AN5834

Application note

LC sensor metering implementation on STM32U5 Series featuring LPBAM

Introduction

This application note describes the LC sensor metering feature included in the STM32U5 Nucleo board NUCLEO-U575ZI-Q.

The STM32U5 Series microcontrollers combine ultra-low-power with performance, offering a complete set of analog and digital peripherals. These peripherals can operate in autonomous mode down to Stop 2 mode, a subset of which is used to build the LC sensor-based meter. This is the starting point for the implementation of gas or water meter applications, based on the inductive reading of a rotating mechanical wheel.

The demonstration based on the STM32U5 can be executed with a single USB cable (type A to B) connecting the host PC to the NUCLEO-U575ZI-Q, and a custom LC sensor board.

This application note is delivered with the X-CUBE-LCSENSOR expansion package dedicated to STM32U5xx microcontrollers using the STM32CubeMX (6.6.1) tool with the LPBAM interface.

1 General information

The STM32U5 Series microcontrollers are based on the Arm® Cortex®-M33 processor.

Note: *Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.*

Table 1. **Acronyms and terms**

Acronym	Definition	Description
Vmid	Middle voltage ($V_{DD}/2$)	It allows energizing LC sensor. It is delivered by DAC peripheral output for STM32U5. To save power consumption, Vmid is disabled when the LC sensor is not activated.
Vcmp	Comparator threshold voltage	It is set above Vmid to provide a noise margin.
Texcit	Excitation pulse width	It allows driving LC sensor in output push-pull.
Tcapture	Time to perform the measurement	During this time, the comparator collects oscillations, compares them to Vcmp and delivers pulses to the low power timer.
Count	Pulse count number	It is generated by the LC sensor and conditioned by the comparator.
CountDetect	Detection threshold for the count number	It allows defining the sensor state (metal or no metal).
CountMax	Maximum number of counted pulses, without metal, in proximity of the LC sensor	-
CountMin	Minimum number of counted pulses, with metal, in proximity of the LC sensor	-

Reference documents

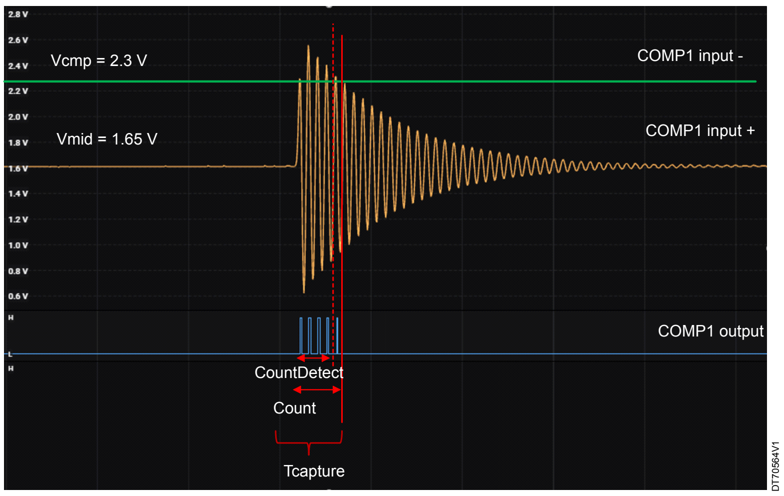
[1] Application note *Demonstration of LC sensor for gas or water metering based on STM32L073Z-EVAL and STM32L476RG-NUCLEO boards* (AN4636)

[2] Application note *How to build STM32 LPBAM application using STM32CubeMX* (AN5816)

2 LC sensor metering principle

An excitation pulse managed by LPGPIO drives LC (see Figure 1). Once the pulse is delivered, the LPGPIO is isolated. The LC oscillation has been generated. The COMP peripheral collects the analog oscillations and compares them to a reference voltage (Vcmp), delivering a digital signal on the comparator output. The low power timer peripheral then counts these pulses.

Figure 1. LC sensor oscillations



When a metal is in the proximity of the LC sensor, a part of the magnetic field emitted by the inductor is absorbed, and the energy is lost in the metal target. The oscillation amplitude and the number of collected pulses are consequently lower compared to the case without metal.

3 STM32U5xx-Nucleo description

3.1 Hardware required

This demonstration uses the NUCLEO-U575ZI-Q board and custom board for LC sensing with external components. The NUCLEO-U575ZI-Q board can be connected to the USB cable of the host PC.

The detection accessory board MB1199 can be used to simulate the presence of metal over the LC sensor. It must be oriented with the copper layer on the top side and placed over the inductor during the detection tests.

3.2 Hardware settings for LC sensor custom board

3.2.1 Requirements

A custom board with LC sensors must be developed and connected to the STM32U5 I/Os. External components must be selected to minimize deviations due to external conditions (such as temperature, humidity). The characteristics of the components used for this demonstration are detailed below.

- LC sensors: all sensors must have identical Ls/Cs values. These values have been selected according to the comparator input bandwidth and the current consumption needed to excite the LC sensor.
  - Ls = 140 µH used in this application (recommended value is 470 µH)
  - Cs = 220 pF
- Reference capacitor: used to store energy for LC sensors oscillations.
  - Cref = 470 nF
- Serial resistors: the LC sensor is connected to comparator inputs pin through a serial resistor that limits the DC current necessary to start oscillations. This current must be below the GPIO maximum current capability.
  - R = 150 Ω (for V_{DD} = 3 V)
- Protection diodes (1N4148): external diodes must be connected between LPGPIO excitation pin and comparator inputs pin.

3.2.2 LC sensor schematic on STM32U5

Figure 2. One or two sensors schematic

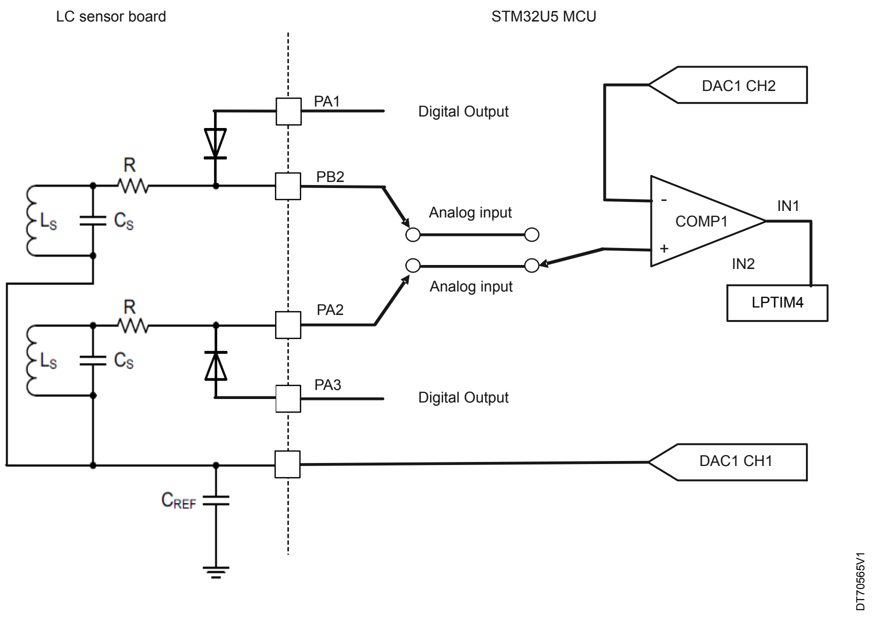
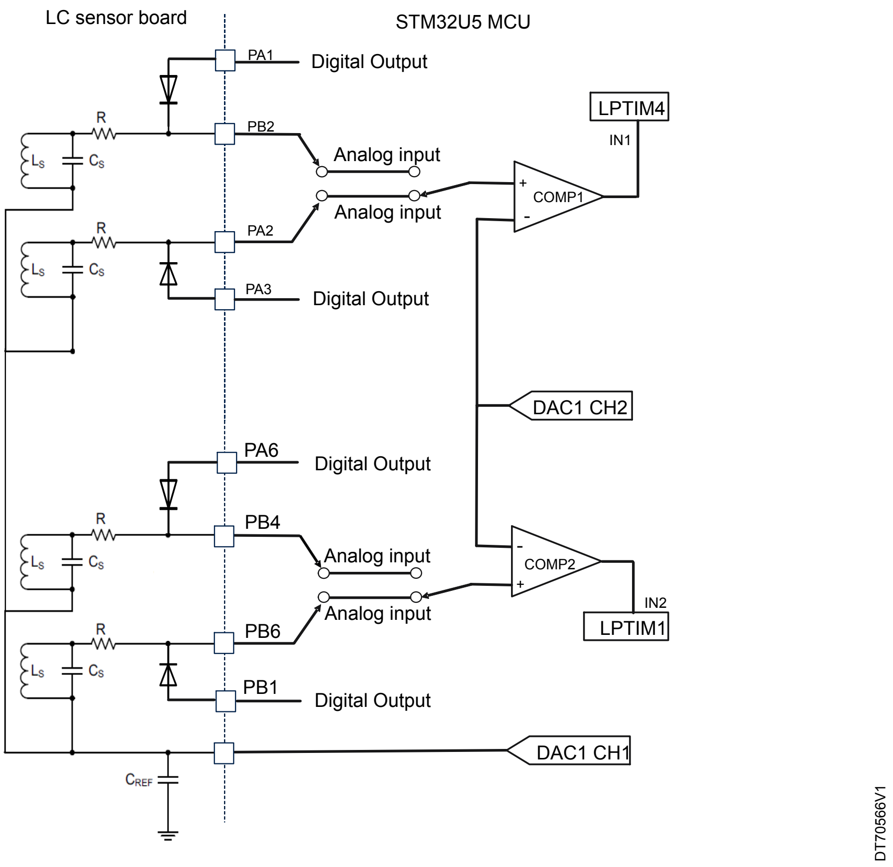


Figure 3. **Up to four sensors schematic**



For the LC sensor custom board, the main difference between STM32L4/L0 and STM32U5 is that the excitation comes from another pin connected to the noninverting comparator input.

The 1N4148 diode is used to isolate between the excitation pin and the noninverting comparator input.

3.2.3 LC sensor signals connection on STM32U5

Table 2. **Comparator threshold connection**

Mode	GPIO_MODE_ANALOG		Signal
Internal connection	DAC_CH2_PA5	COMP1_INPUT_MINUS_DAC1_CH2	Vcmp
		COMP2_INPUT_MINUS_DAC1_CH2	

Table 3. **Internal signals in output mode**

Mode	GPIO_MODE_OUTPUT		Signal
Internal connection	COMP1_OUT_PB0	LPTIM4_INPUT1SOURCE	Count
	LPTIM3_CH1	LPDMA1_CH0_TRIGGER	Trigger
External connection	COMP2_OUT_PB11	LPTIM1_INPUT1SOURCE_PB5	Count

Table 4. **LC sensor excitation pulses and energization connections**

Mode	GPIO_MODE_ANALOG	GPIO_MODE_OUTPUT_PP	Signal
External connection	COMP1_INP2_PB2	LPGPIO.P0 PA1	Texcit(sensor 1)
	COMP1_INP3_PA2	LPGPIO.P1 PA3	Texcit(sensor 2)
	COMP2_INP1_PB4	LPGPIO.P2 PA6	Texcit(sensor 3)
	COMP2_INP2_PB6	LPGPIO.P3 PB1	Texcit(sensor 4)
	DAC1_CH1_PA4	(Connect to LC sensor I/O pin)	Vmid

3.3 Application principles

3.3.1 Overview

An excitation pulse managed by LPGPIO drives LC sensors. Once the pulse is delivered, the LC oscillation is generated.

The COMP peripheral collects the analog oscillations and compares them to a reference voltage (Vcmp).

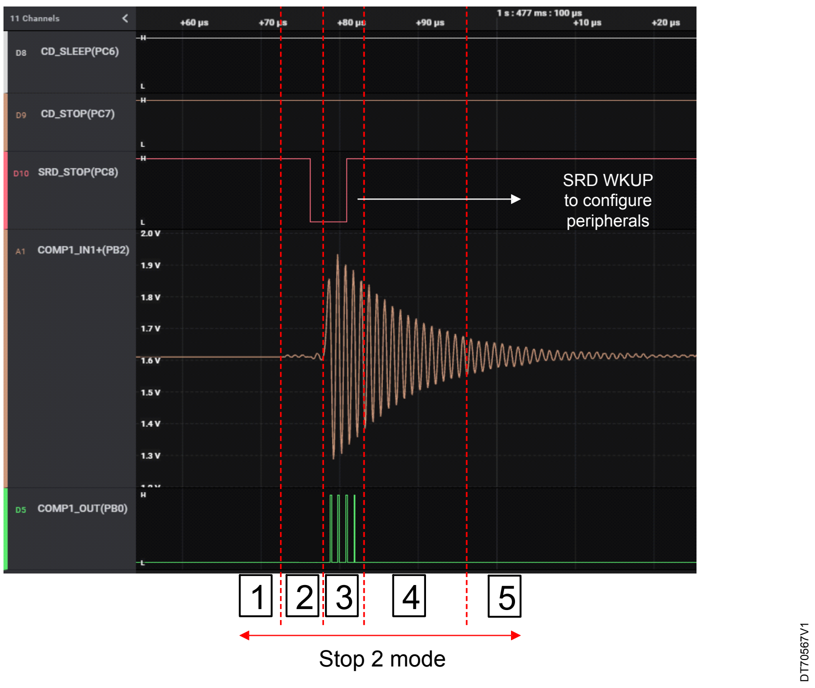
The internal DAC generates the Vmid level, which can be stopped after measurement to reduce the power consumption.

Once started, the amplitude of these oscillations decays exponentially until the signal comes back to Vmid bias.

3.3.2 LC sensor metering capture window

To save power consumption, the LC sensor metering is activated periodically, and the microcontroller is put in Stop 2 mode (low-power mode demonstration selected).

Figure 4. **LC measurement sequence**



For each measurement, as indicated in Figure 4, the following steps are executed:

1. The LC sensor metering is activated, and the microcontroller is put in Stop 2 mode (low-power mode).
2. The MCU still in Stop 2 mode and enables the peripherals used by the application, with a waiting time inserted using the LPTIM trigger to let peripherals stabilize. After that, the LC sensor excitation pulse is delivered (refer to Excitation time: Texcit). Then start the measurement (refer to Capture window: Tcapture).

3. During the measurement phase, the MCU is in Stop 2 mode and the LPTIMER increments its counter according to the number of oscillations. The collection of the LC oscillations and pulses count is done.
4. After the pulse counts are collected, all peripherals used by the application are disabled using LPDMA execution nodes in Stop 2 mode.
5. The LC sensor metering is disabled, and the microcontroller is still the Stop 2 mode.

