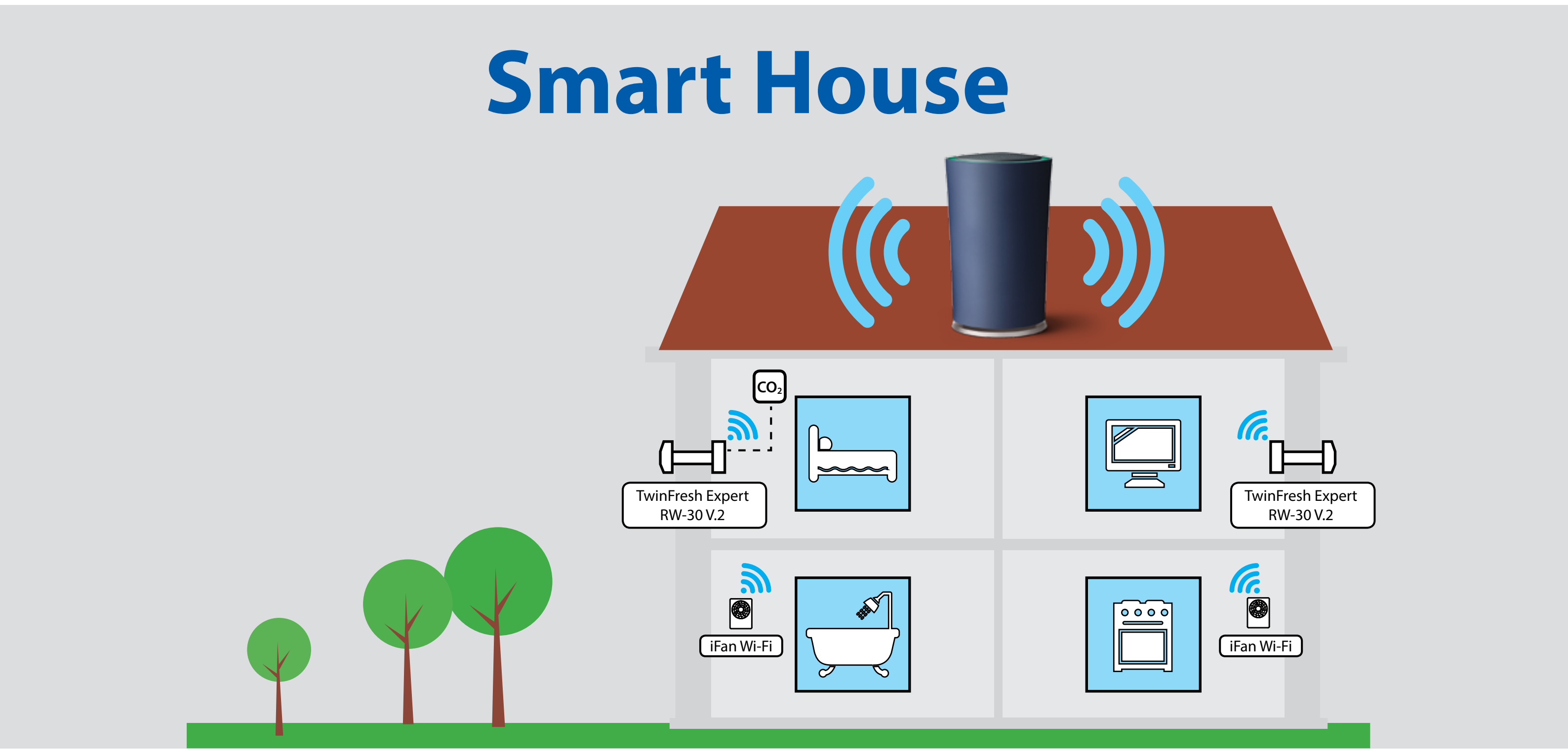
CONNECTION GUIDE

IFAN WI-FI



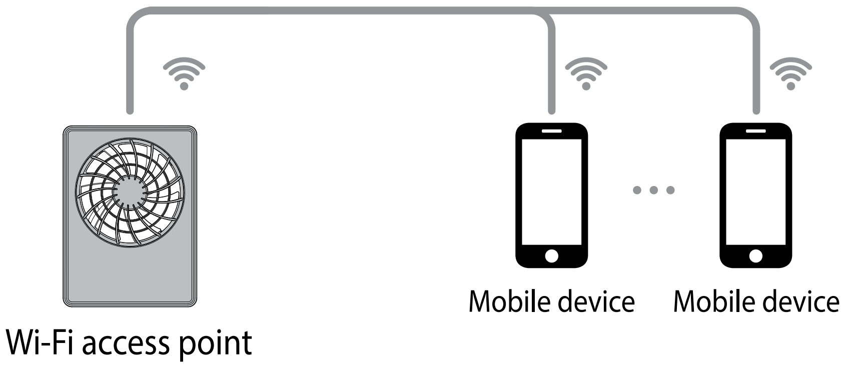
Connection to a Smart Home system

VENTS

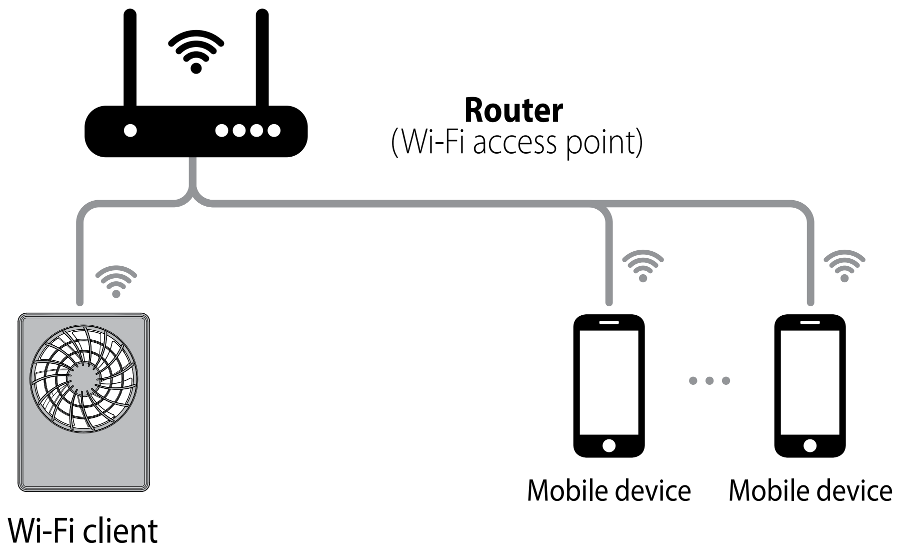
CONTENTS

CONNECTION AND SETUP

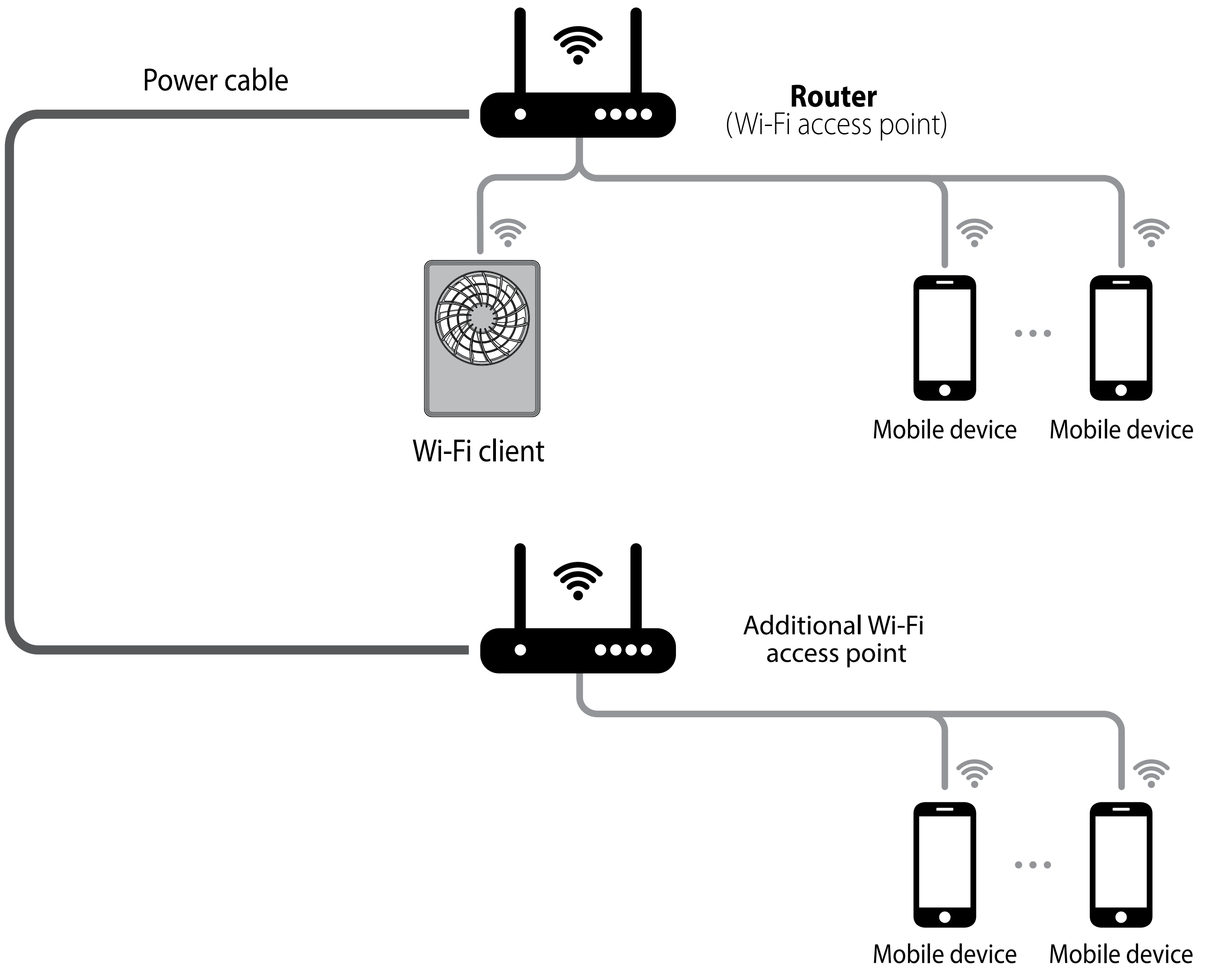
Example 1: Pattern of direct connection of the fan to the BMS Smart Home system without using a router.
Set up the fan to operate Wi-Fi in the access point mode (see the User's manual for the fan).
Note: maximum possible number of connected control devices is eight.



Example 2: connection via router with one Wi-Fi access point.
The fan, mobile devices and the BMS Smart Home system are connected to the Wi-Fi access point of the network router.



Example 3: BMS Smart Home system connection via router with several Wi-Fi access points.



