1. Features

- Firmware for RGB LED controller
- Voltage control (PWM) in 255 steps (0-255) for each channel
- MASTER channel adjustable in 255 steps as well
- Possibility to set up minimum and maximum value for each channel
- Last state memory
- Adjustable dimming time for each channel
- 37 control instructions
- 3 blocking instructions
- 4 timers (1 for each channel) for instruction execution delay 1s-24h
- Allows defining up to 128 CAN messages which can indirectly control the module
- Settable power up states
- Uptime counter
- Health check monitor
- Transmit (42 messages) and receive (42 messages) FIFO buffers

2. Compatibility

- Firmware for **UNIV 3.8.0.x** module
- Firmware can be uploaded into processor with bootloader version 3.1 or compatible.

3. Firmware version

UNIV 3 . 8 . 0 . 0

- 0 – firmware version
- 0 – application version
- 8 – application type (RGB LED controller)
- 3 – processor version
- UNIV – universal processor

4. Communication Frames (messages)

4.1. RGB LED controller message

The module sends message to the bus, when the status of any channel changes.

Table 1. RGB LED controller frame (0x308)

Frame type	Flags				Module	Group	D0	D1	D2	D3	D4	D5	D6	D7
0x308	3	2	1	0	Node Nr	Group Nr	0xFF	0xFF	CHANNEL	STATUS	RELAY	INSTR1	INSTR2	TIMER

0x308 – universal module frame, RGB LED controller

- 3 - - not used flag, read as "0"
- 2 - - not used flag, read as "0"
- 1 - - not used flag, read as "0"
- 0 RE - response flag, flag is equal "1" if node was requested. If flag is equal „0" it means that status of output has just changed.

- Node Nr - message sender node number
- Group Nr - message sender group number
- CHANNEL - output channel (0x01 – red, 0x02 – green, 0x03 – blue, 0x04 - MASTER)
- STATUS - current status of output 0x00 – 0xFF
- PRZEK - current status of all channels (0x00 – if all channels are turned off, 0xFF – if master channel and any colour channe is on) – this byte can be used to toggle relay which turns on the power supply for the LED. This byte is only available in MASTER channel frame
- INSTR1 - instruction that is waiting for execution, or 0xFF if none instruction
- INSTR2 - second byte of instruction that is waiting for execution, or 0xFF
- TIMER - delay value of waiting instruction, or 0x00 if none waiting

4.2. Status request

Status of module can be checked by sending from computer STATUS REQUEST frame (0x109) (Table 2).

Table 2. STATUS REQUEST frame (0x109).

Frame type	Flags	Module	Group	D0	D1	D2	D3	D4	D5	D6	D7
0x109	0x0	COMP ID1	COMP ID2	0xXX	0xXX	Node Nr	Group Nr	0xXX	0xXX	0xXX	0xXX

0x1090 – STATUS REQUEST frame

COMP ID1 - computer identifier (must be unique on the network)
COMP ID2 - computer identifier (must be unique on the network)

Node Nr - node number of requested module

Group Nr - group number of requested module

0xXX - inessential data

As response the module will send RGB LED frames. The meaning of bytes is the same as in Table 1.

Table 3. Response to STATUS REQUEST

Frame type	Flags	Module	Group	D0	D1	D2	D3	D4	D5	D6	D7
0x308	0x1	Node Nr	Group Nr	0xFF	0xFF	0x01	STATUS	0xFF	INSTR1	INSTR2	TIMER1
0x308	0x1	Node Nr	Group Nr	0xFF	0xFF	0x02	STATUS	0xFF	INSTR1	INSTR2	TIMER2
0x308	0x1	Node Nr	Group Nr	0xFF	0xFF	0x03	STATUS	0xFF	INSTR1	INSTR2	TIMER3
0x308	0x1	Node Nr	Group Nr	0xFF	0xFF	0x04	STATUS	RELAY	INSTR1	INSTR2	TIMER4

4.3. Uptime request

Table 4. UPTIME REQUEST (0x113).