3.3.3 Parameters to define

Some parameters must be defined according to the application requirements:

DAC configuration

Two channels are used to generate Vmid and Vcmp levels.

- Vmid for reference level to V_{DD}/2: channel 1 connected to external pins, PA4.
- Vcmp for threshold comparator level: channel 2 connected to on-chip comparator inverting input.

The DAC peripheral is configured in sample and hold mode and managed by LPDMA nodes. It is enabled at each LC measurement and disabled at the end of LC measurement.

Reference voltage (Vmid)

The first DAC output is configured to deliver reference voltage to energize LC sensors at V_{DD}/2.

$$DAC\,Vmid\left(toprogram\right) = 4096 \times \left(\frac{Vmid}{V_{DD}}\right) = \frac{4096}{2} = 2048 = 0x800$$

Comparator threshold voltage (Vcmp)

The second DAC output is configured to deliver the threshold voltage (Vcmp) needed by the comparator. It is set higher than V_{DD}/2 bias voltage to provide a noise margin.

In the following example, DAC output is configured to deliver Vcmp = Vmid +750 mV.

$$DAC\,Vcmp\left(toprogram\right) = 4096 \times \left(\frac{Vcmp}{V_{DD}}\right) = \frac{4096}{3\,V} \times 2.25V = 3072 = 0xC00$$

Note: The two DAC channels are configured during the peripheral initialization.

DAC Vmid refresh time (Trefresh)

To save power consumption, the DAC output is disabled at the end of LC measurements. During this phase, the DAC output level is in analog state (high impedance) and the Vmid level is hold by Cref capacitor. Vmid drop (Dv) depends on the holding time (time between two measurements). Before each measurement, a DAC refresh time after DAC enabling is required to reach the expected Vmid value.

For more details on Trefresh and Dv computing, refer to document [1].

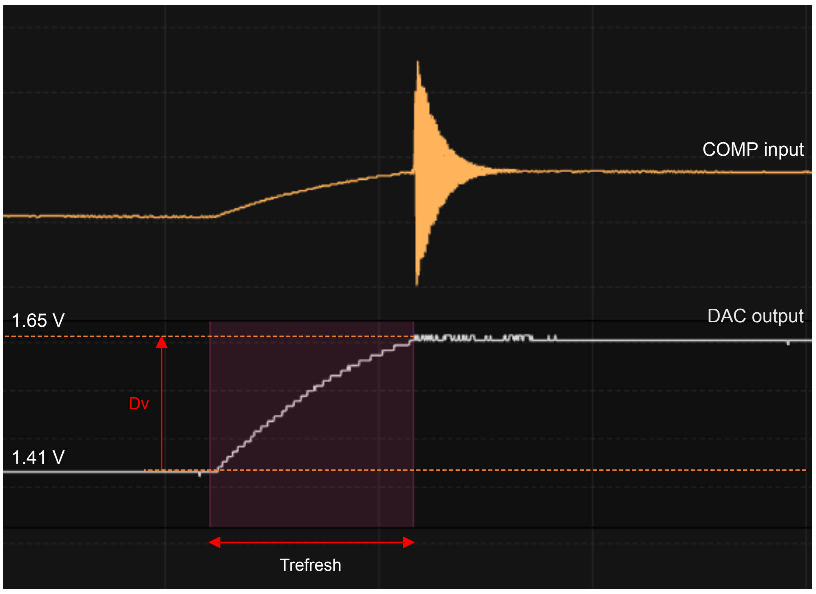


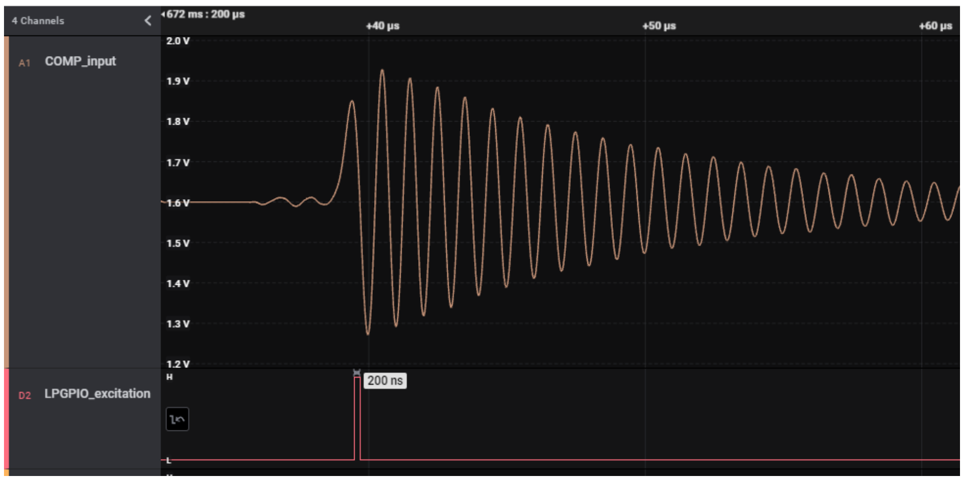
Figure 5. **DAC output**

Excitation time: Texcit

LC sensor excitation can be done by an external LPGPIO pin connected to the noninverting comparator input. The excitation is generated when writing 1 then 0 directly into the LPGPIO_ODR using LPDMA with MSI clock = 24 MHz.

Once the pulse is delivered, the LC oscillation is generated.

Figure 6. **Excitation pulse managed by LPGPIOs with Texcit = 200 ns**



Capture window: Tcapture

$$Tcapture\_seconds = \left(\frac{CountMax}{Freqosc}\right) \times \left(1 + x\right)$$

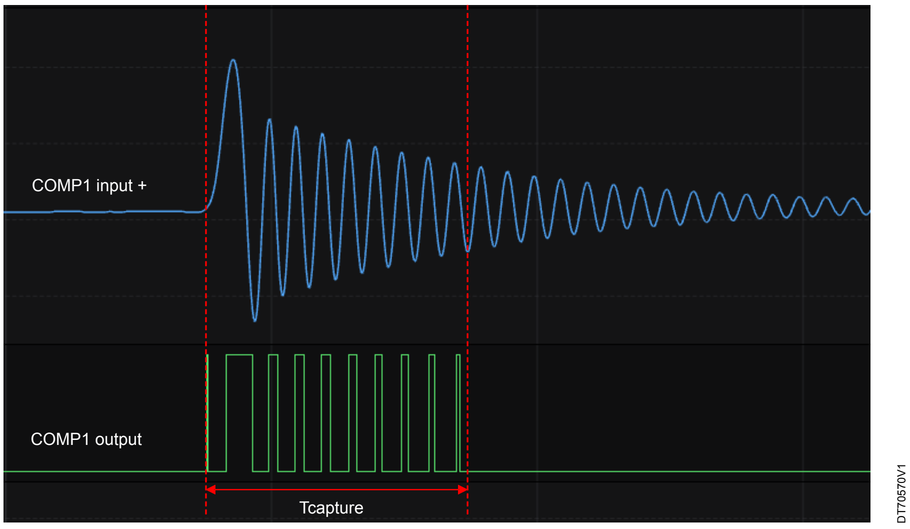
where x is the margin in percent (default value is 5 %) and CountMax is determined during the calibration phase.

The theoretical oscillation frequency is determined as:

$$Fresqosc = \frac{1}{2 \times \pi \times \sqrt{LS \times Cs}}$$

In this example, the inductor Ls is 140 μH, and the capacitor Cs is 220 pF. The oscillation frequency is then approximately 899 kHz.

Figure 7. **Tcapture**

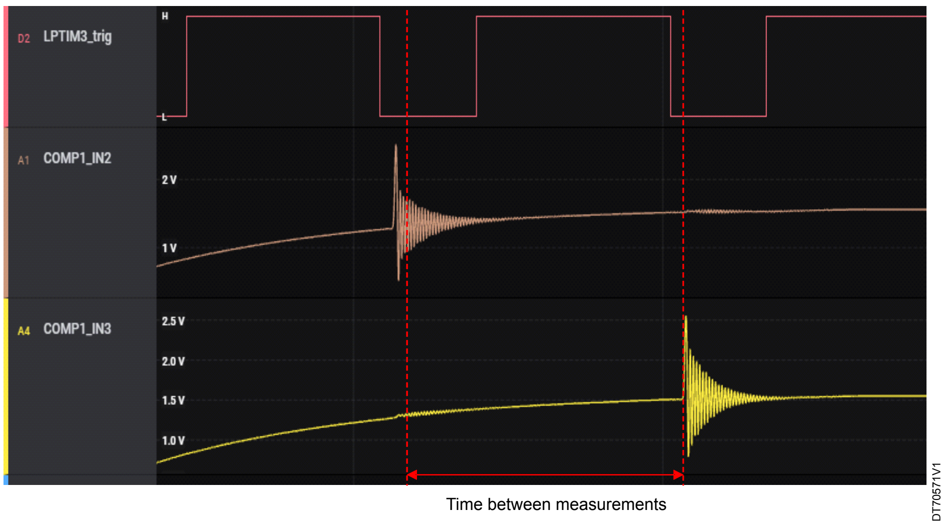


Time between measurements

Measurements must be done sequentially to avoid cross-effects between LC sensors. If more than one sensor is used, a waiting time between the measurements must be added to account for the stabilization time of the next sensor.

The time between sensor is managed with the low-power timer (LPTIM3) configured to trigger the comparator on the falling edge.

Figure 8. **Time between sensors: 100 μs between sensor 1 (PB2) and sensor 2 (PA2)**



3.3.4 Calibration

The calibration is done with and without metal in proximity of LC sensor.

The aim of the calibration is to adjust the comparator threshold level to have a predefined pulses count number. Based on this pulse count number, the detection threshold is set, and the capture time adjusted for power saving.

For more details about calibration, refer to document [1].

4 STM32CubeMX with LPBAM interface

4.1 STM32U5xx peripherals used by the application

This application example uses the following STM32U5 peripherals with the settings described below.

GPIOs

Some GPIOs are used to control the signals needed for this LC sensor metering example:

- PA4 as Vmid generator (DAC output) at $V_{DD}/2$ reference voltage (IO1)
- PB2, PA2, PB4, and PB6 are alternatively used as analog GPIO (positive comparator inputs COMP1 and COMP2).
- PA1, PA3, PA6, and PB1 are used as LPGPIO outputs to drive the LC sensors.
- PB5 configured as LPTIM1 alternate function mode and connected externally to PB11 COMP2 output.
- PC6, PC7, and PC8 are configured as debug pin in Stop 2 mode.

DAC

DAC_OUT1 and DAC_OUT2 are used to generate:

- $V_{DD}/2$ reference voltage (Vmid)
- input threshold voltage needed by the comparator

This reference voltage provides a voltage value of the preset threshold near $V_{DD}/2$ but with a necessary positive margin to eliminate noise. It calibrates the threshold value according to different LC network responses and environment variations. DAC outputs are configured in sample and hold mode and enabled/disabled via LPDMA nodes.

COMP1 and/or COMP2

Comparators are connected to LC sensors. 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. This results in digital pulses delivered on the comparator output. This signal is then transmitted to the LPTIM. Comparators are connected to the LC sensor network.

LPTIM1 and/or LPTIM4

LPTIM1 and/or LPTIM4 timers are configured to count the pulses generated by the LC sensor oscillations, and conditioned by the 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 to the previous value to get the number of collected pulses for the corresponding measurement.

LPTIM3

The LPTIM3 timer is configured to generate the precise time used for the LC sensor metering sequence. This LPTIM is configured in PWM mode. The repetition counter is configured depending on the number of sensors used in the application. It is configured also to repeat the LC sensor acquisition at regular intervals.

LPDMA

Three DMA channels are used to configure peripherals and store the number of collected pulses in SRAM4.

- LPDMA_Channel1: used to configure all the peripherals in Stop 2 mode without using the CPU with sequential nodes
- LPDMA_Channel2: used to store the number of collected pulses for sensor1 and sensor2 in SRAM4
- LPDMA_Channel3: used to store the number of collected pulses for sensor3 and sensor4 in SRAM4

RTC

The RTC peripheral is used to start the calibration at regular intervals before the MCU performs the LC measurements. The RTC clock is connected to the LSE clock which frequency is 32.768 kHz.

The wakeup timer is used to perform LC measurements. The wakeup frequency corresponds to the LC sensor capture frequency. It is set to 500 Hz in this example. This capture frequency can be increased or decreased depending on the rotating wheel maximum speed. Modifications on the capture frequency must lead to a change of the average power consumption.

TIM6

TIM6 is configured to generate precise temporizations used for the LC sensor metering sequence.

This timer is configured in OnePulseCounter mode to generate event at the end of pulse capture time.

Note: Once the calibration is done, the RTC and TIM6 must be disabled.

SRAM4

The memory used in this application is SRAM4.

SRAM4
RAM_start = 0x28000000
RAM_end = 0x28003FFF

4.2 LPBAM LC sensor application description

4.2.1 Principle

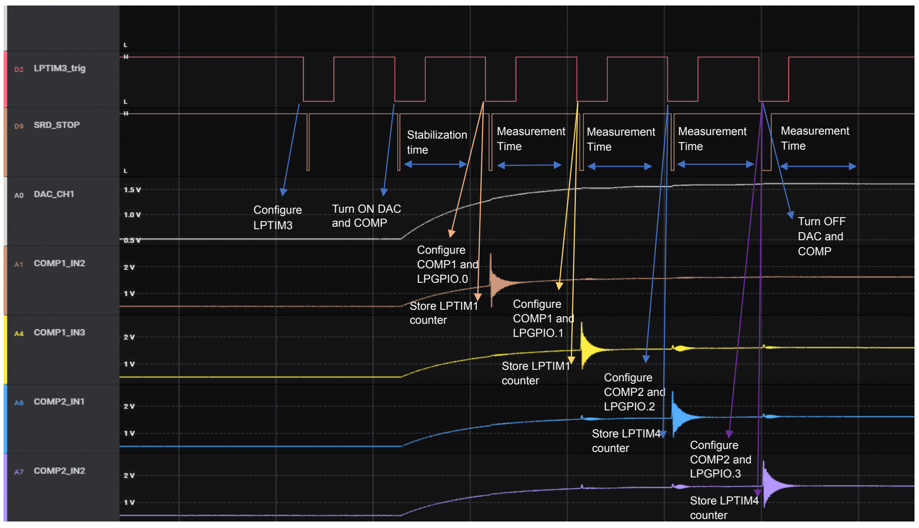
After reset, the main application configures the needed resources for the main application such as: ICACHE, GPIO, LPDMA, DAC, COMP, and LPTIM.

