234 Feel ModBus protocol

Content

1 Modbus RTU

1.1 Devices address

The device addresses are used to describe and distinguish the device in a bus system. The device address must be sent with each read or write command.

Address types	Address
Standard	0x01
Adjustable	0x01...0xF7
Broadcast	0x00

Standard

The default address is set by the manufacturer and should be changed to a free bus address.

Adjustable

In addition to the default address (0x01), addresses from 1 to 247 (0x01...0xF7) are freely selectable.

Broadcast

The device always responds to the broadcast address 0. With several bus stations connected, read commands to the broadcast address must not be used.
A write command to this address is accepted by all connected devices. In this case, no response is sent. Registers written in this way can only be checked by reading them back.

1.2 RS485 Settings

Default Settings:

Baud rate	**19200**
Data bits	8
Parity	Even
Stop bits	1

1.3 Telegram structure

A telegram always consists of an address part, the command (FC = function code), the data part and the checksum. Telegram elements of fixed length are grayed out. All data read out must be evaluated in the MSB First formatting.

1.3.1 Read register

Address	Function code	Register Start Address		Register length		Checksum	
1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	1 byte
		MSB	LSB	MSB	LSB	LSB	MSB

1.3.2 Reply

Address	Function code	Data length	First data register		...		Last data register		Checksum	
1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	1 byte
	FC	Byte count	MSB	LSB	MSB	LSB	MSB	LSB	LSB	MSB

1.3.3 Response in case of error

In case of an error 128 (0x80) is added to the function code and the error code is transmitted in the data part.
Response to read register with FC3 in case of error, telegram structure slave → master

Address	Function code	Data length	Error code	Checksum	
1 byte	1 byte	1 byte	1 byte	1 byte	1 byte
	FC + 0x80	0x01	01, 02, 03 or 04	LSB	MSB

1.4 Data type

At the communication with ModBus RTU both opposite sides must always know in which data type the requested or written data belong. Only then the data can be interpreted correctly. Furthermore you have to know how long the data are.
Actually all data types can be used. (int8, uint8, int16, uint16, float, string etc.)

2 Function Codes

The Codes function tells the device what action to perform. For example, read a single register.
The different register types inform the device about the register size. Furthermore it is defined whether the value in the register may be changed via the ModBus protocol or not.

Function Code [FC]	Register type	Access	Size
1 / 0x01	Singel Output Coil	Read	1bit
3 / 0x03	Singel Holding Register	Read	16Bits
4 / 0x04	Multiple input registers	Read	16Bits
5 / 0x05	Singel Output Coil	Writing	1bit
6 / 0x06	Single Holding Register	Writing	16Bits
15 / 0x0F	Multiple output coils	Writing	1bit
16 / 0x10	Multiple holding registers	Writing	16Bits

2.1 FC1 (0x01) Read Coils

This function code is used to read output coils.
The response is always in bytes. If only one coil is read, the slave responds with one byte. The status of the read coil is in the LSB of the byte. If several coils are read at the same time, these coils fill up the byte. The LSB corresponds to the register start address and the MSB to the start address + 7. If more than 8 coils are read, the following bytes are filled according to the same scheme.

2.1.1 Request

Protocol data	Data length	Value
Function code	1 byte	0x01
Start address	2 bytes	0x0000...0xFFFF
Number of registers	2 bytes	0x0001...0x07D0

2.1.2 Reply

Protocol data	Data length	Value
Function code	1 byte	0x01
Byte count	1 byte	N = Number of bytes sent
Coil status	n bytes	n = N or N+1

2.1.3 Error response

Protocol data	Data length	Value	
Function code	1 byte	0x81	
Error code	1 byte	Function code not supported	0x01
		Start address not allowed	0x02
		Inadmissible number of registers	0x03
		Read error output coil	0x04

2.1.4 Example

The registers 0x03...0x0E are to be read.

Request		Reply	
Protocol data	Data	Protocol data	Data
Function code	0x01	Function code	0x01
Start address MSB	0x00	Byte count	0x02
Start address LSB	0x03	Data Byte 1 (0x0B...0x03)	0xCF
Number of registers MSB	0x00	Data Byte 2 (0x0E...0x0C)	0x04
Number of registers LSB	0x0B		

For the registers 0x0B to 0x03 this results in a bit sequence of 1100 1111 where the last 1 is register 0x03. For the registers 0x0E to 0x0C the following bit pattern 0000 0100 results.

