UM2023 User manual

STM32CubeL0 STM32L073Z-EVAL demonstration firmware

Introduction

STMCube™ initiative was originated by STMicroelectronics to ease developers' life by reducing development efforts, time and cost. STM32Cube covers STM32 portfolio.

STM32Cube includes:

- The STM32CubeMX, a graphical software configuration tool that allows to generate C initialization code using graphical wizards.
- A comprehensive embedded software platform, delivered per Series (such as STM32CubeL0 for STM32L0 Series)
  - The STM32CubeL0 HAL, an STM32 abstraction layer embedded software, ensuring maximized portability across STM32 portfolio
  - A consistent set of middleware components such as RTOS, USB, STMTouch and FatFs
  - All embedded software utilities coming with a full set of examples

The STM32CubeL0 demonstration platform running on the STM32L073Z-EVAL evaluation board is built around the STM32Cube HAL BSP and the FatFs middleware component.

This demonstration embeds five applications showing several features supported by the STM32L073 device and using some of the peripherals present on the STM32L073Z-EVAL evaluation board. These five applications are:

- Low-power application
- Thermometer application
- LPUART wake-up application
- LC sensor metering application
- Pressure application

Contents

1 STM32CubeL0 main features 6

2 Getting started with the demonstration 7

2.1 Hardware requirements 7

2.2 Hardware settings of the STM32L073Z-EVAL board 8

2.3 microSD card setup 8

3 Firmware package of the demonstration 11

3.1 Demonstration repository 11

3.2 Programming a firmware application 12

3.3 Software components used by the demonstration 12

3.3.1 STM32L073Z-EVAL Board Support Package 12

3.3.2 Peripheral Board Support Package 13

3.3.3 HAL drivers 13

4 Functional description of the demonstration 15

4.1 Overview 15

4.2 Overview of the different applications 17

4.3 LPUART wake-up application 18

4.3.1 Overview 18

4.3.2 Functional description 18

4.4 Low-power application 21

4.4.1 User interaction 21

4.4.2 Functional description 22

4.5 Thermometer application 23

4.5.1 User interaction 23

4.5.2 Functional description 24

4.6 LC sensor metering application 25

4.6.1 Overview 25

4.6.2 LC sensor metering in standard mode 26

4.6.3 LC sensor metering in low-power mode 27

4.6.4 Block diagram of the LC sensor metering 27

4.6.5 Application principle 28

4.6.6 Capture window of the LC sensor metering 30

4.7 Pressure application 31

4.7.1 User interaction 31

4.7.2 Functional description 32

5 Revision history 33

List of tables

Table 1. Files to be copied into the microSD card 9
Table 2. Document revision history 33

List of figures

Figure 1. STM32Cube block diagram 6
Figure 2. Demonstration starting with the STM32L073Z-EVAL evaluation board 7
Figure 3. Jumper configuration 8
Figure 4. Icons of the applications 9
Figure 5. Behavior of the demonstration with a wrong microSD card setup 10
Figure 6. Folder structure. 11
Figure 7. Demonstration main menu 15
Figure 8. Application selection using the joystick. 16
Figure 9. Structure of the demonstration menu 17
Figure 10. LPUART application (step 1 up to step 5) 20
Figure 11. LPUART application (step 6 up to step 8) 20
Figure 12. Power mode (user interaction) 21
Figure 13. Thermometer application 23
Figure 14. Thermometer application behavior 24
Figure 15. Using detection accessory board MB1199 with LC sensor metering 26
Figure 16. LC sensor metering standard mode 27
Figure 17. Block diagram of the LC sensor metering. 28
Figure 18. Main signals during LC sensor activation 29
Figure 19. Oscillations collected with LC sensor in air on PB4 29
Figure 20. Oscillations collected with LC sensor in proximity of metal on PB4 30
Figure 21. Display of the pressure application. 31
Figure 22. The three different views associated to the pressure application 32

1 STM32CubeL0 main features

STM32CubeL0 gathers together, in a single package, all the generic embedded software components required to develop an application on STM32L0 microcontrollers. In line with the STM32Cube initiative, this set of components is highly portable, not only within the STM32L0 Series but also to other STM32 Series.

STM32CubeL0 is fully compatible with the STM32CubeMX code generator that allows users to generate initialization code. The package includes a low-level hardware abstraction layer (HAL) that covers the microcontroller hardware, together with an extensive set of examples running on the STMicroelectronics boards.

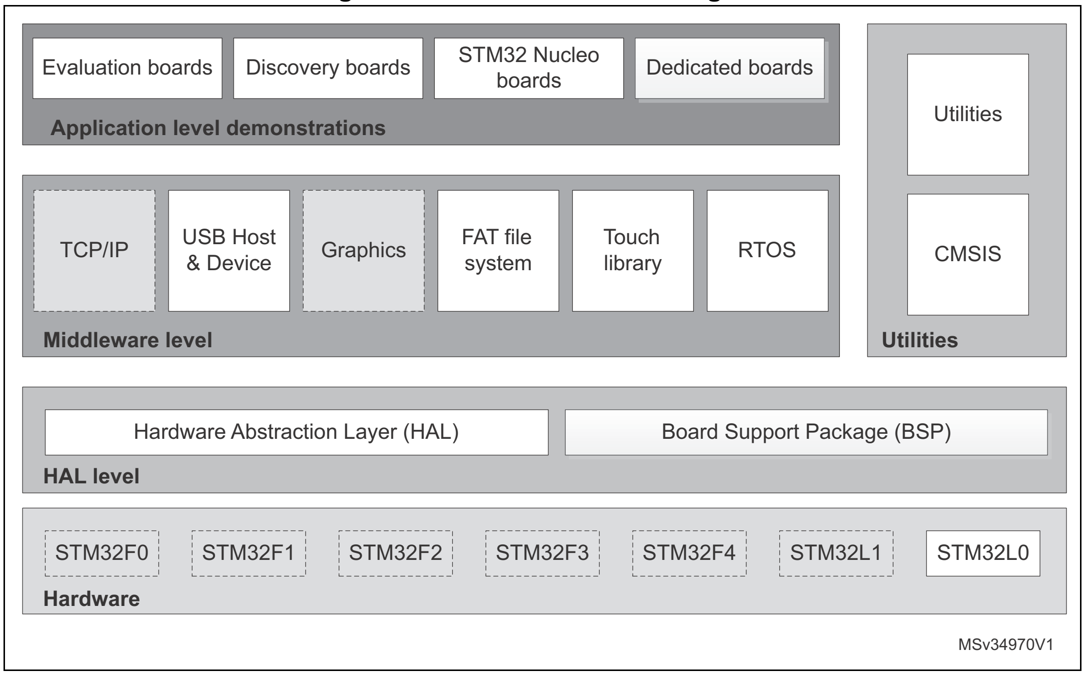
For user convenience the HAL is available in an open-source BSD license.

STM32CubeL0 package features a set of middleware components with the corresponding examples. They come with very permissive license terms:

- Full USB device stack supporting many classes (HID, MSC, CDC and DFU)
- CMSIS-RTOS implementation with FreeRTOS open source solution
- FAT file system based on open source FatFs solution
- STMTouch touch-sensing solution

The demonstration described in this document is included in the STM32CubeL0 package. The block diagram of STM32Cube is shown in *Figure 1*.

Figure 1. STM32Cube block diagram



2 Getting started with the demonstration

2.1 Hardware requirements

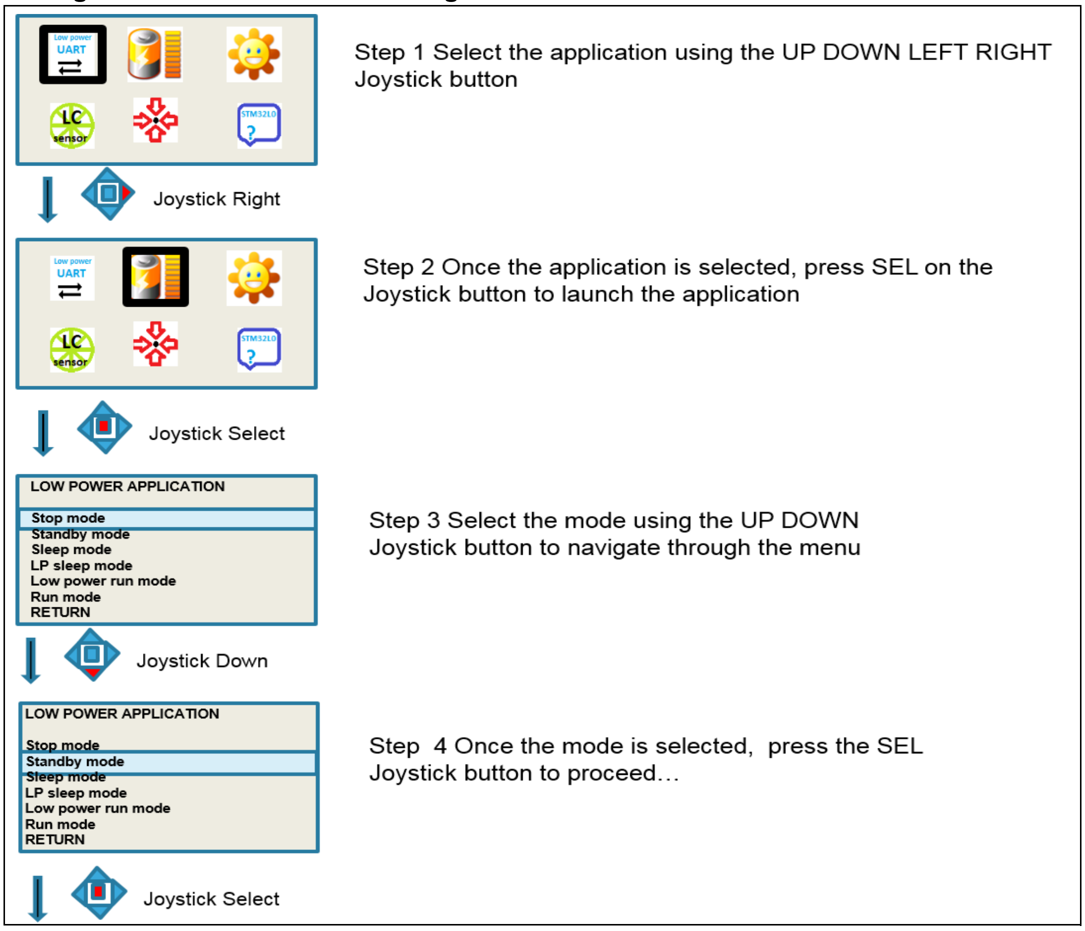
The hardware requirements to start the demonstration are the following:

- STM32L073Z-EVAL evaluation board (see *Figure 2*)
- One microSD card - SDHC (up to 32 Gbytes)
- One USB cable to power up the STM32L073Z-EVAL board from the ST-LINK USB

The STM32L073Z-EVAL evaluation board is connected to the USB cable to the host PC and does not need any external power supply. A detection accessory board MB1199 is also provided with the evaluation board and is used to simulate the presence of metal over the LC sensor, when running the LC sensor application.

The demonstration displays icons which are stored on a microSD card. The microSD card must contain several .bmp files available under the firmware package directory: /Utilities/Media/Pictures folder. For more details refer to *Section 3: Firmware package of the demonstration*.

