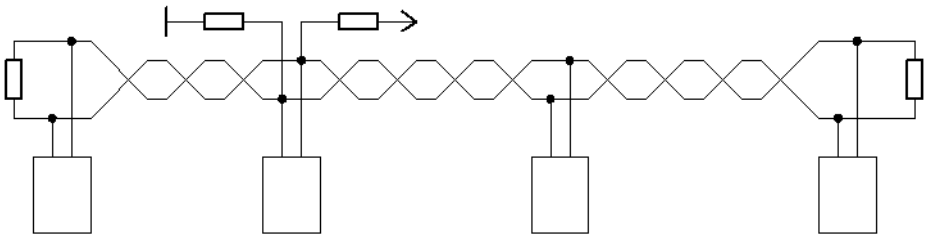
Technical Documentation



KSR MODBUS

Beluk GmbH
Taubenstrasse 1
86956 Schongau
Germany

Tel.: +49/(0)8861/2332-0
Fax: +49/(0)8861/2332-22
E-Mail: blr@beluk.de
Web: http://www.beluk.de

Document history

Date	Name	Revision	Change
21.12.04	CE	01	Initial document release
17.03.05	ATh	02	Features of new software revision (1.06) added to manual, general updates
11.05.05	ATh	03	General updates and settings via MODBUS added
10.01.06	ATh	04	General updates
21.09.07	Le	05	New connector on MODBUS hardware
16.02.12	Le	06	Add address space faultrecorder
19.07.18	SO	07	Content correction

Contents

Important Information!

If the above sign appears besides a text passage in the manual the reader is strongly advised to read the corresponding information, as it may be very important for usage of the device. It can contain safety advice or other information for the correct handling of the device. If the information is disregarded, the device may be inoperable or even damaged!

Additional documentation for the MODBUS protocol can be found at www.modbus.org. The MODBUS standards are also available from there.

1 Overview

The MODBUS Extension of the KSR offers the possibility to read values from the device and modify the settings of the device.

This document describes the transmission by use of the MODBUS-protocol. This protocol defines methods for data transmission and access control, but doesn't restrict the user to one single physical transmission system. In case of the KSR, RS485 is used on the physical layer. As this is a bus-capable interface it is possible to connect more than one KSR to a single pair of wire and access the units by use of an ID number.

A lot of commercial devices and PLCs are able to use the MODBUS protocol, either as bus master or slave. Various SCADA solutions are also available from different vendors. So, the integration of the KSR in an existing bus-system or setting up a new bus system is only a minor issue.

2 MODBUS / RS485

The implementation basically consists of two parts:

- The RS485 transmission is used for serial data transport. It is able to interconnect more than one device in a bus-like configuration. The RS485 protocol offers its “services” to the higher-level MODBUS protocol.
- The MODBUS protocol uses the underlying serial data transport layer (RS485 in this case) to communicate with several bus devices. It defines commands, address structures and data structures to access the slave device.

2.1 The physical layer - RS485 (defined in EIA485/ISO8482)

RS485 offers basic serial data transport to the higher-level MODBUS protocol layers. It is therefore called the “physical layer” of the bus system. Higher layers use the lower physical layer as a basic “service“ for data transport.

RS485 uses two data wires for serial transmission. Each of them is driven to 0V or 5V by the transmitting device. The two data wires always have different voltage levels. One state (one wire 5V, other wire GND) represents the logic “OFF”

state. The two wires exchange their voltages for the logic “ON” state. This differential transmission mode makes the RS485 bus very resistant against electro-magnetic distortions and therefore allows long transmission distances of more than 1000 metres.

The data transmission rates of the KSR can be selected between 1200, 2400, 9600, 19200 or 38400 baud. The parity can be selected between even, odd and no parity. All bus devices need to use the same settings. Standard settings are: 9600 baud and even parity.

There exist two different types of RS485:

- 2-wire RS485: This type uses only two data wires, which form one data channel. This means, that, after sending a request, the bus master has to deactivate its transmitter to make the data line free for the answering device. (Half-duplex mode)
- 4-wire RS485: this types uses one data line (=two wires) for the master->slave direction and another one (two more wires) for the slave ->master direction. The KSR does not support 4-wire RS485.

