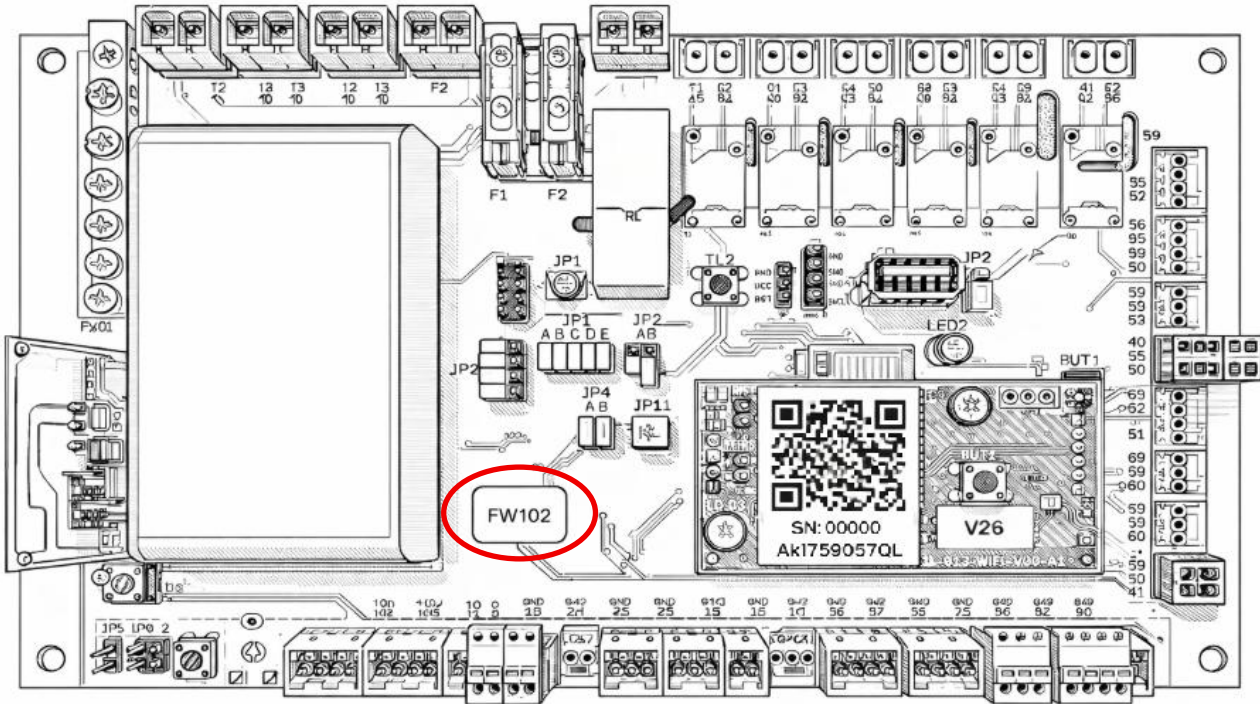


Contents

| | |
|--|----|
| Introduction | 2 |
| General description of the Modbus protocol | 3 |
| Communication modes..... | 3 |
| Modbus Data Unit (PDU) | 3 |
| Modbus function codes | 4 |
| General Modbus address space..... | 5 |
| Modbus on RS485..... | 5 |
| RS485 Bus Default Settings | 5 |
| Connecting the wire pins to the RJ45 connector for connection to XCONT-HUB | 5 |
| Modbus RTU frame structure on RS485 bus | 6 |
| CRC calculation | 6 |
| List of Modbus functions | 7 |
| Modbus address space for WIFI units..... | 7 |
| List of input registers | 7 |
| DCFG registers of configurability bitwise..... | 9 |
| List of UCFG storage registers..... | 10 |
| List of DCFG storage registers | 11 |
| A more detailed description of the meaning of individual registers | 13 |
| Description of input registers | 13 |
| - UI Status..... | 13 |
| - Condition Vent..... | 13 |
| - AFP status | 13 |
| - AQS_state | 14 |
| - Selection of the AQS type according to HW jumpers (jumpers)..... | 14 |
| Description of UCFG storage registers..... | 15 |
| Front panel..... | 15 |
| Description of DCFG storage registers | 15 |
| Bits | 15 |
| Description, syntax and example of use of used Modbus functions | 16 |
| (0x04) Read Input Registers Function | 16 |
| (0x03) Read Retention Registers function | 17 |
| (0x10) Write Multiple Retention Registers function..... | 18 |

Introduction

This document is used to describe the Modbus protocol used for units (XF2; XH3) with WIFI control (3001-013). The version of this manual is intended for units with **Firmware version 102 and above**. The FW version number is indicated on the self-adhesive label located on the PCB and in the input register of the FW version.



