eHouse LAN, WiFi, PRO
Communication protocol over TCP/IP+UDP

Date 2016.10.24. Most current revision is available at:
http://www.isys.pl/download/ehouse-lan-protocol-en.pdf

Open Protocol for communication to eHouse LAN (Ethernet), WiFi, PRO (Ethernet or WiFi) Home Automation system

eHouse LAN, WiFi, PRO Home Automation / Building Management system enables direct communication to the controller from external hardware/software (SmartPhones, PC, PADS, Web Browsers, Smart TV, etc) with current protocol via:

- **TCP/IP** – sending control commands/events with/without authentication Client-> Server (eHouse Controller)
- **UDP** – eHouse controllers (LAN, WiFi) statuses and logs are broadcasted with non connection protocol over LAN/WiFi network (eHouse Controller → Remote Management Panels)

Command/Event submission protocol over TCP/IP:
TCP/IP protocol has been used requiring Client connection (PC, SmartPhone, Pad or other Smart Devices) to Server (eHouse LAN, WiFi, PRO Controller).
Communication Steps Client → Server;
1) Client establish communication over TCP/IP (sockets) with Server on default port 9876 (although it can be changed for whole installation and all controllers, we suggest to remain it unchanged for comfort usage)
2) Server sends unique 6 bytes of **challenge** with time stamp used for authorization to controller for most secure methods (Challenge code)
3) Client send response for the challenge in following form frame:

- repeated 6 bytes of challenge, received from the server (index 0..5)
- 6 bytes of response depending on authorization type (*): (index 6..11)
- character 0x0d (13) (index 12)
- size of commands – multiply by 10 (max 16 events/command) (index 13)

4) Client is waiting for confirmation response from server (Controller):
- '+' - command/event was added to execution queue
- '-' - command/event was NOT added to execution queue, eg. no space left (**)
- NO response from server or timeout (**)

5) Client initiate closing connection sending 0x00 to the server
6) Client disconnect TCP/IP connection from the server
(**) - all communication steps must be repeated after min. 1s.

Authorization methods to eHouse LAN controllers (*):

1. Challenge-Response – dynamic code authorization
2. XOR Password – dynamically hashed password
3. Plain Password – simple password
4. No Authorization – None

Activation of a particular method of verification must be set in the configuration of each controller. It is permitted only selected type of authentication and safer authorizations.
For example: Selecting the authentication method **Challenge-Response** for controller blocks all other methods. Selecting **XOR Password** authentication allows the use of : **XOR Password** and **Challenge-Response**.

Challenge-Response: dynamically changed code (unique for each installation)
The client receives Challenge (6B) from server after connecting.
Client sends repeated challenge from the server (6B) and add encrypted reply (6B) in the following form (Server Challenge XOR unique installation code, eHouse calculated with unique data). Both the server code and the answer is unique and includes a time stamp. Commands with a long delay are ignored. This mode is available only in the original applications eHouse: eHouse.exe (RS-485), eHouseWiFi.exe (LAN, WiFi) and eHousePRO. In the case of large installations and license agreements it is possible to generate an individual algorithm takes into account the License Code (Vendor Code), a unique and granted to each Partner. We do not distribute full authentication algorithm to keep strong security level of eHouse system.

XOR Password: dynamically encrypted password (hashed by variable code):
The client receives challenge (6B) from the server after connecting.
Client sends repeated challenge from the server (6B) and add an encrypted reply (6B) in the following form (Server Challenge XOR password for the installation). Both the server code and the answer is unique and includes a time stamp and commands with a large delay are ignored. The password is not sent directly by communication links (LAN, WiFi, Internet). This mode allows you to create your own software and secure authentication to the controller of eHouse applications: eHouse Java, Android and individual client applications to access from Internet. This authentication algorithm is made available to clients and source code in many programming languages can be found at http://www.isys.pl/download/ .

