thermokon
Sensortechnik GmbH

Description

for

STC-RS485 MODBUS

Version “32RX, 32TX, 32VA”

Introduction

This document describes the Modbus↔EnOcean gateway STC-RS485 Modbus from hardware version V3.1 and firmware V3.3.0.0 (or higher).

For older versions of the STC-RS485 Modbus (or activated compatibility mode) please refer to the following document:

http://www.thermokon.de/ftp/STC65-RS485-Modbus/doc/Specification_STC65-RS485-Modbus_32RX8TX.pdf

Errata

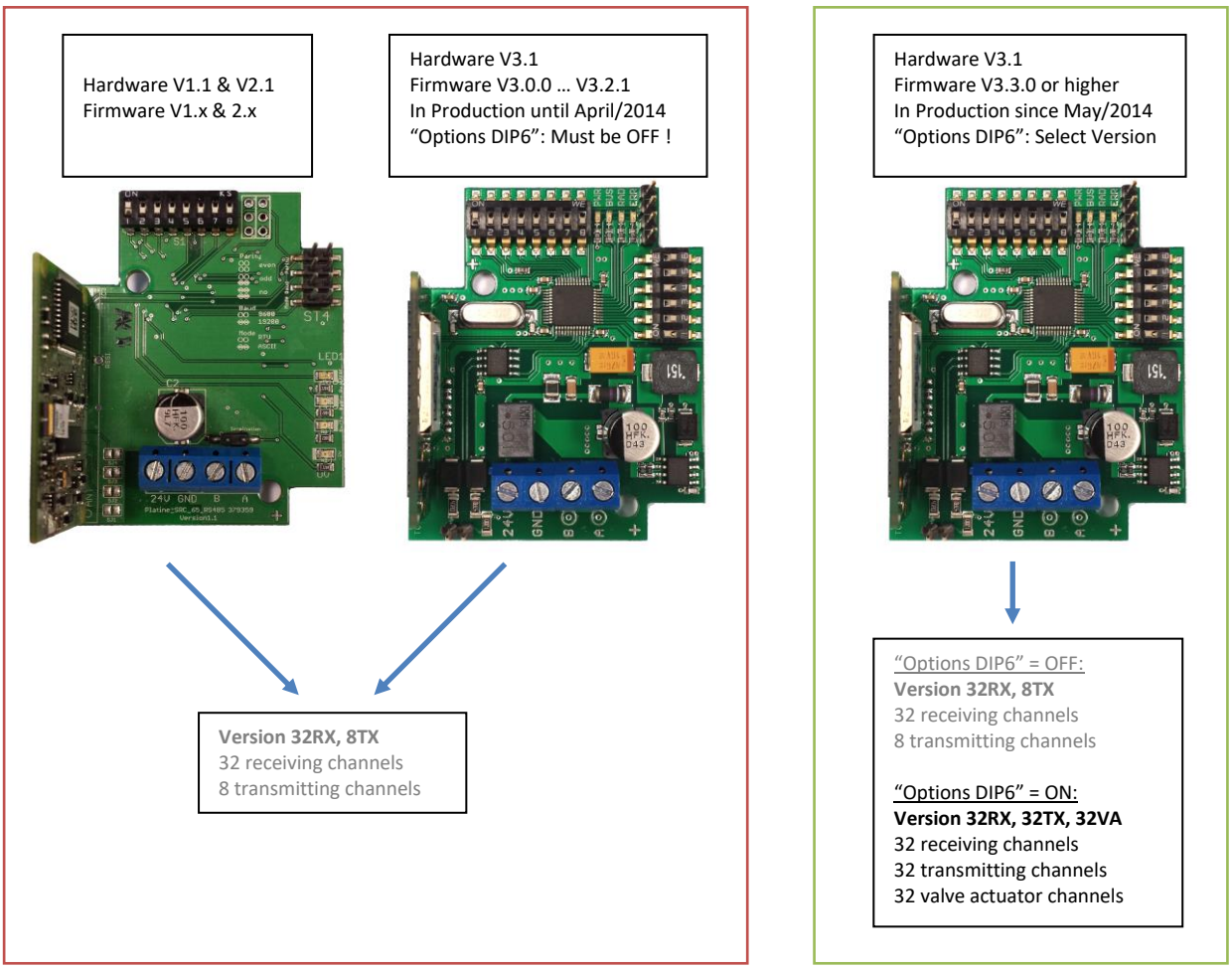
Version	Date	Description
A	28.05.2014	First Revision
B	08.12.2014	Corrected values for valve actuator (service flags for EEP A5-20-01)
C	13.07.2015	Description for SmartAck functionality added (firmware 4.0.0.0 or higher)
D	06.09.2017	Description for UTE telegrams (ORG 0xD4) added (firmware 4.1.0.0 or higher)
E	03.01.2018	• Description for SmartAck Mailbox registers added (register 15 & 16 Register of each TX channel / firmware 4.1.1.0 or higher) • Description for addressed EnOcean telegrams added (register 13 & 14 Register of each TX channel / firmware 4.2.0.0 or higher)
F	20.08.2020	Description for the current valve position (VA1…VA32) added (firmware 4.4.0.0 or higher).
G	26.07.2023	Description for firmware v4.6.0 VA channels now also support temperatue setpoint and actual value in addition to absolute valve position. This makes it possible to use the internal PI controller of the SAB0x valve actuators.
H	27.11.2023	Description for firmware v4.7.0 added - receive and send EnOcean signal telegrams.

Index

1 General

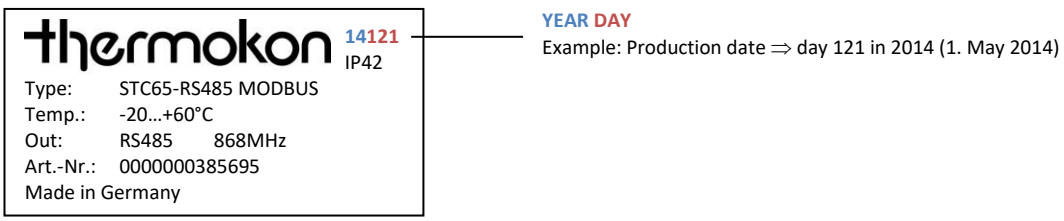
1.1 Version Overview

The following chart shows the version history of the STC65-RS485 Modbus. This document is valid for the version 32RX, 32TX, 32VA which can be selected from version “Hardware 3.1” and “Firmware 3.3.0” or higher.
For older versions of the STC-RS485 Modbus (or activated compatibility mode) please refer to the following document:
http://www.thermokon.de/ftp/STC65-RS485-Modbus/doc/Specification_STC65-RS485-Modbus_32RX8TX.pdf



1.2 Production Date

The production date is printed on the type plate.



