

# 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

UL2033\*#Xv

UL2023\*#Xv

# INDEX

fPort	Function	Transmission	Page
24	Status	↑↑*	3
	Legacy status		4
25	Usage	↑↑	9
49	Configuration request	↓↓**	12
	LDR		12
	DIG		13
	Calendar		14
	Status		15
	Profile		16
	Default dim		17
	Usage		18
	Holiday		19
	Boot delay		20
	Defaults		21
	META position		22
	Multicast		23
50	Configuration	↑↓***	24
	LDR		25
	DIG		26
	Calendar		28
	Status		30
	Profile		31
	Time		34
	Default dim		25
	Usage		37
	Holiday		38
	Boot delay		40
	Defaults		41
	META position		43
	Multicast		45
	Clear config.		50
	Factory reset		51
51	Update mode	↓↓	52
60	Command	↑↓	53
	DALI status		53
	Dimming		57
	Custom DALI request		58
	Custom DALI command		63
	Request status		64
	Request interfaces		66
	Read driver memory		69
	Write driver memory		70
	Timed dimming		71
61	Event notification	↑↑	72
	DIG alert		73
	LDR alert		74
	DALI alert		75
99	System packets	↑↑	77
	Boot		77
	Config. failed		80

\* ↑↑ - 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 ..
<i>device_unix_epoch</i>				<i>status_field</i>	<i>down-link_rssi</i>	<i>down-link_snr</i>	<i>temperature</i>	<i>analog_interfaces</i>	<i>thr*</i>	<i>ldr*</i>	<i>profile_1</i>					<i>profile_n**</i>
type: uint32 Number of seconds that have elapsed since January 1, 1970 (midnight UTC/GMT)				See fig. 1	type: uint8 unit: dBm convert: *-1	type: int8 unit: dB	type: int8 unit: °C	See fig. 2	type: uint8	type: uint8	<i>profile_id</i>	<i>profile_version</i>	<i>dali_address_short</i>	<i>days_active</i>	<i>dim_level</i>	See <i>profile_1</i>
											type: uint8 0..254 255 - no_profile	type: uint8 0..240 ***	type: uint8 value - masked ****	See fig. 3	type: uint8 0..100 unit: %	

1.

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

2.

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

3.

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

4.

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

\* 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”

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 11	Byte 12	Byte 13	Byte 14	Byte 15	Byte ..
<i>device_unix_epoch</i>				<i>status_field</i>	<i>down-link_rssi</i>	<i>profile_1</i>					<i>profile_n**</i>
type: uint32 Number of seconds that have elapsed since January 1, 1970 (midnight UTC/GMT)				See fig. 1	type: uint8 unit: dBm convert: *-1	<i>profile_id</i> type: uint8 0..254 255 - no_profile	<i>profile_version</i> type: uint8 0..240 ***	<i>dali_address_short</i> type: uint8 value - masked ****	<i>days_active</i> See fig. 3	<i>dim_level</i> type: uint8 0..100 unit: %	See <i>profile_1</i>

## Message sample

### Message in base64

```
39QdXgBLBBUCrgUFCv8yAwMG/wA=
```

### Message decoded to HEX

```
DFD41D5E0004B041502AE05050AFF32030306FF00
```

*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

0<sub>B</sub>00000010

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

0<sub>B</sub>11111111

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	<i>active_energy_total</i>	uint32, Wh
1	<i>active_energy_instant</i>	uint16, W
2	<i>load_side_energy_total</i>	uint32, Wh
3	<i>load_side_energy_instant</i>	uint16, W
4	<i>power_factor_instant</i>	uint8 ***
5	<i>system_voltage</i>	uint8, V
6	<i>driver_operating_time</i>	uint32, sec
7	<i>lamp_on_time</i>	uint32, sec

\* If driver supports. Also drivers need to have short addresses assigned.

\*\* 0xFF values measured by the controller.