Frame type	Flags	Module	Group	D0	D1	D2	D3	D4	D5	D6	D7
0x113	0x0	COMP ID1	COMP ID2	0xXX	0xXX	Node Nr	Group Nr	0xXX	0xXX	0xXX	0xXX

0x1130 – UPTIME REQUEST frame

COMP ID1 - computer identifier (must be unique on the network)
COMP ID2 - computer identifier (must be unique on the network)

Node Nr - node number of requested module

Group Nr - group number of requested module

0xXX - inessential data

Table 5. Response to UPTIME REQUEST (0x113).

Frame type	Flags	Module	Group	D0	D1	D2	D3	D4	D5	D6	D7
0x113	0x1	Node Nr	Group Nr	0xFF	0xFF	0xFF	0xFF	UPTIME3	UPTIME2	UPTIME1	UPTIME0

0x1131 – Response to UPTIME REQUEST frame

Node Nr - node number on the network
Group Nr - group number of the node on the network

UPTIME - (UPTIME3*256^3+UPTIME2*256^2+UPTIME1*256^1+UPTIME3*256^0) in seconds

4.4. Health check request

Table 6. HEALTH CHECK - STATUS REQUEST (0x115).

Frame type	Flags	Module	Group	D0	D1	D2	D3	D4	D5	D6	D7
0x115	0x0	COMP ID1	COMP ID2	0x01	0xXX	Node Nr	Group Nr	0xXX	0xXX	0xXX	0xXX

0x1150 – HEALTH CHECK REQUEST frame

COMP ID1 - computer identifier (must be unique on the network)
COMP ID2 - computer identifier (must be unique on the network)

0x01 - status request

Node Nr - node number of requested module

Group Nr - group number of requested module

0xXX - inessential data

As response the module will send two frames (Table 7).

Table 7. Response to HEALTH CHECK - STATUS REQUEST (0x115).

Frame type	Flags	Module	Group	D0	D1	D2	D3	D4	D5	D6	D7
0x115	0x1	Node Nr	Group Nr	0x01	RXCNT	TXCNT	RXCNTMX	TXCNTMX	CANINTCNT	RXERRCNT	TXERRCNT

0x1151 – Response to HEALTH CHECK REQUEST frame

Node Nr - node number on the network
Group Nr - group number of the node on the network

0x01 - frame 1 (current values)
RXCNT - current level of receive FIFO buffer
TXCNT - current level of transmit FIFO buffer
RXCNTMX - maximum level of receive FIFO buffer since power up
TXCNTMX - maximum level of transmit FIFO buffer since power up
CANINTCNT - number of CAN interface restarts since power up
RXERRCNT - current receive errors register
TXERRCNT - current transmit errors register

Frame type	Flags	Module	Group	D0	D1	D2	D3	D4	D5	D6	D7
0x115	0x1	Node Nr	Group Nr	0x02	0xFF	0xFF	RXCNTMXE	TXCNTMXE	CANINTCNTE	RXERRCNTE	TXERRCNTE

0x1151 – Response to HEALTH CHECK REQUEST frame

Node Nr - node number on the network
Group Nr - group number of the node on the network

0x02 - frame 2 (maximum values saved in eeprom memory)
RXCNTMXE - maximum ever level of receive FIFO buffer
TXCNTMXE - maximum ever level of transmit FIFO buffer
CANINTCNTE - maximum ever number of CAN interface restarts
RXERRCNTE - maximum ever receive errors
TXERRCNTE - maximum ever transmit errors

To clear maximum values saved in eeprom memory the frame shown in Table 8 must be sent. There is no response to this message.

Table 8. HEALTH CHECK - CLEAR REQUEST (0x115).

