XC800 Family

AP08102

DALI Control Gear Software Stack

Application Note

V1.1, 2010-09

Microcontrollers

Edition 2010-09

Published by
Infineon Technologies AG
81726 Munich, Germany

LEGAL DISCLAIMER

THE INFORMATION GIVEN IN THIS APPLICATION NOTE IS GIVEN AS A HINT FOR THE IMPLEMENTATION OF THE INFINEON TECHNOLOGIES COMPONENT ONLY AND SHALL NOT BE REGARDED AS ANY DESCRIPTION OR WARRANTY OF A CERTAIN FUNCTIONALITY, CONDITION OR QUALITY OF THE INFINEON TECHNOLOGIES COMPONENT. THE RECIPIENT OF THIS APPLICATION NOTE MUST VERIFY ANY FUNCTION DESCRIBED HEREIN IN THE REAL APPLICATION. INFINEON TECHNOLOGIES HEREBY DISCLAIMS ANY AND ALL WARRANTIES AND LIABILITIES OF ANY KIND (INCLUDING WITHOUT LIMITATION WARRANTIES OF NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OF ANY THIRD PARTY) WITH RESPECT TO ANY AND ALL INFORMATION GIVEN IN THIS APPLICATION NOTE.

Information

For further information on technology, delivery terms and conditions and prices, please contact the nearest Infineon Technologies Office (**www.infineon.com**).

Warnings

Due to technical requirements, components may contain dangerous substances. For information on the types in question, please contact the nearest Infineon Technologies Office.

Infineon Technologies components may be used in life-support devices or systems only with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

XC82x
Revision History: V1.1 2010-09

Previous Version(s): V1.0 2010-06

Page	Subjects (major changes since last revision)
12	Chapter 3.4.4 is updated.

Trademarks

TriCore® is a trademark of Infineon Technologies AG.

We Listen to Your Comments
Is there any information in this document that you feel is wrong, unclear or missing? Your feedback will help us to continuously improve the quality of this document. Please send your proposal (including a reference to this document) to:
mcdocu.comments@infineon.com

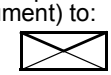

Table of Contents

1 Overview . . . 5

2 DALI Protocol . . . 7
2.1 Receiving DALI Forward Frame . . . 7
2.2 Sending Backward Frame . . . 7

3 DALI Control Gear Software Stack . . . 8
3.1 Software Structure . . . 8
3.2 Hardware Abstraction Layer Configuration . . . 8
3.2.1 GPIO Module . . . 8
3.2.2 Timer 0 and Timer 2 Modules . . . 9
3.2.2.1 Timer 0 . . . 9
3.2.2.2 Timer 2 . . . 9
3.3 DALI Protocol Module . . . 9
3.4 DALI Commands Handler Module . . . 10
3.4.1 Process Command . . . 10
3.4.2 Fading Control . . . 11
3.4.3 DALI Variables . . . 11
3.4.4 Memory Banks . . . 12

4 API Routines . . . 13
4.1 Arc Power Level . . . 13
4.2 Program DALI Variables . . . 13
4.3 Status of Lighting Device . . . 13
4.4 Light On . . . 13

5 Conditional Compilation Option . . . 14

6 Preparing Software Stack for LED Application . . . 15
6.1 Hardware Setup . . . 15
6.2 Hardware Abstraction Layer . . . 16
6.2.1 ADC Initialisation . . . 16
6.2.2 CCU6 Initialisation . . . 16
6.3 Application Layer . . . 17
6.4 Compiler Options and Linker Address Allocation . . . 17
6.5 Software Package . . . 17

7 Summary . . . 18

8 Acronyms Abbreviations and Special Terms . . . 18

9 References . . . 18

10 Appendix . . . 19
10.1 Flow Charts . . . 19
10.2 DALI Variables . . . 20

1 Overview

Digital Addressable Lighting Interface (DALI) is a communication protocol for lighting control in buildings. The interface was first described in Annex E, IEC60929 standard for fluorescent lamp ballast. Subsequently, it was updated to the new standard IEC-62386, to include other lighting devices, such as LED and HID for example. The standard for control interface of electronic control gears was published in June 2009. The standard for lighting control devices is scheduled to be published in 2012.

DALI requires only a pair of wires to form the bus for communication to all devices on a single DALI network. Each piece of operating equipment with a DALI interface can be communicated with, over DALI, individually. Using a bi-directional data exchange, a DALI controller can query and set the status of each connected lighting device. As a standalone system, DALI can be operated with a maximum of 64 devices. Alternatively, DALI can be used as a subsystem via DALI gateways for connection to building management systems.

