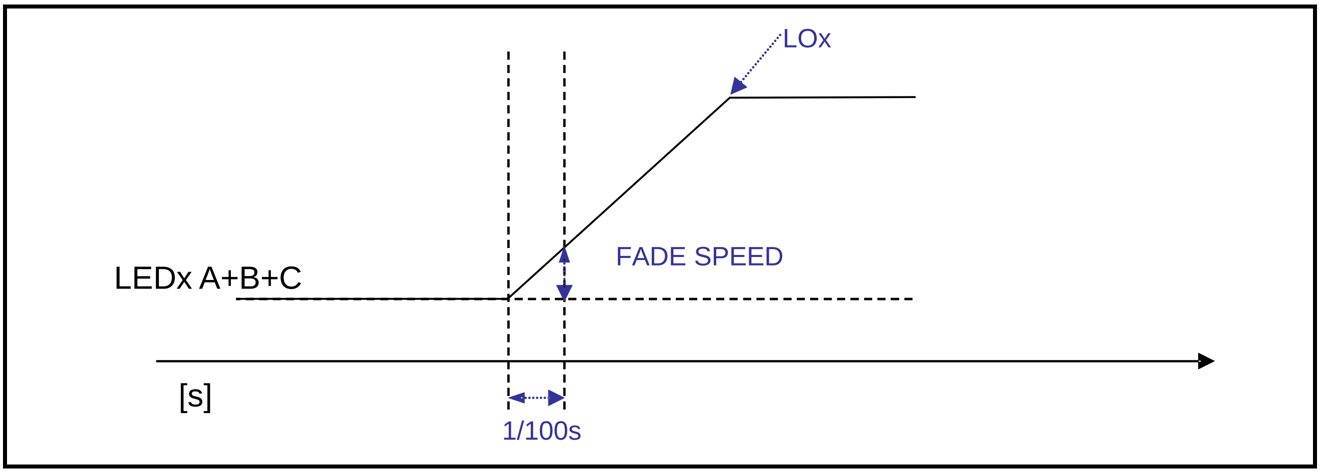


Illustration: timing diagram of mode FADE

9.1.6 LED Modus RANDOM

In this mode the module generates random values for each of the three outputs of a LED group. For this random number guessing process, you can setup a time interval. If this time interval expires the system dices new random values for the three outputs of the affected LED group. The time interval is defined by the register MINIMUM TIMEx and the register MAXIMUM TIMEx in seconds. The system generates a random time interval between those two parameters. If the time expires, the system dices new random values for the three registers RLOx of the affected LED group. Then the system fades the current values stored in the registers CLOx to the new random values RLOx. This fade ramp is defined in the register FADE SPEEDx. The setup is done in steps per 1/100s. If you read the registers CLOx while fading, you will get every value change from the old values to the new end values RLOx. Also the register IS FADE ACTIVE 4x00014 will return a 1 while fading is running at least on one of the three output channels of the LED group. When the module reaches the new values, reading of the registers CLOx will return the same values as stored in the registers RLOx. Also the register value of IS FADE ACTIVEx will be 0. The diced values in the registers RLOx will be in the range of 0 to LOx.

Steps for RANDOM:

Step 1: Dice three random numbers in the range of 0..LOx and store the values in RLOx
Step 2: Dice a random wait period between MINIMUM TIMEx and MAXIMUM TIMEx in seconds
Step 3: Fade up or down from the actual output values CLOx to the new end values RLOx
Step 4: If the random wait period is over, continue with step 1

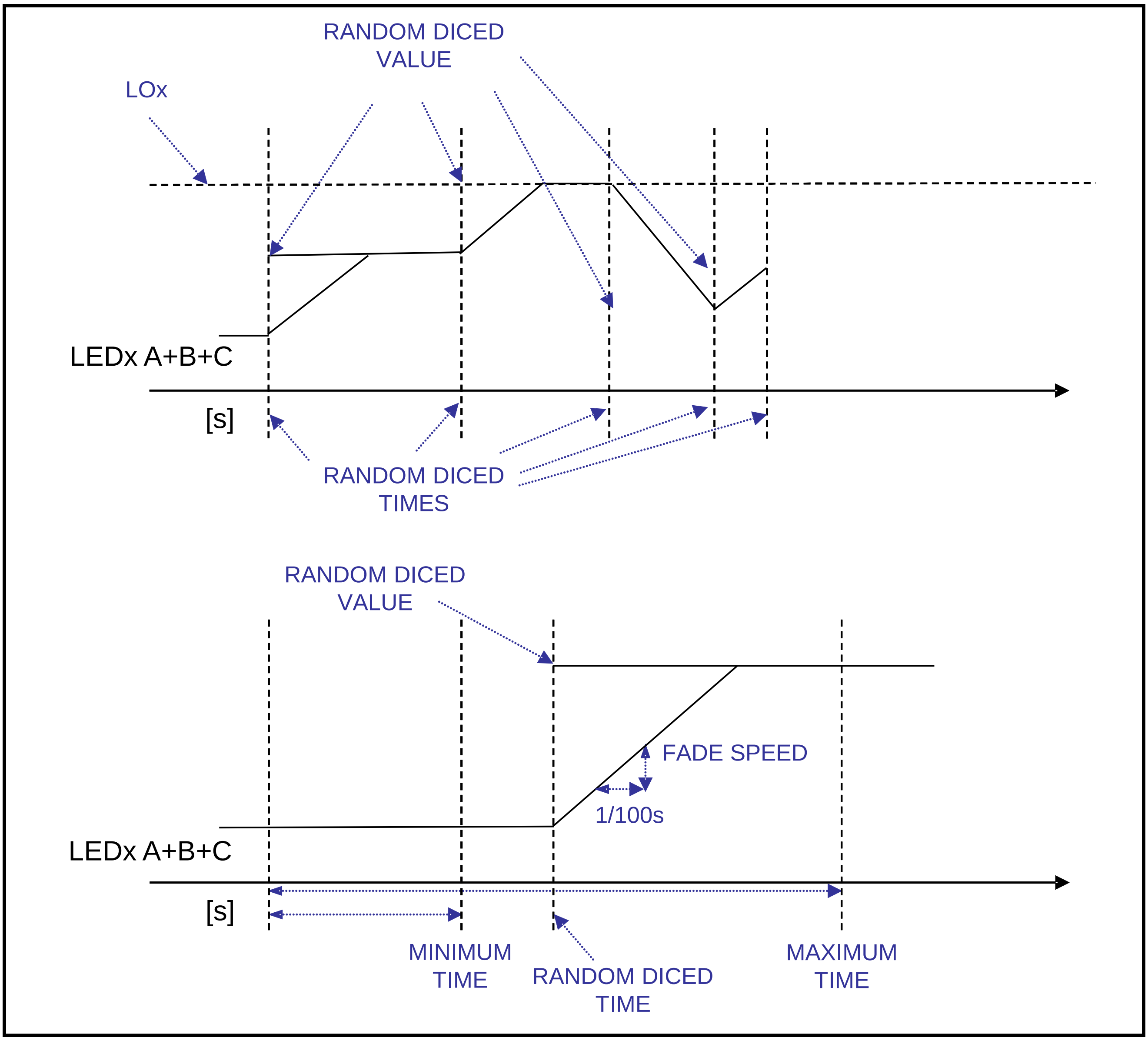


Illustration: timing diagram of mode RANDOM

9.1.7 LED Modus SEQUENCE

In this mode, the module creates a sequential flash light with the three PWM outputs of a LED group. The outputs flashes between the three set points LOx A, LOx B and LOx C and 0 of the affected LED group in sequence. In the first ON phase the module sets the real output CLOx A to the set point LOx A, the other two outputs are set to 0. This phase lasts for MINIMUM TIMEx in 1/10s. While this period of time, the current value register CLOx A delivers the same value as stored in LOx A, and the other two current value registers CLOx B and CLOx C deliver the value 0. Then the module switches all three outputs to 0 for a time period defined with the register MAXIMUM TIMEx in 1/10s (OFF time period). While this period of time, all three output registers CLOx deliver the value 0. Now the system repeats the ON phase with the next set point register LOx B. The two registers CLOx A and CLOx C are set to 0 in this phase. Next the OFF time period is executed. The last phase is the ON phase with the register LOx C. The two registers CLOx A and CLOx B are 0 in this phase. The last OFF time period is executed. This three times ON/OFF cycle is repeated endlessly.

Steps for SEQUENCE:

Step 1: Output the three set points LOx A, 0, 0 to the three PWM outputs
Step 2: wait for MINIMUM TIMEx in 1/10s
Step 3: Output the values 0, 0, 0 to the three PWM outputs
Step 4: wait for MAXIMUM TIMEx in 1/10s
Step 5: Output the three set points 0, LOx B, 0 to the three PWM outputs
Step 6: wait for MINIMUM TIMEx in 1/10s
Step 7: Output the values 0, 0, 0 to the three PWM outputs
Step 8: wait for MAXIMUM TIMEx in 1/10s
Step 9: Output the three set points 0, 0, LOx C to the three PWM outputs
Step 10: wait for MINIMUM TIMEx in 1/10s
Step 11: Output the values 0, 0, 0 to the three PWM outputs
Step 12: wait for MAXIMUM TIMEx in 1/10s
Step 13: continue with step 1

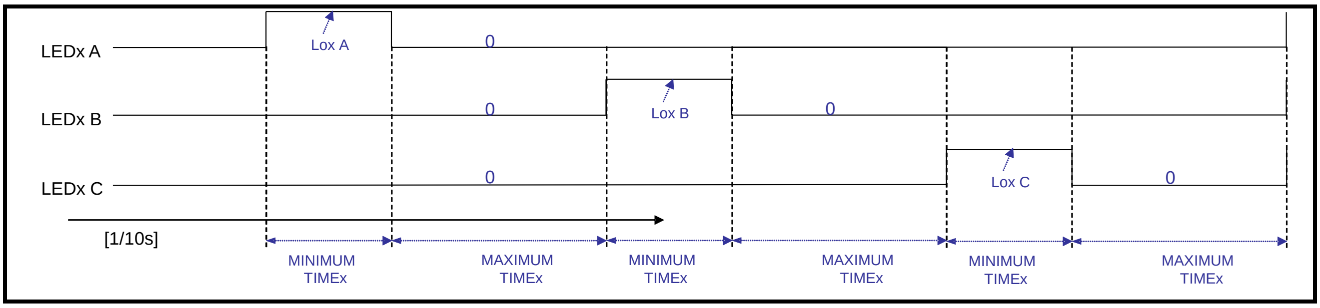


Illustration: timing diagram of mode SEQUENCE

9.2 Technical data

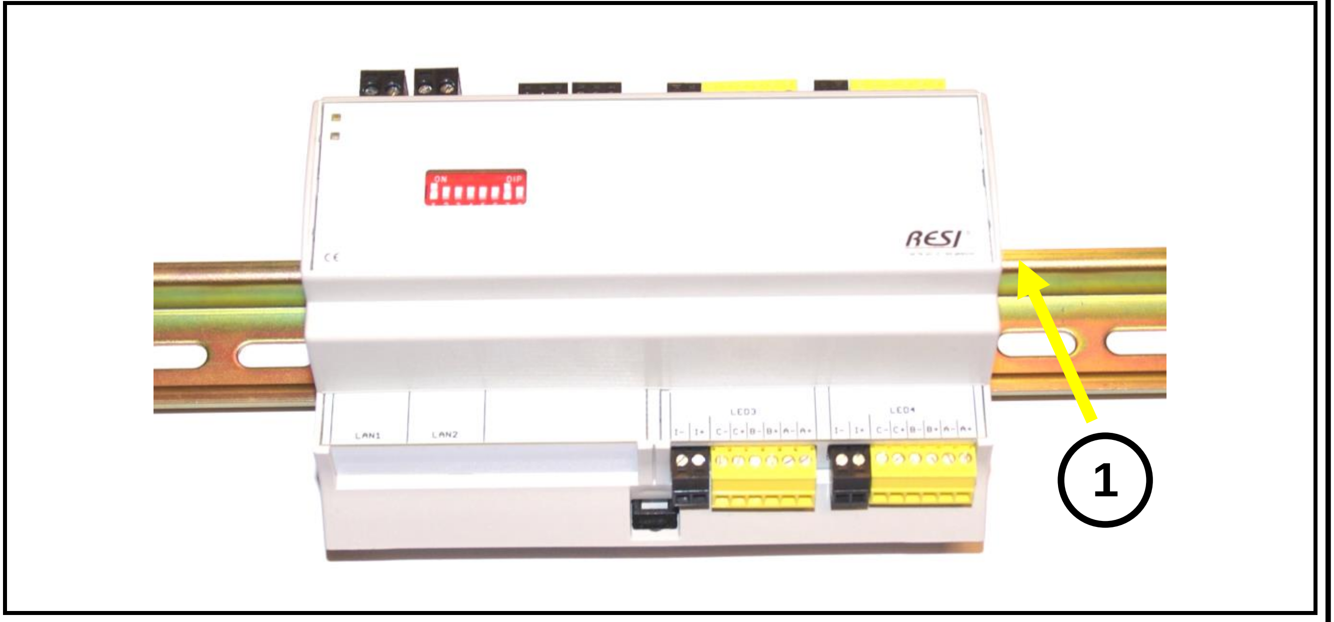
Technical Data			
Power supply			
Supply voltage	12-48V= +/-10%	Storage temperature	-20...80 °C
Power LED	Yes	Operating Temperature	0...60°C
Power consumption	<0.8W	Humidity	25...90 % rH non-condensing
		Protection Class	IP20 (EN 60529)
		Dimensions LxWxH	143mm x 110mm x 62mm
		Weight	260g
		Mounting	On DIN EN50022 rail or wall mounting
ASCII/Modbus Interface			
Protocol	ASCII or Modbus/RTU		
Type	RS485		
Baud rates	4800 to 256000Bd/8/N or E/1		
Cable Connection	Via removable clamps		
LED indicator	Yes		
Galvanic insulation	No		
LED stripe output			
Total amount of outputs	12 individual dimmable outputs		
LED groups	4 LED groups organized with 3 individual dimmable outputs each with own power supply		
Amount of outputs per LED group	3 individual dimmable outputs		
Signal	PWM with 400Hz		
LED Stripes	RGB Dual white Mono color		
LED connection	Via common anode		
Output voltage	0..48Vdc		
Output current	Max. 5A per LED output		
LED power supply	0..48Vdc,max 15A 180W@12Vdc 360W@24vdc 720W@48Vdc		
Cable connection	Via removable clamps		
Galvanic insulation to the rest of the module	Yes All LED groups are internally coupled via the common ground		
LED indicator	No		
Clamps			
Clamp wire cross section	Max. 1,5 mm²		
Tightening torque	Max. 0.5Nm	**CE conformity**	Yes

9.3 Assembling

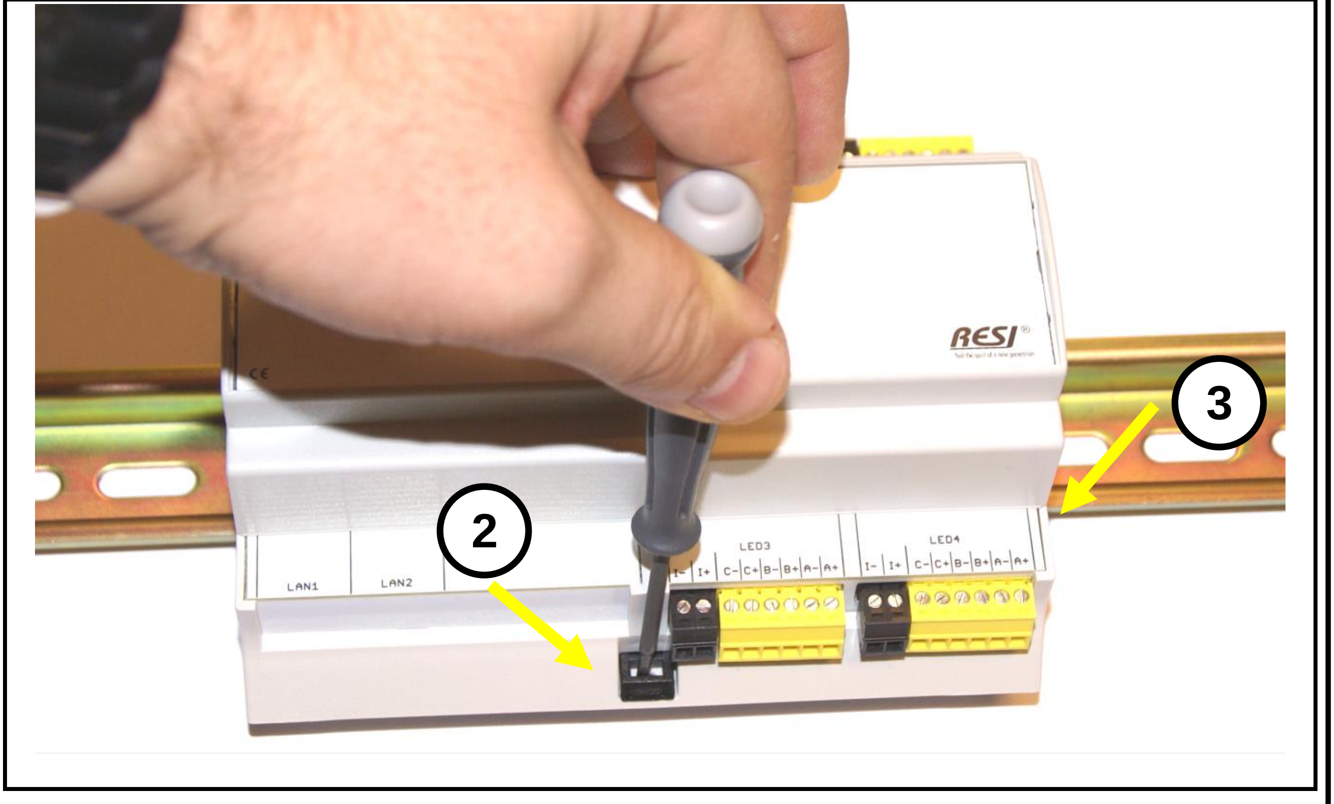
Our IO modules are designed for mounting onto a 35mm DIN-EN50022 rail or for wall mounting. Please not, that in the following mounting description we use only symbolic photos of our IO modules.

9.3.1 Mounting of a DIN EN50022 rail

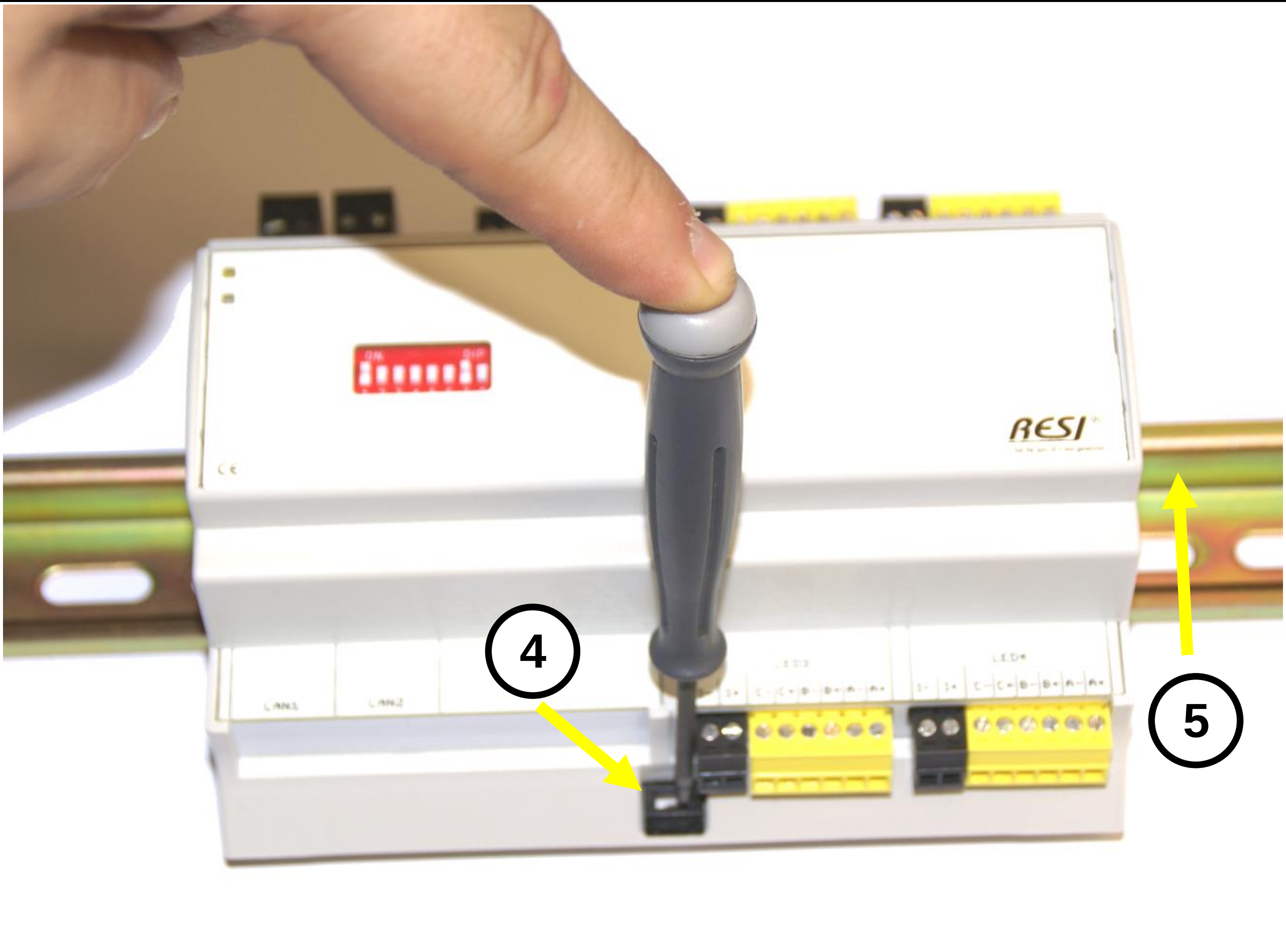
First snap in the top part of the module into the DIN rail (1). The bottom part of the module is not snapped into the DIN rail at this moment.



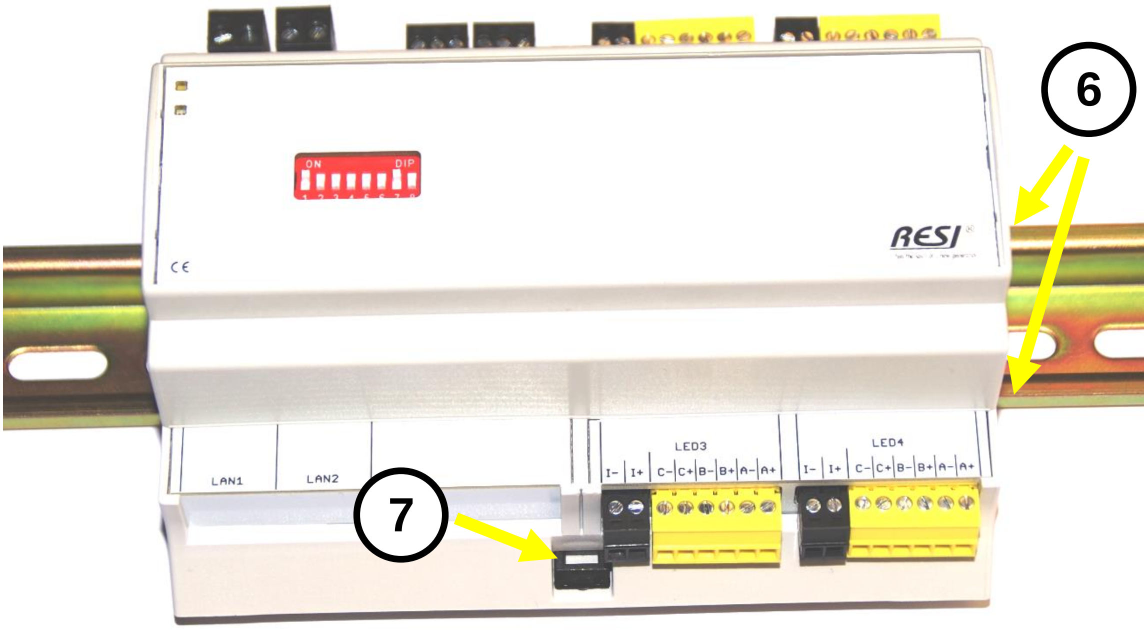
Then open the black hook with a screw driver (2). Now press the module with the opened hook onto the DIN rail until both sides of the module snap into the DIN rail (3). Release the screw driver now. The hook snaps into the DIN rail and the module is now mounted correctly onto the DIN rail.



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.

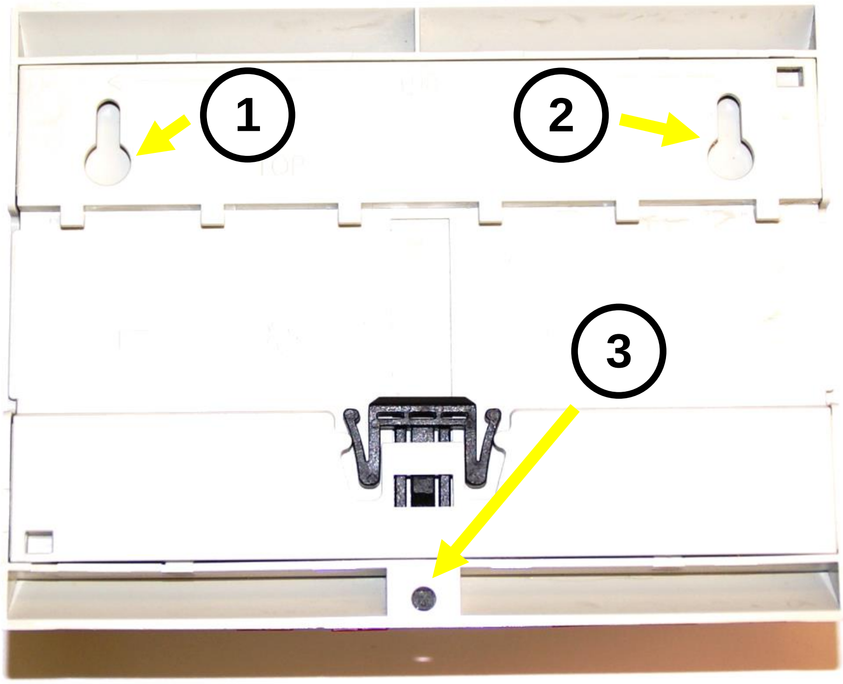


The module is correctly mounted, if the module has snapped into the DIN rail on both sides of the housing (6) and if the hook has snapped in too (7).

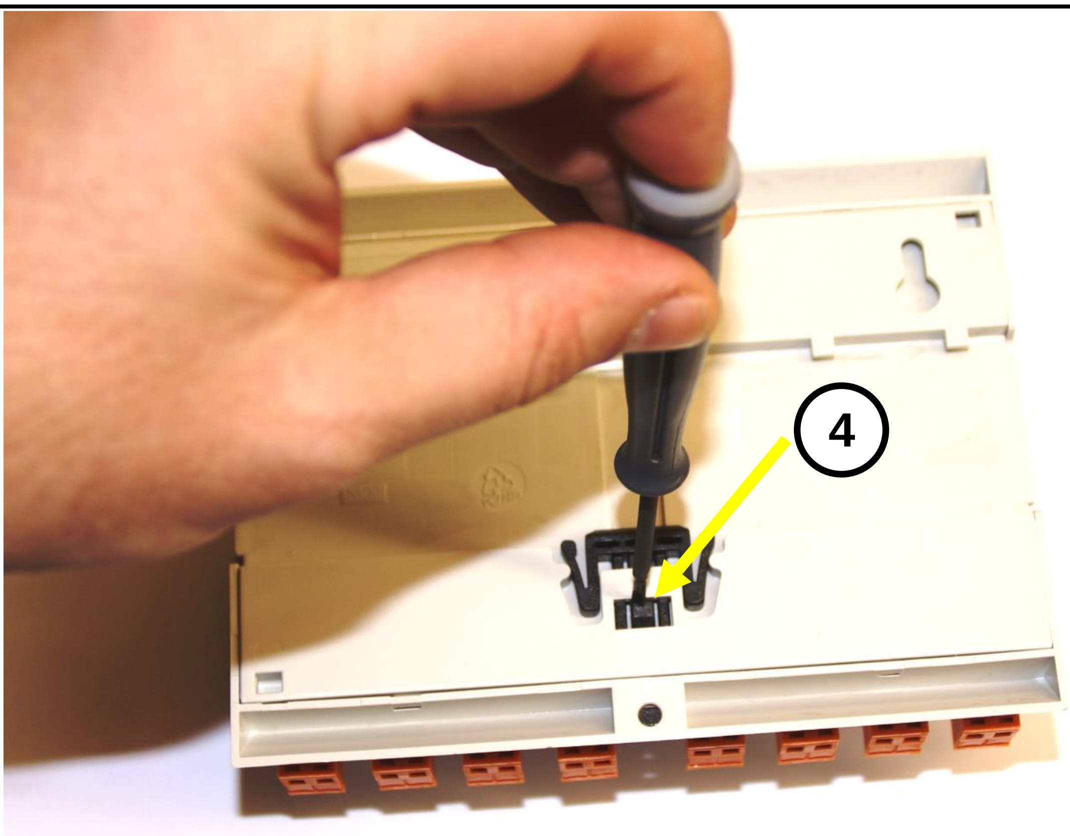


9.3.2 Wall mounting

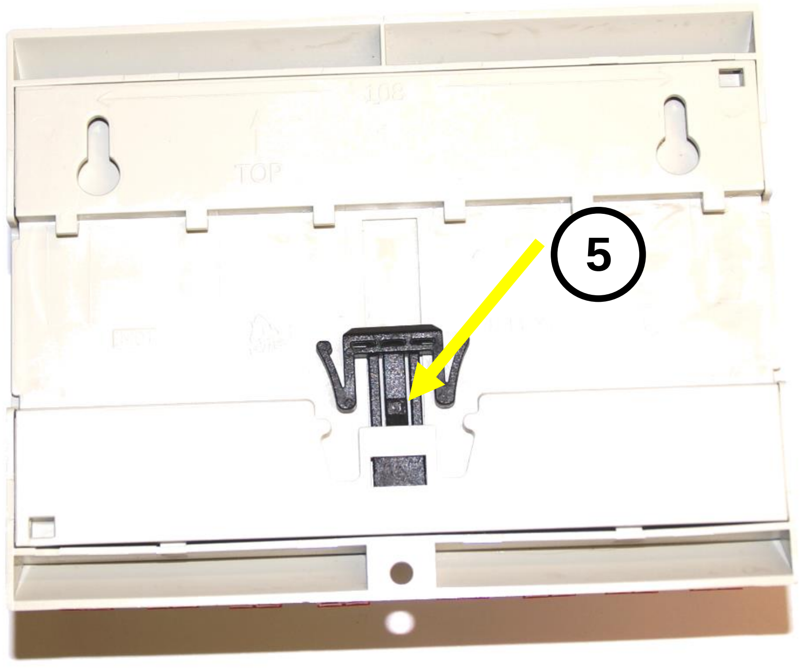
Our modules can also be mounted onto a wall. Turn over the module as shown in the picture below:

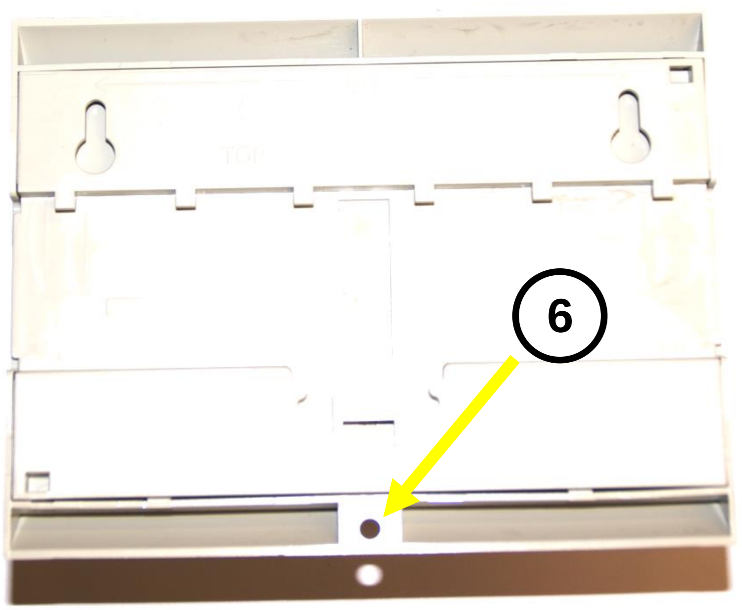


You will notice, that there are two holes for wall hooks or screws on the top side of the housing. (1) and (2). On the bottom side you will notice a small hole for a screw to fix the housing on the wall from the front (3). But first we have to remove the hook, which blocks the screw hole in the housing.



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.

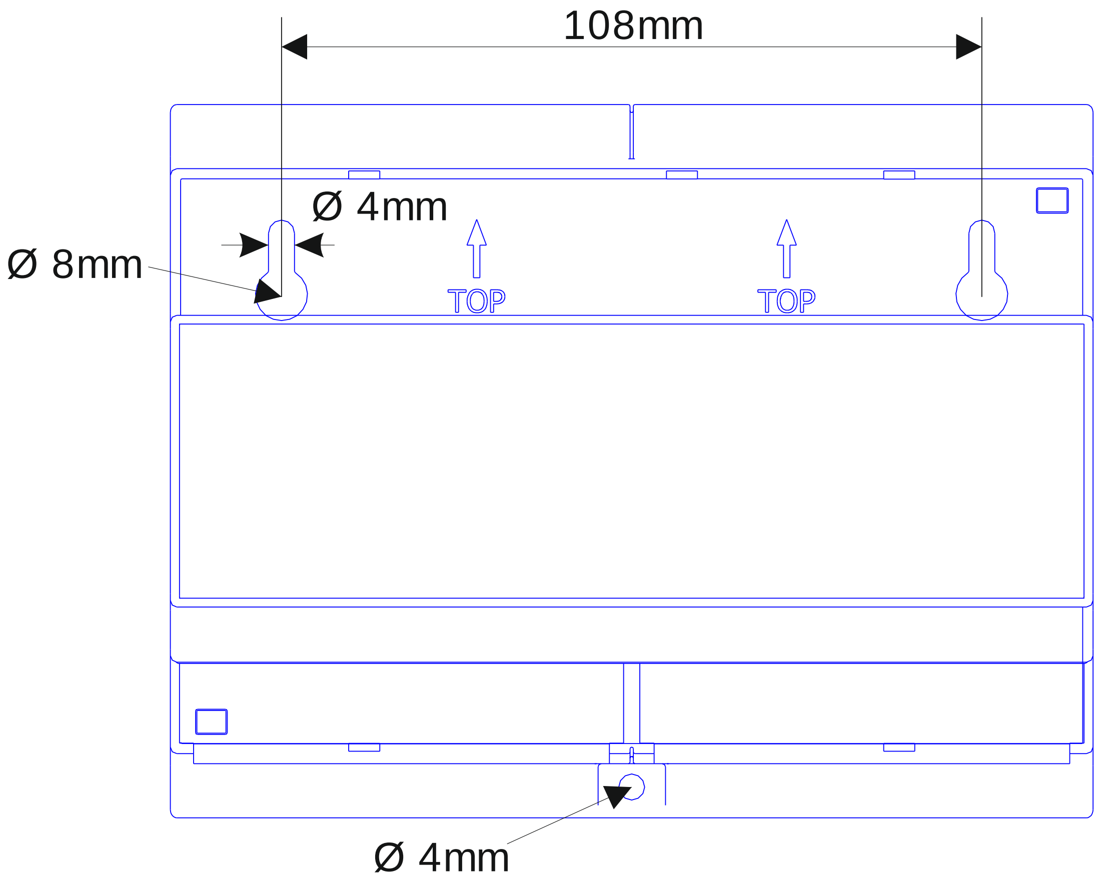


Illustration: Bottom view of the module with holes for wall mounting

9.4 Connection diagram

In the below drawing you will find the correct cabling of the IO module.

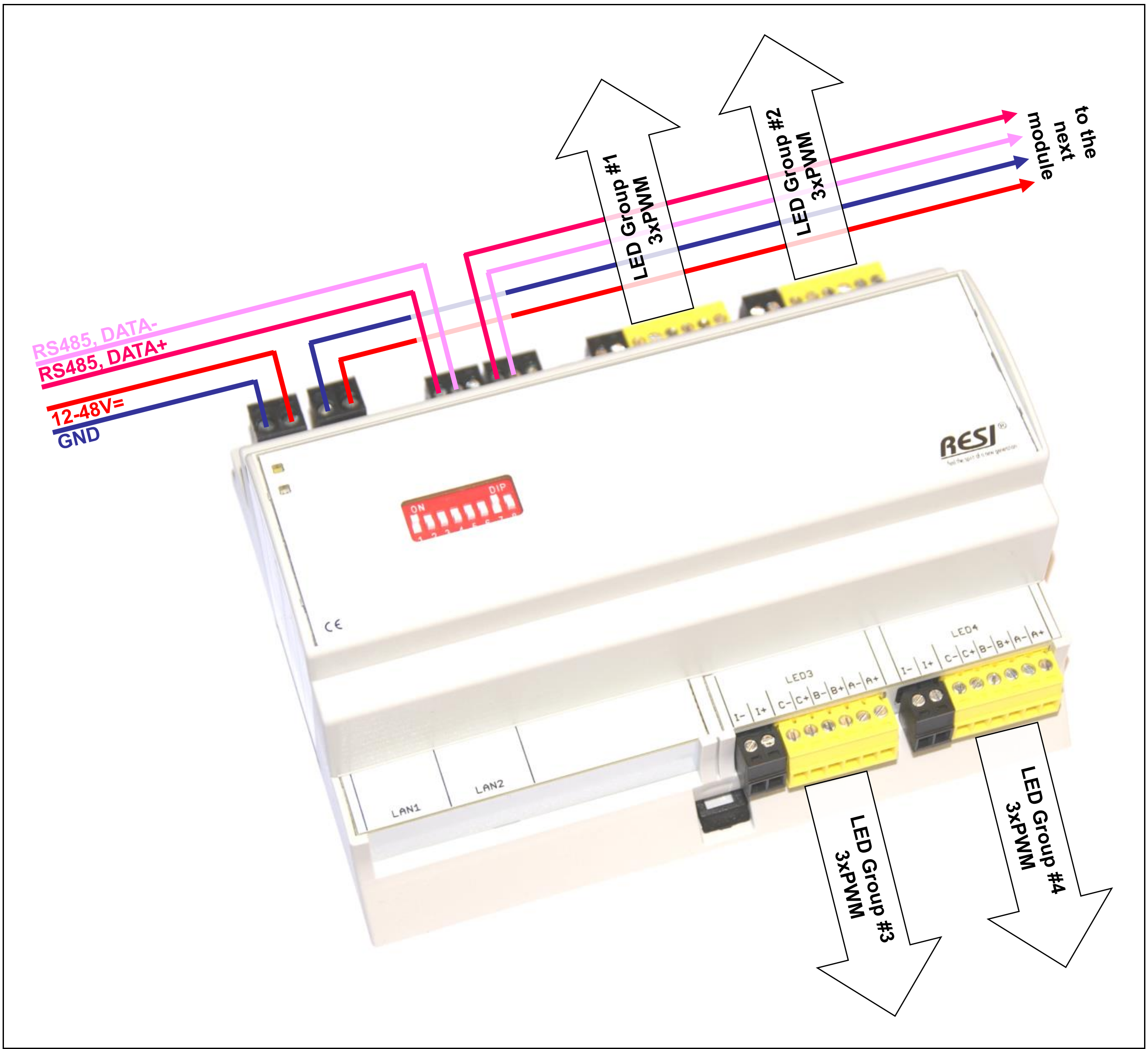


Illustration: cabling of the IO module

9.5 Clamps, DIP switch settings an LED indicators

The IO module offers the following clamps:

CLAMPS	RESI-4LED-MODBUS, RESI-4LED-ASCII
L+ M-	Power supply via two separated plug-in 2-pin terminal blocks. For daisy chain IN and OUT power supply of many modules L+: 12-48 V= M-: Ground
SIO1 1=A+ 2=B- 3=GND	RS485 ASCII or MODBUS/RTU serial interface IN A+: RS485 DATA+ signal B-: RS485 DATA- signal GND: RS485 ground signal
SIO2 1=A+ 2=B- 3=GND	RS485 ASCII or MODBUS/RTU serial interface OUT A+: RS485 DATA+ signal B-: RS485 DATA- signal GND: RS485 ground signal
LED GROUP #1 LED1 I+, I- A+, A- B+, B- C+, C-	Dimmable LED stripe group with three PWM outputs: I+, I-: Power supply 0..48Vdc, max. 15A A+,A-: PWM output A max. 5A, A+ common anode B+,B-: PWM output B max. 5A, B+ common anode C+,C-: PWM output C max. 5A, C+ common anode HINT: The clamps I+, A+, B+ and C+ are internally bridged together! The LED stripes must have a common anode pin.
LED GROUP #2 LED2 I+, I- A+, A- B+, B- C+, C-	Dimmable LED stripe group with three PWM outputs: I+, I-: Power supply 0..48Vdc, max. 15A A+,A-: PWM output A max. 5A, A+ common anode B+,B-: PWM output B max. 5A, B+ common anode C+,C-: PWM output C max. 5A, C+ common anode HINT: The clamps I+, A+, B+ and C+ are internally bridged together! The LED stripes must have a common anode pin.
LED GROUP #3 LED3 I+, I- A+, A- B+, B- C+, C-	Dimmable LED stripe group with three PWM outputs: I+, I-: Power supply 0..48Vdc, max. 15A A+,A-: PWM output A max. 5A, A+ common anode B+,B-: PWM output B max. 5A, B+ common anode C+,C-: PWM output C max. 5A, C+ common anode HINT: The clamps I+, A+, B+ and C+ are internally bridged together! The LED stripes must have a common anode pin.
LED GROUP #4 LED4 I+, I- A+, A- B+, B- C+, C-	Dimmable LED stripe group with three PWM outputs: I+, I-: Power supply 0..48Vdc, max. 15A A+,A-: PWM output A max. 5A, A+ common anode B+,B-: PWM output B max. 5A, B+ common anode C+,C-: PWM output C max. 5A, C+ common anode HINT: The clamps I+, A+, B+ and C+ are internally bridged together! The LED stripes must have a common anode pin.

Table: Description of the terminal blocks of the IO module

<table>
<tr><th>DIP+LED</th><th>RESI-4LED-MODBUS, RESI-4LED-ASCII</th></tr>
<tr><td>DIP SWITCH
1=ADR0
2=ADR1
3=ADR2
4=ADR3
5=BR0
6=BR1
7=BR2
8=PARITY</td><td>DIP switch to setup the IO module
ADR: 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:
<table>
<tr><td>ADR3</td><td>ADR2</td><td>ADR1</td><td>ADR0</td><td>MODBUS/RTU unit number or ASCII bus number</td></tr>
<tr><td>OFF</td><td>OFF</td><td>OFF</td><td>OFF</td><td>Internal MODBUS/RTU unit number is used from the FLASH memory in the range of 0 to 255.</td></tr>
<tr><td>OFF</td><td>OFF</td><td>OFF</td><td>ON</td><td>1</td></tr>
<tr><td>OFF</td><td>OFF</td><td>ON</td><td>OFF</td><td>2</td></tr>
<tr><td>OFF</td><td>OFF</td><td>ON</td><td>ON</td><td>3</td></tr>
<tr><td>OFF</td><td>ON</td><td>OFF</td><td>OFF</td><td>4</td></tr>
<tr><td>OFF</td><td>ON</td><td>OFF</td><td>ON</td><td>5</td></tr>
<tr><td>OFF</td><td>ON</td><td>ON</td><td>OFF</td><td>6</td></tr>
<tr><td>OFF</td><td>ON</td><td>ON</td><td>ON</td><td>7</td></tr>
<tr><td>ON</td><td>OFF</td><td>OFF</td><td>OFF</td><td>8</td></tr>
<tr><td>ON</td><td>OFF</td><td>OFF</td><td>ON</td><td>9</td></tr>
<tr><td>ON</td><td>OFF</td><td>ON</td><td>OFF</td><td>10</td></tr>
<tr><td>ON</td><td>OFF</td><td>ON</td><td>ON</td><td>11</td></tr>
<tr><td>ON</td><td>ON</td><td>OFF</td><td>OFF</td><td>12</td></tr>
<tr><td>ON</td><td>ON</td><td>OFF</td><td>ON</td><td>13</td></tr>
<tr><td>ON</td><td>ON</td><td>ON</td><td>OFF</td><td>14</td></tr>
<tr><td>ON</td><td>ON</td><td>ON</td><td>ON</td><td>15</td></tr>
</table>
BAUD RATE: Those three DIP switches BR2-BR0 define the MODBUS/RTU or ASCII baud rate for the communication:
<table>
<tr><td>BR2</td><td>BR1</td><td>BR0</td><td>MODBUS/RTU or ASCII Baudrate</td></tr>
<tr><td>OFF</td><td>OFF</td><td>OFF</td><td>4800bd</td></tr>
<tr><td>OFF</td><td>OFF</td><td>ON</td><td>9600bd</td></tr>
<tr><td>OFF</td><td>ON</td><td>OFF</td><td>19200bd</td></tr>
<tr><td>OFF</td><td>ON</td><td>ON</td><td>38400bd</td></tr>
<tr><td>ON</td><td>OFF</td><td>OFF</td><td>57600bd</td></tr>
<tr><td>ON</td><td>OFF</td><td>ON</td><td>115200bd</td></tr>
<tr><td>ON</td><td>ON</td><td>OFF</td><td>230400bd</td></tr>
<tr><td>ON</td><td>ON</td><td>ON</td><td>256000bd</td></tr>
</table>
PARITY: This DIP switch PARITY defines the MODBUS/RTU or ASCII parity for the communication:
<table>
<tr><td>PARITY</td><td>MODBUS/RTU or ASCII parity</td></tr>
<tr><td>OFF</td><td>NONE</td></tr>
<tr><td>ON</td><td>EVEN</td></tr>
</table>
HINT: After changing on of the DIP switches, the module restarts completely and reinitializes the serial interface. You will notice that the WHITE LED will be on for approximately 2 seconds, before this LED will flash with a one second cycle.</td></tr>
<tr><td>LED WHITE</td><td>This LED will flash with a cycle of 1 seconds to show normal mode of the module</td></tr>
<tr><td>LED GREEN</td><td>This LED will flash shortly, whenever the module receives a valid telegram on the serial interface.</td></tr>
<tr><td>LED RED</td><td>If this LED flashes cyclically, there is a module error detected by the firmware</td></tr>
</table>

Table: Description of the DIP switch functions and the indication LEDs on the IO module

9.6 Dimensions of the module

In the below drawing you will find the dimensions of the IO module.

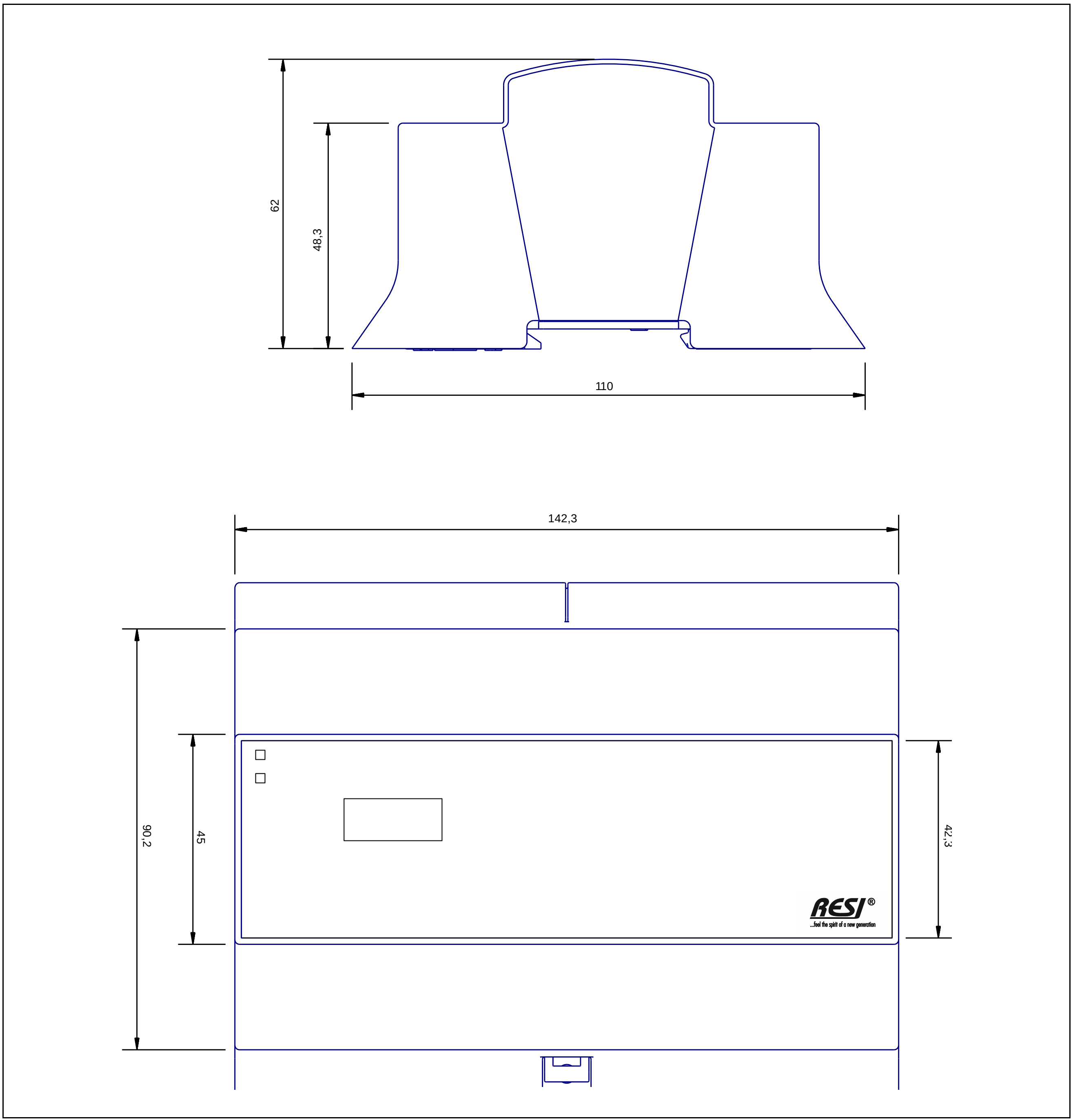


Illustration: Dimensions of the IO module in mm

Dimensions	
Dimensions of the housing L x B x H (mm)	143 x 110 x 62
Weight	260 g
Color	Grey, RAL7035
Material	Self-extinguish PC/ABS, DIN 43880
Protection class	IP20 based on DIN 40050/EN 60529

Table: technical data of the housing

9.8 Power supply of the module

In the below drawing you will find how to connect the module to a power supply.

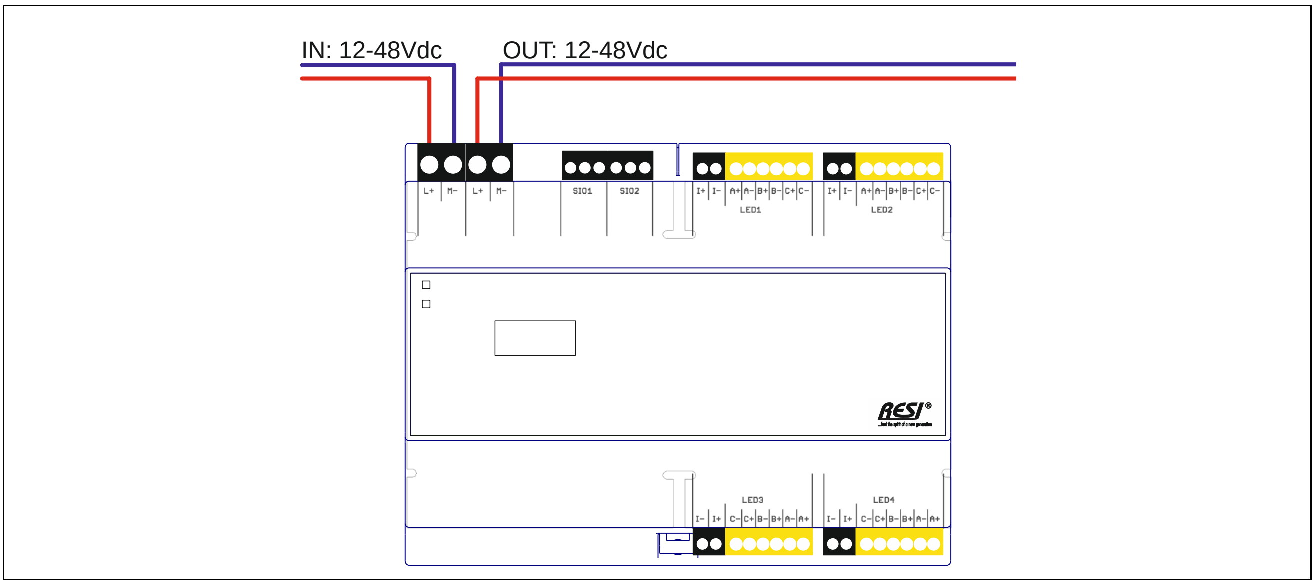


Illustration: Power supply of the IO module

The module offers two 2-pin plug-in terminals for connecting the power supply to the module. It is designed to create a daisy chain power supply with many modules.

9.9 Serial RS485 connection

In this drawing you see the cabling of the serial RS485 bus line. In the module both SIO terminal block are bridged.

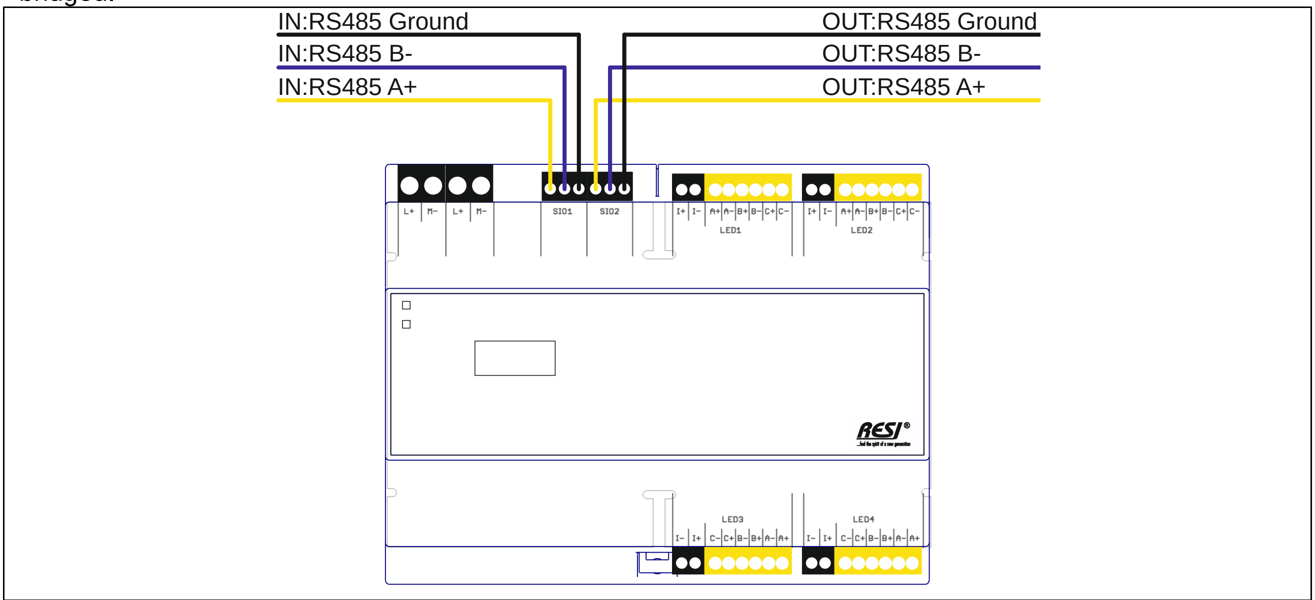


Illustration: RS485 bus cabling of the IO module

The module offers two plug-in 3-pin terminals to connect a RS485 bus line to the module. It was designed to create a daisy chain bus line with many modules. Don’t forget, that a RS485 bus line needs a line termination at the end of both lines!

9.10 Connection of LED stripes

IN the below drawings we will show the cabling of various types of LED stripes to our module. We will discuss how to connect different types of LED stripes to our module in detail.

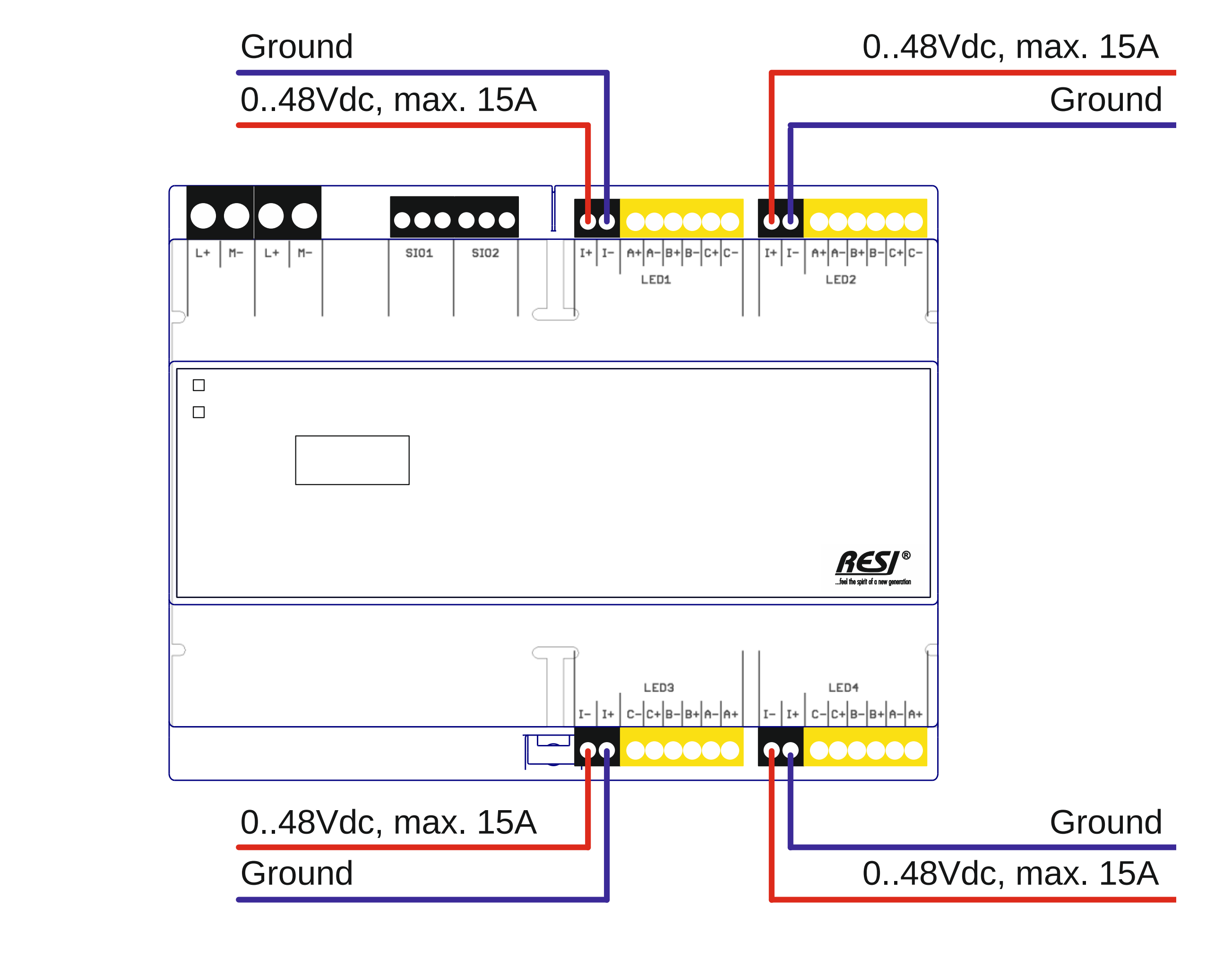


Illustration: Power supply for the LED stripes

The module offers 4 independed LED groups LED1, LED2 LED3 and LED4. Each of the LED groups offers three individual dimmable outputs A, B and C.

For the LED stripes of each of the four LED groups you can use individual power supplies or one for all four LED groups together. It depends on the power consumption and the voltage level of the used LED stripes. Therefore each LED group offers two terminals I+ and I-. Depending on the type of LED stripe, you can use different types of power supplies. It is very important, that the maximum current, which the power supply delivers, must not exceed 15A! Due to this the following limitations arise for powering the LED stripes:

- LED stripes for 12Vdc voltage: 12Vdc*15A -> max. 180W power supply
- LED stripes for 24Vdc voltage: 24Vdc*15A -> max. 360W power supply
- LED stripes for 48Vdc voltage: 48Vdc*15A -> max. 720W power supply

But be careful! Each dimmable PWM output can only drive max. 5A current for dimming!

IMPORTANT: Each of the four LED groups can have its own power supply. They can have different voltage levels for each LED group e.g. 12vdc on LED group LED1 and 24Vdc on LED group LED2. But due to the design of the module all four ground terminals (I-) are bridged internally!

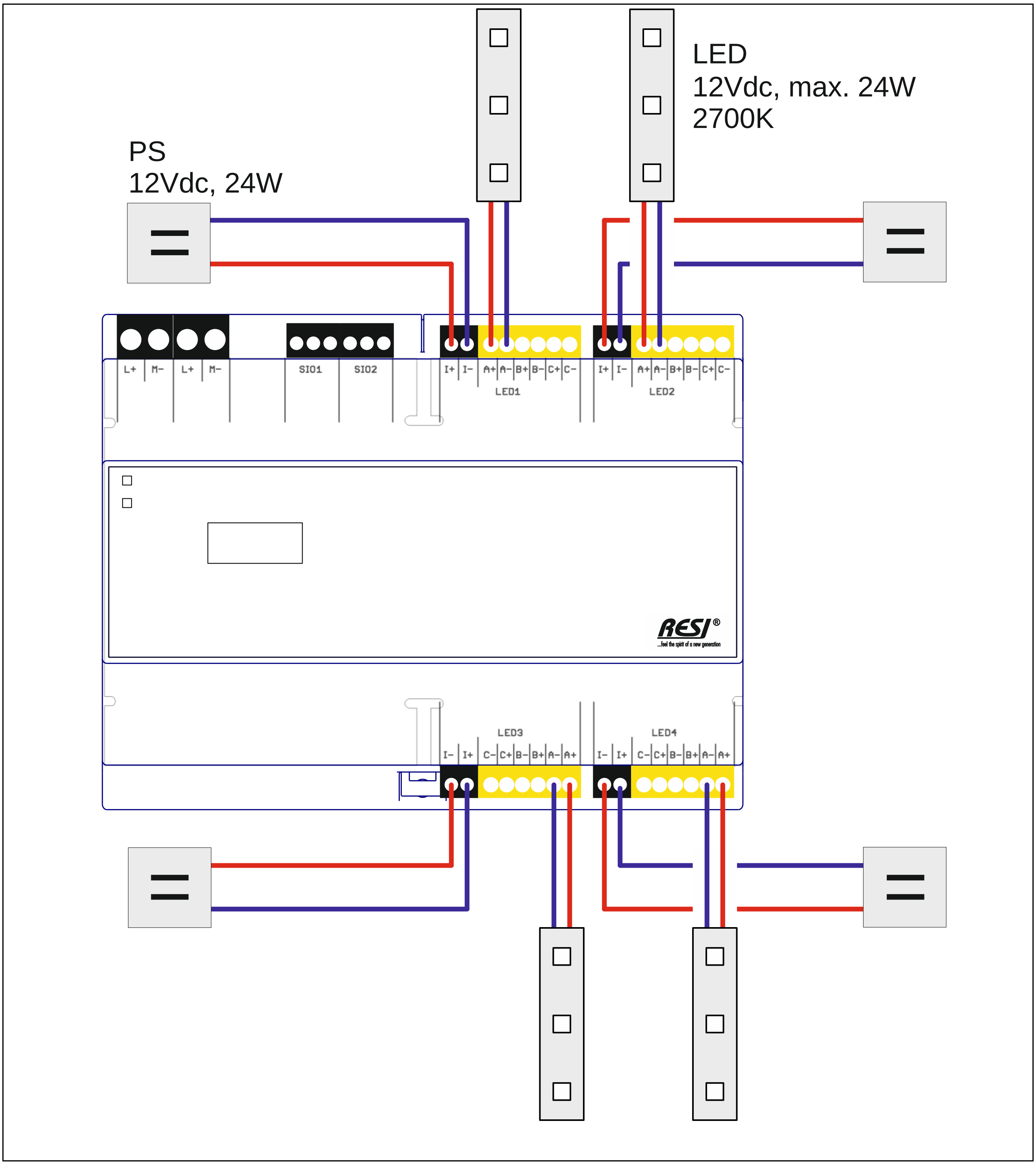


Illustration: Cabling of one 12Vdc LED stripe with 24W power consumption, luminous color 2700K, per LED group. Due to the reason, that the LED stripe consumes only 24W, we use also a 24W power supply. This results in an input current for the clamps I+ and I- of 2A (This is far below 15A and therefore ok). The output current flow over the terminal A is 2A (<5A, again this is ok).