Figure 2. Demonstration starting with the STM32L073Z-EVAL evaluation board



Step 1 Select the application using the UP DOWN LEFT RIGHT Joystick button

Step 2 Once the application is selected, press SEL on the Joystick button to launch the application

Step 3 Select the mode using the UP DOWN Joystick button to navigate through the menu

Step 4 Once the mode is selected, press the SEL Joystick button to proceed…

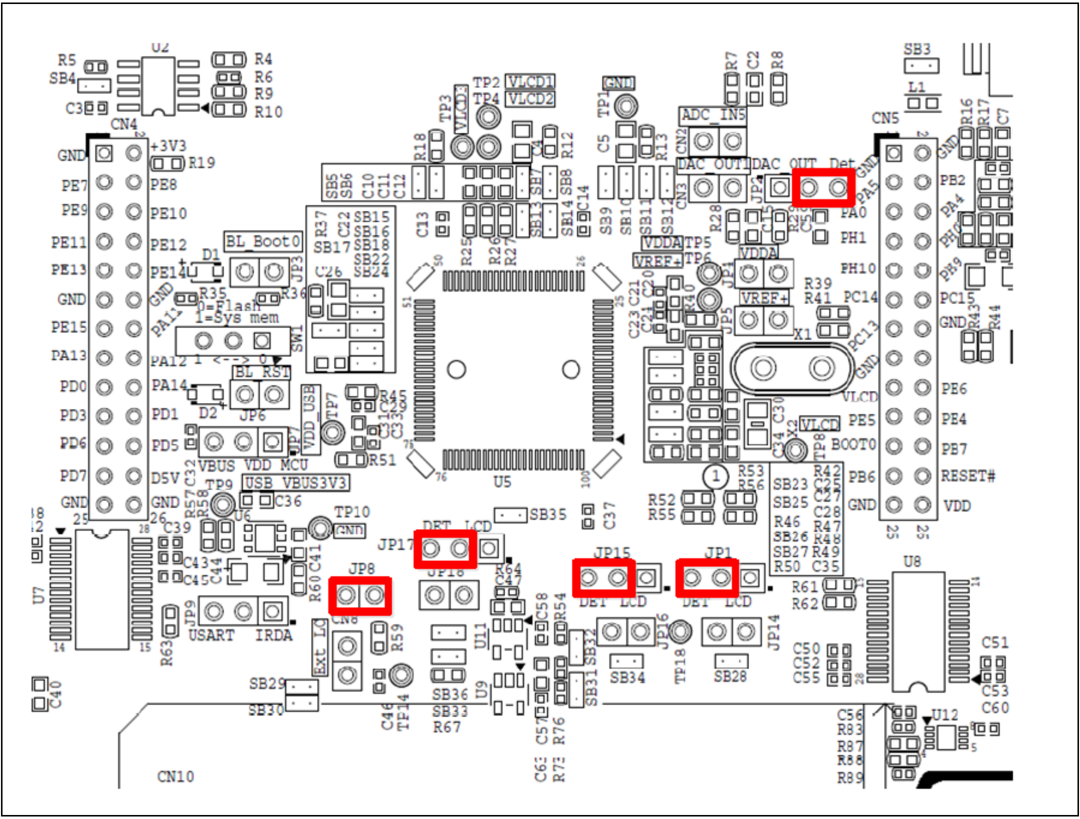
2.2 Hardware settings of the STM32L073Z-EVAL board

The SBx solder bridges should be set in their initial factory configuration. All JP1, JP2, JP15 and JP17 jumpers must be set in the "DET" position (detection) and JP8 fitted as shown in *Figure 3: Jumper configuration*, where these jumpers are marked in red.

The LCD glass must be put in I/O position.

If the user wants to measure the average current during the LC sensor metering in low-power mode (few µA range), an ammeter can be inserted on JP10 between pin 1 (V_{DD}) and 2 (V_{DD}-MCU). Pay attention when measuring pulsed current. Should some tests be performed with an external LC sensor, remove SB29 and SB30 and connect the new LC cell on CN8 (external LC sensor).

Figure 3. Jumper configuration



2.3 microSD card setup

To run the demonstration, the microSD card must contain several .bmp files available inside the firmware package directory: /Utilities/Media/Pictures.

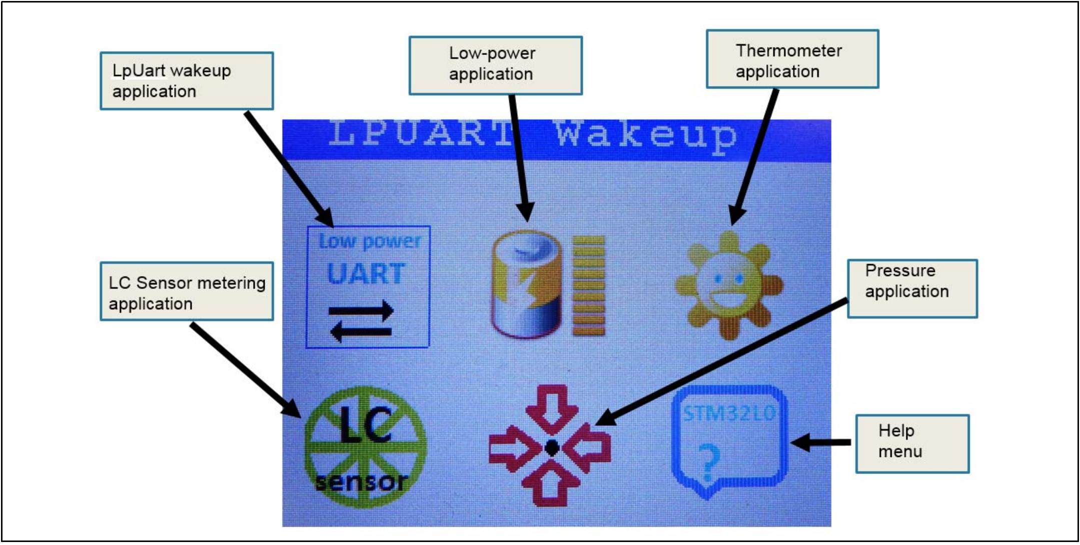
Table 1 shows the list of the mandatory files, which must be copied into the microSD card:

Table 1. Files to be copied into the microSD card

Firmware package files to copy	Location inside microSD card
/Utilities/Media/Pictures/BMP_64x64/Help.bmp	STFILES/Help.bmp
/Utilities/Media/Pictures/BMP_64x64/LCsensor.bmp	STFILES/LCsensor.bmp
/Utilities/Media/Pictures/BMP_64x64/Power.bmp	STFILES/Power.bmp
/Utilities/Media/Pictures/BMP_64x64/Pressure.bmp	STFILES/Pressure.bmp
/Utilities/Media/Pictures/BMP_64x64/Temp.bmp	STFILES/Temp.bmp
/Utilities/Media/Pictures/BMP_64x64/Uart.bmp	STFILES/Uart.bmp
/Utilities/Media/Pictures/BMP_128x160/STLogo.bmp	STFILES/STLogo.bmp

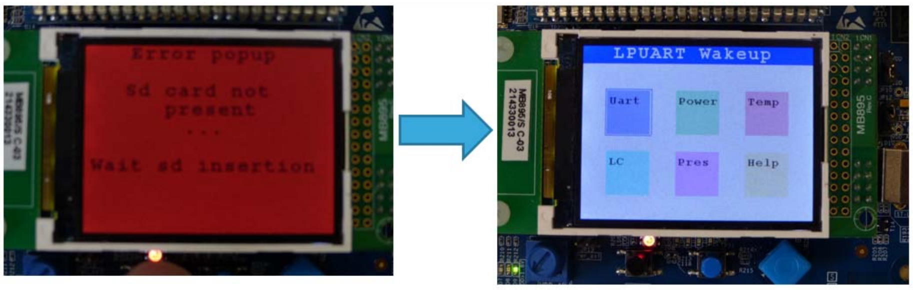
During the startup phase of the demonstration, the user gets the following icons displayed on the STM32L073Z-EVAL evaluation board. Each icon corresponds to a specific application, as shown in *Figure 4*:

Figure 4. Icons of the applications



If the microSD card is not correctly inserted or well programmed, an error message is displayed, and basic icons are displayed instead (see *Figure 5: Behavior of the demonstration with a wrong microSD card setup*). Nevertheless, in this case, it is still possible to launch the different applications present inside the demonstration.

Figure 5. Behavior of the demonstration with a wrong microSD card setup

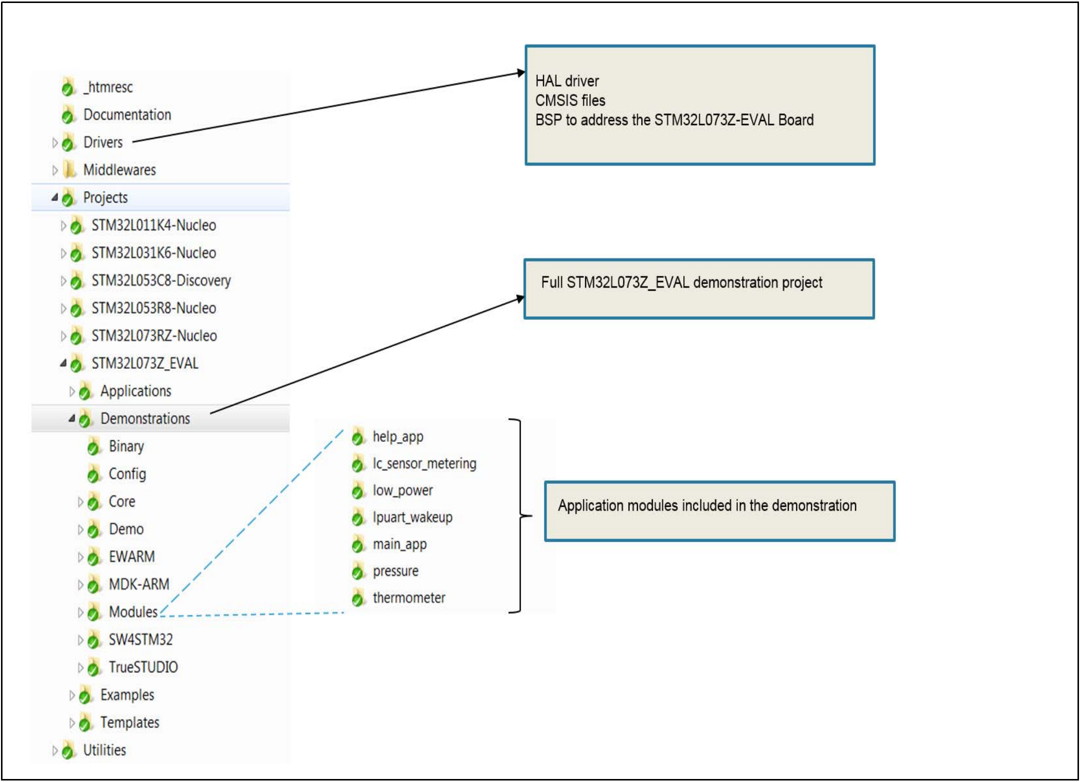


3 Firmware package of the demonstration

3.1 Demonstration repository

The demonstration is provided within the STM32CubeL0 firmware package inside the STM32L073Z-EVAL project, as shown in *Figure 6*.

Figure 6. Folder structure



The demonstration sources are located in the project folders of the STM32Cube package corresponding to the STM32L073Z-EVAL evaluation board. The sources are grouped into several folders described as follows:

1. **Binary**: demonstration binary file in Hex format.
2. **Config**: contains configuration files including the standard stm32l0xx_hal_conf.h file.
3. **Core**: contains the files in charge of the user interface (menu management, microSD card management in charge of displaying the different icons).
4. **Demo**: contains the main.c file (demonstration entry point) and the ARM® Cortex®-M0+ exception handlers requested for this demonstration.
5. **EWARM, MDK-ARM, SW4STM32, TrueSTUDIO**: inside these folders are located the startup files, the project configuration files and the linker files associated to each toolchain.
6. **Modules**: contains the source code associated to the different applications supported by the demonstration. There is a dedicated folder for each application: help_app, lc_sensor_metering, low_power, lpuart_wakeup,main_app, pressure and thermometer.

