

CONNECTING TO THE SMART HOME SYSTEM

Freshpoint
Freshpoint Eco

EN CONNECTION INSTRUCTION

CONTENTS

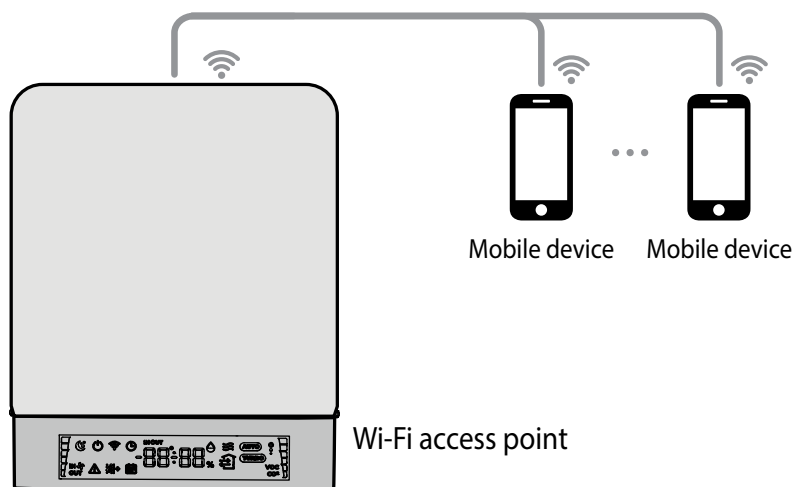
Connection and setup	2
Network parameters.....	3
Packet structure.....	4
Examples of using special commands in the data block.....	5
Examples of a complete packet.....	6
Table of parameters	7
Example of packet processing in C	12

CONNECTION AND SETUP

Example 1: diagram of direct connection of the fan to the BMS Smart House system without a router.

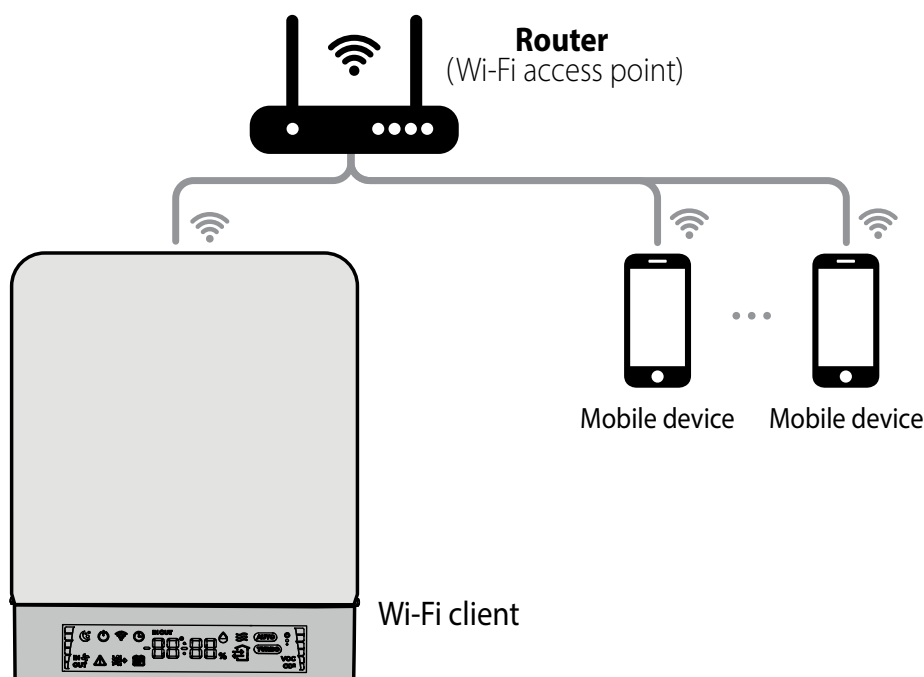
Configure the fan to operate via Wi-Fi as an access point (see the fan user manual).

