CANx / LoRa 433 MHz 8 x Analog inputs / Digital ouputs

ENG - Data sheet
Issue date 26.11.2021

Application

Universal 8 channel IO device is designed to be used in building automation applications as an extension module to LogicMachine series devices based on CAN FT bus. The configuration and monitoring of the device is done through separate LogicMachine CANx application. The device is designed for DIN-rail mounting and requires 4 DIN-units.

Types of product

CAN-UIO16	Universal canX bus IO module 16 AI/DO

Standards and norms compliance

CE conformity:	EMBS-CE-190223/03 Electromagnetic compatibility
EMC:	EN61000-6-1 EN61000-6-3
PCT	Certificate

Technical data:

Power supply:	12-32 VDC
Power consumption (at 24 V)	15 mA (LoRa not active)

	30 mA (peak LoRa activity)	
DC overvoltage protection:	±50 V	
Interface:	Universal Inputs/Outputs	8
	Analog input resolution	12bits
	Digital output current	350 mA (max 2 A per whole device)
	CAN FT	1
LoRa specification	Power on transmitter	1.6-50 mW (software adjustable)
	Frequency range	433-434,750 MHz
	Channel bandwidth	125 / 250 / 500 kHz
	Carrier frequency step	125 kHz
	Spreading factor	7-12
Clamps:	CAN FT	CAN FT Connection Terminal 0.8mm2
	Inputs/Outputs	3.5mm2
	Power supply	5 mm2
Enclosure:	Material:	Polyamide
	Color:	Gray
	Dimensions:	54(W)x100(H)x68(L) mm
Protection:	IP20 according to EN 60529	
Usage temperature:	-5C ... +55C	
Storage temperature:	-20C ... +70C	
Net weight:	86g	
Gross weight:	97g	

⚠ Caution

Security advice

The installation and assembly of electrical equipment may only be performed by skilled electrician. The devices must not be used in any relation with equipment that supports, directly or indirectly, human health or life or with application that can result danger of people, animals or real value

Mounting advice

The devices are supplied in operational status. The cables connections included can be clamped to the housing if required.

Electrical connection

The devices are constructed for the operation of protective low voltage (SELV). Grounding of device not needed. When switching the power supply on or off, power surges must be avoided.

Default settings

Line ID: 0

Node ID: 1

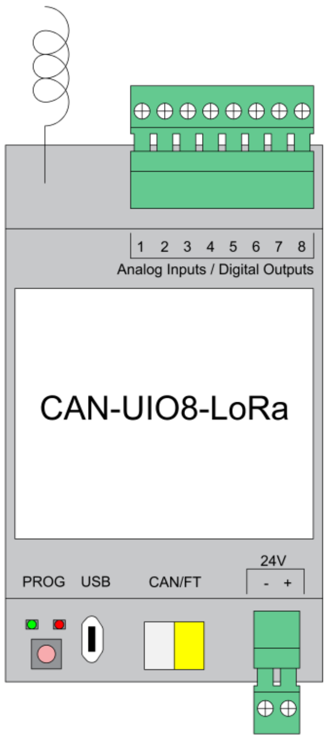
Max. number of group addresses per object : 16

Reset to defaults

Press programming button for 5 seconds, the RED LED blinks 2 times, then release button - GREEN lights up shortly.

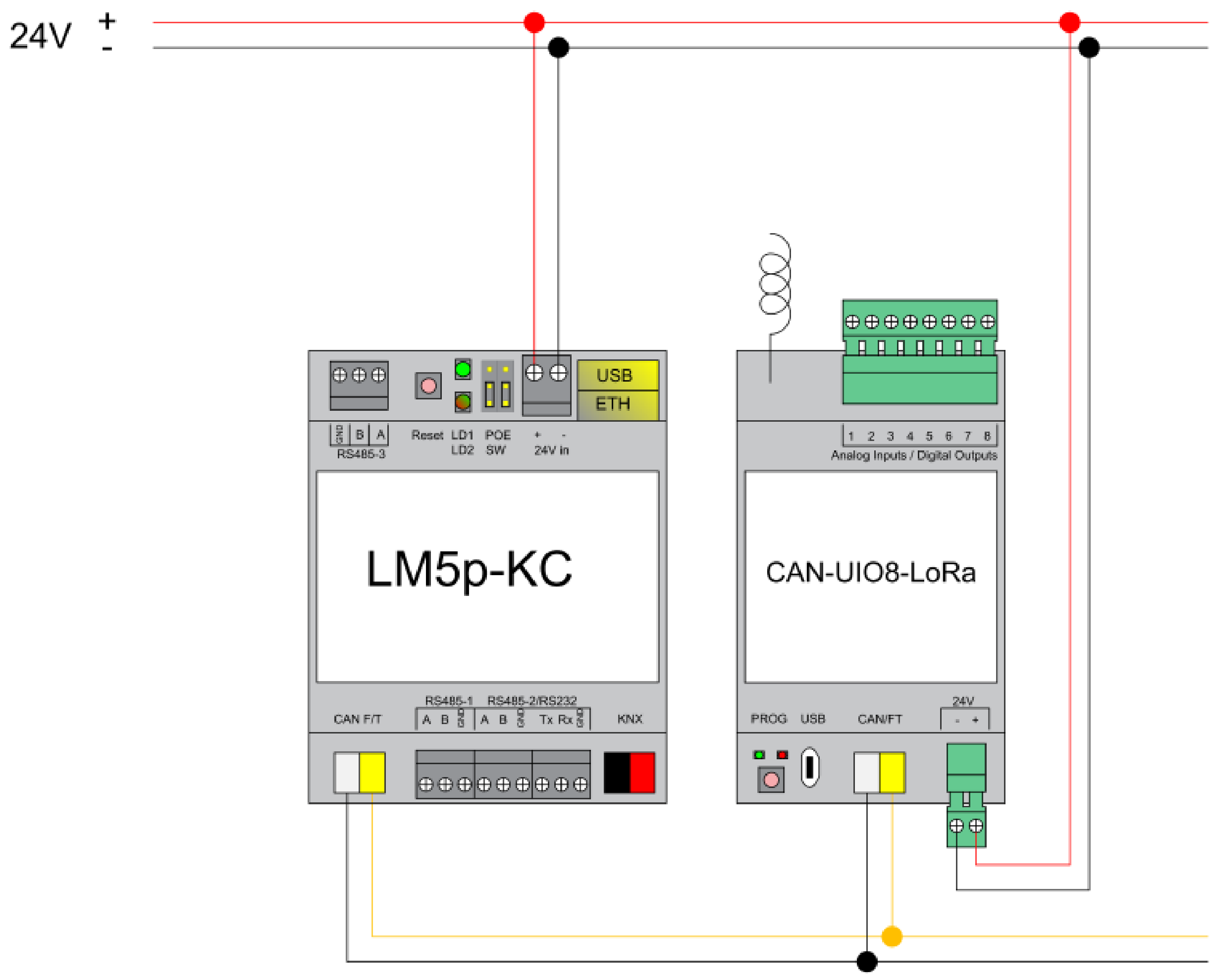
Programming physical address

Press *Tools* → *Write device address* from CANx app. Choose address and press *Write*. Then press programming button shortly on the device, GREEN LED lights up shortly. The LED is switched off automatically in 1 second which means address is written.

