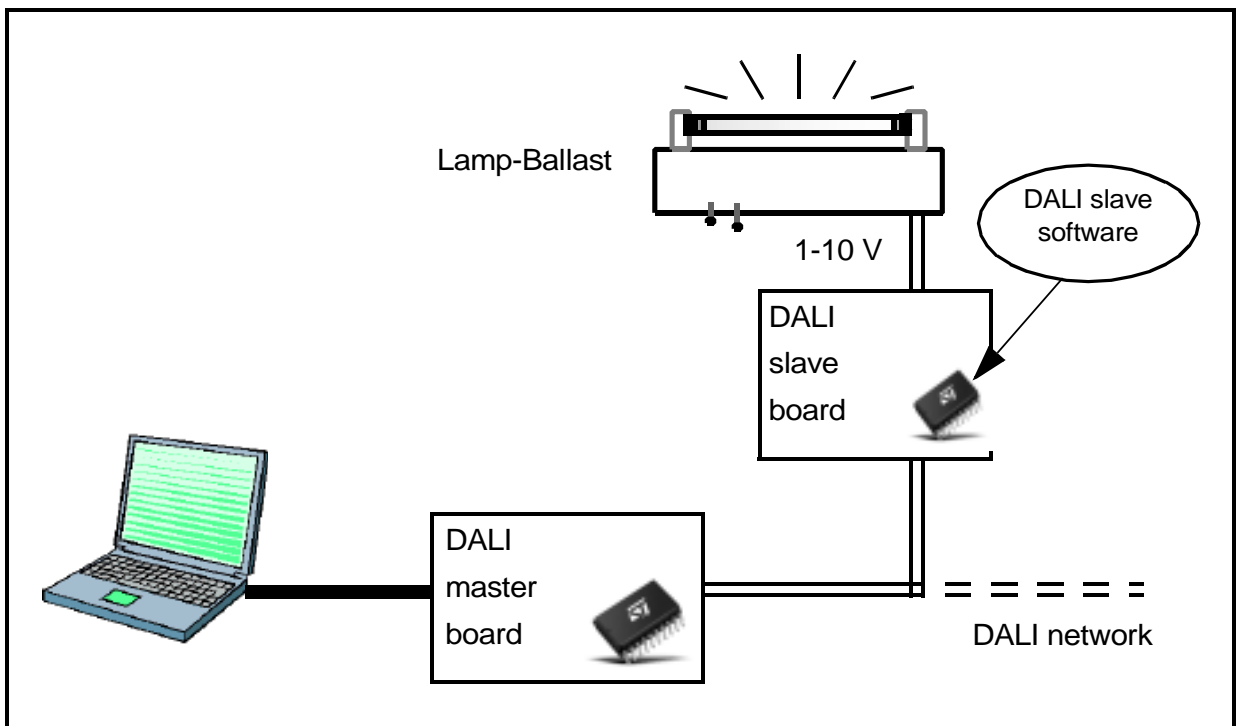


SOFTWARE IMPLEMENTATION FOR ST7DALI-EVAL

INTRODUCTION

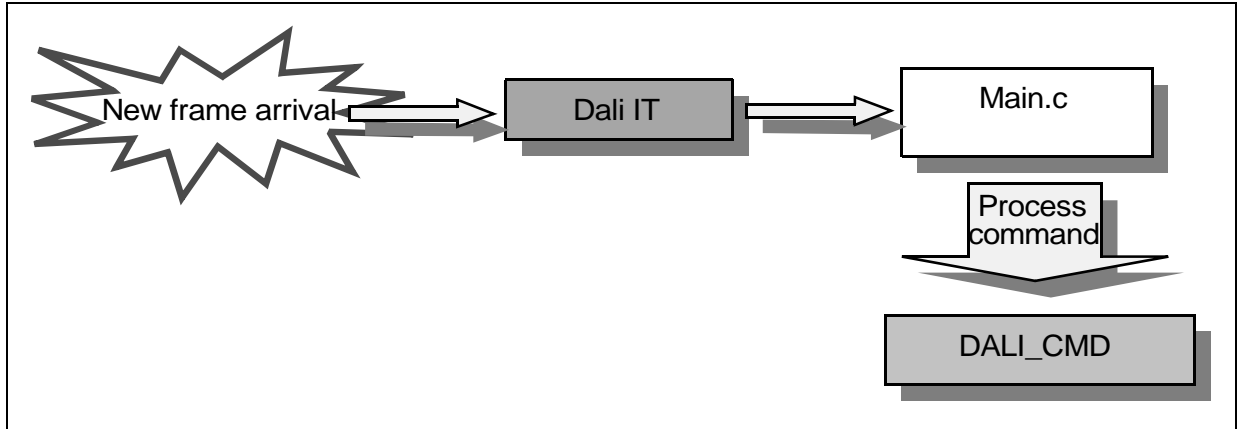
This application note describes a software example for driving a DALI slave board using an ST7DALI (ST7LITE2 family) microcontroller. It is supplied with the kit ST7DALI-EVAL and can be ordered with the code ST7DALI-EVAL.

The software is written in C language and is compatible with both Metrowerks and Cosmic compilers.



1 PROCESS OVERVIEW

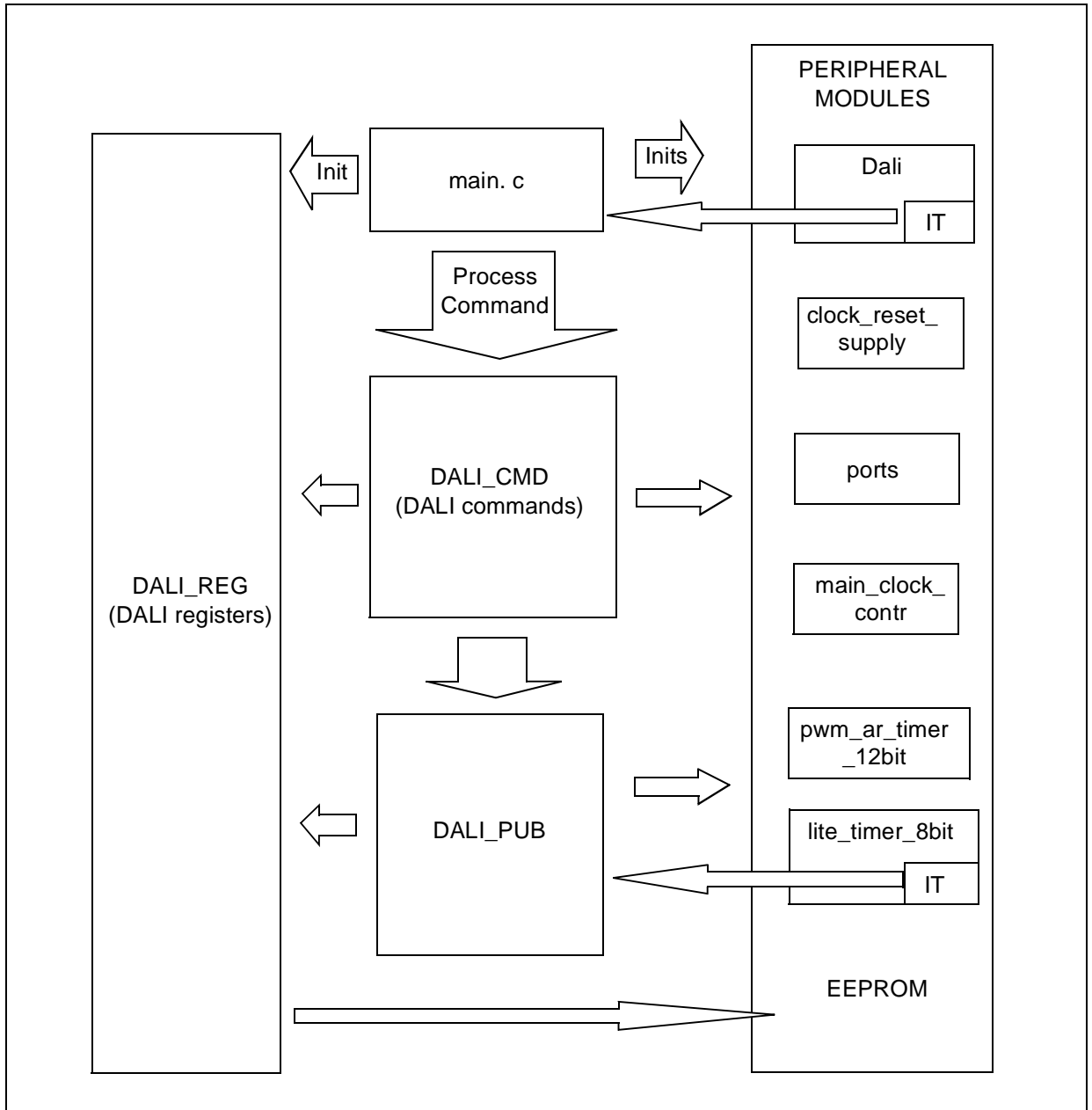
First of all, the DALI slave software initializes the slave system; afterwards, when the DALI master board sends a forward frame (basically 1 address byte and 1 data byte), and the DALI slave board receives the forward frame, a DALI interrupt (IT) is generated. This interrupt sets a flag, when this flag is set, the program checks whether the command is addressed to this ballast or not. If the command is addressed to this ballast the command handling process is started and the DALI slave board reacts to the received command.



