1. Features

- Firmware for light dimmer
- Voltage control in 256 steps (0-255)
- 4 defined dimmer characteristics
- Possibility to program user defined characteristic
- Adjustable minimum and maximum values
- Last state memory
- Adjustable dimming time for each channel
- 10 control instructions
- 3 blocking instructions
- 1 timer 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

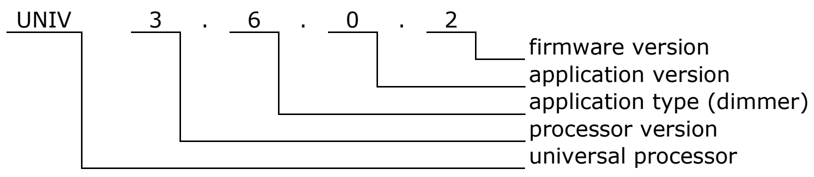
New:

- Operators <-small or equal and> greater than or equal were added to support boxes
- The flash of the light source that occurred when the set dimmer value was saved to the eeprom memory has been eliminated

2. Compatibility

- Firmware is compatible to previous UNIV 3.6.0.1
- Firmware for **UNIV 3.6.0.x** module
- Firmware can be uploaded into processor with bootloader version 3.1 or compatible.

3. Firmware version



4. Communication Frames (messages)

4.1. Dimmer message

The module sends message to the bus, when the dimmer state changes.

Table 1. Dimmer frame (0x306)

Frame type	Flags	Module	Group	D0	D1	D2	D3	D4	D5	D6	D7
0x306	3 2 1 0	Node Nr	Group Nr	0xFF	0xFF	CHANNEL	STATUS	0xFF	INSTR1	INSTR2	TIMER

0x306 – dimmer frame

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 (always 0x01 in this device)

STATUS - current status of output 0x00 – 0xFF

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

Table 2. Dimmer error frame
The module sends message to the bus, when the dimmer error changes.

Frame type	Flags	Module	Group	D0	D1	D2	D3	D4	D5	D6	D7
0x306	0x0	Node Nr	Group Nr	0xFF	0xFF	**0xF0**	ERROR	0xFF	0xFF	0xFF	0xFF

0xF0	- error frame
ERROR	0x00 – <00000000> - no error 0x01 – <00000001> (bit 0) – 230V mains problem 0x02 – <00000010> (bit 1) - overheating

Table 3. Dimmer transistor conduction time
The frame is sent only in response to STATUS REQUEST

Frame type	Flags	Module	Group	D0	D1	D2	D3	D4	D5	D6	D7
0x306	0x1	Node Nr	Group Nr	0xFF	0xFF	**0xFE**	COND1	COND0	0xFF	0xFF	0xFF

0xFE	MOSFET - transistor conduction time frame
COND	- COND1*256 + COND0 – transistor conduction time frame in mains half sine wave (in microseconds)

Table 4. Mains frequency frame
The frame is sent only in response to STATUS REQUEST

Frame type	Flags	Module	Group	D0	D1	D2	D3	D4	D5	D6	D7
0x306	0x1	Node Nr	Group Nr	0xFF	0xFF	**0xFF**	FREQ1	FREQ0	0xFF	0xFF	0xFF

0xFF	- mains frequency frame
FREQ	- FREQ1*256 + FREQ0 – mains sine wave half period time (value in microseconds)

4.2. Status request

Status of module can be checked by sending from computer STATUS REQUEST frame (0x109) (Table 5).

Table 5. 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 dimmer frames. The meaning of bytes is the same as in Table 1,2,3,4.

Table 6. Response to STATUS REQUEST

Frame type	Flags	Module	Group	D0	D1	D2	D3	D4	D5	D6	D7
0x306	0x1	Node Nr	Group Nr	0xFF	0xFF	CHANNEL	STATUS	0xFF	INSTR1	INSTR2	TIMER
Frame type	**Flags**	**Module**	**Group**	**D0**	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**
0x306	0x1	Node Nr	Group Nr	0xFF	0xFF	**0xF0**	ERROR	0xFF	0xFF	0xFF	0xFF
Frame type	**Flags**	**Module**	**Group**	**D0**	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**
0x306	0x1	Node Nr	Group Nr	0xFF	0xFF	**0xFE**	COND1	COND0	0xFF	0xFF	0xFF
Frame type	**Flags**	**Module**	**Group**	**D0**	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**
0x306	0x1	Node Nr	Group Nr	0xFF	0xFF	**0xFF**	FREQ1	FREQ0	0xFF	0xFF	0xFF

4.3. Uptime request

Table 7. 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 8. 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 9. 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 10).

Table 10. 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 11 must be sent. There is no response to this message.