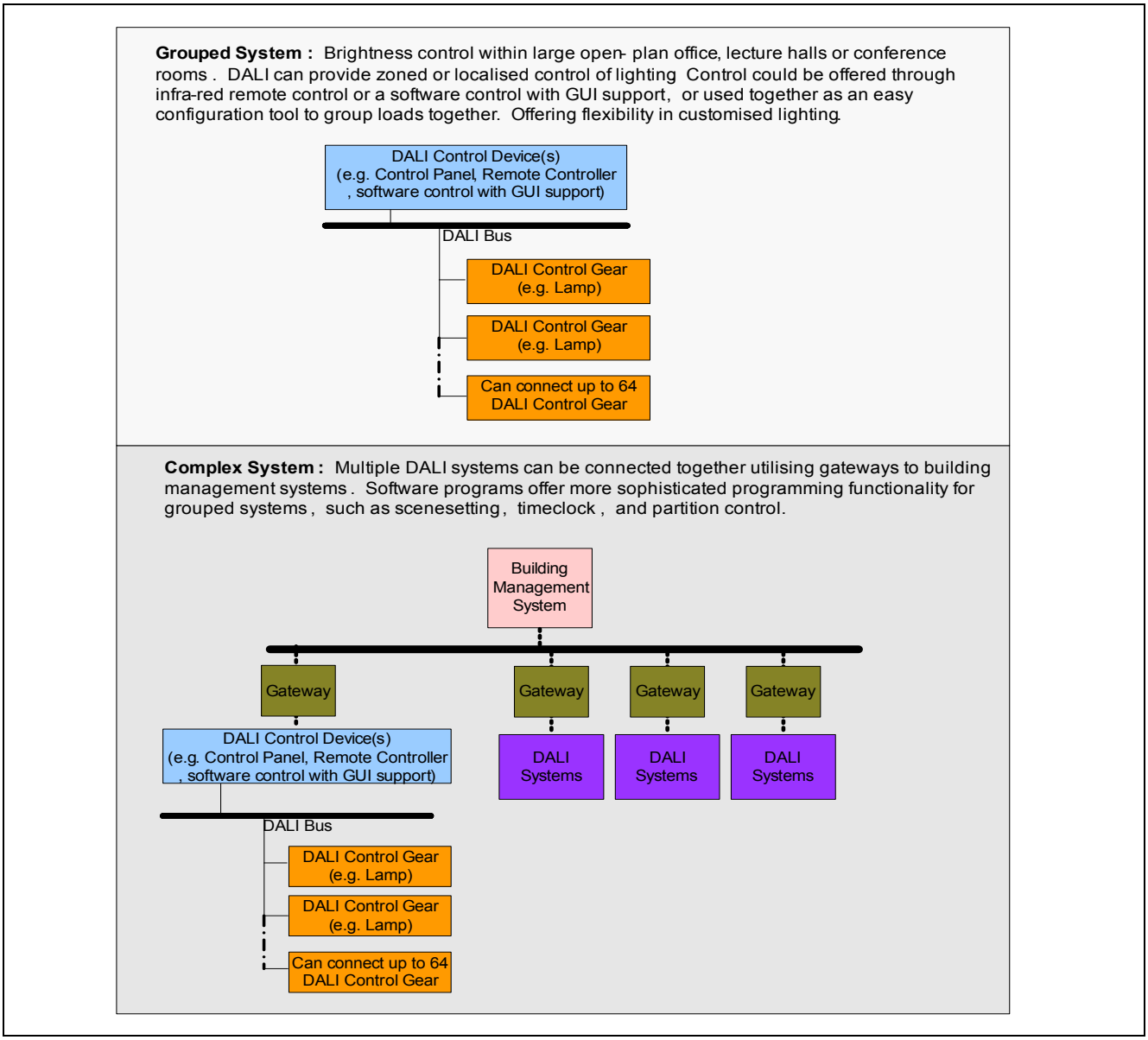

Figure 1 DALI System Types

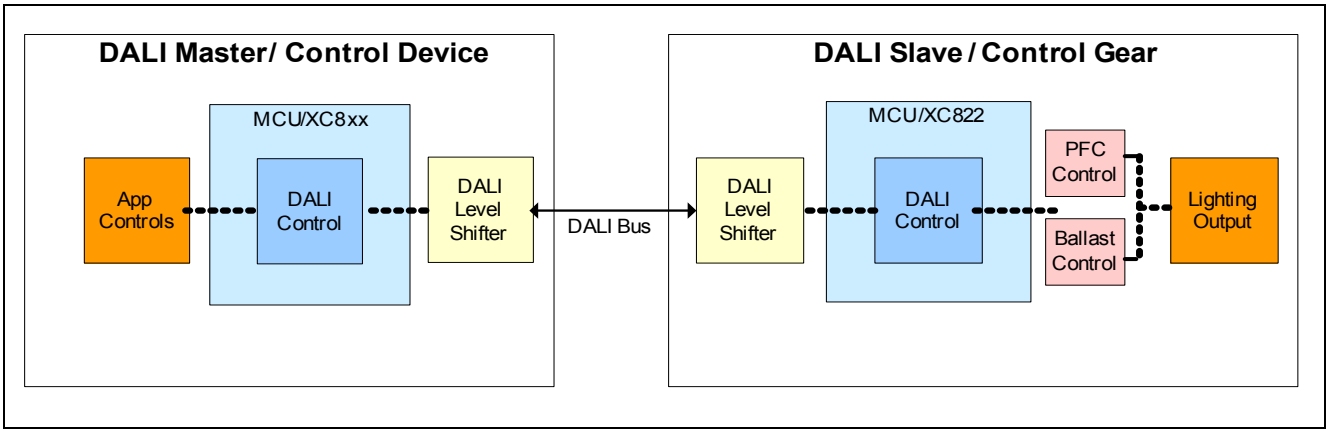

Figure 2 Block Diagram for DALI Control Device and Control Gear

This document describes the DALI Software Stack for Control Gear solution, created to provide an application example for LED control. This solution is based on the published IEC standard and is implemented with the Infineon XC82x microcontroller.

The following items are required for use with this application note:

- 1x XC82x Easy Kit

Figure 3 XC82x Easy Kit

Other application notes of interest are:

- AP08104: Guide to using DALI LightNet tool
- AP08105: DALI Demo using Touch Sense Control

2 DALI Protocol

DALI uses the Manchester encoded unidirectional serial protocol with a transmission rate of 1200bps ± 10%.

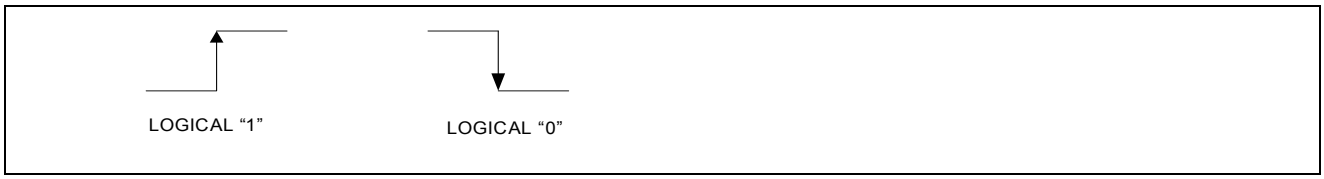

Figure 4 Bi-phase levels: ‘Logical 1’, ‘Logical 0’

2.1 Receiving DALI Forward Frame

Forward frame is the command frame received from the DALI master. It consists of 19-bits.

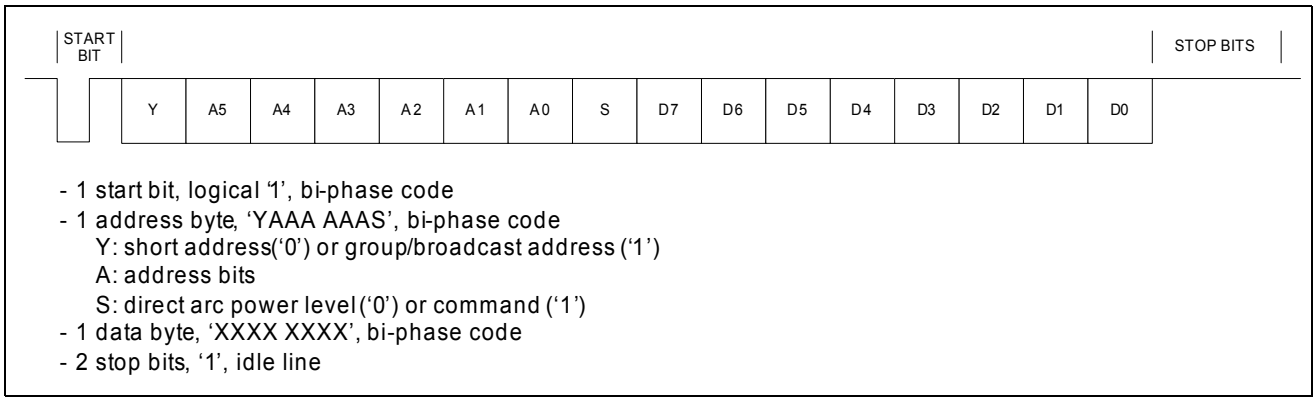

Figure 5 Forward Frame Format

2.2 Sending Backward Frame

The Backward frame is sent only after the reception of a query command or a write memory command. It consists of 11-bits.

