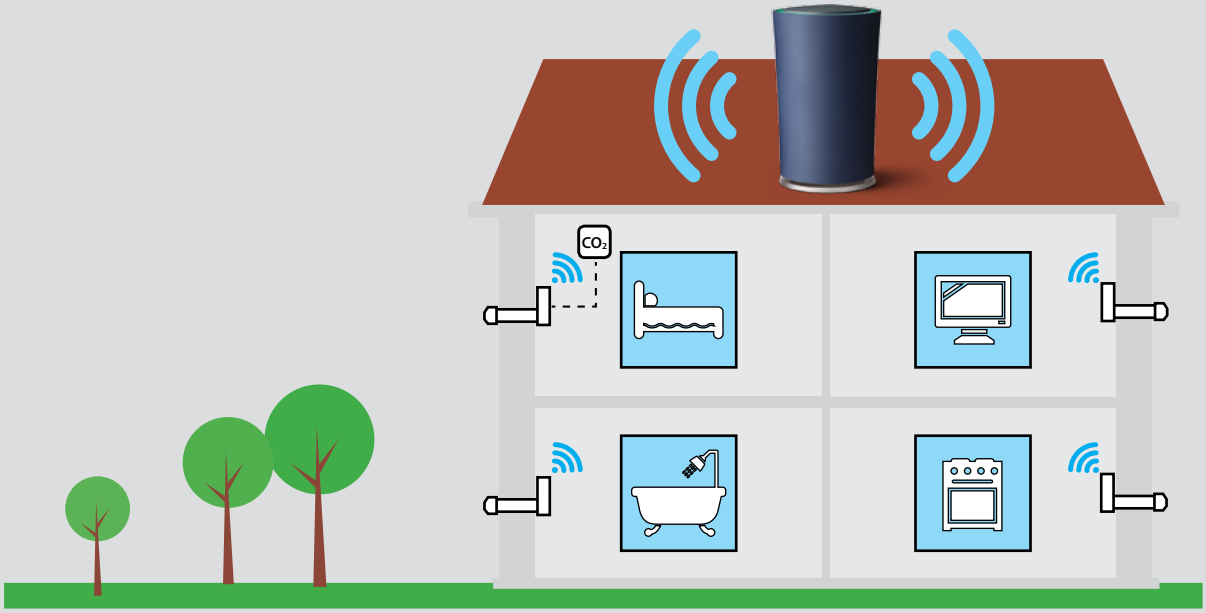
CONNECTION GUIDE

MICRA 100 WiFi



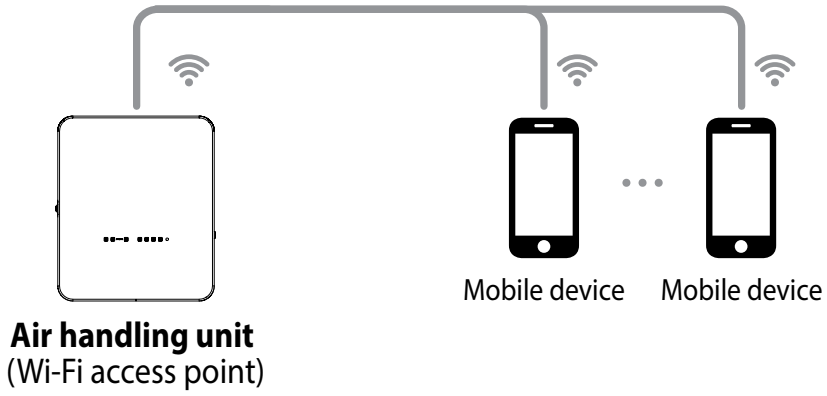
Connection to a "Smart Home" system

VENTS

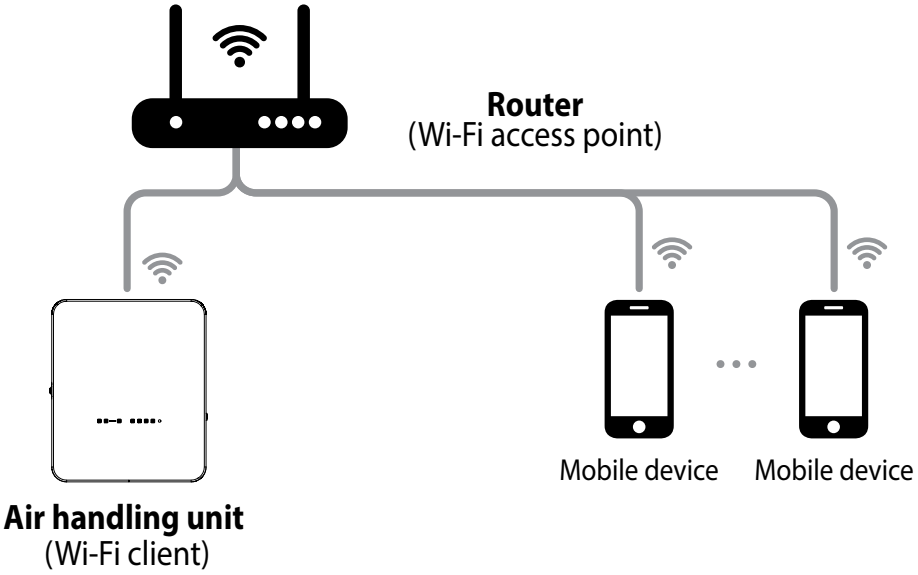
CONTENTS

CONNECTION AND SETUP

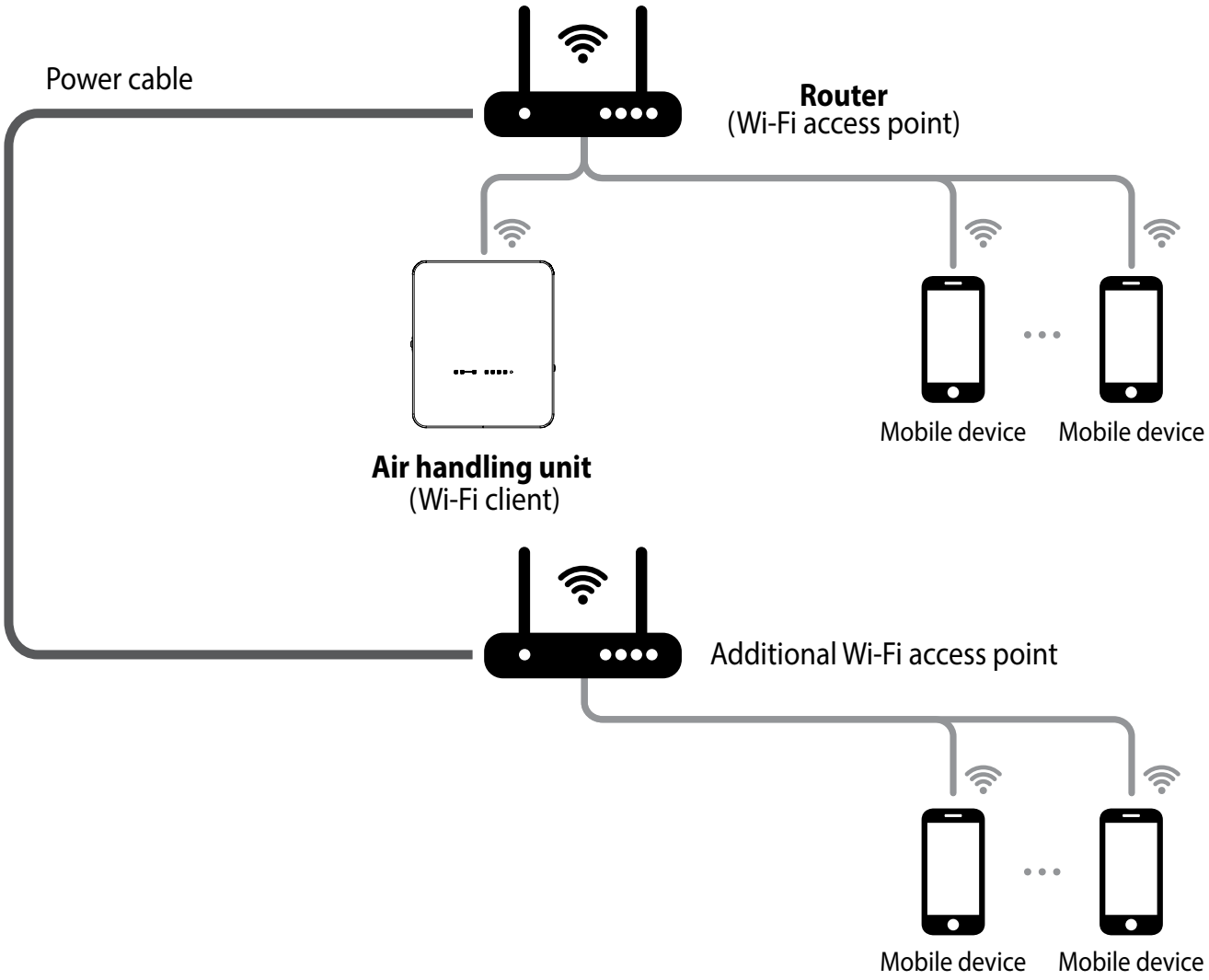
Example 1: pattern of direct connection of the unit to the Smart Home system without using a router.
Set up the unit to operate Wi-Fi in the access point mode (see the User's manual for the unit).
Note: maximum possible number of connected control devices is eight.



