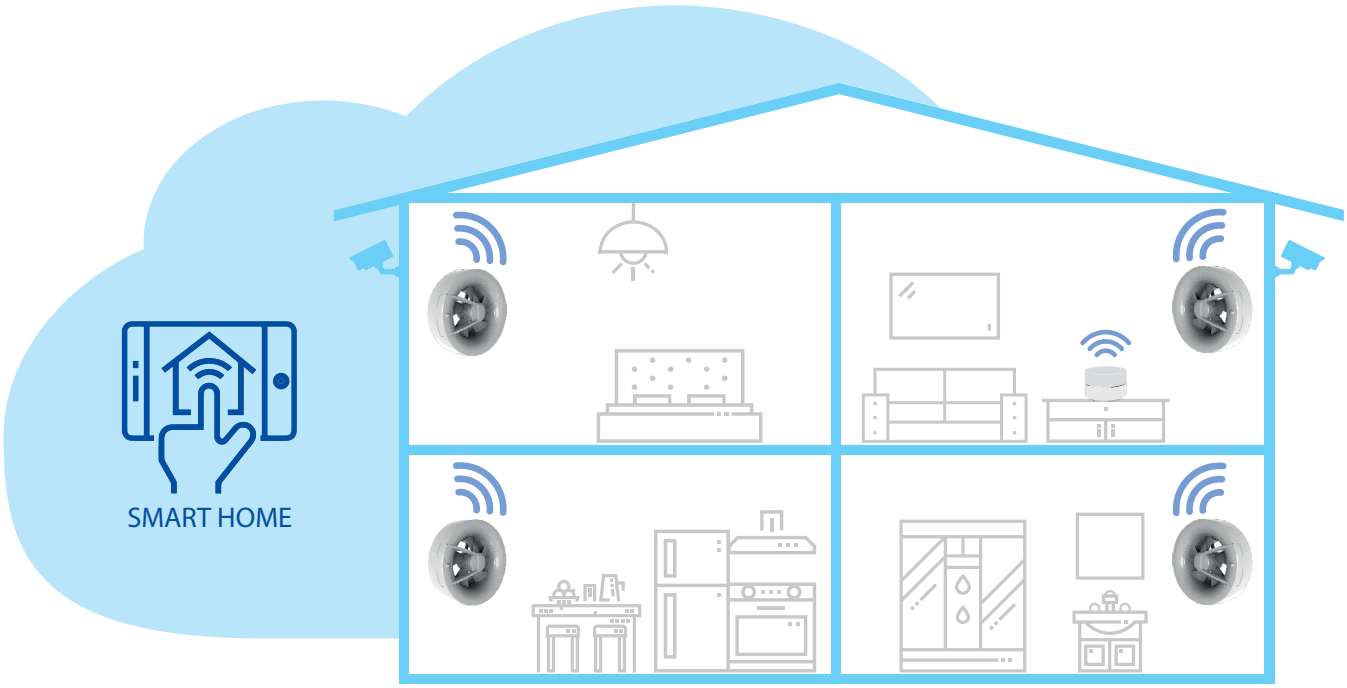


CONNECTING TO THE SMART HOME SYSTEM

O2 Supreme

EN CONNECTION INSTRUCTION

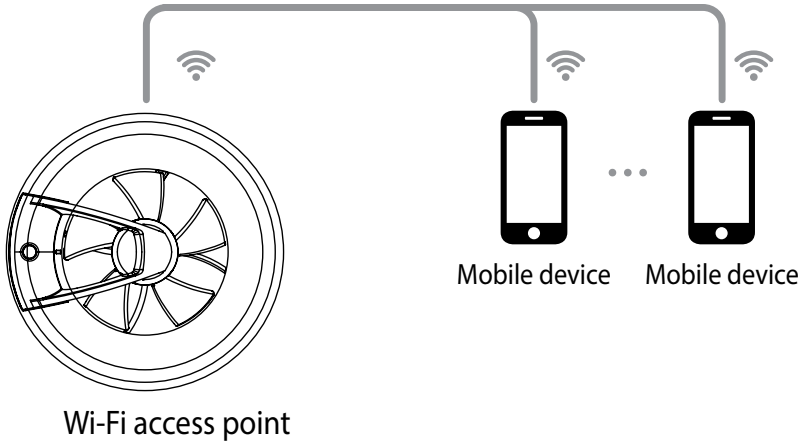
BLAUBERG

CONTENTS

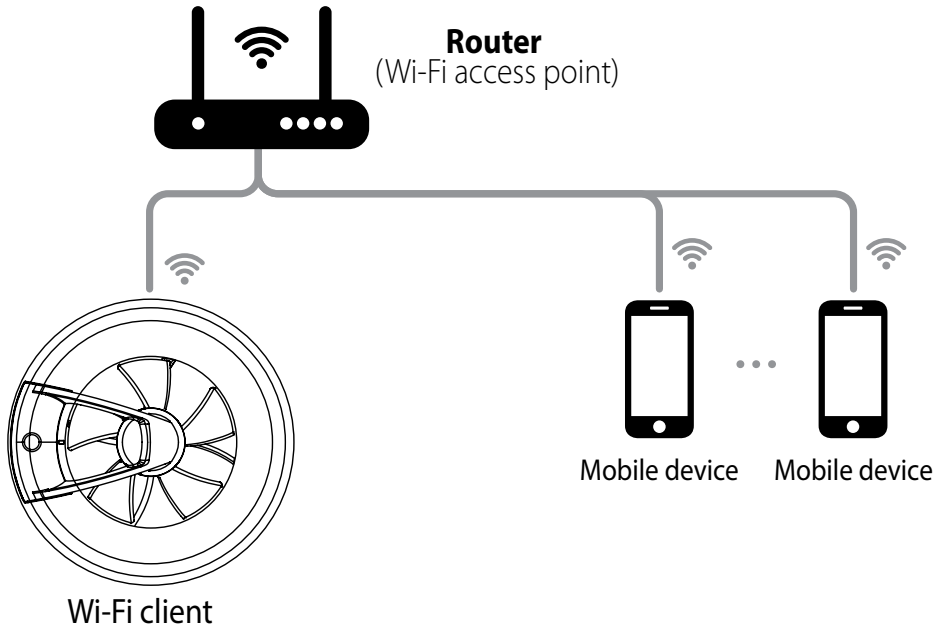
Connection and setup 2
Network parameters 3
Packet structure 4
Examples of using special commands in the data block 5
Examples of a complete packet 6
Table of parameters 7
Example of packet processing in C 10

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: 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	TYPE	SIZE ID	ID	SIZE PWD	PWD	FUNC	DATA	Chksum L	Chksum H

