UM1728
User manual

STM32F1xx DALI slave interface

Introduction

This document describes the implementation of the digital addressable lighting interface (DALI) into the STM32F1xx 32-bit microcontroller family.

The DALI slave library for STM32F1xx microcontrollers simplifies integration of the DALI slave interface into customer applications. The implementation of DALI into STM32, together with the various STM32 features (peripherals, computation power, communication interfaces), is mainly used in light control applications (example, electronic ballast control).

The STM32 DALI slave library was tested according to the DALI specification.

The DALI slave library comes with a simple application example (DALI slave device). The example was designed (and tested) for use with the following evaluation boards:

- STM32VLDISCOVERY
- DALI transceiver board (see UM1032: STEVAL-ILM001V1 hardware module)

The application example controls the light of the on-board LED diode. Light intensity is controlled by the PWM method using a built-in timer. The external DALI master device must control this application example (which is the DALI slave device). The DALI master devices were tested using the ST7DALI evaluation kit (master device board and PC software).

Useful information and links about DALI interface can be found on http://www.dali-ag.org.

Contents

List of figures

1 DALI information

1.1 DALI standard overview

DALI is an international standard (IEC 62386) lighting control system that provides a single interface for electronic control gears (light sources) and devices (lighting controllers).

The DALI standard enables dimmable ballasts, transformers, relay modules, emergency fittings and controllers from different manufacturers to be mixed and matched into a single control system. A DALI system provides designers, installers, building owners, facility managers and end-users a powerful and flexible digital lighting system with security of supply from many sources.

The DALI standard is overseen by the “AG-DALI” activity group which comprises engineers, manufacturers, and institutions working in the field of digital lamp/ lighting control.

More information about the DALI standard can be found in the following documents:

- IEC 62386
- NEMA standard 243-2004

The following sections provide an overview of the DALI standard. They describe the basic principles of the DALI interface.

Note: To better understand the STM32 DALIF1xx slave library, a knowledge of the DALI interface specification is essential. This application note does not provide a description of this specification.

1.1.1 DALI purpose and properties

The DALI protocol was designed to control modern light sources using a computer.

Functions include:

- Dimming
- ON/ OFF switching
- Grouping lights to a common control
- Scene storage and selection

The DALI design properties include:

- Simple wiring using standard electrical installation cables
- No special wiring topology (as with power electrical cables)
- Simple installation giving cable polarity independence
- Automated light source addressing
- Use of low cost microcontrollers on the light source side to minimize cost of light source
- Use of a simple protocol to control light dimming and switching.

1.1.2 DALI physical layer

The DALI interface consists of a physical layer from two wires. This is a simple installation for which the polarity is independent.

The protocol used on these cables is a standard serial protocol.There is 1 start bit, 8 data bits and 2 stop bits.The communication speed is fixed at 1200 Bd.Manchester coding is used for better resynchronization: rising edge is logical 1 and falling edge is logical 0.

Bytes are grouped into frames. One frame usually consists of 1 or 2 bytes which is either data only (answer from the device) or address + data (command to device).

Voltage levels present on DALI communication wires are higher than the transistor-transistor logic (TTL) levels that are usually used. This is due to better noise immunity because of higher interference present on nearby power installation cables. Voltage levels are defined as follows:

- Low level state
  - -4.5 to 4.5 V (transmitter)
  - -6.5 to 6.5 V (receiver)
- High level state
  - 11.5 to 20.5 V (transmitter)
  - 9.5 to 22.5 V (receiver)

Low level state is dominant on the DALI bus. The device can force this level to the DALI bus by shortening the DALI wires. Consequently, current levels are defined for devices used on the DALI bus. There are two device types:

1. DALI master or DALI slave communication devices. In these devices the current is sunk from the DALI bus. They consume a maximum of 2 mA to receive high level state and sink a minimum of 250 mA to transmit low level state.
2. DALI power supply. These devices power the DALI bus and are the source current for it. Supply is limited to a maximum of 250 mA.

The maximum length of the DALI bus depends on the cables used for the DALI communication wires (for example, a 1.5 mm2 cable (which is typically used in light installation cables) allows a maximum bus length of up to 300 m.). The length is linearly dependent on the cross section of the conductor.

Collisions between several DALI masters on the DALI bus are solved on the basis of timing priorities. When a collision is detected (the DALI master should check the sent data), communication is muted for a period of time according to the assigned master priority. There are five defined priority levels: 12 ms, 13 ms, 14 ms, 15 ms, and 16 ms. The longest waiting time has less priority.