To get started, here is some useful information to help you troubleshoot any problems:

Only those registers that are available on the drive can be read from the unit. Otherwise, the unit responds with an error response with error code 0x02 – Illegal Data Address.

The unit requires a certain amount of time to process the request, so it is necessary to take into account 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. Normally, the response time is around 4ms

In case the unit does not communicate, make sure that the frames you send are correct and check that you keep pauses of at least 4ms on the communication bus for correct detection of the ends of the frame.

The bus operates in the so-called A-Z mode. **Half-duplex**. This means that it is not able to accept further requests until it comes to responding to the previous modbus framework.

To check or verify the correctness of **modbus crc calculations**, it is possible to use an on-line calculator:

<https://www.lammertbies.nl/comm/info/crc-calculation.html>

The calculator must be switched to entering HEX characters and the subsequent result CRC-16 (Modbus) 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) are present on the bus. Communication is always initiated by the master device. The slave only responds to the requirements of the master device. Modbus uses Big-endian data representation. This means that for items larger than 1 B, the highest byte is sent first, and the lowest byte is sent afterwards.

Communication modes

Unicast mode:

The master addresses **one specific slave device** using its Modbus address. The slave processes the message and responds.

Modbus Data Unit (PDU)

| Modbus function | Date |
|-----------------|--------|
| 1 B | N* 1 B |

The Modbus protocol defines three basic types of PDUs:

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

The **Modbus function** field contains the code of the given Modbus function. The data field then according to the given Modbus address function, number of variables, variable values and others

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

The **Modbus function** field contains the **same value** as in the received Request PDU. The data section then contains, depending on the given Modbus function, operating values, read inputs, coils, etc....

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

The **Modbus function** field contains the **Modbus function value from the Request PDU plus 0x80** to indicate a failure. The data section then **identifies the error**.

Error codes in Exception Response PDU

| Code | Function code type | Meaning |
|------|--|---|
| 0x01 | Illegal Modbus Function | The required Modbus function is not supported by the server (unit) |
| 0x02 | Illegal data address | The specified address (coil, register...) is outside the server-supported range |
| 0x03 | Illegal data value | The data being transmitted is invalid |
| 0x04 | Device failure | An unrecoverable error occurred while executing the request |
| 0x05 | Confirmation | Code intended for use in programming. The server reports receipt of a valid request, but it will take longer to execute |
| 0x06 | The device is busy | Code intended for use in programming. The server is busy executing a long-running command. |
| 0x08 | Memory parity error | Code intended to be used when working with files. The server encountered a parity error when trying to read the file |
| 0x0A | Gateway - Transmission path unavailable | Code designed to work with the gateway. The gateway is not able to allocate an internal transfer path from the input port to the output port. It is probably overloaded or set incorrectly. |
| 0x0B | Gateway - Destination device is not responding | Code designed to work with the gateway. The destination device is not responding, it may not be present. |

Modbus function codes

- 1) **Public feature 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. The uniqueness of the code is not guaranteed.

Code ranges of Modbus functions

| 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 |

General Modbus address space

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

| Table | Description | Access | Address space (not a requirement) |
|-------------------|-------------|----------------|-----------------------------------|
| Discrete inputs | 1-bit | Read-only | 0x2710 to 0x4E1F |
| Coils | 1-bit | Read and write | 0x0000 to 0x270F |
| Input registers | 16-bit | Read-only | 0x7530 to 0x9C3F |
| Storage registers | 16-bit | Read and write | 0x9C40 to 0xC34F |

Modbus on RS485

RS485 Bus Default Settings

The Modbus RTU protocol defines the default settings of a serial line 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 rate 4800, 9600 and the option of no parity.