NETWORK PARAMETERS

Data is exchanged via the UDP protocol (with broadcast support).

Master device IP address:

- 192.168.4.1 — if the master device runs without a router (connection pattern 1).
- If the master device is connected via a router (connection pattern 2), the IP address is set up via a mobile application (see the User's manual) and can be defined as static or dynamic (DHCP).

Master device port: 4000.

Maximum packet size: 256 bytes.

PACKET STRUCTURE

0xFD	0xFD	TYPE	SIZE ID	ID	SIZE PWD	PWD	FUNC	DATA	Chksum L	Chksum H

0xFD 0xFD: packet beginning character (2 bytes).

TYPE: protocol type (1 byte). Value = 0x02.

SIZE ID: ID block size (1 byte). Value = 0x10.

ID: controller ID. This number is printed on the label (16 characters) applied to the control circuit board or the unit casing.

You can also substitute the ID with «DEFAULT_DEVICEID» code word. The ID can be used:

- To control the master device if it runs without a router (connection pattern 1).
- To search for master devices on the network if a router is used (connection pattern 2). In this case, the device will respond to two parameters only: 0x007C and 0x00B9 (see parameter table).

SIZE PWD: PWD block size (1 byte). Possible values: from 0x00 to 0x08.

PWD: device password (permissible characters: "0...9", "a...z", and "A...Z"). The default password is "1111".
This password can be changed via the mobile application from the **Connection —> At home —> Settings menu** (see the unit data sheet).