Figure 1. Example of DALI device connections

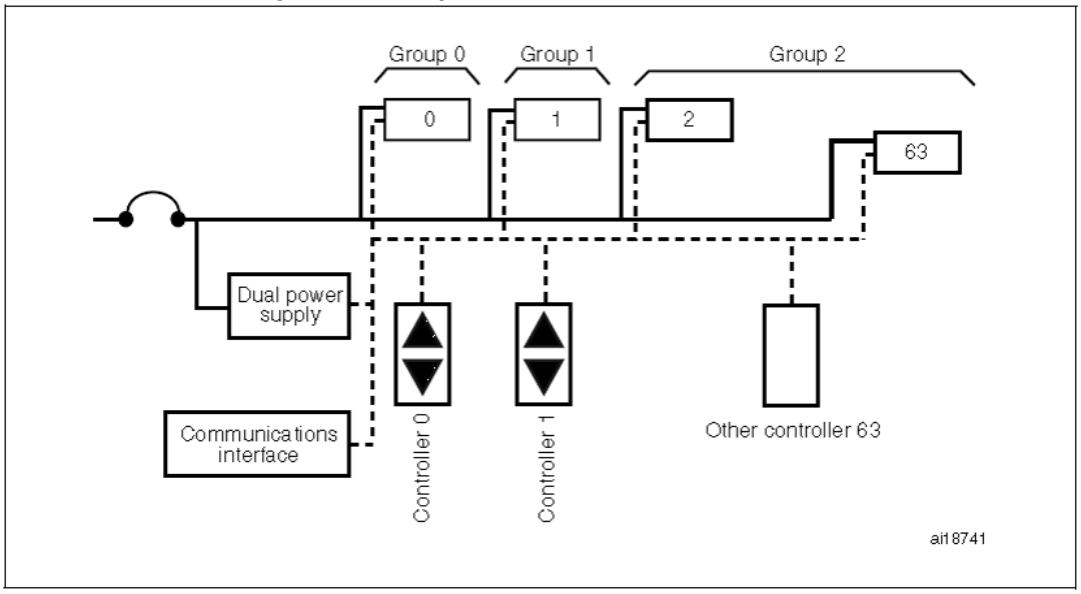
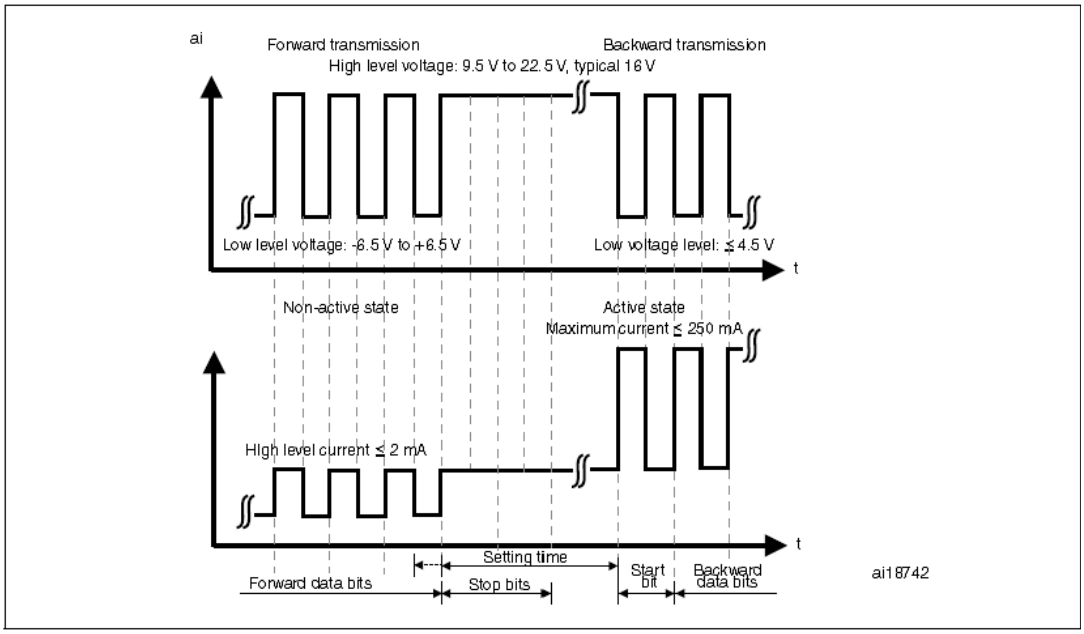


Figure 2. Voltage and currents on the DALI bus



1.1.3 DALI stack layer

The DALI stack is the higher DALI layer which implements DALI commands, DALI structures, timing management, and error management according to the DALI specification. The most important feature of the DALI stack layer is the DALI commands. These commands are used for:

- Direct light control such as dimming, on/off switching, and scene selection.
- Device configuration to set DALI variables, read device properties, address assignments, and query device status.

The DALI stack layer is fully implemented in microcontroller software.

1.2 STM32 DALI slave library overview

The STM32F1xx DALI slave library contains instructions for implementing software of the DALI protocol into STM32 microcontrollers.

The DALI slave library consists of two layers:

- Upper layer (DALI stack layer)
- Hardware layer (I/O pin driver layer)

The upper layer of the DALI slave library (also known as the DALI stack layer), consists of routines for processing the required DALI commands and other necessary control functions (including timing control, error management, memory management, and light control). The hardware layer (also known as the direct I/O pin driver layer), contains the physical layer implementation. This is necessary because STM32 microcontrollers do not support DALI communication peripherals.

2 Structure of final user application

The final user application uses the DALI slave library and so, consists of the following three layers:

- Main user application
- DALI stack layer
- I/O pin driver layer

2.1 User application

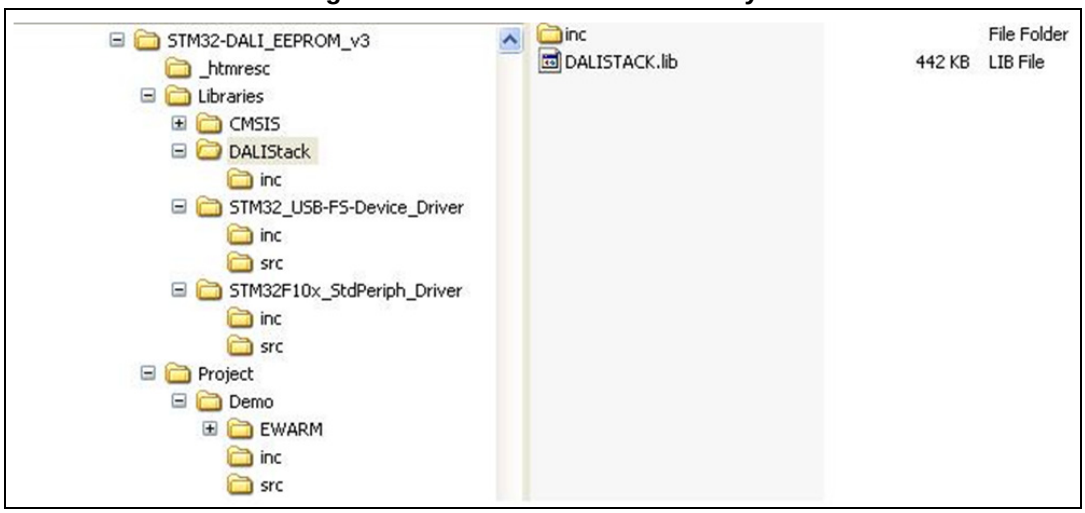
The structure of the user application is comprised completely of IAR project files. The application code is in one source file, “main.c”, in which the whole user ballast control is implemented. This code calls the DALI slave library functions.

The user application example included in the DALI slave library package is designed to work with the “STM32 discovery kit” and “STEVAL_ILM001V1 (DALI transceiver board)”. The STM32 discovery kit is a small printed circuit board (PCB) with an integrated STM32F100RBT6 device which demonstrates STM32 capabilities. The DALI transceiver board is connected to the STM32 discovery kit as a physical layer extension. It provides voltage level conversion and optical isolation between the DALI bus and the STM32 discovery kit (according to the DALI specification requirements).