2 GENERAL STRUCTURE OF THE DALI SLAVE SOFTWARE

The following diagram shows the different software modules and their relationships.

Figure 1. General Block Diagram



Note: The names of DALI_CMD module functions are prefixed by DALIC_. DALI_REG module functions are prefixed by DALIR_, and DALI_PUB module functions are prefixed by DALIP_.

3 MODULE DESCRIPTIONS

3.1 MAIN.C MODULE

For DALI communication, the microcontroller has to monitor the low voltage state on the DALI bus (it is not allowed to be more than 500ms), and so it needs a process to differentiate a frame reception from a bus down.

This module calls all the initialization routines in the other modules, then it calls the routines to switch on the red and green LEDs on the board, and finally it enters an infinite loop.

This loop could be divided in two parts, the management of the low voltage condition of the DALI BUS and fade rate plus the management of the new DALI frame reception.

The management of the low voltage condition and fade rate is obtained using a state machine, synchronized with a period of 1 ms, in fact the `lite_timer_IT_state` variable is set to one every 1 ms.

To better understand how it works, please refer to [Figure 12](#) in appendix A.

To manage the DALI frame reception, it checks the "dali_receive_status" flag in order to see if a new forward frame has been received by the microcontroller (MCU). If so, it calls the "DALIC_isTalkingToMe" function to check whether the command is addressed to this ballast or not. If it is, it switches on the green LED and starts the command handling process; otherwise it switches on the red LED. Finally it resets the "dali_receive_status" flag to restart the cycle.

3.2 DALI_CMD MODULE

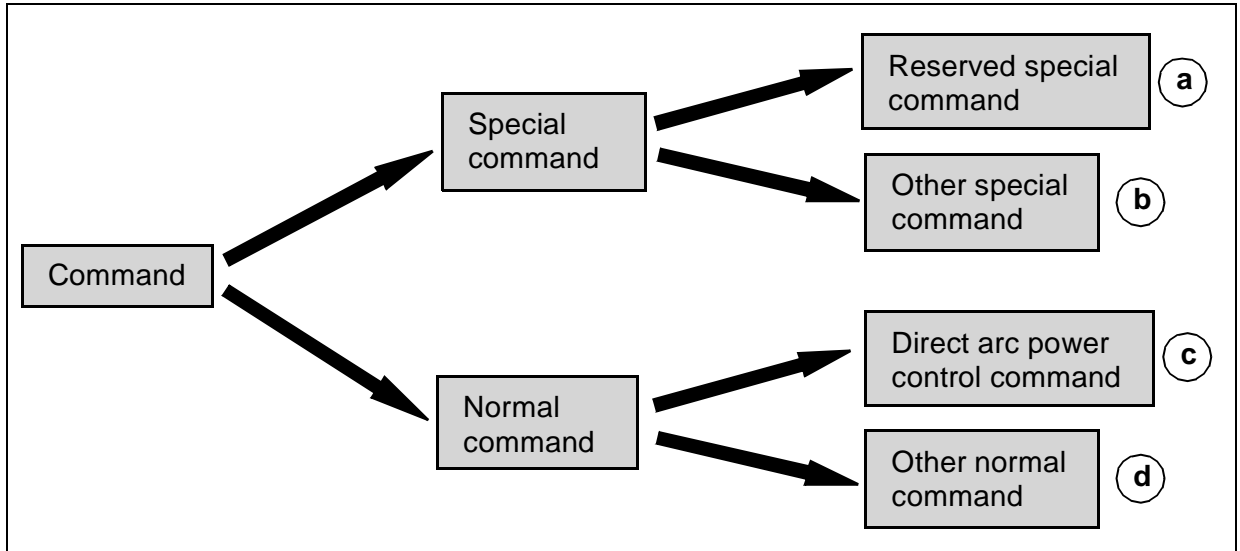
The main purpose of this module is to handle the DALI commands.

DALI_CMD contains several functions, most of them handle a particular DALI command, but three of them are called from outside the module (from main.c):

- 1) The "DALIC_Init", function initializes the ballast at its "POWER_ON_LEVEL".
- 2) The "DALIC_isTalkingToMe" function, checks whether the command is addressed to this ballast or not.
- 3) The "DALIC_ProcessCommand" function is the first step in the process of executing a command. It checks if a repetition fault has occurred (according to the specification, another command between an expected repetition is ignored and leads to the cancellation of the repetition sequence) and it checks whether the command is a special one or a normal one.

The sequence continues as shown in [Figure 2](#):

Figure 2. DALI_CMD Module Sequence

**Notes:**

- a) "DALIP_Reserved_Special_Function" is called. This function is empty, it is reserved for future needs.
- b) The function for handling a special command is called through the "special_jt" table.
- c) The "DALIC_Direct_Arc" function handles this kind of command.
- d) The function for handling a normal command, is called through the "normal_jt" table (that points to the function).

Note: In the case of COSMIC compilation, this is divided into three tables to avoid "long array" problems (this COSMIC problem has since been solved).

3.3 DALI_REG MODULE

In accordance with the DALI specification, some data variables giving information on the ballast and its status have to be stored in memory; we refer to these variables as DALI registers. The DALI_REG module handles the reading and writing of the DALI registers. It allows access to any of the DALI registers in the same way, wherever they are physically situated (ROM, EEPROM, RAM).

Normally you don't need to use the functions of this module directly (you can access most of these registers using the functions defined for this purpose in the DALI_PUB module). However, in case you need direct access to the registers, you can find the description of these functions and how to use them in Appendix B.

3.4 PERIPHERAL MODULES

The purpose of these modules is to handle the different peripherals and hardware blocks of the MCU, each one is in charge of a specific block.

All these modules have the same file structure. Each module has three files:

- xxx_hr.h where all or part of the registers and the register bits are defined.
- xxx.h that contains the declaration of the public functions and constants.
- xxx.c that contains the function routines.

(xxx represents the name of the module)

3.4.1 “clock_reset_supply” module

This module handles the “RC Oscillator Control” Register (RCCR) and the “System Integrity Control/Status” Register (SICSR). In the current version, it just configures the RCCR to calibrate the RC oscillator frequency. Two factory calibration values are stored in the first two EEPROM addresses, however you can calibrate the RC with a different value. Please refer to the ST7FDALI datasheet SUPPLY, RESET AND CLOCK MANAGEMENT section for more details.

3.4.2 “dali” module

This module handles the DALI peripheral.

It initializes the peripheral (DALI_Init function), and contains the DALI interrupt routine that handles the arrival of a forward frame. It also contains the (Send_DALI_Frame) function that allows sending backward frames to the master board. Please refer to the ST7FDALI datasheet, section DALI COMMUNICATION MODULE for more details.

3.4.3 “eeprom” module

This module handles the EEPROM of the MCU.

It initializes the EEPROM, so it saves the DALI registers (their reset values) in the EEPROM the first time that the program is started. It also contains the functions in charge of the physical reading and writing of the EEPROM. When you use the EEPROM, you don't need to use these functions directly, since a group of functions for handling the EEPROM is defined in the DALI_PUB module.