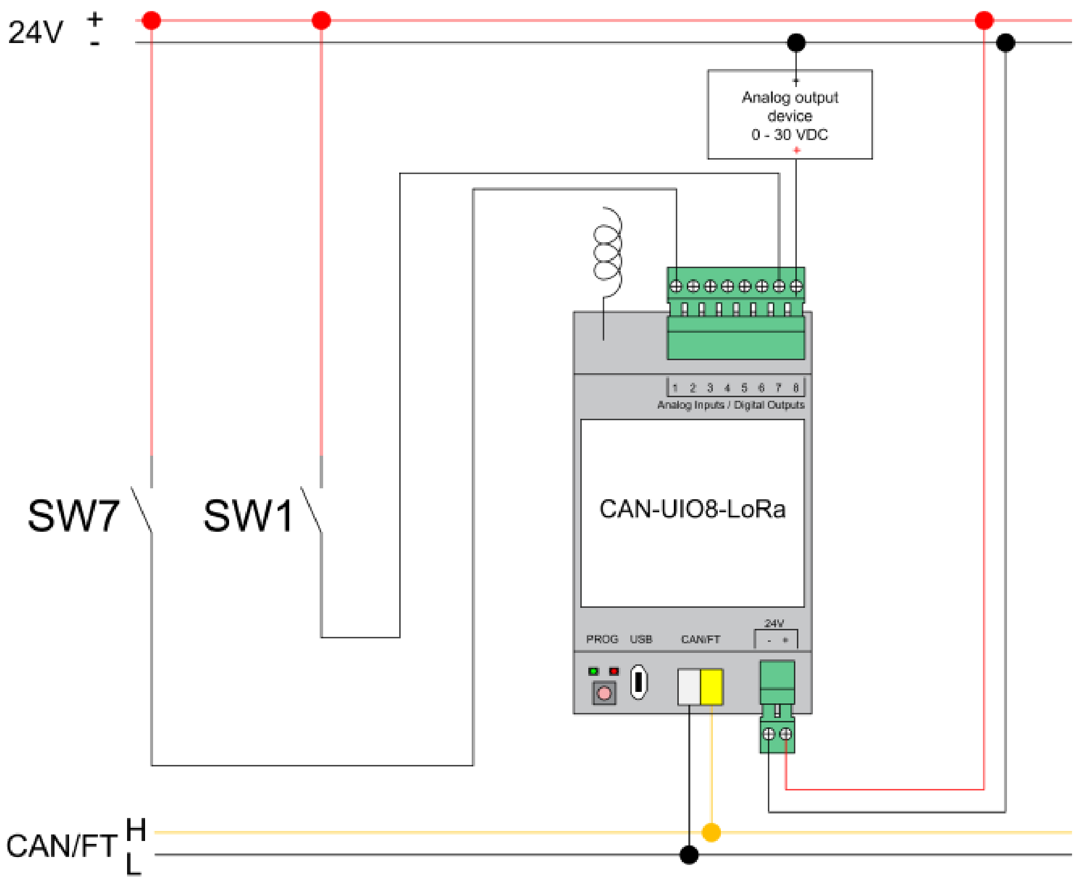


1. Connection diagrams

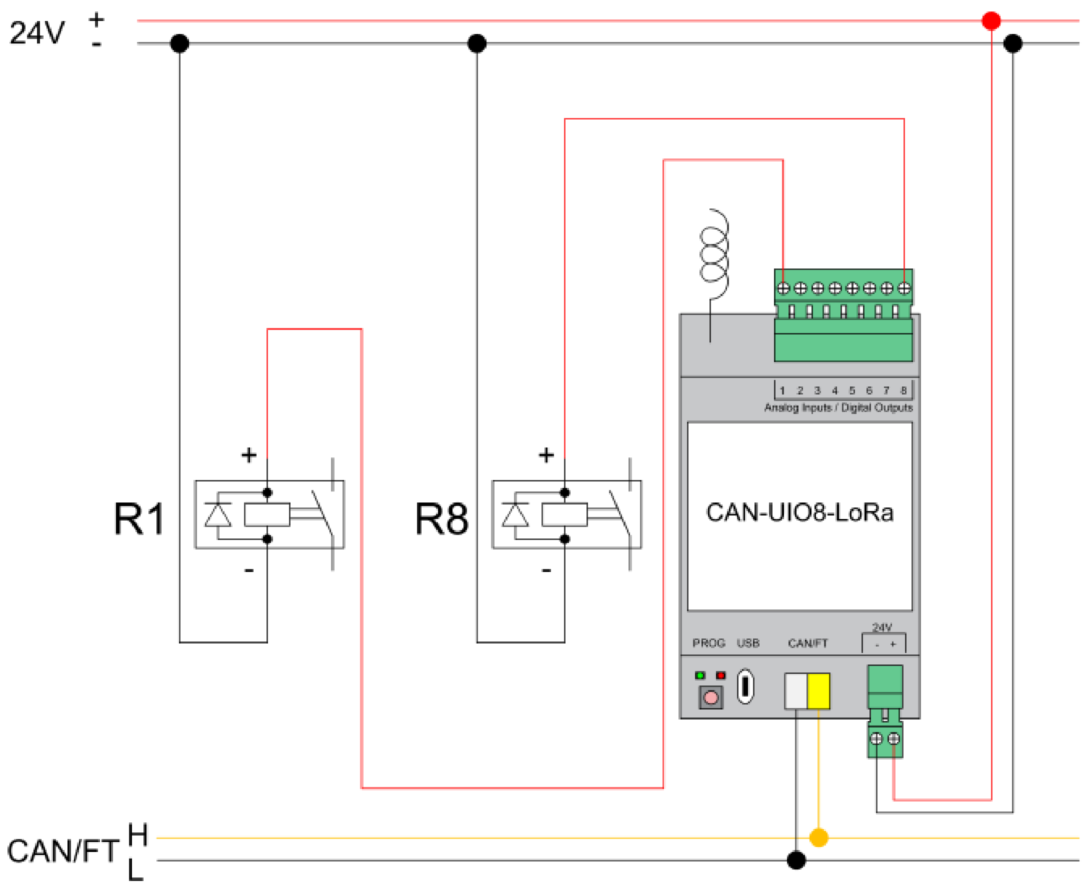
CAN FT connection



Digital / Analog input

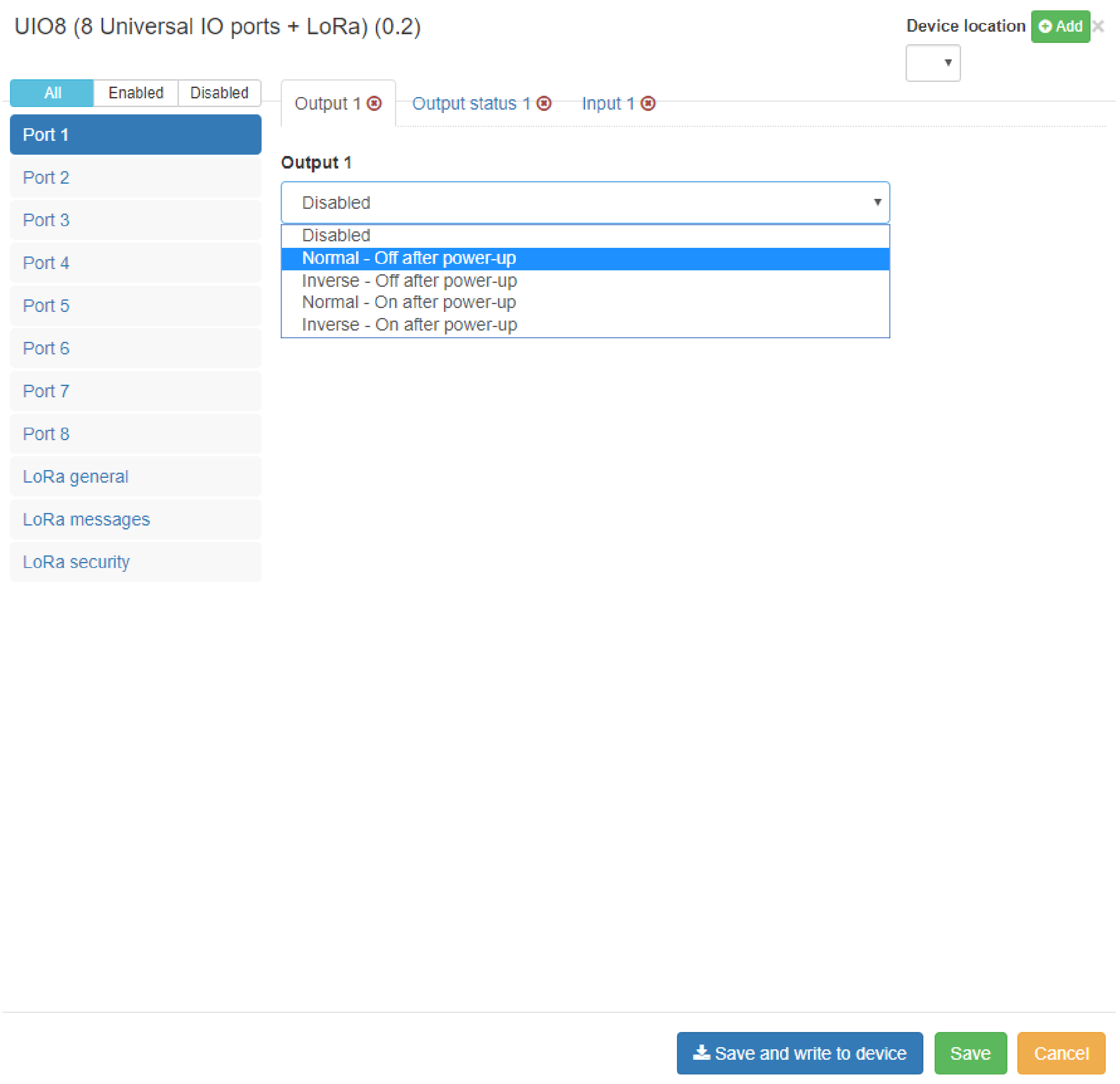


Digital output (e.g. external contactor control)



2. canX software settings

Digital output


UIO8 (8 Universal IO ports + LoRa) (0.2)

Device location

Output 1

Disabled

- Disabled
- Normal - Off after power-up
- Inverse - Off after power-up
- Normal - On after power-up
- Inverse - On after power-up

Default flags: read (R), write (W), transmit (T)

Output mode:

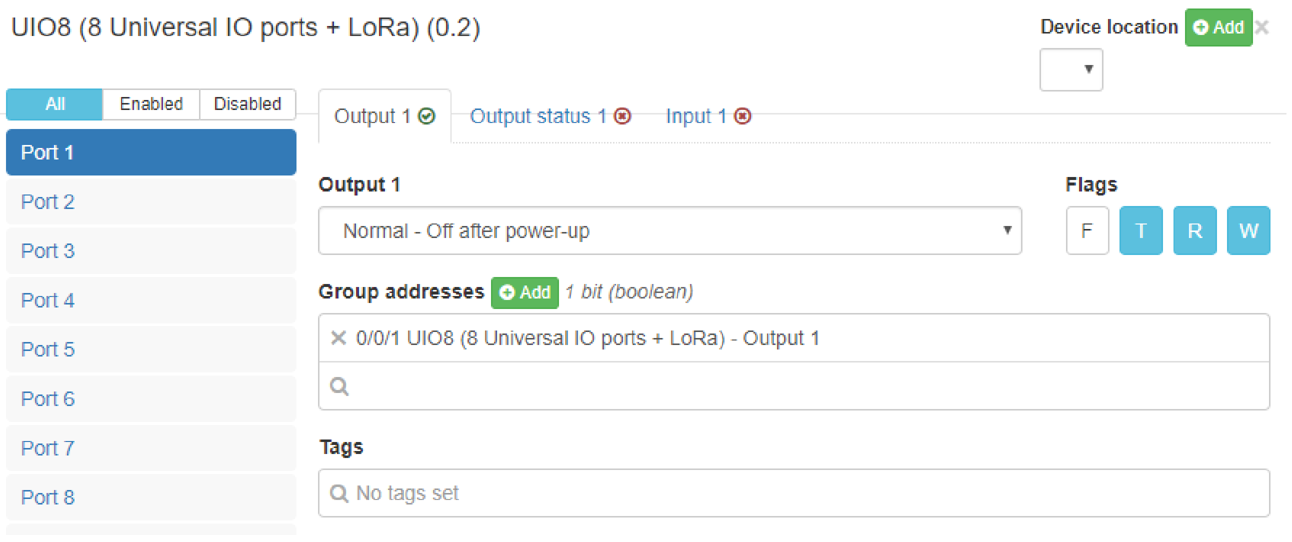
Normal – Off after power-up

Inverse – Off after power-up

Normal – On after power-up

Inverse – On after power-up

Group addresses – you can assign group addresses from the predefined list or add manually by clicking on ADD button. You can assign max 16 group addresses to one object / output.



UIO8 (8 Universal IO ports + LoRa) (0.2)

Device location

Output 1

Normal - Off after power-up

Flags

Group addresses 1 bit (boolean)

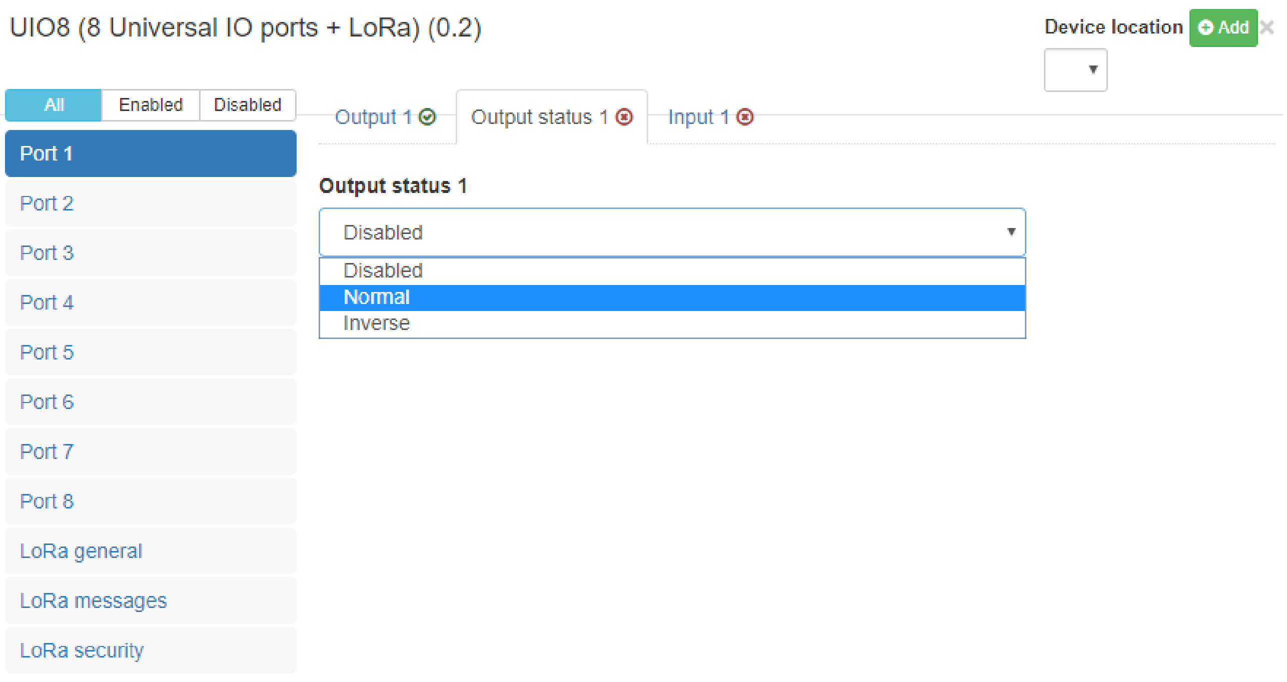
0/0/1 UIO8 (8 Universal IO ports + LoRa) - Output 1

Tags

No tags set

<u>Digital output status</u>

Status (response after read command) will return a real measurement value (1 – for high voltage, 0 – for no voltage)


UIO8 (8 Universal IO ports + LoRa) (0.2)

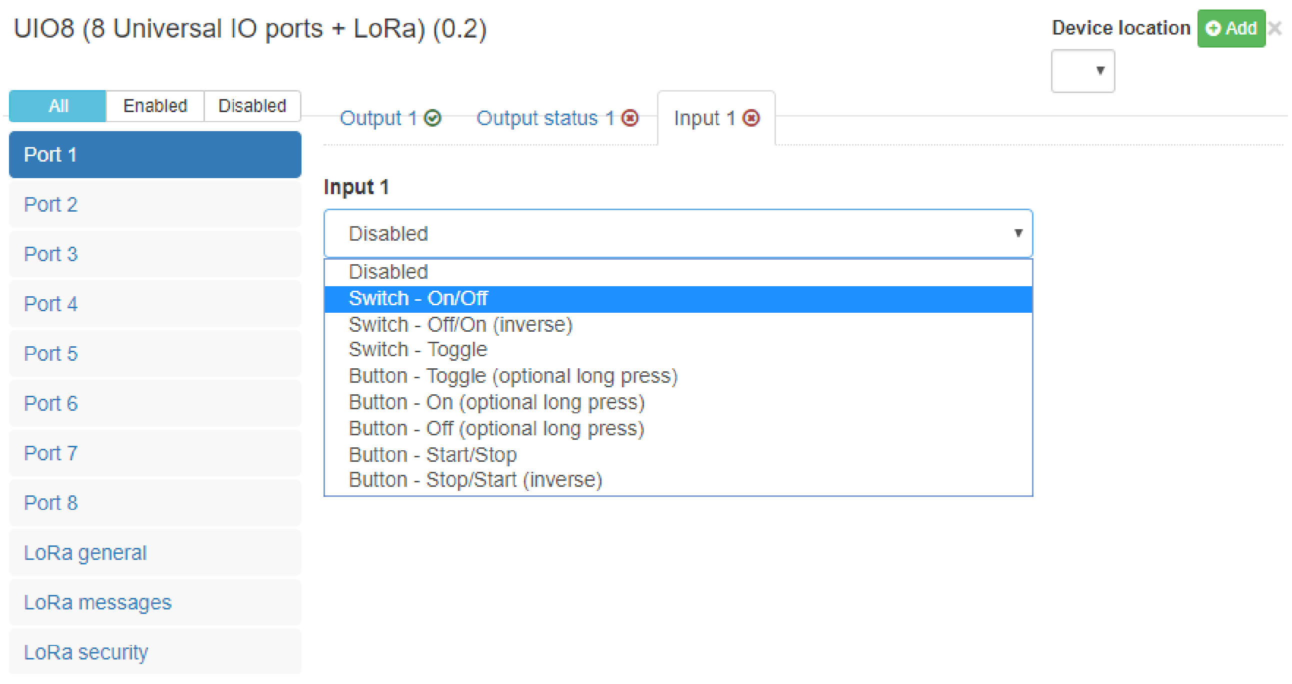
Output status 1: Disabled (options: Disabled, Normal, Inverse)