IMPORTANT: Dot forget, that all external power supplies are connected internally with the terminal I- of all four LED groups! Please also remember, that all anodes with the terminals I+, A+, B+ and C+ are tied together in one LED group!

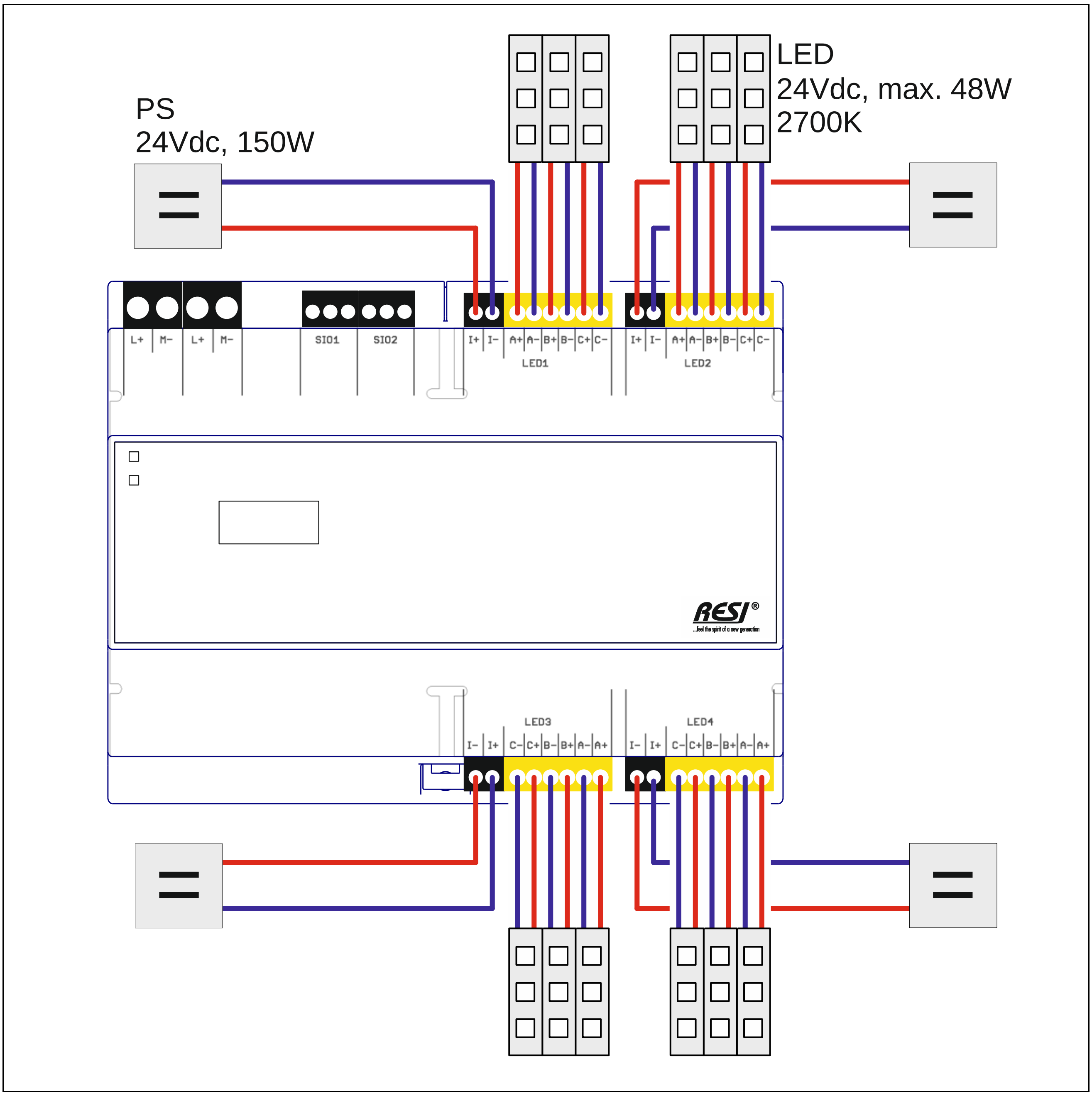


Illustration: Cabling of three 24Vdc LED stripes with 48W power consumption of each LED stripe per LED group. Each of the three LED stripes of a LED group can be dimmed individually. This example uses all three outputs A, B and C of a LED group. Each of the three LED stripes consumes a maximum of 48W of power. We use a power supply with 3x48W -> 150W. The input current, which flows over the terminals I+ and I- is now max. 6.25A. That's far beyond the limit of 15A and ok. Due to the fact, that on each output we have only connected a 48W LED stripe, the current flow over each of the three outputs A, B and C is max. 2A. That's again is beyond the limit of 5A and ok.

IMPORTANT: Dot forget, that all external power supplies are connected internally with the terminal I- of all four LED groups! Please also remember, that all anodes with the terminals I+, A+, B+ and C+ are tied together in one LED group!

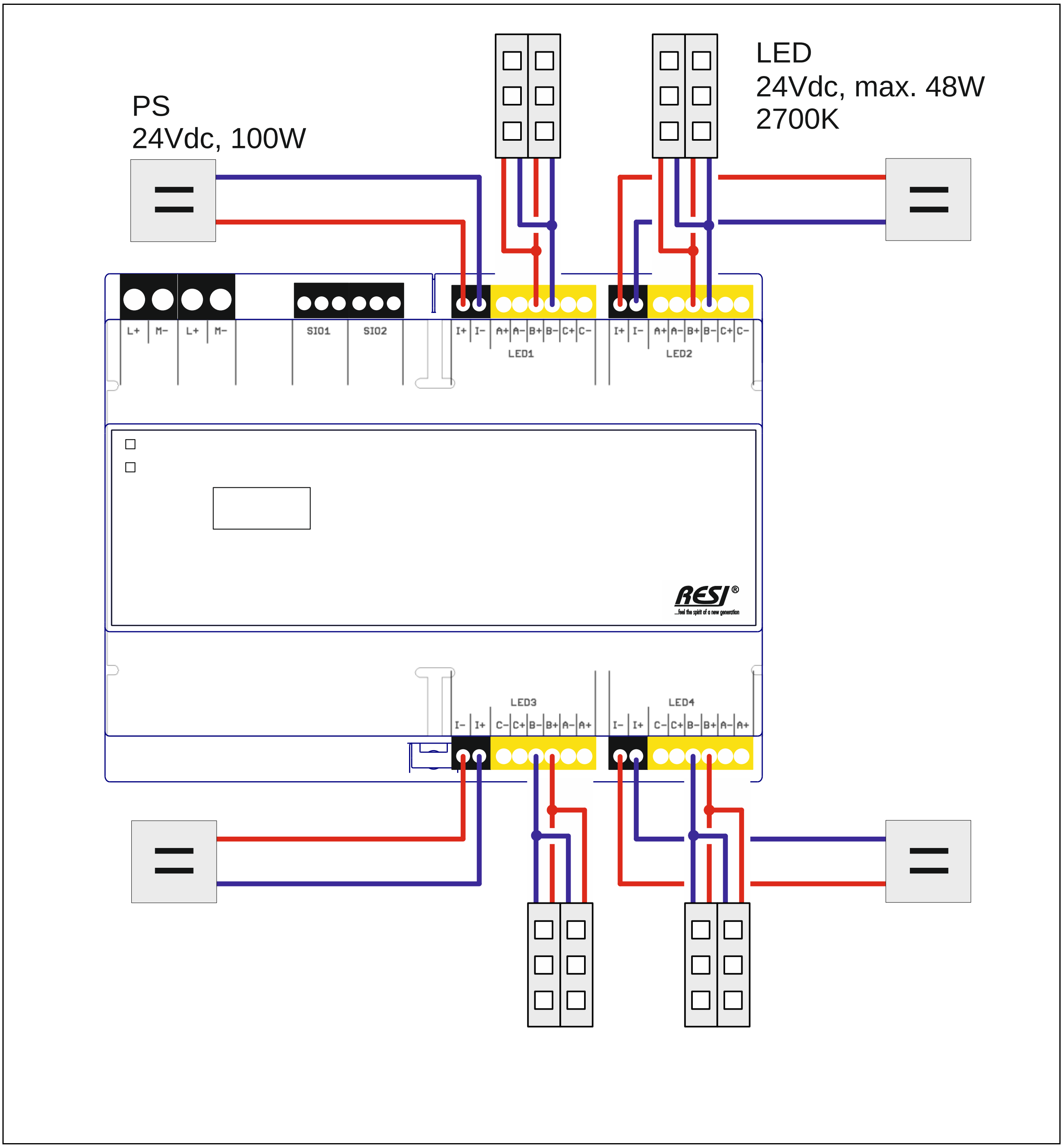


Illustration: Cabling of two 24Vdc LED stripes with 48W power consumption of each LED stripe on the output B per LED group. Both LED stripes are dimmed as a group with output B of the LED group. We use a 100W power supply on the input I+ and I-. This results in an input current of 4.17A, which is again below the maximum current of 15A and ok. Now we have connected two LED stripes to one output. This results in a total power consumption of 96W on the output B. This leads to an output current of 4A. That's again lower than 5A and ok.

IMPORTANT: Dot forget, that all external power supplies are connected internally with the terminal I- of all four LED groups! Please also remember, that all anodes with the terminals I+, A+, B+ and C+ are tied together in one LED group!

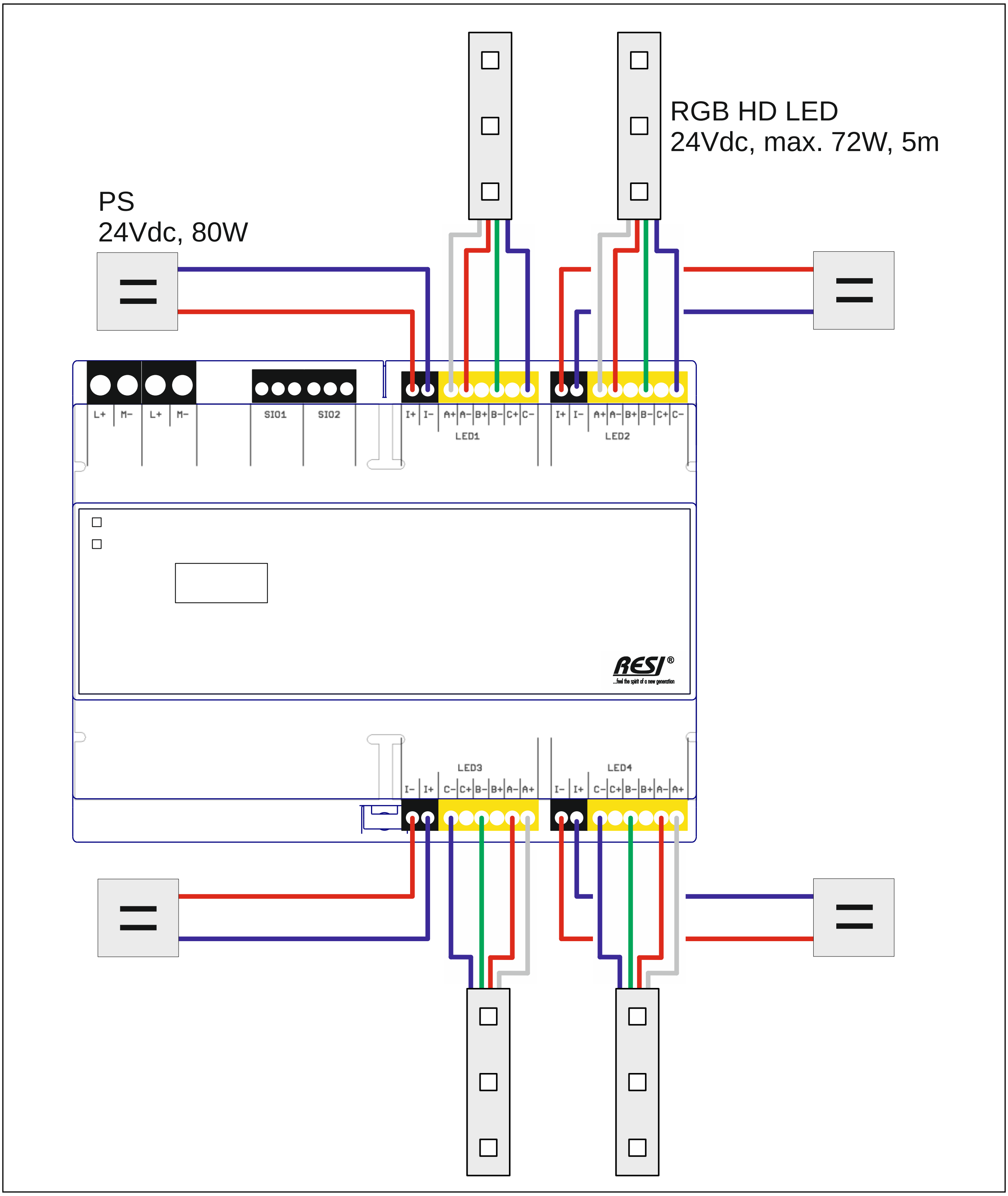


Illustration: In this example we use a RGB LED HD stripe per LED group. This stripe offers three dimmable channels, one for the color red, one for green and one for blue. The common anode of the LED stripe is connected to the terminal A+ of our module. The 80W power supply delivers a maximum current of 3,34A. This is far beyond the allowed 15A and ok. Each output channel must drive only 1/3rd of the 72W total power consumption of the LED stripe. This equals to 24W, resulting in a maximum current for each output of 1A. Again this is far beyond the allowed 5A for each channel and ok.

IMPORTANT: Dot forget, that all external power supplies are connected internally with the terminal I- of all four LED groups! Please also remember, that all anodes with the terminals I+, A+, B+ and C+ are tied together in one LED group!

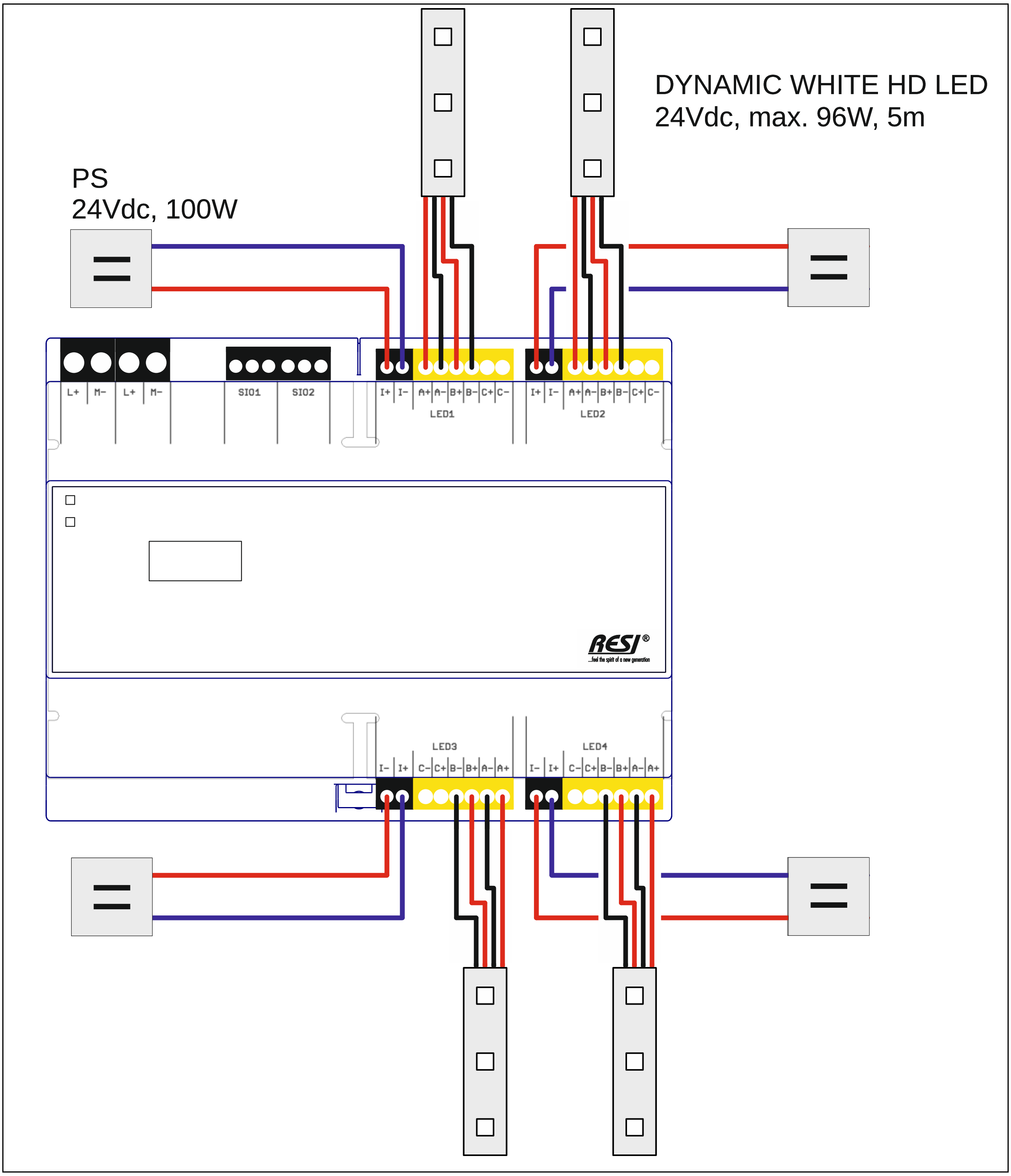


Illustration: Cabling of a dynamic white LED stripe per LED group. This type of LED stripe combines two different LED types with different luminant colors into one product. Each of the two build-in LED stripes consumes 48W of power and you can mix up different luminant colors mostly from warm white to cold white. Connect the four cables of the LED stripe a shown in the above drawing. We also connect the both anodes of the dual LED stripe to the clamps A+ and B+. The cable for warm white is connected to the output A- and the cable for the cold white LEDs is connected to the output B+. The output clamps C+ and C- stay unconnected. Due to the fact, that the outputs A and B have to drive only 48W each, the maximum output current per channel is 2A. This is far beyond the allowed 5A and ok. The input current on the clamps I+ and I- lies by maximum 4.16A with the 100W power supply. This again is far under the allowed 15A and ok.

IMPORTANT: Dot forget, that all external power supplies are connected internally with the terminal I- of all four LED groups! Please also remember, that all anodes with the terminals I+, A+, B+ and C+ are tied together in one LED group!

9.11 Assignment of the channel numbers to the output clamps

Here you can find a definition, how the channel numbers are mapped to the output terminals.

LED Group	Clamp	Group Number	Channel Number
LED1	A+ A-	1	1
LED1	B+ B-	1	2
LED1	C+ C-	1	3
LED2	A+ A-	2	4
LED2	B+ B-	2	5
LED2	C+ C-	2	6
LED3	A+ A-	3	7
LED3	B+ B-	3	8
LED3	C+ C-	3	9
LED4	A+ A-	4	10
LED4	B+ B-	4	11
LED4	C+ C-	4	12

9.12 Functional description

This IO module communicates with a host system with the MODBUS/RTU slave protocol. All versions of the Io module with RESI-xxx-ASCII offer an additional protocol: An ASCII slave protocol with simple text string. The communication runs over a RS485 interface (half duplex) or over a RS232 interface (full duplex).

For the communication with ASCII texts, the host sends ASCII messages with a special start character # (0x23, 35dec) and a special end character (0x0d,13dec or CARRIAGE RETURN) to the module. The module uses also this special start and end characters to answer to the host request. Consult the below noted detailed command descriptions. In the ASCII protocol mode, the host can send messages with or without a bus number.

For communication with the MODBUS/RTU slave, the module offers the following MODBUS functions:

- 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)

HINT:
The functions READ HOLDING REGISTER and PRESET MULTIPLE REGISTERS are restricted to max. 125 register per request! The functions READ INPUT STATUS, READ COIL STATUS and FORCE MULTIPLE COILS are restricted to 2000 coils or inputs (bits) per request.

9.13 ASCII protocol description

9.13.1 Overview

The IO module communicates with simple ASCII commands. The following special characters are used in this documentation:

stand for the **Hashtag** ASCII character 35dec or 0x23
: stand for the **colon** ASCII character 58dec or 0x3A
= stand for the **equal sign** with the ASCII code 61ec or 0x3D
- stand for the **minus sign** with the ASCII code 45dec or 0x2D
, stands for the **comma** with the ASCII code 44dec or 0x2C
<CR> stand for **CARRIAGE RETURN** ASCII character 13dec or 0x0D. In the following text we use the representation CR.
<SP> stands for the **SPACE**. This is the space character with the ASCII code 32dec or 0x20. In the following text we will use the representation☐.
<ADR> will be used as the current **bus address** of the module. The bus number can be transmitted decimal or hexadecimal and is separated with a comma (ASCII code 44dec or 0x2C) from the rest of the command. Hexadecimal numbers always start with 0x. Its only allowed to use the ASCII characters ‚0‘-‚9‘ 48dec to 57dec, 0x30-0x39 and ‚A‘ to ‚F‘, 65dec to 70dec, 0x41-0x46. All modules react to the broadcast bus address 0 and to its own bus number. With a DIP switch, the user can easily change between the internal stored bus number in the FLASH and the fix bus number 255. Consult the DIP switch description for more details.

9.13.2 Communication sequence

In general the IO module sends no ASCII characters without a request from a host. So the host is the master of the communication and the IO module is always answering to host requests as a slave. If only one IO module is connected to a host (e.g. Via RS232), you can dump the bus number in the protocol. If you use a RS485 interface, more than one module can be connected to the host. Therefore a bus number in the request frame of the host is always necessary.

The structure of the command look like this:

The host sends a command or a command with parameters without a bus address to the module:
#<Command><CR> or
#<Command>:<Parameter><CR>

The module answers, if it feels addressed, with the following answer telegram:
#<Answer><CR>

If the bus number is used, the module answers with:
#<ADR>,<Command><CR> or
#<ADR>,<Command>:<Parameter><CR>

The module answers with:
#<ADR>,<Answer><CR>

The bus address lies in the range of 1dec to 255dec or 0x00 to 0xFF hexadecimal. The setup is done with our free configuration software MODBUSConfigurator. Each module reacts also to the broadcast bus address 0.

For each command, we define two different writings. A long version and a short version, to avoid unnecessary traffic on the bus. For example to request the software version of the module you can use the command VERSION or the short command VER.

9.13.3 Request VERSION

This command returns the current software version of the module.

Host long version:
#VERSION<CR> or
#<ADR>,VERSION<CR>

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

Answer:
#VERSION:<HIGH>.<MED>.<LOW><CR> or
#<ADR>,VERSION:<HIGH>,<MED>,<LOW><CR>

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

Samples:
→ **#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$_{CR}$**

9.13.4 Request module TYPE

This command returns the current type of the module.

Host long version:
#TYPE<CR> or
#<ADR>,TYPE<CR>

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

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

<TYP> defines the current type of the module. Currently RESI-4LED-ASCII

Samples:
→ **#TYPE$_{CR}$**
← **#TYPE:RESI-4LED-ASCII$_{CR}$**

→ **#255,TYP$_{CR}$**
← **#255,TYPE:RESI-4LED-ASCII$_{CR}$**

9.13.5 Table of all ASCII commands

Here you will find a possible ASCII commands of the module. We use here only the version with bus number. That you can avoid the bus number, we have discussed earlier in this document. If an argument as the extension Dec, it will be returned as a decimal number, If an argument has the extension Hex, then this argument is returned as a hexadecimal number. Many command returns the argument in decimal and hexadecimal representation. So the host can select, what kind of number conversion, it will handle in its software.

Direction	**ASCII command**
Host	**#<BusAdr>,VER**$_{CR}$ **#<BusAdr>,VERSION**$_{CR}$
Answer	**#<BusAdr>,VERSION:<VersionHi>.< VersionMed>.<VersionLo>**$_{CR}$
	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)
Host	**#<BusAdr>,TYP**$_{CR}$ **#<BusAdr>,TYPE**$_{CR}$
Answer	**#<BusAdr>,TYPE:RESI-4LED-ASCII**$_{CR}$
	Returns the current type of the module
Host	**#<BusAdr>,OWN**$_{CR}$ **#<BusAdr>,OWNER**$_{CR}$
Answer	**#<BusAdr>,OWNER:RESI**$_{CR}$
	Returns the owner of the module
Host	**#<BusAdr>,CRE**$_{CR}$ **#<BusAdr>,CREATOR**$_{CR}$
Answer	**#<BusAdr>,CREATOR:DI HC SIGL,MSC**$_{CR}$
	Returns the creator of the module

Direction	**ASCII command**
Host	**#<BusAdr>,COPY$_{CR}$** **#<BusAdr>,COPYRIGHT$_{CR}$**
Answer	**#<BusAdr>,COPYRIGHT:2015-16 BY RESI AND DI HC SIGL,MSC WWW.RESI.CC$_{CR}$**
	Returns a copyright note about the module
Host	**#<BusAdr>,GDIP$_{CR}$** **#<BusAdr>,GET☐DIP$_{CR}$**
Answer	**#<BusAdr>,GDIP:<DIPSwitchDec>,<DIPSwitchHex>$_{CR}$**
	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 (=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)
Host	**#<BusAdr>,GLOS$_{CR}$** **#<BusAdr>,GET☐LOS$_{CR}$**
Answer	**#<BusAdr>,GLOS:<LO1Dec>,<LO2Dec>, … ,<LO12Dec>,** **<LO1Hex>,<LO2Hex>, … ,<LO12Hex>$_{CR}$**
	Returns the current value of all twelve set point values of the PWM outputs LO1, LO2 .. LO12 as decimal and hexadecimal number LO1Dec LO1Hex The current set point of the dimmable output 1 in the range of 0 to 4095 or 0x000 to 0xFFF LO2Dec LO2Hex The current set point of the dimmable output 2 in the range of 0 to 4095 or 0x000 to 0xFFF … LO12Dec LO12Hex The current set point of the dimmable output 12 in the range of 0 to 4095 or 0x000 to 0xFFF
Host	**#<BusAdr>,GLOx$_{CR}$** **#<BusAdr>,GET☐LOx$_{CR}$**
Answer	**#<BusAdr>,GLOx:<LOxDec>,<LOxHex>$_{CR}$**
x	**1..12**
	Returns the current set point value of the PWM output LOx as decimal and hexadecimal number LOxDec LOxHex The current set point of the dimmable output LOx in the range of 0 to 4095 or 0x000 to 0xFFF
Host	**#<BusAdr>,SLOx:<LOxValue>$_{CR}$** **#<BusAdr>,SET☐LOx:<LOxValue>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
x	**1..12**
	Stores the new value LOxValue into the set point register LOx. LOxValue the new set point value for the register LOx in the range of 0..4095 or 0x000 to 0xFFF

Direction	**ASCII command**
Host	**#<BusAdr>,SMODEx:<MODEx>$_{CR}$** **#<BusAdr>,SET□MODEx:<MODEx>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
x	**1..4**
	Sets the mode for the three LED outputs for one of the four LED groups LED1, LED2, LED3 and LED4 to the new mode MODEx. MODEx The new mode for the LED module =0: OFF: All three outputs of the affected LED group are immediately switched to 0 =1: ON: All three outputs of the affected LED group are dimmed to the values LOx immediately =2: FLASH: All three outputs of the affected LED group flashes in the rhythm of the parameterized minimum and maximum times with the three set point values LOx. =3: FADE: All three outputs of the affected LED group fade with the current speed FADE SPEEDx to the new values LOx. =4: RANDOM: All three outputs of the affected LED group dices a random number for each channel in the range of 0 to LOx. Then the three outputs fade to the new values with the current FADE SPEEDx. After a random pause between the configured minimum and maximum time in seconds, this procedure will be repeated. =5: SEQUENCE: All three outputs of the affected LED group flashes successively with the three set points LOx. The three outputs are on for the time period MIN TIMEx in 1/10s. In between the three outputs are 0 for a time period MAXTIMEx in 1/10s.
Host	**#<BusAdr>,GMODEx$_{CR}$** **#<BusAdr>,GET□MODEx$_{CR}$**
Answer	**#<BusAdr>,GMODEx:<MODExDec>,<MODExHex>$_{CR}$**
x	**1..4**
	Returns the current mode of the affected LED group LED1..LED4. MODExDec MODExHex The current mode of the affected LED group. =0: OFF: All three outputs of the affected LED group are immediately switched to 0 =1: ON: All three outputs of the affected LED group are dimmed to the values LOx immediately =2: FLASH: All three outputs of the affected LED group flashes in the rhythm of the parameterized minimum and maximum times with the three set point values LOx. =3: FADE: All three outputs of the affected LED group fade with the current speed FADE SPEEDx to the new values LOx. =4: RANDOM: All three outputs of the affected LED group dices a random number for each channel in the range of 0 to LOx. Then the three outputs fade to the new values with the current FADE SPEEDx. After a random pause between the configured minimum and maximum time in seconds, this procedure will be repeated. =5: SEQUENCE: All three outputs of the affected LED group flashes successively with the three set points LOx. The three outputs are on for the time period MIN TIMEx in 1/10s. In between the three outputs are 0 for a time period MAXTIMEx in 1/10s.

Direction	**ASCII command**
Host	**#<BusAdr>,SFADEx:<FADEx>$_{CR}$** **#<BusAdr>,SET□FADEx:<FADEx>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
x	**1..4**
	Sets the new fading speed for the fading in the two modes FADE and RANDOM for the affected LED group LED1..LED4 FADE The new speed for fading in steps per 1/100s.
Host	**#<BusAdr>,GFADEx$_{CR}$** **#<BusAdr>,GET□FADEx$_{CR}$**
Answer	**#<BusAdr>,GFADEx:<FADExDec>,<FADExHex>$_{CR}$**
x	**1..4**
	Returns the current fade speed of the LED module in steps per 1/100s for the affected LED group LED1..LED4. FADEDec FADEHex The current fade speed in steps per 1/100s.
Host	**#<BusAdr>,SMINTx:<MINTIMEx>$_{CR}$** **#<BusAdr>,SET□MIN□TIMEx:<MINTIMEx>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
x	**1..4**
	Sets the new minimum time for the LED group LED1..LED4. This time is used in the three modes FLASH, RANDOM and SEQUENCE. MINTIMEx The new value for the minimum time. In the modes FLASH and SEQUENCE, this time defines the ON time span of the three outputs with the three values LOx. The OFF time span with the three values 0 is defined with the MAXTIMEx parameter. The parameter specifies a time in 1/10s. In the mode RANDOM this time defines the minimum time span between two random value changes. The parameter specifies a time span in seconds.
Host	**#<BusAdr>,GMINTx$_{CR}$** **#<BusAdr>,GET□MIN□TIMEx$_{CR}$**
Answer	**#<BusAdr>,GMINTx:<MINTIMExDec>,<MINTIMExHex>$_{CR}$**
x	**1..4**
	Returns the current defined minimum time of the affected LED group LED1..LED4. MINTIMExDec MINTIMExHex The current value for the minimum time. In the modes FLASH and SEQUENCE in 1/10s In the mode RANDOM in seconds.

Direction	**ASCII command**
Host	**#<BusAdr>,SMAXTx:<MAXTIMEx>$_{CR}$** **#<BusAdr>,SET☐MAX☐TIMEx:<MAXTIMEx>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
x	**1..4**
	Sets the new maximum time for the LED group LED1..LED4. This time is used in the three modes FLASH, RANDOM and SEQUENCE. MAXTIMEx The new value for the maximum time. In the modes FLASH and SEQUENCE, this time defines the OFF time span of the three outputs with the three values LOx. The ON time span with the three values LOx is defined with the MAXTIMEx parameter. The parameter specifies a time in 1/10s. In the mode RANDOM this time defines the minimum time span between two random value changes. The parameter specifies a time span in seconds.
Host	**#<BusAdr>,GMAXTx$_{CR}$** **#<BusAdr>,GET☐MAX☐TIMEx$_{CR}$**
Answer	**#<BusAdr>,GMAXTx:<MAXTIMExDec>,<MAXTIMExHex>$_{CR}$**
x	**1..4**
	Returns the current defined maximum time for the affected LED group LED1..LED4. MAXTIMEDec MAXTIMEHex The current value for the maximum time. In the modes FLASH and SEQUENCE in 1/10s In the mode RANDOM in seconds.
Host	**#<BusAdr>,STIMESx:<MINTIMEx>,<MAXTIMEx>$_{CR}$** **#<BusAdr>,SET☐TIMESx:<MINTIMEx>,<MAXTIMEx>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
x	**1..4**
	Sets the new minimum and maximum times for the LED group LED1..LED4 for the three modes FLASH, RANDOM and SEQUENCE. MINTIMEx The new value for the minimum time. In the modes FLASH and SEQUENCE, this time defines the ON time span of the three outputs with the three values LOx. The OFF time span with the three values 0 is defined with the MAXTIMEx parameter. The parameter specifies a time in 1/10s. In the mode RANDOM this time defines the minimum time span between two random value changes. The parameter specifies a time span in seconds.. MAXTIMEx The new value for the maximum time. In the modes FLASH and SEQUENCE, this time defines the OFF time span of the three outputs with the three values LOx. The ON time span with the three values LOx is defined with the MAXTIMEx parameter. The parameter specifies a time in 1/10s. In the mode RANDOM this time defines the minimum time span between two random value changes. The parameter specifies a time span in seconds.
Host	**#<BusAdr>,GTIMESx$_{CR}$** **#<BusAdr>,GET☐TIMESx$_{CR}$**
Answer	**#<BusAdr>,GTIMES:<MINTIMExDec>,<MAXTIMExDec>,** **<MINTIMExHex>,<MAXTIMExHex>$_{CR}$**
x	**1..4**
	Returns the current minimum and maximum times of the LED group LED1..LED4 . MINTIMEDec MINTIMEHex The current value for the minimum time. In the modes FLASH and SEQUENCE in 1/10s. In the mode RANDOM in seconds. MAXTIMEDec MAXTIMEHex The current value for the maximum time. In the modes FLASH and SEQUENCE in 1/10s. In the mode RANDOM in seconds.

Direction	**ASCII command**
Host	**#<BusAdr>,SALLx:<MODEx>,<LOxA>,<LOxB>,<LOxC>, <MINTIMEx>,<MAXTIMEx>,<FADEx>$_{CR}$ #<BusAdr>,SET☐ALLx:<MODEx>,<LOxA>,<LOxB>,<LOxC>, <MINTIMEx>,<MAXTIMEx>,<FADEx>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
x	**1..4**
	Sets all values for one LED group LED1..LED4 with one command.

MODEx	The new mode for the affected LED group =0: OFF: All three outputs of the affected LED group are immediately switched to 0 =1: ON: All three outputs of the affected LED group are dimmed to the values LOx immediately =2: FLASH: All three outputs of the affected LED group flashes in the rhythm of the parameterized minimum and maximum times with the three set point values LOx. =3: FADE: All three outputs of the affected LED group fade with the current speed FADE SPEEDx to the new values LOx. =4: RANDOM: All three outputs of the affected LED group dices a random number for each channel in the range of 0 to LOx. Then the three outputs fade to the new values with the current FADE SPEEDx. After a random pause between the configured minimum and maximum time in seconds, this procedure will be repeated. =5: SEQUENCE: All three outputs of the affected LED group flashes successively with the three set points LOx. The three outputs are on for the time period MIN TIMEx in 1/10s. In between the three outputs are 0 for a time period MAXTIMEx in 1/10s.
LOxA	The new value of the PWM output LOx A in the range of 0..4095 or 0x000 to 0xFFF
LOxB	The new value of the PWM output LOx B in the range of 0..4095 or 0x000 to 0xFFF
LOxC	The new value of the PWM output LOx C in the range of 0..4095 or 0x000 to 0xFFF
MINTIMEx	The new value for the minimum time. In the modes FLASH and SEQUENCE, this time defines the ON time span of the three outputs with the three values LOx. The OFF time span with the three values 0 is defined with the MAXTIMEx parameter. The parameter specifies a time in 1/10s. In the mode RANDOM this time defines the minimum time span between two random value changes. The parameter specifies a time span in seconds.
MAXTIMEx	The new value for the maximum time. In the modes FLASH and SEQUENCE, this time defines the OFF time span of the three outputs with the three values LOx. The ON time span with the three values LOx is defined with the MAXTIMEx parameter. The parameter specifies a time in 1/10s. In the mode RANDOM this time defines the minimum time span between two random value changes. The parameter specifies a time span in seconds.
FADEx	The new speed for fading in steps per 1/100s.

Direction	ASCII command
Host	**#<BusAdr>,GALLx$_{CR}$** **#<BusAdr>,GET□ALLx$_{CR}$**
Answer	**#<BusAdr>,GALL:<MODExDec>,<LOxADec>,<LOxBDec>,<LOxCDec>,<MINTIMExDec>,<MAXTIMExDec>,<FADExDec>,<CLOxADec>,<CLOxBDec>,<CLOxCDec>,<RLOxADec>,<RLOxBDec>,<RLOxCDec>,<MODExHex>,<LOxAHex>,<LOxBHex>,<LOxCHex>,<MINTIMExHex>,<MAXTIMExHex>,<FADExHex>,<CLOxAHex>,<CLOxBHex>,<CLOxCHex>,<RLOxAHex>,<RLOxBHex>,<RLOxCHex>$_{CR}$**
x	**1..4**
	Returns the current values of one LED group LED1..LED4 in one answer. MODExDec MODExHex The new mode for the affected LED group. See MODEx description ion command GALL LOxADec LOxAHex The new value of the PWM output LOx A in the range of 0..4095 or 0x000 to 0xFFF LOxBDec LOxHex The new value of the PWM output LOx B in the range of 0..4095 or 0x000 to 0xFFF LOxCDec LOxCHex The new value of the PWM output LOx C in the range of 0..4095 or 0x000 to 0xFFF MINTIMExDec MINTIMExHex The new value for the minimum time. In the modes FLASH and SEQUENCE, this time defines the ON time span of the three outputs with the three values LOx. The OFF time span with the three values 0 is defined with the MAXTIMEx parameter. The parameter specifies a time in 1/10s. In the mode RANDOM this time defines the minimum time span between two random value changes. The parameter specifies a time span in seconds. MAXTIMExDec MAXTIMExHex The new value for the maximum time. In the modes FLASH and SEQUENCE, this time defines the OFF time span of the three outputs with the three values LOx. The ON time span with the three values LOx is defined with the MAXTIMEx parameter. The parameter specifies a time in 1/10s. In the mode RANDOM this time defines the minimum time span between two random value changes. The parameter specifies a time span in seconds. FADExDec FADExHex The new speed for fading in steps per 1/100s. CLOxADec CLOxAHex The current output value of the output A in the range of 0 to 4095 or 0x000 to 0xFFF including dimming and mode information. CLOxBDec CLOxBHex The current output value of the output A in the range of 0 to 4095 or 0x000 to 0xFFF including dimming and mode information. CLOxCDec CLOxCHex The current output value of the output A in the range of 0 to 4095 or 0x000 to 0xFFF including dimming and mode information. RLOxADec RLOxAHex The last diced value in mode RANDOM for output A in the range of 0 to 4095 or 0x000 to 0xFFF. RLOxBDec RLOxBHex The last diced value in mode RANDOM for output B in the range of 0 to 4095 or 0x000 to 0xFFF. RLOxCDec RLOxCHex The last diced value in mode RANDOM for output C in the range of 0 to 4095 or 0x000 to 0xFFF.

Direction	ASCII command
Host	**#<BusAdr>,GCLOS$_{CR}$** **#<BusAdr>,GET□CURRENT□LOS$_{CR}$**
Answer	**#<BusAdr>,GCLOS: <CLO1Dec>,<CLO2Dec>, … ,<CLO12CDec>, <CLO1Hex>,<CLO2Hex>, .. ,<CLO12Hex>$_{CR}$**
	Returns all current values of all twelve outputs O1, O2 up to O12 on the IO module. CLO1Dec CLO1Hex The real value of the output O1 in the range of 0 to 4095 or 0x000 to 0xFFF, including all fading and all modes. CLO2Dec CLO2Hex The real value of the output O2 in the range of 0 to 4095 or 0x000 to 0xFFF, including all fading and all modes. … CLO12Dec CLO12Hex The real value of the output O12 in the range of 0 to 4095 or 0x000 to 0xFFF, including all fading and all modes.
Host	**#<BusAdr>,GRLOS$_{CR}$** **#<BusAdr>,GET□RANDOM□LOS$_{CR}$**
Answer	**#<BusAdr>,GRLOS: <RLO1Dec>,<RLO2Dec>, … ,<RLO12Dec>, <RLO1Hex>,<RLO2Hex>, … ,<RLO12Hex>$_{CR}$**
	Returns the last diced values for the twelve outputs O1, O2 up to O12 in mode RANDOM. RLO1Dec RLO1Hex The last diced number in mode RANDOM for output 1 in the range of 0 to 4095 or 0x000 to0xFFF. RLO2Dec RLO2Hex The last diced number in mode RANDOM for output 2 in the range of 0 to 4095 or 0x000 to0xFFF. … RLO12Dec RLO12Hex The last diced number in mode RANDOM for output 12 in the range of 0 to 4095 or 0x000 to0xFFF.
Host	**#<BusAdr>,GCLOx$_{CR}$** **#<BusAdr>,GET□CURRENT□LOx$_{CR}$**
Answer	**#<BusAdr>,GCLOx:<CLOxDec>,<CLOxHex>$_{CR}$**
x	**1..12**
	Returns the current value of the channel Ox. CLOxDec CLOxHex The real value of the output Ox in the range of 0 to 4095 or 0x000 to 0xFFF, including all fading and all modes.
Host	**#<BusAdr>,GRLOx$_{CR}$** **#<BusAdr>,GET□RANDOM□LOx$_{CR}$**
Answer	**#<BusAdr>,GRLOx:<RLOxDec>,<RLOxHex>$_{CR}$**
x	**1..12**
	Returns the last diced values for the output Ox in mode RANDOM. RLOxDec RLOxHex The last diced number in mode RANDOM for output Ox in the range of 0 to 4095 or 0x000 to0xFFF.

Direction	ASCII command
Host	**#<BusAdr>,SMBADR:<MBUnit>$_{CR}$** **#<BusAdr>,SET□MODBUS□ADDRESS:<MBUnit>$_{CR}$**
Answer	**#<BusAdr>,OK $_{CR}$**
	Writes the unit address into the FLASH memory of the module. The new unit address for MODBUS/RTU or ASCII mode is only used immediately, if the DIP switch setting of the bus address is 0. Otherwise the unit address is defined by the DIP settings. The unit address ranges from 0dec to 255dec.
Host	**#<BusAdr>,GMBADR$_{CR}$** **#<BusAdr>,GET□MODBUS□ADDRESS$_{CR}$**
Answer	**#<BusAdr>,GMBADR:<MBUnitDec>,<MBFLASHDec>,<MBUnitHex>,<MBFLASHHex>$_{CR}$**
	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. MBUnitDec MBUnitHex — The current used MODBUS/RTU unit or ASCII address for communication MBFLASHDec MBFLASHHex — The internal stored MODBUS/RTU unit address or ASCII address from the FLASH memory, if the DIP switch is 0.
Host	**#<BusAdr>,RST$_{CR}$** **#<BusAdr>,RESET$_{CR}$**
Answer	**none**
	Executes a software reset (Reboot) of the module.

9.14 MODBUS – register description

9.14.1 Table of inputs and coils

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 0x00001 to 0x65536. Inputs are usually noted with 1x00001 to 1x65536. 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: 0x00100 for the coil 100, 1x00100 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.

Register	Description
0x00001 1x00001 I:0 R/O LED1_ISFADING	Is currently a fading in the LED group LED1 active =0:no fading is active, =1:Fading is active
0x00002 1x00002 I:1 R/O LED2_ISFADING	Is currently a fading in the LED group LED2 active =0:no fading is active, =1:Fading is active
0x00003 1x00003 I:2 R/O LED3_ISFADING	Is currently a fading in the LED group LED3 active =0:no fading is active, =1:Fading is active
0x00004 1x00004 I:3 R/O LED4_ISFADING	Is currently a fading in the LED group LED4 active =0:no fading is active, =1:Fading is active

Register	Description
0x02001 1x02001 I:2000 R/O DIP1	Current state of DIP switch 1 =0:Dip switch is OFF, =1: Dip switch is ON
0x02002 1x02002 I:2001 R/O DIP2	Current state of DIP switch 2 =0:Dip switch is OFF, =1: Dip switch is ON
0x02003 1x02003 I:2002 R/O DIP3	Current state of DIP switch 3 =0:Dip switch is OFF, =1: Dip switch is ON
0x02004 1x02004 I:2003 R/O DIP4	Current state of DIP switch 4 =0:Dip switch is OFF, =1: Dip switch is ON
0x02005 1x02005 I:2004 R/O DIP5	Current state of DIP switch 5 =0:Dip switch is OFF, =1: Dip switch is ON
0x02006 1x02006 I:2005 R/O DIP6	Current state of DIP switch 6 =0:Dip switch is OFF, =1: Dip switch is ON
0x02007 1x02007 I:2006 R/O DIP7	Current state of DIP switch 7 =0:Dip switch is OFF, =1: Dip switch is ON
0x02008 1x02008 I:2007 R/O DIP8	Current state of DIP switch 8 =0:Dip switch is OFF, =1: Dip switch is ON

9.14.2 Table of holding/input registers

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.

Register	Description
4x00001 3x00001 I:0 R/W LO1	Current set point of LED group LED1 PWM output A. 0..4095 or 0x000...0xFFF for 0% to 100% brightness If you write onto this register, you define a new set point value for the PWM output
4x00002 3x00002 I:1 R/W LO2	Current set point of LED group LED1 PWM output B. 0..4095 or 0x000...0xFFF for 0% to 100% brightness If you write onto this register, you define a new set point value for the PWM output
4x00003 3x00003 I:2 R/W LO3	Current set point of LED group LED1 PWM output C. 0..4095 or 0x000...0xFFF for 0% to 100% brightness If you write onto this register, you define a new set point value for the PWM output
4x00004 3x00004 I:3 R/W LO4	Current set point of LED group LED2 PWM output A. 0..4095 or 0x000...0xFFF for 0% to 100% brightness If you write onto this register, you define a new set point value for the PWM output
4x00005 3x00005 I:4 R/W LO5	Current set point of LED group LED2 PWM output B. 0..4095 or 0x000...0xFFF for 0% to 100% brightness If you write onto this register, you define a new set point value for the PWM output
4x00006 3x00006 I:5 R/W LO6	Current set point of LED group LED2 PWM output C. 0..4095 or 0x000...0xFFF for 0% to 100% brightness If you write onto this register, you define a new set point value for the PWM output

Register	Description
4x00007 3x00007 I:6 R/W LO7	Current set point of LED group LED3 PWM output A. 0..4095 or 0x000...0xFFF for 0% to 100% brightness If you write onto this register, you define a new set point value for the PWM output
4x00008 3x00008 I:7 R/W LO8	Current set point of LED group LED3 PWM output B. 0..4095 or 0x000...0xFFF for 0% to 100% brightness If you write onto this register, you define a new set point value for the PWM output
4x00009 3x00009 I:8 R/W LO9	Current set point of LED group LED3 PWM output C. 0..4095 or 0x000...0xFFF for 0% to 100% brightness If you write onto this register, you define a new set point value for the PWM output
4x00010 3x00010 I:9 R/W LO10	Current set point of LED group LED4 PWM output A. 0..4095 or 0x000...0xFFF for 0% to 100% brightness If you write onto this register, you define a new set point value for the PWM output
4x00011 3x00011 I:10 R/W LO11	Current set point of LED group LED4 PWM output B. 0..4095 or 0x000...0xFFF for 0% to 100% brightness If you write onto this register, you define a new set point value for the PWM output
4x00012 3x00012 I:11 R/W LO12	Current set point of LED group LED4 PWM output C. 0..4095 or 0x000...0xFFF for 0% to 100% brightness If you write onto this register, you define a new set point value for the PWM output

Register	Description
4x00013 3x00013 I:12 R/W MODE1	Current mode of LED group LED1 Current mode of LED group LED1 =0:OFF: It doesn't matter which values are stored in the set point registers for the outputs A, B and C, the real output values of the LED group are always 0. =1:ON: The module outputs immediately the values of the three set point registers LOx A, LOx B and LOx C to the three PWM outputs. =2:FLASH: Fort he defined time span MINTIMEx in 1/10s alle three outputs of the LED group are set to the set point values LOx A, LOx B and LOx C. Then the thre outputs are set to 0 for the timespan defined in the register MAXTIMEx in 1/10s. This cycle is repeated endlessly. =3: FADE: if you write a new value into one of the three output set point registers LOx A, LOx B or LOx C, the system increments / decrements every 1/100s the current output values with the value FADESPEEDx to the new setpoint values, until the output registers CLOx A=LOx A and CLOx B=LOx B and CLOx C=LOx C. . The fadespeed is defined in steps per 1/100s. =4:RANDOM: In this mode the module dices after a random time span in the range of MINTIMEx and MAXTIMEx in seconds three new random values for the registers RLOx A, RLOx B and RLOx C. This rules are used: New diced value for RLOx is between 0 and LOx Then the module fades with the defined FADESPEEDx to the new dimming values RLOx. =5: SEQUENCE: Every channel flashes sequentially for MINTIMEx 1/10s. In between the module switches off all three outputs for MAXTIMEx in 1/10s. If you write onto this register, you define a new mode for this LED group.
4x00014 3x00014 I:13 R/W MODE2	Current mode of LED group LED2 See mode description in register MODE1 If you write onto this register, you define a new mode for this LED group.
4x00015 3x00015 I:14 R/W MODE3	Current mode of LED group LED3 See mode description in register MODE1 If you write onto this register, you define a new mode for this LED group.
4x00016 3x00016 I:15 R/W MODE4	Current mode of LED group LED4 See mode description in register MODE1 If you write onto this register, you define a new mode for this LED group.

Register	Description
4x00017 3x00017 I:16 R/W FADESPEED1	Current dimming or fading speed for the outputs in mode FADE and RANDOM in steps per 1/100s. The smallest value is 1. Every 1/100s the system add/subtracts this FADESPEED1 value from the three outputs CLO1, CLO2 and CLO3. So the value 1 in FADESPEED1 means, that if CLO1 hast the start value 0 and the new value LO1 is 4095, the fade up process will last for 40.95 seconds. This is the slowest fading speed of the module. A value of 4095 or more defines the fastest fade speed. After 1/100s the new value will be valid. If you write on this register, you will redefine the FADESPEED1
4x00018 3x00018 I:17 R/W FADESPEED2	Current dimming or fading speed for the outputs in mode FADE and RANDOM in steps per 1/100s. The smallest value is 1. Every 1/100s the system add/subtracts this FADESPEED2 value from the three outputs CLO4, CLO5 and CLO6. So the value 1 in FADESPEED2 means, that if CLO4 hast the start value 0 and the new value LO4 is 4095, the fade up process will last for 40.95 seconds. This is the slowest fading speed of the module. A value of 4095 or more defines the fastest fade speed. After 1/100s the new value will be valid. If you write on this register, you will redefine the FADESPEED2
4x00019 3x00019 I:18 R/W FADESPEED3	Current dimming or fading speed for the outputs in mode FADE and RANDOM in steps per 1/100s. The smallest value is 1. Every 1/100s the system add/subtracts this FADESPEED3 value from the three outputs CLO7, CLO8 and CLO9. So the value 1 in FADESPEED3 means, that if CLO7 hast the start value 0 and the new value LO7 is 4095, the fade up process will last for 40.95 seconds. This is the slowest fading speed of the module. A value of 4095 or more defines the fastest fade speed. After 1/100s the new value will be valid. If you write on this register, you will redefine the FADESPEED3
4x00020 3x00020 I:19 R/W FADESPEED4	Current dimming or fading speed for the outputs in mode FADE and RANDOM in steps per 1/100s. The smallest value is 1. Every 1/100s the system add/subtracts this FADESPEED4 value from the three outputs CLO10, CLO11 and CLO12. So the value 1 in FADESPEED4 means, that if CLO10 hast the start value 0 and the new value LO10 is 4095, the fade up process will last for 40.95 seconds. This is the slowest fading speed of the module. A value of 4095 or more defines the fastest fade speed. After 1/100s the new value will be valid. If you write on this register, you will redefine the FADESPEED4
4x00021 3x00021 I:20 R/W MINTIME1	For the mode FLASH and SEQUENCE this value means the ON time of the flashing or sequence cycle in 1/10s. For the mode RANDOM this value defines the minimum time span for a new dicing cycle. The dicing cycle dices new random values for the registers RLO1, RLO2 and RLO3 and fades to this new brightness values. The time span is defined in seconds. If you write to this register, you will redefine this value.
4x00022 3x00022 I:21 R/W MINTIME2	For the mode FLASH and SEQUENCE this value means the ON time of the flashing or sequence cycle in 1/10s. For the mode RANDOM this value defines the minimum time span for a new dicing cycle. The dicing cycle dices new random values for the registers RLO4, RLO5 and RLO6 and fades to this new brightness values. The time span is defined in seconds. If you write to this register, you will redefine this value.
4x00023 3x00023 I:22 R/W MINTIME3	For the mode FLASH and SEQUENCE this value means the ON time of the flashing or sequence cycle in 1/10s. For the mode RANDOM this value defines the minimum time span for a new dicing cycle. The dicing cycle dices new random values for the registers RLO7, RLO8 and RLO9 and fades to this new brightness values. The time span is defined in seconds. If you write to this register, you will redefine this value.

Register	Description
4x00024 3x00024 I:23 R/W MINTIME4	For the mode FLASH and SEQUENCE this value means the ON time of the flashing or sequence cycle in 1/10s. For the mode RANDOM this value defines the minimum time span for a new dicing cycle. The dicing cycle dices new random values for the registers RLO10, RLO11 and RLO12 and fades to this new brightness values. The time span is defined in seconds. If you write to this register, you will redefine this value.
4x00025 3x00025 I:24 R/W MAXTIME1	For the mode FLASH and SEQUENCE this value means the OFF time of the flashing or sequence cycle in 1/10s. For the mode RANDOM this value defines the maximum time span for a new dicing cycle. The dicing cycle dices new random values for the registers RLO1, RLO2 and RLO3 and fades to this new brightness values. The time span is defined in seconds. If you write to this register, you will redefine this value.
4x00026 3x00026 I:25 R/W MAXTIME2	For the mode FLASH and SEQUENCE this value means the OFF time of the flashing or sequence cycle in 1/10s. For the mode RANDOM this value defines the maximum time span for a new dicing cycle. The dicing cycle dices new random values for the registers RLO4, RLO5 and RLO6 and fades to this new brightness values. The time span is defined in seconds. If you write to this register, you will redefine this value.
4x00027 3x00027 I:26 R/W MAXTIME3	For the mode FLASH and SEQUENCE this value means the OFF time of the flashing or sequence cycle in 1/10s. For the mode RANDOM this value defines the maximum time span for a new dicing cycle. The dicing cycle dices new random values for the registers RLO7, RLO8 and RLO9 and fades to this new brightness values. The time span is defined in seconds. If you write to this register, you will redefine this value.
4x00028 3x00028 I:27 R/W MAXTIME4	For the mode FLASH and SEQUENCE this value means the OFF time of the flashing or sequence cycle in 1/10s. For the mode RANDOM this value defines the maximum time span for a new dicing cycle. The dicing cycle dices new random values for the registers RLO10, RLO11 and RLO12 and fades to this new brightness values. The time span is defined in seconds. If you write to this register, you will redefine this value.
4x00029 3x00029 I:28 R/O ISFADING1	Is one of the three outputs of the LED group LED1 currently fading from one value to another value =0:no fading is active, =1:fading is running
4x00030 3x00030 I:29 R/O ISFADING2	Is one of the three outputs of the LED group LED2 currently fading from one value to another value =0:no fading is active, =1:fading is running
4x00031 3x00031 I:30 R/O ISFADING3	Is one of the three outputs of the LED group LED3 currently fading from one value to another value =0:no fading is active, =1:fading is running
4x00032 3x00032 I:31 R/O ISFADING4	Is one of the three outputs of the LED group LED4 currently fading from one value to another value =0:no fading is active, =1:fading is running

Register	Description
4x00033 3x00033 I:32 R/O CLO1	The real value on the LED PWM output LED1 A including diming and the current mode. 0..4095 or 0x000...0xFFF for 0% to 100% brightness
4x00034 3x00034 I:33 R/O CLO2	The real value on the LED PWM output LED1 B including diming and the current mode. 0..4095 or 0x000...0xFFF for 0% to 100% brightness
4x00035 3x00035 I:34 R/O CLO3	The real value on the LED PWM output LED1 C including diming and the current mode. 0..4095 or 0x000...0xFFF for 0% to 100% brightness
4x00036 3x00036 I:35 R/O CLO4	The real value on the LED PWM output LED2 A including diming and the current mode. 0..4095 or 0x000...0xFFF for 0% to 100% brightness
4x00037 3x00037 I:36 R/O CLO5	The real value on the LED PWM output LED2 B including diming and the current mode. 0..4095 or 0x000...0xFFF for 0% to 100% brightness
4x00038 3x00038 I:37 R/O CLO6	The real value on the LED PWM output LED2 C including diming and the current mode. 0..4095 or 0x000...0xFFF for 0% to 100% brightness
4x00039 3x00039 I:38 R/O CLO7	The real value on the LED PWM output LED3 A including diming and the current mode. 0..4095 or 0x000...0xFFF for 0% to 100% brightness
4x00040 3x00040 I:39 R/O CLO8	The real value on the LED PWM output LED3 B including diming and the current mode. 0..4095 or 0x000...0xFFF for 0% to 100% brightness
4x00041 3x00041 I:40 R/O CLO9	The real value on the LED PWM output LED3 C including diming and the current mode. 0..4095 or 0x000...0xFFF for 0% to 100% brightness
4x00042 3x00042 I:41 R/O CLO10	The real value on the LED PWM output LED4 A including diming and the current mode. 0..4095 or 0x000...0xFFF for 0% to 100% brightness
4x00043 3x00043 I:42 R/O CLO11	The real value on the LED PWM output LED4 B including diming and the current mode. 0..4095 or 0x000...0xFFF for 0% to 100% brightness
4x00044 3x00044 I:43 R/O CLO12	The real value on the LED PWM output LED4 C including diming and the current mode. 0..4095 or 0x000...0xFFF for 0% to 100% brightness

Register	Description
4x00045 3x00045 I:44 R/O RLO1	The last diced random number for the LED PWM output LED1 A in mode RANDOM. 0..4095 or 0x000...0xFFF for 0% to 100% brightness
4x00046 3x00046 I:45 R/O RLO2	The last diced random number for the LED PWM output LED1 B in mode RANDOM. 0..4095 or 0x000...0xFFF for 0% to 100% brightness
4x00047 3x00047 I:46 R/O RLO3	The last diced random number for the LED PWM output LED1 C in mode RANDOM. 0..4095 or 0x000...0xFFF for 0% to 100% brightness
4x00048 3x00048 I:47 R/O RLO4	The last diced random number for the LED PWM output LED2 A in mode RANDOM. 0..4095 or 0x000...0xFFF for 0% to 100% brightness
4x00049 3x00049 I:48 R/O RLO5	The last diced random number for the LED PWM output LED2 B in mode RANDOM. 0..4095 or 0x000...0xFFF for 0% to 100% brightness
4x00050 3x00050 I:49 R/O RLO6	The last diced random number for the LED PWM output LED2 C in mode RANDOM. 0..4095 or 0x000...0xFFF for 0% to 100% brightness
4x00051 3x00051 I:50 R/O RLO7	The last diced random number for the LED PWM output LED3 A in mode RANDOM. 0..4095 or 0x000...0xFFF for 0% to 100% brightness
4x00052 3x00052 I:51 R/O RLO8	The last diced random number for the LED PWM output LED3 B in mode RANDOM. 0..4095 or 0x000...0xFFF for 0% to 100% brightness
4x00053 3x00053 I:52 R/O RLO9	The last diced random number for the LED PWM output LED3 C in mode RANDOM. 0..4095 or 0x000...0xFFF for 0% to 100% brightness
4x00054 3x00054 I:53 R/O RLO10	The last diced random number for the LED PWM output LED4 A in mode RANDOM. 0..4095 or 0x000...0xFFF for 0% to 100% brightness
4x00055 3x00055 I:54 R/O RLO11	The last diced random number for the LED PWM output LED4 B in mode RANDOM. 0..4095 or 0x000...0xFFF for 0% to 100% brightness
4x00056 3x00056 I:55 R/O RLO12	The last diced random number for the LED PWM output LED4 C in mode RANDOM. 0..4095 or 0x000...0xFFF for 0% to 100% brightness

Register	Description
4x02001 3x02001 I:2000 R/O DIP SWITCH	Current state of the DIP switch 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)
4x06001 3x06001 I:6000 W/O RESET SYSTEM	If you write to this register the module does a software restart (soft reset)
4x65222 3x65222 I:65221 R/W MODBUS UNIT ADDRESS	If you read this register, you will receive the current defined MODBUS/RTU unit or ASCII address of the internal FLASH memory. If you write to this register, you can define a new MODBUS/RTU unit address or ASCII address for the internal FLASH memory. This address is only valid after a system reboot or soft reset and if the DIP switches for the bus address are in position 0.

10 RESI-16DI8RO-MODBUS, RESI-16DI8RO-ASCII

10.1 Product description

This IO module offers the following features:

- 16 digital inputs for 16-48Vdc signals
- 8 bistable relay outputs with special power relays
- Maximum switching power: max. 250Vac, max. 16A
- 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
- 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
- RESI-16DI8RO-MODBUS: MODBUS/RTU slave protocol
- RESI-16DI8RO-ASCII: MODBUS/RTU slave and text based ASCII protocol
- Power supply 12-48Vdc
- DIP switch for setting the baud rates, the type of interface and the bus number
- LED indicator for the communication
- Mounting onto EN50022 DIN rail or wall mounting

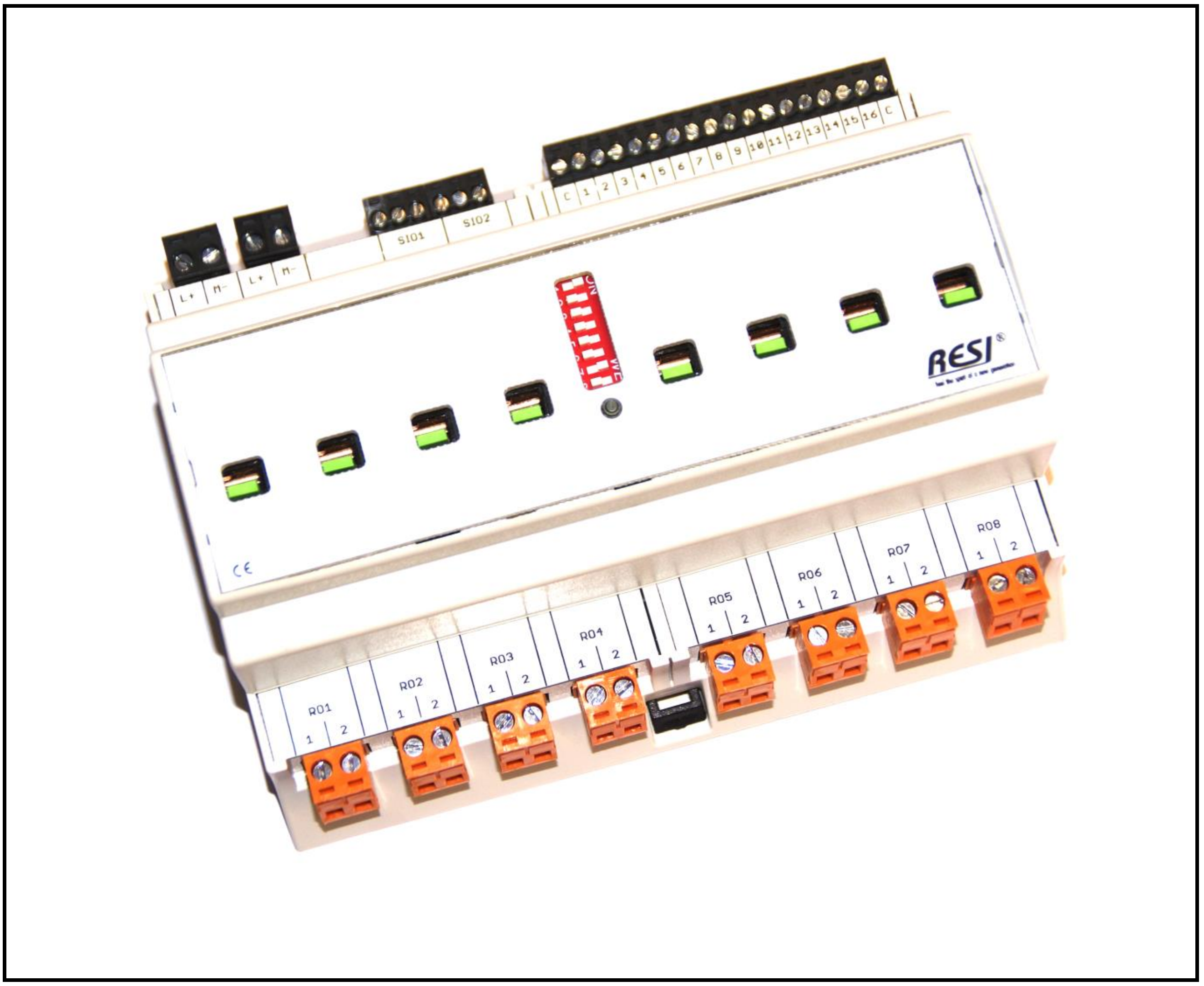


Illustration: Our IO module

10.1.1 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.