3.2 Programming a firmware application

To program the STM32L073Z-EVAL evaluation board with the demonstration, proceed as follows:

- Install the preferred Integrated Development Environment (IDE)
- Install the ST-LINK driver available at the ST website *www.st.com*

There are two methods of programming the STM32L073Z-EVAL evaluation board:

- **Method 1**:
  Upload the STM32CubeL0_Demo_STM32L073Z-EVAL.hex from the firmware package available under Projects\STM32L073Z-EVAL\Demonstrations\Binary, using the preferred in-system programming tool.
- **Method 2**:
  Choose one of the two supported toolchains (IAR™/ Keil®/SW4STM32) and follow the steps below:
  - Open the application folder: Projects\STM32L073Z-EVAL\Demonstrations\
  - Chose the desired IDE project (EWARM for IAR, MDK-ARM™ for Keil, SW4STM32 for System Workbench STM32)
  - Load the project file (for example Project.eww for EWARM)
  - Rebuild all files
  - Load the project image
  - Run the program

3.3 Software components used by the demonstration

3.3.1 STM32L073Z-EVAL Board Support Package

For the STM32L073Z-EVAL evaluation board, a dedicated Board Support Package (BSP) implementing the board capabilities is available inside the firmware package. This BSP (available under /Driver/BSP/STM32L073Z_EVAL) supports the following items:

- eeprom (not used for the demonstration)
- lcd glass (not used for the demonstration)
- lcd (used for the demonstration)
- current measurement (used for the demonstration)
- I/O configuration including push-button, leds and joystick setup (used for the demonstration)
- microSD card (used for the demonstration)
- temperature sensor (used for the demonstration)

The microSD slot available on the STM32L073Z-EVAL evaluation board uses 4-wire SPI to communicate with the STM32L073. The microSD is controlled by this BSP, which exports in a generic way the SDIO operations needed for its process.

The STM32L073Z-EVAL BSP uses the services of the specific peripheral BSP to control in particular the LCD (hx8347), the MFX (mfxstm32l152), and the temperature sensor (stlm75).

3.3.2 Peripheral Board Support Package

The demonstration uses several peripherals present on the STM32L073Z-EVAL evaluation board. Some of these peripherals are managed by a specific BSP.

- **hx8347**: this BSP controls the HX8347D LCD device. It is available under /Driver/BSP/component/hx8347
- **mfxstm32l152**: this BSP controls the MFXSTM32L152 I/O Expander devices. The joystick and the current measurement are managed through the MFX component available on the STM32L073Z-EVAL evaluation board. This BSP is available under /Driver/BSP/component/mfxstm32l152
- **stlm75**: this BSP controls the stlm75 temperature sensor. It is available under /Driver/BSP/component/stlm75

The MFX component is controlled by the STM32L0773Z component through an I^2C bus.

3.3.3 HAL drivers

The demonstration uses several HAL drivers to control the different IPs present on the STM32L073xx device. The following list details the usage of some of the IPs requested for the demonstration.

GPIOs

GPIOs are used in particular for LED activation and push-button management.
Some specific GPIOs are also used to control the signals needed for the LC sensor metering application:

- PE4 set as output GPIO to drive the LED LD1.
- PD7 set as output GPIO to control the $V_{DD}/2$ reference bias. This GPIO power supplies the operational amplifier that is designed to generate the $V_{DD}/2$ reference voltage. This allows switching on or off this amplifier and save power consumption.
- PB4 is alternatively used as output GPIO (by-default function) and as non-inverting comparator input (alternate function). As GPIO, it drives the LC sensor. As comparator input, it senses the excitation pulse, when delivered.

DAC

DAC_OUT1 is used to generate the input threshold voltage needed by the comparator. Oscillations signals from LC sensor are referenced to $V_{DD}/2$. The DAC peripheral allows to provide a preset threshold voltage value near $V_{DD}/2$.

COMP2

COMP2 is connected to the LC sensor network. As soon as the excitation pulse has been delivered, the COMP2 positive input collects analog oscillations referenced to $V_{DD}/2$ bias, while its negative input is fed by the DAC threshold voltage. Finally, the LC sensor oscillations are conditioned and as a consequence digital pulses are delivered on the comparator output. This signal is then transmitted to the LPTIM. COMP2 is used in high-speed mode, according to the signal frequency emitted by the LC. The DAC output is internally connected to the inverting input of COMP2 and PB4 to its non-inverting input.

LPTIM

LPTIM is configured to count the pulses generated by the LC sensor oscillations and conditioned by the COMP2 comparator. The LPTIM internal counter is read before the LC sensor generates the oscillations. At the end of the oscillations period, the new LPTIM counter value is subtracted from the previous value, to get the number of collected pulses for the corresponding measurement. This avoids a time-consuming LPTIM reset.

RTC

On the LC sensor application, the RTC peripheral is used to wake up the microcontroller at regular intervals once the STM32 has entered Stop mode. The RTC clock is connected to the LSE clock, whose frequency is 32.768 kHz. The wake-up frequency corresponds to the LC sensor capture frequency and it is set to 32 Hz for this example.

The RTC is also used on the low-power application, to show how to exit from low- power mode.

TIM

TIM is configured to generate precise temporizations used for the LC sensor metering sequence especially in standard demonstration mode.

LPUART

LPUART is used on LPUART wake-up application. Through this UART, messages are exchanged between the board and the hyperterminal.

SPI

The SPI is used in particular when accessing to the microSD card.

I^2C

The I^2C interface is used in particular when accessing the MFX companion chip (joystick management and current measurement).

ADC

The ADC is used for the pressure application.

4 Functional description of the demonstration

4.1 Overview

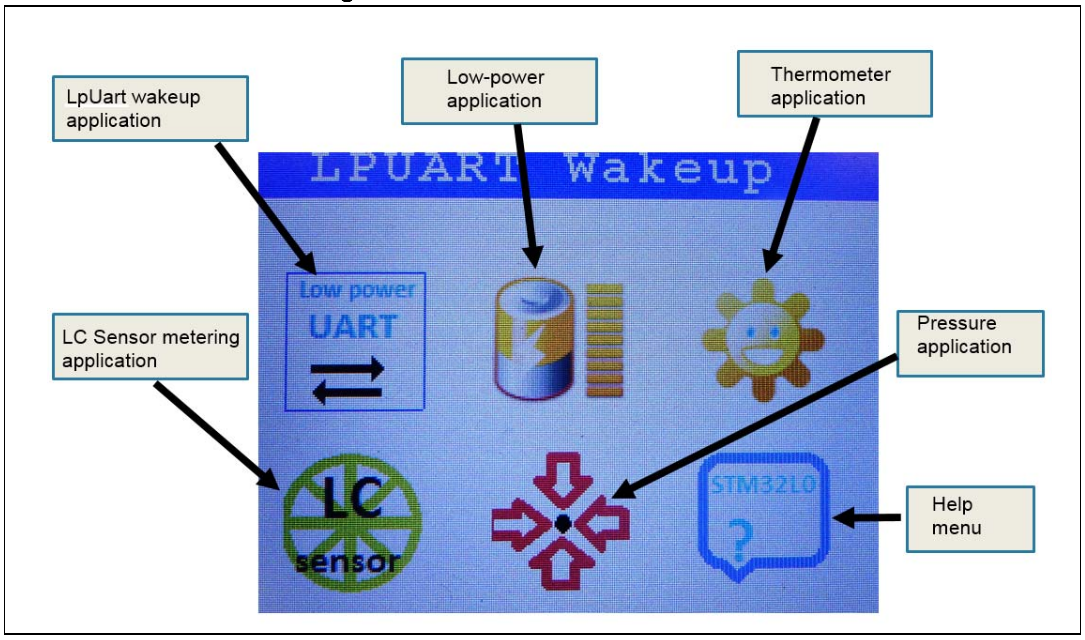
The purpose of the demonstration is to bring out the capabilities of the STM32L073 microcontroller. It uses several peripherals present on the STM32L073Z-EVAL evaluation board.

At start, when launching the STM32L073Z-EVAL demonstration, six icons are displayed on the TFT LCD present on the board. Through these icons it is possible to execute different independent applications:

- Low-power
- Thermometer
- LPUART wake-up
- LC sensor metering
- Pressure

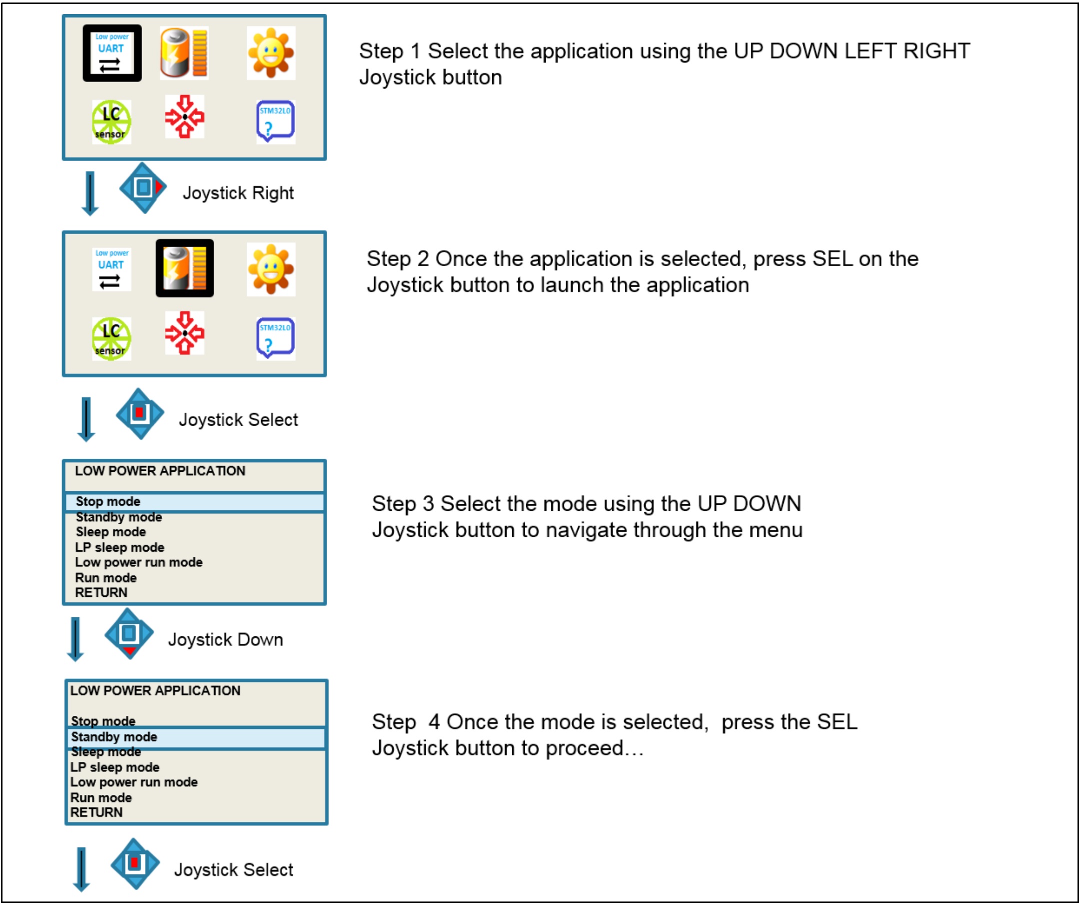
The UP, DOWN, RIGHT and LEFT joystick directions allow the user to navigate among these icons. To select one application, the user has to press SEL on the joystick.

Figure 7. Demonstration main menu



The joystick also enables the navigation through the different submenus, as shown in *Figure 8*.