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



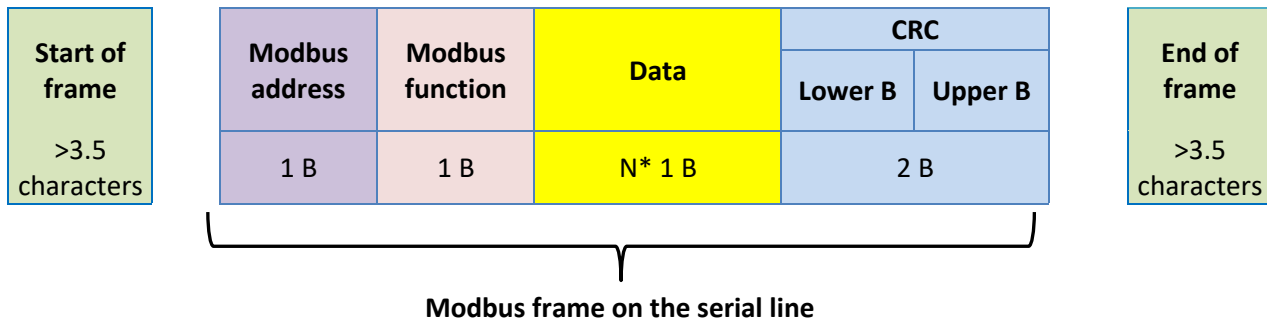
Pins 1, 2, 7, 8 must be left unplugged

Pins 3, 4 – RS485 bus data wire A

pins 5, 6 – RS485 bus data wire B

Modbus RTU frame structure on RS485 bus

In Modbus RTU mode, 1 B is composed of two four-bit hexa characters. The transmission of the Modbus frame is started and ends with a pause on the bus longer than 3.5 characters. During the transmission of the frame, the spaces between each character must not be larger than 1.5 characters.



Addressing Slave Devices:

| Address | Meaning |
|------------|----------------------------|
| 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 left.

CRC calculation

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

1. Initialize the 16-bit CRC register on 0xFFFF.
2. We perform XOR of the first 8 bits of the frame with the bottom byte of the CRC register and store the result in the CRC register.
3. Move the CRC register by 1 b to the right (towards the LSB), fill in the MSB CRC register by 0. We capture and evaluate the lowest bit that fell out due to the shift.
4. If this bit was equal to 1, we perform an XOR between the CRC register and the value of 0xA001 (generating polynomial= $1+x^2+x^{15}+x^{16}$). We save the result again in the CRC register.
5. Repeat steps 3 and 4 until eight CRC register shifts are performed.
6. We perform XOR of the next 8 bits of the frame with the bottom flat of the CRC register and repeat steps 3 through 5.
7. We continue like this until the last byte of the frame.
8. The resulting CRC value of the calculation is stored in the CRC register.
9. When placing a CRC value in a Modbus frame, it is necessary to swap the upper and lower bytes of the CRC register (see the structure of the Modbus RTU frame on the serial line).

List of Modbus functions

| Intended use | Function code | Command | Public/User |
|-----------------|---------------|----------------------------------|-------------|
| In. Registers | 0x04 | Read input registers | Public |
| Hold. Registers | 0x03 | Read holding registers | Public |
| | 0x10 | Write multiple holding registers | Public |

