AN11174

DALI slave using the LPC111x

Rev. 2 — 6 March 2013

Application note

Document information

Info	Content
Keywords	Lighting networks, DALI, CORTEX M0, ARM, LPC111x, Microcontroller
Abstract	This application note describes how to create a DALI slave device with the OM13026 LPC111x DALI slave board.

Revision history

Rev	Date	Description
2	20130306	• Editorial updates throughout. • Updated Section 3.
1	20120301	• Initial version.

Contact information

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

1. Introduction

The aim of this document is to describe how to create a *DALI slave device* with the OM13026 DALI slave board.

It covers the following aspects:

- Short introduction into DALI lighting networks
- OM13026 LPC111x DALI slave board design description
- Global software overview
- Supported DALI command set
- Software configuration items

This application note does *not* include items covering the light generation aspect, especially:

- Description of lighting drivers (CFL, SSL)
- Lamp feedback (color, temperature)

1.1 Overview

This application note is intended for usage with the DALI slave board OM13026 as depicted in Fig 1.

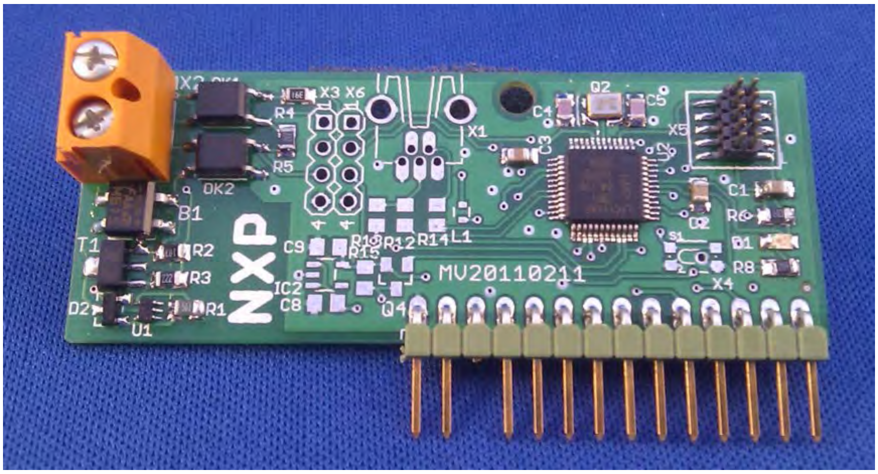


Fig 1. OM13026 LPC111x DALI Slave board

The OM13026 board is an example implementation of an isolated physical layer for the DALI bus with a Cortex M0 LPC111x microcontroller for the DALI protocol handling, and many I/O functions to steer external lighting drivers for solid state or compact fluorescent lighting applications.

The board is not pre-programmed with the DALI software described in this application note. The user should reprogram the board with the slave software from DALI SDK 2.

1.2 DALI

This section briefly describes the Digital Addressable Lighting Interface (DALI) to understand the terms and concepts used in the other chapters of this document. For more information on the DALI standard, see References [1], [2], and [3].

1.2.1 Bus structure

DALI uses a wired bus structure to create a communication path between control devices (switches or central gateways) and control gear (lighting units). The topology is not specified; it can be a star, bus or point to point. A ring structure is not allowed.

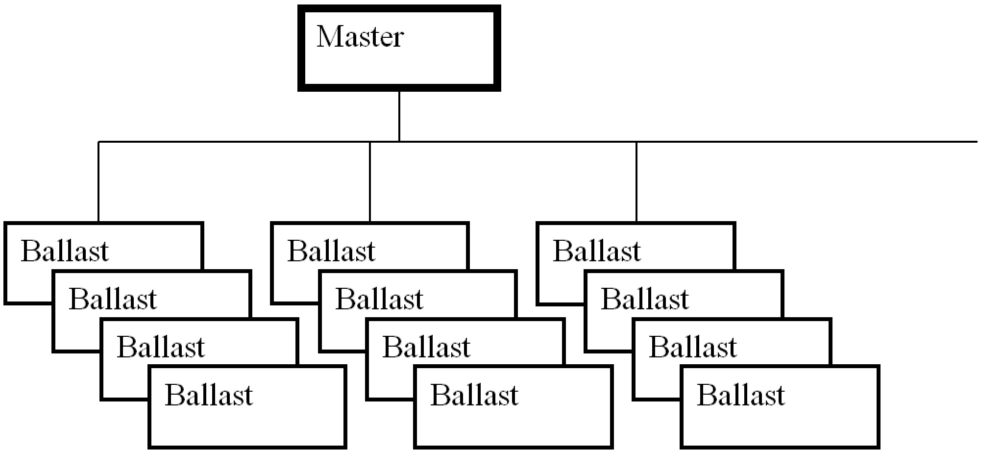


Fig 2. Example bus of control device and multiple control gear

For installation purposes, the DALI signal wiring can be combined with mains connections in one cable as shown in Fig 3.

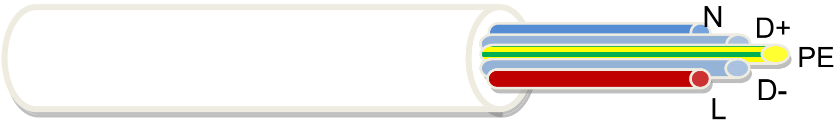


Fig 3. DALI wires combined with mains power wires

1.2.2 Physical layer

The DALI bus consists of two wires on which data is transmitted in frames. There are two different frame types: a “forward” frame (2 bytes sent by the master to the slave) and a “backward” frame (1 byte sent by the slave to the master on request of the master).

DALI uses a bi-phase (also called Manchester) encoding, which means that the data is transmitted using the edges of the signal. A rising edge indicates a ‘1’, and a falling edge indicates a ‘0’ (see Fig 4).

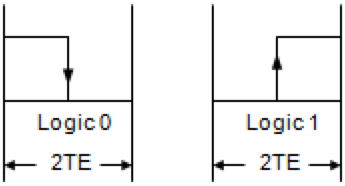


Fig 4. Bi-phase encoding of data

The encoded bits are actually represented by high and low voltage levels on the bus. Typically, the low voltage is 0 V and the high voltage is 16 V. The maximum and minimum bus voltage at both the transmitting unit and the receiving unit are given in Fig 5.

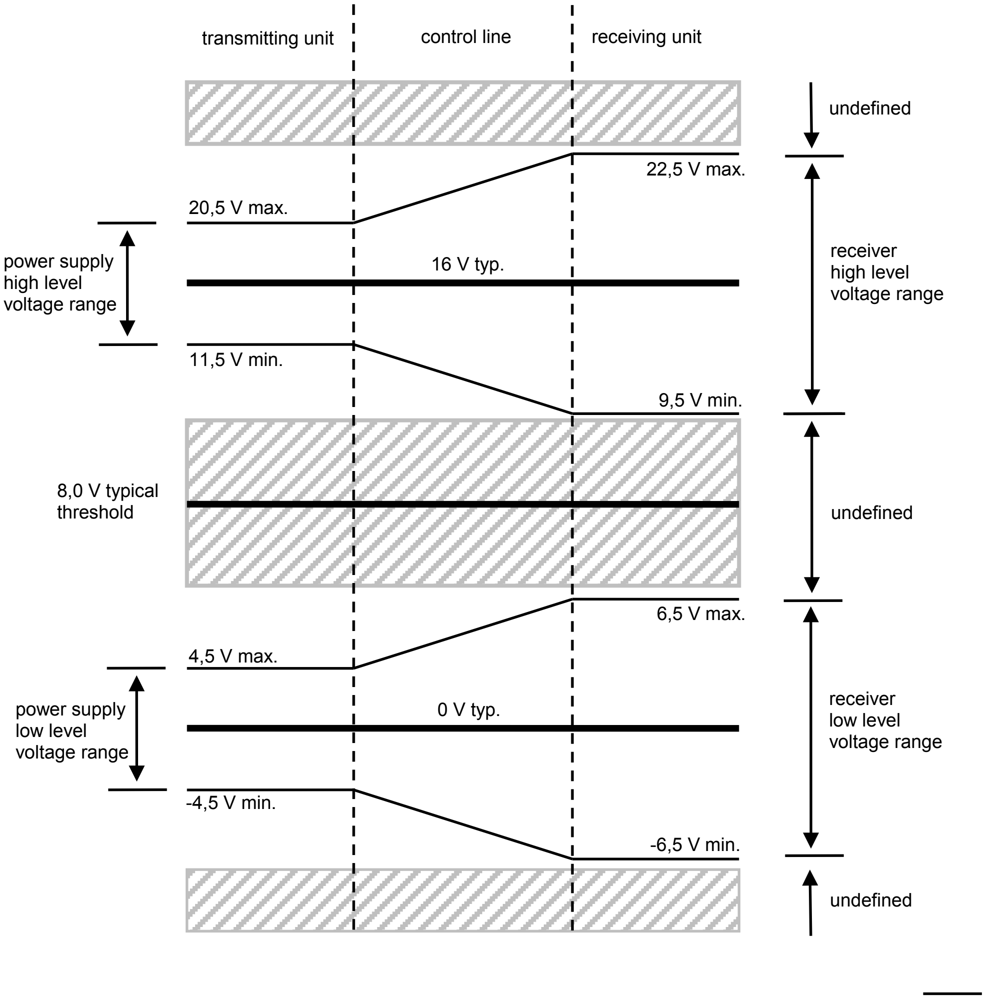


Fig 5. Bus voltage levels

The bus is powered via a bus power supply, which can be part of the master on the bus and has a current limit of 250 mA. Whenever a control gear on the bus wants to modulate the bus voltage, it will short circuit the bus to create a low voltage level. If a high level is desired, it will put its transmission stage in a high impedance state.

1.2.3 Logical layer

In a typical application, a DALI-bus consists of one controller (master), and multiple slaves (normally TL-ballasts). It can control up to 64 different slaves (ballasts) within the same control system. It’s possible to transmit commands to single ballasts or to a group of ballasts.