Default flags: read (R), transmit (T)

Output status: Disabled, Normal, Inverse

Group addresses – you can assign group addresses from the predefined list or add manually by clicking on ADD button. You can assign max 16 group addresses to one object / output status

<u>Input mode</u>


UIO8 (8 Universal IO ports + LoRa) (0.2)

Device location Add

All | Enabled | Disabled

- Port 1
- Port 2
- Port 3
- Port 4
- Port 5
- Port 6
- Port 7
- Port 8
- LoRa general
- LoRa messages
- LoRa security

Output 1 | Output status 1 | Input 1

Input 1

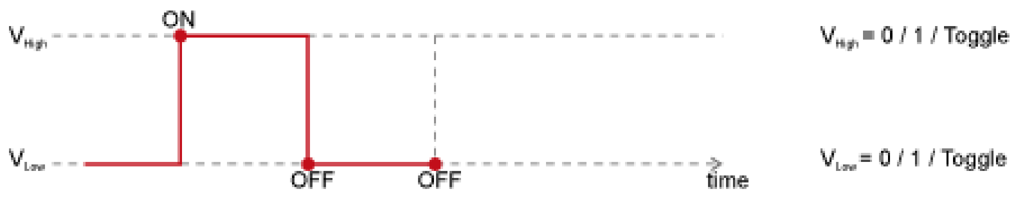
Disabled

- Disabled
- Switch - On/Off
- Switch - Off/On (inverse)
- Switch - Toggle
- Button - Toggle (optional long press)
- Button - On (optional long press)
- Button - Off (optional long press)
- Button - Start/Stop
- Button - Stop/Start (inverse)

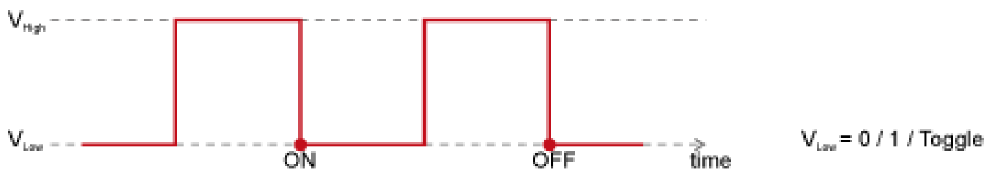
Default flags: read (R), write (W), transmit (T)

Input mode:

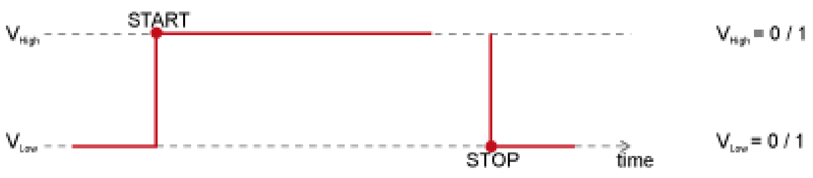
Switch on/off – send 1 to bus if switched ON or 0 if switched OFF
Switch off/on (inverse) – send 0 to bus if switched ON or 1 if switched OFF
Switch Toggle - change status to inverted with every push



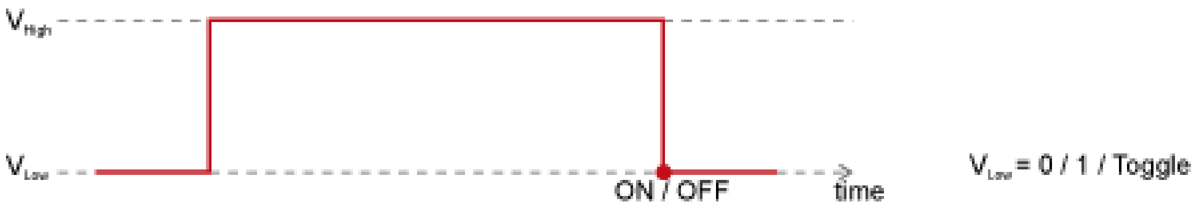
Button Toggle (optional long press) – change status to inverted with every push
Button On (optional long press) – push 1 to bus every pulse
Button Off (optional long press) – push 0 to bus every pulse

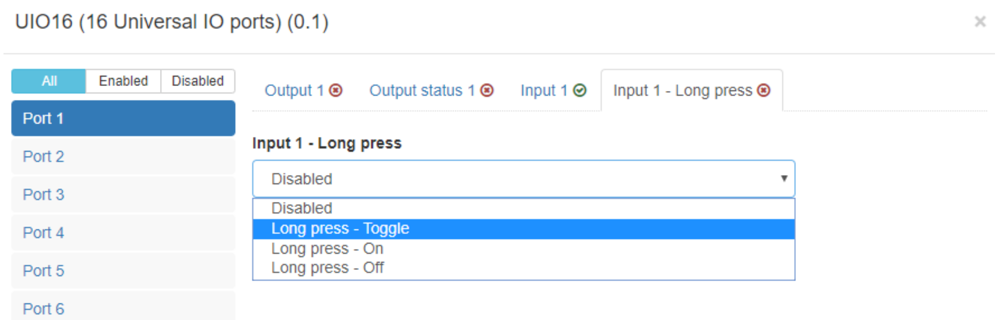


Button Start/Stop – send 1 when pushed and 0 when released
Button Stop/Start (inverse) – send 0 when pushed and 1 when released



Button long press toggle - Send 0 or 1 to bus with every long press
Button long press send 1 - Send 1 with every long press
Button long press send 0 - Send 0 with every long press