Figure 8. Application selection using the joystick



Step 1 Select the application using the UP DOWN LEFT RIGHT Joystick button

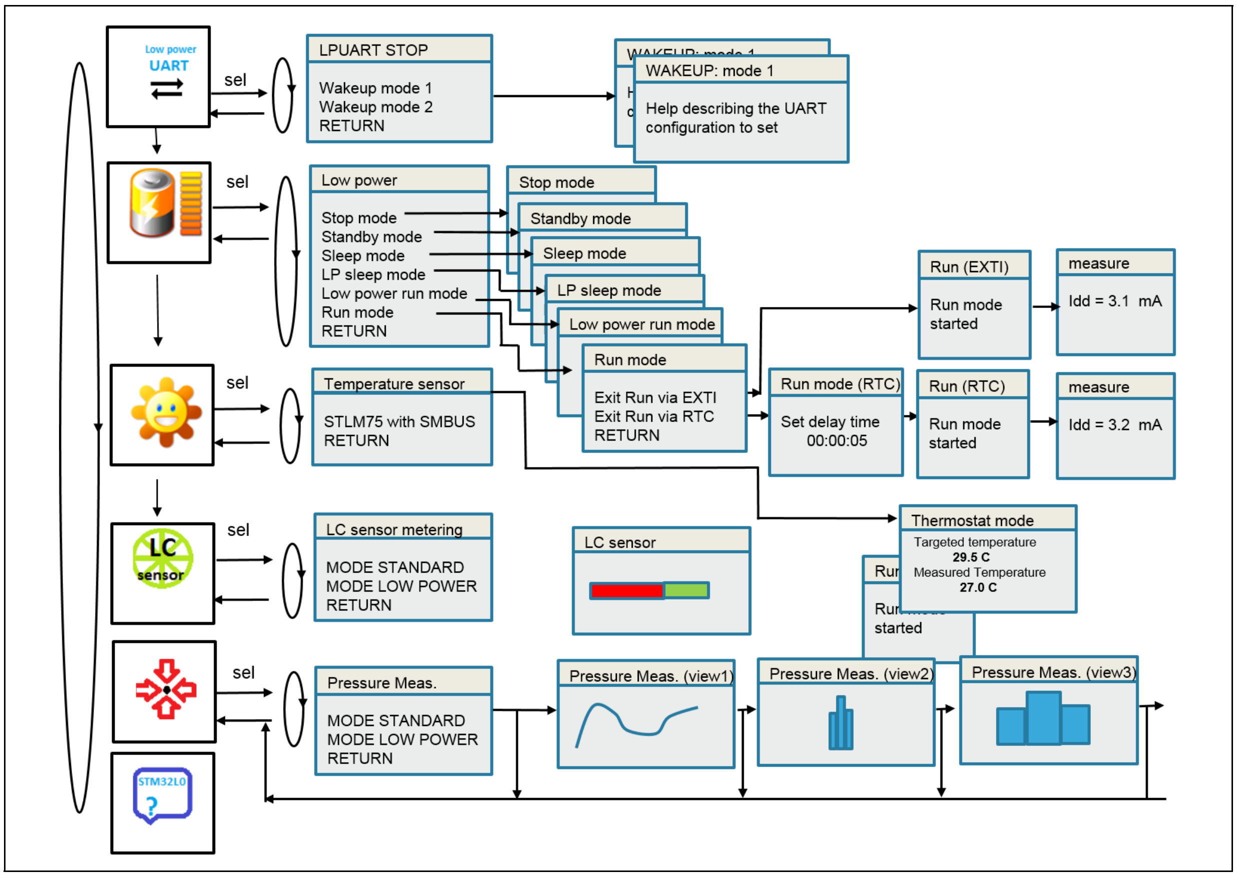
Step 2 Once the application is selected, press SEL on the Joystick button to launch the application

Step 3 Select the mode using the UP DOWN Joystick button to navigate through the menu

Step 4 Once the mode is selected, press the SEL Joystick button to proceed…

The menu structure of the whole demonstration is shown in *Figure 9*.

Figure 9. Structure of the demonstration menu



4.2 Overview of the different applications

Through this demonstration, it is possible to launch five independent applications:

- LPUART wake-up application: this application demonstrates the wake-up of the platform from a terminal connected to CN7 (RS232) in different modes. After wake-up, the current consumption measured by a companion chip (mfx) during Stop mode is displayed.
  For more details refer to *Section 4.3: LPUART wake-up application*.
- Low-power application: this application allows to enter in the different low-power modes supported by the STM32L073xx. The wake-up from low-power mode can be ordered by an EXTI or an alarm. After wake-up, the current consumption measured by a

companion chip (mfx) is displayed. For more details refer to *Section 4.4: Low-power application*.

- Thermometer application: this application displays the temperature (sensor stlm75) using an SMBUS. For more details refer to *Section 4.5: Thermometer application*.
- LC sensor metering: this application shows how the LC sensor is managed. Two modes are available: standard mode and low-power mode:
  - Standard mode shows graphically on the display the LC sensor sensitivity to metal on the display
  - Low-power mode minimizes the power consumption of the detector to achieve years of operation. For more details refer to *Section 4.6: LC sensor metering application*.
- Pressure application: this application demonstrates the capability of the 16-bit ADC oversampling mode from the one available in the STL32L073xx component. The current air pressure value is measured from the sensor U1 available on the STM32L073Z-EVAL evaluation board. For more details refer to *Section 4.7: Pressure application*.

4.3 LPUART wake-up application

4.3.1 Overview

This application demonstrates the wake-up of the platform from a terminal connected to CN7 (RS232) in different modes. After waking up, the current consumption measured by a companion chip (mfx) during Stop mode is displayed on the STM32L073Z-EVAL evaluation board.

To run this application user needs a PC with an hyperterminal software installed. This hyperterminal must be correctly set according to the instructions displayed on TFT screen, for example:

- BaudRate = 9600
- Data = 7 bits
- Parity = ODD
- Stop = 1 bit
- HwFlowCtl = NONE for recall.

Moreover, the LCD glass (MB979) must be moved down in I/O position on CN10 and CN14 connectors. For correct application behavior, the jumper JP10 must be moved to IDD position and the jumper JP12 must be moved to +3V3 position.

4.3.2 Functional description

- Step 1: configuration of the UART.
  The LPUART1 available on the STM32L073xx is configured for the communication with the hyperterminal.
- Step 2: selection of the wake-up mode.
  Though the menu displayed by the application, the user can select one of the two following wake-up modes:
  - **wake-up mode 1**: in this case, the STM32L073xx is waked up on a start-bit detection. The wake-up event is set to UART_WAKEUP_ON_STARTBIT. In this

  case, any character sent from the hyperterminal back to the STM32L073Z-EVAL board wakes up the STM32L073xx.
  - **wake-up mode 2**: in this case, the STM32L073xx is waked up on a specific character. The wake-up event is set to UART_WAKEUP_ON_ADRESS. In this case, only the character 'R' sent from the hyperterminal back to the STM32L073Z-EVAL board wakes up the STM32L073xx.

- Step 3: configuration of the system.
  At this stage, the system performs the following actions:
  - Configuration of the wake-up from UART mode using the HAL_UARTEx_StopModeWakeUpSourceConfig() function.
  - Enabling of the UART wakeup from stop interrupt.
  - Saving of the GPIO configuration
- Step 4: message displayed to guide the user.
  On the STM32L073Z-EVAL board, the following message is displayed:

  Please set JP10 to IDD, JP12 to +3V3 and connect an Hyperterminal with following settings: BaudRate = 9600, Data = 7 bits, Parity = ODD, Stop = 1 bit, HwFlowCtl = NONE.

  The user must follow the instruction above, before pressing the SEL joystick button to reach the next step.
- Step 5: sending a message to the hyperterminal.
  Once the user has correctly setup the connection between the STM32L073Z-EVAL board and the PC through the CN7 (RS232) connector, the STM32L073 LPUART sends a message to the hyperterminal. The sending of this message by the LPUART1 is done in interrupt mode.
- Step 6: entering in low-power mode.
  At this stage, the system enters in Stop mode. Just before entering in Stop mode a request to perform a current measurement is sent to the mfx companion chip.
- Step 7: wake-up of the system.
  If the user has selected the 'wakeup mode 1' (refer to step 2), any character typed on the hyperterminal wakes up the STM32L073Z-EVAL board.
  If the user has selected the 'wakeup mode 2' (refer to Step 2), only the character 'R', typed on the hyperterminal, will wake up the STM32L073Z-EVAL board.
- Step 8: restore the previous configuration.
  Once the system is waked up, the GPIOs are set back to the state they were just before entering in low-power mode. The clock is also set back to its default configuration: CONFIG_CLOCK_HSE_32M.
  When exiting from low-power mode, the application returns to the state it was before entering in low-power mode.
- Step 9: display the current.
  The value of the current consumption measured by the mfx companion chip is displayed on the STM32L073Z-EVAL evaluation board.

Figure 10. LPUART application (step 1 up to step 5)

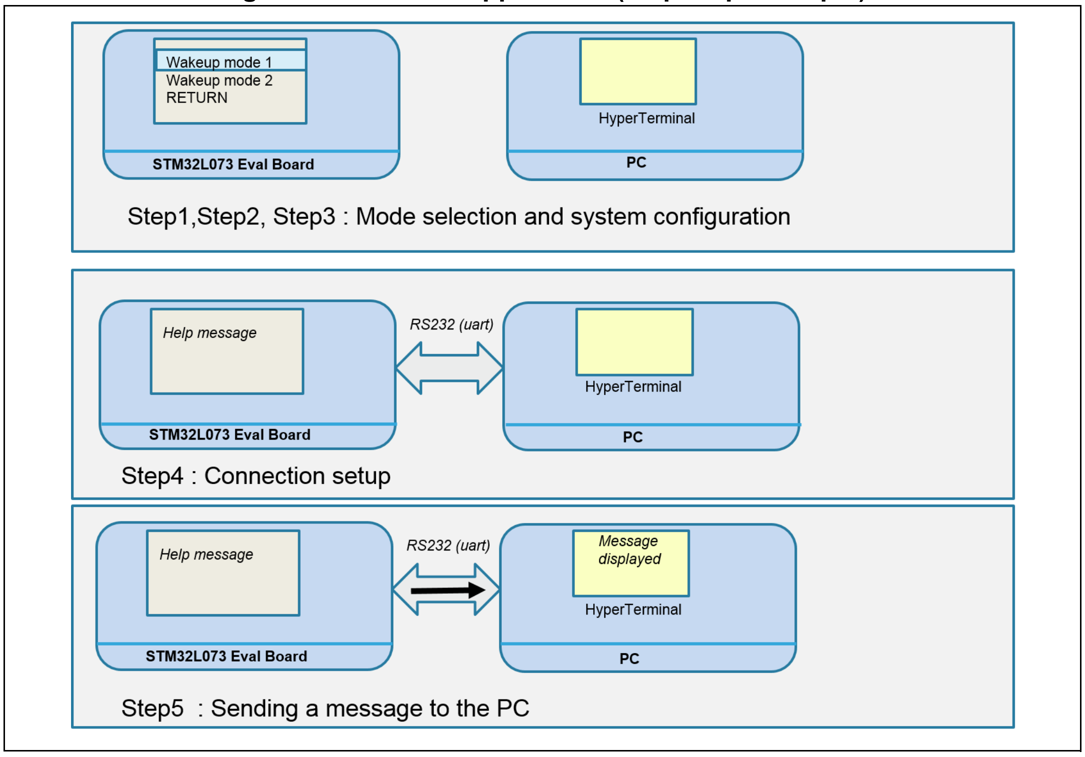
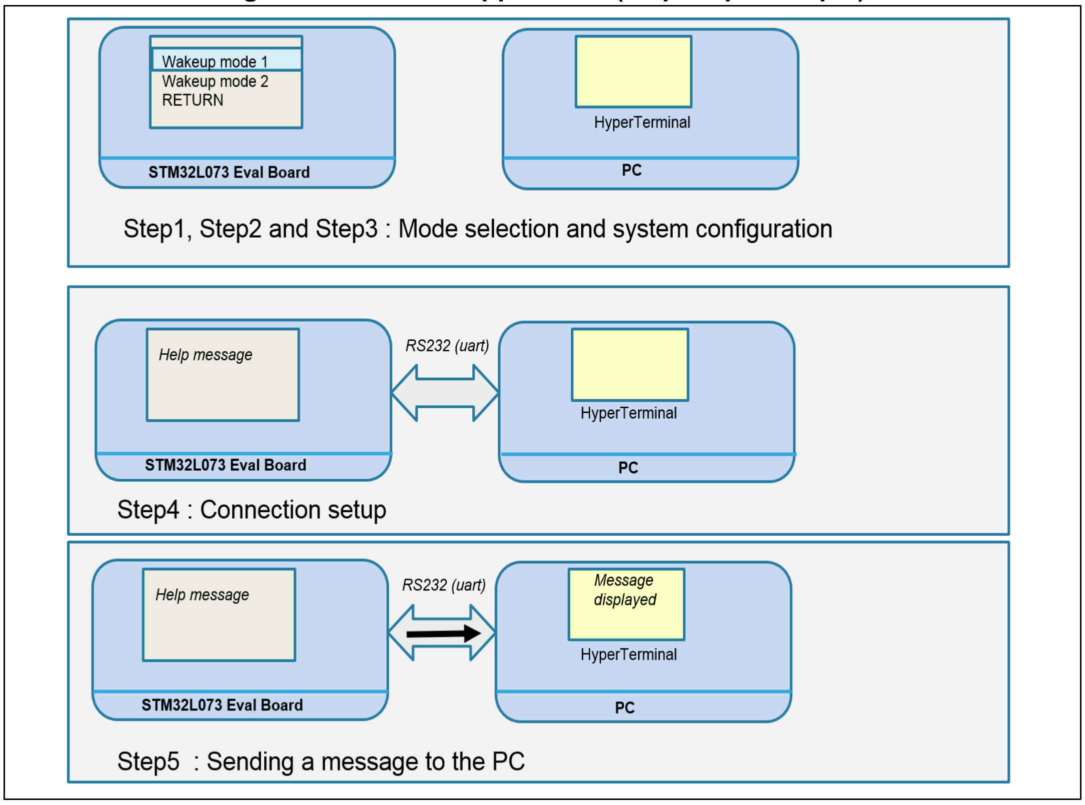