1.3 Device Description

The unit STC-RS485 Modbus serves as a bidirectional gateway between EnOcean radio devices and the RS485 Modbus. With this device, commands and measured values of EnOcean wireless sensors can be transferred to the Modbus, for example to connect them to a programmable logic controller (PLC). Also EnOcean wireless actuators can be controlled via Modbus.

The STC-RS485 Modbus is organized as follows:

32 receiving channels (RX1…RX32) - link to EnOcean sensors:

- Push buttons (switching, dimming, shutter, scene)
- Window contacts and handles
- Access card and other switches
- Temperature, humidity and light sensors
- Presence sensors
- Gas and environmental sensors (weather station)
- Room control devices
- Automated counter devices
- Digital inputs
- …

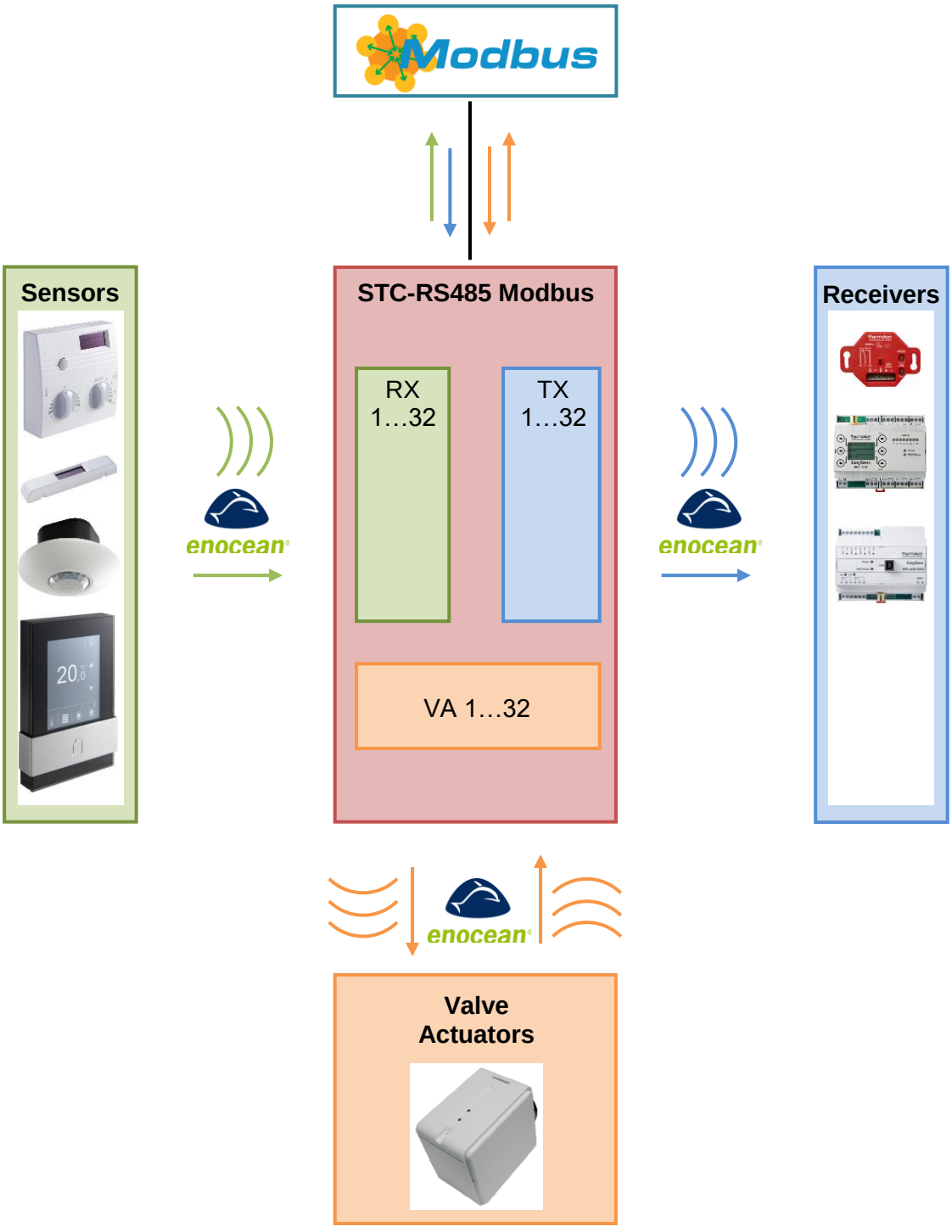
32 transmitting channels (TX1…TX32) - link to EnOcean receivers:

- Switch actuators
- Light actuators
- Dimmer actuators
- Shutter actuators
- Heating & cooling controllers
- FanCoil controllers
- …

32 Valve Actuator channels (VA1…VA32) - link to EnOcean wireless valve actuators:

- EnOcean wireless valve actuator with EnOcean Profile A5-20-01,
  i.e. Thermokon SAB02

The configuration of the device and the channels can be done by using the Thermokon STC-RS485 Modbus configuration software (http://www.thermokon.de/ftp/STC65-RS485-Modbus/stc-rs485_modbus_setup.exe) or directly through the Modbus register by a PLC.


Modbus
Sensors
STC-RS485 Modbus
RX
1…32
TX
1…32
VA 1…32
Receivers
enocean
enocean
enocean
Valve
Actuators

1.4 Hardware Installation

The STC65-RS485 Modbus gateway 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 STC-RS485-Modbus and the data sheet wiring_rs485_network.pdf.

1.5 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.