Example 2: connection via router with one Wi-Fi access point.
Air handling units, mobile devices and a "Smart Home" system are connected to the Wi-Fi access point of the network router.



Example 3: 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 the mobile application (see unit data sheet) 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) consists 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.

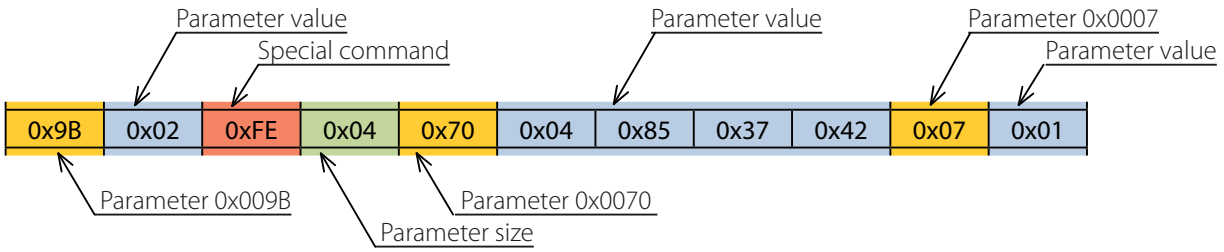
Value :parameter value (the default size of value is 1 byte). Byte ordering from least significant byte to most significant byte..

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

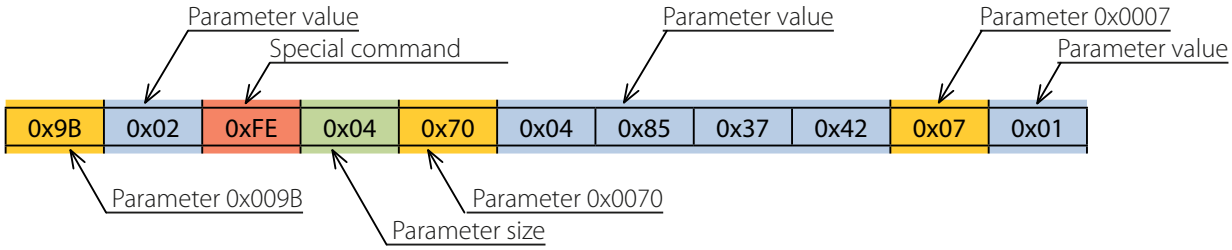
Write request (FUNC = 0x03) for parameters number 0x009B, 0x0070, and 0x0007



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.

