thermokon
Sensortechnik GmbH

Interface Description
for
SRC/STC RS485 MODBUS

1 Errata

Version	Date	Description
B	21.10.2008	(1) STC Firmware V2.0 • transmit functionality • 8 channels for transmitting data from Modbus-network to EasySens-network (2) SRC and STC-Description
C	24.11.2011	Corrections SAB02
D	30.11.2011	Appendices
E	14.10.2013	Appendices to new Hardware (Rev B) and Firmware 3.0.0
F	02.12.2013	Additions to the register description

2 Introduction

This documentation describes the serial interface of the radio receiver SRC-RS485 MODBUS and the radio transceiver STC-RS485 MODBUS. The MODBUS-Protocol developed by the company Modicon is an open protocol for the communication of several intelligent devices on master-slave-basis.
Both support the mapping of up to 32 EasySens sensors in a MODBUS-network and the transceiver supports in addition up to 8 EasySens transmitters for data transmission from a EasySens network in a MODBUS-network.
Further information and definitions on the MODBUS can be obtained under www.modbus.org
If a SRC-RS485 is used the chapters 2.6.2 and 6 of this description are irrelevant.

<u>**From firmware version 1.4 keys can be evaluated.**</u>

3 Device Description

3.1 Hardware Installation

The receiver and the transceiver can be connected by means of a twisted-pair cable (line resistance 120 Ohm). For detailed information on installation and mounting, please see the product data sheet SRC-RS485-Modbus resp. STC-RS485-Modbus and the data sheet wiring_rs485_network.pdf.

3.2 RS485 Transceiver

The maximum number of bus participants without use of a repeater is preset by the RS485-transceiver. The transceiver used allows 32 devices per bus segment at maximum.

3.3 Protocol

The receiver module SRC-485-MODBUS and the transceiving module STC-RS485-MODBUS are slave-bus participants, only allowed to send to the bus on demand of the master. The protocol corresponds to the defaults of:
- MODBUS Application Protocol Specification V1.1
- MODBUS via Serial Line Specification & Implementation guide V1.0

3.4 Configuration Options

By means of dip switches the device can be adapted to the respective bus topology. The following can be adjusted:

- bus address of the device (1 - 247)
- bus terminating resistor 120 Ohm
- transmission mode RTU or ASCII
- baud rate 9600, 19200, 38400 or 57600 (38400 and 57600 only available on hardware Rev B and Firmware 3.0.0)
- even parity, odd parity or no parity

As the data sheet contains a detailed description on position and meaning of the jumpers, please refer to the file „Produktblatt_src_rs485.pdf“ resp. „Produktblatt_stc_rs485.pdf“.

Important remarks for operation in the Master/Slave-System:

!! The bus address must be differently adjusted for each device
!! Transmission mode, baud rate and parity must be identical

3.5 Control Commands Supported

The following MODBUS – control commands are supported:

Description	Function Code	
Read bits	01 (hex)	1 (dec)
Read register	03 (hex)	3 (dec)
Write individual bit	05 (hex)	5 (dec)
Write individual register	06 (hex)	6 (dec)
Write several bits	0F (hex)	15 (dec)
Write several registers	10 (hex)	16 (dec)

Table 1

Note:
Function codes 02 and 04 are not available since hardware Rev B and firmware 3.0.0!

3.6 Data Administration

All data in a MODBUS-Slave are allocated to addresses. The access to the data (read or write) is made by the corresponding control command and the identification of the corresponding data address.

3.6.1 Sensor Data

3.6.1.1 Register Allocation of Sensor Data

According to the definition, a register in a MODBUS device consists of 16 bit. The data for administration of up to 32 Thermokon EasySens sensors are lying in the registers 1 - 320, whereas 10 registers are allocated to each sensor (see table 1):

Sensor 1 Register 1 - 10_{dez}
Sensor 2 Register 11 - 20_{dez}
:
Sensor 32 Register 311 - 320_{dez}

Holding Register – Modbus Command 03_{hex}, 06_{hex} and 10_{hex}

<table>
<tr><th rowspan="2">Register</th><th rowspan="2">Data Address</th><th colspan="8">MSB</th><th colspan="8">LSB</th><th></th></tr>
<tr><th>Bit 15</th><th>Bit 14</th><th>Bit 13</th><th>Bit 12</th><th>Bit 11</th><th>Bit 10</th><th>Bit 09</th><th>Bit 08</th><th>Bit 07</th><th>Bit 06</th><th>Bit 05</th><th>Bit 04</th><th>Bit 03</th><th>Bit 02</th><th>Bit 01</th><th>Bit 00</th><th></th></tr>
<tr><td>1 R/W</td><td>0</td><td colspan="8">not used</td><td colspan="8">ORG</td><td rowspan="10">Data Sensor 1</td></tr>
<tr><td>2 R/W</td><td>1</td><td colspan="8">ID-Byte-3</td><td colspan="8">ID-Byte-2</td></tr>
<tr><td>3 R/W</td><td>2</td><td colspan="8">ID-Byte-1</td><td colspan="8">ID-Byte-0</td></tr>
<tr><td>4 R</td><td>3</td><td colspan="8">not used</td><td colspan="8">Data-Byte-3</td></tr>
<tr><td>5 R</td><td>4</td><td colspan="8">not used</td><td colspan="8">Data-Byte-2</td></tr>
<tr><td>6 R</td><td>5</td><td colspan="8">not used</td><td colspan="8">Data-Byte-1</td></tr>
<tr><td>7 R</td><td>6</td><td colspan="8">not used</td><td colspan="8">Data-Byte-0</td></tr>
<tr><td>8 R</td><td>7</td><td colspan="8">Receive-Time-Byte-1</td><td colspan="8">Receive-Time-Byte-0</td></tr>
<tr><td>9 R/W</td><td>8</td><td colspan="8">not used</td><td colspan="8">Actuator channel</td></tr>
<tr><td>10 R</td><td>9</td><td colspan="8">not used</td><td colspan="8">not used</td></tr>
<tr><td colspan="19">:
:</td></tr>
<tr><td>311R/W</td><td>310</td><td colspan="8">not used</td><td colspan="8">ORG</td><td rowspan="10">Data Sensor 32</td></tr>
<tr><td>312 R/W</td><td>311</td><td colspan="8">ID-Byte-3</td><td colspan="8">ID-Byte-2</td></tr>
<tr><td>313 R/W</td><td>312</td><td colspan="8">ID-Byte-1</td><td colspan="8">ID-Byte-0</td></tr>
<tr><td>314 R</td><td>313</td><td colspan="8">not used</td><td colspan="8">Data-Byte-3</td></tr>
<tr><td>315 R</td><td>314</td><td colspan="8">not used</td><td colspan="8">Data-Byte-2</td></tr>
<tr><td>316 R</td><td>315</td><td colspan="8">not used</td><td colspan="8">Data-Byte-1</td></tr>
<tr><td>317 R</td><td>316</td><td colspan="8">not used</td><td colspan="8">Data-Byte-0</td></tr>
<tr><td>318 R</td><td>317</td><td colspan="8">Receive-Time-Byte-1</td><td colspan="8">Receive-Time-Byte-0</td></tr>
<tr><td>319 R/W</td><td>318</td><td colspan="8">not used</td><td colspan="8">Actuator channel</td></tr>
<tr><td>320 R</td><td>319</td><td colspan="8">not used</td><td colspan="8">not used</td></tr>
</table>

Table 2: Register allocation of sensor data

3.6.1.2 Identification Code

The first 3 registers receive the identification code of a sensor, which identifies each sensor clearly. It consists of ORG-byte (device identification 1 byte / 4 byte sensor) and the ID-bytes 0 to 3.

These registers are marked by „R/W“ and have read and write access. These data are stored in the EEPROM and remain unchanged after a voltage reset, thus.

3.6.1.3 Data-Bytes Sensors (ORG = 6 oder ORG = 7)

The following four registers contain the sensor data bytes 0 – 3. The meaning of the data and how they can be processed is depending on the sensor type. Thus, please see the corresponding data sheets. The registers in question are marked by „R“ and can only be read via the Modbus.

Data-Byte 0

- For digital values, e.g. SR04 xx with presence key
- With self-holding function: status change of presence key is stored in the device until the next Modbus inquiry and is sent

Data-Byte 1

- Temperature
- Resolution 0 – 255 Bit. For the measuring range, please see the data sheet of the sensor
- It is possible to invert temperature (see chapter 3.6.1.9)

Data-Byte 2

- Set point adjuster with SR04 xx

- Humidity withSR04 rH

Data-Byte 3

- Fan speed step with SR04 xx
- Set point adjuster with SR04 rH
- Window contact SRW01

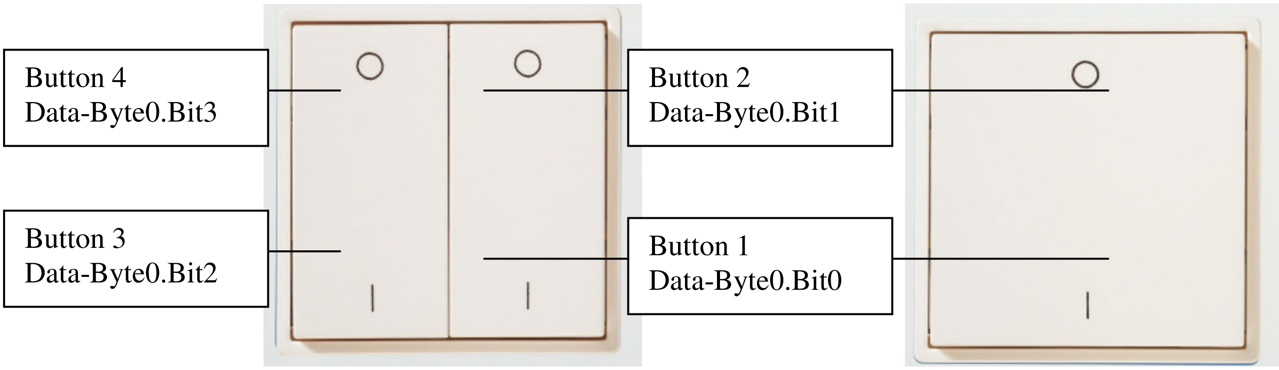
A key evaluation for light control does not make any sense, as the Master-Slave-System of the Modbus is too slow. Button actuations could get lost in dependence of the inquiry interval.