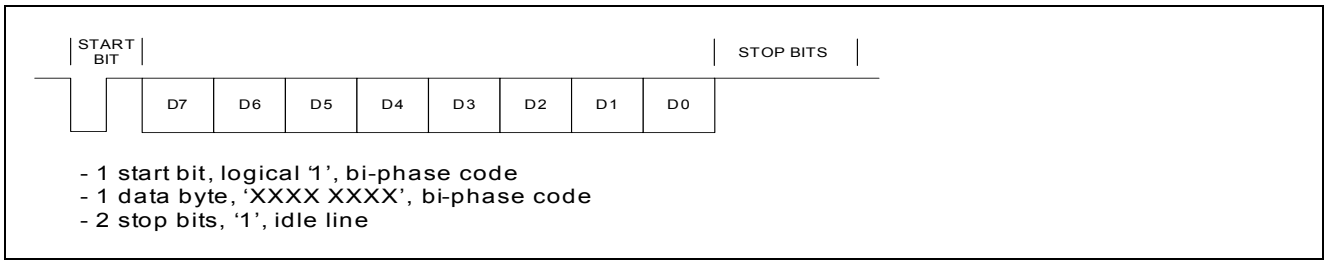

Figure 6 Backward Frame Format

3 DALI Control Gear Software Stack

This chapter describes the DALI Software Stack implemented for XC82x / XC83x devices.

3.1 Software Structure

Figure 7 shows the interface of the different software modules in the XC82x / XC83x lighting application software. It consists of peripheral modules, the DALI Software Stack, and an application layer.

The DALI Software Stack consists of the DALI protocol and commands handler.

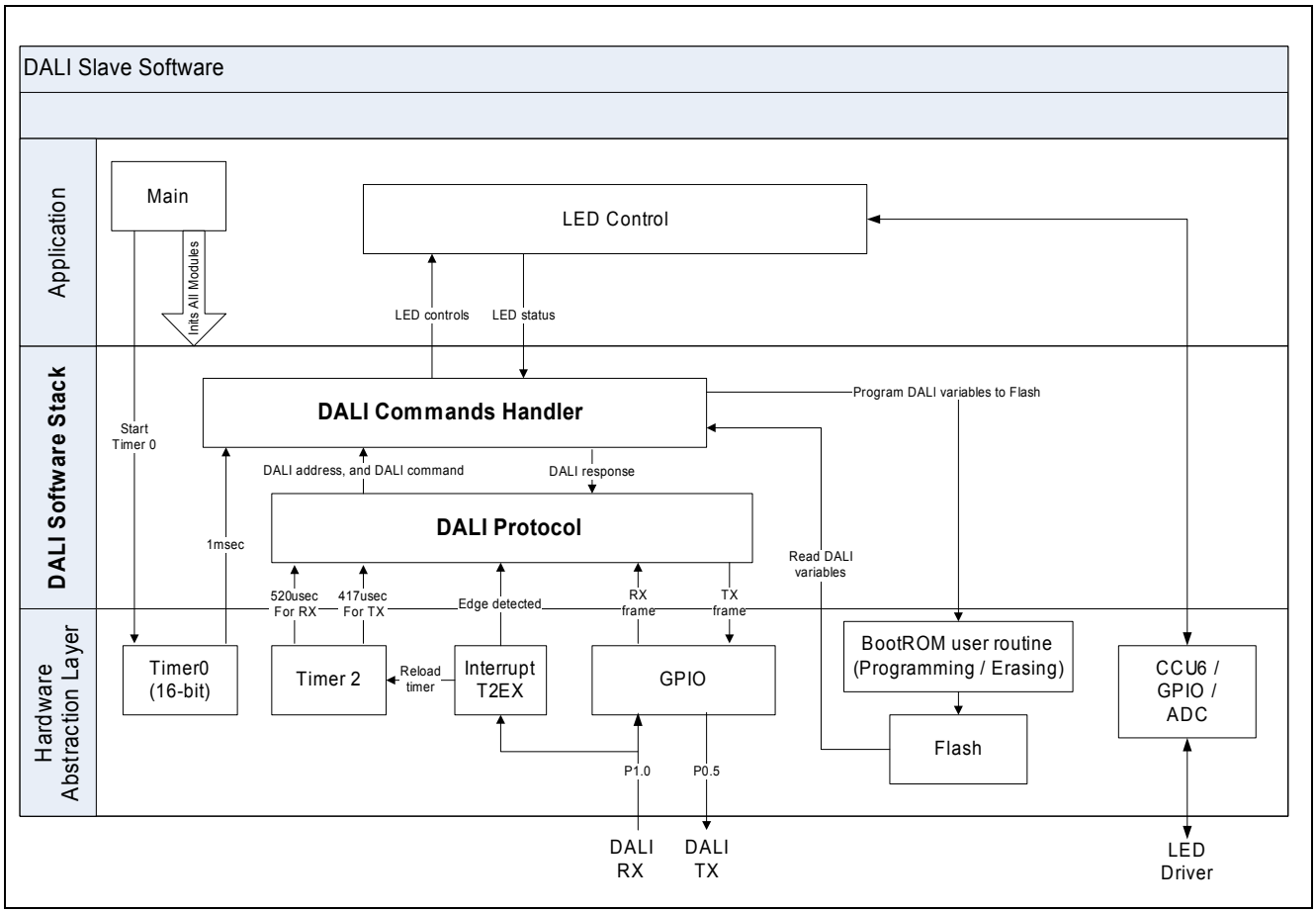

Figure 7 Software Structure for the XC82x / XC83x

3.2 Hardware Abstraction Layer Configuration

The config.h file stores the configuration of the GPIO and timers. The file can be modified according to the application requirements.

3.2.1 GPIO Module

The default GPIO settings for DALI receive and transmit are P1.0 and P0.5 respectively. For the DALI receive, T2EX (Timer 2 External pin), must be mapped to these pins. Depending on the application and package used, the GPIO for DALI interface can be re-configured to other port pins. Refer to **Table 1** for the GPIO pins that have T2EX as input function and can be used for DALI receive.