\*\*\* Needs to be divided 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

```
BAMAAAAAAAAAGA xUUAAAAAA==
```

### 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

0<sub>B</sub>00000011

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	<i>ldr_config_request</i>

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	<i>dig_config_request</i>

### 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	<i>calendar_config_request</i>

## 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	<i>status_config_request</i>

## 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
<i>profile_config_request</i>	<i>profile_id</i>
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	<i>default_dim_config_request</i>

## 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	<i>usage_config_request</i>

## 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	<i>holiday_config_request</i>

## 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	<i>boot_delay_config_request</i>

## 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	<i>defaults_config_request</i>

## 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	<i>meta_pos_config_request</i>

## 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
<i>multicast_config_request</i>	<i>multicast_device</i>
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
<i>ldr_config_packet</i>	<i>switch_thresholds</i>		<i>switch_behaviour</i>
	<i>high*</i> (switch off / clear alert)	<i>low*</i> (switch on / create alert)	
0x01	type: uint8 0xFF - disabled	type: uint8 0xFF - disabled	See fig. 1

\*High and low are raw ADC values.

1.

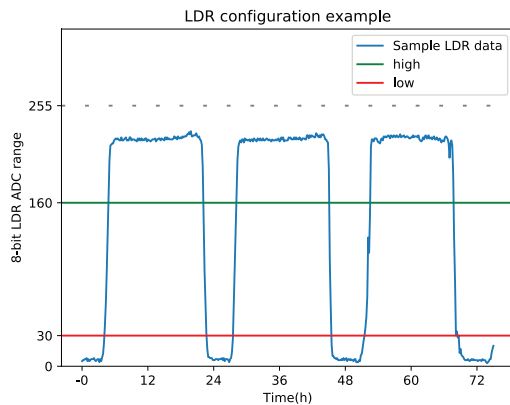
Bit #	Function	Value
0	RFU	
1	RFU	
2	<i>trigger_alert</i>	0: false 1: true
3	RFU	
4	RFU	
5	RFU	
6	RFU	
7	RFU	

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



## Message sample

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



This chart visualizes 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)

01A03004

Control value in base64 to control after sending

AaAwBA==

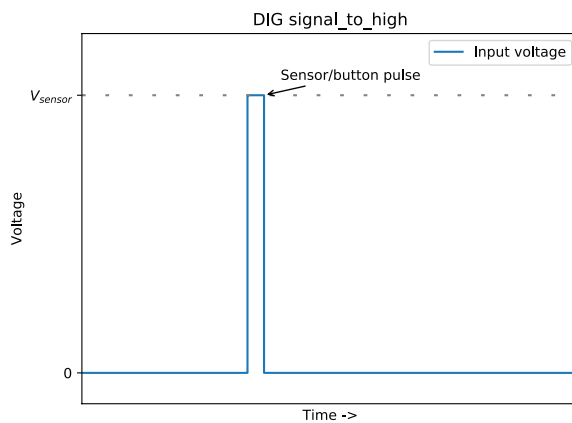
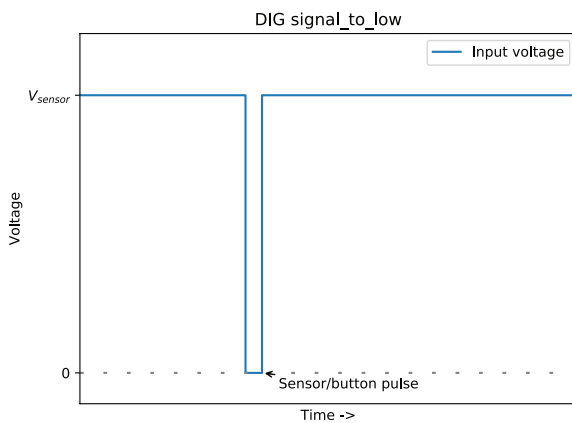
## DIG configuration packet

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
<i>dig_config_packet</i>	<i>switch_time</i>		<i>behaviour</i>	<i>dali_address_short</i>	<i>dim_level</i>
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	<i>switch_point*</i>	0: signal_to_low 1: signal_to_high
2	<i>trigger_alert</i>	0: false 1: true
3	RFU	
4	RFU	
5	RFU	
6	RFU	
7	RFU	

Following charts show typical *switch\_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)

032C0102FE32

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
<i>calendar_config_packet</i>	<i>sunrise_offset</i>	<i>sunset_offset</i>	<i>latitude</i>		<i>longitude</i>	
0x06*	type: int8 unit: minutes	type: int8 unit: minutes	type: int16 unit: degrees (2 decimals * 100)**		type: int16 unit: degrees (2 decimals * 100)**	

\*0x06FFFFFFFF 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

BuIe1hmzCQ==

## Status reporting configuration

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
<i>status_config_packet</i>	<i>status_interval</i>			
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)

07|100E0000

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_con- fig_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.