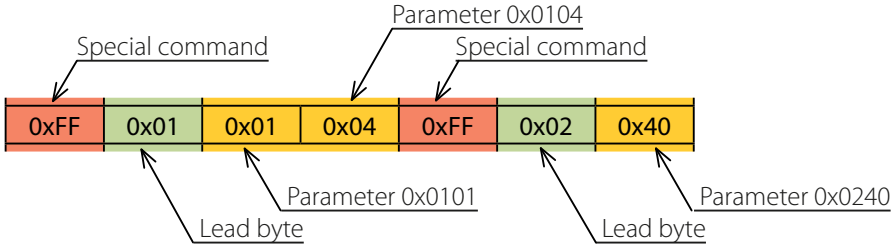
Controller response (FUNC = 0x06) to write request



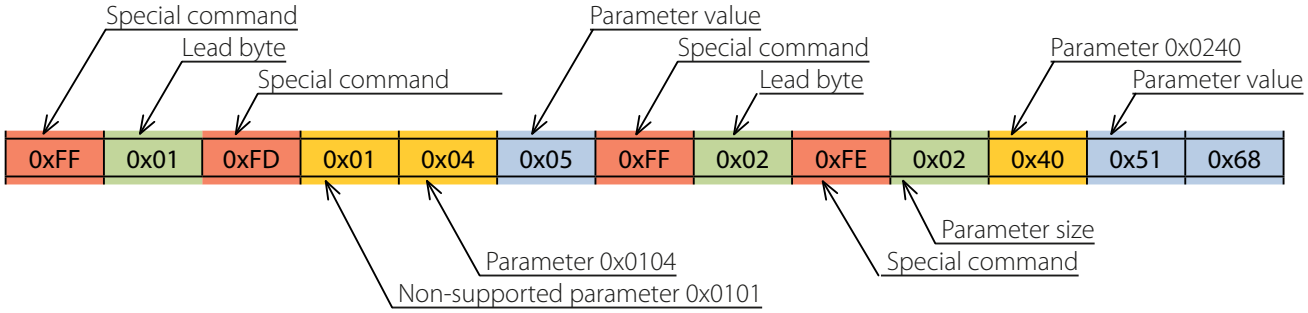
Controller response details:

- 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



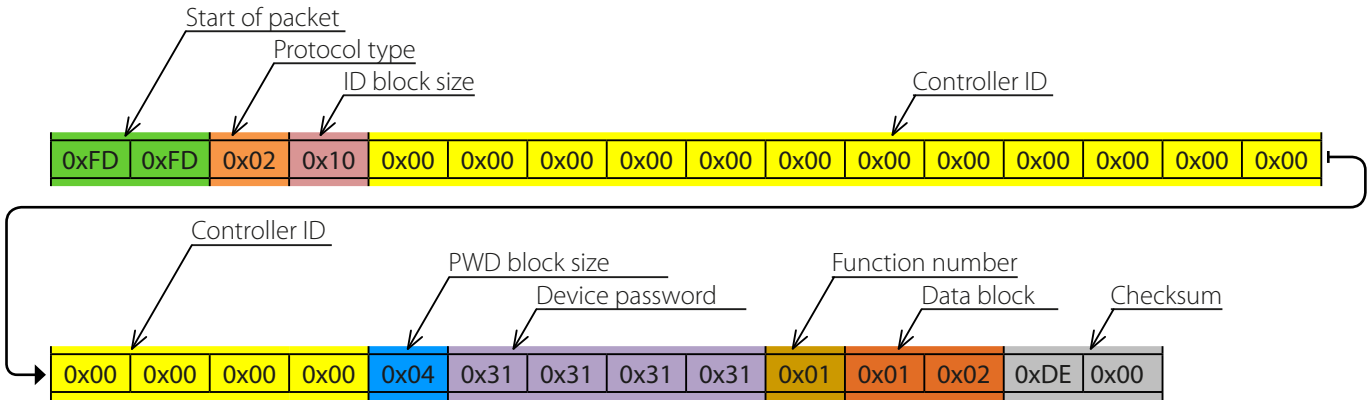
Controller response details:

- 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.