Table 1 DALI Receive GPIO pin selection

Package	GPIO can be used for DALI RX
TSSOP-28	P0.4, P0.6, P1.0, P2.0, P2.7, P3.2
DSO-24	P0.4, P0.6, P1.0, P2.0, P3.2
DSO-20	P0.4, P0.6, P1.0, P2.0
TSSOP-16	P0.4, P0.6, P1.0, P2.0

Default settings:

1. DALI Transmit is on P0.5

#define DALI_TX_PIN P0_5

#define READ_DALI_TX_PIN P0_5

2. DALI Receive is on P1.0

#define DALI_RX_DATAIN P1_DATAIN

#define READ_DALI_RX_PIN ACC_b0

#define T2EX_SEL 0x02

3.2.2 Timer 0 and Timer 2 Modules

The default settings for Timer 0 and Timer 2 are based on F_{PCLK} = 24MHz. If a different peripheral clock frequency is selected, the settings for Timer 0 and Timer 2 have to be changed accordingly.

3.2.2.1 Timer 0

The Timer 0 serves as a 1 msec tick. The main loop is run every 1 msec (see **Figure 13**). The Interrupt Service Routine carries out the timing countdown required in the Software Stack.

Default settings:

Timer 0 = 0xD120, 1 msec tick

3.2.2.2 Timer 2

The Manchester encoding and decoding of the DALI frame is performed in the DALI Software Stack with the use of the Timer 2 module. The Timer 2 external pin, T2EX, is connected to the DALI RX to detect the falling / rising edge of the forward frame.

Default settings:

Timer 2 = 0xC4F0, 520 usec for DALI Reception.

Timer 2 = 0xD8F0, 417 usec for DALI Transmission.

3.3 DALI Protocol Module

The ‘T2_viTmr2()’ function handles the decoding and encoding of the DALI frames.

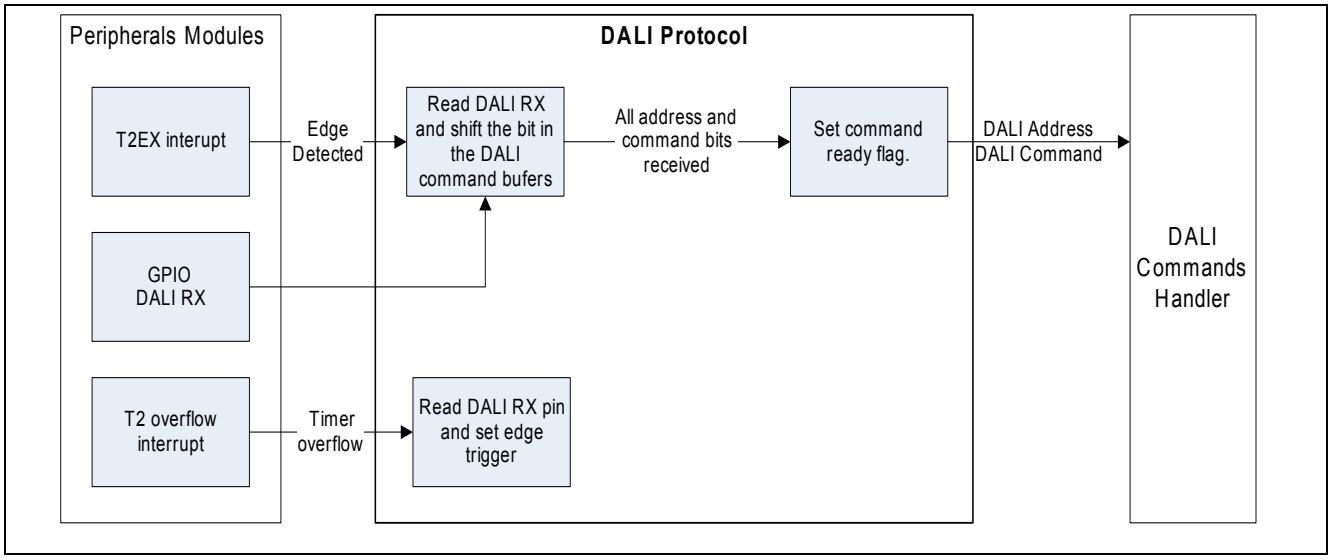

Figure 8 Decoding DALI Forward Frame

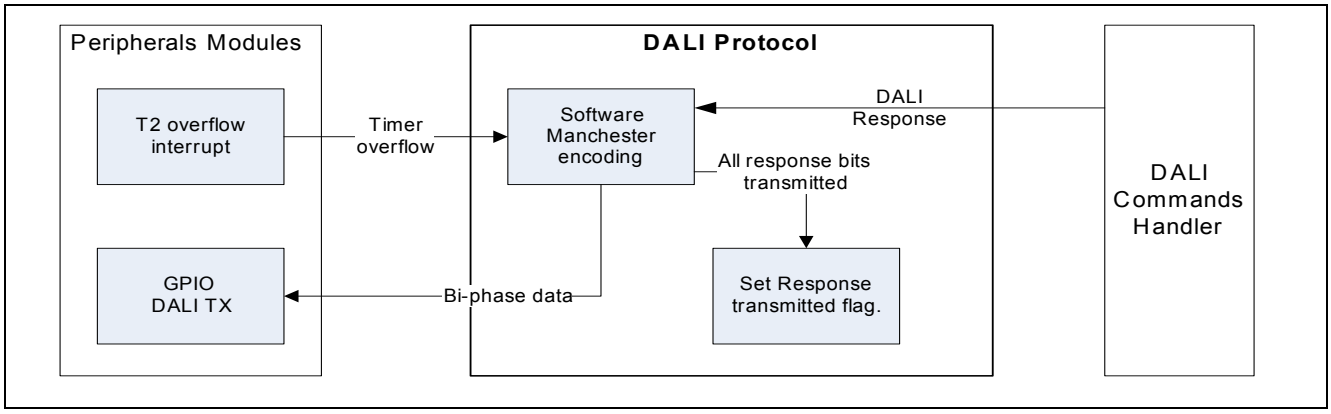

Figure 9 Encoding DALI Backward Frame

3.4 DALI Commands Handler Module

The main purpose of this module is to process the DALI address and command data from the DALI protocol module. This module contains application interface routines and DALI functions for fade control, arc power control, and read / write memory.

3.4.1 Process Command

The address byte of the DALI command is used to distinguish between a normal and a special command. This check is implemented in the ‘DALI_vProcess_cmd()’ function. **Figure 10** shows the flow of this function.