3.6.1.4 Data-Bytes Keys (ORG = 5)

The following four registers include the key data Data-Bytes 0 - 3.

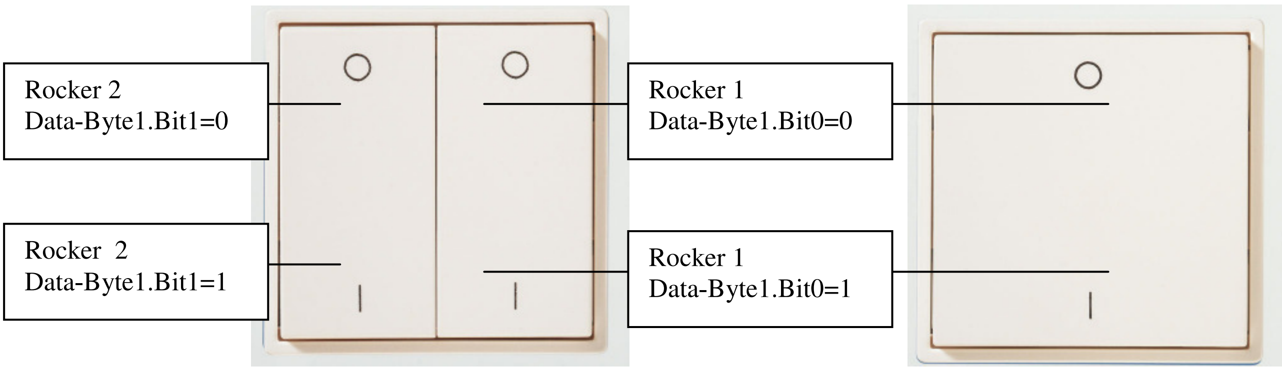
Data-Byte 0

- Current status of keys
- Button function
- All status changes of the key are stored in the device till the next Modbus inquiry and are sent, then.
- After an inquiry of the register, the Data-Byte0 is reset, unless a button is still pressed.
- bit = 1 ==> button pressed, bit = 0 ==> button not pressed



Data-Byte 1

- Current status of rocker
- Switch function
- Button I: Bit0/Bit1 = 1
- Button O: Bit0/Bit1 = 0



Data-Byte 2

- Current status of button
- Button function – status changes of the button are stored in the device till the next Modbus inquiry and are sent, then.
- The button pressed last is stored as RAW value.
- The allocation of the RAW values to the respective button is shown in the data sheet of the keys.

Data-Byte 3

- Current status of button
- The allocation of the RAW values to the respective button is shown in the data sheet of the keys.
- Pressed buttons are not buffered.

Due to the fact, that the Master-Slave-System is too slow with the Modbus, it might come to delays with button actuations.

3.6.1.5 Sensor Monitoring Time

The respective eight register "receive time" shows, how many time went by since the last radio telegram of the sensor was received.

Data that are marked by „not used" are always output with the value „0" with data output.

3.6.1.6 Actuator Channel

A value within the range 1… 8 causes that with the telegram receipt of a learned-in sensor the data of the corresponding transmit channel is automatically sent. Example of use is the direct coupling of a sensor and a SAB01-actuator.

3.6.1.7 Register Allocation Modbus-Configuration

Holding Register – Modbus Command 03_{hex}, 06_{hex} and 10_{hex}

Register	Data Address	MSB															LSB	
		Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 09	Bit 08	Bit 07	Bit 06	Bit 05	Bit 04	Bit 03	Bit 02	Bit 01	Bit 00	
321R/W	320	Min-Response-Time-Byte-1								Min-Response-Time-Byte-0								min. response time

Table 3: Register allocation for minimum response time

Register 321 has read and write access and defines the minimum time (ms) that must pass by before a slave responses to a master's inquiry. These data are stored in the EEPROM and remain unchanged even after a voltage reset. Preset value: 10 ms, smallest allowed value 5 ms.

Note: Since hardware Rev B and firmware 3.0.0 this parameter is only available in ASCII mode.

3.6.1.8 Bit Allocation for Sensor Learn Mode

The bit values listed in table 4 are marked by „R/W“ and have read and write access. If for example Bit1 is described by the value „1“ the learn mode for sensor 1 is activated.
In the learn mode the receiver waits for a learn-in radio telegram of a sensor, which is produced by pushing the learn button at the sensor. With a successful transmission of the radio telegram the receiver describes the identification code of the sensor in the corresponding register (see table 1 and chapter 4 : Learning of sensors).

Coils – Modbus Command 01_{hex}, 05_{hex} and $0F_{hex}$

Bit	Data Address	Value = 1 ==> Learn mode active
1 R/W	0	Learn mode Sensor 1
2 R/W	1	Learn mode Sensor 2
	:	
32 R/W	31	Learn mode Sensor 32

Table 4

3.6.1.9 Bit Allocation for Configuration „Invert of Data Byte 1“

For Thermokon temperature sensors and room operating panels data byte 1 is used for the transmission of temperature values. The sensor types SR04 (without relative air humidity) and SR65 send the temperature value inverted i.e. the minimum temperature value corresponds to the value 255 in data byte 1 and the maximum temperature value corresponds to the value 0 in data byte 1 (please see the corresponding product data sheets).

The configuration bits 33 to 64 offer the possibility to invert the temperature value for each sensor, so that the temperature is output proportional to the values 0 to 255. These data are stored in the EEPROM and are maintained after power on reset.

Coils – Modbus Command 01_{hex}, 05_{hex} and $0F_{hex}$

Bit	Data Address	Storage area	Value = 1 ==> Invert data byte 1
33 R/W	32	EEPROM	Sensor 1
34 R/W	33	EEPROM	Sensor 2
		:	
64 R/W	63	EEPROM	Sensor 32

Table 5

3.6.2 EasySens Transmitter Data

3.6.2.1 Register Allocation of Transmitter Data

The registers of the transmitters are set up in the same way as the sensors. According to the definition, a MODBUS device consists of 16 bit. In registers 401-480 data for displaying of up to 8 EasySens transmitters can be found, whereas 10 registers are allocated to each transmitter (see table 2):

Sender 1	Register 401 - 410dez
Sender 2	Register 411 -420dez
:	
Sender 8	Register 471 - 480dez

The write instruction “Write several registers (0x10)“ can be used for all registers of a transmitter. Only the data for the ORG-Byte, Data-Bytes and STATUS-Byte are taken over.
The data received by the Modbus network are sent according to the EnOcean protocol.
In order to release a transmission, the corresponding transmission bit must be set (see chapter 2.6.2.5).

Holding Register – Modbus Command 03_{hex}, 06_{hex} and 10_{hex}

<table>
<tr><td rowspan="2">Register</td><td rowspan="2">Data Address</td><td colspan="8">MSB</td><td colspan="8">LSB</td><td></td></tr>
<tr><td>Bit 15</td><td>Bit 14</td><td>Bit 13</td><td>Bit 12</td><td>Bit 11</td><td>Bit 10</td><td>Bit 09</td><td>Bit 08</td><td>Bit 07</td><td>Bit 06</td><td>Bit 05</td><td>Bit 04</td><td>Bit 03</td><td>Bit 02</td><td>Bit 01</td><td>Bit 00</td><td></td></tr>
<tr><td colspan="19"></td></tr>
<tr><td>401 R/W</td><td>400</td><td colspan="8">not used</td><td colspan="8">ORG</td><td rowspan="10">Data Sender 1</td></tr>
<tr><td>402 R</td><td>401</td><td colspan="8">ID-Byte-3</td><td colspan="8">ID-Byte-2</td></tr>
<tr><td>403 R</td><td>402</td><td colspan="8">ID-Byte-1</td><td colspan="8">ID-Byte-0</td></tr>
<tr><td>404 R/W</td><td>403</td><td colspan="8">not used</td><td colspan="8">Data-Byte-0</td></tr>
<tr><td>405 R/W</td><td>404</td><td colspan="8">not used</td><td colspan="8">Data-Byte-1</td></tr>
<tr><td>406 R/W</td><td>405</td><td colspan="8">not used</td><td colspan="8">Data-Byte-2</td></tr>
<tr><td>407 R/W</td><td>406</td><td colspan="8">not used</td><td colspan="8">Data-Byte-3</td></tr>
<tr><td>408 R/W</td><td>407</td><td colspan="8">not used</td><td colspan="8">STATUS</td></tr>
<tr><td>409 R</td><td>408</td><td colspan="8">not used</td><td colspan="8">not used</td></tr>
<tr><td>410 R</td><td>409</td><td colspan="8">not used</td><td colspan="8">not used</td></tr>
<tr><td colspan="19">:
:</td></tr>
<tr><td>471 R/W</td><td>470</td><td colspan="8">not used</td><td colspan="8">ORG</td><td rowspan="10">Data Sender 8</td></tr>
<tr><td>472 R</td><td>471</td><td colspan="8">ID-Byte-3</td><td colspan="8">ID-Byte-2</td></tr>
<tr><td>473 R</td><td>472</td><td colspan="8">ID-Byte-1</td><td colspan="8">ID-Byte-0</td></tr>
<tr><td>474 R/W</td><td>473</td><td colspan="8">not used</td><td colspan="8">Data-Byte-0</td></tr>
<tr><td>475 R/W</td><td>474</td><td colspan="8">not used</td><td colspan="8">Data-Byte-1</td></tr>
<tr><td>476 R/W</td><td>475</td><td colspan="8">not used</td><td colspan="8">Data-Byte-2</td></tr>
<tr><td>477 R/W</td><td>476</td><td colspan="8">not used</td><td colspan="8">Data-Byte-3</td></tr>
<tr><td>478 R/W</td><td>477</td><td colspan="8">not used</td><td colspan="8">STATUS</td></tr>
<tr><td>479 R</td><td>478</td><td colspan="8">not used</td><td colspan="8">not used</td></tr>
<tr><td>480 R</td><td>479</td><td colspan="8">not used</td><td colspan="8">not used</td></tr>
</table>