3.4.4 “lite_timer_8bit” module

This module handles the Lite Timer peripheral (Two 8-bit upcounters for timing purposes). In the current version, it just uses upcounter 1.

It initializes the Lite Timer so that the interrupt routine is run every 1 ms, this routine carries out the various countdowns required by the program.

3.4.5 “main_clock_contr” module

This module contains the Main Clock Control initialization routine that handles the Main Clock Control/Status Register (MCCSR). In the current version, it enables the MCO output clock; i.e. the CPU clock signal can be seen on this pin.

3.4.6 “ports” module

This module handles the MCU I/O ports. In the current version of the software, this application just uses the PA1 and PA2 ports; they are used as outputs for switching the LEDs on and off, to indicate whether a forward frame has been addressed to this ballast or not.

3.4.7 “pwm_ar_timer_12bit” module

This module controls the 12-bit autoreload timer.

In this application the 12-bit autoreload timer is used for generating a PWM signal to control the power level of the lamp (the PWM duty cycle determines the DC output level of the slave board).

So, the purpose of this module is to initialize the timer and set up the PWM duty cycle according to the command received.

3.5 DALI_PUB MODULE

This module has to be modified (if needed) by the user. The following sections describe the main parts of the module.

3.5.1 ROM registers

According the DALI specification, the DALI “version number” and “physical min. level” registers have to be stored in ROM, these two values are defined in this module by the “ROMRegs” table as follows: ROMRegs[]={0,25}, where the first value is the “version number” and the second is the “physical min. level”, you can modify these values according to the ballast used.

3.5.2 Fading functions

“DALIP_LaunchTimer”: starts a countdown in the Lite Timer interrupt routine (“lite_timer_8bit” module), so that the “DALIP_TimerCallback” function is called every 1ms. The parameter passed represents the number of times that the function “DALIP_TimerCallback” will be called, but if you pass 0xFF the function will be called every 1ms non stop until the function “DALIP_DoneTimer” is called.

“DALIP_DoneTimer”: is used to stop the timer started by “DALIP_LaunchTimer”. It should be called as soon as the process is finished to let the MCU enter in SLOW-WAIT-MODE (in order to save power) if it has nothing more to do.

“DALIP_TimerCallback”: handles the fading effect; basically, it increases or decreases (one step) the arc power level every “DALIP_iChangeEvery” ms (according to the DALI command processed and the fade time/rate selected), until the required power level is reached.

3.5.3 Arc power control functions

The following functions are the last step in the process of all “arc power control” commands: DALIP_Direct_Arc, DALIP_Off, DALIP_Up, DALIP_Down, DALIP_Step_Up, DALIP_Step_Down, DALIP_Step_Down_And_Off, DALIP_On_And_Step_Up. They set the new power level and update the “actual dim level” register. DALIP_Off, DALIP_Up and DALIP_Direct_Arc (if necessary) calculate the “DALIP_iChangeEvery” value used by the “DALIP_TimerCallback” function, this value is calculated according the current fade time/rate value to carry out the fading process.

Note: All the above functions correspond to DALI commands described in the DALI specification.

3.5.4 DALI register access functions

The following functions allow you to write and read most of the specific DALI registers (see DALI spec.). If you need to access the registers directly, refer to the DALI_REG module description.

The Write Functions pass just one parameter to update a particular DALI register and returns nothing:

Table 1. List of DALI Register Write functions

Write-Functions	DALI register affected
DALIP_SetArc	Actual dim level
DALIP_SetBallastStatusFlag	Status information (bit 0)
DALIP_SetLampFailureFlag	Status information (bit 1)
DALIP_SetLampPowerOnFlag	Status information (bit 2)
DALIP_SetFadeReadyFlag	Status information (bit 4)
DALIP_SetPowerFailureFlag	Status information (bit 7)

Note: For Flag Registers, pass 0 to clear the Bit and !=0 to set it.

The Read Functions return the current value of a particular DALI register:

Table 2. List of DALI Register Read functions

Read-Function	DALI register affected
DALIP_GetArc	Actual dim level
DALIP_GetFadeTime	Fade time
DALIP_GetFadeRate	Fade rate
DALIP_GetMaxLevel	Max level
DALIP_GetMinLevel	Min level
DALIP_GetPowerOnLevel	Power on level

DALIP_GetSysFailureLevel	System failure level
DALIP_GetStatus	Status information
DALIP_GetVersion	Version number
DALIP_GetPhysMinLevel	Physical min. level

3.5.5 EEPROM access functions

Using these functions, you can read and write to the connected EEPROM. The addressing range is from 0 to the return-value of DALIP_EEPROM_Size. Accesses outside that range will be ignored.

Note: The returned EEPROM-Size is the actual size minus a few bytes that are used for saving the DALI-Registers. There is maximum size of 256 bytes for the connected EEPROM. (Bigger ones work too, but only the lower 256 bytes can be accessed).

Caution: If a Page-Write exceeds the addressing range, the WHOLE Write Operation will be ignored!

DALIP_EEPROM_Size

Return: Highest address that can be passed to an EEPROM-Access-Command

DALIP_Read_E2

Reads one byte from the passed address

Param1: Address to be read

Return: Data byte read from the EEPROM

DALIP_Write_E2

Writes one byte to the passed address

Param1: Address to write to

Param2: Data byte to be written

DALIP_Write_E2_Buffer

Writes a sequence of Bytes (uses the page-write-operation of the EEPROM to be faster)

Param1: First Address to write to

Param2: Number of Bytes to be written

Param3: Pointer to the first byte of the array that contains the data

3.5.6 Reserved functions

Many commands in the DALI specification are reserved for future needs. When a forward frame calls for one of these commands, one of the following functions is called. Either "DALIP_Reserved_Function" if the reserved command is not special (commands in the range

0-255) or “DALIP_Reserved_Special_Function” if it’s a special command (commands in the range 256-287).

At present these functions are empty.

3.5.7 Other functions

DALIP_Is_Physically_Selected

This function returns 1 if the device is physically selected, otherwise it returns 0.

Since the first case it is not yet implemented, at present it always returns 0.

DALIP_What_Device_Type

This function returns a number that corresponds with the type of the device used. At present it returns 0 that means “device for fluorescent lamps” according to the DALI specification, so you should specify here the type if it is a different one.

