Table of Contents

Introduction

This document is used to describe the Modbus protocol used for XROOM units (control panel marked as XCONT-CP-E and XCONT-CP-B). The version of this manual is intended for **units with Firmware version 200 and above**. The FW version number is indicated on a self-adhesive label on the PCB.

Let's start with some useful information for troubleshooting:

Only those registers that are available on the unit can be read. Otherwise, the unit responds with an error code 0x02 - Illegal data address.

The unit requires a certain amount of time to process the request, so it is necessary to allow sufficient time for the unit to respond. The time before the unit responds varies depending on the selected modbus function and the number of read/written registers. Typical response time is approx. 4 ms

In case the unit is not communicating, make sure that the frames you are sending are correct and check that you are observing pauses of at least 4 ms on the communication bus for correct detection of frame ends.

The bus operates in the so-called **Half-duplex** mode. This means that it is not able to accept further requests until the previous modbus frame has been responded.

To **check** or verify the correctness of **the modbus crc calculations**, it is possible to use the on-line calculator:
https://www.lammertbies.nl/comm/info/crc-calculation.html
It is necessary to switch the calculator to HEX mode and the subsequent CRC-16 (Modbus) result has the upper and lower bytes swapped in the Modbus frame.

General description of the Modbus protocol

The Modbus protocol is a Master-Slave protocol. Only 1 master and up to 247 slave devices (in our case units) may be present on the bus. Communication is always initiated by the master device. The slave only responds to requests from the master device. Modbus uses a Big-endian data representation. This means that for items larger than 1 B, the highest byte is sent first and the lowest byte afterwards.

Communication Modes

Unicast mode:

The master contacts **a single specific slave device** via its Modbus address. The slave processes the message and responds.

Protocol Data Unit (PDU)

Modbus functions	Data
1 B	N* 1 B

The Modbus protocol defines three basic types of PDU:

1) **Request PDU -** Used to contact one or more slave devices by the master.

   The Modbus function field contains the code of the specific Modbus function. The data field then according to the Modbus performs the function of address, number of variables, values of variables, and other

2) **Response PDU -** Used to send a **positive response** to the received Request PDU from the slave device.

   The **Modbus function** field contains the **same value** as in the received Request PDU. According to the given Modbus function operating values, read inputs, coils ..

3) **Exception Response PDU -** Used to send a **negative response** by the slave device to the received Request PDU.

   The **Modbus function** field contains the **value of the Modbus function from the Request PDU + 0x80** as a failure indication. The data part then **identifies the error**.

Error codes in Exception Response PDU

Code	*Function code type*	*Meaning*
0x01	Illegal Modbus functions	The required Modbus function is not supported by the server (unit)
0x02	Illegal data address	The specified address (coil, register ...) is outside the range supported by the server
0x03	Illegal data value	Transmitted data is invalid
0x04	Equipment failure	An unrecoverable error occurred during the execution of the request
0x05	Confirmation	Code to be used in programming. The server reports receiving a valid request, but it will take longer to execute
0x06	Device busy.	Code to be used in programming. The server is busy executing a long-running command.
0x08	Memory parity error	Code to be used for working with files. The server detected a parity error when trying to read the file
0x0A	Gateway - transmission path unavailable	Code for working with the gateway. The gateway is unable to reserve an internal transmission path from the input port to the output port. It is probably overloaded or incorrectly set.
0x0B	Gateway - the target device does not match	Code for working with the gateway. The target device is not responding, probably not present.

Modbus function codes

1) **Public function codes** - They are clearly defined and publicly documented. Their uniqueness is guaranteed. They also contain some unused codes for future use.
2) **User-defined function codes** - Allow the user to implement a function that is not defined by the protocol. Code uniqueness is not guaranteed.

Modbus function code ranges

Function code	*Function code type*
1 ... 64	Public function codes
65 ... 72	User-defined function codes
73 ... 100	Public function codes
101 ... 110	User-defined function codes
111 ... 127	Public function codes

Modbus General Address Space

The Modbus protocol address space is based on a set of tables with characteristic meanings. The following four basic tables are defined:

Table	*Description*	*Access*	*Address space (not required)*
Discrete inputs	1-bit	Only reading	0x2710 to 0x4E1F
Spools	1-bit	Reading and writing	0x0000 to 0x270F
Input registers	16-bit	Only reading	0x7530 to 0x9C3F
Preservation registers	16-bit	Reading and writing	0x9C40 to 0xC34F