1.6 Protocol

The gateway modules STC65-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:

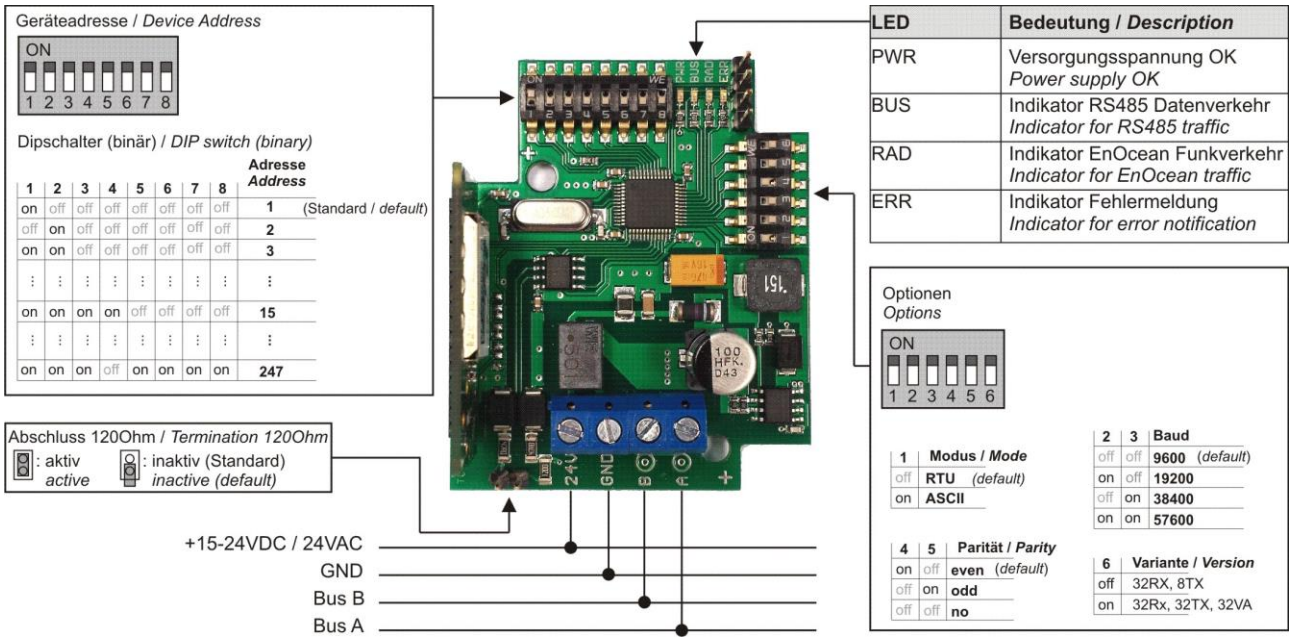
- MODBUS Application Protocol Specification V1.1
- MODBUS via Serial Line Specification & Implementation guide V1.0

1.7 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 ASCII or RTU
- baud rate 9600, 19200, 38400 or 57600
- even parity, odd parity or no parity

1.8 Hardware Overview



DIP6 of the "Options" dip-switch must be switched ON (32RX, 32TX, 32VA mode), to use the functionality and register mapping which is described in this document!

If DIP6 is switched OFF, the STC-RS485 Modbus works in compatibility mode (32RX, 8TX).
In this case, please refer to the following document:
http://www.thermokon.de/ftp/STC65-RS485-Modbus/doc/Spezifikation_STC65-RS485-Modbus_32RX8TX.pdf

1.9 LED Error Codes

LED PWR	LED BUS	LED RAD	LED ERR	Description
flashing	flashing	x	flashing	Wrong Modbus communication. Please check parity, baud rate, mode, …
off	on	off	on	Wrong Modbus settings. Please check address, mode, parity, …
off	off	on	on	No communication to EnOcean RF module.

2 Modbus Register Mapping

2.1 General Registers (Holding Registers)

Modbus Holding Registers	
Register Data Address	**Description**
0 R	#Firmware Version e.g.: 0x3300 = Version 3.3.0.0
1 R	#Device Coding 0x0500
2 R/W	Operating Time since last reset [seconds] HIGH
3 R/W	Operating Time since last reset [seconds] LOW
4 R/W	Receiving Channels Telegram Counter (all IDs) HIGH
5 R/W	Receiving Channels Telegram Counter (all IDs) LOW
6 R/W	Receiving Channels Telegram Counter (stored IDs only) HIGH
7 R/W	Receiving Channels Telegram Counter (stored IDs only) LOW
8 R/W*	#EnOcean Base-ID-Byte-3
9 R/W*	#EnOcean Base-ID-Byte-2
10 R/W*	#EnOcean Base-ID-Byte-1
11 R/W*	#EnOcean Base-ID-Byte-0
12 R	*reserved*
13 R	*reserved*
14 R	*reserved*
15 R/W	#Repeater function (0=OFF, 1=1-Level, 2=2-Level) Sets the integrated EnOcean repeater to off, 1-level or 2-level.
16 R/W	#Min. Response Time
17 R/W	#Reset RX Coil necessary (factory setting: 0)
18 R/W	Transmitting Channels Telegram Counter (all telegrams)
19 R/W	Transmitting Channels Telegram Counter (by coil triggered telegrams only)

Table 1
Modbus Holding Registers 0…19

The registers marked by a rhomb (#) are configuration properties which keep their values even after a restart of the device.

Firmware

Read out the firmware Version of the STC-RS485 Modbus.

Device Coding

Coding to identify the STC-RS485 Modbus in the Modbus network.

Operating Time

Operating time counter – counts up every second.

Telegram Counter (all IDs)

Counts up on every EnOcean telegram received.

Telegram Counter (stored IDs only)

Counts up on every EnOcean telegram of a known (ID stored in one of the 32 RX channels) EnOcean sender.

Repeater function

The STC-RS485 Modbus provides the option to activate a one or two-level repeater for EnOcean radio telegrams.

1-level repeater: If a telegram received is a valid and an original one (not yet repeated), the telegram is repeated after a random delay.

2-level repeater: If a telegram received is valid and an original one or repeated once, the telegram is repeated after a random delay.

2-level repeating function should only be activated if really needed! Otherwise the system function can be compromised by collisions of telegrams.
The repeated telegram is marked as “repeated” by an increased repeater counter.

Min. Response Time (for ASCII Mode only)

Delay time before the serial transmitter of the STC65 is enabled. This is required because the STC65 is so fast that there is no time between receiving and transmitting the frame. If the master is too slow with enabling its receiver then he will not receive the response correctly.