2.2 FC3(0x03) Read Holding Registers

This function code is used to read holding registers. A register address always corresponds to 2 bytes, i.e. 16 bits. Therefore at least 2 bytes are sent as response. The first byte corresponds to the MSB and the second byte to the LSB.
If data with a smaller data type than 16 bits is stored in such a register, two bytes are still sent. The first byte is then empty.

2.2.1 Request

Protocol data	Data length	Value
Function code	1 byte	0x03
Start address	2 bytes	0x0000...0xFFFF
Number of registers	2 bytes	0x0001...0x007D (125)

2.2.2 Reply

Protocol data	Data length	Value
Function code	1 byte	0x03
Byte count	1 byte	2 x N
Register content	N x 2 bytes	

2.2.3 Error response

Protocol data	Data length	Value	
Function code	1 byte	0x83	
Error code	1 byte	Function code not supported	0x01
		Start address not allowed	0x02
		Inadmissible number of registers	0x03
		Read error holding register	0x04

2.2.4 Example

The registers 0x20...0x22 are to be read.

Request		Reply	
Protocol data	Data	Protocol data	Data
Function code	0x03	Function code	0x03
Start address MSB	0x00	Byte count	0x06
Start address LSB	0x20	Register content MSB 0x20	0x46
Number of registers MSB	0x00	Register content LSB 0x20	0x53
Number of registers LSB	0x03	Register content MSB 0x21	0x4D
		Register content LSB 0x21	0x20
		Register content MSB 0x22	0x41
		Register content LSB 0x22	0x47

So also whole strings can be read. In this case two ascii characters are stored in one register. In this case now "FSM AG". The "F" and "S" are in register 0x20, the "M" and "space" in register 0x21 and the "A" and "G" in register 0x22.

2.3 FC4(0x04) Read Input Registers

This function code is used to read input registers. A register address always corresponds to 2 bytes, i.e. 16 bits. Therefore at least 2 bytes are sent as response. The first byte corresponds to the MSB and the second byte to the LSB.
If data with a smaller data type than 16 bits is stored in such a register, two bytes are still sent. The first byte is then empty.

2.3.1 Request

Protocol data	Data length	Value
Function code	1 byte	0x04
Start address	2 bytes	0x0000...0xFFFF
Number of registers	2 bytes	0x0001...0x007D (125)

2.3.2 Reply

Protocol data	Data length	Value
Function code	1 byte	0x04
Byte count	1 byte	2 x N
Register content	N x 2 bytes	

2.3.3 Error response

Protocol data	Data length	Value	
Function code	1 byte	0x84	
Error code	1 byte	Function code not supported	0x01
		Start address not allowed	0x02
		Inadmissible number of registers	0x03
		Read error input register	0x04

2.3.4 Example

Register 16 is to be read. The content is defined as uint16.

Request		Reply	
Protocol data	Data	Protocol data	Data
Function code	0x04	Function code	0x04
Start address MSB	0x00	Byte count	0x02
Start address LSB	0x10	Register content MSB 0x10	0x65
Number of registers MSB	0x00	Register content LSB 0x10	0x53
Number of registers LSB	0x01		

As answer we get 0x65 in the first byte of the register and 0x53 in the second byte. This must now be merged to a uint16. So we get 0x6553 which is decimal 25939.

2.4 FC5 (0x05) Write Single Coil

This function code is used to write individual output coils.
Two bytes are always sent. However, there are only two valid data that can be sent. 0x0000 (OFF/0) and 0xFF00 (ON/1). Other data are invalid and are treated as errors.
The normal response to a successful write to the register is an echo of the sent request.

2.4.1 Request

Protocol data	Data length	Value
Function code	1 byte	0x05
Register address	2 bytes	0x0000...0xFFFF
Coil status	2 bytes	0x0000 or 0xFF00

2.4.2 Reply

Protocol data	Data length	Value
Function code	1 byte	0x05
Register address	2 bytes	0x0000...0xFFFF
Coil status	2 bytes	0x0000 or 0xFF00

2.4.3 Error response

Protocol data	Data length	Value	
Function code	1 byte	0x85	
Error code	1 byte	Function code not supported	0x01
		Start address not allowed	0x02
		Illegal register content	0x03
		Writing error output coil	0x04

2.4.4 Example

A 1 is to be written into register 5 or the status is to be set to ON.