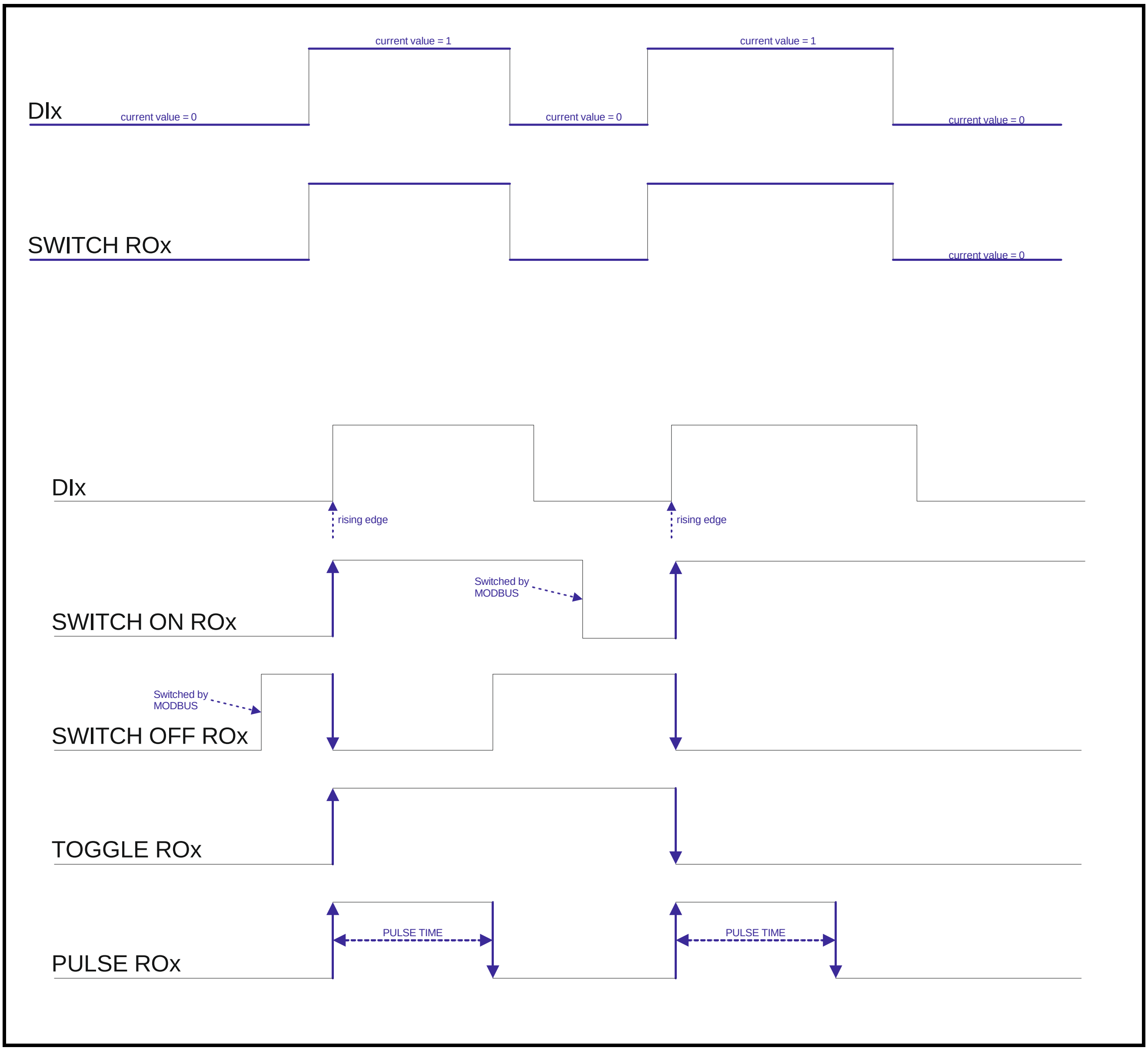


Illustration: Internal logic functions

10.1.1.1 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.

10.1.1.2 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.

10.1.1.3 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 MOBUS 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 MOBUS 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.

10.1.1.4 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 MOBUS 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 MOBUS 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 pushbutton 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

10.1.1.5 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 MOBUS 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 MOBUS 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.

10.1.1.6 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 MOBUS 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 pushbutton switches to the digital inputs DI1 and DI2 and press one of them, the current status of the relay output RO4 is inverted.

10.1.1.7 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 MOBUS 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 pushbutton switches 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.

10.2 Technical data

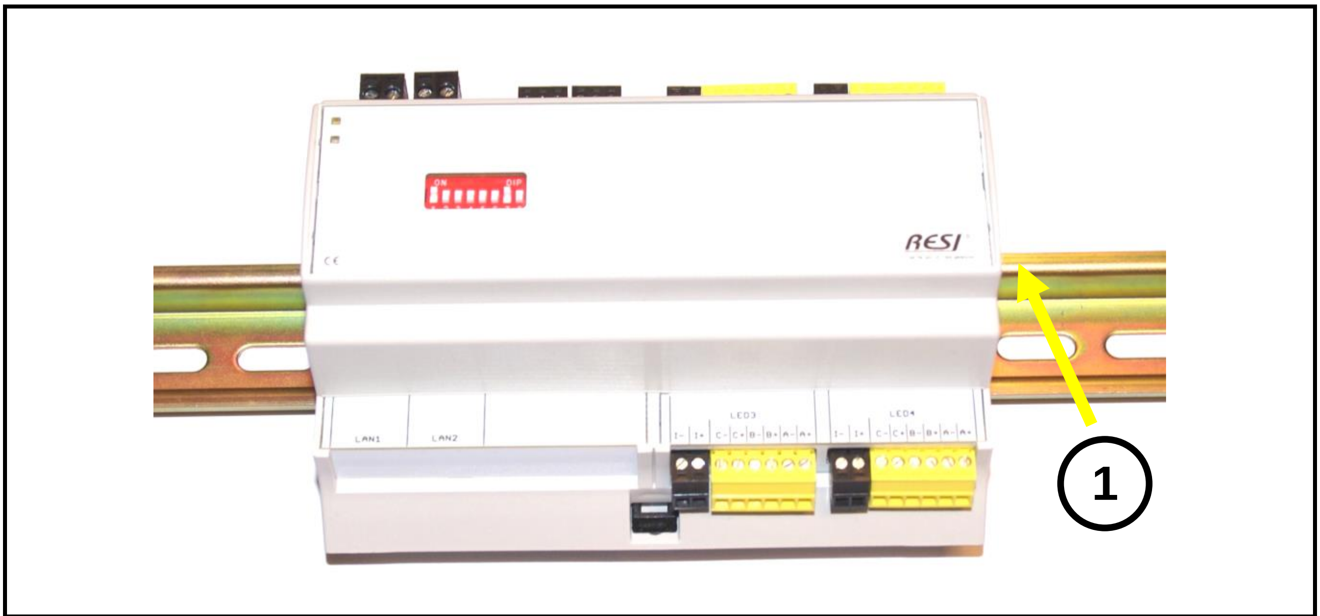
Technical Data			
Power supply			
Supply voltage	12-48V= +/-10%	Storage temperature	-20...85 °C
Power LED	Ja	Operating Temperature	0...60°C
Power consumption	<0.5W	Humidity	25...90 % rH non-condensing
		Protection Class	IP20 (EN 60529)
		Dimensions LxWxH	143mm x 110mm x 62mm
		Weight	560g
		Mounting	On DIN EN50022 rail or wall mounting
ASCII/Modbus Interface		**Relay outputs**	
Protocol	ASCII or Modbus/RTU	Number of outputs	8 bistable relays for socket-outlets and light applications
Type	RS485	Relay type	Bistable with manual operation
Baud rates	4800 to 256000Bd/8/N or E/1	incandescent electric lamp load	Max 4.800 W
Cable Connection	Via removable clamps	Capacitive load	Max. 200µF
LED indicator	Yes	Maximum voltage	250Vac
Galvanic insulation	No	Maximum current	16A
		Mechanical lifetime	10^6 cycles of operation
		Contact material	$AgSnO_2$
		Insulation	Creepage and clearance distance 8mm
		Output power per channel:	
		Incandescent lamp	4.800 W
Digital inputs		Fluorescent lamp	
Total amount of inputs	16	not compensated	5.000 W
Sampling rate	Every 10ms	Fluorescent lamp	
Input voltage range	12-48V= +/-10%	parallel compensated	2.500 W / 200 µF
Input current	approx.. 1mA per channel	Fluorescent lamp	
Logic levels	0: <3V=	duo-combination	2 x 5.000 W
	1: >5V=	Halogen lamp (230VAC)	5.000 W
Cable connection	Via 18-pin plug-in terminal block	Low voltage halogen lamp with transformer	2.000 VA
Galvanic insulation	No	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
		Cable connection	Via 8 2-pin plug-in terminal blocks
		Galvanic insulation	Yes, with the relay
Clamps			
Clamp wire cross section	Max. 1,5 mm²		
Tightening torque	Max. 0.5Nm	**CE conformity**	Yes

10.3 Assembling

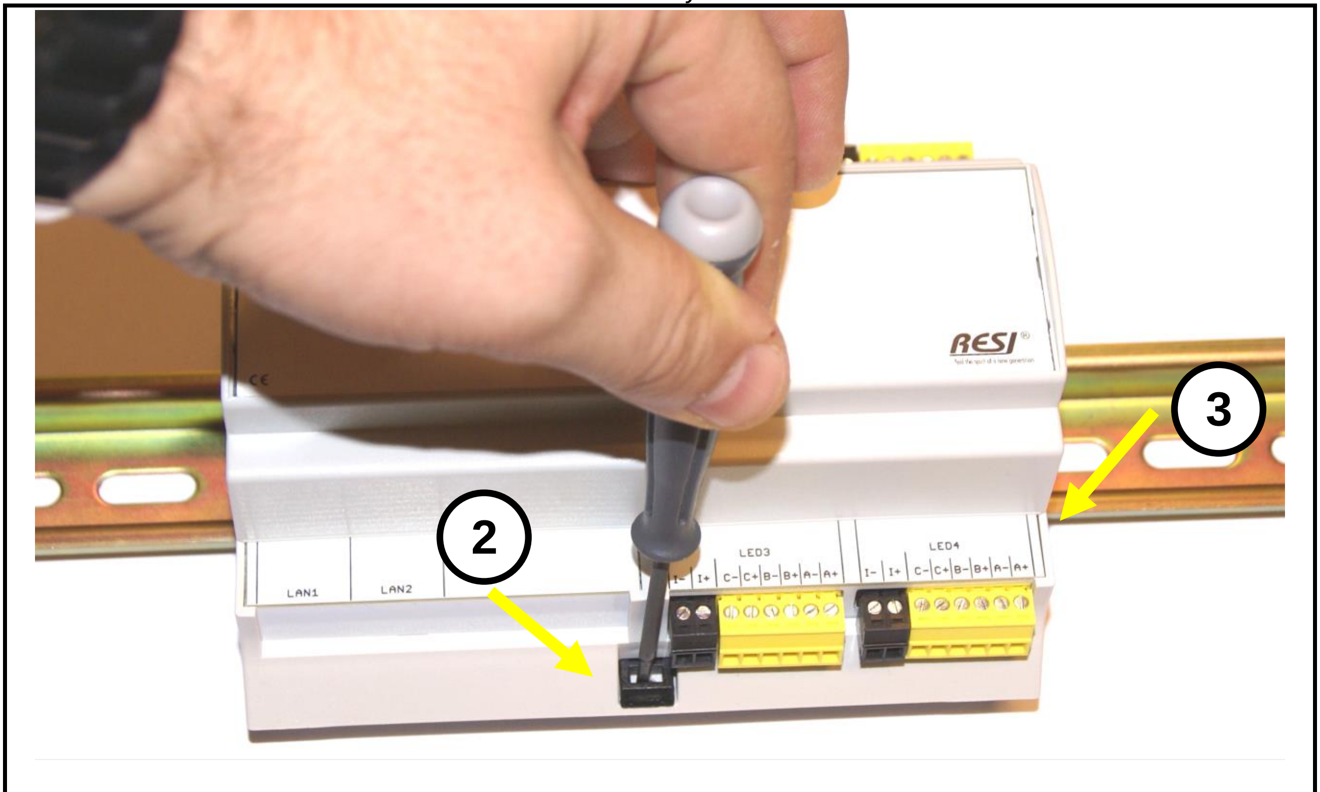
Our IO modules are designed for mounting onto a 35mm DIN-EN50022 rail or for wall mounting. Please not, that in the following mounting description we use only symbolic photos of our IO modules.

10.3.1 Mounting of a DIN EN50022 rail

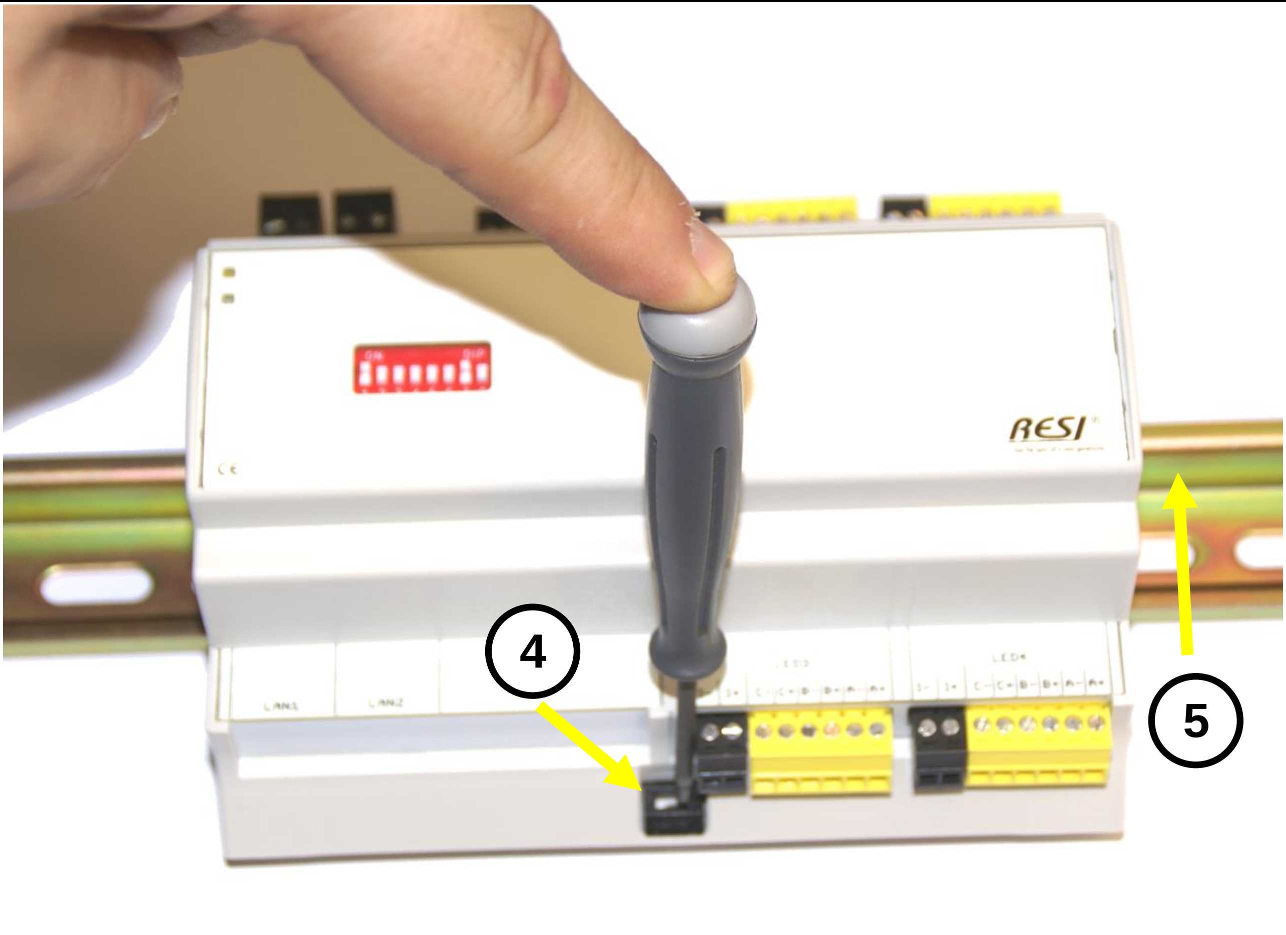
First snap in the top part of the module into the DIN rail (1). The bottom part of the module is not snapped into the DIN rail at this moment.



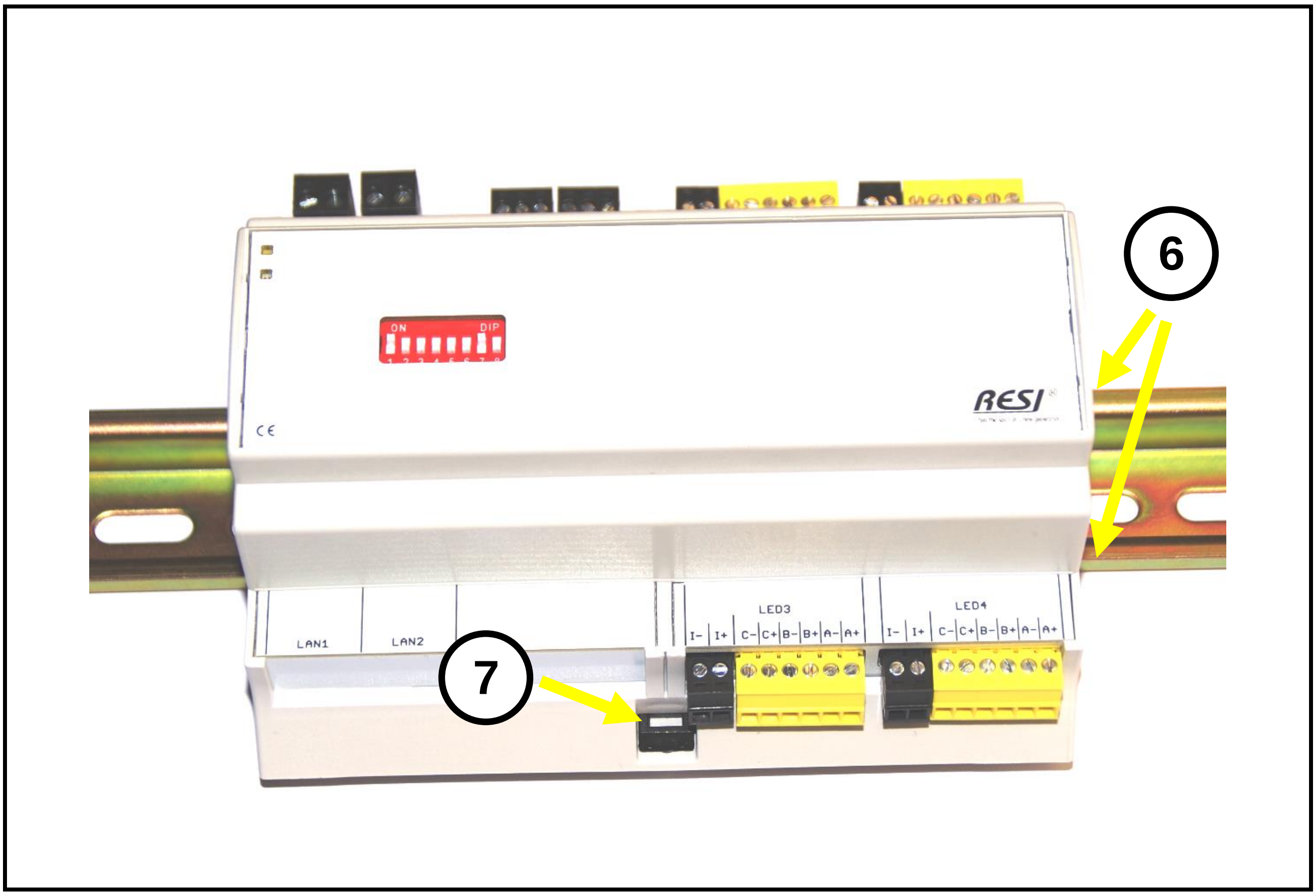
Then open the black hook with a screw driver (2). Now press the module with the opened hook onto the DIN rail until both sides of the module snap into the DIN rail (3). Release the screw driver now. The hook snaps into the DIN rail and the module is now mounted correctly onto the DIN rail.



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.

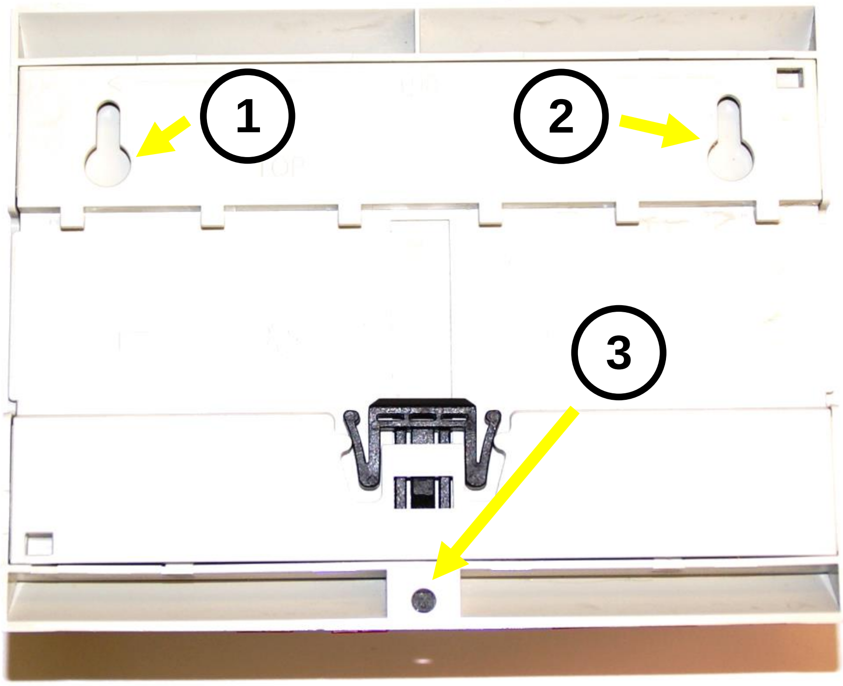


The module is correctly mounted, if the module has snapped into the DIN rail on both sides of the housing (6) and if the hook has snapped in too (7).

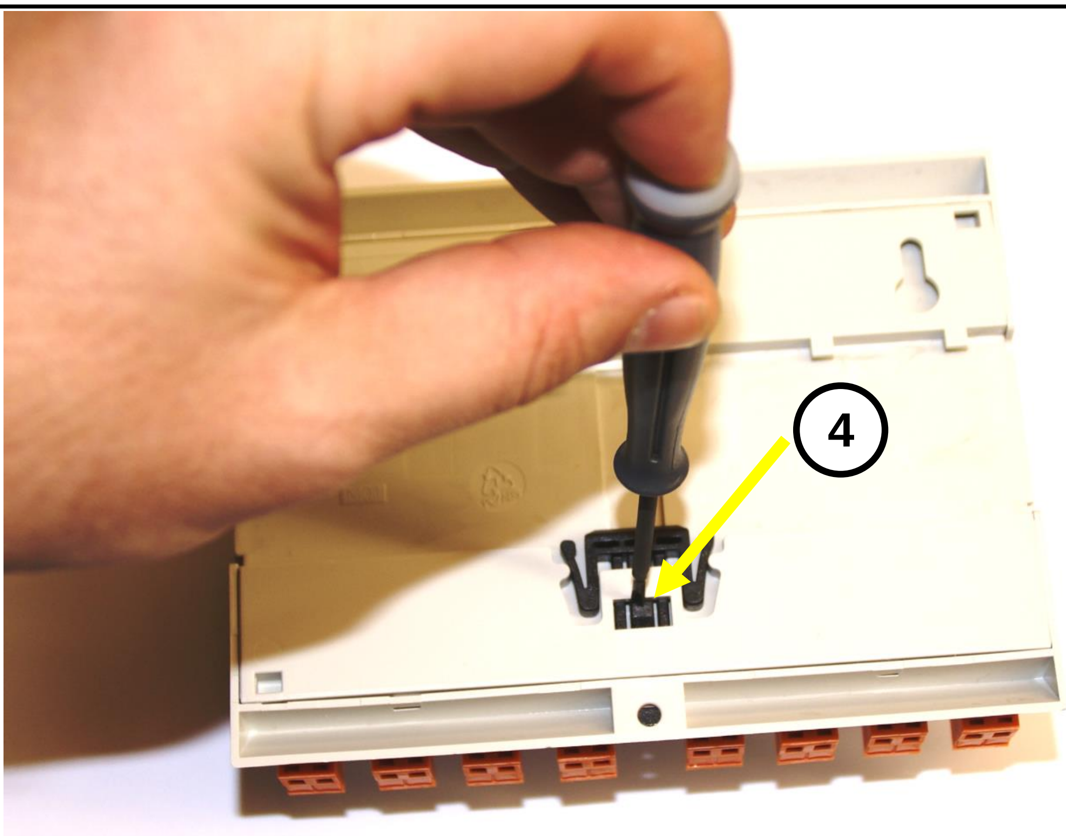


10.3.2 Wall mounting

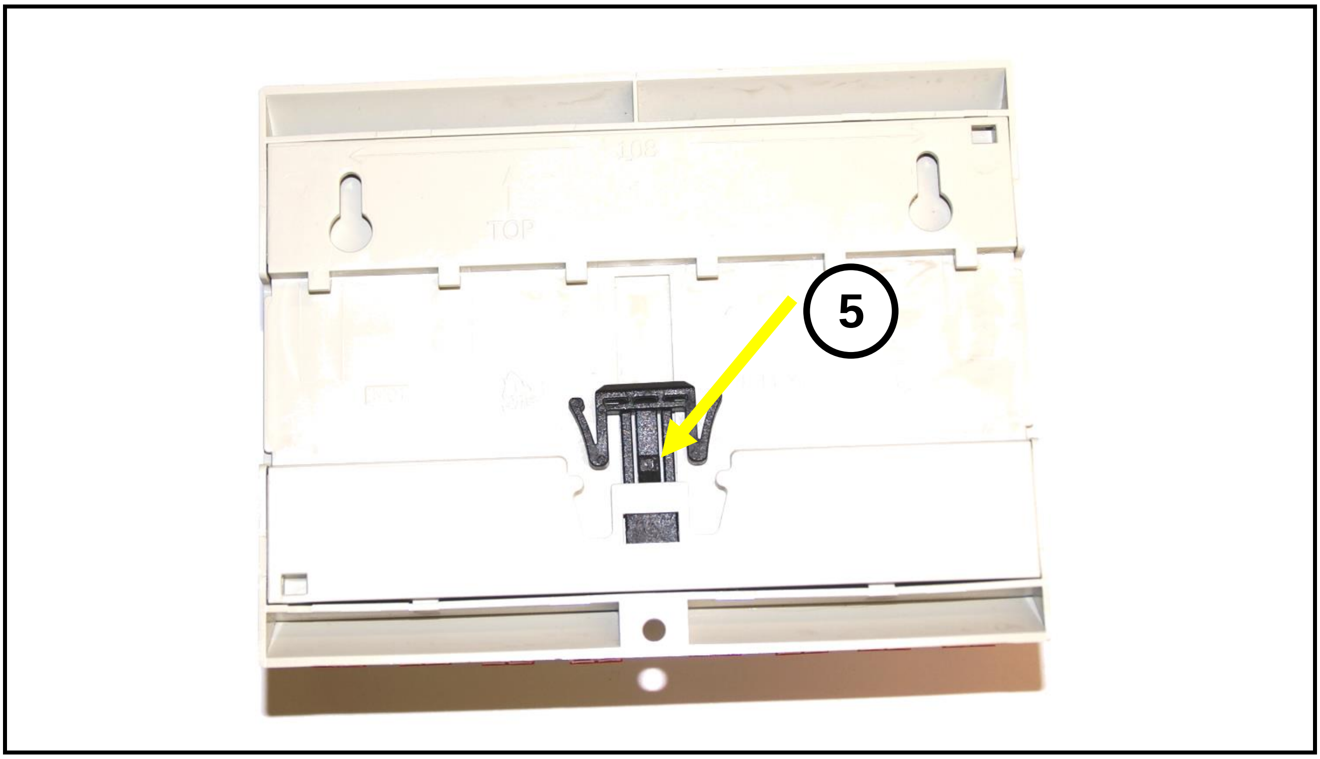
Our modules can also be mounted onto a wall. Turn over the module as shown in the picture below:

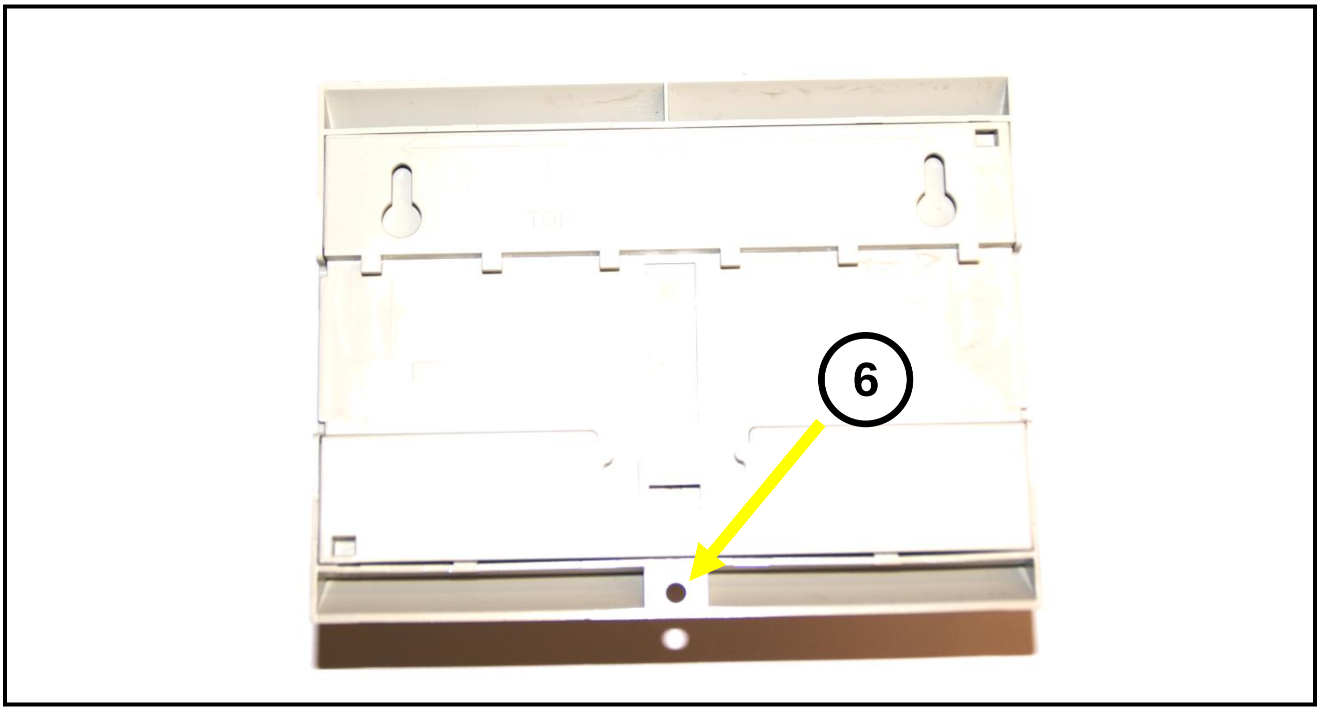


You will notice, that there are two holes for wall hooks or screws on the top side of the housing. (1) and (2). On the bottom side you will notice a small hole for a screw to fix the housing on the wall from the front (3). But first we have to remove the hook, which blocks the screw hole in the housing.



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.

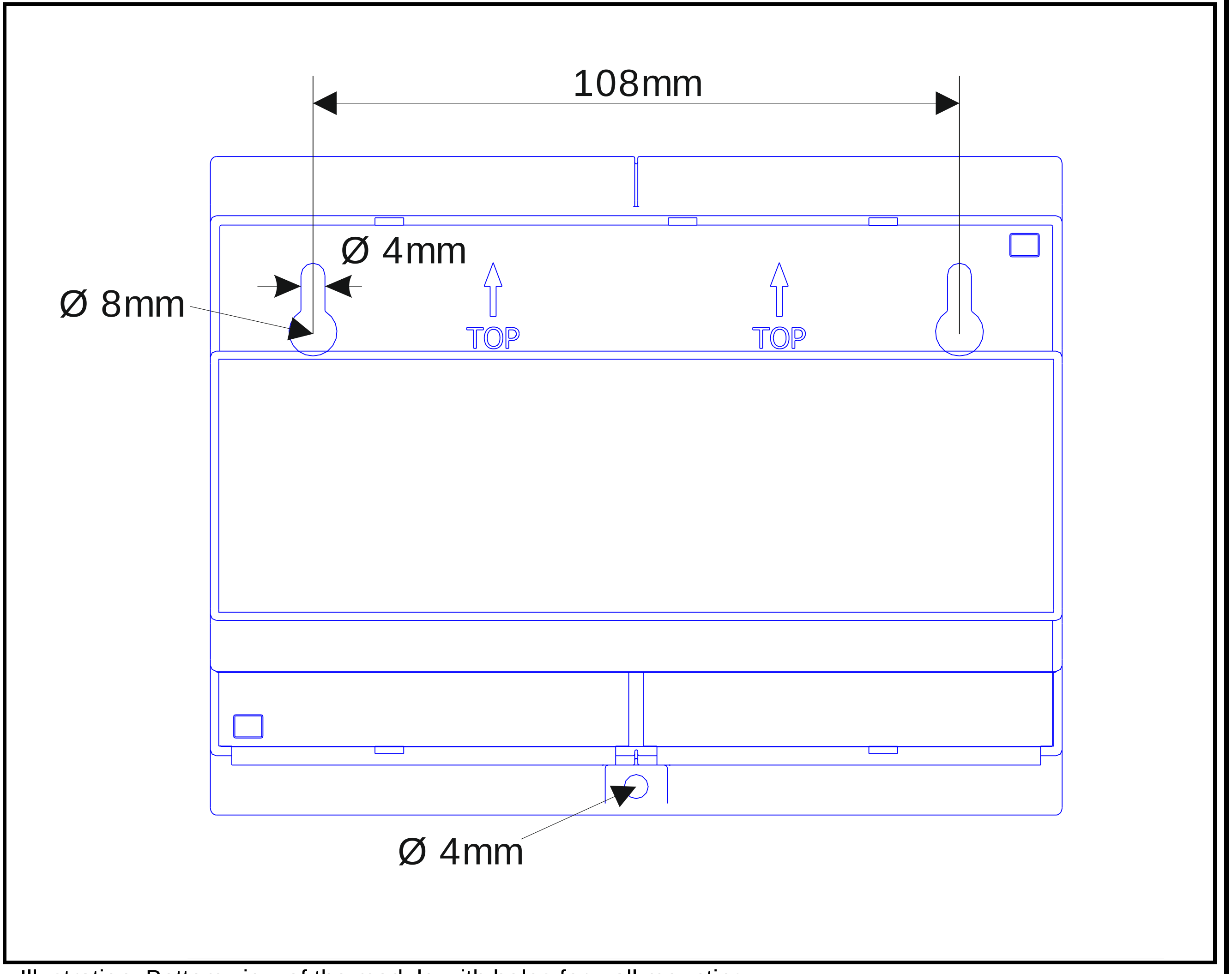


Illustration: Bottom view of the module with holes for wall mounting

10.4 Connection diagram

In the below drawing you will find the correct cabling of the IO module.

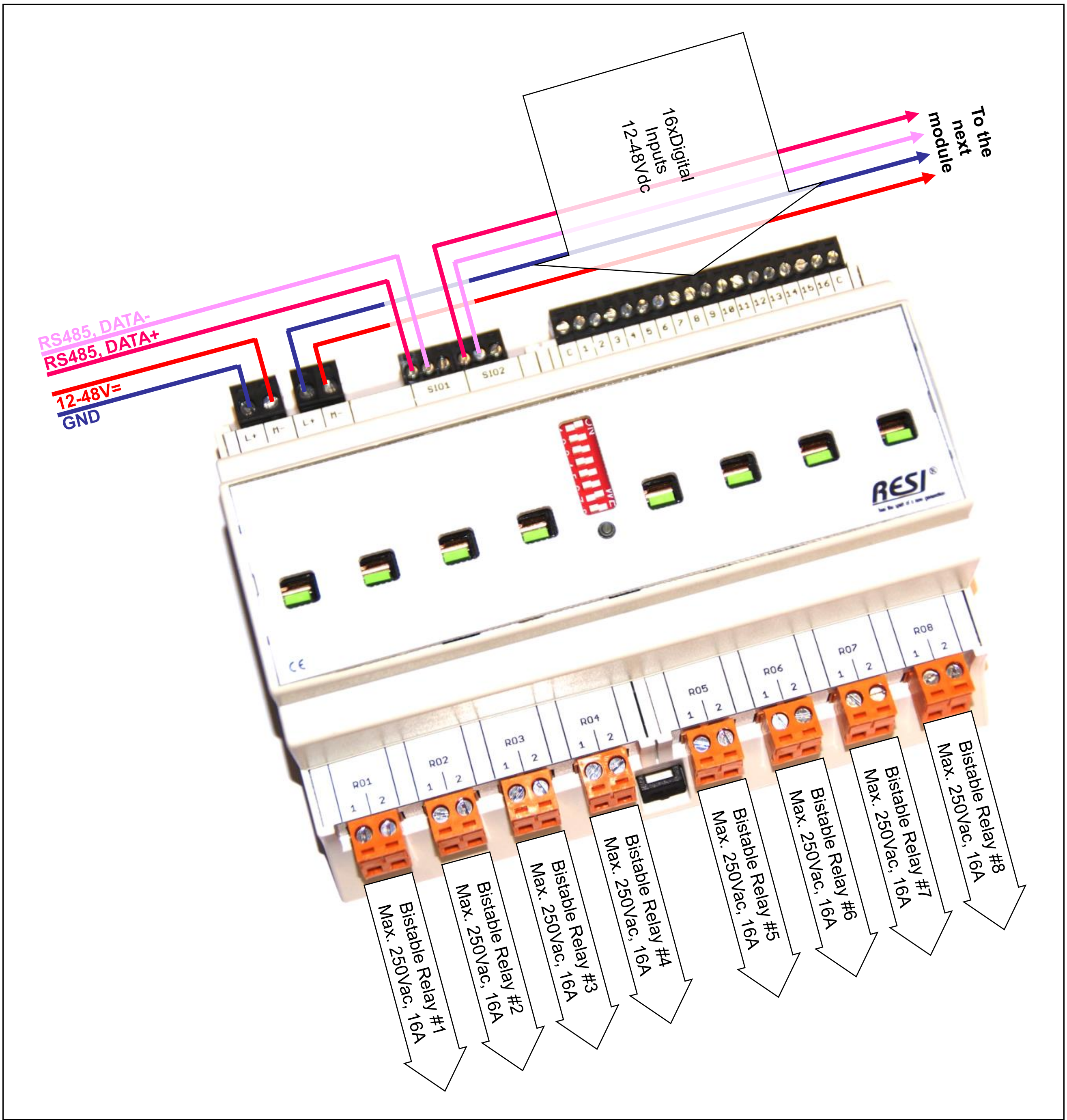


Illustration: cabling of the IO module

10.5 Clamps, DIP switch settings an LED indicators

The IO module offers the following clamps:

CLAMPS	RESI-16DI8RO-MODBUS, RESI-16DI8RO-ASCII
L+ M-	Power supply via two separated plug-in 2-pin terminal blocks. For daisy chain IN and OUT power supply of many modules L+: 12-48 V= M-: Ground
SIO1 1=A+ 2=B- 3=GND	RS485 ASCII or MODBUS/RTU serial interface IN A+: RS485 DATA+ signal B-: RS485 DATA- signal GND: RS485 ground signal
SIO2 1=A+ 2=B- 3=GND	RS485 ASCII or MODBUS/RTU serial interface OUT A+: RS485 DATA+ signal B-: RS485 DATA- signal GND: RS485 ground signal
DI C=GND 1=DI1 2=DI2 3=DI3 4=DI4 5=DI5 6=DI6 7=DI7 8=DI8 9=DI9 10=DI10 11=DI11 12=DI12 13=DI13 14=DI14 15=DI15 16=DI16 C=GND	16 digital inputs for 12-48Vdc signals C: Ground of the module DI1-DI16: Digital inputs 0=open or GND, 1=+12Vdc..+48Vdc
RO1 1=Relay+ 2=Relay-	Bistable relay output 1 with 2-pin plug in terminal block 1: Switching contact of the relay + 2: Switching contact of the relay -
RO2 1=Relay+ 2=Relay-	Bistable relay output 2 with 2-pin plug in terminal block 1: Switching contact of the relay + 2: Switching contact of the relay -
RO3 1=Relay+ 2=Relay-	Bistable relay output 3 with 2-pin plug in terminal block 1: Switching contact of the relay + 2: Switching contact of the relay -
RO4 1=Relay+ 2=Relay-	Bistable relay output 4 with 2-pin plug in terminal block 1: Switching contact of the relay + 2: Switching contact of the relay -
RO5 1=Relay+ 2=Relay-	Bistable relay output 5 with 2-pin plug in terminal block 1: Switching contact of the relay + 2: Switching contact of the relay -
RO6 1=Relay+ 2=Relay-	Bistable relay output 6 with 2-pin plug in terminal block 1: Switching contact of the relay + 2: Switching contact of the relay -
RO7 1=Relay+ 2=Relay-	Bistable relay output 7 with 2-pin plug in terminal block 1: Switching contact of the relay + 2: Switching contact of the relay -
RO8 1=Relay+ 2=Relay-	Bistable relay output 6 with 2-pin plug in terminal block 1: Switching contact of the relay + 2: Switching contact of the relay -

Table: Description of the terminal blocks of the IO module

The IO module offers also an 8-pin DIP switch and a dual color LED indicator:

<table>
<tr><th>DIP+LED</th><th>RESI-16DI8RO-MODBUS, RESI-16DI8RO-ASCII</th></tr>
<tr><td>DIP SWITCH
1=ADR0
2=ADR1
3=ADR2
4=ADR3
5=BR0
6=BR1
7=BR2
8=PARITY</td><td>DIP switch to setup the IO module
ADR: 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:
<table>
<tr><td>ADR3</td><td>ADR2</td><td>ADR1</td><td>ADR0</td><td>MODBUS/RTU unit number or ASCII bus number</td></tr>
<tr><td>OFF</td><td>OFF</td><td>OFF</td><td>OFF</td><td>Internal MODBUS/RTU unit number is used from the FLASH memory in the range of 0 to 255.</td></tr>
<tr><td>OFF</td><td>OFF</td><td>OFF</td><td>ON</td><td>1</td></tr>
<tr><td>OFF</td><td>OFF</td><td>ON</td><td>OFF</td><td>2</td></tr>
<tr><td>OFF</td><td>OFF</td><td>ON</td><td>ON</td><td>3</td></tr>
<tr><td>OFF</td><td>ON</td><td>OFF</td><td>OFF</td><td>4</td></tr>
<tr><td>OFF</td><td>ON</td><td>OFF</td><td>ON</td><td>5</td></tr>
<tr><td>OFF</td><td>ON</td><td>ON</td><td>OFF</td><td>6</td></tr>
<tr><td>OFF</td><td>ON</td><td>ON</td><td>ON</td><td>7</td></tr>
<tr><td>ON</td><td>OFF</td><td>OFF</td><td>OFF</td><td>8</td></tr>
<tr><td>ON</td><td>OFF</td><td>OFF</td><td>ON</td><td>9</td></tr>
<tr><td>ON</td><td>OFF</td><td>ON</td><td>OFF</td><td>10</td></tr>
<tr><td>ON</td><td>OFF</td><td>ON</td><td>ON</td><td>11</td></tr>
<tr><td>ON</td><td>ON</td><td>OFF</td><td>OFF</td><td>12</td></tr>
<tr><td>ON</td><td>ON</td><td>OFF</td><td>ON</td><td>13</td></tr>
<tr><td>ON</td><td>ON</td><td>ON</td><td>OFF</td><td>14</td></tr>
<tr><td>ON</td><td>ON</td><td>ON</td><td>ON</td><td>15</td></tr>
</table>
BAUD RATE: Those three DIP switches BR2-BR0 define the MODBUS/RTU or ASCII baud rate for the communication:
<table>
<tr><td>BR2</td><td>BR1</td><td>BR0</td><td>MODBUS/RTU or ASCII Baudrate</td></tr>
<tr><td>OFF</td><td>OFF</td><td>OFF</td><td>4800bd</td></tr>
<tr><td>OFF</td><td>OFF</td><td>ON</td><td>9600bd</td></tr>
<tr><td>OFF</td><td>ON</td><td>OFF</td><td>19200bd</td></tr>
<tr><td>OFF</td><td>ON</td><td>ON</td><td>38400bd</td></tr>
<tr><td>ON</td><td>OFF</td><td>OFF</td><td>57600bd</td></tr>
<tr><td>ON</td><td>OFF</td><td>ON</td><td>115200bd</td></tr>
<tr><td>ON</td><td>ON</td><td>OFF</td><td>230400bd</td></tr>
<tr><td>ON</td><td>ON</td><td>ON</td><td>256000bd</td></tr>
</table>
PARITY: This DIP switch PARITY defines the MODBUS/RTU or ASCII parity for the communication:
<table>
<tr><td>PARITY</td><td>MODBUS/RTU or ASCII parity</td></tr>
<tr><td>OFF</td><td>NONE</td></tr>
<tr><td>ON</td><td>EVEN</td></tr>
</table>
HINT: After changing on of the DIP switches, the module restarts completely and intialises the serial interface. You will notice that the WHITE LED will be on for approximately 2 seconds, before this LED will flash with a one second cycle.</td></tr>
<tr><td>LED indicator
WHITE
RED</td><td>The LED indicator consists out of a dual color LED with the following states:
OFF: Module has no power or module is defect
FLASH 1S WHITE/OFF: Module has power and works correct
SHORT RED: A MODBUS/RTU or ASCII telegram is received or transmitted
2S RED: The DIP switch was changed or the module does a reboot</td></tr>
</table>

Table: Description of the DIP switch functions and the indication LEDs on the IO module

10.7 Dimensions of the module

In the below drawing you will find the dimensions of the IO module.

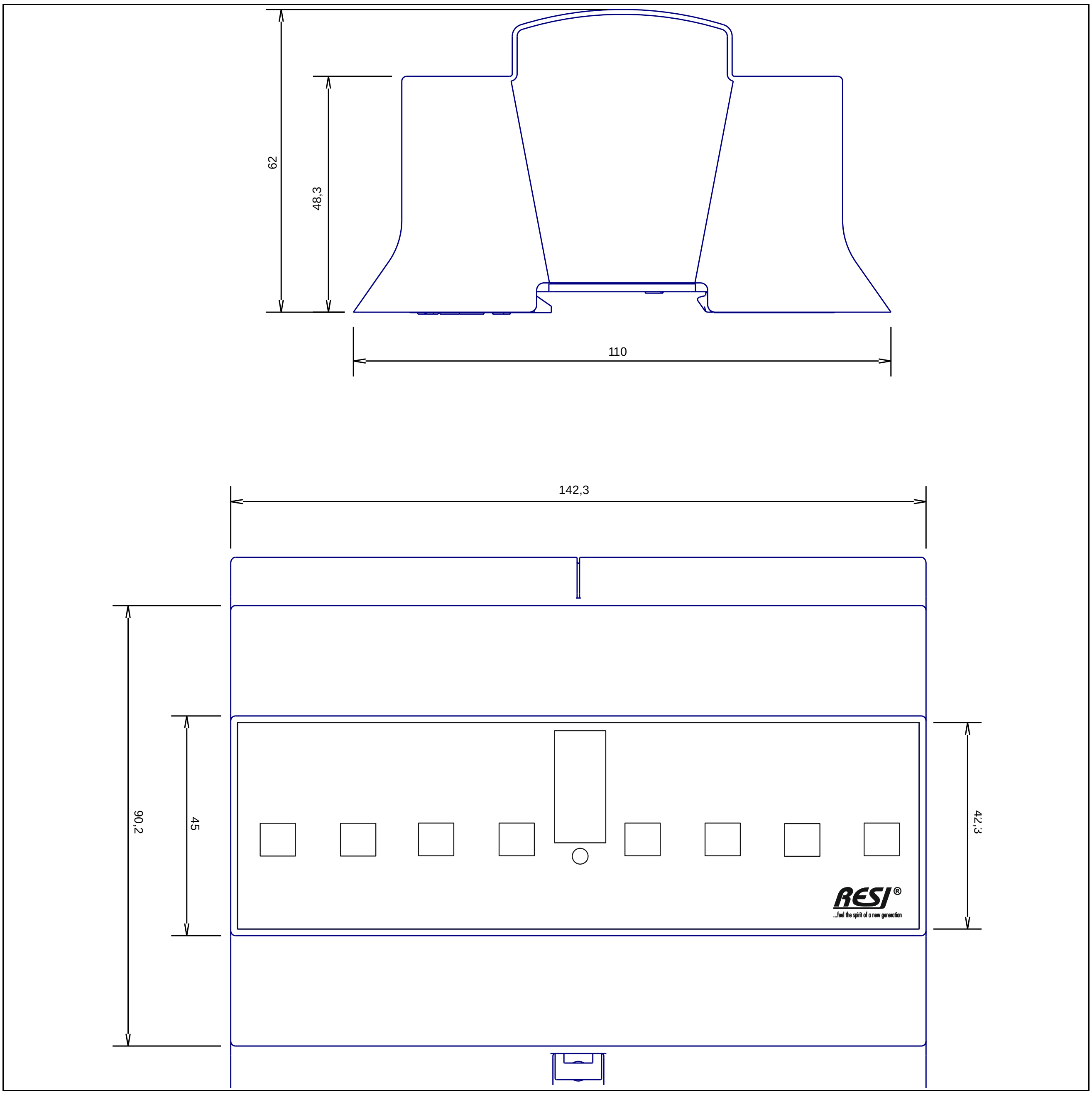


Illustration: Dimensions of the IO module in mm

Dimensions	
Dimensions of the housing L x B x H (mm)	143 x 110 x 62
Weight	560 g
Color	Grey, RAL7035
Material	Self-extinguish PC/ABS, DIN 43880
Protection class	IP20 based on DIN 40050/EN 60529

Table: technical data of the housing

10.8 Power supply of the module

In the below drawing you will find how to connect the module to a power supply.

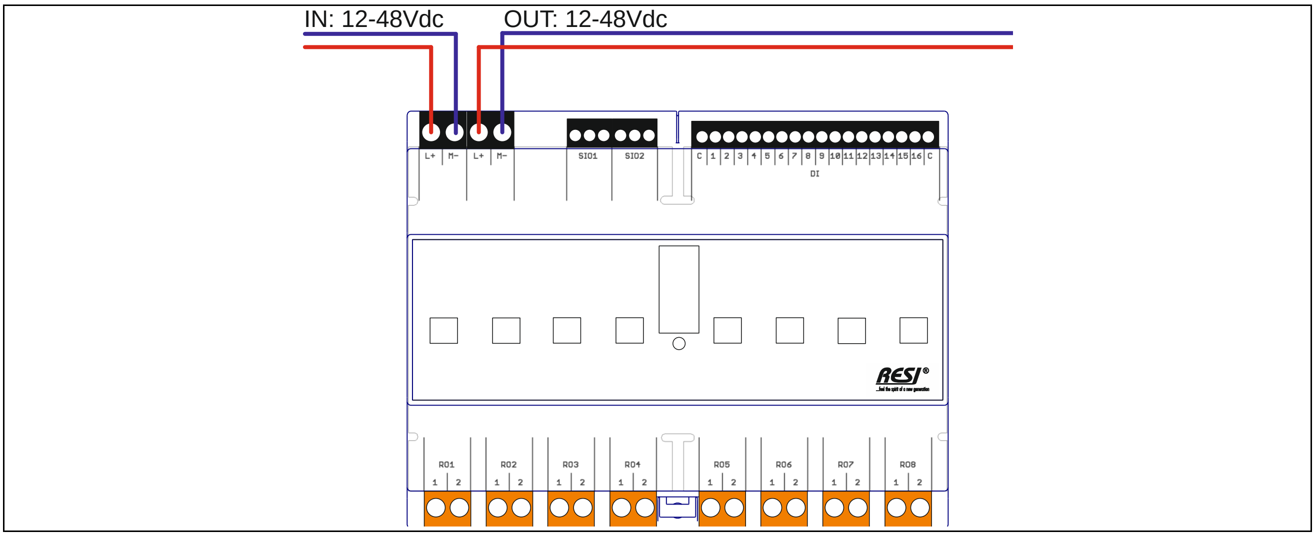


Illustration: Power supply of the IO module

The module offers two 2-pin plug-in terminals for connecting the power supply to the module. It is designed to create a daisy chain power supply with many modules.

10.9 Serial RS485 connection

In this drawing you see the cabling of the serial RS485 bus line. In the module both SIO terminal block are bridged.

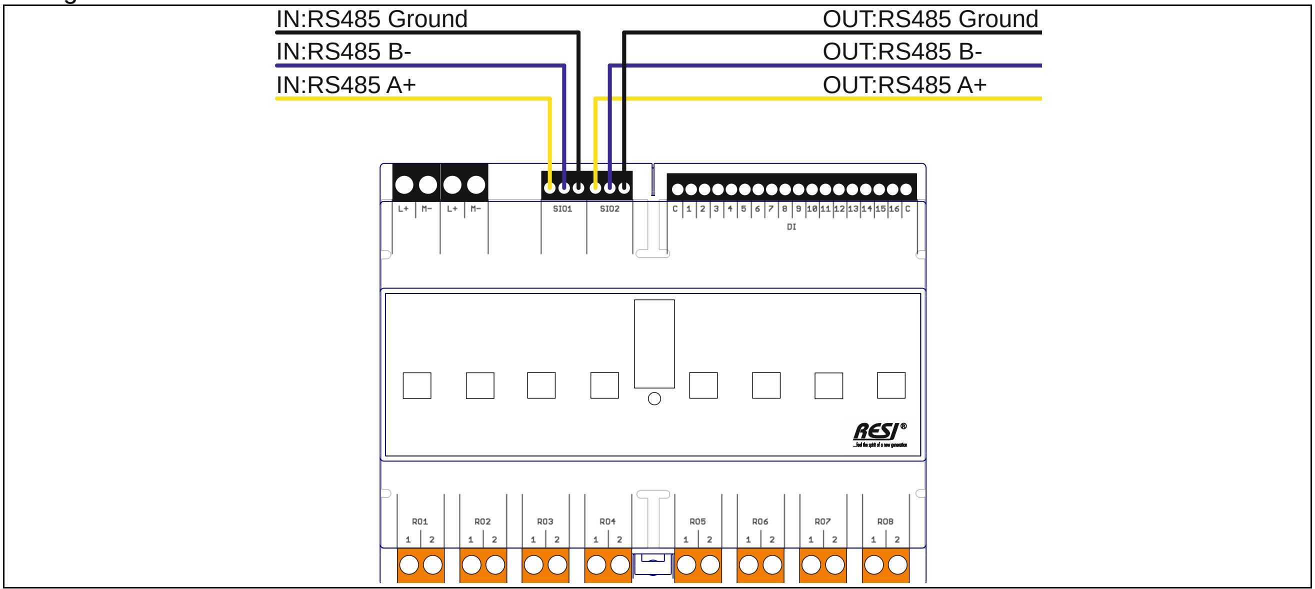


Illustration: RS485 bus cabling of the IO module

The module offers two plug-in 3-pin terminals to connect a RS485 bus line to the module. It was designed to create a daisy chain bus line with many modules. Don't forget, that a RS485 bus line needs a line termination at the end of both lines!

10.10 Cabling of the digital inputs of the module

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.

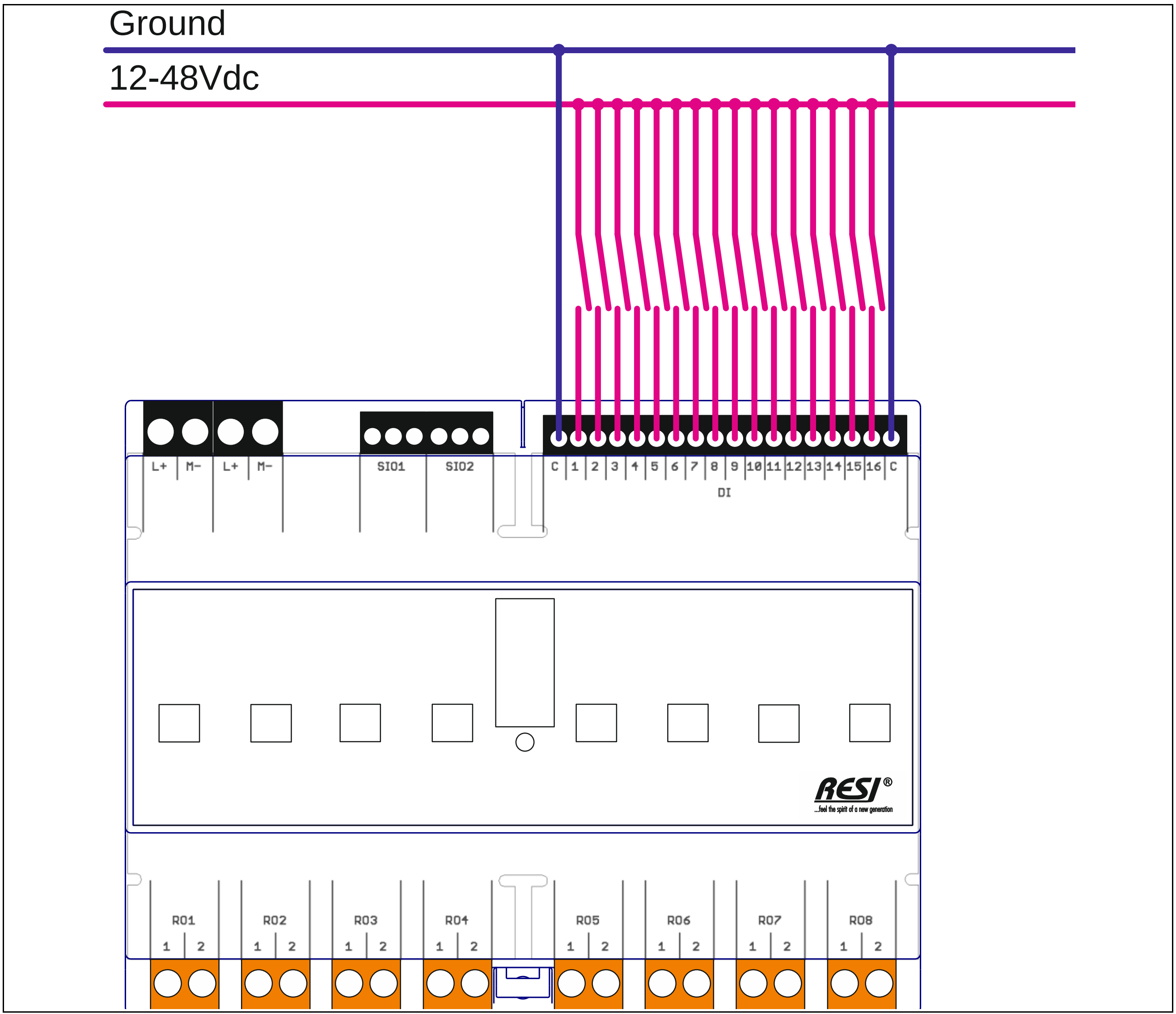


Illustration: Cabling of the digital inputs of the IO module

10.11 Cabling of the relay outputs of the module

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

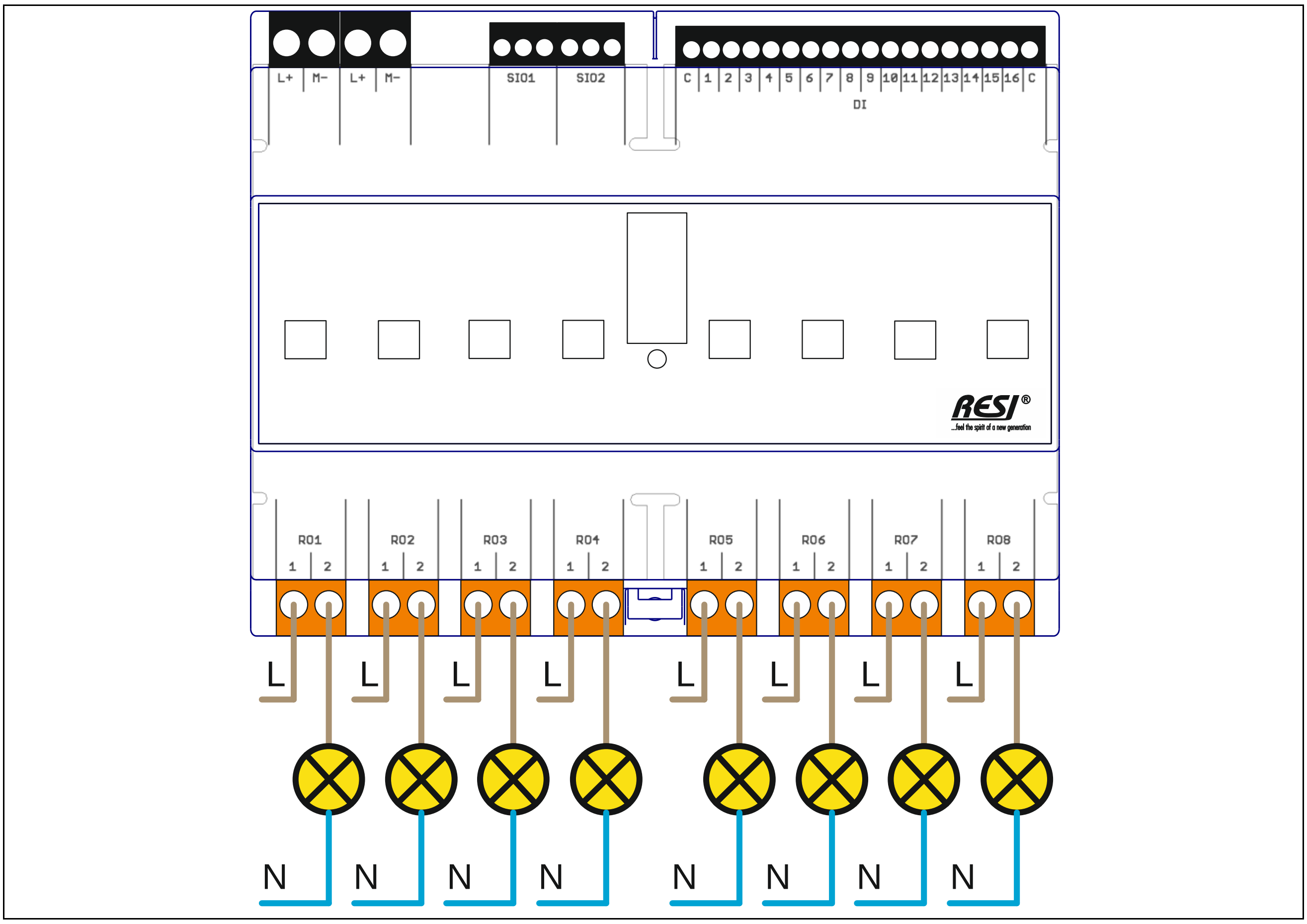


Illustration: Cabling of the bistable power relays of the IO module

10.12 Functional description

This IO module communicates with a host system with the MODBUS/RTU slave protocol. All versions of the Io module with RESI-xxx-ASCII offer an additional protocol: An ASCII slave protocol with simple text string. The communication runs over a RS485 interface (half duplex) or over a RS232 interface (full duplex).

