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

DALI RS485 gateway

ENG - Data sheet
Issue date 8.02.2013

Application

DALI (Digitally Addressable Lighting Interface) RS485 gateway is a device specially designed for management and control of dimmable lights. 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

DALI/RS485	DALI RS485 gateway

Standards and norms compliance

EMC:	EN61000-6-1 EN61000-6-3

Technical data:

Power supply:	18-28V DC 16-18V DC	Power supply for gateway Power supply for the DALI bus
Interface: RS485 interface	RS485 DALI output Maximum DALI RS485 Gateways	1 1 16

DALI interface	Maximum count of Ballasts per one DALI RS485 gateway	64
Clamps:	Power supply:	1.5mm2
	DALI output:	1.5mm2
	RS485:	1.5mm2
	DALI power supply	1.5mm2
Enclosure:	Material:	Polyamide
	Color:	Gray
	Dimensions:	36(W)x91(H)x56(L) mm
Protection:	IP20 according to EN 60529	
Usage temperature:	-5C ... +55C	
Storage temperature:	-20C ... +70C	
Weight:	50g	

 **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.

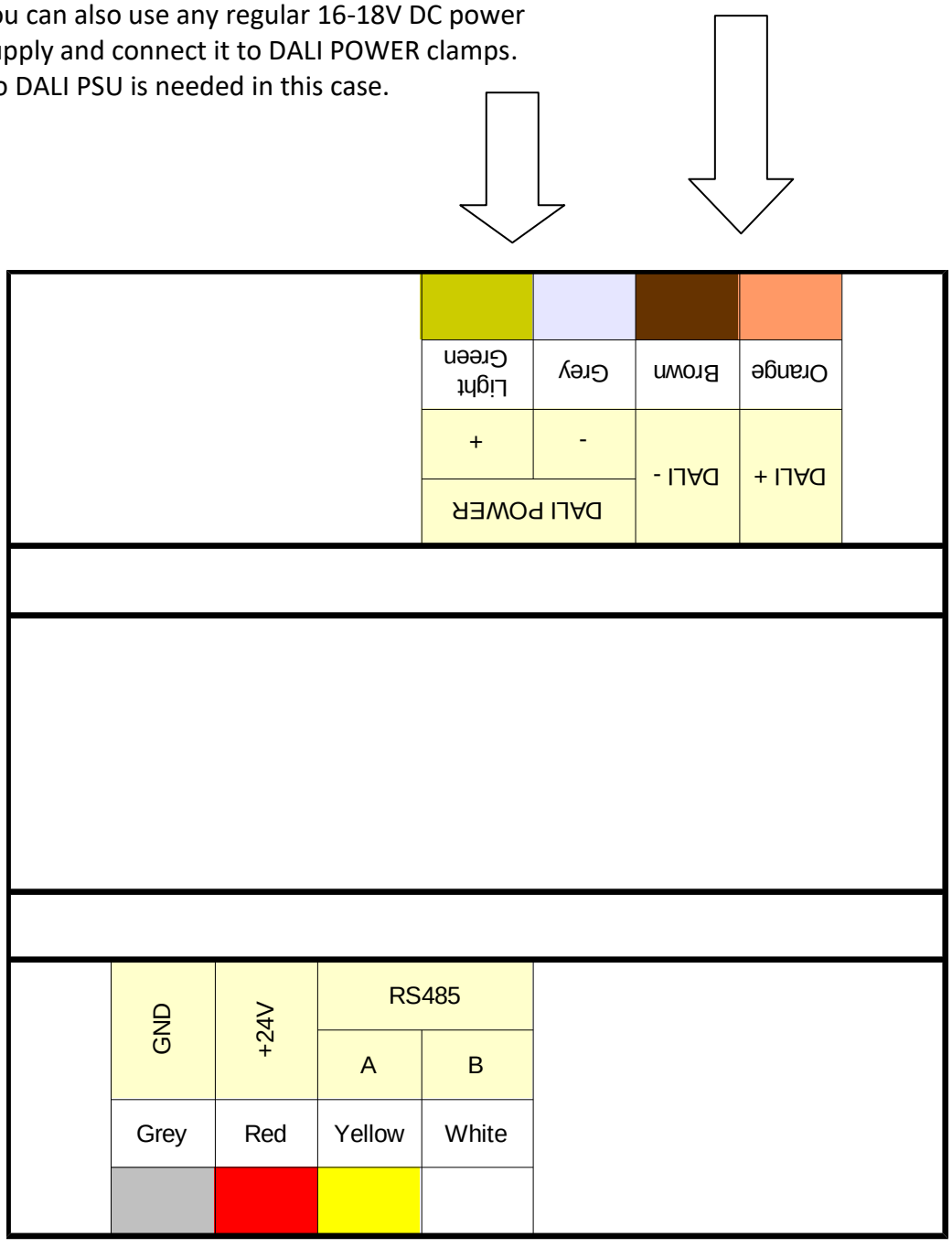
Electrical connection

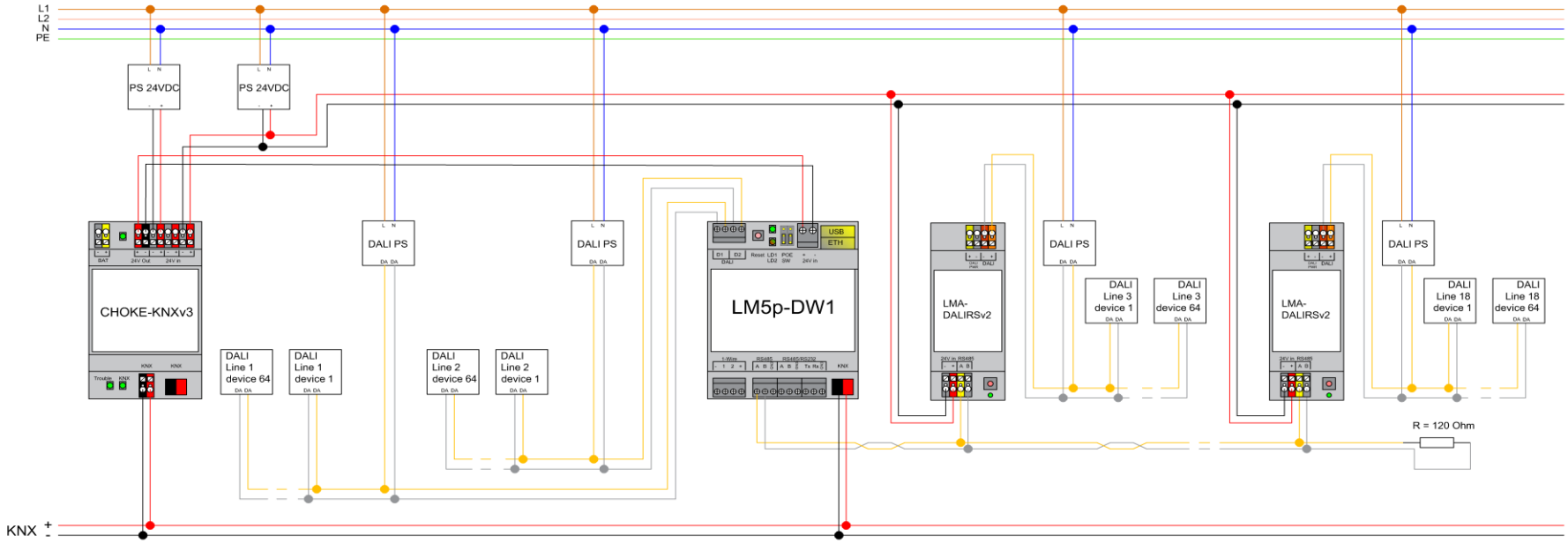
The devices are constructed for the operation of protective low voltage (SELV). Grounding of device not needed. When switching the power supply on or off, power surges must be avoided.

Terminal connection scheme

If DALI power supply is used, you can connect directly to the DALI port

You can also use any regular 16-18V DC power Supply and connect it to DALI POWER clamps. No DALI PSU is needed in this case.