Note: the maximum number of control devices that can be connected is eight.



Example 2: Connection diagram using a router with one Wi-Fi access point.

The fan, smartphones, and the BMS Smart House system are connected to the router's Wi-Fi access point.



NETWORK PARAMETERS

Data is exchanged over the UDP transport protocol (broadcasting is supported).

IP address of the device:

- 192.168.4.1 - when the device operates without a router (connection scheme No. 1);
- when the device is connected to a router (connection scheme No. 2), the IP address is configured using the smartphone application (see the product data sheet) and can be set as static or dynamic (DHCP).

The device port is 4000.

The maximum packet size is 256 bytes.

PACKET STRUCTURE



0xFD **0xFD** – packet beginning (2 bytes).

TYPE – protocol type (1 byte). The value is 0x02

SIZE ID – **ID** block size (1 byte). The value is 0x10.

ID – controller ID. This number is located on a sticker (represented as 16 characters) that is affixed to the control board or to the product casing).

It is also possible to use the code word "DEFAULT_DEVICEID" as an ID number. It can be used:

- to control the device if it operates without a router (connection example #1);
 - to search for devices in the network if a router is used (connection example #2);
- at the same time, the device will only respond to two parameters: 0x007C and 0x00B9 (see the parameters table).

SIZE PWD – розмір блока **PWD** (1 байт). Possible value range: from 0x00 to 0x08.

PWD – device password (valid characters include: "0...9", "a...z", "A...Z"). The default password is 1111.

This password can be changed in the Settings menu of the smartphone app.

FUNC – function number (1 byte). Determines the action to be performed with the data and the structure of the **DATA** block:

- 0x01 - reading parameters;
- 0x02 - writing parameters. The controller does not respond to the request for the status of these parameters;
- 0x03 - writing parameters with the subsequent response of the controller on the status of the specified parameters;
- 0x04 - incrementing the parameters followed by the response of the controller on the status of the specified parameters;
- 0x05 - decrementing the parameters followed by the controller response about the state of the specified parameters;
- 0x06 - controller response to the request (FUNC = 0x01,0x03,0x04,0x05).

DATA – the data block. It consists of parameter numbers and their values:

if FUNC = 0x01 or 0x04 or 0x05:



if FUNC = 0x02 or 0x03 or 0x06:



The parameter numbers (see the table of parameters) roughly consist of two bytes (the high byte is virtual). By default, the high byte of each parameter number in each new packet is 0x00. The high byte can be changed within the same packet using the special command 0xFF (see below).

P – the low byte of the parameter number. Possible value range: 0x00 – 0xFB. The values 0xFC - 0xFF are special commands:

0xFC – change the function number (**FUNC**). The next byte must be a new function number from 0x01 to 0x05. It is used to organize several functions with different actions within a single package;

0xFD – the parameter is not supported by the controller. The next byte is the low byte of the unsupported parameter. It is used when the controller responds (**FUNC** = 0x06) to a request to read or write a non-existent parameter;

0xFE – change the size of the **Value** parameter value for one subsequent parameter. The next byte must be the new parameter size, followed by the low-order byte of the parameter number, and then the **Value** itself;

0xFF – change the high byte for parameter numbers within the same packet. The next byte must be the new high byte.

Value – the value of the parameter (by default, 1 byte). The byte sequence is from low to high.