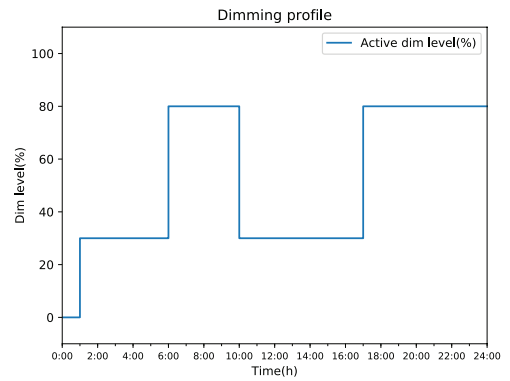
Bit #	Parameter	Value
0	<i>holiday</i>	0 - not active 1 - active
1	<i>mon</i>	
2	<i>tue</i>	
3	<i>wed</i>	
4	<i>thu</i>	
5	<i>fri</i>	
6	<i>sat</i>	
7	<i>sun</i>	

\* Maximum of 10 steps are allowed. Steps need to be in chronological 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 address 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 address 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

0E00011110

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|06|1E|2450|3C|1E|6650
```

Control value in base64 to control after sending

```
CBYD/h4AAAYeJFA8HmZQ
```

## Time configuration packet\*

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
<i>time_config_packet</i>	<i>device_unix_epoch</i>			
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)

09|681A9C59

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
<i>default_dim_config_packet</i>	<i>default_dim</i>	<i>RFU</i>
0x0A	unit: % type: uint8 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)

0A|00|00

Control value in base64 to control after sending

CgAA

## Usage reporting configuration\*

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
<i>usage_config_packet</i>	<i>usage_interval</i>			<i>system_voltage</i>	
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

CxAOAAADm

## Holiday configuration packet

This setting affects Holiday bit in Profile configuration packet.

Byte 0	Byte 1	Byte 2	Byte ..	Byte ..
<i>holiday_config_packet</i>	<i>holiday_1</i>		<i>holiday_n*</i>	
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 1<sup>st</sup>, February 24<sup>th</sup>, May 1<sup>st</sup>, June 23<sup>rd</sup>, August 20<sup>th</sup> and December 23<sup>rd</sup> as holidays.

Select Header HEX code

0C

*holiday\_1*

Choose desired day

January 1<sup>st</sup>

Convert to day number in year

1

Convert value to HEX

0x0001

Flip value for LSB

0100

*holiday\_2*

Choose desired day

February 24<sup>th</sup>

Convert to day number in year

55

Convert value to HEX

0x0037

Flip value for LSB

3700

*holiday\_3*

Choose desired day

May 1<sup>st</sup>

Convert to day number in year

121

Convert value to HEX

0x0079

Flip value for LSB

7900

*holiday\_4*

Choose desired day

June 23<sup>rd</sup>

Convert to day number in year

174

Convert value to HEX

0x00AE

Flip value for LSB

AE00

*holiday\_5*

Choose desired day

August 20<sup>th</sup>

Convert to day number in year

234

Convert value to HEX

0x00E2

Flip value for LSB

E200

*holiday\_6*

Choose desired day

December 23<sup>rd</sup>

Convert to day number in year

357

Convert value to HEX

0x0165

Flip value for LSB

6501

Compile message for sending (HEX)

0c010037007900AE00E2006501

Control value in base64 to control after sending

DAEANwB5AK4A4gB1AQ==

## Boot delay configuration packet

Byte 0	Byte 1
<i>boot_delay_config_packet</i>	<i>boot_delay_range</i>
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 configuraion packet

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

\*Same as default dim configuraion packet

1.

Bit #	Function	Value
0	<i>default_dim</i>	0: not present in packet 1: configured
1	RFU	
2	<i>fade</i>	
3	RFU	
4	<i>legacy_mode**</i>	
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 seconds 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
<i>packet_type</i>	<i>configured_parameters</i>	<i>gps_position</i>		<i>address</i>	
		<i>latitude</i>	<i>longitude</i>	<i>length</i>	<i>data[0]..data[length-1]</i>
0x10 <i>meta_pos_config_packet</i>	See fig. 1	type: int32 value: *10 <sup>7</sup> when: <i>gps_position</i> = <i>configured</i>  0x7FFFFFFF - not_configured	type: int32 value: *10 <sup>7</sup> when: <i>gps_position</i> = <i>configured</i>  0x7FFFFFFF - not_configured	type: uint8 unit: B when: <i>address</i> = <i>configured</i>  max=38	type: char array (utf-8) when: <i>address</i> = <i>configured</i>

1.

Bit #	Parameter	Value
0	<i>gps_position</i>	0 - <i>not_configured</i> 1 - <i>configured</i>
1	<i>address</i>	
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 multiplied 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 multiplied 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

EAF2wjhPiyDg==

## Multicast configuration packet

Byte 0	Byte 1	Byte 2..Byte 5	Byte 6..Byte 21	Byte 22..Byte 37
<i>multicast_config_packet</i>	<i>multicast_device</i>	<i>DevAddr</i>	<i>NwkSKey</i>	<i>AppSKey</i>
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
<i>clear_config_packet</i>	<i>ldr_config</i>
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)
<code>clear_config_packet</code>	<code>profile_config</code>	<code>dali_address_short</code>	<code>profile_id</code>
0xFF	0x04	type: uint8 value: masked 0xFF - <i>all_profiles</i>	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
<code>clear_config_packet</code>	<code>holiday_config</code>
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
<code>clear_config_packet</code>	<code>multicast_config</code>	<code>multicast_device</code>
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
<i>dali_status_request</i>	1 <sup>st</sup> device		.. device	
	<i>dali_address_short</i>	<i>dali_status</i>	<i>dali_address_short</i>	<i>dali_status</i>
0x00	type: uint8 value: value_masked	See fig. 1	type: uint8 value: value_masked	See fig. 1

1.

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

## Message sample

Message in base64