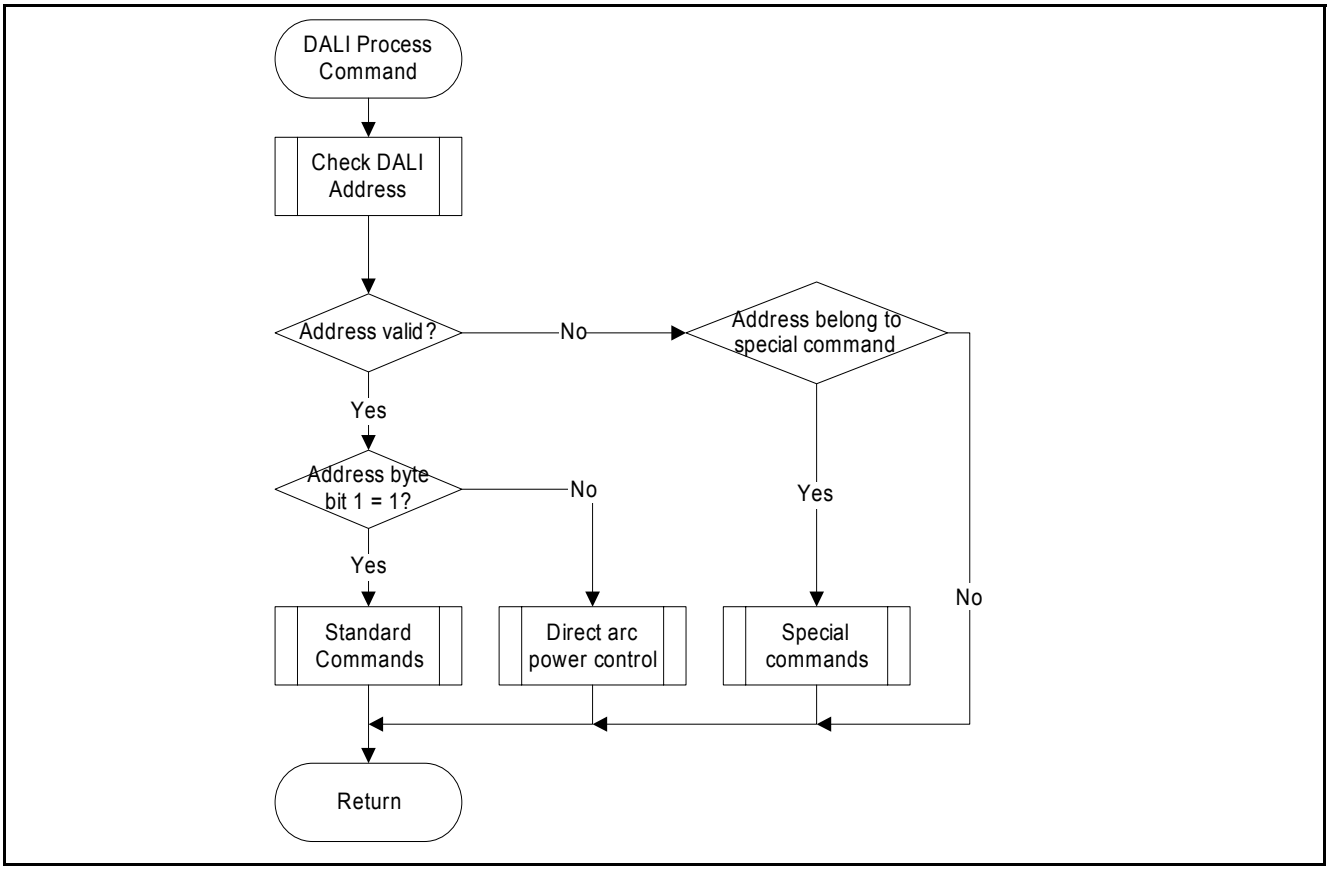

Figure 10 Process DALI command

3.4.2 Fading Control

According to the DALI standard there are 15 different fade times and fade rates that control dimming. The fade control is executed every 1 msec during the fading process. It updates the current arc power and stops the fading process once the target arc power is reached.

The fade time per step is calculated using the formula:

$$\text{Fade time per step} = \frac{\text{Fade Time}}{\text{Target Arc Power} - \text{Current Arc Power}}$$

To resolve the truncation error the formula is modified to:

$$\text{Fade time per 2 steps} = \frac{\text{Fade Time}}{(\text{Target Arc Power} - \text{Current Arc Power}) / 2} + 0.5$$

3.4.3 DALI Variables

There are 34 DALI variables defined in the standard (see **Table 11**). The data from these 34 variables is returned to the DALI master, when requested through the query commands. 29 of the 34 variables can be updated by the DALI master through the configuration and special commands.

3.4.4 Memory Banks

The DALI standard supports up to 256 memory banks, up to 256 bytes for each memory bank. Due to limited flash size, the implementation can support only 2 memory banks; Bank 0 and Bank 1.

For a device with a flash size of 4K bytes:

- Bank 0 is 32 bytes and with read-only access.
- Bank 1 is 95 bytes and with read / write access.

For a device with a flash size of 8K bytes:

- Bank 0 is 256 bytes and with read-only access.
- Bank 1 is 95 bytes and with read / write access.

The default data of Memory Bank 0 is defined in the file called “DALI_memory_bank0.c“. The default data of Memory Bank 1 is defined in the file called “DALI_Flash_Sect_default.c“.

Table 2 Memory Bank Address Allocation

Device	Flash Size	Memory Bank 0	Memory Bank 1 (Default)
XC82x / XC83x	4K Bytes	0x0EC0 to 0x0EDF (32 bytes)	0x0F20 to 0x0F7E (95 bytes)
XC83x	8K Bytes	0x1E00 to 0x1EFF (256 bytes)	0x1F20 to 0x1F7E (95 bytes)

At power-up, the device will retrieve the Memory Bank 1 data from the flash and place it into XRAM. Any changes to the Memory Bank 1 data requested by the DALI Master, will be carried out in XRAM. The routine Program_DALI_Variables() is to be called to copy the Memory Bank 1 data from the XRAM into the flash. This could be done before the device goes into power down mode or in the power-loss situation.

4 API Routines

The DALI Software Stack provides 4 API routines:

- DALI_arc_power() - return the required arc power level.
- PROGRAM_DALI_variables() - to program the DALI variables and Memory Bank 1 data to flash if they are updated by the DALI master.
- LIGHT_status() - to update the DALI Software Stack on the status of the lighting device.
- LIGHT_on() - to indicate that the lighting device is turned on.

4.1 Arc Power Level

The DALI Software Stack provides a routine that the user application can call to get the requested arc power level, DALI_arc_power(). If the return value is zero, the lamp/LED has to be turned off.

Table 3 Update arc power level to the Application layer

Routine	DALI_arc_power
Inputs	-
Return	8-bits data: 0, minimum arc power level - maximum arc power level

4.2 Program DALI Variables

This routine programs the updated DALI variables and Memory Bank 1 data, to the flash. If all the variables / data remain unchanged, there will be no programming action.

