Nordic Automation Systems

LUMINAIRE CONTROLLER UL20xx FW 1.0.x+ COMMUNICATION PROTOCOLS

This document describes the LoRaWAN® application protocol for the following NAS devices:

UL2002
UL2014
UL2020
UL2030

FW version >=1.0.3

INDEX

* ⇈ - Uplink
** ⇊ - Downlink
*** ⇅ - Bidirectional

fPort - 24 Status packet

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10	Byte 11	Byte 12	Byte 13	Byte 14	Byte 15	Byte ..
device_unix_epoch				*status_field*	*down-link_rssi*	*down-link_snr*	*tem-pera-ture*	*analog_inter-faces*	*thr**	*ldr**	*profile_1*					*pro-file_n***
type: uint32 Number of seconds that have elapsed since January 1, 1970 (midnight UTC/GMT)				See fig. 1	type: uint8 unit: dBm con-vert: *-1	type: int8 unit: dB	type: int8 unit: °C	See fig. 2	type: uint8	type: uint8	*pro-file_id* type: uint8 0..254 255 - no_pro-file	*profile_ version* type: uint8 0..240 ***	*dali_ad-dress_ short* type: uint8 value - masked ****	*days_ active* See fig. 3	*dim_ level* type: uint8 0..100 unit: %	See *pro-file_1*

1.

Bit #	Parameter	Value
0	*dali_error_external* (lamp/led, ballast)	0 - ok 1 - error
1	*dali_error_con-nection*	
2	*ldr_state*	0 - off 1 - on
3	***thr_state***	0 - off 1 - on
4	*dig_state*	0 - off 1 - on
5	*hardware_error*	0 - ok 1 - error
6	*firmware_error*	
7	*relay_state*	0 - open 1 - closed

2.

Bit #	Parameter	Value
0	*thr*	0 - not reported 1 - reported
1	*ldr*	0 - not reported 1 - reported
2	*od*	0 - open 1 - closed
3	RFU	
4	RFU	
5	RFU	
6	RFU	
7	RFU	

3.

Bit #	Parameter	Value
0	*holiday*	0 - not active 1 - active
1	*mon*	
2	*tue*	
3	*wed*	
4	*thu*	
5	*fri*	
6	*sat*	
7	*sun*	

4.

profile_version	Out of sequence reason		profile_version	Out of sequence reason
240..245	RFU		251	*thr_active*
246	*ballast_not_found*		252	*dig_active*
247	*calendar_active*		253	*manual_active*
248	*default_dim_active*		254	*value_differ*
249	*profile_not_active*		255	*unknown*
250	*ldr_active*			

* only reported if the parameter is marked as reported in *analog_interfaces*
** profile blocks are repeated as many times as there are profiles/drivers
*** *profile_version* values higher than 240 are used when the actual dimming value is not according to the profile to explain the reason why. See fig 4. for more details.
**** See dali address mapping for more details.

Legacy status packet

0.6.x FW compatible status packet. To enable this format, check “legacy_mode” bit in “Defaults configuration packet”

<table>
<tr><th>Byte 0</th><th>Byte 1</th><th>Byte 2</th><th>Byte 3</th><th>Byte 4</th><th>Byte 5</th><th>Byte 11</th><th>Byte 12</th><th>Byte 13</th><th>Byte 14</th><th>Byte 15</th><th>Byte ..</th></tr>
<tr><td colspan="4">device_unix_epoch</td><td>status_field</td><td>down-link_rssi</td><td colspan="5">profile_1</td><td>profile_n**</td></tr>
<tr><td colspan="4" rowspan="2">type: uint32
Number of seconds that have elapsed since January 1, 1970 (midnight UTC/GMT)</td><td rowspan="2">See fig. 1</td><td rowspan="2">type: uint8
unit: dBm
convert: *-1</td><td>profile_id</td><td>profile_ver-sion</td><td>dali_ad-dress_short</td><td>days_active</td><td>dim_level</td><td rowspan="2">See pro-file_1</td></tr>
<tr><td>type: uint8
0..254
255 - no_profile</td><td>type: uint8
0..240 ***</td><td>type: uint8
value - masked
****</td><td>See fig. 3</td><td>type: uint8
0..100
unit: %</td></tr>
</table>

Message sample

Message in base64

```
39QdXgBLBBUCrgUFCv8yAwMG/wA=
```

Message decoded to HEX

```
DFD41D5E004B041502AE05050AFF32030306FF00
```

device_unix_epoch `DFD41D5E` flip for MSB

```
0x5E1DD4DF
```

HEX message converted to decimal (epoch)

```
1579013343 (seconds)
```

```
Epoch time converted to date
```

```
14 January 2020 14:49:03 (UTC)
```

status_field HEX message

```
0x00
```

HEX message converted to binary

```
0B00000000
```

Binary converted to statuses (LSB)

```
0 : dali_error_external - ok
0 : dali_error_connection - ok
0 : ldr_state - off
0 : thr_state - off
0 : dig_state - off
0 : hardware_error - ok
0 : software_error - ok
0 : relay_state - open
```

downlink_rssi HEX message

```
0x4B
```

HEX message converted to decimal

```
75
```

Decimal value multiplied by -1

```
-75 (dBm)
```

downlink_snr HEX message

```
0x04
```

HEX message converted to signed decimal

```
4 (dB)
```

temperature HEX message

```
0x15
```

HEX message converted to signed decimal

```
21 (°C)
```

analog_interfaces HEX message

```
0x02
```

HEX message converted to binary

```
0B00000010
```

Binary converted to statuses (LSB)

```
0 : RFU
1 : ldr - reported
0 : RFU
0 : RFU
0 : RFU
0 : RFU
0 : RFU
0 : RFU
```

ldr HEX message

```
0xAE
```

HEX message converted to decimal

```
174
```

profile_1 HEX message

```
0x05050AFF32
```

profile_id in HEX

```
0x05
```

converted to decimal

```
5
```

profile_version in HEX

```
0x05
```

converted to decimal

```
5
```

dali_address_short in HEX

```
0x0A
```

converted to DALI address

```
Single address 5
```

days_active setting in HEX

```
0xFF
```

converted to binary

```
0B11111111
```

Binary converted to statuses (LSB)

```
1 : holiday - active
1 : mon - active
1 : tue - active
1 : wed - active
1 : thu - active
1 : fri - active
1 : sat - active
1 : sun - active
```

dim_level in HEX

```
0x32
```

converted to decimal

```
50%
```

profile_2 HEX message

```
0x030306FF00
```

profile_id in HEX

```
0x03
```

converted to decimal

```
3
```

profile_version in HEX

```
0x03
```

converted to decimal

```
3
```

dali_address_short in HEX

```
0x06
```

converted to DALI address

```
Single address 3
```

days_active setting in HEX

```
0xFF
```

converted to binary

```
0B11111111
```

Binary converted to statuses (LSB)

```
1 : holiday - active
1 : mon - active
1 : tue - active
1 : wed - active
1 : thu - active
1 : fri - active
1 : sat - active
1 : sun - active
```

dim_level in HEX

```
0x00
```

converted to decimal

```
0%
```

fPort 25 - Usage Message*

Byte 0**	Byte 1	Byte ..	Byte n
dali_address_short	reported_fields	Payload	
type: uint8 value -masked	See fig. 1		

1.

Bit #	Parameter	Type and unit
0	*active_energy_total*	uint32, Wh
1	*active_energy_instant*	uint16, W
2	*load_side_energy_total*	uint32, Wh
3	*load_side_energy_instant*	uint16, W
4	*power_factor_instant*	uint8 ***
5	*system_voltage*	uint8, V
6	*driver_operating_time*	uint32, sec
7	*lamp_on_time*	uint32, sec

* If driver supports. Also drivers need to have short addresses assigned.
** 0xFF values measured by the controller.
*** Needs to be dived by 100.

Usage parameters are reported after driver address in order of the reported fields mapping. If more than one driver are connected, then next drivers address is reported after the last parameter of the first driver and followed by the parameters in the same logic.