The LPBAM scenario description using several LPDMA execution nodes inserted in one queue is described below:

1. The LPTIM3 peripheral is configured to generate pulses at regular intervals.
2. When triggered on the falling edge of the LPTIM pulse, the COMP and DAC peripherals are switched ON. A delay is inserted (using the LPTIM pulse) to wait the peripherals stabilization times before they can be operational.
3. On the next falling edge of the LPTIM pulse, the COMP peripheral is configured using another LPDMA execution node. The I/O pin placed in LPGPIO output mode delivers the excitation pulse. Set 1 then 0 the LPGPIO pin using two consecutive nodes as soon as the pulse is applied on the LC network. The LC oscillation is generated.
4. At the end of measurements, DAC and COMP1/2 are disabled using the LPDMA node to minimize power consumption.

Figure 9 illustrates the principle of the LC sensor application for four sensors.

Figure 9. **LC sensor application signal debug for four sensors**

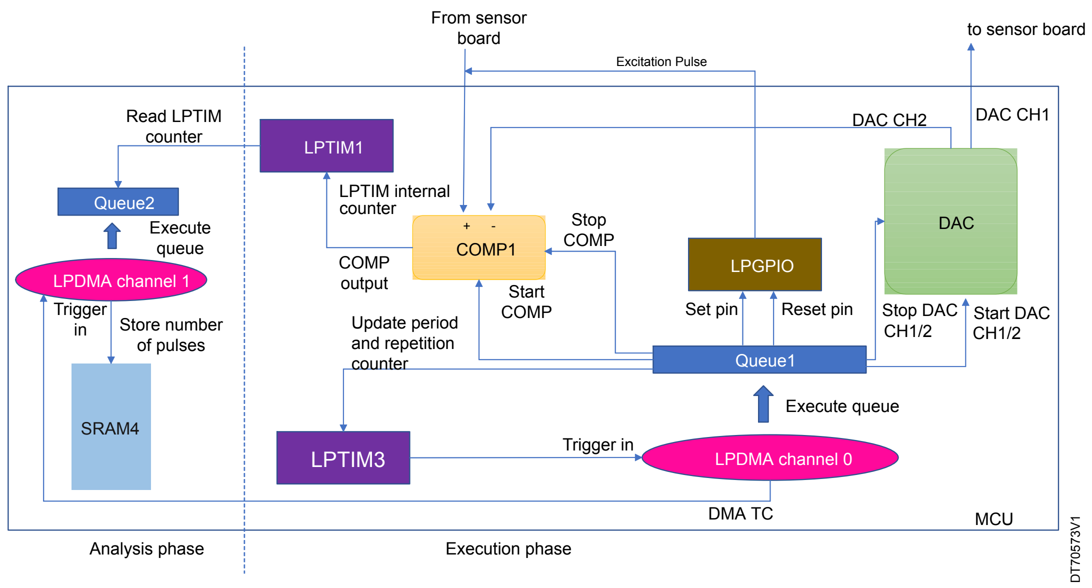


4.2.2 Implementation

To implement this scenario, the user must follow the steps below:

1. Configure LPTIM3 (using HAL) to generate the PWM signal trigger.
2. Configure DAC channel 1 to generate an external signal and channel 2 to generate an internal signal.
3. Configure COMP1 and COMP2 (using HAL).
4. Configure LPTIM1/4 counter (using HAL) to increment each valid clock pulse on the LPTIM external input 1.
5. Build an LPBAM queue 1 (with no DMA trigger start condition):
   a. updates the LPTIM3 period and repetition counter.
   b. enables DAC channels with LPTIM3 trigger condition.
   c. starts COMP (select input Minus) with LPTIM3 trigger condition.
   d. sets the LPGPIO pin.
   e. resets the LGPIO pin.
   f. reads the LPTIM1/4 counter (nodes are added manually).
   g. disables DAC channels and COMP.

Figure 10. LC Sensor application block diagram



4.3 STM32CubeMX LPBAM LC sensor application building

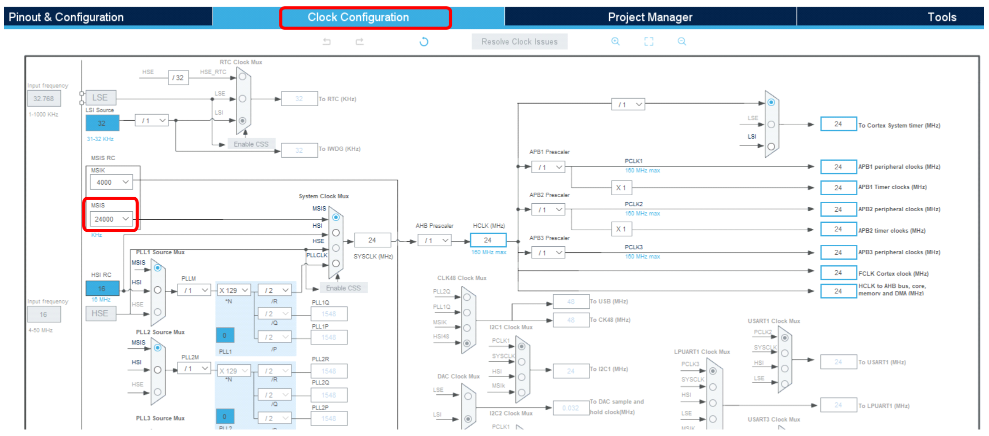
This section explains how to build the LC sensor application using the STM32CubeMX tool with the LPBAM interface.

To start an application from scratch using STM32CubeMX, refer to document [2].

The STM32CubeMX tool entry point is always the standard view. The user needs to configure the clock system at 24 MHz.

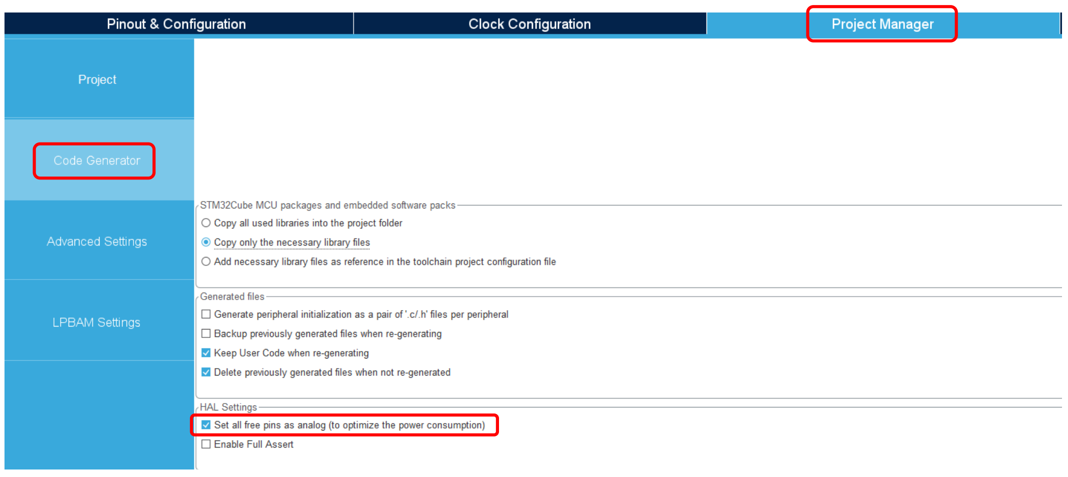
1. Go to *Clock configuration* and choose *MSIS* and *MSIK 24000*.

Figure 11. Clock configuration



2. Set all free pins as analog to save power consumption.
Go to *Project Manager*, then *Code Generator*, and select *Set all free pins as analog*.

Figure 12. GPIO configuration



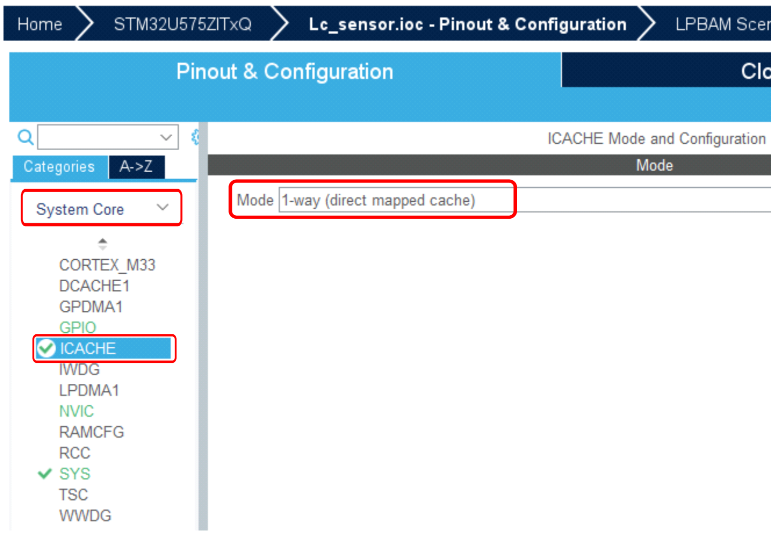
STM32Cube MCU packages and embedded software packs
- ○ Copy all used libraries into the project folder
- ◉ Copy only the necessary library files
- ○ Add necessary library files as reference in the toolchain project configuration file

Generated files
- ☐ Generate peripheral initialization as a pair of '.c/.h' files per peripheral
- ☐ Backup previously generated files when re-generating
- ☑ Keep User Code when re-generating
- ☑ Delete previously generated files when not re-generated

HAL Settings
- ☑ Set all free pins as analog (to optimize the power consumption)
- ☐ Enable Full Assert

3. Enable the ICACHE peripheral in one way configuration.
Click on *System Core*, then *ICACHE* peripheral, and change the *Mode* to *1-way (direct mapped cache)*.

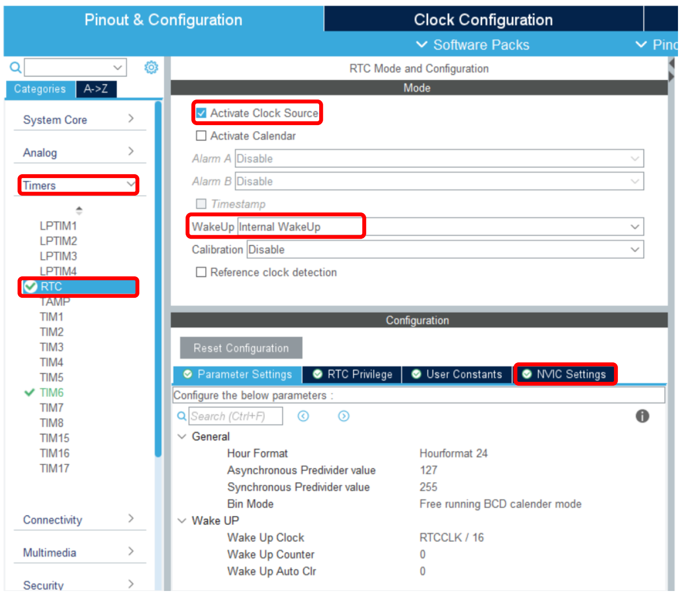
Figure 13. ICACHE activation



Mode 1-way (direct mapped cache)

4. Configure RTC to generate a wakeup interrupt at regular interval for LC sensor calibration.
Click on *Timers*, then *RTC* peripheral. Click on *Activate Clock Source*, then select *Internal WakeUp* for *WakeUp* field. After that, enable the RTC interrupt in *NVIC Settings*.

Figure 14. RTC configuration



☑ Activate Clock Source
☐ Activate Calendar
Alarm A Disable
Alarm B Disable
☐ Timestamp
WakeUp Internal WakeUp
Calibration Disable
☐ Reference clock detection

Configure the below parameters :

General

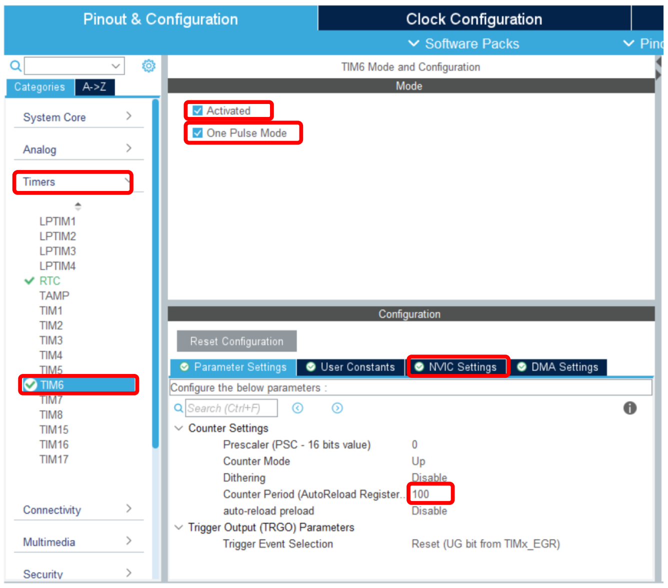
Hour Format	Hourformat 24
Asynchronous Predivider value	127
Synchronous Predivider value	255
Bin Mode	Free running BCD calender mode

Wake UP

Wake Up Clock	RTCCLK / 16
Wake Up Counter	0
Wake Up Auto Clr	0

5. Configure TIM6 to generate precise temporizations used for the LC sensor metering sequence. Click on *Timers*, then *TIM6* peripheral. Click on *Activated* and *One Pusle Mode*, then configure *Counter Period* equal to *100* . After that, enable the TIM6 global interrupt in *NVIC Settings*.

Figure 15. TIM6 configuration



☑ Activated
☑ One Pulse Mode

Configure the below parameters :

Counter Settings

Prescaler (PSC - 16 bits value)	0
Counter Mode	Up
Dithering	Disable
Counter Period (AutoReload Register...	100
auto-reload preload	Disable

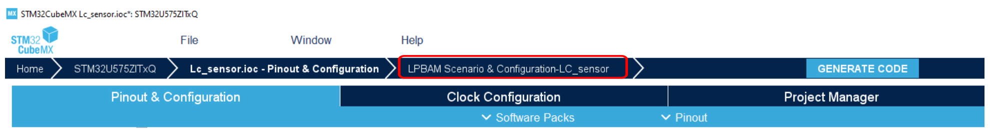
Trigger Output (TRGO) Parameters

Trigger Event Selection	Reset (UG bit from TIMx_EGR)

At this step, the main project system is configured.