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 the **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 – PWD block size (1 byte). 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.
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:

P1	P2	Pn

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

P1	Value 1	P2	Value 2	Pn	Value n

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;

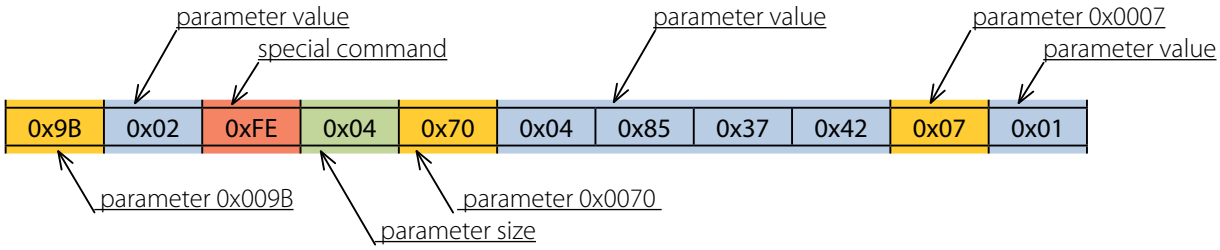
0xFF – change the high byte for parameter numbers within the same packet.
The next byte must be the new high byte.

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 low byte of the checksum.
Checksum H is the high byte of the checksum.

