

Embedded Systems SIA, VAT No LV40003411103
47. Katolu str., Riga, LV 1003, LATVIA
Phone: +371 67648888, fax: +371 67205036, e-mail: sales@openrb.com

CANx / LoRa DALI gateway

ENG - Data sheet
Issue date 21.09.2018

Application

DALI (Digitally Addressable Lighting Interface) CANx gateway is a device specially designed for management and control of dimmable lights via CAN FT bus and wirelessly over LoRa 433. In a typical application, a DALI-bus consists of one gateway (master), and multiple slaves. In DALI-bus segment a master can control up to 64 individually addressable slaves who are also called (digital addressable) ballasts. The DALI standard enables compiling these slaves into: 16 light scenes (incl. dimming values and transitional periods) and 16 lighting groups (multiple assignments of the devices are possible).



Types of product	
CAN-DALI-LoRa	CANx / LoRa DALI gateway
Technical data	
Power supply for gateway	12-32V DC
Power consumption (at 24 V)	5.5 mA (stand-by), 20 mA (full load with LoRa)
Power supply for the DALI bus	16-18V DC
DC overvoltage protection:	50 V
Wrong wiring polarity protection	Yes
Interfaces and operating elements	
DALI output	1

Maximum count of ballasts per one CANx DALI gateway	64
USB	1 microUSB for upgrade firmware flashing
CAN FT	1
LED	1 – CPU load, 1 - Error
Programming/reset button	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 and enclosure	
CAN FT Terminal	0.8mm ²
DALI output	5 mm ²
Power supply	5 mm ²
Color	Gray
Dimensions	70(W)x100(H)x68(L) mm
Protection	IP20 according to EN 60529
Usage temperature	-5C ... +55C
Storage temperature	-20C ... +70C
Net weight:	86 g
Gross weight	97 g
Standards and norms compliance	
CE conformity	EMBS-CE-190223/10 Electromagnetic compatibility
EMC	EN61000-6-1, EN61000-6-3



Caution

Security advice

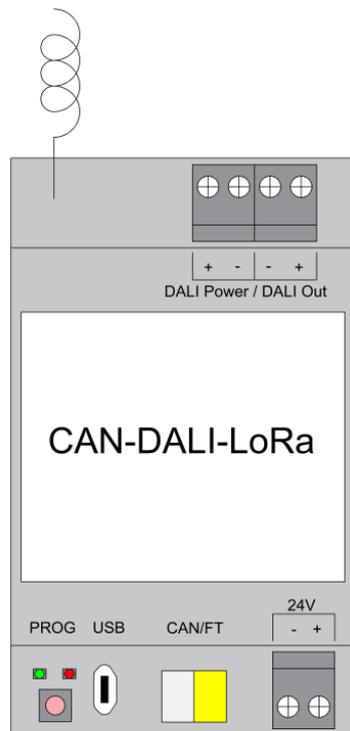
The installation and assembly of electrical equipment may only be performed by skilled electrician. The devices must not be used in any relation with equipment that supports, directly or indirectly, human health or life or with application that can result danger of people, animals or real value