Figure 11. LPUART application (step 6 up to step 8)

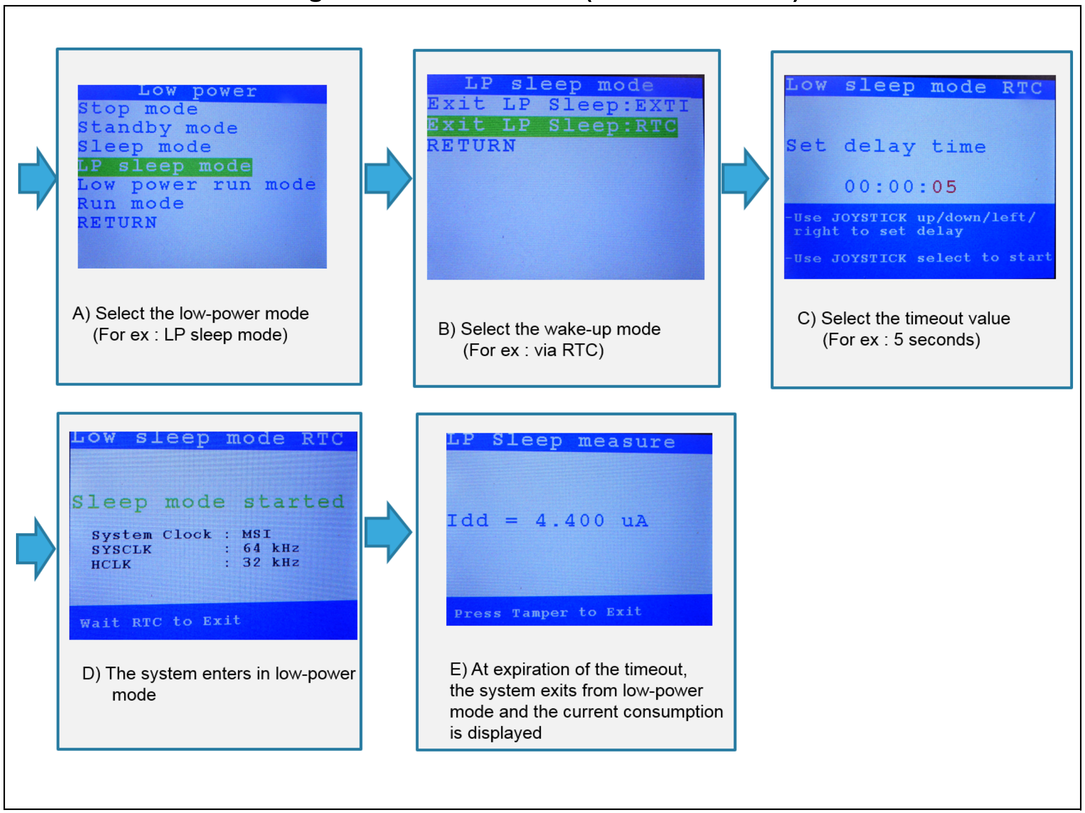


4.4 Low-power application

4.4.1 User interaction

Through the different menus, the user selects the low-power mode and the way the system exits the low-power mode. When the system exits the low-power mode, the current consumption measured is displayed. The purpose of this application is to show how to enter and exit the various low-power modes supported by the STM32L0L073 (see *Figure 12*).

Figure 12. Power mode (user interaction)



```
Low power
Stop mode
Standby mode
Sleep mode
LP sleep mode
Low power run mode
Run mode
RETURN
```

A) Select the low-power mode (For ex : LP sleep mode)

```
LP sleep mode
Exit LP Sleep:EXTI
Exit LP Sleep:RTC
RETURN
```

B) Select the wake-up mode (For ex : via RTC)

```
Low sleep mode RTC

Set delay time

00:00:05

-Use JOYSTICK up/down/left/
 right to set delay
-Use JOYSTICK select to start
```

C) Select the timeout value (For ex : 5 seconds)

```
Low sleep mode RTC

Sleep mode started

System Clock  :  MSI
SYSCLK        :  64 kHz
HCLK          :  32 kHz

Wait RTC to Exit
```

D) The system enters in low-power mode

```
LP Sleep measure

Idd = 4.400 uA

Press Tamper to Exit
```

E) At expiration of the timeout, the system exits from low-power mode and the current consumption is displayed

4.4.2 Functional description

When the low-power mode application is called, the following steps occur:

- Step 1: selection of the low-power mode.
  Through the menu displayed by the application, the user selects the low-power mode he wants to use. The user can select one of the following modes:
  - Run mode: default mode after a power reset
  - Low-power-run mode: regulator in low-power mode, limited clock frequency, limited number of peripherals
  - Sleep mode: ARM® Cortex®-M0+ core stopped, peripherals kept running
  - Low-power-sleep mode: ARM® Cortex®-M0+core stopped, limited clock frequency, limited number of peripherals running, regulator in low-power mode, flash stopped
  - Stop mode: all clocks are stopped, regulator running, regulator in low-power mode
  - Standby mode: V_{CORE} domain powered off
- Step 2: selection of the wake-up mode (EXTI or RTC).
  Through the menu displayed by the application, the user selects the wake-up mode. He has two choices:
  - via EXTI: in this case, the user has to press on the push-button to generate an interrupt which wakes up the device.
  - via an RTC alarm: in this case, the user has to set a timeout value using the user interface (selection via the joystick). By default the timeout value is set to five seconds. At expiration of the timeout, the RTC triggers an interrupt that wakes up the system.
- Step 3: configuration of the system.
  At this stage, the application performs the following tasks:
  - Sending of a current measurement request. A request for current measurement is sent to the mfx companion chip via the I^2C interface. The effective current measurement starts after a delay defined at the configuration of the mfx.
  - Saving of the GPIO context. The GPIO configuration is stored internally. The aim is to be able to keep the same GPIO configuration before entering low-power mode and after exiting low-power mode.
  - Configuration of the GPIO (associated to the push-button) and configuration of the associated EXTI line, if the wake-up mode requested is EXTI.
  - Configuration of the RTC and enabling of the RTC alarm interrupt, if the wake-up mode requested is RTC.
  - Setting of a new clock configuration before entering low-power mode when it is necessary to optimize the power consumption. (For instance, CONFIG_CLOCK_MSI_64K when entering low-power-sleep mode)
  - Stopping of the systick interrupt if it is necessary to avoid being waked up involuntarily.
- Step 4: enter low-power mode.
  At this stage, the system enters in low-power mode. Depending on the wake-up mode selected, the system exits from this low-power mode after the RTC timeout expiration or after a push-button action from the user.
- Step 5: restore the previous configuration.
  Once the system is waked up, the GPIOs are set back to the state they were just before entering low-power mode. The clock is also set back to its default configuration:

CONFIG_CLOCK_HSE_32M.
When exiting from low-power mode, the application returns to the state it was before entering in low-power mode. This is valid for all power modes except in standby mode. When exiting from standby mode the system performs a reset.

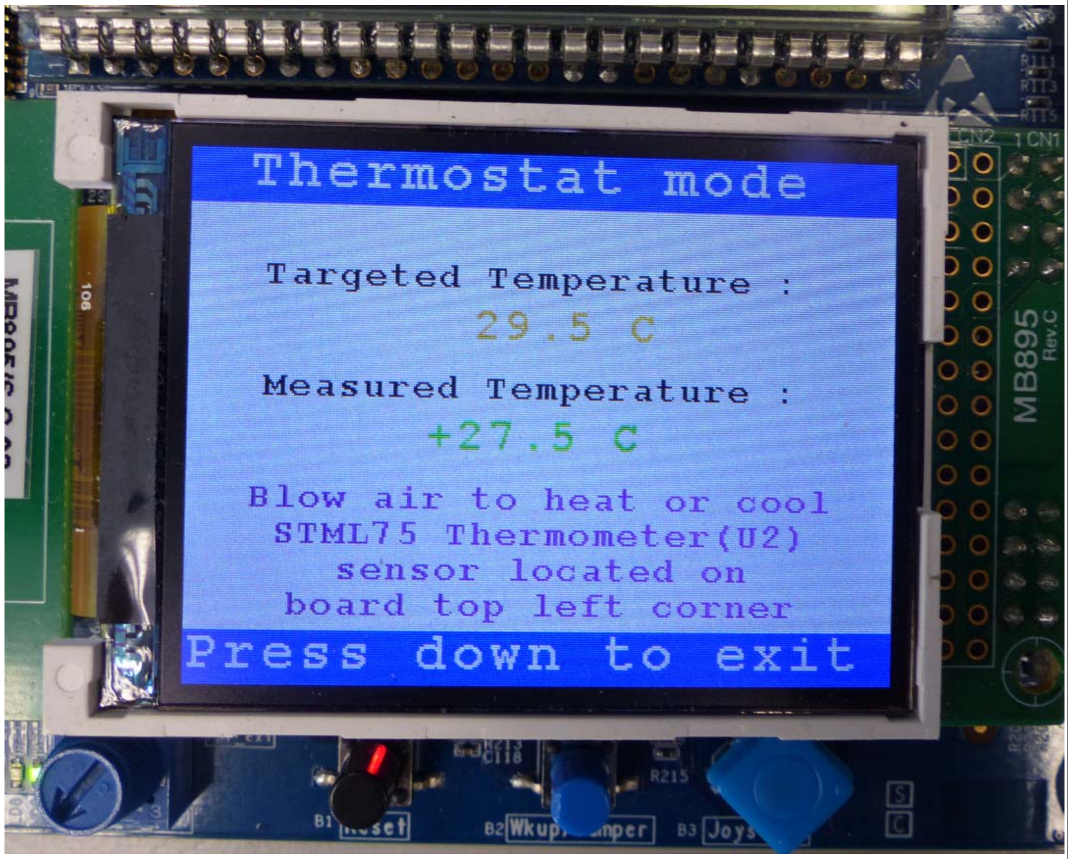
- Step 6: display the current.
  The value of the current consumption measured by the mfx companion chip is displayed on the STM32L073Z-EVAL evaluation board.