Modbus on RS485 bus

Default RS485 bus settings

The Modbus RTU protocol defines the default serial link settings as follows:

Baud rate 19200
1 start bit
8 data bits
1 even parity bit
1 stop bit

The above parameters are the default settings of the unit

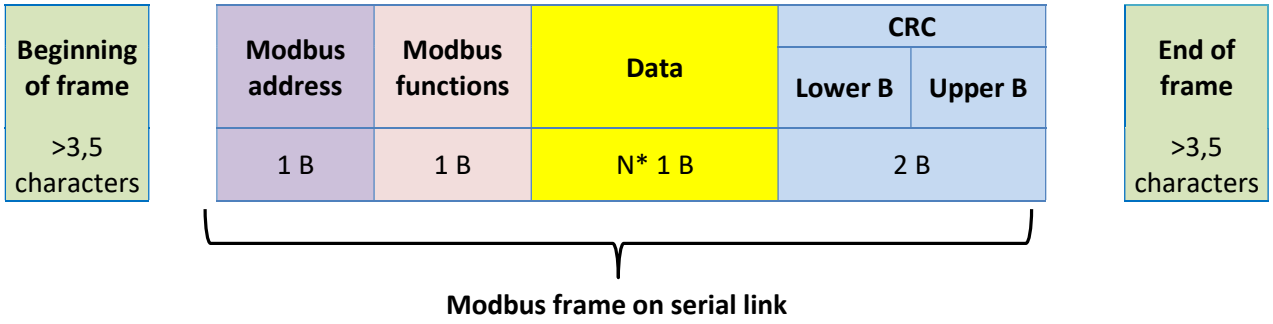
It is also possible to set baud rates 4800, 9600 and no parity option.

Connecting the wire pins to the RJ45 connector for connection to XCONT-HUB

Pins 1, 2, 7, 8 must be left unconnected
Pins 3, 4 - RS485 bus data wire A
Pins 5, 6 - RS485 bus data wire B

Structure of the Modbus RTU frame on the RS485 bus

In Modbus RTU mode, 1 B consists of two four-bit hex characters. The Modbus framework transmission is started and ended with a delay on the bus longer than 3.5 characters. During the transmission of the frame, the spaces between the characters shall not exceed 1.5 characters.



Addressing slave devices:

address	*significance*
0	Broadcast address
1 to 247	Individual slave addresses
248 to 255	Reserved

In Modbus RTU frames carrying responses intended for the master device, the Modbus address of the corresponding slave device is maintained.

CRC Calculation

The CRC calculation is performed from the entire frame including the Modbus slave address, the Modbus function and the data part of the frame

1. Initialization of the 16-bit CRC register to 0xFFFF.
2. XOR the first 8 bits of the frame with the lower byte of the CRC register and store the result in the CRC registry.
3. Shift the CRC register 1 b to the right (towards LSB), fill the MSB of the CRC register with 0. Capture and evaluate the lowest bit that has dropped out by the shift.
4. If this bit was equal to 1, perform XOR between the CRC register and the value 0xA001 (generating polynomial= 1+x2+x15+x16). The result is stored again in the CRC register.
5. Repeat steps 3 and 4 until eight CRC register shifts have been performed.
6. XOR the next 8 bits of the frame with the lower byte of the CRC register and repeat steps 3 to 5.
7. Continue like this till the last byte of the frame.
8. The result of the CRC calculation is stored in the CRC register.
9. When placing the CRC value in the Modbus frame, the upper and lower bytes of the CRC register must be swapped (see the structure of the Modbus RTU frame on the serial link).

List of Modbus functions			
Intention	***Function code***	***Command***	***Public/User***
Entry registers	0x04	Read input registers	Public
Storage registers	0x03	Read retention registers	Public
	0x10	Write multiple retention registers	Public