Message sample

Message in base64

`BAMAAAAAAAAGAxUUAAAAAA==`

Message decoded to HEX

`04|03|00000000|0000|06|03|15140000|0000`

Address 1

dali_address_short in HEX

`0x04`

Profile HEX address converted to DALI address

Single address 2

reported_fields in HEX

`0x03`

Reported fields converted to binary

`0B00000011`

Binary converted to statuses (LSB)

```
1 : active_energy_total - sent
1 : active_energy_instant - sent
0 : load_side_energy_total - not sent
0 : load_side_energy_instant - not sent
0 : power_factor_instant - not sent
0 : system_voltage - not sent
0 : driver_operating_time - not sent
0 : lamp_on_time - not sent
```

Active energy total `00000000` message flip for MSB

`0x00000000`

HEX message converted to decimal

`0 (Wh)`

Current consumption `0000` message flip for MSB

`0x0000`

HEX message converted to decimal

`0 (W)`

Address 2

DALI address in HEX

`0x06`

Profile HEX address converted to DALI address

Single address 3

Reported fields in HEX

```
03
```

Reported fields converted to binary

```
0B00000011
```

Binary converted to statuses (LSB)

```
1 : active_energy_total - sent
1 : active_energy_instant - sent
0 : load_side_energy_total - not sent
0 : load_side_energy_instant - not sent
0 : power_factor_instant - not sent
0 : system_voltage - not sent
0 : driver_operating_time - not sent
0 : lamp_on_time - not sent
```

Active energy total `15140000` message flip for MSB

```
0x00001415
```

HEX message converted to decimal

```
5141 (Wh)
```

Current consumption `0000` message flip for MSB

```
0x0000
```

HEX message converted to decimal

```
0 (W)
```

fPort 49 - Configuration Request Message

LDR configuration request

Byte 0	Operation
0x01	*ldr_config_request*

Message sample

Message goal: Request LDR configuration

Header

Select Header HEX code

```
0x01
```

Compile message for sending (HEX)

```
0x01
```

Control value in base64 to control after sending

```
AQ==
```

Response

Sent to fPort in the same format as *ldr_config_packet*

DIG configuration request

Byte 0	Operation
0x03	*dig_config_request*

Message sample

Message goal: Request DIG configuration

Header

Select Header HEX code

```
0x03
```

Compile message for sending (HEX)

```
0x03
```

Control value in base64 to control after sending

```
Aw==
```

Response

Sent to fPort 50 in the same format as *dig_config_packet*

Calendar configuration request

Byte 0	Operation
0x06	*calendar_config_request*

Message sample

Message goal: Request calendar configuration

Header

Select Header HEX code

```
0x06
```

Compile message for sending (HEX)

```
0x06
```

Control value in base64 to control after sending

```
Bg==
```

Response

Sent to fPort 50 in the same format as *calendar_config_packet*

Status configuration request

Byte 0	Operation
0x07	*status_config_request*

Message sample

Message goal: Request status configuration

Header

Select Header HEX code

```
0x07
```

Compile message for sending (HEX)

```
0x07
```

Control value in base64 to control after sending

```
Bw==
```

Response

Sent to fPort 50 in the same format as *status_config_packet*

Profile configuration request

Byte 0	Byte 1
profile_config_request	*profile_id*
0x08	type: uint8 0..254 (255 - request list of profile IDs)

Message sample

Message goal: Request configuration for *profile_id* 6

Header

Select Header HEX code

```
0x08
```

profile_id

Convert *profile_id* 6 to HEX

```
0x06
```

Compile message for sending (HEX)

```
0x0806
```

Control value in base64 to control after sending

```
CAY=
```

Response

Sent to fPort 50 in the same format as *profile_config_packet*

Default dim

Byte 0	Operation
0x0A	*default_dim_config_request*

Message sample

Message goal: Request defaults configuration

Header

Select Header HEX code

```
0x0A
```

Compile message for sending (HEX)

```
0x0A
```

Control value in base64 to control after sending

```
Cg==
```

Response

Sent to fPort 50 in the same format as *default_dim_config_packet*

Usage configuration request

Byte 0	Operation
0x0B	*usage_config_request*

Message sample

Message goal: Request usage configuration

Header

Select Header HEX code

```
0x0B
```

Compile message for sending (HEX)

```
0x0B
```

Control value in base64 to control after sending

```
Cw==
```

Response

Sent to fPort 50 in the same format as *usage_config_packet*

Holiday configuration request

Byte 0	Operation
0x0C	*holiday_config_request*

Message sample

Message goal: Request holiday configuration

Header

Select Header HEX code

```
0x0C
```

Compile message for sending (HEX)

```
0x0C
```

Control value in base64 to control after sending

```
DA==
```

Response

Sent to fPort 50 in the same format as *holiday_config_packet*

Boot delay configuration request

Byte 0	Operation
0x0D	*boot_delay_config_request*

Message sample

Message goal: Request boot delay configuration

Header

Select Header HEX code

```
0x0D
```

Compile message for sending (HEX)

```
0x0D
```

Control value in base64 to control after sending

```
DQ==
```

Response

Sent to fPort 50 in the same format as *boot_delay_config_packet*

Defaults configuration request

Byte 0	Operation
0x0E	*defaults_config_request*

Message sample

Message goal: Request default configuration

Header

Select Header HEX code

```
0x0E
```

Compile message for sending (HEX)

```
0x0E
```

Control value in base64 to control after sending

```
Dg==
```

Response

Sent to fPort 50 in the same format as *defaults_config_packet*

Position configuration request

Byte 0	Operation
0x13	*meta_pos_config_request*

Message sample

Message goal: Request position configuration

Header

Select Header HEX code

```
0x13
```

Compile message for sending (HEX)

```
0x13
```

Control value in base64 to control after sending

```
Ew==
```

Response

Sent to fPort 50 in the same format as *meta_pos_config_packet*

Multicast configuration request

Byte 0	Byte 1
multicast_config_request	*multicast_device*
0x52	type: uint8 1..4

Message sample

Message goal: Request configuration for *multicast_device* 2

Header

Select Header HEX code

```
0x52
```

multicast_device

Convert *multicast_device* 2 to HEX

```
0x02
```

Compile message for sending (HEX)

```
0x5202
```

Control value in base64 to control after sending

```
UgI=
```

Response

Sent to fPort 50 in the same format as *multicast_config_packet*

fPort 50 - Configuration Message

LDR configuration packet

This configuration is the “LDR” dim source in luminaire control chart. Please see appendix 1 for more information.

Byte 0	Byte 1	Byte 2	Byte 3
ldr_config_packet	*switch_thresholds*		*switch_behaviour*
	*high** (switch off / clear alert)	*low** (switch on / create alert)	
0x01	type: uint8 0xFF - disabled	type: uint8 0xFF - disabled	See fig. 1

*High and low are raw ADC values.

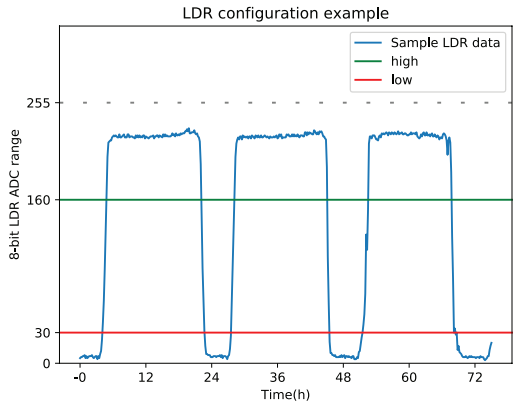
1.

Bit #	Function	Value
0	RFU	
1	RFU	
2	*trigger_alert*	0: false 1: true
3	RFU	
4	RFU	
5	RFU	
6	RFU	
7	RFU	

This feature requires hardware support. Not all devices are supported.

Message sample

Message goal: Configure LDR to control luminaires and enable alerts.



This chart vizualizes typical outdoor LDR behaviour.

Actual LDR values vary by device type and installation location.