L1
L2
N
PE
PS 24VDC
PS 24VDC
CHOKE-KNXv3
DALI PS
DALI PS
LM5p-DW1
USB
ETH
LMA-DALIRSv2
DALI PS
DALI Line 3 device 1
DALI Line 3 device 64
LMA-DALIRSv2
DALI PS
DALI Line 18 device 1
DALI Line 18 device 64
DALI Line 1 device 64
DALI Line 1 device 1
DALI Line 2 device 64
DALI Line 2 device 1
R = 120 Ohm
KNX + -

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

Usage

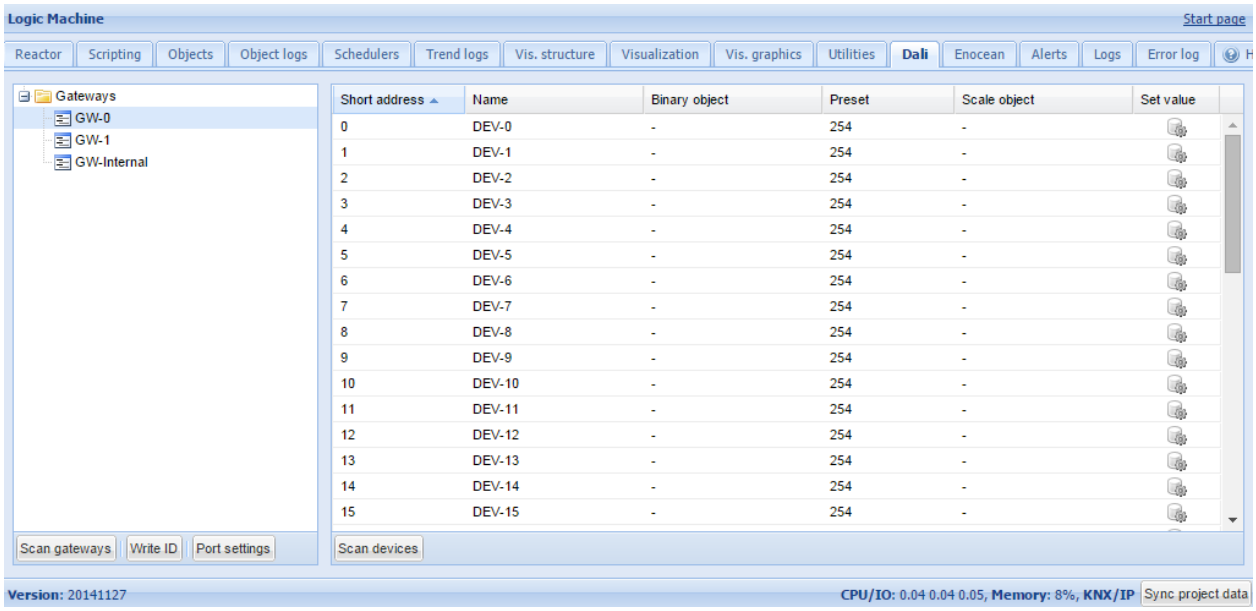
Note! Make sure Modbus RTU is not enabled for the same RS-485 if you are using external DALI RS485 gateways connected to LM's RS-485 port.

Note! Make sure you have series connection for all DALI gateways.

Note! In case you have long RS-485 lines between LM and DALI gateway, make sure to use 120 Ohm termination resistors on LM and on the farthest DALI gateway.

Note! Make sure GND are connected (if you use separate power supplies for LM and DALI gw)

DALI configuration is located in DALI tab.



- ***Scan gateways*** - scans for currently connected gateways, address mapping for missing devices is deleted automatically
- ***Write ID*** - allows setting a unique address for each gateway
- ***Scan devices*** - scans for currently connected DALI devices to the selected gateway, assigns short address automatically. You can also set not to overwrite existing addresses during scan
- ***Port settings*** – serial port name if there are external DALI-RS-485 interfaces connected

Make sure you define Port name in **Port settings** when using RS485 DALI gateway. If you use Reactor with 1 x RS485, the port name will be '/dev/RS485'. If you use LM4, the port name is either '/dev/RS485-1', '/dev/RS485-2' or '/dev/RS485-3'.

Then write for each external RS-485/DALI gateway its unique ID. This is done by pressing programming button on RS485/DALI gateway and pushing **Write ID** on LM.

Then you should press **Scan gateways** button on LM. Your programmed gateway should appear.

For each DALI device, you can set a custom name and map to binary on/off and scale object. This allows communication with DALI devices from KNX bus and visualization without any additional scripts.

DALI object mapping

Once DALI objects are scanned, you can click on corresponding object and perform the configuration.