FUNC: the function number (1 byte). It defines the action with the data and the **DATA** block structure:
0x01: parameter read.
0x02: parameter write. The controller does not send any response regarding the status of the given parameters.
0x03: parameter write with subsequent controller response regarding the status of the given parameters.
0x04: parameter increment with subsequent controller response regarding the status of the given parameters.
0x05: parameter decrement with subsequent controller response regarding the status of the given parameters.
0x06: controller response to the request (FUNC = 0x01, 0x03, 0x04, 0x05).

DATA: 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

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

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

0xFC: change function (FUNC) number. The following byte must be the new function number ranging from 0x01 to 0x05. This command is used to organise several functions with different actions into a single packet.

0xFD: parameter not supported by the controller. The following byte is the low byte of the non-supported parameter. This command is used in controller response (FUNC = 0x06) to a non-supported parameter read or write request.

0xFE: change a size of the Value parameter value for one parameter which follows. The following byte must be the new parameter size followed by the low byte of the parameter number, and then - by the Value itself.

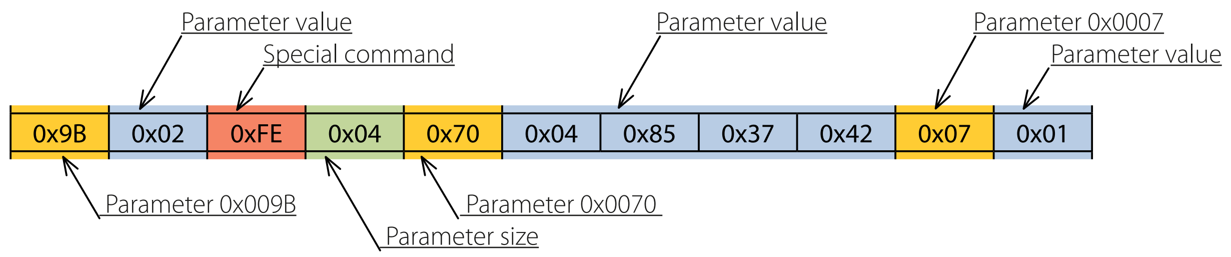
0xFF: change the high byte for parameter numbers within a single packet. The following byte must be the new high byte.

Value: parameter value (the default size of value is 1 byte). Byte ordering from least significant byte to most significant byte.

Chksum L Chksum H: check sum (2 bytes). This is calculated as the total of bytes beginning with the TYPE byte and ending with the final byte of the DATA block.
Chksum L: checksum low byte.
Chksum H: checksum high byte.

EXAMPLES OF USING SPECIAL COMMANDS IN THE DATA BLOCK

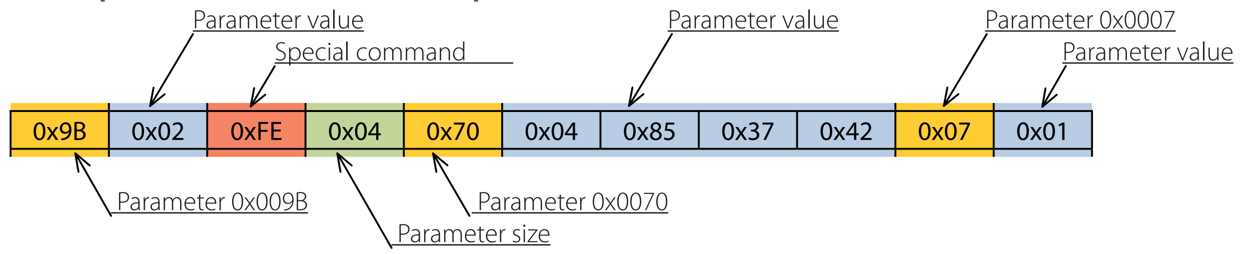
Write request (FUNC = 0x03) for parameters number 0x009B, 0x0070, and 0x0007



Write request details:

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

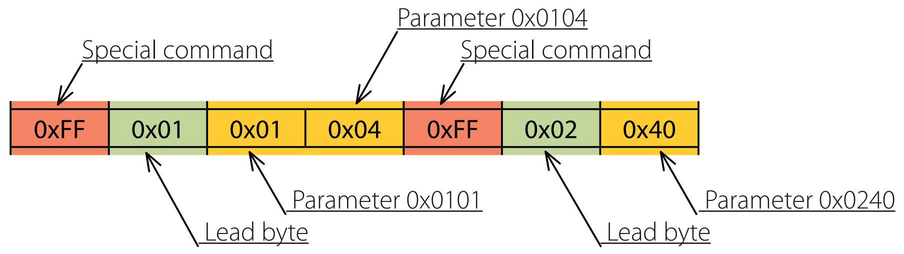
Controller response (FUNC = 0x06) to write request



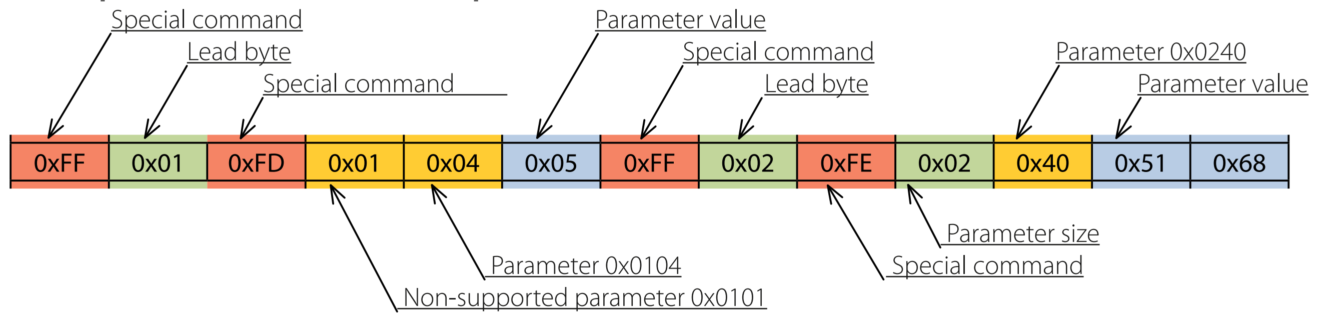
Controller response details:

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

Read request (FUNC = 0x01) for parameters number 0x0101, 0x0104, and 0x0240



Controller response (FUNC = 0x06) to write request



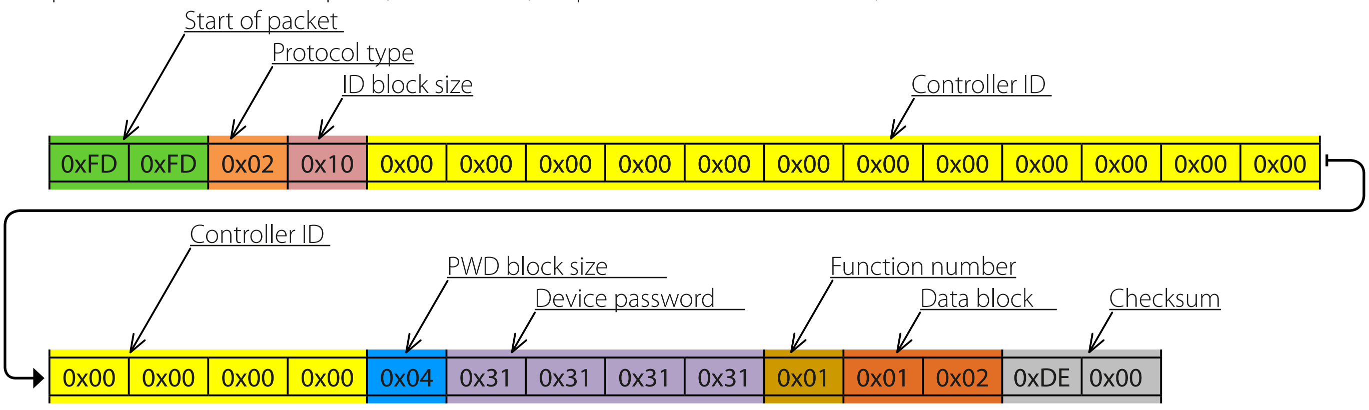
Controller response details:

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

COMPLETE PACKET EXAMPLES

Sending “Smart Home —> Controller” packet

This packet contains a read request (FUNC = 0x01) for parameters number: 0x0001, 0x0002.