Frame type	Flags	Module	Group	D0	D1	D2	D3	D4	D5	D6	D7
0x115	0x0	COMP ID1	COMP ID2	0x02	0xXX	Node Nr	Group Nr	0xXX	0xXX	0xXX	0xXX

0x1150 – HEALTH CHECK REQUEST frame

COMP ID1 - computer identifier (must be unique on the network)
COMP ID2 - computer identifier (must be unique on the network)

0x02 - clear request
Node Nr - node number of requested module
Group Nr - group number of requested module
0xXX - inessential data

5. Module control

The module can be controlled directly from PC, or indirectly by other modules.

5.1. Control instruction

The table below shows all instructions, which can be executed by the module. Some of them can be executed only with direct control and other with indirect control (through other modules).

Table 9. Module control instructions

Instrukcja	Instruction Codding								Note	Control	
	INSTR1	INSTR2	INSTR3	INSTR4	INSTR5	INSTR6	INSTR7	INSTR8		Direct	Indirect
SET LED1 (R) TO...	**0x00**	**STATE**	**TIMER**	0xXX	0xXX	0xXX	0xXX	0xXX	Sets the state of the channel immediately at the level specified by the byte **STATE** (0-255). Instructions may be delayed if the **TIMER** is not zero.	√	√
SET LED2 (G) TO...	**0x01**	**STATE**	**TIMER**	0xXX	0xXX	0xXX	0xXX	0xXX			
SET LED3 (B) TO...	**0x02**	**STATE**	**TIMER**	0xXX	0xXX	0xXX	0xXX	0xXX			
SET MASTER TO...	**0x03**	**STATE**	**TIMER**	0xXX	0xXX	0xXX	0xXX	0xXX			
TOGGLE LED1	**0x04**	0xXX	**TIMER**	0xXX	0xXX	0xXX	0xXX	0xXX	If the channel is on, it will be turned off. If it is off, it will be turned to the maximum or the last memorized value (if state memory is set in the configuration). Instructions may be delayed if the **TIMER** is not zero.	√	√
TOGGLE LED2	**0x05**	0xXX	**TIMER**	0xXX	0xXX	0xXX	0xXX	0xXX			
TOGGLE LED3	**0x06**	0xXX	**TIMER**	0xXX	0xXX	0xXX	0xXX	0xXX			
TOGGLE MASTER	**0x07**	0xXX	**TIMER**	0xXX	0xXX	0xXX	0xXX	0xXX			
STEP DOWN LED1 BY ...	**0x08**	**VALUE**	**TIMER**	0xXX	0xXX	0xXX	0xXX	0xXX	Channel state will be reduced by the value indicated in the **VALUE** byte.	√	√
STEP DOWN LED2 BY ...	**0x09**	**VALUE**	**TIMER**	0xXX	0xXX	0xXX	0xXX	0xXX			
STEP DOWN LED3 BY ...	**0x0A**	**VALUE**	**TIMER**	0xXX	0xXX	0xXX	0xXX	0xXX			
STEP DOWN MASTER BY ...	**0x0B**	**VALUE**	**TIMER**	0xXX	0xXX	0xXX	0xXX	0xXX			
STEP UP LED1 BY ...	**0x0C**	**VALUE**	**TIMER**	0xXX	0xXX	0xXX	0xXX	0xXX	Channel state will be increased by the value indicated in the **VALUE** byte.	√	√
STEP UP LED2 BY ...	**0x0D**	**VALUE**	**TIMER**	0xXX	0xXX	0xXX	0xXX	0xXX			
STEP UP LED3 BY ...	**0x0E**	**VALUE**	**TIMER**	0xXX	0xXX	0xXX	0xXX	0xXX			
STEP UP MASTER BY ...	**0x0F**	**VALUE**	**TIMER**	0xXX	0xXX	0xXX	0xXX	0xXX			
SET LED1 (R) SOFTLY TO...	**0x10**	**STATE**	**TIMER**	0xXX	0xXX	0xXX	0xXX	0xXX	Sets the state of the channel slowly to the level specified by the byte **STATE** (0-255). Instructions may be delayed if the **TIMER** is not zero.	√	√
SET LED2 (G) SOFTLY TO...	**0x11**	**STATE**	**TIMER**	0xXX	0xXX	0xXX	0xXX	0xXX			
SET LED3 (B) SOFTLY TO...	**0x12**	**STATE**	**TIMER**	0xXX	0xXX	0xXX	0xXX	0xXX			
SET MASTER SOFTLY TO...	**0x13**	**STATE**	**TIMER**	0xXX	0xXX	0xXX	0xXX	0xXX			
STOP LED1	**0x14**	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	It stops instruction which is being executed eg. START or SET SOFTLY TO ...	√	√
STOP LED2	**0x15**	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX			
STOP LED3	**0x16**	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX			
STOP MASTER	**0x17**	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX			