Simplified example:

```
unsigned char ServerChallenge[6];          //Server response after connection of client
unsigned char Password[6];                 //password for verification 6 bytes
unsigned char ClientResponse[170];         //response to server from client
unsigned char CommandToSend[10];           //command/event to send to server
int socket;                                //bsd socket
unsigned char buff[20];                    //receive of client buffer
recv(socket,ServerChallenge,6,0);
memcpy(ClientResponse,ServerChallenge,6);        //copy ServerChallenge to client responce
for (char i=0;i<6;i++)                           //hashing client response with password
        {
        ClientResponse[6+i]=ServerChallenge[i]^Password[i];
        }
ClientResponse[12]=13;      //static field
ClientResponse[13]=10;      //size of one command (10B)
memcpy((unsigned char *)&ClientResponse[14],CommandToSend,10); //copy command code
send(socket,ClientResponse,14+ClientResponse[13],0);    //send to BSD socket
buff[0]=0;
if (recv(socket,buff,1,0)<0)    //error receiving response
        {
        close(socket);
        return -1;
        }
else
        if (buff[0]=='+')
                {       //event received  by server confirmation OK
                }
        else            //no confirmation of event/command
                {       //should be retried
                }
        buff[0]=0;                      //Send Close command and disconnect
        send(socket,buff,1,0);
        flush(socket);
        Delay100ms;
        close(socket);
```

Plain Password: Simple password:
The client receives challenge (6B) from the server after connecting.
Client sends repeated challenge from the server (6B) and open password for the system (6B). Both the server code and the answer is unique, however, the password is sent directly by communication links (LAN, WiFi, Internet). This eliminate repeated packets and re-transmissions and simple sabotage. This mode allows you to create your own software and a simple authorization to the controller in the individual customer's applications and XOR-Password applications eHouse: eHouse Java, Android. It should be avoided when using public links (Internet). This authentication algorithm is made available to clients and source code in many programming languages can be found at http://www.isys.pl/download/ .

Simplified example:

```
unsigned char ServerChallenge[6];          //Server response after connection of client
unsigned char Password[6];                 //password for verification 6 bytes
unsigned char ClientResponse[170];         //response to server from client
unsigned char CommandToSend[10];           //command/event to send to server
int socket;                                //bsd socket
unsigned char buff[20];                    //receive of client buffer
recv(socket,ServerChallenge,6,0);          //receive Challenge from server
memcpy(ClientResponse,ServerChallenge,6);      //copy ServerChallenge to client responce
for (char i=0;i<6;i++)                         //adding plain password
        {
        ClientResponse[6+i]=Password[i];       //plain password without encryption
        }
ClientResponse[12]=13;      //static field
ClientResponse[13]=10;      //size of one command (10B;
memcpy((unsigned char *)&ClientResponse[14],CommandToSend,10); //copy command code
send(socket,ClientResponse,14+ClientResponse[13],0);    //send to BSD socket
buff[0]=0;
if (recv(socket,buff,1,0)<0)    //error receiving response
        {
        close(socket);
        return -1;
        }
else
        if (buff[0]=='+')
                {       //event received  by server confirmation OK
                }
        else            //no confirmation of event/command
                {       //should be retried
                }
        buff[0]=0;                  //Send Close command and disconnect
        send(socket,buff,1,0);
        flush(socket);
        Delay100ms;
        close(socket);
```

No Authorization: No authorization to controller:
1) The client receives the challenge (6B) from the server after connecting.
2) Client sends repeated response from the server (6B) and what Possible (6B) to keep the length of the communication frame. The data passing through the communication links (LAN, WiFi, Internet) are not checked by the server. This mode allows you to test your own software in the initial stage of development without logging in to the controller. Also allows for simple authentication (PLAIN-Password) to the controller in the individual customer's applications and XOR-Password applications eHouse: eHouse Java, Android. It should not be used in production systems especially with Internet access. It is available to customers with source code in many programming languages can be found at http://www.isys.pl/download/ .

Simplified example:

```
unsigned char ServerChallenge[6];          //Server response after connection of client
unsigned char Password[6];                 //password for verification 6 bytes
unsigned char ClientResponse[170];         //response to server from client
unsigned char CommandToSend[10];           //command/event to send to server
int socket;                                //bsd socket
unsigned char buff[20];                    //receive of client buffer
recv(socket,ServerChallenge,6,0);          //receive Challenge from server
//memcpy(ClientResponse,ServerChallenge,6);      //copy ServerChallenge to client responce
//for (char i=0;i<6;i++)                         //adding plain password
//      {
//      ClientResponse[6+i]=Password[i];         //plain password without encryption
//      }
//ClientResponse[12]=13;     //static field*/
for (char i=0;i<13;i++) {ClientResponse[6+i]=0;}//anything here to keep proper frame length
ClientResponse[13]=10;       //size of one command (10B;
memcpy((unsigned char *)&ClientResponse[14],CommandToSend,10); //copy command code
send(socket,ClientResponse,14+ClientResponse[13],0);     //send to BSD socket
buff[0]=0;
if (recv(socket,buff,1,0)<0)    //error receiving response
        {
        close(socket);
        return -1;
        }
else
        if (buff[0]=='+')
                {       //event received  by server confirmation OK
                }
        else            //no confirmation of event/command
                {       //should be retried
                }
                buff[0]=0;                  //Send Close command and disconnect
                send(socket,buff,1,0);
                flush(socket);
                Delay100ms;
                close(socket);
```