Every bit takes two periods TE. The defined bit rate of DALI is 1200 bps. So a 1 bit period (2TE) is ~834 µsec. A frame is started by a start bit, and ends with two high-level stop bits (no change of phase). Data is transmitted with the MSB first. Between frames, the bus is in idle (high) state (see Fig 6).

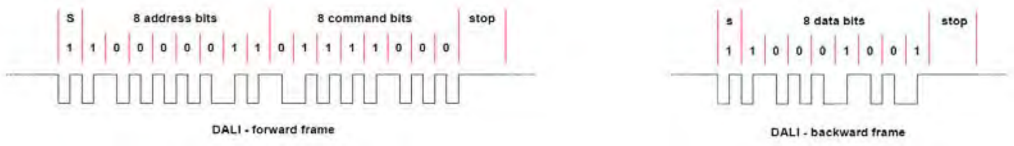


Fig 6. Physical encoding

Additional protocol timing requirements for transmission are:

- The settling time between two subsequent forward frames shall be at least 9.17 ms. This means that 4 forward frames with accompanying periods of 9.17 ms shall fit exactly in 100 ms.
- The settling time between forward and backward frames (transition from forward to backward) shall be between 2.92 ms and 9.17 ms. After sending the forward frame, the master unit will wait for 9.17 ms. If no backward frame has been started after 9.17 ms this is interpreted as “no answer” from slave.

The settling time between backward and forward frames (transition from backward to forward) shall be at least 9.17 ms (see Fig 7).

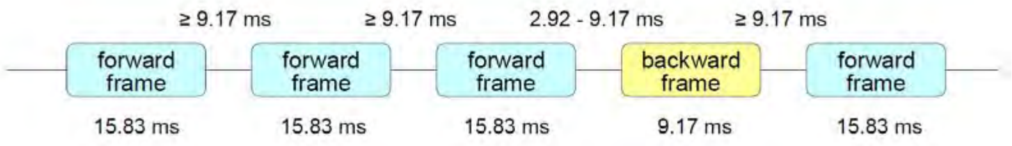
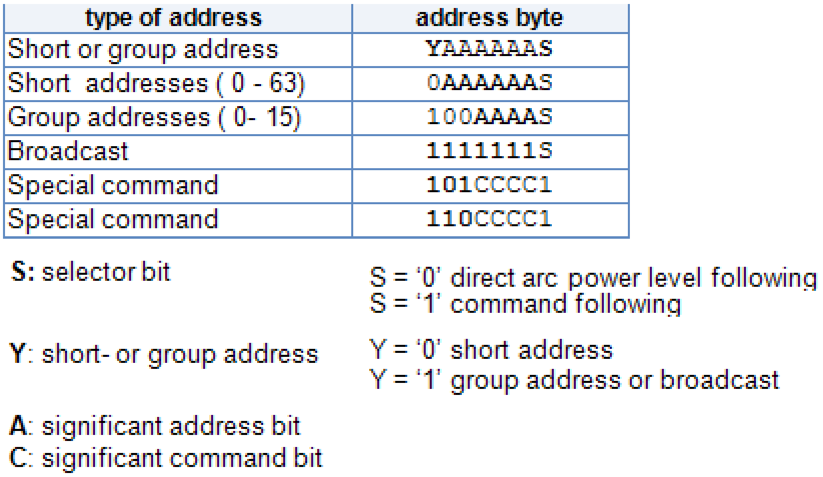


Fig 7. Timing of forward and backward frames

Every DALI slave is able to react to a short address, 16 group addresses and broadcast. For addressing the following scheme is shown in Fig 8:



type of address	address byte
Short or group address	YAAAAAAS
Short addresses (0 - 63)	0AAAAAAS
Group addresses (0- 15)	100AAAAS
Broadcast	1111111S
Special command	101CCCC1
Special command	110CCCC1

S: selector bit

S = '0' direct arc power level following
S = '1' command following

Y: short- or group address

Y = '0' short address
Y = '1' group address or broadcast

A: significant address bit
C: significant command bit

Fig 8. DALI addressing types

After the address byte, a second byte follows the forward frame. This second byte contains the direct arc power level for the ballast or a command byte depending on the selector bit. There are four types of commands:

1. Direct/Indirect arc power control commands – used to set ballast power level
2. Configuration commands – configures the ballast (for example: add to a group or store level). Command must be repeated within 100 ms, otherwise it's ignored.
3. Query commands – ask slave (ballast) for status information (for example: power level or version number). The slave can send a backward frame.
4. Special commands – used to initialize and setup the ballast, some must be repeated within 100 ms, and some require an answer from the slave. Most commands are only processed within 15 minutes after an "INITIALIZE" command is received.

2. Hardware description

This section describes the hardware of the DALI slave board. First the physical layer of the board will be discussed followed the complete board description.

2.1 Physical layer

The schematic of the DALI physical layer from board OM13026 is depicted in Fig 9. The DALI bus connects to the connections D. The microcontroller LPC111x is connected to the signals DALI1_TX and DALI1_RX.

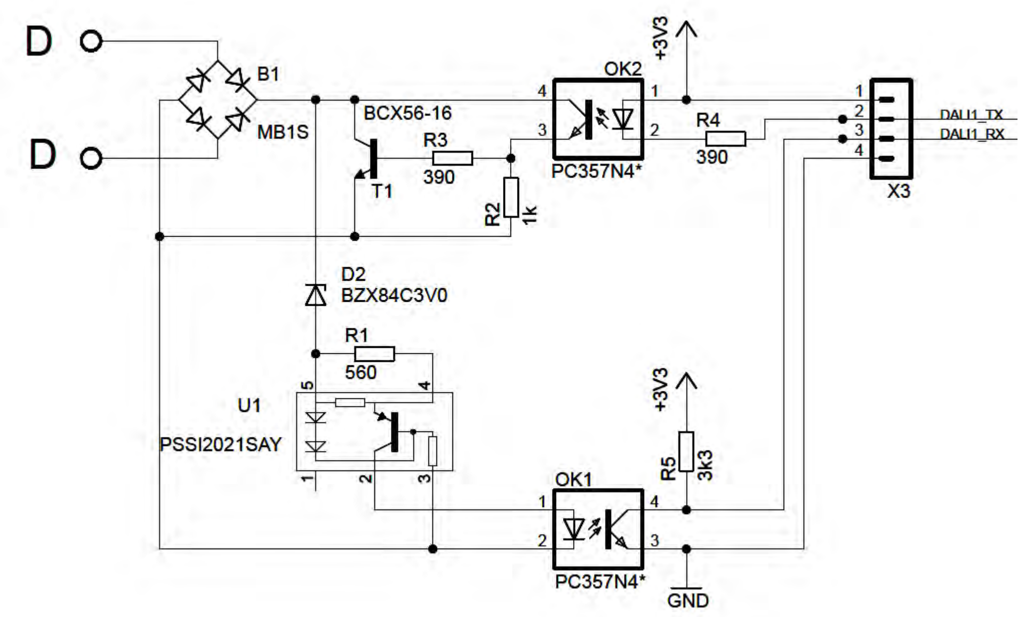


Fig 9. Physical layer schematic

By usage of bridge rectifier B1, the design is made polarity independent. Both terminals D are interchangeable.

The upper part of Fig 9 contains the transmission part of the DALI slave. It is created around T1, R2, R3, OK2 and R4. The signal DALI1_TX is driven by the microcontroller at 0 V or 3.3 V. For low signals of DALI1_TX, Optocoupler OK2 will connect the junction of R2 and R3 to the DALI bus. This will create a drive current for the base of T1 that will start to conduct and short circuit the DALI bus via bridge B1. When the signal DALI1_TX is high, transistor T1 will not conduct and the bus will be in the ‘high’ state. The resistor R3 of 390 Ohm is chosen such that T1 will sufficiently go into saturation and thus create a low voltage level on the bus while having a low voltage drop on T1. This keeps the power dissipation of T1 limited as it should be able to sink the maximum DALI bus current of 250 mA. Also, for this reason, T1 should have a high h_{FE} of minimum 100. In this way no additional cooling area is needed.

The reception path is shown in the lower part of Fig 9. It is created around optocoupler OK1, U1, R1, R5 and Zener diode D2. When the DALI bus is idle (high) a constant current source of about 1 mA is created using U1 and R1. This current is used to drive optocoupler OK1 that signals the level of the DALI bus to the microcontroller via DALI1_RX. The current source limits the maximal current load the circuit creates when not in transmission mode. Zener D2 and bridge rectifier B1 drop the received bus voltage to a level to guarantee that a low level voltage of 6.5 V does not drive the optocoupler.

The circuit around OK1 and R3 creates an inverted signal to the microcontroller: A high DALI bus level will connect DALI1_RX to low. A low DALI bus level will create a high signal on DALI1_RX.

All components are chosen to withstand several factors (70 V to 80 V) of the highest allowed DALI bus voltage level of 22.5 V. The physical layer does not contain overvoltage protection or suppression components. This is left up to the reader.

The optocouplers create isolation between the microcontroller side and the DALI bus. The isolation is sufficient for evaluation of the DALI software stack when the microcontroller is connected in a non isolated way to the mains supply; any re-use of this design should be made compliant to the isolation requirements as specified in section 5.4 of Reference [1].

2.2 Microcontroller

The section of the design which handles the incoming and outgoing DALI messages and controls the lighting is shown in Fig 10. The DALI1_RX signal from the DALI physical interface is connected to a 32-bit timer/capture unit CT32B0 of U2. The DALI1_TX signal is connected to a general purpose IO pin that is software controlled to generate the DALI signal timing. Connector X6 makes is possible to connect a second, different, physical interface to the timer/capture unit CT32B1 of U2.

Debugging and flashing connection is provided by means of header X5, which complies with the 10-pin SWD standard as supported by many flash and software tools.

