R004_b

UPWT Protocol – connecting to UPWT LED displays

1. Basic information

Data transmission is performed, using RS232, in one direction (to the display) at a rate of 9600 bps (parameters 8N1). Or in both directions using RS485 (9600 bps, parameters 8N1) and Ethernet (TCP, UDP, default port 2101).

There are five types of communication frames:

- Frame for sending text „international” and characters indicating weight (of fixed width),
- Frame for sending brightness settings,
- Frame giving the display an address for communication RS232/CL, RS422, RS485,
- Frame for the command of displaying the address for communication RS232/CL, RS422, RS485,
- Frame for the command of checking the correctness of the display operation.

2. Display communication protocol

The frame for sending text „international” and characters indicating weight (of fixed width) looks as follows:

Byte No.	0	1	2	3–9	10 : 10+(2*(x+1))	10+(2*x)+2 : 10+(2*x)+5
Description	1st byte of frame start	2nd byte of frame start	3rd byte of frame start	Control data	User text characters with a terminating character	Sum CRC_8 or CRC_32

The following is the detailed construction of the data frame:

Byte No.		Value [dec]	Description
0		8	First byte of frame start
1		12	Second byte of frame start
2		116	Third byte of frame start – command identifier
3		0-1	Display line number (for a display with a single line the assumed value is 1)
4		0-4	scroll_0: 0 – auto scroll – automatic recognition if the text is to be scrolled. If it is to be displayed statically, it will be aligned to the right edge of the display 1 – scroll force – forces scrolling the text, even if it does fit the display and could be shown as static 2 – scroll off – text always static – if it is too long, the beginning of the text will be aligned to the left side of the display and the ending of the text (not fitting) will not be shown 3 – scroll auto center – automatic recognition if the text is to be scrolled. If it is to be displayed statically, it will be centered. 4 – scroll off center – text always static, centered
5	Bits: 1-3	0-4	timeout - the indication on the display breaks in communication for more than: 0 - option disabled, 1 - 10 seconds, 2 - 30 seconds, 3 - 60 seconds, 4 - 180 seconds.
	Bits: 0	0-1	scroll_1: 0 - text is changed immediately after receiving the frame, 1 - text is changed after the previous text has flown .
6		0-19	Text scroll speed – essential when scroll_0 equals 0 or 1

7	0-1, 8	Should CRC_32 sum be checked: 0 – do not check CRC sum 1 – check CRC_8 sum 8 – check CRC_32 sum	
8	0-255	Target device address (0 – default address, 255 – broadcast)	
9	0	Free byte – to use later	
10 : 11+(2*(x-1)) x>0	String with a code page	User text, a character contains 2 bytes because of fonts of different languages and a symbol of fonts of fixed width. The text has variable length, it can contain maximum 100 letters (200 bytes). If the inscription does not contain any characters (x=0), then, in the field number 10 and 11, there is a terminating character	
10+(2*x) + 0	0	Terminating character (user text end symbol)	
10+(2*x) + 1	0		
10+(2*x) + 2	0–255	MSB CRC_32 calculated from bytes (3 - 10+(2*x) + 1)	If the byte on position 7 equals 0, CRC_32 values may have any value, but must still be sent .
10+(2*x) + 3	0–255	CRC_32 calculated from bytes (3 - 10+(2*x) + 1)	
10+(2*x) + 4	0–255	CRC_32 calculated from bytes (3 - 10+(2*x) + 1))	
10+(2*x) + 5	0–255	LSB CRC_32 calculated from bytes (3 - 10+(2*x) + 1) CRC_8 – if the value of field 7 equals 8, then fields: 10+(2*x) + 2, 10+(2*x) + 3, 10+(2*x) + 4 should have value 0	

Sample frames: „#008#012#116#001#000#001#009#000#255#000$00-$00#143$001$003$005$00$00$01$02$03$04”
„#008#012#116#001#001#001#009#000#255#000$00#143$00T$00e$00x$00t$00$00$01$02$03$04”
„#008#012#116#001#002#001#009#000#255#000$00#142$00>$14#002$14#003$14#005$00$00$01$02$03$04”

For testing purposes, users can use Hercules SETUP utility http://www.hw-group.com/products/hercules/index_en.html
Use *Ethernet cable then* select *TCP Client* then *Connect* 192.168.0.11 and Port 2101).
Note: We offer no guarantee or support concerning Hercules. Hercules is Freeware - owned and distributed by hw-group.com

The frame for sending brightness settings for the computer system:

Byte No.	0	1	2	3–7	8
Description	1st byte of frame start	2nd byte of frame start	3rd byte of frame start	Control data	Sum CRC_8

The following is the detailed construction of the display brightness settings' frame:

Byte No.	Value [dec]	Description
0	8	First byte of frame start
1	12	Second byte of frame start
2	124	Third byte of frame start
3	0-10;	Brightness control: 0 – automatic control is on, data from field 4 are significant 1-10 – manually set fixed brightness (the biggest number indicates the greatest brightness)
4	0-3	Brightness control: 0-3 – automatic control set profile (0– profile 1… 3 – profile 4) – field is significant when field 3 has value 0
5	0	Free byte – to use later
6	0-255	Target device address (0 – default address, 255 – broadcast)
7	0-1	Should CRC_8 sum be checked: 0 – do not check CRC sum 1 – check CRC_8 sum
8	0–255	CRC_8 calculated from bytes from 3 – 7. It is essential when the value of field 7 equals 1