Modbus address space for WIFI units

List of input registers

| Input register | Address register | Description | Units | Format | R/W |
|----------------------|------------------|--|------------------|--------|-----|
| FW version | 0x7530 | Firmware Version in the Unit | - | uint16 | R |
| UI Status | 0x7531 | User Interface States | - | uint16 | R |
| Vent status | 0x7532 | Fan states | - | uint16 | R |
| AFP status | 0x7533 | Frost protection status | - | uint16 | R |
| AQS_state | 0x7534 | Air quality sensor status register: bit 0, 1- total AQS bit 2, 3 – AQS status 2 bit 4, 5 – AQS status 1 bit 6- 15 – unused | - | uint16 | R |
| fan_aqs_flow | 0x7535 | Required flow value according to AQS values | 0.1 m3/h | uint16 | R |
| fan1_vent voltage | 0x7536 | Required fan voltage1 | 0.01V | uint16 | R |
| fan2_vent voltage | 0x7537 | Required fan voltage2 | 0.01V | uint16 | R |
| fan1_poh_cd_voltage | 0x7538 | Required Voltage Aftercooling, Fan 1 | 0.01V | uint16 | R |
| fan2_poh_cd_voltage | 0x7539 | Required Voltage Aftercooling, Fan 2 | 0.01V | int16 | R |
| fan1_preh_cd_voltage | 0x753A | Required voltage of aftercooling, preheating fan 1 | 0.01V | uint16 | R |
| fan2_preh_cd_voltage | 0x753B | Required voltage of aftercooling, preheating fan 2 | 0.01V | uint16 | R |
| fan_req_flow | 0x753C | Required flow rate (based on AQS and user request) | 0.1m3/h | uint16 | R |
| act_AQS1 | 0x753D | Current value of the AQS1 sensor | 1ppm/0.1%/1Bq/m3 | uint16 | R |
| act_AQS2 | 0x753E | Current value of the AQS2 sensor | 1ppm/0.1%/1Bq/m3 | uint16 | R |
| Te | 0x753F | Outdoor air temperature | 0.1°C | int16 | R |
| Ti | 0x7540 | Indoor air temperature | 0.1°C | int16 | R |
| T_antifreeze | 0x7541 | Frost protection sensor temperature | 0.1°C | int16 | R |
| FAN1.act_AO | 0x7542 | Current fan voltage 1 | 0.01V | uint16 | R |
| FAN1.stav | 0x7543 | Fan Status 1: 0 = OFF 1 = START 2 = ON 3 = ERROR | - | uint16 | R |

List of input registers

| Input register | Address register | Description | Units | Format | R/W |
|---------------------|------------------|--|-------|--------|-----|
| FAN2.act_AO | 0x7544 | Current fan voltage 2 | 0.01V | uint16 | R |
| FAN2.stav | 0x7545 | Fan status 2: 0 = OFF 1 = START 2 = ON 3 = ERROR | - | uint16 | R |
| preheater.cd_timer | 0x7546 | Preheating Cooldown Timer: bit 0 - 15 - timer value | 1s | uint16 | R |
| postheater.cd_timer | 0x7547 | Postheating Cooldown Timer: bit 0 - 15 - Timer Value | 1s | uint16 | R |
| timer.summer_mode | 0x7548 | Summer mode duration timer: bit 0 - 15 - timer value | 1s | uint16 | R |
| timer.boost_mode | 0x7549 | Boost mode duration timer: bit 0 - 15 - timer value | 1s | uint16 | R |
| preheater.timer | 0x754A | Preheat relay timer: bit 0 - 15 - preheat relay timer value | 1s | uint16 | R |
| postheater.timer | 0x754B | Relay Timer postheat: bit 0 - 15 Reheating Relay Timer Value | 1s | uint16 | R |
| FILTER_ELAPSED_TIME | 0x754C | Filter Active Time Timer | 1h | uint16 | R |
| aqs.guard_timer | 0x754D | Timing of the AQS: bit state protection time 0 - 15 - timer value | 1s | uint16 | R |
| afp.timer High | 0x754E | Upper register of the preheat state timer: bit 0 - 15 - lower 16 bits of the timer value | 1s | uint16 | R |
| afp.timer Low | 0x754F | Lower register of the preheat state timer: bits 0 - 15 - upper 16 bits of the timer value | 1s | uint16 | R |
| act_startup_timer | 0x7550 | Timing of the protection interval after restarting the device: bit 0 - 15 - timer value | 1s | uint16 | R |
| Relay | 0x7551 | Relay state bits: bit 0 - relay status preheat bit 1 - relay status postheat bits 2 – 15 – unused | - | uint16 | R |
| Status 1 | 0x7552 | Status register 1 bit 0 – err_FAN1 (0=OK, 1=error) bit 1 – err_FAN2 (0=OK, 1=error) bit 2 – err_Ti (0=OK, 1=error) bit 3 – err_Te (0=OK, 1=error) bit 4 – err_Ta (0=OK, 1=error) bit 5 – err_max_Te – exceeding max channel temperature (0=OK, 1=error) bit 6 – err_AQS1 (0=OK, 1=error) bit 7 – err_AQS2 (0=OK, 1=error) bit 8 – Filter (0=replace, 1=OK) bit 9 – AFP max (0=no, 1= yes) bit 10 – 15 - reserve | | uint16 | R |
| fan_max_flow | 0x7553 | Maximum fan flow rate (flow rate at control voltage 10V) | | uint16 | R |
| Jumpers 1 | 0x7554 | HW jumpers (1=JP set, 0= JP unmounted): | - | uint16 | R |