Reset RX Coil necessary

If Register 17 “Reset RX Coil necessary” is set to >0:
After you have read out the values of a RX channel, the corresponding monitoring coil must be reset because this is the way how the STC65 detects that the receiving channel was read by the Modbus master!

2.1.1 EnOcean Telegram Buffer

Each EnOcean receiver channel (RX channel) stores up to 6 EnOcean telegrams in a FIFO memory which can be read out via Modbus. In case that 6 telegrams are stored (which were not read via Modbus) and a further telegram will be received, the oldest telegram will be overwritten.
By each reading of a receiver channel, one place in the EnOcean telegram buffer will become free.
If all telegrams were read out the Modbus registers contain the last received telegram. The Modbus registers 20, 22, 24, ..., 82 contain the number of stored new messages for each receiver channel.
The monitoring Coils 100 ... 131 indicate that at least one new EnOcean telegram is available at this channel. If all telegrams have been read out the corresponding coil switches automatically to 0.

If Register 17 "Reset RX Coil necessary" is set to >0:
After you have read out the values of a RX channel, the corresponding monitoring coil must be reset because this is the way how the STC65 detects that the receiving channel was read by the Modbus master!

2.2 EnOcean Receiving Information Registers (Input Registers)

Modbus Input Registers																	
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	
20 R	Number of available EnOcean Telegrams at channel RX1																
21 R	RSSI Level of the last received EnOcean telegram at channel RX1																[-dbm]
…																	
82 R	Number of available EnOcean Telegrams at channel RX32																
83 R	RSSI Level of the last received EnOcean telegram at channel RX32																[-dbm]
84 R	*reserved*																
85 R	RSSI Level of the last received EnOcean telegram at channel VA1																[-dbm]
…																	
146 R	*reserved*																
147 R	RSSI Level of the last received EnOcean telegram at channel VA32																[-dbm]

Table 2
Modbus Input Registers 20…147

2.3 EnOcean Receiving Data Registers (RX1…RX32)

Modbus Holding Registers																	
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	
200 R/W									#FUNCTION								Data Sensor 1 RX1
201 R/W									#TYPE								
202 R/W									#MANUFACTURER								
203 R/W	#ID-Byte-3								#ID-Byte-2								
204 R/W	#ID-Byte-1								#ID-Byte-0								
205 R/W									#Response Channel								
206 R/W	Receive-Time-Byte																
207 R/W									#ORG								
208 R									Status								
209 R									Number of DataBytes **								
													Data-Byte 0 *				
210 R	Data-Byte 1 **								Data-Byte 0 **								
					reserved								Data-Byte 1 *				
211 R	Data-Byte 3 **								Data-Byte 2 **								
					reserved								Data-Byte 2 *				
212 R	Data-Byte 5 **								Data-Byte 4 **								
					reserved								Data-Byte 3 *				
213 R	Data-Byte 7 **								Data-Byte 6 **								
					reserved								*reserved*				
214 R	Data-Byte 9 **								Data-Byte 8 **								
					reserved								*reserved*				
215 R	Data-Byte 11 **								Data-Byte 10 **								
					reserved								*reserved*				
216 R	Data-Byte 13 **								Data-Byte 12 **								
					reserved								*reserved*				
217 R									SIGNAL-Telegram received								
218 R	SIGNAL-Telegram Data-Byte 1								SIGNAL-Telegram Data-Byte 0								
219 R	*reserved*								*reserved*								
…																	
820 R/W									#FUNCTION								Data Sensor 32 RX32
821 R/W									#TYPE								
822 R/W									#MANUFACTURER								
823 R/W	#ID-Byte-3								#ID-Byte-2								
824 R/W	#ID-Byte-1								#ID-Byte-0								
825 R/W									#Response Channel								
826 R/W	Receive-Time-Byte-1								Receive-Time-Byte-0								
827 R/W									#ORG								
828 R									Status								
829 R									Number of DataBytes **								
													Data-Byte 0 *				
830 R	Data-Byte 1 **								Data-Byte 0 **								
					reserved								Data-Byte 1 *				
831 R	Data-Byte 3 **								Data-Byte 2 **								
					reserved								Data-Byte 2 *				
832 R	Data-Byte 5 **								Data-Byte 4 **								
					reserved								Data-Byte 3 *				
833 R	Data-Byte 7 **								Data-Byte 6 **								
					reserved								*reserved*				
834 R	Data-Byte 9 **								Data-Byte 8 **								
					reserved								*reserved*				
835 R	Data-Byte 11 **								Data-Byte 10 **								
					reserved								*reserved*				

836 R	Data-Byte 13 **	*reserved*	Data-Byte 12 **	*reserved*	
837 R			SIGNAL-Telegram received		
838 R	SIGNAL-Telegram Data-Byte 1		SIGNAL-Telegram Data-Byte 0		
839 R	*reserved*		*reserved*		

Table 3
Modbus Holding Registers 200…839

***Blue** marked descriptions are for ORG F6 (RPS), D5 (1BS) and A5 (4BS) EnOcean telegrams only!
****Red** marked descriptions are for ORG D2 (VLD) and D1 (MSC) EnOcean telegrams only!

The registers marked by a rhomb (#) are configuration properties which keep their values even after a restart of the device.

2.3.1 Sensor Identification

The registers FUNCTION, TYPE, MANUFACTURER, ID-ByteX and ORG identify each sensor clearly (see EnOcean "Standardization_EnOcean_Communication_Profiles" for further information).

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

2.3.2 Response Channel

A value within the range 1… 32 causes that with the telegram receipt of a sensor telegram the data of the corresponding transmit channel is automatically sent.

2.3.3 Receive Time

The respective register "receive time" shows the time (in seconds) that passed by since the last radio telegram of the sensor was received.

2.3.4 SIGNAL-Telegram

The receipt of an EnOcean signal telegram is signaled via this register.
0: No signal telegram has been received since the register was last read out.
1: At least 1 signal telegram has been received since the register was last read out.

Note: The content of this register is automatically reset to 0 after it has been read out.

2.3.5 SIGNAL-Telegram Data

This register contains the data of the last SIGNAL telegram received. Only SIGNAL telegrams with a maximum of 2 data bytes are fully supported.