The main task of the user application is to initialize the DALI stack and to implement the light intensity control function (callback function). This function is ballast dependent. The user application also periodically controls some event flags. If the DALI slave library sets an event flag and it indicates a pending event, the user application must call the appropriate service routines (from the DALI slave library) to execute the necessary commands.

The user application example uses timer 3 to control the light intensity of the LED diode present on the STM32 discovery kit. Timer 3 generates the pulse-width modulation (PWM) for the LED diode (using the callback function). The main program loop checks event flags, executes actions for active flags, and provides power management (Low power state if the DALI bus is quiet) and error management (reports hardware errors). For all these purposes the user application calls functions from the DALI slave library.

Figure 3. Structure of DALI stack layer



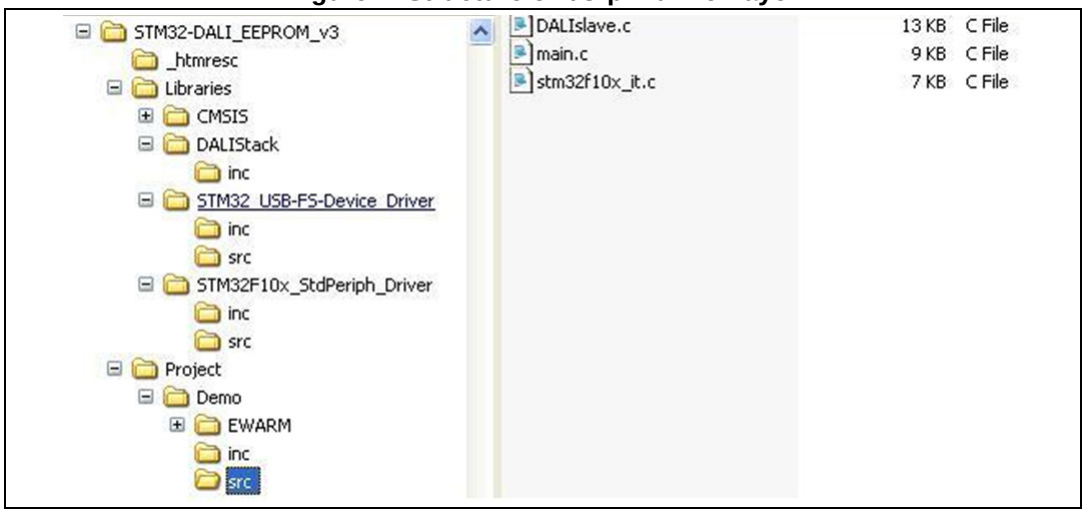
The DALI stack is hardware independent. It calls the I/O pin driver for low level hardware dependent functions. After initialization of the DALI stack layer (after calling initialization functions), most of the tasks run automatically. Tasks include receiving commands and running timer tasks (light dimming, timeouts).

Note: To get the complete source code of DALI STACK library, please contact nearest sales office.

2.2 I/O pin driver layer

The I/O pin driver layer of the DALI slave library implements physical and link layers of the DALI interface. The DALI protocol requires two pins for communication. The I/O pin driver layer implements reception and transmission through direct control of two arbitrary GPIO pins. It also controls the required DALI speed and correct bit timing, solves DALI bus error recovery, and collects received bits into DALI frames (using address and command bytes). *Figure 4* displays the structure of the DALI I/O pin driver layer

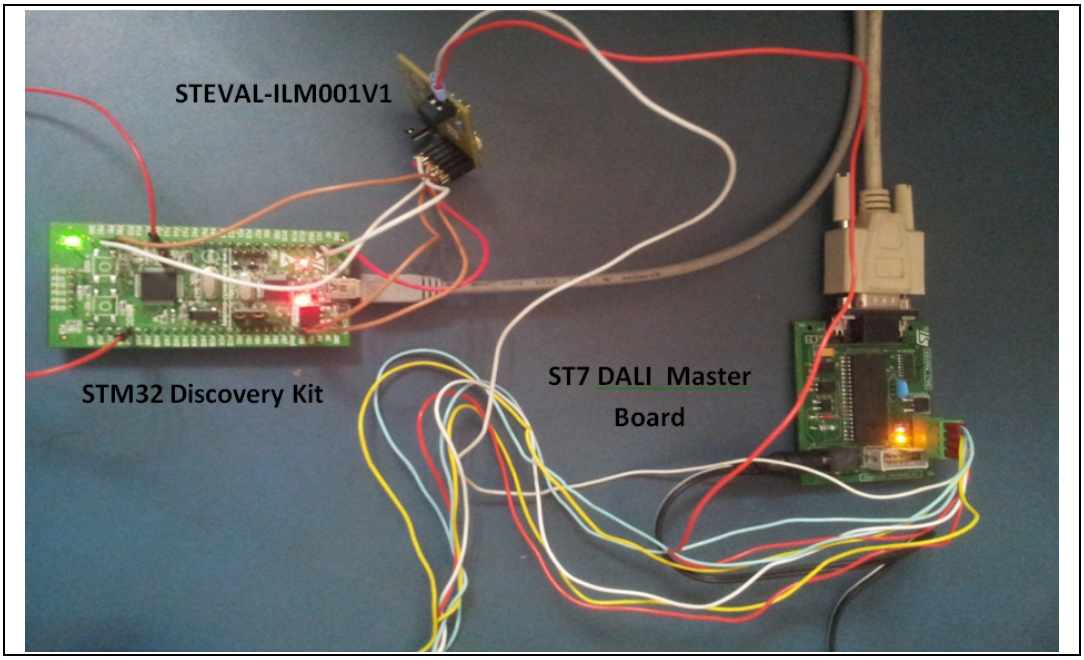
Figure 4. Structure of I/O pin driver layer



The I/O pin driver layer occupies one timer (systick timer), one GPIO interrupt, one timer interrupt, and direct control of two GPIO pins (assignment of which is software selectable).

Timing events are invoked inside the timer interrupt service routine. Such events control the I/O pin driver layer behavior by calling callback functions from the DALI stack layer at given 1 ms intervals (timer ticks).