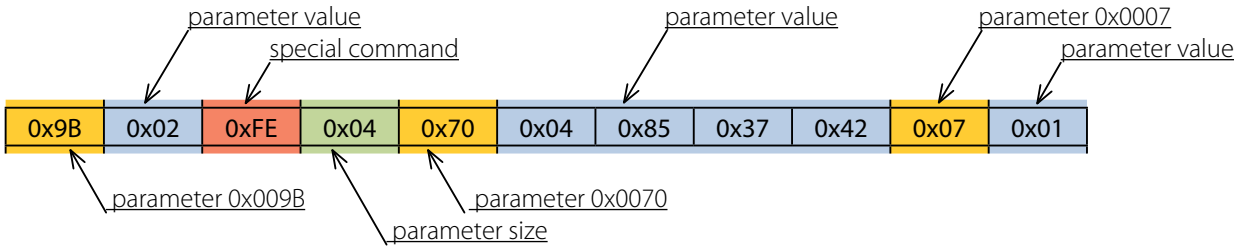
EXAMPLES OF USING SPECIAL COMMANDS IN THE DATA BLOCK

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



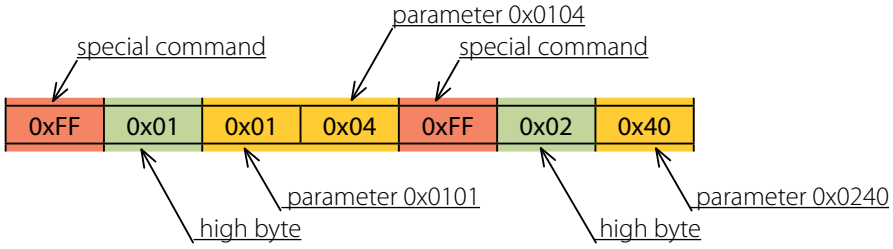
In the request for writing:
- Assign value 0x009B to parameter 0x02.
- Assign value 0x0070 to parameter 0x42378504. The value size is 4 bytes, as indicated by the special command 0xFE + 0x04.
- Assign value 0x0007 to parameter 0x01.

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

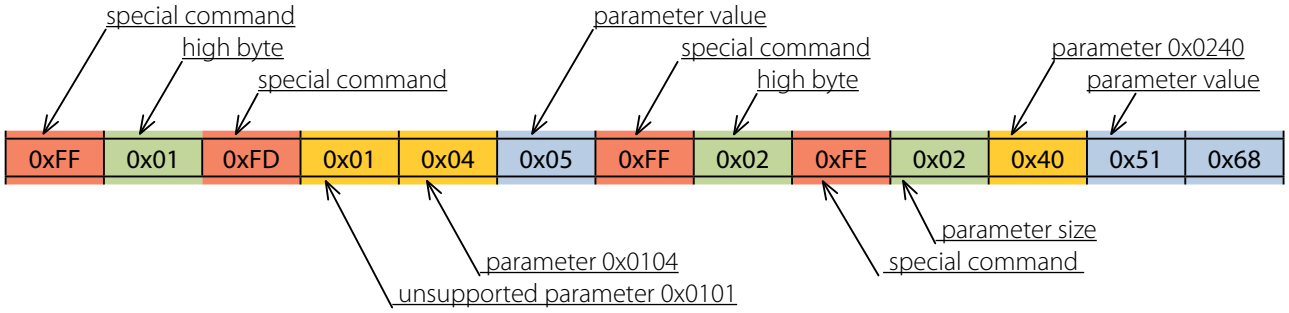


In the controller response:
- Parameter 0x009B has the value 0x02.
- Parameter 0x0070 has the value 0x42378504.
- Parameter 0x0007 has the value 0x01. Request for writing (FUNC = 0x01) parameters 0x0101, 0x0104, 0x0240

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



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

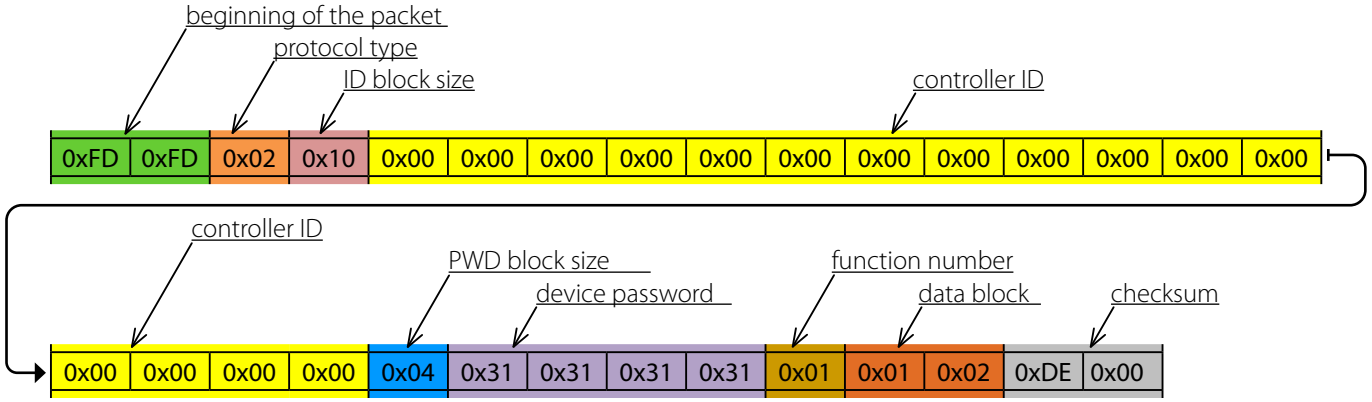


In the controller response:
- The parameter 0x0101 is not supported by the controller. This is indicated by the special command 0xFD.
- Parameter 0x0104 has the value 0x05.
- Parameter 0x0240 has the value 0x6851. The value size is 2 bytes, as indicated by the special command 0xFE + 0x02.