Address space of Modbus XCONT-CP units

List of input registers					
Inp. register	***Address of register***	***Description***	***of unit***	***Format***	***R/W***
FW version	0x7530	Firmware version in the unit	-	uint16	R
act_state 1	0x7531	Status register 1: bit 0 - 3 - user interface status bit 4 - 7 - previous user interface status bit 8 - 11 - fan status bit 12 - 15 - previous fan status	-	uint16	R
act_state 2	0x7532	Status register 2: bit 0 - 3 - pre-heating status bit 4 - 7 - previous pre-heating status bit 8 - 11 - last pre-heating status with fans running bit 12 - 15 - fan balance status	-	uint16	R
act_state 3	0x7533	Status register 3: bit 0 - 2 - reheat state bit 3 - 5 - previous reheat state bit 6 - filter state bit 7 - previous filter state bit 8 - UI state change flag bit 9 - fan state change flag bit 10 - preheat state change flag bit 11 - superheat status change flag bit 12 - filter status change flag bit 13 - max. fan imbalance overflow flag according to ventilation request bit 14 - max. fan imbalance overflow flag according to reheating request bit 15 - last frost protection direction change flag	-	uint16	R
AQS_state	0x7534	Air quality sensor status register: bit 0 - CO2 sensor error bit 1 - RH sensor error bit 2 - active ventilation request from AQS bit 3 - active CO2 level bit 4 - active RH level bit 5 - active Radon level	- - - - - - 0.01 V	uint16	R

List of input registers					
Inp. register	***Address of register***	***Description***	***of unit***	***Format***	***R/W***
		bit 6- 15 - required fan speed according to AQS values			
set_FAN_SPEED	0x7535	Required fan voltage (according to user and AQS settings)	0.01 V	uint16	R
set_T_ROOM	0x7536	Required temperature	0.1°C	int16	R
act_CO2	0x7537	Current CO2 sensor value	1ppm	uint16	R
act_RH	0x7538	Current RH sensor value	0.1%RH	uint16	R
act_RADON	0x7539	Current Radon sensor value	1Bq/m3	uint16	R
act_T_ROOM	0x753A	Room temperature sensor: bit 0 - 13 - temperature sensor value bit 14 - 15 - sensor status	 0.1°C -	int16	R
act_T_EXHAUST	0x753B	Frost protection temperature sensor: bit 0 - 13 - temperature sensor value bit 14 - 15 - sensor status	 0.1°C -	int16	R
FAN1.act_AO	0x753C	Current fan voltage 1	0.01 V	uint16	R
FAN1.act_tacho	0x753D	Fan feedback 1: bit 0 - tacho input status bit 1 - 15 - tacho input error timer	 1-OK, 0-NG 0.01s	uint16	R
FAN2.act_AO	0x753E	Current fan voltage 2	0.01 V	uint16	R
FAN2.act_tacho	0x753F	Fan feedback 2: bit 0 - tacho input status bit 1 - 15 - tacho input error timer	 1-OK, 0-NG 0.01s	uint16	R
timer.act_cool_preheat	0x7540	Preheat precooling timer: bit 0 - timer enable bit 1 - 15 - timer value	 - 1s	uint16	R
timer.act_cool_postheat	0x7541	Reheat aftercooling timer: bit 0 - timer enable bit 1 - 15 - timer value	 - 1s	uint16	R
timer.act_summer_mode	0x7542	Summer mode duration timer: bit 0 - timer enable bit 1 - 15 - timer value	 - 1s	uint16	R
timer.act_boost_mode	0x7543	Summer mode duration timer: bit 0 - timer enable bit 1 - 15 - timer value	 - 1s	uint16	R
timers_1	0x7544	Boost signalling and preheat relay timers: bit 0 - enable boost signalling timer bit 1 - 7 - boost signalling timer value bit 8 - preheat relay timer enable bit 9 - 15 - preheat relay timer value	 - 1s - 1s	uint16	R
timers_2	0x7545	Reheat and AQS protection time relay timers bit 0 - enable the reheat relay timer bit 1 - 7 - value of the reheat relay timer bit 8 - enable the AQS protection time timer bit 9 - 15 - value of the AQS protection time timer	 - 1s - 1s	uint16	R
act_FILTER_ELAPSED_TIME	0x7546	Filter active time timer	1h	uint16	R

List of input registers					
Inp. register	***Address of register***	***Description***	***of unit***	***Format***	***R/W***
act_ui_timer	0x7547	Timing of user interface state transitions: bit 0 - timer enable bit 1 - 15 - timer value	- 0.01s	uint16	R
act_preheat_timer High	0x7548	Upper register of the preheat state timer: bit 0 - timer enable bit 1 - 15 - lower 15 bits of the timer value	- 1s	uint16	R
act_preheat_timer Low	0x7549	Lower register of the preheat timer: bits 0 - 15 - upper 16 bits of the timer value	1s	uint16	R