Table 6: Register allocation to transmission data

3.6.2.2 Identification Code

Registers 2 and 3 of each sensor have a unique identification code, which is derived from the EasySens module. The allocation is as follows: Sender 1 = BasisID+0, Sender1 = BasisID+1, …, Sender8 = BasisID+7

These registers are marked by ”R“ and can only be read.

3.6.2.3 ORG-Byte and Data-Bytes for Sender

The ORG-Register determines which telegram type (ORG-Byte) should be sent. There are four registers available for the data.
The meaning of the data bytes are different and depend on the values to be transmitted. Please note the corresponding description of the data to be sent.

The registers are marked by ”R/W“ and have read and write access.

3.6.2.4 Status-Byte

In the Status-Byte additional information can be transmitted according to the EnOcean protocol. The register is marked by ”R/W“ and has a read and write access.

3.6.2.5 Send Telegram

The bit values listed in table 7 are marked by “R/W“ and have a read and write access. For sending a telegram, the Coil 1 must be set. After a successful transmission, the Coil is automatically reset to 0.

Coils – Modbus Command 01_{hex}, 05_{hex} and $0F_{hex}$

Bit	Data-Address	Value = 1 ==> Transmission mode active
65 R/W	64	Transmission bit Sender 1
66 R/W	65	Transmission bit Sender 2
	:	
72R/W	71	Transmission bit Sender 8

Table 7: Transmission bits

4 Data Transmission

4.1 Master/Slave Protocol

One master and one or more slaves are connected to the serial bus. The communication between master and slave is exclusively controlled by the master. The slaves are only allowed to send if they have been addressed by the master before. Slaves only send back to the master, never to another slave.

4.2 Data Frame

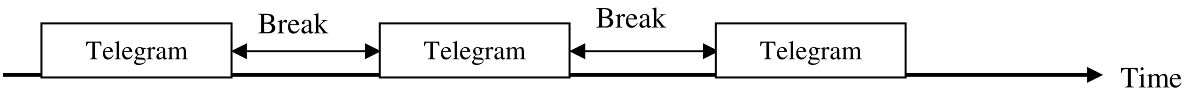
The datas are sent to the bus in accordance to severely defined defaults:

Address	Control Command	Data	Checksum

In general, a MODBUS telegram starts with the address of the slave, followed by a control command (e.g. read register) and the data. By means of the checksum at the telegram end, the bus participants can recognize transmission errors.

4.3 Transmission Mode RTU

In the transmission mode RTU telegrams are separated by means of transmission breaks.



The period of the transmission breaks for separating telegrams is depending on the adjusted baud rate and amounts to 3,5 * word transmission time (11 bit). With 9600 baud at least 4 ms must pass by and with 19200 at least 2 ms. must pass by between two telegrams.

4.3.1 Telegram Layout

Address	Command Control	Data	Checksum	
1 Byte	1 Byte	0 - 100 byte	CRC Low	CRC High

4.3.2 Calculation of CRC-Checksum

The CRC checksum (Cyclicyal Redundancy Check) is calculated by the sender out of all bytes transmitted and is attached to the message.

The receiver re-calculates the CRC checksum and compares it with the checksum received. If the values do not correspond, a transmission error is assumed and the data received are rejected.

The least significant byte of the 16 bit large checksum is set to the penultimate location and the most significant byte is set at last location.

Calculation of checksum (Programming example in C):

```
crc = 0xFFFF;                           // CRC-Check, Initialisierung
for(i = 0; i < Telegrammlänge-2; i++)
        crc = crc_calc(crc, Telegrammdaten[i]);

crc_low  = crc & 0x00FF;                // Low-Byte
crc_high = (crc & 0xFF00) >> 8;         // High-Byte

// Funktionsdefinition CRC Berechnen
unsigned int    crc_calc(unsigned int   crc_temp, unsigned int data)
{
        unsigned int    Index_CC=0;             // Schleifenzähler
        unsigned int    LSB=0;                  // Hilfsvariable

        // Exclusive-Oder des 8Bit-Char mit den unteren 8Bit von CRC
        crc_temp = ( ( crc_temp ^ data) | 0xFF00) & (crc_temp | 0x00FF) ;

        for(Index_CC = 0; Index_CC<8; Index_CC++)
        {
                LSB = (crc_temp & 0x0001);
                crc_temp        >>= 1;
                if(LSB)
                crc_temp = crc_temp ^ 0xA001;           // calculation polynominal für CRC16
        }

        return(crc_temp);
}
```

4.4 Transmission Mode ASCII

The ASCII transmission mode does not make that high demands on the computer speed of the bus participants. The telegrams are not separated by break times, but by ASCII control characters.

4.4.1 Telegram Layout

The ASCII control character „ : “ always identifies the beginning of a telegram. The ASCII control characters „CR“ and „LF“ identify the end of a telegram. The telegram data are output hexa-decimal in the ASCII format:

e.g..: 197dez (1Byte) = C5hex (1 Byte) = C (1 Byte) 5 (1 Byte) ASCII

As one data byte is displayed by 2 ASCII characters, the number of data bytes to be transmitted is doubled compared with the RTU mode.

Start 1 char	Address 2 char	Control command	Data 0 - 2 x 100 char	Checksum LRC 2 char	End 2 char
:					**CR LF**

4.4.2 Calculation of LRC-Checksum

The LRC checksum (Longitudinal Redundancy Check) is calcualted by the sender out of all bytes transmitted (without „:“, „CR“, „LF“) and pasted in the message of „CR„ and „LF“. The receiver re-calculates the LRC checksum and compares it with the checksum received. If the values do not correspond, a transmission error is assumed and the data received are rejected.

The most significat ASCII character of the 8 bit large checksum is sent in the telegram before the least significant ASCII character.

Calculation of checksum (programming example in C):

```
lrc = 0;
for(i = 1; i < Telegrammlänge -4; i++)
        lrc = lrc + Telegrammdaten [i];

lrc = 0xFF - lrc;
lrc = lrc + 1;
```

5 Learning-in of Sensors

The receiver only administers the data of radio sensors, whichs identification codes are known, i.e. the codes stored in the EEPROM. In accordance with table 1, 10 registers are allocated to each sensor, whereas the first three registers contain the identification code.

The sensor identification code is either described to the register by a MODBUS telegram or is indepenetly stored in the learn mode out of a received „learn radio telegram".

5.1 Learning-in via MODBUS – Write Command

By the control command „Write Register" (10hex) the identification code can directly be described into the corresponding register. The identification code (ORG-Byte and ID-Bytes) clearly identifies each sensor and is marked on the device lable of the radio sensors.

Example: Learn-in of sensor 2 with ID = 01 23 D5 E7 (hex) and ORG-Byte = 07 (hex)

Master - Telegram in transmission mode RTU:

Slave Address	Command	Start Address		Number of Registers		Number of Bytes	Data Register 0A		Data Register 0B		Data Register 0C		Checksum	
		H Byte	L Byte	H Byte	L Byte		H Byte	L Byte	H Byte	L Byte	H Byte	L Byte	L CRC	H CRC
02	10	00	0A	00	03	06	00	07	01	23	D5	E7	CRC	

Slave – Response telegram in transmission mode RTU:

Slave Address	Command	Start address		Number of Register		Checksum	
		H Byte	L Byte	H Byte	L Byte	L CRC	H CRC
02	10	00	0A	00	03	CRC	

If a radio telegram of the sensor with the ID = 01 23 D5 E7 and ORG = 7 is received, the measuring values are described in the corresponding data bytes and the monitoring timer is set back to the value „0".

5.2 Learning-in via the Learn Button of the Radio Sensor

By means of the control command „ Write Bit(s)“ (0Fhex) one learn bit (or more) can be described with the value „1“. Thereby the receiver is set in the learn mode for one sensor selected. Within the learn mode the receiver waits for a radio telegram of a sensor, by which the learn button was actuated and writes the identification code received directly into the corresponding registers.