Mounting advice

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

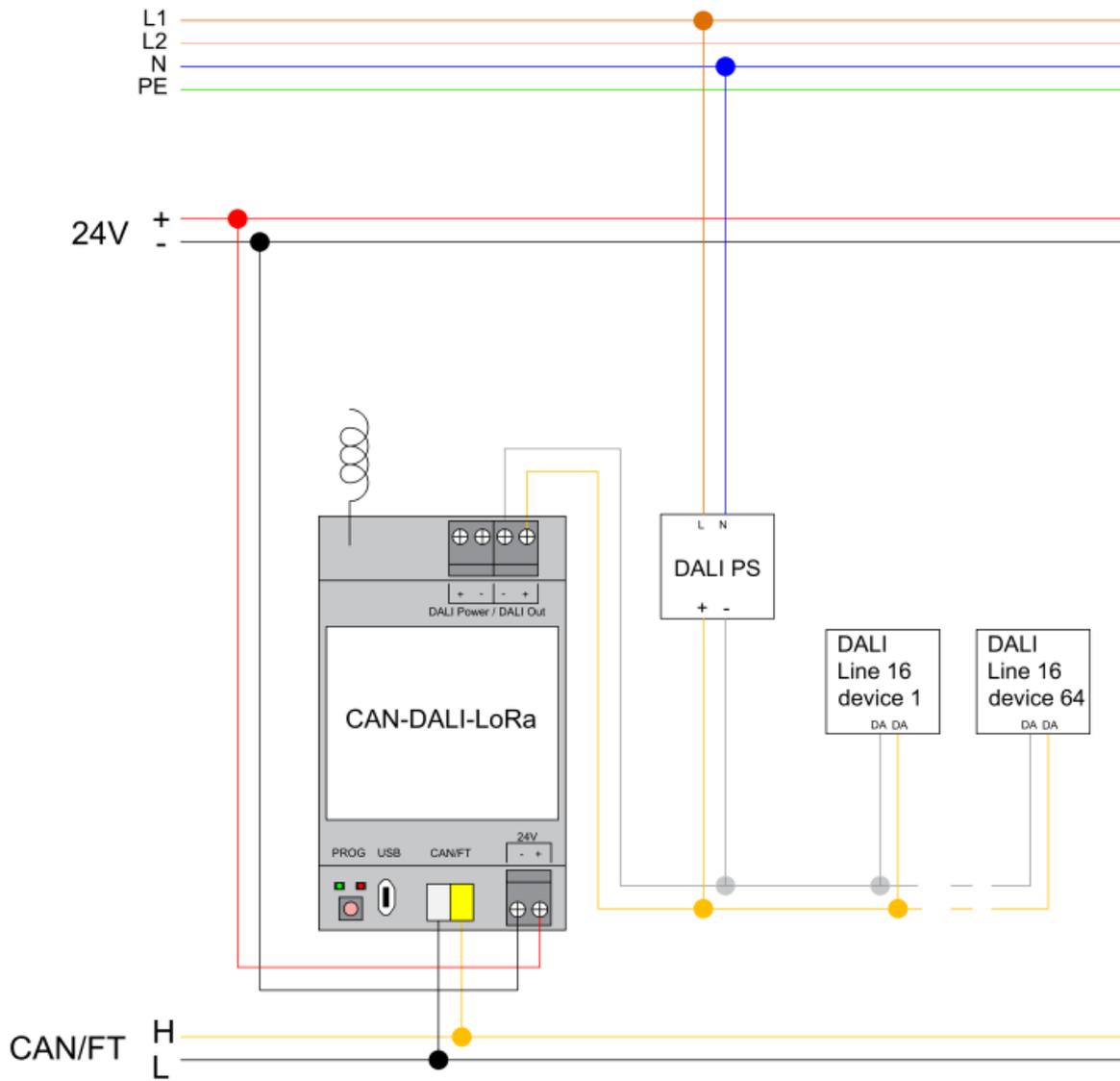
Electrical connection

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.