6. Build the LPBAM LC_sensor application.
   a. Click on *LPBAM Scenario & Configuration* panel.

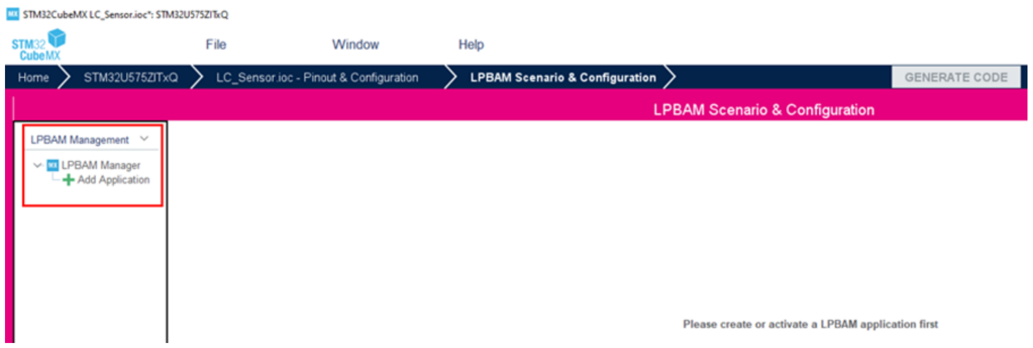
Figure 16. LPBAM scenario and configuration selection



   b. To add an LPBAM application, go to *LPBAM Management*, then *LPBAM Manager*, and click on + *Add Application*.
      When adding an LPBAM application, the STM32CubeMX tool shows the LBPAM view. As for the standard view, it contains *LPBAM Scenario & Configuration*, *Pinout & Configuration*, and *Clock Configuration* panels. The naming chosen in the project should be reused in code-generated APIs and variables. This ensures consistency between STM32CubeMX LPBAM tool views and LPBAM generated application code.

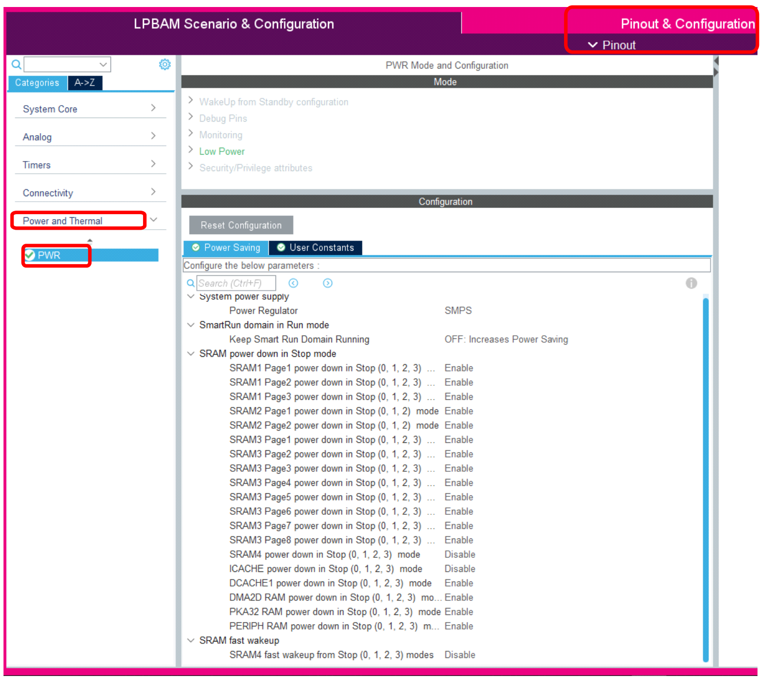
Note: It is recommended to choose the correct user application naming to allow a clear and readable generated code.
   1. Change the name of the application from LpbamAp1 to LC_sensor.
   2. Change the name of the Scenario to LC_sensor_app.

Figure 17. Application and scenario naming


STM32CubeMX LC_Sensor.ioc*: STM32U575ZITxQ
STM32 CubeMX
File
Window
Help
Home
STM32U575ZITxQ
LC_Sensor.ioc - Pinout & Configuration
LPBAM Scenario & Configuration
GENERATE CODE
LPBAM Scenario & Configuration
LPBAM Management
LPBAM Manager
Add Application
Please create or activate a LPBAM application first
DT71530V1

   c. Configure the system power to reach the lowest power consumption.
      Go to *Pinout and Configuration*, then *Power and Thermal*, and *PWR*:
      i. Select SMPS as power regulator.
      ii. Enable SRAM power down in Stop mode for all SRAMs except SRAM4 and ICACHE.

Figure 18. **LPBAM PWR configuration**



```
Configure the below parameters :
System power supply
    Power Regulator                                   SMPS
SmartRun domain in Run mode
    Keep Smart Run Domain Running                     OFF: Increases Power Saving
SRAM power down in Stop mode
    SRAM1 Page1 power down in Stop (0, 1, 2, 3) ...   Enable
    SRAM1 Page2 power down in Stop (0, 1, 2, 3) ...   Enable
    SRAM1 Page3 power down in Stop (0, 1, 2, 3) ...   Enable
    SRAM2 Page1 power down in Stop (0, 1, 2) mode     Enable
    SRAM2 Page2 power down in Stop (0, 1, 2) mode     Enable
    SRAM3 Page1 power down in Stop (0, 1, 2, 3) ...   Enable
    SRAM3 Page2 power down in Stop (0, 1, 2, 3) ...   Enable
    SRAM3 Page3 power down in Stop (0, 1, 2, 3) ...   Enable
    SRAM3 Page4 power down in Stop (0, 1, 2, 3) ...   Enable
    SRAM3 Page5 power down in Stop (0, 1, 2, 3) ...   Enable
    SRAM3 Page6 power down in Stop (0, 1, 2, 3) ...   Enable
    SRAM3 Page7 power down in Stop (0, 1, 2, 3) ...   Enable
    SRAM3 Page8 power down in Stop (0, 1, 2, 3) ...   Enable
    SRAM4 power down in Stop (0, 1, 2, 3) mode        Disable
    ICACHE power down in Stop (0, 1, 2, 3) mode       Disable
    DCACHE1 power down in Stop (0, 1, 2, 3) mode      Enable
    DMA2D RAM power down in Stop (0, 1, 2, 3) mo...   Enable
    PKA32 RAM power down in Stop (0, 1, 2, 3) mode    Enable
    PERIPH RAM power down in Stop (0, 1, 2, 3) m...   Enable
SRAM fast wakeup
    SRAM4 fast wakeup from Stop (0, 1, 2, 3) modes    Disable
```

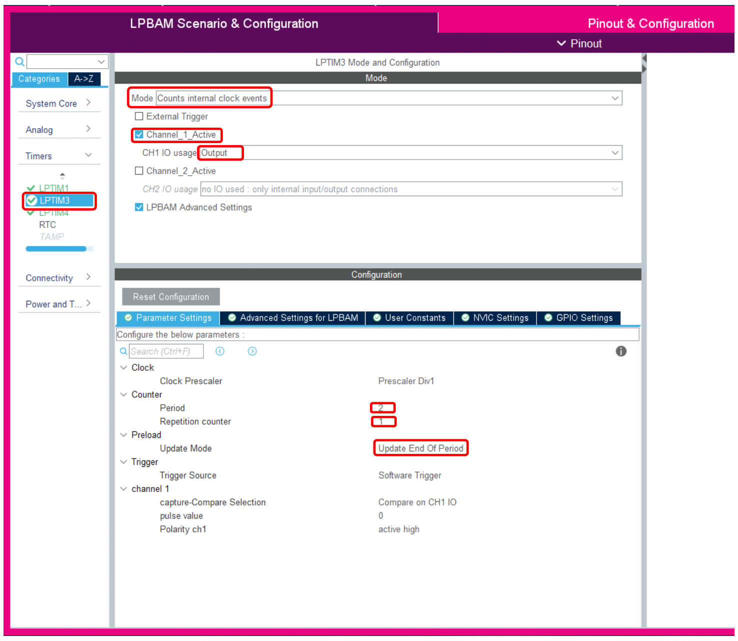
At this step, the system is optimized in terms of consumption and ready to host the LPBAM application operating. LPBAM The configuration of each peripheral (using HAL driver) is then needed.

4.3.1 LPTIM3 configuration

1. Click on *Timers*, then on *LPTIM3*. The *LPTIM3 Mode and Configuration* window appears.
2. Under *Mode*, change the mode from disable to *Counts internal clock events*.
   a. Select the *Channel_1_Active*.
   b. Select *CH1 IO usage* from *no IO used* to *Output* (for debug reasons).

3. Under *Configuration*, go to *Parameter Settings*:
   a. Configure *Period* value to 2.
   b. Configure *Repetition counter* to 1.
   c. Select update mode as *Update End Of period*.

Figure 19. LPTIM3 activation and parameter settings



LPTIM3 Mode and Configuration

Mode

Mode: Counts internal clock events
☐ External Trigger
☑ Channel_1_Active
CH1 IO usage: Output
☐ Channel_2_Active
CH2 IO usage: no IO used : only internal input/output connections
☑ LPBAM Advanced Settings

Configuration

Configure the below parameters :

```
Clock
    Clock Prescaler                Prescaler Div1
Counter
    Period                         2
    Repetition counter             1
Preload
    Update Mode                    Update End Of Period
Trigger
    Trigger Source                 Software Trigger
channel 1
    capture-Compare Selection      Compare on CH1 IO
    pulse value                    0
    Polarity ch1                   active high
```

The LPTIM3 is configured to generate two different sequences:

- Sequence 1: the period value is equal to 100 ms. The repetition counter is equal to 0.
- Sequence 2: The period value is equal to 100 µs. The repetition counter is equal to 5 (if the demo is up to four sensors).

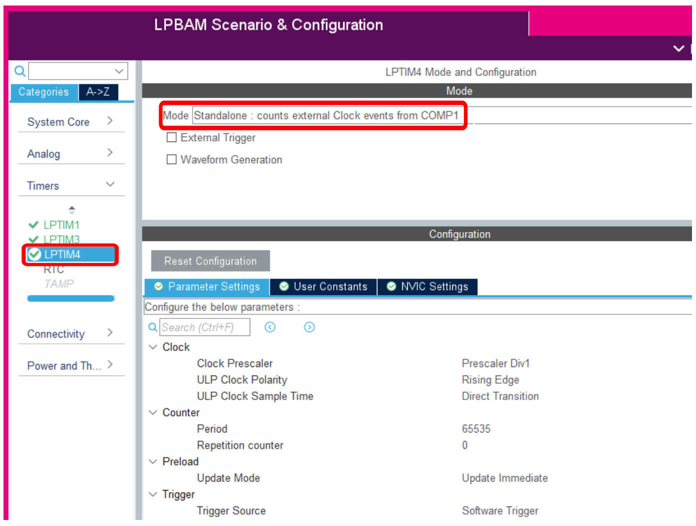
Note: The LPTIM3 must be reconfigured each time with these two sequences using LPDMA execution nodes.

4.3.2 LPTIM4 configuration

1. Click on *Timers*, then on *LPTIM4*. The *LPTIM4 Mode and Configuration* window appears.
2. Under *Mode*, change the mode from disable to *Counts external Clock events from COMP1*.

3. Under *Configuration*, go to *Parameter Settings*.

Note: *Parameter Settings remain by default with an LPTIM period value equal to 0xFFFF.*


LPBAM Scenario & Configuration

LPTIM4 Mode and Configuration

Mode

Mode: Standalone : counts external Clock events from COMP1

☐ External Trigger

☐ Waveform Generation

Configuration

Configure the below parameters :

Clock	
Clock Prescaler	Prescaler Div1
ULP Clock Polarity	Rising Edge
ULP Clock Sample Time	Direct Transition
Counter	
Period	65535
Repetition counter	0
Preload	
Update Mode	Update Immediate
Trigger	
Trigger Source	Software Trigger

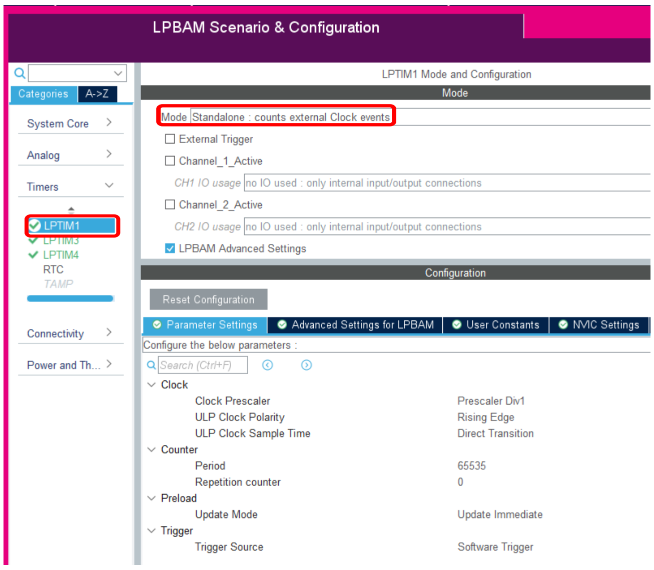
Figure 20. **LPTIM4 activation and parameter settings**

The LPTIM4 is configured to count the pulses generated by the LC sensor oscillations from the output comparator (COMP1).

4.3.3 LPTIM1 configuration

1. Click on *Timers*, then on *LPTIM1*. The *LPTIM1 Mode and Configuration* window appears.
2. Under *Mode*, change the mode from disable to *Counts external Clock events*.
3. Under *Configuration*, go to *Parameter Settings*.

Note: *Parameter Settings remain by default with an LPTIM period value equal to 0xFFFF.*


LPBAM Scenario & Configuration

LPTIM1 Mode and Configuration

Mode

Mode: Standalone : counts external Clock events

☐ External Trigger

☐ Channel_1_Active

CH1 IO usage: no IO used : only internal input/output connections

☐ Channel_2_Active

CH2 IO usage: no IO used : only internal input/output connections

☑ LPBAM Advanced Settings

Configuration

Configure the below parameters :

Clock	
Clock Prescaler	Prescaler Div1
ULP Clock Polarity	Rising Edge
ULP Clock Sample Time	Direct Transition
Counter	
Period	65535
Repetition counter	0
Preload	
Update Mode	Update Immediate
Trigger	
Trigger Source	Software Trigger

Figure 21. **LPTIM1 activation and parameter settings**

The LPTIM1 is configured to count the pulses generated by the LC sensor oscillations from the output comparator (COMP2).