Both types, 2wire and 4wire, need another line to be connected, although it is not mentioned explicitly: the common ground GND. So, for the 2wire version you need a cable with 3 wires, for the 4wire version one with 5 wires! You should use a shielded cable, but never use the shield for GND connection. It should only be connected to protective ground to reduce electromagnetic influences.

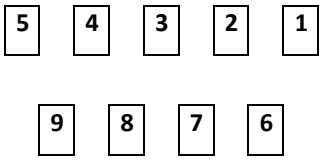
The RS485 bus interconnects more than one device (typically up to 32). To accomplish this, the several data signals have to be interconnected for all bus devices. These are the two data lines and the common ground GND. All devices are connected to the bus in parallel. Avoid using taps, as they tend to be the source of transmission errors if they are too long. You should always prefer direct connection of the device to the main bus wire.

One bus cable with all its devices is called a “bus segment”. Several segments can be interconnected by using “repeaters”.

2.1.1 Connection

There are two possible MODBUS interfaces:

a) 9-pin sub-d



PIN1 +5V (only for data line bias, **do never supply any other external circuits from this voltage output!**)

PIN2 GROUND for biasing and as common ground for all bus participants.

PIN5 D (B) - data signal B

PIN9 D (A) - data signal A

b) 3-pin connector

This connection variant uses a 3-pin connector. The connections can be seen in the picture. To use the MODBUS, one must connect the data lines + and - and the common ground (middle pin).

2.1.2 Line Termination

One very important issue is the termination of the bus line. This is definitely needed for a working bus system to cancel out echoes from the line ends, which would distort data signals. To terminate the bus cable, one must add a resistor at each end of the bus cable. The value of the resistor must match the cables impedance. At most times, 120 Ω is a good value to start with. Connect the termination resistor between data wires at each end of the bus segment.

Some devices, especially bus converters have built-in resistors. Please check the manuals for all devices used on the bus. If these internal resistors cannot be disabled, this has a very important influence on your bus: you must place these devices at one of the ends of the bus! As the bus has only two ends, you can use only two devices with fixed resistors per bus segment!

2.1.3 Line biasing

Another important issue is line biasing. If no device is actually transmitting, the data wires are left at floating. Because of the termination resistors, both will have nearly the same voltage. This could result in spurious data signals because of external influences. Line biasing is used to give the data wires a defined “off”-state in this case.

Two resistors of approx. 500-600Ω have to be connected from D(+) line to +5 V and from D(-) to GND. The two bias resistors are needed only once per bus, the position of the bias resistors is not important. They may be placed anywhere on the bus, even in the “middle”. Please check in the manuals of all bus devices, if resistors are internally provided!

The KSR has the voltages 5V and GND available on its bus connector, so these two resistors can be soldered inside the connector case.

2.1.4 Communication Indicator

The yellow LED on the backside of the device indicates an active transmission. It flashes only, if the device is actually communication with the bus master.

The communication indicator LED is available for both connector variants.

2.2 The MODBUS protocol

2.2.1 MODBUS - Description

The MODBUS protocol uses the RS485 as an underlying physical layer and implements the data transmission control mechanisms. It is therefore located on layer 2 ("link layer") of the OSI layer model for data exchange systems.

2.2.2 Serial data format and framing

The data is transmitted in fixed frames. The frames are separated by the bus being inactive for at least 3,5 characters. All data is organized in "protocol data units" (PDUs), which are transmitted over the serial bus system by the underlying physical protocol layer.

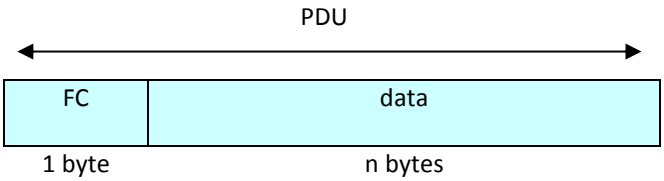


Illustration 1 : "protocol data unit" - PDU

The PDU consists of two parts:

- The "function code" (FC) gives a command, which defines, what the slave unit has to do.
- The data block consists of the corresponding data to a FC. Its usage depends on the FC, it can contain pure data but also register addresses for slave data access.