List of UCFG retention registries					
Ret. registry	***Address of register***	***Description***	***range***	***Default value***	***R/W***
front panel	0x9C40	Parameters set on the front panel: bit 0 - flag power on bit 1 - flag AQS auto/manual bit 2 - flag active summer mode bit 3 - flag summer mode auto off bit 4 - flag active boost mode bit 5 - flag touch button lock bit 6 - 9 - fan level bit 10 -15 - temperature level	1-ON/ 0-OFF 1-AU/0-MAN 1-YES/0-NO 1-YES/0-NO 1-YES/0-NO 1-LOCK/0-NO - -	2	R/W R/W R/W R R/W R/W R/W R/W
set_CO2	0x9C41	CO2 value at which the fans switch on (1ppm)	600-1,000	800	R/W
set_RH	0x9C42	RH value at which the fans switch on (0.1%RH)	500-750	650	R/W
set_RADON	0x9C43	Radon value at which the fans switch on (1Bq/m3)	300 - 450	350	R/W

List of DCFG retention registries					
Ret. registry	***Address of register***	***Description***	***range***	***Default value***	***R/W***
bits	0x9C50	Bit setting items: bit 0 - 2 - not used bit 3 - flag automatic shutdown after reset bit 4 - flag permanently active fans at min speed bit 5 - 6- modbus baud rate setting bit 7 - modbus parity setting bit 8 -15 modbus address setting	 - 1-YES/0-NO 1-YES/0-NO - 1-NONE/0-EVE 1 - 247	 3 0 1	 R/W R/W R/W R/W R/W R/W
corr_T_ROOM	0x9C51	Room temperature correction (0.1°C)	-100 - 100	0	R/W
set_SUMMER_ MODE_DURATION	0x9C52	Summer mode duration (1s)	3600 - 32400	28800	R/W
set_BOOST_ MODE_DURATION	0x9C53	Boost mode duration (1s)	300 - 3600	600	R/W
set_FILTER_ LIFETIME	0x9C54	Filter lifetime (1h)	2200 - 8800	4400	R/W
set_POSTHEAT_ FAN1_SPEED	0x9C55	FAN1 fan voltage at reheat demand (0.01V)	XR100 250 - 400 XR250 250-500	250	R/W

Detailed description of the meaning of individual registers

Description of input registers

- ***act_state 1***

  the UI and previous UI state bits can take on values:
  0 = UI_STATE_OFF - unit off
  1 = UI_STATE_OFF_COOLING - unit off preheating or reheating cooling is in progress
  2 = UI_STATE_FAN1_ERROR - fan error 1
  3 = UI_STATE_FAN2_ERROR - fan error 2
  4 = UI_STATE_T_ROOM_ERROR - room error sensor
  5 = UI_STATE_T_EXHAUST_ERROR - frost sensor error
  6 = UI_STATE_ACTIVE_LOCK - active touch button lock
  7 = UI_STATE_ACT_DEACT_LOCK - touch button lock deactivation status
  8 = UI_STATE_SHOW_SETTINGS - wake-up status of the controller, displays the current settings 9 = UI_STATE_SET_FAN - fan range setting mode
  10 = UI_STATE_SET_HEAT - temperature range setting mode
  11 = UI_STATE_RUN normal power ON mode if no button is pressed
  12 = UI_STATE_SERVICE_MENU - service menu mode fan status bits and previous fan status bits can take on values:
  0 = FAN_STATE_OFF - the unit is OFF
  1 = FAN_STATE_OFF_COOL - the unit is OFF, fans are preheating or reheating
  2 = FAN_STATE_FAN1_ERROR - fan 1 error
  3 = FAN_STATE_FAN2_ERROR - fan 2 error
  4 = FAN_STATE_ACTIVE - fans active in normal mode
  5 = FAN_STATE_ACTIVE_ANTIFREEZE - fans active in antifreeze mode
  6 = FAN_STATE_INACTIVE - inactive fans in normal mode
  7 = FAN_STATE_INACTIVE_ANTIFREEZE - inactive fans in antifreeze mode
  8 = FAN_STATE_SUMMER_MODE - fans in summer mode
  9 = FAN_STATE_BOOST_MODE - fans in Boost mode