Example: Switch sensor 29 into learn mode (Bit 29 = 1, bit address = 28)

Master - Telegram transmission mode RTU:

Slave Address	Command	Start address		Number of Bitsr		Number of Bytes	Data	Checksum	
		H Byte	L Byte	H Byte	L Byte		H Byte	L CRC	H CRC
02	0F	00	1C	00	01	01	01	CRC	

Slave – Response telegram in transmission mode RTU:

Slave Address	Command	Start address		Number of Bits		Checksum	
		H Byte	L Byte	H Byte	L Byte	L CRC	H CRC
02	0F	00	1C	00	01	CRC	

After receipt of a radio learn telegram the learn bit is automatically deleted. Thus, it is not necessary to send a new telegram for setting back the learn bits.

6 Read Out of Data

All registers and bit values described in chapter 2.6 have read access, whereas different control commands are used for reading out the registers and bits.

6.1 Read Out of Registers

By means of the control command "Read Register" (03hex) 1 to 50 registers can be read out. If the master tries to read out more than 50 registers, the slave responds with a error telegram (error code 02hex).

Example: Read out data of sensor 29 (Register 281_{dez} (addr. = 0118_{hex}) to 290_{dez} (addr. = 0121_{hex}))

Master - Telegram in mode RTU		**Slave – Response telegram in mode RTU**			
Description	Value (Hex)	Description			Value (Hex)
Slave address	02	Slave address			02
Command	03	Command			03
Start address High	01	Number of bytes			14
Start address Low	18	Register value High	(0118)	not used	00
Number of register High	00	Register value Low	(0118)	ORG	07
Number of register Low	0A	Register value High	(0119)	ID-Byte-3	01
Checksum Low	CRC	Register value Low	(0119)	ID-Byte-2	23
Checksum High		Register value High	(011A)	ID-Byte-1	D5
		Register value Low	(011A)	ID-Byte-0	E7
		Register value High	(011B)	not used	00
		Register value Low	(011B)	Data-Byte-3	E7
		Register value High	(011C)	not used	00
		Register value Low	(011C)	Data-Byte-2	2A
		Register value High	(011D)	not used	00
		Register value Low	(011D)	Data-Byte-1	5F
		Register value High	(011E)	not used	00
		Register value Low	(011E)	Data-Byte-0	0F
		Register value High	(011F)	Receive-Time	01
		Register value Low	(011F)	Receive-Time	20
		Register value High	(0120)	not used	00
		Register value Low	(0120)	not used	00
		Register value High	(0121)	not used	00
		Register value Low	(0121)	not used	00
		Checksum Low			CRC
		Checksum High			

6.2 Read Out of Bits

By means of the control command „Read Bits“ (01_{hex}) one bit or more bits (see table 3 in chapter 2.4.3) can be read out.

Example: Read out learn bit of sensors 29-30 (29_{dez} (addr. = $0001C_{hex}$) to 30_{dez} (addr. = $0001D_{hex}$))

Master – Telegram in mode RTU		**Slave – Reply telegram in mode RTU**	
Description	Value (Hex)	Description	Value (Hex)
Device	02	Device	02
Command	01	Command	01
Start address High	00	Number of bytes	01
Start address Low	1C	Bit values 0,0,0,0,0,0,Bit29,Bit28	03
Number of Bits High	00	Checksum Low	CRC
Number of Bits Low	02	Checksum High	
Checksum Low	CRC		
Checksum High			

7 Transmission of Data

7.1 Write Register

By means of the write instruction “Write Multiple Registers(0x10)“ the registers of a transmitter can be written. Optionally each register can be written individually (instruction „Write Single Register“ (0x06)) .

Transmitter 1 (Register 401-410)

Master - Telegram in Mode RTU		**Slave – Response Telegram in Mode RTU**	
Description	Value (Hex)	Description	Value (Hex)
Slave Address	02	Slave Address	02
Command	10	Command	10
Start address high	01	Start address high	01
Start address low	90	Start address low	90
Number Register High	00	Number Register High	00
Number Register Low	0A	Number Register Low	0A
Number Bytes	14	Check sum Low	CRC
Value Register1 High	00	Check sum High	
Value Register1 Low (ORG)	07		
Value Register2 High	00		
Value Register2 Low	00		
Value Register3 High	00		
Value Register3 Low	00		
Value Register4 High	00		
Value Register4 Low (DATABYTE3)	AB		
Value Register5 High	00		
Value Register5 Low (DATABYTE2)	08		
Value Register6 High	00		
Value Register6 Low (DATABYTE1)	13		
Value Register7 High	00		
Value Register7 Low (DATABYTE0)	00		
Value Register8 High	00		
Value Register8 Low (STATUS)	00		
Value Register9 High	00		
Value Register9 Low	00		
Value Register10 High	00		
Value Register10 Low	00		
Check sum Low	CRC		
Check sum High			

7.2 Triggering of Transmissions

By means of the control command "Write Bit(s) " (0Fhex or 05hex) a transmission process can be triggered by setting one or various transmission bits with the value "1". The corresponding values of the transmission register are sent in an EasySens telegram. Afterwards, the transmission bit is automatically reset to 0 by the transceiver, i.e. no necessity to reset the same by another telegram.

Example: Send value of Sender 1 (Bit 65 = 1, i.e. data-address 64)

Master - Telegram in transmission mode RTU:

Slave Address	Command	Start address		Number of Bits		Number of Bytes	Data	Check sum	
		H Byte	L Byte	H Byte	L Byte		H Byte	L CRC	H CRC
02	0F	00	40	00	01	01	01	CRC	

Slave – response telegram in transmission mode RTU:

Slave Adresse	Command	Start address		Number of Bits		Checksum	
		H Byte	L Byte	H Byte	L Byte	L CRC	H CRC
2	0F	00	40	00	01	CRC	

7.3 EnOcean Telegram

According to the values sent, the following radio telegram is transmitted. In the example the ID of the transmitter is: 0xFFED8F00

SYNC-BYTE 1		0xA5
SYNC-BYTE 0		0x5A
H-SEQ	LENGTH	0x0B
ORG		0x07
DATA-BYTE3		0xAB
DATA-BYTE2		0x08
DATA-BYTE1		0x13
DATA-BYTE0		0x00
ID-BYTE3		0xFF
ID-BYTE2		0xED
ID-BYTE2		0x8F
ID-BYTE0		0x00
STATUS		0x00
Check sum		CS

8 Configuration Software

By means of a RS485-interface (e.g. RS232-RS485-level converter e.g. ADAM-4520) it is possible to access to the Modbus by the configuration software. The configuration software is not obligatory necessary for the installation of the SRC-RS485 resp. STC-RS485 Modbus. It is possible to use any program producing Modbus telegrams and which is suitable to learn-in sensors.

9 Software Installation

For the installation of the configuration software, the setup file „setup.exe“ must be started. Please note that you must have administrator rights for the installation. During the installation, please follow the screen instructions.

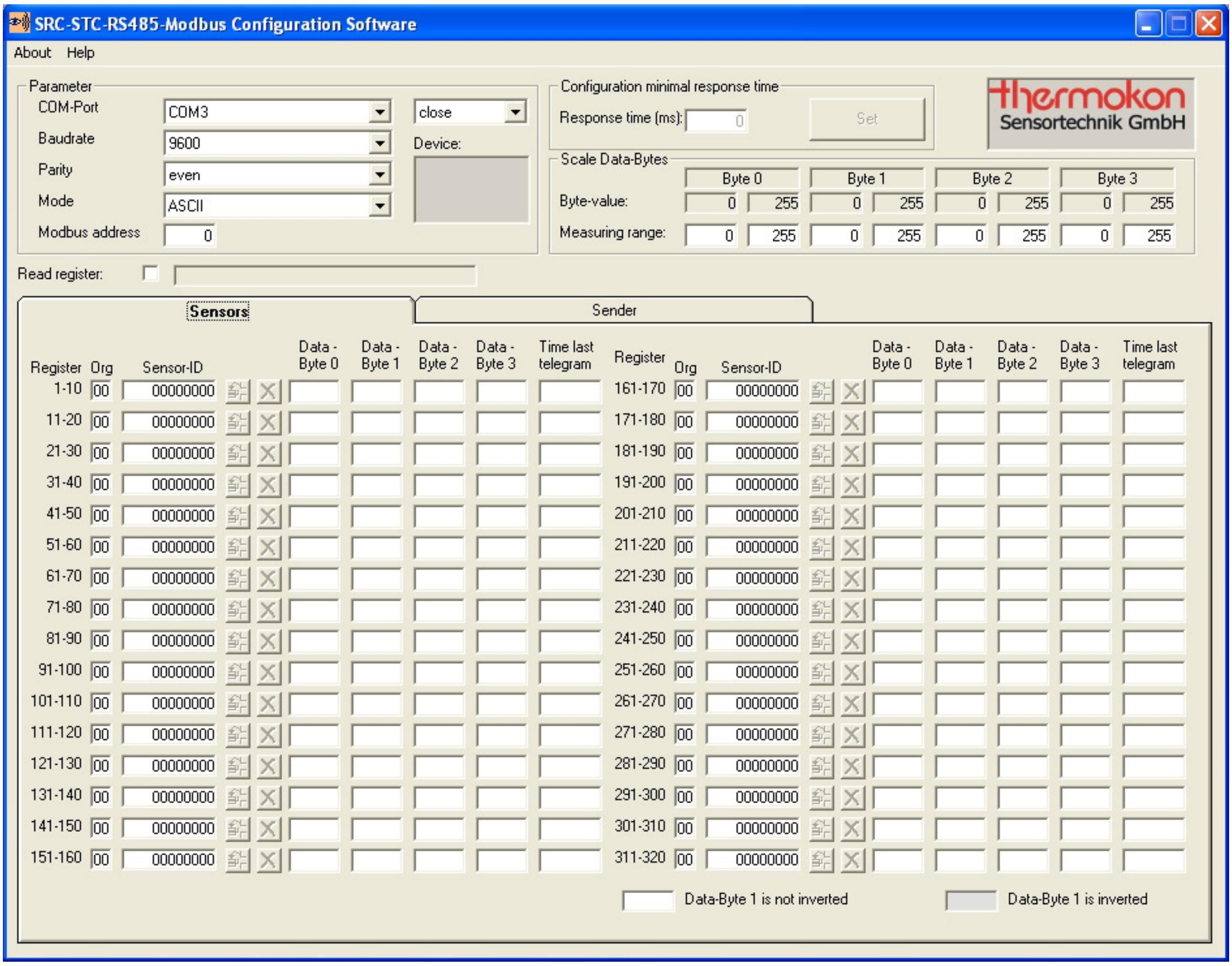
After a successful operation, the configuration software can be started via the
“Starting Menu/Programs/Thermokon“