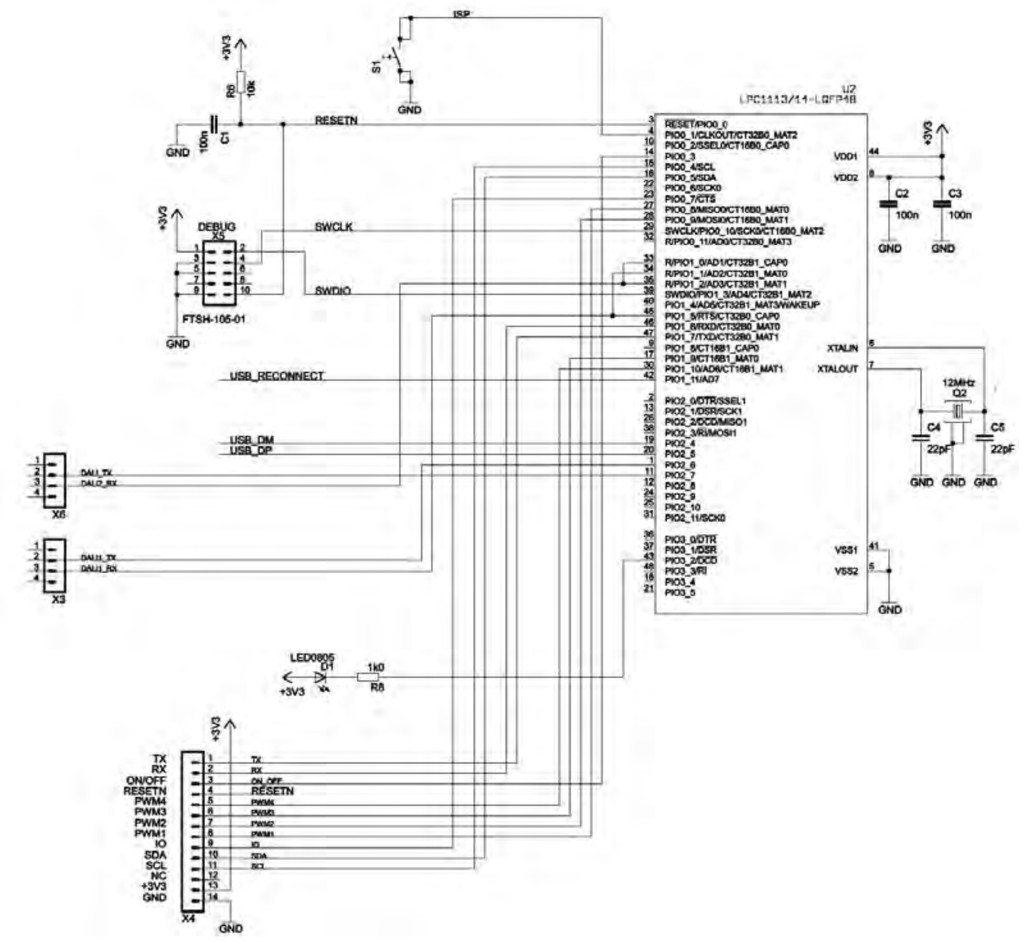


Fig 10. Microcontroller connections

The OM13026 board can derive its microcontroller clock from a) an external connected crystal (circuit Q2, C4, C5) or b) an internal RC oscillator. This is configurable in the application software.

The function of header X4 is twofold: it contains the 3.3 V supply on pins 13 and 14 and contains many input/output controls on pins 1 to 11. The board should be powered by an external 3.3 V supply. The OM13026 board consumes 9 mA of current at 48 MHz, and 4 mA at 12 MHz.

The input/output controls are listed in Table 1. It shows the default function of the pins on header X4.

Table 1. Header X4 description

Pin	Description	Direction	LPC1114 pin	Remark
1	TX	O	PIO1_7/TXD/CT32B0_MAT1	UART
2	RX	I	PIO1_6/RXD/CT32B0_MAT0	UART
3	ON_OFF	O	PIO0_3	
4	RESETN	I/O	nReset/PIO0_0	
5	PWM1	O	PIO0_8/MISO0/CT16B0_MAT0	
6	PWM2	O	PIO0_9/MOSI0/CT16B0_MAT1	
7	PWM3	O	PIO1_9/CT16B1_MAT0	
8	PWM4	O	PIO1_10/AD6/CT16B1_MAT1	
9	IO	I	PIO0_7/nCTS	LAMP FAILURE
10	SDA	I/O	PIO0_5/SDA	I2C
11	SCL	O	PIO0_4/SCL	I2C
12	nc			
13	+3V3	I	VDD1,2	
14	GND	I	VSS1,2	

Up to four PWM signals are available to independently drive different lighting units. The frequency and resolution of the signals is software programmable. The ON_OFF signal can be used independently from the PWM signals to switch a lighting driver into an OFF or ON state. The I^{2}C-bus pins can be used to externally connect other devices like EEPROM or a temperature sensor. An analog A/D input is available via pin 8. The IO signal on pin 9 is used as a Lamp Failure digital input pin by the software, high (logical '1') meaning lamp OK, low (logical '0') meaning lamp FAIL.

UART, I^{2}C and A/D converter functionality is not included in the software release.

2.3 Board layout

The layout of the board is given in Fig 11. On header X4, pin 12 is removed to function as key to circumvent misplacement when attaching it to a lighting driver.

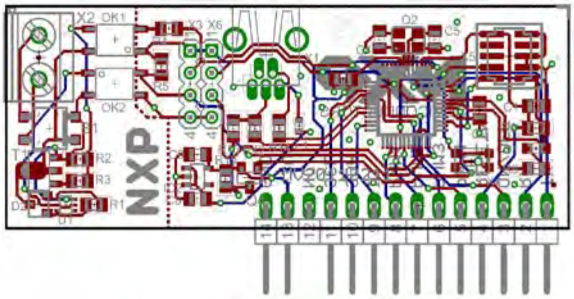


Fig 11. PCB layout

In Fig 11, the optical isolated DALI physical interface is placed on the left side of the dotted line. On the right side of the board the LPC1114 microcontroller with the 10-pin SWD debug/programming header is placed. The middle part of the printed circuit board is not populated and not necessary for use as DALI slave unit. Header X6 is meant to connect a second, different DALI physical layer or can be used for an additional GPIO or 32-bit timer output. The other components are used when U2 is mounted with the pin compatible LPC1343.

The PCB does not have milling at the isolation border between the microcontroller side and the DALI bus interface. It is advised to recheck the layout design to comply with the isolation requirements as specified in section 5.4 of Reference [1].

2.4 Component list

The list of components is given in Table 2. The oscillator circuit around C4, C5 and Q2 is listed as optional; the microcontroller can also use its internal reference clock to generate its internal clock. This is configurable by software.

The USB circuitry around components IC2, L1, X1, Q4, R15, R12, R14, R13, C8 and C9 is not mounted: this part of the circuit only applies when the footprint of U2 is mounted with the pin compatible LPC1343, which is not the case for the OM13026 board.

Table 2. List of components

Part ref	Description	Manufacturer	Package	Remarks
B1	MB1S		SOIC-4	
C1	100nF		0805	
C2	100nF		0805	
C3	100nF		0805	
C4	22pF		0805	optional
C5	22pF		0805	optional
C8	4u7/10V/X7R		0805	not mounted
C9	2u2/6V3/X7R		0805	not mounted
D1	LED0805		CHIPLED_0805	
D2	BZX84C3V0	NXP Semiconductors	TO236	
IC2	SA57000-33D	NXP Semiconductors		not mounted
L1	BLM18AG601S		0603	not mounted
OK1	PC357N4	Sharp		
OK2	PC357N4	Sharp		
Q2	12MHz			optional
Q4	BSH205	NXP Semiconductors	SOT23	
R1	560R		0805	
R2	1k		0805	
R3	390R		0805	
R4	390R		0805	
R5	3k3		0805	
R6	10k		0805	
R8	1k0		0805	
R12	33R		0805	not mounted
R13	1k5		0805	not mounted
R14	33R		0805	not mounted
R15	nc		0805	
S1	KMR211G	C&K Components		not mounted
T1	BCX56-16	NXP Semiconductors	SOT89-BCE	

Part ref	Description	Manufacturer	Package	Remarks
U1	PSSI2021SAY	NXP Semiconductors	SOT353	
U2	LPC111x	NXP Semiconductors	LQFP48	
X1	565790519	MOLEX	MINI-AB USB	not mounted
X2	MKDSN1,5/2-5			
X3	MA04-1R			not mounted
X4	MA14-1W			pin 12 removed
X5	FTSH-105-01	SAMTEC		
X6	MA04-1R			not mounted

3. Software description

This chapter describes the structure and components of the software project for usage with hardware board OM13026 to create a DALI control gear. After a small software overview the DALI stack will be explained in detail. The other components of the example software project are not discussed in depth.

3.1 Decomposition

The decomposition of the software is given in Fig 12.

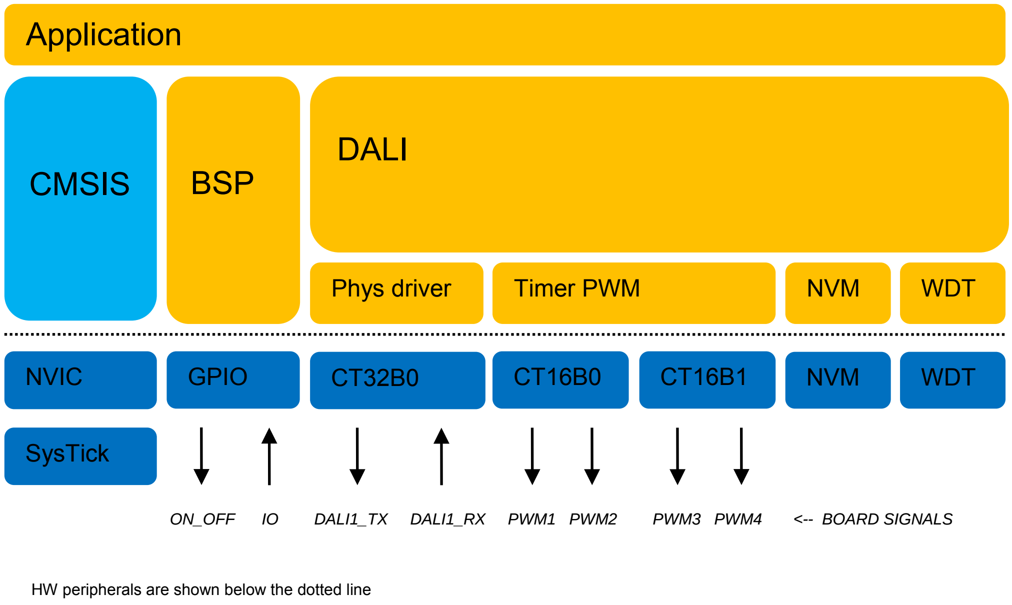


Fig 12. DALI Slave system decomposition

For the software interface of the hardware peripherals, such as NVIC, SysTick and PLLs of the LPC111x, the Cortex Microcontroller Software Interface Standard (CMSIS) is used.