Table 4 Program DALI variables routine

Routine	PROGRAM_DALI_variables()
Inputs	-
Return	-
Resources used	32 bytes of IRAM for buffering data to be programmed
Execution Time	14 msec - Flash programming done. 2.4 usec - Flash programming not done.

4.3 Status of Lighting Device

If the lighting device is faulty or disconnected, this routine is to be called.

Table 5 Status of Lamp / LED

Routine	LIGHT_status()
Inputs	0 - Lighting Device OK 1 - Lighting Device faulty or disconnected
Return	-

4.4 Light On

This routine will set bit 2 of the DALI status byte, indicating that the lighting device is powered on. The user application needs to call this routine after the pre-heating and ignition process of the lighting device are complete.

Table 6 Light On

Routine	LIGHT_on()
Inputs	-
Return	-

5 Conditional Compilation Option

Four conditional compilation preprocessor functions are defined.

- **XC83X_8K** - Flash size is 8Kbytes and the Memory banks are in upper 4K.
- **SPECIAL_MODE_EN -** DALI special commands (commands 258 to 270) are supported.
- **MEMORY_BANK_EN** - Read Memory Map access commands supported (commands 197, 273, 274)
- **WRITE_MEMORY_EN** - Write to Memory Map commands supported (commands 275)

6 Preparing Software Stack for LED Application

This chapter describes an LED application example that uses the DALI Software Stack with the XC822 Easy Kit board from Infineon.

6.1 Hardware Setup

This section describes the hardware setup for this LED example. The XC822 Easy Kit board has a built in boost converter which is used to drive the LED. The PWM signal is generated using the CCU6 module and output through Port 1.2 to control the brightness of the LED.

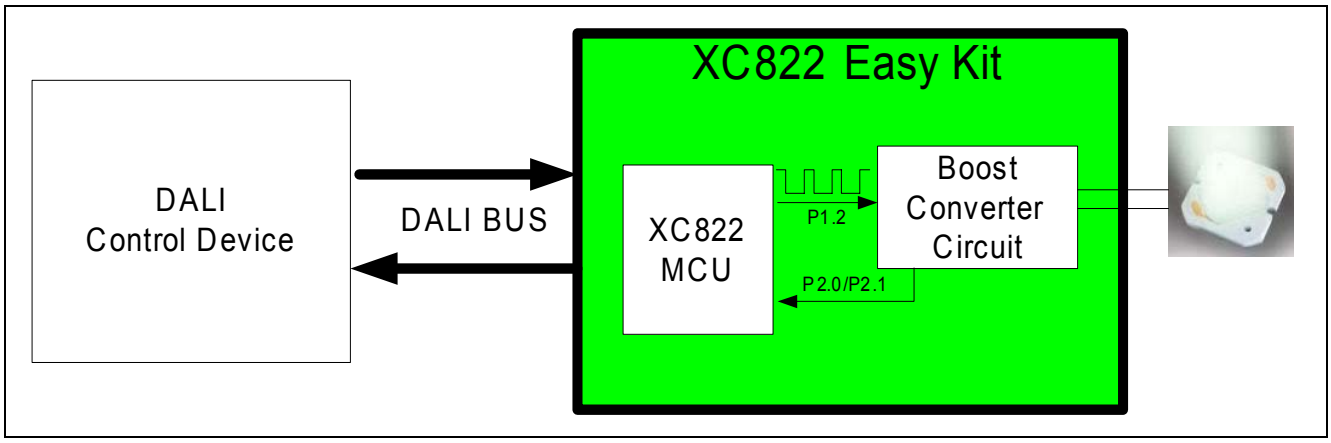

Figure 11 Overview of hardware setup

On the XC822 Easy Kit board, the jumper BC_EN is connected to enable the boost converter and the LED hardware module is connected to VOUT connector as shown in **Figure 12**.

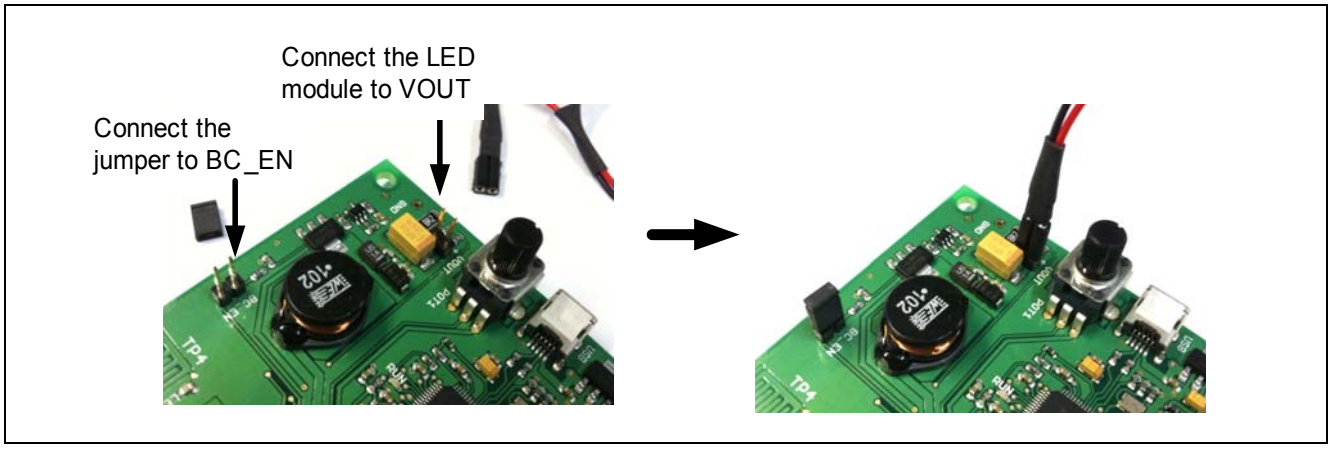

Figure 12 Connecting LED and boost converter jumper

Table 7 shows the XC822 MCU pins usage in this LED example on the Easy Kit board.

Table 7 XC822 MCU Pins Assignment

Pin Number	Port Pin	Pin Functionality
1	P0.5	DALI Transmit
2	P0.6	SPD Input / Output,
3	P2.3	-

Table 7 XC822 MCU Pins Assignment (cont'd)

Pin Number	Port Pin	Pin Functionality
4	P2.2	-
5	P2.1	ADC Channel 1, LED control voltage feedback
6	P2.0	ADC Channel 0, LED control current feedback
7	P1.0	DALI Receive
8	P1.2	CCU6 Channel 1 PWM output
9	V_{DDP}	I/O Port Supply
10	V_{SSP} / V_{SSC}	I/O Port Ground / Core Supply Ground
11	V_{DDC}	Core Supply Output
12	P0.0	-
13	P0.1	-
14	P0.2	-
15	P0.3	-
16	P0.4	-

6.2 Hardware Abstraction Layer

In the configuration file CONFIG.H, there are changes to the Timer 2 configuration, as shown in Table 8.