Figure 5. Hardware for testing DALI: STM32 discovery kit with DALI transceiver board and ST7 DALI master board



3 Function description

This section describes the most important STM32F1xx DALI slave library functions for the user.

3.1 I/O pin driver layer functions

The I/O pin driver layer implements hardware dependent functions (designed for use with STM32F1xx devices and the DALI transceiver board).

3.1.1 STM32F10x_it.c

These files contain two interrupt service routines (ISR):

void SysTick_Handler(void)

This function:

- Generates timer ticks for the DALI bit receive and transmit functions
- Calls, in given intervals, the transmit/receive function (sampling is eight times per DALI bit).
- Generates timer ticks for the upper DALI layer timing by calling callback functions fr om the DALI stack layer
- Checks for DALI interface failure such as loss of voltage on the DALI bus for more than 500 ms (disconnection from DALI bus).

Void EXTI0_IRQHandler(void)

This function:

- Calls receiving routines if there is a voltage level change on the DALI RX pin
- Detects the start of the DALI packet. This ISR is inactive during DALI packet reception.It is activated after DALI packet reception

3.1.2 DALIslave.c

These file contains the reception and transmission routines for building the DALI packet from received bits according to the DALI specification. It also contains the initialization functions of the DALI I/O pin driver layer.

The functions of these files are given below in bold.

Void init_DALI (port_out, pin_out, invert_out, port_in, pin_in, invert_in, DataReceivedFunction, ErrorFunction,RTC_1ms_Function)

This function:

- Initializes the DALI I/O pin driver layer
- Defines the DALI RX and DALI TX pins and their inversion.

This depends on the physical connection to the DALI bus (i.e. the converter between the DALI bus and the I/O pins).

- Defines the callback functions:
  - if the complete DALI packet is received (address, command/data)
  - if an error occurs on the DALI bus (loss of idle voltage)
  - if 1 ms callback timer ticks are generated for the DALI stack layer
- After this function is called, the I/O pin driver is able to receive DALI packets and can call upper layer callbacks. This function should be called as th e last initialization function (after DALI stack initialization).

bool get_DALIIN(),

bool get_DALIOUT(),

void set_DALIOUT(pin_value)

This set of functions:

- Obtains/sets the logical level from/to the DALI pin (DALI Rx and DALI Tx)
- Accepts the hardware implementation of the driver (its eventual inversion)

void receive_data (void)

This function is called from the interrupt routine when the first start bit of DALI packet occurs. It:

- Starts the receiving process

void send_data (byteToSend)

This function:

- Starts the transmission process

void receive_tick (void)

This function is called from systick interrupt service routine when receiving has already started. It:

- Performs bit receiving from the DALI RX pin for building the DALI packet
- Checks the packet errors on the DALI bus

void send_tick (void)

This function is called from systick service routine when transmitting has already started. It:

- Performs transmitting of DALI packet to DALI bus (bit coding to physical layer)

void check_interface_failure(void)

This function:

- Checks the DALI bus idle voltage presence (500 ms disconnection from the DALI bus)
- Calls error callback function if an error occurs

u8 get_timer_count(void)

This function:

- Returns the current systick timer content
- Is used for general purpose use, for example, random number generation

u8 get_flag(void)

This function:

- Returns the current DALI I/O driver state (receiving, transmitting)

void check_interface_failure(void)

This function:

- Checks the DALI bus idle voltage presence (500 ms disconnection from the DALI bus)
- Calls error callback function if an error occurs

u8 get_timer_count(void)

This function:

- Returns the current systick timer content
- Is used for general purpose use, for example, random number generation

u8 get_flag(void)

This function:

- Returns the current DALI I/O driver state (receiving, transmitting)

3.2 DALI stack layer functions

The DALI stack layer implements all DALI commands processed on the logical level according to the DALI specification.

3.2.1 dali.c

These main files contain the most commonly used functions that are called from the user application. They contain initialization of the entire DALI slave library, DALI slave library event flag signaling, command execution calls, and reporting of user hardware error.

void DALI_Init(LightControlFunction)

This function:

- Initializes the entire DALI slave library (both I/O pin driver layer and DALI stack layer and is called from the user application as the main initialization function
- Initializes all necessary modules of the DALI slave library
- Starts execution of receiving commands from the DALI bus, timing management, callback calls, and setting of event flags.

void DALI_InterruptConfig (void)

This function:

- Configures the used IRQ Channels and sets their priority.

u8 DALI_CheckAndExecuteReceivedCommand(void)

This function should be cyclically called from the main user application in the main program loop. It:

- Checks if there is a pending received DALI packet. If so, the given DALI command from the packet is executed.
- Returns the execution status
  - 0 = no pending command
  - 1 = pending DALI command executed
  - 2 = DALI bus error present and processed

u8 DALI_CheckAndExecuteReceivedCommand(void)

This function should be cyclically called from the main user application in the main program loop. It:

- Checks if there is a pending received DALI packet. If so, the given DALI command from the packet is executed.
- Returns the execution status
  - 0 = no pending command
  - 1 = pending DALI command executed
  - 2 = DALI bus error present and processed

u8 DALI_CheckAndExecuteTimer(void)

This function should be cyclically called from the main user application in the main program loop at least every 1 ms. It:

- Checks if there is a pending timer flag. If so, the given timer callbacks (fading functions, timeouts, ...) are executed.
- Returns the number of pending remaining 1 ms timer cycles for a given timer task (example, number of timer cycles to finish light dimming cycle).

void DALI_halt(void)

This function contains the entry to the microcontroller low power state (halt mode without timer interrupt). The user must check before calling this function that there are no pending timer cycles and DALI packets.

void DALI_Set_Lamp_Failure(failure)

This function:

- Sets or resets the failure state of the lighting element (for example, a damaged bulb).

u8 Get_DALI_Random(void)

This function:

- Returns a random number
- Is used for general purpose use

3.2.2 dali_cmd.c

These files contain all DALI commands according to the DALI specification. In addition, they contain auxiliary functions for command processing such as checking command parameters and correct timing.

Many functions execute a given DALI command and other functions execute auxiliary commands. A given function is called automatically when the user calls the DALI_CheckAndExecuteReceivedCommand(void) function.