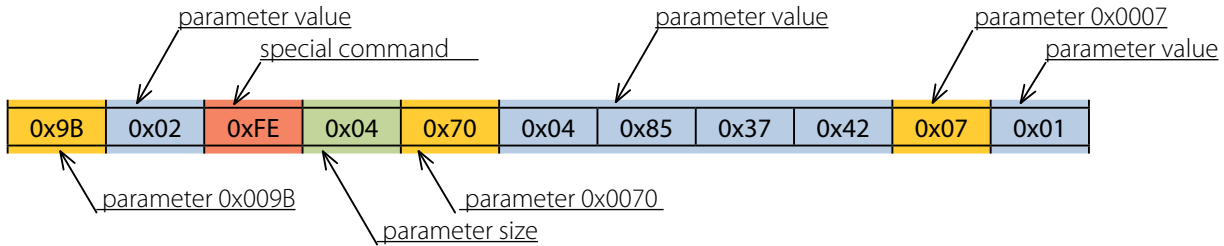
Chksum L **Chksum H** – checksum (2 bytes). It is calculated as the sum of bytes starting with the **TYPE** byte and ending with the last byte of the **DATA** block.

Checksum L is the lowest byte of the checksum.

Checksum H is the high byte of the checksum.

EXAMPLES OF USING SPECIAL COMMANDS IN THE DATA BLOCK

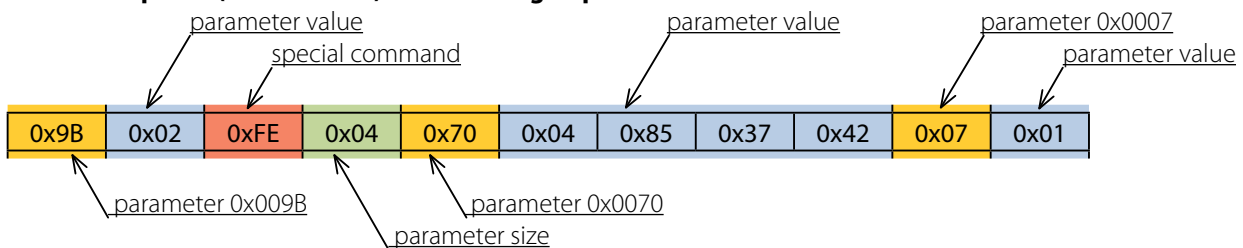
Request for writing (FUNC = 0x03) parameters 0x009B, 0x0070, 0x0007



In the request for writing:

- Assign value 0x009B to parameter 0x02.
- Assign value 0x0070 to parameter 0x42378504. The value size is 4 bytes, as indicated by the special command 0xFE + 0x04.
- Assign value 0x0007 to parameter 0x01.

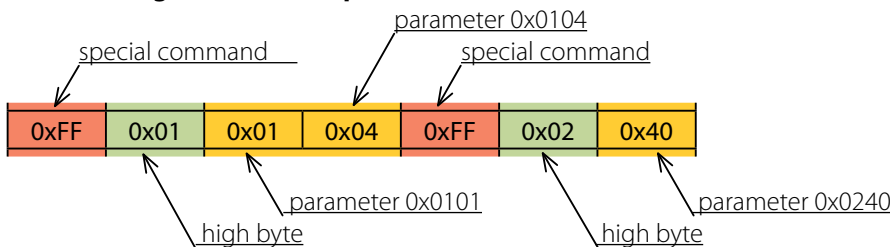
The controller response (FUNC = 0x06) to the writing request



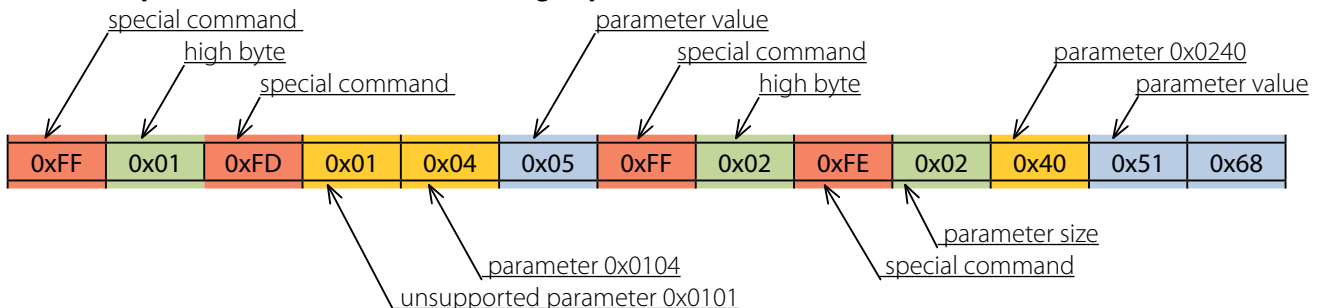
In the controller response:

- Parameter 0x009B has the value 0x02.
- Parameter 0x0070 has the value 0x42378504. The value size is 4 bytes, as indicated by the special command 0xFE + 0x04.
- Parameter 0x0007 has the value 0x01.

Request for reading (FUNC = 0x01) parameters 0x0101, 0x0104, 0x0240



The controller response (FUNC = 0x06) to the reading request



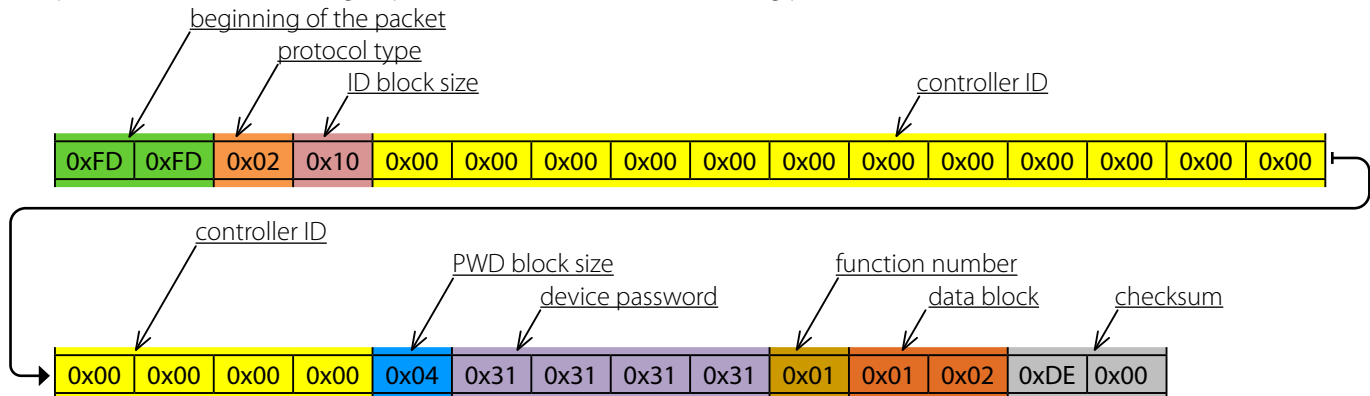
In the controller response:

- The parameter 0x0101 is not supported by the controller. This is indicated by the special command 0xFD.
- Parameter 0x0104 has the value 0x05.
- Parameter 0x0240 has the value 0x6851. The value size is 2 bytes, as indicated by the special command 0xFE + 0x02.

