

## 1. Features

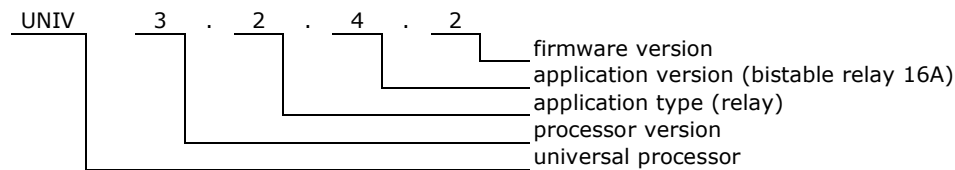
- Firmware for 6 bistable relay module
- 3 control instructions (on, off, toggle)
- 3 blocking instructions
- 6 timers (1 for each relay) for instruction execution delay 1s-24h
- Allows defining up to 128 CAN messages which can indirectly control the module
- Settable relay power up state
- Settable relay coil polarity
- Uptime counter
- Health check monitor
- Transmit (42 messages) and receive (42 messages) FIFO buffers



## 2. Compatibility

- Firmware compatible with previous one UNIV 3.2.4.1
- The operators < - less or equal and > - greater or equal were added to the boxes
- Firmware can be uploaded into processor with bootloader version 3.1 or compatible.

## 3. Firmware version



## 4. Communication Frames (messages)

### 4.1. Relay message

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

Table 1. RELAY frame (0x302)

Frame type	Flags	Module	Group	D0	D1	D2	D3	D4	D5	D6	D7
0x302	3 2 1 0	Node Nr	Group Nr	0xFF	0xFF	CHANNEL	STATUS	0xFF	INSTR1	INSTR2	TIMER

0x302	- universal module frame, relay
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

STATUS - actual status of outputs 0x00 - relay off, 0xFF - relay on

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 RELAY 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
0x302	0x1	Node Nr	Group Nr	0xFF	0xFF	0x01	STATUS	0xFF	INSTR1	INSTR2	TIMER1
0x302	0x1	Node Nr	Group Nr	0xFF	0xFF	0x02	STATUS	0xFF	INSTR1	INSTR2	TIMER2
0x302	0x1	Node Nr	Group Nr	0xFF	0xFF	0x03	STATUS	0xFF	INSTR1	INSTR2	TIMER3
0x302	0x1	Node Nr	Group Nr	0xFF	0xFF	0x04	STATUS	0xFF	INSTR1	INSTR2	TIMER4
0x302	0x1	Node Nr	Group Nr	0xFF	0xFF	0x05	STATUS	0xFF	INSTR1	INSTR2	TIMER5
0x302	0x1	Node Nr	Group Nr	0xFF	0xFF	0x06	STATUS	0xFF	INSTR1	INSTR2	TIMER6

## 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** -  $(UPTIME3*256^3+UPTIME2*256^2+UPTIME1*256^1+UPTIME0*256^0)$  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. In both situation all instructions in the table below can be used. Blocking instructions can't be used in direct control.

**5.1. Control instruction**

The Table 9 shows instructions, which can be executed by the module.

Table 9. Module control instructions

Instruction	Instruction Coddng								Note	Control	
	INSTR1	INSTR2	INSTR3	INSTR4	INSTR5	INSTR6	INSTR7	INSTR8		Direct	Indirect
TURN OFF RELAY	0x00	CHANNEL	TIMER	0xXX	0xXX	0xXX	0xXX	0xXX	It will turn off chosen relays, and the rest will stay unchanged.		
TURN ON RELAY	0x01	CHANNEL	TIMER	0xXX	0xXX	0xXX	0xXX	0xXX	It will turn on chosen relays, and the rest will stay unchanged.	√	√
TOGGLE RELAY	0x02	CHANNEL	TIMER	0xXX	0xXX	0xXX	0xXX	0xXX	It will toggle chosen relays, and the rest will stay unchanged.		

0xXX – inessential data

CHANNEL	Note
0x01	- <00000001> - only relay K1
0x02	- <00000010> - only relay K2
0x03	- <00000011> - relay K1 & K2
0x04	- <00000100> - only relay K3
...	
0x3F	- <00111111> - relays K1,K2,K3,K4,K5,K6

bit <0> - relay K1, bit <1> - relay K2, bit <2> - relay K3  
bit <3> - relay K4, bit <4> - relay K5, bit <5> - relay K6

TIMER	Note
0x00	- instruction will be executed immediately
0x01	- instruction will be executed with 1s delay
...	
0xFF	- instruction will be executed with 24h delay

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 relay 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.

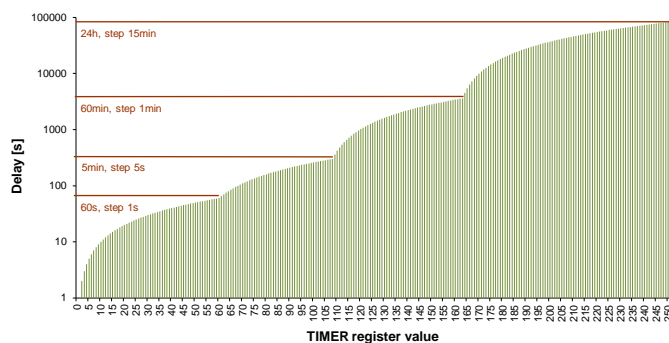


Figure 1. Delay/timer register relationship

### 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. The module can be also controlled from HAPCAN Programmer.

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

Below are parameters that can be configured with this firmware. 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. Relay Names

Each relay can be named with 32 chars.

### 6.4. Power up values

It is possible to configure relay states at startup after power loss. At startup relays can be set to ON, OFF or to the last set value. The last set value must be unchanged for at least 6s before power failure.

### 6.5. Relay coil polarity

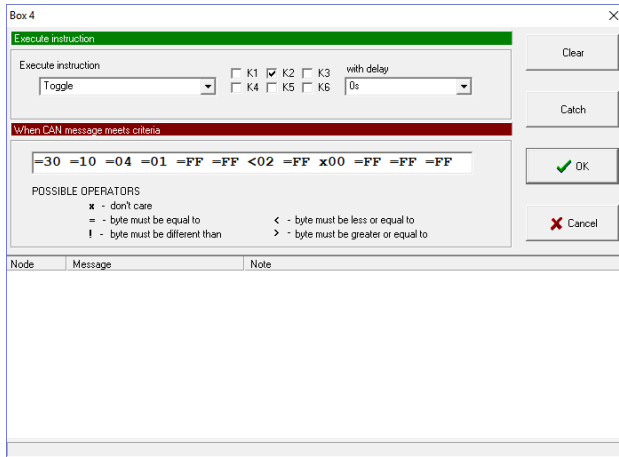
For each relay individually, you can determine the polarity of the coil: normal (if NO contact is closed when positive potential is connected to A1) or reversed (if NO contact is closed when positive potential is connected to A2).

### 6.6. Text notes.

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

### 6.7. 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.



In the figure on a side, the toggle instruction has been programmed, which changes the current state of the K2 relay 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 are used for this (Table 9). In this way, you can control which boxes are to be checked after receiving the message.

## 7. License



HAPCAN Home Automation Project firmware, Copyright (C) 2018 [hapcan.com](http://hapcan.com)

This program is free firmware: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.



This firmware is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this firmware. If not, see <http://www.gnu.org/licenses/gpl-3.0.html>.

## 8. Document version

File	Note	Date
univ_3-2-4-2a.pdf	Original version	March 2018