```
AAIEBgIMAg==
```

Message decoded to hex

```
00|0204|0602|0C02
```

Header decoded

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

1<sup>st</sup> 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
```

2<sup>nd</sup> 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
```

3<sup>rd</sup> Device status

*dali\_address\_short* in HEX

0x0C

converted to device

Single device 6

*dali\_status* in HEX

0x02

HEX message converted to binary

0<sub>B</sub>00000010

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
<i>dimming_command</i>	<i>dali_address_short</i>	<i>dim_level</i>	<i>dali_address_short.</i>	<i>dim_level</i>
0x01	type: uint8 value: value_masked	type: uint8 unit: % 0..100 0xFF - Resume*	type: uint8 value: value_masked	type: uint8 unit: % 0..100 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

1<sup>st</sup> Query

*dali\_address\_short*

Convert single device 36 to HEX

0x48

Convert Max level query code 161 to HEX

0xA1

2<sup>nd</sup> Query

*dali\_address\_short*

Convert single device 36 to HEX

0x48

Convert Min level query code 162 to HEX

0xA2

3<sup>rd</sup> Query

*dali\_address\_short*

Convert single device 36 to HEX

0x48

Convert Power on level query code 163 to HEX

0xA3

4<sup>th</sup> Query

*dali\_address\_short*

Convert single device 36 to HEX

0x48

Convert System failure level query code 164 to HEX

0xA4

## 5<sup>th</sup> 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
<i>custom_dali_request</i>	<i>dali_address_short</i>	<i>dali_query</i>	<i>dali_answer</i>	<i>dali_address_short</i>	<i>dali_query</i>	<i>dali_answer</i>
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

1<sup>st</sup> 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

2<sup>nd</sup> 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

3<sup>rd</sup> 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

4<sup>th</sup> 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

5<sup>th</sup> 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
<i>custom_dali_command</i>	<i>dali_command</i>	
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
<i>request_status</i>	<i>functions</i>
0x05	See fig. 1

1.

Bit #	Parameter	Value
0	<i>usage</i>	0 - not requested 1 - requested
1	<i>status</i>	
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

0<sub>B</sub>00000001

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
<i>request_interfaces</i>	<i>interfaces</i>
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
<i>request_interfaces</i>	<i>dig</i>	<i>value</i>	<i>ldr</i>	<i>value</i>	<i>thr</i>	<i>value</i>	<i>relay</i>	<i>value</i>
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	<i>main relay</i>	0: open 1: closed
1	<i>od_relay</i>	
2	<i>RFU</i>	
3	<i>RFU</i>	
4	<i>RFU</i>	
5	<i>RFU</i>	
6	<i>RFU</i>	
7	<i>RFU</i>	

## Message sample

Message in base64