Request		Reply	
Protocol data	Data	Protocol data	Data
Function code	0x05	Function code	0x05
Register address MSB	0x00	Register address MSB	0x00
Register -Address LSB	0x04	Register address LSB	0x04
Coil Status MSB	0xFF	Coil Status MSB	0xFF
Coil status LSB	0x00	Coil status LSB	0x00

2.5 FC6 (0x06) Write Single Register

This function code is used to write holding registers. A register address always corresponds to 2 bytes, i.e. 16 bits. Therefore, at least 2 bytes must be sent. The first byte corresponds to the MSB and the second byte to the LSB.
If data with a smaller data type than 16 bits is stored in such a register, two bytes are still sent. The first byte is then empty. The normal response to a successful write of the register is an echo of the sent request.

2.5.1 Request

Protocol data	Data length	Value
Function code	1 byte	0x06
Start address	2 bytes	0x0000...0xFFFF
Register content	2 bytes	0x0000...0xFFFF

2.5.2 Reply

Protocol data	Data length	Value
Function code	1 byte	0x06
Register address	2 bytes	0x0000...0xFFFF
Register content	2 bytes	0x0000...0xFFFF

2.5.3 Error response

Protocol data	Data length	Value	
Function code	1 byte	0x86	
Error code	1 byte	Function code not supported	0x01
		Start address not allowed	0x02
		Illegal register content	0x03
		Writing error holding register	0x04

2.5.4 Example

Register 4 is to be written with the string "OK". To do this, the hex values for "O" and "K" must be determined from an ASCII table and the bytes must then be written with them. For "O" and "K" this would be 0x4F and 0x4B.

Request		Reply	
Protocol data	Data	Protocol data	Data
Function code	0x06	Function code	0x06
Register address MSB	0x00	Register address MSB	0x00
Register address LSB	0x04	Register address LSB	0x04
Register content MSB	0x4F	Register content MSB	0x4F
Register content LSB	0x4B	Register content LSB	0x4B

2.6 FC15 (0x0F) Write Multiple Coils

This function code is used to write multiple output coils.
Whole bytes are always written. As with reading output coils, the status of the lowest register is written in the LSB of the first byte. After that the byte is filled up with the next 7 registers. If this byte is full, the next register is written again in the LSB of the second byte.

2.6.1 Request

Protocol data	Data length	Value
Function code	1 byte	0x0F
Start address	2 bytes	0x0000...0xFFFF
Number of registers	2 bytes	0x0001...0x07B0
Byte count	1 byte	N
Coil status	N x 1 byte	

2.6.2 Reply

Protocol data	Data length	Value
Function code	1 byte	0x0F
Start address	2 byte	0x0000...0xFFFF
Number of registers	n bytes	0x0001...0x07B0

2.6.3 Error response

Protocol data	Data length	Value	
Function code	1 byte	0x8F	
Error code	1 byte	Function code not supported	0x01
		Start address not allowed	0x02
		Inadmissible number of registers	0x03
		Multiple Output Coil Write Error	0x04

2.6.4 Example

The registers 0x13...0x1E are to be written.

Request		Reply	
Protocol data	Data	Protocol data	Data
Function code	0x0F	Function code	0x0F
Start address MSB	0x00	Byte count	0x02
Start address LSB	0x13	Data Byte 1 (0x0B...0x03)	0xCF
Number of registers MSB	0x00	Data Byte 2 (0x0E...0x0C)	0x04
Number of registers LSB	0x0B		

For the registers 0x1B to 0x13 this results in a bit sequence of 1100 1111 where the last 1 is register 0x13. For the registers 0x1E to 0x1C the following bit pattern 0000 0100 results.

2.7 FC16 (0x10) Write Multiple registers

This function code is used to write several holding registers. This is especially needed for values whose data type is larger than 16bit. e.g.: an int32. A register address always corresponds to 2 bytes, i.e. 16 bits. Therefore at least 2 bytes must be sent. The first byte corresponds to the MSB and the second byte to the LSB. If all registers are written successfully, the start address and the number of the written registers is answered.

2.7.1 Request

Protocol data	Data length	Value
Function code	1 byte	0x10
Start address	2 bytes	0x0000...0xFFFF
Number of registers	2 bytes	0x0001...0x007B
Byte count	1 byte	2 x N
Coil status	N x 2 bytes	value

2.7.2 Reply

Protocol data	Data length	Value
Function code	1 byte	0x10
Start address	2 byte	0x0000...0xFFFF
Number of registers	2 bytes	0x0001...0x007B

2.7.3 Error response

Protocol data	Data length	Value	
Function code	1 byte	0x90	
Error code	1 byte	Function code not supported	0x01
		Start address not allowed	0x02
		Inadmissible number of registers	0x03
		Write error multiple input register	0x04

2.7.4 Example

Registers 5 and 6 are to be written.