Select Header HEX code

```
0x01
```

switch_thresholds

high

Convert threshold value `160` to HEX

```
A0
```

low

Convert threshold value `48` to HEX

```
30
```

functions

Function selection

```
0 : RFU
0 : RFU
1 : trigger_alert - enable
0 : RFU
0 : RFU
0 : RFU
0 : RFU
0 : RFU
```

Selection converted to binary

```
0B00000100
```

Selection converted to HEX

```
0x04
```

Compile message for sending (HEX)

```
01|A0|30|04
```

Control value in base64 to control after sending

```
AaAwBA==
```

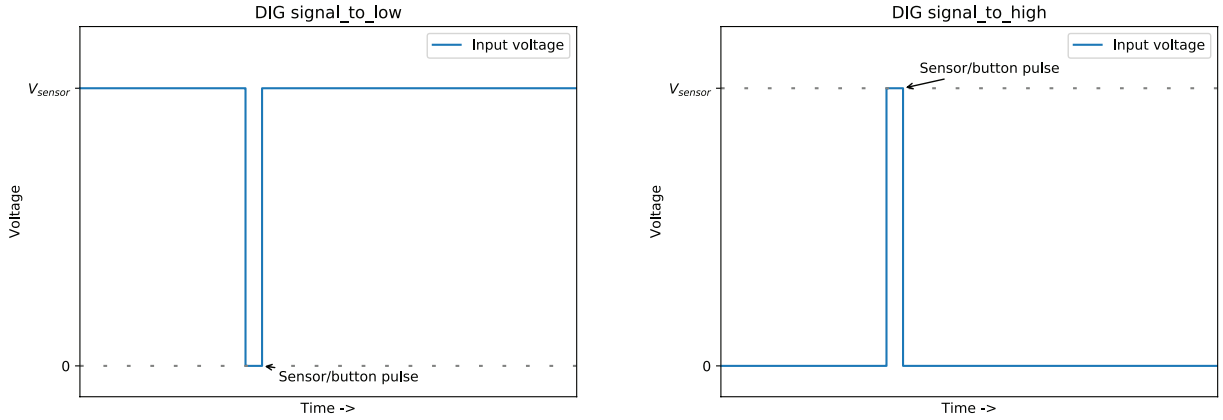
DIG configuration packet

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
dig_config_packet	*switch_time*		*behaviour*	*dali_address_short*	*dim_level*
0x03	type: uint16 unit: seconds 0xFF disabled		See fig. 1	type: uint8 value: value_masked	type: uint8 0..100 unit: %

1.

Bit #	Function	Value
0	RFU	
1	*switch_point**	0: signal_to_low 1: signal_to_high
2	*trigger_alert*	0: false 1: true
3	RFU	
4	RFU	
5	RFU	
6	RFU	
7	RFU	

Following charts show typical swith_point behaviour



* V_{sensor} is Device specific. For event to be registered, pulse duration must be at least 100ms.

This configuration is the “DIG” dim source in luminaire control chart. Please see appendix 1 for more information.
This feature requires hardware support. Not all devices are supported.

Message sample

Message goal: Configure switch on signal to high. Set all lamps to 50% for 5 minutes.

Select Header HEX code

```
0x03
```

switch_time

Convert value `300` to HEX

```
0x012C
```

Flip value for LSB

```
2C01
```

behaviour

Function selection

```
0 : RFU
1 : switch_point - signal_to_high
0 : trigger_alert - false
0 : RFU
0 : RFU
0 : RFU
0 : RFU
0 : RFU
```

Selection converted to binary

```
0B00000010
```

Selection converted to HEX

```
0x02
```

dali_address_short

Convert `Broadcast` adress to HEX

```
0xFE
```

dim_level

Convert `50%` from to HEX

```
0x32
```

Compile message for sending (HEX)

```
03|2C01|02|FE|32
```

Control value in base64 to control after sending

```
AywBAv4y
```

Calendar configuration

This configuration controls luminaires. Please see appendix 1 for more information.

For correct behaviour, requires the device clock to be set. Check Time configuration packet

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
calendar_config_packet	*sunrise_offset*	*sunset_offset*	*latitude*		*longitude*	
0x06*	type: int8 unit: minutes	type: int8 unit: minutes	type: int16 unit: degrees (2 decimals * 100)**		type: int16 unit: degrees (2 decimals * 100)**	

*0x06FFFFFFFFFFFF will disable the sunrise/sunset calendar
**2 decimals times 100. Example : 59.425284 > 59.43 > 5943

Message sample

Message goal: Set lights to turn off 30 minutes before sunrise and turn on 30 minutes after sunrise.
Set calendar to Viimsi, Estonia.

Select Header HEX code

```
0x06
```

sunrise_offset

Convert offset `-30` to HEX

```
E2
```

sunset_offset

Convert offset `30` to HEX

```
1E
```

latitude

Convert coordinates `65.500226` to 2 decimals

```
65.50
```

Multiply with 100 to get rid of the decimal places

```
6550
```

Convert to HEX

```
0x1996
```

Flip value to LSB

```
9619
```

longitude

Convert coordinates `24.833547` to 2 decimals

```
24.83
```

Multiply with 100 to get rid of the decimal places

```
2483
```

Convert to HEX

```
0x09B3
```

Flip value to LSB

```
B309
```

Compile message for sending (HEX)

```
06E21E9619B309
```

Control value in base64 to control after sending

```
BuIelhmzCQ==
```

Status reporting configuration

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
status_config_ packet	*status_interval*			
0x07	type: uint32 unit: seconds minimum - 10min			

Message sample

Message goal: Set status reporting to 1 hour

Select Header HEX code

```
0x07
```

status_interval

Convert interval `3600` to HEX

```
0x00000E10
```

Flip value to LSB

```
100E0000
```

Compile message for sending (HEX)

```
07100E0000
```

Control value in base64 to control after sending

```
BxAOAAA=
```

Profile configuration packet

This configuration is the “Profile” dim source in luminaire control chart. Please see appendix 1 for more information.

For correct behaviour, requires the device clock to be set. Check Time configuration packet

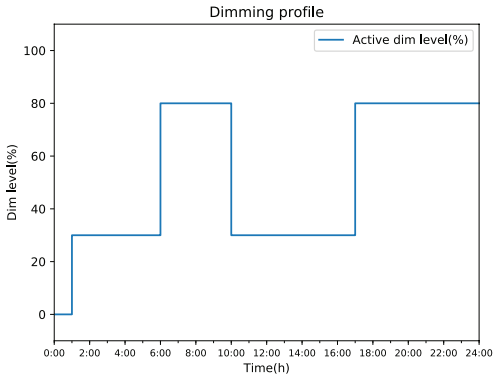
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte ..	Byte ..
profile_config_packet	profile_id	profile_version	dali_address_short	days_active	dimming_step_1		dimming_step_n*	
					step_time**	dim_level	step_time**	dim_level
0x08	type: uint8 0..254	type: uint8 0..240	type: uint8 value: masked	See fig. 1	type: uint8 value: mapped	type: uint8 unit: % 0..100	type: uint8 value: mapped	type: uint8 unit: % 0..100

1.

Bit #	Parameter	Value
0	*holiday*	0 - not active 1 - active
1	*mon*	
2	*tue*	
3	*wed*	
4	*thu*	
5	*fri*	
6	*sat*	
7	*sun*	

* Maximum of 10 steps are allowed. Steps need to be in chonological order.
** UTC time. Step time is 10 min increment from 00:00:00. Look at the profile step time chart for more details.

Message sample



Message goal: Apply profile with *profile_id* 22 *profile_version* 3 to Broadcast addrress to dim the lights (00:00 to 0%, 01:00 to 30%, 06:00 to 80%, 10:00 to 30%, 17:00 to 80%) on Mondays, Tuesdays, Wednesdays and Thursdays. UTC time. See chart for visualization.

Select Header HEX code

```
08
```

profile_id

Convert `22` to HEX

```
0x16
```

profile_version

Convert `3` to HEX

```
0x03
```

dali_address_short

Convert `Broadcast` adress to HEX

```
0xFE
```

days_active

selection

```
0 : holiday - not active
1 : mon - active
1 : tue - active
1 : wed - active
1 : thu - active
0 : fri - not active
0 : sat - not active
0 : sun - not active
```

Selection converted to binary

```
0B00011110
```

Selection converted to HEX

```
0x1E
```

dimming_steps

Choose dimming steps

```
00:00 @ 0%, 01:00 @ 30%, 06:00 @ 80%, 10:00 @ 30%, 17:00 @ 80%
```

Convert steps to decimal(see step time conversion map)

```
0 @ 0%, 6 @ 30%, 36 @ 80%, 60 @ 30%, 102 @ 80%
```

Convert to Hex(first byte step_time, second dim_level)

```
0000 061E 2450 3C1E 6650
```

Compile message for sending (HEX)

```
08|16|03|FE|1E|0000|061E|2450|3C1E|6650
```

Control value in base64 to control after sending

```
CBYD/h4AAAYeJFA8HmZQ
```

Time configuration packet*

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
time_config_packet	*device_unix_epoch*			
0x09	type: uint32 Number of seconds that have elapsed since January 1, 1970 (midnight UTC/GMT)			

* In LoRaWAN networks with support for LoRaWAN 1.0.3+ (DeviceTimeReq) the clock is automatically synced from network.