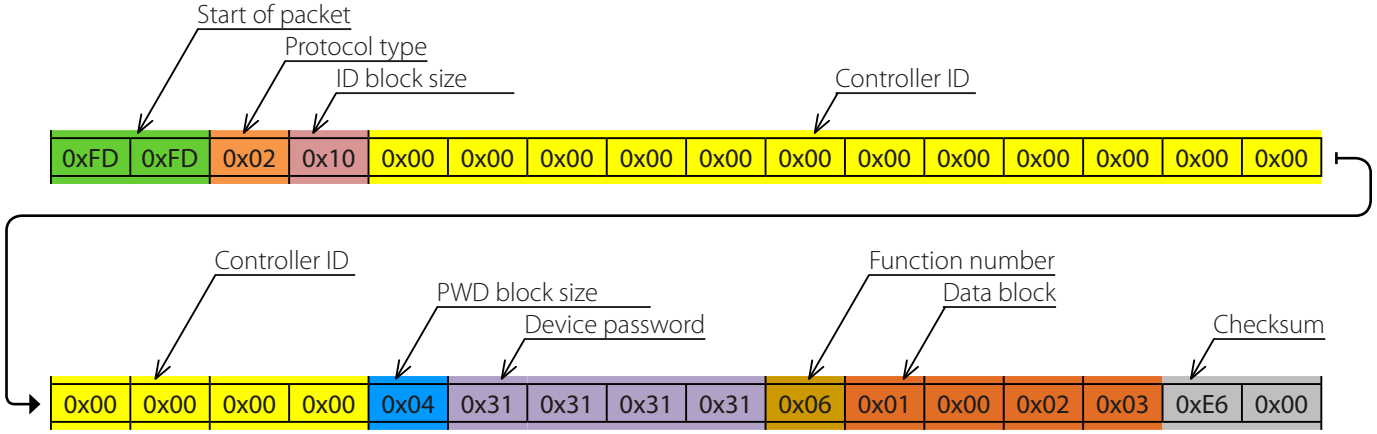


Request details:

- Checksum: 0x00DE.

Sending "Controller –> Smart Home" packet

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



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	Unit On/Off	0 - Off 1 - On 2 - Invert	1
2/0x0002	R/W/RW/INC/DEC	Speed mode	1 - Speed 1 2 - Speed 2 3 - Speed 3 4 - Speed 4 5 - Speed 5	1
3/0x0003	R/W/RW/INC/DEC	Maximum speed number	3, 5	1
6/0x0006	R	Boost-mode. The unit switches to Boost speed for the Boost mode turn-off delay time (see parameters 70, 71, 102)	0 - Off 1 - On 2 - Invert	1
7/0x0007	R/W/RW	Timer On/Off	0 - Off 1 - On 2 - Invert	1
8/0x0008	R/W/RW/INC/DEC	Timer mode	0 - Standby 1 - Speed 1 2 - Speed 2 3 - Speed 3 4 - Speed 4 5 - Speed 5	1
9/0x0009	R/W/RW/INC/DEC	Timer setpoint (minutes)	0...59 min	1
10/0x000A	R/W/RW/INC/DEC	Timer setpoint (hours)	0...23 hours	1
11/0x000B	R	Current timer countdown time	Byte 1 - seconds (0…59) Byte 2 - minutes (0…59) Byte 3 - hours (0…23)	3
13/0x000D	R/W/RW/INC/DEC	Room temperature setpoint in timer mode	0 - ventilation only, +15 ... + 30 °C	1
20/0x0014	R/W/RW	Control with a BOOST switch	0 - Off 1 - On 2 - Invert	1
21/0x0015	R/W/RW	Control by a fire alarm sensor	0 - Off 1 - On 2 - Invert	1
24/0x0018	R/W/RW/INC/DEC	Room temperature setpoint in normal mode	+15...+30 °C	1
29/0x001D	R/W/RW/INC/DEC	Selecting a temperature sensor for controlling room temperature	0 - in exhaust duct (ExAirIn), 1 - external sensor in the control panel (Ext), 2 - in supply duct (SuAirOut)	1
30/0x001E	R	Current temperature controlling the room temperature	-32768 - sensor is missing +32767 - short circuit	signed 2 (must be divided by 10, one decimal place)