APPENDIX A. FLOWCHARTS

Figure 3. Main.c Flowchart

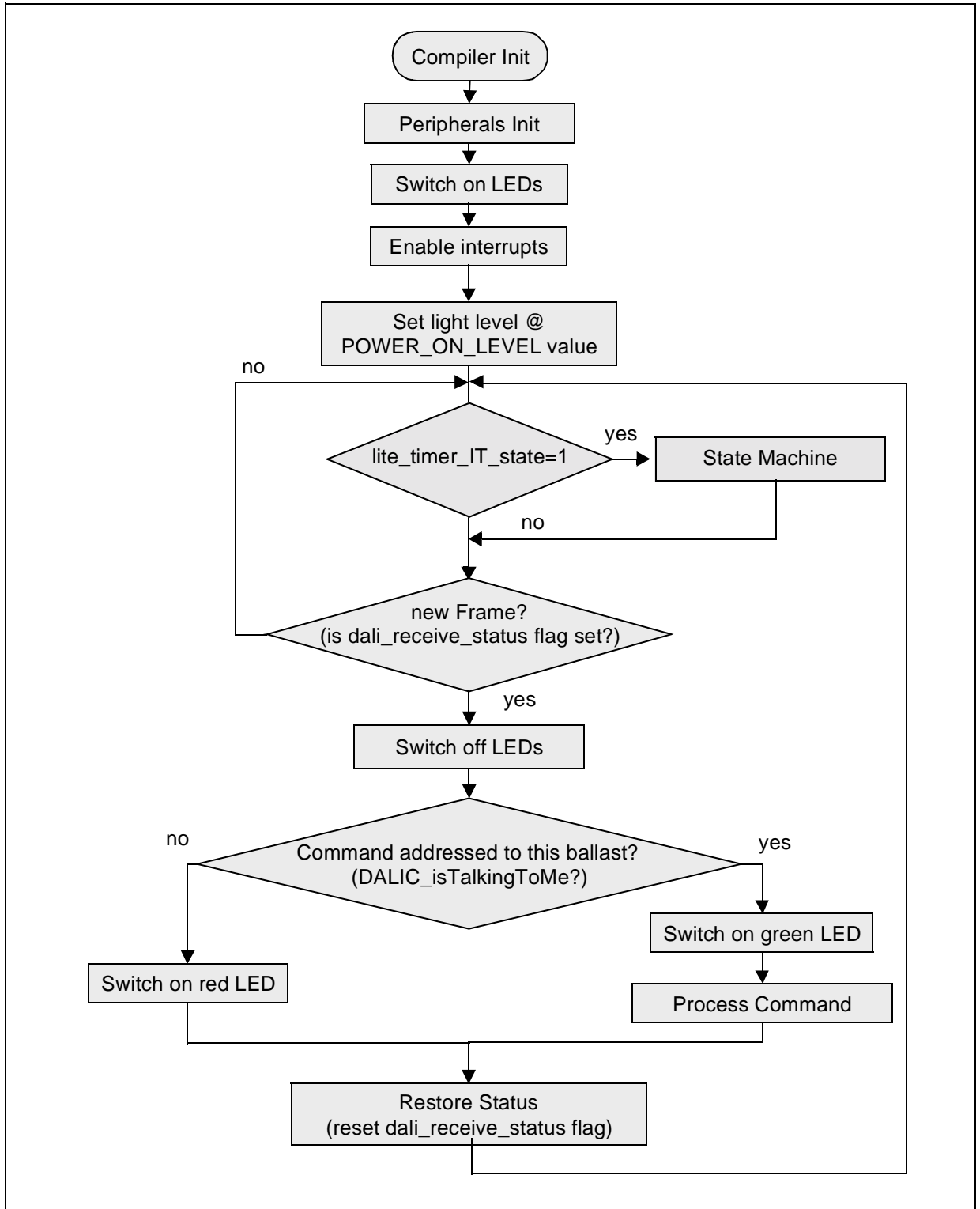


Figure 4. DALIC_is_talking_to_me @ dali_cmd.c

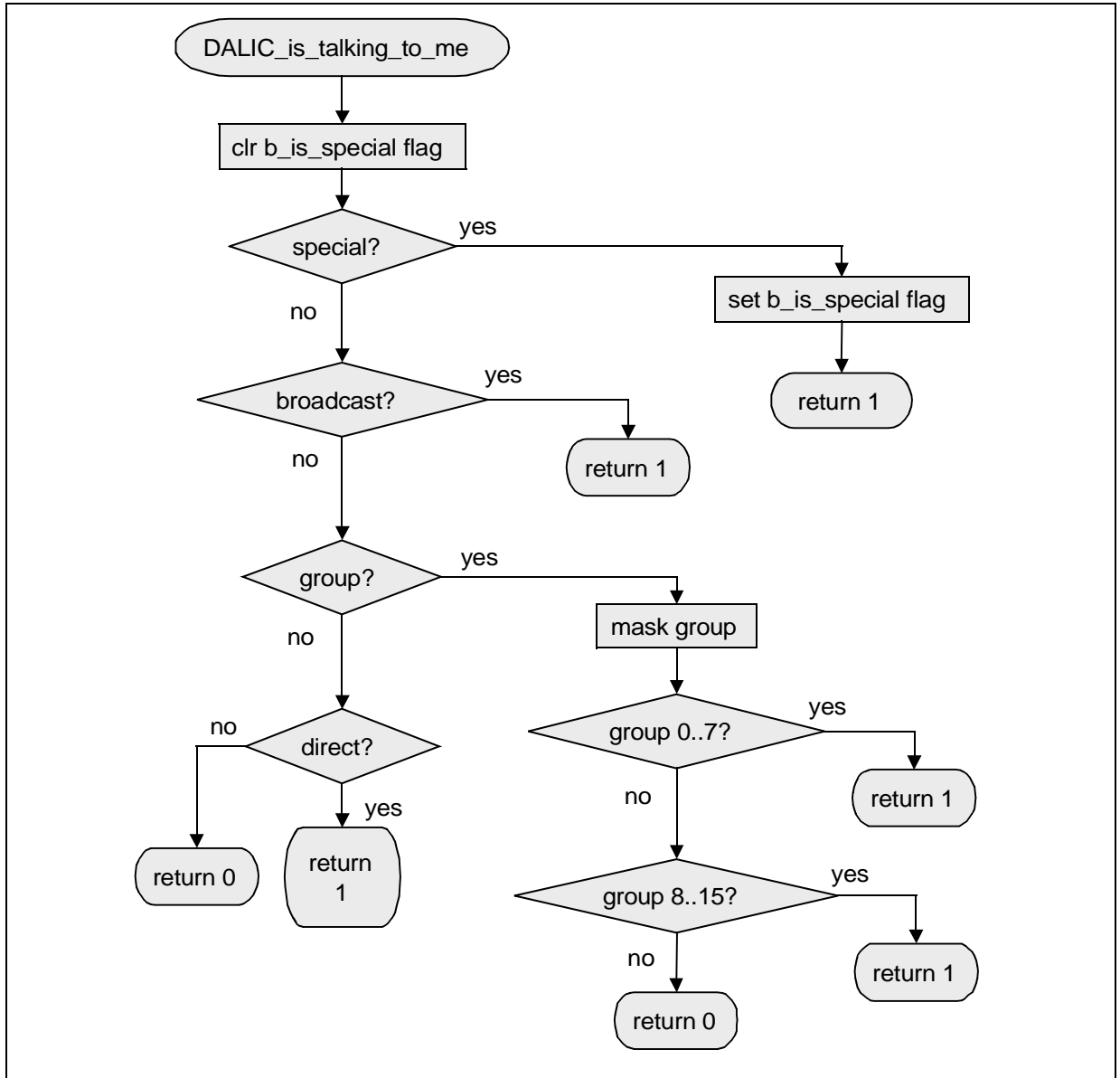


Figure 5. DALIC_ProcessCommand @ dali_cmd.c

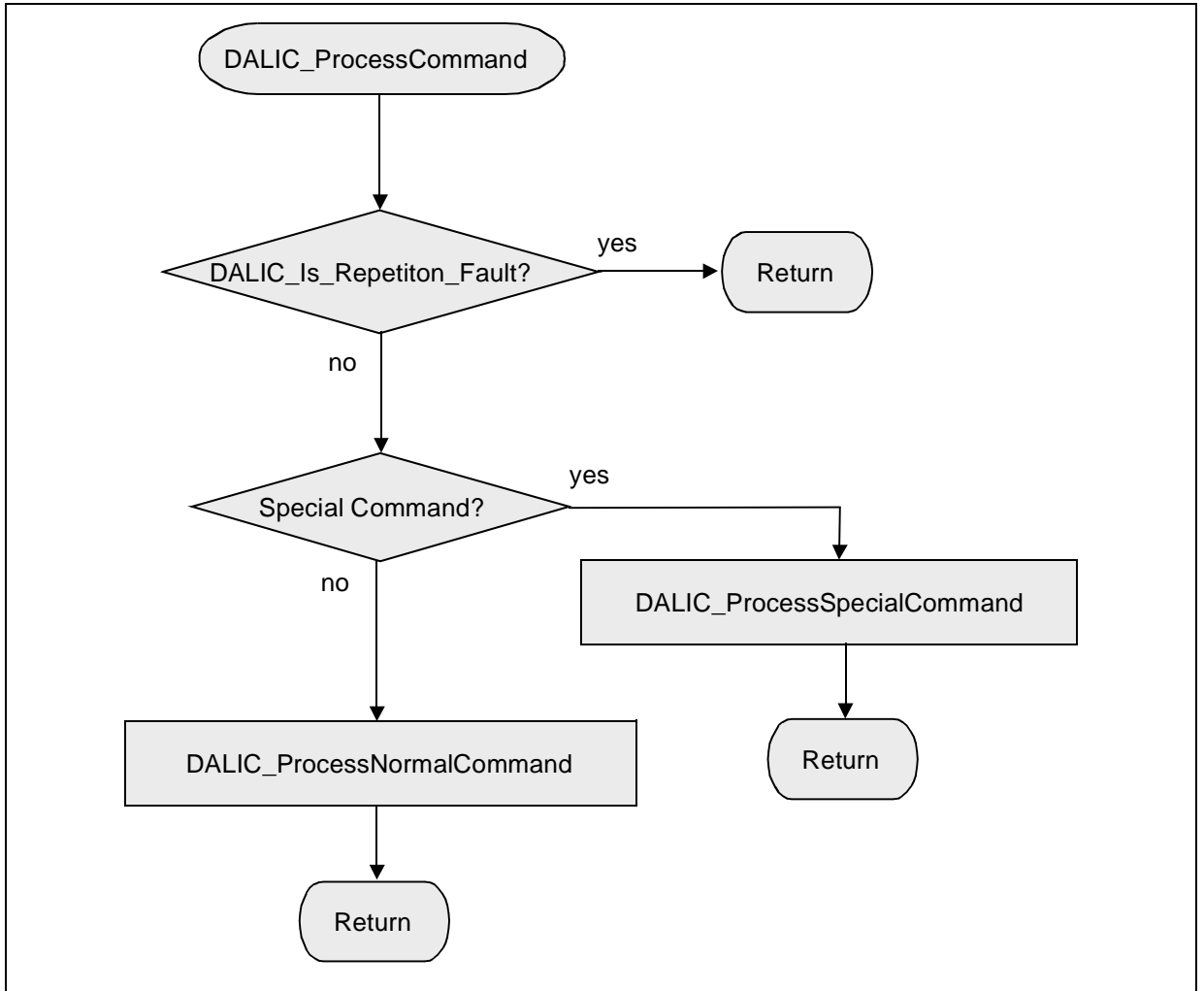


Figure 6. DALIC_ProcessSpecialCommand @ dali_cmd.c

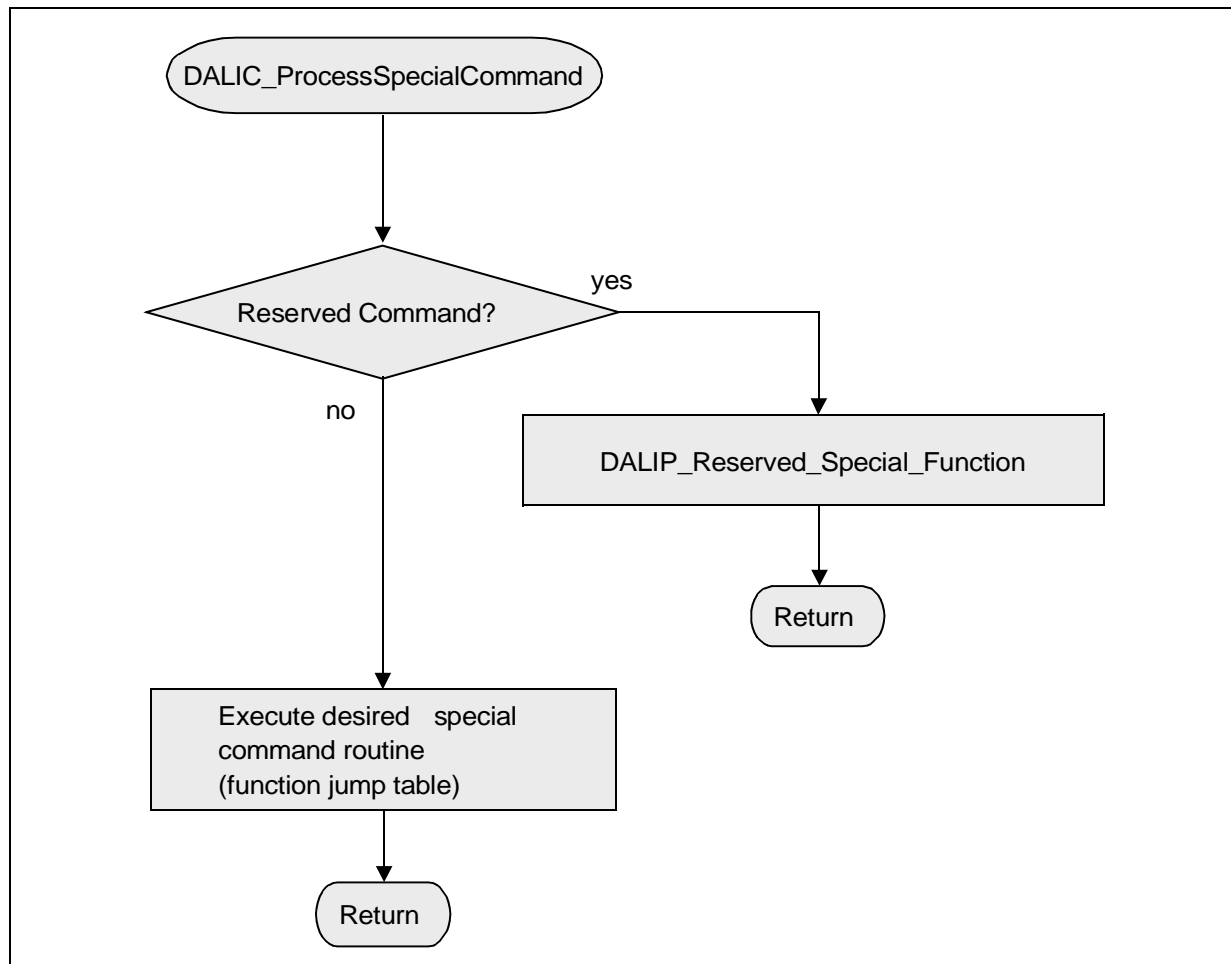


Figure 7. DALIC_ProcessNormalCommand @ dali_cmd.c

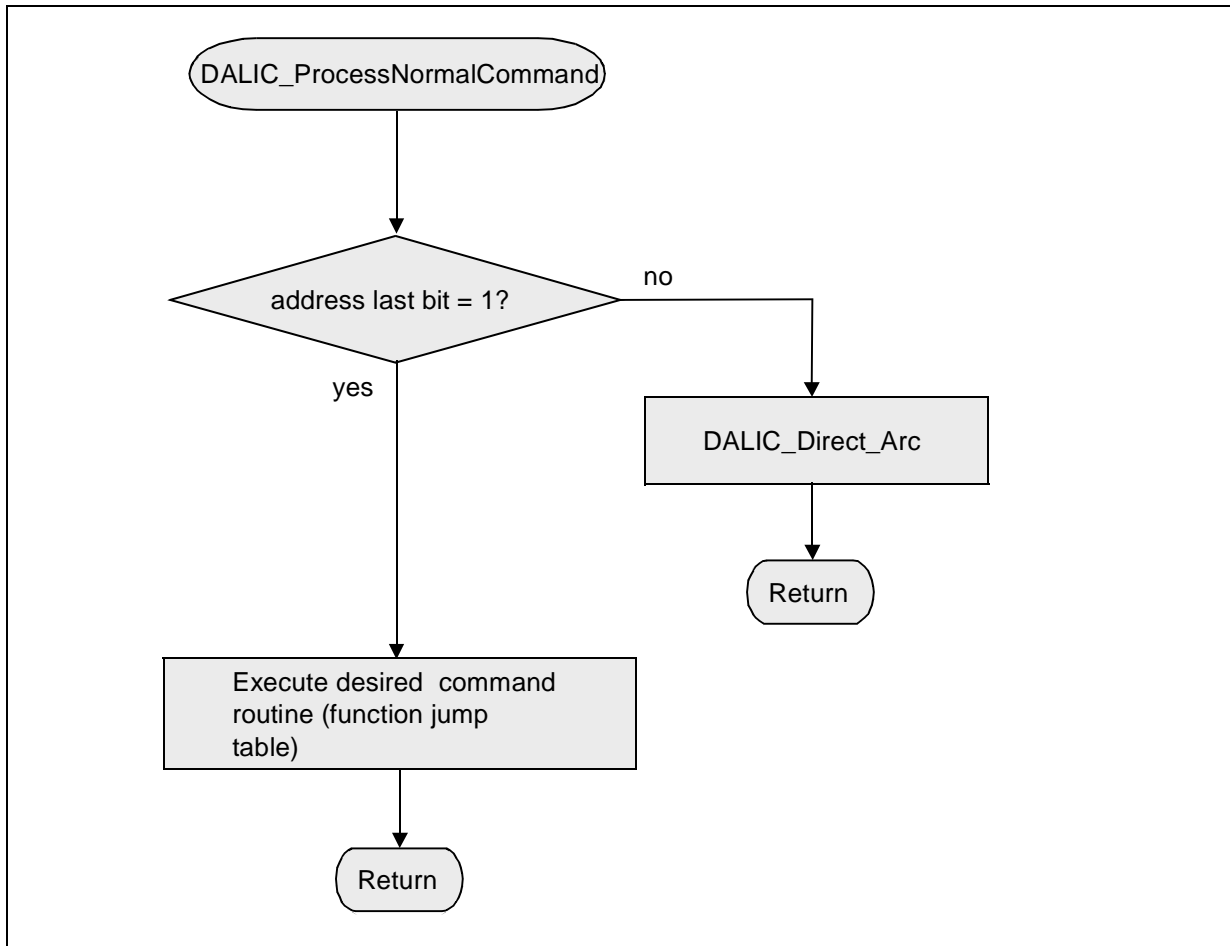


Figure 8. DALIC_Is_Repetition_Fault @ dali_cmd.c

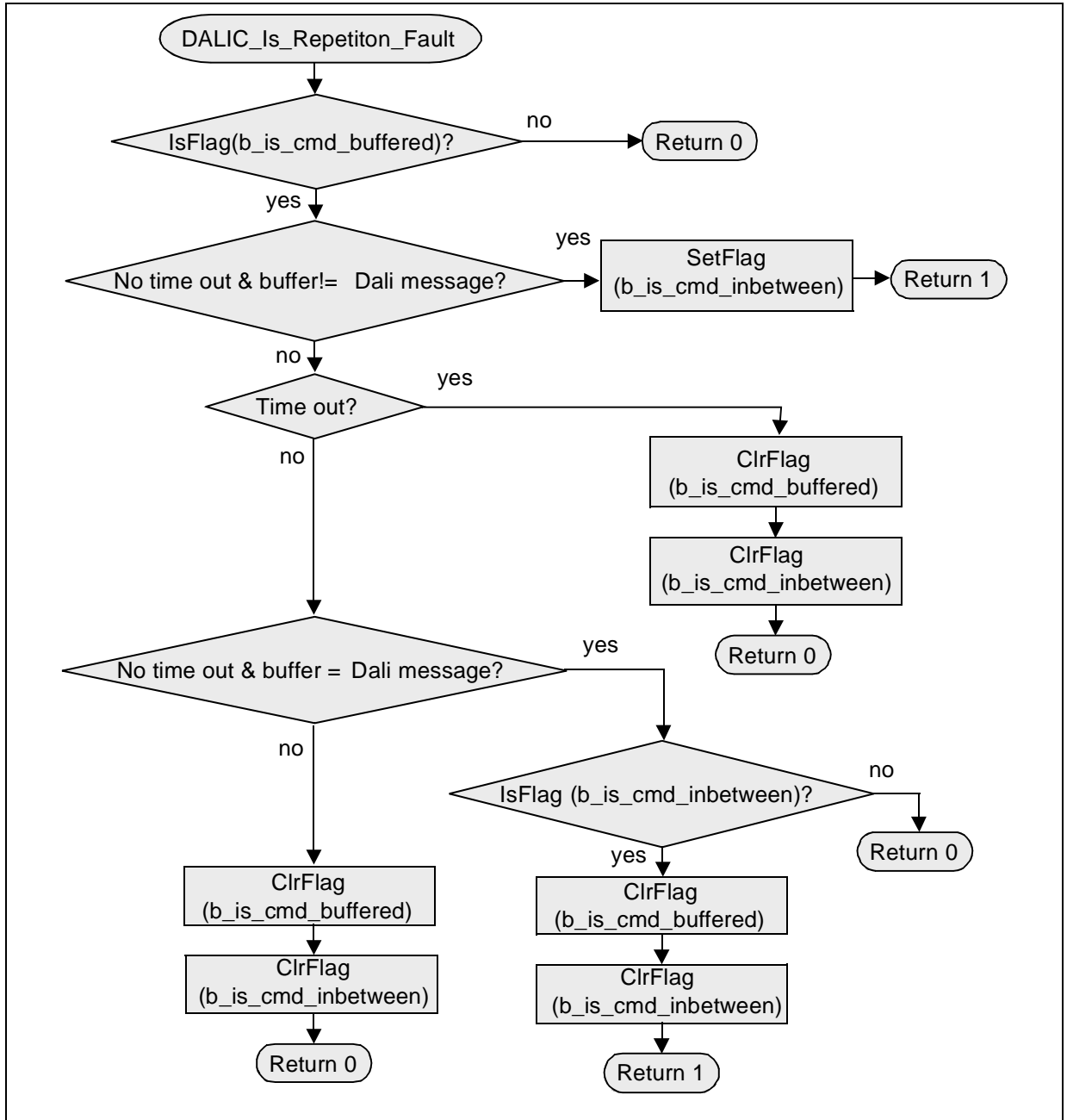


Figure 9. DALIC_Direct_Arc @ dali_cmd.c

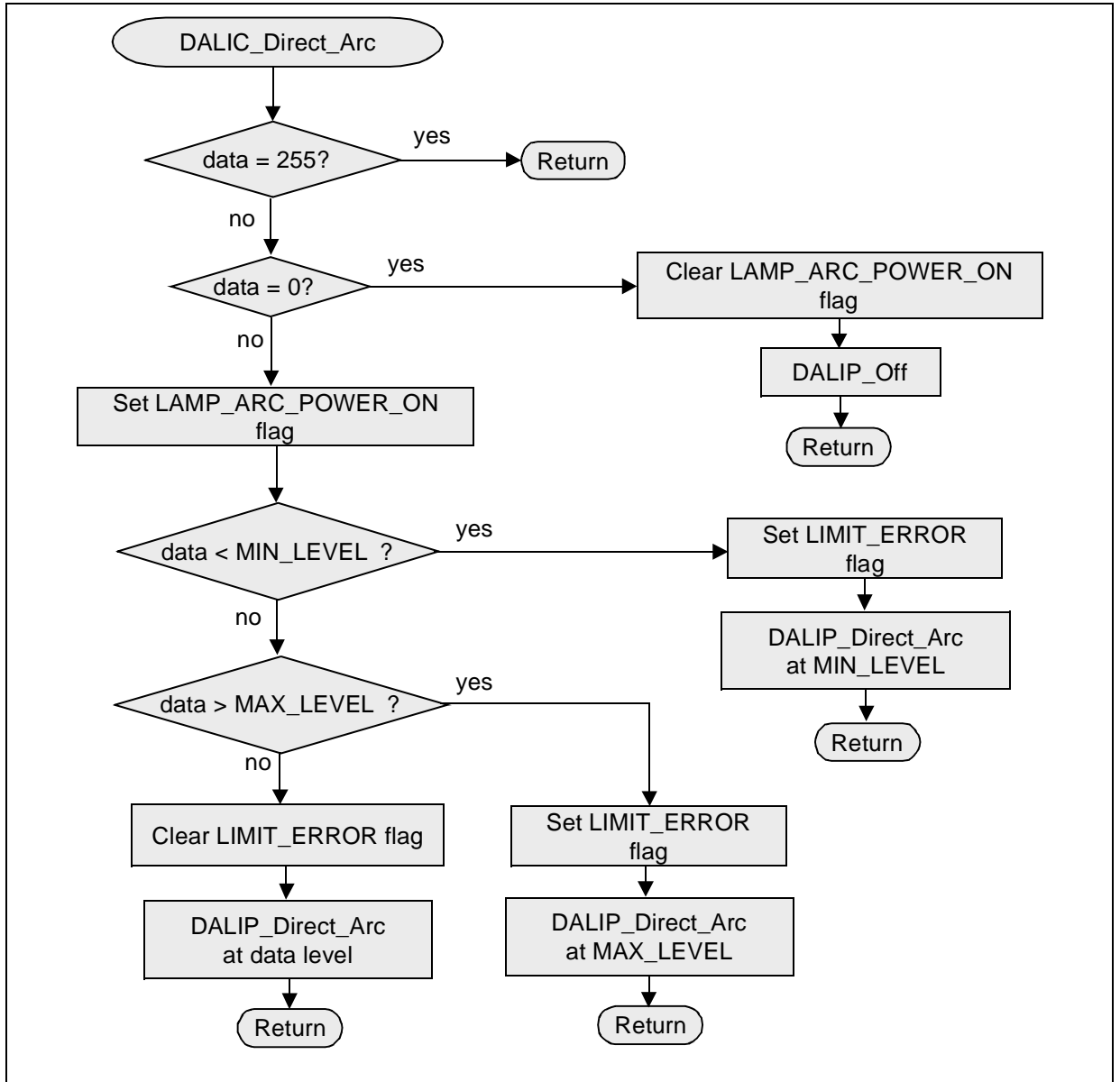


Figure 10. DALIC_Is_Repeated @ dali_cmd.c

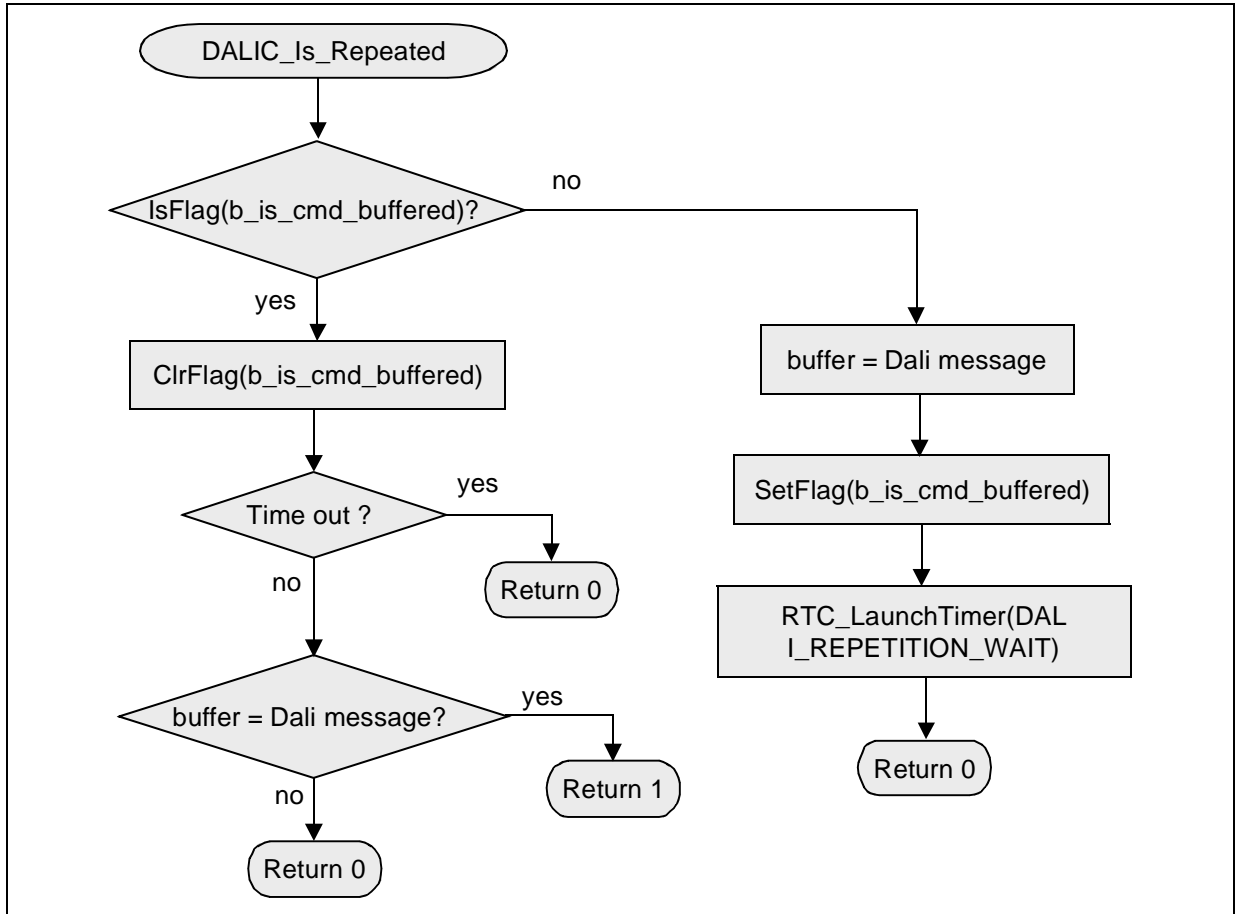


Figure 11. DALIP_Direct_Arc @ DALI_PUB.C

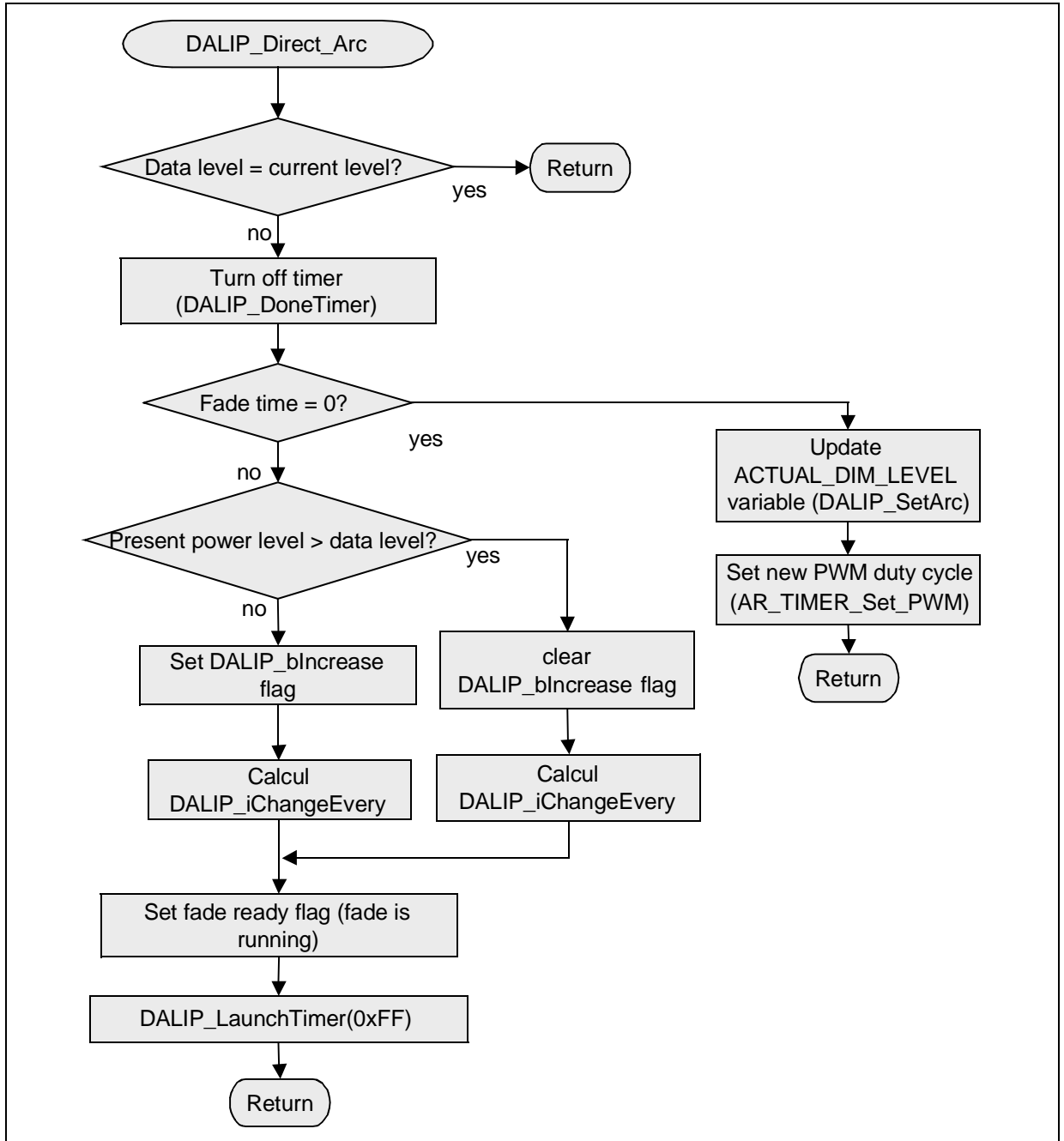
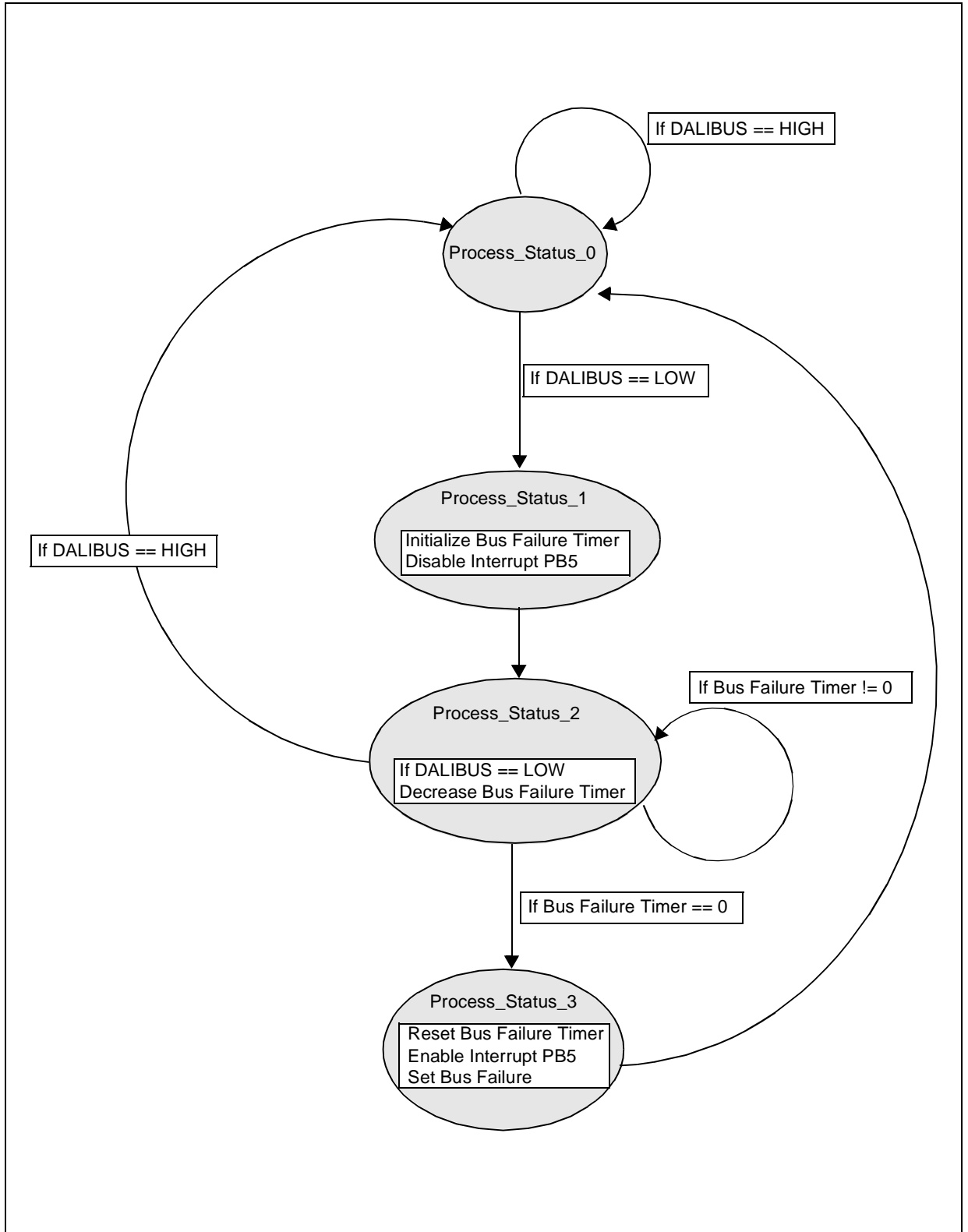


Figure 12. State Machine diagram