Device name – name of the DALI device
Binary (ON/OFF) object – map to KNX binary object
Preset for binary ON – preset on binary ON
Scale (0-100%) object – map to KNX scale object

You can set up specific value by clicking on this icon 

Access DALI bus from scripts

If you want to access DALI devices from other scripts, you can use **dalicmd** function.

```
res, err = dalicmd(gwid, cmd, params)
```

Parameters

- `gwid` *(number/string)* gateway id: gateway number or `internal` when internal DALI exists
- `cmd` *(string)* command to send, refer to command table for possible values
- `params` *(table)* command parameters

Params (Lua table):

- `addrtype` *(string)* address type, only required for addressable commands, possible values: `short` `group` `broadcast`
- `address` *(number)* short or group address
- `value` *(number)* additional value to send

3 addressing modes are supported

- `broadcast` all slaves should react: `{ addrtype = 'broadcast' }`
- `short` only one slave having a unique short address should react: `{ addrtype = 'short', address = SLAVE_ID }`
- `group` several slaves belonging to a group should react: `{ addrtype = 'group', address = GROUP_ID }`

Command types

If command is addressable, it's possible to provide address type and address in `params` table.

If command expects a reply it must be addressed so only one slave can reply, otherwise a collision will happen. In case of success, reply is a binary string, usually consisting of a single byte. You can convert it to number like this:

```
-- query status of slave with short address 5 on the internal DALI bus
res, err = dalicmd('internal', 'querystatus', { addrtype = 'short', address = 5 })
-- read ok
if res then
  status = res:byte()
end
```

If command has a value range, params table must have a value field which is an integer in the specified range. For example, `arc` command accepts a value from 0 to 254:

```
-- set level to 42 for all slave on the internal DALI bus
dalicmd('internal', 'arc', { addrtype = 'broadcast', value = 42 })
```

Setting DTR

For commands where DTR is needed prior to executing command, use `setdtr` command to set the value:

```
-- set dtr for ballast 5 to 200

dalicmd('internal', 'setdtr', { addrtype = 'short', address = 5, value = 200 })
```

Example (use gateway with id 1, switch all ballasts off, set ballast with short address 5 to full on)

```
require('user.dali')


dalicmd(1, 'arc', { addrtype = 'broadcast', value = 0 })

dalicmd(1, 'arc', { addrtype = 'short', address = 5, value = 254 })
```

Example (set maximum value for ballast 5 to value 200; the ballast is connected on internal DALI gateway on LogicMachine)

```
require('user.dali')


dalicmd('internal', 'setdtr', { addrtype = 'short', address = 5, value = 200 })
dalicmd('internal', 'storemax', { addrtype = 'short', address = 5 })
```

Example (log all ballast short addresses which are connected to internal DALI gateway)

```
require('user.dali')
res, err = dalicmd('internal', 'queryshortaddr', { addrtype = 'broadcast' })
if res then
   log(res:byte())
else
   log(err)
end
```

Example (add 4 DALI short addressed to one group with nr. 7)

```
require('user.dali')
dalicmd('internal', 'addtogroup', { addrtype = 'short', address = 1, value = 7 })
dalicmd('internal', 'addtogroup', { addrtype = 'short', address = 2, value = 7 })
dalicmd('internal', 'addtogroup', { addrtype = 'short', address = 3, value = 7 })
dalicmd('internal', 'addtogroup', { addrtype = 'short', address = 4, value = 7 })
```

Setting group 7 to a certain value:

```
require('user.dali')

value = event.getvalue()

value = math.floor(value * 2.54)

dalicmd('internal', 'arc', { addrtype = 'group', address = 7, value = value })
```

Example (functions for calling and saving scenes, can be used not only for DALI)

First, you have to define each scene via a Lua table where each item is a table with two items: group address and value. Each scene has a unique id which can be a number or a string.

callscene(id) sets all objects in given scene to their specified value. First it looks for a saved scene in storage and uses default values if storage is empty.

savescene(id) gets current values of all objects from given scene and saves the whole scene in storage.

```
scenes = {}

scenes[1] = {

   { '15/1/1', 50 },

   { '15/1/2', 75 },

   { '15/1/3', 90 },

}

function callscene(id)

  local key, scene

  key = 'scene_' .. id

  scene = storage.get(key, scenes[ id ])

  if type(scene) ~= 'table' then

    alert('Scene ' .. id .. ' not found')

    return

  end

  for _, item in ipairs(scene) do

    grp.write(item[ 1 ], item[ 2 ])
```

```
  end
end


function savescene(id)
  local key, scene
  scene = scenes[ id ]
  if type(scene) ~= 'table' then
    alert('Scene ' .. id .. ' not found')
    return
  end
  for i, item in ipairs(scene) do
    scene[ i ][ 2 ] = grp.getvalue(item[ 1 ])
  end
  key = 'scene_' .. id
  storage.set(key, scene)
end
```