The PDU defines a single data unit, which has to reach a certain bus device in order to perform a function. The transportation differs, dependent on the physical layer that is used.

To be able to control the transmission, the PDU is extended with additional blocks of data for transmission control purposes. For RS485, this extension results in the "application data unit" (ADU).

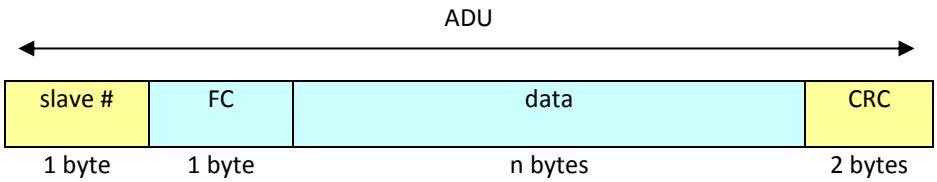


Illustration 2: "application data unit" - ADU

The application data unit, as it is used with the serial transmission over RS485, contains two additional blocks of data:

- The first field specifies the target for the data block, the "slave number" (=slave address)
- The transmission is additionally secured by a CRC16 error correction code.

2.2.3 Serial transmission modes

The protocol defines two different encodings for the frames' data contents. **The KSR always uses the RTU-mode! ASCII mode is not implemented and is mentioned here only for the purpose of completeness.**

"Remote terminal unit" (RTU)

In this transmission mode every 8bit data word contains two 4-bit hexadecimal numbers. They are transmitted as one complete byte; a maximum transmission density is reached. With every data word, the following information is transmitted:

- 1 start bit
- 8 data bits, "least significant bit" first
- 1 parity bit (if set)
- 1 stop bit for parity even or odd / 2 if parity is none to compensate missing parity bit

"American standard code for information interchange" (ASCII)

In ASCII-mode, the two 4-byte nibbles of an 8bit data word are transmitted separately in ASCII-code representation. A data byte which contents $5B_{hex}$ will be divided into two parts and transmitted separately as one byte each. The result is that TWO data bytes are transmitted with contents 35_{hex} (=ASCII-Code "5") and 42_{hex} (=ASCII-Code "B"). This data mode is intended for compatibility reasons and makes debugging on the transmission line easier but it also decreases transmission speed significantly.

2.2.4 Function codes

As already mentioned, the data packet contains "function codes" which specify a command from the bus master to the bus slave. The slave executes the command (if possible) and then answers with the same function code in the reply to acknowledge the command. The valid range for function codes is specified from 1 to 255, but only a part of it is actually really used. Please refer to the MODBUS specifications for detailed Information. If it is impossible for a slave to execute a command, it replies with an exception (=an error code). The function code of an exception packet is the function code of the received command, which caused the error, but it was changed in a certain way: The most significant bit is set by the slave to signal the error condition to the master. The contents of the data block specify the error in more detail.

The KSR supports function codes 03_{hex} (read holding register), 04_{hex} (read input register) and 06_{hex} (write single register).

2.2.5 Exception codes

If a slave is not able to execute a command, which was sent by the master, it answers with exception codes. A full list of codes can be found in the MODBUS specification. We do not include this list here, because the master software will be able to handle most exceptions automatically. If one has to program the MODBUS master stack by himself he will need the full specifications, and with that, he gets the full list of ERROR codes.

2.2.6 Master-Slave protocol

For communication, a master-slave protocol is used. Only the bus master is permitted to initiate a transfer. The "master" starts data exchange by sending a command to a slave by transmitting a data frame with the corresponding function code (=command) to the slave, which will then execute it.

- The unicast-mode is normally used to communicate on a Modbus system. One single slave is addressed by the slave number in the master's data packet. The valid address range is between 1 and 247. The slave then executes the command and answers by sending a data packet as acknowledge back to the master.