| List of input registers | | | | | |
|--------------------------------|-------------------------|---|--------------|---------------|------------|
| Input register | Address register | Description | Units | Format | R/W |
| | | bit 0 – JP1 A bit 1 – JP1 B bit 2 – JP1 C bit 3 – JP1 D bit 4 – JP1 E bit 5 – JP2 A bit 6 – JP2 B bit 7 – JP3 A bit 8 – JP3 B bit 9 – JP3 C bit 10 – JP3 D bit 11 – JP4 A bit 12 – JP4 B bit 13 – JP5 A bit 14 – JP5 B bit 15 – JP6 | | | |
| Digital inputs 1 | 0x7555 | Digital input values (1= active, 0= inactive): bit 0 – DI-AUTO bit 1 – EXT1 bit 2 – EXT2 bit 3 – EXT3 bit 4 – EXT4 bit 5 – RF IN1 bit 6 – RF IN2 bit 7 – RF IN3 bit 8 – RF IN4 bit 9 – TL1 bit 10 – WIFI present bit 11 – BOOTLOADER 1 bit 12 – BOOTLOADER 2 bit 13 to 15 – reserve | - | uint 16 | R |
| | | | | | |

| DCFG registers of configurability bitwise | | | | | |
|--|-------------------------|--|--|---------------|------------|
| Input Register | Address register | Description | Scope | Format | R/W |
| Register 1 | 0x7700 | Bit items: bit 0 - register adjustability 0x9C50 bit 1 - register adjustability 0x9C51 bit 2 - register adjustability 0x9C52 bit 3 - register adjustability 0x9C53 bit 4 - register adjustability 0x9C54 bit 5 - register adjustability 0x9C55 bit 6 - register adjustability 0x9C56 bit 7 - register adjustability 0x9C57 bit 8 - register adjustability 0x9C58 bit 9 - register adjustability 0x9C59 | 1-YES/0-NO 1-YES/0-NO 1-YES/0-NO 1-YES/0-NO 1-YES/0-NO 1-YES/0-NO 1-YES/0-NO 1-YES/0-NO 1-YES/0-NO | uint16 | R |

DCFG registers of configurability bitwise

| Input Register | Address register | Description | Scope | Format | R/W |
|----------------|------------------|---|------------|--------|-----|
| | | bit 10 - register adjustability 0x9C5A bit 11 - register adjustability 0x9C5B bit 12 - register adjustability 0x9C5C bit 13 - register adjustability 0x9C5D bit 14 - register adjustability 0x9C5E bit 15 - register adjustability 0x9C5F | | | |
| Register 2 | 0x7701 | Bit items: bit 0 - register adjustability 0x9C60 bit 1 - register adjustability 0x9C61 bit 2 - register adjustability 0x9C62 bit 3 - register adjustability 0x9C63 bit 4 - register adjustability 0x9C64 bit 5 - register adjustability 0x9C65 bit 6 - register adjustability 0x9C66 bit 7 - register adjustability 0x9C67 bit 8 - register adjustability 0x9C68 bit 9 - register adjustability 0x9C69 bit 10 - register configurability 0x9C6A bit 11 - register configurability 0x9C6B bit 12-15 - reserve | 1-YES/0-NO | uint16 | R |