Protocol of events/commands for eHouse System

Standard event / command of eHouse system (DirectEvent) have a length of 10B (bytes)
Meaning of the bytes of the event:
1) ADRH - Address H - high component controller address (MSB)
2) ADRL - Address L - low component controller address (LSB)
3) command / event code
4) argument 1 for the command
5) argument 2 for the command
6) argument 3 for the command
7) argument 4 for the command
8) argument 5 for the command
9) argument 6 for the command
10) argument 7 for the command

Due to the frequent need to integrate different variants of eHouse system we apply some standardization and simplify addressing concept. This is especially important for addresses allocated to drivers and standard ports of eHouse. Due to the long-term development of eHouse system since 2000 and the need to combine different communication interfaces, it was necessary to standardize the use of the latest versions could natively support the standards used in the earlier versions and enabling development and expansion after years.

Although, you can give any address to controllers ADRH, ADRL in range (1..254) 0x01..0xfe you should avoid changing component ADRH, because it is used by eHouse system applications to detect variant of controller (RS-485, CAN, RF PRO , LAN, WiFi).

ADR H (1, 2, 55) – eHouse RS-485
ADR H (0x7f..0x8f) – eHouse CAN
ADR H (0x70..0x7e) – eHouse RF
Other ADRH addresses - eHouse LAN, WiFi, PRO (the same for each controller for working in network mask **255.255.255.0**. IP address is created following way: **192.168.ADRH.ADRL**
For LAN, WiFi, Pro controller even ADRL has meaning:
ADR L (100..199) – by default is allocated to eHouse WiFi
ADR L (200) – by default is allocated to eHouse PRO
ADR L (201..249) – by default is allocated to eHouse LAN (EthernetRoomManager)
ADR L (250..254) –by default is allocated to eHouse LAN (CommManager)

Commands / events depend on the type of eHouse architecture (LAN, PRO, CAN / RF, WiFi, RS-485) and the type of driver
(CommManager, RoomManager, HeatManager, small controllers)

The commands contained in this document apply only to eHouse LAN drivers : (CommManager, LevelManager, EthernetRoomManager).

Commands eHouse LAN: – depends of controller variant LM, CM, ERM and firmware version

Name	Decimal Code	Meaning
EXECPROFILE	2	execute output program change
SETLIGHT	4	set dimmer
SETADC	12	set ADC levels
SETOUTPUT	33	change output state
IRALIAS	39	send IR command
INCDECPROFILE	96	Increment / decrements Program
ADC_PROGRAM	97	set ADC program
SECU_PROGRAM	98	run rollers program + zone – CM only
ZONE_CHANGE	99	zone change - CM only
ROLLERSSINGLE	100	single rollers execution – CM only
ROLLERSMULTI	101	rollers program execution – CM only
BATCH_ROM	104	run multiple events from ROM bank
MODIFY_LIGHS	105	Change Dimmers Settings
SETSINGLELIGHT	106	Set single Dimmer Value
SET_TIME	107	Obsolete
SETDMX	108	Set DMX Single Chanel
SETDMXRGB	109	Set DMX 3 channels
SENDDALICMD	242	Send DALI Command
SETDALI	243	Set DALI Single Chanel
SETDALIRGB	244	Set DALI RGB
MODADC	245	Modify ADC Levels
SETDALISCENE	246	Set DALI Program/scene
SETDMXSCENE	247	Set DMX Program/scene
EVENT_SUBMIT_TCP_STATUS	248	send DATA to log analyzer if connected
EVENT_SEND_TCP	249	send DATA to opened TCP Query connection
EVENT_SEND_UDP	250	broadcast DATA via UDP
EVENT_SEND_TO_LOG	251	send DATA to log analyzer if sconnected
SENDIRCOMMAND	252	send infrared code (RC Control)
SCAN_IR	253	scan IR remote controller code
RESETDEV	254	reset device