Note: The COMP2 output is not connected internally with the input of the LPTIM. The solution is to configure the LPTIM to count external clock events and connect externally the input of LPTIM1 with the output of COMP2.

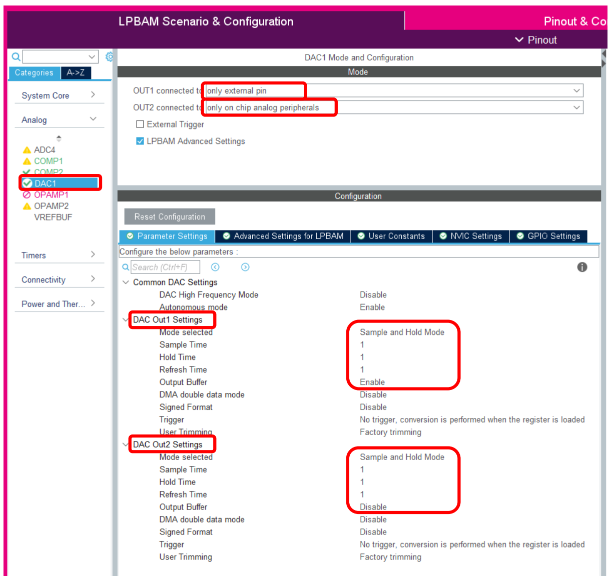
4.3.4 DAC configuration

The DAC configuration for this scenario is described below:

1. Click on *Analog*, then on *DAC*. The *DAC Mode and Configuration* window appears.
2. Under *Mode*:
   a. Select *only external pin* for OUT1 connected to.
   b. Select *only on chip analog peripherals* for OUT2 connected to.
3. Under *Configuration*, go to *Parameter Settings*.
   Configure *DAC Out1 Settings*:
   – Mode selected: Sample and Hold Mode
   – Sample Time: 1
   – Hold Time: 1
   – Refresh Time: 1
   – Output Buffer: Enable

   Configure *DAC Out2 Settings*:
   – Mode selected: Sample and Hold Mode
   – Sample Time: 1
   – Hold Time: 1
   – Refresh Time: 1
   – Output Buffer: Disable

Figure 22. DAC configuration



DAC1 Mode and Configuration

Mode

OUT1 connected to: only external pin
OUT2 connected to: only on chip analog peripherals
☐ External Trigger
☑ LPBAM Advanced Settings

Configuration

Configure the below parameters :

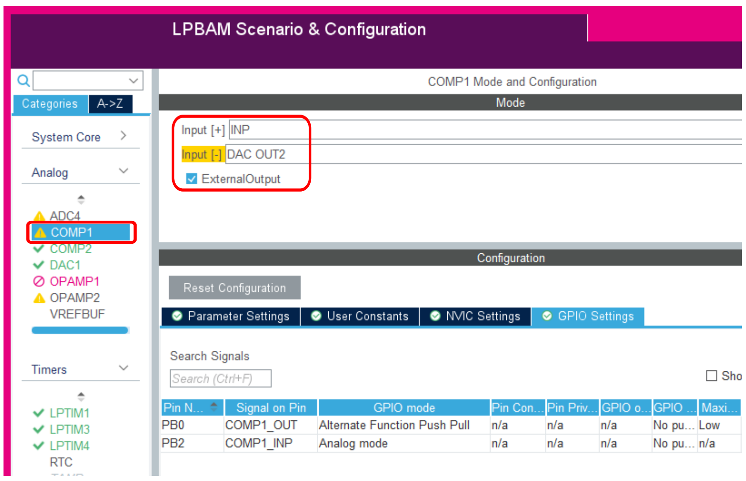
Common DAC Settings	
DAC High Frequency Mode	Disable
Autonomous mode	Enable
DAC Out1 Settings	
Mode selected	Sample and Hold Mode
Sample Time	1
Hold Time	1
Refresh Time	1
Output Buffer	Enable
DMA double data mode	Disable
Signed Format	Disable
Trigger	No trigger, conversion is performed when the register is loaded
User Trimming	Factory trimming
DAC Out2 Settings	
Mode selected	Sample and Hold Mode
Sample Time	1
Hold Time	1
Refresh Time	1
Output Buffer	Disable
DMA double data mode	Disable
Signed Format	Disable
Trigger	No trigger, conversion is performed when the register is loaded
User Trimming	Factory trimming

4.3.5 COMP1 configuration

1. Click on *Analog*, then on *COMP1*. The *COMP1 Mode and Configuration* window appears.

2. Under *Mode*:
   a. Select *INP* as Input [+] and *DAC OUT2* as Input [-].
   b. Select *ExternalOutput* to generate the COMP1 output signal (for debug reasons).

Figure 23. **COMP1 parameter settings**



COMP1 Mode and Configuration

Mode

Input [+] INP
Input [-] DAC OUT2
✓ ExternalOutput

Configuration

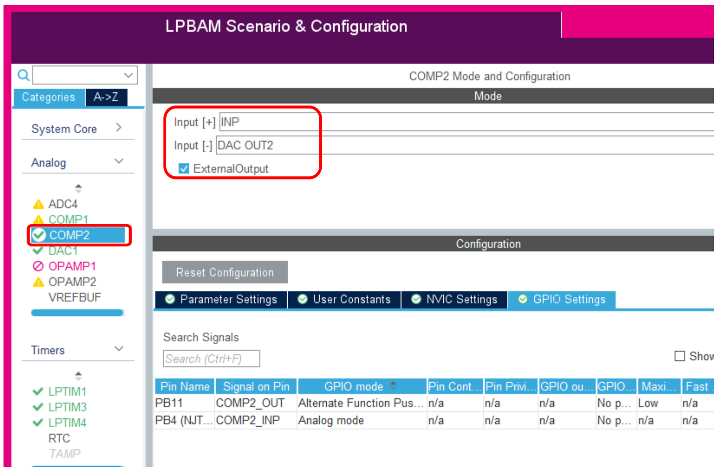
Pin N...	Signal on Pin	GPIO mode	Pin Con...	Pin Priv...	GPIO o...	GPIO ...	Maxi...
PB0	COMP1_OUT	Alternate Function Push Pull	n/a	n/a	n/a	No pu...	Low
PB2	COMP1_INP	Analog mode	n/a	n/a	n/a	No pu...	n/a

DT70582V1

4.3.6 COMP2 configuration

1. Click on *Analog*, then on *COMP2*. The *COMP2 Mode and Configuration* window appears.
2. Under *Mode*:
   a. Select *INP* as Input [+] and *DAC OUT2* as Input [-].
   b. Select *ExternalOutput* to generate the COMP2 output signal (mandatory to be connected externally to the LPTIM1 input).

Figure 24. **COMP2 parameter settings**



COMP2 Mode and Configuration

Mode

Input [+] INP
Input [-] DAC OUT2
✓ ExternalOutput

Configuration

Pin Name	Signal on Pin	GPIO mode	Pin Cont...	Pin Privi...	GPIO ou...	GPIO...	Maxi...	Fast
PB11	COMP2_OUT	Alternate Function Pus...	n/a	n/a	n/a	No p...	Low	n/a
PB4 (NJT...	COMP2_INP	Analog mode	n/a	n/a	n/a	No p...	n/a	n/a

DT70583V1

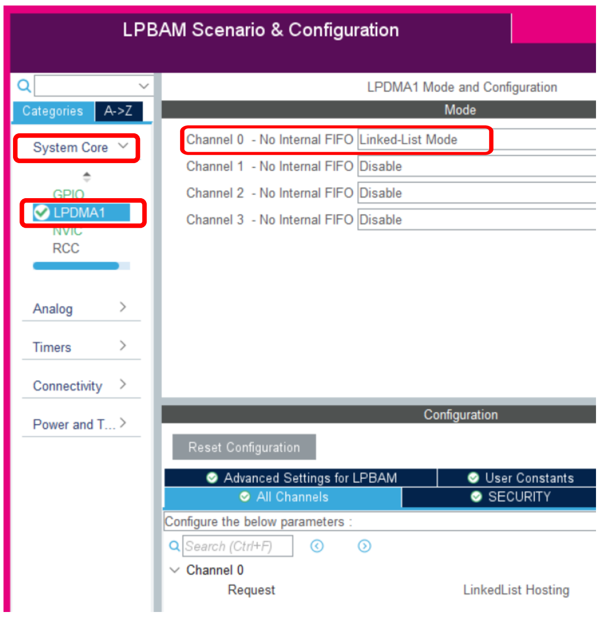
4.3.7 LPDMA configuration

Three LDPMA channels are configured in linked-list mode:

1. Click on *System Core*, then on *LPDMA1*. The *LPDMA1 Mode and Configuration* window appears.

2. Under *Mode*, select *Linked-List Mode* for Channel 0 - No Internal FIFO.

Figure 25. **LPDMA configuration**


LPBAM Scenario & Configuration

LPDMA1 Mode and Configuration

Mode

Channel 0 - No Internal FIFO	Linked-List Mode
Channel 1 - No Internal FIFO	Disable
Channel 2 - No Internal FIFO	Disable
Channel 3 - No Internal FIFO	Disable

Configuration

Configure the below parameters :

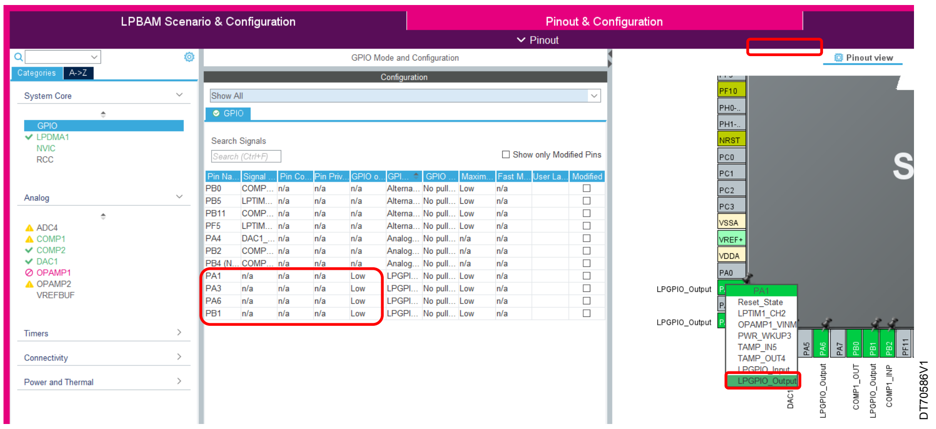
Channel 0
Request LinkedList Hosting

4.3.8 GPIO configuration

Four LPGPIO pins must be configured in output mode (one LPGIO pin for each sensor). The other pins are configured automatically when the peripherals are configured.

1. Click on *System Core*, then on *GPIO*. The *GPIO Mode and Configuration* window appears.
2. Click on *Pinout view*, and select *LPGIO Output* for PA1, PA3, PA6, and PB1.

Figure 26. **GPIO configuration**


GPIO Mode and Configuration

Configuration

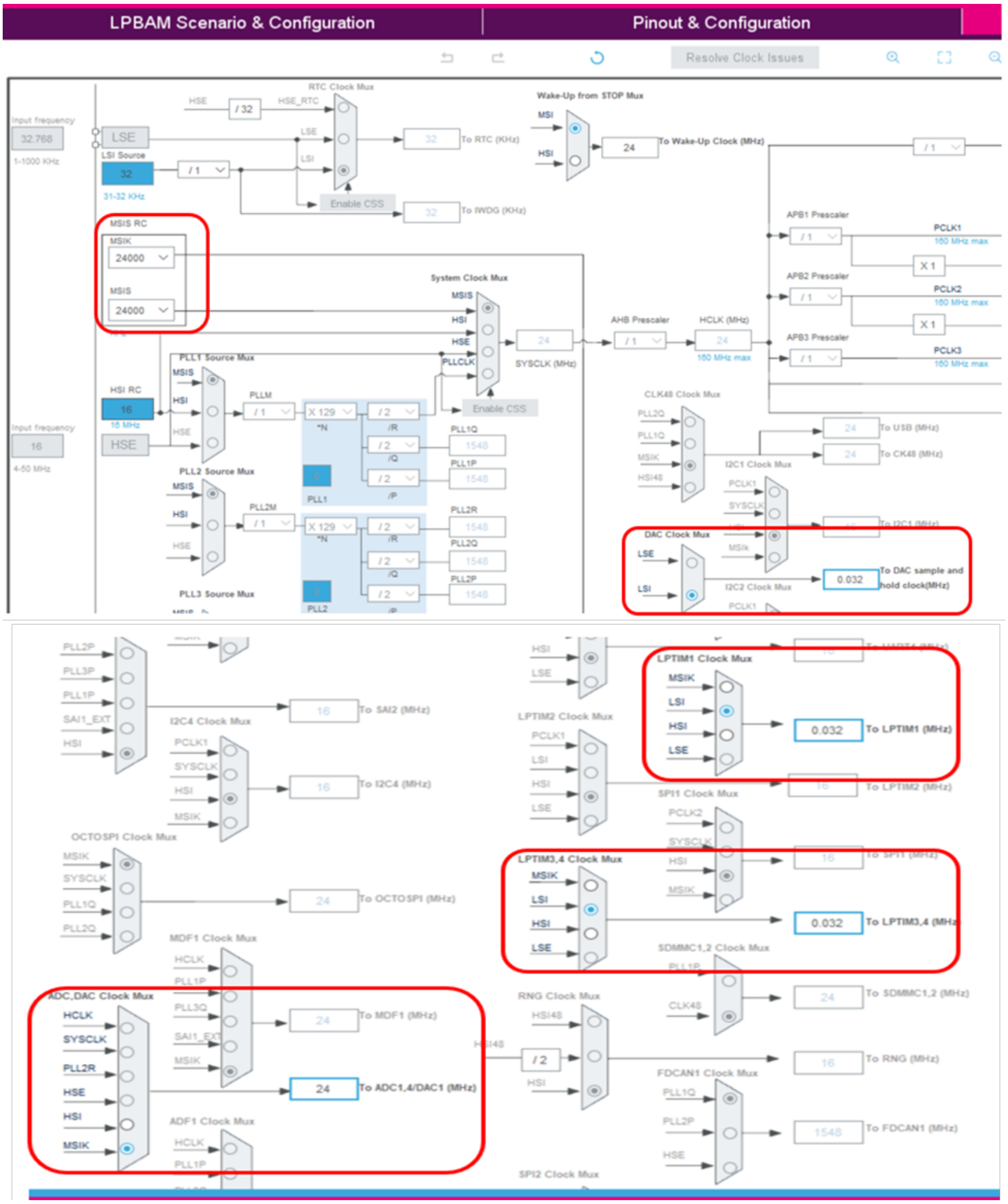
Pin Na...	Signal ...	Pin Co...	Pin Priv...	GPIO o...	GPI...	GPIO ...	Maxim...	Fast M...	User La...	Modified
PB0	COMP...	n/a	n/a	n/a	Alterna...	No pull...	Low	n/a		☐
PB5	LPTIM...	n/a	n/a	n/a	Alterna...	No pull...	Low	n/a		☐
PB11	COMP...	n/a	n/a	n/a	Alterna...	No pull...	Low	n/a		☐
PF5	LPTIM...	n/a	n/a	n/a	Alterna...	No pull...	Low	n/a		☐
PA4	DAC1_...	n/a	n/a	n/a	Analog...	No pull...	n/a	n/a		☐
PB2	COMP...	n/a	n/a	n/a	Analog...	No pull...	n/a	n/a		☐
PB4 (N...	COMP...	n/a	n/a	n/a	Analog...	No pull...	n/a	n/a		☐
PA1	n/a	n/a	n/a	Low	LPGPI...	No pull...	Low	n/a		☐
PA3	n/a	n/a	n/a	Low	LPGPI...	No pull...	Low	n/a		☐
PA6	n/a	n/a	n/a	Low	LPGPI...	No pull...	Low	n/a		☐
PB1	n/a	n/a	n/a	Low	LPGPI...	No pull...	Low	n/a		☐

At this step, the DAC, COMP1/2 and LPTIM1/3/4 are configured and still needed to configure their own clock.