EXAMPLES OF A COMPLETE PACKET

Sending the "Smart House -> Controller" packet

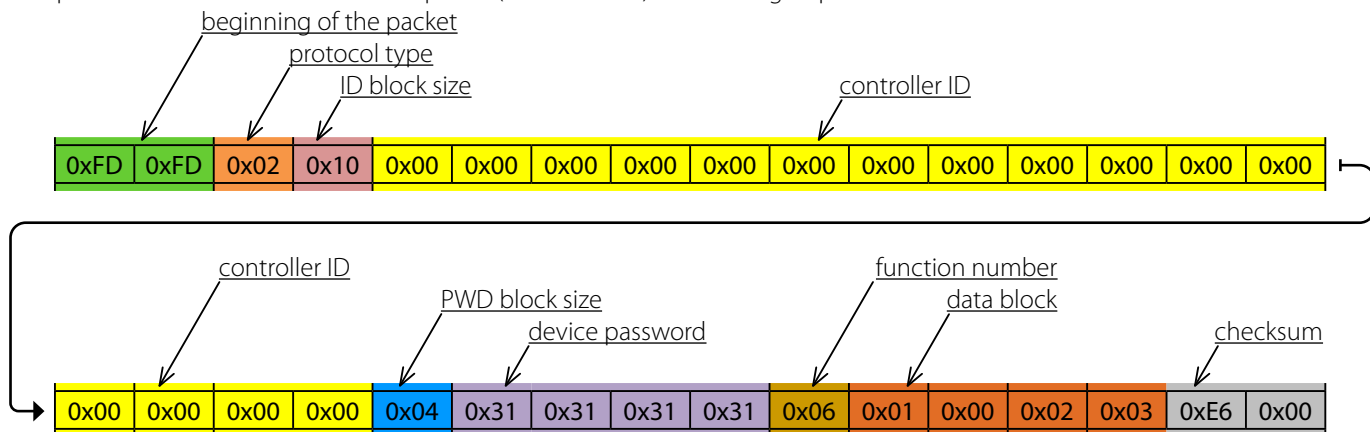
This packet constitutes a reading request (FUNC = 0x01) for the following parameters: 0x0001, 0x0002.



The request includes:
Checksum: 0x00DE.

Sending the "Controller -> Smart House" packet

This packet contains the controller's response (FUNC = 0x06) to a reading request.



In the controller response:

- Parameter 0x0001 has the value 0x00.
- Parameter 0x0002 has the value 0x03.
- Checksum: 0x00E6.

TABLE OF PARAMETERS

Functions:	R – 0x01	INC – 0x04	RW – 0x03	W – 0x02	DEC – 0x05
Parameter number [Dec./Hex.]	Functions	Description	Possible value range:	Size [bytes]	
1/0x0001	R/W/RW	Turn on/off the unit	0 – off 1 – on 2 – invert	1	
2/0x0002	R/W/RW/INC/DEC	Speed mode	1 – first speed, 2 – second speed, 3 – third speed, 4 – fourth speed, 5 – fifth speed, 255 – manual speed setting mode (see parameter 68)	1	
7/0x0007	R/W/RW/INC/DEC	Timer	0 – off, 1 – night, 2 – turbo	1	
11/0x000B	R	Current timer countdown time	1st byte – seconds (0..59), 2nd byte – minutes (0..59), 3rd byte – hours (0..23)	3	
15/0x000F	R/W/RW	Humidity sensor-based control	0 – off 1 – on 2 – invert	1	
17/0x0011	R/W/RW	CO ₂ sensor-based control	0 – off 1 – on 2 – invert	1	
25/0x0019	R/W/RW/INC/DEC	Humidity threshold setting	40 .. 80 RH%	1	
26/0x001A	R/W/RW/INC/DEC	CO ₂ threshold setting	400...2000 ppm	2	
31/0x001F	R	Current outdoor air temperature	-32768 – no sensor, +32767 – short circuit	signed 2 (must be divided by 10, one decimal place)	
32/0x0020	R	Current supply air temperature downstream of the reheater	-32768 – no sensor, +32767 – short circuit	signed 2 (must be divided by 10, one decimal place)	
33/0x0021	R	Current exhaust air temperature at the inlet	-32768 – no sensor, +32767 – short circuit	signed 2 (must be divided by 10, one decimal place)	
34/0x0022	R	Current exhaust air temperature at the outlet	-32768 – no sensor, +32767 – short circuit	signed 2 (must be divided by 10, one decimal place)	
36/0x0024	R	Current RTC battery voltage	0...5000 mV	2	
37/0x0025	R	Current room humidity	0 – 100 RH%	1	
39/0x0027	R	Current indoor CO ₂ level	0...2000 ppm	2	
58/0x003A	R/W/RW/INC/DEC	Supply fan speed at 1st speed	10...100 %	1	
59/0x003B	R/W/RW/INC/DEC	Extract fan speed at 1st speed	10...100 %	1	