APPENDIX B. DALI_REG MODULE FUNCTIONS

The program makes use of the following functions of the DALI_REG module to read/write the DALI registers easily:

DALIR_ReadReg

Purpose: it reads the value of one register.

Parameters: name of the register to be read

Returns: value of the register

DALIR_WriteReg

Purpose: It writes to one of the DALI-Registers

Parameters: Name of the register to be write, new value

Returns: --

DALIR_WriteStatusBit

Purpose: To reset/clear one bit of the DALIREG_STATUS_INFORMATION register.

Parameters: name of the bit, 0 to reset the bit and !=0 to set it.

Returns: --

DALIR_ReadStatusBit

Purpose: It reads the value of one bit of the DALIREG_STATUS_INFORMATION register.

Parameters: name of the bit to be read.

Returns: bit value (0 or 1).

Other DALI_REG functions:

DALIR_Init

Purpose: it initialises the RAM-registers to zero.

Parameters: --

Returns: --

DALIR_ResetRegs

Purpose: It initialises the DALI registers to their "reset value" (the "reset value" for each register is specified the DALI specification)

Parameters: --

Returns: --

DALIR_LoadRegsFromE2

Purpose: It loads the DALIREG_SHORT_ADDRESS into the "short_addr" variable (function used just in the EEPROM init).

Parameters: --

Returns: --

DALIR_DeleteShort

Purpose: It writes 0xFF in the DALIREG_SHORT_ADDRESS register, which means that no address is ascribed to this ballast (mask).

Parameters: --

Returns: --

The names of the registers are defined in dali_regs.h as follows:

RAM-Registers:

DALIREG_ACTUAL_DIM_LEVEL (1 byte)

DALIREG_SEARCH_ADDRESS (3 bytes)

DALIREG_STATUS_INFORMATION (1 byte):

Bit 0: DALIREG_STATUS_BALLAST

Bit 1: DALIREG_STATUS_LAMP_FAILURE

Bit 2: DALIREG_STATUS_LAMP_ARC_POWER_ON