START LED1	**0x18**	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	Instruction START begins typical control process. If within 400ms from START instruction, controller receives STOP instruction then it toggles channel's state (exactly the same as TOGGLE instruction). If after that time instruction STOP is not received then channel is brightened (if previous state was min or 0), or channel is dimmed (if previous state was max). It gives possibility to control with one button. If button is pressed for less than 400ms the channel will turn on or off. If button is pressed for longer than 400ms, then channel will dim or brighten.	√	√
START LED2	**0x19**	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX			
START LED3	**0x1A**	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX			
START MASTER	**0x1B**	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX			
SET LED1 SPEED TO ...	**0x1C**	**VALUE**	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	It sets dimming time of particular channel. **VALUE** byte (0-255) defines time when channel changes its state from 0 to 255. This byte can hold value from 0 – 255 which corresponds to 1s – 256s.	√	√
SET LED2 SPEED TO ...	**0x1D**	**VALUE**	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX			
SET LED3 SPEED TO ...	**0x1E**	**VALUE**	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX			
SET MASTER SPEED TO ...	**0x1F**	**VALUE**	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX			
SET RGB TO...	**0x20**	**STATE1**	**STATE2**	**STATE3**	**TIMER**	0xXX	0xXX	0xXX	It sets immediately values of three colour channels. Instruction may be delayed if the **TIMER** is not zero.	√	√
SET SOFTLY RGB TO...	**0x21**	**STATE1**	**STATE2**	**STATE3**	**TIMER**	0xXX	0xXX	0xXX	It sets slowly values of three colour channels. Instruction may be delayed if the **TIMER** is not zero.	√	√
SET RGB SPEED TO...	**0x22**	**VALUE**	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	It sets dimming time of three colour channels. **VALUE** byte (0-255) defines time when channel changes its state from 0 to 255. This byte can hold value from 0 – 255 which corresponds to 1s – 256s.	√	√
RGB SPEED UP	**0x23**	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	It changes dimming speed of three colour channels to next lower value	√	√
RGB SPEED DOWN	**0x24**	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	It changes dimming speed of three colour channels to next higher value	√	√
PROGRAM	**0x25**	**0x00**	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	It stops any running program.	√	√
		0x01	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	Program 1 – every program calling causes colour changing with sequence: 1- red 2- green 3- blue 4- yellow 5- cyan 6- violet 7- white		
		0x02	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	Program 2 – first program calling runs automatic colour changing with current dimming speed beginning from red colour. The second program calling stops colour. Another calling starts colour changing with reversed order.		
ENABLE BOX	**0xDD**	**BoxX**	**BoxY**	0xXX	0xXX	0xXX	0xXX	0xXX	It enables chosen boxes – these boxes will be compared with next received message from the bus.		√
DISABLE BOX	**0xDE**	**BoxX**	**BoxY**	0xXX	0xXX	0xXX	0xXX	0xXX	It disables chosen boxes – these boxes will be passed when next message arrives from the bus.		√
TOGGLE BOX	**0xDF**	**BoxX**	**BoxY**	0xXX	0xXX	0xXX	0xXX	0xXX	It toggles boxes – enables when they are disabled and vice versa		√

0xXX – inessential data

BoxX	Note
0x00	- from Box 1
0x01	- from Box 2
...	
0x7F	- from Box 128

BoxY	Note
0x00	+ 0 -(and not anyone else)
0x01	+ 1 -(and 1 following)
...	
0x7F	+127 -(and 127 following)

5.2. Timer

Each channel has its own timer which can delay execution of the instruction. Delay can be chosen between 1s-24h set in TIMER register. Drawing below shows delay dependence of TIMER register.