4.5 Thermometer application

4.5.1 User interaction

This application displays the temperature (sensor stlm75) in radiator thermostat way. Thanks to SMBUS, a management of the temperature setpoint tracking is done with interruption events, which displays messages and Min/Max temperature values.

Figure 13. Thermometer application



Thermostat mode

Targeted Temperature :
29.5 C

Measured Temperature :
+27.5 C

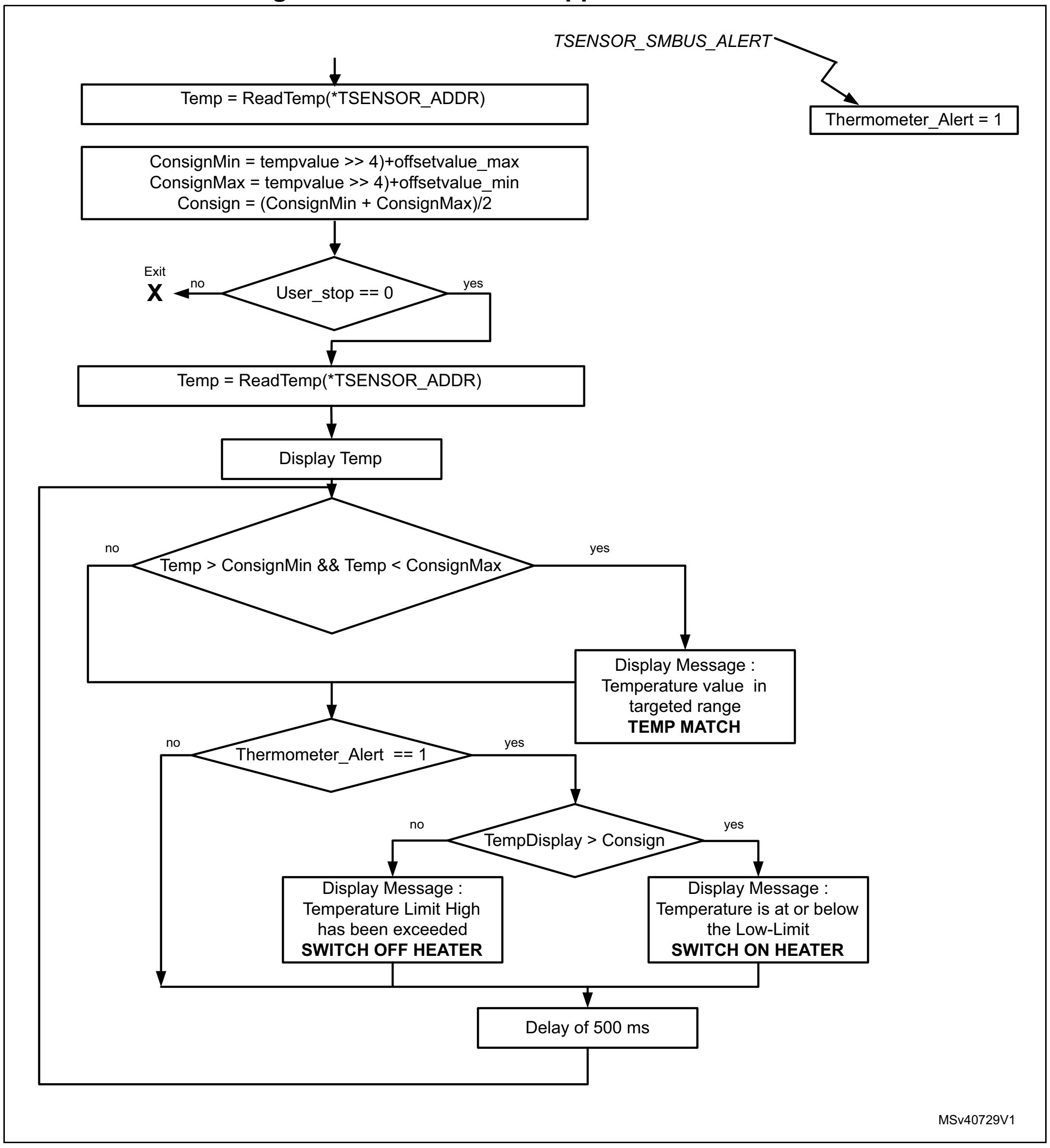
Blow air to heat or cool
STML75 Thermometer(U2)
sensor located on
board top left corner

Press down to exit

4.5.2 Functional description

This application is based on a dedicated driver (stm32l0xx_tsensor.c) which manages the SMBUS temperature sensor (stlm75) mounted on the STM32L073Z-EVAL board. This driver implements a high-level communication layer for initializing, reading or getting the status of the temperature sensor. This driver implements also the SMBUS ALERT protocol. Three signals are used to control the stlm75 device: SDA, SCL and OS/INT. Every 500 ms, the temperature is read and if the value is in between two values (min and max) a specific message is displayed. Moreover, a SMBUS alert is triggered when a specific threshold is reached.

Figure 14. Thermometer application behavior


TSENSOR_SMBUS_ALERT
Thermometer_Alert = 1
Temp = ReadTemp(*TSENSOR_ADDR)
ConsignMin = tempvalue >> 4)+offsetvalue_max
ConsignMax = tempvalue >> 4)+offsetvalue_min
Consign = (ConsignMin + ConsignMax)/2
Exit
X
no
User_stop == 0
yes
Temp = ReadTemp(*TSENSOR_ADDR)
Display Temp
no
Temp > ConsignMin && Temp < ConsignMax
yes
Display Message : Temperature value in targeted range TEMP MATCH
no
Thermometer_Alert == 1
yes
no
TempDisplay > Consign
yes
Display Message : Temperature Limit High has been exceeded SWITCH OFF HEATER
Display Message : Temperature is at or below the Low-Limit SWITCH ON HEATER
Delay of 500 ms
MSv40729V1

4.6 LC sensor metering application

4.6.1 Overview

This application shows how the LC sensor is managed. Two modes are available:

- Standard mode shows graphically on the display the LC sensor sensitivity to metal
- Low-power mode minimizes the power consumption (but in this case the status is not displayed as done in standard mode)

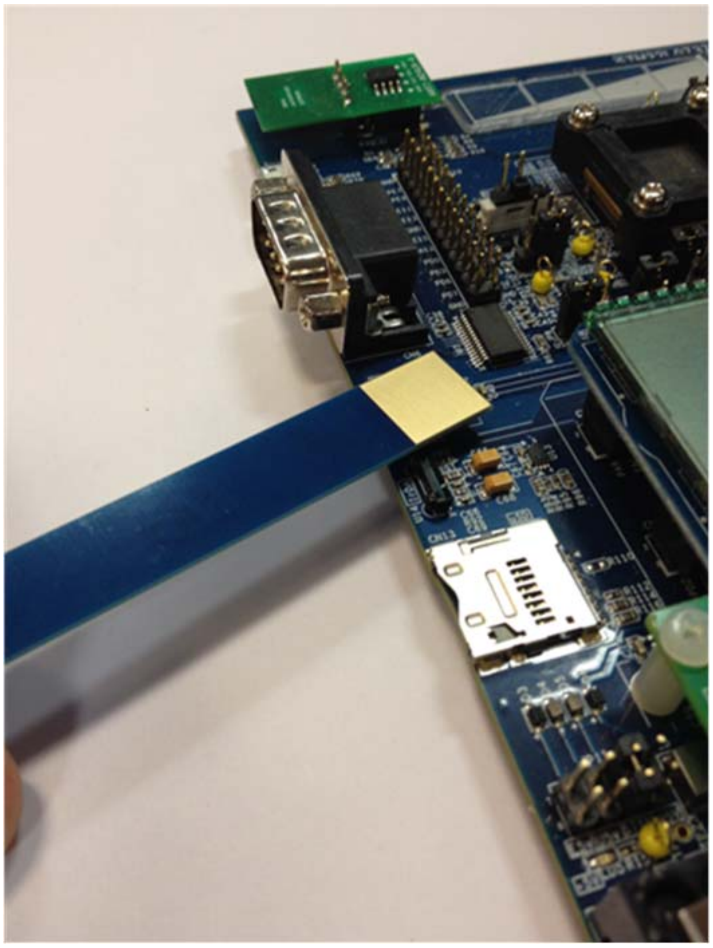
In both cases a metal strip must be placed over the LC sensor.

Standard and low-power modes are similar except that standard mode does not optimize the power consumption and that a graphical demonstration is available on the LCD TFT screen.

When using the detection accessory board MB1199, the metal strip must be positioned over the LC sensor as shown in *Figure 15: Using detection accessory board MB1199 with LC sensor metering*.

Note: More details on this application can be found in the “Demonstration on LC sensor for gas or water metering usage based on a STM32L073Z-EVAL evaluation board” Application note (AN4636).

Figure 15. Using detection accessory board MB1199 with LC sensor metering



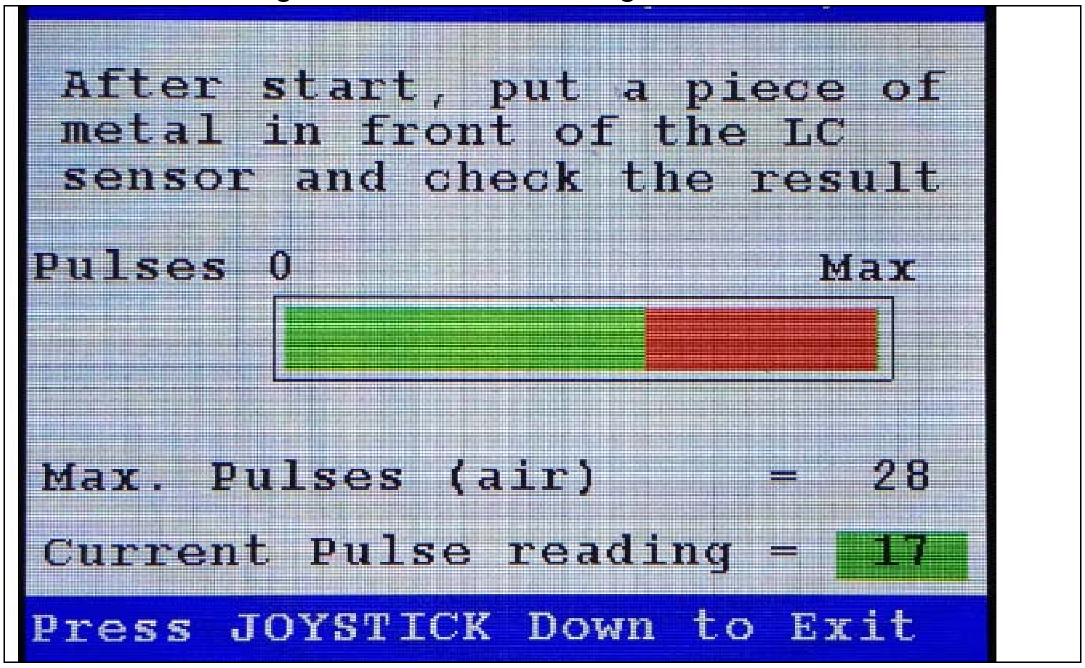
4.6.2 LC sensor metering in standard mode

The *Figure 16: LC sensor metering standard mode* shows the menu displayed when entering the mode standard. A bar graph shows the number of collected pulses (green area). The maximum graduation (“Max. Pulses” label) is reached when the LC sensor is in air (no metal). The missing pulses, due to the proximity of the metal, are displayed in the red area. The current pulse reading gives the number of pulses, especially when a piece of metal is placed above the LC sensor. If the amplitude of the damped oscillations reaches 20% less than the one of the undamped oscillations in air, then the LED LD1 is lit as detection flag (green LED near B2 push-button) and the current pulse reading value is turned to green.