For the communication with ASCII texts, the host sends ASCII messages with a special start character # (0x23, 35dec) and a special end character (0x0d,13dec or CARRIAGE RETURN) to the module. The module uses also this special start and end characters to answer to the host request. Consult the below noted detailed command descriptions. In the ASCII protocol mode, the host can send messages with or without a bus number.

For communication with the MODBUS/RTU slave, the module offers the following MODBUS functions:

- 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)

HINT:
The functions READ HOLDING REGISTER and PRESET MULTIPLE REGISTERS are restricted to max. 125 register per request! The functions READ INPUT STATUS, READ COIL STATUS and FORCE MULTIPLE COILS are restricted to 2000 coils or inputs (bits) per request.

10.13 ASCII protocol description

10.13.1 Overview

The IO module communicates with simple ASCII commands. The following special characters are used in this documentation:

stand for the **Hashtag** ASCII character 35dec or 0x23
: stand for the **colon** ASCII character 58dec or 0x3A
= stand for the **equal sign** with the ASCII code 61ec or 0x3D
- stand for the **minus sign** with the ASCII code 45dec or 0x2D
, stands for the **comma** with the ASCII code 44dec or 0x2C
<CR> stand for **CARRIAGE RETURN** ASCII character 13dec or 0x0D. In the following text we use the representation CR.
<SP> stands for the **SPACE**. This is the space character with the ASCII code 32dec or 0x20. In the following text we will use the representation□.
<ADR> will be used as the current **bus address** of the module. The bus number can be transmitted decimal or hexadecimal and is separated with a comma (ASCII code 44dec or 0x2C) from the rest of the command. Hexadecimal numbers always start with 0x. Its only allowed to use the ASCII characters ‚0‘-‚9‘ 48dec to 57dec, 0x30-0x39 and ‚A‘ to ‚F‘, 65dec to 70dec, 0x41-0x46. All modules react to the broadcast bus address 0 and to its own bus number. With a DIP switch, the user can easily change between the internal stored bus number in the FLASH and the fix bus number 255. Consult the DIP switch description for more details.

10.13.2 Communication sequence

In general the IO module sends no ASCII characters without a request from a host. So the host is the master of the communication and the IO module is always answering to host requests as a slave. If only one IO module is connected to a host (e.g. Via RS232), you can dump the bus number in the protocol. If you use a RS485 interface, more than one module can be connected to the host. Therefore a bus number in the request frame of the host is always necessary.

The structure of the command look like this:

The host sends a command or a command with parameters without a bus address to the module:
#<Command><CR> or
#<Command>:<Parameter><CR>

The module answers, if it feels addressed, with the following answer telegram:
#<Answer><CR>

If the bus number is used, the module answers with:
#<ADR>,<Command><CR> or
#<ADR>,<Command>:<Parameter><CR>

The module answers with:
#<ADR>,<Answer><CR>

The bus address lies in the range of 1dec to 255dec or 0x00 to 0xFF hexadecimal. The setup is done with our free configuration software MODBUSConfigurator. Each module reacts also to the broadcast bus address 0.

For each command, we define two different writings. A long version and a short version, to avoid unnecessary traffic on the bus. For example to request the software version of the module you can use the command VERSION or the short command VER.

10.13.3 Request VERSION

This command returns the current software version of the module.

Host long version:
#VERSION<CR> or
#<ADR>,VERSION<CR>

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

Answer:
#VERSION:<HIGH>.<MED>.<LOW><CR> or
#<ADR>,VERSION:<HIGH>,<MED>,<LOW><CR>

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

Samples:
→ **#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$_{CR}$**

10.13.4 Request module TYPE

This command returns the current type of the module.

Host long version:
#TYPE<CR> or
#<ADR>,TYPE<CR>

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

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

<TYP> defines the current type of the module. Currently RESI-16DI8RO-ASCII

Samples:
→ **#TYPE$_{CR}$**
← **#TYPE:RESI-16DI8RO-ASCII$_{CR}$**

→ **#255,TYP$_{CR}$**
← **#255,TYPE:RESI-16DI8RO-ASCII$_{CR}$**

10.13.5 Table of all ASCII commands

Here you will find a possible ASCII commands of the module. We use here only the version with bus number. That you can avoid the bus number, we have discussed earlier in this document. If an argument as the extension Dec, it will be returned as a decimal number, If an argument has the extension Hex, then this argument is returned as a hexadecimal number. Many command returns the argument in decimal and hexadecimal representation. So the host can select, what kind of number conversion, it will handle in its software.

Direction	ASCII command
Host	**#<BusAdr>,VER**$_{CR}$ **#<BusAdr>,VERSION**$_{CR}$
Answer	**#<BusAdr>,VERSION:<VersionHi>.< VersionMed>.<VersionLo>**$_{CR}$
	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)
Host	**#<BusAdr>,TYP**$_{CR}$ **#<BusAdr>,TYPE**$_{CR}$
Answer	**#<BusAdr>,TYPE:RESI-16DI8RO-ASCII**$_{CR}$
	Returns the current type of the module
Host	**#<BusAdr>,OWN**$_{CR}$ **#<BusAdr>,OWNER**$_{CR}$
Answer	**#<BusAdr>,OWNER:RESI**$_{CR}$
	Returns the owner of the module
Host	**#<BusAdr>,CRE**$_{CR}$ **#<BusAdr>,CREATOR**$_{CR}$
Answer	**#<BusAdr>,CREATOR:DI HC SIGL,MSC**$_{CR}$
	Returns the creator of the module

Direction	**ASCII command**
Host	**#<BusAdr>,COPY$_{CR}$** **#<BusAdr>,COPYRIGHT$_{CR}$**
Answer	**#<BusAdr>,COPYRIGHT:2015-16 BY RESI AND DI HC SIGL,MSC WWW.RESI.CC$_{CR}$**
	Returns a copyright note about the module
Host	**#<BusAdr>,GDIP$_{CR}$** **#<BusAdr>,GET☐DIP$_{CR}$**
Answer	**#<BusAdr>,GDIP:<DIPSwitchDec>,<DIPSwitchHex>$_{CR}$**
	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 (=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)
Host	**#<BusAdr>,GDIS$_{CR}$** **#<BusAdr>,GET☐DIS$_{CR}$**
Answer	**#<BusAdr>,GDIS:<DISDec>,<DISHex>$_{CR}$**
	Returns the current state of all 16 digital inputs as decimal number and as hexadecimal number. DISDec DISHex The current state of all digital inputs: Bit 0: State of DI1 (=0:OFF, =1:ON) Bit 1: State of DI2 (=0:OFF, =1:ON) Bit 2: State of DI3 (=0:OFF, =1:ON) Bit 3: State of DI4 (=0:OFF, =1:ON) Bit 4: State of DI5 (=0:OFF, =1:ON) Bit 5: State of DI6 (=0:OFF, =1:ON) Bit 6: State of DI7 (=0:OFF, =1:ON) Bit 7: State of DI8 (=0:OFF, =1:ON) Bit 8: State of DI9 (=0:OFF, =1:ON) Bit 9: State of DI10 (=0:OFF, =1:ON) Bit 10: State of DI11 (=0:OFF, =1:ON) Bit 11: State of DI12 (=0:OFF, =1:ON) Bit 12: State of DI13 (=0:OFF, =1:ON) Bit 13: State of DI14 (=0:OFF, =1:ON) Bit 14: State of DI15 (=0:OFF, =1:ON) Bit 15: State of DI16 (=0:OFF, =1:ON)
Host	**#<BusAdr>,GDIx$_{CR}$** **#<BusAdr>,GET☐DIx$_{CR}$**
Answer	**#<BusAdr>,GDIx:<DIxDec>,<DIxHex>$_{CR}$**
x	**1..16**
	Returns the current state of the digital input Dix as decimal number and as hexadecimal number. X stands for the desired digital input between 1 and 16. DIxDec DIxHex The current state of the digital input x: =0: Digital input is OFF =1: Digital input is ON

Direction	ASCII command
Host	**#<BusAdr>,GROS$_{CR}$** **#<BusAdr>,GET□ROS$_{CR}$**
Answer	**#<BusAdr>,GROS:<ROSDec>,<ROSHex>$_{CR}$**
	Returns the current state of the eight relay outputs as decimal number and as hexadecimal number. ROSDec ROSHex The current state of the eight relay outputs: Bit 0: State of RO1 (=0:OFF, =1:ON) Bit 1: State of RO2 (=0:OFF, =1:ON) Bit 2: State of RO3 (=0:OFF, =1:ON) Bit 3: State of RO4 (=0:OFF, =1:ON) Bit 4: State of RO5 (=0:OFF, =1:ON) Bit 5: State of RO6 (=0:OFF, =1:ON) Bit 6: State of RO7 (=0:OFF, =1:ON) Bit 7: State of RO8 (=0:OFF, =1:ON)
Host	**#<BusAdr>,SROS:<OutAllROS>$_{CR}$** **#<BusAdr>,SET□ROS:<OutAllROS>$_{CR}$**
Answer	**#OK$_{CR}$**
	Sets all eight bistable relay to the new state <OutAllROS> OutAllROS The new state for all bistable relay outputs: Bit 0: State for RO1 (=0:RO to OFF, =1:RO to ON) Bit 1: State for RO2 (=0:RO to OFF, =1:RO to ON) Bit 2: State for RO3 (=0:RO to OFF, =1:RO to ON) Bit 3: State for RO4 (=0:RO to OFF, =1:RO to ON) Bit 4: State for RO5 (=0:RO to OFF, =1:RO to ON) Bit 5: State for RO6 (=0:RO to OFF, =1:RO to ON) Bit 6: State for RO7 (=0:RO to OFF, =1:RO to ON) Bit 7: State for RO8 (=0:RO to OFF, =1:RO to ON)
Host	**#<BusAdr>,GROx$_{CR}$** **#<BusAdr>,GET□ROx$_{CR}$**
Answer	**#<BusAdr>,GROx:<ROxDec>,<ROxHex>$_{CR}$**
x	**1..8**
	Returns the current state of the relay output ROx as decimal number and as hexadecimal number. X stands for the number of the relay output from 1 to 8. ROxDec ROxHex The current state of the bistable relay output ROx: =0: relay output is OFF =1: relay output is ON
Host	**#<BusAdr>,SROx:<Out>$_{CR}$** **#<BusAdr>,SET□ROx:<Out>$_{CR}$**
Answer	**#OK$_{CR}$**
x	**1..8**
	Sets the new state for relay output ROx. The state is defined with <Out>. X stands for the affected relay output between 1 and 8. Out The new state of the relay output ROx: =0: relay output is OFF =1: relay output is ON

Direction	**ASCII command**
Host	**#<BusAdr>,GCROx$_{CR}$** **#<BusAdr>,GET□CYCLES□ROx$_{CR}$**
Answer	**#<BusAdr>,GCROx:<CROxDec>,<CROxHex>$_{CR}$**
x	**1..8**
	The module counts the amount of switching cycles for each relay output in a non-volatile internal FRAM memory separately. With this command you can read out the current amount of switching cycles for the relay output #x as decimal number and as hexadecimal number. You cannot reset or delete this counter! CROxDec CROxHex The current amount of switching cycles of the output x
Host	**#<BusAdr>,RDIx$_{CR}$** **#<BusAdr>,RISE□DIx$_{CR}$**
Answer	**#<BusAdr>,RDIx:<RDIxDec>,<RDIxHex>$_{CR}$**
x	**1..16**
	Returns the current counter for rising edges on the digital input x since last power on of the module as decimal number and as hexadecimal number. RDIxDec RDIxHex The current amount of counted rising edges on the digital input x
Host	**#<BusAdr>,FDIx$_{CR}$** **#<BusAdr>,FALL□DIx$_{CR}$**
Answer	**#<BusAdr>,FDIx:<FDIxDec>,<FDIxHex>$_{CR}$**
x	**1..16**
	Returns the current counter for falling edges on the digital input x since last power on of the module as decimal number and as hexadecimal number. FDIxDec FDIxHex The current amount of counted falling edges on the digital input x
Host	**#<BusAdr>,RC$_{CR}$** **#<BusAdr>,RESET□COUNTERS$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
	This command deletes all counters for rising and falling edges of the 16 digital inputs in the module to 0.
Host	**#<BusAdr>,SSMODE:<Mode>$_{CR}$** **#<BusAdr>,SET□SPECIAL□MODE:<Mode>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
	The module offers an internal logic, which can combine digital inputs with the rely outputs. With this command you can switch on or off this internal logic execution. Mode The new mode for the logic execution =0: No logic is executed =1: Internal logic execution is activated
Host	**#<BusAdr>,GSMODE$_{CR}$** **#<BusAdr>,GET□SPECIAL□MODE$_{CR}$**
Answer	**#<BusAdr>,GSMODE:<ModeDec>,<ModeHex>$_{CR}$**
	Returns the current status of the internal logic execution of the module. ModeDec ModeHex The current status of the internal logic execution =0: No logic is executed =1: Internal logic execution is activated

Direction	**ASCII command**
Host	**#<BusAdr>,RSTSPCMODE$_{CR}$** **#<BusAdr>,RESET☐SPECIAL☐MODE$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
	This command switches off the internal logic execution and deletes the complete configuration registers for the mapping of the digital inputs to the relay outputs of the internal logic execution to 0.
Host	**#<BusAdr>,SSWITCHx:<DIPattern>$_{CR}$** **#<BusAdr>,SET☐SWITCHx:<DIPattern>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
x	**1..8**
	This command defines a mapping table for the output relay x with all 16 digital inputs of the module for the logic function SWITCH. SWITCH function: Is the mapped digital input high (1), the relay output will be turned on (set to 1). Is the mapped digital input low (0), the relay output will be turned off (set to 0). It makes sense to map only one digital input to a relay output, because this function works with the current state of the digital input, not with a rising edge like the other logic functions. DIPattern A 16 bit value as decimal or hexadecimal number Each bit stands for one of the 16 digital inputs Bit 0: =1: Current value of DI1 switches ROx ON/OFF, =0: DI1 is ignored Bit 1: =1: Current value of DI2 switches ROx ON/OFF, =0: DI2 is ignored Bit 2: =1: Current value of DI3 switches ROx ON/OFF, =0: DI3 is ignored Bit 3: =1: Current value of DI4 switches ROx ON/OFF, =0: DI4 is ignored Bit 4: =1: Current value of DI5 switches ROx ON/OFF, =0: DI5 is ignored Bit 5: =1: Current value of DI6 switches ROx ON/OFF, =0: DI6 is ignored Bit 6: =1: Current value of DI7 switches ROx ON/OFF, =0: DI7 is ignored Bit 7: =1: Current value of DI8 switches ROx ON/OFF, =0: DI8 is ignored Bit 8: =1: Current value of DI9 switches ROx ON/OFF, =0: DI9 is ignored Bit 9: =1: Current value of DI10 switches ROx ON/OFF, =0: DI10 is ignored Bit 10: =1: Current value of DI11 switches ROx ON/OFF, =0: DI11 is ignored Bit 11: =1: Current value of DI12 switches ROx ON/OFF, =0: DI12 is ignored Bit 12: =1: Current value of DI13 switches ROx ON/OFF, =0: DI13 is ignored Bit 13: =1: Current value of DI14 switches ROx ON/OFF, =0: DI14 is ignored Bit 14: =1: Current value of DI15 switches ROx ON/OFF, =0: DI15 is ignored Bit 15: =1: Current value of DI16 switches ROx ON/OFF, =0: DI16 is ignored
Host	**#<BusAdr>,GSWITCHx$_{CR}$** **#<BusAdr>,GET☐SWITCHx$_{CR}$**
Answer	**#<BusAdr>,GSWITCHx:<DIPatternDec>,<DIPatternHex>$_{CR}$**
x	**1..8**
	This command returns the current mapping of all 16 digital inputs to the relay output x (1..8) for the logic function SWITCH. DIPatternDec DIPatternHex refer to DIPattern in the command SET SWITCHx

Direction	**ASCII command**
Host	**#<BusAdr>,SSONx:<DIPattern>$_{CR}$** **#<BusAdr>,SET□SWITCH□ONx:<DIPattern>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
x	**1..8**
	This command defines a mapping table for the output relay x with all 16 digital inputs of the module for the logic function SWITCH ON. SWITCH ON function: If the module detects a rising edge on one of the mapped digital inputs, it sets the corresponding relay output to high (1). DIPattern A 16 bit value as decimal or hexadecimal number Each bit stands for one of the 16 digital inputs Bit 0: =1: Rising edge on DI1 switches ROx ON, =0: DI1 is ignored Bit 1: =1: Rising edge on DI2 switches ROx ON, =0: DI2 is ignored Bit 2: =1: Rising edge on DI3 switches ROx ON, =0: DI3 is ignored Bit 3: =1: Rising edge on DI4 switches ROx ON, =0: DI4 is ignored Bit 4: =1: Rising edge on DI5 switches ROx ON, =0: DI5 is ignored Bit 5: =1: Rising edge on DI6 switches ROx ON, =0: DI6 is ignored Bit 6: =1: Rising edge on DI7 switches ROx ON, =0: DI7 is ignored Bit 7: =1: Rising edge on DI8 switches ROx ON, =0: DI8 is ignored Bit 8: =1: Rising edge on DI9 switches ROx ON, =0: DI9 is ignored Bit 9: =1: Rising edge on DI10 switches ROx ON, =0: DI10 is ignored Bit 10: =1: Rising edge on DI11 switches ROx ON, =0: DI11 is ignored Bit 11: =1: Rising edge on DI12 switches ROx ON, =0: DI12 is ignored Bit 12: =1: Rising edge on DI13 switches ROx ON, =0: DI13 is ignored Bit 13: =1: Rising edge on DI14 switches ROx ON, =0: DI14 is ignored Bit 14: =1: Rising edge on DI15 switches ROx ON, =0: DI15 is ignored Bit 15: =1: Rising edge on DI16 switches ROx ON, =0: DI16 is ignored
Host	**#<BusAdr>,GSONx$_{CR}$** **#<BusAdr>,GET□SWITCH□ONx$_{CR}$**
Answer	**#<BusAdr>,GSONx:<DIPatternDec>,<DIPatternHex>$_{CR}$**
x	**1..8**
	This command returns the current mapping of all 16 digital inputs to the relay output x (1..8) for the logic function SWITCH ON. DIPatternDec DIPatternHex refer to DIPattern in the command SET SWITCH ONx

Direction	**ASCII command**
Host	**#<BusAdr>,SSOFFx:<DIPattern>$_{CR}$** **#<BusAdr>,SET☐SWITCH☐OFFx:<DIPattern>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
x	**1..8**
	This command defines a mapping table for the output relay x with all 16 digital inputs of the module for the logic function SWITCH OFF. SWITCH OFF function: If the module detects a rising edge on one of the mapped digital inputs, it sets the corresponding relay output to low (1). DIPattern A 16 bit value as decimal or hexadecimal number Each bit stands for one of the 16 digital inputs Bit 0: =1: Rising edge on DI1 switches ROx OFF, =0: DI1 is ignored Bit 1: =1: Rising edge on DI2 switches ROx OFF, =0: DI2 is ignored Bit 2: =1: Rising edge on DI3 switches ROx OFF, =0: DI3 is ignored Bit 3: =1: Rising edge on DI4 switches ROx OFF, =0: DI4 is ignored Bit 4: =1: Rising edge on DI5 switches ROx OFF, =0: DI5 is ignored Bit 5: =1: Rising edge on DI6 switches ROx OFF, =0: DI6 is ignored Bit 6: =1: Rising edge on DI7 switches ROx OFF, =0: DI7 is ignored Bit 7: =1: Rising edge on DI8 switches ROx OFF, =0: DI8 is ignored Bit 8: =1: Rising edge on DI9 switches ROx OFF, =0: DI9 is ignored Bit 9: =1: Rising edge on DI10 switches ROx OFF, =0: DI10 is ignored Bit 10: =1: Rising edge on DI11 switches ROx OFF, =0: DI11 is ignored Bit 11: =1: Rising edge on DI12 switches ROx OFF, =0: DI12 is ignored Bit 12: =1: Rising edge on DI13 switches ROx OFF, =0: DI13 is ignored Bit 13: =1: Rising edge on DI14 switches ROx OFF, =0: DI14 is ignored Bit 14: =1: Rising edge on DI15 switches ROx OFF, =0: DI15 is ignored Bit 15: =1: Rising edge on DI16 switches ROx OFF, =0: DI16 is ignored
Host	**#<BusAdr>,GSOFFx$_{CR}$** **#<BusAdr>,GET☐SWITCH☐OFFx$_{CR}$**
Answer	**#<BusAdr>,GSOFFx:<DIPatternDec>,<DIPatternHex>$_{CR}$**
x	**1..8**
	This command returns the current mapping of all 16 digital inputs to the relay output x (1..8) for the logic function SWITCH OFF. DIPatternDec DIPatternHex refer to DIPattern in the command SET SWITCH OFFx

Direction	ASCII command
Host	**#<BusAdr>,STOGGLEx:<DIPattern>$_{\text{CR}}$** **#<BusAdr>,SET□TOGGLEx:<DIPattern>$_{\text{CR}}$**
Answer	**#<BusAdr>,OK$_{\text{CR}}$**
x	**1..8**
	This command defines a mapping table for the output relay x with all 16 digital inputs of the module for the logic function TOGGLE. TOGGLE function: If the module detects a rising edge on one of the mapped digital inputs, it inverts the current state of the corresponding relay output. DIPattern A 16 bit value as decimal or hexadecimal number Each bit stands for one of the 16 digital inputs Bit 0: =1: Rising edge on DI1 inverts ROx, =0: DI1 is ignored Bit 1: =1: Rising edge on DI2 inverts ROx, =0: DI2 is ignored Bit 2: =1: Rising edge on DI3 inverts ROx, =0: DI3 is ignored Bit 3: =1: Rising edge on DI4 inverts ROx, =0: DI4 is ignored Bit 4: =1: Rising edge on DI5 inverts ROx, =0: DI5 is ignored Bit 5: =1: Rising edge on DI6 inverts ROx, =0: DI6 is ignored Bit 6: =1: Rising edge on DI7 inverts ROx, =0: DI7 is ignored Bit 7: =1: Rising edge on DI8 inverts ROx, =0: DI8 is ignored Bit 8: =1: Rising edge on DI9 inverts ROx, =0: DI9 is ignored Bit 9: =1: Rising edge on DI10 inverts ROx, =0: DI10 is ignored Bit 10: =1: Rising edge on DI11 inverts ROx, =0: DI11 is ignored Bit 11: =1: Rising edge on DI12 inverts ROx, =0: DI12 is ignored Bit 12: =1: Rising edge on DI13 inverts ROx, =0: DI13 is ignored Bit 13: =1: Rising edge on DI14 inverts ROx, =0: DI14 is ignored Bit 14: =1: Rising edge on DI15 inverts ROx, =0: DI15 is ignored Bit 15: =1: Rising edge on DI16 inverts ROx, =0: DI16 is ignored
Host	**#<BusAdr>,GTOGGLEx$_{\text{CR}}$** **#<BusAdr>,GET□TOGGLEx$_{\text{CR}}$**
Answer	**#<BusAdr>,GTOGGLEx:<DIPatternDec>,<DIPatternHex>$_{\text{CR}}$**
x	**1..8**
	This command returns the current mapping of all 16 digital inputs to the relay output x (1..8) for the logic function TOGGLE. DIPatternDec DIPatternHex refer to DIPattern in the command SET TOGGLEx

Direction	**ASCII command**
Host	**#<BusAdr>,SPULSEx:<DIPattern>$_{CR}$** **#<BusAdr>,SET☐PULSEx:<DIPattern>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
x	**1..8**
	This command defines a mapping table for the output relay x with all 16 digital inputs of the module for the logic function PULSE. PULSE function: If the module detects a rising edge on one of the mapped digital inputs, it switches the corresponding relay output x on for a defined PULSE TIME. This time is individual for each relay output. After the time has passed by, the relay output is switched to low automatically (0). DIPattern A 16 bit value as decimal or hexadecimal number Each bit stands for one of the 16 digital inputs Bit 0: =1: Rising edge on DI1 creates a pulse on ROx, =0: DI1 is ignored Bit 1: =1: Rising edge on DI2 creates a pulse on ROx, =0: DI2 is ignored Bit 2: =1: Rising edge on DI3 creates a pulse on ROx, =0: DI3 is ignored Bit 3: =1: Rising edge on DI4 creates a pulse on ROx, =0: DI4 is ignored Bit 4: =1: Rising edge on DI5 creates a pulse on ROx, =0: DI5 is ignored Bit 5: =1: Rising edge on DI6 creates a pulse on ROx, =0: DI6 is ignored Bit 6: =1: Rising edge on DI7 creates a pulse on ROx, =0: DI7 is ignored Bit 7: =1: Rising edge on DI8 creates a pulse on ROx, =0: DI8 is ignored Bit 8: =1: Rising edge on DI9 creates a pulse on ROx, =0: DI9 is ignored Bit 9: =1: Rising edge on DI10 creates a pulse on ROx, =0: DI10 is ignored Bit 10: =1: Rising edge on DI11 creates a pulse on ROx, =0: DI11 is ignored Bit 11: =1: Rising edge on DI12 creates a pulse on ROx, =0: DI12 is ignored Bit 12: =1: Rising edge on DI13 creates a pulse on ROx, =0: DI13 is ignored Bit 13: =1: Rising edge on DI14 creates a pulse on ROx, =0: DI14 is ignored Bit 14: =1: Rising edge on DI15 creates a pulse on ROx, =0: DI15 is ignored Bit 15: =1: Rising edge on DI16 creates a pulse on ROx, =0: DI16 is ignored
Host	**#<BusAdr>,GPULSEx$_{CR}$** **#<BusAdr>,GET☐PULSEx$_{CR}$**
Answer	**#<BusAdr>,GPULSEx:<DIPatternDec>,<DIPatternHex>$_{CR}$**
x	**1..8**
	This command returns the current mapping of all 16 digital inputs to the relay output x (1..8) for the logic function PULSE. DIPatternDec DIPatternHex refer to DIPattern in the command SET PULSEx

Direction	**ASCII command**
Host	**#<BusAdr>,SPTIMEx:<Time>$_{CR}$** **#<BusAdr>,SET□PULSE□TIMEx:<Time>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
x	**1..8**
	This command defines for the output relay x (1..8) a time in 1/10s. This time is used to switch off the relay output automatically, after it was switched on by a rising edge on one of the mapped digital inputs. PULSE function: If the module detects a rising edge on one of the mapped digital inputs, it switches the corresponding relay output x on for a defined PULSE TIME. This time is individual for each relay output. After the time has passed by, the relay output is switched to low automatically (0). Time a time in 1/10s for the duration of the pulse
Host	**#<BusAdr>,GPTIMEx$_{CR}$** **#<BusAdr>,GET□PULSE□TIMEx$_{CR}$**
Answer	**#<BusAdr>,GPTIMEx:<TimeDec>,<TimeHex>$_{CR}$**
x	**1..8**
	This command returns the current duration for relay output x (1..8) for logic function PULSE. TimeDec TimeHex a time in 1/10s for the duration of the pulse
Host	**#<BusAdr>,GPTIMERx$_{CR}$** **#<BusAdr>,GET□PULSE□TIMERx$_{CR}$**
Answer	**#<BusAdr>,GPTIMERx:<TimerDec>,<TimerHex>$_{CR}$**
x	**1..8**
	This command returns the rest of the ON pulse duration in 1/10s of the running pulse on relay output x (1..8) for the logic function PULSE. PULSE function: If the module detects a rising edge on one of the mapped digital inputs, it switches the corresponding relay output x on for a defined PULSE TIME. This time is individual for each relay output. After the time has passed by, the relay output is switched to low automatically (0). TimerDec TimerHex The rest of the time period in 1/10s for the duration of the current pulse

Direction	**ASCII command**
Host	**#<BusAdr>,SMBADR:<MBUnit>$_{CR}$** **#<BusAdr>,SET□MODBUS□ADDRESS:<MBUnit>$_{CR}$**
Answer	**#<BusAdr>,OK $_{CR}$**
	Writes the unit address into the FLASH memory of the module. The new unit address for MODBUS/RTU or ASCII mode is only used immediately, if the DIP switch setting of the bus address is 0. Otherwise the unit address is defined by the DIP settings. The unit address ranges from 0dec to 255dec.
Host	**#<BusAdr>,GMBADR$_{CR}$** **#<BusAdr>,GET□MODBUS□ADDRESS$_{CR}$**
Answer	**#<BusAdr>,GMBADR:<MBUnitDec>,<MBFLASHDec>,<MBUnitHex>,<MBFLASHHex>$_{CR}$**
	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. MBUnitDec MBUnitHex: The current used MODBUS/RTU unit or ASCII address for communication MBFLASHDec MBFLASHHex: The internal stored MODBUS/RTU unit address or ASCII address from the FLASH memory, if the DIP switch is 0.
Host	**#<BusAdr>,RST$_{CR}$** **#<BusAdr>,RESET$_{CR}$**
Answer	**none**
	Executes a software reset (Reboot) of the module.

10.14 MODBUS – register description

10.14.1 Table of inputs and coils

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 0x00001 to 0x65536. Inputs are usually noted with 1x00001 to 1x65536. 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: 0x00100 for the coil 100, 1x00100 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.

Register	Description
0x00001 1x00001 I:0 R/O DI1	Current state of the digital input DI1 =0:DI is OFF, =1:DI is ON
0x00002 1x00002 I:1 R/O DI2	Current state of the digital input DI2 =0:DI is OFF, =1:DI is ON
0x00003 1x00003 I:2 R/O DI3	Current state of the digital input DI3 =0:DI is OFF, =1:DI is ON
0x00004 1x00004 I:3 R/O DI4	Current state of the digital input DI4 =0:DI is OFF, =1:DI is ON
0x00005 1x00005 I:4 R/O DI5	Current state of the digital input DI5 =0:DI is OFF, =1:DI is ON
0x00006 1x00006 I:5 R/O DI6	Current state of the digital input DI6 =0:DI is OFF, =1:DI is ON
0x00007 1x00007 I:6 R/O DI7	Current state of the digital input DI7 =0:DI is OFF, =1:DI is ON
0x00008 1x00008 I:7 R/O DI8	Current state of the digital input DI8 =0:DI is OFF, =1:DI is ON

Register	Description
0x00009 1x00009 I:8 R/O DI9	Current state of the digital input DI9 =0:DI is OFF, =1:DI is ON
0x00010 1x00010 I:9 R/O DI10	Current state of the digital input DI10 =0:DI is OFF, =1:DI is ON
0x00011 1x00011 I:10 R/O DI11	Current state of the digital input DI11 =0:DI is OFF, =1:DI is ON
0x00012 1x00012 I:11 R/O DI12	Current state of the digital input DI12 =0:DI is OFF, =1:DI is ON
0x00013 1x00013 I:12 R/O DI13	Current state of the digital input DI13 =0:DI is OFF, =1:DI is ON
0x00014 1x00014 I:13 R/O DI14	Current state of the digital input DI14 =0:DI is OFF, =1:DI is ON
0x00015 1x00015 I:14 R/O DI15	Current state of the digital input DI15 =0:DI is OFF, =1:DI is ON
0x00016 1x00016 I:15 R/O DI16	Current state of the digital input DI16 =0:DI is OFF, =1:DI is ON

Register	Description
0x00017 1x00017 I:16 R/W RO1	Current state of the bistable relay output RO1 =0:RO is OFF, =1:RO is ON Writing on this register changes the state of the relay output
0x00018 1x00018 I:17 R/W RO2	Current state of the bistable relay output RO2 =0:RO is OFF, =1:RO is ON Writing on this register changes the state of the relay output
0x00019 1x00019 I:18 R/W RO3	Current state of the bistable relay output RO3 =0:RO is OFF, =1:RO is ON Writing on this register changes the state of the relay output
0x00020 1x00020 I:19 R/W RO4	Current state of the bistable relay output RO4 =0:RO is OFF, =1:RO is ON Writing on this register changes the state of the relay output
0x00021 1x00021 I:20 R/W RO5	Current state of the bistable relay output RO5 =0:RO is OFF, =1:RO is ON Writing on this register changes the state of the relay output
0x00022 1x00022 I:21 R/W RO6	Current state of the bistable relay output RO6 =0:RO is OFF, =1:RO is ON Writing on this register changes the state of the relay output
0x00023 1x00023 I:22 R/W RO7	Current state of the bistable relay output RO7 =0:RO is OFF, =1:RO is ON Writing on this register changes the state of the relay output
0x00024 1x00024 I:23 R/W RO8	Current state of the bistable relay output RO8 =0:RO is OFF, =1:RO is ON Writing on this register changes the state of the relay output

Register	Description
0x00025 1x00025 I:24 R/O DIP1	Current state of DIP switch 1 =0:Dip switch is OFF, =1: Dip switch is ON
0x00026 1x00026 I:25 R/O DIP2	Current state of DIP switch 2 =0:Dip switch is OFF, =1: Dip switch is ON
0x00027 1x00027 I:26 R/O DIP3	Current state of DIP switch 3 =0:Dip switch is OFF, =1: Dip switch is ON
0x00028 1x00028 I:27 R/O DIP4	Current state of DIP switch 4 =0:Dip switch is OFF, =1: Dip switch is ON
0x00029 1x00029 I:28 R/O DIP5	Current state of DIP switch 5 =0:Dip switch is OFF, =1: Dip switch is ON
0x00030 1x00030 I:29 R/O DIP6	Current state of DIP switch 6 =0:Dip switch is OFF, =1: Dip switch is ON
0x00031 1x00031 I:30 R/O DIP7	Current state of DIP switch 7 =0:Dip switch is OFF, =1: Dip switch is ON
0x00032 1x00032 I:31 R/O DIP8	Current state of DIP switch 8 =0:Dip switch is OFF, =1: Dip switch is ON
0x00100 1x00100 I:99 R/W RESET COUNTER	Reset, resetting the internal edge counters for all 16 digital inputs. While reading always 0.

10.14.2 Table of holding/input registers

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.

Register	Description
4x00001 3x00001 I:0 R/O RISE DI1	Counter for rising edges on the digital input DI1. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00002 3x00002 I:1 R/O FALL DI1	Counter for falling edges on the digital input DI1. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00003 3x00003 I:2 R/O RISE DI2	Counter for rising edges on the digital input DI2. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00004 3x00004 I:3 R/O FALL DI2	Counter for falling edges on the digital input DI2. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00005 3x00005 I:4 R/O RISE DI3	Counter for rising edges on the digital input DI3. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00006 3x00006 I:5 R/O FALL DI3	Counter for falling edges on the digital input DI3. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00007 3x00007 I:6 R/O RISE DI4	Counter for rising edges on the digital input DI4. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00008 3x00008 I:7 R/O FALL DI4	Counter for falling edges on the digital input DI4. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00009 3x00009 I:8 R/O RISE DI5	Counter for rising edges on the digital input DI5. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00010 3x00010 I:9 R/O FALL DI5	Counter for falling edges on the digital input DI5. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00011 3x00011 I:10 R/O RISE DI6	Counter for rising edges on the digital input DI6. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00012 3x00012 I:11 R/O FALL DI6	Counter for falling edges on the digital input DI6. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.

Register	Description
4x00013 3x00013 I:12 R/O RISE DI7	Counter for rising edges on the digital input DI7. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00014 3x00014 I:13 R/O FALL DI7	Counter for falling edges on the digital input DI7. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00015 3x00015 I:14 R/O RISE DI8	Counter for rising edges on the digital input DI8. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00016 3x00016 I:15 R/O FALL DI8	Counter for falling edges on the digital input DI8. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00017 3x00017 I:16 R/O RISE DI9	Counter for rising edges on the digital input DI9. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00018 3x00018 I:17 R/O FALL DI9	Counter for falling edges on the digital input DI9. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00019 3x00019 I:18 R/O RISE DI10	Counter for rising edges on the digital input DI10. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00020 3x00020 I:19 R/O FALL DI10	Counter for falling edges on the digital input DI10. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00021 3x00021 I:20 R/O RISE DI11	Counter for rising edges on the digital input DI11. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00022 3x00022 I:21 R/O FALL DI11	Counter for falling edges on the digital input DI11. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00023 3x00023 I:22 R/O RISE DI12	Counter for rising edges on the digital input DI12. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00024 3x00024 I:23 R/O FALL DI12	Counter for falling edges on the digital input DI12. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.

Register	Description
4x00025 3x00025 I:24 R/O RISE DI13	Counter for rising edges on the digital input DI13. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00026 3x00026 I:25 R/O FALL DI13	Counter for falling edges on the digital input DI13. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00027 3x00027 I:26 R/O RISE DI14	Counter for rising edges on the digital input DI14. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00028 3x00028 I:27 R/O FALL DI14	Counter for falling edges on the digital input DI14. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00029 3x00029 I:28 R/O RISE DI15	Counter for rising edges on the digital input DI15. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00030 3x00030 I:29 R/O FALL DI15	Counter for falling edges on the digital input DI15. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00031 3x00031 I:30 R/O RISE DI16	Counter for rising edges on the digital input DI16. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00032 3x00032 I:31 R/O FALL DI16	Counter for falling edges on the digital input DI16. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00100 3x00100 I:99 R/W RESET COUNTER	Reset, resetting the internal edge counters for all 16 digital inputs. While reading always 0.

Register	Description
4x00101 3x00101 I:100 R/O DIS	Current state of all digital inputs Bit 0: =0:DI1 is OFF, =1:DI1 is ON Bit 1: =0:DI2 is OFF, =1:DI2 is ON Bit 2: =0:DI3 is OFF, =1:DI3 is ON Bit 3: =0:DI4 is OFF, =1:DI4 is ON Bit 4: =0:DI5 is OFF, =1:DI5 is ON Bit 5: =0:DI6 is OFF, =1:DI6 is ON Bit 6: =0:DI7 is OFF, =1:DI7 is ON Bit 7: =0:DI8 is OFF, =1:DI8 is ON Bit 8: =0:DI9 is OFF, =1:DI9 is ON Bit 9: =0:DI10 is OFF, =1:DI10 is ON Bit 10: =0:DI11 is OFF, =1:DI11 is ON Bit 11: =0:DI12 is OFF, =1:DI12 is ON Bit 12: =0:DI13 is OFF, =1:DI13 is ON Bit 13: =0:DI14 is OFF, =1:DI14 is ON Bit 14: =0:DI15 is OFF, =1:DI15 is ON Bit 15: =0:DI16 is OFF, =1:DI16 is ON
4x00102 3x00102 I:101 R/W ROS	Current state of all relay outputs Bit 0: =0:RO1 is OFF, =1:RO1 is ON Bit 1: =0:RO2 is OFF, =1:RO2 is ON Bit 2: =0:RO3 is OFF, =1:RO3 is ON Bit 3: =0:RO4 is OFF, =1:RO4 is ON Bit 4: =0:RO5 is OFF, =1:RO5 is ON Bit 5: =0:RO6 is OFF, =1:RO6 is ON Bit 6: =0:RO7 is OFF, =1:RO7 is ON Bit 7: =0:RO8 is OFF, =1:RO8 is ON Bit 8-15: always 0 Write on this register sets all eight relay to a new state
4x00103 3x00103 I:102 R/O DIP	Current state of the DIP switch 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) Bit 8-15: always 0

Register	Description
4x00201-4x00202 3x00201-3x00202 I:200-201 R/O COUNTER RO1	Current counter of switching cycles for relay output RO1 Data format: 32 bit unsigned integer with 32-Bit format 0x12345678 -> register 0:0x1234 register 1:0x5678
4x00203-4x00204 3x00203-3x00204 I:202-203 R/O COUNTER RO2	Current counter of switching cycles for relay output RO2 Data format: 32 bit unsigned integer with 32-Bit format 0x12345678 -> register 0:0x1234 register 1:0x5678
4x00205-4x00206 3x00205-3x00206 I:204-205 R/O COUNTER RO3	Current counter of switching cycles for relay output RO3 Data format: 32 bit unsigned integer with 32-Bit format 0x12345678 -> register 0:0x1234 register 1:0x5678
4x00207-4x00208 3x00207-3x00208 I:206-207 R/O COUNTER RO4	Current counter of switching cycles for relay output RO4 Data format: 32 bit unsigned integer with 32-Bit format 0x12345678 -> register 0:0x1234 register 1:0x5678
4x00209-4x00210 3x00209-3x00210 I:208-209 R/O COUNTER RO5	Current counter of switching cycles for relay output RO5 Data format: 32 bit unsigned integer with 32-Bit format 0x12345678 -> register 0:0x1234 register 1:0x5678
4x00211-4x00212 3x00211-3x00212 I:210-211 R/O COUNTER RO6	Current counter of switching cycles for relay output RO6 Data format: 32 bit unsigned integer with 32-Bit format 0x12345678 -> register 0:0x1234 register 1:0x5678
4x00213-4x00214 3x00213-3x00214 I:212-213 R/O COUNTER RO7	Current counter of switching cycles for relay output RO7 Data format: 32 bit unsigned integer with 32-Bit format 0x12345678 -> register 0:0x1234 register 1:0x5678
4x00215-4x00216 3x00215-3x00216 I:214-215 R/O COUNTER RO8	Current counter of switching cycles for relay output RO8 Data format: 32 bit unsigned integer with 32-Bit format 0x12345678 -> register 0:0x1234 register 1:0x5678

Register	Description
4x00221-4x00222 3x00221-3x00222 I:220-221 COUNTER RO1	Current counter of switching cycles for relay output RO1 Data format: 32 bit unsigned integer with 32-Bit format 0x12345678 -> register 0:0x5678 register 1:0x1234
4x00223-4x00224 3x00223-3x00224 I:222-223 COUNTER RO2	Current counter of switching cycles for relay output RO2 Data format: 32 bit unsigned integer with 32-Bit format 0x12345678 -> register 0:0x5678 register 1:0x1234
4x00225-4x00226 3x00225-3x00226 I:224-225 COUNTER RO3	Current counter of switching cycles for relay output RO3 Data format: 32 bit unsigned integer with 32-Bit format 0x12345678 -> register 0:0x5678 register 1:0x1234
4x00227-4x00228 3x00227-3x00228 I:226-227 COUNTER RO4	Current counter of switching cycles for relay output RO4 Data format: 32 bit unsigned integer with 32-Bit format 0x12345678 -> register 0:0x5678 register 1:0x1234
4x00229-4x00230 3x00229-3x00230 I:228-229 COUNTER RO5	Current counter of switching cycles for relay output RO5 Data format: 32 bit unsigned integer with 32-Bit format 0x12345678 -> register 0:0x5678 register 1:0x1234
4x00231-4x00232 3x00231-3x00232 I:230-231 COUNTER RO6	Current counter of switching cycles for relay output RO6 Data format: 32 bit unsigned integer with 32-Bit format 0x12345678 -> register 0:0x5678 register 1:0x1234
4x00233-4x00234 3x00233-3x00234 I:232-233 COUNTER RO7	Current counter of switching cycles for relay output RO7 Data format: 32 bit unsigned integer with 32-Bit format 0x12345678 -> register 0:0x5678 register 1:0x1234
4x00235-4x00236 3x00235-3x00236 I:234-235 COUNTER RO8	Current counter of switching cycles for relay output RO8 Data format: 32 bit unsigned integer with 32-Bit format 0x12345678 -> register 0:0x5678 register 1:0x1234

Register	Description
4x20001 3x20001 I:20000 SWITCH RO1	Mapping table for the output relay RO1 with all 16 digital inputs of the module for the logic function SWITCH. SWITCH function: Is the mapped digital input high (1), the relay output will be turned on (set to 1). Is the mapped digital input low (0), the relay output will be turned off (set to 0). It makes sense to map only one digital input to a relay output, because this function works with the current state of the digital input, not with a rising edge like the other logic functions. Each bit stand for one digital input: Bit 0=DI1, Bit 1=DI2,…Bit 15=DI16. If the bit is 1, the digital input will be used in this logic function. If this bit is 0, the output will be ignored. Writing on this register saves the mapping in the non-volatile internal FRAM memory.
4x20002 3x20002 I:20001 SWITCH RO2	Mapping table for the output relay RO2 with all 16 digital inputs of the module for the logic function SWITCH. Refer to SWITCH RO1
4x20003 3x20003 I:20002 SWITCH RO3	Mapping table for the output relay RO3 with all 16 digital inputs of the module for the logic function SWITCH. Refer to SWITCH RO1
4x20004 3x20004 I:20003 SWITCH RO4	Mapping table for the output relay RO4 with all 16 digital inputs of the module for the logic function SWITCH. Refer to SWITCH RO1
4x20005 3x20005 I:20004 SWITCH RO5	Mapping table for the output relay RO5 with all 16 digital inputs of the module for the logic function SWITCH. Refer to SWITCH RO1
4x20006 3x20006 I:20005 SWITCH RO6	Mapping table for the output relay RO6 with all 16 digital inputs of the module for the logic function SWITCH. Refer to SWITCH RO1
4x20007 3x20007 I:20006 SWITCH RO7	Mapping table for the output relay RO7 with all 16 digital inputs of the module for the logic function SWITCH. Refer to SWITCH RO1
4x20008 3x20008 I:20007 SWITCH RO8	Mapping table for the output relay RO8 with all 16 digital inputs of the module for the logic function SWITCH. Refer to SWITCH RO1

Register	Description
4x20009 3x20009 I:20008 TOGGLE RO1	Mapping table for the output relay RO1 with all 16 digital inputs of the module for the logic function TOGGLE. TOGGLE function: If the module detects a rising edge on one of the mapped digital inputs, it inverts the current state of the corresponding relay output. Each bit stand for one digital input: Bit 0=DI1, Bit 1=DI2,…Bit 15=DI16. If the bit is 1, the digital input will be used in this logic function. If this bit is 0, the output will be ignored. Writing on this register saves the mapping in the non-volatile internal FRAM memory.
4x20010 3x20010 I:20009 TOGGLE RO2	Mapping table for the output relay RO2 with all 16 digital inputs of the module for the logic function SWITCH. Refer to TOGGLE RO1
4x20011 3x20011 I:20010 TOGGLE RO3	Mapping table for the output relay RO3 with all 16 digital inputs of the module for the logic function SWITCH. Refer to TOGGLE RO1
4x20012 3x20012 I:20011 TOGGLE RO4	Mapping table for the output relay RO4 with all 16 digital inputs of the module for the logic function SWITCH. Refer to TOGGLE RO1
4x20013 3x20013 I:20012 TOGGLE RO5	Mapping table for the output relay RO5 with all 16 digital inputs of the module for the logic function SWITCH. Refer to TOGGLE RO1
4x20014 3x20014 I:20013 TOGGLE RO6	Mapping table for the output relay RO6 with all 16 digital inputs of the module for the logic function SWITCH. Refer to TOGGLE RO1
4x20015 3x20015 I:20014 TOGGLE RO7	Mapping table for the output relay RO7 with all 16 digital inputs of the module for the logic function SWITCH. Refer to TOGGLE RO1
4x20016 3x20016 I:20015 TOGGLE RO8	Mapping table for the output relay RO8 with all 16 digital inputs of the module for the logic function SWITCH. Refer to TOGGLE RO1

Register	Description
4x20017 3x20017 I:20016 SWITCH ON RO1	Mapping table for the output relay RO1 with all 16 digital inputs of the module for the logic function SWITCH ON. SWITCH ON function: If the module detects a rising edge on one of the mapped digital inputs, it sets the corresponding relay output to high (1). Each bit stand for one digital input: Bit 0=DI1, Bit 1=DI2,…Bit 15=DI16. If the bit is 1, the digital input will be used in this logic function. If this bit is 0, the output will be ignored. Writing on this register saves the mapping in the non-volatile internal FRAM memory.
4x20018 3x20018 I:20017 SWITCH ON RO2	Mapping table for the output relay RO2 with all 16 digital inputs of the module for the logic function SWITCH ON. Refer to SWITCH ON RO1
4x20019 3x20019 I:20018 SWITCH ON RO3	Mapping table for the output relay RO3 with all 16 digital inputs of the module for the logic function SWITCH ON. Refer to SWITCH ON RO1
4x20020 3x20020 I:20019 SWITCH ON RO4	Mapping table for the output relay RO4 with all 16 digital inputs of the module for the logic function SWITCH ON. Refer to SWITCH ON RO1
4x20021 3x20021 I:20020 SWITCH ON RO5	Mapping table for the output relay RO5 with all 16 digital inputs of the module for the logic function SWITCH ON. Refer to SWITCH ON RO1
4x20022 3x20022 I:20021 SWITCH ON RO6	Mapping table for the output relay RO6 with all 16 digital inputs of the module for the logic function SWITCH ON. Refer to SWITCH ON RO1
4x20023 3x20023 I:20022 SWITCH ON RO7	Mapping table for the output relay RO7 with all 16 digital inputs of the module for the logic function SWITCH ON. Refer to SWITCH ON RO1
4x20024 3x20024 I:20023 SWITCH ON RO8	Mapping table for the output relay RO8 with all 16 digital inputs of the module for the logic function SWITCH ON. Refer to SWITCH ON RO1

Register	Description
4x20025 3x20025 I:20024 SWITCH OFF RO1	Mapping table for the output relay RO1 with all 16 digital inputs of the module for the logic function SWITCH OFF. SWITCH OFF function: If the module detects a rising edge on one of the mapped digital inputs, it sets the corresponding relay output to low (1). Each bit stand for one digital input: Bit 0=DI1, Bit 1=DI2,…Bit 15=DI16. If the bit is 1, the digital input will be used in this logic function. If this bit is 0, the output will be ignored. Writing on this register saves the mapping in the non-volatile internal FRAM memory.
4x20026 3x20026 I:20025 SWITCH OFF RO2	Mapping table for the output relay RO2 with all 16 digital inputs of the module for the logic function SWITCH OFF. Refer to SWITCH OFF RO1
4x20027 3x20027 I:20026 SWITCH OFF RO3	Mapping table for the output relay RO3 with all 16 digital inputs of the module for the logic function SWITCH OFF. Refer to SWITCH OFF RO1
4x20028 3x20028 I:20027 SWITCH OFF RO4	Mapping table for the output relay RO4 with all 16 digital inputs of the module for the logic function SWITCH OFF. Refer to SWITCH OFF RO1
4x20029 3x20029 I:20028 SWITCH OFF RO5	Mapping table for the output relay RO5 with all 16 digital inputs of the module for the logic function SWITCH OFF. Refer to SWITCH OFF RO1
4x20030 3x20030 I:20029 SWITCH OFF RO6	Mapping table for the output relay RO6 with all 16 digital inputs of the module for the logic function SWITCH OFF. Refer to SWITCH OFF RO1
4x20031 3x20031 I:20030 SWITCH OFF RO7	Mapping table for the output relay RO7 with all 16 digital inputs of the module for the logic function SWITCH OFF. Refer to SWITCH OFF RO1
4x20032 3x20032 I:20031 SWITCH OFF RO8	Mapping table for the output relay RO8 with all 16 digital inputs of the module for the logic function SWITCH OFF. Refer to SWITCH OFF RO1

Register	Description
4x20033 3x20033 I:20032 PULSE RO1	Mapping table for the output relay RO1 with all 16 digital inputs of the module for the logic function PULSE. PULSE function: If the module detects a rising edge on one of the mapped digital inputs, it switches the corresponding relay output x on for a defined PULSE TIME. This time is individual for each relay output. After the time has passed by, the relay output is switched to low automatically (0). Each bit stand for one digital input: Bit 0=DI1, Bit 1=DI2,…Bit 15=DI16. If the bit is 1, the digital input will be used in this logic function. If this bit is 0, the output will be ignored. Writing on this register saves the mapping in the non-volatile internal FRAM memory.
4x20034 3x20034 I:20033 PULSE RO2	Mapping table for the output relay RO2 with all 16 digital inputs of the module for the logic function PULSE. Refer to PULSE RO1
4x20035 3x20035 I:20034 PULSE RO3	Mapping table for the output relay RO3 with all 16 digital inputs of the module for the logic function PULSE. Refer to PULSE RO1
4x20036 3x20036 I:20035 PULSE RO4	Mapping table for the output relay RO4 with all 16 digital inputs of the module for the logic function PULSE. Refer to PULSE RO1
4x20037 3x20037 I:20036 PULSE RO5	Mapping table for the output relay RO5 with all 16 digital inputs of the module for the logic function PULSE. Refer to PULSE RO1
4x20038 3x20038 I:20037 PULSE RO6	Mapping table for the output relay RO6 with all 16 digital inputs of the module for the logic function PULSE. Refer to PULSE RO1
4x20039 3x20039 I:20038 PULSE RO7	Mapping table for the output relay RO7 with all 16 digital inputs of the module for the logic function PULSE. Refer to PULSE RO1
4x20040 3x20040 I:20039 PULSE RO8	Mapping table for the output relay RO8 with all 16 digital inputs of the module for the logic function PULSE. Refer to PULSE RO1

Register	Description
4x20065-4x20066 3x20065-3x20066 I:20064-20065 R/W PULSE TIME RO1	Current time for pulse length in logic function pulse in 1/10s for relay output RO1. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x1234 register 1:0x5678 Writing a 32 bit value on both registers stores the new time into the non-volatile FRAM memory permanently.
4x20067-4x20068 3x20067-3x20068 I:20066-20067 R/W PULSE TIME RO2	Current time for pulse length in logic function pulse in 1/10s for relay output RO2. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x1234 register 1:0x5678 Writing a 32 bit value on both registers stores the new time into the non-volatile FRAM memory permanently.
4x20069-4x20070 3x20069-3x20070 I:20068-20069 R/W PULSE TIME RO3	Current time for pulse length in logic function pulse in 1/10s for relay output RO3. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x1234 register 1:0x5678 Writing a 32 bit value on both registers stores the new time into the non-volatile FRAM memory permanently.
4x20071-4x20072 3x20071-3x20072 I:20070-20071 R/W PULSE TIME RO4	Current time for pulse length in logic function pulse in 1/10s for relay output RO4. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x1234 register 1:0x5678 Writing a 32 bit value on both registers stores the new time into the non-volatile FRAM memory permanently.
4x20073-4x20074 3x20073-3x20074 I:20072-20073 R/W PULSE TIME RO5	Current time for pulse length in logic function pulse in 1/10s for relay output RO5. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x1234 register 1:0x5678 Writing a 32 bit value on both registers stores the new time into the non-volatile FRAM memory permanently.
4x20075-4x20076 3x20075-3x20076 I:20074-20075 R/W PULSE TIME RO6	Current time for pulse length in logic function pulse in 1/10s for relay output RO6. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x1234 register 1:0x5678 Writing a 32 bit value on both registers stores the new time into the non-volatile FRAM memory permanently.
4x20077-4x20078 3x20077-3x20078 I:20076-20077 R/W PULSE TIME RO7	Current time for pulse length in logic function pulse in 1/10s for relay output RO7. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x1234 register 1:0x5678 Writing a 32 bit value on both registers stores the new time into the non-volatile FRAM memory permanently.
4x20079-4x20080 3x20079-3x20080 I:20078-20079 R/W PULSE TIME RO8	Current time for pulse length in logic function pulse in 1/10s for relay output RO8. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x1234 register 1:0x5678 Writing a 32 bit value on both registers stores the new time into the non-volatile FRAM memory permanently.

Register	Description
4x20081-4x20082 3x20081-3x20082 I:20080-20081 R/W PULSE TIME RO1	Current time for pulse length in logic function pulse in 1/10s for relay output RO1. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x5678 register 1:0x1234 Writing a 32 bit value on both registers stores the new time into the non-volatile FRAM memory permanently.
4x20083-4x20084 3x20083-3x20084 I:20082-20083 R/W PULSE TIME RO2	Current time for pulse length in logic function pulse in 1/10s for relay output RO2. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x5678 register 1:0x1234 Writing a 32 bit value on both registers stores the new time into the non-volatile FRAM memory permanently.
4x20085-4x20086 3x20085-3x20086 I:20084-20085 R/W PULSE TIME RO3	Current time for pulse length in logic function pulse in 1/10s for relay output RO3. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x5678 register 1:0x1234 Writing a 32 bit value on both registers stores the new time into the non-volatile FRAM memory permanently.
4x20087-4x20088 3x20087-3x20088 I:20086-20087 R/W PULSE TIME RO4	Current time for pulse length in logic function pulse in 1/10s for relay output RO4. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x5678 register 1:0x1234 Writing a 32 bit value on both registers stores the new time into the non-volatile FRAM memory permanently.
4x20089-4x20090 3x20089-3x20090 I:20088-20089 R/W PULSE TIME RO5	Current time for pulse length in logic function pulse in 1/10s for relay output RO5. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x5678 register 1:0x1234 Writing a 32 bit value on both registers stores the new time into the non-volatile FRAM memory permanently.
4x20091-4x20092 3x20091-3x20092 I:20090-20091 R/W PULSE TIME RO6	Current time for pulse length in logic function pulse in 1/10s for relay output RO6. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x5678 register 1:0x1234 Writing a 32 bit value on both registers stores the new time into the non-volatile FRAM memory permanently.
4x20093-4x20094 3x20093-3x20094 I:20092-20093 R/W PULSE TIME RO7	Current time for pulse length in logic function pulse in 1/10s for relay output RO7. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x5678 register 1:0x1234 Writing a 32 bit value on both registers stores the new time into the non-volatile FRAM memory permanently.
4x20095-4x20096 3x20095-3x20096 I:20094-20095 R/W PULSE TIME RO8	Current time for pulse length in logic function pulse in 1/10s for relay output RO8. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x5678 register 1:0x1234 Writing a 32 bit value on both registers stores the new time into the non-volatile FRAM memory permanently.

Register	Description
4x20097-4x20098 3x20097-3x20098 I:20096-20097 R/O PULSE TIMER RO1	Current remaining time for the running pulse length in logic function pulse in 1/10s for relay output RO1. If no pulse is running this register is 0. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x1234 register 1:0x5678
4x20099-4x20100 3x20099-3x20100 I:20098-20099 R/O PULSE TIMER RO2	Current remaining time for the running pulse length in logic function pulse in 1/10s for relay output RO2. If no pulse is running this register is 0. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x1234 register 1:0x5678
4x20101-4x20102 3x20101-3x20102 I:20100-20101 R/O PULSE TIMER RO3	Current remaining time for the running pulse length in logic function pulse in 1/10s for relay output RO3. If no pulse is running this register is 0. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x1234 register 1:0x5678
4x20103-4x20104 3x20103-3x20104 I:20102-20103 R/O PULSE TIMER RO4	Current remaining time for the running pulse length in logic function pulse in 1/10s for relay output RO4. If no pulse is running this register is 0. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x1234 register 1:0x5678
4x20105-4x20106 3x20105-3x20106 I:20104-20105 R/O PULSE TIMER RO5	Current remaining time for the running pulse length in logic function pulse in 1/10s for relay output RO5. If no pulse is running this register is 0. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x1234 register 1:0x5678
4x20107-4x20108 3x20107-3x20108 I:20106-20107 R/O PULSE TIMER RO6	Current remaining time for the running pulse length in logic function pulse in 1/10s for relay output RO6. If no pulse is running this register is 0. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x1234 register 1:0x5678
4x20109-4x20110 3x20109-3x20110 I:20108-20109 R/O PULSE TIMER RO7	Current remaining time for the running pulse length in logic function pulse in 1/10s for relay output RO7. If no pulse is running this register is 0. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x1234 register 1:0x5678
4x20111-4x20112 3x20111-3x20112 I:20110-20111 R/O PULSE TIMER RO8	Current remaining time for the running pulse length in logic function pulse in 1/10s for relay output RO8. If no pulse is running this register is 0. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x1234 register 1:0x5678

Register	Description
4x20113-4x20114 3x20113-3x20114 I:20112-20113 R/O PULSE TIMER RO1	Current remaining time for the running pulse length in logic function pulse in 1/10s for relay output RO1. If no pulse is running this register is 0. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x5678 register 1:0x1234
4x20115-4x20116 3x20115-3x20116 I:20114-20115 R/O PULSE TIMER RO2	Current remaining time for the running pulse length in logic function pulse in 1/10s for relay output RO2. If no pulse is running this register is 0. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x5678 register 1:0x1234
4x20117-4x20118 3x20117-3x20118 I:20116-20117 R/O PULSE TIMER RO3	Current remaining time for the running pulse length in logic function pulse in 1/10s for relay output RO3. If no pulse is running this register is 0. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x5678 register 1:0x1234
4x20119-4x20120 3x20119-3x20120 I:20118-20119 R/O PULSE TIMER RO4	Current remaining time for the running pulse length in logic function pulse in 1/10s for relay output RO4. If no pulse is running this register is 0. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x5678 register 1:0x1234
4x20121-4x20122 3x20121-3x20122 I:20120-20121 R/O PULSE TIMER RO5	Current remaining time for the running pulse length in logic function pulse in 1/10s for relay output RO5. If no pulse is running this register is 0. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x5678 register 1:0x1234
4x20123-4x20124 3x20123-3x20124 I:20122-20123 R/O PULSE TIMER RO6	Current remaining time for the running pulse length in logic function pulse in 1/10s for relay output RO6. If no pulse is running this register is 0. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x5678 register 1:0x1234
4x20125-4x20126 3x20125-3x20126 I:20124-20125 R/O PULSE TIMER RO7	Current remaining time for the running pulse length in logic function pulse in 1/10s for relay output RO7. If no pulse is running this register is 0. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x5678 register 1:0x1234
4x20127-4x20128 3x20127-3x20128 I:20126-20127 R/O PULSE TIMER RO8	Current remaining time for the running pulse length in logic function pulse in 1/10s for relay output RO8. If no pulse is running this register is 0. Data format: 32 Bit unsigned integer with storage format 0x12345678 -> register 0:0x5678 register 1:0x1234

Register	Description
4x21001 3x21001 I:21000 R/W ENABLE LOGIC FUNCTIONS	Only if this register contains 1, the internal logic functions are executed. If there is a 0 value in this register, no logic is executed in the module. Writing onto this register stores the new value in the non-volatile FRAM memory.
4x21002 3x21002 I:21001 R/W CLEAR ALL LOGIC FUNCTIONS	Reading this register delivers always the value 0. Writing a value onto this register clears all mapping and configuration tables for logic functions and sets the register ENABLE LOGIC FUNCTIONS to 0. The mapping and configuration tables in the non-volatile FRAM memory are set to 0 permanently! After a reboot of the module. No logic functions are executed and the configuration remains 0.