Parameter number [Dec./Hex.]	Functions	Description	Possible value range:	Size [bytes]
60/0x003C	R/W/RW/INC/DEC	Supply fan speed at 2nd speed	10...100 %	1
61/0x003D	R/W/RW/INC/DEC	Extract fan speed at 2nd speed	10...100 %	1
62/0x003E	R/W/RW/INC/DEC	Supply fan speed at 3rd speed	10...100 %	1
63/0x003F	R/W/RW/INC/DEC	Extract fan speed at 3rd speed	10...100 %	1
68/0x0044	R/W/RW/INC/DEC	Fan speed in manual speed setting mode.	10...100 %	1
74/0x004A	R	Supply fan speed	0 ... 5000 rpm	2
75/0x004B	R	Extract fan speed	0 ... 5000 rpm	2
99/0x0063	R/W/RW/INC/DEC	Setting the filter change timer	0, 70 .. 365 days	2
100/0x0064	R	Filter change timer countdown	1st byte – minutes (0...59), 2nd byte – hours (0...23), 3rd, 4th byte – days (0...365)	4
101/0x0065	W	Reset the filter change timer countdown	1 – execute	1
104/0x0068	R/W/RW	Heater control	0 – off 1 – on 2 – invert	1
111/0x006F	R/W/RW	RTC time	1st byte – RTC seconds (0...59), 2nd byte – RTC minutes (0...59), 3rd byte – RTC hours (0...23)	3
112/0x0070	R/W/RW	RTC calendar	1st byte – RTC day (1...31), 2nd byte – RTC day of the week (1...7), 3rd byte – RTC month (1...12), 4th byte – RTC year (0...99)	4
114/0x0072	R/W/RW	Weekly schedule mode	0 – off 1 – on 2 – invert	1
119/0x0077	R/W/RW	<p>Schedule settings</p> <p>In the read request, the special command 0xFE must be used and the value of parameter 0x02 must be specified to select the required day of the week and time period number.</p> <p>The write request uses all 6 bytes, as does the controller's response.</p> <p>The start of the first time period is always 00:00, and the start of any subsequent period is the end of the previous one.</p> <p>The end of the last time period is always 24:00.</p>	<p>1st byte – day of the week: 0 – all days (write only), 1 – Monday, 2 – Tuesday, 3 – Wednesday, 4 – Thursday, 5 – Friday, 6 – Saturday, 7 – Sunday, 8 – Mon...Fri (write only), 9 – Sat...Sun (write only)</p> <p>2nd byte – period number: 1...4</p> <p>3rd byte – speed 0 – standby 1...5</p> <p>4th byte – reserved</p> <p>5th byte – minutes until the end of the interval: 0...59</p> <p>6th byte – hours until the end of the interval: 0...23</p>	6