If the detection accessory board is placed over the LC sensor (copper layer on top), then the variation in amplitude between damped and undamped oscillations is approximately 40%. All relevant signals are monitored to evaluate the detection, including:

- $V_{DD}/2$ reference bias controlled by PD7
- PB4 signals where oscillations are present (can be monitored on JP8 jumper)
- DAC feeding the COMP2 negative input as a comparator threshold voltage
- TP18 COMP2 comparator output wired to the LPTIM input

Figure 16. LC sensor metering standard mode


After start, put a piece of metal in front of the LC sensor and check the result

Pulses 0 Max

Max. Pulses (air) = 28

Current Pulse reading = 17

Press JOYSTICK Down to Exit

4.6.3 LC sensor metering in low-power mode

In this demonstration the LCD TFT screen does not display any information from the LC sensor metering system but the goal is to evaluate essentially the power consumption and the different signals, while the LC sensor feature is active to mimic a real application case.

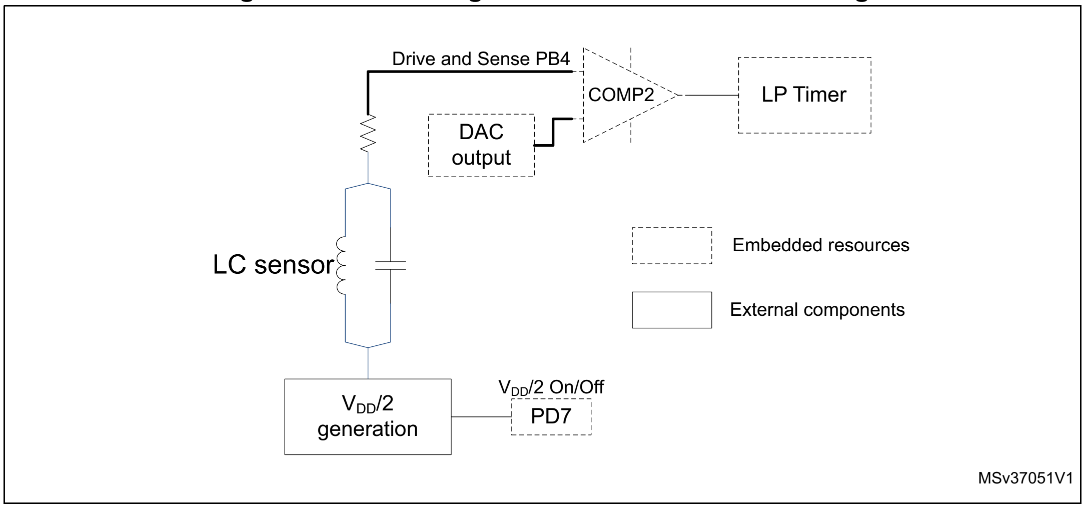
To measure the power consumption on the V_{DD} signal of the STM32, the JP10 jumper must be removed and an ammeter must be placed between pin 1 (+) and pin 2 (-) of the JP10 connector.

This ammeter measures the current delivered by the V_{DD} power supply to the STM32 exclusively. Knowing that the application changes from STM32 running steps to STM32 Stop modes, the power consumption observed on the microcontroller is not constant. While the microcontroller is executing some processes in RUN mode, the power consumption can be up to few mA. On the other hand, while the microcontroller is entering Stop mode (with RTC on LSE enabled), the power consumption reaches approximately 1.2 µA max. (25°C).

4.6.4 Block diagram of the LC sensor metering

Figure 17: Block diagram of the LC sensor metering shows the description of the LC sensor metering block diagram.

Figure 17. Block diagram of the LC sensor metering



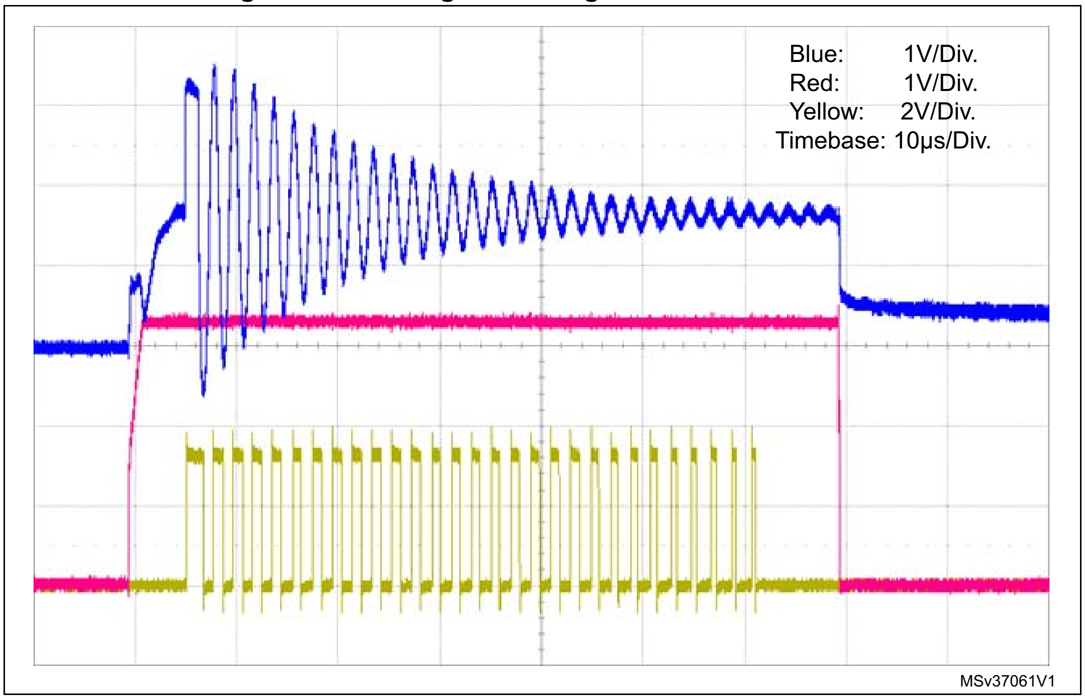
The LC sensor metering feature uses a LC network connected to a V_{DD}/2 bias voltage (AC ground). This V_{DD}/2 voltage is generated by an external operational amplifier, wired as a voltage follower and externally controlled by the pin PD7 of the microcontroller. This allows switching ON and OFF the V_{DD}/2 generation circuit, to save power consumption, when the LC sensor is not activated.

On the other side, the LC sensor is connected to PB4 pin through a serial resistor that limits the DC current necessary to start oscillations. This current must be below the GPIO maximum current capability (20 mA). PB4 is used both as a GPIO drive pin and as a sense pin connected to the comparator. When used as a drive pin, the PB4 GPIO is set as an output pin and delivers a positive excitation pulse to the LC sensor, causing oscillation; on the other end when PB4 is set as the comparator non-inverting input, the objective is to get the signals of the oscillations as soon as it starts. The COMP2 comparator is internally connected to the DAC, as an input threshold signal on its inverting input. The threshold signal is set approximately above 100 mV the VDD/2 bias voltage to provide a noise margin. Finally, the pulse train on the comparator output is redirected to the low-power timer input to be counted.

4.6.5 Application principle

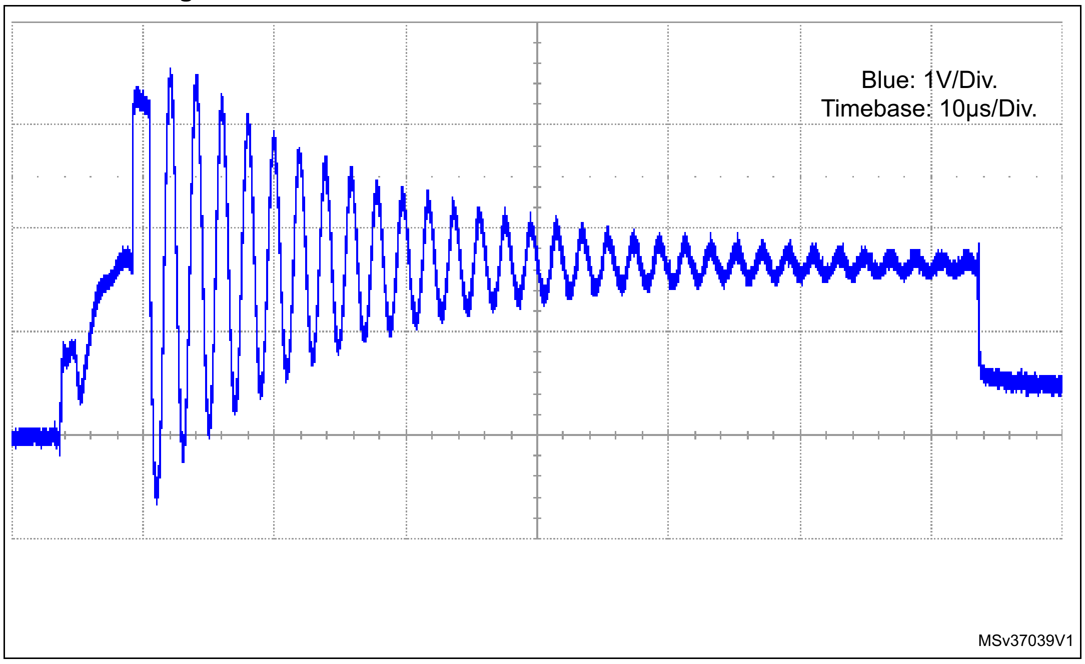
As described in *Section 4.6.4* the LC sensor is driven by an excitation pulse managed by the PB4 GPIO. The pulse needs approximately a minimum width of 2 µs, so that the oscillations amplitude can reach and exceed the power signals rails (V_{DD} and GND). Once this pulse is delivered, the GPIO is reconfigured from push-pull output to alternate the function analog input. The oscillations then appear on the comparator non-inverting input, which is high-impedance. *Figure 18: Main signals during LC sensor activation* shows the main signals, including the PB4 waveform (blue signal), the V_{DD}/2 controlled by the PD7 (red signal) and the COMP2 comparator output (yellow signal). The V_{DD}/2 is first turned on (red signal), then the oscillations are generated by the PB4 (blue signal) and collected by the LPTIM after being conditioned by the comparator (yellow signal).