Parameter number [Dec./Hex.]	Functions	Description	Possible values	Size [bytes]
31/0x001F	R	Current intake air temperature at the unit inlet	-32768 - sensor is missing +32767 - short circuit	signed 2 (must be divided by 10, one decimal place)
32/0x0020	R	Current supply air temperature at the unit outlet (downstream of the heat exchanger/downstream of the heater)	-32768 - sensor is missing +32767 - short circuit	signed 2 (must be divided by 10, one decimal place)
33/0x0021	R	Current extract air temperature at the unit inlet	-32768 - sensor is missing +32767 - short circuit	signed 2 (must be divided by 10, one decimal place)
34/0x0022	R	Current exhaust air temperature at the unit outlet	-32768 - sensor is missing +32767 - short circuit	signed 2 (must be divided by 10, one decimal place)
50/0x0032	R	Current Boost switch status	0 - Off 1 - On	1
51/0x0033	R	Current fire alarm sensor status	0 - Off 1 - On	1
54/0x0036	R/W/RW/INC/DEC	Minimum fan speed	0...100 %	1
55/0x0037	R/W/RW/INC/DEC	Minimum fan speed	0...100 %	1
58/0x003A	R/W/RW/INC/DEC	Supply fan speed in Speed 1 mode	min…max %	1
59/0x003B	R/W/RW/INC/DEC	Extract fan speed in Speed 1 mode	min … max %	1
60/0x003C	R/W/RW/INC/DEC	Supply fan speed in Speed 2 mode	min … max %	1
61/0x003D	R/W/RW/INC/DEC	Extract fan speed in Speed 2 mode	min … max %	1
62/0x003E	R/W/RW/INC/DEC	Supply fan speed in Speed 3 mode	min … max %	1
63/0x003F	R/W/RW/INC/DEC	Extract fan speed in Speed 3 mode	min … max %	1
64/0x0040	R/W/RW/INC/DEC	Supply fan speed in Speed 4 mode	min … max %	1
65/0x0041	R/W/RW/INC/DEC	Extract fan speed in Speed 4 mode	min … max %	1
66/0x0042	R/W/RW/INC/DEC	Supply fan speed in Speed 5 mode	min … max %	1
67/0x0043	R/W/RW/INC/DEC	Extract fan speed in Speed 5 mode	min … max %	1
69/0x0045	R/W/RW/INC/DEC	Fan speed while blowing electric heaters	min … max %	1
70/0x0046	R/W/RW/INC/DEC	Supply fan speed in BOOST mode	min … max %	1

Parameter number [Dec./Hex.]	Functions	Description	Possible values	Size [bytes]
71/0x0047	R/W/RW/INC/DEC	Extract fan speed in BOOST mode	min … max %	1
96/0x0060	R/W/RW/INC/DEC	Main heater type	0 - turn off 1 - electric (fixed value)	1
99/0x0063	R/W/RW/INC/DEC	Setting filter replacement timer	0, 70 .. 365 days with an interval of 5 days	2
100/0x0064	R	Timer countdown to filter replacement	Byte 1 - minutes (0…59) Byte 2 - hours (0…23) Byte 3 and Byte 4 - days (0…365)	4
101/0x0065	W	Reset timer countdown to filter replacement	Any byte	1
102/0x0066	R/W/RW/INC/DEC	Setpoint of the Boost mode turn-off delay	0 … 60 min	1
103/0x0067	R/W/RW/INC/DEC	Setpoint of the Boost mode turn-on delay	0 … 15 min	1
104/0x0068	R/W/RW	Temperature control in normal mode	0 - Off 1 - On 2 - Invert	1
106/0x006A	R	TE5 temperature	-32768 - sensor is missing +32767 - short circuit	signed 2 (must be divided by 10, one decimal place)
111/0x006F	R/W/RW	RTC time	Byte 1 - RTC seconds Byte 2 - RTC minutes Byte 3 - RTC hours	3
112/0x0070	R/W/RW	RTC calendar	Byte 1 - RTC number Byte 2 - RTC day of the week Byte 3 - RTC month Byte 4 - RTC year	4
114/0x0072	R/W/RW	Weekly schedule mode	0 - Off 1 - On 2 - Invert	1
115/0x0073	R	Weekly schedule speed	0 - Standby 1 - Speed 1 2 - Speed 2 3 - Speed 3 4 - Speed 4 5 - Speed 5	1
116/0x0074	R	Weekly schedule temperature setup	0 - only ventilation, +15 ... +30 °C	1