List of UCFG storage registers

| Hold. Register | Address 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 lock touch button bits 6 - 9 - fan speed 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/WR R/W R/W R/W R/W |
| set_CO2 | 0x9C41 | CO2 value at which fans switch (1ppm) | 400-1000 | 800 | R/W |
| set_RH | 0x9C42 | RH value at which fans switch on (0.1%RH) | 450-750 | 650 | R/W |
| set_Radon | 0x9C43 | Radon value at which fans switch on (1Bq/m3) | 100-450 | 350 | R/W |

| List of DCFG storage registers | | | | | |
|---------------------------------------|-------------------------|---|---|----------------------|--|
| Hold. Register | Address register | Description | Scope | Default Value | R/W |
| Bits | 0x9C50 | Bit Settings: bit 0 - 2 - Not Used bit 3 - Automatic shutdown flag after reset bit 4 - Flag permanently active fans at min speed bit 5 - 6- Modbus Baud Rate 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 value 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_FAN_SPEED | 0x9C53 | Fan voltage in Boost mode (0.01V) | XF: 500 – 800 XH: 600 - 900 | XF: 800 XH: 900 | R/W |
| set_BOOST_MODE_FAN_FLOW | 0x9C54 | Fan flow in Boost mode (1m3/h) | XF: xx – xx XH: xx - xx | XF: xx XH: xx | R/W |
| set_BOOST_MODE_DURATION | 0x9C55 | Duration of Boost Mode (1s) | 30 – 3600 | 60 | R/W |
| set_FAN_OFFSET_STATIC | 0x9C56 | Fan Distribution Settings (%) | 0 – 35 | 0 | R/W |
| set_FILTER_LIFETIME | 0x9C57 | Filter life (1h) | 2200 – 8800 | 4400 | R/W |
| set_NOMINAL_FLOW | 0x9C58 | Nominal Unit Flow (1m3/h) | Min-max units | | R/W |
| set_DIRECTION_L/R | 0x9C59 | Setting the unit left/right/according to the HW switch | 0 = according to HW switch (switch set to left) 1 = according to HW switch (switch set to right) 2 = left 3= right | 0 | R/W |
| Factory_reset | 0x9C5A | Writing 1 resets the configuration to its default values | 1 | 0 | R/W |
| AQS1_type | 0x9C5B | Connected sensor type AQS1 | 0 = according to HW 1 = CO2 2 = RH 3 = RADON | 0 | R/W |
| AQS2_tyte | 0x9C5C | Connected sensor type AQS2 | 0 = according to HW 1 = CO2 2 = RH 3 = RADON | 0 | R/W |
| Filter reset | 0x9C5D | Filter reset (write 1 to reset the filter) | 1 | 0 | R/W |

| List of DCFG storage registers | | | | | |
|---------------------------------------|--------|---|-----|---|-----|
| min_airflow | 0x9C5E | Setting the Low Flow Level for EFF/EHF Units | 0-7 | 3 | R/W |
| max_airflow | 0x9C5F | High Flow Level Setting for EFF/EHF Units | 0-7 | 7 | R/W |
| nominal_flow_by_EXT4_disable | 0x9C60 | Disable Nominal Flow Switching Using EXT4: 0 – Enabled 1 – Disabled | 0,1 | 0 | R/W |

A more detailed description of the meaning of individual registers

Description of input registers

- **UI Status**

UI Status Register may take values:

- 0 = UI_STATE_OFF – Unit Off
- 1 = UI_STATE_SOFT_OFF – Unit Apparently Off, Running at Minimum Speed
- 2 = UI_STATE_ON_SLEEP - Normal Power ON Mode If No Button Is Pressed
- 3 = UI_STATE_ON_WAKEUP – Normal ON Mode, Controller Wake Status, Displays Current Settings
- 4 = UI_STATE_ON_ERR – Unit Error Status
- 5 = UI_STATE_ACTIVE_LOCK – Touch Button Lock Active
- 6 = UI_STATE_LOCK_CHANGE – Status Activating/Deactivating the Touch Button Lock
- 7 = UI_STATE_SET_FAN – Fan Range Setting Mode
- 8 = UI_STATE_DEFAULT_AIRFLOW – Fan Range Setting Status for Default Airflow Function
- 9 = UI_STATE_SERVICE_MENU – Service Menu Mode
- 10 = UI_STATE_CUST_MENU_BOOST_FLOW - User Menu with Boost Mode Flow Setting
- 11 = UI_STATE_CUST_MENU_BOOST_TIME - User Menu with Boost Mode Duration Setting
- 12 = UI_STATE_CUST_MENU_FAN_OFFSET - User Menu with Distribution Settings fans
- 13 = UI_STATE_CUST_MENU_TEMP_AUTO_BYPASS - User menu with temperature set settings for the automatic bypass function