Message sample

Message goal: Set device clock to 22 August 2017 11:50:00

Select Header HEX code

```
0x09
```

device_unix_epoch

Choose desired time

```
22 August 2017 11:50:00 (UTC)
```

Convert to epoch

```
1503402600
```

Covert to hex

```
0x599C1A68
```

Flip value for LSB

```
681A9C59
```

Compile message for sending (HEX)

```
09681A9C59
```

Control value in base64 to control after sending

```
CWganFk=
```

Default dim configuration packet

This configuration controls luminaires. Please see appendix 1 for more information.

Byte 0	Byte 1	Byte 2
default_dim_config_packet	*default_dim*	*RFU*
0x0A	type: uint8 unit: % 0..100 (Default 100%)	

Message sample

Message goal: Set default light to 0%.

Select Header HEX code

```
0x0A
```

default_dim

Convert `0%` from decimal to HEX

```
0x00
```

Compile message for sending (HEX)

```
0A0000
```

Control value in base64 to control after sending

```
CgAA
```

Usage reporting configuration*

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
usage_config_packet	*usage_interval*				*system_voltage*
0x0B	unit: seconds		type: uint32 minimum - 10min	0x00 - disabled	type: uint8 unit: volts

* usage can be reported only if the driver supports metering

Message sample

Message goal: Set usage reporting to 1 hour and system voltage to 230V.

Select Header HEX code

```
0x0B
```

usage_interval

Convert `3600` to HEX

```
0x00000E10
```

Flip value to LSB

```
100E0000
```

system_voltage

```
0xE6
```

Compile message for sending (HEX)

```
0B100E0000E6
```

Control value in base64 to control after sending

```
CxAOAADm
```

Holiday configuration packet

This setting affects Holiday bit in Profile configuration packet.

Byte 0	Byte 1	Byte 2	Byte ..	Byte ..
holiday_config_packet	*holiday_1*		*holiday_n**	
0x0C	type: uint16 day number 1..365		type: uint16 day number 1..365	

* Holiday are marked with days from January 1. Maximum of 25 holidays is supported. Sending in configuration for new holidays will replace old config

Message sample

Message goal: Configure January 1st, Febraury 24th, May 1st, June 23rd, August 20th and December 23rd as holidays.

Select Header HEX code

```
0C
```

holiday_1

Choose desired day

```
January 1st
```

Convert to day number in year

```
1
```

Convert value to HEX

```
0x0001
```

Flip value for LSB

```
0100
```

holiday_2

Choose desired day

```
February 24th
```

Convert to day number in year

```
55
```

Convert value to HEX

```
0x0037
```

Flip value for LSB

```
3700
```

holiday_3

Choose desired day

`May 1st`

Convert to day number in year

`121`

Convert value to HEX

`0x0079`

Flip value for LSB

`7900`

holiday_4

Choose desired day

`June 23rd`

Convert to day number in year

`174`

Convert value to HEX

`0x00AE`

Flip value for LSB

`AE00`

holiday_5

Choose desired day

`August 20th`

Convert to day number in year

`234`

Convert value to HEX

`0x00E2`

Flip value for LSB

`E200`

holiday_6

Choose desired day

`December 23rd`

Convert to day number in year

`357`

Convert value to HEX

`0x0165`

Flip value for LSB

`6501`

Compile message for sending (HEX)

`0C|0100|3700|7900|AE00|E200|6501`

Control value in base64 to control after sending

`DAEANwB5AK4A4gBlAQ==`

Boot delay configuration packet

Byte 0	Byte 1
boot_delay_config_packet	*boot_delay_range*
0x0D	type: uint8 unit: seconds default: 0

Message sample

Message goal: Set boot delay to 120 seconds.

Select Header HEX code

```
0x0D
```

boot_delay_range

Convert `120` to HEX

```
0x78
```

Compile message for sending (HEX)

```
0D78
```

Control value in base64 to control after sending

```
DXg=
```

Defaults configuration packet

*Same as default dim configutaion packet

Byte 0	Byte 1	Byte 2	Byte 3
defaults_config_packet	*configured_parameters*	*default_dim**	*fade*
0x0E	See fig. 1	type: uint8 0..100 unit: %	See fig. 2

*Same as default dim configutaion packet

1.

Bit #	Function	Value
0	*default_dim*	0: not present in packet 1: configured
1	RFU	
2	*fade*	
3	RFU	
4	*legacy_mode***	
5	RFU	
6	RFU	
7	RFU	

** 1 = Status packet is sent in 0.6.x compatible format. Check “Legacy status packet”. Should only be used if compatibility is desired.
0 = The default status packet format in 1.0.x. Check “fPort/type - 24 Status packet

2.

Value	Fade time (sec)		Value	Fade time (sec)
0	< 0.71		9	11.31
1	0.71		10	16
2	1		11	22.63
3	1.41		12	32
4	2		13	45.25
5	2.83		14	64
6	4		15	90.51
7	5.66		255	ignore (default)
8	8			

In case of value FF(255) UL20xx will not change the value of fade time in DALI ballasts.

Message sample

Message goal: Set *default_dim* to 0% and *fade* to 2.83 seconds.

Select Header HEX code

```
0x0E
```

configured_parameters

selection

```
1 : default_dim - configured
0 : RFU
1 : fade - configured
0 : RFU
0 : legacy_mode - not set
0 : RFU
0 : RFU
0 : RFU
```

Selection converted to binary

```
0B00000101
```

Selection converted to HEX

```
0x05
```

default_dim

Convert `0%` to HEX

```
0x00
```

fade

Convert `2.83` secods to HEX

```
0x05
```

Compile message for sending (HEX)

```
0E|05|00|05
```

Control value in base64 to control after sending

```
DgUABQ==
```

Meta position configuration packet

Byte 0	Byte 1	Byte 2..5	Byte 6..9	Byte 10	Byte 11..x
packet_type	*configured_parameters*	*gps_position*		*address*	
		latitude	*longitude*	*length*	*data[0]..data[length-1]*
0x10 *meta_pos_config_packet*	See fig. 1	type: int32 value: $*10^7$ when: *gps_position* = *configured* 0x7FFFFFFF - not_configured	type: int32 value: $*10^7$ when: *gps_position* = *configured* 0x7FFFFFFF - not_configured	type: uint8 unit: B when: *address* = *configured* max=38	type: char array (utf-8) when: *address* = *configured*

1.

Bit #	Parameter	Value
0	*gps_position*	0 - *not_configured* 1 - *configured*
1	*address*	
2	RFU	
3	RFU	
4	RFU	
5	RFU	
6	RFU	
7	RFU	

Message sample

Message goal: Set the device coordinates to 59.4286400, 24.6610052

Header

Select Header HEX code

```
0x10
```

configured_parameters

```
1 : gps_position - configured
0 : address - not_configured
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
```

Selection converted to binary

```
0B00000001
```

Selection converted to HEX

```
0x01
```

gps_position

latitude decimal value (7 decimal places)

```
59.4286400
```

Decimal value multipleid by 10000000

```
594286400
```

Decimal value converted to HEX

```
0x236C1740
```

HEX message flip for LSB

```
0x40176C23
```

longitude decimal value (7 decimal places)

```
24.6610052
```

Decimal value multipleid by 10000000

```
246610052
```

Decimal value converted to HEX

```
0x0EB2F884
```

HEX message flip for LSB

```
0x84F8B20E
```

Compile message for sending (HEX)

```
10|01|40176C23|84F8B20E
```

Control value in base64 to control after sending

```
EAFAF2wjhPiyDg==
```

Multicast configuration packet

Byte 0	Byte 1	Byte 2..Byte 5	Byte 6..Byte 21	Byte 22..Byte 37
multicast_config_packet	*multicast_device*	*DevAddr*	*NwkSKey*	*AppSKey*
0x52	0x01..0x04*	uint32	HEX	HEX

* Upto 4 different sets of Multicast keys can be configured. When the same set number is used, then the old keys are overwritten.