The most important functions of these files are given below in bold. For information about other functions, the user can refer inside these source files and to the DALI specification.

void DALIC_Init(void)

This function:

- Initializes the entire DALI command modules

u8 DALIC_isTalkingToMe(void)

This function:

- Detects if the received DALI packet is addressed to the device

void DALIC_ProcessCommand(void)

This function:

- Selects the correct command execution according to received data

void DALIC_Direct_Arc(val),

void DALIC_Direct_Arc_NoFade(val)

This function is called from many DALI commands as a request for light intensity change. It:

- Executes the light intensity control

3.2.3 dali_pub.c

These files contain the lower level DALI stack layer functions. The user can modify some functions if required. All functions of these files are called from other DALI stack modules (mainly from the “dali_cmd.c”) as requests to execute lower level commands.

The most important functions of these files are given below in bold.

void DALIP_Init(LightControlFunction)

This function:

- Initializes the public module
- Contains as an input parameter a user callback function for light control:
  - it is called if there is a need to change the light level
  - the user implements it (implementation is user hardware dependent)
  - it has only one input parameter (unsigned int) which is the requested light level
  - (see header file “dali_pub.h” for this function type definition).

void DALIP_Direct_Arc(u8 val)

This function is called from the “dali_cmd.c”. It:

- Executes the light intensity control using a logarithmic curve

u8 DALIP_Getxxx(void),

void DALIP_Setxxx(u8 val)

This set of functions:

- Reads or sets a given “xxx” DALI parameter (usually a DALI register or flag in RAM)

void DALIP_Off(void),

void DALIP_On_And_Step_Up(void),

void DALIP_Step_Down_And_Off(void),

void DALIP_Recall_Max_Level(void),

void DALIP_Recall_Min_Level(void),

void DALIP_Up(void),

void DALIP_Down(void),

void DALIP_Step_Up(void),

void DALIP_Step_Down(void)

This set of functions

- Implements the DALI commands for controlling light level

3.2.4 dali_reg.c

These files contain the DALI register management functions. The DALI registers can be stored in RAM, EEPROM, or ROM memory (see the DALI specification definitions). The functions of these files implement automatic memory selection depending on the register index and register read and write routines.

void DALIR_Init(void)

This function:

- Initializes the DALI register module
- Clears all RAM registers

void DALIR_ResetRegs(void)

This function:

- Sets all registers to the default state (according to the DALI specification)

u8 DALIR_ReadReg(idx),

void DALIR_WriteReg(idx, newval),

u8 DALIR_ReadStatusBit(bit),

void DALIR_WriteStatusBit(bit, val),

u8 DALIR_ReadEEPROMReg(idx),

void DALIR_WriteEEPROMReg(idx, val)

This set of functions:

- Is used for register reading and writing

void DALIR_DeleteShort(void)

This function:

- Clears the DALI short address (unassigned address state)

3.2.5 eeprom.c

These files contain the EEPROM (flash emulated as EEPROM) management, read and write routines.

void EEPROM_Init (void)

This function:

- Initializes the MCU for emulated EEPROM
- Initializes the variables related to emulated EEPROM

void E2_ResetEEPROM(void)

This set of functions:

- Resets the EEPROM header

uint16_t E2_WriteMem(u16 VirtAddress, u16 Data),

void E2_WriteBurst(u16 addr, u16 times, u16* buf)

This set of functions:

- Writes data to EEPROM

u16 E2_ReadMem(u16 VirtAddress,uint16_t* Data)

This set of functions:

- Read data from EEPROM

static FLASH_Status EE_Format(void)

This set of functions:

- Erases PAGE0 and PAGE1 and writes VALID_PAGE header to PAGE0

static uint16_t EE_FindValidPage(uint8_t Operation)

This set of functions:

- Find valid Page for write or read operation

static uint16_t EE_VerifyPageFullWriteVariable(uint16_t VirtAddress, uint16_t Data)

This set of functions:

- Verify if active page is full and Writes variable in EEPROM

static uint16_t EE_PageTransfer(uint16_t VirtAddress, uint16_t Data)

This set of functions:

- Transfers last updated variables data from the full Page to an empty one.

3.2.6 lite_timer_8bit.c

These files are the timing module. They contain all virtual timers which are needed for DALI timing such as intervals for the dimming function and DALI timeouts. Reference timer ticks come from the DALI I/O pin driver layer.

void Timer_Lite_Init(void)

This function:

- Initializes the timing module
- Initializes the internal variables to their default state

void RTC_LaunchTimer(timer_value)

This function:

- Initializes a timeout for DALI command repetition

void RTC_LaunchUserTimer(TimerCount),

void RTC_DoneUserTimer(void)

This set of functions:

- Dims the update intervals (start and stop)

void RTC_LaunchBigTimer(mins)

This function:

- Initializes the timeout (usually 15 minutes) - expiration of the DALI “Initialize” command

void RTC_LaunchDAPCTimer(void)

This function:

- Initializes the timeout (200 ms) - expiration of the DALI “DAPC sequence” command

void PowerOnTimerReset(void)

This function:

- Stops the power-on timeout (600 ms) which was started during power-on (using the Timer_Lite_Init() function)

void Lite_timer_Interrupt(void)

This function is the main timer function servicing all timers. It:

- Is a callback function that receives 1 ms timer ticks from the DALI I/O pin driver layer
- Runs in interrupt
- Updates the timer event flag used for updating the virtual timers (1 ms each)

u8 Process_Lite_timer_IT(void)

This function is called cyclically from the user main program loop in 1 ms intervals when the timer event flag is signalled. It:

- Services timer events (if necessary)
- Calls callbacks or updates virtual timer values

3.2.7 dali_config.c

These files contain the DALI stack module configurations. The user can change them according to his needs, the final ballast hardware/software implementation, the light element used, the type of light control, the STM32 pinout, and the user/device description information.