The board support package (BSP) does low level board setup, keeps track of the system uptime, and exports functions to the application such as toggle the heartbeat LED, etc.

The DALI component implements the DALI stack, handles the brown out IRQ, handles all the DALI timeouts on the SysTick IRQ and handles the fading on the Timer16_0_IRQ.

The DALI physical driver handles the reception and transmission of DALI frames using the CT32B0 timer/capture unit.

The timer PWM driver uses the 16 bit timer units CT16B0 and CT16B1 to generate pulse width modulated (PWM) signals to control the ballast.

General purpose IO pins can be used to signal a lighting driver to switch on or off, and to read lamp failure information into the DALI component.

The non volatile memory (NVM) component implements functionality to read and write data from the highest sector in the internal flash memory.

The application, the ‘main’ of the program, first calls the *SystemInit* function of CMSIS, then it calls the *bsp_init* function, then the DALI stack is setup by calling the function *DALI_Init*, and finally the function *DALI_CommandHandler* is called from within the programs main loop.

3.2 Component structure

The DALI component *libDALI_LPC111x.a* is responsible for:

- reading DALI forward frames from the DALI physical interface
- acting upon received commands
- driving the connected lighting unit via PWM signals
- persistent storage of DALI device parameters on brown out event
- sending DALI backward frames via the DALI physical interface

The functions for receiving forward frames and transmitting backward frames from the physical DALI bus interface are bundled in the file *DALI_PhysDriver.c*. The systems main execution loop, which can be found in *main.c,* calls the function *DALI_CommandHandler*, which reads forward frames via the DALI physical driver. The forward frame parsing and assembly of the backward frame is done in *libDALI_LPC111x.a*. The actual control of the ballast is also done in *libDALI_LPC111x.a*. Storage of non volatile parameters is done using *nvm.c*.

All the files that are related to implementing the DALI specification have a name that starts with *DALI*. The relations between the files are given in Fig 13.

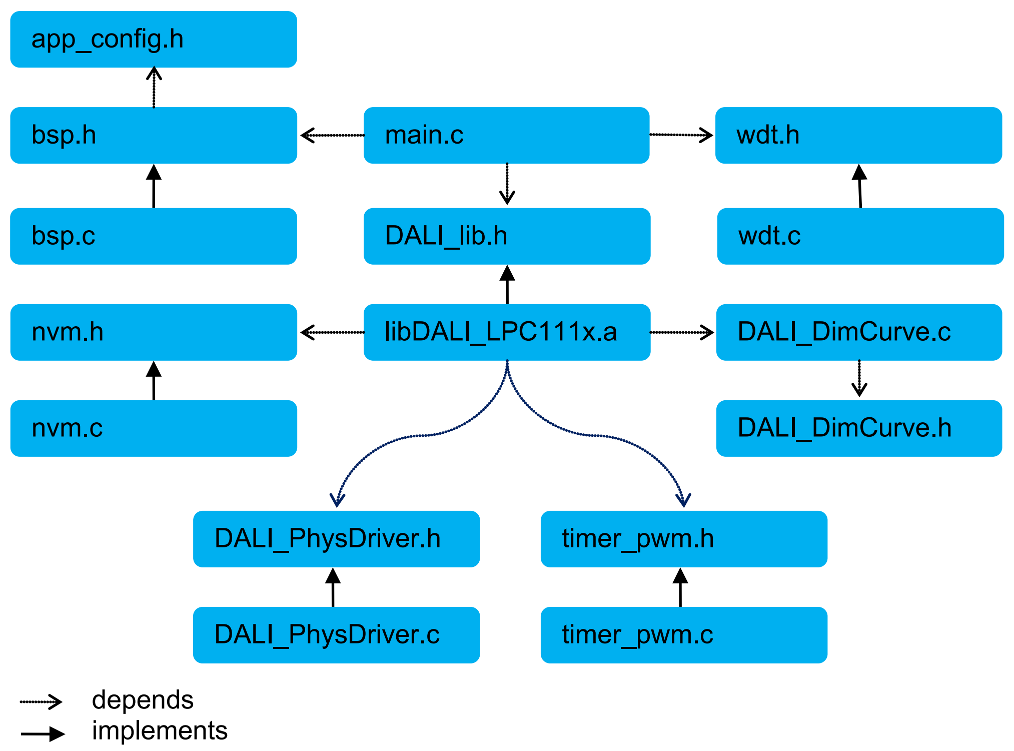


Fig 13. DALI Slave file dependencies

The file *DALI_lib.h* defines the external interfaces of the DALI component and type definitions for the setup of the component.

3.3 Run time flow

A simplified software execution flow of the DALI Slave is shown in Fig 14.

All functions of the left hand side in Fig 14, the "application program thread", are called from the function *main* located in file *main.c.* First the clock structure of the microcontroller is initialized with *system_init* , next the system tick and microcontroller I/O are initialized with *bsp_init.* The DALI communication and ballast configuration is setup and used to initialize the DALI communication: the exact configuration options are described in section 3.10.

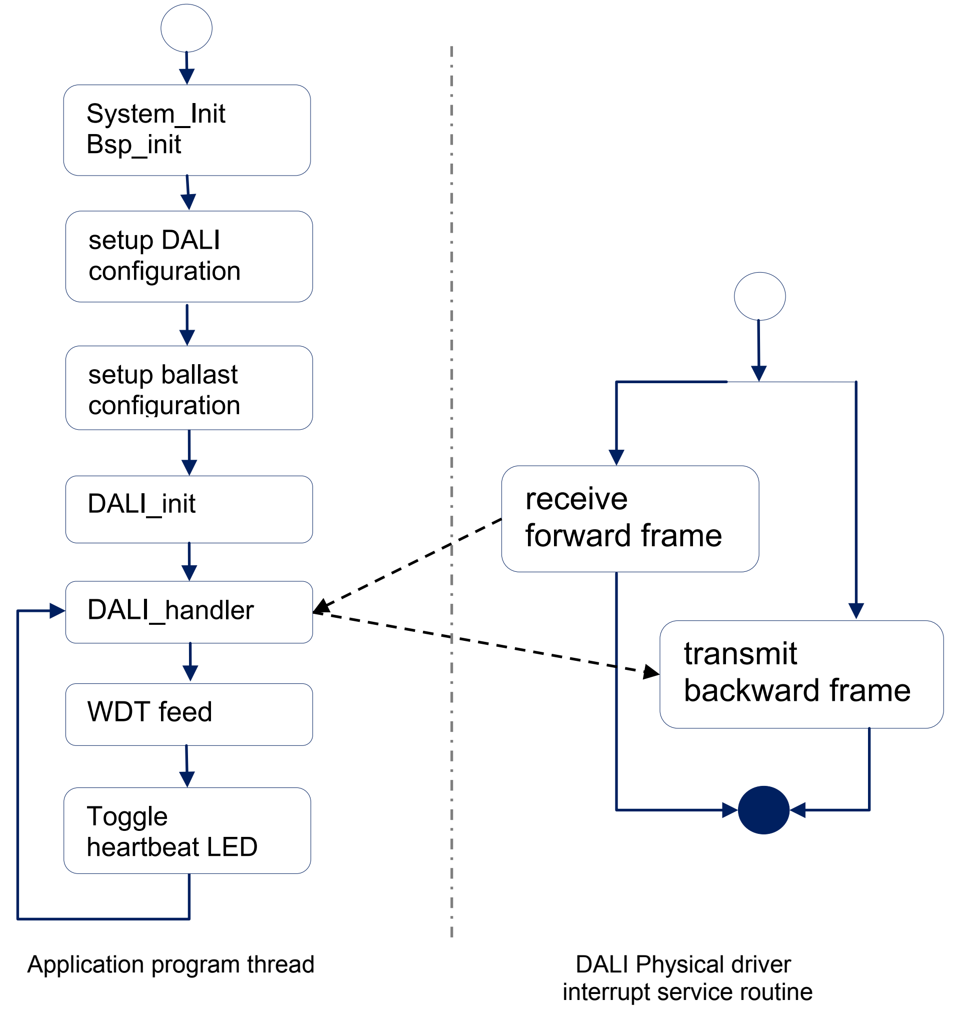


Fig 14. Simplified DALI Slave execution flow

After the initialization the *DALI handler* function waits for forward frames and takes care of initiating possible backframe communication. The “interrupt service routine” for forward frame reception and backward frame transmission is implemented by *Timer32_0_IRQ Handler* and is located in *DALI_PhysDriver.c:* this is indicated in the right hand side of Fig 14. The DALI handler returns after a timeout or if a DALI command has been received and interpreted. Next the watchdog is fed and a heartbeat LED is toggled. In the case of no DALI communication the heartbeat LED toggles with a constant rate. In the case of DALI communication it is toggled with the rate of each received frame on the bus.

By default the DALI physical driver is in receive mode and will switch to transmit mode when a backward frame must be send. When the transmission of the backward frame is ready the DALI physical driver automatically switches back to receive mode.