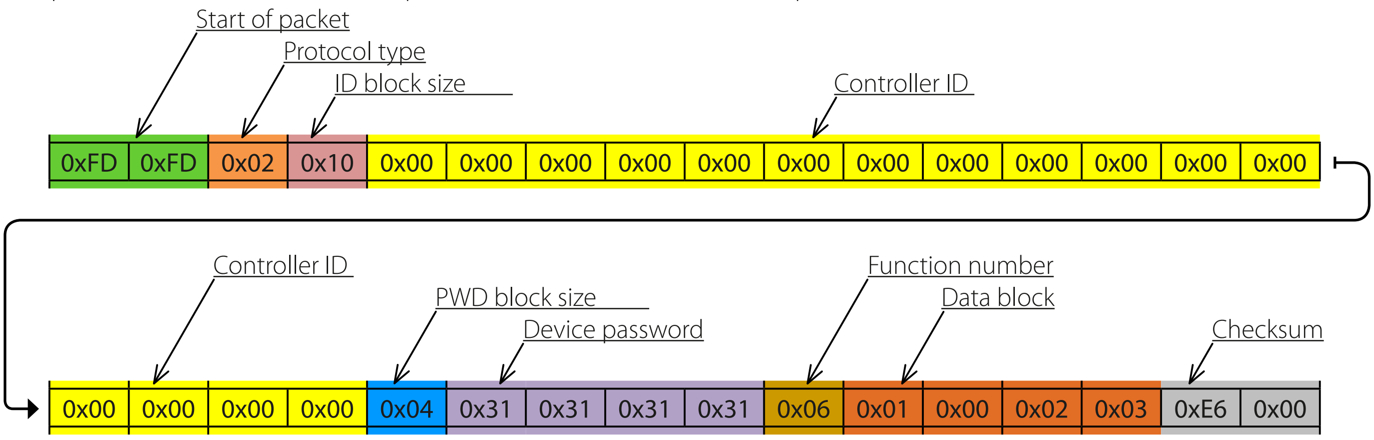
0xFD	0xFD	0x02	0x10	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
0x00	0x00	0x00	0x00	0x04	0x31	0x31	0x31	0x31	0x01	0x01	0x02	0xDE	0x00		

Request details:

- Checksum: 0x00DE.

Sending “Controller —> Smart Home” packet

This packet contains the controller response (FUNC = 0x06) to the write request.



0xFD	0xFD	0x02	0x10	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
0x00	0x00	0x00	0x00	0x04	0x31	0x31	0x31	0x31	0x06	0x01	0x00	0x02	0x03	0xE6	0x00

Controller response details:

- Parameter 0x0001 equals 0x00.
- Parameter 0x0002 equals 0x03.
- Checksum: 0x00E6.

PARAMETER TABLE

Functions:

R - 0x01	**INC** - 0x04	**RW** - 0x03
W - 0x02	**DEC** - 0x05	

Parameter number [Dec./Hex.]	Functions	Description	Possible values	Size [bytes]
1/0x0001	R/W/RW	Fan On/Off	0-Off 1-On 2-Invert	1
2/0x0002	R	Battery status	0 - discharged (absent) 1-normal rate of charge	1
3/0x0003	R/W/RW	24 hours mode selection	0-Off 1-On 2-Invert	1
4/0x0004	R	Current fan speed (rpm)	0...6000 RPM	2
5/0x0005	R/W/RW	BOOST mode On/Off	0-Off 1-On 2-Invert	1
6/0x0006	R	Current BOOST timer countdown in seconds	0...86400 seconds	3
7/0x0007	R	Current status of the built-in timer	0-Off 1-On	1
8/0x0008	R	Current status of fan operation by humidity sensor	0-Off 1-On	1
10/0x000A	R	Current status of fan operation by temperature sensor	0-Off 1-On	1
11/0x000B	R	Current status of fan operation by motion sensor	0-Off 1-On	1
12/0x000C	R	Current status of fan operation by signal from an external switch	0-Off 1-On	1
13/0x000D	R	Current status of fan operation in interval ventilation mode	0-Off 1-On	1
14/0x000E	R	Current status of fan operation in SILENT mode	0-Off 1-On	1
15/0x000F	R/W/RW	Permission of operation based on humidity sensor readings	0-Off 1-in automatic mode 2-in manual mode	1
17/0x0011	R/W/RW	Permission of operation based on temperature sensor readings	0-Off 1-On 2-Invert	1
18/0x0012	R/W/RW	Permission of operation based on motion sensor readings	0-Off 1-On 2-Invert	1
19/0x0013	R/W/RW	Permission of operation based on signal from an external switch	0-Off 1-On 2-Invert	1
24/0x0018	R/W/RW/INC/DEC	Max speed setpoint	30...100 %	1
26/0x001A	R/W/RW/INC/DEC	Silent speed setpoint	30...100 %	1