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.

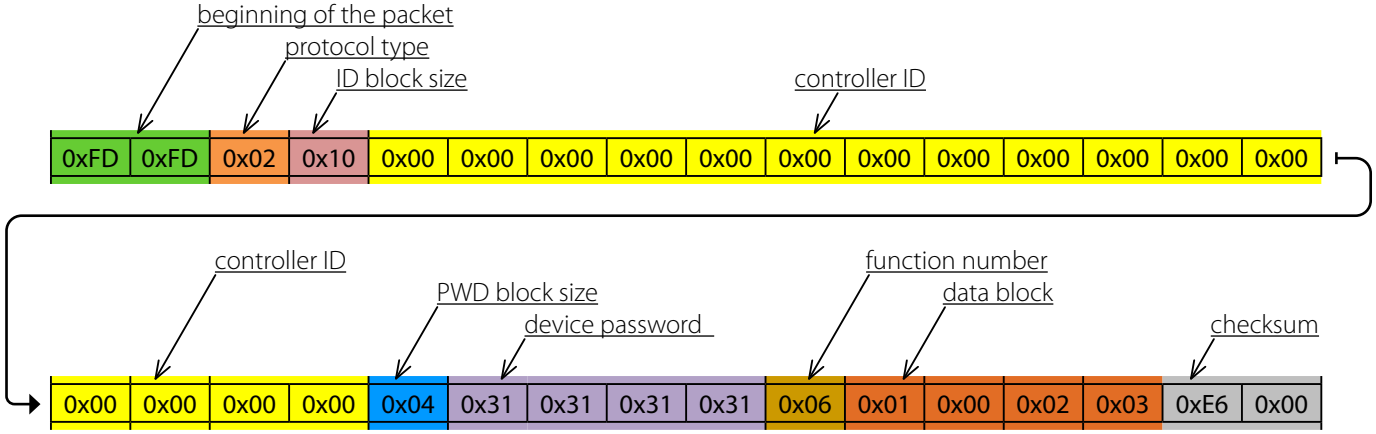


The request includes:

- 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]
6/0x0006	R/W/RW	Boost mode	0 – off 1 – on 2 – invert	1
7/0x0007	R	Indicator of operation of the turn-off delay timer	0 – off 1 – on	1
11/0x000B	R	Current countdown time of the Boost timer	1st byte – seconds (0...59), 2nd byte – minutes (0...59), 3rd byte – hours (0...23)	3
15/0x000F	R/W/RW/INC/DEC	Control by humidity sensor	0 – off, 1 – auto, 2 – manual setting (see parameter 25)	1
25/0x0019	R/W/RW/INC/DEC	Setting the humidity threshold	40 … 80 RH %	1
33/0x0021	R	Current room temperature	-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
75/0x004B	R	Fan speed	0 ... 5000 rpm	2
102/0x0066	R/W/RW/INC/DEC	Setting the turn-off delay timer	0 … 60 min	1
111/0x006F	R/W/RW	RTC time	1st byte – seconds (0...59), 2nd byte – minutes (0...59), 3rd byte – hours (0...23)	3
124/0x007C	R	Search for devices on the local Ethernet network. The response contains the device ID number	Text ("0…9", "A…F")	16
125/0x007D	R/W/RW	Device password	Text ("0…9", "a…z", "A…Z")	0-8
131/0x0083	R	Low battery warning indicator	0 – off 1 – on,	1
133/0x0085	R/W/RW	Permission to work via the cloud server	0 – off 1 – on 2 – invert	1
134/0x0086	R	Version and date of the main controller firmware	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 settings to factory defaults	1 – execute	1
148/0x0094	R/W/RW/INC/DEC	Wi-Fi operation 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