Byte No. (0..9) and meaning

0	1	2	3	4	5	6	7	8	9
ADRH	**ADRL**	**CMD**	**Arg 1**	**Arg 2**	**Arg 3**	**Arg 4**	**Arg 5**	**Arg 6**	**Arg 7**

Controller Address **Command** **Arguments for current eHouse command**

EXECPROFILE	2	execute output program change

Run program/light scene for on/off outputs and LED dimmers

0	1	2	3	4	5	6	7	8	9
ADRH	**ADRL**	**2**	**Arg 1**	0	0	0	0	0	0

CMD - 2
Arg 1 – number of program 0..23

SETLIGHT	4	Set Dimmer

Set LED/RGB dimmer values - 3 channel

0	1	2	3	4	5	6	7	8	9
ADRH	**ADRL**	**4**	**Arg 1**	**Arg 2**	**Arg 3**	**Arg 4**	0	0	0

CMD = 4
Arg 1 – first channel no.
Arg 2 – dimmer value R (0..100)
Arg 3 – dimmer value G (0..100)
Arg 4 – dimmer value B (0..100)

SETADC	12	Set ADC levels

Set high/low thresholds for ADC (measurement/regulation)

0	1	2	3	4	5	6	7	8	9
ADRH	**ADRL**	**12**	**Arg 1**	**Arg 2**	**Arg 3**	**Arg 4**	**Arg 5**	0	0

CMD - 12
Arg 1 – ADC input no. 1..16
Arg 2 – higher byte MSB of low threshold
Arg 3 – lower byte LSB of low threshold
Arg 4 – higher byte MSB of high threshold
Arg 5 - lower byte LSB of high threshold

Values of thresholds are calculated as:
ADCLow =(Arg2<<8) +Arg3
ADCHigh =(Arg4<<8) +Arg5

Value ADCLow, ADCHigh of thresholds as ADC 10 bit binary data <0..1024) depends on the sensor type and the scale of mapping. For standard sensors (voltage measurement, temperature sensor, lighting we suggest to use data from **eHouseWiFi.exe** application).

SETOUTPUT	33	execute output change

Set single on/off output

0	1	2	3	4	5	6	7	8	9
ADRH	**ADRL**	**33**	**Arg 1**	**Arg 2**	**Arg 3**	**Arg 4**	0	0	0

CMD – 33 or 1
Arg 1 – output number 1..128
Arg 2 – output state
0 – off
1 – on
2 – toggle (invert state)
Arg 3 – Timeout LSB
Arg 4 – Timeout MSB

Timeout=(Arg4<<8)+Arg3
Timeout – time in seconds for automatic turning off output

IRALIAS	39	Send IR Command

Send IR macro - 4 IR codes from local database

0	1	2	3	4	5	6	7	8	9
ADRH	**ADRL**	**39**	**Arg 1**	**Arg 2**	**Arg 3**	**Arg 4**	0	0	0

CMD – 39
Arg 1 – index of first code
Arg 2 – index of second code
Arg 3 – index of third code
Arg 4 – index of fourth code

INCDECPROFILE	96	Increments/decrements Program

Increments / Decrements number of program, light scene

0	1	2	3	4	5	6	7	8	9
ADRH	**ADRL**	**96**	**Arg 1**	0	0	0	0	0	0

CMD - 96
Arg 1 – direction
1 – increments by 1
2 – decrements by 1

ADC_PROGRAM	97	execute ADC measurement program change

Run ADC Measurement / Regulation program

0	1	2	3	4	5	6	7	8	9
ADRH	**ADRL**	**97**	**Arg 1**	0	0	0	0	0	0

CMD – 97
Arg 1 – number of ADC program 0..11

SECU_PROGRAM	**98**	**execute security/rollers program change**

Drive Security/Servo/Drive program with zone (CM only)

0	1	2	3	4	5	6	7	8	9
ADRH	**ADRL**	**98**	**Arg 1**	0	0	0	0	0	0

CMD - 98
Arg 1 – number of program 0..23

ZONE_CHANGE	**99**	**execute security zone change**

Set Security Zone (CM)

0	1	2	3	4	5	6	7	8	9
ADRH	**ADRL**	**99**	**Arg 1**	**Arg 2**	0	0	0	0	0

CMD - 99
Arg 1 – number of security zone 0..23
Arg 2 – delay 0 - none, other ~30s

ROLLERSSINGLE	100	execute drive/servo change

Change roller/servo state (CM only)