- Not in any case the master can receive an answer to his query: in multicast-mode all slaves on the bus are addressed in parallel. They all execute the same command, but none of them will respond. A multicast transfer is initiated by the master by using "0" as slave number.

2.2.7 KSR - MODBUS setup

If your device has MODBUS support, an additional entry is available in the “setup” - menu of the device. After entering it, you will have the possibility to select the following items:

- ADDRESS: This is the devices slave address (slave ID). The valid range is 1-247.
- BAUD RATE: Select the baud rate here. The valid range is 1200 - 38400 baud.
- PARITY: Select the parity to be EVEN, ODD or NONE.

The settings for baud rate and parity must be the same for all bus devices; the address must be unique for each device.

2.2.8 Address space

The data in the KSR is organized and accessed by means of addresses. Each address accesses one data word. The data words are always 16bits wide.

The KSR does not differentiate the addresses between the function codes. There is one big address space available and to access each address’s data, any valid function code can be used. Nevertheless, the data will only make sense when interpreted the correct way!

The data can be of the following types:

- REAL: this is a 32bit floating-point number, as defined in IEEE Standard 754
- UINT16: this is a 16bit unsigned integer value
- SINT16: this is a 16bit signed integer value
- SINT32: this is a 32bit signed integer value
- LONG: this is a 32bit integer value
- Struct: this is a predefined structure

As the data is organized in 16 bit wide words, a set of sequential addresses has to be read for longer data items. For these, the base address is given in the tables. To read a REAL with base address 12, one has to read two 16bit words from addresses 12 and 13. These two values need to be concatenated to form the desired result of 32 bits. Most SCADA software packages or PLCs can do this task for you.

There exist different types of addresses:

- **The MODBUS address always starts with 0 and can go up to 65535. It can be used with any function code.**
- **Certain PLCs lack correct handling of the 0 and therefore add 1 to the address. So, their addresses are always (MODBUS address +1) and start with 1.**
- **Some SCADA tools add an offset to determine the function code, which shall be used to access the device at the given address. They also sometimes add 1 to the MOSBUS address.**
  As an example, address 40001 would be "read MODBUS address 0 with function code 03_{hex}", 30012 would be "read MODBUS address 11 with function code 04_{hex}".
  Please refer to your software's manual to find out the correct addresses.

  The following tables always give the MODBUS addresses mentioned first in above list.

2.2.9 Measured values

The measured values are available beginning from address 0 in intervals of 6 data words. For each value in this table, the maximum and minimum values are also available. To read the maximum, just add 2 to the address of a value, for the minimum add 4. (Example: to get the minimum voltage L1-N, read from address 00034 = 00030+4). All these values can be accessed with function codes 03_{hex} and 04_{hex}.

Address	Value	Words	Type	Unit
00000	Current I-L1	2	REAL	A
00006	Current I-L2	2	REAL	A
00012	Current I-L3	2	REAL	A
00018	Current I-UB	2	REAL	A
00024	Voltage V-L1N	2	REAL	V
00030	Voltage V-L2N	2	REAL	V
00036	Voltage V-L3N	2	REAL	V
00042	Voltage V-L12	2	REAL	V
00048	Voltage V-L23	2	REAL	V
00054	Voltage V-L31	2	REAL	V

00060	Fundamental current If-L1	2	REAL	A
00066	Fundamental current If-L2	2	REAL	A
00072	Fundamental current If-L3	2	REAL	A
00078	Fundamental current If-UB	2	REAL	A
00084	Total harmonic distortion THD-I1	2	REAL	%
00090	Total harmonic distortion THD-I2	2	REAL	%
00096	Total harmonic distortion THD-I3	2	REAL	%
00102	Total harmonic distortion THD-IUB	2	REAL	%
00108	Total harmonic distortion THD-V1N	2	REAL	%
00114	Total harmonic distortion THD-V2N	2	REAL	%
00120	Total harmonic distortion THD-V3N	2	REAL	%
00126	Ambient temperature T	2	REAL	°C
00132	Damped current Ith-L1	2	REAL	A
00138	Damped current Ith-L2	2	REAL	A
00144	Damped current Ith-L3	2	REAL	A
00150	Damped current Ith-UB	2	REAL	A

All these values are mapped to another structure with includes only the actual meas values without min. and maximal values. It starts at adress 2816 and looks like this (analog to the structure with min. and max. values).