Operating systems supported:	Windows9x; WindowsNT; WindowsMe; Windows2000; WindowsXP; WindowsServer, Windows 7

10 Configuration of Transceivers

10.1 Configuration Software

By means of the configuration software sensors can be learned-in to different registers and the transmitter channels can be verified. Register, sensor data and transmitter can be read out and scaled for display. In total, the SRC- and the STC-RS485-Modbus have 320 registers for sensors and the STC in additon 80 for transmitters. One sensor including all data takes 10 registers. Thus, the registers 1-10; 11-20 ... 311-320; stand for one sensor. The transmitter data also take 10 registers each, starting with register address 401. Sender1 takes 401-410, Sender2 411-420,…,Sender8 471-480. The load of the individual registers including the data bytes of the sensors and transmitters is described in chapter 2.6.



Picture 10-1: Configuration Software

10.2 Parameter Frame

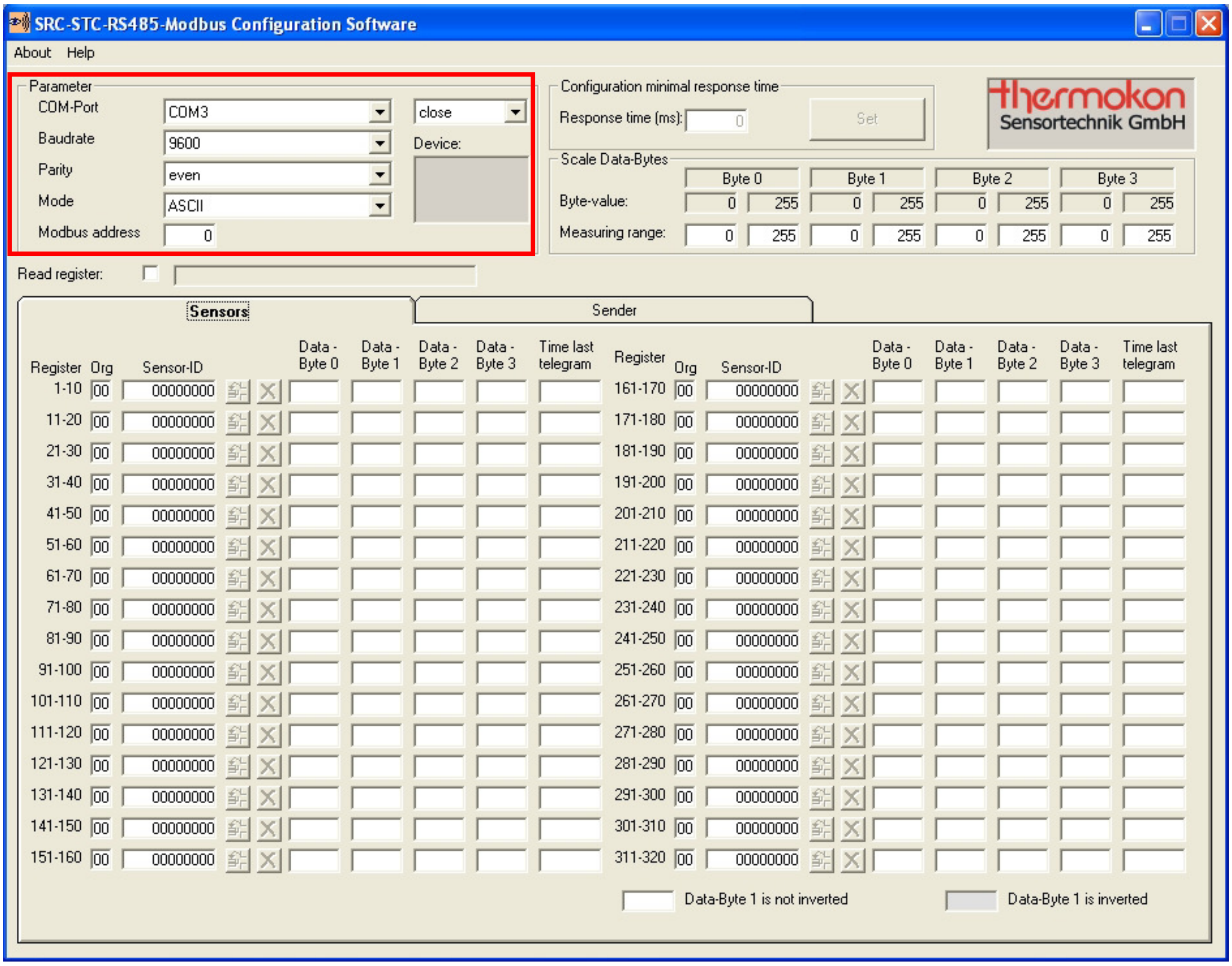
It is possible to access the Modbus by the configuration software by means of a COM-Port. In the "Parameter" frame hardware settings can be made. To build up a connection, the settings must be in accordance with the Modbus receiver.

The following options are available:

- COM-Port
- Baud rate
- Parity for setting of non-parity, even or odd parity
- Modus for setting of ASCII or RTU transmission
- Modbus address

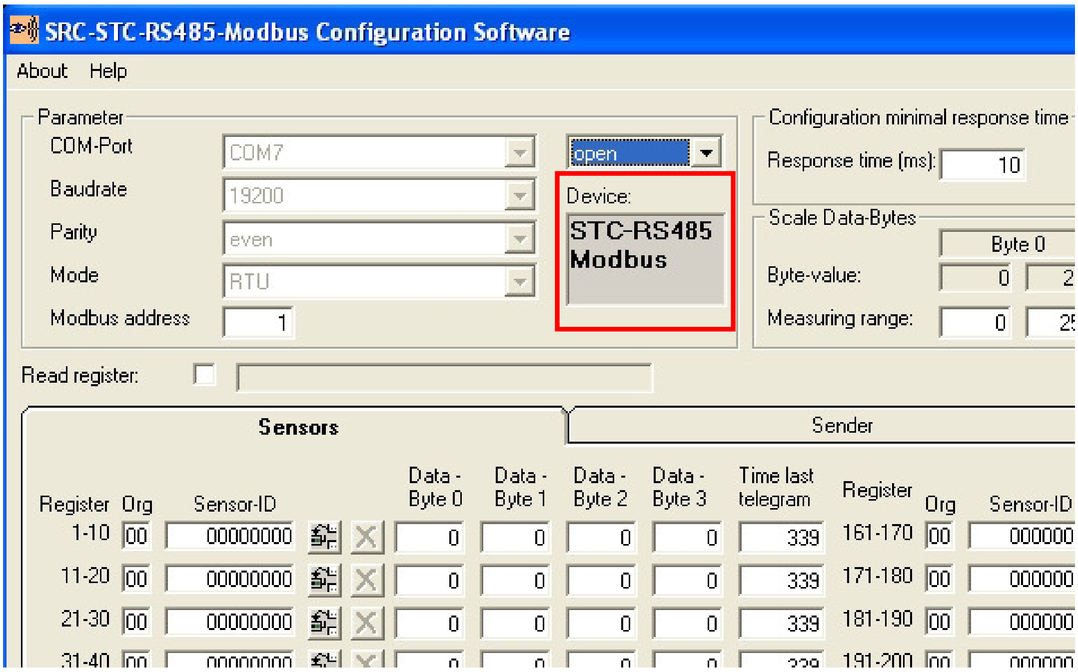
In the field „Modbus address" the address of the Modbus receiver to be configured is entered (value between 0 and 255).

Via the menu behind „COM-Port" the port can be opened „open" and closed „close".