Request		Reply	
Protocol data	Data	Protocol data	Data
Function code	0x10	Function code	0x10
Start address MSB	0x00	Start address MSB	0x00
Start address LSB	0x05	Start address LSB	0x05
Number of registers MSB	0x00	Number of registers MSB	0x00
Number of registers LSB	0x04	Number of registers LSB	0x02
Byte count	0x08		
Register content MSB 5	0x8D		
Register content LSB 5	0xFF		
Register content MSB 6	0x89		
Register content LSB 6	0x98		

Two int32 values have now been sent here. Thus 0x8DFF8998 (-1912632936) was written into register 5 and 6.

3 Register Maps

3.1 Output coil

Read by means of FC1 and write by means of FC5 and FC15.

Temperature sensor		
Register address	**Meaning/Content**	**Type**
0	**Temperature sensor adjustment @negative amplitude** Triggers an adjustment of the negative amplitude and thus changes the slope of the linearization line between zero point and negative amplitude	Bit
1	**Temperature sensor @ 0°C** Triggers a correction of the temperature sensor to clean up the offset "1"→ start the operation The register can be read to check whether the process has been completed. "0"→ Process completed "1"→ Process not completed	Bit
2	**Temperature Sensor Adjustment @Positive Amplitude** Triggers an adjustment of the positive amplitude and thus changes the slope of the linearization line between zero point and positive amplitude	Bit

Humidity sensor		
Register address	**Meaning/Content**	**Type**
3	**Humidity sensor adjustment @11.3%rH** Triggers an adjustment at 11.3%rH and thus changes the slope of the linearization line	Bit
4	**Humidity sensor adjustment @75.3%rH** Triggers an adjustment at 75.3%rH and thus changes the slope of the linearization line	Bit

Pressure sensor		
Register address	**Meaning/Content**	**Type**
5	**Pressure sensor adjustment @negative amplitude** Triggers an adjustment of the negative amplitude and thus changes the slope of the linearization line between zero point and negative amplitude	Bit
6	**Zero pressure sensor** Triggers a zero point correction of the pressure sensor to clean up the zero offset. "1"→ start the zeroing process The register can be read to check whether the zeroing process has been completed. "0"→ Zeroing process completed "1"→ Zeroing process not completed	Bit
7	**Pressure Sensor Adjustment @Positive Amplitude** Triggers an adjustment of the positive amplitude and thus changes the slope of the linearization line between zero point and positive amplitude	Bit

8	**Pressure sensor automatic zeroing** "0"→ Automatic zeroing off "1"→ Automatic zeroing on	Bit

CO2 sensor		
Register address	**Meaning/Content**	**Type**
9	**CO2 sensor adjustment @amplitude** Triggers an adjustment of the amplitude and thus changes the slope of the linearization line	Bit
10	**CO2 sensor adjustment @500ppm** Triggers an adjustment of the amplitude and thus changes the slope of the linearization line	Bit

Border contacts		
Register address	**Meaning/Content**	**Type**
11	**Limit contact 1 Active/not active** "0"→ Limit contact not active "1"→ Border Contact Active	Bit
12	**Limit contact 2 Active/not active** "0"→ Limit contact not active "1"→ Border Contact Active	Bit
13	**Limit contact 1 actuated/not actuated** "0"→ Limit contact not actuated "1"→ Limit contact actuated	Bit
14	**Limit contact 2 actuated/not actuated** "0"→ Limit contact not actuated "1"→ Limit contact actuated	Bit

3.2 Input Registers:

Read only with FC4.