- ***act_state 2***

  bits of the preheat state, the previous preheat state and the last preheat state with active fans can take values:
  0 = PREHEAT_STATE_OFF - the unit is OFF
  1 = PREHEAT_STATE_OFF_COOL, the unit is OFF, preheating aftercooling
  2 = PREHEAT_STATE_T_ROOM_ERROR - room temperature sensor error
  3 = PREHEAT_STATE_T_EXHAUST_ERROR - frost sensor error
  4 = PREHEAT_STATE_ACTIVE_LOW - first level of frost protection with active preheating
  5 = PREHEAT_STATE_ACTIVE_MED1 - second level of frost protection with fan balancing
  6 = PREHEAT_STATE_ACTIVE_MED2 - third level of frost protection with FAN 1 switched off
  7 = PREHEAT_STATE_ACTIVE_HIGH - highest level of protection with both FAN 1 and FAN 2 switched off
  8 = PREHEAT_STATE_INACTIVE - inactive frost protection, active fans
  9 = PREHEAT_STATE_INACTIVE_FAN_OFF - inactive frost protection, inactive fans
  10 = PREHEAT_STATE_COOL - preheat aftercooling, inactive fan requirement
  11 = PREHEAT_STATE_COOL_TO_MED2 - preheat aftercooling when FAN 1 is off, transition to MED2

12 = PREHEAT_STATE_COOL_TO_INACTIVE - aftercooling between LOW and INNACTIVE states
Bits of the fan unbalance degree symbolize the voltage difference between FAN 1 and FAN 2 with a step of 0.5 V.

- ***act_state 3***

  bits of the reheat state and the previous reheat state can take values:
  0 = POSTHEAT_STATE_OFF - the unit is OFF
  1 = POSTHEAT_STATE_OFF_COOL - the unit is OFF, reheat cooling
  2 = POSTHEAT_STATE_T_ROOM_ERROR - room sensor error
  3 = POSTHEAT_STATE_T_EXHAUST_ERROR - antifreeze sensor error
  4 = POSTHEAT_STATE_ACTIVE - active reheat
  5 = POSTHEAT_STATE_ACTIVE_ANTIFREEZE_MED2 - active reheat in antifreeze 3. Stages
  6 = POSTHEAT_STATE_INACTIVE - inactive reheat
  7 = POSTHEAT_STATE_COOL - reheat aftercooling

- ***AQS_state***

  bits 0 and 1 - indicate the error state of the AQS if the sensor is enabled (according to the unit type XX,CO,RH,CR or in the service menu)
  bit 2 - indicates that one of the AQS is active (activates the fans, if automatic mode)
  bit 3 - active CO2 level - the sensor value exceeded the level required to activate the fan
  bit 4 - active RH level - the sensor value exceeded the level required to activate the fan
  bit 5 - active Radon level - the sensor value exceeded the level required to activate the fan
  bit 6-15 - calculated rate of voltage demand of the fans according to the AQS

- ***act_T_ROOM and act_T_EXHAUST***

  bit 0 - 13 - represent the value of the given temperature
  bit 14 - 15 - sensor status - 0 = ok, 1 = sensor disconnected, 2 = sensor shorted

Description of UCFG retention registers

Front panel

bit 0 - flag power on - indicates whether the unit is on or off (1 = ON, 0 = OFF). The unit can be switched on or off remotely by writing. bit 1 - flag AQS auto/manual - indicates the currently selected fan mode (1 = automatic, according to AQS, 0 = manual). You can change the mode by enrolling here. bit 2 - flag of active summer mode - indicates active summer mode (1 = summer mode active, 0 = summer mode inactive). The summer mode can be activated/deactivated by writing (if the conditions for activation are met) bit 3 - flag summer mode auto off - signals the automatic termination of the summer mode. Not for writing. When writing, leave set to the current value

bit 4 - active boost mode flag - indicates active boost mode (1 = active boost mode, 0 = inactive boost mode). By writing it is possible to activate / deactivate (if the conditions are met)

bit 5 - touch button lock flag - indicates an active "child" lock (1 = button lock active, 0 = button lock inactive). Can be changed by entry

bit 6 - 9 - fan level level can be changed by writing - indicates the currently selected fan level. Can be changed by entry.