Address	Value	Words	Type	Unit
02816	Current I-L1	2	REAL	A
02818	Current I-L2	2	REAL	A
02820	Current I-L3	2	REAL	A
…	…	…	…	…
…	…	…	…	…

2.2.10 Harmonics

Harmonics are stored in separate arrays of FLOAT values for each current / voltage. The table below gives the corresponding base addresses. Each data array contains 64 values with 2 words length each. The first table entry holds the fundamental. As all values are normalized to 100%=fundamental, this first value always reads “100.0”. After the fundamental (=harmonic of order 1), the other harmonics follow in increasing order up to the 63^{d} harmonic. Remember that each FLOAT value occupies 2 words, so always increase the address by 2 for the next value.

If the current or voltage is too small to calculate valid harmonics from it, the value at the base address (= the fundamental) reads 0.0% instead of 100.0%. This indicates, that the higher harmonics for the current or voltage are also invalid!

All these values can be accessed with function codes $03_{hex\ and}$ 04_{hex}.

Address	Value	Words	Type	Unit
1538 (0602_{hex})	Base address for harmonics I - L1	64*2	REAL	%
1668 (0684_{hex})	Base address for harmonics I - L2	64*2	REAL	%
1798 (0706_{hex})	Base address for harmonics I - L3	64*2	REAL	%
1928 (0788_{hex})	Base address for harmonics I - N	64*2	REAL	%
2058 ($080A_{hex}$)	Base address for harmonics V - L1-N	64*2	REAL	%
2188 ($088C_{hex}$)	Base address for harmonics V - L2-N	64*2	REAL	%
2318 ($090E_{hex}$)	Base address for harmonics V - L3-N	64*2	REAL	%

2.2.11 Parameter settings

The parameters of the device can also be set via MODBUS. The parameters are stored from address 512 in UINT16 format. The table below gives the corresponding addresses.

All these values can be accessed with function codes 03_{hex}, 04_{hex} and 06_{hex}.

Address	Value	Words	Type	Unit
512	Parameter for ct-I123 ratio	1	UINT16	-
513	Parameter for vt ratio	1	UINT16	-
514	Parameter for ct-IUB ratio	1	UINT16	-
515	Parameter for thermic tau	1	UINT16	-

2.2.12 Relay settings

The behaviour of the output relays can also be set via MODBUS. It is stored in a binary bit mask, which is available from address 769 in UINT16 format. The least significant bit represents relay 1, the next relays follow in the bits with higher significance. In the case of relay inverting “0” means normally open, “1” means normally closed. In the case of relay reset mode “0” means automatic reset, “1” means manual reset.

All these values can be accessed with function codes 03_{hex}, 04_{hex} and 06_{hex}.

Address	Value	Words	Type	Unit
769	Relay normally open/closed	1	UINT16	-
770	Relay manual/automatic reset	1	UINT16	-

2.2.13 Alarm settings

The alarm data is stored in separate structs for each alarm. The table below gives the corresponding base address for each alarm. All these values can be accessed with function codes 03_{hex}, 04_{hex} and 06_{hex}.

Address	Value	Words	Type	Unit
1024	Base address for alarm data 1	6	Struct	-
1030	Base address for alarm data 2	6	Struct	-
1036	Base address for alarm data 3	6	Struct	-
1042	Base address for alarm data 4	6	Struct	-
1048	Base address for alarm data 5	6	Struct	-
1054	Base address for alarm data 6	6	Struct	-
1060	Base address for alarm data 7	6	Struct	-
1066	Base address for alarm data 8	6	Struct	-
1072	Base address for alarm data 9	6	Struct	-
1078	Base address for alarm data 10	6	Struct	-
1084	Base address for alarm data 11	6	Struct	-
1090	Base address for alarm data 12	6	Struct	-
1096	Base address for alarm data 13	6	Struct	-
1102	Base address for alarm data 14	6	Struct	-
1108	Base address for alarm data 15	6	Struct	-
1114	Base address for alarm data 16	6	Struct	-
1120	Base address for alarm data 17	6	Struct	-
1126	Base address for alarm data 18	6	Struct	-
1132	Base address for alarm data 19	6	Struct	-
1138	Base address for alarm data 20	6	Struct	-
1144	Base address for alarm data 21	6	Struct	-
1150	Base address for alarm data 22	6	Struct	-
1156	Base address for alarm data 23	6	Struct	-
1162	Base address for alarm data 24	6	Struct	-
1168	Base address for alarm data 25	6	Struct	-
1174	Base address for alarm data 26	6	Struct	-
1180	Base address for alarm data 27	6	Struct	-
1186	Base address for alarm data 28	6	Struct	-