The interrupt driven reception of DALI frames is done symbol after symbol, because one bit is bi-phase coded into two symbols. Multiple calls of the ISR routine concatenate the received symbols until a raw forward frame is received. The same holds for the transmission part of the ISR: it puts a raw backward frame on the bus, symbol after symbol by subsequent invocations of the ISR. After reception the raw forward frame is converted into the original forward frame.

Additionally the system has a system tick ISR, which can be found in *bsp.c*, that keeps track of the system uptime. The function *DALI_SysTickHook* , exported by the DALI component, is coupled to this system tick ISR to be able to handle DALI timeout checking. This coupling is realized by calling the function *bsp_set_systick_hook*.

The interrupt Timer16_0_IRQ of the PWM timer is used to reprogram the duty cycle, and for lighting fading effects by doing interpolation steps between DALI arc power levels. The lighting fading is done by the DALI component. The DALI component is coupled to this PWM timer interrupt by passing a callback function when calling *Timer_PWM_Start* (part of “init ballast”).

3.4 DALI reception and transmission

The physical layer circuit of Fig 9 is connected to three pins of the microcontroller unit. The incoming *DALI1_RX* signal is connected to a capture input of the timer capture unit and to GPIO input PIO1_1. The *DALI1_TX* signal is connected to GPIO output PIO2_6.

3.4.1 Inter frame timing

The match register 0 (MR0) of the timer capture unit is used to create the minimal frame delay between forward frames (see Fig 7). The MR0 is set to 22Te. When this match register generates an interrupt, the ISR reception state is set to receive mode and capture interrupts are enabled.

3.4.2 Forward frame reception

The capture function of the timer capture unit is used to measure the pulse duration of the incoming DALI message; the timer capture unit is set to a clock of 1 MHz. This results in a 1 microsecond accurate timing measurement for the DALI bus timing. Upon the first transition of the DALI1_RX line the timer counter is reset. During each following DALI1_RX transition, either to low or to high, the ISR is triggered and the duration of the pulse is measured using capture register 0 (CR0). A bitstream is generated of which the contents depend on the duration of the pulse (due to the Manchester encoding of the DALI bus signal). After receiving 17 Manchester encoded bits the ISR detects the stopbits with a duration of 4*Te using match register 2 (MR2). When the ISR triggers on MR2 after 4*Te, and there has been no DALI line activity, the forward frame bi-phase bits are stored by the ISR for processing by the main loop. Fig 15 illustrates the relation between the bus timing and the corresponding timer capture and match registers in use.

3.4.3 Command handling

The main loop blocks until a forward frame is returned by the DALI physical driver. The bi-phase encoded forward frame (raw forward frame), as received from the ISR, is decoded into a forward frame by the DALI physical driver. The DALI component decodes the forward frame into a command with data. This command is then parsed and the corresponding action is performed.

3.4.4 Backward frame transmission

When the command handler decides that a backward frame should be transmitted, it calls the DALI physical driver function *DALI_SendBackwardFrame.* This function encodes the response into a Manchester encoded message, sets the DALI physical driver in transmit mode and enables the interrupt match register 2 (MR2) with an initial time of 12*Te (must be between 7*Te and 22*Te). On the first time MR2 fires the ISR is triggered and the MR2 value is set to 1*Te. On each subsequent interrupt generated by MR2 the DALI1_TX GPIO is set by the ISR to the corresponding high/low level of the Manchester encoded backward frame. When all bi-phase encoded bits of the backward frame are transmitted the DALI physical driver state is set to receive mode.

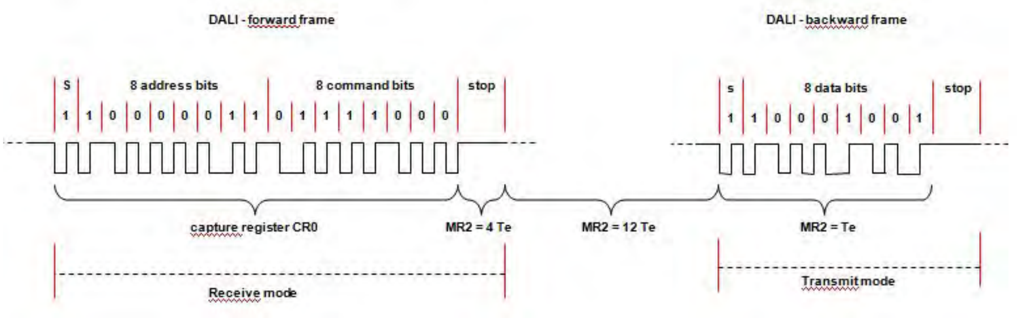


Fig 15. Physical encoding/decoding using capture timer unit

3.5 Interrupts

In total four interrupt sources are used by the DALI Slave software. The following table shows the interrupts that are used by the DALI Slave software, and describes their priority and function.

Table 3. DALI slave interrupts in use

Interrupt	Priority	ISR in file	Description
BOD_IRQ	0 (highest)	libDALI_LPC111x.a	Brown Out Detect IRQ, used to store DALI device parameters in NVM. Triggers only on Brown Out.
TIMER16_0_IRQ	1	timer_pwm.c	Reprograms the PWM1, PWM2, PWM3 and PWM4 signals, used for driving lighting units, and handles the DALI fading. Triggers with the PWM frequency.
TIMER32_0_IRQ	2	DALI_PhysDriver.c	Handles RX and TX on DALI physical interface #1. Triggers on DALI bus activity.
SysTick_IRQ	3 (lowest)	bsp.c	Used to generate the system uptime and to check the DALI timeouts. Triggers every 10 msec.

3.6 Lighting driver control

The software provides two ways to control lighting drivers: a digital ON/OFF (high/low) output signal, and up to four pulse width modulated (PWM) output signals to steer the brightness level of a light driver. Usage of the ON/OFF signal is optional (configurable in the software). The PWM signals are generated using the 16 bit timer units. All the functionality of lighting driver control is contained in the DALI component.

Timer capture unit CT16B0 is used for PWM signals 1 and 2; timer capture unit CT16B1 is used for generating PWM signals 3 and 4. For each of these timer capture units the match register MR3 is used to set the PWM frequency. Match registers MR0 and MR1 of both timer units are used to set the duty cycle of the PWM signals. Only the interrupt of MR3 of CT16B0 is used to reload the 16 bit timer units with new duty cycle settings. For more information on the mode of operation of the timer capture unit we refer to the timer chapter in the LPC111x user manual [4].

Each DALI arc power level is translated via a look up table to a duty cycle setting of the timer unit. For each output a separate lookup table can be used. The lookup tables are defined in *DALI_DimCurve.h*. When using the DALI fade features, the software calculates intermediate steps for creation of smooth fade effects. This is done in the DALI component.

Generation of lamp failure information is lighting driver dependent. Any lamp failure information can be coupled to the DALI stack via digital input pin PIO0_7 (IO), which is read out in the function *bsp_read_lamp_fail* in *bsp.c*. A low level on this digital input pin is interpreted as lamp failure.

3.7 Supported commands

The following tables give an overview of the supported commands of the accompanying software. If there is no remark the command is implemented.

Table 4. Indirect arc power control commands

Command nr	Description	Remarks
0	OFF	
1	UP	
2	DOWN	
3	STEP UP	
4	STEP DOWN	
5	RECALL MAX LEVEL	
6	RECALL MIN LEVEL	
7	STEP DOWN AND OFF	
8	ON AND STEP UP	
9	ENABLE DAPC SEQUENCE	
10-15	RESERVED COMMANDS	
16-31	GO TO SCENE	

Table 5. Configuration commands

Command nr	Description	Remarks
32	RESET	
33	STORE ACTUAL LEVEL IN DTR	
34-41	RESERVED COMMANDS	
42	STORE DTR AS MAX LEVEL	
43	STORE DTR AS MIN LEVEL	
44	STORE DTR AS SYSTEM FAILURE LEVEL	
45	STORE DTR AS POWER ON LEVEL	
46	STORE DTR AS FADE TIME	
47	STORE DTR AS FADE RATE	
48-63	RESERVED COMMANDS	
64-79	STORE DTR AS SCENE	
80-95	REMOVE FROM SCENE	
96-111	ADD TO GROUP	
112-127	REMOVE FROM GROUP	
128	STORE DTR AS SHORT ADR	
129	ENABLE WRITE MEMORY	
130-143	RESERVED COMMANDS	

Table 6. Query commands

Command nr	Description	Remarks
144	QUERY STATUS	
145	QUERY CONTROL GEAR	
146	QUERY LAMP FAILURE	
147	QUERY LAMP POWER ON	
148	QUERY LIMIT ERROR	
149	QUERY RESET STATE	
150	QUERY MISSING SHORT ADR	
151	QUERY VERSION NUMBER	
152	QUERY CONTENT DTR	
153	QUERY DEVICE TYPE	
154	QUERY PHYS MIN LEVEL	
155	QUERY POWER FAILURE	
156	QUERY CONTENT DTR1	
157	QUERY CONTENT DTR2	
158-159	RESERVED COMMANDS	
160	QUERY ACTUAL LEVEL	
161	QUERY MAX LEVEL	
162	QUERY MIN LEVEL	
163	QUERY POWER ON LEVEL	

Command nr	Description	Remarks
164	QUERY SYSTEM FAILURE LEVEL	
165	QUERY FADE TIME/RATE	
166-175	RESERVED COMMANDS	
176-191	QUERY SCENE LEVEL	
192	QUERY GROUPS 0-7	
193	QUERY GROUPS 8-15	
194	QUERY RANDOM ADDRESS (H)	
195	QUERY RANDOM ADDRESS (M)	
196	QUERY RANDOM ADDRESS (L)	
197	READ MEMORY LOCATION	
198-223	RESERVED COMMANDS	
224-254	APPLICATION EXTENDED COMMANDS	IEC 62386-201 supported
255	QUERY EXTENDED VERSION NUMBER	

Table 7. Special commands

Command nr	Description	Remarks
256	TERMINATE	
257	DATA TRANSFER REGISTER	
258	INITIALISE	
259	RANDOMISE	
260	COMPARE	
261	WITHDRAW	
262-263	RESERVED COMMANDS	
264	SEARCH ADDRESS H	
265	SEARCH ADDRESS M	
266	SEARCH ADDRESS L	
267	PROGRAM SHORT ADDRESS	
268	VERIFY SHORT ADDRESS	
269	QUERY SHORT ADDRESS	
270	PHYSICAL SELECTION	By means of IO (PIO0_7)
271	RESERVED	
272	ENABLE DEVICE TYPE X	
273	DATA TRANSFER REGISTER 1	
274	DATA TRANSFER REGISTER 2	
275	WRITE MEMORY LOCATION	Read only memory bank 0 supported

3.8 NVM storage

Each DALI ballast has a series of properties called 'variables' (see table 6 of IEC 62386-102) e.g. actual level, power level, min/max level, short address etc. These properties are persistently stored in the nonvolatile flash memory of the microcontroller. For this, sector 7 of the flash memory is used as storage.