Message sample

Message goal: Set Multicast group 1 keys to DevAddr: 82840c70,
NwkSKey: 82840c7056429b143d21974557f93a53, AppSKey: 82840c70c08494b931fe2fa6f8835c6a

Select Header HEX code

```
0x52
```

multicast_device

Convert multicast_device 1 to HEX

```
0x01
```

DevAddr in HEX

```
11223344
```

HEX message flip for LSB

```
44332211
```

NwkSKey in HEX

```
82840C7056429B143D21974557F93A53
```

AppSKey in HEX

```
82840C70C08494B931FE2FA6F8835C6A
```

Compile message for sending (HEX)

```
52|01|44332211|82840C7056429B143D21974557F93A53|
82840C70C08494B931FE2FA6F8835C6A
```

Control value in base64 to control after sending

```
UgFEMyIRgoQMcFZCmxQ9IZdFV/k6U4KEDHDAhJS5Mf4vpviDXGo=
```

Clear LDR configuration request

Byte 0	Byte 1
clear_config_packet	*ldr_config*
0xFF	0x01

Message sample

Message goal: Clear LDR configuration

Select *clear_config_packet* HEX code

```
0xFF
```

Select *ldr_config* HEX code

```
0x01
```

Compile message for sending (HEX)

```
FF01
```

Control value in base64 to control after sending

```
/wE=
```

Clear DIG configuration request

Byte 0	Byte 1
clear_config_packet	*dig_config*
0xFF	0x03

Message sample

Message goal: Clear DIG configuration

Select *clear_config_packet* HEX code

```
0xFF
```

Select *dig_config* HEX code

```
0x03
```

Compile message for sending (HEX)

```
FF03
```

Control value in base64 to control after sending

```
/wM=
```

Clear profile configuration request

Byte 0	Byte 1	Byte 2	Byte 3 (optional)
clear_config_packet	*profile_config*	*dali_address_short*	*profile_id*
0xFF	0x04	type: uint8 value: masked 0xFF - *all_profiles*	type: uint8 0..254

Message sample

Message goal: Clear all profiles from single device 5

Select *clear_config_packet* HEX code

```
0xFF
```

Select *profile_config* HEX code

```
0x04
```

dali_address_short

Convert `single device 5` address to HEX

```
0x0A
```

Compile message for sending (HEX)

```
FF040A
```

Control value in base64 to control after sending

```
/wQK
```

Clear holiday configuration request

Byte 0	Byte 1
clear_config_packet	*holiday_config*
0xFF	0x06

Message sample

Message goal: Clear holiday configuration

Select *clear_config_packet* HEX code

```
0xFF
```

Select *holiday_config* HEX code

```
0x06
```

Compile message for sending (HEX)

```
FF06
```

Control value in base64 to control after sending

```
/wY=
```

Clear profile configuration request

Byte 0	Byte 1	Byte 2
clear_config_packet	*multicast_config*	*multicast_device*
0xFF	0x52	type: uint8 01..04 0xFF - all_multicast_devices

Message sample

Message goal: Clear all multicast addresses

Select *clear_config_packet* HEX code

```
0xFF
```

Select *multicast_config* HEX code

```
0x52
```

multicast_device

Convert `all devices` to HEX

```
0xFF
```

Compile message for sending (HEX)

```
FF52FF
```

Control value in base64 to control after sending

```
/1L/
```

Factory reset

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
clear_config_packet	*reset_device*	*device_serial*			
0xFF	0xFF	type: uint32 format: hex			

Message sample

Message goal: Restore factory configuration

Select *clear_config_packet* HEX code

```
0xFF
```

Select *reset_device* HEX code

```
0xFF
```

device_serial

```
5083000D
```

message flip for LSB

```
0x0D008350
```

Compile message for sending (HEX)

```
FFFF0D008350
```

Control value in base64 to control after sending

```
//8NAINQ
```

fPort 51 - OTA Activation Message

Byte 0	Comment
activate_ota	
0xFF	Activates OTA for 2 minutes

Message sample

Message goal: Enable OTA mode

Header

Select Header HEX code

```
FF
```

Compile message for sending (HEX)

```
FF
```

Control value in base64 to control after sending

```
/w==
```

fPort 60 - Command Message

Query detailed DALI status

Byte 0	Byte 1
dali_status_request	dali_address_short*
0x00	type: uint8 value: value_masked

* Value 254(0xFE) Queries all the automatically detected ballasts and sends the statuses as an array.

Message sample

Message goal: Get status of all connected DALI devices

Select *dali_status_request* HEX code

```
0x00
```

dali_address_short

Convert `Broadcast` address to HEX

```
0xFE
```

Compile message for sending (HEX)

```
00FE
```

Control value in base64 to control after sending

```
AP4=
```

Response:

Byte 0	Byte 1		Byte 2		Byte..		Byte n
dali_status_request	1st device				.. device		
	dali_address_short		*dali_status*		*dali_address_short*		*dali_status*
0x00	type: uint8 value: value_masked		See fig. 1		type: uint8 value: value_masked		See fig. 1

1.

Bit #	Parameter	Value
0	*control_gear_failure*	0: false 1: true
1	*lamp_failure*	
2	*lamp_on*	
3	*limit_error*	
4	*fade_running*	
5	*reset_state*	
6	*short_address*	
7	*power_cycle_seen*	

Message sample

Message in base64

```
AAIEBgIMAg==
```

Message decoded to hex

```
00|02|04|06|02|0C|02
```

Header decoded

```
Answer for detailed DALI status request
```

1st Device status

dali_address_short in HEX

```
0x02
```

converted to device

```
Single device 1
```

dali_status in HEX

```
0x04
```

HEX message converted to binary

```
0B00000100
```

Binary converted to statuses (LSB)

```
0 : control_gear_failure - false
0 : lamp_failure - false
1 : lamp_on - true
0 : limit_error - false
0 : fade_running - false
0 : reset_state - false
0 : short_address - false
0 : power_cycle_seen - false
```

2nd Device status

dali_address_short in HEX

```
0x06
```

converted to device

```
Single device 3
```

dali_status in HEX

```
0x02
```

HEX message converted to binary

```
0B00000010
```

Binary converted to statuses (LSB)

```
0 : control_gear_failure - false
1 : lamp_failure - true
0 : lamp_on - false
0 : limit_error - false
0 : fade_running - false
0 : reset_state - false
0 : short_address - false
0 : power_cycle_seen - false
```

3rd Device status

dali_address_short in HEX

```
0x0C
```

converted to device

```
Single device 6
```

dali_status in HEX

```
0x02
```

HEX message converted to binary

```
0B00000010
```

Binary converted to statuses (LSB)

```
0 : control_gear_failure - false
1 : lamp_failure - true
0 : lamp_on - false
0 : limit_error - false
0 : fade_running - false
0 : reset_state - false
0 : short_address - false
0 : power_cycle_seen - false
```

Set Dimming level

This command is the “MANUAL” dim source in luminaire control chart. Please see appendix 1 for more information.

Byte 0	Byte 1	Byte 2	Byte..	Byte n
dimming_command	*dali_address_short*	*dim_level*	*dali_address_short.*	*dim_level*
0x01	type: uint8 value: value_masked	type: uint8 0..100 unit: % 0xFF - Resume*	type: uint8 value: value_masked	type: uint8 0..100 unit: % 0xFF - Resume*

* Will tell the controller to resume, to the state it was before overriding the dimming with manual commands.