Do not set a value of bit 8 for Boost mode. Boost mode is activated by bit 4

Bit 6-9 values	XROOM100 fan voltage	XROOM250 fan voltage
0	0 V	0 V
1	2 V	2.5 V
2	2.4 V	3.1 V
3	2.8 V	3.7 V
4	3.3 V	4.3 V
5	3.7 V	5.1 V
6	4.1 V	5.5 V
7	4.5 V	6 V
8 - BOOST mode	10 V	10 V

bit 10-15 - Temperature level - For units of E type, this indicates the current selected temperature level. Can be changed by entry. The table below shows the approximate temperature values assigned to each level

Bit 10-15 values	Corresponding temperature set point
0	Heating off
1	5°C
2	19°C
3	19.5°C
4	20°C
…	…
10	23°C
11	23.5°C
12	24°C
13	28°C

Description of DCFG retention registers

bits

bit 0 - 2 - not used - When writing, replace with 0 bit 3 - auto shutdown flag after reset - indicates whether the unit will automatically shut down or restore the previous state in case of an unexpected reset. (1 = automatically switch off, 0 = previous state). Can be changed by entry
bit 4 - flag of permanently active fans to min speed - it indicates the mode when fans cannot be switched off. The fans always run at minimum speed (1 = fans always at minimum speed, 0 = fans off). Can be changed by entry.
bit 5 - 6- modbus baud rate setting - modbus baud rate setting (0 = disabled value, 1 = 4800, 2 = 9600, 3 = 19200). Can be changed by entry
bit 7 - modbus parity setting can be changed by writing - indicates the modbus parity setting (0 = even parity, 1 = no parity). Can be changed by entry.
bit 8 -15 of the modbus address setting can be changed by writing - it indicates the current modbus address of the unit.

For bits 5 to 15, in the case of entry, the unit immediately changes operation according to the new parameters. Therefore, if any of the parameters are changed, the unit will typically stop communicating until the network master changes the parameters.

Description, syntax and example of the Modbus functions used

(0x04) Read input registers Function

This function is used to read the contents of a contiguous block of input registers. The request specifies the address of the first register and the number of registers. In the response, two bytes correspond to each register corresponds.

1) Request PDU

Modbus functions 0x04 1 B	1st register address see List of input reg. 2 B	No. of registers 1 to max. 13 2 B

Example of reading the act_CO2 and act_RH input registers:

0x04	act_CO2		No. of registers = 2	
	0x75	0x37	0x00	0x02

2) Response PDU

Modbus functions 0x04 1 B	Number of bytes 2*N 1 B	Register states 2*N B

N = Number of registers (see Request PDU)

Example of the response to reading the act_CO2 and act_RH input registers:

0x04	Number of bytes	CO2 = 980 ppm		RH = 335 ‰	
	0x04	0x03	0xD4	0x01	0x4F

3) Exception Response PDU

Modbus function |0x80 0x84 1 B	Error code 1, 2, 3 or 4 1 B

(0x03) Read retention registers Function

This function is used to read the contents of a contiguous block of retention registers. The request specifies the address of the first register and the number of registers. In the response, two bytes correspond to each register corresponds.

1) Request PDU

Modbus functions 0x03 1 B	1st register address see List of retention reg. 2 B	No. of registers 1 to max. 13 2 B

Example of reading the set_CO2 and set_RH retention registers:

0x03	Set_CO2		No. of registers = 2	
	0x9C	0x41	0x00	0x02

2) Response PDU

Modbus functions 0x03 1 B	Number of bytes 2*N 1 B	Register states 2*N B

N = Number of registers (see Request PDU)

Example response to reading the set_CO2 and set_RH retention registers:

0x03	Number of bytes	CO2=750		RH=550	
	0x06	0x02	0xEE	0x02	0x26

3) Exception Response PDU

Modbus function |0x80 0x83 1 B	Error code 1, 2, 3 or 4 1 B

(0x10) Write multiple retention registers Function

This function is used to write a contiguous block of retention registers. The request specifies the address of the first register to be written, the number of registers, and the values to be written. The normal response contains the starting address and the number of registers written.

1) Request PDU

Modbus functions 0x10 1 B	1st register address see List of retention reg. 2 B	No. of registers 1 to max. 11 2 B	Number of bytes 2*N 1 B	Register states 2*N B

N = Number of registers

Example of writing set_FILTER_LIFE_TIME retention registers

0x10	Set_FIL_LT		No. of registers = 1		Number of bytes = 2	8800h	
	0x9C	0x55	0x00	0x01	0x04	0x22	0x60

2) Response PDU

Modbus functions 0x10 1 B	1st register address see Request PDU 2 B	No. of registers see Request PDU 2 B

N = Number of registers

Example of a response to writing the set_FILTER_LIFE_TIME retention registers:

0x10	Set_FIL_LT		No. of registers = 1	
	0x9C	0x55	0x00	0x01

3) Exception Response PDU

Modbus function |0x80 0x90 1 B	Error code 1, 2, 3 or 4 1 B