Parameter number [Dec./Hex.]	Functions	Description	Possible value range:	Size [bytes]
155/0x009B	R/W/RW	DHCP of the Wi-Fi module	0 - STATIC, 1 - DHCP, 2 - invert	1
156/0x009C	R/W/RW	Set the 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	Exit the Wi-Fi module configuration 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
185/0x00B9	R	Device type	**0 - 65535,** **13 - O2 Supreme**	2
252/0x00FC 253/0x00FD 254/0x00FE 255/0x00FF	**Special commands**			
772/0x0304	R	Indicator of excess humidity	0 – off 1 – on	1
781/0x030D	R/W/RW	Mode 24	0 – off 1 – on 2 – invert	1
782/0x030E	R	Light sensor triggering indicator	0 – off 1 – on	1
783/0x030F	R	Motion detector operation indicator	0 – off 1 – on	1
784/0x0310	R	Interval ventilation operation indicator	0 – off 1 – on	1
785/0x0311	R	Indicator of the Silent mode operation	0 – off 1 – on	1
786/0x0312	R	Air quality deterioration indicator	0 – off 1 – on	1
787/0x0313	R/W/RW	Control by light sensor	0 – off 1 – on 2 – invert	1
788/0x0314	R/W/RW	Control by motion sensor	0 – off 1 – on 2 – invert	1
789/0x0315	R/W/RW/INC/DEC	Control by air quality sensor	0 – off 1 – auto, 2 - manual setting (see parameter 799)	1

Parameter number [Dec./Hex.]	Functions	Description	Possible value range:	Size [bytes]
790/0x0316	R/W/RW	Interval ventilation mode	0 – off 1 – on 2 – invert	1
791/0x0317	R/W/RW	Silent mode	0 – off 1 – on 2 – invert	1
792/0x0318	R/W/RW	Start of Silent mode	1st byte – seconds (0...59), 2nd byte – minutes (0...59), 3rd byte – hours (0...23)	3
793/0x0319	R/W/RW	End of Silent mode	1st byte – seconds (0...59), 2nd byte – minutes (0...59), 3rd byte – hours (0...23)	3
794/0x031A	R/W/RW	Air flow when the RH sensor is triggered	3 - 60 m³/h , 4 - 90 m³/h, 5 - 115 m³/h	1
795/0x031B	R/W/RW	Air flow when motion/light sensor is triggered	2 - 40 m³/h, 3 - 60 m³/h , 4 - 90 m³/h, 5 - 115 m³/h	1
796/0x031C	R/W/RW	Air flow when the air quality sensor is triggered	3 - 60 m³/h , 4 - 90 m³/h, 5 - 115 m³/h	1
797/0x031D	R/W/RW	Air flow during interval ventilation	1 - 20 m³/h, 2 - 40 m³/h, 3 - 60 m³/h	1
798/0x031E	R/W/RW	Air flow in 24 hour mode	1 - 20 m³/h, 2 - 40 m³/h, 3 - 60 m³/h	1
799/0x031F	R/W/RW/INC/DEC	Air quality sensor setpoint	50 - 500 index	2
800/0x0320	R	Current air quality level	0 - 500 index	2
803/0x0323	R	Indicator of excess temperature	0 – off 1 – on	1
804/0x0324	R/W/RW	Temperature sensor control	0 – off 1 – on 2 – invert	1
805/0x0325	R/W/RW/INC/DEC	Temperature sensor setpoint	18 - 36 °C	1
815/0x032F	R/W/RW	Air flow when the temperature sensor is triggered	3 - 60 m³/h, 4 - 90 m³/h, 5 - 115 m³/h	1

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

//********* Checking 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 i DATA *********************//
                for(; r_pos < receive_data_size - 2; r_pos++)
                {
                    //=========== Special commandи ===========//
                    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)
                    {
```

```
                    page = receive_data[++r_pos];
                    continue;
                }
                //=== new value of the parameter size
                else if(receive_data[r_pos] == BGCP_CMD_SIZE)
                {
                    param_size = receive_data[++r_pos];
                    r_pos++;
                }
                //=== if the parameter is unsupported
                else if(receive_data[r_pos] == BGCP_CMD_NOT_SUP)
                {
                    r_pos++;
                    //******* processing of unsupported parametersв ******//
                    param = (uint16_t)(page << 8) | (uint16_t)(receive_data[r_pos]);
                    switch(param)
                    {
                       case 0x0001:
                          break;
                       case 0x0002:
                          break;
                        ...
                    }
                    //***********************//
                    continue;
                }
                jump_size = param_size;
                //===================================//

                //******* processing of supported parameters ******//
                param = (uint16_t)(page << 8) | (uint16_t)(receive_data[r_pos]);
                switch(param)
                {
                    case 0x0001:
                        State_Power = receive_data[r_pos+1];
                        break;
                    case 0x0002:
                        State_Speed_mode = receive_data[r_pos+1];
                        break;
                    ...
                }
                //***********************//
                r_pos += jump_size;
            }
            //*********************************************************//
        }
    }
}
```

BLAUBERG

www.blaubergventilatoren.de
B255-1EN-01