Table 11. 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 12. Module control instructions

Instruction	Instruction Coding								Note	Control	
	INSTR1	INSTR2	INSTR3	INSTR4	INSTR5	INSTR6	INSTR7	INSTR8		Direct	Indirect
SET TO…	**0x00**	**STATE**	**TIMER**	0xXX	0xXX	0xXX	0xXX	0xXX	Sets the state of the dimmer immediately at the level specified by the byte **STATE** (0-255). The SOFT START feature takes 1s to turn device on and off. Instructions may be delayed if the **TIMER** is not zero.	√	√
TOGGLE	**0x01**	0xXX	**TIMER**	0xXX	0xXX	0xXX	0xXX	0xXX	If dimmer 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.	√	√
STEP DOWN BY …	**0x02**	**VALUE**	**TIMER**	0xXX	0xXX	0xXX	0xXX	0xXX	Dimmer state will be reduced by the value indicated in the **VALUE** byte.	√	√
STEP UP BY …	**0x03**	**VALUE**	**TIMER**	0xXX	0xXX	0xXX	0xXX	0xXX	Dimmer state will be increased by the value indicated in the **VALUE** byte.	√	√
STOP	**0x04**	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	It stops instruction which is being executed eg. START or SET SOFTLY TO …	√	√
START	**0x05**	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 dimmer 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 dimmer will turn on or off. If button is pressed for longer than 400ms, then dimmer will dim or brighten.	√	√
SET SOFTLY TO…	**0x06**	**STATE**	**TIMER**	0xXX	0xXX	0xXX	0xXX	0xXX	Sets the dimmer statel slowly to the level specified by the byte **STATE** (0-255). Instructions may be delayed if the **TIMER** is not zero.	√	√
SET MINIMUM TO…	**0x07**	**VALUE**	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	Sets dimmer minimum value. Byte **VALUE** (0-255).	√	√
SET MAXIMUM TO …	**0x08**	**VALUE**	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	Sets dimmer maximum value. Byte **VALUE** (0-255).	√	√
SET DIMMING SPEED TO …	**0x09**	**VALUE**	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	It sets dimming time. **VALUE** byte (0-255) defines time when dimmer changes its state from 0 to 255. This byte can hold value from 0 – 255 which corresponds to 1s – 256s.	√	√
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.

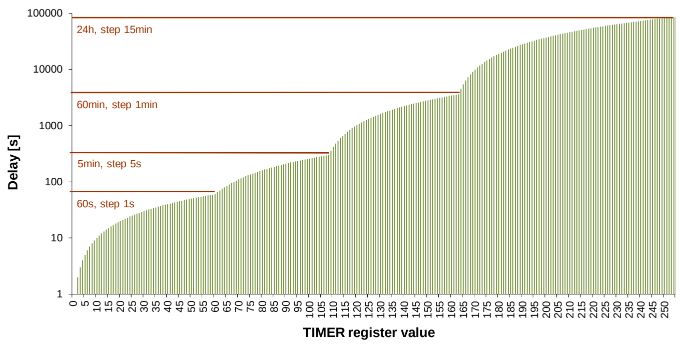


Figure 1. Delay/timer register relationship

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 13. 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 12.

6. Configuration

Parameters that can be configured with this firmware. Configuration process can be made 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).

6.2. Module description

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

6.3. Dimmer channel name

The only one channel of this device can be named with 32 chars.

6.4. Power up minimum and maximum values

It is possible to set minimum and maximum values.

6.5. Power up dimming speed

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

6.6. Power up state

It is possible to configure dimmer state 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.

6.7. Last state memory

The dimmer last state 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.

6.8. Dimmer characteristics

One of five dimmer characteristics can be chosen. For the light control is it advised to choose square or incandescent curve. Only few types of LED bulbs might require "LED" curve.

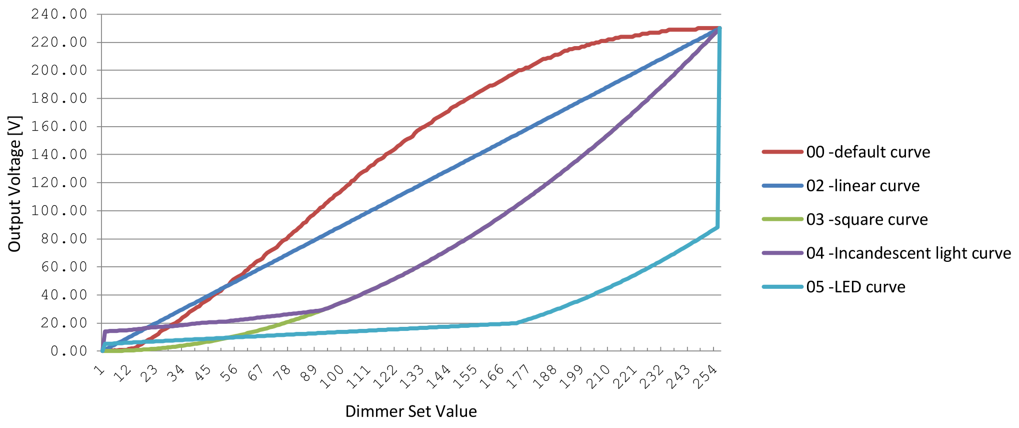


Figure 2. Defined dimmer characteristics.