1. Go to *Clock Configuration Panel*, then configure *MSIS* and *MSIK = 24000*.

2. Select *LSI* as DAC and LPTIM1/3/4 clock, and *MSIK* as ADC and DAC clock.

Figure 27. **Clock configuration**



All LPBAM peripherals are then configured and ready for scenario queues building.
It is mandatory to ensure that all used resources are configured before starting the scenario building.

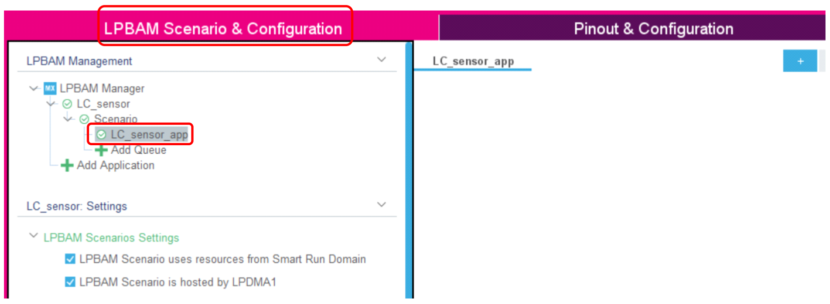
4.4 Scenario queues building

The queue naming in the project is reused in code generated APIs and variables.

Note: *The queue name is suffixed automatically by "_Q" at the generated code.*

Go to *LPBAM Scenario & Configuration*, and change the name of Queue1 to *LC_sensor_app*.

Figure 28. Queue naming



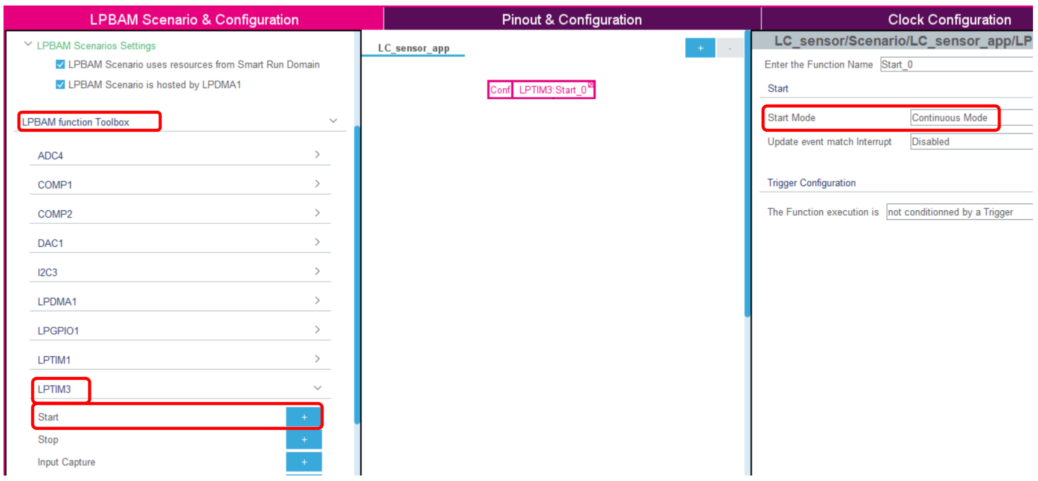
LPBAM Scenario uses resources from Smart Run Domain ✓
LPBAM Scenario is hosted by LPDMA1 ✓

For this queue, several nodes are created to implement the LC sensor scenario. First, three nodes must be added to *Start*. Then, configure the period and the repetition counter of LPTIM3 to trigger the system at regular interval.

4.4.1 LPTIM3 node configuration

1. Click on *LPBAM function Toolbox*. The function list opens.
2. Click on *LPTIM3*. The scenario mode opens.
   a. Click on + of Start to add the configuration. A node named *LPTIM3:Start_1* is created and added to the queue.
   b. Select *Continuous Mode* as Start Mode.

Figure 29. Start LPTIM3 node



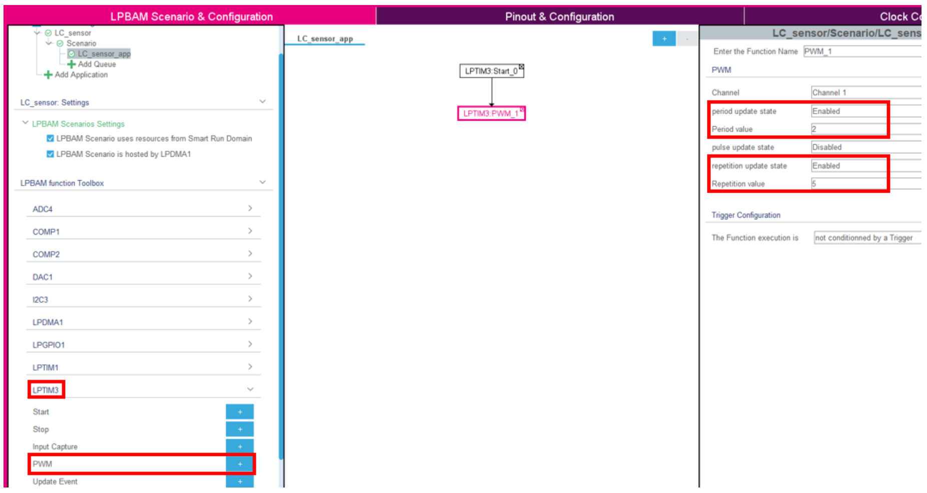
LPBAM Scenario uses resources from Smart Run Domain ✓
LPBAM Scenario is hosted by LPDMA1 ✓

Enter the Function Name	Start_0
Start Mode	Continuous Mode
Update event match Interrupt	Disabled
The Function execution is	not conditionned by a Trigger

Note: *Change the node name Start_1 to Start_0.*

3. Under *LPTIM3*, add *PWM*. A node named *LPTIM:PWM_1* is created and added to the queue.
   Configure *LPTIM:PWM_1*:
   – Channel: Channel 1
   – period update state: Enabled
   – Period value: 2 (to generate a PWM signal with a period equal to 100 µs)
   – Repetition update state: Enabled
   – Repetition value: 5 (according to the number of sensors used)

Figure 30. LPTIM3 PWM_1 configuration



Enter the Function Name PWM_1

PWM

Channel	Channel 1
period update state	Enabled
Period value	2
pulse update state	Disabled
repetition update state	Enabled
Repetition value	5

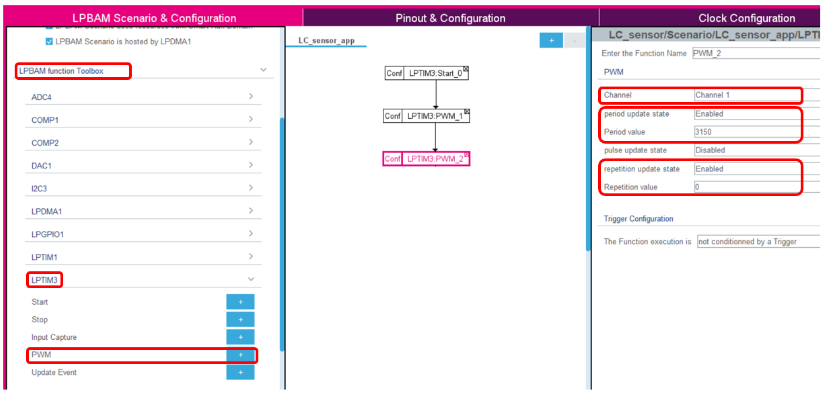
Trigger Configuration

The Function execution is not conditionned by a Trigger

4. Configure a second *PWM* to trigger the LC sensor acquisition at regular interval. A node named *LPTIM3:PWM_2* is created and added to the queue.
   Configure *LPTIM3:PWM_2*:
   – Channel: Channel 1
   – period update state: Enabled
   – Period value: 3150 (to generate a PWM signal with a period equal to 100 ms)
   – Repetition update state: Enabled
   – Repetition value: 0 (to generate a pulse)

Note: *The period value depends on the sampling period. In the example below, the sampling period is equal to 100 Hz.*

Figure 31. LPTIM3 PWM_2 node configuration



Enter the Function Name PWM_2

PWM

Channel	Channel 1
period update state	Enabled
Period value	3150
pulse update state	Disabled
repetition update state	Enabled
Repetition value	0

Trigger Configuration

The Function execution is not conditionned by a Trigger

4.4.2 DAC1 node configuration

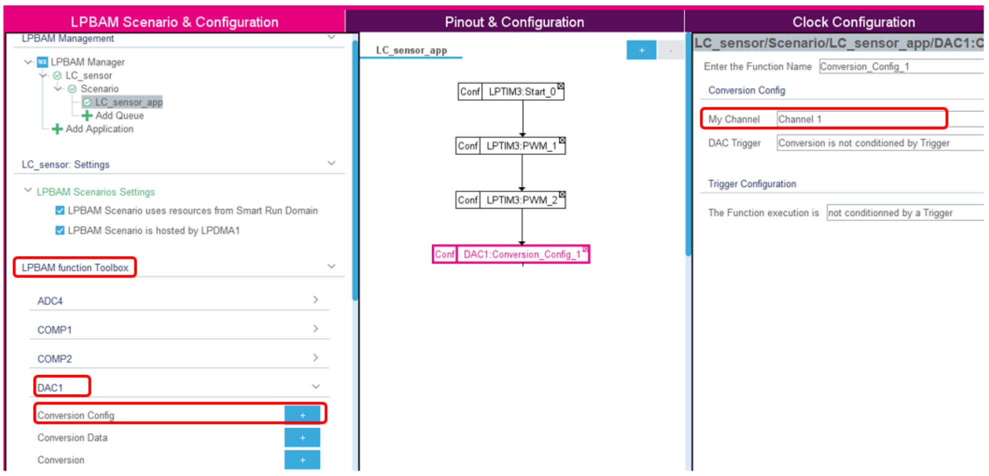
Under *LPBAM function Toolbox*, select *DAC1*. The scenario mode opens.

Click on + of *Conversion Config* to add the configuration. A node named *DAC1:Conversion_Config_1* is created and added to the queue.

Configure *Conversion_Config_1*:

- My Channel: Channel 1

Figure 32. DAC1 node configuration



LPBAM Scenario uses resources from Smart Run Domain
LPBAM Scenario is hosted by LPDMA1

Enter the Function Name: Conversion_Config_1
My Channel: Channel 1
DAC Trigger: Conversion is not conditioned by Trigger
The Function execution is: not conditionned by a Trigger

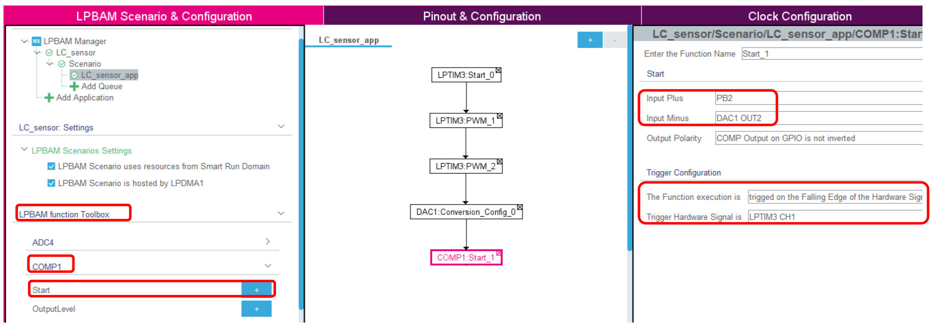
Note: There is a limitation in the STM32CubeMX LPBAM interface to configure both DAC channels at the same time. The solution is to create manually a node to enable both DAC channels.

4.4.3 COMP1 configuration for sensor 1

Under *LPBAM function Toolbox*, select *COMP1*. The scenario mode opens.

1. Click on + of *Start* to add the configuration. A node named *COMP1:Start_1* is created and added to the queue.

Figure 33. COMP1 node configuration for sensor 1



LPBAM Scenario uses resources from Smart Run Domain
LPBAM Scenario is hosted by LPDMA1

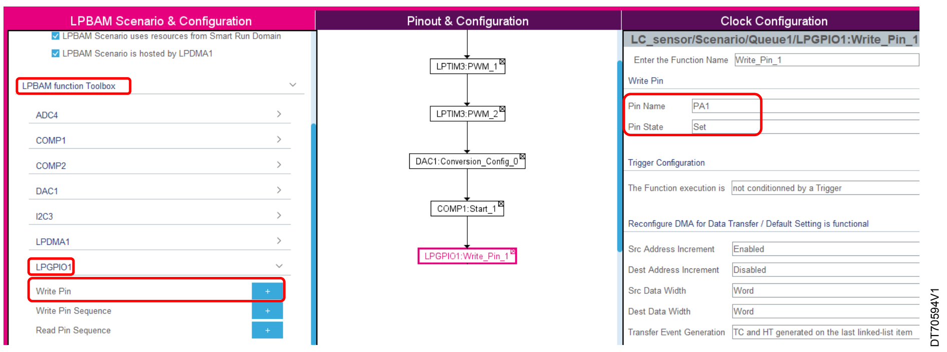
Enter the Function Name: Start_1
Input Plus: PB2
Input Minus: DAC1 OUT2
Output Polarity: COMP Output on GPIO is not inverted
The Function execution is: trigged on the Falling Edge of the Hardware Sig
Trigger Hardware Signal is: LPTIM3 CH1

4.4.4 LPGPIO1 node configuration for sensor 1

To start the LC oscillation for sensor 1, two successive nodes must be created to set and reset the LPGPIO excitation pin (PA2).