Message sample

Message goal: Set all luminaires to 100%

Select *dimming_command* HEX code

```
0x01
```

dali_address_short

Convert `Broadcast` to HEX

```
0xFE
```

dim_level

Convert `100%` to HEX

```
0x64
```

Compile message for sending (HEX)

```
01FE64
```

Control value in base64 to control after sending

```
Af5k
```

Custom DALI request

Byte 0	Byte 1	Byte 2	Byte..	Byte n
custom_dali_request	dali_address_short	dali_query	dali_address_short	dali_query
0x03	type: uint8 value: value_masked	type: uint8	type: uint8 value: value_masked	type: uint8

Message sample

Message goal: Ask Max level, Min Level, Power on level, System failure level and Fade time/rate from device 36

Select custom_dali_request HEX code

```
0x03
```

1st Query

dali_address_short

Convert `Single device 36` to HEX

```
0x48
```

Convert Max level query code `161` to HEX

```
0xA1
```

2nd Query

dali_address_short

Convert `Single device 36` to HEX

```
0x48
```

Convert Min level query code `162` to HEX

```
0xA2
```

3rd Query

dali_address_short

Convert `Single device 36` to HEX

```
0x48
```

Convert Power on level query code `163` to HEX

```
0xA3
```

4th Query

dali_address_short

Convert `Single device 36` to HEX

```
0x48
```

Convert System failure level query code `164` to HEX

```
0xA4
```

5th Query

dali_address_short

Convert `Single device 36` to HEX

```
0x48
```

Convert Fade time/rate query code `165` to HEX

```
0xA5
```

Compile message for sending (HEX)

```
03|48A1|48A2|48A3|48A4|48A5
```

Control value in base64 to control after sending

```
A0ihSKJIo0ikSKU=
```

Response:

Byte 0	Byte 1	Byte 2	Byte 3	Byte..	Byte..	Byte n
custom_dali_request	*dali_address_short*	*dali_query*	*dali_answer*	*dali_address_short*	*dali_query*	*dali_answer*
0x03	type: uint8 value: masked	type: uint8	type: uint8	type: uint8 value: masked	type: uint8	type: uint8

Message sample

Message in base64

```
BEih/kiiqEij/kik/kilBw==
```

Message decoded to hex

```
04|48A1FE|48A2A8|48A3FE|48A4FE|48A507
```

Header `04` decoded

```
Answer for custom DALI request
```

1st Answer

dali_address_short in HEX

0x48

converted to device

Single device 36

dali_query in HEX

0xA1

HEX value converted to decimal

161

Decimal value translated to query

Max level

dali_answer in HEX

0xFE

HEX value converted to decimal

254

2nd Answer

dali_address_short in HEX

0x48

converted to device

Single device 36

dali_query in HEX

0xA2

HEX value converted to decimal

162

Decimal value translated to query

Min level

dali_answer in HEX

0xA8

HEX value converted to decimal

168

3rd Answer

dali_address_short in HEX

0x48

converted to device

Single device 36

dali_query in HEX

0xA3

HEX value converted to decimal

163

Decimal value translated to query

Power on level

dali_answer in HEX

0xFE

HEX value converted to decimal

254

4th Answer

dali_address_short in HEX

0x48

converted to device

Single device 36

dali_query in HEX

0xA4

HEX value converted to decimal

164

Decimal value translated to query

Failure level

dali_answer in HEX

0xFE

HEX value converted to decimal

254

5th Answer

dali_address_short in HEX

0x48

converted to device

Single device 36

dali_query in HEX

0xA5

HEX value converted to decimal

165

Decimal value translated to query

Fade time/rate

dali_answer in HEX

0x07

HEX value converted to status

<0.7s / 45 steps/s

Custom DALI command

Byte 0	Byte 1	Byte n
custom_dali_command	*dali_command*	
0x04		

Message sample

Message goal: Set Single device 1 minimum light level to 127.

Select *custom_dali_command* HEX code

```
0x04
```

dali_command in HEX

```
027F0321032B
```

Compile message for sending (HEX)

```
04027F0321032B
```

Control value in base64 to control after sending

```
BAJ/AyEDKw==
```

Request status

Byte 0	Byte 1
request_status	*functions*
0x05	See fig. 1

1.

Bit #	Parameter	Value
0	*usage*	0 - *not requested* 1 - *requested*
1	*status*	
2	RFU	
3	RFU	
4	RFU	
5	RFU	
6	RFU	
7	RFU	

Message sample

Message goal: Request usage message.

Select *request_status* HEX code

```
0x05
```

function selection

```
1 : usage - requested
0 : status - not requested
0 : RFU
0 : RFU
0 : RFU
0 : RFU
0 : RFU
0 : RFU
```

Selection converted to binary

```
0B00000001
```

Selection converted to HEX

```
0x01
```

Compile message for sending (HEX)

```
0501
```

Control value in base64 to control after sending

```
BQE=
```

Request interface values

Byte 0	Byte 1
request_interfaces	*interfaces*
0x06	0xFF - all interfaces

Message sample

Message goal: Request all interface values.

Select *request_interfaces* HEX code

```
0x06
```

interface selection in HEX

```
0xFF
```

Compile message for sending (HEX)

```
06FF
```

Control value in base64 to control after sending

```
Bv8=
```

Response:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
request_inter-faces	*dig*	*value*	*ldr*	*value*	*thr*	*value*	*relay*	*value*
0x06	0x01	0x00 - off 0x01 - on 0xFF = N/A	0x02	type: uint8 0..254 0xFF = N/A	0x03	0xFF - n/a	0x04	See fig 1.

1.

Bit #	Parameter	Value
0	*main relay*	0: open 1: closed
1	*od_relay*	
2	*RFU*	
3	*RFU*	
4	*RFU*	
5	*RFU*	
6	*RFU*	
7	*RFU*	

Message sample

Message in base64

```
BgEAAkUD/wQA
```

Message decoded to hex

```
06|0100|0245|03FF|0400
```

Header `06` decoded

```
Answer for request interface values
```

`1st` interface HEX value 01 converted to value

```
dig
```

value in HEX

```
0x00
```

HEX value converted to state

```
off
```

`2nd` interface HEX value 02 converted to value

```
ldr
```

value in HEX

```
0x45
```

HEX value converted to state

```
69
```

`3rd` interface HEX value 03 converted to value

```
thr
```

value in HEX

```
0xFF
```

HEX value converted to state

```
n/a
```

`4th` interface HEX value 04 converted to value

```
relay
```

value in HEX

```
0x00
```

HEX value converted to state

```
off
```

Read from driver memory

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
read_memory	*dali_address_short*	*memory_bank*	*memory_address*	*read_size*
0x07	type: uint8 value: value_masked	type: uint8	type: uint8	type: uint8*

* Consider the answer length when choosing the amount of bytes to read.

Message sample

Message goal is to read GTIN from driver/ballast (address 0x4(shifted 2)), memory bank 0x0 address 0x3.
6 bytes.

Compile message

```
07|04|00|03|06
```

Response:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte ..	Byte n
read_memory	*dali_address_short*	*memory_bank*	*memory_address*	*read_size*	*memory_value*	
0x07	type: uint8 value: masked	type: uint8	type: uint8	type: uint8	type: HEX	

Message response sample

```
0704000306|07edface82e5
```

Header from request GTIN = 8718696481509

Write to driver memory*

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4..n
write_memory	*dali_address_short*	*memory_bank*	*memory_address*	*memory_value*
0x08	type: uint8 value: value_masked	type: uint8	type: uint8	type: uint8*

* If the operation fails, only the header is returned, with no contents.

Timed Dimming

This command is the “MANUAL” dim source in luminaire control chart. Please see appendix 1 for more information.

Byte 0	Byte 1	Byte 2	Byte 3	Byte..	Byte ..	Byte n
timed_dimming_ command	*dali_address_short*	*dim_level*	*duration*	*dali_address_short.*	*dim_level*	*duration*
0x09	type: uint8 value: masked	type: uint8 0..100 unit: %	type: uint8 unit: minutes	type: uint8 value: masked	type: uint8 0..100 unit: %	type: uint8 unit: minutes

Message sample

Message goal: Set all luminaires to 100% for 15 minutes

Select *timed_dimming_command* HEX code

```
0x09
```

dali_address_short

Convert `Broadcast` to HEX

```
0xFE
```

dim_level

Convert `100%` to HEX

```
0x64
```

duration

Convert `15` to HEX

```
0x0F
```

Compile message for sending (HEX)

```
09fe640f
```

Control value in base64 to control after sending

```
Cf5kDw==
```

fPort 61 - Event Notification Message

General packet structure

Byte 0	Byte 1	Byte ..	Byte n
packet_type	*parameters*	*payload*	
type: HEX	See fig. 1	type: packet specific	

1.