2.3.6 Data-Bytes Interpretation

EnOcean telegrams of type D5 (1BS) and A5 (4BS) are transferred 1:1 (raw) to the respective Modbus registers without interpretation.

Telegrams of type F6 (RPS) will be interpreted as follows:

2.3.6.1 EnOcean switch

Data-Byte 0

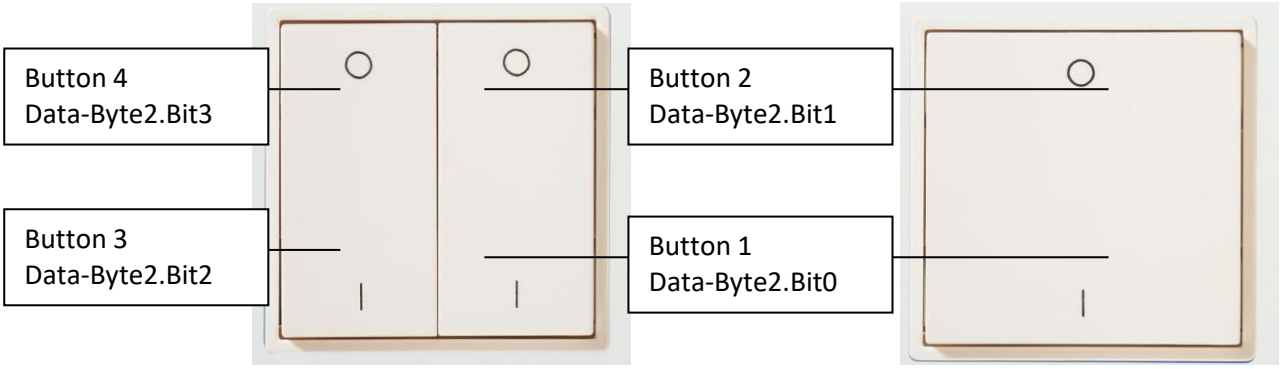
- Current status of button (raw values)
- Pressed buttons are not buffered.

Data-Byte 1

- Current status of button (raw values)
- Button function – status changes of the button are stored in the device till the next Modbus inquiry is received and are sent by then.

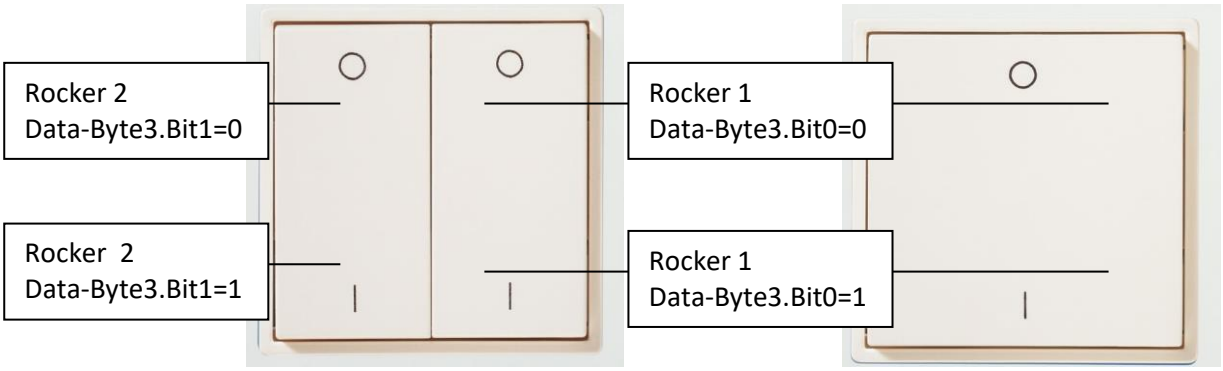
Data-Byte 2

- Current status of keys
- Button function
- All status changes of the key are stored in the device till the next Modbus inquiry is received and are sent by 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 3

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



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

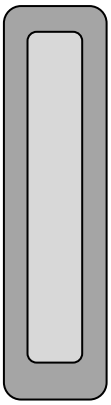
2.3.6.2 EnOcean Window Handle

Data-Byte 0

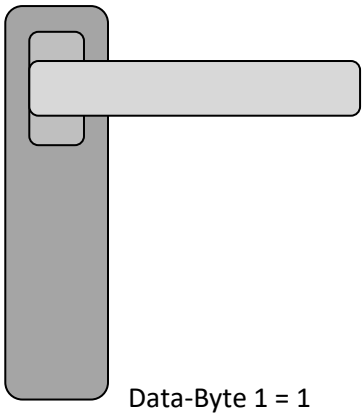
- Current status of the window handle (raw).

Data-Byte 1

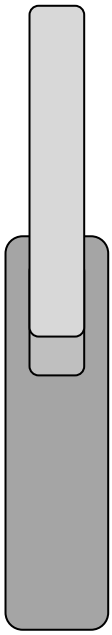
- Current status the window handle.