The most important variables and definitions in these files are given below in bold.

const u16 DALIP_ArcTable[]

This variable is the logarithmic table for light control. It:

- Implements the logarithmic table according to the DALI specification
- Converts 8-bit linear values to 16-bit logarithmic output for direct light power control (see the DALI specification).

const u32 DALIP_FadeTimeTable[]

This variable:

- Implements the fade time table according to the DALI specification

const u16 DALIP_FadeRateTable[]

This variable:

- Implements the fade rate table according to the DALI specification

const u8 DaliRegDefaults[]

This variable:

- Implements the default DALI register content according to the DALI specification

#define USE_ARC_TABLE

This definition:

- Enables the logarithmic table to be used instead of direct light control

#define OUT_DALI_PORT,

#define OUT_DALI_PIN,

#define INVERT_OUT_DALI,

#define IN_DALI_PORT,

#define IN_DALI_PIN,

#define INVERT_IN_DALI

This set of definitions:

- Contains DALI Rx and DALI Tx signal assignments to given pins according to user requirements. The Rx pin must have interrupt capability.
- Define if a pin is inverted or not on the user hardware DALI transceiver board

#define DEVICE_TYPE

This definition:

- Defines the ballast type according to the DALI specification (see DALI specification for valid ballast type numbers).

#define DALI_VERSION_NUMBER_ROM

This definition:

- Defines the DALI version

#define PHYSICAL_MIN_LEVEL_ROM

This definition:

- Defines the minimum light level which is limited by the ballast hardware capability (according to the DALI specification).

4 Application setup

1. Preload the program into the STM32 discovery kit (details in the DALI SW DALI description).

Note: The current version was compiled with IAR 6.3.

2. Select jumper J3 (of STEVAL-ILM001V1).
   a) 1-2 shorted, the direct power supply is selected. It means the module is always supplied with the same supply voltage as a controlling microcontroller (on STM32-discovery kit).
   b) 2-3 shorted, a controlled supply is selected. The module is supplied through the GPIO pin of the microcontroller, allowing to disconnect the receiver optocoupler when not needed (e.g. during transmission) to lower the overall system consumption.
3. Connect the DALI network cable on connector J1 (DALI DA). There is no need to pay particular attention to wire polarity as there is an input diode bridge on the module.
4. Make the connections of J2 connector of STEVAL-ILM001V1 and STM32 discovery as given in the table below and also connect the PC9 pin of the STM32 discovery board to the PA6 pin of the same board to see the effect on LED3 on STM32 discovery board as shown in *Figure 9*.

Table 1. Connections of STEVAL-ILM001V1 and STM32 discovery board

STEVAL-ILM001V1 pins(J2)	STM32 discovery pins
VDD (pin 1)	3V3
GND (pin 2)	GND
RX_Supply (pin 5)	PB9
S1 (pin 6)	PB8
DALI_TX (pin 9)	PB1
DALI_RX(pin 10)	PB0

Figure 6. Bottom side connectors

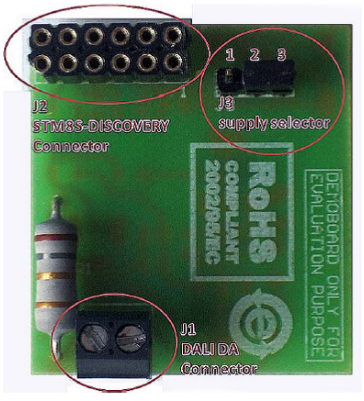
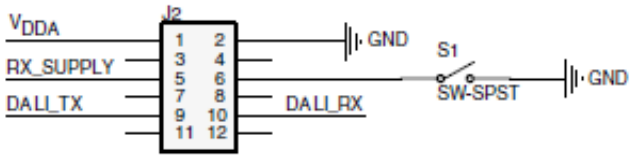


Figure 7. Schematic of connector J2



5. Connect the STM32-discovery board to any USB (no SW or driver installation required, it is used only as the supply for basic evaluation).

Figure 8. STM32-discovery board

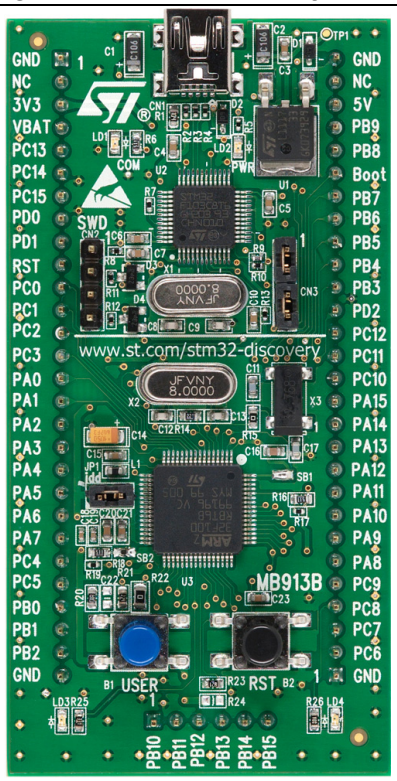


Figure 9. Slave device setup (STEVAL-ILM001V1 and STM32-discovery board)