Example (Binary dimmer for DALI lamps to be able dim DALI lamp from physical pushbutton)

1) Add *bindimmer* function to Common functions

```
function bindimmer(up, down, out, event)
  local main, rev, step, val, new, delay
  step = 10 -- in %
  delay = 0.5 -- in seconds
  -- ignore "stop" command
  val = tonumber(event.datahex, 16)
  if val == 0 then
    return
  end
  -- up, normal mode
  if event.dst == up then
    main, rev = up, down
```

```
-- down, reverse step
elseif event.dst == down then
  main, rev = down, up
  step = -step
-- invalid object
else
  return
end
-- current output object value
val = grp.getvalue(out) or 0
while true do
  -- main object in "stop" state
  if not grp.getvalue(main) then
    return
  end
  -- reverse object in "start" state
  if grp.getvalue(rev) then
    return
  end
  -- get new value
  new = math.min(100, val + step)
  new = math.max(0, new)
  -- no change, stop
  if new == val then
    return
  end
  -- write new value
  val = new
  grp.write(out, new, dt.scale)
  -- wait for next run
  os.sleep(delay)
```

```
  end

end
```

2) Create 3 objects:
   1/1/1 - binary (dim up)
   1/1/2 - binary (dim down)
   1/1/3 - 1-byte scale (output)

3) Create an event script for each binary object:

```
bindimmer('1/1/1', '1/1/2', '1/1/3', event)
```

4) You can tune *step* and *delay* variables in *bindimmer* function to adjust dimming speed.

DALI commands

Command	Description	Addressable	Reply	Value
arc	direct arc power control	+		0..254
off	turn off	+		
up	turn on	+		
down	down	+		
stepup	step up	+		
stepdown	step down	+		
recallmin	recall max level	+		
recallmax	recall min level	+		
stepdownoff	step down and off	+		
stepupon	on and step up	+		
gotoscene	go to scene			0..15
reset	reset	+		
storeactual	store actual level in the dtr	+		
storemax	store the dtr as max level	+		
storemin	store the dtr as min level	+		
storesystemfailure	store the dtr as system failure level	+		
storepoweron	store the dtr as power on level	+		
storefadetime	store the dtr as fade time	+		
storefaderate	store the dtr as fade rate	+		
storescene	store the dtr as scene	+		0..15
removescene	remove from scene	+		0..15
addtogroup	add to group	+		0..15
removefromgroup	remove from group	+		0..15
storeshortaddress	store dtr as short address	+		

querystatus	query status	+	+	
queryballast	query ballast	+	+	
querylampfailure	query lamp failure	+	+	
querylamppoweron	query lamp power on	+	+	
querylimiterror	query limit error	+	+	
queryresetstate	query reset state	+	+	
querymissingshort	query missing short address	+	+	
queryversion	query version number	+	+	
querydtr	query content dtr	+	+	
querydevicetype	query device type	+	+	
queryphysicalmin	query physical minimum level	+	+	
querypowerfailure	query power failure	+	+	
queryactual	query actual level	+	+	
querymax	query max level	+	+	
querymin	query min level	+	+	
querypoweron	query power on level	+	+	
querysystemfailure	query system failure level	+	+	
queryfadetimerate	query fade time / fade rate	+	+	
queryscene	query scene level (scenes 0-15)	+	+	0..15
querygroupslow	query groups 0-7	+	+	
querygroupshigh	query groups 8-15	+	+	
queryrandomaddrh	query random address (h)	+	+	
queryrandomaddrm	query random address (m)	+	+	
queryrandomaddrl	query random address (l)	+	+	
terminate	terminate			
setdtr	set data transfer register (dtr)			0..255
initialise	initialise			
randomise	randomise			
compare	compare		+	
withdraw	withdraw			
searchaddrh	set search address (h)			0..255
searchaddrm	set search address (m)			0..255
searchaddrl	set search address (l)			0..255
programshortaddr	program short address			0..63
verifyshortaddr	verify short address		+	0..63
queryshortaddr	query short address		+	
physicalselection	physical selection			
enabledevicetype	enable device type x			0..255