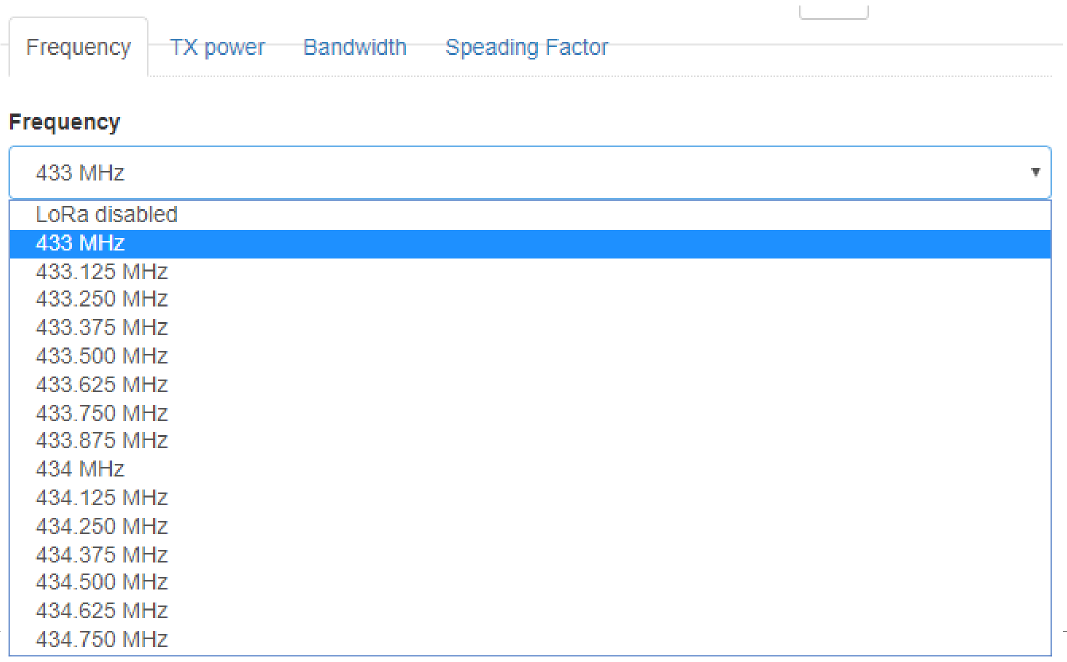


Input 1 - Long press: Disabled

Disabled
Long press - Toggle
Long press - On
Long press - Off

LoRa General settings

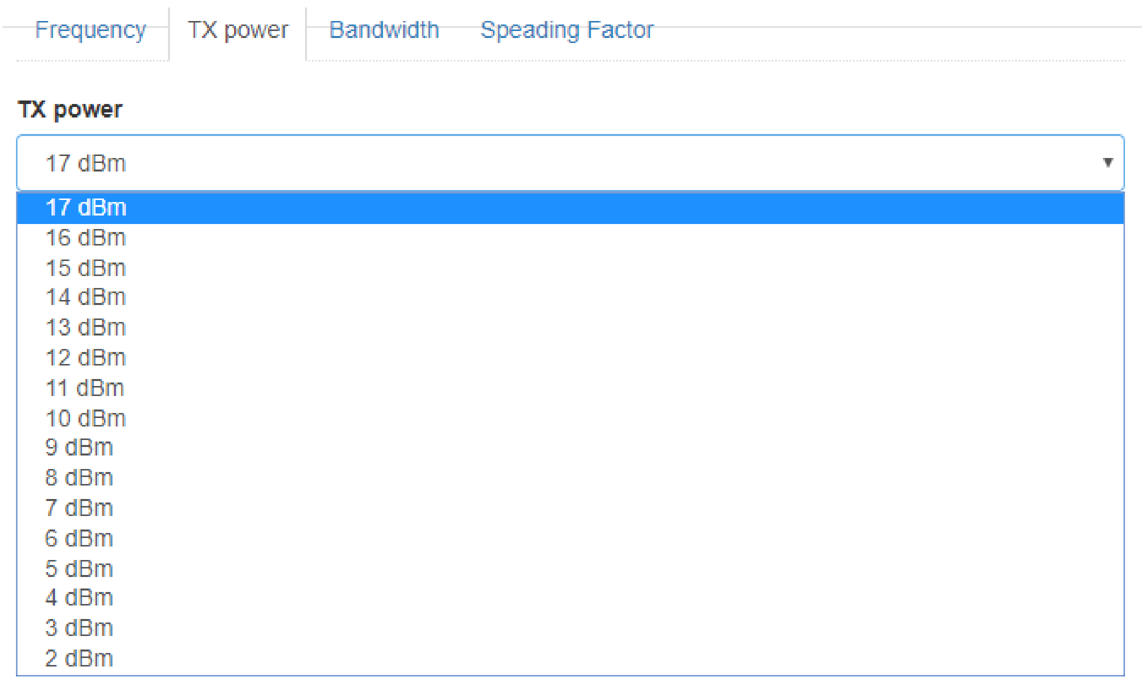
Frequency – define the frequency LoRa will operate in. Frequency should be equal on transmitter and receiver(-s).



Frequency: 433 MHz

LoRa disabled
433 MHz
433.125 MHz
433.250 MHz
433.375 MHz
433.500 MHz
433.625 MHz
433.750 MHz
433.875 MHz
434 MHz
434.125 MHz
434.250 MHz
434.375 MHz
434.500 MHz
434.625 MHz
434.750 MHz

TX power – output power of LoRa transceiver

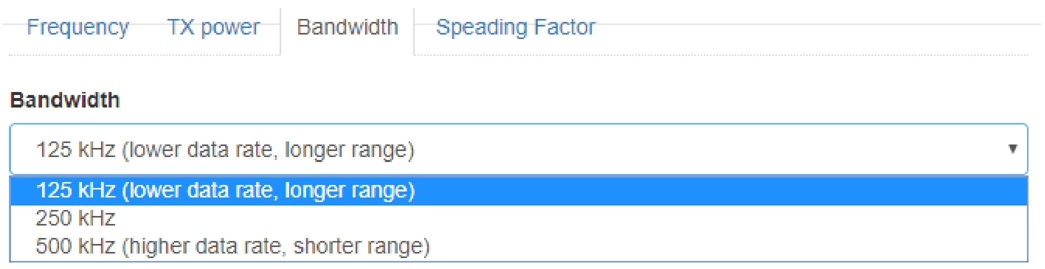


TX power

17 dBm

17 dBm
16 dBm
15 dBm
14 dBm
13 dBm
12 dBm
11 dBm
10 dBm
9 dBm
8 dBm
7 dBm
6 dBm
5 dBm
4 dBm
3 dBm
2 dBm

Bandwidth – define the bandwidth of the channel. The lower the bandwidth – the lower the data rate / longer the distance. Bandwith should be equal on transmitter and receiver(-s).