1192	Base address for alarm data 29	6	Struct	-
1198	Base address for alarm data 30	6	Struct	-
1204	Base address for alarm data 31	6	Struct	-
1210	Base address for alarm data 32	6	Struct	-

Each alarm data consists of a struct containing 5 values composed of different data types. Every alarm is organized in the same way. The table below shows the content of an alarm.

Address	Value	Words	Type	Unit
Base address + 0	Source[1]	1	SINT16	-
Base address + 1	Limit	2	SINT32	-
Base address + 3	T-On x 100	1	UINT16	s
Base address + 4	T-Off x 100	1	UINT16	s
Base address + 5	Output[2]	1	UINT16	-

The limit can be set in the normal way by sending a value of the type SINT32. The values for T-On and T-Off can also be set directly by a value of the type UINT16, but remember to use the factor 100. The following table shows the code for the different sources.

Code	Source	Type	Factor
-1	inactive	SINT16	-
0	Current I-L1	SINT16	100
1	Current I-L2	SINT16	100
2	Current I-L3	SINT16	100
3	Current I-UB	SINT16	100
4	Voltage V-L1N	SINT16	1
5	Voltage V-L2N	SINT16	1
6	Voltage V-L3N	SINT16	1

[1] see table for source

[2] see illustration for output

7	Voltage V-L12	SINT16	1
8	Voltage V-L23	SINT16	1
9	Voltage V-L31	SINT16	1
10	Fundamental current If-L1	SINT16	100
11	Fundamental current If-L2	SINT16	100
12	Fundamental current If-L3	SINT16	100
13	Fundamental current If-UB	SINT16	100
14	Total harmonic distortion THD-I1	SINT16	100
15	Total harmonic distortion THD-I2	SINT16	100
16	Total harmonic distortion THD-I3	SINT16	100
17	Total harmonic distortion THD-IUB	SINT16	100
18	Total harmonic distortion THD-V1N	SINT16	100
19	Total harmonic distortion THD-V2N	SINT16	100
20	Total harmonic distortion THD-V3N	SINT16	100
21	Ambient temperature T	SINT16	1
22	Damped current Ith-L1	SINT16	100
23	Damped current Ith-L2	SINT16	100
24	Damped current Ith-L3	SINT16	100
25	Damped current Ith-UB	SINT16	100

The following illustration shows the necessary code for the different outputs.

Output

x	x	HOLD	MASK	TYPE	TYPE	INV	DISP	R8	R7	R6	R5	R4	R3	R2	R1

bit16 bit1

R1 – R8: Output relay 1 – 8. “0” represents relay inactive if trip or alarm occurs, relay active if “1”.

DISP: Output display. “0” represents display inactive if trip or alarm occurs, display active if “1”.

INV: “0” represents value > limit, “1” means value < limit.

TYPE: Code 00 marks OL

Code 01 marks UB

Code 10 marks OV

Code 11 marks UV

MASK: “0” represents trip, “1” means alarm.

HOLD: “0” represents alarm automatic reset, “1” means alarm manual reset.

All settings send by MODBUS are used immediately, but remember this information is only stored in the working memory, after a power blackout these settings will be lost. To store the settings durable, you have to store the data in the non-volatile memory.

2.2.14 Settings storage

For storing the settings durable in the non-volatile storage (EPROM) you have to use the table below.