Bit 3: DALIREG_STATUS_LIMIT_ERROR

Bit 4: DALIREG_STATUS_FADE_READY

Bit 5: DALIREG_STATUS_RESET_STATE

Bit 6: DALIREG_STATUS_MISSING_SHORT

Bit 7: DALIREG_STATUS_POWER_FAILURE

E²PROM-Registers:

DALIREG_POWER_ON_LEVEL (1 byte)

DALIREG_SYSTEM_FAILURE_LEVEL (1 byte)

DALIREG_MIN_LEVEL (1 byte)

DALIREG_MAX_LEVEL (1 byte)

DALIREG_FADE_RATE (1 byte)

DALIREG_FADE_TIME (1 byte)

DALIREG_SHORT_ADDRESS (1 byte)

DALIREG_RANDOM_ADDRESS (3 bytes)

DALIREG_GROUP_0_7 (1 byte)

DALIREG_GROUP_8_15 (1 byte)

DALIREG_SCENE (16 bytes)

ROM-Registers:

DALIREG_VERSION_NUMBER (1 byte)

DALIREG_PHYS_MIN_LEVEL (1 byte)

“THE PRESENT NOTE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING CUSTOMERS WITH INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM TO SAVE TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING FROM THE CONTENT OF SUCH A NOTE AND/OR THE USE MADE BY CUSTOMERS OF THE INFORMATION CONTAINED HEREIN IN CONNECTION WITH THEIR PRODUCTS.”

Information furnished is believed to be accurate and reliable. However, STMicroelectronics assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of STMicroelectronics. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. STMicroelectronics products are not authorized for use as critical components in life support devices or systems without express written approval of STMicroelectronics.

The ST logo is a registered trademark of STMicroelectronics.

All other names are the property of their respective owners

© 2004 STMicroelectronics - All rights reserved

STMicroelectronics GROUP OF COMPANIES

Australia – Belgium - Brazil - Canada - China – Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan -
Malaysia - Malta - Morocco - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States

www.st.com