Parameter number [Dec./Hex.]	Functions	Description	Possible value range:	Size [bytes]
124/0x007C	R	Search for devices on the local Ethernet network. The response contains the device ID	Text ("0...9", "A...F")	16
125/0x007D	R/W/RW	Device password for the Ethernet network	Text ("0...9", "a...z", "A...Z")	0-8
126/0x007E	R	Motor running hours	1st byte – minutes (0..59), 2nd byte – hours (0..23) 3rd, 4th byte – days (0...65535)	4
127/0x007F	R	List of current alarms/warnings	1st byte – code 2nd byte – type: 1-alarm , 2-warning	0,2,4...
128/0x0080	W	Reset faults	1 – execute	1
129/0x0081	R	Heater status	0 – off, 1 – on,	1
131/0x0083	R	Fault/warning indicator	0 – none, 1 – alarm (has higher priority) 2 – warning	1
132/0x0084	R	Air quality status	1st byte – RH: 0-normal, 1-over setpoint, 2nd byte – CO ₂ : 0-normal, 1-over setpoint, 3rd byte – reserved, 4th byte – reserved, 5th byte – VOC (AQI): 0-normal, 1-over setpoint,	5
133/0x0085	R/W/RW	Enable operation via cloud server	0 – off, 1 – on, 2 – invert	1
134/0x0086	R	Controller firmware version and date	1st byte – firmware version (major) 2nd byte – firmware version (minor), 3rd byte – day, 4th byte – month, 5th, 6th byte – year	6
135/0x0087	W	Restore all parameters to factory settings	1 – execute	1
136/0x0088	R	Filter status	0 – clean, 1 – soiled	1
148/0x0094	R/W/RW/INC/DEC	Wi-Fi operating mode	1 – client, 2 – access point	1
149/0x0095	R/W/RW	Wi-Fi name in client mode	Text	1...32
150/0x0096	R/W/RW	Wi-Fi password	Text	8...64
153/0x0099	R/W/RW	Wi-Fi data encryption type	48 – OPEN, 50 – WPA_PSK, 51 – WPA2_PSK, 52 – WPA_WPA2_PSK	1
154/0x009A	R/W/RW/INC/DEC	Wi-Fi frequency channel	1...13	1
155/0x009B	R/W/RW	Wi-Fi module DHCP	0 – STATIC, 1 – DHCP, 2 – invert	1

Parameter number [Dec./Hex.]	Functions	Description	Possible value range:	Size [bytes]
156/0x009C	R/W/RW	Set IP address of the Wi-Fi module	1st byte – 0...255, 2nd byte – 0...255, 3rd byte – 0...255, 4th byte – 0...255,	4
157/0x009D	R/W/RW	Subnet mask of the Wi-Fi module	1st byte – 0...255, 2nd byte – 0...255, 3rd byte – 0...255, 4th byte – 0...255,	4
158/0x009E	R/W/RW	Default gateway of the Wi-Fi module	1st byte – 0...255, 2nd byte – 0...255, 3rd byte – 0...255, 4th byte – 0...255,	4
160/0x00A0	W	Apply new Wi-Fi settings and exit the Wi-Fi module setup mode	1 – execute	1
162/0x00A2	W	Quit the Wi-Fi module setup mode without applying new Wi-Fi settings	1 – execute	1
163/0x00A3	R	Current IP address of the Wi-Fi module	1st byte – 0...255, 2nd byte – 0...255, 3rd byte – 0...255, 4th byte – 0...255,	4
183/0x00B7	R/W/RW/INC/DEC	Fan rotation direction	0 – ventilation, 1 – regeneration, 2 – supply, 3 – extract	1
185/0x00B9	R	Device type	0 – 65535, 17 – Freshpoint 160, 20 – Freshpoint Eco 160, 22 – Freshpoint 200, 24 – Freshpoint Eco 200	2
252/0x00FC	Special commands			
253/0x00FD				
254/0x00FE				
255/0x00FF				
297/0x0129	R	Recovery efficiency	0...100 %	1
298/0x012A	R/W/RW	Restore all fan speed modes' settings to factory defaults	1 – execute	1
770/0x0302	R/W/RW	Set time for night timer	0th byte – minutes 1st byte – hours	2
771/0x0303	R/W/RW	Set time for Turbo timer	0th byte – minutes 1st byte – hours	2
774/0x0306	R	Current speed in schedule mode	0...3	1
779/0x030B	R	Frost protection status	0 – inactive, 1 – active	1
789/0x0315	R/W/RW	Air quality sensor-based (VOC) control	0 – off 1 – on 2 – invert	1
799/0x031F	R/W/RW/INC/DEC	Air quality sensor (VOC) setpoint	50-250 index	2
800/0x0320	R	Current air quality level (VOC)	0 – 500 index	2