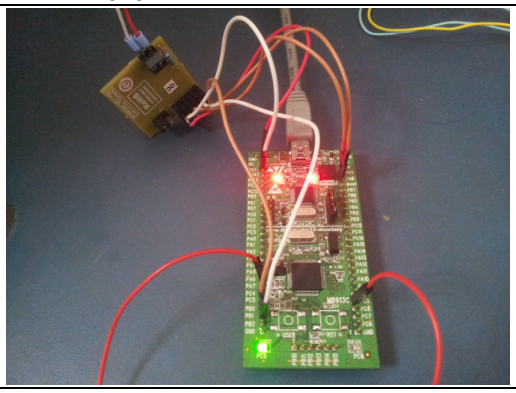
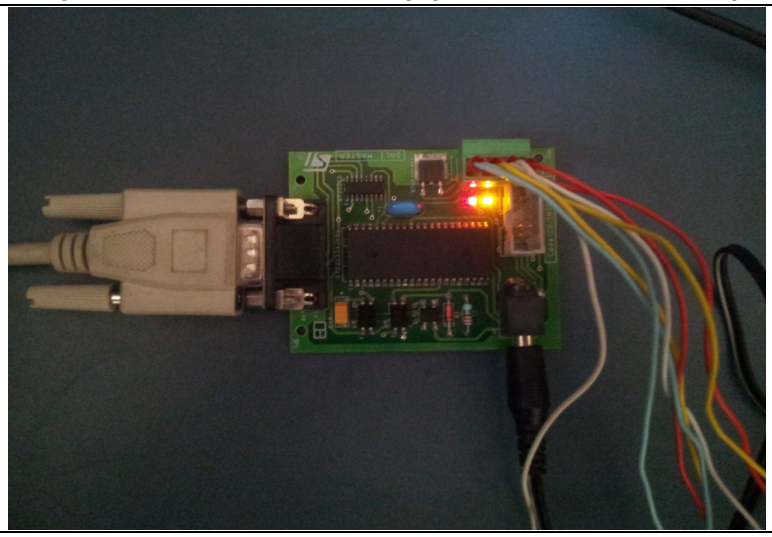
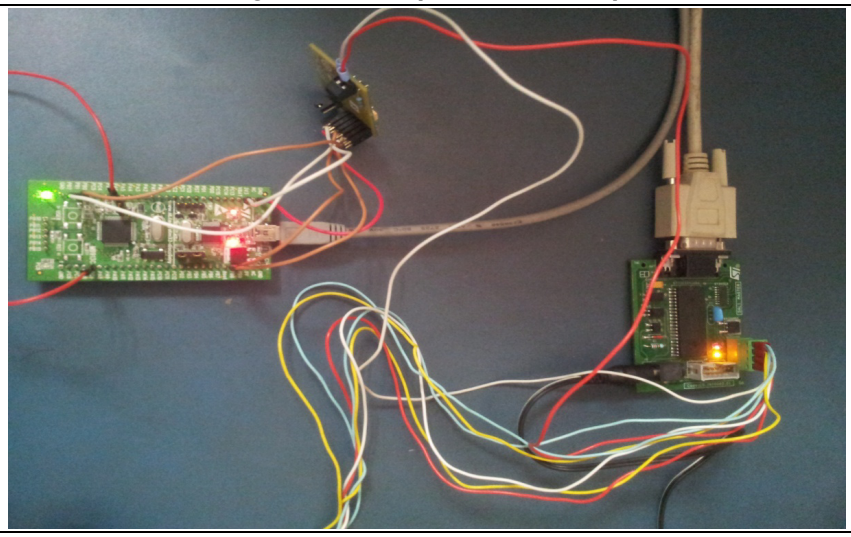


Figure 10. Master device setup (ST7 DALI master board)



6. For the master device connect the serial cable (RS-232) to the PC and the ST7 DALI master board and also connect the 12V adapter to the master board as can be seen in *Figure 10*.
7. The kit is now ready for operation (two LEDs should be ON, one on the STEVAL-ILM001V and one on the STM32-discovery).

Figure 11. Complete DALI setup



5 Application description

The hardware combination (STM32-DISCOVERY + STEVAL-ILM001V1) works as a DALI slave device. The LD3 on the STM32-DISCOVERY kit is used as a light source. Its brightness, fade-in/out times, etc., as defined by the DALI standard, can be controlled through DALI to simulate a normal lamp behavior.

The STEVAL-ILM001V1 provides all the functions required by the DALI standards. The most important of which are:

- Level translation from DALI voltage levels (-6.5 to 22.5 V) to microcontroller levels (3.3 V / 5 V logic)
- Proper rise/fall times for the communication
- Current consumption limit (2 mA max.)
- Overvoltage protection for misconnection of rated mains voltage to DALI DA connectors.

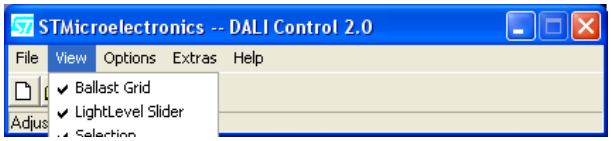
6 Running the PC software for DALI-STM32

The ST7 DALI user interface is available with STM8 DALI example (AN3298). Use the “PowerControl” file in the following path of STM8 package: STM8Sx_AN3298_FW_V2.0.0\Utilities\PC_Software\ST7 DALI Master.

The user interface consists of a main window and 5 other windows. The main window is always visible.

Select “View” menu to choose other visible windows.

Figure 12. Main window



Following windows options appear in drop down menu:

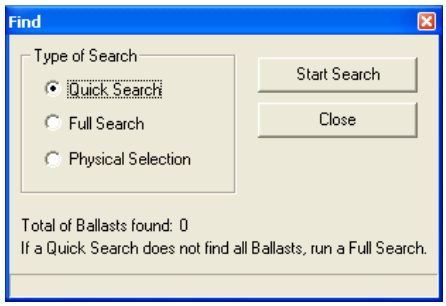
Ballast grid window: ballasts found on the bus after a search light level slider window: buttons and sliders for adjusting light level parameters selection window: buttons for selecting groups scene window: buttons for selecting scenesDALI logger window: displays all DALI commands sent.

6.1 Basic commands

Addressing the ballast

Select “Extras” menu in the main window and select item “Search for ballasts…”. The following window appears:

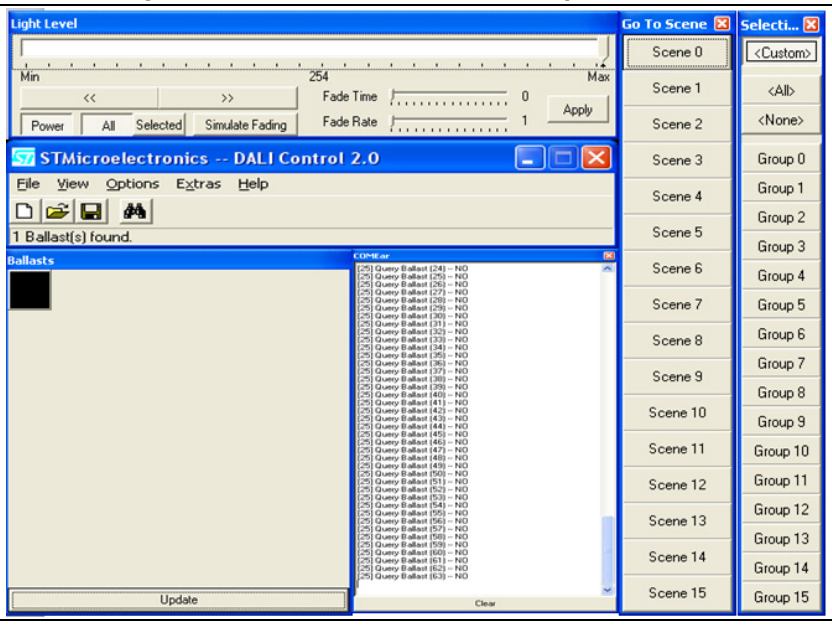
Figure 13. Search for ballast window



Run a “Full Search”. When it is finished, run a “Quick Search”.