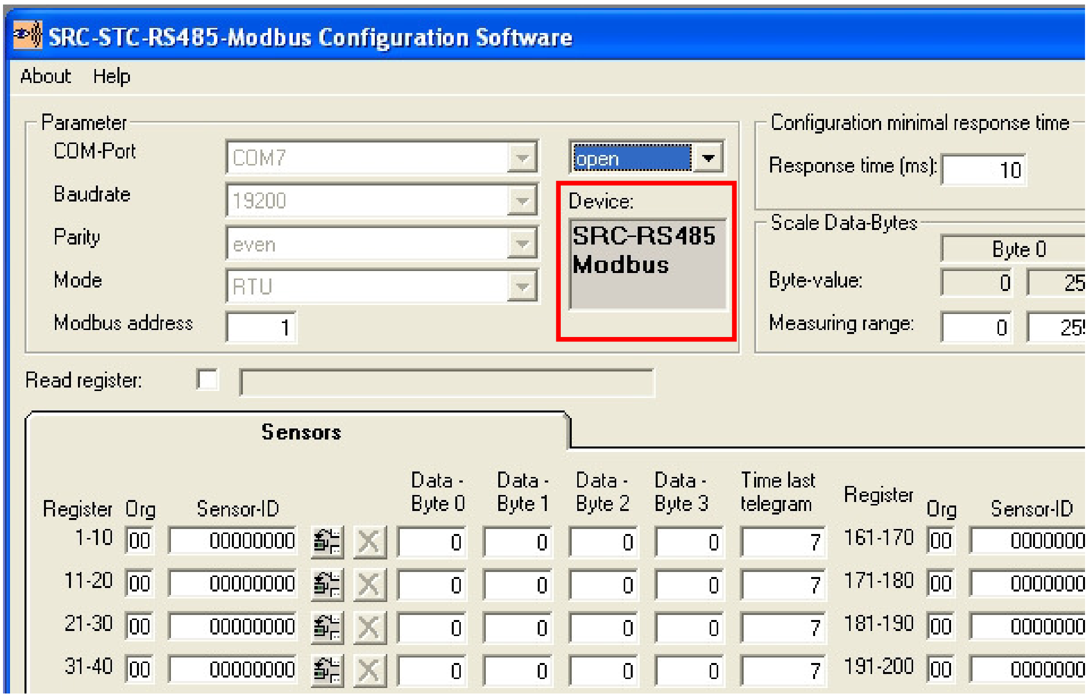


Picture 10-2: Parameter Frame

After connecting the device the device type appears. The configuration software automatically detects the connected device type.



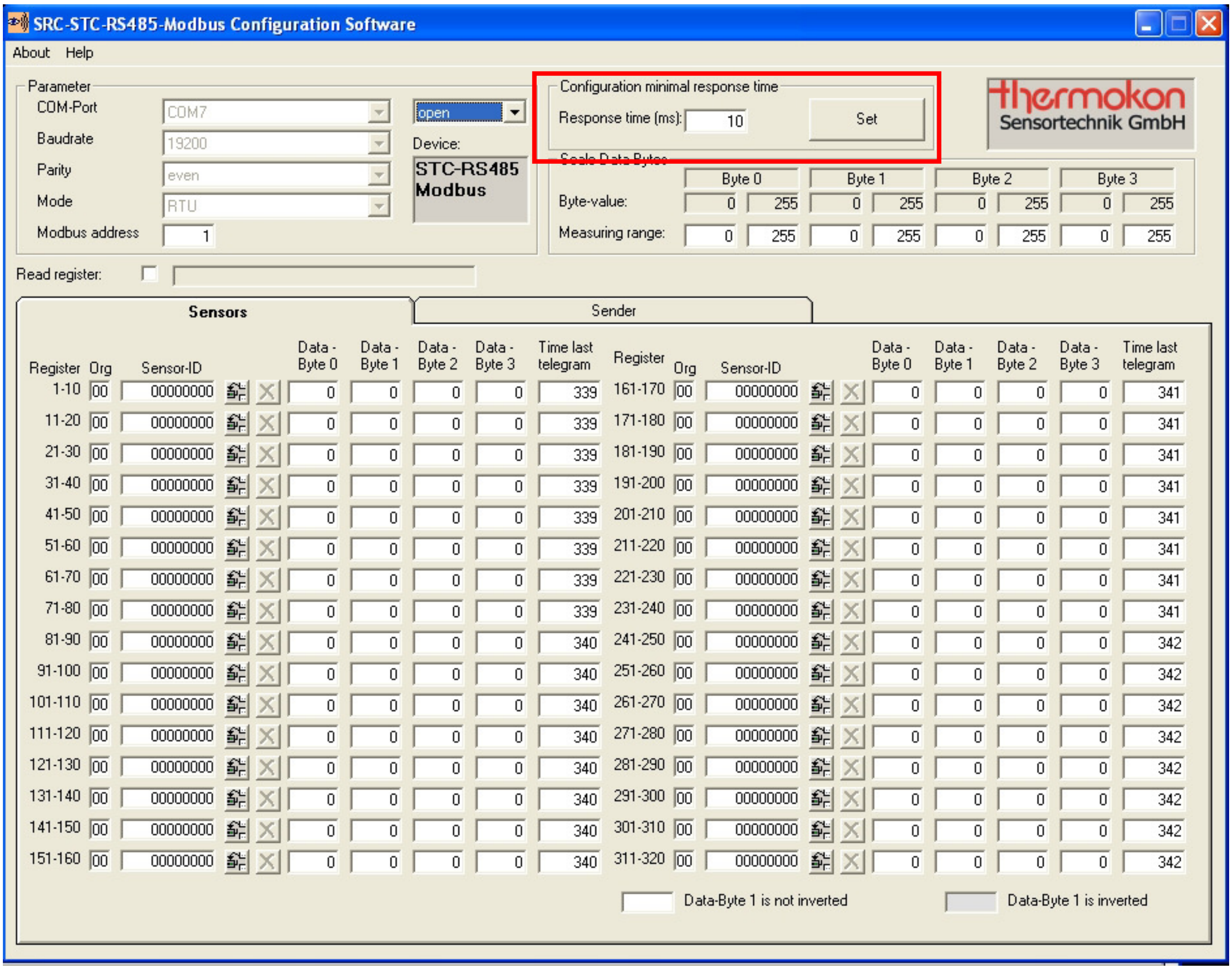
Picture 10-3: Device type STC



Picture 10-4: Device type SRC

10.3 Minimal Response Time

In the frame “configuration of minimal response time“ the register 321 can be adjusted. The response time is the minimal time (ms) that must pass before a slave is allowed to answer to a master inquiry. Preset value: 10 ms, smallest permitted value 5 ms. By means of the button “Setting“ the new settings of the minimal response time are taken over.



Picture 10-5: Minimal Response Time

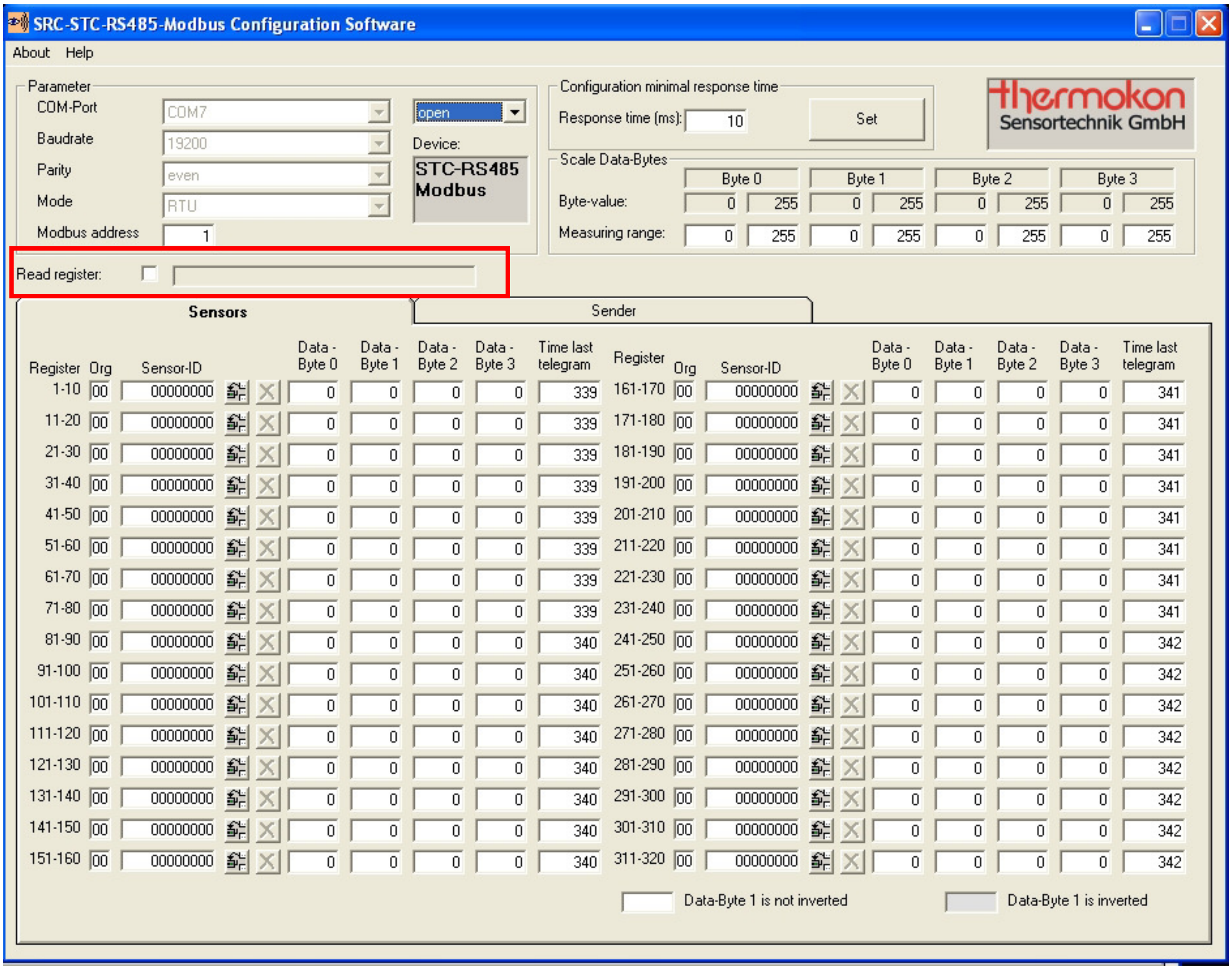
10.4 Read Out of Register

If a hook is set with “Read out of register“, all registers are read out one after the other and the data of the sensors are shown in the configuration software. If a scalling is entered, the data bytes are scalled.