Bit #	Parameter	Value
0	RFU	
1	RFU	
2	RFU	
3	RFU	
4	*length*	*0..15*
5		
6		
7		

Digital interface alert

Byte 0	Byte 1	Byte 2	Byte 3
dig_alert	*parameters*	*counter*	
0x80	0x20	*uint16*	

Message sample

Message in base64

```
gAA=
```

Message decoded to HEX

```
80200600
```

Byte 2-3 flip LSB

```
0006 - event happened 6 times
```

LDR interface alert

Byte 0	Byte 1	Byte 2	Byte 3
ldr_alert	*parameters*	*state*	*ldr_value*
0x81	0x20	0x00 - off 0x01 - on	type: uint8

Message sample

Message in base64

```
gSAAeQ==
```

Message decoded to HEX

```
81|20|00|79
```

Header value 0x81 converted to packet type

```
ldr_alert
```

parameters in HEX

```
0x20
```

converted to binary

```
0B00100000
```

Binary converted to statuses (LSB)

```
0 : RFU
0 : RFU
0 : RFU
0 : RFU
0 : length -  [0B0010]
1 :
0 :
0 :
```

Payload length binary value converted to HEX

```
0x02
```

converted to value

```
2 (bytes)
```

state value 0x00 converted to state

```
off
```

ldr_value in HEX

```
0x79
```

converted to decimal

```
121
```

DALI driver alert

Byte 0	Byte 1	Byte 2	Byte 3		
dali_driver_status	*parameters*	*dali_address_short*	*dali_status*	*dali_address_short*	*dali_status*
0x83	0x20..0xE0*	type: uint8 value: value_masked	See fig. 1	type: uint8 value: value_masked	See fig. 1

1.

Bit #	Parameter	Value
0	*control_gear_failure*	0: ok 1: alert
1	*lamp_failure*	
2	*lamp_on***	
3	*limit_error***	
4	*fade_running***	
5	*reset_state***	
6	*short_address***	
7	*power_cycle_seen***	

* Depending of the count of included statuses. Maximum of 7 status messages can be sent at the same time
** Do not trigger alert.

Message sample

Message in base64

```
gSAAeQ==
```

Message decoded to HEX

```
83|20|02|02
```

Header value 0x83 converted to packet type

```
dali_driver_status
```

parameters in HEX

```
0x20
```

converted to binary

```
0B00100000
```

Binary converted to statuses (LSB)

```
0 : RFU
0 : RFU
0 : RFU
0 : RFU
0 : length -  [0B0010]
1 :
0 :
0 :
```

Payload length binary value converted to HEX

```
0x02
```

converted to value

```
2 (bytes)
```

dali_address_short value 0x02 converted to device

```
Single device 1
```

dali_status in HEX

```
0x02
```

converted to binary

```
0B00000010
```

Binary converted to statuses (LSB)

```
0 : control_gear_failure - ok
1 : lamp_failure - alert
0 : lamp_on - ok
0 : limit_error - ok
0 : fade_running - ok
0 : reset_state - ok
0 : short_address - ok
0 : power_cycle_seen - ok
```

fPort 99 - System Message

Boot message

Byte 0	Byte 1.. Byte 4	Byte 5..Byte 7	Byte 8..Byte 11	Byte 12	Byte 13	Byte 14	Byte 15	Byte 16
packet_type	*device_serial*	*firmware_version*	*device_unix_epoch*	*device_config*	*optional_features*	*dali_info*	*driver_info*	*reset_reason*
0x00 boot_packet	type: uint32 format: hex	format: hex	type: uint32 Seconds elapsed since Jan 1, 1970 (midnight UTC/GMT)	See fig.1	See fig. 2	See fig. 3	See fig. 4	See fig. 5

1.

Bit #	Config	Desc.
00	DD	dali
01	DC	dali_nc
02	*DO*	dali_no
03	*AC*	analog_nc
04	*AO*	analog_no
05	*UC*	dali_analog_nc
06	*UO*	dali_analog_no
07	*UU*	dali_analog_nc_no

2.

Bit #	Parameter	Value
0	RFU	0: not present 1: present
1	RFU	
2	*dig*	
3	*ldr*	
4	open_drain	
5	metering	
6	RFU	
7	*custom_request*	

3.

Bit #	Parameter	Value
0	*bus_supply*	0x00..0x6F type: uint unit: V value: scaled
1		
2		
3		0x70..0x7F - error codes 0x70..0x7D - RFU 0x7E - *bus_high* 0x7F - *dali_error* (int DALI supply error / DALI shorted / DALI overloaded)
4		
5		
6		
7	*bus_power*	0 - internal 1 - external

4.

Bit #	Parameter	Value
0	*device_count* (with short addresses assigned)	type: uint7
1		
2		
3		
4		
5		
6		
7	*unaddressed_devices*	0 - not found 1 - found

5.

Bit #	Parameter	Value
0	RFU	0: false 1: true
1	*watchdog_reset*	
2	*soft_reset* (e.g. from DFU)	
3	RFU	
4	RFU	
5	RFU	
6	RFU	
7	*RFU*	

Message sample

Message in base64

```
AA0Ag1ABAQHzfyBeAAz+AQQ=
```

Message decoded to hex

```
00|0D008350|010101|F37F205E|00|0C|FE|01|04
```

Header `00` decoded

```
boot_packet
```

device_serial `0D008350` message flip for MSB

```
0x5083000D
```

firmware_version

major version in HEX

```
0x01
```

HEX value converted to decimal

```
1
```

minor version in HEX

```
0x01
```

HEX value converted to decimal

```
1
```

patch version in HEX

```
0x01
```

HEX value converted to decimal

```
1
```

device_unix_epoch `F37F205E` message flip for MSB

```
0x5E207FF3
```

HEX message converted to decimal (epoch)

```
1579188211 (seconds)
```

```
Epoch time converted to date
```

```
January 16, 2020 15:23:31 (UTC)
```

device_config configuration

Hardware setup in HEX

```
00
```

HEX value converted to setup

```
DALI only
```

optional_features in HEX

```
0x0C
```

converted to binary

```
0B00001100
```

Cells marked with X contain address bits

Binary converted to config

```
0 : RFU
0 : RFU
1 : dig - present
1 : ldr - present
0 : RFU
0 : RFU
0 : RFU
0 : custom_request - not present
```

driver_info in HEX

```
0x01
```

converted to binary

```
0B00000001
```

Binary converted to statuses (LSB)

```
1 : device_count - [0B0000001]
0 :
0 :
0 :
0 :
0 :
0 :
0 : unadressed_devices - not found
```

device_count binary value converted to HEX

```
0x01
```

converted to value

```
1 (addressed device)
```

optional_features in HEX

```
0x04
```

converted to binary

```
0B00000100
```

Binary converted to config

```
0 : RFU
0 : watchdog_reset - false
1 : soft_reset - true
0 : RFU
0 : RFU
0 : RFU
0 : RFU
0 : RFU
```

Downlink failed message

Byte 0	Byte 1	Byte 2
packet_type	*packet_from_fport*	*parse_error_code*
0x13 config_failed_packet	type: uint8	See fig. 1

1.

Value	Error	C
2	unknown_fport	
3	packet_size_short	
4	packet_size_long	
5	value_error	
6	protocol_parse_error	
7	reserved_flag_set	
8	invalid_flag_combination	
9	unavailable_feature_request	
10	unsupported_header	
11	unreachable_hw_request	
12	address_not_available	
13	internal_error	
14	packet_size_error	
128	no_room	profile conf. errors
129	id_seq	
130	dest	
131	days	
132	step_count	
133	step_value	
134	step_unsorted	
135	days_overlap	