11 RESI-32DI-MODBUS, RESI-32DI-ASCII

11.1 Product description

This IO module offers the following features:

- 32 digital inputs for 12-48Vdc signals
- Galvanic insulated RS485 interface for communication with a host system
- RESI-32DI-MODBUS: MODBUS/RTU slave protocol
- RESI-32DI-ASCII: MODBUS/RTU slave and text based ASCII protocol
- Power supply 12-48Vdc
- DIP switch for setting the baud rates, the type of interface and the bus number
- LED indicator for the communication
- Mounting onto EN50022 DIN rail or wall mounting

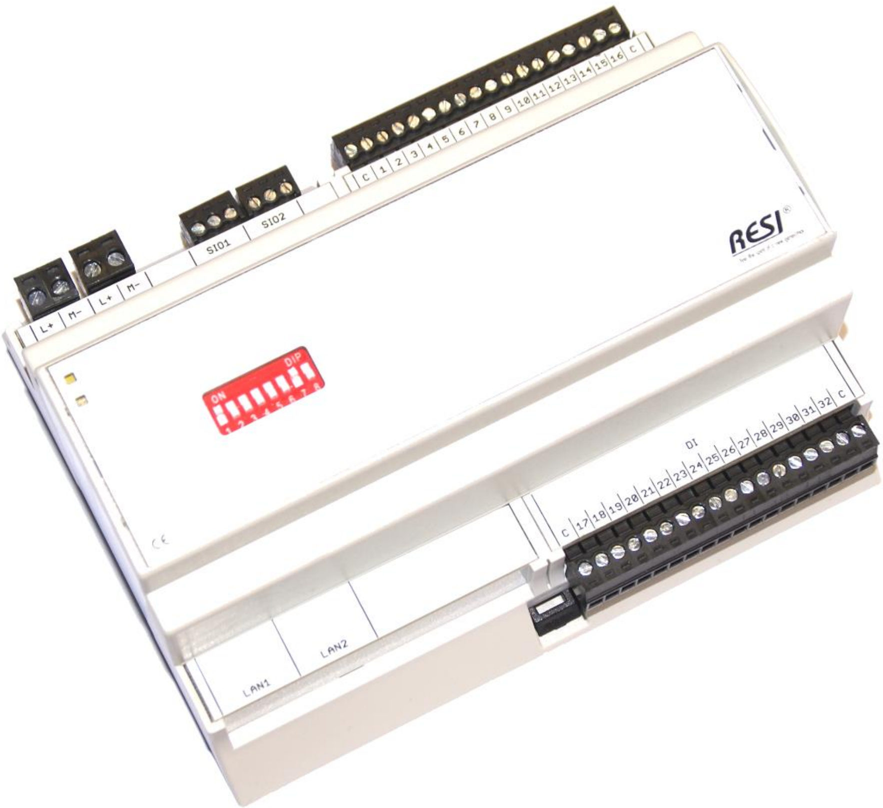


Illustration: Our IO module

11.2 Technical data

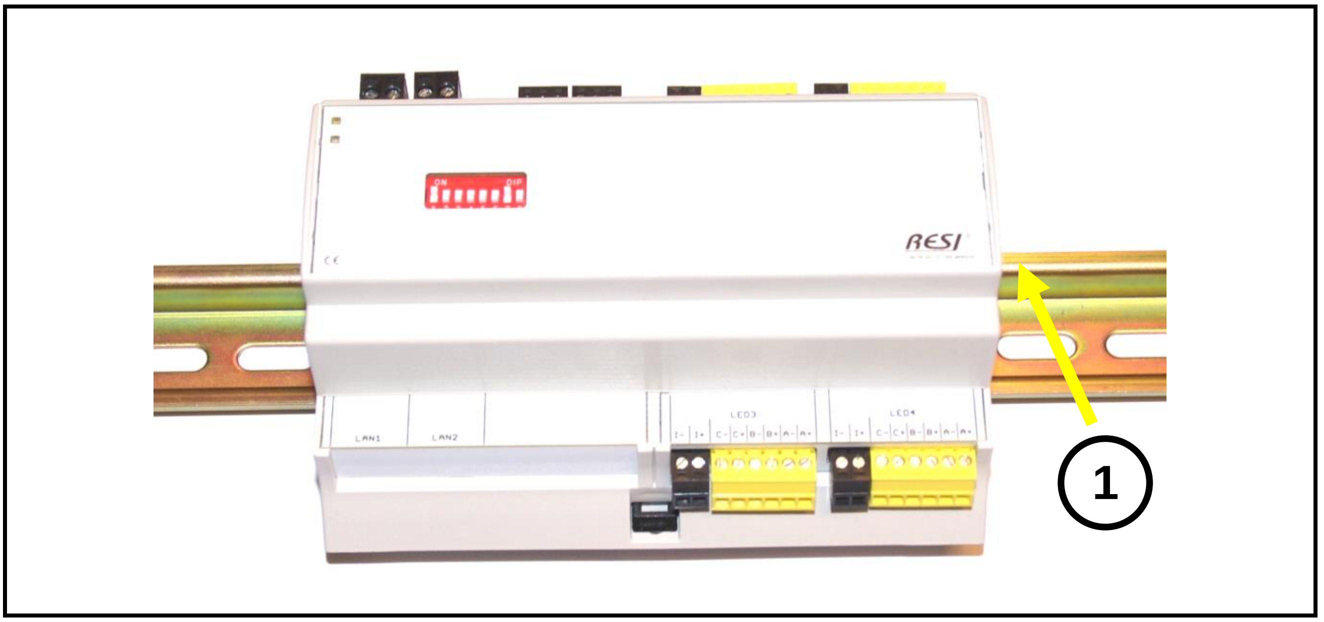
Technical Data			
Power supply			
Supply voltage	12-48V= +/-10%	Storage temperature	-20...85 °C
Power LED	Ja	Operating Temperature	0...60°C
Power consumption	<0.5W	Humidity	25...90 % rH non-condensing
		Protection Class	IP20 (EN 60529)
		Dimensions LxWxH	143mm x 110mm x 62mm
		Weight	260g
		Mounting	On DIN EN50022 rail or wall mounting
ASCII/Modbus Interface			
Protocol	ASCII or Modbus/RTU		
Type	RS485		
Baud rates	4800 to 256000Bd/8/N or E/1		
Cable Connection	Via removable clamps		
LED indicator	Yes		
Galvanic insulation	No		
Digital inputs			
Total amount of inputs	32		
Sampling rate	Every 10ms		
Input voltage range	12-48V= +/-10%		
Input current	approx. 1mA per channel		
Logic levels	0: <3V= 1: >5V=		
Cable connection	Via two 18-pin plug-in terminal blocks		
Galvanic insulation	No		
Clamps			
Clamp wire cross section	Max. 1,5 mm²		
Tightening torque	Max. 0.5Nm	**CE conformity**	Yes

11.3 Assembling

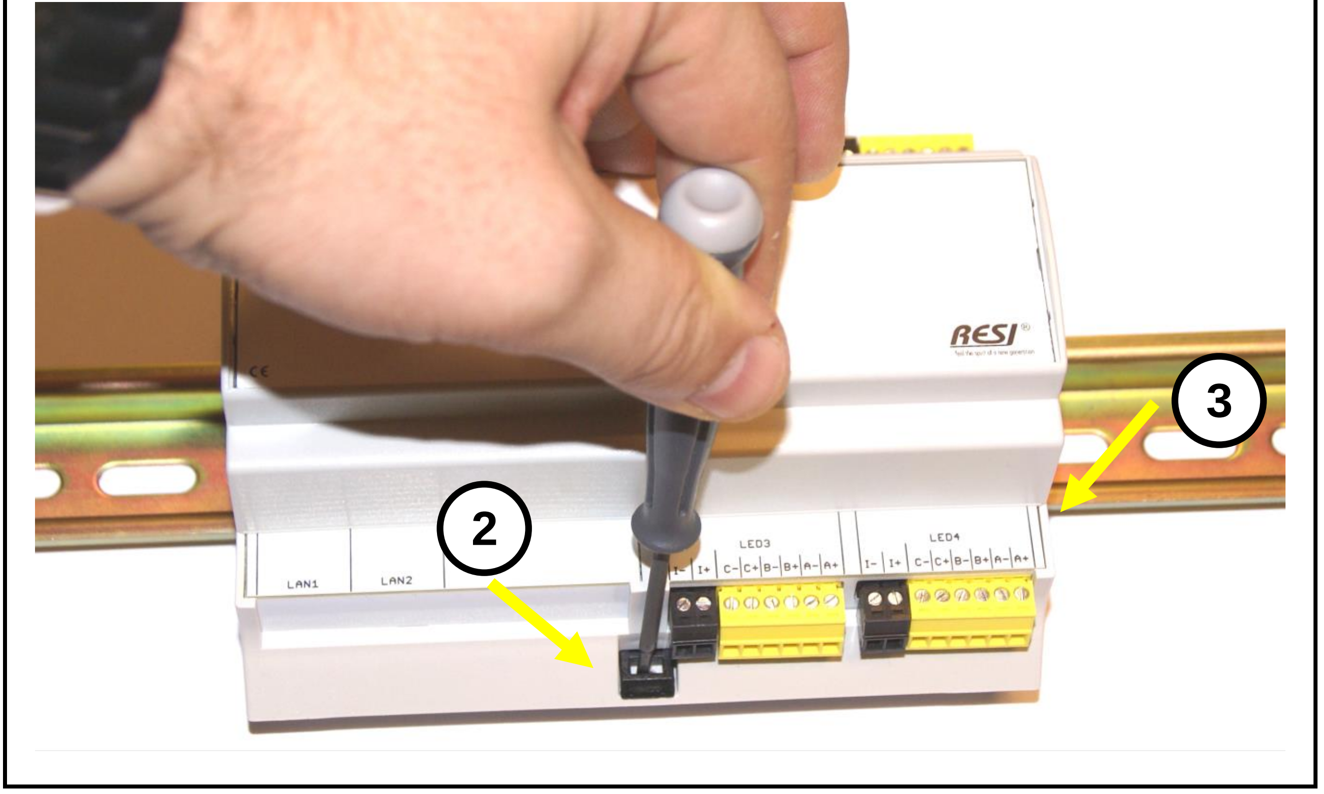
Our IO modules are designed for mounting onto a 35mm DIN-EN50022 rail or for wall mounting. Please not, that in the following mounting description we use only symbolic photos of our IO modules.

11.3.1 Mounting of a DIN EN50022 rail

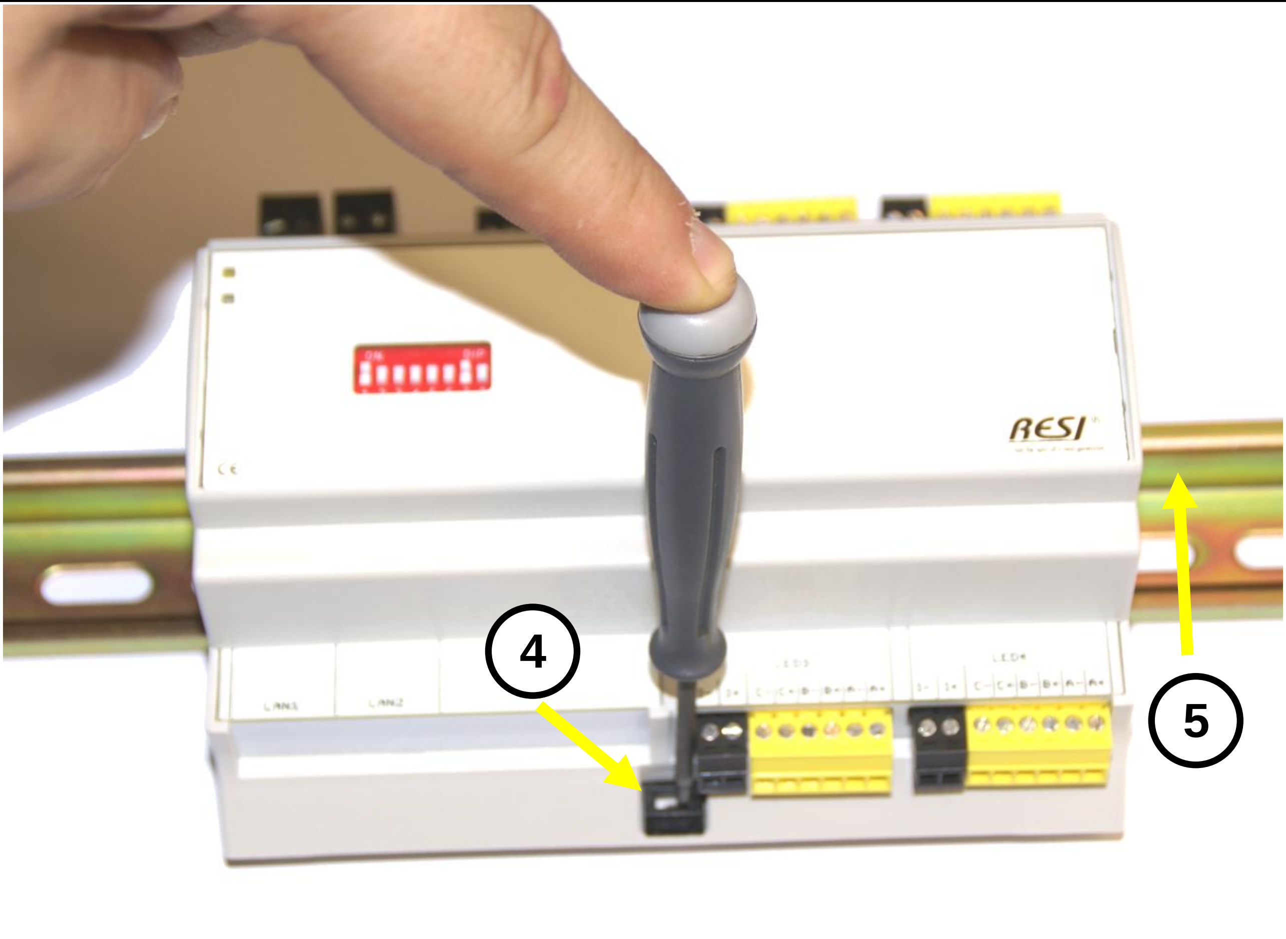
First snap in the top part of the module into the DIN rail (1). The bottom part of the module is not snapped into the DIN rail at this moment.



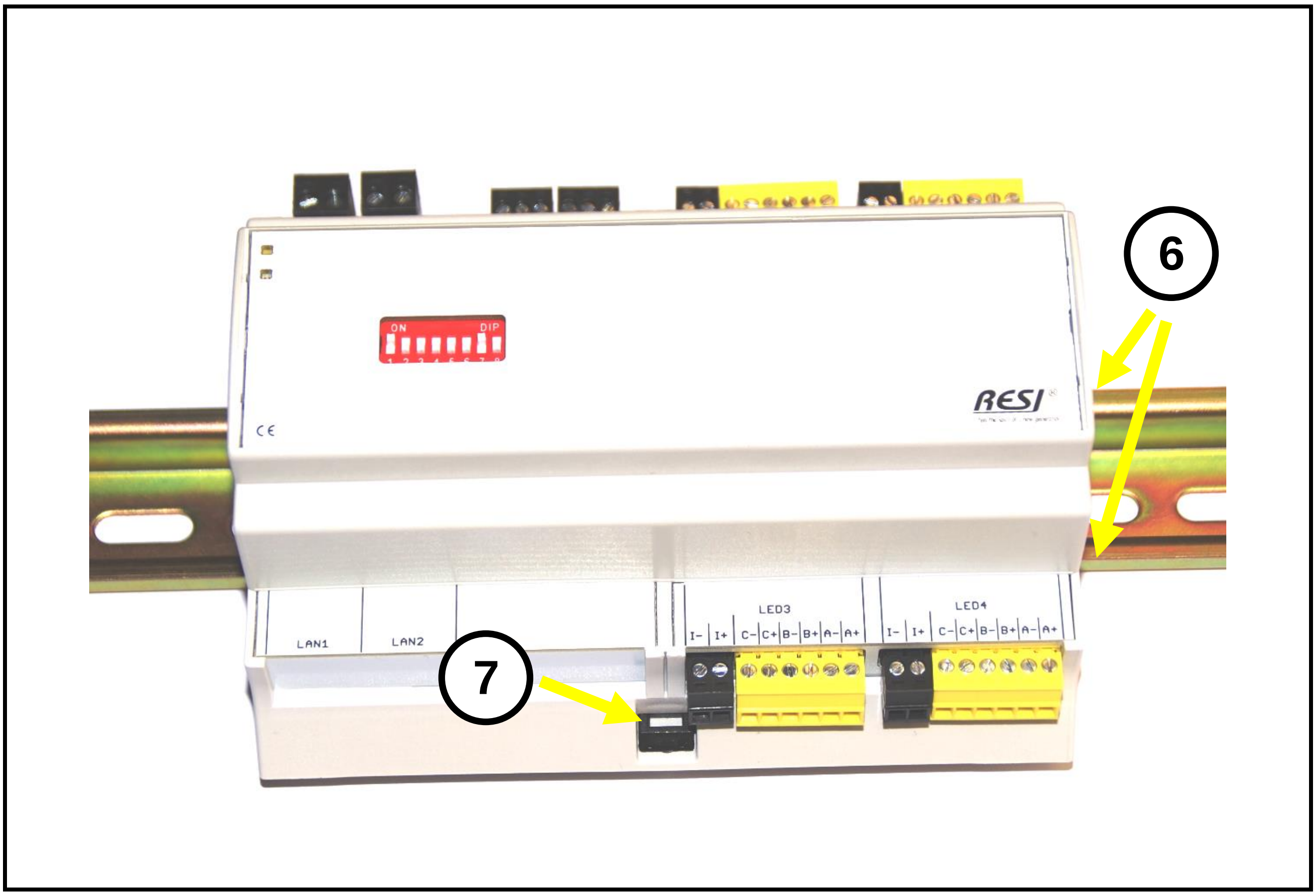
Then open the black hook with a screw driver (2). Now press the module with the opened hook onto the DIN rail until both sides of the module snap into the DIN rail (3). Release the screw driver now. The hook snaps into the DIN rail and the module is now mounted correctly onto the DIN rail.



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.

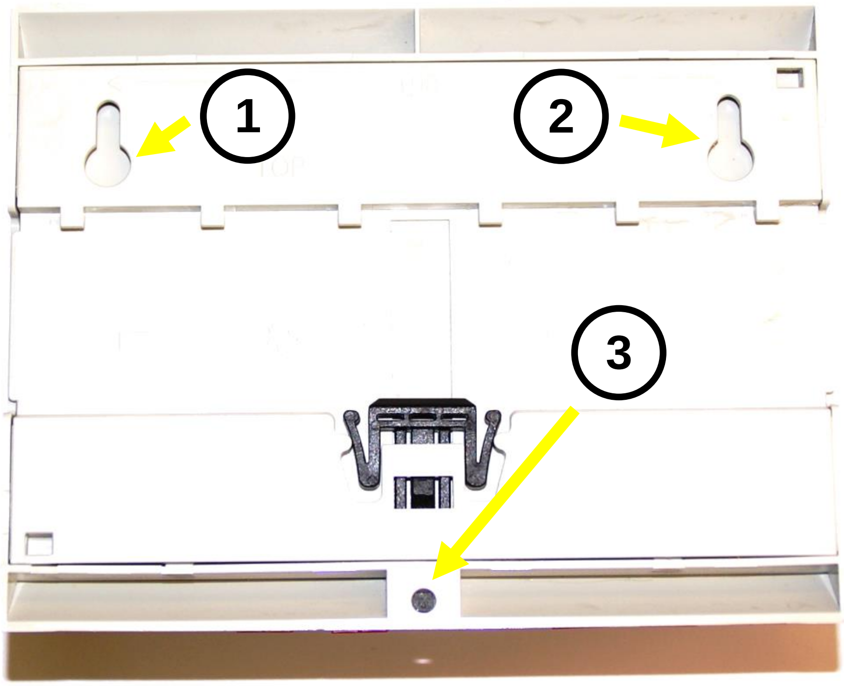


The module is correctly mounted, if the module has snapped into the DIN rail on both sides of the housing (6) and if the hook has snapped in too (7).

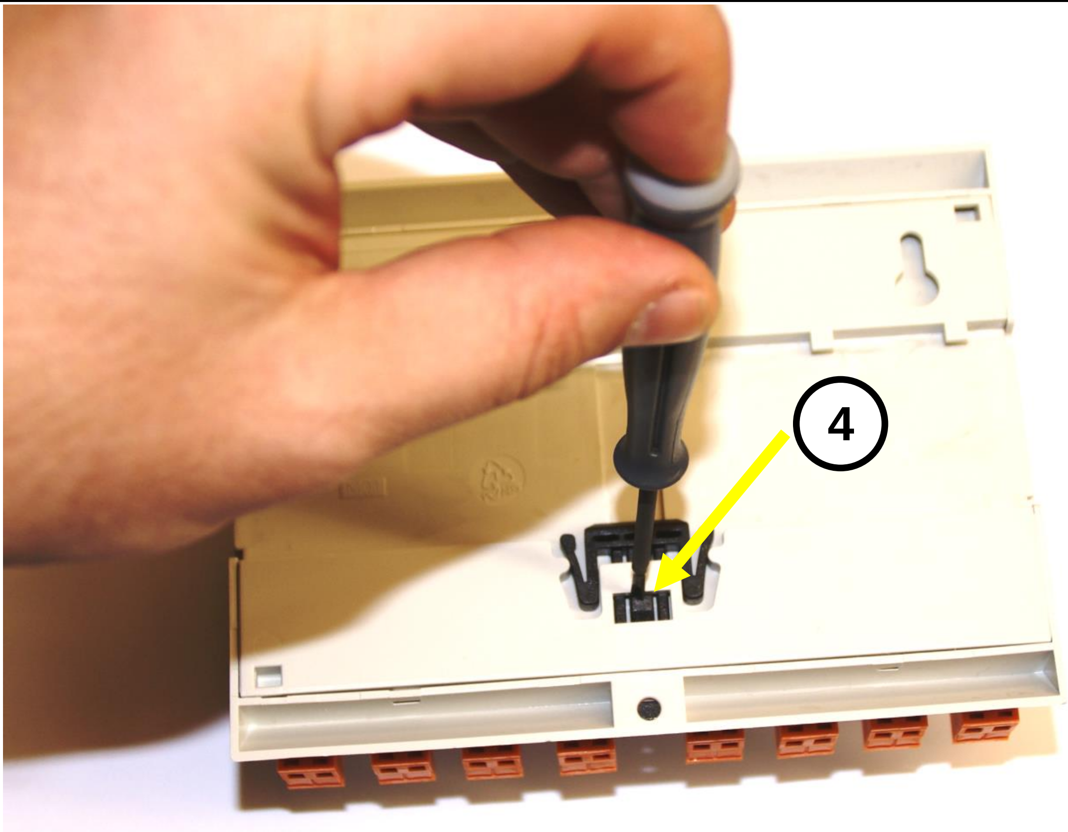


11.3.2 Wall mounting

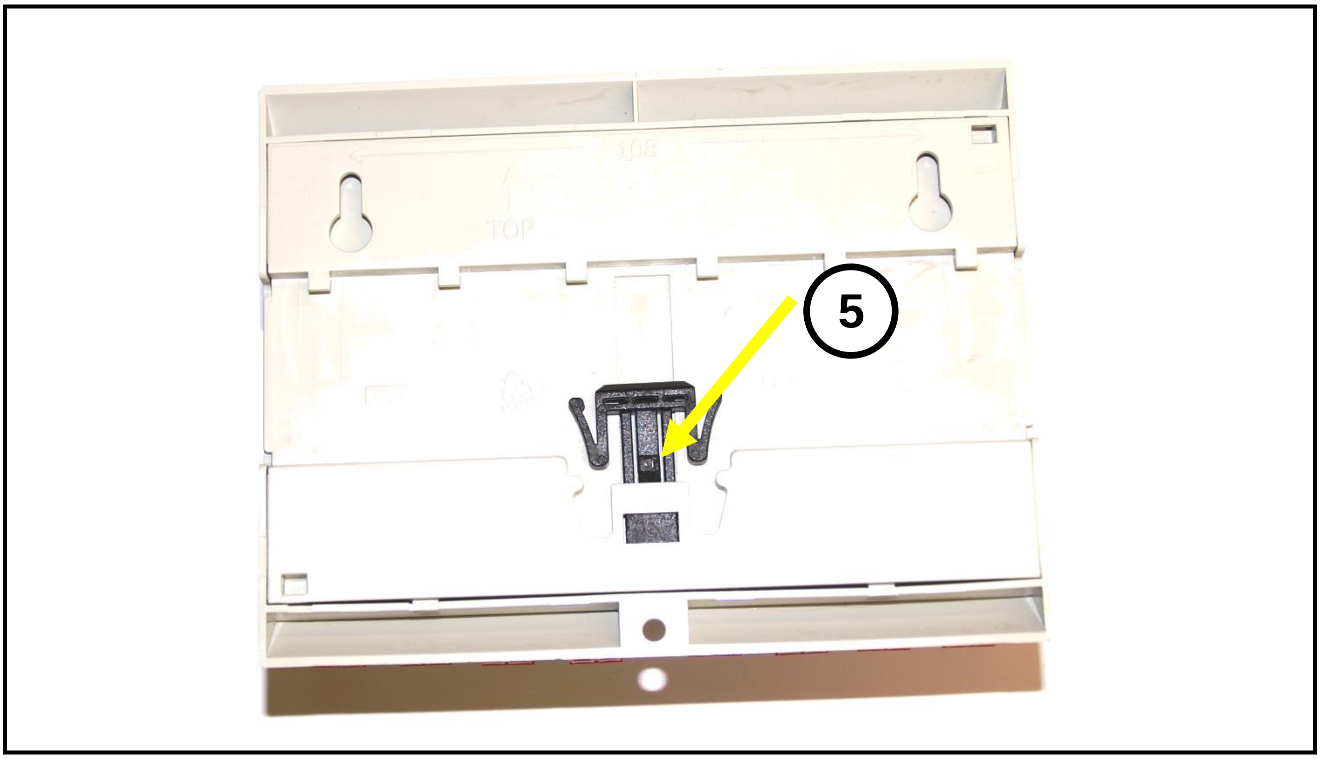
Our modules can also be mounted onto a wall. Turn over the module as shown in the picture below:

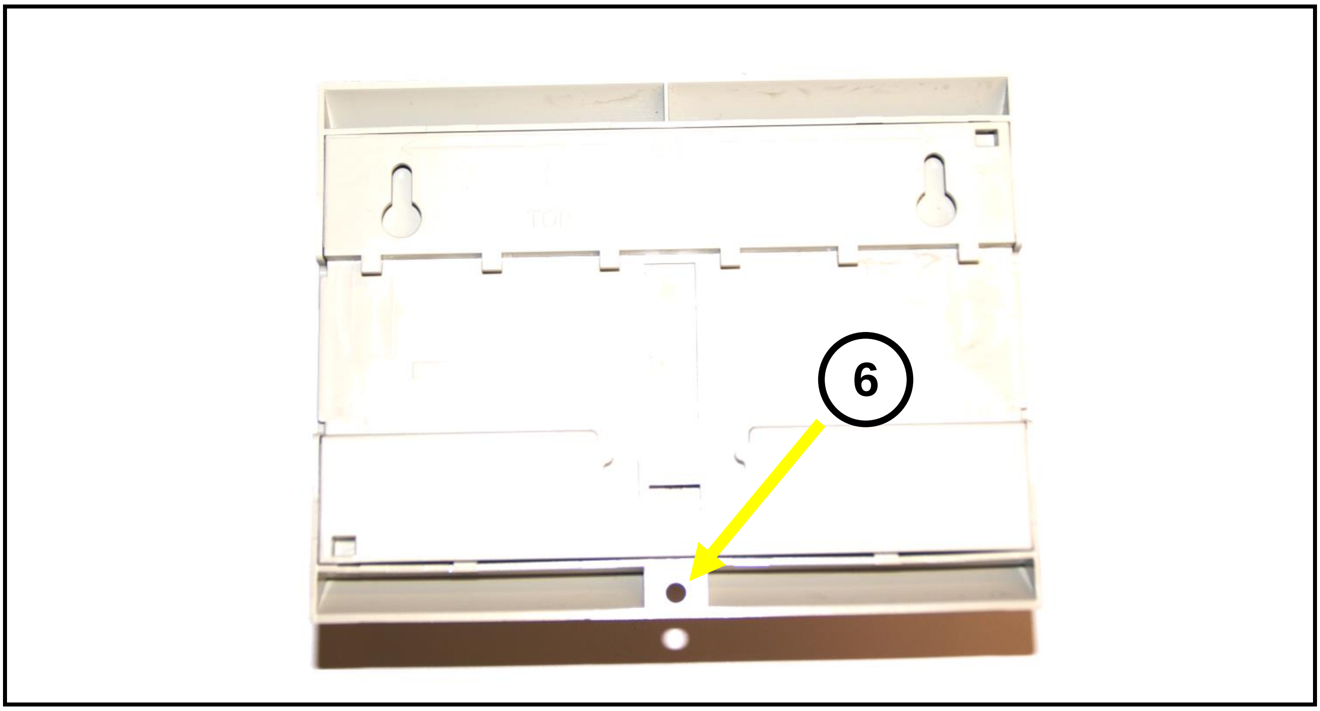


You will notice, that there are two holes for wall hooks or screws on the top side of the housing. (1) and (2). On the bottom side you will notice a small hole for a screw to fix the housing on the wall from the front (3). But first we have to remove the hook, which blocks the screw hole in the housing.



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.

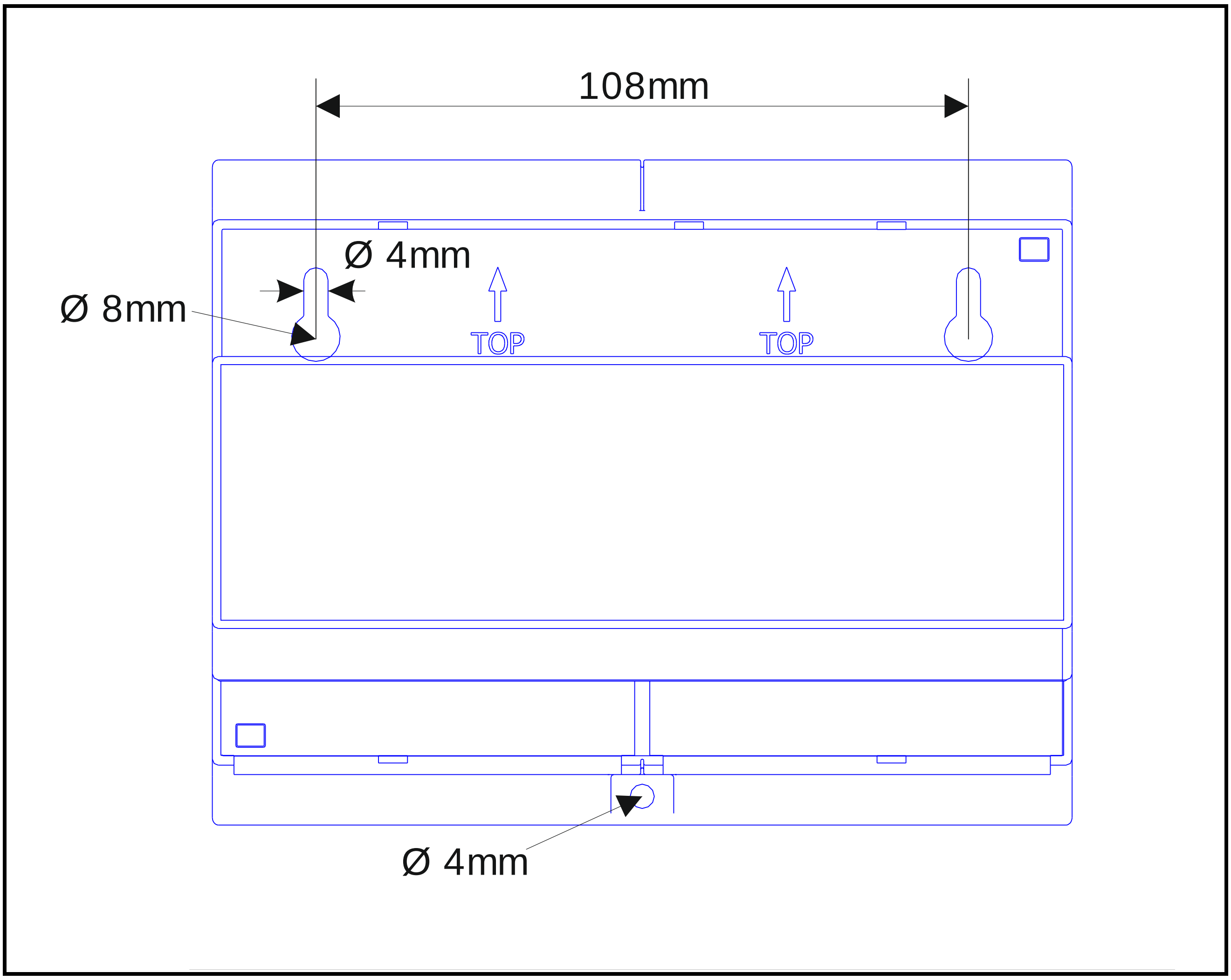


Illustration: Bottom view of the module with holes for wall mounting

11.4 Connection diagram

In the below drawing you will find the correct cabling of the IO module.

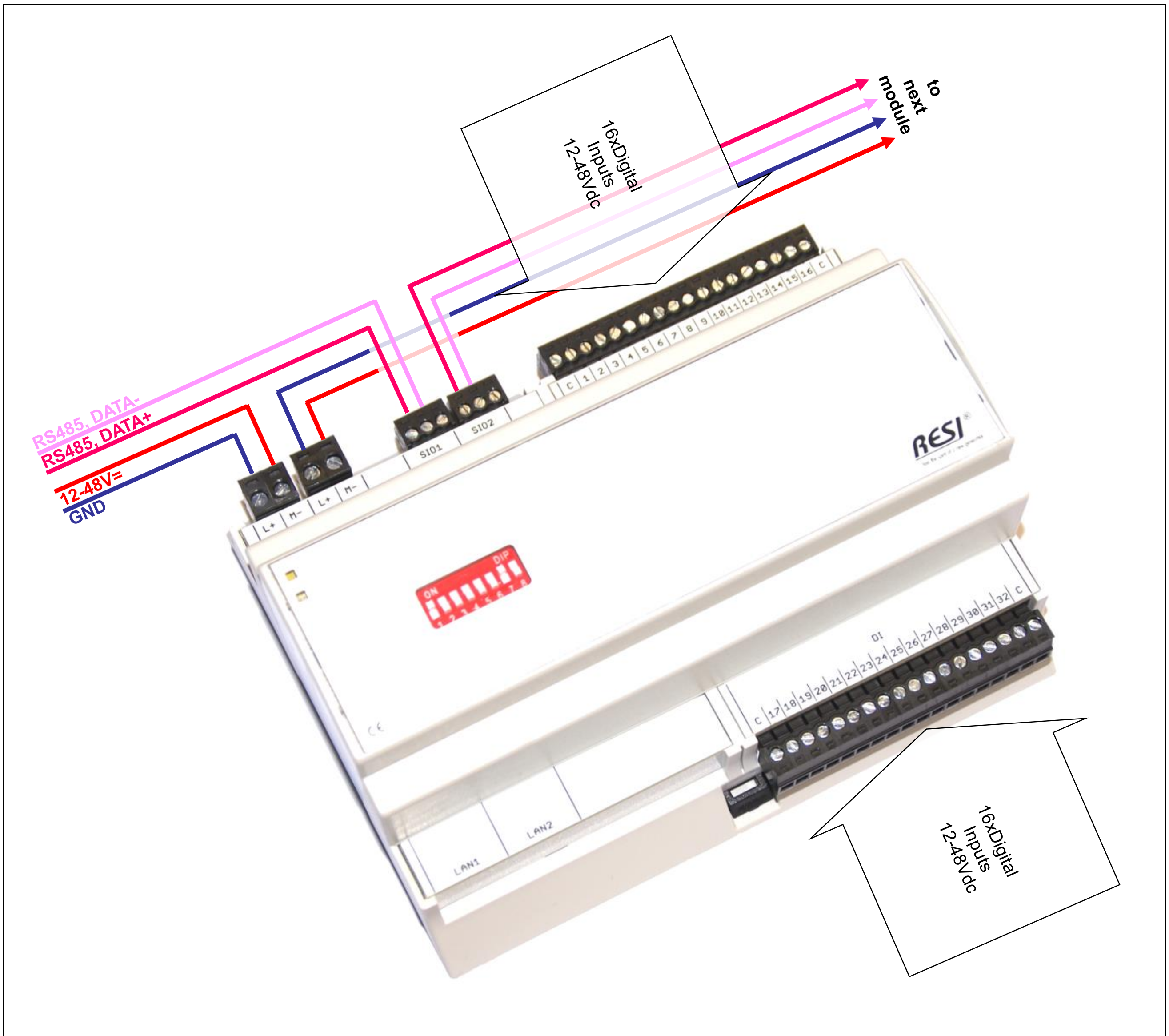


Illustration: cabling of the IO module

11.5 Clamps, DIP switch settings an LED indicators

The IO module offers the following clamps:

CLAMPS	RESI-32DI-MODBUS, RESI-32DI-ASCII
L+ M-	Power supply via two separated plug-in 2-pin terminal blocks. For daisy chain IN and OUT power supply of many modules L+: 12-48 V= M-: Ground
SIO1 1=A+ 2=B- 3=GND	RS485 ASCII or MODBUS/RTU serial interface IN A+: RS485 DATA+ signal B-: RS485 DATA- signal GND: RS485 ground signal
SIO2 1=A+ 2=B- 3=GND	RS485 ASCII or MODBUS/RTU serial interface OUT A+: RS485 DATA+ signal B-: RS485 DATA- signal GND: RS485 ground signal
DI C=GND 1=DI1 2=DI2 3=DI3 4=DI4 5=DI5 6=DI6 7=DI7 8=DI8 9=DI9 10=DI10 11=DI11 12=DI12 13=DI13 14=DI14 15=DI15 16=DI16 C=GND	16 digital inputs for 12-48Vdc signals C: Ground of the module DI1-DI16: Digital inputs 0=open or GND, 1=+12Vdc..+48Vdc
DI C=GND 1=DI17 2=DI18 3=DI19 4=DI20 5=DI21 6=DI22 7=DI23 8=DI24 9=DI25 10=DI26 11=DI27 12=DI28 13=DI29 14=DI30 15=DI31 16=DI32 C=GND	16 digital inputs for 12-48Vdc signals C: Ground of the module DI17-DI32: Digital inputs 0=open or GND, 1=+12Vdc..+48Vdc

Table: Description of the terminal blocks of the IO module

The IO module offers also an 8-pin DIP switch and a dual color LED indicator:

<table>
<tr><th>DIP+LED</th><th>RESI-32DI-MODBUS, RESI-32DI-ASCII</th></tr>
<tr><td>DIP SWITCH
1=ADR0
2=ADR1
3=ADR2
4=ADR3
5=BR0
6=BR1
7=BR2
8=PARITY</td><td>DIP switch to setup the IO module
ADR: 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:
<table>
<tr><td>ADR3</td><td>ADR2</td><td>ADR1</td><td>ADR0</td><td>MODBUS/RTU unit number or ASCII bus number</td></tr>
<tr><td>OFF</td><td>OFF</td><td>OFF</td><td>OFF</td><td>Internal MODBUS/RTU unit number is used from the FLASH memory in the range of 0 to 255.</td></tr>
<tr><td>OFF</td><td>OFF</td><td>OFF</td><td>ON</td><td>1</td></tr>
<tr><td>OFF</td><td>OFF</td><td>ON</td><td>OFF</td><td>2</td></tr>
<tr><td>OFF</td><td>OFF</td><td>ON</td><td>ON</td><td>3</td></tr>
<tr><td>OFF</td><td>ON</td><td>OFF</td><td>OFF</td><td>4</td></tr>
<tr><td>OFF</td><td>ON</td><td>OFF</td><td>ON</td><td>5</td></tr>
<tr><td>OFF</td><td>ON</td><td>ON</td><td>OFF</td><td>6</td></tr>
<tr><td>OFF</td><td>ON</td><td>ON</td><td>ON</td><td>7</td></tr>
<tr><td>ON</td><td>OFF</td><td>OFF</td><td>OFF</td><td>8</td></tr>
<tr><td>ON</td><td>OFF</td><td>OFF</td><td>ON</td><td>9</td></tr>
<tr><td>ON</td><td>OFF</td><td>ON</td><td>OFF</td><td>10</td></tr>
<tr><td>ON</td><td>OFF</td><td>ON</td><td>ON</td><td>11</td></tr>
<tr><td>ON</td><td>ON</td><td>OFF</td><td>OFF</td><td>12</td></tr>
<tr><td>ON</td><td>ON</td><td>OFF</td><td>ON</td><td>13</td></tr>
<tr><td>ON</td><td>ON</td><td>ON</td><td>OFF</td><td>14</td></tr>
<tr><td>ON</td><td>ON</td><td>ON</td><td>ON</td><td>15</td></tr>
</table>
BAUD RATE: Those three DIP switches BR2-BR0 define the MODBUS/RTU or ASCII baud rate for the communication:
<table>
<tr><td>BR2</td><td>BR1</td><td>BR0</td><td>MODBUS/RTU or ASCII Baudrate</td></tr>
<tr><td>OFF</td><td>OFF</td><td>OFF</td><td>4800bd</td></tr>
<tr><td>OFF</td><td>OFF</td><td>ON</td><td>9600bd</td></tr>
<tr><td>OFF</td><td>ON</td><td>OFF</td><td>19200bd</td></tr>
<tr><td>OFF</td><td>ON</td><td>ON</td><td>38400bd</td></tr>
<tr><td>ON</td><td>OFF</td><td>OFF</td><td>57600bd</td></tr>
<tr><td>ON</td><td>OFF</td><td>ON</td><td>115200bd</td></tr>
<tr><td>ON</td><td>ON</td><td>OFF</td><td>230400bd</td></tr>
<tr><td>ON</td><td>ON</td><td>ON</td><td>256000bd</td></tr>
</table>
PARITY: This DIP switch PARITY defines the MODBUS/RTU or ASCII parity for the communication:
<table>
<tr><td>PARITY</td><td>MODBUS/RTU or ASCII parity</td></tr>
<tr><td>OFF</td><td>NONE</td></tr>
<tr><td>ON</td><td>EVEN</td></tr>
</table>
HINT: After changing on of the DIP switches, the module restarts completely and intialises the serial interface. You will notice that the WHITE LED will be on for approximately 2 seconds, before this LED will flash with a one second cycle.</td></tr>
<tr><td>LED WHITE</td><td>This LED will flash with a cycle of 1 seconds to show normal mode of the module</td></tr>
<tr><td>LED GREEN</td><td>This LED will flash shortly, whenever the module receives a valid telegram on the serial interface.</td></tr>
<tr><td>LED RED</td><td>If this LED flashes cyclically, there is a module error detected by the firmware</td></tr>
</table>

Table: Description of the DIP switch functions and the indication LEDs on the IO module

11.7 Dimensions of the module

In the below drawing you will find the dimensions of the IO module.

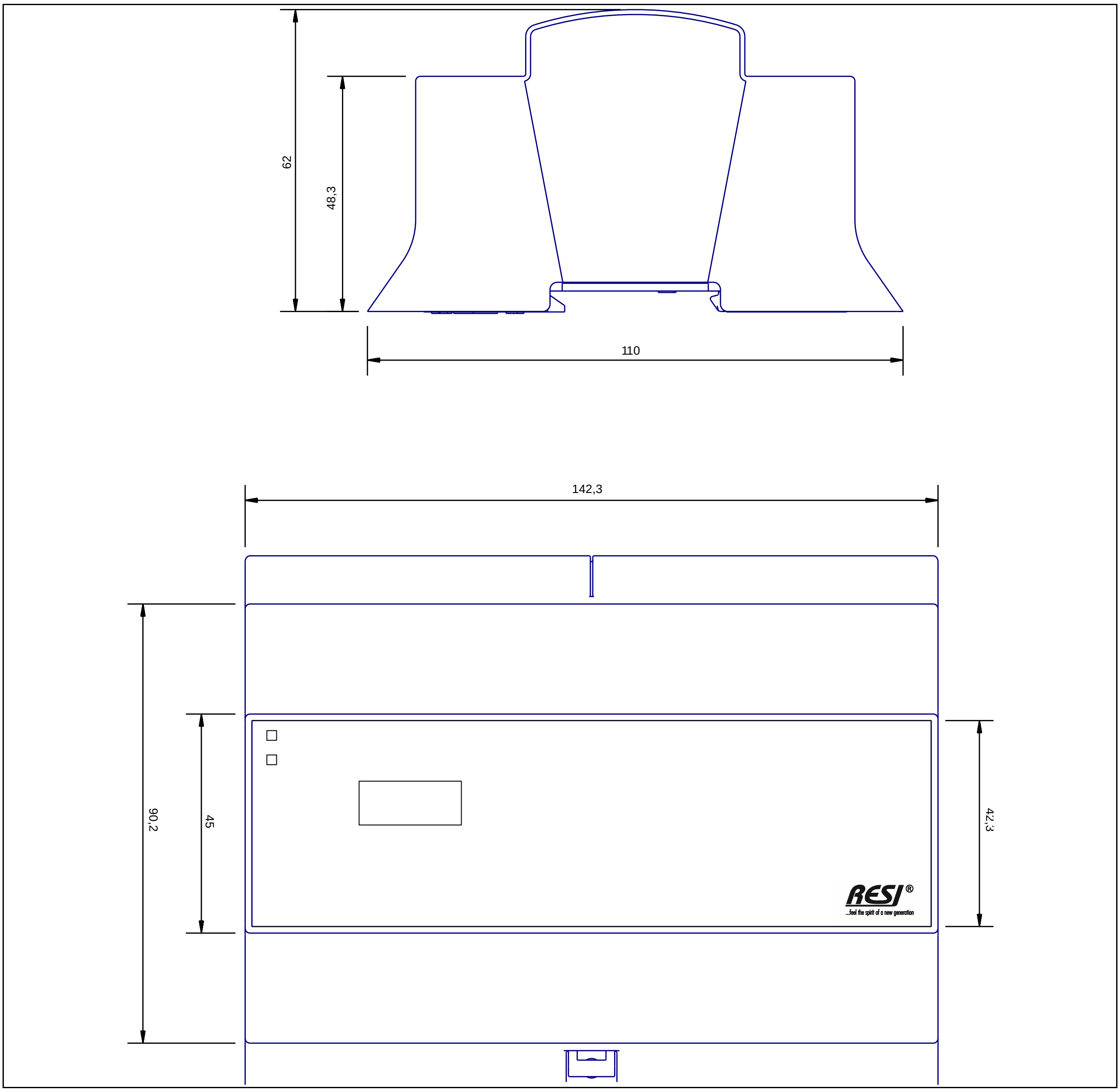


Illustration: Dimensions of the IO module in mm

Dimensions	
Dimensions of the housing L x B x H (mm)	143 x 110 x 62
Weight	260 g
Color	Grey, RAL7035
Material	Self-extinguish PC/ABS, DIN 43880
Protection class	IP20 based on DIN 40050/EN 60529

Table: technical data of the housing

11.8 Power supply of the module

In the below drawing you will find how to connect the module to a power supply.

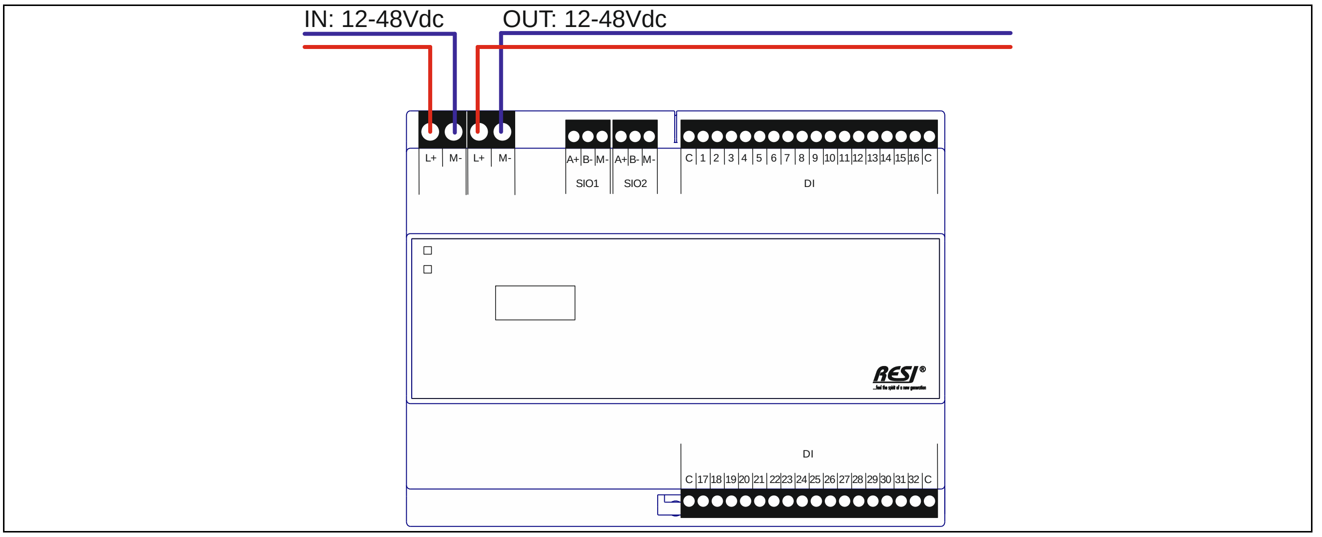


Illustration: Power supply of the IO module

The module offers two 2-pin plug-in terminals for connecting the power supply to the module. It is designed to create a daisy chain power supply with many modules.

11.9 Serial RS485 connection

In this drawing you see the cabling of the serial RS485 bus line. In the module both SIO terminal block are bridged.

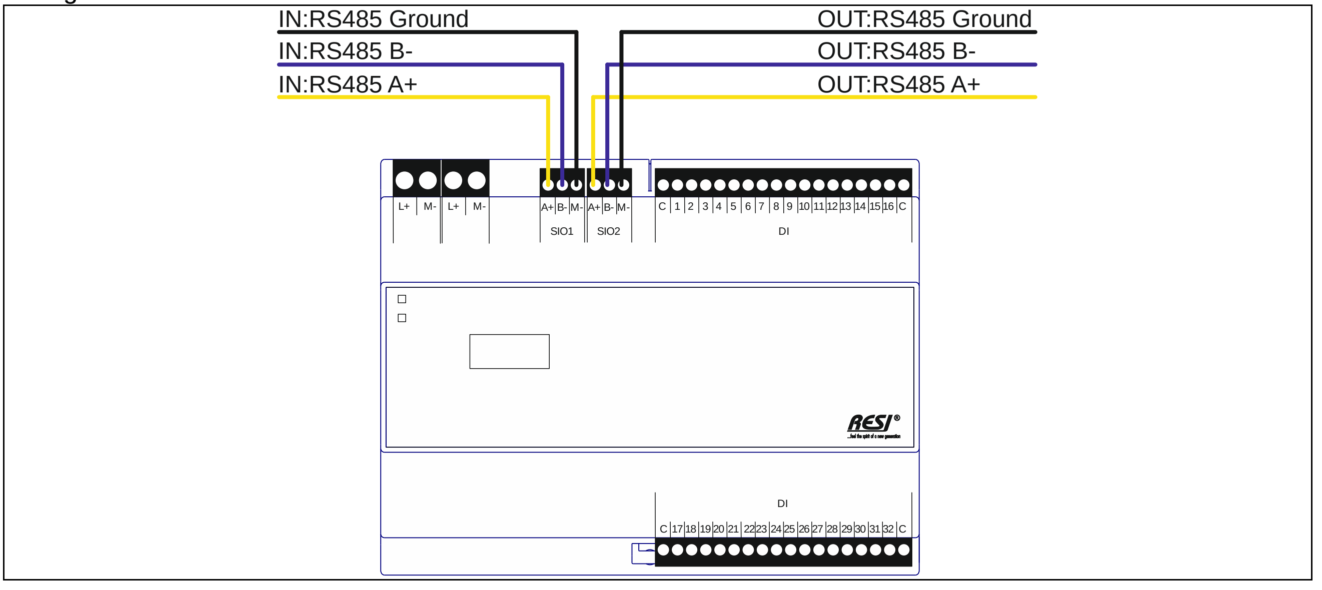


Illustration: RS485 bus cabling of the IO module

The module offers two plug-in 3-pin terminals to connect a RS485 bus line to the module. It was designed to create a daisy chain bus line with many modules. Don’t forget, that a RS485 bus line needs a line termination at the end of both lines!

11.10 Cabling of the digital inputs of the module

In the below drawing you see the cabling of the 32 digital inputs of the module. Both terminals C are internally connected to the ground signal.

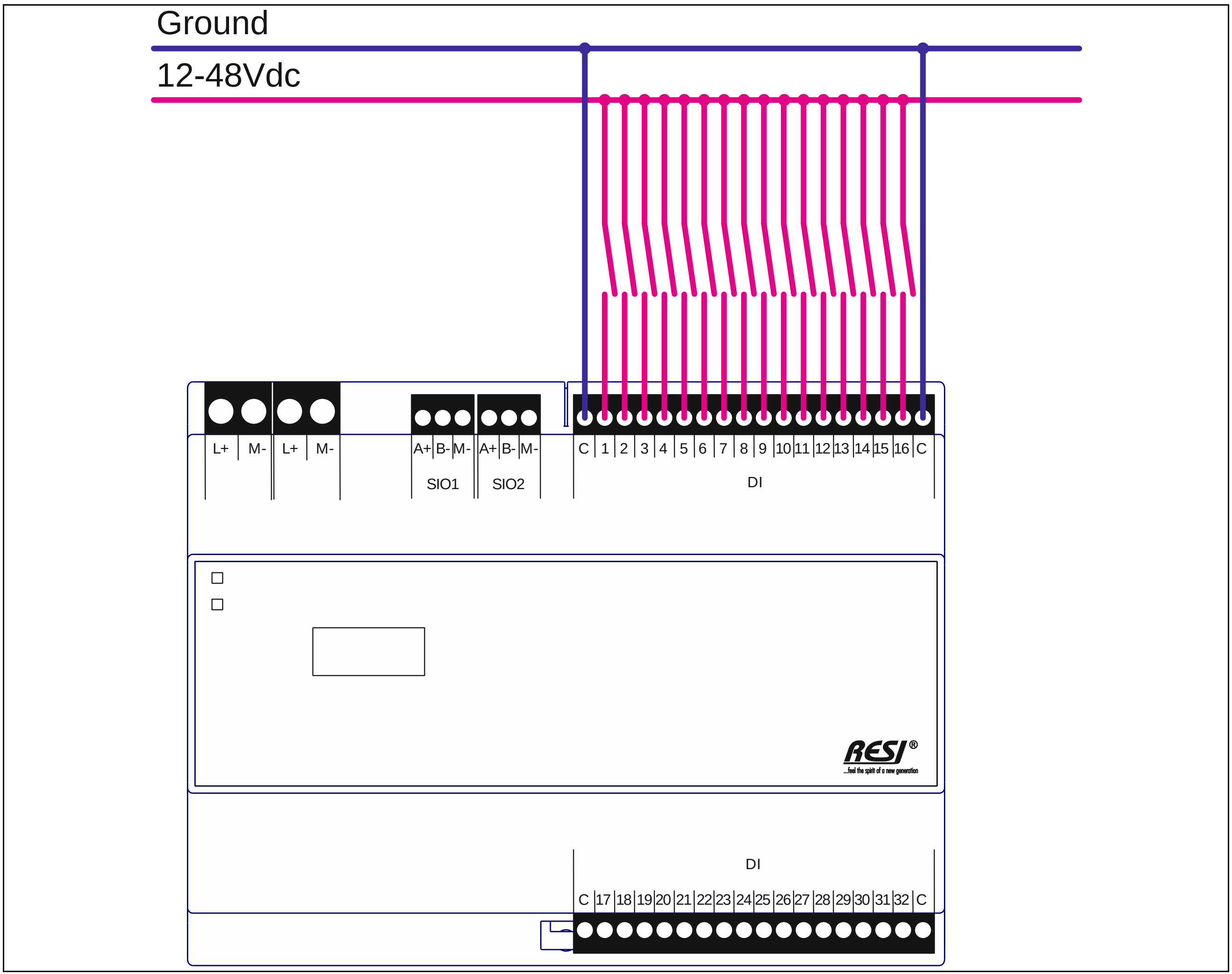


Illustration: Cabling of the first 16 digital inputs of the IO module

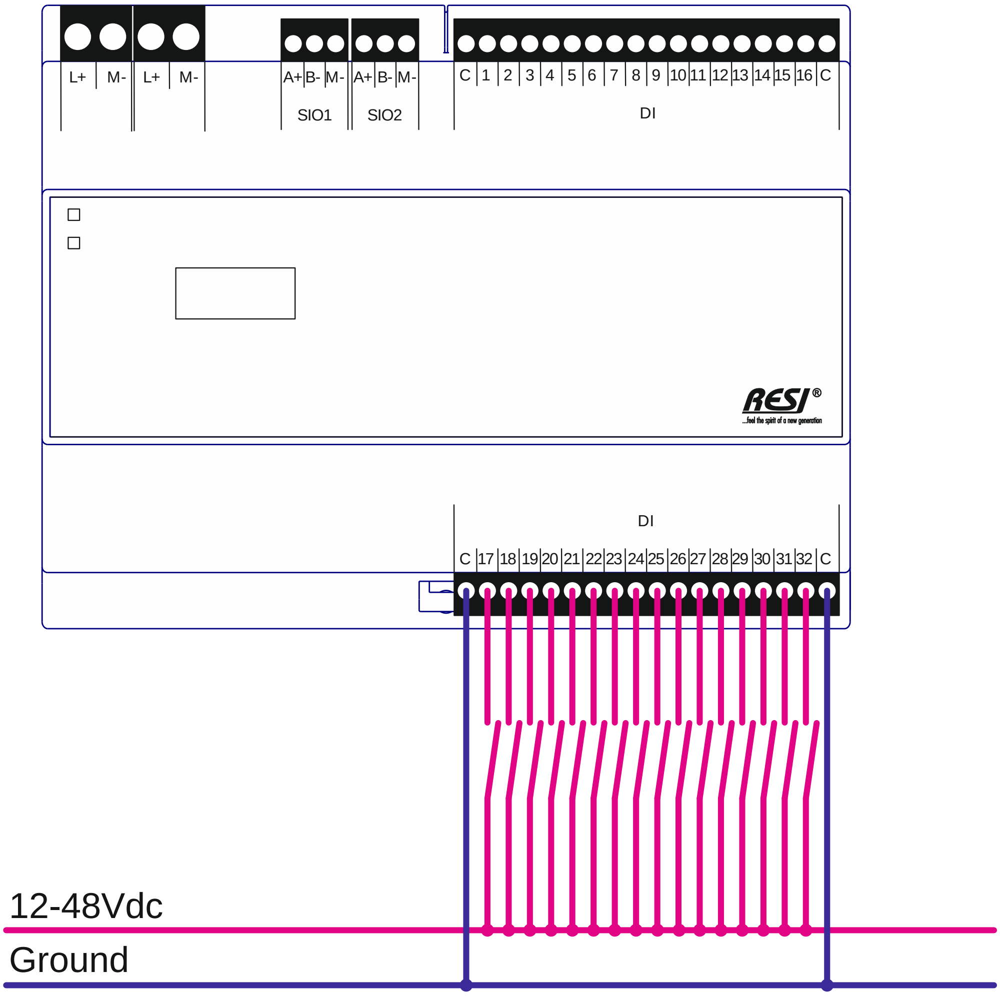


Illustration: Cabling of the second 16 digital inputs of the IO module

11.11 Functional description

This IO module communicates with a host system with the MODBUS/RTU slave protocol. All versions of the Io module with RESI-xxx-ASCII offer an additional protocol: An ASCII slave protocol with simple text string. The communication runs over a RS485 interface (half duplex) or over a RS232 interface (full duplex).

For the communication with ASCII texts, the host sends ASCII messages with a special start character # (0x23, 35dec) and a special end character (0x0d,13dec or CARRIAGE RETURN) to the module. The module uses also this special start and end characters to answer to the host request. Consult the below noted detailed command descriptions. In the ASCII protocol mode, the host can send messages with or without a bus number.

For communication with the MODBUS/RTU slave, the module offers the following MODBUS functions:

- 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)

HINT:
The functions READ HOLDING REGISTER and PRESET MULTIPLE REGISTERS are restricted to max. 125 register per request! The functions READ INPUT STATUS, READ COIL STATUS and FORCE MULTIPLE COILS are restricted to 2000 coils or inputs (bits) per request.

11.12 ASCII protocol description

11.12.1 Overview

The IO module communicates with simple ASCII commands. The following special characters are used in this documentation:

stand for the **Hashtag** ASCII character 35dec or 0x23
: stand for the **colon** ASCII character 58dec or 0x3A
= stand for the **equal sign** with the ASCII code 61ec or 0x3D
- stand for the **minus sign** with the ASCII code 45dec or 0x2D
, stands for the **comma** with the ASCII code 44dec or 0x2C
<CR> stand for **CARRIAGE RETURN** ASCII character 13dec or 0x0D. In the following text we use the representation CR.
<SP> stands for the **SPACE**. This is the space character with the ASCII code 32dec or 0x20. In the following text we will use the representation☐.
<ADR> will be used as the current **bus address** of the module. The bus number can be transmitted decimal or hexadecimal and is separated with a comma (ASCII code 44dec or 0x2C) from the rest of the command. Hexadecimal numbers always start with 0x. Its only allowed to use the ASCII characters ‚0‘-‚9‘ 48dec to 57dec, 0x30-0x39 and ‚A‘ to ‚F‘, 65dec to 70dec, 0x41-0x46. All modules react to the broadcast bus address 0 and to its own bus number. With a DIP switch, the user can easily change between the internal stored bus number in the FLASH and the fix bus number 255. Consult the DIP switch description for more details.

11.12.2 Communication sequence

In general the IO module sends no ASCII characters without a request from a host. So the host is the master of the communication and the IO module is always answering to host requests as a slave. If only one IO module is connected to a host (e.g. Via RS232), you can dump the bus number in the protocol. If you use a RS485 interface, more than one module can be connected to the host. Therefore a bus number in the request frame of the host is always necessary.

The structure of the command look like this:

The host sends a command or a command with parameters without a bus address to the module:
#<Command><CR> or
#<Command>:<Parameter><CR>

The module answers, if it feels addressed, with the following answer telegram:
#<Answer><CR>

If the bus number is used, the module answers with:
#<ADR>,<Command><CR> or
#<ADR>,<Command>:<Parameter><CR>

The module answers with:
#<ADR>,<Answer><CR>

The bus address lies in the range of 1dec to 255dec or 0x00 to 0xFF hexadecimal. The setup is done with our free configuration software MODBUSConfigurator. Each module reacts also to the broadcast bus address 0.

For each command, we define two different writings. A long version and a short version, to avoid unnecessary traffic on the bus. For example to request the software version of the module you can use the command VERSION or the short command VER.

11.12.3 Request VERSION

This command returns the current software version of the module.

Host long version:
#VERSION<CR> or
#<ADR>,VERSION<CR>

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

Answer:
#VERSION:<HIGH>.<MED>.<LOW><CR> or
#<ADR>,VERSION:<HIGH>,<MED>,<LOW><CR>

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

Samples:
→ **#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$_{CR}$**

11.12.4 Request module TYPE

This command returns the current type of the module.

Host long version:
#TYPE<CR> or
#<ADR>,TYPE<CR>

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

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

<TYP> defines the current type of the module. Currently RESI-32DI-ASCII

Samples:
→ **#TYPE$_{CR}$**
← **#TYPE:RESI-32DI-ASCII$_{CR}$**

→ **#255,TYP$_{CR}$**
← **#255,TYPE:RESI-32DI-ASCII$_{CR}$**

11.12.5 Table of all ASCII commands

Here you will find a possible ASCII commands of the module. We use here only the version with bus number. That you can avoid the bus number, we have discussed earlier in this document. If an argument as the extension Dec, it will be returned as a decimal number, If an argument has the extension Hex, then this argument is returned as a hexadecimal number. Many command returns the argument in decimal and hexadecimal representation. So the host can select, what kind of number conversion, it will handle in its software.