In the field “Time last telegram“ the time since the last receipt of the sensor telegram is shown (in seconds).

If the temperature is inverted (data byte 1), the same is shown by a grey-coloured field. Values which are not inverted are shown in a white coloured field.

If there are communication problems, an error message is output in the field next to the field “Read out of register“.



Picture 10-6: Read out of Sensor

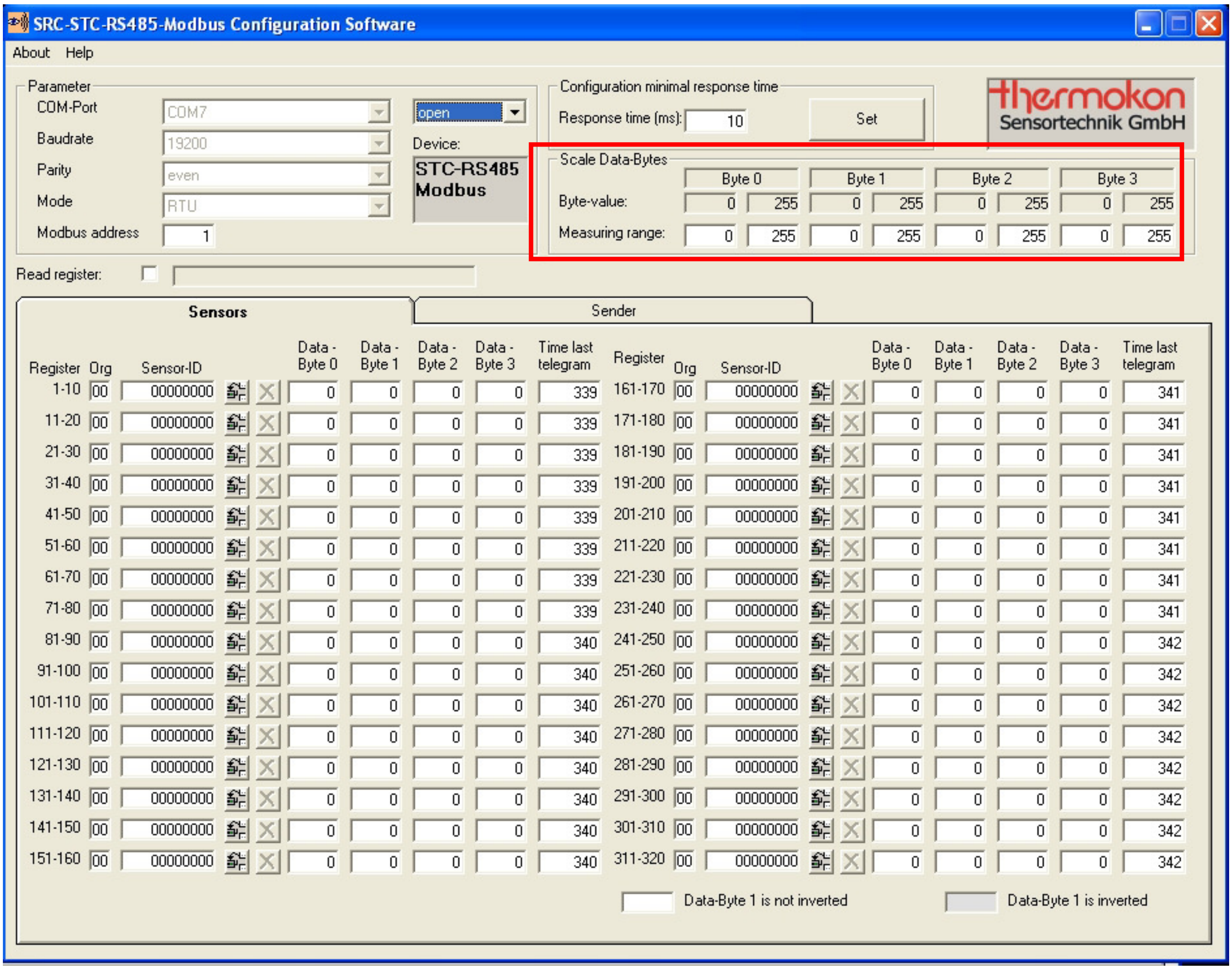
10.5 Sensor Frame

10.5.1 Scalling of Data Byte Frame

In the „Scalling Data Byte“ frame the individual data bytes of the sensors are scaled. The scalling is only designed for an easier display of the sensor data.

Example.: Scale for an outdoor temperature sensor the measuring range from –20°C to +60°C.

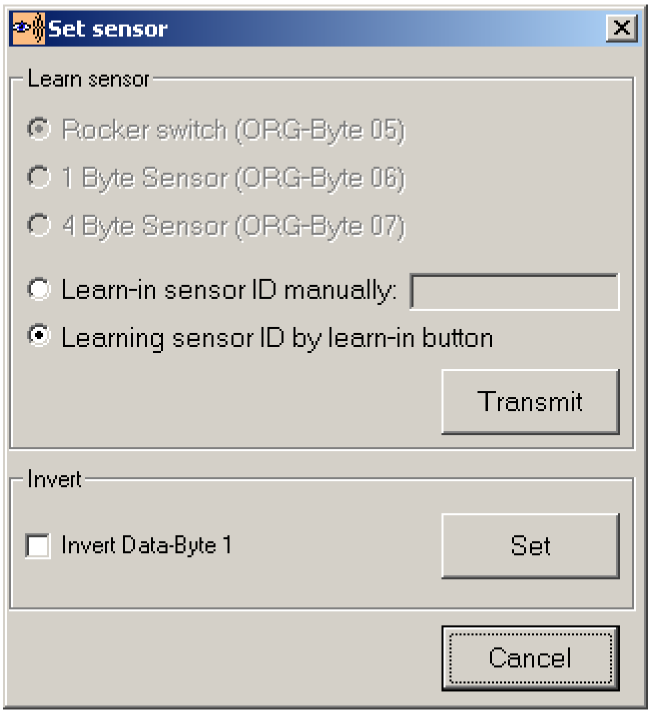
The allocation of the individual data bytes can be found in the corresponding data sheet of the sensor manufacturer.



Picture 10-7: Scalling

10.5.2 Learning-in of Sensors into the SRC/STC-RS485 Modbus

Sensors can either be connected to the receiver manually by entering the sensor ID or by pressing the learn-in button. In order to learn-in a sensor, the learn-in button in the main menu must be pressed.

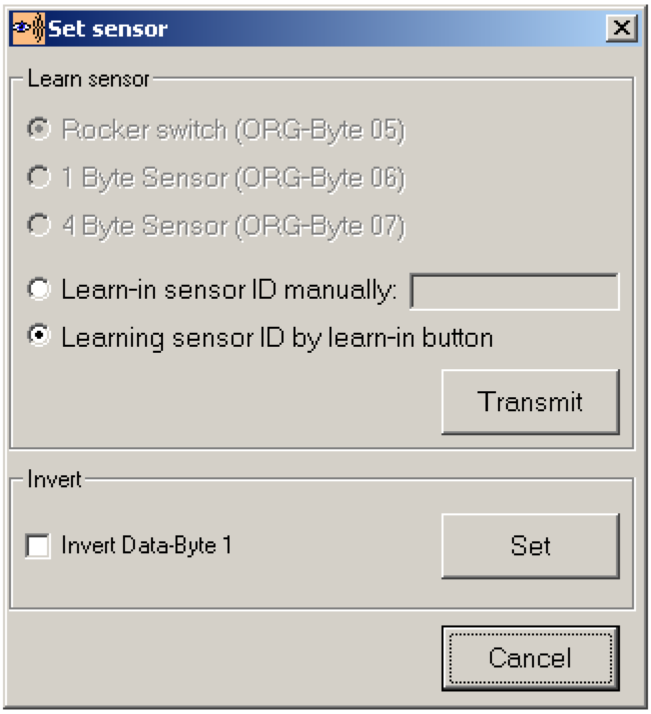


Picture 10-8: Adjusting of Sensor

- Entering of Sensor ID
    - Button (ORG-Byte 05) e.g. PTM100
    - 1 Byte Sensor (ORG-Byte 06) e.g. window contact
    - 4-Byte-Sensor (ORG-Byte 07) e.g. SR04x
    - Enter sensor ID
        - consisting of a 4-byte hexadecimal number
        - e.g. 0x00004E7A
    - By pressing the "Learn-in" button the sensor is saved in the receiver
- Entering of Sensor ID by means of "Learn-In" Button
    - By pressing the "Learn-in" button, the receiver is set into the learn mode.
    - By pressing the "Learn-in" button on the sensor or by actuating a button on the switch, the sensor/switch can be learned-in into the receiver.