All nonvolatile variables are stored in one record of 256 bytes in the flash sector. When one of the variables is changed, the complete record is stored in the flash sector at the moment that the BOD_IRQ triggers, just before power down. This prevents the DALI device from not responding to DALI frames; during flash erase/write no other tasks can be done for a short timeframe on the processor core of the microcontroller. As the record is relatively small in respect to the sector size, multiple records fit in one sector. The software routines in *nvm.c* ensure that only the most recent record is used. The record contains the firmware version number. As the contents of the flash sector are 0xFF after an erase, it is guaranteed that the software always finds the most recent record when searching downwards from the highest address of the flash sector 7. Nonvolatile records are stored incremental from the start of the sector.

When using multiple DALI addresses in one physical device (e.g. for a RGB lamp) all variables of each logical DALI address are stored separately.

3.9 Configuration options

To simplify the usage of the software, several pre-configured ballast configurations (LAMP_TYPE) are included. The ballast configurations vary the numbers of DALI devices in the slave, the frequency of the PWM signals, and enable usage of the on/off signal. These configurations can be set in the file *app_config.h*. Only one configuration can be active at any time.

Table 8. Pre-defined ballast configuration options

Configuration (LAMP_TYPE)	# control gear	Ballast drive		
		PWM signals		On/off signal
		Frequency (Hz)	Inverted	
CFL	1	2197	No	Yes
SSL1523	1	300	Yes	No
UBA3070_1CH	1	733	Yes	No
UBA3070_4CH	4	733	Yes	No
DRIVER_3CH	3	733	No	No

The options of Table 8 set several ballast and communication configurations. A more elaborate overview of the configuration options is given in Table 9.

Table 9. Additional configuration options

Configuration	Description	Value	File
LPC_CORE_CLOCKSPEED_HZ	CPU clock frequency	48000000	app_config.h
MAJOR_FIRMWARE_VERSION_NR	Control gear major version number	2	app_config.h
MINOR_FIRMWARE_VERSION_NR	Control gear minor version number	0	app_config.h
GTIN_BYTE_X	Global Trade Item Number	0	app_config.h

Configuration	Description	Value	File
LAMP_TYPE	Lamp type	UBA3070_1CH	app_config.h
NUM_DALI_DEVICES	Number of supported Dali devices in control gear	1	app_config.h
NR_OF_DIM_CURVES	Number of used DIM curve tables	1	app_config.h
PWM_OUTPUT_FREQUENCY_HZ	Frequency of output PWM1 till PWM4	733	app_config.h
GPIO_ON_OFF	Usage of PIO0_3	disabled	app_config.h
DALI_PHYSICAL_MIN_LEVEL	Minimum arc power level	1	app_config.h
TE	Bit length in microseconds	417	DALI_PhysDriver.c
MIN_TE	Minimal Te length	300	DALI_PhysDriver.c
MAX_TE	Maximal Te length	550	DALI_PhysDriver.c
MIN_2TE	Minimal length for 1 bit	668	DALI_PhysDriver.c
MAX_2TE	Maximal length for 1 bit	1000	DALI_PhysDriver.c

The configuration options from the file *app_config.h* (described in Table 9) are passed to the DALI component via the *DALI_SETUP* structure in *main.c*. It is possible to use only one DIM curve table in a 3 (RGB) or 4 (RGBW) PWM output channel DALI Slave. The DIM Curve tables included in this DALI SDK apply to the PWM frequencies as described in Table 8 for a CPU clock frequency of 48 MHz.

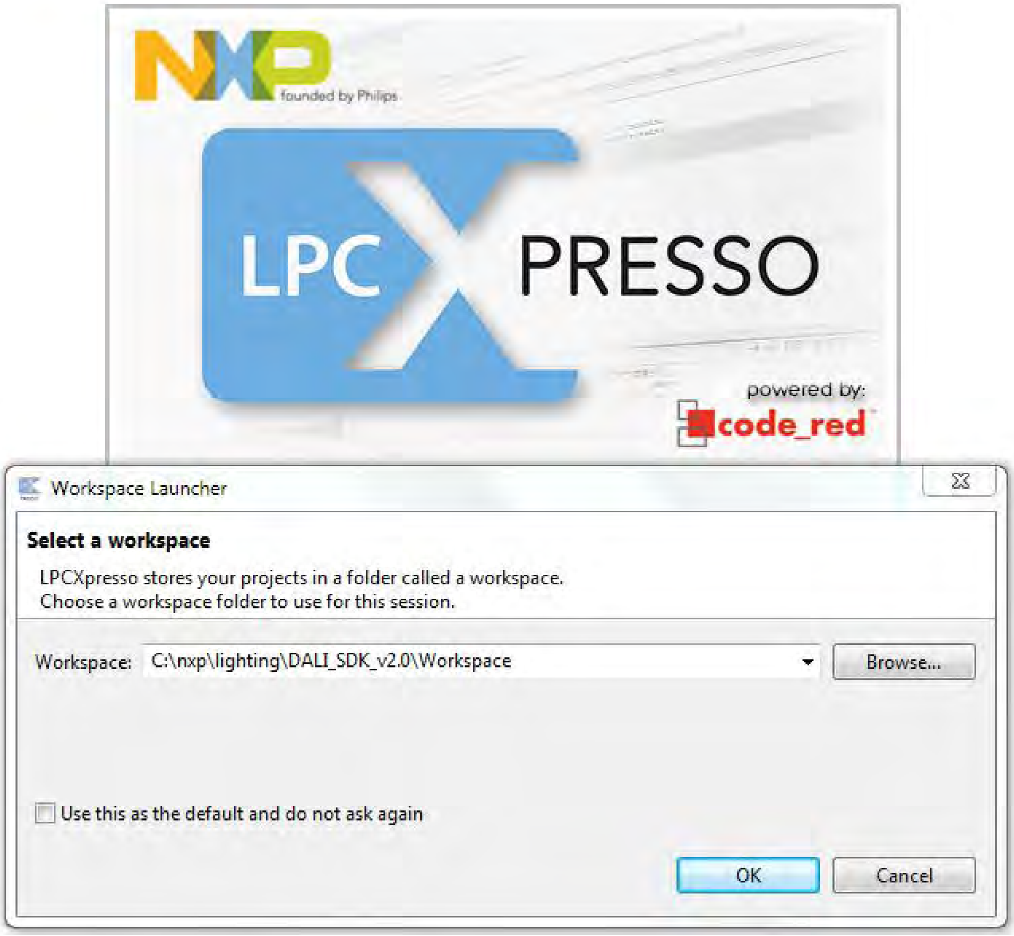
The DALI Slave supports memory bank 0, address 0x0 till address 0xE. Memory bank 0 contains the 6 bytes Global Trade Information Number (GTIN), a 2 bytes firmware version number of the control gear, and a four bytes serial number (see table 3 of IEC 62386-102).

3.10 Building the software

The software tree includes project files for

- LPCXpresso v5.0.10_1066, and
- IAR Embedded Workbench for ARM v6.40

When using LPCXpresso for building the DALI slave, use the workspace location C:\nxp\lighting\DALI_SDK_v2.0\Workspace as shown in Fig 16.



Select a workspace

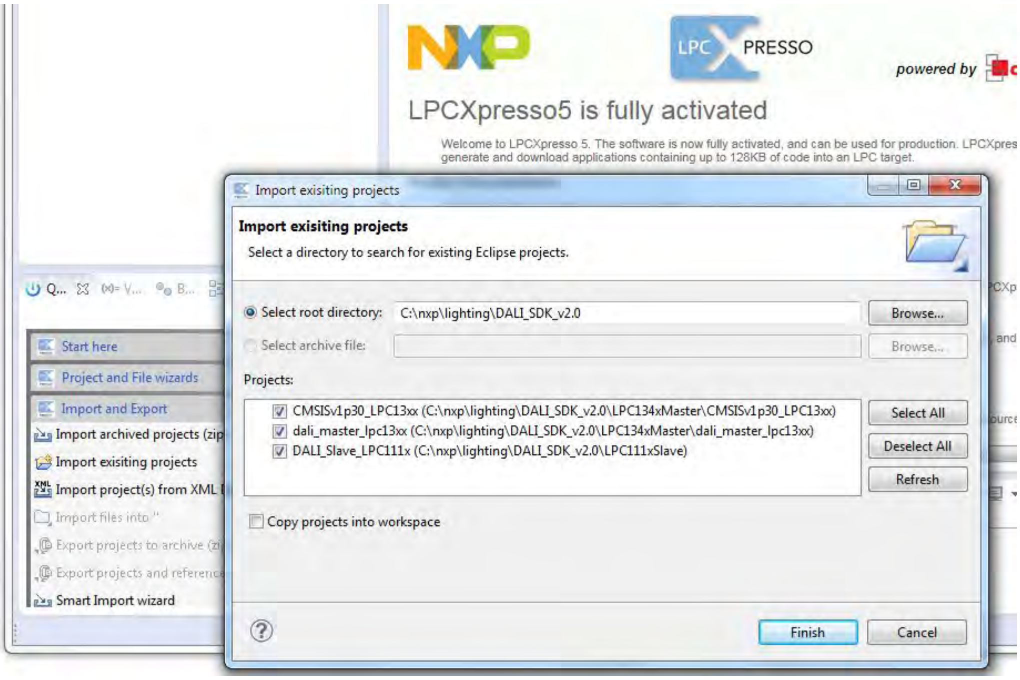
LPCXpresso stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\nxp\lighting\DALI_SDK_v2.0\Workspace

☐ Use this as the default and do not ask again

Fig 16. Start LPCXpresso and select the DALI SDK workspace folder

After clicking the OK button, open the panel “Import and Export” from the Quickstart Panel in the lower left corner of the LPCXpresso window. From this panel use the quick link “*Import existing projects*” and select as root folder the SDK installation path. Make sure to uncheck the tick mark “*Copy projects into workspace*” and select the projects that are shown in Fig 17. Click on the Finish button to import the selected projects.