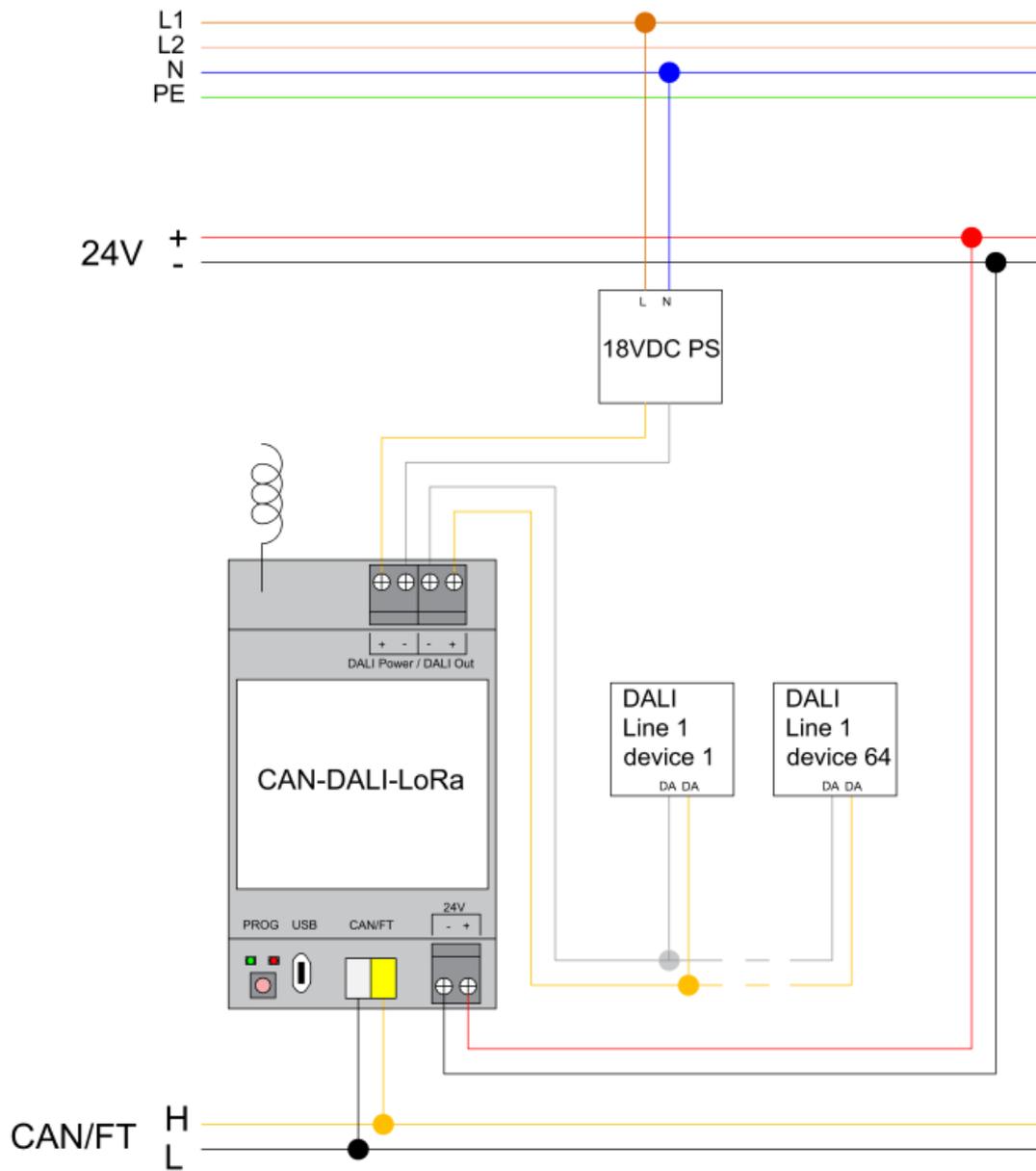


Connection diagram

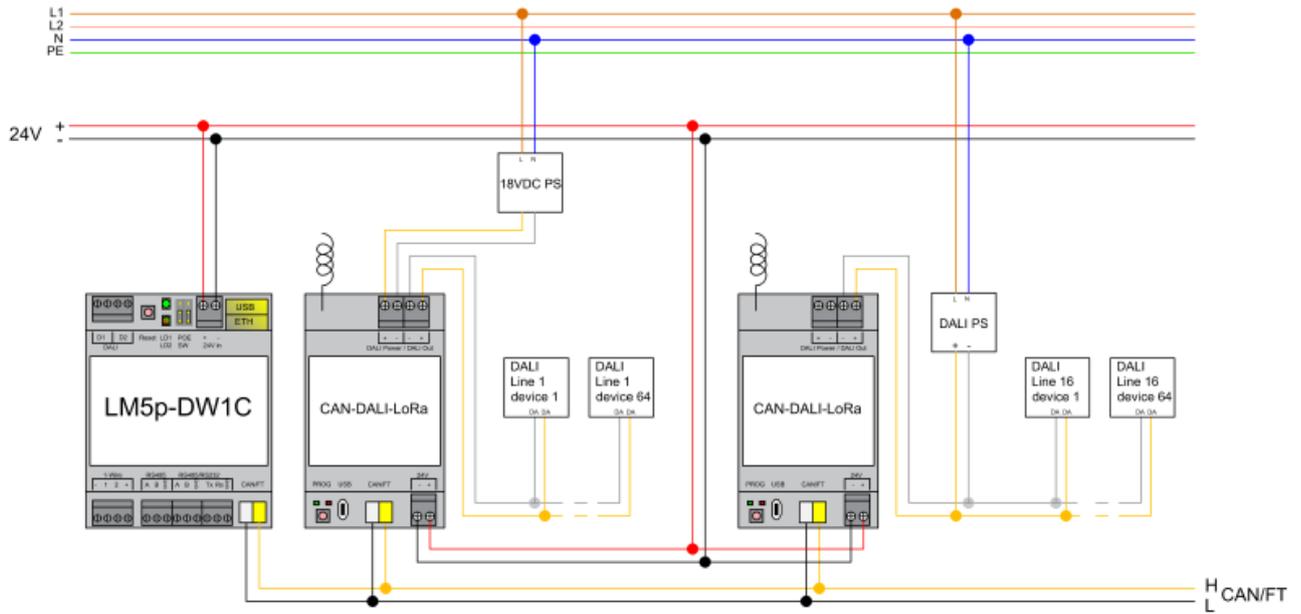
External DALI power supply



Internal DALI power supply



CAN FT connection



Software configuration

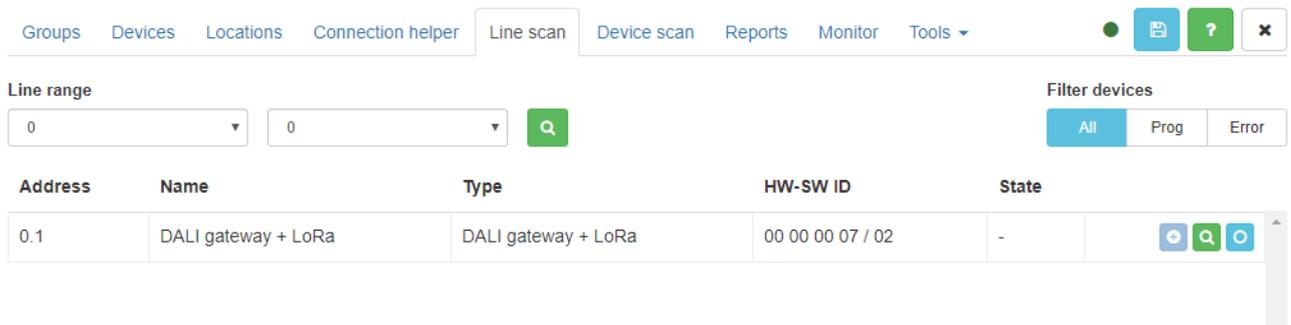
Two apps are mandatory for the CANx DALI Gateway:

- CANx app
- CANx DALI gateway

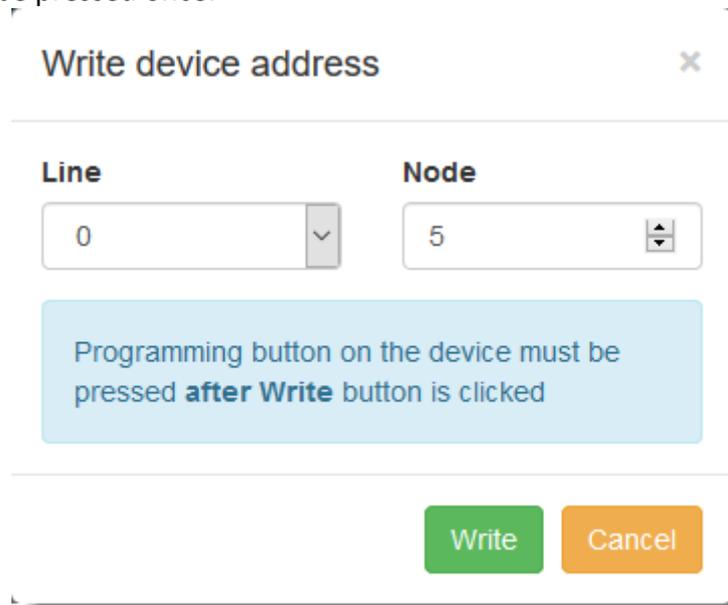
1. CANx app

As first step CANx DALI gateway must be configured in CANx app

- Open the app and scan the line under **Line scan** tab. The gateway will be found as 0.1 default address.



- Readdress the gateway to desire number by selecting **Tools -> Write device address**. Set the node number and press **Write**. After this procedure programming button on the DALI gateway has to be pressed once.



- Add device to Project by pressing 
- Go to Devices tab and select Configure 
- Set the **optional** association to DALI power supply short circiue status.