10.5.3 Inverting of Temperature

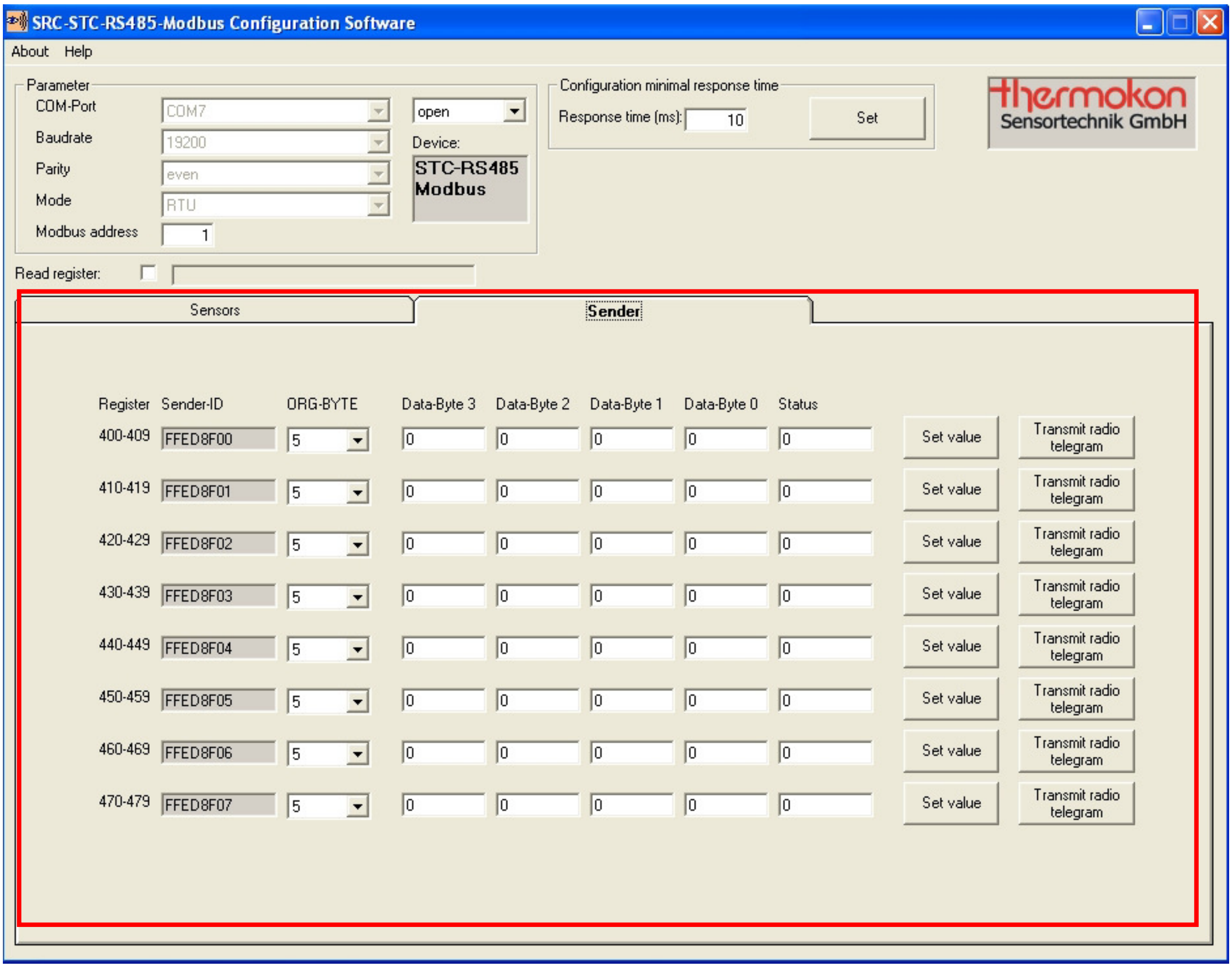
In order to invert data byte 1 (temperature) of a sensor, the learn-in button must be pressed. In the frame “Inverting“ the temperature can be inverted by activating the hook. By pressing the button “Setting“ the settings are taken over and transmitted to the receiver.



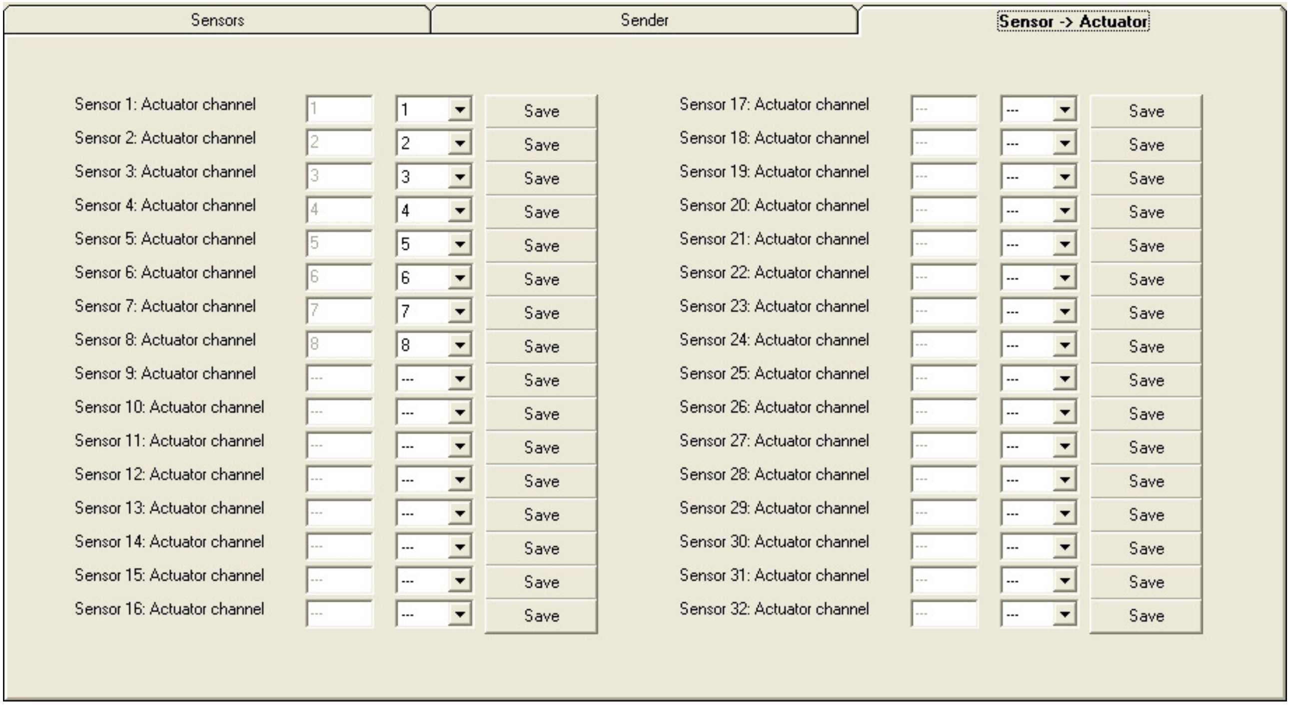
Picture 10-9: Setting of Sensor

10.6 Transmitter

In the register card “Sender“ radio telegrams can be sent via 8 available transmission channels. After a successful connection to the STC, the corresponding registers are read out first. In the fields „Sender-ID“ of the channels the corresponding identification codes are shown. These ID´s are automatically derived from the ID of the transmitting module and cannot be changed. The fields ORG-BYTE, Data-Byte3 to Data-Byte0 and Status are changeable. The valid values are lying between 0 and 225. Changed values are marked in red. By pushing the button “Take over value“ the values are written in the corresponding registers of the STC. The radio telegram is triggered by actuation of the button “Send radio telegram”.



Picture 10-10: 8 radio transmitters



Picture 10-11 Actuator to sensor assignment

11 Annex

11.1.1 Learning-in of SAB02

After having received a learning telegram of the actuator, the following sequence must be returned from the gateway (STC65) to the actuator within the period of time specified in the data sheet of the SAB02:

ORG-BYTE	=	0x07
DataByte3	=	0x80
DataByte2	=	0x08
Data Byte1	=	0x02,
Data Byte0	=	0xF0

There are 2 possibilities for triggering the above telegram.

(a) Use of Thermokon Configuration Software
1. Under the rider "Sender", the above mentioned values are entered -> "Take over value"
2. Afterwards, the learning telegram of the actuator must be triggered as described in the data sheet of the SAB01.
3. Now, the telegram of 1 must be sent by the gateway within the period of time stated in data sheet of the SAB01 -> "Send radio telegram"

(b) With a DDC or Modbus Tool on the PC
1. An actuator channel is assigned to a sensor channel. This is done via the eighth register of a sensor (see chapter 2.6.11 and 10.5) => Register 8 *(Actuator channel)* for Sensor 1, Register 18 for Sensor 2 etc. . In this register, the actuator channel (1-8) is registerd.
2.Now the values must be entered as described above into the registers by means of the DDC/Modbus tool (e.g. actuator channel 1 => Register 401 and 404-407).
3.The gateway is adjusted in that way, that a telegram with the registered value is automatically sent to the allocated actuator channel upon receipt of the sensor , i.e. there is no time problem with the generation of the sending telegram (see (a).3.).

Notice: The assignment should be removed after a successful learning-in procedure because otherwise a telegram will be returned to the actuator instantly after every telegram received on the sensor channel!