Under *LPBAM function Toolbox*, select *LPGPIO1*. The scenario mode opens.

1. Click on + of *Write Pin* to add the configuration. A node named *LGPIO1:Write_Pin_1* is created and added to the queue.
   Configure *LGPIO1:Write_Pin_1*:
   – Pin Name: PA1
   – Pin State: Set

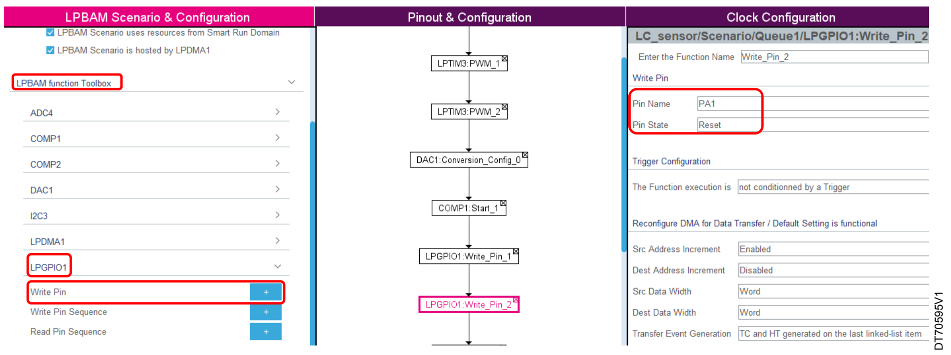


LC_sensor/Scenario/Queue1/LPGPIO1:Write_Pin_1

Enter the Function Name	Write_Pin_1
Write Pin	
Pin Name	PA1
Pin State	Set
Trigger Configuration	
The Function execution is	not conditionned by a Trigger
Reconfigure DMA for Data Transfer / Default Setting is functional	
Src Address Increment	Enabled
Dest Address Increment	Disabled
Src Data Width	Word
Dest Data Width	Word
Transfer Event Generation	TC and HT generated on the last linked-list item

Figure 34. LPGPIO1 node 1 configuration for sensor 1

2. Add another *Write Pin*. A node named *LPGPIO1:Write_Pin_2* is created and added to the queue.
   Configure *LPGPIO1:Write_Pin_2*:
   – Pin Name: PA1
   – Pin State: Reset



LC_sensor/Scenario/Queue1/LPGPIO1:Write_Pin_2

Enter the Function Name	Write_Pin_2
Write Pin	
Pin Name	PA1
Pin State	Reset
Trigger Configuration	
The Function execution is	not conditionned by a Trigger
Reconfigure DMA for Data Transfer / Default Setting is functional	
Src Address Increment	Enabled
Dest Address Increment	Disabled
Src Data Width	Word
Dest Data Width	Word
Transfer Event Generation	TC and HT generated on the last linked-list item

Figure 35. LPGPIO1 node 2 configuration for sensor 1

4.4.5 COMP1 node configuration for sensor 2

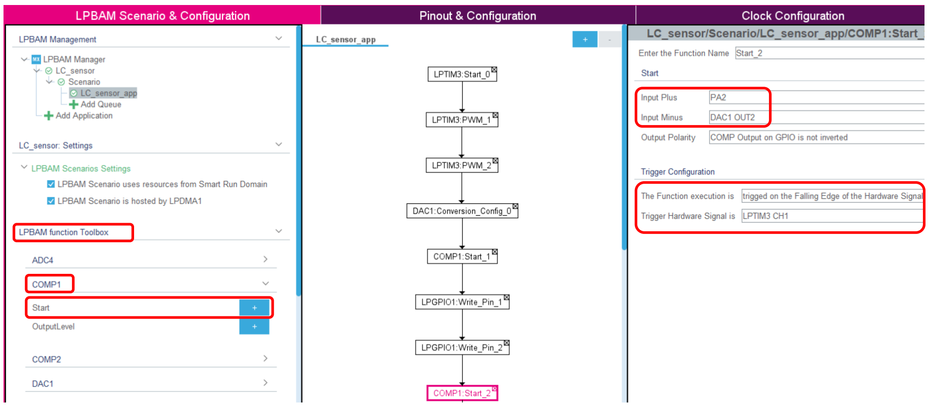
To add a second sensor, configure COMP1 with different input and configure another LPGPIO as detailed in Section 3.2.3 .

The next node named *COMP1:Start_2* is created and added to the queue to start COMP1 with different input Plus.

Under *LPBAM function Toolbox*, select *COMP1*. The scenario mode opens.

1. Click on + of *Start* to add the configuration. A node named *COMP1:Start_2* is created and added to the queue.
   Configure *COMP1:Start_2*:
   – Input Plus: PA2
   – Input Minus: DAC1 OUT2
   – Trigger Hardware Signal is: LPTIM CH1 on the falling Edge (to create a delay between two sensors)

Figure 36. COMP1 node configuration for sensor 2



☑ LPBAM Scenario uses resources from Smart Run Domain
☑ LPBAM Scenario is hosted by LPDMA1

LC_sensor/Scenario/LC_sensor_app/COMP1:Start_

Enter the Function Name: Start_2

Start
- Input Plus: PA2
- Input Minus: DAC1 OUT2
- Output Polarity: COMP Output on GPIO is not inverted

Trigger Configuration
- The Function execution is: trigged on the Falling Edge of the Hardware Signal
- Trigger Hardware Signal is: LPTIM3 CH1

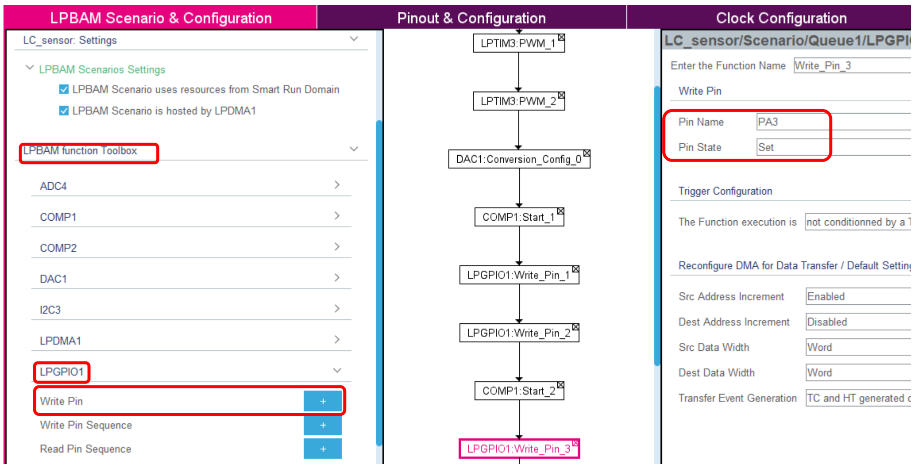
4.4.6 LPGPIO1 node configuration for sensor 2

To start the LC oscillation for sensor 2, two successive nodes must be created to set and reset the LPGPIO excitation pin (PA3).

Under *LPBAM function Toolbox*, select *LPGPIO1*. The scenario mode opens.

1. Click on + of *Write Pin* to add the configuration. A node named *LPGPIO1:Write_Pin_3* is created and added to the queue.
   Configure *LPGPIO1:Write_Pin_3*:
   – Pin Name: PA3
   – Pin State: Set

Figure 37. LPGPIO1 node 3 configuration for sensor 2



☑ LPBAM Scenario uses resources from Smart Run Domain
☑ LPBAM Scenario is hosted by LPDMA1

LC_sensor/Scenario/Queue1/LPGPI

Enter the Function Name: Write_Pin_3

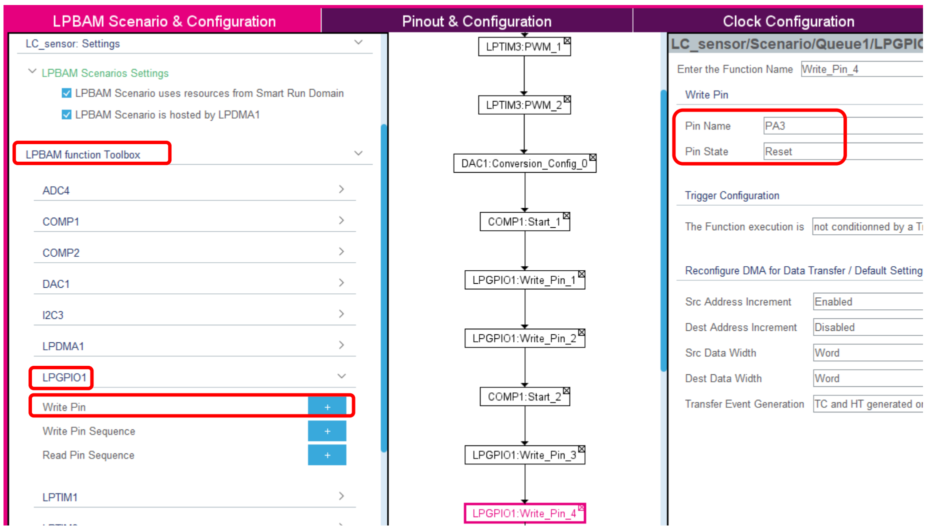
Write Pin
- Pin Name: PA3
- Pin State: Set

Trigger Configuration
- The Function execution is: not conditionned by a T

Reconfigure DMA for Data Transfer / Default Setting
- Src Address Increment: Enabled
- Dest Address Increment: Disabled
- Src Data Width: Word
- Dest Data Width: Word
- Transfer Event Generation: TC and HT generated c

2. Add another *Write Pin*. A node named *LPGPIO1:Write_Pin_4* is created and added to the queue. Configure *LPGPIO1:Write_Pin_4*:
   – Pin Name: PA3
   – Pin State: Reset

Figure 38. LPGPIO1 node 4 configuration for sensor 2



LC_sensor: Settings

LPBAM Scenarios Settings
- ☑ LPBAM Scenario uses resources from Smart Run Domain
- ☑ LPBAM Scenario is hosted by LPDMA1

LPBAM function Toolbox
- ADC4
- COMP1
- COMP2
- DAC1
- I2C3
- LPDMA1
- LPGPIO1
  - Write Pin +
  - Write Pin Sequence +
  - Read Pin Sequence +
- LPTIM1

LC_sensor/Scenario/Queue1/LPGPIO

Enter the Function Name	Write_Pin_4
Write Pin	
Pin Name	PA3
Pin State	Reset
Trigger Configuration	
The Function execution is	not conditionned by a T
Reconfigure DMA for Data Transfer / Default Setting	
Src Address Increment	Enabled
Dest Address Increment	Disabled
Src Data Width	Word
Dest Data Width	Word
Transfer Event Generation	TC and HT generated o

4.4.7 COMP2 node configuration for sensor 3

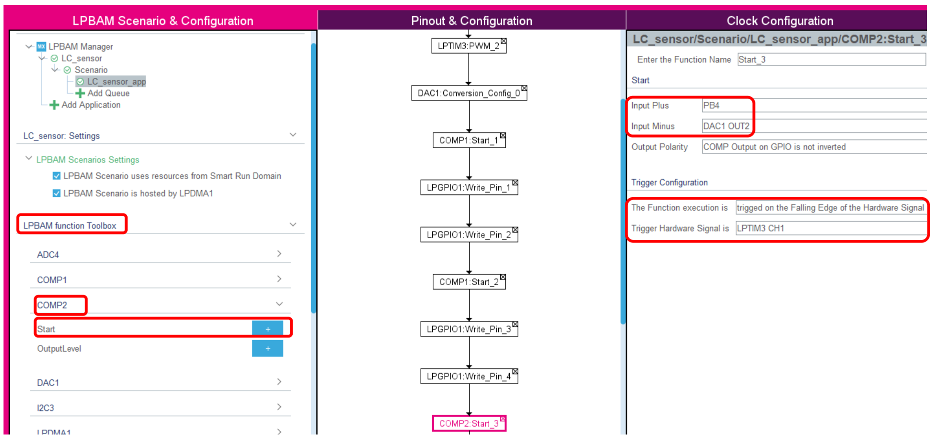
To add a third sensor, configure COMP2 with different input and configure another LPGPIO as detailed in Section 3.2.3 .

The next node named *COMP2:Start_3* is created and added to the queue to start COMP2 with specific input.

Under *LPBAM function Toolbox*, select *COMP2*. The scenario mode opens.

1. Click on + of *Start* to add the configuration. A node named *COMP2:Start_3* is created and added to the queue.
   Configure *COMP2:Start_3*:
   – Input Plus: PB4
   – Input Minus: DAC1 OUT2
   – Trigger Hardware Signal is: LPTIM CH1 on the falling Edge (to create a delay between two sensors)

Figure 39. COMP2 node configuration for sensor 3



LPBAM Manager
- LC_sensor
  - Scenario
    - LC_sensor_app
    - Add Queue
  - Add Application

LC_sensor: Settings

LPBAM Scenarios Settings
- ☑ LPBAM Scenario uses resources from Smart Run Domain
- ☑ LPBAM Scenario is hosted by LPDMA1

LPBAM function Toolbox
- ADC4
- COMP1
- COMP2
  - Start +
  - OutputLevel +
- DAC1
- I2C3
- LPDMA1

LC_sensor/Scenario/LC_sensor_app/COMP2:Start_3

Enter the Function Name	Start_3
Start	
Input Plus	PB4
Input Minus	DAC1 OUT2
Output Polarity	COMP Output on GPIO is not inverted
Trigger Configuration	
The Function execution is	trigged on the Falling Edge of the Hardware Signal
Trigger Hardware Signal is	LPTIM3 CH1

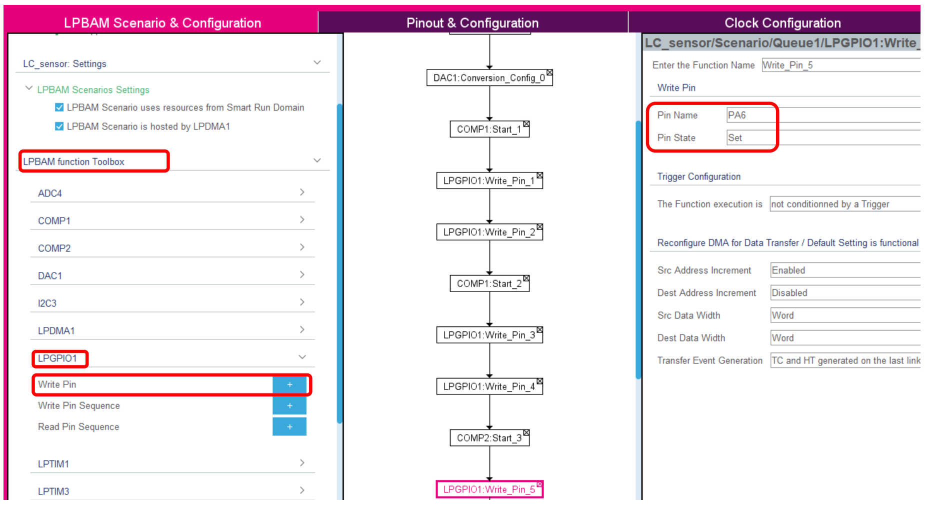
4.4.8 LPGPIO1 node configuration for sensor 3

To start the LC oscillation for sensor 3, three successive nodes must be created to set and reset the LPGPIO excitation pin (PA6).