General information		
Register address	**Meaning/Content**	**Type**
0 -> 5	**Firmware version** Returns the firmware version of the sensor in string format.	12 x Uint8
6 -> 10	**Hardware version** Returns the hardware version of the sensor in string format.	10 x Uint8
11 -> 20	**Manufacturer serial number** Returns the manufacturer serial number of the sensor in string format.	20 x Uint8

VOC sensor		
Register address	**Meaning/Content**	**Type**
30 -> 39	**VOC sensor type** Returns the type of the VOC sensor in string format.	20 x Uint8
40	**Status VOC sensor** "0"→ Within the permissible measuring range "1"→ below the minimum VOC index "2"→ above the maximum VOC index	Uint8
41	**Raw measured value VOC sensor** Returns the raw value from the VOC sensor	int16
42	**Percentage measured value VOC sensor** Returns the percentage measurement value from the set range for the VOC sensor. Output contains 2 decimal places e.g.: 4573 = 45.73%	int16
43 -> 44	**Real measured value VOC sensor** Returns the measured value in the set unit without decimal places	Int32
45 -> 49	**Unit VOC sensor** Reads the unit of the sensor in string format.	10 x Uint8

Temperature sensor		
Register address	**Meaning/Content**	**Type**
50 -> 59	**Temperature sensor type** Returns the type of the temperature sensor in string format.	20 x Uint8
60	**Temperature sensor status** "0"→ Within the set measuring range "1"→ below the minimum temperature "2"→ above the maximum temperature	Uint8
61	**Raw measured value temperature sensor** Returns the raw value from the temperature sensor	int16
62	**Percentage measured value temperature sensor** Returns the percentage measurement value of the set range for the temperature sensor. Output contains 2 decimal places e.g.: 8523 = 85.23%	int16
63 -> 64	**Real measured value temperature sensor** Returns the measured value in the set unit. Output contains 2 decimal places	Int32

	e.g. 2568 = 25.68°C	
65 -> 69	**Unit temperature sensor** Reads the unit of the sensor in string format.	10 x Uint8

Humidity sensor		
Register address	**Meaning/Content**	**Type**
70 -> 79	**Humidity sensor type** Returns the type of the humidity sensor in string format.	20 x Uint8
80	**Humidity sensor status** "0"→ Within the set measuring range "1"→ below the minimum humidity "2"→ above the maximum humidity	Uint8
81	**Raw measured value humidity sensor** Returns the raw value from the humidity sensor	int16
82	**Percentage measured value humidity sensor** Returns the percentage of the set range for the humidity sensor. Output contains 2 decimal places e.g.: 8523 = 85.23%	int16
83 -> 84	**Real measured value humidity sensor** Returns the measured value in the set unit. Output contains 2 decimal places e.g. 5127 = 51.27%	Int32
85 -> 89	**Unit humidity sensor** Reads the unit of the sensor in string format.	10 x Uint8

Pressure sensor		
Register address	**Meaning/Content**	**Type**
90 -> 99	**Pressure sensor type** Returns the type of the pressure sensor in string format.	20 x Uint8
100	**Pressure sensor status** "0"→ Within the set measuring range "1"→ under the minimum pressure "2"→ above the maximum pressure	Uint8
101	**Raw measured value pressure sensor** Returns the raw value from the pressure sensor	int16
102	**Percentage measured value pressure sensor** Returns the percentage measurement value of the set range for the pressure sensor. Output contains 2 decimal places e.g.: 8523 = 85.23%	int16
103 -> 104	**Real measured value pressure sensor** Returns the measured value in the set unit. Output contains 2 decimal places e.g. 25445 = 254.45Pa	Int32
105 -> 109	**Unit pressure sensor** Reads the unit of the sensor in string format.	10 x Uint8

CO2 sensor		
Register address	**Meaning/Content**	**Type**
110 -> 119	**CO2 sensor type**	20 x Uint8

	Returns the type of the CO2 sensor in string format.	
120	**Status CO2 sensor** "0"→ Within the set measuring range "1"→ under the minimum pressure "2"→ above the maximum pressure	Uint8
121	**Raw measured value CO2 sensor** Returns the raw value from the CO2 sensor	int16
122	**Percentage measured value CO2 sensor** Returns the percentage measurement value of the set range for the CO2 sensor. Output contains 2 decimal places e.g.: 8523 = 85.23%	int16
123 -> 124	**Real measured value CO2 sensor** Returns the measured value in the set unit without decimal places e.g. 951 = 951 ppm	Int32
125 -> 129	**Unit CO2 sensor** Reads the unit of the sensor in string format.	10 x Uint8

3.3 Holding Registers:

Read with FC3, write with FC6 (of one register) or with FC16 (of several registers).