Parameter number [Dec./Hex.]	Functions	Description	Possible value range:	Size [bytes]
1024/0x0400	R/W/RW/INC/DEC	Screen backlight brightness (Manual)	1...100	1
1025/0x0401	R/W/RW	Activation of the sound emitter (response to control commands).	0 – off 1 – on 2 – invert	1
1026/0x0402	R/W/RW	Screen backlight mode	0 – Auto, 1 – Manual, 2 – invert	1
1027/0x0403	R/W/RW/INC/DEC	Selecting the temperature sensor to display on the device screen	0 – alternating, 1 – supply air, 2 – extract air	1
1028/0x0404	R/W/RW/INC/DEC	Selecting the air quality sensor to display on the device screen	0 – alternating, 1 – CO ₂ , 2 – VOC	1
1029/0x0405	R/W/RW/INC/DEC	Display of time or temperature/humidity on the screen	0 – alternating 1 – time 2 – temperature and humidity	1
1030/0x0406	R/W/RW	Display of time in Standby mode on the screen	0 – off 1 – on	1
1031/0x0407	R/W/RW	Enabling/disabling all screen display	0 – off 1 – on 2 – interval during which the indication is off	1
1032/0x0408	R/W/RW	Start time for turning off the screen display	0th byte – minutes (0...59), 1st byte – hours (0...23)	2
1033/0x0409	R/W/RW	End time for turning off the screen display	0th byte – minutes (0...59), 1st byte – hours (0...23)	2

EXAMPLE OF PACKET PROCESSING IN C

```
//===== Special commands =====//
#define BGCP_CMD_PAGE                0xFF
#define BGCP_CMD_FUNC                0xFC
#define BGCP_CMD_SIZE                0xFE
#define BGCP_CMD_NOT_SUP             0xFD
//=====//

#define BGCP_FUNC_RESP                0x06

uint8_t receive_data[256];
uint16_t receive_data_size;
uint8_t State_Power;
uint8_t State_Speed_mode;
char current_id[17] = "002D6E1B34565815"; // controller ID

//***** Checksum verification and beginning of the packet *****/
uint8_t check_protocol(uint8_t *data, uint16_t size)
{
    uint16_t i, chksum1 = 0, chksum2 = 0;
    if((data[0] == 0xFD) && (data[1] == 0xFD))
    {
        for(i = 2; i <= size-3; i++)
            chksum1 += data[i];
        chksum2 = (uint16_t)(data[size-1] << 8) | (uint16_t)(data[size-2]);
        if(chksum1 == chksum2)
            return 1;
        else
            return 0;
    }
    else
        return 0;
}
//*****//

int main(void)
{
    ...

    if(check_protocol(receive_data, receive_data_size) == 1) // Checksum
    {
        if(receive_data[2] == 0x02) // Protocol type
        {
            if(memcmp(&receive_data[4], current_id, receive_data[3]) == 0) // ID
            {
                uint16_t jump_size = 0, page = 0, param, param_size, r_pos;
                uint8_t flag_check_func = 1, BGCP_func;

                r_pos = 4 + receive_data[3];
                r_pos += 1 + receive_data[r_pos]; // Location in the array where the FUNC block starts
                //***** FUNC & DATA *****/
                for(; r_pos < receive_data_size - 2; r_pos++)
                {
                    //===== Special commands =====//
                    param_size = 1;
                    //==== new function number
                    if((flag_check_func == 1) || (receive_data[r_pos] == BGCP_CMD_FUNC))
                    {
                        if(receive_data[r_pos] == BGCP_CMD_FUNC)
                            r_pos++;
                        flag_check_func = 0;
                        BGCP_func = receive_data[r_pos];
                        if(BGCP_func != BGCP_FUNC_RESP) // if the function number is not supported
                            break;
                        continue;
                    }
                    //==== new value of the high byte for parameter numbers
                    else if(receive_data[r_pos] == BGCP_CMD_PAGE)
                    {

```