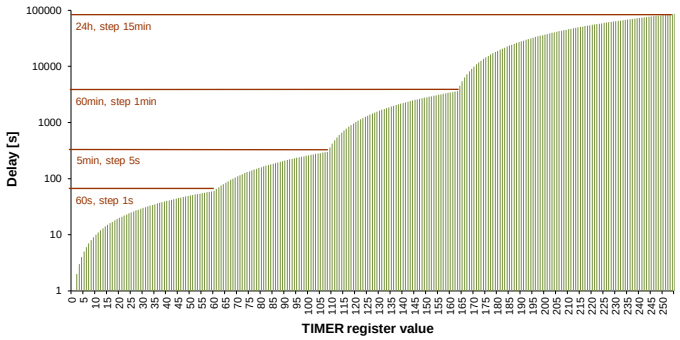

Figure 1. Delay/timer register relationship

5.3. Direct control

It is possible to control module by sending DIRECT CONTROL message. The message contains instruction, which will be executed by the module.

Table 10. DIRECT CONTROL frame (0x10A).

Frame type	Flags	Module	Group	D0	D1	D2	D3	D4	D5	D6	D7
0x10A	0x0	COMP ID1	COMP ID2	INSTR1	INSTR2	Node Nr	Group Nr	INSTR3	INSTR4	INSTR5	INSTR6

0x10A – DIRECT CONTROL frame

COMP ID1 - computer identifier (must be unique on the network)

COMP ID2 - computer identifier (must be unique on the network)

Node Nr - node number of requested module

Group Nr - group number of requested module

INSTR1-6 - instruction to be executed (byte1)

5.4. Indirect control

Indirect control means that module will react to messages sent by other modules on the network. It depends on configuration programmed into the module boxes (memory cells).
This firmware has feature to set simple conditions of executing instruction. To do so, you can use blocking instruction (0xDD – 0xDF) shown in Table 9.

6. Configuration

Parameters that can be configured with this firmware:

- Module identifier (module number and group number);
- Module description (16 chars);
- Minimum and maximum values;
- Power up dimming speed;
- Power up states;
- Last state memory;
- Text notes;
- Linking device with other modules (indirect control of module).

Configuration process can be done using HAPCAN Programmer.

6.1. Module identifier

Every module on the network must have unique identifier. The identifier is made of two bytes, module number (1 byte) and group number (1 byte). Identifier of the Ethernet Interface can be changed in HAPCAN Programmer in software settings.

6.2. Module description

Every module can have 16 char description, which makes easier for user (programmer) to distinguish nodes.

6.3. Minimum and maximum values

It is possible to set minimum and maximum values for each channel.

6.4. Power up dimming speed

This parameter defines how fast channel goes from value 0 to value 255. The dimming speed can be chosen between 1s and 256s with 1s step.

6.5. Power up states
It is possible to configure channel states at startup after power loss. At startup values can be chosen between 0 and 255 or the or the last state saved in non-volatile memory can be taken. The last state value must be unchanged for at least 6s before power failure.

6.6. Last state memory
The last state of any channel can be remembered. In this mode, when channel is being switched on, it sets to the value that was before switching off. In no state memory mode it sets to the maximum value.

6.7. Text notes.
Up to 1024 characters can be written into processor's memory.

6.8. Linking devices
The module has 128 memory cells (boxes). Each box can contain information about message sent by other node, and instruction which will be executed when that message is received.

7. License

8. Document version

File	Note	Date
univ_3-8-0-0a.pdf	Original version	January 2014