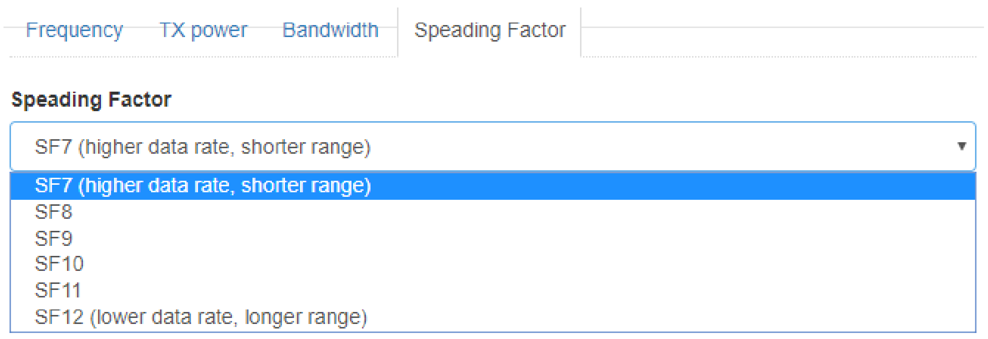


Bandwidth

125 kHz (lower data rate, longer range)

125 kHz (lower data rate, longer range)
250 kHz
500 kHz (higher data rate, shorter range)

Spreading factor - The basic principle of spread spectrum is that each bit of information is encoded as multiple chirps. Within the given bandwidth the relationship between the bit and chirp rate for LoRa modulation may differ between spreading factor (SF) 7 to 12. Spreading factor should be equal on transmitter and receiver(-s).



Speading Factor

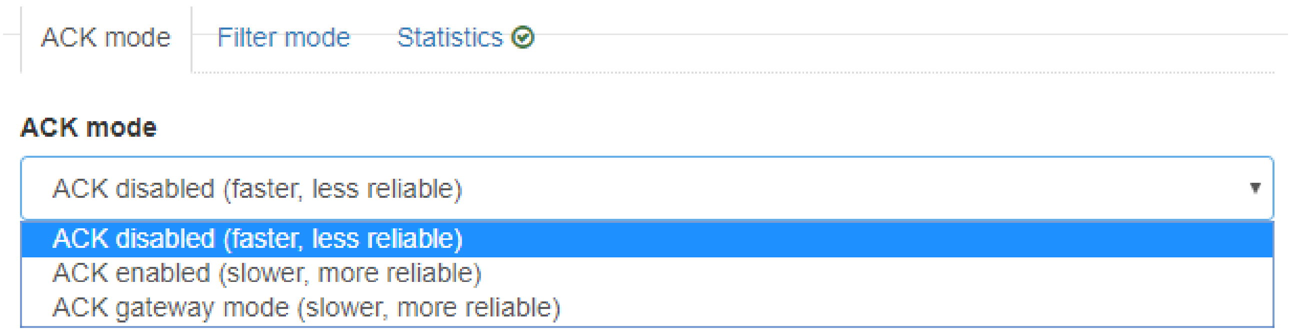
SF7 (higher data rate, shorter range)

SF7 (higher data rate, shorter range)
SF8
SF9
SF10
SF11
SF12 (lower data rate, longer range)

LoRa Messages

ACK mode – message acknowledgement mode

ACK disabled - no ACK will be done (faster and less reliable communication)
ACK enabled - each message will be acknowledged (slower, more reliable)
ACK gateway mode – the node will retransmit ACK to the next node

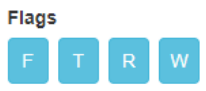


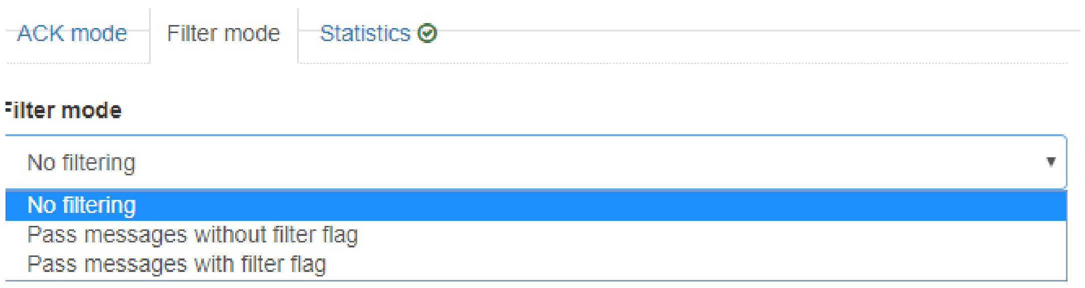
ACK mode

ACK disabled (faster, less reliable)

ACK disabled (faster, less reliable)
ACK enabled (slower, more reliable)
ACK gateway mode (slower, more reliable)

Filter mode – define either to pass messages with F (Filter) flag enabled in object settings



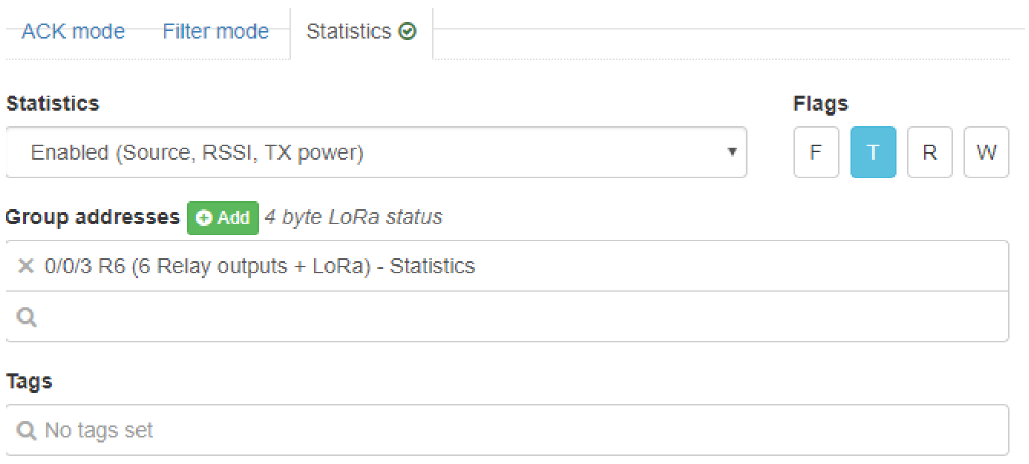


Filter mode

No filtering

- No filtering
- Pass messages without filter flag
- Pass messages with filter flag