All these values can be accessed with function codes 03_{hex}, 04_{hex} and 06_{hex}.

Address	Value	Words	Type	Unit
2560	Store parameter data in EPROM	1	UINT16	-
2561	Store alarm and relay data in EPROM	1	UINT16	-

If you send “24276” to the address the corresponding data will be stored durable in the EPROM. After the correct takeover this is confirmed by “1” in the referring word.

2.2.15 Read out fault recorder

The following table shows the base address for all stored values. X represents the storage number

Address	Value	Words	Type	Unit
10000+28*X	Alarm number	1	UINT16	-
10001+28*X	on / off event	1	UINT16	-
10002+28*X	month_year of event	1	hex	-
10003+28*X	day_hour of event	1	hex	-
10004+28*X	min_sec of event	1	hex	-
10005+28*X	source (number between 1-49 see table 2)	1	SINT16	-
10006+28*X	limit	2	Float	
10008+28*X	max. value	2	Float	
10010+28*X	time delay	2	Float	s
10012+28*X	flags (relays see table 3 for meaning of the bits)	1	SINT16	-
10013+28*X	Voltage L12	2	Float	V
10015+28*X	Voltage L23	2	Float	V
10017+28*X	Voltage L31	2	Float	V
10019+28*X	Current I1	2	Float	A
10021+28*X	Current I2	2	Float	A
10023+28*X	Current I3	2	Float	A
10025+28*X	Current IN	2	Float	A

Source	Meaning	Source	Meaning
0	I-L1	14	THD1
1	I-L2	15	THD2
2	I-L3	16	THD3
3	I-UB	17	THD4
4	U-L1N	18	THDU1N
5	U-L2N	19	THDU2N
6	U-L3N	20	THDU3N
7	U-L12	21	T
8	U-L23	22	Ith-1
9	U-L31	23	Ith-2
10	If-1	24	Ith-3
11	If-2	25	Ith-UB
12	If-3	26	Iuc-4
13	If-Ub		

Table 3

Alarm Flags

x	Disable by DI	Store in FR	Reset Display alarm	Alarm Type	Alarm Type	Alarm Trigger	Display Alarm	Relay 8	Relay 7	Relay 6	Relay 5	Relay 4	Relay 3	Relay 2	Relay 1
bit16															bit1

Relay 1 – Relay 8: Output relay 1 – 8. “0” represents relay inactive if trip or alarm occurs, relay active if “1”.

Display Alarm: Output display. “0” represents display inactive if trip or alarm occurs, display active if “1”.

Alarm Trigger: “0” represents value > limit, “1” means value < limit.

Alarm Type: Code 00 marks OL

Code 01 marks UB

Code 10 marks OV

Code 11 marks UV

Reset Display Alarm: “0” means auto reset, “1” means manual reset.

Store in FR: “0” represents "not active" fault recording for this setting, “1” means "active" fault recorder for this setting.

Disable by DI: "0" means for this alarm the blocking via DI is "not active", "1" represents "active".

There are other values present in the devices memory than the ones mentioned above. They can contain important setup data for the device. Do never write to any address if you’re not sure what data it contains!

3 Trouble Shooting

If the bus connection isn't working correctly, please check the following points:

1. If there is no communication at all, then the error must be looked for between the EMM5 and the PC!

   Possible causes can be:

   - Check adjustment of baud rate, parity and address at the EMM5, possibly make changes in the configuration
   - Possibly the bus lines A and B are interchanged, if necessary rectify
   - Check adjustments of the converter RS485/RS232, possibly use the data sheet of the converter
   - Perhaps there is a multiple reservation of the interface at the PC, if necessary stop this multiple reservation
   - Check termination and bias resistors, if necessary rectify

2. Does the cable of the bus connection have any damages? All plug connections are correct? If necessary replace.
3. Is the pin assignment of the RS485 connection correct? If necessary rectify.
4. The shielding of the bus connection may not be grounded. If this is the case, separate the shielding from the grounding.
5. Is the communication possible, but there are problems with the software of the customer, then please check the following points:
   - Check adjustment of bus address, parity and baud rate in the software
   - Check data format