General		
Register address	**Meaning/Content**	**Type**
0 -> 9	**Customer serial number** Memory for the customer's serial number in string format.	20 x Uint8
10 -> 11	**Baud rate** Sets the baud rate of the RS485 interface. Values from 3000...200,000 baud are possible	Uint32
12	**ModBus address** Sets the ModBus address. Address 0x00 is considered as broadcast address and cannot be used Default address is 0x01 Addresses from 0x01 to 0xF7 are possible	Uint8

VOC sensor		
Register address	**Meaning/Content**	**Type**
20	**Unit VOC sensor** Setting the unit of the sensor. "0"→ "VOC"	Uint8
21 -> 22	**Lower measuring range VOC sensor** Reading or writing the lower measuring range in the set unit of the VOC sensor without decimal places. e.g. 35 = 35	Int32
23 -> 24	**Upper measuring range VOC sensor** Reading or writing the upper measuring range in the set unit of the VOC sensor without decimal places. e.g. 85 = 85	Int32
25	**Switching mode limit contact 1 of the VOC sensor** "0"→ Sensor has no effect on limit contact "1"→ a switching threshold: Upper switching threshold > measured value = limit contact off "2"→ a switching threshold: Upper switching threshold > measured value = limit contact on "3"→ two switching thresholds: Lower switching threshold < measured value > upper switching threshold = limit contact on "4"→ two thresholds: Lower switching threshold < measured value > upper switching threshold = limit contact on	Uint8
26	**Lower switching threshold limit contact 1 of the VOC sensor** Sets the lower switching threshold of limit contact 1 for the sensor. The value is given as a percentage with 2 decimal places. e.g.: 1000 = 10.00%	Int16
27	**Upper switching threshold Limit contact 1 of the VOC sensor** Sets the lower switching threshold of limit contact 1 for the sensor. The value is given as a percentage with 2 decimal places.	Int16

	e.g.: 9000 = 90.00%	
28	**Switching mode limit contact 2 of the VOC sensor** "0"→ Sensor has no effect on limit contact "1"→ a switching threshold: Upper switching threshold > measured value = limit contact off "2"→ a switching threshold: Upper switching threshold > measured value = limit contact on "3"→ two switching thresholds: Lower switching threshold < measured value > upper switching threshold = limit contact on "4"→ two thresholds: Lower switching threshold < measured value > upper switching threshold = limit contact on	Uint8
29	**Lower switching threshold limit contact 2 of the VOC sensor** Sets the lower switching threshold of limit contact 2 for the sensor. The value is given as a percentage with 2 decimal places. e.g.: 1000 = 10.00%	Int16
30	**Upper switching threshold Limit contact 2 of the VOC sensor** Sets the lower switching threshold of limit contact 2 for the sensor. The value is given as a percentage with 2 decimal places. e.g.: 9000 = 90.00%	Int16

Temperature sensor		
Register address	**Meaning/Content**	**Type**
40	**Unit temperature sensor** Setting the unit of the sensor. "0"→ "degK" "1"→ "degC" "2"→ "degF"	Uint8
41 -> 42	**Lower measuring range temperature sensor** Reading or writing the lower measuring range in the set unit of the temperature sensor with 2 decimal places. e.g. -4000 = -40.00 °C	Int32
43 -> 44	**Upper measuring range temperature sensor** Reading or writing the upper measuring range in the set unit of the temperature sensor with 2 decimal places. e.g. 8000 = 80.00 °C	Int32
45	**Switching mode limit contact 1 of the temperature sensor** "0"→ Sensor has no effect on limit contact "1"→ a switching threshold: Upper switching threshold > Measured value = Limit contact off "2"→ a switching threshold: Upper switching threshold > measured value = limit contact on "3"→ two switching thresholds: Lower switching threshold < measured value > upper switching threshold = limit contact on "4"→ two thresholds: Lower switching threshold < measured value > upper switching threshold = limit contact on	Uint8

46	**Lower switching threshold limit contact 1 of the temperature sensor** Sets the lower switching threshold of limit contact 1 for the sensor. The value is given as a percentage with 2 decimal places. e.g.: 1000 = 10.00%	Int16
47	**Upper switching threshold limit contact 1 of the temperature sensor** Sets the lower switching threshold of limit contact 1 for the sensor. The value is given as a percentage with 2 decimal places. e.g.: 9000 = 90.00%	Int16
48	**Switching mode limit contact 2 of the temperature sensor** "0"→ Sensor has no effect on limit contact "1"→ a switching threshold: Upper switching threshold > Measured value = Limit contact off "2"→ a switching threshold: Upper switching threshold > measured value = limit contact on "3"→ two switching thresholds: Lower switching threshold < measured value > upper switching threshold = limit contact on "4"→ two thresholds: Lower switching threshold < measured value > upper switching threshold = limit contact on	Uint8
49	**Lower switching threshold limit contact 2 of the temperature sensor** Sets the lower switching threshold of limit contact 2 for the sensor. The value is given as a percentage with 2 decimal places. e.g.: 1000 = 10.00%	Int16
50	**Upper switching threshold limit contact 2 of the temperature sensor** Sets the lower switching threshold of limit contact 2 for the sensor. The value is given as a percentage with 2 decimal places. e.g.: 9000 = 90.00%	Int16