Direction	ASCII command
Host	**#<BusAdr>,VER**$_{CR}$ **#<BusAdr>,VERSION**$_{CR}$
Answer	**#<BusAdr>,VERSION:<VersionHi>.< VersionMed>.<VersionLo>**$_{CR}$
	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)
Host	**#<BusAdr>,TYP**$_{CR}$ **#<BusAdr>,TYPE**$_{CR}$
Answer	**#<BusAdr>,TYPE:RESI-32DI-ASCII**$_{CR}$
	Returns the current type of the module
Host	**#<BusAdr>,OWN**$_{CR}$ **#<BusAdr>,OWNER**$_{CR}$
Answer	**#<BusAdr>,OWNER:RESI**$_{CR}$
	Returns the owner of the module
Host	**#<BusAdr>,CRE**$_{CR}$ **#<BusAdr>,CREATOR**$_{CR}$
Answer	**#<BusAdr>,CREATOR:DI HC SIGL,MSC**$_{CR}$
	Returns the creator of the module

Direction	ASCII command
Host	**#<BusAdr>,COPY$_{CR}$** **#<BusAdr>,COPYRIGHT$_{CR}$**
Answer	**#<BusAdr>,COPYRIGHT:2015-16 BY RESI AND DI HC SIGL,MSC WWW.RESI.CC$_{CR}$**
	Returns a copyright note about the module
Host	**#<BusAdr>,GDIP$_{CR}$** **#<BusAdr>,GET□DIP$_{CR}$**
Answer	**#<BusAdr>,GDIP:<DIPSwitchDec>,<DIPSwitchHex>$_{CR}$**
	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 (=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)
Host	**#<BusAdr>,GDIS$_{CR}$** **#<BusAdr>,GET□DIS$_{CR}$**
Answer	**#<BusAdr>,GDIS:<DISDec>,<DISHex>$_{CR}$**
	Returns the current state of all 32 digital inputs as decimal number and as hexadecimal number. DISDec DISHex The current state of all digital inputs: Bit 0: State of DI1 (=0:OFF, =1:ON) Bit 1: State of DI2 (=0:OFF, =1:ON) Bit 2: State of DI3 (=0:OFF, =1:ON) Bit 3: State of DI4 (=0:OFF, =1:ON) Bit 4: State of DI5 (=0:OFF, =1:ON) Bit 5: State of DI6 (=0:OFF, =1:ON) Bit 6: State of DI7 (=0:OFF, =1:ON) Bit 7: State of DI8 (=0:OFF, =1:ON) Bit 8: State of DI9 (=0:OFF, =1:ON) Bit 9: State of DI10 (=0:OFF, =1:ON) Bit 10: State of DI11 (=0:OFF, =1:ON) Bit 11: State of DI12 (=0:OFF, =1:ON) Bit 12: State of DI13 (=0:OFF, =1:ON) Bit 13: State of DI14 (=0:OFF, =1:ON) Bit 14: State of DI15 (=0:OFF, =1:ON) Bit 15: State of DI16 (=0:OFF, =1:ON) Bit 16: State of DI17 (=0:OFF, =1:ON) Bit 17: State of DI18 (=0:OFF, =1:ON) Bit 18: State of DI19 (=0:OFF, =1:ON) Bit 19: State of DI20 (=0:OFF, =1:ON) Bit 20: State of DI21 (=0:OFF, =1:ON) Bit 21: State of DI22 (=0:OFF, =1:ON) Bit 22: State of DI23 (=0:OFF, =1:ON) Bit 23: State of DI24 (=0:OFF, =1:ON) Bit 24: State of DI25 (=0:OFF, =1:ON) Bit 25: State of DI26 (=0:OFF, =1:ON) Bit 26: State of DI27 (=0:OFF, =1:ON) Bit 27: State of DI28 (=0:OFF, =1:ON) Bit 28: State of DI29 (=0:OFF, =1:ON) Bit 29: State of DI30 (=0:OFF, =1:ON) Bit 30: State of DI31 (=0:OFF, =1:ON) Bit 31: State of DI32 (=0:OFF, =1:ON)

Direction	ASCII command
Host	**#<BusAdr>,GDIx$_{CR}$** **#<BusAdr>,GET□DIx$_{CR}$**
Answer	**#<BusAdr>,GDIx:<DIxDec>,<DIxHex>$_{CR}$**
x	**1..32**
	Returns the current state of the digital input Dix as decimal number and as hexadecimal number. X stands for the desired digital input between 1 and 32. DIxDec DIxHex — The current state of the digital input x: =0: Digital input is OFF =1: Digital input is ON
Host	**#<BusAdr>,RDIx$_{CR}$** **#<BusAdr>,RISE□DIx$_{CR}$**
Answer	**#<BusAdr>,RDIx:<RDIxDec>,<RDIxHex>$_{CR}$**
x	**1..32**
	Returns the current counter for rising edges on the digital input x since last power on of the module as decimal number and as hexadecimal number. RDIxDec RDIxHex — The current amount of counted rising edges on the digital input x
Host	**#<BusAdr>,FDIx$_{CR}$** **#<BusAdr>,FALL□DIx$_{CR}$**
Answer	**#<BusAdr>,FDIx:<FDIxDec>,<FDIxHex>$_{CR}$**
x	**1..32**
	Returns the current counter for falling edges on the digital input x since last power on of the module as decimal number and as hexadecimal number. FDIxDec FDIxHex — The current amount of counted falling edges on the digital input x
Host	**#<BusAdr>,RC$_{CR}$** **#<BusAdr>,RESET□COUNTERS$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
	This command deletes all counters for rising and falling edges of the 32 digital inputs in the module to 0.

Direction	**ASCII command**
Host	**#<BusAdr>,SMBADR:<MBUnit>$_{CR}$** **#<BusAdr>,SET□MODBUS□ADDRESS:<MBUnit>$_{CR}$**
Answer	**#<BusAdr>,OK $_{CR}$**
	Writes the unit address into the FLASH memory of the module. The new unit address for MODBUS/RTU or ASCII mode is only used immediately, if the DIP switch setting of the bus address is 0. Otherwise the unit address is defined by the DIP settings. The unit address ranges from 0dec to 255dec.
Host	**#<BusAdr>,GMBADR$_{CR}$** **#<BusAdr>,GET□MODBUS□ADDRESS$_{CR}$**
Answer	**#<BusAdr>,GMBADR:<MBUnitDec>,<MBFLASHDec>,<MBUnitHex>,<MBFLASHHex>$_{CR}$**
	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. MBUnitDec MBUnitHex: The current used MODBUS/RTU unit or ASCII address for communication MBFLASHDec MBFLASHHex: The internal stored MODBUS/RTU unit address or ASCII address from the FLASH memory, if the DIP switch is 0.
Host	**#<BusAdr>,RST$_{CR}$** **#<BusAdr>,RESET$_{CR}$**
Answer	**none**
	Executes a software reset (Reboot) of the module.

11.13 MODBUS – register description

11.13.1 Table of inputs and coils

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 0x00001 to 0x65536. Inputs are usually noted with 1x00001 to 1x65536. 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: 0x00100 for the coil 100, 1x00100 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.

Register	Description
0x00001 1x00001 I:0 R/O DI1	Current state of the digital input DI1 =0:DI is OFF, =1:DI is ON
0x00002 1x00002 I:1 R/O DI2	Current state of the digital input DI2 =0:DI is OFF, =1:DI is ON
0x00003 1x00003 I:2 R/O DI3	Current state of the digital input DI3 =0:DI is OFF, =1:DI is ON
0x00004 1x00004 I:3 R/O DI4	Current state of the digital input DI4 =0:DI is OFF, =1:DI is ON
0x00005 1x00005 I:4 R/O DI5	Current state of the digital input DI5 =0:DI is OFF, =1:DI is ON
0x00006 1x00006 I:5 R/O DI6	Current state of the digital input DI6 =0:DI is OFF, =1:DI is ON
0x00007 1x00007 I:6 R/O DI7	Current state of the digital input DI7 =0:DI is OFF, =1:DI is ON
0x00008 1x00008 I:7 R/O DI8	Current state of the digital input DI8 =0:DI is OFF, =1:DI is ON

Register	Description
0x00009 1x00009 I:8 R/O DI9	Current state of the digital input DI9 =0:DI is OFF, =1:DI is ON
0x00010 1x00010 I:9 R/O DI10	Current state of the digital input DI10 =0:DI is OFF, =1:DI is ON
0x00011 1x00011 I:10 R/O DI11	Current state of the digital input DI11 =0:DI is OFF, =1:DI is ON
0x00012 1x00012 I:11 R/O DI12	Current state of the digital input DI12 =0:DI is OFF, =1:DI is ON
0x00013 1x00013 I:12 R/O DI13	Current state of the digital input DI13 =0:DI is OFF, =1:DI is ON
0x00014 1x00014 I:13 R/O DI14	Current state of the digital input DI14 =0:DI is OFF, =1:DI is ON
0x00015 1x00015 I:14 R/O DI15	Current state of the digital input DI15 =0:DI is OFF, =1:DI is ON
0x00016 1x00016 I:15 R/O DI16	Current state of the digital input DI16 =0:DI is OFF, =1:DI is ON

Register	Description
0x00017 1x00017 I:16 R/O DI17	Current state of the digital input DI17 =0:DI is OFF, =1:DI is ON
0x00018 1x00018 I:17 R/O DI18	Current state of the digital input DI18 =0:DI is OFF, =1:DI is ON
0x00019 1x00019 I:18 R/O DI19	Current state of the digital input DI19 =0:DI is OFF, =1:DI is ON
0x00020 1x00020 I:19 R/O DI20	Current state of the digital input DI20 =0:DI is OFF, =1:DI is ON
0x00021 1x00021 I:20 R/O DI21	Current state of the digital input DI21 =0:DI is OFF, =1:DI is ON
0x00022 1x00022 I:21 R/O DI22	Current state of the digital input DI22 =0:DI is OFF, =1:DI is ON
0x00023 1x00023 I:22 R/O DI23	Current state of the digital input DI23 =0:DI is OFF, =1:DI is ON
0x00024 1x00024 I:23 R/O DI24	Current state of the digital input DI24 =0:DI is OFF, =1:DI is ON

Register	Description
0x00025 1x00025 I:24 R/O DI25	Current state of the digital input DI25 =0:DI is OFF, =1:DI is ON
0x00026 1x00026 I:25 R/O DI26	Current state of the digital input DI26 =0:DI is OFF, =1:DI is ON
0x00027 1x00027 I:26 R/O DI27	Current state of the digital input DI27 =0:DI is OFF, =1:DI is ON
0x00028 1x00028 I:27 R/O DI28	Current state of the digital input DI28 =0:DI is OFF, =1:DI is ON
0x00029 1x00029 I:28 R/O DI29	Current state of the digital input DI29 =0:DI is OFF, =1:DI is ON
0x00030 1x00030 I:29 R/O DI30	Current state of the digital input DI30 =0:DI is OFF, =1:DI is ON
0x00031 1x00031 I:30 R/O DI31	Current state of the digital input DI31 =0:DI is OFF, =1:DI is ON
0x00032 1x00032 I:31 R/O DI32	Current state of the digital input DI32 =0:DI is OFF, =1:DI is ON

Register	Description
0x00033 1x00033 I:32 R/O DIP1	Current state of DIP switch 1 =0:Dip switch is OFF, =1: Dip switch is ON
0x00034 1x00034 I:33 R/O DIP2	Current state of DIP switch 2 =0:Dip switch is OFF, =1: Dip switch is ON
0x00035 1x00035 I:34 R/O DIP3	Current state of DIP switch 3 =0:Dip switch is OFF, =1: Dip switch is ON
0x00036 1x00036 I:35 R/O DIP4	Current state of DIP switch 4 =0:Dip switch is OFF, =1: Dip switch is ON
0x00037 1x00037 I:36 R/O DIP5	Current state of DIP switch 5 =0:Dip switch is OFF, =1: Dip switch is ON
0x00038 1x00038 I:37 R/O DIP6	Current state of DIP switch 6 =0:Dip switch is OFF, =1: Dip switch is ON
0x00039 1x00039 I:38 R/O DIP7	Current state of DIP switch 7 =0:Dip switch is OFF, =1: Dip switch is ON
0x00040 1x00040 I:39 R/O DIP8	Current state of DIP switch 8 =0:Dip switch is OFF, =1: Dip switch is ON
0x00100 1x00100 I:99 R/W RESET COUNTER	Reset, resetting the internal edge counters for all digital inputs. While reading always 0.

11.13.2 Table of holding/input registers

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.

Register	Description
4x00001 3x00001 I:0 R/O RISE DI1	Counter for rising edges on the digital input DI1. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00002 3x00002 I:1 R/O FALL DI1	Counter for falling edges on the digital input DI1. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00003 3x00003 I:2 R/O RISE DI2	Counter for rising edges on the digital input DI2. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00004 3x00004 I:3 R/O FALL DI2	Counter for falling edges on the digital input DI2. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00005 3x00005 I:4 R/O RISE DI3	Counter for rising edges on the digital input DI3. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00006 3x00006 I:5 R/O FALL DI3	Counter for falling edges on the digital input DI3. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00007 3x00007 I:6 R/O RISE DI4	Counter for rising edges on the digital input DI4. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00008 3x00008 I:7 R/O FALL DI4	Counter for falling edges on the digital input DI4. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00009 3x00009 I:8 R/O RISE DI5	Counter for rising edges on the digital input DI5. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00010 3x00010 I:9 R/O FALL DI5	Counter for falling edges on the digital input DI5. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00011 3x00011 I:10 R/O RISE DI6	Counter for rising edges on the digital input DI6. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00012 3x00012 I:11 R/O FALL DI6	Counter for falling edges on the digital input DI6. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.

Register	Description
4x00013 3x00013 I:12 R/O RISE DI7	Counter for rising edges on the digital input DI7. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00014 3x00014 I:13 R/O FALL DI7	Counter for falling edges on the digital input DI7. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00015 3x00015 I:14 R/O RISE DI8	Counter for rising edges on the digital input DI8. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00016 3x00016 I:15 R/O FALL DI8	Counter for falling edges on the digital input DI8. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00017 3x00017 I:16 R/O RISE DI9	Counter for rising edges on the digital input DI9. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00018 3x00018 I:17 R/O FALL DI9	Counter for falling edges on the digital input DI9. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00019 3x00019 I:18 R/O RISE DI10	Counter for rising edges on the digital input DI10. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00020 3x00020 I:19 R/O FALL DI10	Counter for falling edges on the digital input DI10. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00021 3x00021 I:20 R/O RISE DI11	Counter for rising edges on the digital input DI11. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00022 3x00022 I:21 R/O FALL DI11	Counter for falling edges on the digital input DI11. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00023 3x00023 I:22 R/O RISE DI12	Counter for rising edges on the digital input DI12. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00024 3x00024 I:23 R/O FALL DI12	Counter for falling edges on the digital input DI12. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.

Register	Description
4x00025 3x00025 I:24 R/O RISE DI13	Counter for rising edges on the digital input DI13. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00026 3x00026 I:25 R/O FALL DI13	Counter for falling edges on the digital input DI13. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00027 3x00027 I:26 R/O RISE DI14	Counter for rising edges on the digital input DI14. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00028 3x00028 I:27 R/O FALL DI14	Counter for falling edges on the digital input DI14. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00029 3x00029 I:28 R/O RISE DI15	Counter for rising edges on the digital input DI15. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00030 3x00030 I:29 R/O FALL DI15	Counter for falling edges on the digital input DI15. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00031 3x00031 I:30 R/O RISE DI16	Counter for rising edges on the digital input DI16. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00032 3x00032 I:31 R/O FALL DI16	Counter for falling edges on the digital input DI16. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.

Register	**Description**
4x00033 3x00033 I:32 R/O RISE DI17	Counter for rising edges on the digital input DI13. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00034 3x00034 I:33 R/O FALL DI17	Counter for falling edges on the digital input DI13. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00035 3x00035 I:34 R/O RISE DI18	Counter for rising edges on the digital input DI14. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00036 3x00036 I:35 R/O FALL DI18	Counter for falling edges on the digital input DI14. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00037 3x00037 I:36 R/O RISE DI19	Counter for rising edges on the digital input DI15. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00038 3x00038 I:37 R/O FALL DI19	Counter for falling edges on the digital input DI15. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00039 3x00039 I:38 R/O RISE DI20	Counter for rising edges on the digital input DI16. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00040 3x00040 I:39 R/O FALL DI20	Counter for falling edges on the digital input DI16. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.

Register	Description
4x00041 3x00041 I:40 R/O RISE DI21	Counter for rising edges on the digital input DI13. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00042 3x00042 I:41 R/O FALL DI21	Counter for falling edges on the digital input DI13. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00043 3x00043 I:42 R/O RISE DI22	Counter for rising edges on the digital input DI14. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00044 3x00044 I:43 R/O FALL DI22	Counter for falling edges on the digital input DI14. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00045 3x00045 I:44 R/O RISE DI23	Counter for rising edges on the digital input DI15. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00046 3x00046 I:45 R/O FALL DI23	Counter for falling edges on the digital input DI15. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00047 3x00047 I:46 R/O RISE DI24	Counter for rising edges on the digital input DI16. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00048 3x00048 I:47 R/O FALL DI24	Counter for falling edges on the digital input DI16. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.

Register	Description
4x00049 3x00049 I:48 R/O RISE DI25	Counter for rising edges on the digital input DI13. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00050 3x00050 I:49 R/O FALL DI25	Counter for falling edges on the digital input DI13. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00051 3x00051 I:50 R/O RISE DI26	Counter for rising edges on the digital input DI14. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00052 3x00052 I:51 R/O FALL DI26	Counter for falling edges on the digital input DI14. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00053 3x00053 I:52 R/O RISE DI27	Counter for rising edges on the digital input DI15. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00054 3x00054 I:53 R/O FALL DI27	Counter for falling edges on the digital input DI15. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00055 3x00055 I:54 R/O RISE DI28	Counter for rising edges on the digital input DI16. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00056 3x00056 I:55 R/O FALL DI28	Counter for falling edges on the digital input DI16. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.

Register	Description
4x00057 3x00057 I:56 R/O RISE DI29	Counter for rising edges on the digital input DI13. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00058 3x00058 I:57 R/O FALL DI29	Counter for falling edges on the digital input DI13. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00059 3x00059 I:58 R/O RISE DI30	Counter for rising edges on the digital input DI14. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00060 3x00060 I:59 R/O FALL DI30	Counter for falling edges on the digital input DI14. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00061 3x00061 I:60 R/O RISE DI31	Counter for rising edges on the digital input DI15. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00062 3x00062 I:61 R/O FALL DI31	Counter for falling edges on the digital input DI15. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00063 3x00063 I:62 R/O RISE DI32	Counter for rising edges on the digital input DI16. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00064 3x00064 I:63 R/O FALL DI32	Counter for falling edges on the digital input DI16. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00100 3x00100 I:99 R/W RESET COUNTER	Reset, resetting the internal edge counters for all digital inputs. While reading always 0.

Register	Description
4x00101 3x00101 I:100 R/O DIS1..16	Current state of digital inputs 1..16 Bit 0: =0:DI1 is OFF, =1:DI1 is ON Bit 1: =0:DI2 is OFF, =1:DI2 is ON Bit 2: =0:DI3 is OFF, =1:DI3 is ON Bit 3: =0:DI4 is OFF, =1:DI4 is ON Bit 4: =0:DI5 is OFF, =1:DI5 is ON Bit 5: =0:DI6 is OFF, =1:DI6 is ON Bit 6: =0:DI7 is OFF, =1:DI7 is ON Bit 7: =0:DI8 is OFF, =1:DI8 is ON Bit 8: =0:DI9 is OFF, =1:DI9 is ON Bit 9: =0:DI10 is OFF, =1:DI10 is ON Bit 10: =0:DI11 is OFF, =1:DI11 is ON Bit 11: =0:DI12 is OFF, =1:DI12 is ON Bit 12: =0:DI13 is OFF, =1:DI13 is ON Bit 13: =0:DI14 is OFF, =1:DI14 is ON Bit 14: =0:DI15 is OFF, =1:DI15 is ON Bit 15: =0:DI16 is OFF, =1:DI16 is ON
4x00102 3x00102 I:101 R/O DIS17..32	Current state of digital inputs 17..32 Bit 0: =0:DI17 is OFF, =1:DI17 is ON Bit 1: =0:DI18 is OFF, =1:DI18 is ON Bit 2: =0:DI19 is OFF, =1:DI19 is ON Bit 3: =0:DI20 is OFF, =1:DI20 is ON Bit 4: =0:DI21 is OFF, =1:DI21 is ON Bit 5: =0:D22 is OFF, =1:DI22 is ON Bit 6: =0:DI23 is OFF, =1:DI23 is ON Bit 7: =0:DI24 is OFF, =1:DI24 is ON Bit 8: =0:DI25 is OFF, =1:DI25 is ON Bit 9: =0:DI26 is OFF, =1:DI26 is ON Bit 10: =0:DI27 is OFF, =1:DI27 is ON Bit 11: =0:DI28 is OFF, =1:DI28 is ON Bit 12: =0:DI29 is OFF, =1:DI29 is ON Bit 13: =0:DI30 is OFF, =1:DI30 is ON Bit 14: =0:DI31 is OFF, =1:DI31 is ON Bit 15: =0:DI32 is OFF, =1:DI32 is ON
4x00103 3x00103 I:102 R/O DIP	Current state of the DIP switch 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) Bit 8-15: always 0

12 RESI-14RI-MODBUS, RESI-14RI-ASCII

12.1 Product description

This IO module offers the following features:

- 14 digital inputs for 24-250Vac/dc signals
- Each digital input is galvanic insulated to all other digital inputs
- Galvanic insulated RS485 interface for communication with a host system
- RESI-14RI-MODBUS: MODBUS/RTU slave protocol
- RESI-14RI-ASCII: MODBUS/RTU slave and text based ASCII protocol
- Power supply 12-48Vdc
- DIP switch for setting the baud rates, the type of interface and the bus number
- LED indicator for the communication
- Mounting onto EN50022 DIN rail or wall mounting

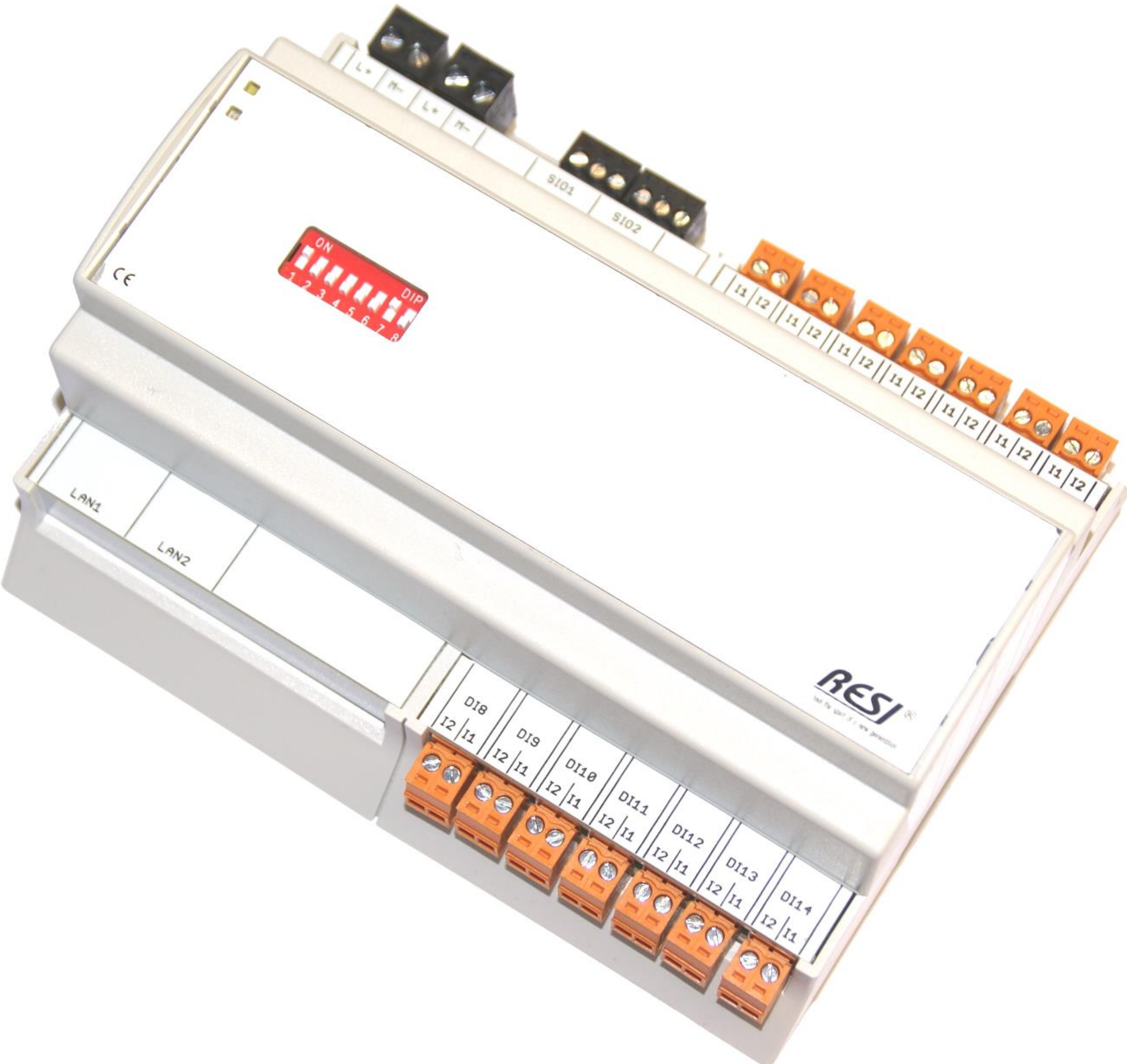


Illustration: Our IO module

12.2 Technical data

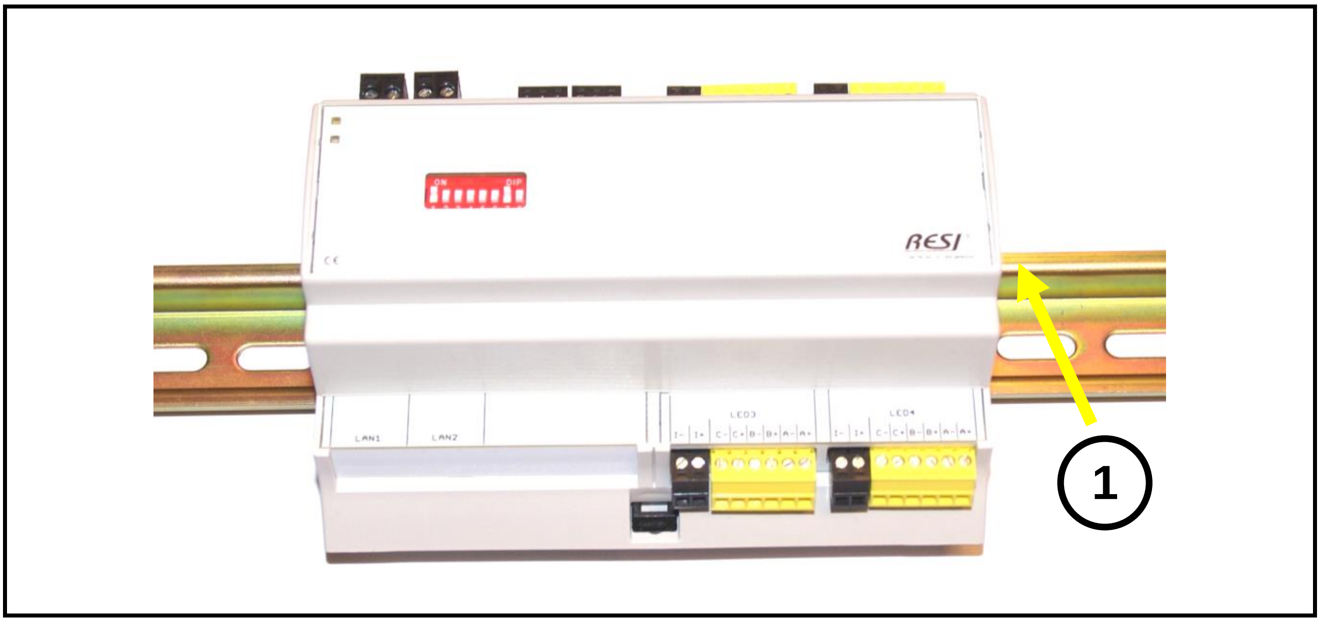
Technical Data			
Power supply			
Supply voltage	12-48V= +/-10%	Storage temperature	-20...85 °C
Power LED	Ja	Operating Temperature	0...60°C
Power consumption	<0.5W	Humidity	25...90 % rH non-condensing
		Protection Class	IP20 (EN 60529)
		Dimensions LxWxH	143mm x 110mm x 62mm
		Weight	265g
		Mounting	On DIN EN50022 rail or wall mounting
ASCII/Modbus Interface			
Protocol	ASCII or Modbus/RTU		
Type	RS485		
Baud rates	4800 to 256000Bd/8/N or E/1		
Cable Connection	Via removable clamps		
LED indicator	Yes		
Galvanic insulation	No		
Digital inputs			
Total amount of inputs	14		
Sampling rate	Every 10ms		
Input voltage range DC	24-250Vdc +/-10%		
Input current DC	per channel approx. 1.0mA@20V= approx. 1.6mA@24V= approx. 1.9mA@32V= approx. 2.1mA@250V=		
Input power consumption DC	max. 0.6W/channel		
Logic levels DC	0: <3V= 1: >20V=		
Input voltage range AC	24-250Vac +/-10%		
Input current AC	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 AC	max. 0.6W/channel		
Logic levels AC	0: <3V~ 1: >20V~		
Cable connection	Each input separately via 2-pin plug-in terminal block in orange		
Galvanic insulation	Yes, to the module and to each other digital input		
Clamps			
Clamp wire cross section	Max. 1,5 mm²		
Tightening torque	Max. 0.5Nm	**CE conformity**	Yes

12.3 Assembling

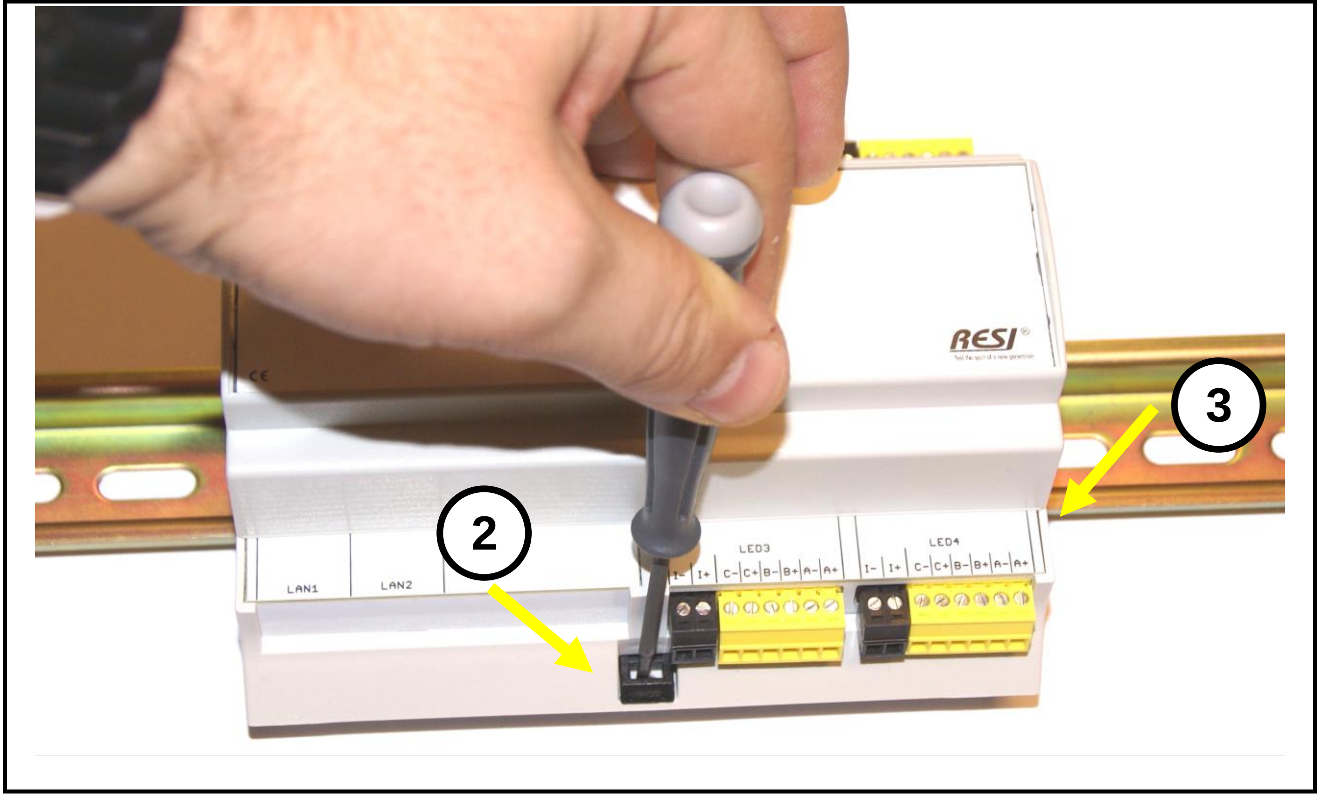
Our IO modules are designed for mounting onto a 35mm DIN-EN50022 rail or for wall mounting. Please not, that in the following mounting description we use only symbolic photos of our IO modules.

12.3.1 Mounting of a DIN EN50022 rail

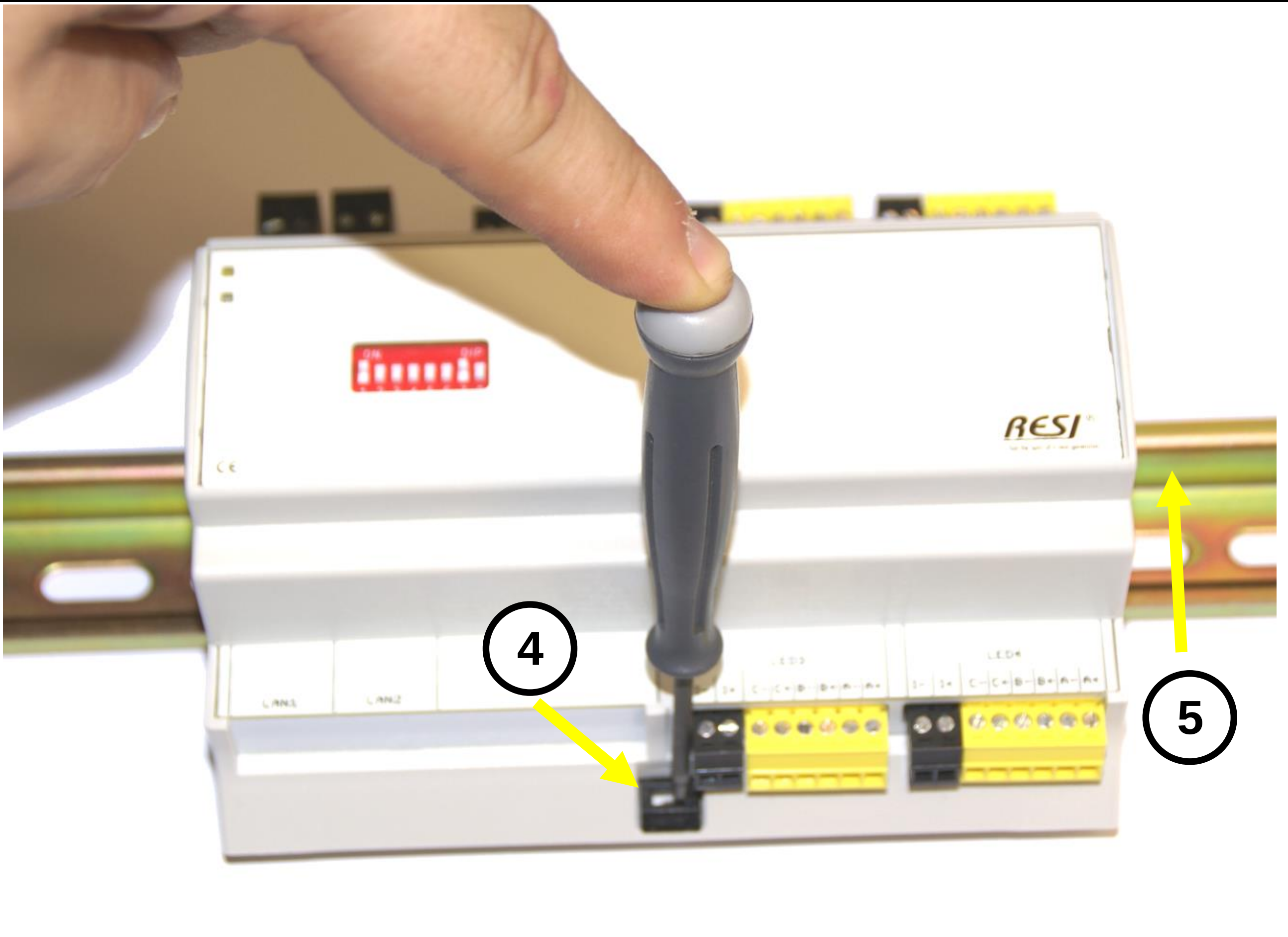
First snap in the top part of the module into the DIN rail (1). The bottom part of the module is not snapped into the DIN rail at this moment.



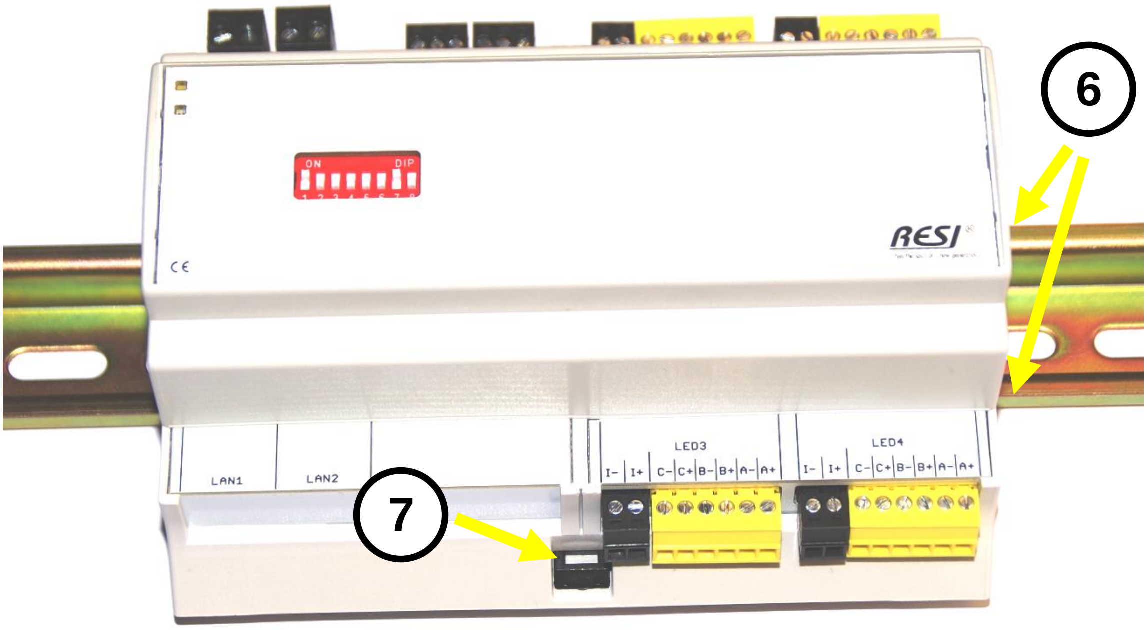
Then open the black hook with a screw driver (2). Now press the module with the opened hook onto the DIN rail until both sides of the module snap into the DIN rail (3). Release the screw driver now. The hook snaps into the DIN rail and the module is now mounted correctly onto the DIN rail.



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.

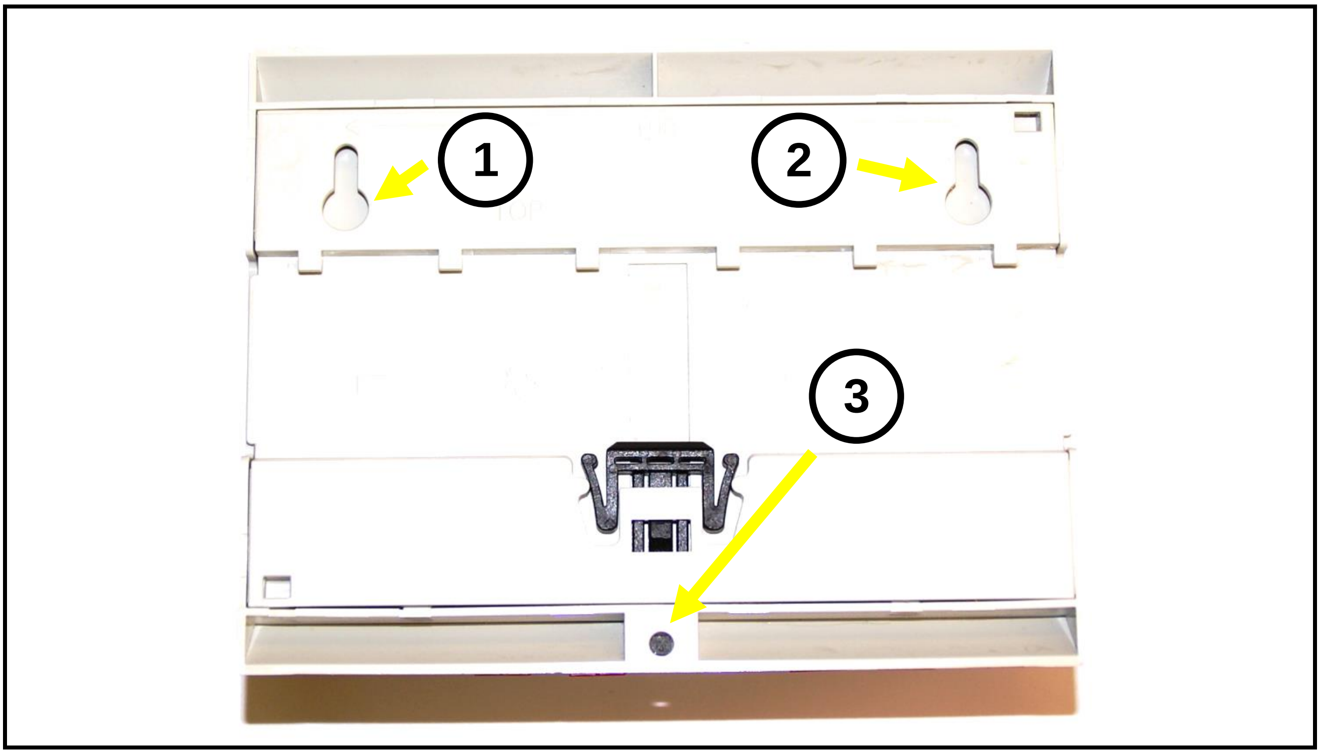


The module is correctly mounted, if the module has snapped into the DIN rail on both sides of the housing (6) and if the hook has snapped in too (7).

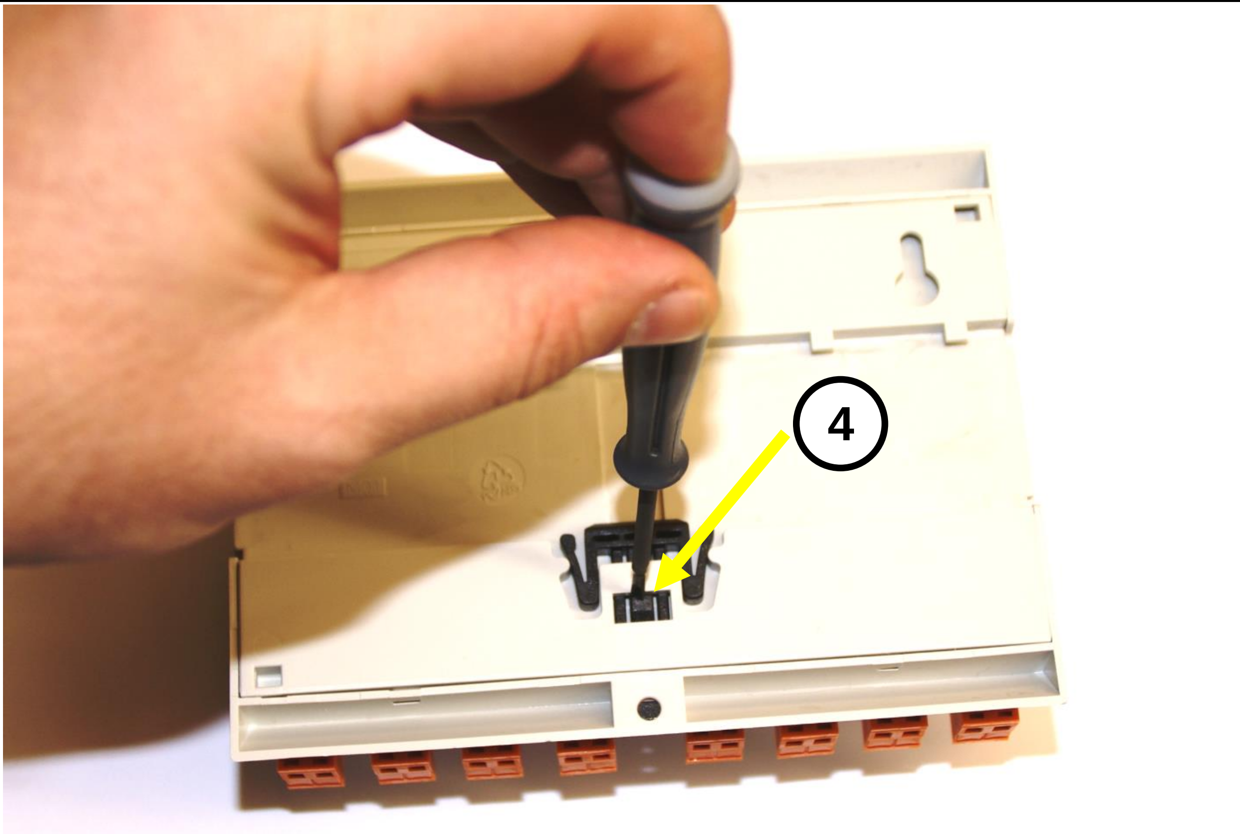


12.3.2 Wall mounting

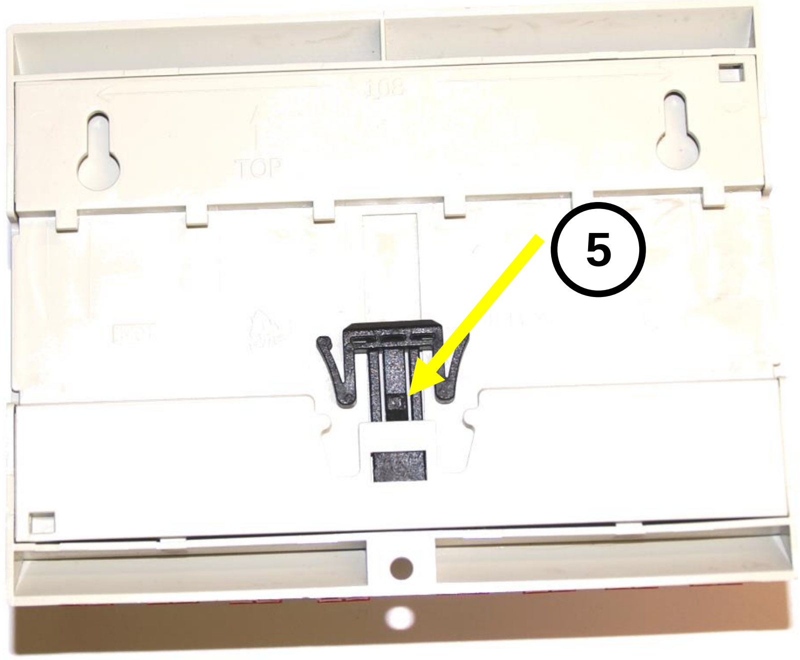
Our modules can also be mounted onto a wall. Turn over the module as shown in the picture below:

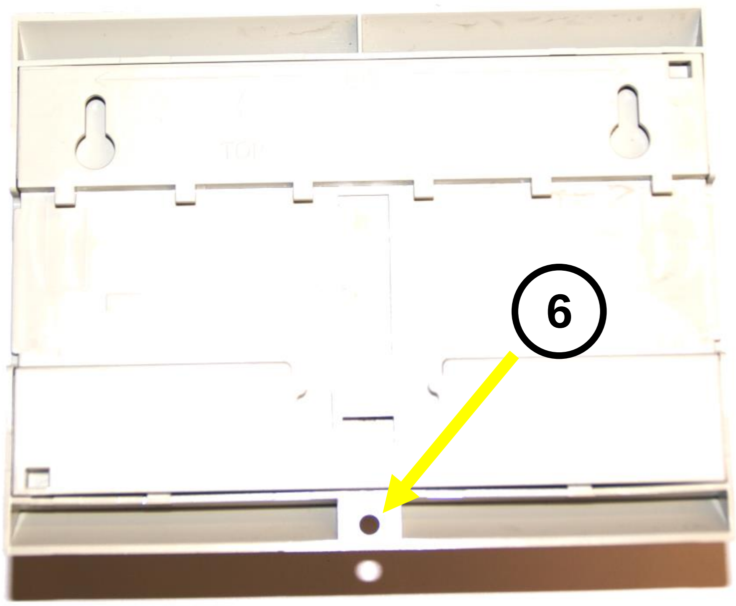


You will notice, that there are two holes for wall hooks or screws on the top side of the housing. (1) and (2). On the bottom side you will notice a small hole for a screw to fix the housing on the wall from the front (3). But first we have to remove the hook, which blocks the screw hole in the housing.



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.

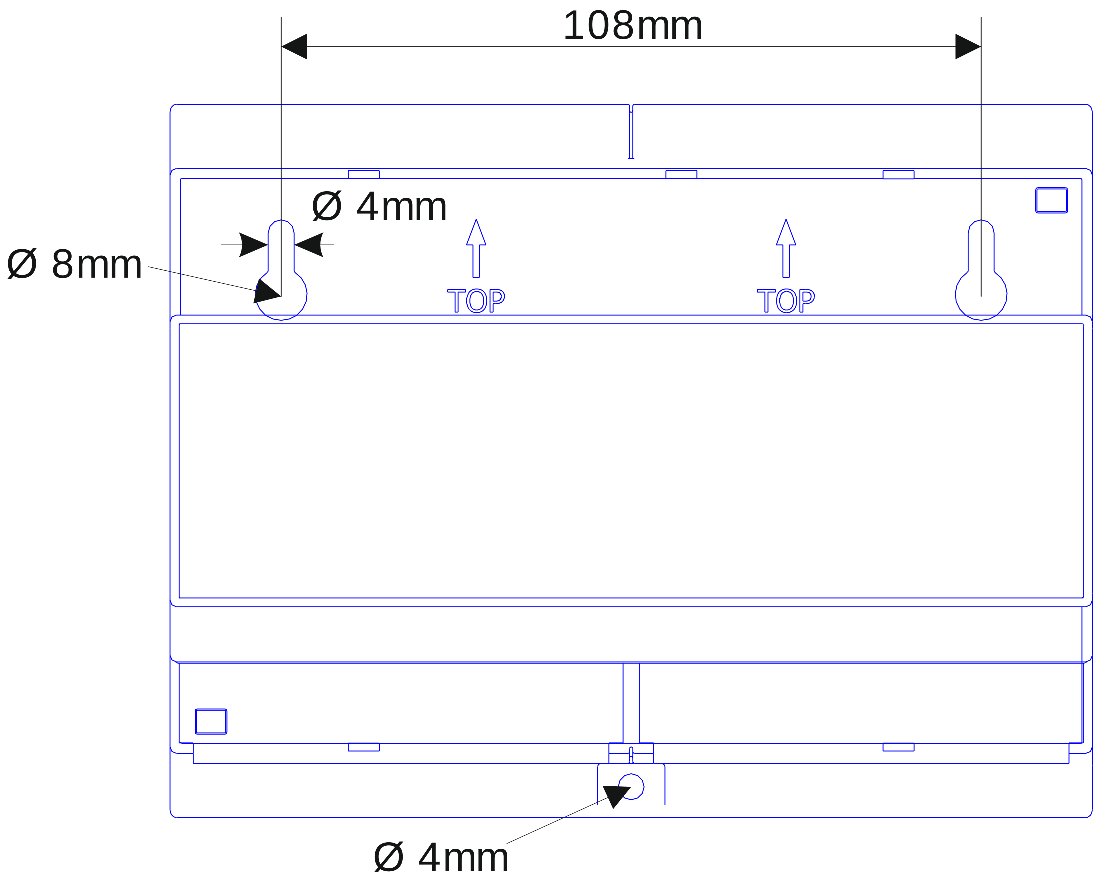


Illustration: Bottom view of the module with holes for wall mounting

12.4 Connection diagram

In the below drawing you will find the correct cabling of the IO module.

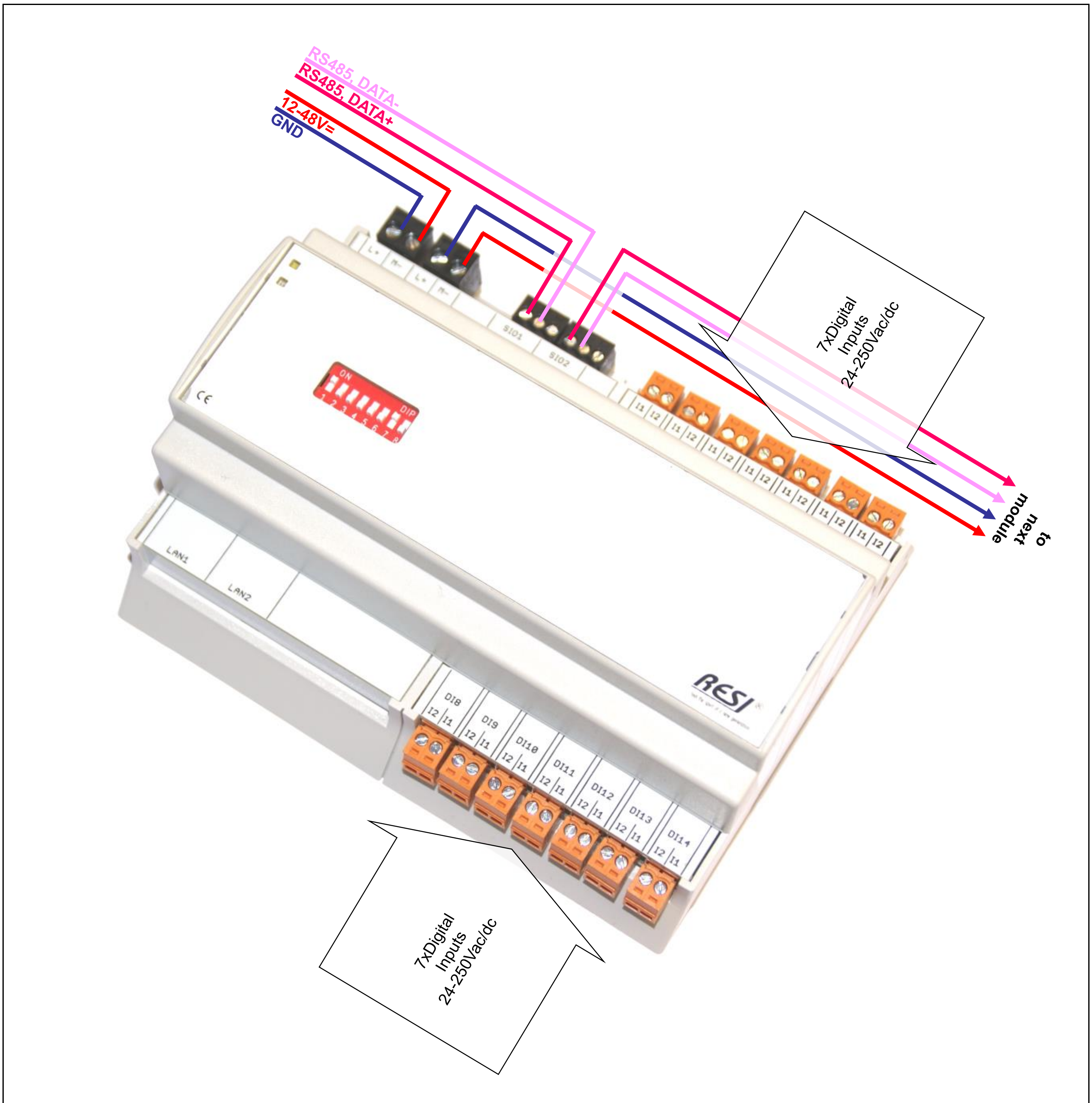


Illustration: cabling of the IO module

12.5 Clamps, DIP switch settings an LED indicators

The IO module offers the following clamps:

CLAMPS	**RESI-14RI-MODBUS, RESI-14RI-ASCII**
L+ M-	Power supply via two separated plug-in 2-pin terminal blocks. For daisy chain IN and OUT power supply of many modules L+: 12-48 V= M-: Ground
SIO1 1=A+ 2=B- 3=GND	RS485 ASCII or MODBUS/RTU serial interface IN A+: RS485 DATA+ signal B-: RS485 DATA- signal GND: RS485 ground signal
SIO2 1=A+ 2=B- 3=GND	RS485 ASCII or MODBUS/RTU serial interface OUT A+: RS485 DATA+ signal B-: RS485 DATA- signal GND: RS485 ground signal
DI1 I1=L+ or L I2=M- or N	Digital input 1 for AC/DC signals I1: AC/DC signal I2: Ground or neutral wire
DI2 I1=Signal + I2=Signal -	Digital input 2 for AC/DC signals I1: AC/DC signal I2: Ground or neutral wire
DI3 I1=Signal + I2=Signal -	Digital input 3 for AC/DC signals I1: AC/DC signal I2: Ground or neutral wire
DI4 I1=Signal + I2=Signal -	Digital input 4 for AC/DC signals I1: AC/DC signal I2: Ground or neutral wire
DI5 I1=Signal + I2=Signal -	Digital input 5 for AC/DC signals I1: AC/DC signal I2: Ground or neutral wire
DI6 I1=Signal + I2=Signal -	Digital input 6 for AC/DC signals I1: AC/DC signal I2: Ground or neutral wire
DI7 I1=Signal + I2=Signal -	Digital input 7 for AC/DC signals I1: AC/DC signal I2: Ground or neutral wire
DI8 I1=Signal + I2=Signal -	Digital input 8 for AC/DC signals I1: AC/DC signal I2: Ground or neutral wire
DI9 I1=Signal + I2=Signal -	Digital input 9 for AC/DC signals I1: AC/DC signal I2: Ground or neutral wire
DI10 I1=Signal + I2=Signal -	Digital input 10 for AC/DC signals I1: AC/DC signal I2: Ground or neutral wire
DI11 I1=Signal + I2=Signal -	Digital input 11 for AC/DC signals I1: AC/DC signal I2: Ground or neutral wire
DI12 I1=Signal + I2=Signal -	Digital input 12 for AC/DC signals I1: AC/DC signal I2: Ground or neutral wire
DI13 I1=Signal + I2=Signal -	Digital input 13 for AC/DC signals I1: AC/DC signal I2: Ground or neutral wire
DI14 I1=Signal + I2=Signal -	Digital input 14 for AC/DC signals I1: AC/DC signal I2: Ground or neutral wire

Table: Description of the terminal blocks of the IO module

The IO module offers also an 8-pin DIP switch and a dual color LED indicator:

<table>
<tr><th>DIP+LED</th><th>RESI-14RI-MODBUS, RESI-14RI-ASCII</th></tr>
<tr><td>DIP SWITCH
1=ADR0
2=ADR1
3=ADR2
4=ADR3
5=BR0
6=BR1
7=BR2
8=PARITY</td><td>DIP switch to setup the IO module
ADR: 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:
<table>
<tr><th>ADR3</th><th>ADR2</th><th>ADR1</th><th>ADR0</th><th>MODBUS/RTU unit number or ASCII bus number</th></tr>
<tr><td>OFF</td><td>OFF</td><td>OFF</td><td>OFF</td><td>Internal MODBUS/RTU unit number is used from the FLASH memory in the range of 0 to 255.</td></tr>
<tr><td>OFF</td><td>OFF</td><td>OFF</td><td>ON</td><td>1</td></tr>
<tr><td>OFF</td><td>OFF</td><td>ON</td><td>OFF</td><td>2</td></tr>
<tr><td>OFF</td><td>OFF</td><td>ON</td><td>ON</td><td>3</td></tr>
<tr><td>OFF</td><td>ON</td><td>OFF</td><td>OFF</td><td>4</td></tr>
<tr><td>OFF</td><td>ON</td><td>OFF</td><td>ON</td><td>5</td></tr>
<tr><td>OFF</td><td>ON</td><td>ON</td><td>OFF</td><td>6</td></tr>
<tr><td>OFF</td><td>ON</td><td>ON</td><td>ON</td><td>7</td></tr>
<tr><td>ON</td><td>OFF</td><td>OFF</td><td>OFF</td><td>8</td></tr>
<tr><td>ON</td><td>OFF</td><td>OFF</td><td>ON</td><td>9</td></tr>
<tr><td>ON</td><td>OFF</td><td>ON</td><td>OFF</td><td>10</td></tr>
<tr><td>ON</td><td>OFF</td><td>ON</td><td>ON</td><td>11</td></tr>
<tr><td>ON</td><td>ON</td><td>OFF</td><td>OFF</td><td>12</td></tr>
<tr><td>ON</td><td>ON</td><td>OFF</td><td>ON</td><td>13</td></tr>
<tr><td>ON</td><td>ON</td><td>ON</td><td>OFF</td><td>14</td></tr>
<tr><td>ON</td><td>ON</td><td>ON</td><td>ON</td><td>15</td></tr>
</table>
BAUD RATE: Those three DIP switches BR2-BR0 define the MODBUS/RTU or ASCII baud rate for the communication:
<table>
<tr><th>BR2</th><th>BR1</th><th>BR0</th><th>MODBUS/RTU or ASCII Baudrate</th></tr>
<tr><td>OFF</td><td>OFF</td><td>OFF</td><td>4800bd</td></tr>
<tr><td>OFF</td><td>OFF</td><td>ON</td><td>9600bd</td></tr>
<tr><td>OFF</td><td>ON</td><td>OFF</td><td>19200bd</td></tr>
<tr><td>OFF</td><td>ON</td><td>ON</td><td>38400bd</td></tr>
<tr><td>ON</td><td>OFF</td><td>OFF</td><td>57600bd</td></tr>
<tr><td>ON</td><td>OFF</td><td>ON</td><td>115200bd</td></tr>
<tr><td>ON</td><td>ON</td><td>OFF</td><td>230400bd</td></tr>
<tr><td>ON</td><td>ON</td><td>ON</td><td>256000bd</td></tr>
</table>
PARITY: This DIP switch PARITY defines the MODBUS/RTU or ASCII parity for the communication:
<table>
<tr><th>PARITY</th><th>MODBUS/RTU or ASCII parity</th></tr>
<tr><td>OFF</td><td>NONE</td></tr>
<tr><td>ON</td><td>EVEN</td></tr>
</table>
HINT: After changing on of the DIP switches, the module restarts completely and intialises the serial interface. You will notice that the WHITE LED will be on for approximately 2 seconds, before this LED will flash with a one second cycle.</td></tr>
<tr><td>LED WHITE</td><td>This LED will flash with a cycle of 1 seconds to show normal mode of the module</td></tr>
<tr><td>LED GREEN</td><td>This LED will flash shortly, whenever the module receives a valid telegram on the serial interface.</td></tr>
<tr><td>LED RED</td><td>If this LED flashes cyclically, there is a module error detected by the firmware</td></tr>
</table>

Table: Description of the DIP switch functions and the indication LEDs on the IO module

12.7 Dimensions of the module

In the below drawing you will find the dimensions of the IO module.

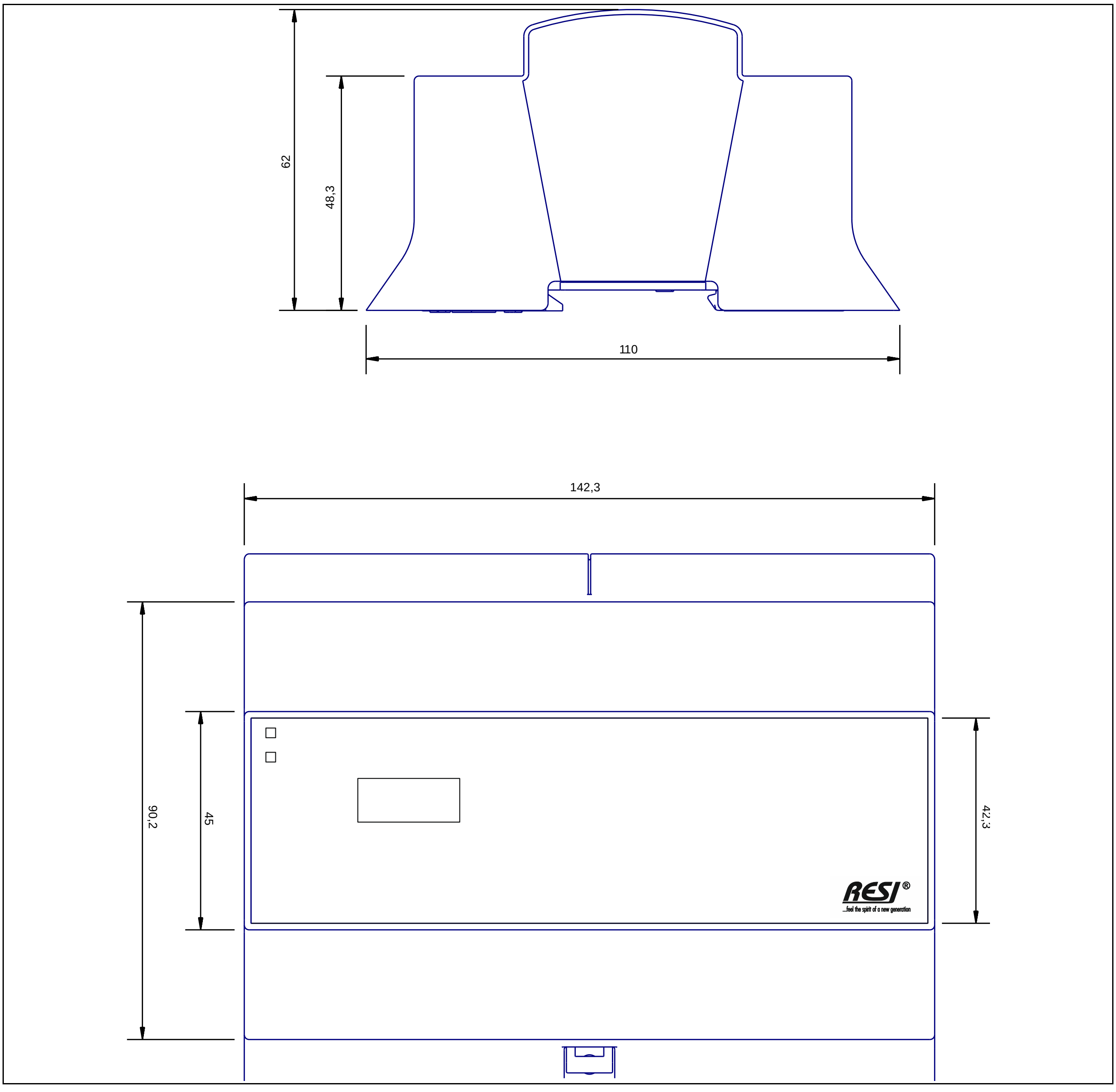


Illustration: Dimensions of the IO module in mm

Dimensions	
Dimensions of the housing L x B x H (mm)	143 x 110 x 62
Weight	265 g
Color	Grey, RAL7035
Material	Self-extinguish PC/ABS, DIN 43880
Protection class	IP20 based on DIN 40050/EN 60529

Table: technical data of the housing

12.8 Power supply of the module

In the below drawing you will find how to connect the module to a power supply.