- **Vent status**

The ventilation status can be valued:

- 0 = VENT_STATE_OFF – Inactive ventilation
- 1 = VENT_STATE_SOFT_OFF - Ventilation mode when fans run to a minimum during the unit is switched off
- 2 = VENT_STATE_MAN – Manual ventilation mode
- 3 = VENT_STATE_AUTO – Automatic ventilation mode according to AQS
- 4 = VENT_STATE_SUMMER - Ventilation mode summer mode
- 5 = VENT_STATE_BOOST – Temporary intensive ventilation mode
- 6 = VENT_STATE_AFP – Ventilation mode during frost protection
- 7 = VENT_STATE_EXT1 – Special ventilation mode for Ekoair units when activating EXT1
- 8 = VENT_STATE_EXT2 – Special ventilation mode for Ekoair units when activating EXT2

- **AFP status**

The frost protection status can be valued:

- 0 = AFP_STATE_OFF – inactive frost protection
- 1 = AFP_STATE_STANDBY – temporarily inactive frost protection, waiting for fan request
- 2 = AFP_STATE_1_WO - AFP status with active preheating, fans without distribution, AFP deterioration
- 3 = AFP_STATE_1_BE - AFP status with active preheating, fans without distribution, AFP status improvement
- 4 = AFP_STATE_2A_WO - AFP status with FAN1 decrease, active preheating, AFP deterioration
- 5 = AFP_STATE_2A_BE - AFP status with FAN1 decrease, active preheating, AFP improvement
- 6 = AFP_STATE_2B_WO - AFP status with FAN2 increase, active preheating, AFP deterioration
- 7 = AFP_STATE_2B_BE – AFP status with increasing FAN2, active preheating, improvement in AFP status

8 = AFP_STATE_2B_BE_PH3 - AFP state with FAN2 increase, active preheat, AFP state improvement, after switching from state 3

9 = AFP_STATE_3_WO - Condition with FAN1 off, inactive preheat, AFP deterioration

10 = AFP_STATE_3_BE - Condition with FAN1 off, inactive preheat, improvement in AFP

11 = AFP_STATE_4 - Condition with both FAN1 and FAN2 off, Inactive preheating

- **AQS_state**

bits 0 and 1 – signal the overall state of AQS

bits 2 and 3 – signal the state of AQS2

bits 4 and 5 – signal the state of AQS1

Bits can take these values

0 = AQS_STATE_OFF = inactive ventilation requirement according to the AQS sensor

1 = AQS_STATE_ON = active ventilation requirement according to the AQS sensor

2 = AQS_STATE_ABSENT – AQS sensor absent

3 = AQS_STATE_ERR – AQS sensor error

- **Selection of the AQS type according to HW jumpers (jumpers)**

| | JP2 A | JP4 A |
|--------------|--------|-------|
| AQS1 = CO2 | 1 | 1 |
| AQS1 = RH | 0 | 1 |
| AQS1 = Radon | 0 or 1 | 0 |

| | JP2 B | JP4 B |
|--------------|--------|-------|
| AQS2 = CO2 | 1 | 1 |
| AQS2 = RH | 0 | 1 |
| AQS2 = Radon | 0 or 1 | 0 |

Description of UCFG storage registers

Front panel

- bit 0 - flag power on – signals 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 – signals the current selected fan mode (1 = automatic, according to AQS, 0 = manual). This setting can be changed via a command.
- bit 2 - flag of active summer mode – signals active summer mode (1 = summer mode active, 0 = summer mode inactive). By writing, the summer mode can be activated/deactivated (if the conditions for activation are met)
- bit 3 - flag summer mode auto off – signals the automatic end of summer mode. It is not used for writing. When writing, leave the current value set to
- bit 4 - flag of active boost mode – 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 – signals an active "child" lock (1 = button lock active, 0 = button lock inactive). This setting can be changed via a command.
- bit 6 - 9 – fan speed level - signals the current selected fan level. Can be changed. Do not set the value of 8 for Boost mode. Boost mode is activated by the 4
- bit 10 -15 – temperature level– For E type units, it signals the current selected temperature level. This setting can be changed via a command.