0	1	2	3	4	5	6	7	8	9
ADRH	**ADRL**	**100**	**Arg 1**	**Arg 2**	0	0	0	0	0

CMD - 100
Arg 1 – drive number n=(0..35)
Arg 2 – direction:
0 – no movement
1 – output activation **(n*2+1) / Open**
2 – output activation **(n*2+2) / Close**
3 – STOP
Somfy – both outputs activation **(n*2+1)** & **(n*2+2)**
DIRECT – both outputs deactivation **(n*2+1)** & **(n*2+2)**

To control single drive, 2 lines (output) numbers (n*2+1) and (n*2+2) are used. CM driver configuration must be set - all outputs as drives (Somfy or Direct).
Caution: setting improper mode drives Somfy vs. Direct can damage the drive. If Somfy mode is chosen, 2 directions lines are activated simultaneously for (STOP) condition.

ROLLERSMULTI	101	execute multiple drives change

Simultaneous running rollers settings (only CM)

0	1	2	3	4	5	6	7	8	9
ADRH	ADRL	101	Arg 1	Arg 2	Arg 3	Arg 4	Arg 5	Arg 6	Arg 7

Arg 1..7 – setting drive line – 1 bit for each output/direction) bit-wise notation.

ArgX (1..7) => n =X-1 (0..6) - bits

Bit [MSb..LSb]	7	6	5	4	3	2	1	0
Output No.	8n+8	8n+7	8n+6	8n+5	8n+4	8n+3	8n+2	8n+1
Drive No.	N/4+4	N/4+4	N/4+3	N/4+3	N/4+2	N/4+2	N/4+1	N/4+1
Direction	Close	Open	Close	Open	Close	Open	Close	Open

To control drive no. **x** 2 lines (output) numbers (x*2+1) and (x*2+2) are used. CM driver configuration must be set - all outputs as drives (Somfy or Direct).
Caution: setting improper mode drives Somfy / Direct can damage the drive. If Somfy mode is chosen 2 directions lines are activated simultaneously for (STOP) condition.

BATCH_ROM	104	execute batch of commands/events

Running Macro of multiple commands from Flash memory of controller

0	1	2	3	4	5	6	7	8	9
ADRH	**ADRL**	**104**	**Arg 1**	**Arg 2**	0	0	0	0	0

CMD - 104
Arg 1 – no. of events from memory of Macros (0..max)
Arg 2 – count of events to run (1..max)

MODIFY_LIGHTS	105	Modify Dimmers Settings

Modify LED i DMX dimmers (ERM only)

0	1	2	3	4	5	6	7	8	9
ADRH	**ADRL**	**105**	**Arg 1**	**Arg 2**	**Arg 3**	0	0	0	0

CMD - 105
Arg 1 – dimmer no. 1..3 (internal LED dimmer) and 4..24 (external DMX dimmer)
Arg 2 – operation
0 - set decrease mode of brightness level (to stop or reach the minimum level) with a step change in **Arg 3** field
1 - set to increase mode of brightness level (to stop or reach the maximum level) with a step change in the **Arg 3** field
2 - stop (stop change)
3 - setting the value of the field **Arg 3**
4 - switch dimmer - if movement than stops, if was stopped it triggers a change in the opposite direction to the last change
5 - reduces once the level of the dimmer value by step in **Arg 3**
6 - increases once the level of the dimmer value by step in **Arg 3**
Arg 3 - the value of step when changing or setting level

SETSINGLELIGHT	106	Set single dimmer light level

Set Light Level for internal LED dimmer or external DMX dimmer

0	1	2	3	4	5	6	7	8	9
ADRH	**ADRL**	**106**	**Arg 1**	**Arg 2**	0	0	0	0	0

CMD - 106
Arg 1 – no. dimmer (1..max) 1..3 (internal), 4..max (external DMX)
Arg 2 – level of dimmer (0..100%)

SETDMX	108	Set DMX dimmer settings

Change DMX dimmer settings

0	1	2	3	4	5	6	7	8	9
ADRH	**ADRL**	**108**	**Arg 1**	**Arg 2**	**Arg 3**	0	0	0	0

CMD - 108
Arg 1 – no. (3..Max)
Arg 2 – operation
0 – set decrements mode by step in **Arg 3** field
1 – set increment mode by step in **Arg 3** field
2 – stop movement of dimmer
3 – set dimmer value from **Arg 3** field
Arg 3 – step or value for dimmer