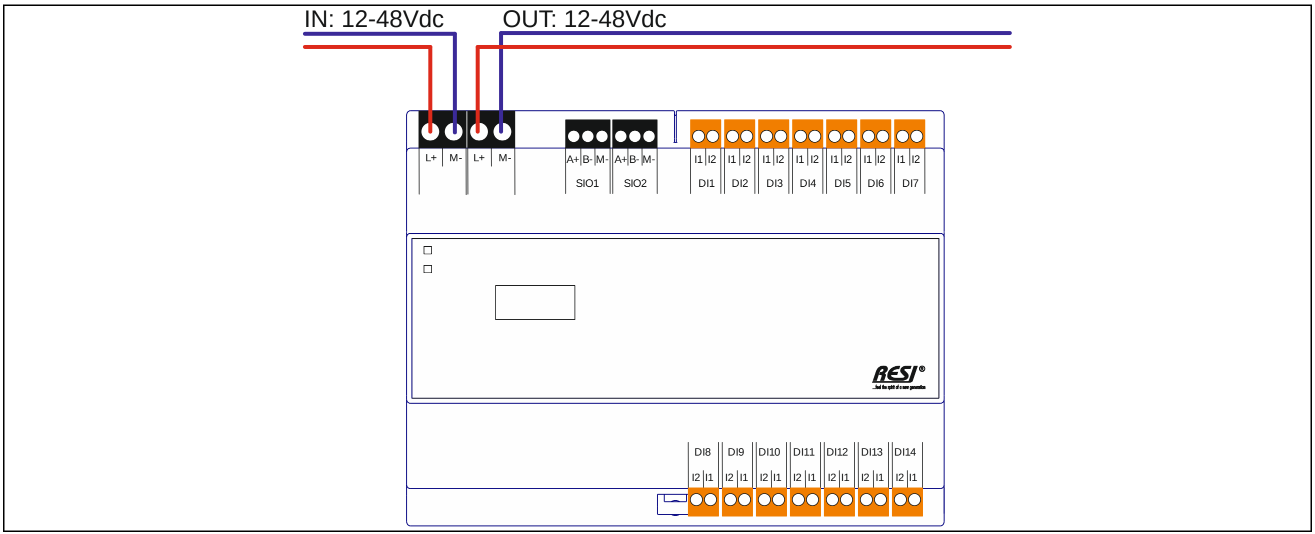


Illustration: Power supply of the IO module

The module offers two 2-pin plug-in terminals for connecting the power supply to the module. It is designed to create a daisy chain power supply with many modules.

12.9 Serial RS485 connection

In this drawing you see the cabling of the serial RS485 bus line. In the module both SIO terminal block are bridged.

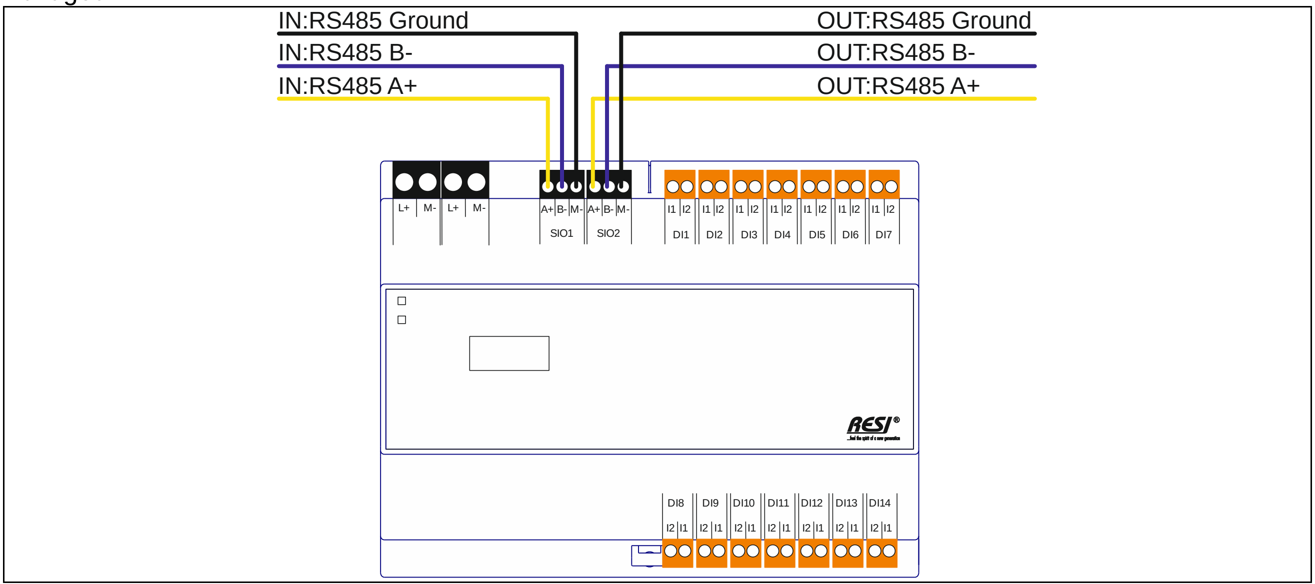


Illustration: RS485 bus cabling of the IO module

The module offers two plug-in 3-pin terminals to connect a RS485 bus line to the module. It was designed to create a daisy chain bus line with many modules. Don't forget, that a RS485 bus line needs a line termination at the end of both lines!

12.10 Cabling of the digital inputs of the module with DC signals

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

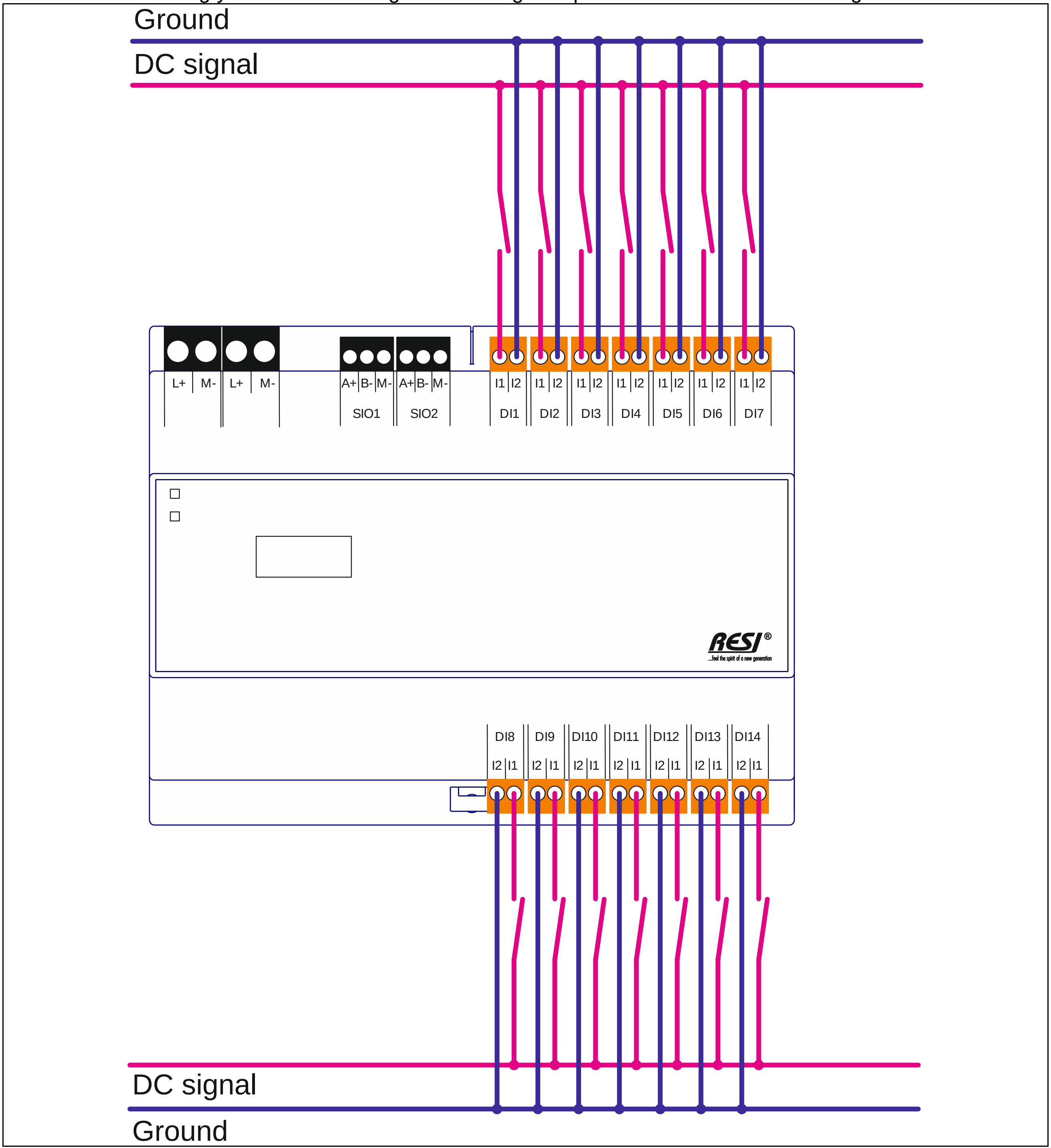


Illustration: Cabling of the digital inputs of the IO module with DC signals

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!

12.11 Cabling of the digital inputs of the module with AC signals

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

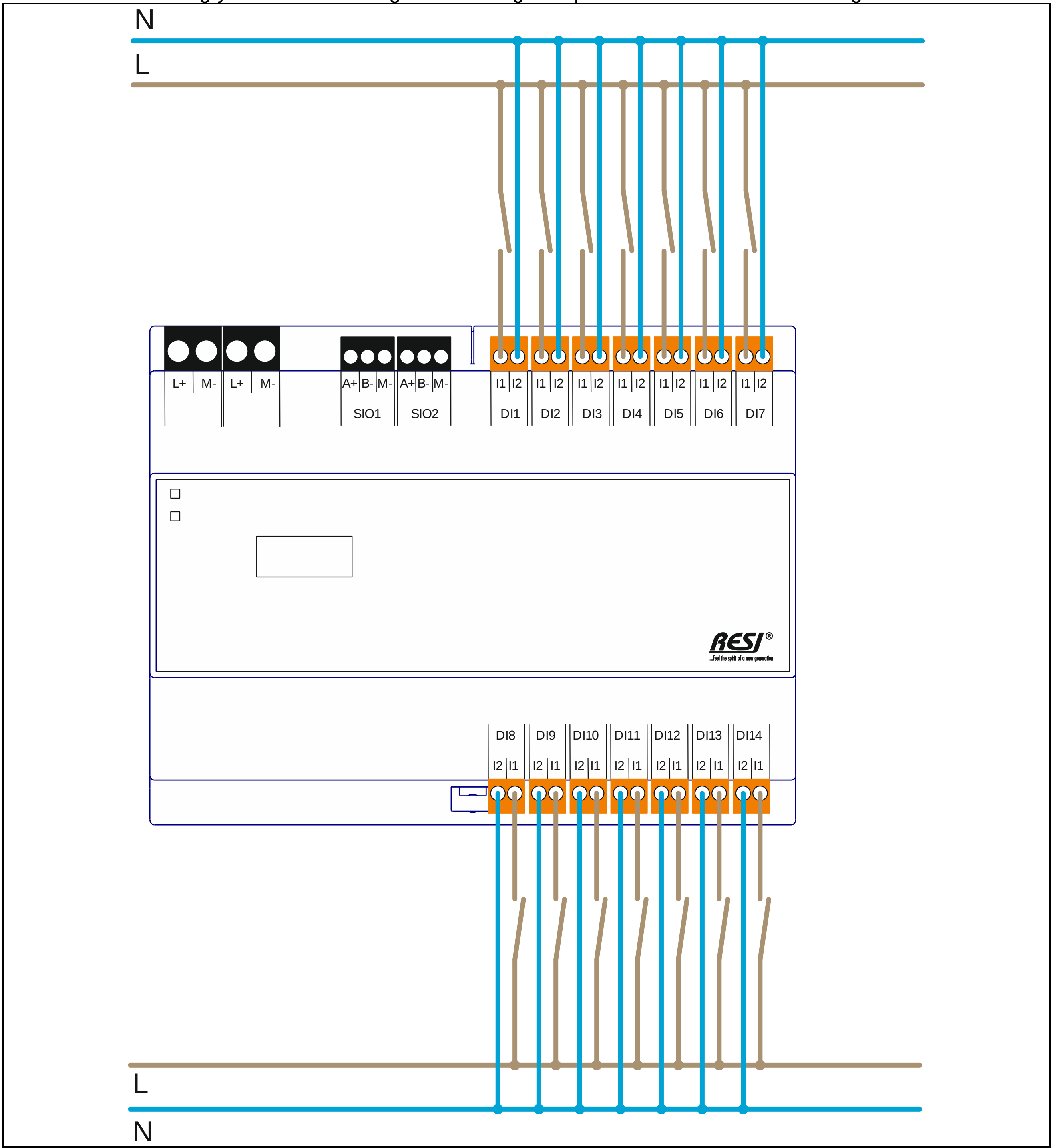


Illustration: Cabling of the digital inputs of the IO module with AC signals

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!

12.12 Functional description

This IO module communicates with a host system with the MODBUS/RTU slave protocol. All versions of the Io module with RESI-xxx-ASCII offer an additional protocol: An ASCII slave protocol with simple text string. The communication runs over a RS485 interface (half duplex) or over a RS232 interface (full duplex).

For the communication with ASCII texts, the host sends ASCII messages with a special start character # (0x23, 35dec) and a special end character (0x0d,13dec or CARRIAGE RETURN) to the module. The module uses also this special start and end characters to answer to the host request. Consult the below noted detailed command descriptions. In the ASCII protocol mode, the host can send messages with or without a bus number.

For communication with the MODBUS/RTU slave, the module offers the following MODBUS functions:

- 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)

HINT:
The functions READ HOLDING REGISTER and PRESET MULTIPLE REGISTERS are restricted to max. 125 register per request! The functions READ INPUT STATUS, READ COIL STATUS and FORCE MULTIPLE COILS are restricted to 2000 coils or inputs (bits) per request.

12.13 ASCII protocol description

12.13.1 Overview

The IO module communicates with simple ASCII commands. The following special characters are used in this documentation:

stand for the **Hashtag** ASCII character 35dec or 0x23
: stand for the **colon** ASCII character 58dec or 0x3A
= stand for the **equal sign** with the ASCII code 61ec or 0x3D
- stand for the **minus sign** with the ASCII code 45dec or 0x2D
, stands for the **comma** with the ASCII code 44dec or 0x2C
<CR> stand for **CARRIAGE RETURN** ASCII character 13dec or 0x0D. In the following text we use the representation CR.
<SP> stands for the **SPACE**. This is the space character with the ASCII code 32dec or 0x20. In the following text we will use the representation☐.
<ADR> will be used as the current **bus address** of the module. The bus number can be transmitted decimal or hexadecimal and is separated with a comma (ASCII code 44dec or 0x2C) from the rest of the command. Hexadecimal numbers always start with 0x. Its only allowed to use the ASCII characters ‚0‘-‚9‘ 48dec to 57dec, 0x30-0x39 and ‚A‘ to ‚F‘, 65dec to 70dec, 0x41-0x46. All modules react to the broadcast bus address 0 and to its own bus number. With a DIP switch, the user can easily change between the internal stored bus number in the FLASH and the fix bus number 255. Consult the DIP switch description for more details.

12.13.2 Communication sequence

In general the IO module sends no ASCII characters without a request from a host. So the host is the master of the communication and the IO module is always answering to host requests as a slave. If only one IO module is connected to a host (e.g. Via RS232), you can dump the bus number in the protocol. If you use a RS485 interface, more than one module can be connected to the host. Therefore a bus number in the request frame of the host is always necessary.

The structure of the command look like this:

The host sends a command or a command with parameters without a bus address to the module:
#<Command><CR> or
#<Command>:<Parameter><CR>

The module answers, if it feels addressed, with the following answer telegram:
#<Answer><CR>

If the bus number is used, the module answers with:
#<ADR>,<Command><CR> or
#<ADR>,<Command>:<Parameter><CR>

The module answers with:
#<ADR>,<Answer><CR>

The bus address lies in the range of 1dec to 255dec or 0x00 to 0xFF hexadecimal. The setup is done with our free configuration software MODBUSConfigurator. Each module reacts also to the broadcast bus address 0.

For each command, we define two different writings. A long version and a short version, to avoid unnecessary traffic on the bus. For example to request the software version of the module you can use the command VERSION or the short command VER.

12.13.3 Request VERSION

This command returns the current software version of the module.

Host long version:
#VERSION<CR> or
#<ADR>,VERSION<CR>

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

Answer:
#VERSION:<HIGH>.<MED>.<LOW><CR> or
#<ADR>,VERSION:<HIGH>,<MED>,<LOW><CR>

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

Samples:
→ **#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**$_{CR}$

12.13.4 Request module TYPE

This command returns the current type of the module.

Host long version:
#TYPE<CR> or
#<ADR>,TYPE<CR>

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

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

<TYP> defines the current type of the module. Currently RESI-14RI-ASCII

Samples:
→ **#TYPE**$_{CR}$
← **#TYPE:RESI-14RI-ASCII**$_{CR}$

→ **#255,TYP**$_{CR}$
← **#255,TYPE:RESI-14RI-ASCII**$_{CR}$

12.13.5 Table of all ASCII commands

Here you will find a possible ASCII commands of the module. We use here only the version with bus number. That you can avoid the bus number, we have discussed earlier in this document. If an argument as the extension Dec, it will be returned as a decimal number, If an argument has the extension Hex, then this argument is returned as a hexadecimal number. Many command returns the argument in decimal and hexadecimal representation. So the host can select, what kind of number conversion, it will handle in its software.

Direction	**ASCII command**
Host	**#<BusAdr>,VER$_{CR}$** **#<BusAdr>,VERSION$_{CR}$**
Answer	**#<BusAdr>,VERSION:<VersionHi>.< VersionMed>.<VersionLo>$_{CR}$**
	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)
Host	**#<BusAdr>,TYP$_{CR}$** **#<BusAdr>,TYPE$_{CR}$**
Answer	**#<BusAdr>,TYPE:RESI-14RI-ASCII$_{CR}$**
	Returns the current type of the module
Host	**#<BusAdr>,OWN$_{CR}$** **#<BusAdr>,OWNER$_{CR}$**
Answer	**#<BusAdr>,OWNER:RESI$_{CR}$**
	Returns the owner of the module
Host	**#<BusAdr>,CRE$_{CR}$** **#<BusAdr>,CREATOR$_{CR}$**
Answer	**#<BusAdr>,CREATOR:DI HC SIGL,MSC$_{CR}$**
	Returns the creator of the module

Direction	**ASCII command**
Host	**#<BusAdr>,COPY$_{CR}$** **#<BusAdr>,COPYRIGHT$_{CR}$**
Answer	**#<BusAdr>,COPYRIGHT:2015-16 BY RESI AND DI HC SIGL,MSC WWW.RESI.CC$_{CR}$**
	Returns a copyright note about the module
Host	**#<BusAdr>,GDIP$_{CR}$** **#<BusAdr>,GET□DIP$_{CR}$**
Answer	**#<BusAdr>,GDIP:<DIPSwitchDec>,<DIPSwitchHex>$_{CR}$**
	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 (=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)
Host	**#<BusAdr>,GDIS$_{CR}$** **#<BusAdr>,GET□DIS$_{CR}$**
Answer	**#<BusAdr>,GDIS:<DISDec>,<DISHex>$_{CR}$**
	Returns the current state of all 32 digital inputs as decimal number and as hexadecimal number. DISDec DISHex The current state of all digital inputs: Bit 0: State of DI1 (=0:OFF, =1:ON) Bit 1: State of DI2 (=0:OFF, =1:ON) Bit 2: State of DI3 (=0:OFF, =1:ON) Bit 3: State of DI4 (=0:OFF, =1:ON) Bit 4: State of DI5 (=0:OFF, =1:ON) Bit 5: State of DI6 (=0:OFF, =1:ON) Bit 6: State of DI7 (=0:OFF, =1:ON) Bit 7: State of DI8 (=0:OFF, =1:ON) Bit 8: State of DI9 (=0:OFF, =1:ON) Bit 9: State of DI10 (=0:OFF, =1:ON) Bit 10: State of DI11 (=0:OFF, =1:ON) Bit 11: State of DI12 (=0:OFF, =1:ON) Bit 12: State of DI13 (=0:OFF, =1:ON) Bit 13: State of DI14 (=0:OFF, =1:ON)

Direction	ASCII command
Host	**#<BusAdr>,GDIx**$_{CR}$ **#<BusAdr>,GET☐DIx**$_{CR}$
Answer	**#<BusAdr>,GDIx:<DIxDec>,<DIxHex>**$_{CR}$
x	**1..14**
	Returns the current state of the digital input Dix as decimal number and as hexadecimal number. X stands for the desired digital input between 1 and 14. DIxDec DIxHex The current state of the digital input x: =0: Digital input is OFF =1: Digital input is ON
Host	**#<BusAdr>,RDIx**$_{CR}$ **#<BusAdr>,RISE☐DIx**$_{CR}$
Answer	**#<BusAdr>,RDIx:<RDIxDec>,<RDIxHex>**$_{CR}$
x	**1..14**
	Returns the current counter for rising edges on the digital input x since last power on of the module as decimal number and as hexadecimal number. RDIxDec RDIxHex The current amount of counted rising edges on the digital input x
Host	**#<BusAdr>,FDIx**$_{CR}$ **#<BusAdr>,FALL☐DIx**$_{CR}$
Answer	**#<BusAdr>,FDIx:<FDIxDec>,<FDIxHex>**$_{CR}$
x	**1..14**
	Returns the current counter for falling edges on the digital input x since last power on of the module as decimal number and as hexadecimal number. FDIxDec FDIxHex The current amount of counted falling edges on the digital input x
Host	**#<BusAdr>,RC**$_{CR}$ **#<BusAdr>,RESET☐COUNTERS**$_{CR}$
Answer	**#<BusAdr>,OK**$_{CR}$
	This command deletes all counters for rising and falling edges of the 14 digital inputs in the module to 0.

Direction	**ASCII command**
Host	**#<BusAdr>,SMBADR:<MBUnit>$_{CR}$** **#<BusAdr>,SET□MODBUS□ADDRESS:<MBUnit>$_{CR}$**
Answer	**#<BusAdr>,OK $_{CR}$**
	Writes the unit address into the FLASH memory of the module. The new unit address for MODBUS/RTU or ASCII mode is only used immediately, if the DIP switch setting of the bus address is 0. Otherwise the unit address is defined by the DIP settings. The unit address ranges from 0dec to 255dec.
Host	**#<BusAdr>,GMBADR$_{CR}$** **#<BusAdr>,GET□MODBUS□ADDRESS$_{CR}$**
Answer	**#<BusAdr>,GMBADR:<MBUnitDec>,<MBFLASHDec>,<MBUnitHex>,<MBFLASHHex>$_{CR}$**
	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. MBUnitDec MBUnitHex — The current used MODBUS/RTU unit or ASCII address for communication MBFLASHDec MBFLASHHex — The internal stored MODBUS/RTU unit address or ASCII address from the FLASH memory, if the DIP switch is 0.
Host	**#<BusAdr>,RST$_{CR}$** **#<BusAdr>,RESET$_{CR}$**
Answer	**none**
	Executes a software reset (Reboot) of the module.

12.14 MODBUS – register description

12.14.1 Table of inputs and coils

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 0x00001 to 0x65536. Inputs are usually noted with 1x00001 to 1x65536. 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: 0x00100 for the coil 100, 1x00100 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.

Register	Description
0x00001 1x00001 I:0 R/O DI1	Current state of the digital input DI1 =0:DI is OFF, =1:DI is ON
0x00002 1x00002 I:1 R/O DI2	Current state of the digital input DI2 =0:DI is OFF, =1:DI is ON
0x00003 1x00003 I:2 R/O DI3	Current state of the digital input DI3 =0:DI is OFF, =1:DI is ON
0x00004 1x00004 I:3 R/O DI4	Current state of the digital input DI4 =0:DI is OFF, =1:DI is ON
0x00005 1x00005 I:4 R/O DI5	Current state of the digital input DI5 =0:DI is OFF, =1:DI is ON
0x00006 1x00006 I:5 R/O DI6	Current state of the digital input DI6 =0:DI is OFF, =1:DI is ON
0x00007 1x00007 I:6 R/O DI7	Current state of the digital input DI7 =0:DI is OFF, =1:DI is ON
0x00008 1x00008 I:7 R/O DI8	Current state of the digital input DI8 =0:DI is OFF, =1:DI is ON

Register	Description
0x00009 1x00009 I:8 R/O DI9	Current state of the digital input DI9 =0:DI is OFF, =1:DI is ON
0x00010 1x00010 I:9 R/O DI10	Current state of the digital input DI10 =0:DI is OFF, =1:DI is ON
0x00011 1x00011 I:10 R/O DI11	Current state of the digital input DI11 =0:DI is OFF, =1:DI is ON
0x00012 1x00012 I:11 R/O DI12	Current state of the digital input DI12 =0:DI is OFF, =1:DI is ON
0x00013 1x00013 I:12 R/O DI13	Current state of the digital input DI13 =0:DI is OFF, =1:DI is ON
0x00014 1x00014 I:13 R/O DI14	Current state of the digital input DI14 =0:DI is OFF, =1:DI is ON

Register	Description
0x00033 1x00033 I:32 R/O DIP1	Current state of DIP switch 1 =0:Dip switch is OFF, =1: Dip switch is ON
0x00034 1x00034 I:33 R/O DIP2	Current state of DIP switch 2 =0:Dip switch is OFF, =1: Dip switch is ON
0x00035 1x00035 I:34 R/O DIP3	Current state of DIP switch 3 =0:Dip switch is OFF, =1: Dip switch is ON
0x00036 1x00036 I:35 R/O DIP4	Current state of DIP switch 4 =0:Dip switch is OFF, =1: Dip switch is ON
0x00037 1x00037 I:36 R/O DIP5	Current state of DIP switch 5 =0:Dip switch is OFF, =1: Dip switch is ON
0x00038 1x00038 I:37 R/O DIP6	Current state of DIP switch 6 =0:Dip switch is OFF, =1: Dip switch is ON
0x00039 1x00039 I:38 R/O DIP7	Current state of DIP switch 7 =0:Dip switch is OFF, =1: Dip switch is ON
0x00040 1x00040 I:39 R/O DIP8	Current state of DIP switch 8 =0:Dip switch is OFF, =1: Dip switch is ON
0x00100 1x00100 I:99 R/W RESET COUNTER	Reset, resetting the internal edge counters for all digital inputs. While reading always 0.

12.14.2 Table of holding/input registers

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.

Register	Description
4x00001 3x00001 I:0 R/O RISE DI1	Counter for rising edges on the digital input DI1. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00002 3x00002 I:1 R/O FALL DI1	Counter for falling edges on the digital input DI1. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00003 3x00003 I:2 R/O RISE DI2	Counter for rising edges on the digital input DI2. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00004 3x00004 I:3 R/O FALL DI2	Counter for falling edges on the digital input DI2. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00005 3x00005 I:4 R/O RISE DI3	Counter for rising edges on the digital input DI3. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00006 3x00006 I:5 R/O FALL DI3	Counter for falling edges on the digital input DI3. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00007 3x00007 I:6 R/O RISE DI4	Counter for rising edges on the digital input DI4. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00008 3x00008 I:7 R/O FALL DI4	Counter for falling edges on the digital input DI4. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00009 3x00009 I:8 R/O RISE DI5	Counter for rising edges on the digital input DI5. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00010 3x00010 I:9 R/O FALL DI5	Counter for falling edges on the digital input DI5. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00011 3x00011 I:10 R/O RISE DI6	Counter for rising edges on the digital input DI6. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00012 3x00012 I:11 R/O FALL DI6	Counter for falling edges on the digital input DI6. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.

Register	Description
4x00013 3x00013 I:12 R/O RISE DI7	Counter for rising edges on the digital input DI7. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00014 3x00014 I:13 R/O FALL DI7	Counter for falling edges on the digital input DI7. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00015 3x00015 I:14 R/O RISE DI8	Counter for rising edges on the digital input DI8. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00016 3x00016 I:15 R/O FALL DI8	Counter for falling edges on the digital input DI8. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00017 3x00017 I:16 R/O RISE DI9	Counter for rising edges on the digital input DI9. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00018 3x00018 I:17 R/O FALL DI9	Counter for falling edges on the digital input DI9. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00019 3x00019 I:18 R/O RISE DI10	Counter for rising edges on the digital input DI10. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00020 3x00020 I:19 R/O FALL DI10	Counter for falling edges on the digital input DI10. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00021 3x00021 I:20 R/O RISE DI11	Counter for rising edges on the digital input DI11. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00022 3x00022 I:21 R/O FALL DI11	Counter for falling edges on the digital input DI11. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00023 3x00023 I:22 R/O RISE DI12	Counter for rising edges on the digital input DI12. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00024 3x00024 I:23 R/O FALL DI12	Counter for falling edges on the digital input DI12. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.

Register	Description
4x00025 3x00025 I:24 R/O RISE DI13	Counter for rising edges on the digital input DI13. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00026 3x00026 I:25 R/O FALL DI13	Counter for falling edges on the digital input DI13. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00027 3x00027 I:26 R/O RISE DI14	Counter for rising edges on the digital input DI14. If the module detects a rising edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00028 3x00028 I:27 R/O FALL DI14	Counter for falling edges on the digital input DI14. If the module detects a falling edge on the digital input, this counter is incremented by 1. After power on or a soft reset this counter is set always to 0. With the function RESET COUNTER this counter is also set to 0.
4x00100 3x00100 I:99 R/W RESET COUNTER	Reset, resetting the internal edge counters for all digital inputs. While reading always 0.

Register	Description
4x00101 3x00101 I:100 R/O DIS	Current state of digital inputs 1..14 Bit 0: =0:DI1 is OFF, =1:DI1 is ON Bit 1: =0:DI2 is OFF, =1:DI2 is ON Bit 2: =0:DI3 is OFF, =1:DI3 is ON Bit 3: =0:DI4 is OFF, =1:DI4 is ON Bit 4: =0:DI5 is OFF, =1:DI5 is ON Bit 5: =0:DI6 is OFF, =1:DI6 is ON Bit 6: =0:DI7 is OFF, =1:DI7 is ON Bit 7: =0:DI8 is OFF, =1:DI8 is ON Bit 8: =0:DI9 is OFF, =1:DI9 is ON Bit 9: =0:DI10 is OFF, =1:DI10 is ON Bit 10: =0:DI11 is OFF, =1:DI11 is ON Bit 11: =0:DI12 is OFF, =1:DI12 is ON Bit 12: =0:DI13 is OFF, =1:DI13 is ON Bit 13: =0:DI14 is OFF, =1:DI14 is ON Bit 14: free (=0) Bit 15: free (=0)
4x00102 3x00102 I:101 R/O FREE	Currently unused, always 0
4x00103 3x00103 I:102 R/O DIP	Current state of the DIP switch 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) Bit 8-15: always 0

13 RESI-8CO-MODBUS, RESI-8CO-ASCII

13.1 Product description

This IO module offers the following features:

- 8 mono stable relay outputs with special power relays
- 3 clamps per relay: NO contact, NC contact and common root contact (C)
- Switching power per relay output: max. 30Vdc, max. 250Vac, max. 8A
- Contact material $AgSnO_2$
- Galvanic insulated RS485 interface for communication with a host system
- RESI-8CO-MODBUS: MODBUS/RTU slave protocol
- RESI-8CO-ASCII: MODBUS/RTU slave and text based ASCII protocol
- Power supply 12-48Vdc
- DIP switch for setting the baud rates, the type of interface and the bus number
- LED indicator for the communication
- Mounting onto EN50022 DIN rail or wall mounting

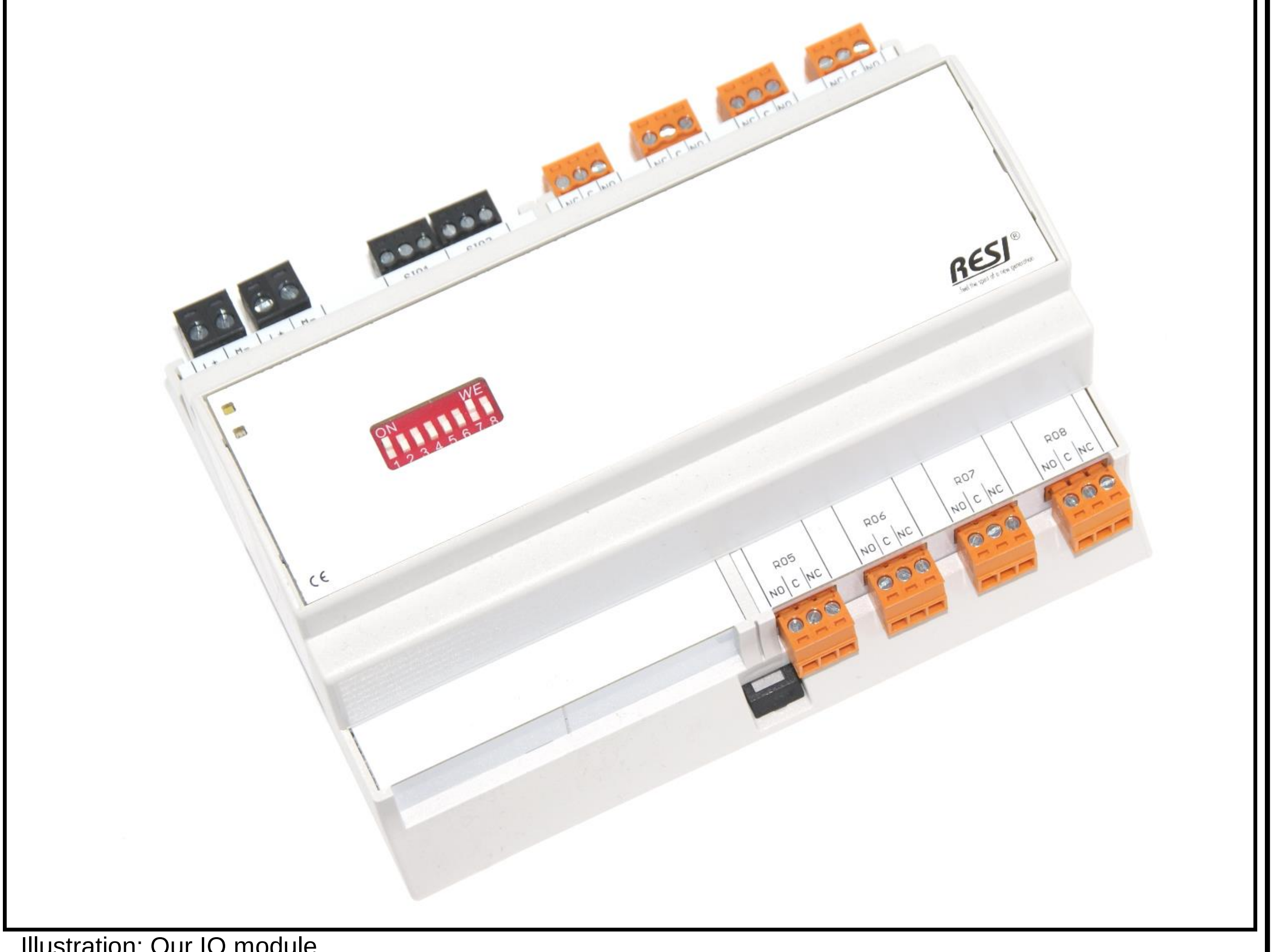


Illustration: Our IO module

13.2 Technical data

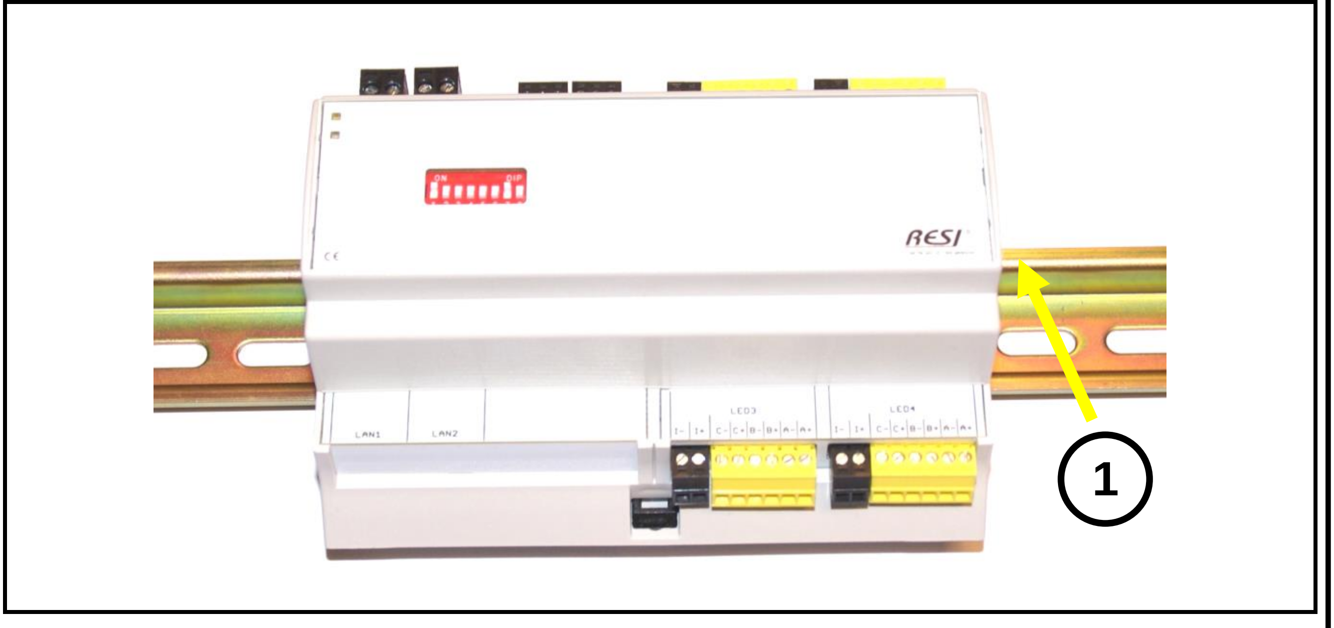
Technical Data			
Power supply			
Supply voltage	12-48V= +/-10%	Storage temperature	-20...85 °C
Power LED	Ja	Operating Temperature	0...60°C
Power consumption	<2.5W	Humidity	25...90 % rH non-condensing
		Protection Class	IP20 (EN 60529)
		Dimensions LxWxH	143mm x 110mm x 62mm
		Weight	300g
		Mounting	On DIN EN50022 rail or wall mounting
ASCII/Modbus Interface		**Relay outputs**	
Protocol	ASCII or Modbus/RTU	Amount of outputs	8
Type	RS485	Relay type	mono stable relay with contacts for NO clamp, NC clamp and common root clamp
Baud rates	4800 to 256000Bd/8/N or E/1	Maximum voltage	250Vac or 30Vdc
Cable Connection	Via removable clamps	Maximum current	8A
LED indicator	Yes	switching cycles	10^7 switching cycles
Galvanic insulation	No	Contact material	$AgSnO_2$
		Cable connection	via 8 2pin removable clamps in orange
		Galvanic insulation	Yes, through the relays
Clamps			
Clamp wire cross section	Max. 1,5 mm²		
Tightening torque	Max. 0.5Nm	**CE conformity**	Yes

13.3 Assembling

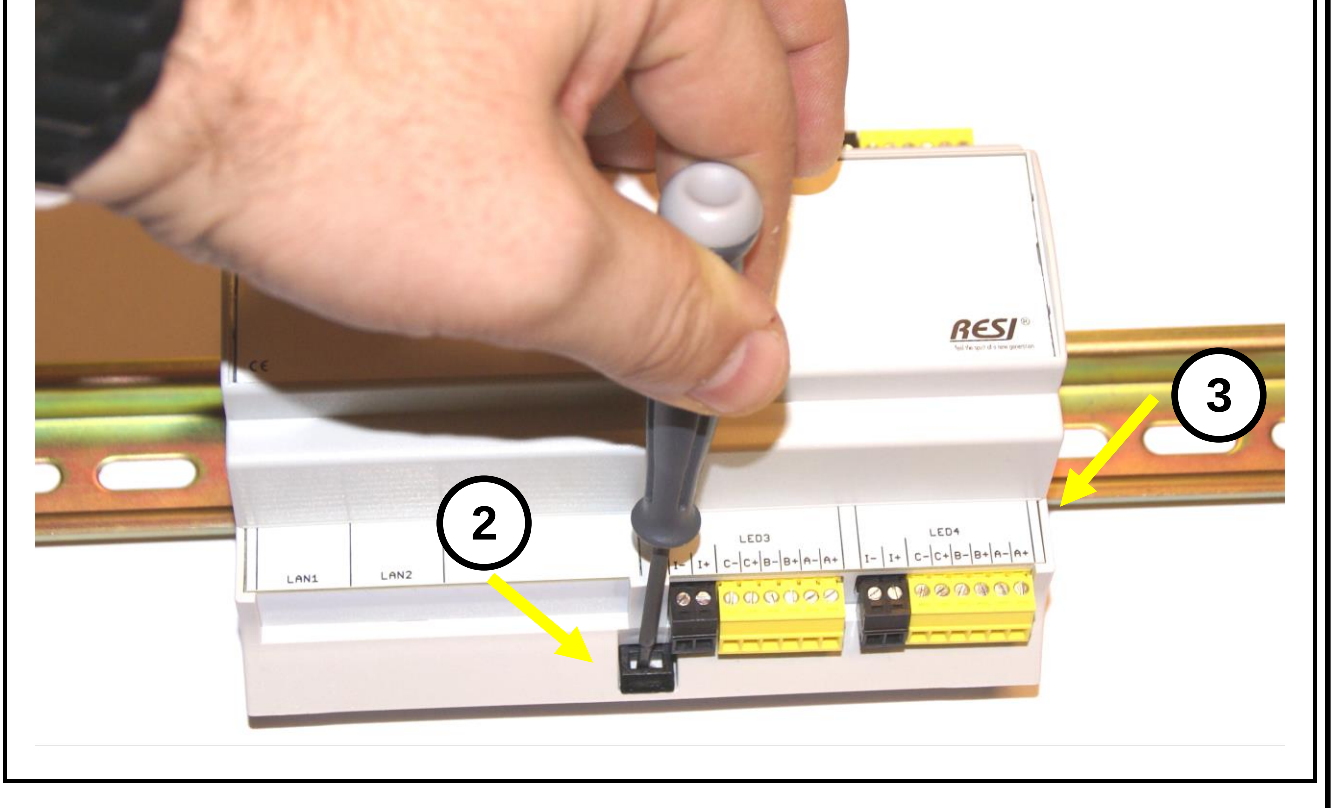
Our IO modules are designed for mounting onto a 35mm DIN-EN50022 rail or for wall mounting. Please not, that in the following mounting description we use only symbolic photos of our IO modules.

13.3.1 Mounting of a DIN EN50022 rail

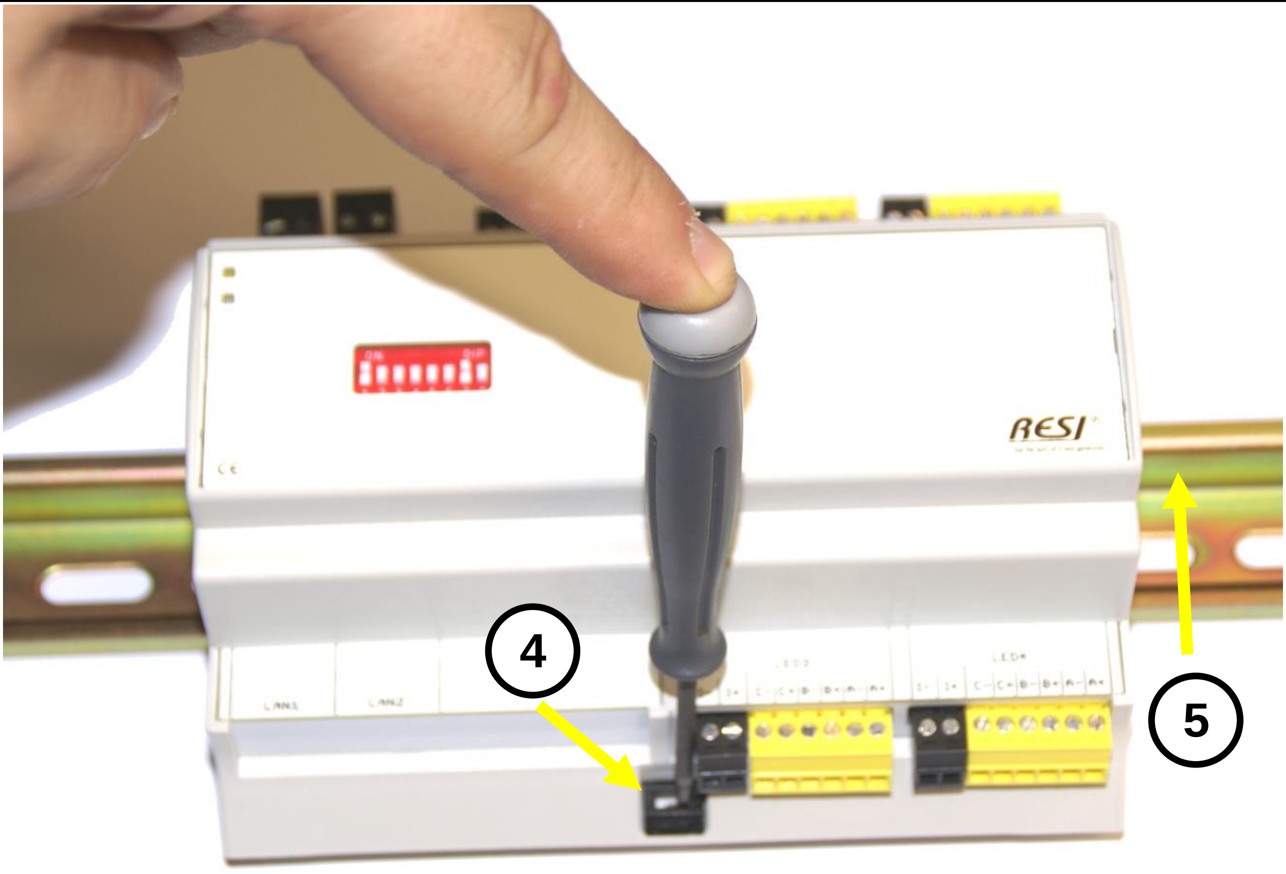
First snap in the top part of the module into the DIN rail (1). The bottom part of the module is not snapped into the DIN rail at this moment.



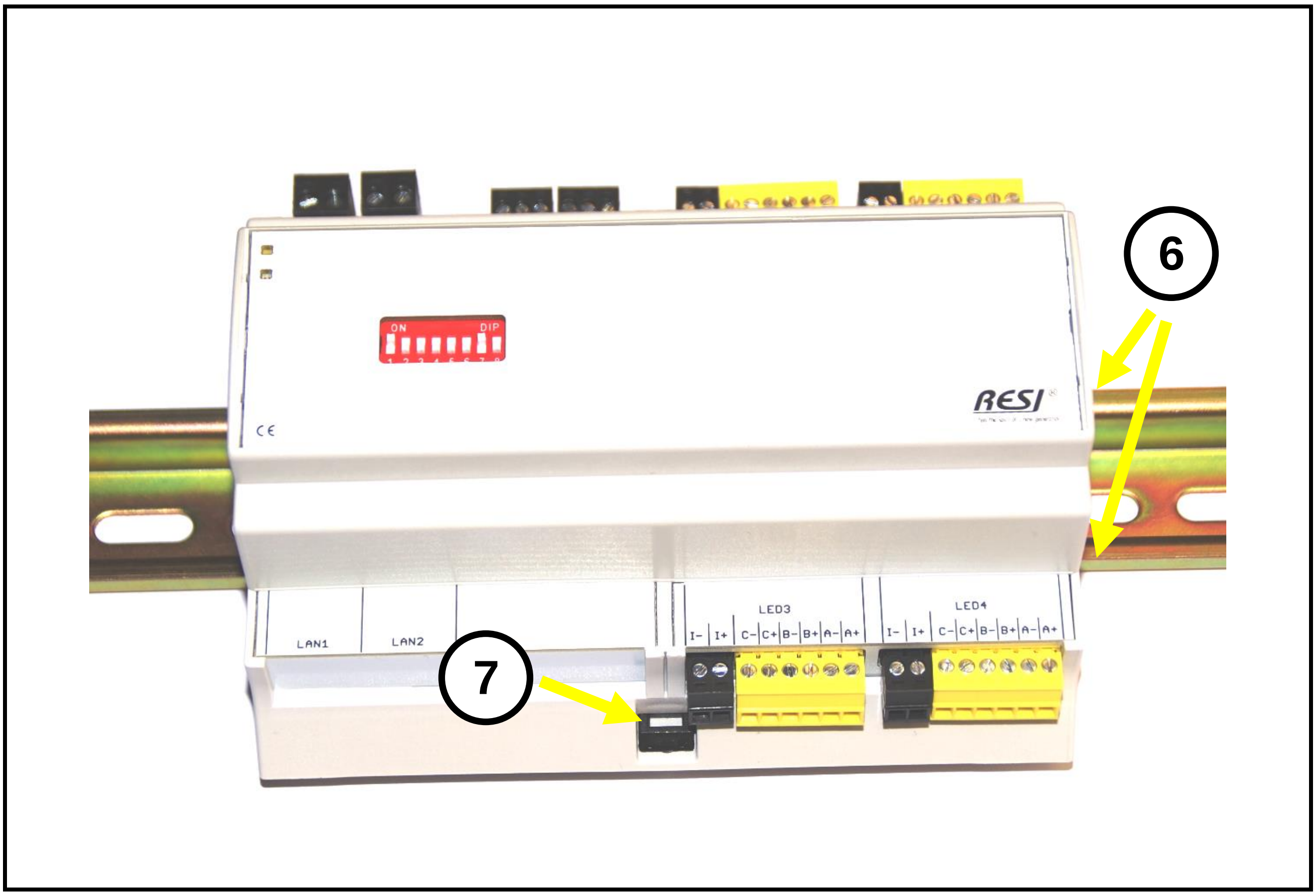
Then open the black hook with a screw driver (2). Now press the module with the opened hook onto the DIN rail until both sides of the module snap into the DIN rail (3). Release the screw driver now. The hook snaps into the DIN rail and the module is now mounted correctly onto the DIN rail.



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.

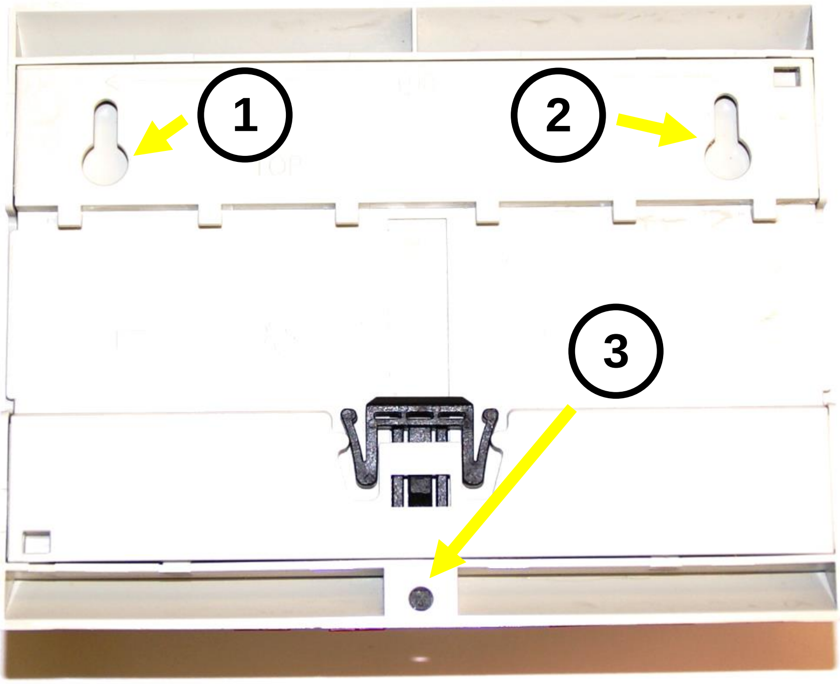


The module is correctly mounted, if the module has snapped into the DIN rail on both sides of the housing (6) and if the hook has snapped in too (7).

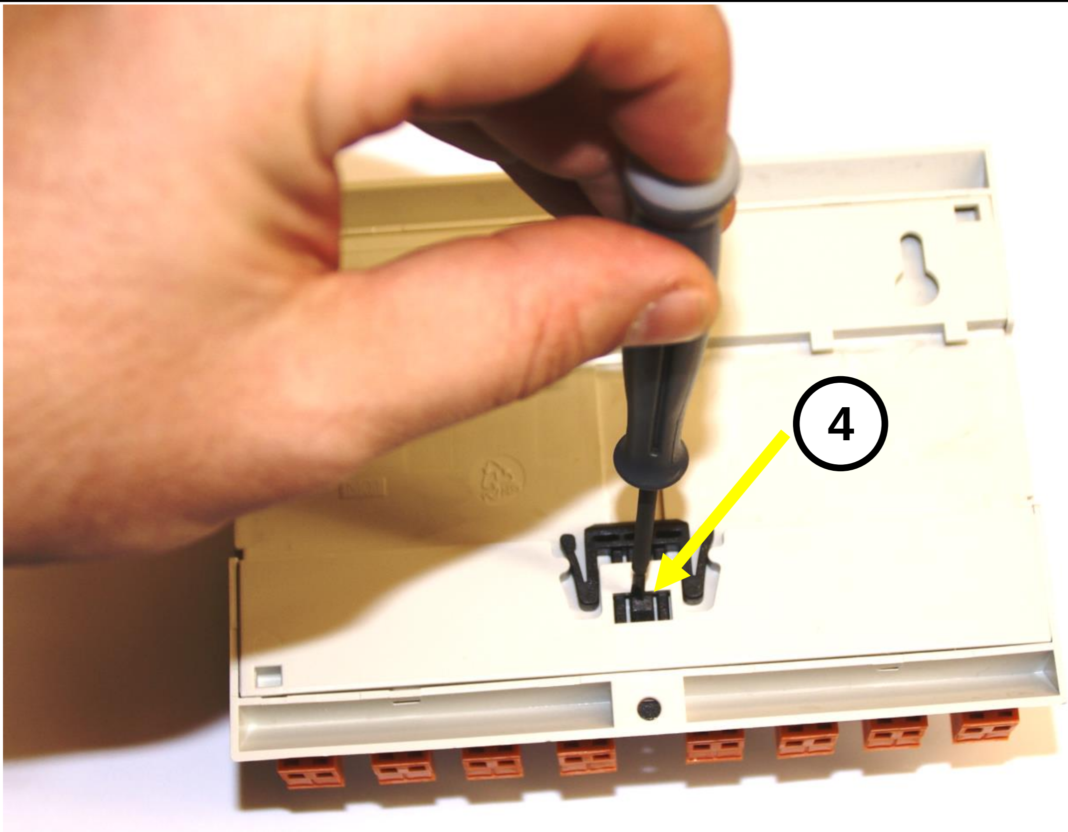


13.3.2 Wall mounting

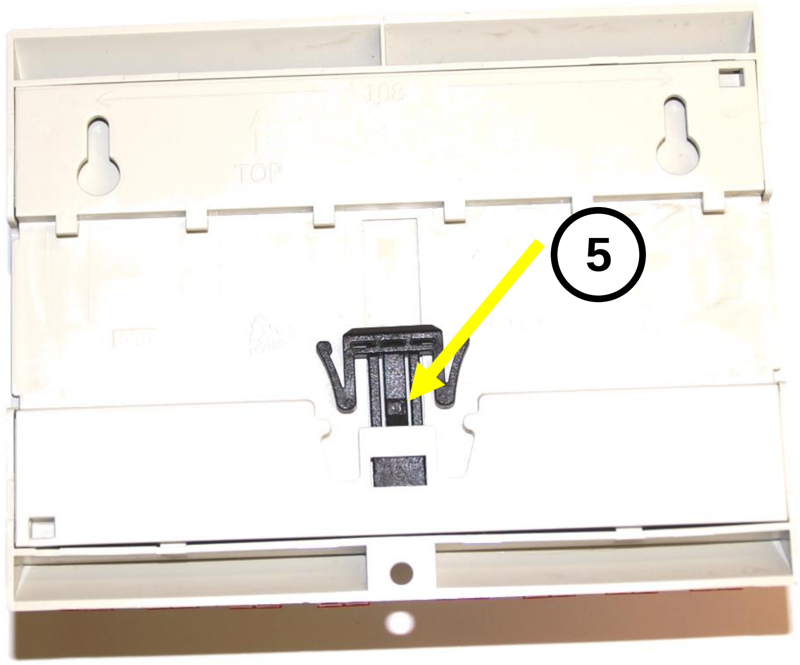
Our modules can also be mounted onto a wall. Turn over the module as shown in the picture below:

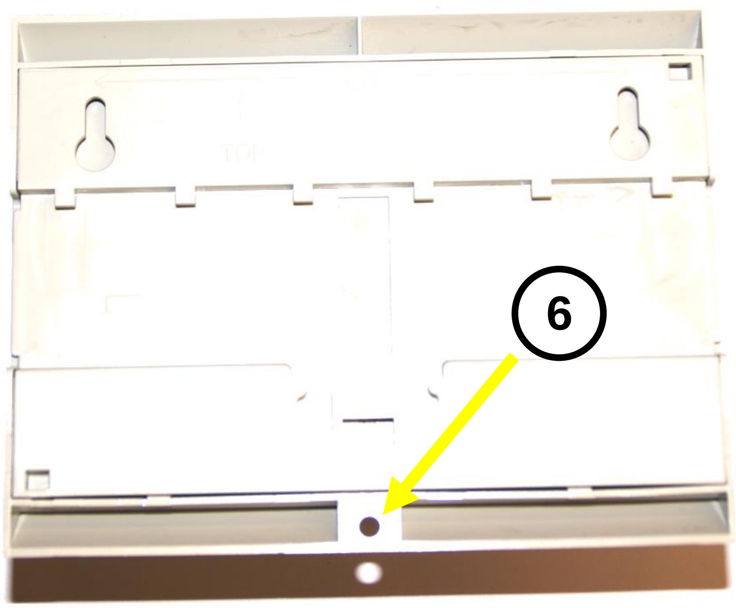


You will notice, that there are two holes for wall hooks or screws on the top side of the housing. (1) and (2). On the bottom side you will notice a small hole for a screw to fix the housing on the wall from the front (3). But first we have to remove the hook, which blocks the screw hole in the housing.



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.

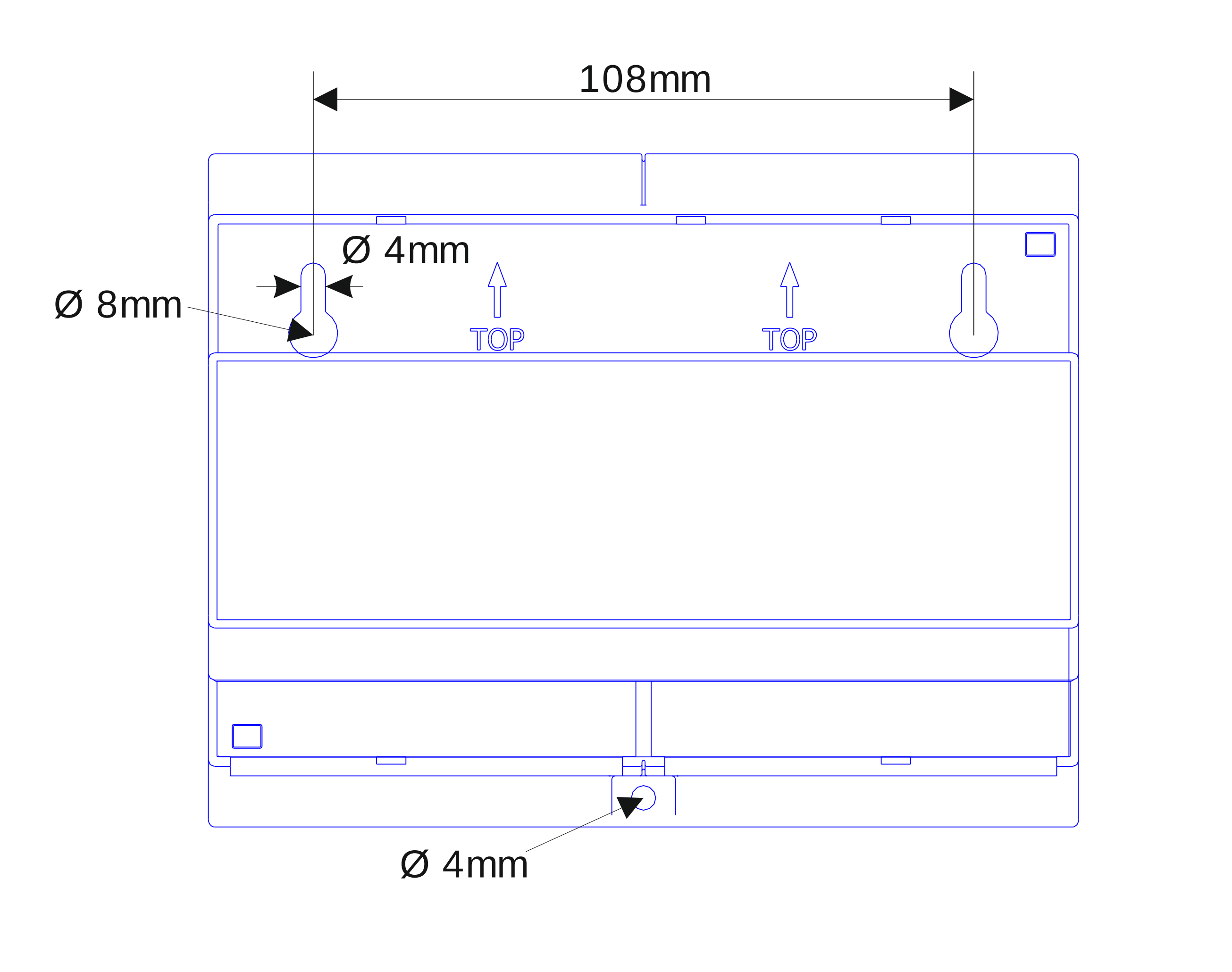


Illustration: Bottom view of the module with holes for wall mounting

13.4 Connection diagram

In the below drawing you will find the correct cabling of the IO module.