Description of DCFG storage registers

bits

- bit 0 - 2 - not used - Replace 0
- bit 3 when writing - automatic shutdown flag after reset - indicates whether the unit will automatically shut down or restore to its previous state in the event of an unexpected reset. (1 = automatically turn off, 0 = previous state). This setting can be changed via a command
- bit 4 - flag of permanently active fans to min speed - signals the mode when the fans cannot be turned off. The fans are still running at minimum speed (1 = fans always at minimum speed, 0 = turn off the fans). This setting can be changed via a command
- bit 5 - 6- modbus baud rate settings – modbus baud rate settings (0 = forbidden value, 1 = 4800, 2 = 9600, 3 = 19200). This setting can be changed via a command
- bit 7 - Modbus parity setting – indicates the Modbus parity setting (0 = even parity, 1 = no parity). This setting can be changed via a command
- bit 8 -15 modbus address settings – it signals the current modbus address of the unit.

For bits 5 to 15, in the event of writing, the unit immediately begins to behave according to the new parameters. In the event of a change in any of the parameters, the unit typically stops communicating until the parameters are changed by the network master.

Description, syntax and example of use of used Modbus functions

(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, each register corresponds to a pair of bytes.

1) Request PDU

| Modbus function | Address of the 1st register | Number of registers |
|-----------------|-----------------------------|---------------------|
| 0x04 | see List In. Reg. | 1 to max 13 |
| 1 B | 2 B | 2 B |

Example of reading input registers act_CO2 and act_RH:

| 0x04 | act_CO2 | Number of registers = 2 |
|------|-----------|-------------------------|
| | 0x75 0x37 | 0x00 0x02 |

2) Response PDU

| Modbus function | Number of bytes | Register statuses |
|-----------------|-----------------|-------------------|
| 0x04 | 2*N | |
| 1 B | 1 B | 2*N B |

N = Number of registers (see Request PDU)

Example of a response to read the input registers act_CO2 and act_RH:

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

3) Exception Response PDU

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

(0x03) Read Retention Registers function

Use this function to read the contents of a contiguous block of retention registers. The request specifies the address of the first registry and the number of registers. In the answer, each register corresponds to a pair of apartments.

1) Request PDU

| Modbus function | Address of the 1st register see List of hold. reg. | Number of registers |
|-----------------|---|---------------------|
| 0x03 | | 1 to max 13 |
| 1 B | 2 B | 2 B |

Example of reading the set_CO2 and set_RH retention registers:

| 0x03 | Set_CO2 | Number of registers = 2 |
|------|-----------|-------------------------|
| | 0x9C 0x41 | 0x00 0x02 |

2) Response PDU

| Modbus function | Number of bytes | Register statuses |
|-----------------|-----------------|-------------------|
| 0x03 | 2*N | |
| 1 B | 1 B | 2*N B |

N = Number of registers (see Request PDU)

Example of a response to read 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 | Error code |
|-----------------|------------|
| 0x80 | |
| 0x83 | 1,2,3 or 4 |
| 1 B | 1 B |

(0x10) Write Multiple Retention Registers function

Use this function 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. A normal response includes the start address and the number of registers written.

1) Request PDU

| Modbus function | Address of the 1st register see List of hold. reg. | Number of registers | Number of bytes | Register statuses |
|-----------------|---|---------------------|-----------------|-------------------|
| 0x10 | | 1 to max 11 | 2*N | |
| 1 B | 2 B | 2 B | 1 B | 2*N B |

N = Number of registers

Example of writing storage registers set_FILTER_LIFE_TIME

| | | | | |
|------|------------|-------------------------|---------------------|-----------|
| 0x10 | Set_FIL_LT | Number of registers = 1 | Number of bytes = 2 | 8800h |
| | 0x9C 0x55 | 0x00 0x01 | 0x02 | 0x22 0x60 |

2) Response PDU

| Modbus function | Address of the 1st register see Request PDU | Number of registers see Request PDU |
|-----------------|--|--|
| 0x10 | | |
| 1 B | 2 B | 2 B |

N = Number of registers

An example of a response to write storage registers set_FILTER_LIFE_TIME:

| | | |
|------|------------|-------------------------|
| 0x10 | Set_FIL_LT | Number of registers = 1 |
| | 0x9C 0x55 | 0x00 0x01 |

3) Exception Response PDU

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