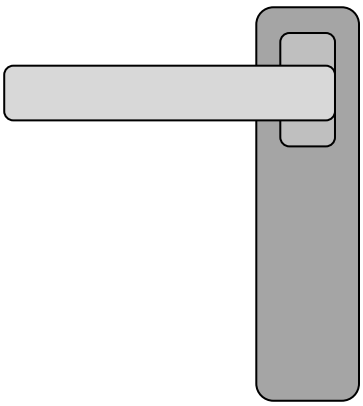
Data-Byte 1 = 0



Data-Byte 1 = 1



Data-Byte 1 = 2



Data-Byte 1 = 1

2.4 EnOcean Sender Data Registers (TX1…TX32)

Modbus Holding Registers																	
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	
1000 R	#ID-Byte-3								#ID-Byte-2								TX1
1001 R	#ID-Byte-1								#ID-Byte-0								
1002 R/W	*reserved*								Learnchannel								
1003 R/W	*reserved*								ORG								
1004 R/W	*reserved*								STATUS								
1005 R/W	*reserved*				*reserved*				Number of DataBytes **				Data-Byte 0 *				
1006 R/W	Data-Byte 1 **				*reserved*				Data-Byte 0 **				Data-Byte 1 *				
1007 R/W	Data-Byte 3 **				*reserved*				Data-Byte 2 **				Data-Byte 2 *				
1008 R/W	Data-Byte 5 **				*reserved*				Data-Byte 4 **				Data-Byte 3 *				
1009 R/W	Data-Byte 7 **				*reserved*				Data-Byte 6 **				Data-Byte 4 *				
1010 R/W	Data-Byte 9 **				*reserved*				Data-Byte 8 **				Data-Byte 5 *				
1011 R/W	Data-Byte 11 **				*reserved*				Data-Byte 10 **				Data-Byte 6 *				
1012 R/W	Data-Byte 13 **				*reserved*				Data-Byte 12 **				*reserved*				
1013 R/W	Destination-ID-Byte-3								Destination-ID-Byte-2								
1014 R/W	Destination-ID-Byte-1								Destination-ID-Byte-0								
1015 R/W	*reserved*								SMACK-Mailbox refresh procedure								
1016 R/W	*reserved*								SMACK-Mailbox refresh								
1017 R/W	*reserved*				*reserved*				*reserved*				*reserved*				
1018 R/W	*reserved*				*reserved*				*reserved*				*reserved*				
1019 R/W	*reserved*				*reserved*				*reserved*				*reserved*				
…																	
1620 R	#ID-Byte-3								#ID-Byte-2								TX32
1621 R	#ID-Byte-1								#ID-Byte-0								
1622 R/W	*reserved*								Learnchannel								
1623 R/W	*reserved*								ORG								
1624 R/W	*reserved*								STATUS								
1625 R/W	*reserved*				*reserved*				Number of DataBytes **				Data-Byte 0 *				
1626 R/W	Data-Byte 1 **				*reserved*				Data-Byte 0 **				Data-Byte 1 *				
1627 R/W	Data-Byte 3 **				*reserved*				Data-Byte 2 **				Data-Byte 2 *				
1628 R/W	Data-Byte 5 **				*reserved*				Data-Byte 4 **				Data-Byte 3 *				
1629 R/W	Data-Byte 7 **				*reserved*				Data-Byte 6 **				Data-Byte 4 *				
1630 R/W	Data-Byte 9 **				*reserved*				Data-Byte 8 **				Data-Byte 5 *				
1631 R/W	Data-Byte 11 **				*reserved*				Data-Byte 10 **				Data-Byte 6 *				
1632 R/W	Data-Byte 13 **				*reserved*				Data-Byte 12 **				*reserved*				
1633 R/W	Destination-ID-Byte-3								Destination-ID-Byte-2								
1634 R/W	Destination-ID-Byte-1								Destination-ID-Byte-0								
1635 R/W	*reserved*								SMACK-Mailbox refresh procedure								
1636 R/W	*reserved*				*reserved*				SMACK-Mailbox refresh								

1637 R/W	*reserved*	*reserved*	*reserved*	*reserved*	
1638 R/W	*reserved*	*reserved*	*reserved*	*reserved*	
1639 R/W	*reserved*	*reserved*	*reserved*	*reserved*	

Table 4
Modbus Holding Registers 1000…1639

***Blue** marked descriptions are for ORG F6 (RPS), D5 (1BS), A5 (4BS) and UTE (D4) EnOcean telegrams only!
****Red** marked descriptions are for ORG D0 (SIG), D1 (MSC) and D2 (VLD) EnOcean telegrams only!

The registers marked by a rhomb (#) are configuration properties which keep their values even after a restart of the device.

2.4.1 Identification Code

The first 2 registers hold the EnOcean ID of the respective transmitting channel.
Each channel has an own EnOcean ID build as: BaseID + Index Transmitting channel

2.4.2 ORG-Byte, Status-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“ have read and write access.

2.4.3 Learn Channel

A value within the range 1… 32 causes that by sending a telegram from the transmitting channel the corresponding receiving channel is automatically set into learn mode.

2.4.4 Destination-ID-Byte 0 … 3

If an addressed EnOcean telegram has to be sent, the EnOcean destination address can be entered in these registers.

2.4.5 SMACK-Mailbox refresh procedure

0: The SmartAck mailbox is updated cyclically.
1: The mailbox is only updated if values have changed.

2.4.6 SMACK-Mailbox refresh

1: Force the SmartAck mailbox to update.

2.5 EnOcean Valve Actuator Data Registers (VA1…VA32)

(Supports EnOcean Profiles A5-20-01, i.e. Thermokon SAB0x, only!)

Modbus Holding Registers																	
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	
2000 R/W									#ORG								VA1
2001 R/W									#FUNC								
2002 R/W									#TYPE								
2003 R/W									#MANUFACTURER ID								
2004 R/W	#ID-Byte 3								#ID-Byte 2								
2005 R/W	#ID-Byte 1								#ID-Byte 0								
2006 R/W	Receive-Time-Byte-1								Receive-Time-Byte-0								
2007 R/W									Set valve position (0…100 means 0...100%) 0% = valve completely closed … 50% = valve half opened … 100% = valve completely opened								
2008 R/W									Set service flags 0: No service 9: Set summer mode 17: Close valve 33: Open valve 65: Lift set 129: Run init sequence								
2009 R									Read status flags Bit 5…7: not used Bit 4: 0=battery bad 1=battery ok Bit 2…3: not used Bit 1: 0=“window closed” detected 1=“window open” detected Bit 0: 0=No error 1=Error detected								
2010 R	Read the value of the valve-actuator built-in temperature sensor (0…400 means 0…40,0°C)																
2011 R									Valve position feedback (current value) 0…100%								
2012 R/W									Operation Mode 0: Valve position (Register 2007) 1: Temperature setpoint & actual temperature (Register 2013 and 2014)								

2013 R/W	Temperature Setpoint 0…400 means 0…40,0°C		
2014 R/W	Actual Temperature Value 0…400 means 0…40,0°C		
2015 R		*reserved*	
2016 R		*reserved*	
2017 R		*reserved*	
2018 R		*reserved*	
2019 R		*reserved*	
…			
2620 R/W		#ORG	VA32
2621 R/W		#FUNC	
2622 R/W		#TYPE	
2623 R/W		#MANUFACTURER ID	
2624 R/W	#ID-Byte 3	#ID-Byte 2	
2625 R/W	#ID-Byte 1	#ID-Byte 0	
2626 R/W	Receive-Time-Byte-1	Receive-Time-Byte-0	
2627 R/W		Set valve position (0…100 means 0...100%) 0% = valve completely closed … 50% = valve half opened … 100% = valve completely opened	
2628 R/W		Set service flags 0: No service 9: Set summer mode 17: Close valve 33: Open valve 65: Lift set 129: Run init sequence	
2629 R		Read status flags Bit 5…7: not used Bit 4: 0=battery bad 1=battery ok Bit 2…3: not used Bit 1: 0=“window closed” detected 1=“window open” detected Bit 0: 0=No error 1=Error detected	
2630 R	Read the value of the valve-actuator built-in temperature sensor (0…400 means 0…40,0°C)		
2631 R		Valve position feedback (current value) 0…100%	
2632 R/W		Operation Mode 0: Valve position (Register 2627) 1: Temperature setpoint & actual temperature (Register 2633 and 2634)	