SETDMXRGB	109	Set DMX RGB Dimmer

Set dimmer DMX / RGB (ERM only)

0	1	2	3	4	5	6	7	8	9
ADRH	**ADRL**	**109**	**Arg 1**	**Arg 2**	**Arg 3**	**Arg 4**	0	0	0

CMD - 109
Arg 1 – dimmer no. DMX/RGB * 3 channel (1..Max)
Arg 2 – dimmer level R (Red)
Arg 3 – dimmer level G (Green)
Arg 4 – dimmer level B (Blue)
Note: For proper work of following function it is necessary to place DMX RGB dimmers at the beginning of a number to the dimmer DMX / RGB was at DMX = 3 * (n-1) +1

SENDDALICMD	242	Send DALI Command

Send DALI command to DALI Bus

0	1	2	3	4	5	6	7	8	9
ADRH	**ADRL**	**242**	**Arg 1**	**Arg 2**	0	0	0	0	0

CMD - 242
Arg 1 – command MSB
Arg 2 – command LSB

SETDALI	243	Set DALI dimmer

Set value of DALI dimmer

0	1	2	3	4	5	6	7	8	9
ADRH	**ADRL**	**243**	**Arg 1**	**Arg 2**	**Arg 3**	0	0	0	0

CMD - 243
Arg 1 – DALI channel no.(1..Max)
Arg 2 – operation
0 – lowering the brightness of the value by **Arg 3** field
1 – increasing the brightness of the value in the **Arg 3** field
3 - set level of dimmer from **Arg 3** field
Arg 3 – step or value for dimmer

SETDALISCENE	246	Set DALI light scene/program

Run DALI Program / Light Scene

0	1	2	3	4	5	6	7	8	9
ADRH	**ADRL**	**246**	**Arg 1**	0	0	0	0	0	0

CMD - 246
Arg 1 – no. DALI program/light scene (1..Max)

SETDMXSCENE	247	Set DMX light scene / program

Set DMX program / light scene

0	1	2	3	4	5	6	7	8	9
ADRH	**ADRL**	**247**	**Arg 1**	0	0	0	0	0	0

CMD - 247
Arg 1 – DMX program / light scene (1..Max)

EVENT_SUBMIT_TCP_STATUS	248	Submit data with TCP/IP & UDP Status

Add DATA to submission of controller status

0	1	2	3	4	5	6	7	8	9
ADRH	ADRL	248	Arg 1	Arg 2	Arg 3	Arg 4	Arg 5	Arg 6	Arg 7

CMD – 248
Arg 1..7 – Binary Data to place in status (STATUS_MORE)

EVENT_SEND_TCP	249	Send Data as Log to TCP/IP Logger

Send 7 bytes to TCPLogAnalyzer

0	1	2	3	4	5	6	7	8	9
ADRH	ADRL	249	Arg 1	Arg 2	Arg 3	Arg 4	Arg 5	Arg 6	Arg 7

CMD - 249
Arg 1 ..7 – 7 bytes of Data to submit to TCPLogAnalyzer application

EVENT_SEND_UDP	**250**	**Send / broadcast Data as log via UDP protocol**

Broadcast 7 bytes of data via UDP as log

0	1	2	3	4	5	6	7	8	9
ADRH	**ADRL**	**250**	**Arg 1**	**Arg 2**	**Arg 3**	**Arg 4**	**Arg 5**	**Arg 6**	**Arg 7**

CMD - 250
Arg 1 ..7 – 7 bytes to broadcast via UDP protocol

EVENT_SEND_TO_LOG	251	Send Data to Log

Send 7 bytes of data to Log

0	1	2	3	4	5	6	7	8	9
ADRH	ADRL	251	Arg 1	Arg 2	Arg 3	Arg 4	Arg 5	Arg 6	Arg 7

CMD - 251
Arg 1 ..7 – 7 bytes to send to Log

SENDIRCOMMAND	252	Send IR Command

Send IR code

0	1	2	3	4	5	6	7	8	9
ADRH	ADRL	252	Arg 1	Arg 2	Arg 3	Arg 4	Arg 5	Arg 6	Arg 7

CMD - 252
Arg 1 – IR Standard (captured by eHouseWiFi application)
Arg 2..7 – IR Code (captured by eHouseWiFi application) MSB..LSB

SCAN_IR	253	SCAN IR Command

Scan IR Command (capture/learn)

0	1	2	3	4	5	6	7	8	9
ADRH	ADRL	253	0	0	0	0	0	0	0

CMD - 253

RESETDEV	254	Reset Device

Reset controller

0	1	2	3	4	5	6	7	8	9
ADRH	ADRL	254	0	0	0	0	0	0	0

CMD - 254

Decoding status of eHouse LAN controllers

Status of eHouse LAN controllers are broadcast automatically via UDP protocol - port 6789 (without connection) to all devices available on LAN network. This port can be changed but we do not recommend changing this port unless it is necessary. The status of the drivers may be sent to the connected TCP/IP panel (client) as well in the same form / frame.