Parameter number [Dec./Hex.]	Functions	Description	Possible values	Size [bytes]
119/0x0077	R/W/RW	Schedule setup	Byte 1 – 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) Byte 2 period number: 1 ... 4 Byte 3 speed number: 0 - standby 1 ... 5 Byte 4 - temperature: 0 - ventilation only, +15...+ 30 °C Byte 5 - minutes to end of period: 0 ... 59 Byte 6 - hours to end of period: 0 ... 23	6
124/0x007C	R	Device search on the local Ethernet network	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 hours	Byte 1 - minutes (0…59) Byte 2 - hours (0…23) Byte 3 and Byte 4 - days (0…65535)	4
127/0x007F	R	List of current alarms/warnings	Byte 1 - code Byte 1 - type: 1- alarm 2 - warning	0, 2, 4…
128/0x0080	W	Reset the alarms	Any byte	1
129/0x0081	R	Heater status	0 - Off 1 - On	1
131/0x0083	R	Alarm/warning indicator	0 - no alarms 1 - alarm (highest priority) 2 - warning	1
133/0x0085	R/W/RW	Control through the cloud server	0 - Off 1 - On 2 - Invert	1
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

Parameter number [Dec./Hex.]	Functions	Description	Possible values	Size [bytes]
135/0x0087	W	Restore factory settings	Any byte	1
136/0x0088	R	Filter condition	0 - clean 3 - filter replacement timer has been activated	1
147/0x0093	R	Presence of Wi-Fi module on the circuit board	0 - not available 1 - available	1
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
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
159/0x009F	R/W/RW	DNS server address for Wi-Fi module	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
161/0x00A1	R	Status of the Wi-Fi module connection to the access point of the router	0 - not connected 1 - connected	1
162/0x00A2	W	Exit Wi-Fi Setup mode without using new Wi-Fi settings	Any byte	1
163/0x00A3	R	Current Wi-Fi module IP address	Byte 1 - 0…255, Byte 2 - 0…255, Byte 3 - 0…255, Byte 4 - 0…255	4
182/0x00B6	R	Status of electric heater blowing (preheating, reheating)	0 - Off 1 - On	1
185/0x00B9	R	Unit type	0x0002	2

Parameter number [Dec./Hex.]	Functions	Description	Possible values	Size [bytes]
240/0x00F0	R/W/RW/INC/DEC	Recirculation damper	0 - recirculation off 1 - recirculation on (only for units with recirculation)	1
252/0x00FC		Special command		
253/0x00FD		Special command		
254/0x00FE		Special command		
255/0x00FF		Special command		
273/0x0111	R	Control device type		2
274/0x0112	R	Control panel 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
1024/0x0400	R/W/RW	Button backlight brightness setpoint	0..80 (20-100 %)	1
1025/0x0401	R/W/RW	Turn on/off the sound generator on the circuit board	0 - Off 1 - On	1
1026/0x0402	R/W/RW	Backlight mode selection	0 - static mode 1 - dynamic mode	1

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;
            }
            //*************************************************************//
        }
    }
}
```

V73-9-1EN-01