2633 R/W	Temperature Setpoint 0…400 means 0…40,0°C		
2634 R/W	Actual Temperature Value 0…400 means 0…40,0°C		
2635 R	*reserved*	*reserved*	
2636 R	*reserved*	*reserved*	
2637 R	*reserved*	*reserved*	
2638 R	*reserved*	*reserved*	
2639 R	*reserved*	*reserved*	

Table 5
Modbus Holding Registers 2000…2639

The registers marked by a rhomb (#) are configuration properties which keep their values even after a restart of the device.

2.6 Set Learn Mode for Receiving Channels (RX1…RX32)

The bit values listed in the table are marked by „R/W“ and have read and write access. If for example Coil 0 is described by the value „1“ the learn mode for receiving channel 1 (RX 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.

Modbus Coils	
Coil Data Address	**Description** 0: learn-mode disabled 1: learn-mode enabled
0 R/W	Learn mode receiving channel RX 1
1 R/W	Learn mode receiving channel RX 2
…	
31 R/W	Learn mode receiving channel RX 32

Table 6
Modbus Coils 0…31

2.7 Set Learn Mode for Valve Actuator Channels (VA1…VA32)

The bit values listed in the table are marked by „R/W“ and have read and write access. If for example coil 32 is described by the value „1“ the learn mode for valve actuator channel 1 (VA 1) is activated.
In the learn mode the receiver waits for a learn-in radio telegram of a valve actuator 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.

Modbus Coils	
Data Address	**Description** 0: learn-mode disabled 1: learn-mode enabled
32 R/W	Learn mode Valve Actuator 1
33 R/W	Learn mode Valve Actuator 2
…	
63 R/W	Learn mode Valve Actuator 32

Table 7
Modbus Coils 32…63

2.8 Monitoring Coils

These coils signalize if new telegrams are available on the respective receiving channel.

Modbus Coils	
Data Address	**Description** 0: no new telegram 1: at least one new telegram is available
100 R/W	New Telegram Sensor at RX 1
101 R/W	New Telegram Sensor at RX 2
…	
131 R/W	New Telegram Sensor at RX 32

Table 8
Modbus Coils 100…131

If Holding Register 17 "Reset RX Coil necessary" is set to >0:
After you have read out the values of a RX channel, the corresponding monitoring coil must be reset because this is the way how the STC65 detects that the receiving channel was read by the Modbus master!

2.9 Send Telegram Coils

The bit values listed in the table are marked by "R/W" and have a read and write access. For sending an EnOcean telegram by one one the transmitting channels TX1…TX32, the Coil must be set. After a successful transmission, the Coil is automatically reset to 0.

Modbus Coils	
Coil Data Address	**Description** 1: send out the values of the respective transmitting channel TX1…TX32
200 R/W	Transmission Bit Sender 1
201 R/W	Transmission Bit Sender 2
…	
231 R/W	Transmission Bit Sender 32

Table 9
Modbus Coils 200…231

3 SmartAck

Starting Firmware Version 4.0.0 STC65-RS485 Modbus Gateway supports the Smart Acknowledge procedure for the bidirectional EnOcean communication.

To use SmartAck with STC65 the following must be noted:

- SmartAck can only be used for the first 15 receiving channels (RX1…RX15).
- If a SmartAck device is learned-in to a receiving channel, the corresponding TX transmitting channel is automatically configured as the “response channel” (.e.g.: SmartAck device on RX5 → TX5 is the response channel).

  Notice: SmartAck response telegrams are sent as addressed telegrams under the ChipID of the EnOcen module in the STC65. Thus, the IDs saved in the TX registers are not valid in the SmartAck mode! The ChipID can be queried via the registers 150 & 151.
- In order to activate SmartAck for a receiving channel, the corresponding Modbus Coil (400…414) must be set.
- If the SmartAck function is activated at a receiving channel or if a SmartAck device is learned-in already, it is not feasible to change the ORG-Byte and the ID of the learned-in device “manually” via the Modbus registers. The learning-in of a SmartAck device must obligatory be made via the Learning-Coils (0…14) – The „Learning-in“by entering the EnOcean device ID into the Modbus register is not sufficient.
- If a SmartAck device should be deleted from a receiving channel, the corresponding Modbus Coil (300…314) must be activated.
- Via the Modbus-Coils 500…514 it can be queried on which channels a SmartAck device is learned-in already.

3.1 Description

3.1.1 Activation of SmartAck Function

Via Coils 400…414 the SmartAck function for the receiving channels RX1…RX15 are activated/deactivated. These Coils are configuration parameters which remain their values even after a device restart.

Modbus Coils	
Data Address	**Description** 0: SmartAck deactivated 1: SmartAck activated
400 R/W	SmartAck Function receiving channel RX 1
401 R/W	SmartAck Function for receiving channel RX 2
…	
414 R/W	SmartAck Function for receiving channel RX 15

3.1.2 Delete EnOcean Device from STC65

Via Coils 300…331 an EnOcean device can be deleted from STC65.

Modbus Coils	
Data Address	**Description** 1: Delete SmartAck device
300 R/W	Delete EnOcean device from RX1
301 R/W	Delete EnOcean device from RX2
…	
331 R/W	Delete EnOcean device from RX32

If for example Coil 300 is written with the value „1“, the EnOcen device learned-in to RX1 is deleted from the STC65. Afterwards, the value of the coil is automatically reset to 0.

3.1.3 Verification to Which Channels a SmartAck Device Is Learned-In

Via Coils 500…514 it can be verified to which receiving channels a SmartAck device is learned-in already.