Humidity sensor		
Register address	**Meaning/Content**	**Type**
60	**Unit humidity sensor** Setting the unit of the sensor. "0"→ "%rH" "1"→ "g/m^3"	Uint8
61 -> 62	**Lower measuring range humidity sensor** Reading or writing the lower measuring range in the set unit of the humidity sensor with 1 decimal place. e.g. 205 = 20.5 %rH	Int32
63 -> 64	**Upper measuring range humidity sensor** Reading or writing the upper measuring range in the set unit of the humidity sensor with 1 decimal place. e.g. 753 = 75.3 %rH	Int32
65	**Switching mode limit contact 1 of the humidity sensor** "0"→ Sensor has no effect on limit contact "1"→ a switching threshold: Upper switching threshold > measured value = limit contact off "2"→ a switching threshold: Upper switching threshold > measured value = limit contact on	Uint8

	"3"→ two switching thresholds: Lower switching threshold < measured value > upper switching threshold = limit contact on "4"→ two thresholds: Lower switching threshold < measured value > upper switching threshold = limit contact on	
66	**Lower switching threshold limit contact 1 of the humidity sensor** Sets the lower switching threshold of limit contact 1 for the sensor. The value is given as a percentage with 2 decimal places. e.g.: 1000 = 10.00%	Int16
67	**Upper switching threshold limit contact 1 of the humidity sensor** Sets the lower switching threshold of limit contact 1 for the sensor. The value is given as a percentage with 2 decimal places. e.g.: 9000 = 90.00%	Int16
68	**Switching mode limit contact 2 of the humidity sensor** "0"→ Sensor has no effect on limit contact "1"→ a switching threshold: Upper switching threshold > measured value = limit contact off "2"→ a switching threshold: Upper switching threshold > measured value = limit contact on "3"→ two switching thresholds: Lower switching threshold < measured value > upper switching threshold = limit contact on "4"→ two thresholds: Lower switching threshold < measured value > upper switching threshold = limit contact on	Uint8
69	**Lower switching threshold limit contact 2 of the humidity sensor** Sets the lower switching threshold of limit contact 2 for the sensor. The value is given as a percentage with 2 decimal places. e.g.: 1000 = 10.00%	Int16
70	**Upper switching threshold limit contact 2 of the humidity sensor** Sets the lower switching threshold of limit contact 2 for the sensor. The value is given as a percentage with 2 decimal places. e.g.: 9000 = 90.00%	Int16

Pressure sensor		
Register address	**Meaning/Content**	**Type**
80	**Unit pressure sensor** Setting the unit of the sensor. "0"→ "Pa" "1"→ "hPa" "1"→ "kPa" "1"→ "mbar" "1"→ "bar" "1"→ "psi" "1"→ "mmH2O"	Uint8
81 -> 82	**Lower measuring range pressure sensor** Reading or writing the lower measuring range in the set unit of the pressure sensor with 4 decimal places.	Int32