Message sample

Message in base64

```
EzIE
```

Message decoded to hex

```
133204
```

Header `13` decoded

config_failed_packet

packet_from_fport in HEX

`0x32`

HEX value converted to decimal

50

parse_error_code in HEX

`0x04`

HEX value converted to decimal

4

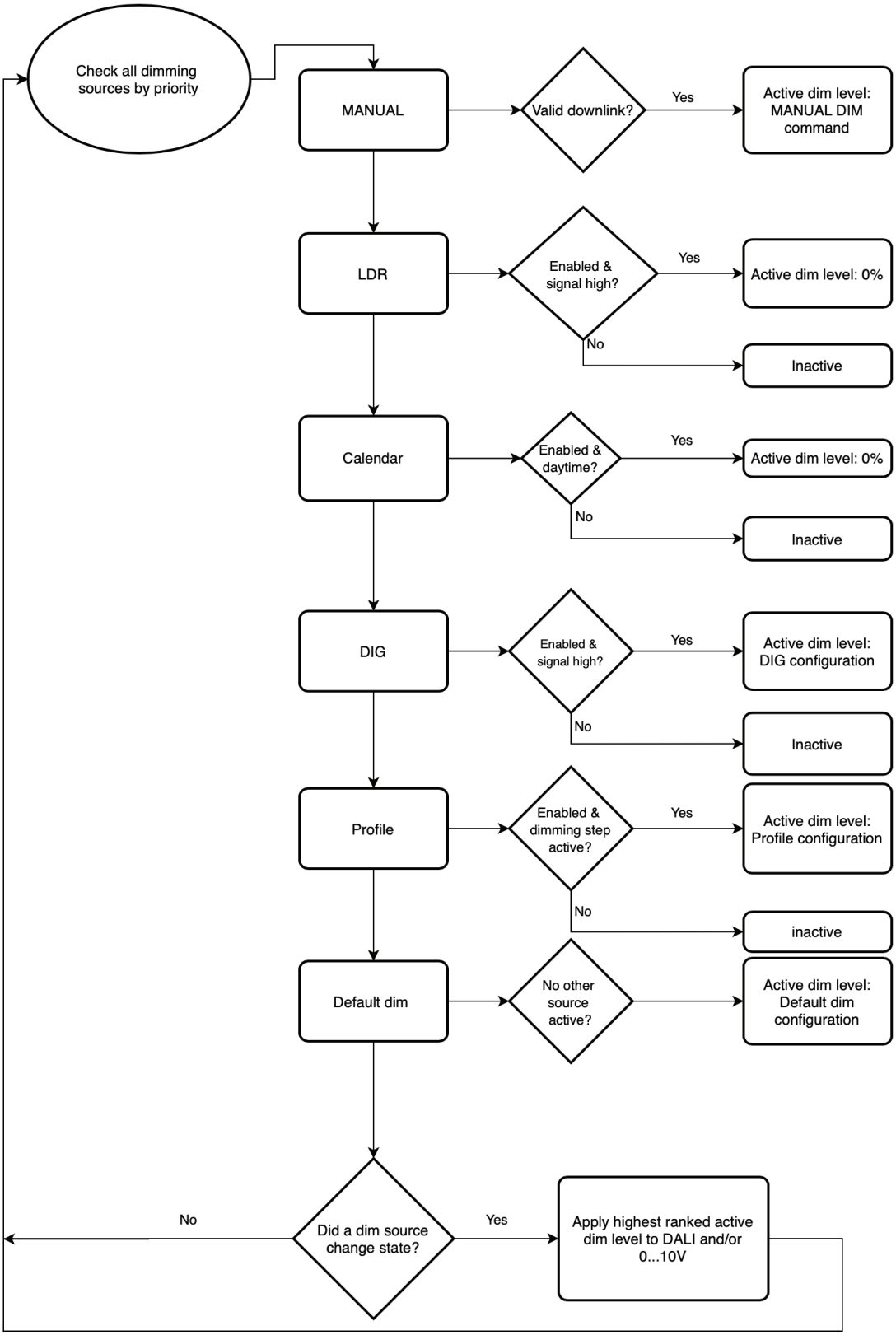
converted to error

packet_size_long

Error message

Byte 0
packet_type
0x14 Could not send full payload

Appendix 1. Luminaire control chart.



DALI Address / Destination mapping

Bit #	Single address	Group address	Broadcast
Addr.	0-63	0-15	n/a
0	0	0	0
1	x	x	1
2	x	x	1
3	x	x	1
4	x	x	1
5	x	0	1
6	x	0	1
7	0	1	1

Broadcast (0xFE) also controls the analog/0..10V interface

Profile step time mapping UTC.

0 - 00:00
1 - 00:10
2 - 00:20
3 - 00:30
4 - 00:40
5 - 00:50
6 - 01:00
7 - 01:10
8 - 01:20
9 - 01:30
10 - 01:40
11 - 01:50
12 - 02:00
13 - 02:10
14 - 02:20
15 - 02:30
16 - 02:40
17 - 02:50
18 - 03:00
19 - 03:10
20 - 03:20
21 - 03:30
22 - 03:40
23 - 03:50
24 - 04:00
25 - 04:10
26 - 04:20
27 - 04:30
28 - 04:40
29 - 04:50
30 - 05:00
31 - 05:10
32 - 05:20
33 - 05:30
34 - 05:40
35 - 05:50
36 - 06:00
37 - 06:10
38 - 06:20
39 - 06:30
40 - 06:40
41 - 06:50
42 - 07:00
43 - 07:10
44 - 07:20
45 - 07:30
46 - 07:40
47 - 07:50
48 - 08:00
49 - 08:10
50 - 08:20
51 - 08:30
52 - 08:40
53 - 08:50
54 - 09:00
55 - 09:10
56 - 09:20
57 - 09:30
58 - 09:40
59 - 09:50
60 - 10:00
61 - 10:10
62 - 10:20
63 - 10:30
64 - 10:40
65 - 10:50
66 - 11:00
67 - 11:10
68 - 11:20
69 - 11:30
70 - 11:40
71 - 11:50
72 - 12:00
73 - 12:10
74 - 12:20
75 - 12:30
76 - 12:40
77 - 12:50
78 - 13:00
79 - 13:10
80 - 13:20
81 - 13:30
82 - 13:40
83 - 13:50
84 - 14:00
85 - 14:10
86 - 14:20
87 - 14:30
88 - 14:40
89 - 14:50
90 - 15:00
91 - 15:10
92 - 15:20
93 - 15:30
94 - 15:40
95 - 15:50
96 - 16:00
97 - 16:10
98 - 16:20
99 - 16:30
100 - 16:40
101 - 16:50
102 - 17:00
103 - 17:10
104 - 17:20
105 - 17:30
106 - 17:40
107 - 17:50
108 - 18:00
109 - 18:10
110 - 18:20
111 - 18:30
112 - 18:40
113 - 18:50
114 - 19:00
115 - 19:10
116 - 19:20
117 - 19:30
118 - 19:40
119 - 19:50
120 - 20:00
121 - 20:10
122 - 20:20
123 - 20:30
124 - 20:40
125 - 20:50
126 - 21:00
127 - 21:10
128 - 21:20
129 - 21:30
130 - 21:40
131 - 21:50
132 - 22:00
133 - 22:10
134 - 22:20
135 - 22:30
136 - 22:40
137 - 22:50
138 - 23:00
139 - 23:10
140 - 23:20
141 - 23:30
142 - 23:40
143 - 23:50

CONTACT INFORMATION

Nordic Automation Systems AS

www.nasys.no

info@nasys.no

REVISION HISTORY

1.0 - First draft

1.1 - Updated usage message and usage configuration (fw 0.6.20)

1.2 - fPort nubers corrected

1.3 - Calendar user configurable

1.4 - Added Multicast configuration

2.0 - Added support for 1.0.0 firmware.

2.1 - Optimised packets. Legacy packets merged with new config. Open-source license info added.

2.2 - Fixed support for 1.0.0 firmware

2.3 - Added luminaire control priority chart, command fixes.

2.4 - Improvements and fixes.

2.4.1 - added samples and fixes.

2.4.2 - added more fixes.

Open-source Licenses used by NAS Products

ArduinoJson

The MIT License (MIT)
Copyright © 2014-2019 Benoit BLANCHON
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

LoRaMac-node

--- Revised BSD License ---
Copyright (c) 2013, SEMTECH S.A.
All rights reserved.
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
• Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
• Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
• Neither the name of the Semtech corporation nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL SEMTECH S.A. BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

nRF5 SDK

Copyright (c) 2010 - 2019, Nordic Semiconductor ASA
All rights reserved.
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form, except as embedded into a Nordic Semiconductor ASA integrated circuit in a product or a software update for such product, must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of Nordic Semiconductor ASA nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.
4. This software, with or without modification, must only be used with a Nordic Semiconductor ASA integrated circuit.
5. Any software provided in binary form under this license must not be reverse engineered, decompiled, modified and/or disassembled.
THIS SOFTWARE IS PROVIDED BY NORDIC SEMICONDUCTOR ASA "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
OF MERCHANTABILITY, NONINFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL NORDIC SEMICONDUCTOR ASA OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Tiny AES in C

This is free and unencumbered software released into the public domain.
Anyone is free to copy, modify, publish, use, compile, sell, or distribute this software, either in source code form or as a compiled binary, for any purpose, commercial or non-commercial, and by any means.
In jurisdictions that recognize copyright laws, the author or authors of this software dedicate any and all copyright interest in the software to the public domain. We make this dedication for the benefit of the public at large and to the detriment of our heirs and
successors. We intend this dedication to be an overt act of relinquishment in perpetuity of all present and future rights to this software under copyright law.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
For more information, please refer to Unlicence.org

Eclipse

Embedded Template Library