Modbus Coils	
Data Address	**Description**
500 R/W	SmartAck device on RX1
501 R/W	SmartAck device on RX2
…	
514 R/W	SmartAck device on RX15

3.1.4 SmartAck Information Register

Via registers 150 & 151 the ChipID of the EnOcean Module can be queried.
In Register 152 the number of SmartAck devices which are saved in the EnOcean module is filed.

Modbus Input Register	
Data Address	**Description**
150 R	ChipID HIGH
151 R	ChipID LOW
152 R	Number of SmartAck devices learned-in in the EnOcean Module

3.2 Examples

3.2.1 Example 1: Learning-In of a SmartAck device type EEP D2-00-01

In this example an EnOcen SmartAck room operating unit with the EnOcean Profile EEP D2-00-01 is learned-in to the receiving channel RX1 of the STC65.

Learning-in Procedure:

1. Activate SmartAck Function for RX1 by setting the Modbus-Coil 400.
2. Set receiving channel RX1 into the learning mode by activating the Modbus-Coil 0 – now RX1 is in the learning mode and waits for a learning telegram of a SmartAck device.
3. Send a SmartAck learning telegram of the room operating unit (see the description of the room operating unit).
4. If the STC65 received the learning telegram, the value of the Modbus-Coils 0 is changed automatically to 0 and in the Registers of RX1 the values of the SmartAck room operating unit are shown.

3.2.2 Example 2: Send values to the SmartAck Room Operating Unit

After the SmartAck room operating unit was learned-in to the receiving channel RX1, data of the STC65 can be transmitted to the room operating unit via the transmitting channel TX1. Therefore, the registers of the transmitting channel TX1 must be written via Modbus according to the EnOcen profile definition. As soon as the room operating unit “wakes up” and requests the data, they are automatically sent to the SmartAck room operating unit by the STC65.

In this example, the following contents are indicated in the display of the SmartAck room operating unit:

- Fan stage Auto 3
- Room occupancy “occupied“
- Room temperature 20,00°C
- Heating symbol

Thereby, the following contents results from the SmartAck response telegram which shall be sent to the room operating unit by the STC after quering:

- ORG 0xD2 (VLD-Telegram)
- Number data bytes: 0x05
- DB0: 0x01 (Heating symbol)
- DB1: 0x07 (20,00°C)
- DB2: 0xD0 (20,00°C)
- DB3: 0x21 (Occupied & Room temperature °C)
- DB4: 0x42 (Fan Auto 3 & Message ID 2)

tx1

Tx = 33212: Err = 16: ID = 1: F = 03: SR = 100ms

	Alias	01000	Alias	01010
0	Register 1000	0xFFC4	Register 1010	0x0000
1	Register 1001	0x0000	Register 1011	0x0000
2	Register 1002	0x0000	Register 1012	0x0000
3	Register 1003	0x00D2	Register 1013	0x0000
4	Register 1004	0x0000	Register 1014	0x0000
5	Register 1005	0x0005	Register 1015	0x0000
6	Register 1006	0x0701	Register 1016	0x0000
7	Register 1007	0x21D0	Register 1017	0x0000
8	Register 1008	0x0042	Register 1018	0x0000
9	Register 1009	0x0000	Register 1019	0x0000

4 Modbus Protocol

4.1 Control Commands Supported

The following MODBUS – control commands are supported:

Description	Function Code	
Read Coil	01 (hex)	1 (dec)
Read Holding Register	03 (hex)	3 (dec)
Read Input Register	04 (hex)	4 (dec)
Write Single Coil	05 (hex)	5 (dec)
Write Single Register	06 (hex)	6 (dec)
Write Multiple Coils	0F (hex)	15 (dec)
Write Multiple Registers	10 (hex)	16 (dec)

Table 10
Supported Modbus Commands

4.2 Data Transmission

4.2.1 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.

4.2.2 Data Frame

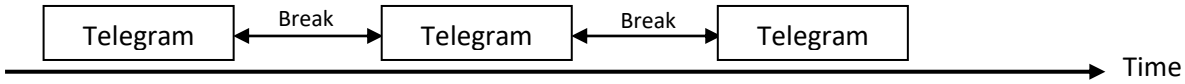
The data are sent to the bus in accordance to severely defined defaults:

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.

4.2.3 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.

4.2.3.1 Telegram Layout

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

4.2.3.2 Calculation of CRC-Checksum

The CRC checksum (Cyclical 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, Init
for(i = 0; i < telegram_length-2; i++)
        crc = crc_calc(crc, telegram_data[i]);

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

// Calculate CRC
unsigned int crc_calc(unsigned int crc_temp, unsigned int data)
{
        unsigned int    Index_CC=0;
        unsigned int    LSB=0;
        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 for CRC16
        }
        return(crc_temp);
}
```

4.2.4 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.

4.2.4.1 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 2 char	Data 0 - 2 x 100 char	Checksum LRC 2 char	End 2 char
:					**CR LF**

4.2.4.2 Calculation of LRC-Checksum

The LRC checksum (Longitudinal Redundancy Check) is calculated 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 significant 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):

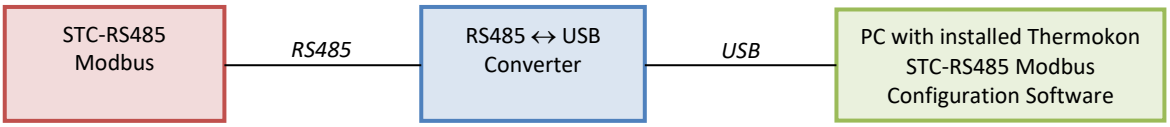
```
lrc = 0;
for(i = 1; i < telegram_length -4; i++)
        lrc = lrc + telegram_data[i];

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

5 Configuration Software

Please download (http://www.thermokon.de/ftp/STC65-RS485-Modbus/stc-rs485_modbus_setup.exe) and install the Thermokon STC-RS485 Modbus configuration software to have easy access to all configuration properties and operation values of the STC-RS485 Modbus. By the configuration software you are able to read/write the Modbus registers of the STC-RS485 Modbus over an easy to handle graphical user interface.

5.1 Connection to PC

Please use a USB to RS485 converter to connect the STC-RS485 Modbus with your computer.



For further information please refer the documentation of the USB to RS485 converter.