Table 8 Configuration of GPIO and Timers in CONFIG.H

Module	Description	Changes
Timer 0	Timer 0 overflow timing	No change
Timer 2	Timer 2 overflow timing	No change
	Define wakeup from power down mode timing	***#define***<*SPACE*>*PWR_DOWN_WAKEUP_TIME*<*SPACE*>***0x0960***
GPIO	Define DALI transmit and DALI receive pins	No change

6.2.1 ADC Initialisation

Two ADC channels are used, Channel 0 and Channel 1. Channel 0 is used to measure the LED current by measuring the analog voltage at AN0. Channel 1 is used to measure the LED voltage by using the ADC differential mode 1.

ADC settings:

- Conversion time = 0.83 usec
- Resolution = 10Bit
- Channel 0 setting
  - Input pin is AN0, P2.0
  - Reference is 1.2Volt + Vssp
- Channel 1 setting
  - Input pin is AN1, P2.1
  - Reference is 1.2V + AN0

6.2.2 CCU6 Initialisation

CCU6 module is used to generate PWM output to control the brightness of the LED.

- Channel 1 is used and output the PWM signal using COUT61 pin.
- PWM period is 66 usec
- Timer12 resolution is 0.021 usec

6.3 Application Layer

In this LED example code, the following features are added:

- Programming of the DALI variables when the V_{DDP} falls below 3.6V. The PROGRAM_DALI_variables() routine is called to program the DALI variables from RAM to flash.
- Enter power-down mode when the LED is turned off and after programming the DALI variables is complete.

The LED control is executed every 5 msec. The current arc power level is read by calling the DALI_arc_power() routine. This arc level together with the LED current feedback (using ADC channel 0) are input into the LED current control software algorithm to change the duty cycle of the PWM output.

6.4 Compiler Options and Linker Address Allocation

For this LED example code, the compiler options XC83X_8K, SPECIAL_MODE_EN, MEMORY_BANK_EN and WRITE_MEMORY_EN are not enabled. This is to fit into the 4K flash of the XC822 Easy Kit.

The address allocation for the DALI variables are defined in the linker file as shown in **Table 9**.

Table 9 Address Allocation in the Linker

Segments	Description	Address
DALI_FLASH_SECT_DEFAULT	DALI variables default values	0x0F01 to 0x0F1C
DALI_PARA	DALI variables reset values	0x0EE0 to 0x0EFF

6.5 Software Package

Table 10 lists the source files in the software package.

Table 10 List of source files

Files	Description
Start_xc.a51	Start up code for XC82x device. This is part of the compiler package
Main.c	Main loop and user application code to control the LED. Refer to **Figure 13**.
DALI_main.c	DALI Software Stack main function loop, refer to **Figure 14**.
DALI.c	DALI software module initialisation and DALI command handling.
DALI_API.c	Contains all the API routines described in **Chapter 4**.
T01.c	Timer 0 module initialisation and 1 msec interrupt service routine
T2	Timer 2 module initialisation and the interrupt service routine that decode or encode the DALI frame.
IO.c	GPIO module initialisation.
ADC.c	ADC module initialisation.
CC6.c	CCU6 module initialisation
PM.c	Power management module initialisation and power down mode entry function.
DALI_para.c	Defined the DALI parameter contents.
DALI_memory_bank0.c	Defined the memory bank 0 contents.

Table 10 List of source files (cont'd)

Files	Description
DALI_Flash_Sect_default.c	Defined the memory bank 1 contents.
config.h	Configuration of GPIO and definitions of timer overflow periods used in timer0 and timer2 modules.

7 Summary

The "DALI Control Gear Software Stack" solution has been developed in accordance with the IEC62386 standards. The description in this application note shows how the solution can be customised for an LED control application. From this example, users will be able to customise the Software Stack to their own lighting application.

8 Acronyms Abbreviations and Special Terms

List of terms and abbreviations used throughout the document:

- API Application Programming Interface
- GPIO General Purpose Input / Output
- MCU Microcontroller Unit
- T2EX Timer 2 External Pin

9 References

[1] IEC62386 Digital addressable lighting interface - Part 101: General requirements - System (Edition 1.0, 2009-06)

[2] IEC62386 Digital addressable lighting interface - Part 102: General requirements - Control gear (Edition 1.0, 2009- 06)

[3] XC82x User Manual version 1.0

[4] XC82x Easy Kit Hardware Manual XC822-TSSOP16 Board, version 1.0

10 Appendix

10.1 Flow Charts

Figure 13 and Figure 14 show the flow of the main loop and the DALI Software Stack.