Parameter number [Dec./Hex.]	Functions	Description	Possible values	Size [bytes]
27/0x001B	R/W/RW/INC/DEC	Interval ventilation speed setpoint	30...100 %	1
29/0x001D	R/W/RW	Interval ventilation mode activation	0-Off 1-On 2-Invert	1
30/0x001E	R/W/RW	Silent mode activation	0-Off 1-On 2-Invert	1
31/0x001F	R/W/RW	Silent Mode start time in seconds	0...86400 seconds	3
32/0x0020	R/W/RW	Silent Mode end time in seconds	0...86400 seconds	3
33/0x0021	R/W/RW	Current time of the fan internal clock in seconds	0...86400 seconds	3
35/0x0023	R/W/RW/INC/DEC	Turn-off delay timer/BOOST setpoint	0-Off 2-5 minutes 3-15 minutes 4-30 minutes 6-60 minutes	1
36/0x0024	R/W/RW/INC/DEC	Turn-on delay timer setpoint	0-Off 1-2 minutes 2-5 minutes	1
37/0x0025	W	Resetting parameters to factory settings	Any byte	1
124/0x007C	R	Device search on the local Ethernet network	Text ("0...9", "A...F")	16
134/0x0086	R	Controller base firmware version and date	Byte 1-firmware version (major) Byte 2-firmware version (minor) Byte 3-day Byte 4-month Byte 5 and Byte 6-year	6
148/0x0094	R/W/RW	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	Wi-Fi frequency channel	1...13	1
155/0x009B	R/W/RW	Wi-Fi module DHCP	0-STATIC 1-DHCP 2-Invert	1
156/0x009C	R/W/RW	IP address assigned to Wi-Fi module	Byte 1-0...255 Byte 2-0...255 Byte 3-0...255 Byte 4-0...255	4

Parameter number [Dec./Hex.]	Functions	Description	Possible values	Size [bytes]
157/0x009D	R/W/RW	Wi-Fi module subnet mask	Byte 1-0...255 Byte 2-0...255 Byte 3-0...255 Byte 4-0...255	4
158/0x009E	R/W/RW	Wi-Fi module main gateway	Byte 1 - 0...255 Byte 2 - 0...255 Byte 3 - 0...255 Byte 4 - 0...255	4
160/0x00A0	W	Apply new Wi-Fi parameters and quit Wi-Fi module Setup Mode	Any byte	1
163/0x00A3	R	Current Wi-Fi module IP address	0...255	4
185/0x00B9	R	Unit type		2

EXAMPLE OF PROCESSING PACKETS WRITTEN 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 and start of packet check ***********//
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]; // Position in array where FUNC block begins
                //********************* FUNC and 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 lead byte value for parameter numbers
                    else if(receive_data[r_pos] == BGCP_CMD_PAGE)
                    {
                       page = receive_data[++r_pos];
                       continue;
                    }
                    //=== New parameter size value
                    else if(receive_data[r_pos] == BGCP_CMD_SIZE)
                    {
                        param_size = receive_data[++r_pos];
                        r_pos++;
                    }
                    //=== If the parameter is not supported
                    else if(receive_data[r_pos] == BGCP_CMD_NOT_SUP)
                    {
                        r_pos++;
                        //******* Processing of non-supported parameters ******//
                        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;
                }
                //***********************************************************//
            }
        }
    }
}
```

V168-1EN-01