- All
- Enabled
- Disabled
- DALI
- LoRa general
- LoRa messages
- LoRa - CANx filtering
- LoRa security

DALI power supply short circuit status (read-only) 🔒

DALI power supply short circuit status (read-only)

Enabled ▼

Flags

F T R W

Group addresses + Add 1 bit (boolean)

✕ 0/0/5 DALI gateway + LoRa - DALI power supply short circuit status (read-only)

🔍

Tags

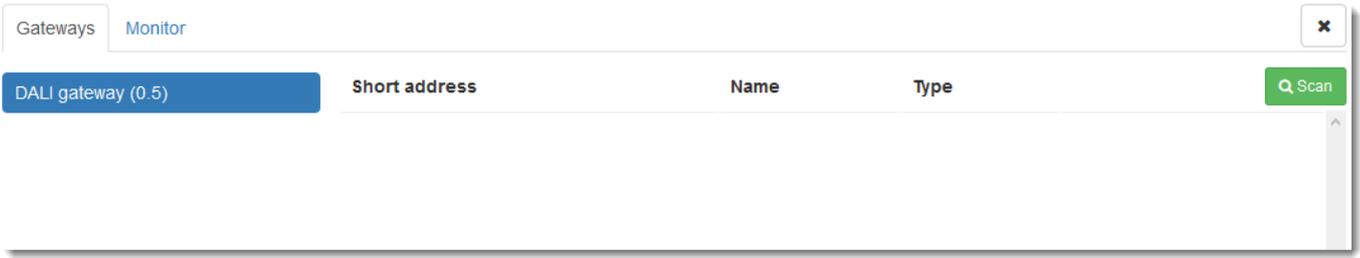
🔍 No tags set

📄 Save and write to device
Save
Cancel

- Once done press 📄 Save and write to device

2. CANx DALI gateway app

- Open the CANx DALI gateway app and the configured DALI gateway will be available to select.



- Press **Scan** to search for connected slaves. Four options are possible

Select scan mode

- Full scan, set short addresses
- Full scan, clear device mapping
- Full scan, keep device mapping
- Partial scan, only add new devices

- Select desired option and scan will proceed. Found slaves will be available as list.

Gateways Monitor

DALI gateway (0.5)

Short address	Name	Type	
0		arc	 
1		arc	 
2		arc	 

Q Scan

- Select  to assign group addresses to slave.
- Give a name to the slave, Select or create new group address for the **On/Off** and **Arc value**. On preset level is the value which will be sent to the slave on **On** command. 254 is equal to 100%. Mapped group addresses are available both in CANx app objects list as well as in LogicMachine objects list.
- Press  to manually control the slave arc level.

Set value of device 0

Arc value (brightness) Mask (no change)

0

Save Cancel

Monitor

Monitor tool is used to monitor and send all commands on the DALI network.

Gateways Monitor

Line: 0 Node: 1 Command: arc Address type: Broadcast Address: Value (0..255): 0

#	Time	Type	Transaction	Address	Raw data	Information
2881	11:16:16.252	RX	36 62 F4 E2 5D 4E A6 40	0.5	11 00	ACK; DATA = 00
2880	11:16:16.219	TX	36 62 F4 E2 5D 4E A6 40	0.5	50 01 A0	ANSW; SHORT 0; CMD = queryactual (160)
2879	11:15:22.719	RX	BC 8B 44 EC 8E 9B 5C 08	0.5	11 00	ACK; DATA = 00
2878	11:15:22.685	TX	BC 8B 44 EC 8E 9B 5C 08	0.5	50 01 A0	ANSW; SHORT 0; CMD = queryactual (160)
2877	11:15:16.900	RX	A4 5F 9B 23 10 DC 0F 5A	0.5	10	ACK
2876	11:15:16.869	TX	A4 5F 9B 23 10 DC 0F 5A	0.5	10 00 00	SHORT 0; ARC; VAL = 0
2875	11:15:11.088	RX	49 BC EB 20 F7 A8 49 71	0.5	11 FE	ACK; DATA = FE

- To send a command select DALI gateway line and a node. Select command, address type, address and respective value.
All the commands correspond to DALI specifications.

LoRa General settings

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

Frequency TX power Bandwidth Spreading Factor

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

Frequency TX power Bandwidth Spreading Factor

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. Bandwidth should be equal on transmitter and receiver(-s).

Frequency TX power Bandwidth Spreading Factor

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).

Frequency TX power Bandwidth Spreading Factor

Spreading 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 Filter mode Statistics 

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

Flags



-ACK mode | Filter mode | Statistics

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

-ACK mode | Filter mode | Statistics

Statistics

Enabled (Source, RSSI, TX power)

Flags



Group addresses Add 4 byte LoRa status

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

Tags

No tags set

Groups | Devices | Locations | Connection helper | Line scan | Device scan | Reports | Monitor | Tools

Name or address | Datatype | Tags | All tags | Any tag | Location | Exact | Incl. sub | Properties | E | R | P |

Address	Name	Datatype	Tags	Value	Properties	Import KNX project	Add
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('aplibs.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()
```