LPCXpresso5 is fully activated

Welcome to LPCXpresso 5. The software is now fully activated, and can be used for production. LPCXpres
generate and download applications containing up to 128KB of code into an LPC target.

Import exisiting projects

Select a directory to search for existing Eclipse projects.

● Select root directory: C:\nxp\lighting\DALI_SDK_v2.0

○ Select archive file:

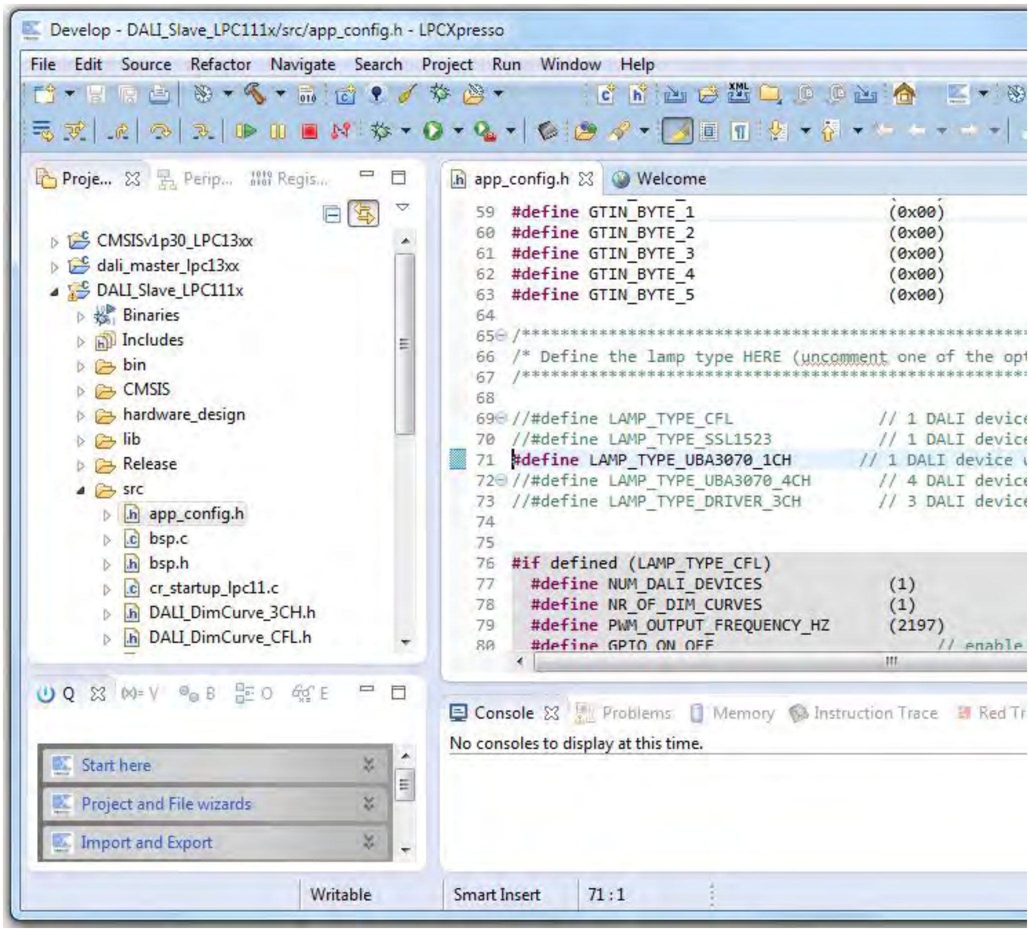
Projects:

☑ CMSISv1p30_LPC13xx (C:\nxp\lighting\DALI_SDK_v2.0\LPC134xMaster\CMSISv1p30_LPC13xx)
☑ dali_master_lpc13xx (C:\nxp\lighting\DALI_SDK_v2.0\LPC134xMaster\dali_master_lpc13xx)
☑ DALI_Slave_LPC111x (C:\nxp\lighting\DALI_SDK_v2.0\LPC111xSlave)

☐ Copy projects into workspace

Fig 17. Importing projects for DALI slave

Before building the software make sure that it is configured correctly. Open the file *app_config.h* and uncomment the LAMP_TYPE definition that matches your requirement, as shown in Fig 18.



```
#define GTIN_BYTE_1                         (0x00)
#define GTIN_BYTE_2                         (0x00)
#define GTIN_BYTE_3                         (0x00)
#define GTIN_BYTE_4                         (0x00)
#define GTIN_BYTE_5                         (0x00)

/*****************************************************
/* Define the lamp type HERE (uncomment one of the opt
/*****************************************************

//#define LAMP_TYPE_CFL                    // 1 DALI device
//#define LAMP_TYPE_SSL1523                // 1 DALI device
#define LAMP_TYPE_UBA3070_1CH            // 1 DALI device u
//#define LAMP_TYPE_UBA3070_4CH            // 4 DALI device
//#define LAMP_TYPE_DRIVER_3CH             // 3 DALI device


#if defined (LAMP_TYPE_CFL)
  #define NUM_DALI_DEVICES                  (1)
  #define NR_OF_DIM_CURVES                  (1)
  #define PWM_OUTPUT_FREQUENCY_HZ           (2197)
  #define GPIO_ON_OFF                            // enable
```

No consoles to display at this time.

Fig 18. Configure DALI slave

Now select the desired build configuration (Debug/Release build), as shown in Fig 19.

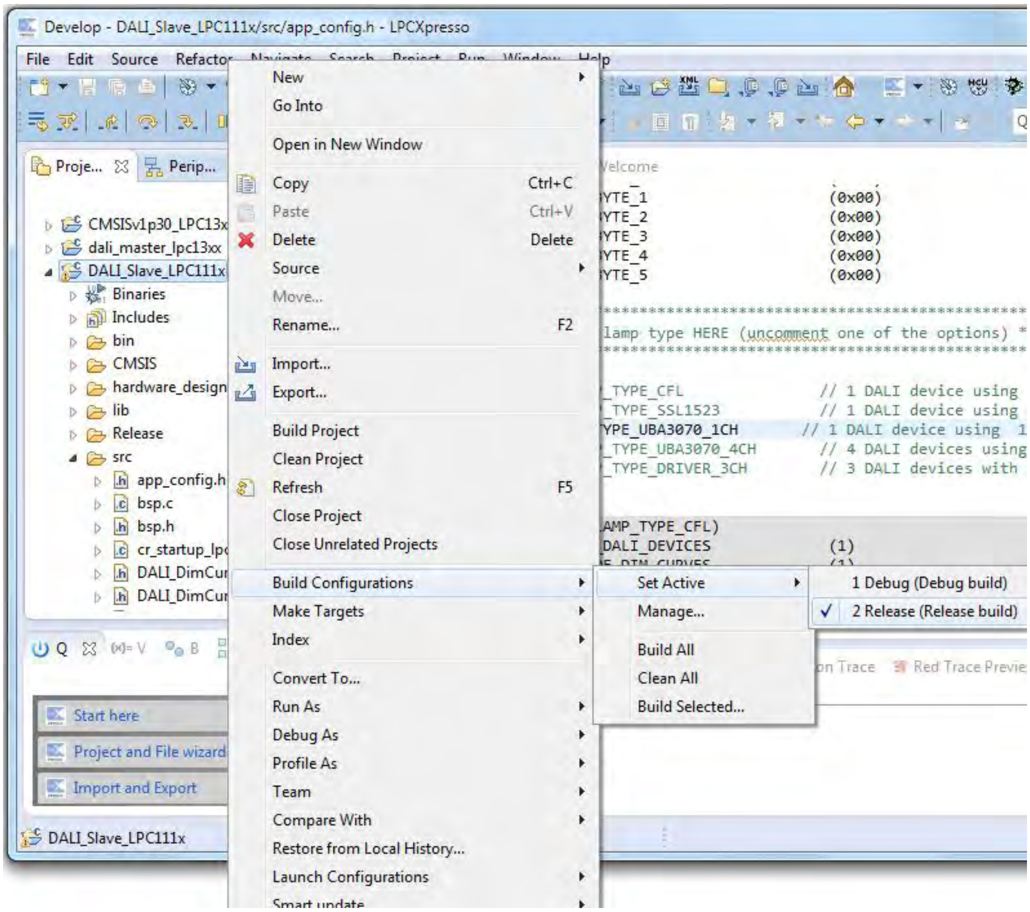


Fig 19. Build configuration of DALI slave

Now start the build process as shown in Fig 20.