Sample frames:

„#008#012#124#010#000#000#255#000#000” // broadcast command: set brightness
„#008#012#125#128#000#000#000#000#000” //command to display of address 0: change brightness

The frame giving the display the address for communication RS232/CL, RS422, RS485:

Byte No.	0	1	2	3–6	7
Description	1st byte of frame start	2nd byte of frame start	3rd byte of frame start	Control data	Sum CRC_8

The following is the detailed construction of the frame giving the display the address for communication RS232/CL, RS422, RS485:

Byte No.	Value [dec]	Description
0	8	First byte of frame start
1	12	Second byte of frame start
2	125	Third byte of frame start
3	0-254	New display address
4	0	Free byte – to use later
5	0-255	Target device address (0 – default address, 255 – broadcast)
6	0-1	Should CRC_8 sum be checked: : 0 – do not check CRC sum 1 – check CRC_8 sum
7	0–255	CRC_8 calculated from bytes from 3 – 6. It is essential when the value of field 6 equals 1

Sample frames:

„#008#012#125#128#000#255#000#000”   // broadcast command: new address 128
„#008#012#125#128#000#000#000#000”   //command to display of address 0: change address to 128

The frame for the command of displaying the address for communication RS232/CL, RS422, RS485:

Byte No.	0	1	2	3–5	6
Description	1st byte of frame start	2nd byte of frame start	3rd byte of frame start	Control data	Sum CRC_8

The following is the detailed construction of the frame for the command of displaying the address for communication RS232/CL, RS422, RS485:

Byte No.	Value [dec]	Description
0	8	First byte of frame start
1	12	Second byte of frame start
2	126	Third byte of frame start
3	0-255	Target device address (0 – default address, 255 – broadcast)
4	0	Free byte – to use later
5	0-1	Should CRC_8 sum be checked: 0 – do not check CRC sum 1 – check CRC_8 sum
6	0–255	CRC_8 calculated from bytes from 3 – 5. It is essential when the value of field 5 equals 1

Sample frames:

„#008#012#126#255#000#000#000”   // broadcast command: show address
„#008#012#126#000#000#000#000”   //command to display of address 0: show address

The frame of the command of checking the correctness of the display operation:

Byte No.	0	1	2	3–5	6
Opis	1st byte of frame start	2nd byte of frame start	3rd byte of frame start	Control data	Sum CRC_8

The following is the detailed construction of the frame for the command of checking the correctness of the display operation:

Byte No.	Value [dec]	Description
0	8	First byte of frame start
1	12	Second byte of frame start

2	127	Third byte of frame start
3	0-255	Target device address (0 – default address, 255 – broadcast)
4	0	Free byte – to use later
5	0-1	Should CRC_8 sum be checked: 0 – do not check CRC sum 1 – check CRC_8 sum
6	0–255	CRC_8 calculated from bytes from 3 – 5. It is essential when the value of field 5 equals 1

„#008#012#127#255#000#000#000”   // broadcast command: show address
„#008#012#127#000#000#000#000”   // command to display of address 0: show address

2.1. Confirmation in two-way communication

Confirmation of receiving data for two-way communication:

- ‘O’ - correct data reception
- ‘C’ - CRC sum error
- ‘E’ - transmitted data error (command recognized as wrong or argument values from outside the range)

3. CRC checksum calculation

3.1. crc8 checksum calculation

In listing 3.1 an exemplary code for calculation of crc8 checksum for three bytes: byte1, byte 2 and byte3 was presented. crc_8 checksum makes XOR of byte1, byte2 and byte3.

```
uint8_t crc_8;
uint8_t byte1=1;
uint8_t byte2=2;
uint8_t byte3=3;

int main(void)
{
        crc_8 = 0; // zeroing of the variable containing the controlled sum value

        crc_8 ^= byte1;
        crc_8 ^= byte2;
        crc_8 ^= byte3;
// after the above operations crc_8 variable will contain the calculated checksum value
        return 0;
}
```

List. 3.1. An example of a code performing calculation of CRC8 checksum

3.2. crc32 checksum calculation

An exemplary code performing calculation of CRC32 checksum is shown in listing 3.2.

```
uint32_t crc_32;
uint8_t crc_32_shift;
uint8_t byte1=1;
uint8_t byte2=2;
uint8_t byte3=3;

int main(void)
{
        crc_32_shift = 0;
        crc_32 = 0; // zeroing of the variable containing the controlled sum value

        crc_32 ^= ((uint32_t) byte1) << crc_32_shift++;
        if (crc_32_shift > 24) crc_32_shift = 0;

        crc_32 ^= ((uint32_t) byte2) << crc_32_shift++;
        if (crc_32_shift > 24) crc_32_shift = 0;

        crc_32 ^= ((uint32_t) byte3) << crc_32_shift++;
        if (crc_32_shift > 24) crc_32_shift = 0;

// after the above operations crc_32 variable will contain the checksum value calculated for three bytes: byte1, byte2, byte3

        return 0;
}
```

List. 3.2. An example of a code performing calculation of CRC32 checksum

Each data byte is converted to 32-byte form. The 32-byte form is rolled left by value of crc_32_shift variable. This is followed by XOR operation for crc_32 variable and previously rolled variable. The crc32_shift is 8-byte variable, which is incremented after each rolling operation. If crc32_shift variable value will exceed 24 it should be zeroed.