The master will find a ballast on the network. The ballast window shows the ballast as shown in *Figure 16*.

Figure 14. Screen overview after a quick search



6.2 Light level

User can choose to change the light level of ballast by pushing the “All” button in the “Light Level” window and change the intensity level using the slider button.

User can also check the fading of the ballast by pushing the “Simulate Fading” button in the light level window and vary the intensity using the slider button.

Figure 15. Light level window

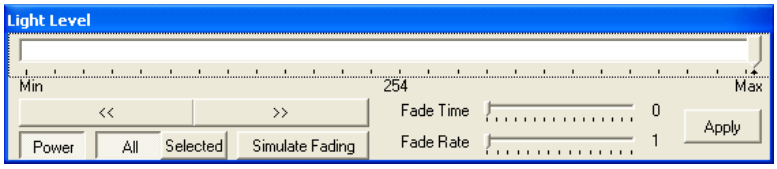


Figure 16. Ballasts window after clicking on update

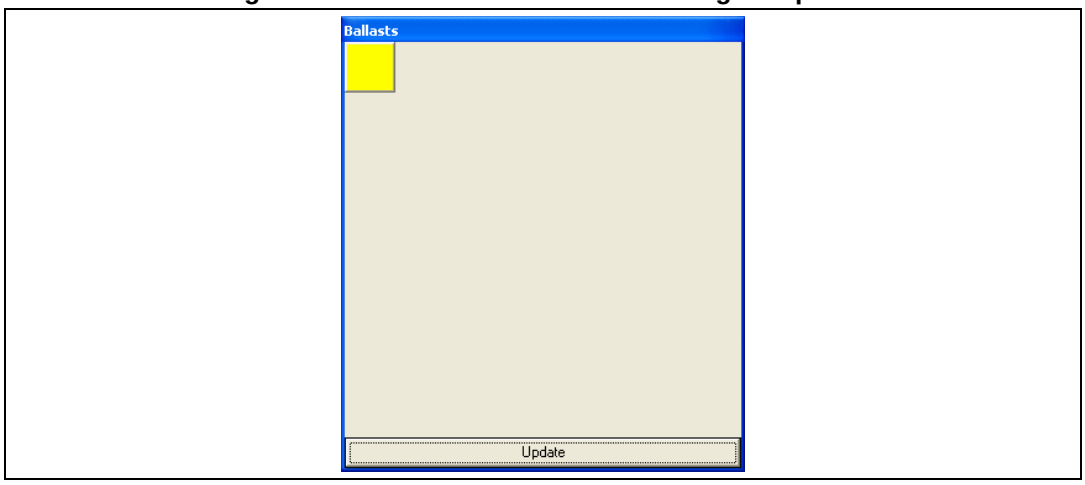
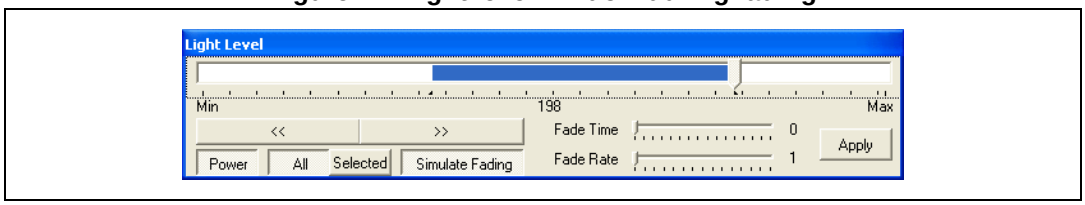


Figure 17. Light level window during fading



Physical selection of ballast can also be done with the help of the button provided on the STEVAL-ILM001V1 board. Select “Extras” menu, item “Search for ballasts …”.and then “Physical selection” in search for ballast window. And then press the button on the board and the ballast is detected by the master device. But during physical selection only one ballast is selected at a time.

7 References

1. AN3298 application note
2. ST7DALI-EVAL evaluation kit user manual
3. UM1032: STEVAL-ILM001V1 hardware module for the STM8S - DISCOVERY interface for DALI communication

8 Revision history

Table 2. Document revision history

Date	Revision	Changes
28-Feb-2014	1	Initial release.

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST’s terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST’S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

ST PRODUCTS ARE NOT DESIGNED OR AUTHORIZED FOR USE IN: (A) SAFETY CRITICAL APPLICATIONS SUCH AS LIFE SUPPORTING, ACTIVE IMPLANTED DEVICES OR SYSTEMS WITH PRODUCT FUNCTIONAL SAFETY REQUIREMENTS; (B) AERONAUTIC APPLICATIONS; (C) AUTOMOTIVE APPLICATIONS OR ENVIRONMENTS, AND/OR (D) AEROSPACE APPLICATIONS OR ENVIRONMENTS. WHERE ST PRODUCTS ARE NOT DESIGNED FOR SUCH USE, THE PURCHASER SHALL USE PRODUCTS AT PURCHASER’S SOLE RISK, EVEN IF ST HAS BEEN INFORMED IN WRITING OF SUCH USAGE, UNLESS A PRODUCT IS EXPRESSLY DESIGNATED BY ST AS BEING INTENDED FOR “AUTOMOTIVE, AUTOMOTIVE SAFETY OR MEDICAL” INDUSTRY DOMAINS ACCORDING TO ST PRODUCT DESIGN SPECIFICATIONS. PRODUCTS FORMALLY ESCC, QML OR JAN QUALIFIED ARE DEEMED SUITABLE FOR USE IN AEROSPACE BY THE CORRESPONDING GOVERNMENTAL AGENCY.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.
Information in this document supersedes and replaces all information previously supplied.
The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

STMicroelectronics group of companies

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

www.st.com