```
BgEAAkUD/wQA
```

Message decoded to hex

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

Header 06 decoded

```
Answer for request interface values
```

1<sup>st</sup> interface HEX value 01 converted to value

```
dig
```

*value* in HEX

```
0x00
```

HEX value converted to state

```
off
```

2<sup>nd</sup> interface HEX value 02 converted to value

```
ldr
```

*value* in HEX

```
0x45
```

HEX value converted to state

```
69
```

3<sup>rd</sup> interface HEX value 03 converted to value

```
thr
```

*value* in HEX

```
0xFF
```

HEX value converted to state

```
n/a
```

4<sup>th</sup> 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
<i>read_memory</i>	<i>dali_address_short</i>	<i>memory_bank</i>	<i>memory_address</i>	<i>read_size</i>
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

```
0704000306
```

Response:

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

### Message response sample

```
070400030607edface82e5
```

Header from request    GTIN = 8718696481509

Write to driver memory\*

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4..n
<i>write_memory</i>	<i>dali_address_short</i>	<i>memory_bank</i>	<i>memory_address</i>	<i>memory_value</i>
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
<i>timed_dimming_command</i>	<i>dali_address_short</i>	<i>dim_level</i>	<i>duration</i>	<i>dali_address_short.</i>	<i>dim_level</i>	<i>duration</i>
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
<i>packet_type</i>	<i>parameters</i>	<i>payload</i>	
type: HEX	See fig. 1	type: packet specific	

1.

Bit #	Parameter	Value
0	RFU	
1	RFU	
2	RFU	
3	RFU	
4	<i>length</i>	0..15
5		
6		
7		



## Digital interface alert

Byte 0	Byte 1	Byte 2	Byte 3
<i>dig_alert</i>	<i>parameters</i>	<i>counter</i>	
0x80	0x20	<i>uint16</i>	

## Message sample

### Message in base64

```
gAA=
```

### Message decoded to HEX

```
80|20|0600
```

Byte 2-3 flip LSB

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

## LDR interface alert

Byte 0	Byte 1	Byte 2	Byte 3
<i>ldr_alert</i>	<i>parameters</i>	<i>state</i>	<i>ldr_value</i>
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	
<i>dali_driver_status</i>	<i>parameters</i>	<i>dali_address_short</i>	<i>dali_status</i>	<i>dali_address_short</i>	<i>dali_status</i>		
0x83	0x20..0xE0*	type: uint8 value: value_masked	See fig. 1	type: uint8 value: value_masked	See fig. 1		

1.

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

\* 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
<i>packet_type</i>	<i>device_serial</i>	<i>firmware_version</i>	<i>device_unix_epoch</i>	<i>device_config</i>	<i>optional_features</i>	<i>dali_info</i>	<i>driver_info</i>	<i>reset_reason</i>
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	<i>dig</i>	
3	<i>ldr</i>	
4	open_drain	
5	metering	
6	RFU	
7	<i>custom_request</i>	

3.

Bit #	Parameter	Value
0	<i>bus_supply</i>	0x00..0x6F type: uint unit: V value: scaled
1		
2		
3		
4		
5		
6		
7	<i>bus_power</i>	0 - internal 1 - external

4.

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

5.

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

## Message sample

Message in base64

```
AA0Ag1ABAQHzyfyBeAAz+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 - [0b00000001]
0 :
0 :
0 :
0 :
0 :
0 :
0 : unaddressed_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
<i>packet_type</i>	<i>packet_from_fport</i>	<i>parse_error_code</i>
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