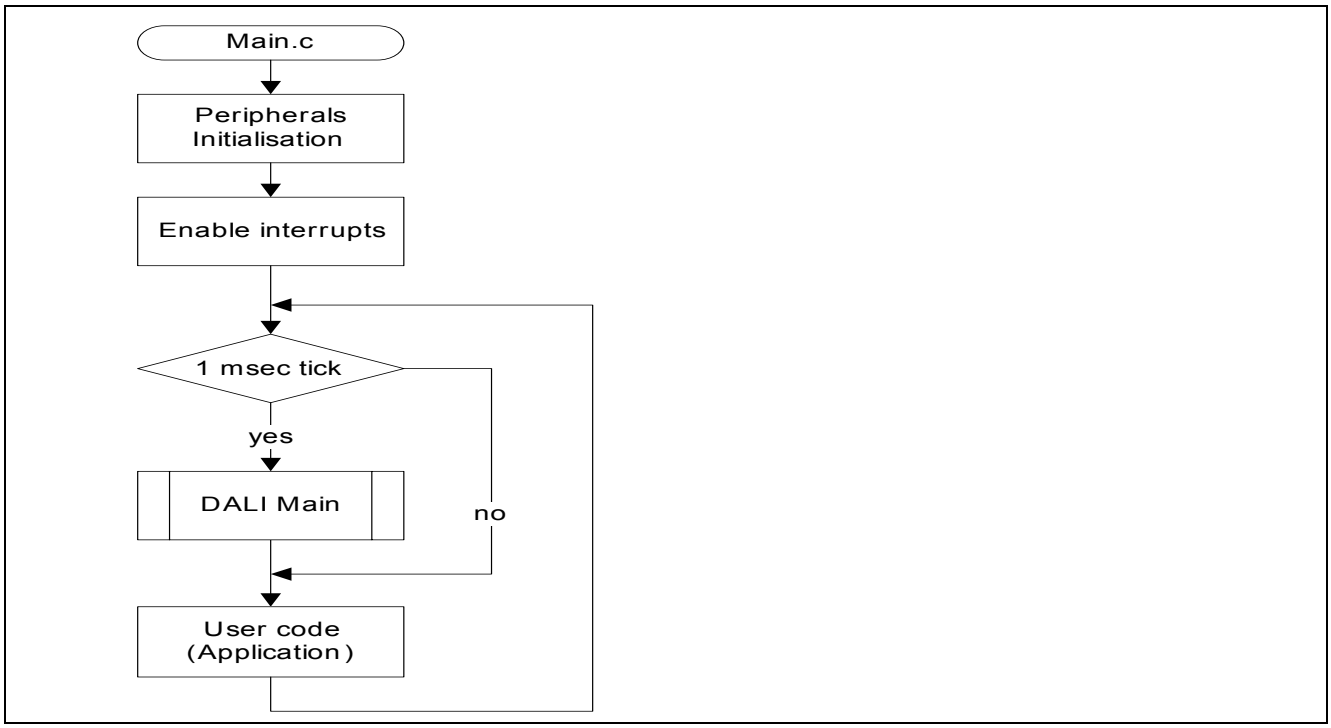

Figure 13 Software flow in main.c

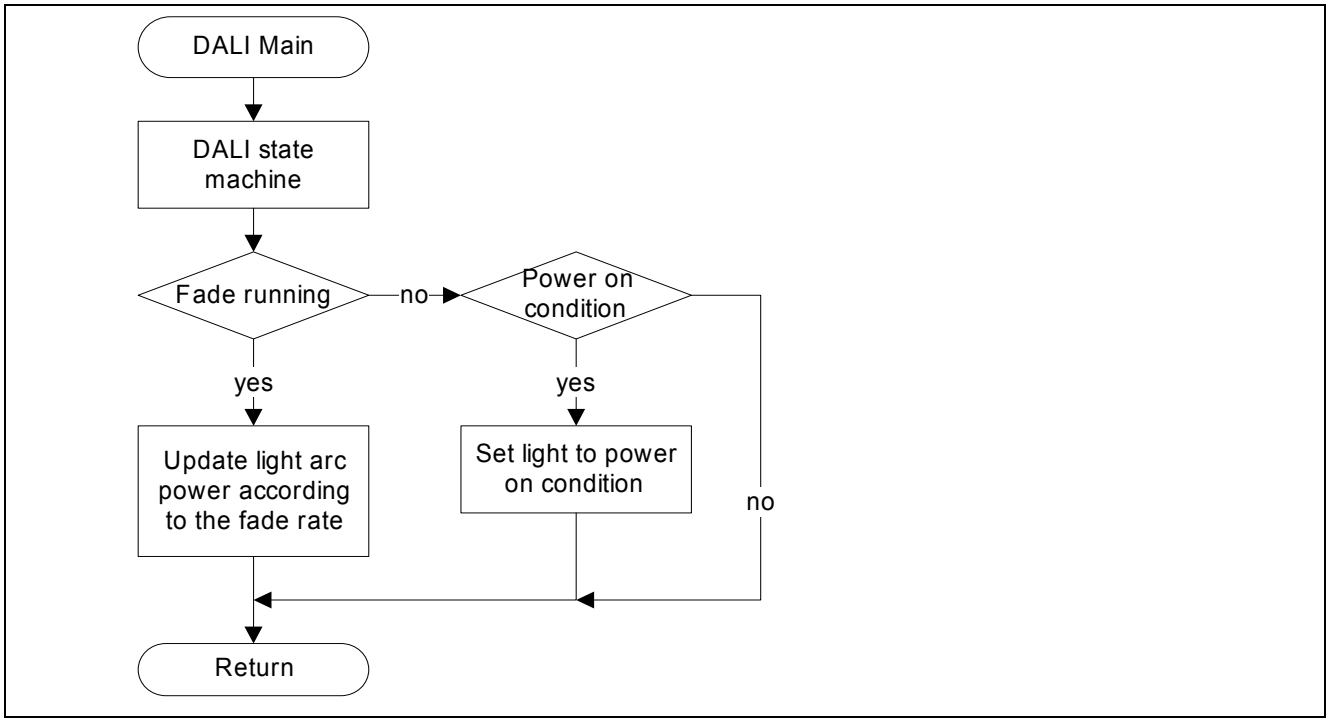

Figure 14 Software Flow in DALI_main.c

10.2 DALI Variables

List of variables defined in the DALI standard are shown in Table 11.

Table 11 Declaration of DALI Variables

Variables	Default values	Reset Values	Range of Validity	Memory
Actual level	Power On level	254	0, min to max	1 Byte RAM
Power On Level	254	254	0 to 255	1 Byte Flash
System Failure Level	254	254	0 to 255	1 Byte Flash
Minimum Level	Physical min	Physical min	Physical min to max	1 Byte Flash
Maximum Level	254	254	MIN to 254	1 Byte Flash
Fade Rate	7	7	1 to 15	1 Byte Flash
Fade Time	0	0	0 to 15	1 Byte Flash
Short Address	255	no change	0 to 63, 255	1 Byte Flash
Search Address	FF FF FFh	FF FF FFh	00 00 00h to FF FF FFh	3 Bytes RAM
Group 0 - 7	0000 0000b	0000 0000b	0 to 255	1 Byte Flash
Group 8 - 15	0000 0000b	0000 0000b	0 to 255	1 Byte Flash
Scene 0	255	255	0 to 255	1 Byte Flash
Scene 1	255	255	0 to 255	1 Byte Flash
Scene 2	255	255	0 to 255	1 Byte Flash
Scene 3	255	255	0 to 255	1 Byte Flash
Scene 4	255	255	0 to 255	1 Byte Flash
Scene 5	255	255	0 to 255	1 Byte Flash
Scene 6	255	255	0 to 255	1 Byte Flash
Scene 7	255	255	0 to 255	1 Byte Flash
Scene 8	255	255	0 to 255	1 Byte Flash
Scene 9	255	255	0 to 255	1 Byte Flash
Scene 10	255	255	0 to 255	1 Byte Flash
Scene 11	255	255	0 to 255	1 Byte Flash
Scene 12	255	255	0 to 255	1 Byte Flash
Scene 13	255	255	0 to 255	1 Byte Flash
Scene 14	255	255	0 to 255	1 Byte Flash
Scene 15	255	255	0 to 255	1 Bytes Flash
Status	1XX0 XXXXb	0X10 0XXXb	0 to 255	1 Byte RAM
DTR	undefined	no change	0 to 255	1 Byte RAM
DTR1	undefined	no change	0 to 255	1 Byte RAM
DTR2	undefined	no change	0 to 255	1 Byte RAM

w w w . i n f i n e o n . c o m

Published by Infineon Technologies AG