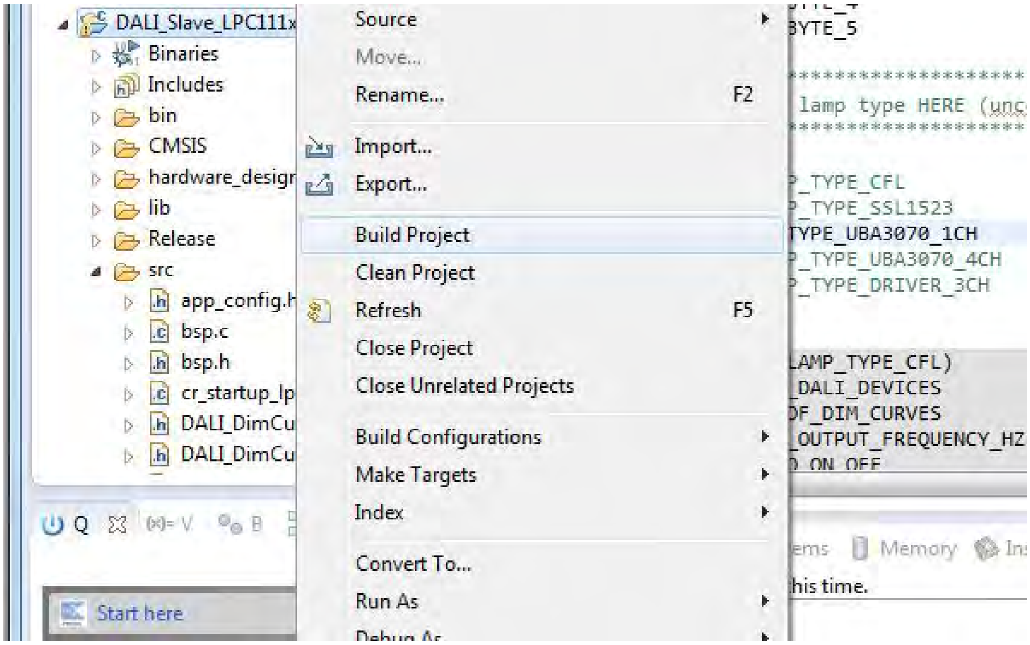
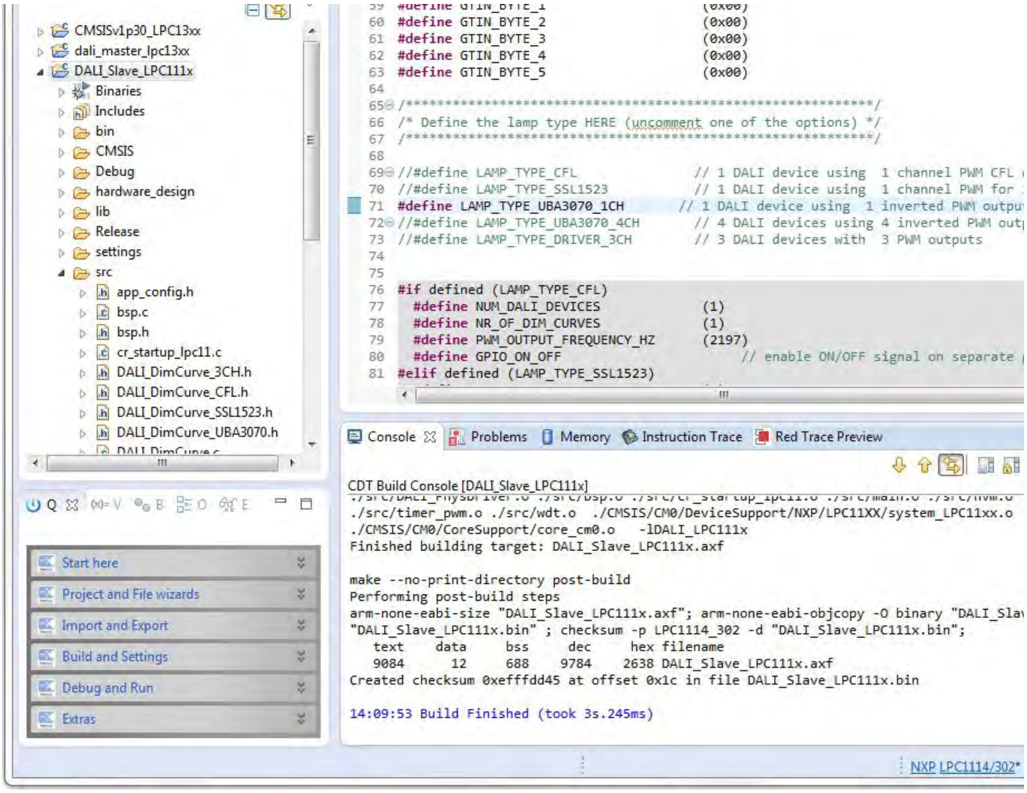


Fig 20. Start build of DALI Slave

The result should look like Fig 21.



```
#define GTIN_BYTE_2                           (0x00)
#define GTIN_BYTE_3                           (0x00)
#define GTIN_BYTE_4                           (0x00)
#define GTIN_BYTE_5                           (0x00)

/*************************************************************/
/* Define the lamp type HERE (uncomment one of the options) */
/*************************************************************/

//#define LAMP_TYPE_CFL                    // 1 DALI device using  1 channel PWM CFL
//#define LAMP_TYPE_SSL1523                // 1 DALI device using  1 channel PWM for
#define LAMP_TYPE_UBA3070_1CH            // 1 DALI device using  1 inverted PWM outpu
//#define LAMP_TYPE_UBA3070_4CH            // 4 DALI devices using 4 inverted PWM out
//#define LAMP_TYPE_DRIVER_3CH             // 3 DALI devices with  3 PWM outputs


#if defined (LAMP_TYPE_CFL)
  #define NUM_DALI_DEVICES                  (1)
  #define NR_OF_DIM_CURVES                  (1)
  #define PWM_OUTPUT_FREQUENCY_HZ           (2197)
  #define GPIO_ON_OFF                                // enable ON/OFF signal on separate
#elif defined (LAMP_TYPE_SSL1523)
```

CDT Build Console [DALI_Slave_LPC111x]

```
./src/timer_pwm.o ./src/wdt.o  ./CMSIS/CM0/DeviceSupport/NXP/LPC11XX/system_LPC11xx.o
./CMSIS/CM0/CoreSupport/core_cm0.o   -lDALI_LPC111x
Finished building target: DALI_Slave_LPC111x.axf

make --no-print-directory post-build
Performing post-build steps
arm-none-eabi-size "DALI_Slave_LPC111x.axf"; arm-none-eabi-objcopy -O binary "DALI_Sla
"DALI_Slave_LPC111x.bin" ; checksum -p LPC1114_302 -d "DALI_Slave_LPC111x.bin";
   text    data     bss     dec     hex filename
   9084      12     688    9784    2638 DALI_Slave_LPC111x.axf
Created checksum 0xefffdd45 at offset 0x1c in file DALI_Slave_LPC111x.bin

14:09:53 Build Finished (took 3s.245ms)
```

Fig 21. Build result of DALI slave

Table 10 shows the size of the firmware in Release mode for different configurations of the software. Note that this table does not contain the size of the flash sector used for NVM storage of DALI device parameters, which is 4 kB on the LPC111x device, and also does not contain the stack size stored in RAM, which is 320 bytes.

Table 10. DALI slave firmware size

Configuration	# control gear	Firmware size		
		Flash (bytes)		RAM (bytes)
		LPCXpresso v5.0.10	IAR v6.40	
CFL	1	9108	7090 + 562	700
SSL1523	1	9080	7066 + 562	700
UBA3070_1CH	1	9084	7070 + 562	700
UBA3070_4CH	4	10644	7102 + 2090	700
DRIVER_3CH	3	10120	7086 + 1582	700

Using Table 10 it can be seen that the LPC1112 with 16 kB of Flash and 2 kB of RAM is sufficient for creating a DALI slave device.

4. Conclusion

This application note illustrates how to create the communication part of a DALI control gear, using evaluation board OM13026 from NXP Semiconductors. It shows an example implementation of a DALI bus physical interface and how to use a timer capture unit of the LPC111x microcontroller to receive and transmit DALI bus frames.

The DALI software that is responsible for the DALI command execution as specified in IEC62386-102 and -201 is described; the relation between the software components, the main execution flow and the coupling to the hardware peripherals of the LPC111x Cortex M0 microcontroller.

All the design information as schematic, Gerber files, software code is part of the DALI SDK 2.0 installer. Project files for the IDE toolsets LPCXpresso and IAR Embedded Workbench for ARM are included in the release.

The resulting firmware image leaves headroom for other additional tasks on the 32-bit Cortex-M0 microcontroller. The current design fits within the memory size of a LPC1112 device from the LPC111x series.

The DALI SDK development kit furthermore includes AN11175: a USB to DALI interface using the LPC1343, and a Getting started guide in UM10553.

5. Document management

5.1 Abbreviations

Table 11. Abbreviations

Acronym	Description
BOD	Brown Out Detection
BSP	Board Support Package
CMSIS	Cortex Microcontroller Software Interface Standard
CPU	Central Processing Unit
CT	Counter Timer
DALI	Digital Addressable Lighting Interface
GPIO	General Purpose Input/Output
HW	Hardware
IDE	Integrated Development Environment
IRQ	Interrupt Request
ISR	Interrupt Service Routine
LED	Light Emitting Diode
MCU	Micro Controller Unit
NVM	NonVolatile Memory
PC	Personal Computer
PCB	Printed Circuit Board
PIO	Input/Output Pin
PLL	Phase Locked Loop
PWM	Pulse Width Modulation
SW	Software
USB	Universal Serial Bus
WDT	Watchdog Timer

5.2 References

[1] IEC62386-101: Digital addressable lighting interface, General requirements – system, Edition 1.0 2009-6, IEC, 2009.

[2] IEC62386-102: Digital addressable lighting interface, General requirements – control gear, Edition 1.0 2009-6, IEC, 2009.

[3] http://www.dali-ag.org/

[4] UM10398: LPC111x/LPC11Cxx User manual, Rev. 12, NXP, 24 September 2012.

6. Legal information

6.1 Definitions

6.2 Disclaimers

6.3 Trademarks

7. List of figures

8. List of tables

9. Contents

Date of release: 6 March 2013
Document identifier: AN11174