13|32|04

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

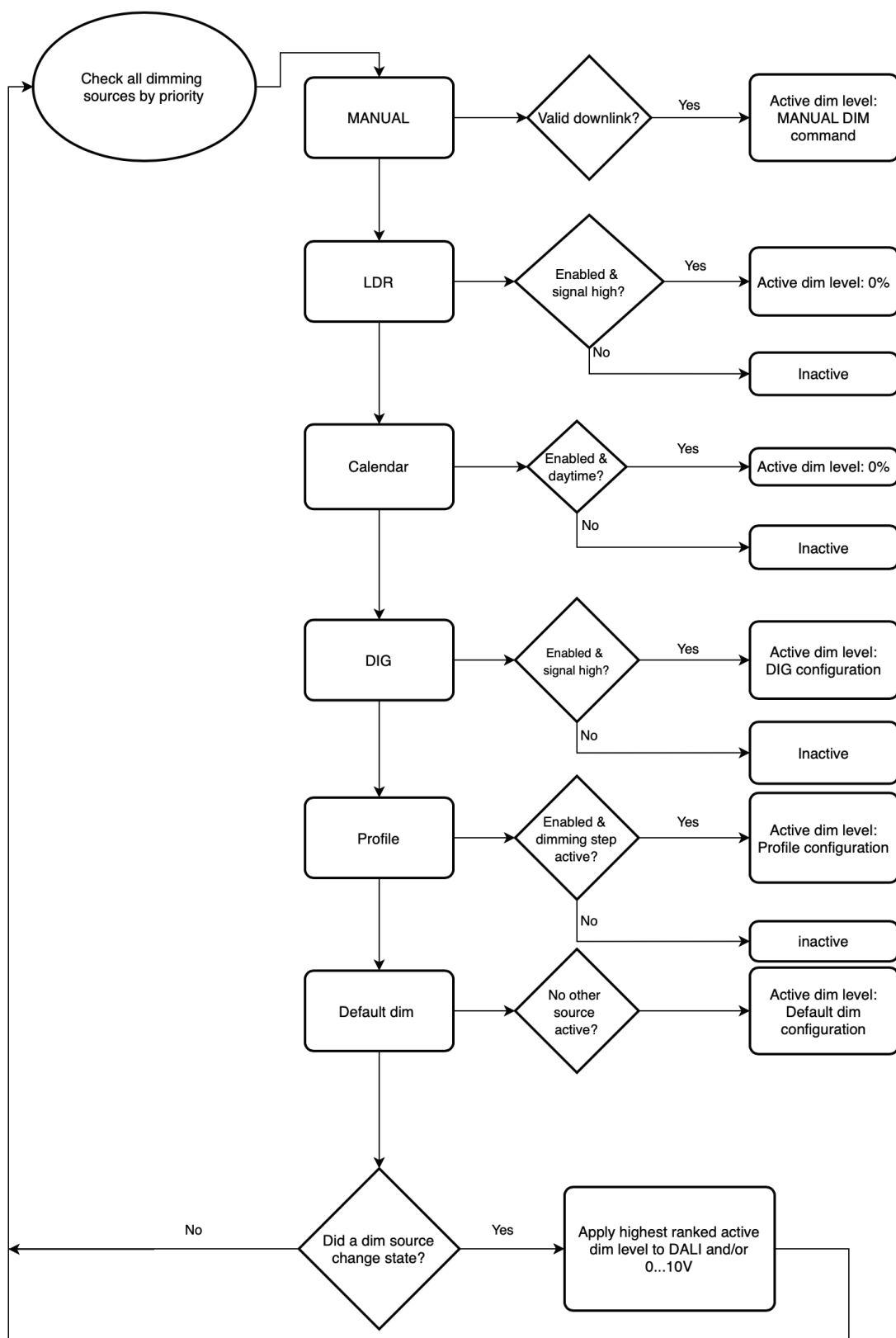
converted to error

*packet\_size\_long*

## Error message

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

# CONTACT INFORMATION

NASYS Ltd

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.

2.4.3 - Added \*Xv devices compatibility

All content contained herein is subject to change without notice. Nordic Automation Systems reserves the right to change or modify the content at any time.

# 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 [Unlicense.org](http://Unlicense.org)

## Eclipse

Copyright (c) 2007, Eclipse Foundation, Inc. and its licensors.

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 Eclipse Foundation, Inc. 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 THE COPYRIGHT OWNER 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.

## Embedded Template Library

The MIT License (MIT)

Copyright (c) 2016 jwellbelove, [https://github.com/ETL\\_CPP/etl](https://github.com/ETL_CPP/etl), <http://www.etlcpp.com>

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.