	e.g. -1000000 = -100.0000 Pa	
83 -> 84	**Upper measuring range pressure sensor** Reading or writing the upper measuring range in the set unit of the pressure sensor with 4 decimal places. e.g. 1000000 = 100.0000 Pa	Int32
85	**Switching mode limit contact 1 of the pressure sensor** "0"→ Sensor has no effect on limit contact "1"→ a switching threshold: Upper switching threshold > measured value = limit contact off "2"→ a switching threshold: Upper switching threshold > measured value = limit contact on "3"→ two switching thresholds: Lower switching threshold < measured value > upper switching threshold = limit contact on "4"→ two thresholds: Lower switching threshold < measured value > upper switching threshold = limit contact on	Uint8
86	**Lower switching threshold limit contact 1 of the pressure sensor** Sets the lower switching threshold of limit contact 1 for the sensor. The value is given as a percentage with 2 decimal places. e.g.: 1000 = 10.00%	Int16
87	**Upper switching threshold limit contact 1 of the pressure sensor** Sets the lower switching threshold of limit contact 1 for the sensor. The value is given as a percentage with 2 decimal places. e.g.: 9000 = 90.00%	Int16
88	**Switching mode limit contact 2 of the pressure sensor** "0"→ Sensor has no effect on limit contact "1"→ a switching threshold: Upper switching threshold > Measured value = Limit contact off "2"→ a switching threshold: Upper switching threshold > measured value = limit contact on "3"→ two switching thresholds: Lower switching threshold < measured value > upper switching threshold = limit contact on "4"→ two thresholds: Lower switching threshold < measured value > upper switching threshold = limit contact on	Uint8
89	**Lower switching threshold limit contact 2 of the pressure sensor** Sets the lower switching threshold of limit contact 2 for the sensor. The value is given as a percentage with 2 decimal places. e.g.: 1000 = 10.00%	Int16
90	**Upper switching threshold limit contact 2 of the pressure sensor** Sets the lower switching threshold of limit contact 2 for the sensor. The value is given as a percentage with 2 decimal places. e.g.: 9000 = 90.00%	Int16

CO2 sensor		
Register address	**Meaning/Content**	**Type**
100	**Unit CO2 sensor**	Uint8

	Setting the unit of the sensor. "0"→ "ppm"	
101 -> 102	**Lower measuring range CO2 sensor** Reading or writing the lower measuring range in the set unit of the CO2 sensor with no decimal places. e.g. 500 = 500 ppm	Int32
103 -> 104	**Upper measuring range CO2 sensor** Reading or writing the upper measuring range in the set unit of the CO2 sensor with no decimal places. e.g. 2000 = 2000 ppm	Int32
105	**Switching mode limit contact 1 of the CO2 sensor** "0"→ Sensor has no effect on limit contact "1"→ a switching threshold: Upper switching threshold > Measured value = Limit contact off "2"→ a switching threshold: Upper switching threshold > measured value = limit contact on "3"→ two switching thresholds: Lower switching threshold < measured value > upper switching threshold = limit contact on "4"→ two thresholds: Lower switching threshold < measured value > upper switching threshold = limit contact on	Uint8
106	**Lower switching threshold limit contact 1 of the CO2 sensor** Sets the lower switching threshold of limit contact 1 for the sensor	Int16
107	**Upper switching threshold Limit contact 1 of the CO2 sensor** Sets the lower switching threshold of limit contact 1 for the sensor	Int16
108	**Switching mode limit contact 2 of the CO2 sensor** "0"→ Sensor has no effect on limit contact "1"→ a switching threshold: Upper switching threshold > measured value = limit contact off "2"→ a switching threshold: Upper switching threshold > measured value = limit contact on "3"→ two switching thresholds: Lower switching threshold < measured value > upper switching threshold = limit contact on "4"→ two thresholds: Lower switching threshold < measured value > upper switching threshold = limit contact on	Uint8
109	**Lower switching threshold limit contact 2 of the CO2 sensor** Sets the lower switching threshold of limit contact 2 for the sensor. The value is given as a percentage with 2 decimal places. e.g.: 1000 = 10.00%	Int16
110	**Upper switching threshold Limit contact 2 of the CO2 sensor** Sets the lower switching threshold of limit contact 2 for the sensor. The value is given as a percentage with 2 decimal places. e.g.: 9000 = 90.00%	Int16

4 Calculate CRC

The following function can be used to calculate the checksum:

```
uint16_t CalculateCRC(uint8_t ByteCount, uint8_t *CRCData)
{
   unsigned char CRCHi = 0xFF;
   unsigned char CRCLo = 0xFF;
   unsigned index;
   unsigned short Bytecount = ByteCount - 2;

   while (bytecount--)
   {
      Index = CRCLo ^ *CRCData++;
      CRCLo = CRCHi ^ CRCHighByte[Index] ;
      CRCHi = CRCLowByte[Index];
   }
   return (CRCHi << 8 | CRCLo);
}
```

5 String structure

Strings are represented in ASCII format and terminated with 0x00.
Form in Ansi-C standard: e.g. "V01.00".
Form in ASCII standard: "V01.00[NULL NULL]".
Form in Hex : 0x 5630 312E 3030 00

5.1 ASCII Table

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]