Status Frame

Size [1B]	AddrH [1B]	AddrL [1B]	Type [1B]	Date [Size B]	CheckSum [2B]
			s	Status	
			l	Log Data	

Indexes of status bytes:

Meaing	[Index]		{size}
STATUS_SIZE	**[0]**		**{1B}**
STATUS_ADDRH	**[1]**		**{1B}**
STATUS_ADDRL	**[2]**		**{1B}**
STATUS_CODE	**[3]**	**// 's'**	**{1B}**
STATUS_DIMMERS	[4]		{20B}
STATUS_DMX_DIMMERS	[STATUS_DIMMERS+3]		{17B}
STATUS_ADC_LEVELS	[24]		{32B}
STATUS_MORE	[56]	//Additional DATA	{8B}
STATUS_ADC2_LEVELS	[64]		{8B}
STATUS_ADC_ETH	**[72]**	**//ADC Measurements results**	**{16 * 2B}**
STATUS_ADC_ETH_END	**[STATUS_ADC_ETH+32]**	**[104]**	
STATUS_OUT_I2C	[STATUS_ADC_ETH_END]	//CM only	{20B}
STATUS_OUT	[STATUS_ADC_ETH_END] [104]	//ERM only	{5B}
STATUS_DMX_DIMMERS2	[STATUS_OUT_I2C+5] [109]	//ERM only	{15B}
STATUS_DALI2	[STATUS_OUT_I2C+5]	//ERM only	{15B}
STATUS_INPUTS_I2C	[STATUS_OUT_I2C+20] [124]	//CM only	{12B}
STATUS_INPUTS	[STATUS_INPUTS_I2C] [124]	//ERM only	{2B}
STATUS_DIMMERS3	[STATUS_INPUTS_I2C+2] [126]	//ERM only	{22B}
STATUS_DALI	[STATUS_INPUTS_I2C+2] [126]	//ERM only	{22B}
STATUS_ALARM_I2C	[STATUS_INPUTS_I2C+12] [136]	//CM only	{12B}
STATUS_WARNING_I2C	[STATUS_ALARM_I2C+12] [148]	//CM only	{12B}
STATUS_MONITORING_I2C	[STATUS_WARNING_I2C+12] [160]	//CM only	{12B}
STATUS_PROGRAM_NR	**[STATUS_MONITORING_I2C+12] [172]**		**{1B}**
STATUS_ZONE_NR	[STATUS_PROGRAM_NR+1] [173]	//CM only	{1B}
STATUS_ADC_PROGRAM	**[STATUS_ZONE_NR+1]**	**[174]**	**{1B}**

eHouse LAN ANY Controllers
eHouse LAN CommManager (CM/LM) ONLY
eHouse LAN EthernetRoomManager (ERM) ONLY

Bits Locations in STATUS_OUT, STATUS_INPUTS fields

Index / Bit Locations	7	6	5	4	3	2	1	0
STATUS_OUT	O8	O7	O6	O5	O4	O3	O2	O1
STATUS_OUT+1	O16	O15	O14	O13	O12	O11	O10	O9
STATUS_OUT+2	O24	O23	O22	O21	O20	O19	O18	O17
STATUS_OUT+3	O32	O31	O30	O29	O28	O27	O26	O25
STATUS_OUT+4						O35	O34	O33
STATUS_INPUTS	IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1
STATUS_INPUTS+1					IN12	IN11	IN10	IN9

Bit fields for CM/LM

STATUS_OUT_I2C + (0..12)
STATUS_INPUTS_I2C + (0..12) - active on/off inputs
STATUS_ALARM_I2C + (0..12) - horn initiated inputs (alarm output)
STATUS_WARNING_I2C + (0..12) – warning light initiated inputs
STATUS_MONITORING_I2C + (0..12) - monitoring initiated inputs (monitoring output)