Statistics – receive statistic information to group address – source address / RSSI signal level / TX power

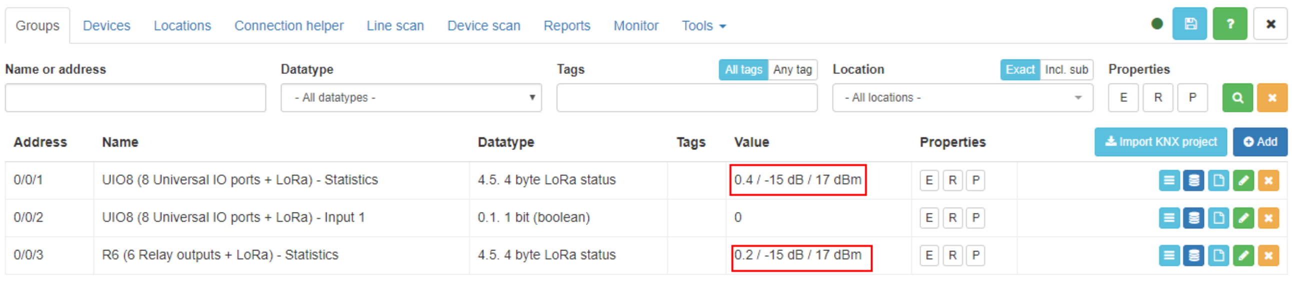


Statistics: Enabled (Source, RSSI, TX power)

Group addresses *4 byte LoRa status*

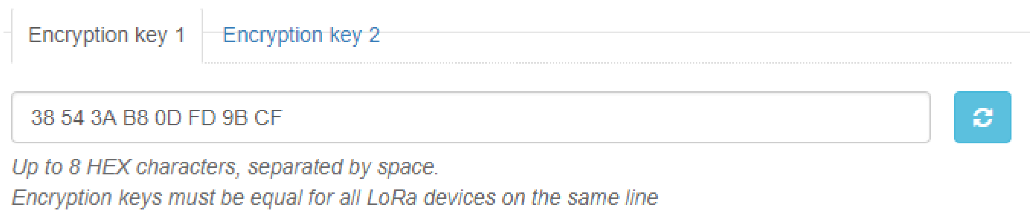
0/0/3 R6 (6 Relay outputs + LoRa) - Statistics

Tags: No tags set



Address	Name	Datatype	Tags	Value	Properties
0/0/1	UIO8 (8 Universal IO ports + LoRa) - Statistics	4.5. 4 byte LoRa status		0.4 / -15 dB / 17 dBm	E R P
0/0/2	UIO8 (8 Universal IO ports + LoRa) - Input 1	0.1. 1 bit (boolean)		0	E R P
0/0/3	R6 (6 Relay outputs + LoRa) - Statistics	4.5. 4 byte LoRa status		0.2 / -15 dB / 17 dBm	E R P

LoRa Security – define security key 1 or/and key 2 in HEX form. Up to 8 HEX characters are supported for each of the keys. Encryption keys must be equal for all LoRa devices on the same line


Encryption key 1 | Encryption key 2

38 54 3A B8 0D FD 9B CF

Up to 8 HEX characters, separated by space.
Encryption keys must be equal for all LoRa devices on the same line

Notification LEDs

- During transmission you can see two LEDs on LoRa device

🟨	Sending LoRa telegram
🟦	Receiving LoRa telegram

- In case statistics is enabled on receiver device and CAN FT line is disconnected from it, both LEDs will light up (receiving telegram from sender, sending telegram with statistics).
- In case ACK is enabled, both orange and blue LEDs will light up.

DALI control commands from scripts

canxdali = require('applibs.canxdali')

canxdali.sendcmds(req)
Sends single or multiple DALI commands to the given gateway.
Returns number of bytes sent or nil plus error message.
This is completely asynchronous function, it adds commands to gateway queue without waiting for returned results.

req table:
lineid - gateway line ID (number, required)
nodeid - gateway node ID (number, required)

command table:
cmd - command name (string, required)
value - command value (number, required for commands with a value)

address - DALI address (string or number, required)
addrtype - address type (string, required if address is a number)

address format:
address can be a string with following format:
s0..s63 - short address, from 0 to 63
g0..g15 - group, from 0 to 15
b - broadcast

if address is a number then *addrtype* is required, it can be either:
short
group
broadcast

Examples:

Send arc with value 0 to DALI short address 15 using gateway 0.1:

```
canxdali = require('applibs.canxdali')

  canxdali.sendcmds({
    lineid = 0,
    nodeid = 1,
    cmd = 'arc',
    address = 's15',
    value = 0,
  })

Send multiple arc commands using gateway 1.42:
  canxdali = require('applibs.canxdali')

  canxdali.sendcmds({
    lineid = 1,
    nodeid = 42,
    cmds = {
      { cmd = 'arc', address = 's0', value = 50 },
      { cmd = 'arc', address = 's4', value = 10 },
    }
  })
```

canxdali.syncsendcmds(req)
Similar to canxdali.sendcmds but waits for each command to complete.
On success returns Lua table with each command result, nil plus error message otherwise.

canxdali.sendqueries(req)
Similar to canxdali.syncsendcmds but checks for each command result, returns a table of values only for query type commands when all commands were successful. Useful for querying DALI device statuses.

canxdali.sethandler(type, fn)
Sets a callback to execute on a specific event.

Callback is executed for each command inside data frame separately.

type - event type (string, required):
 bus - all commands coming from bus side
 busdata - only "bus data" type commands (from other master devices)
 all - all commands coming to/from bus
fn - function to execute, or nil to remove callback (function or nil, required)

canxdali.step()
Waits for a frame or timeout, whichever happens first.
Returns frame or nil plus error message on timeout.
Frame can contain multiple commands when sent to bus.

Example (resident script):

```
if not canxdali then
  function callback(frame)
    log(frame)
  end

  canxdali = require('applibs.canxdali')
  canxdali.sethandler('bus', callback)
end

canxdali.step()
```