Figure 18. Main signals during LC sensor activation



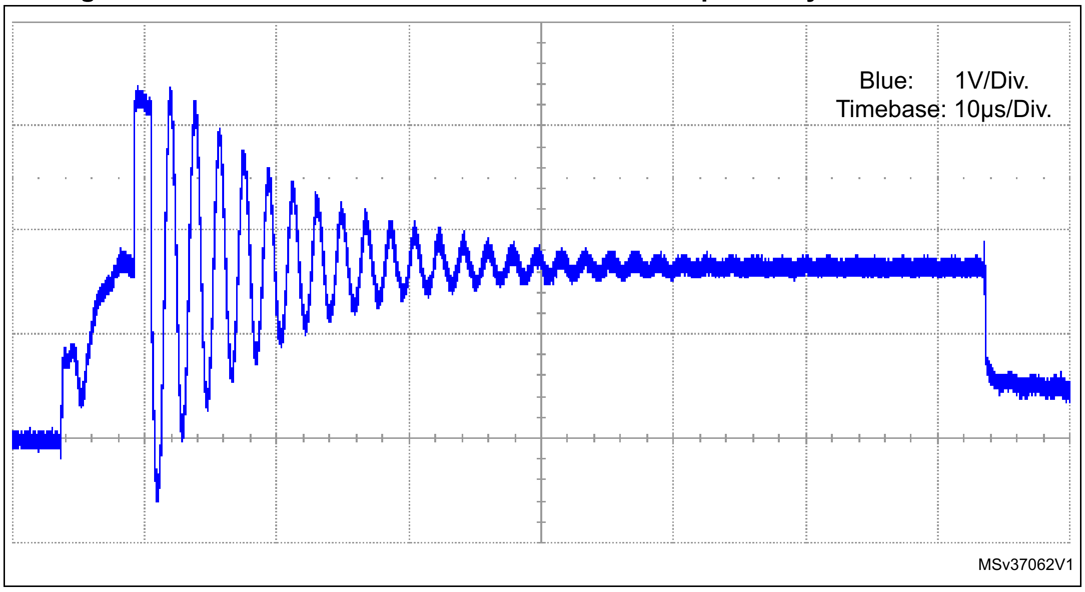
Once started, the amplitude of these oscillations decays exponentially until the signal comes back to $V_{DD}/2$ bias as it is shown in *Figure 19*. This behavior is obtained when the LC sensor is in air and when no metal is already present.

Figure 19. Oscillations collected with LC sensor in air on PB4



When a metal is in the proximity of the LC sensor, a part of the magnetic field emitted by the inductor is absorbed as eddy currents and the energy of the inductor is lost in the metal target. In that case the oscillations observed on the PB4 (measured on JP8) decay more quickly and the number of collected pulses is then lower than in the previous case (see *Figure 20*). The detection is based on the comparison between the detected pulses, obtained in air, and the ones counted in presence of metal. During detection, the amplitude and frequency of these oscillations change. The oscillations are damped, due to the presence of metal. The amplitude decreases with the proximity of metal and the frequency increases (approx. +15%).

Figure 20. Oscillations collected with LC sensor in proximity of metal on PB4



4.6.6 Capture window of the LC sensor metering

To save power consumption, the LC sensor metering is activated periodically and the microcontroller is placed in Stop mode (low-power mode demonstration selected).

The different peripherals used in this application example are then switched on during a capture window and deactivated after, to minimize the power consumption once the measurements have been performed. An arbitrary 32 Hz capture frequency has been selected to activate the LC sensor function. As the LC sensor metering could be used in a gas or water meter detection system, it is considered that this frequency is high enough to evaluate the rotation of a wheel, where only few turns per second can be executed. The RTC wakes up 32 times per second for the LC sensor measurement, with the CPU and selected digital and analog peripherals in RUN mode. The rest of the time is spent in Stop mode.

The capture window lasts approximately 80 µs. During this period, the different peripherals are first switched on and the LC sensor excitation pulse delivered, until the complete pulses collection is acquired. Then the microcontroller is set for the lowest power consumption state.

The sequence is divided into four main steps:

- Enabling of peripherals such as DAC used as COMP2 comparator threshold voltage, $V_{DD}/2$ reference bias ON, COMP2 activation and LPTIM enable
- Generation of an excitation pulse on the LC sensor
- Collection of the LC oscillations and pulses count
- MCU configuration to enter Stop mode and reach the lowest power consumption

Peripherals are initialized at the software start-up, then simply switched on or clocked when the capture window occurs. Due to some peripherals different stabilization times, a switching order is defined, to comply with the corresponding waiting times. For instance, the COMP2 is initialized before the DAC, due to a longer initialization time. Once the capture has been performed, the peripherals are switched off to save power consumption. The “sleep on exit” feature is used to turn the device into Stop mode as soon as the interrupt has been serviced and completed.

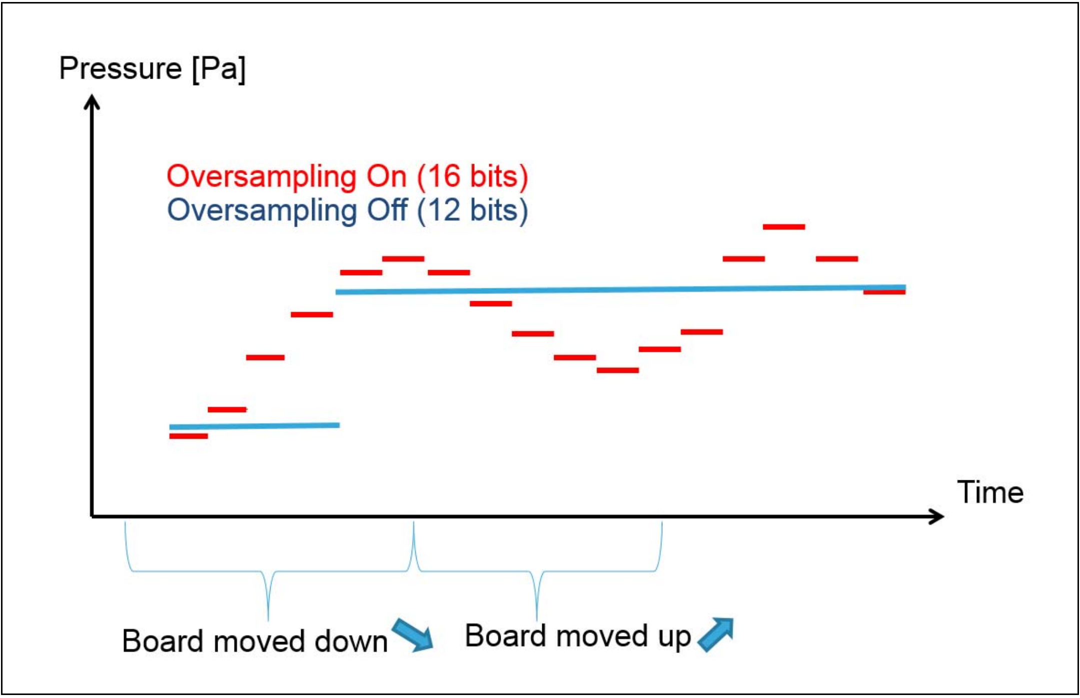
4.7 Pressure application

4.7.1 User interaction

This application demonstrates the capability of the 16-bit ADC oversampling mode from the 12-bit ADC available in the STL32L073xx component. The current air pressure value is measured from the sensor (U1) available on the board (see *Figure 21*).

To modify the pressure, the user can move up the board to 1.5 meters or more and then move it down to the floor.

Figure 21. Display of the pressure application



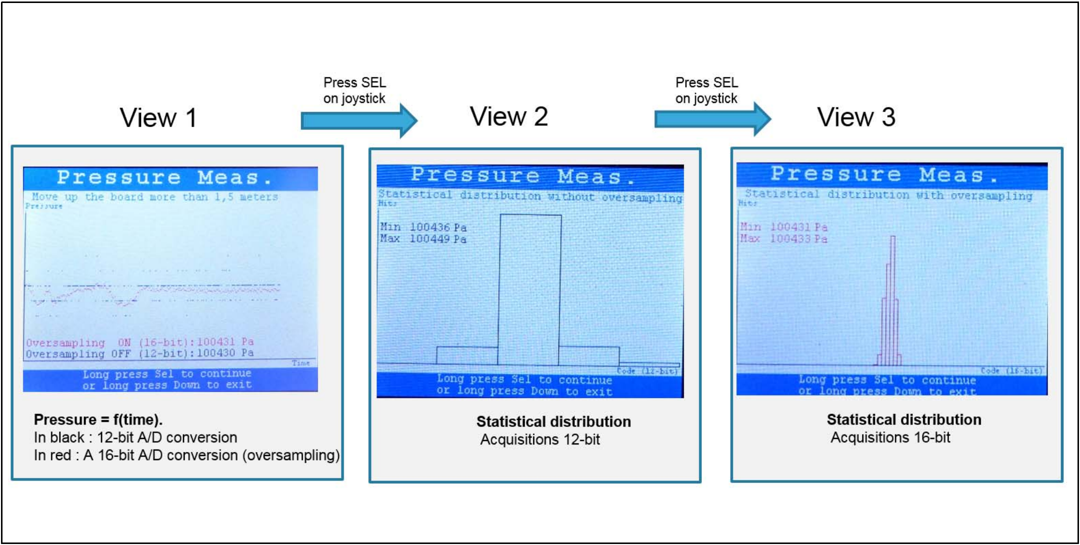
The pressure application supports three different views as shown on *Figure 22: The three different views associated to the pressure application*. It is possible to switch from one view to the next one using the joystick SEL button:

- The first view displays the curve of pressure: Pressure = f(time). Each 200ms, a 12-bit A/D conversion is done and a black point is added. A 16-bit A/D conversion (oversampling) is also done and a red point is added.
- The second view shows the statistical distribution of 2000 12-bit acquisitions.The average value and the standard deviation are calculated for each series of measure:

Min = average – standard deviation
Max = average + standard deviation

- The third view shows the statistical distribution of 2000 16-bit acquisitions (oversampling).The average value and the standard deviation are calculated for each series of measure:
  Min = average – standard deviation
  Max = average + standard deviation

Figure 22. The three different views associated to the pressure application



Pressure Meas.
Move up the board more than 1,5 meters
Oversampling ON (16-bit):100431 Pa
Oversampling OFF (12-bit):100430 Pa
Long press Sel to continue
or long press Down to exit

Pressure = f(time).
In black : 12-bit A/D conversion
In red : A 16-bit A/D conversion (oversampling)

Pressure Meas.
Statistical distribution without oversampling
Min 100436 Pa
Max 100449 Pa
Long press Sel to continue
or long press Down to exit

Statistical distribution
Acquisitions 12-bit

Pressure Meas.
Statistical distribution with oversampling
Min 100431 Pa
Max 100433 Pa
Long press Sel to continue
or long press Down to exit

Statistical distribution
Acquisitions 16-bit

4.7.2 Functional description

The ADC (analog-to-digital converter) present on the STM32L073 uses a 12-bit successive approximation. A built-in hardware oversampler allows to improve analog performances while off-loading the related computational burden from the CPU.

When launching the pressure application, the demonstration shows in real time the pressure variation. Two measures are done every 200ms. The first one with the ADC oversampling disabled (12-bit conversion). The second one with the ADC oversampling enabled (16-bit conversion).

When running the pressure application, the following sequence is performed repetitively until the user presses on the joystick the DOWN button to exit:

- Step 1: the ADC is initialized via the function HAL_ADC_Init() with the oversampling mode disabled
- Step2: the ADC is started via the function HAL_ADC_Start()
- Step 3: the ADC conversion is performed using the function HAL_ADC_PollForConversion() with a timeout value set to 10ms
- Step 4; the ADC is stopped via the function HAL_ADC_Stop()
- Step 5: the ADC converted value is retrieved via the function HAL_ADC_GetValue()
- Step 6: the ADC converted values is displayed
- Step 7: Step 1 up to Step 6 are repeated but with the oversampling mode enabled
- Step 8: a tempo of 200 ms is set before restarting from Step 1

5 Revision history

Table 2. Document revision history

Date	Revision	Changes
11-Oct-2016	1	Initial release.