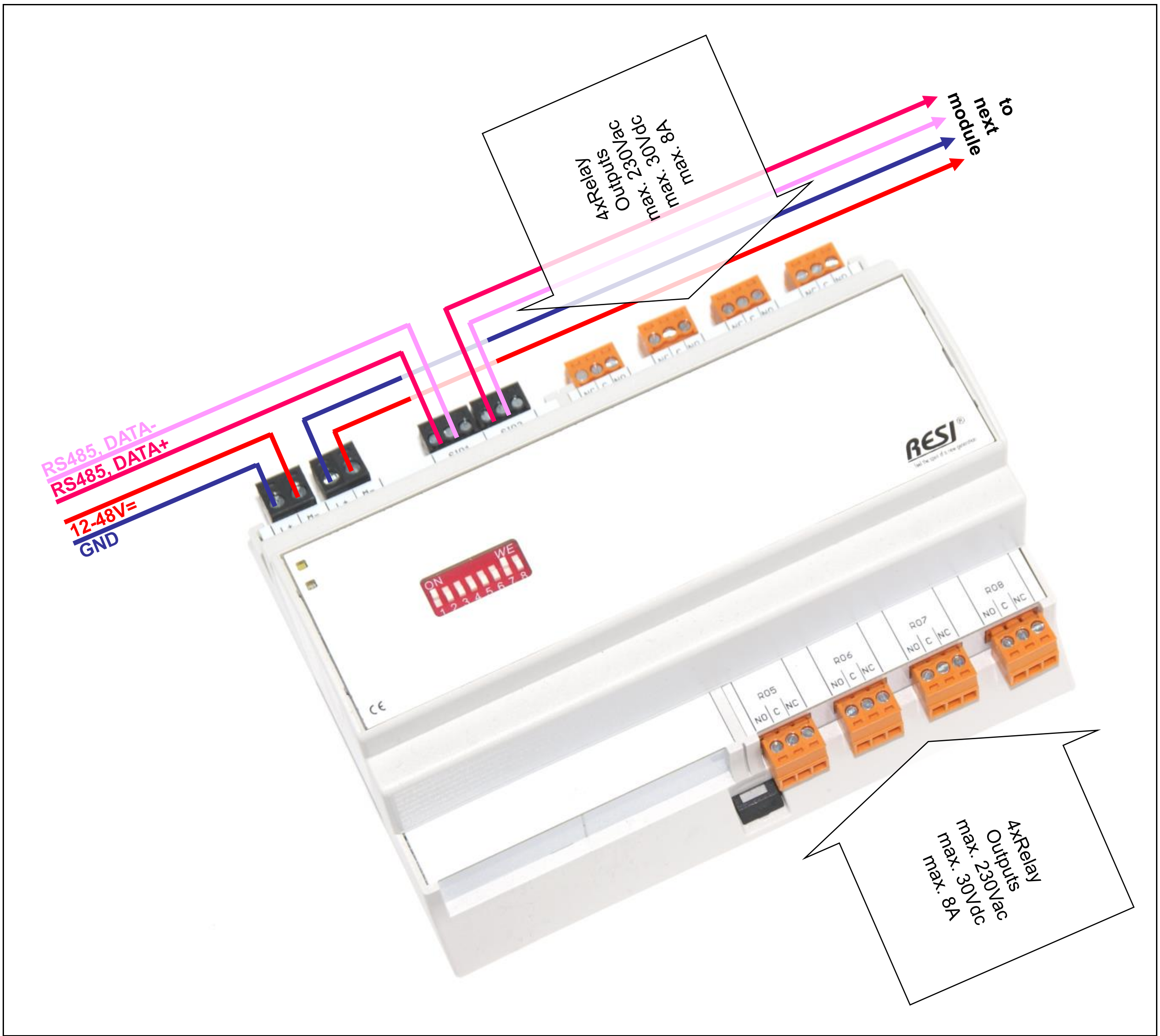


Illustration: cabling of the IO module

13.5 Clamps, DIP switch settings an LED indicators

The IO module offers the following clamps:

CLAMPS	RESI-8CO-MODBUS, RESI-8CO-ASCII
L+ M-	Power supply via two separated plug-in 2-pin terminal blocks. For daisy chain IN and OUT power supply of many modules L+: 12-48 V= M-: Ground
SIO1 1=A+ 2=B- 3=GND	RS485 ASCII or MODBUS/RTU serial interface IN A+: RS485 DATA+ signal B-: RS485 DATA- signal GND: RS485 ground signal
SIO2 1=A+ 2=B- 3=GND	RS485 ASCII or MODBUS/RTU serial interface OUT A+: RS485 DATA+ signal B-: RS485 DATA- signal GND: RS485 ground signal
RO1 NC C NO	Relay output 1 with removable 3pin terminal block NC: Normally closed switching contact of the relay, =OFF: closed, =ON: opened C: common root contact of the relay NO: Normally open switching contact of the relay, =OFF: opened, =ON: closed
RO2 NC C NO	Relay output 2 with removable 3pin terminal block NC: Normally closed switching contact of the relay, =OFF: closed, =ON: opened C: common root contact of the relay NO: Normally open switching contact of the relay, =OFF: opened, =ON: closed
RO3 NC C NO	Relay output 3 with removable 3pin terminal block NC: Normally closed switching contact of the relay, =OFF: closed, =ON: opened C: common root contact of the relay NO: Normally open switching contact of the relay, =OFF: opened, =ON: closed
RO4 NC C NO	Relay output 4 with removable 3pin terminal block NC: Normally closed switching contact of the relay, =OFF: closed, =ON: opened C: common root contact of the relay NO: Normally open switching contact of the relay, =OFF: opened, =ON: closed
RO5 NC C NO	Relay output 5 with removable 3pin terminal block NC: Normally closed switching contact of the relay, =OFF: closed, =ON: opened C: common root contact of the relay NO: Normally open switching contact of the relay, =OFF: opened, =ON: closed
RO6 NC C NO	Relay output 6 with removable 3pin terminal block NC: Normally closed switching contact of the relay, =OFF: closed, =ON: opened C: common root contact of the relay NO: Normally open switching contact of the relay, =OFF: opened, =ON: closed
RO7 NC C NO	Relay output 7 with removable 3pin terminal block NC: Normally closed switching contact of the relay, =OFF: closed, =ON: opened C: common root contact of the relay NO: Normally open switching contact of the relay, =OFF: opened, =ON: closed
RO8 NC C NO	Relay output 8 with removable 3pin terminal block NC: Normally closed switching contact of the relay, =OFF: closed, =ON: opened C: common root contact of the relay NO: Normally open switching contact of the relay, =OFF: opened, =ON: closed

Table: Description of the terminal blocks of the IO module

The IO module offers also an 8-pin DIP switch and a dual color LED indicator:

<table>
<tr><th>DIP+LED</th><th>RESI-8CO-MODBUS, RESI-8CO-ASCII</th></tr>
<tr><td>DIP SWITCH
1=ADR0
2=ADR1
3=ADR2
4=ADR3
5=BR0
6=BR1
7=BR2
8=PARITY</td><td>DIP switch to setup the IO module
ADR: 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:
<table>
<tr><td>ADR3</td><td>ADR2</td><td>ADR1</td><td>ADR0</td><td>MODBUS/RTU unit number or ASCII bus number</td></tr>
<tr><td>OFF</td><td>OFF</td><td>OFF</td><td>OFF</td><td>Internal MODBUS/RTU unit number is used from the FLASH memory in the range of 0 to 255.</td></tr>
<tr><td>OFF</td><td>OFF</td><td>OFF</td><td>ON</td><td>1</td></tr>
<tr><td>OFF</td><td>OFF</td><td>ON</td><td>OFF</td><td>2</td></tr>
<tr><td>OFF</td><td>OFF</td><td>ON</td><td>ON</td><td>3</td></tr>
<tr><td>OFF</td><td>ON</td><td>OFF</td><td>OFF</td><td>4</td></tr>
<tr><td>OFF</td><td>ON</td><td>OFF</td><td>ON</td><td>5</td></tr>
<tr><td>OFF</td><td>ON</td><td>ON</td><td>OFF</td><td>6</td></tr>
<tr><td>OFF</td><td>ON</td><td>ON</td><td>ON</td><td>7</td></tr>
<tr><td>ON</td><td>OFF</td><td>OFF</td><td>OFF</td><td>8</td></tr>
<tr><td>ON</td><td>OFF</td><td>OFF</td><td>ON</td><td>9</td></tr>
<tr><td>ON</td><td>OFF</td><td>ON</td><td>OFF</td><td>10</td></tr>
<tr><td>ON</td><td>OFF</td><td>ON</td><td>ON</td><td>11</td></tr>
<tr><td>ON</td><td>ON</td><td>OFF</td><td>OFF</td><td>12</td></tr>
<tr><td>ON</td><td>ON</td><td>OFF</td><td>ON</td><td>13</td></tr>
<tr><td>ON</td><td>ON</td><td>ON</td><td>OFF</td><td>14</td></tr>
<tr><td>ON</td><td>ON</td><td>ON</td><td>ON</td><td>15</td></tr>
</table>
BAUD RATE: Those three DIP switches BR2-BR0 define the MODBUS/RTU or ASCII baud rate for the communication:
<table>
<tr><td>BR2</td><td>BR1</td><td>BR0</td><td>MODBUS/RTU or ASCII Baudrate</td></tr>
<tr><td>OFF</td><td>OFF</td><td>OFF</td><td>4800bd</td></tr>
<tr><td>OFF</td><td>OFF</td><td>ON</td><td>9600bd</td></tr>
<tr><td>OFF</td><td>ON</td><td>OFF</td><td>19200bd</td></tr>
<tr><td>OFF</td><td>ON</td><td>ON</td><td>38400bd</td></tr>
<tr><td>ON</td><td>OFF</td><td>OFF</td><td>57600bd</td></tr>
<tr><td>ON</td><td>OFF</td><td>ON</td><td>115200bd</td></tr>
<tr><td>ON</td><td>ON</td><td>OFF</td><td>230400bd</td></tr>
<tr><td>ON</td><td>ON</td><td>ON</td><td>256000bd</td></tr>
</table>
PARITY: This DIP switch PARITY defines the MODBUS/RTU or ASCII parity for the communication:
<table>
<tr><td>PARITY</td><td>MODBUS/RTU or ASCII parity</td></tr>
<tr><td>OFF</td><td>NONE</td></tr>
<tr><td>ON</td><td>EVEN</td></tr>
</table>
HINT: After changing on of the DIP switches, the module restarts completely and intialises the serial interface. You will notice that the WHITE LED will be on for approximately 2 seconds, before this LED will flash with a one second cycle.</td></tr>
<tr><td>LED WHITE</td><td>This LED will flash with a cycle of 1 seconds to show normal mode of the module</td></tr>
<tr><td>LED GREEN</td><td>This LED will flash shortly, whenever the module receives a valid telegram on the serial interface.</td></tr>
<tr><td>LED RED</td><td>If this LED flashes cyclically, there is a module error detected by the firmware</td></tr>
</table>

Table: Description of the DIP switch functions and the indication LEDs on the IO module

13.7 Dimensions of the module

In the below drawing you will find the dimensions of the IO module.

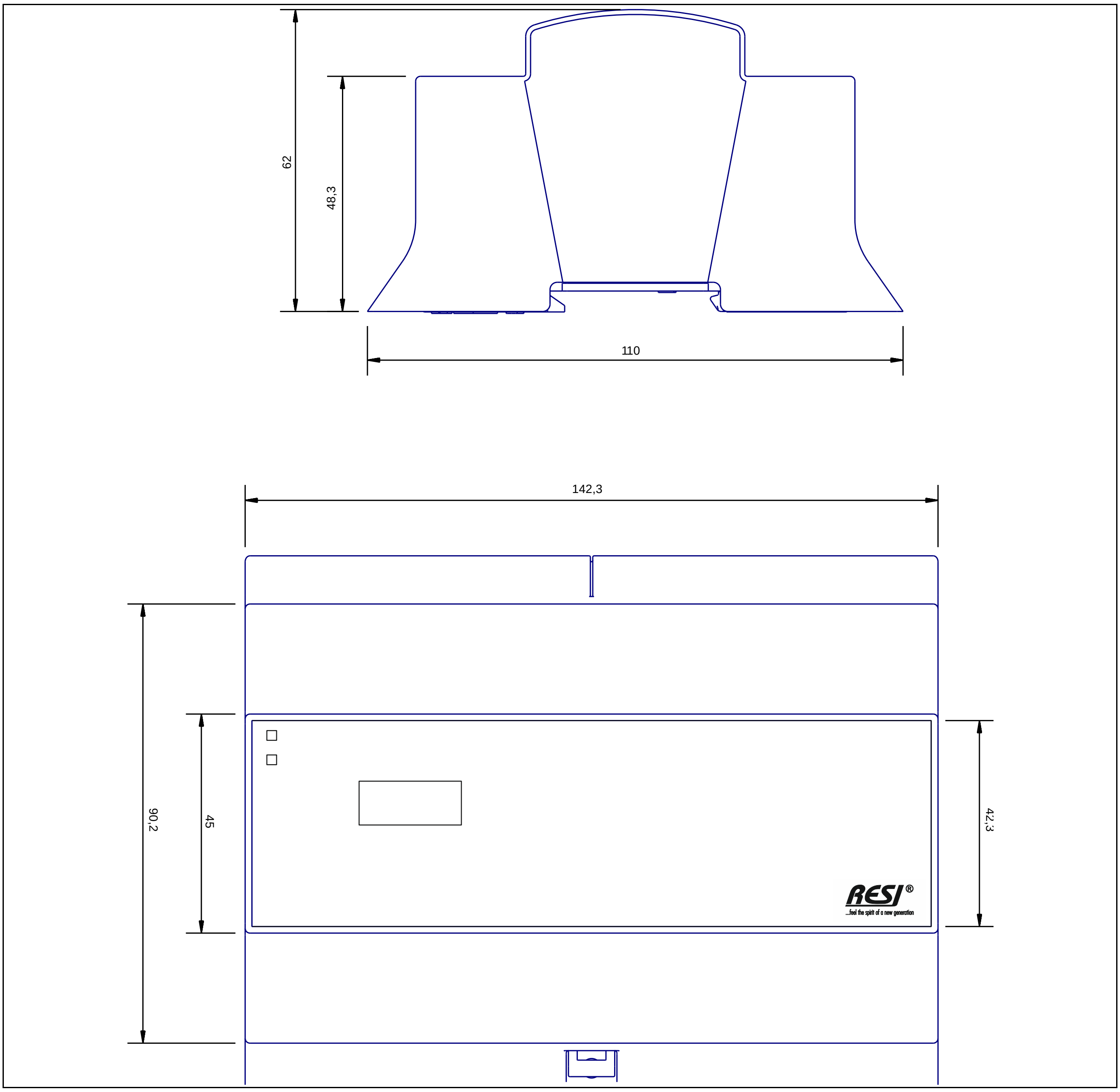


Illustration: Dimensions of the IO module in mm

Dimensions	
Dimensions of the housing L x B x H (mm)	143 x 110 x 62
Weight	300 g
Color	Grey, RAL7035
Material	Self-extinguish PC/ABS, DIN 43880
Protection class	IP20 based on DIN 40050/EN 60529

Table: technical data of the housing

13.8 Power supply of the module

In the below drawing you will find how to connect the module to a power supply.

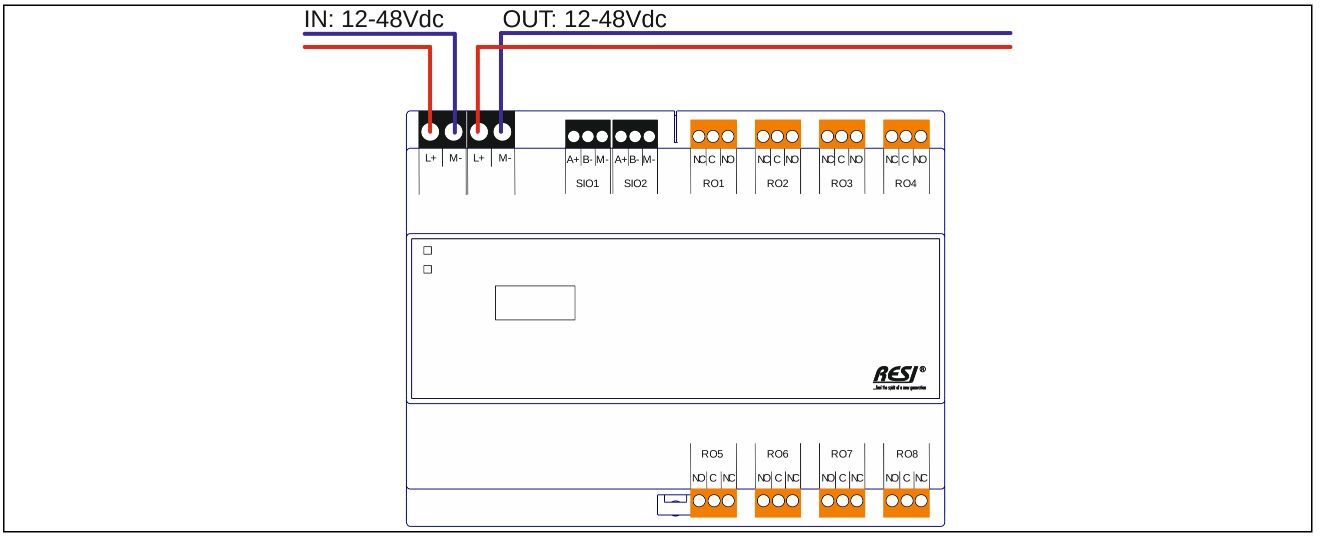


Illustration: Power supply of the IO module

The module offers two 2-pin plug-in terminals for connecting the power supply to the module. It is designed to create a daisy chain power supply with many modules.

13.9 Serial RS485 connection

In this drawing you see the cabling of the serial RS485 bus line. In the module both SIO terminal block are bridged.

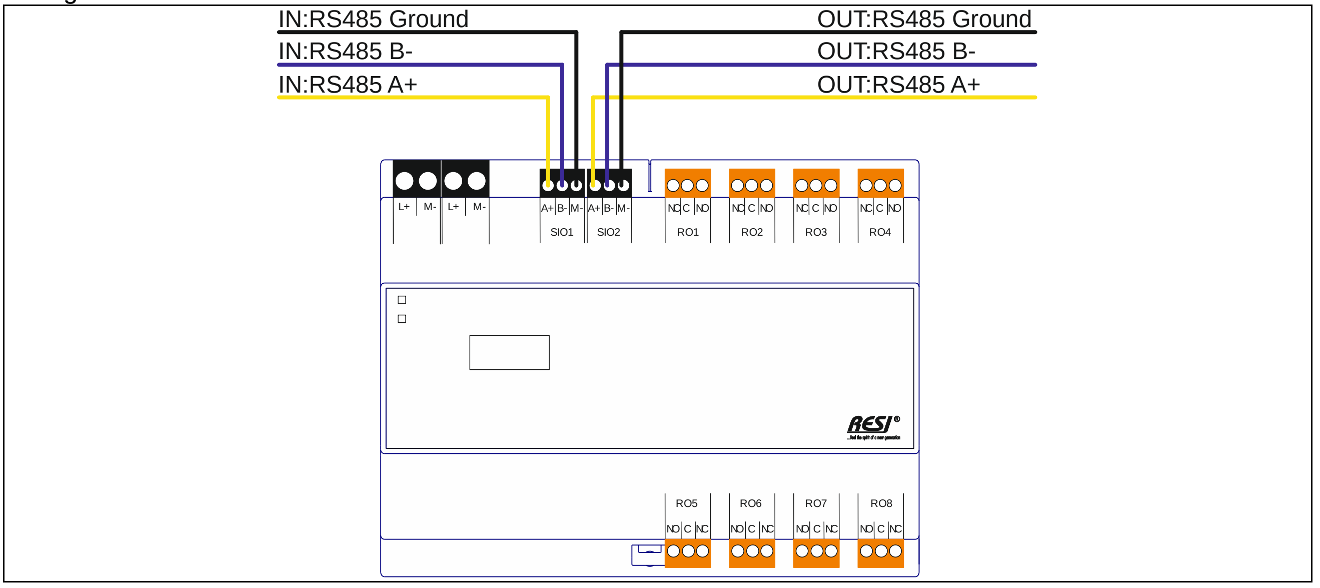
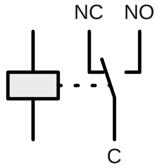


Illustration: RS485 bus cabling of the IO module

The module offers two plug-in 3-pin terminals to connect a RS485 bus line to the module. It was designed to create a daisy chain bus line with many modules. Don’t forget, that a RS485 bus line needs a line termination at the end of both lines!

13.10 Cabling of the relay outputs of the module

In the below drawing you see the cabling of the 8 relay outputs of the module. Each relay offers three contacts: One common root contact, one closing contact (NO) and one opening contact (NC).



If the relay is OFF (powerless), the NC contact is tied to the common root contact (C) and the NO contact is open.

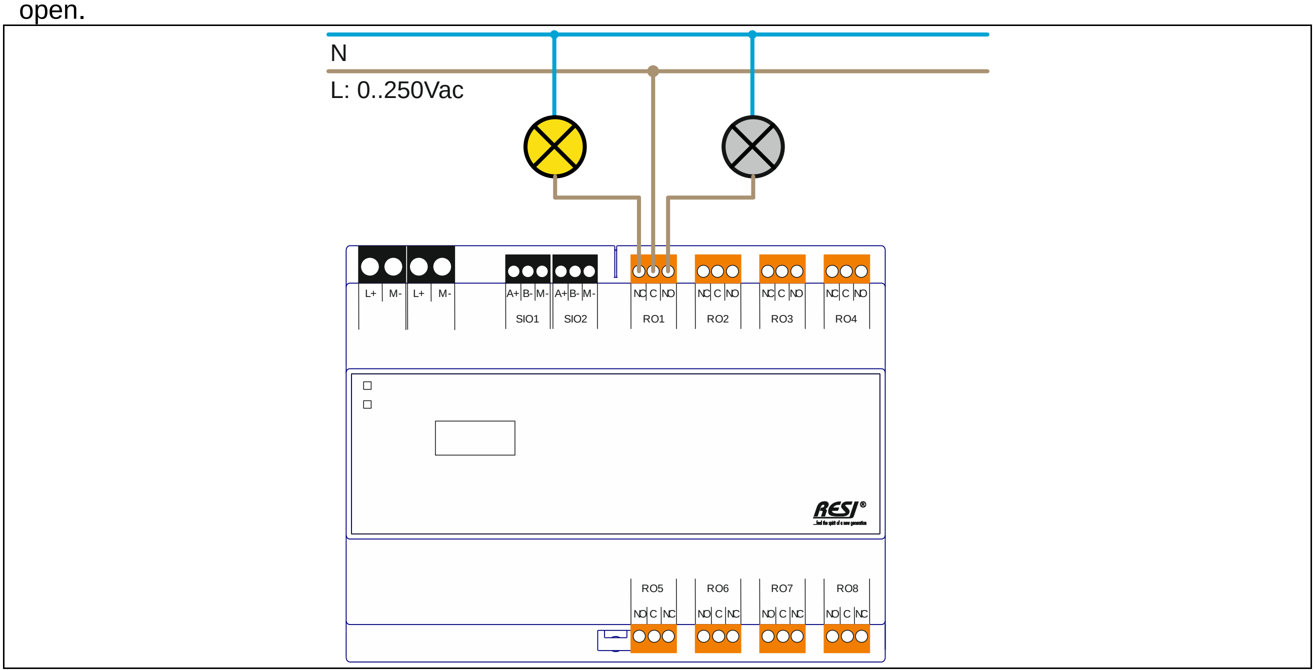


Illustration: Cabling of the relay output 1, relay is OFF

If the relay is under power (ON), then the NC contact is open, and the NO contact ist tied to the common root contact (C).

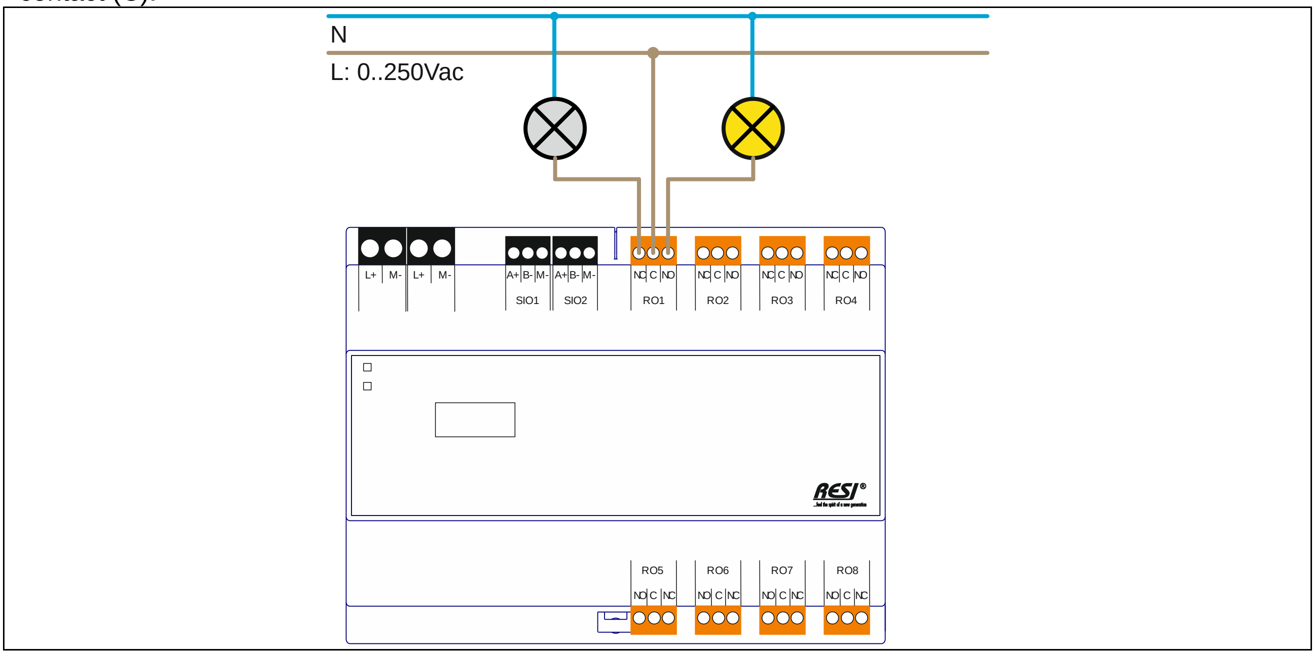


Illustration: Cabling of the relay output 1, relay is ON

The following illustration shows the cabling of all 8 relays using only the NO contact. Only if the relay is ON, the current flows from the root contact to the switching contact NO to the consumer.

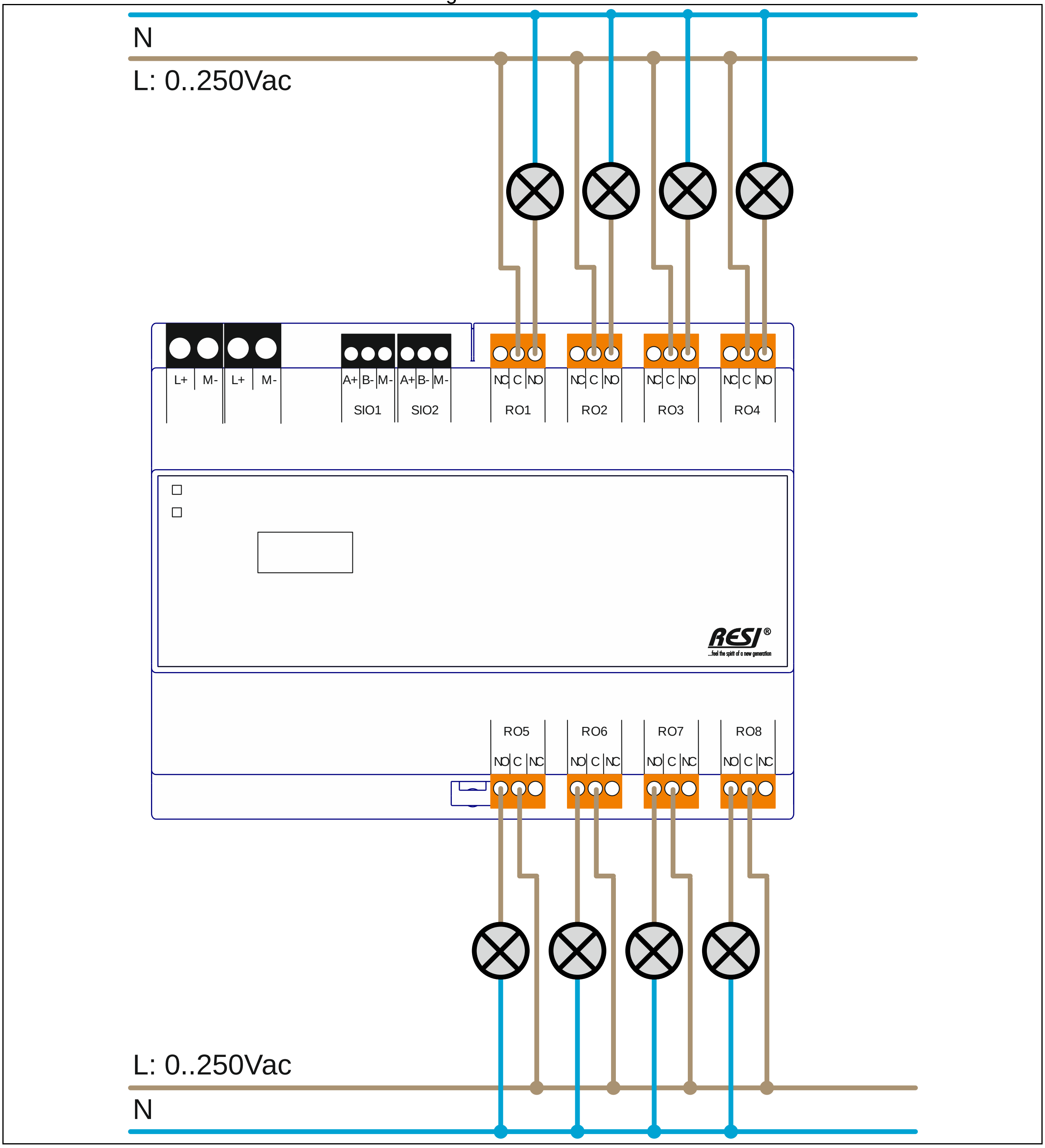


Illustration: Cabling of all 8 relay outputs using the NO contact, all 8 relays are OFF

Here we show a DC cabling of all 8 relay with the NO contacts. Of course you can mix AC and DC signals on the relay outputs of the modules.

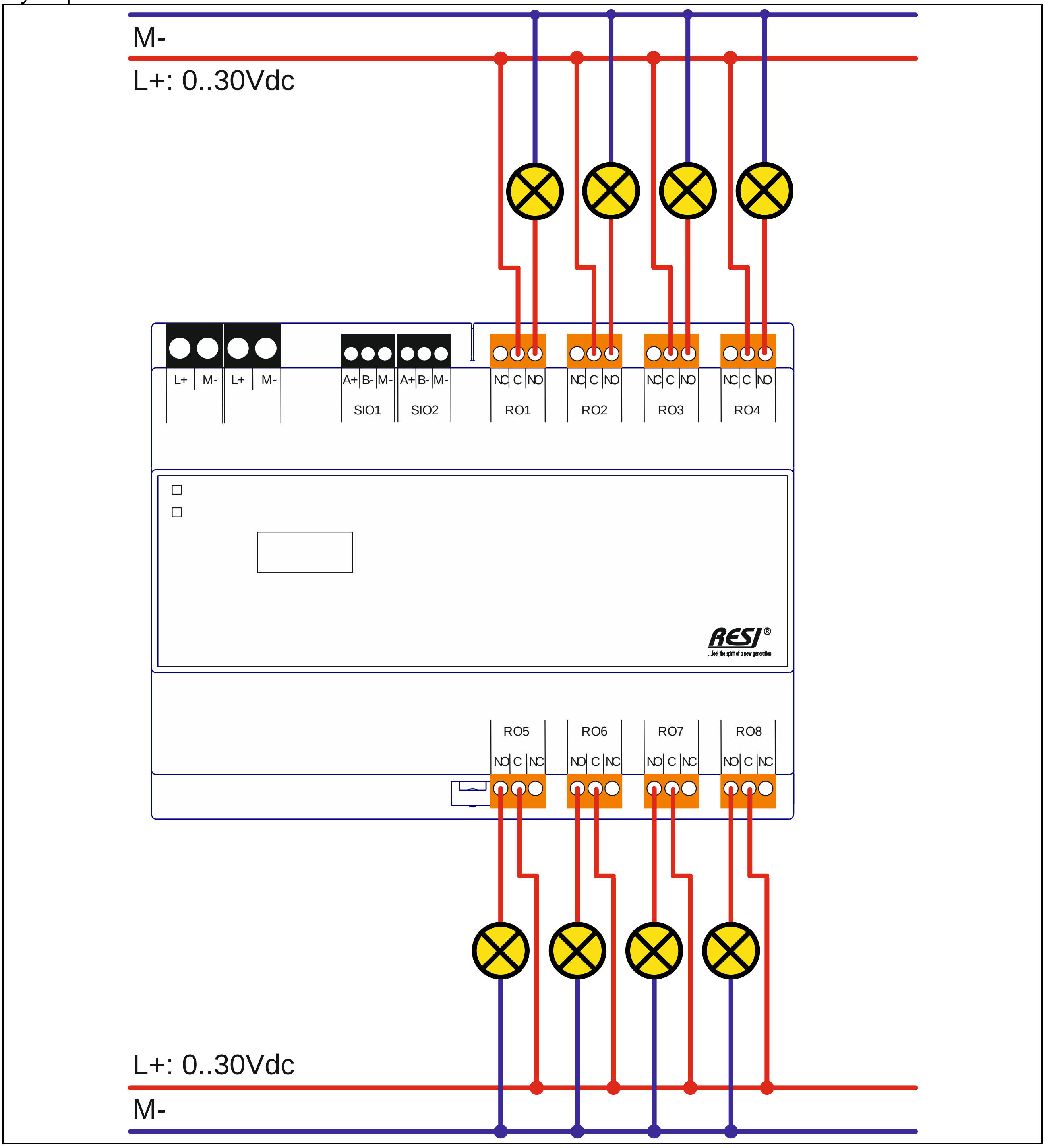


Illustration: Cabling of all 8 relay outputs using the NO contact, all 8 relays are ON

13.11 Functional description

This IO module communicates with a host system with the MODBUS/RTU slave protocol. All versions of the Io module with RESI-xxx-ASCII offer an additional protocol: An ASCII slave protocol with simple text string. The communication runs over a RS485 interface (half duplex) or over a RS232 interface (full duplex).

For the communication with ASCII texts, the host sends ASCII messages with a special start character # (0x23, 35dec) and a special end character (0x0d,13dec or CARRIAGE RETURN) to the module. The module uses also this special start and end characters to answer to the host request. Consult the below noted detailed command descriptions. In the ASCII protocol mode, the host can send messages with or without a bus number.

For communication with the MODBUS/RTU slave, the module offers the following MODBUS functions:

- 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)

HINT:
The functions READ HOLDING REGISTER and PRESET MULTIPLE REGISTERS are restricted to max. 125 register per request! The functions READ INPUT STATUS, READ COIL STATUS and FORCE MULTIPLE COILS are restricted to 2000 coils or inputs (bits) per request.

13.12 ASCII protocol description

13.12.1 Overview

The IO module communicates with simple ASCII commands. The following special characters are used in this documentation:

stand for the **Hashtag** ASCII character 35dec or 0x23
: stand for the **colon** ASCII character 58dec or 0x3A
= stand for the **equal sign** with the ASCII code 61ec or 0x3D
- stand for the **minus sign** with the ASCII code 45dec or 0x2D
, stands for the **comma** with the ASCII code 44dec or 0x2C
<CR> stand for **CARRIAGE RETURN** ASCII character 13dec or 0x0D. In the following text we use the representation CR.
<SP> stands for the **SPACE**. This is the space character with the ASCII code 32dec or 0x20. In the following text we will use the representation☐.
<ADR> will be used as the current **bus address** of the module. The bus number can be transmitted decimal or hexadecimal and is separated with a comma (ASCII code 44dec or 0x2C) from the rest of the command. Hexadecimal numbers always start with 0x. Its only allowed to use the ASCII characters ‚0‘-‚9‘ 48dec to 57dec, 0x30-0x39 and ‚A‘ to ‚F‘, 65dec to 70dec, 0x41-0x46. All modules react to the broadcast bus address 0 and to its own bus number. With a DIP switch, the user can easily change between the internal stored bus number in the FLASH and the fix bus number 255. Consult the DIP switch description for more details.

13.12.2 Communication sequence

In general the IO module sends no ASCII characters without a request from a host. So the host is the master of the communication and the IO module is always answering to host requests as a slave. If only one IO module is connected to a host (e.g. Via RS232), you can dump the bus number in the protocol. If you use a RS485 interface, more than one module can be connected to the host. Therefore a bus number in the request frame of the host is always necessary.

The structure of the command look like this:

The host sends a command or a command with parameters without a bus address to the module:
#<Command><CR> or
#<Command>:<Parameter><CR>

The module answers, if it feels addressed, with the following answer telegram:
#<Answer><CR>

If the bus number is used, the module answers with:
#<ADR>,<Command><CR> or
#<ADR>,<Command>:<Parameter><CR>

The module answers with:
#<ADR>,<Answer><CR>

The bus address lies in the range of 1dec to 255dec or 0x00 to 0xFF hexadecimal. The setup is done with our free configuration software MODBUSConfigurator. Each module reacts also to the broadcast bus address 0.

For each command, we define two different writings. A long version and a short version, to avoid unnecessary traffic on the bus. For example to request the software version of the module you can use the command VERSION or the short command VER.

13.12.3 Request VERSION

This command returns the current software version of the module.

Host long version:
#VERSION<CR> or
#<ADR>,VERSION<CR>

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

Answer:
#VERSION:<HIGH>.<MED>.<LOW><CR> or
#<ADR>,VERSION:<HIGH>,<MED>,<LOW><CR>

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

Samples:
→ **#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$_{CR}$

13.12.4 Request module TYPE

This command returns the current type of the module.

Host long version:
#TYPE<CR> or
#<ADR>,TYPE<CR>

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

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

<TYP> defines the current type of the module. Currently RESI-8CO-ASCII

Samples:
→ #TYPE$_{CR}$
← #TYPE:RESI-8CO-ASCII$_{CR}$

→ #255,TYP$_{CR}$
← #255,TYPE:RESI-8CO-ASCII$_{CR}$

13.12.5 Table of all ASCII commands

Here you will find a possible ASCII commands of the module. We use here only the version with bus number. That you can avoid the bus number, we have discussed earlier in this document. If an argument as the extension Dec, it will be returned as a decimal number, If an argument has the extension Hex, then this argument is returned as a hexadecimal number. Many command returns the argument in decimal and hexadecimal representation. So the host can select, what kind of number conversion, it will handle in its software.

Direction	ASCII command
Host	**#<BusAdr>,VER$_{CR}$** **#<BusAdr>,VERSION$_{CR}$**
Answer	**#<BusAdr>,VERSION:<VersionHi>.< VersionMed>.<VersionLo>$_{CR}$**
	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)
Host	**#<BusAdr>,TYP$_{CR}$** **#<BusAdr>,TYPE$_{CR}$**
Answer	**#<BusAdr>,TYPE:RESI-8CO-ASCII$_{CR}$**
	Returns the current type of the module
Host	**#<BusAdr>,OWN$_{CR}$** **#<BusAdr>,OWNER$_{CR}$**
Answer	**#<BusAdr>,OWNER:RESI$_{CR}$**
	Returns the owner of the module
Host	**#<BusAdr>,CRE$_{CR}$** **#<BusAdr>,CREATOR$_{CR}$**
Answer	**#<BusAdr>,CREATOR:DI HC SIGL,MSC$_{CR}$**
	Returns the creator of the module
Host	**#<BusAdr>,COPY$_{CR}$** **#<BusAdr>,COPYRIGHT$_{CR}$**
Answer	**#<BusAdr>,COPYRIGHT:2015-16 BY RESI AND DI HC SIGL,MSC WWW.RESI.CC$_{CR}$**
	Returns a copyright note about the module
Host	**#<BusAdr>,GDIP$_{CR}$** **#<BusAdr>,GET☐DIP$_{CR}$**
Answer	**#<BusAdr>,GDIP:<DIPSwitchDec>,<DIPSwitchHex>$_{CR}$**
	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 (=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)

Direction	ASCII command
Host	**#<BusAdr>,GROS$_{CR}$** **#<BusAdr>,GET□ROS$_{CR}$**
Answer	**#<BusAdr>,GROS:<ROSDec>,<ROSHex>$_{CR}$**
	Returns the current state of the eight relay outputs as decimal number and as hexadecimal number. ROSDec ROSHex — The current state of the eight relay outputs: Bit 0: State of RO1 (=0:OFF, =1:ON) Bit 1: State of RO2 (=0:OFF, =1:ON) Bit 2: State of RO3 (=0:OFF, =1:ON) Bit 3: State of RO4 (=0:OFF, =1:ON) Bit 4: State of RO5 (=0:OFF, =1:ON) Bit 5: State of RO6 (=0:OFF, =1:ON) Bit 6: State of RO7 (=0:OFF, =1:ON) Bit 7: State of RO8 (=0:OFF, =1:ON)
Host	**#<BusAdr>,SROS:<OutAllROS>$_{CR}$** **#<BusAdr>,SET□ROS:<OutAllROS>$_{CR}$**
Answer	**#OK$_{CR}$**
	Sets all eight bistable relay to the new state <OutAllROS> OutAllROS — The new state for all bistable relay outputs: Bit 0: State for RO1 (=0:RO to OFF, =1:RO to ON) Bit 1: State for RO2 (=0:RO to OFF, =1:RO to ON) Bit 2: State for RO3 (=0:RO to OFF, =1:RO to ON) Bit 3: State for RO4 (=0:RO to OFF, =1:RO to ON) Bit 4: State for RO5 (=0:RO to OFF, =1:RO to ON) Bit 5: State for RO6 (=0:RO to OFF, =1:RO to ON) Bit 6: State for RO7 (=0:RO to OFF, =1:RO to ON) Bit 7: State for RO8 (=0:RO to OFF, =1:RO to ON)
Host	**#<BusAdr>,GROx$_{CR}$** **#<BusAdr>,GET□ROx$_{CR}$**
Answer	**#<BusAdr>,GROx:<ROxDec>,<ROxHex>$_{CR}$**
x	**1..8**
	Returns the current state of the relay output ROx as decimal number and as hexadecimal number. X stands for the number of the relay output from 1 to 8. ROxDec ROxHex — The current state of the bistable relay output ROx: =0: relay output is OFF =1: relay output is ON
Host	**#<BusAdr>,SROx:<Out>$_{CR}$** **#<BusAdr>,SET□ROx:<Out>$_{CR}$**
Answer	**#OK$_{CR}$**
x	**1..8**
	Sets the new state for relay output ROx. The state is defined with <Out>. X stands for the affected relay output between 1 and 8. Out — The new state of the relay output ROx: =0: relay output is OFF =1: relay output is ON

Direction	ASCII command
Host	**#<BusAdr>,PROx:<PulseTimeIn100ms>$_{CR}$** **#<BusAdr>,PULSE□ROx:<PulseTimeIn100ms>$_{CR}$**
Answer	**#<BusAdr>,OK$_{CR}$**
x	**1..8**
	This command switches to relay output ROx on for the pulse duration <PulseTimeIn100ms>*100ms. PulseTimeIn100ms — A duration in 100ms units the corresponding relay output is switched on for this time period
Host	**#<BusAdr>,GPTx$_{CR}$** **#<BusAdr>,GET□PPULSE□TIMERx$_{CR}$**
Answer	**#<BusAdr>,GPTx:<PulseTimeInMSDec>,<PulseTimeInMSHex>$_{CR}$**
x	**1..8**
	Returns the current timer value of the pulse timer for relay output ROx in ms. PulseTimeInMSDec PulseTimeInMSHex — The remaining time of the pulse timer in Milliseconds

Direction	ASCII command
Host	**#<BusAdr>,SMBADR:<MBUnit>$_{CR}$** **#<BusAdr>,SET□MODBUS□ADDRESS:<MBUnit>$_{CR}$**
Answer	**#<BusAdr>,OK $_{CR}$**
	Writes the unit address into the FLASH memory of the module. The new unit address for MODBUS/RTU or ASCII mode is only used immediately, if the DIP switch setting of the bus address is 0. Otherwise the unit address is defined by the DIP settings. The unit address ranges from 0dec to 255dec.
Host	**#<BusAdr>,GMBADR$_{CR}$** **#<BusAdr>,GET□MODBUS□ADDRESS$_{CR}$**
Answer	**#<BusAdr>,GMBADR:<MBUnitDec>,<MBFLASHDec>,<MBUnitHex>,<MBFLASHHex>$_{CR}$**
	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. MBUnitDec MBUnitHex — The current used MODBUS/RTU unit or ASCII address for communication MBFLASHDec MBFLASHHex — The internal stored MODBUS/RTU unit address or ASCII address from the FLASH memory, if the DIP switch is 0.
Host	**#<BusAdr>,RST$_{CR}$** **#<BusAdr>,RESET$_{CR}$**
Answer	**none**
	Executes a software reset (Reboot) of the module.

13.13 MODBUS – register description

13.13.1 Table of inputs and coils

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 0x00001 to 0x65536. Inputs are usually noted with 1x00001 to 1x65536. 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: 0x00100 for the coil 100, 1x00100 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.

Register	Description
0x00001 1x00001 I:0 R/W RO1	Current state of the relay output RO1 =0:RO is OFF, =1:RO is ON Writing on this register changes the state of the relay output
0x00002 1x00002 I:1 R/W RO2	Current state of the relay output RO2 =0:RO is OFF, =1:RO is ON Writing on this register changes the state of the relay output
0x00003 1x00003 I:2 R/W RO3	Current state of the relay output RO3 =0:RO is OFF, =1:RO is ON Writing on this register changes the state of the relay output
0x00004 1x00004 I:3 R/W RO4	Current state of the relay output RO4 =0:RO is OFF, =1:RO is ON Writing on this register changes the state of the relay output
0x00005 1x00005 I:4 R/W RO5	Current state of the relay output RO5 =0:RO is OFF, =1:RO is ON Writing on this register changes the state of the relay output
0x00006 1x00006 I:5 R/W RO6	Current state of the relay output RO6 =0:RO is OFF, =1:RO is ON Writing on this register changes the state of the relay output
0x00007 1x00007 I:6 R/W RO7	Current state of the relay output RO7 =0:RO is OFF, =1:RO is ON Writing on this register changes the state of the relay output
0x00008 1x00008 I:7 R/W RO8	Current state of the relay output RO8 =0:RO is OFF, =1:RO is ON Writing on this register changes the state of the relay output

Register	Description
0x00009 1x00009 I:8 R/O DIP1	Current state of DIP switch 1 =0:Dip switch is OFF, =1: Dip switch is ON
0x00010 1x00010 I:9 R/O DIP2	Current state of DIP switch 2 =0:Dip switch is OFF, =1: Dip switch is ON
0x00011 1x00011 I:10 R/O DIP3	Current state of DIP switch 3 =0:Dip switch is OFF, =1: Dip switch is ON
0x00012 1x00012 I:11 R/O DIP4	Current state of DIP switch 4 =0:Dip switch is OFF, =1: Dip switch is ON
0x00013 1x00013 I:12 R/O DIP5	Current state of DIP switch 5 =0:Dip switch is OFF, =1: Dip switch is ON
0x00014 1x00014 I:13 R/O DIP6	Current state of DIP switch 6 =0:Dip switch is OFF, =1: Dip switch is ON
0x00015 1x00015 I:14 R/O DIP7	Current state of DIP switch 7 =0:Dip switch is OFF, =1: Dip switch is ON
0x00016 1x00016 I:15 R/O DIP8	Current state of DIP switch 8 =0:Dip switch is OFF, =1: Dip switch is ON

13.13.2 Table of holding/input registers

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.

Register	Description
4x00001 3x00001 I:0 R/W ROS	Current state of all relay outputs Bit 0: =0:RO1 is OFF, =1:RO1 is ON Bit 1: =0:RO2 is OFF, =1:RO2 is ON Bit 2: =0:RO3 is OFF, =1:RO3 is ON Bit 3: =0:RO4 is OFF, =1:RO4 is ON Bit 4: =0:RO5 is OFF, =1:RO5 is ON Bit 5: =0:RO6 is OFF, =1:RO6 is ON Bit 6: =0:RO7 is OFF, =1:RO7 is ON Bit 7: =0:RO8 is OFF, =1:RO8 is ON Bit 8-15: always 0 Write on this register sets all eight relay to a new state
4x00101 3x00101 I:0 R/W ROS	Current state of all relay outputs Bit 0: =0:RO1 is OFF, =1:RO1 is ON Bit 1: =0:RO2 is OFF, =1:RO2 is ON Bit 2: =0:RO3 is OFF, =1:RO3 is ON Bit 3: =0:RO4 is OFF, =1:RO4 is ON Bit 4: =0:RO5 is OFF, =1:RO5 is ON Bit 5: =0:RO6 is OFF, =1:RO6 is ON Bit 6: =0:RO7 is OFF, =1:RO7 is ON Bit 7: =0:RO8 is OFF, =1:RO8 is ON Bit 8-15: always 0 Write on this register sets all eight relay to a new state
4x00103 3x00103 I:102 R/O DIP	Current state of the DIP switch 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) Bit 8-15: always 0

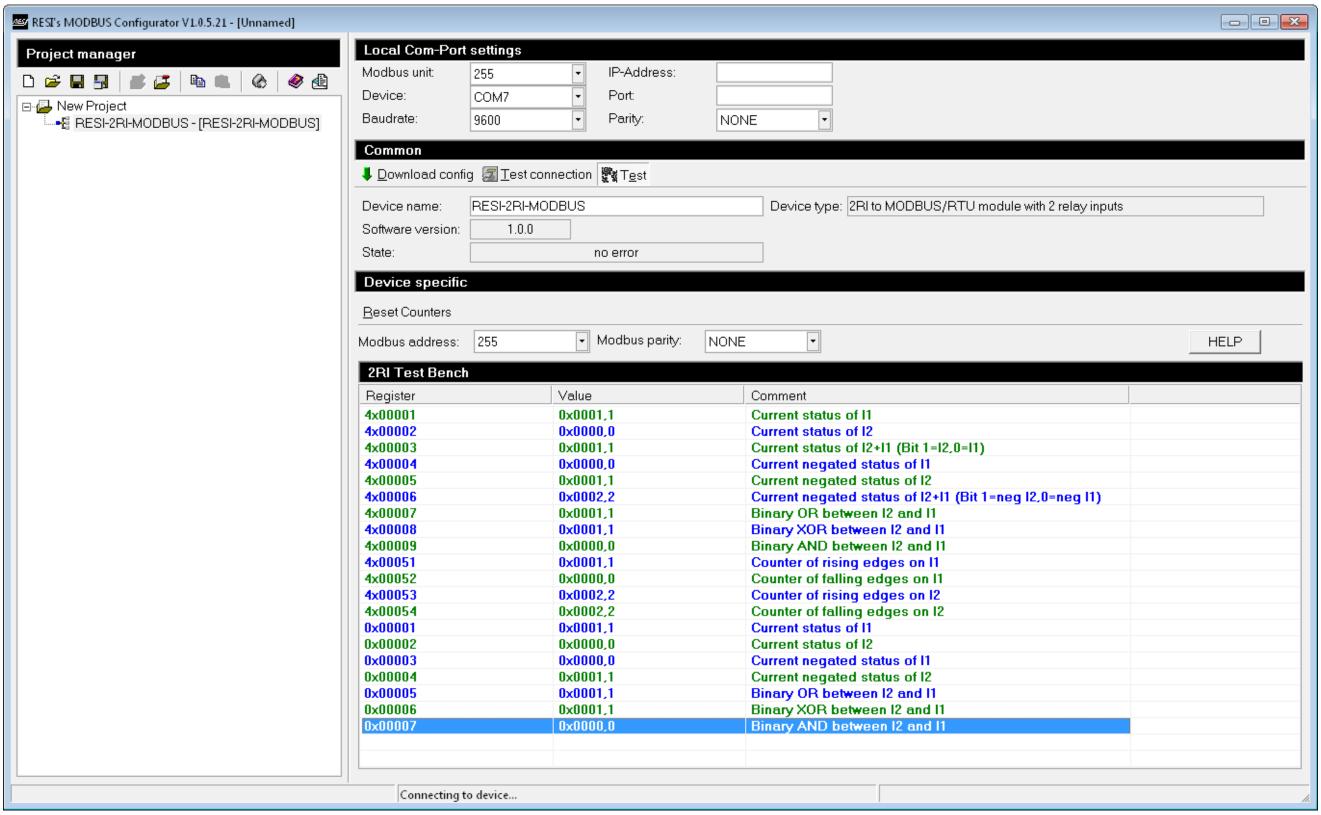
Register	Description
4x00201 3x00201 I:200 R/W PULSE RO1	generate a pulse on relay output 1 in 100ms units (0,1 to 6553,5 Seconds selectable) If you write onto this register, the relay output will be switched on for the desired time in 100ms units.
4x00202 3x00202 I:201 R/W PULSE RO2	generate a pulse on relay output 2 in 100ms units (0,1 to 6553,5 Seconds selectable) If you write onto this register, the relay output will be switched on for the desired time in 100ms units.
4x00203 3x00203 I:202 R/W PULSE RO3	generate a pulse on relay output 3 in 100ms units (0,1 to 6553,5 Seconds selectable) If you write onto this register, the relay output will be switched on for the desired time in 100ms units.
4x00204 3x00204 I:203 R/W PULSE RO4	generate a pulse on relay output 4 in 100ms units (0,1 to 6553,5 Seconds selectable) If you write onto this register, the relay output will be switched on for the desired time in 100ms units.
4x00205 3x00205 I:204 R/W PULSE RO5	generate a pulse on relay output 5 in 100ms units (0,1 to 6553,5 Seconds selectable) If you write onto this register, the relay output will be switched on for the desired time in 100ms units.
4x00206 3x00206 I:205 R/W PULSE RO6	generate a pulse on relay output 6 in 100ms units (0,1 to 6553,5 Seconds selectable) If you write onto this register, the relay output will be switched on for the desired time in 100ms units.
4x00207 3x00207 I:206 R/W PULSE RO7	generate a pulse on relay output 7 in 100ms units (0,1 to 6553,5 Seconds selectable) If you write onto this register, the relay output will be switched on for the desired time in 100ms units.
4x00208 3x00208 I:207 R/W PULSE RO8	generate a pulse on relay output 8 in 100ms units (0,1 to 6553,5 Seconds selectable) If you write onto this register, the relay output will be switched on for the desired time in 100ms units.

Register	Description
4x00301-302 3x00301-302 I:300-301 R/O PULSETIME_RO1	remaining time of the pulse on relay output RO1 in ms. Format: SINT32 0x12345678 -> 1st word: 0x1234 2nd word: 0x5678
4x00303-304 3x00303-304 I:302-303 R/O PULSETIME_RO2	remaining time of the pulse on relay output RO2 in ms. Format: SINT32 0x12345678 -> 1st word: 0x1234 2nd word: 0x5678
4x00305-306 3x00305-306 I:304-305 R/O PULSETIME_RO3	remaining time of the pulse on relay output RO3 in ms. Format: SINT32 0x12345678 -> 1st word: 0x1234 2nd word: 0x5678
4x00307-308 3x00307-308 I:306-307 R/O PULSETIME_RO4	remaining time of the pulse on relay output RO4 in ms. Format: SINT32 0x12345678 -> 1st word: 0x1234 2nd word: 0x5678
4x00309-310 3x00309-310 I:308-309 R/O PULSETIME_RO5	remaining time of the pulse on relay output RO5 in ms. Format: SINT32 0x12345678 -> 1st word: 0x1234 2nd word: 0x5678
4x00311-312 3x00311-312 I:310-311 R/O PULSETIME_RO6	remaining time of the pulse on relay output RO6 in ms. Format: SINT32 0x12345678 -> 1st word: 0x1234 2nd word: 0x5678
4x00313-314 3x00313-314 I:312-313 R/O PULSETIME_RO7	remaining time of the pulse on relay output RO7 in ms. Format: SINT32 0x12345678 -> 1st word: 0x1234 2nd word: 0x5678
4x00315-316 3x00315-316 I:314-315 R/O PULSETIME_RO8	remaining time of the pulse on relay output RO8 in ms. Format: SINT32 0x12345678 -> 1st word: 0x1234 2nd word: 0x5678

Register	Description
4x00317-318 3x00317-318 I:316-317 R/O PULSETIME_RO1	remaining time of the pulse on relay output RO1 in ms. Format: SINT32I 0x12345678 -> 1st word: 0x5678 2nd word: 0x1234
4x00319-320 3x00319-320 I:318-319 R/O PULSETIME_RO2	remaining time of the pulse on relay output RO2 in ms. Format: SINT32I 0x12345678 -> 1st word: 0x5678 2nd word: 0x1234
4x00321-322 3x00321-322 I:320-321 R/O PULSETIME_RO3	remaining time of the pulse on relay output RO3 in ms. Format: SINT32I 0x12345678 -> 1st word: 0x5678 2nd word: 0x1234
4x00323-324 3x00323-324 I:322-323 R/O PULSETIME_RO4	remaining time of the pulse on relay output RO4 in ms. Format: SINT32I 0x12345678 -> 1st word: 0x5678 2nd word: 0x1234
4x00325-326 3x00325-326 I:324-325 R/O PULSETIME_RO5	remaining time of the pulse on relay output RO5 in ms. Format: SINT32I 0x12345678 -> 1st word: 0x5678 2nd word: 0x1234
4x00327-328 3x00327-328 I:326-327 R/O PULSETIME_RO6	remaining time of the pulse on relay output RO6 in ms. Format: SINT32I 0x12345678 -> 1st word: 0x5678 2nd word: 0x1234
4x00329-330 3x00329-330 I:328-329 R/O PULSETIME_RO7	remaining time of the pulse on relay output RO7 in ms. Format: SINT32I 0x12345678 -> 1st word: 0x5678 2nd word: 0x1234
4x00331-332 3x00331-332 I:330-331 R/O PULSETIME_RO8	remaining time of the pulse on relay output RO8 in ms. Format: SINT32I 0x12345678 -> 1st word: 0x5678 2nd word: 0x1234

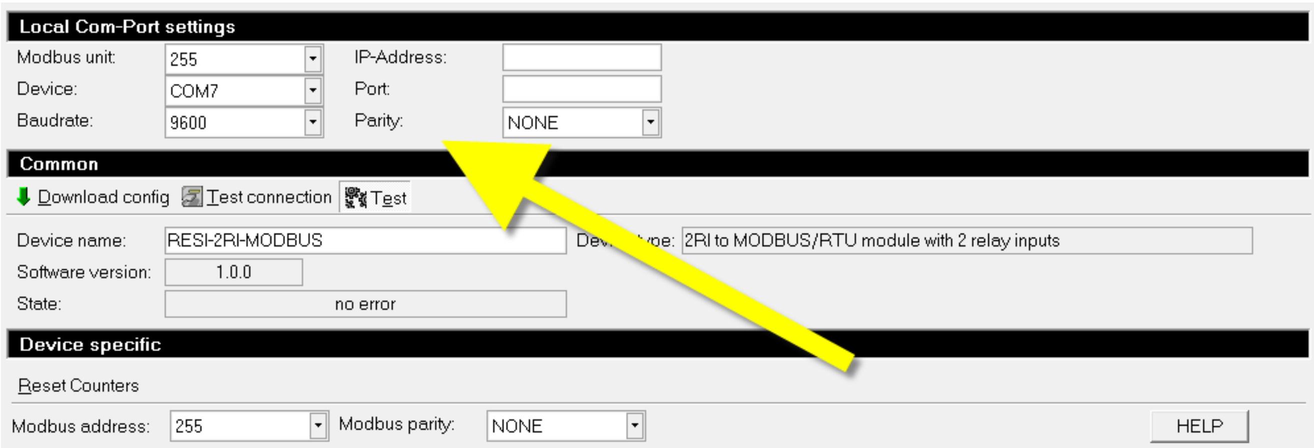
14 RESI's MODBUS Configurator

Almost all of our products can be used together with our MODBUSConfigurator software tool. You can configure and test the modules. If you have established successfully a connection to your module, you will get the following screen:

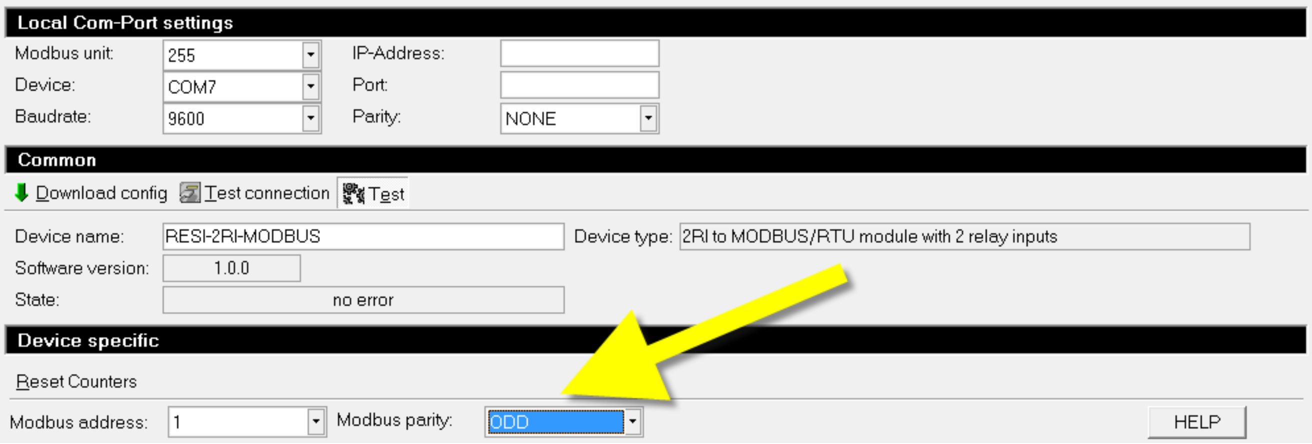


14.1 Changing the MODBUS/RTU Unit ID and the parity

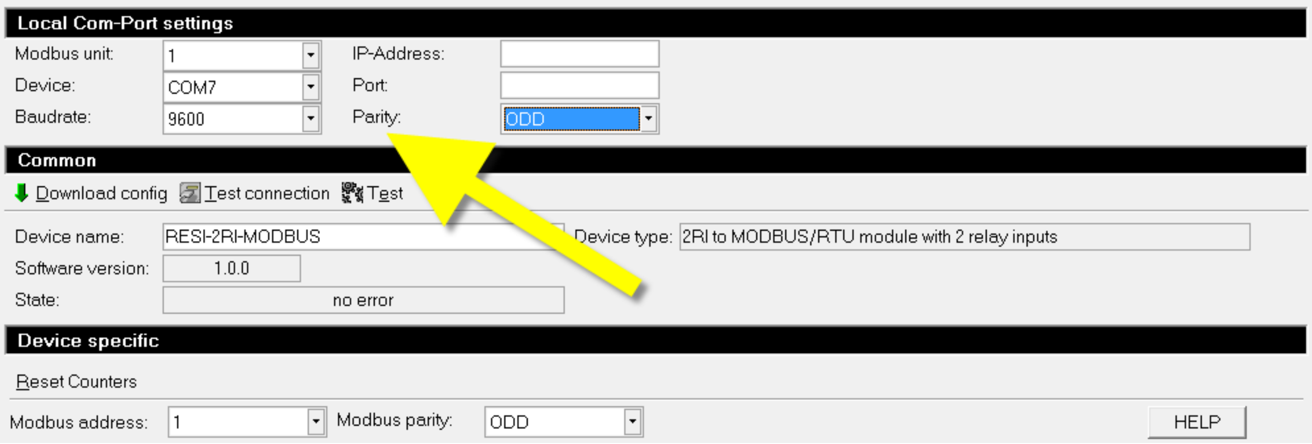
For every module you will see in the upper half of the screen the current communication parameters for the serial interface.



To change these settings, you have to follow this steps. Choose a new MODBUS/RTU unit address in the area „Device Specific“ or change the settings for the parity of the module.



The click onto the button „Download config“ to transfer the new settings into the connected module. If this was successful and the module has done a software reset, the new communication parameters are active in the module. Now you have to adopt the communication parameters in the upper half of the screen to ensure a connection with the new settings to the module:



With the button „Test connection“ you can check, if the new settings work correctly.