6.9. User defined characteristic

When defining user characteristic it is required to set, for each dimmer value, the transistor conduction time in the half period of mains sine wave. This time must be contained within the range 0 - 9215 μs. Value equal or greater than 9215 μs means a continuous conduction of the transistor.

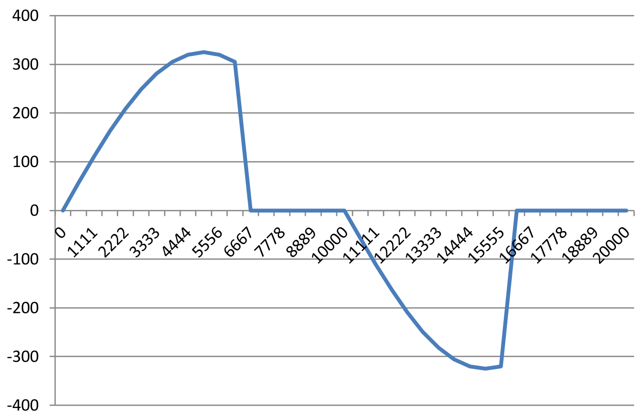


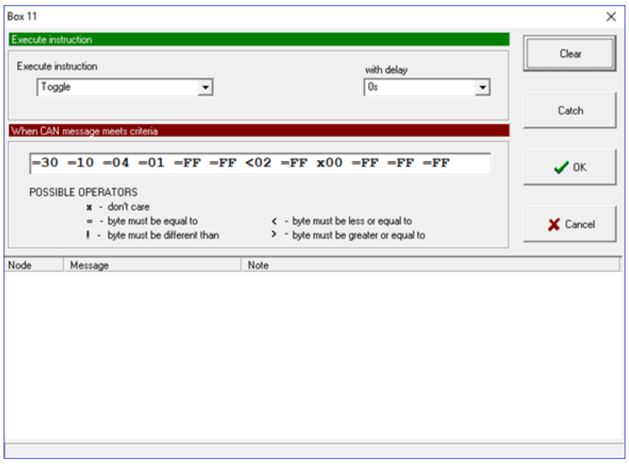
Figure 3. An example of the transistor conduction time equal 6667μs

6.10. Text notes.

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

6.11. Linking devices

The indirect control is used to program the dependencies between modules. It is a controlling the module by messages sent from other modules. Each module when changing its status (eg when button is pressed or released in the button module) sends information that can be used to control the same module, as well as to control other modules located on the bus. For this purpose, the module has 128 boxes (memory cells) into which you can enter messages to which the module is to react when it receives them from the bus. Each box contains information about what message should initiate the action and what instruction should be made when this message is received.



Execute instruction: Toggle
with delay: 0s

When CAN message meets criteria

```
=30 =10 =04 =01 =FF =FF <02 =FF x00 =FF =FF =FF
```

POSSIBLE OPERATORS
x - don't care
= - byte must be equal to
! - byte must be different than
< - byte must be less or equal to
> - byte must be greater or equal to

In the figure on a side, the toggle instruction has been programmed, which changes the current state of the dimmer at the moment of receiving a message from the module button o ID (4.1) - see the description of the BUTTON FRAME in the document "Firmware note" of the button module.

The message bytes are saved in hexadecimal format.

Operators at the front of bytes indicate that the module will react to a message received from the bus in which this byte:
x - can have any value
= - is identical to byte entered here
! - is different than byte entered here
< - is less or equal to byte entered here
> - is greater or equal to byte entered here

In this example:

=30 =10	– button message
=04 =01	– module ID is (4,1)
=FF =FF	– these bytes are not used and in button frame are always 0xFF
<02	– button number is less or equal 2, means button number 1 or 2
=FF	– button is pressed
x00	– it is a byte which gives LED status in the button module. Because we don't want to make instruction execution to be depended on this byte, so this byte can have any value.
=FF =FF =FF	– these bytes are not used and in button frame are always 0xFF

As a result, the module will execute instructions when the module receives a message from the button with ID (4.1), when button number 1 or 2 has been pressed. Instruction is executed regardless of the LED status in the button module.

Each own and received from the bus message informing about the change of module status is checked with each active (enabled) box in the order from box no. 1 to 128. Therefore, you can set the reaction to the same message in two or more boxes. For example, in the first box instruction can be executed without delay, and in the next box another instruction can be executed after the defined time. It is important to keep the logical order of instruction executing to get the desired operation of the module. Calling an instruction that is to be executed without delay will cancel the previously set instruction with a delay.
Boxes can be dynamically activated (enabled) and deactivated (disabled). Blocking instructions 0xDD-0XDF are used for this (Table 12). In this way, you can control which boxes are to be checked after receiving the message**.**

7. License

8. Document version

File	Note	Date
univ_3-6-0-2a.pdf	Original version	February 2019