Under *LPBAM function Toolbox*, select *LPGPIO1*. The scenario mode opens.

1. Click on + of *Write Pin* to add the configuration. A node named *LPGPIO1:Write_Pin_5* is created and added to the queue.
   Configure *LPGPIO1:Write_Pin_5*:
   – Pin Name: PA6
   – Pin State: Set

Figure 40. LPGPIO1 node 5 configuration for sensor 3

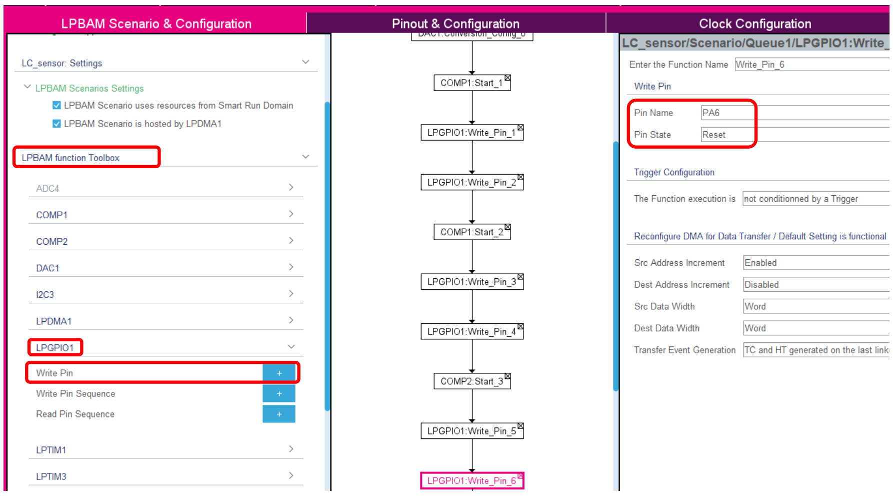


LC_sensor/Scenario/Queue1/LPGPIO1:Write_

Enter the Function Name	Write_Pin_5
Write Pin	
Pin Name	PA6
Pin State	Set
Trigger Configuration	
The Function execution is	not conditionned by a Trigger
Reconfigure DMA for Data Transfer / Default Setting is functional	
Src Address Increment	Enabled
Dest Address Increment	Disabled
Src Data Width	Word
Dest Data Width	Word
Transfer Event Generation	TC and HT generated on the last link

2. Add another *Write Pin*. A node named *LPGPIO1:Write_Pin_6* is created and added to the queue.
   Configure *LPGPIO1:Write_Pin_6*:
   – Pin Name: PA6
   – Pin State: Reset

Figure 41. LPGPIO1 node 6 configuration for sensor 3



LC_sensor/Scenario/Queue1/LPGPIO1:Write_

Enter the Function Name	Write_Pin_6
Write Pin	
Pin Name	PA6
Pin State	Reset
Trigger Configuration	
The Function execution is	not conditionned by a Trigger
Reconfigure DMA for Data Transfer / Default Setting is functional	
Src Address Increment	Enabled
Dest Address Increment	Disabled
Src Data Width	Word
Dest Data Width	Word
Transfer Event Generation	TC and HT generated on the last link

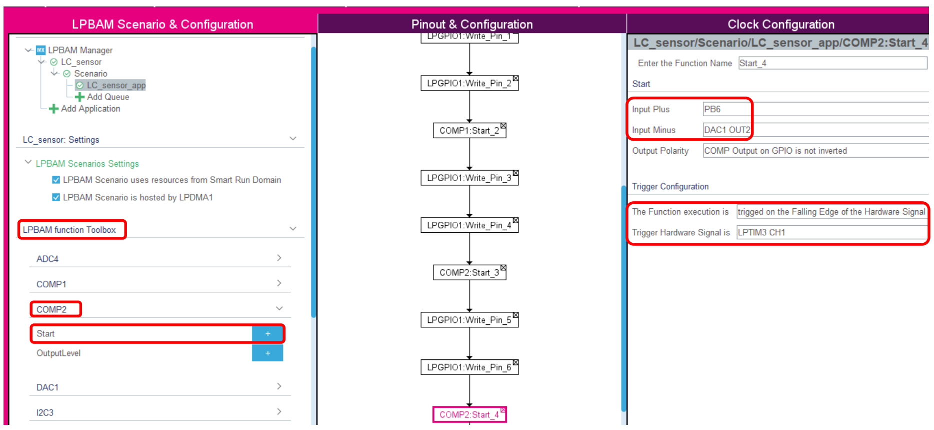
4.4.9 COMP2 node configuration for sensor 4

To add a fourth sensor, configure COMP2 with different input Plus.

The next node named *COMP2:Start_4* is created and added to the queue to start COMP2 with another input Plus.

Under *LPBAM function Toolbox*, select *COMP2*. The scenario mode opens.

1. Click on + of *Start* to add the configuration. A node named *COMP2:Start_4* is created and added to the queue.
   Configure *COMP2:Start_4*:
   – Input Plus: PB6
   – Input Minus: DAC1 OUT2
   – Trigger Hardware Signal is: LPTIM3 CH1 on the falling Edge (to create a delay between two sensors)



☑ LPBAM Scenario uses resources from Smart Run Domain
☑ LPBAM Scenario is hosted by LPDMA1

LC_sensor/Scenario/LC_sensor_app/COMP2:Start_4

Enter the Function Name	Start_4
Start	
Input Plus	PB6
Input Minus	DAC1 OUT2
Output Polarity	COMP Output on GPIO is not inverted
Trigger Configuration	
The Function execution is	trigged on the Falling Edge of the Hardware Signal
Trigger Hardware Signal is	LPTIM3 CH1

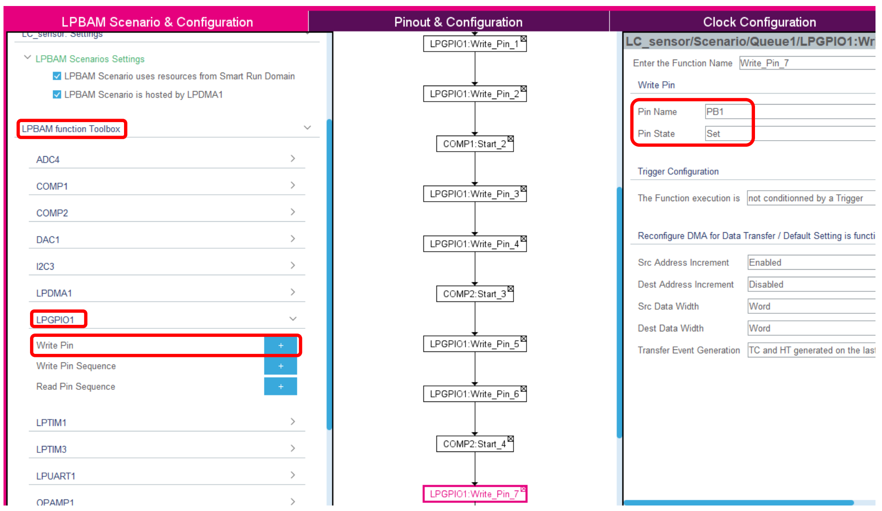
Figure 42. COMP2 node configuration for sensor 4

4.4.10 LGPIO1 node configuration for sensor 4

To start the LC oscillation for sensor 4, two successive nodes must be created to set and reset the LPGPIO excitation pin (PB1).

Under *LPBAM function Toolbox*, select *LPGPIO1*. The scenario mode opens.

1. Click on + of *Write Pin* to add the configuration. A node named *LPGPIO1:Write_Pin_7* is created and added to the queue.
   Configure *LPGPIO1:Write_Pin_7*:
   – Pin Name: PB1
   – Pin State: Set



☑ LPBAM Scenario uses resources from Smart Run Domain
☑ LPBAM Scenario is hosted by LPDMA1

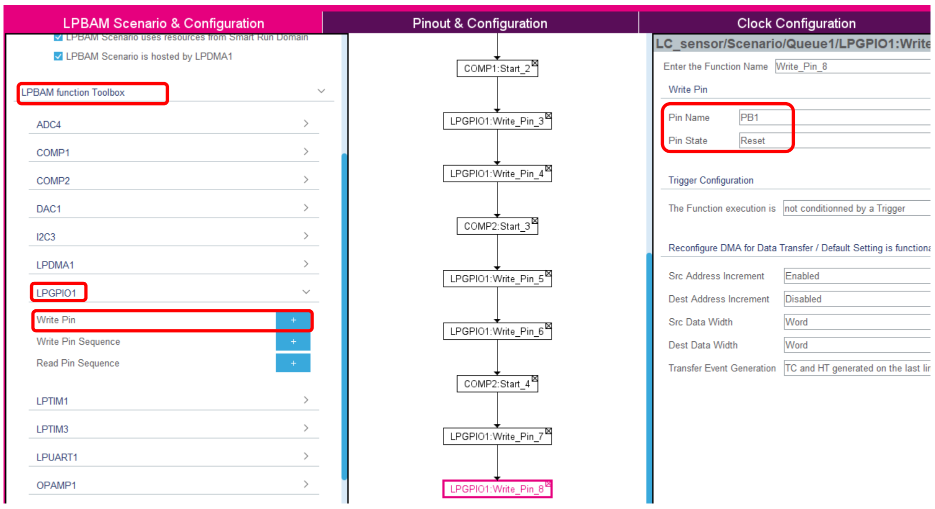
LC_sensor/Scenario/Queue1/LPGPIO1:Wr

Enter the Function Name	Write_Pin_7
Write Pin	
Pin Name	PB1
Pin State	Set
Trigger Configuration	
The Function execution is	not conditionned by a Trigger
Reconfigure DMA for Data Transfer / Default Setting is functi	
Src Address Increment	Enabled
Dest Address Increment	Disabled
Src Data Width	Word
Dest Data Width	Word
Transfer Event Generation	TC and HT generated on the last

Figure 43. LPGPIO1 node 7 configuration for sensor 4

2. Add another *Write Pin*. A node named *LPGPIO1:Write_Pin_8* is created and added to the queue. Configure *LPGPIO1:Write_Pin_8*:
   – Pin Name: PB1
   – Pin State: Reset

Figure 44. **LPGPIO1 node 8 configuration for sensor 4**



LC_sensor/Scenario/Queue1/LPGPIO1:Write

Enter the Function Name: Write_Pin_8

Write Pin

Pin Name: PB1
Pin State: Reset

Trigger Configuration

The Function execution is: not conditionned by a Trigger

Reconfigure DMA for Data Transfer / Default Setting is functiona

Src Address Increment: Enabled
Dest Address Increment: Disabled
Src Data Width: Word
Dest Data Width: Word
Transfer Event Generation: TC and HT generated on the last li

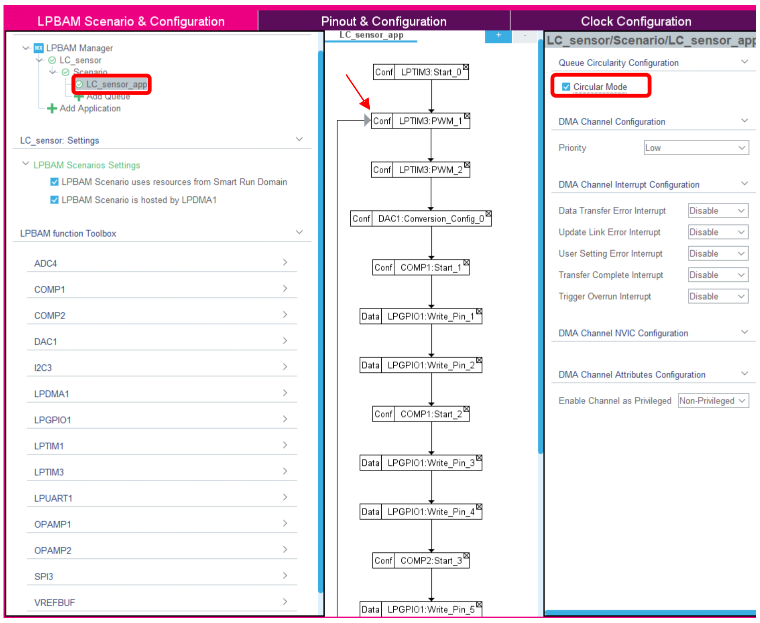
4.4.11 LPDMA circular mode queue

At this step, all the LPBAM application needs with STM32CubeMX are configured. The DMA channel execution mode must be configured in circular mode.

Under *LPBAM Scenario & Configuration*, select *LC_sensor_app*. The node window opens.

1. Check the *Circular Mode* box.
2. Drag the arrow on *LPTIM3:PWM_1* node.

Figure 45. **LPDMA circular mode queue**



LC_sensor/Scenario/LC_sensor_app

Queue Circularity Configuration

☑ Circular Mode

DMA Channel Configuration

Priority: Low

DMA Channel Interrupt Configuration

Data Transfer Error Interrupt: Disable
Update Link Error Interrupt: Disable
User Setting Error Interrupt: Disable
Transfer Complete Interrupt: Disable
Trigger Overrun Interrupt: Disable

DMA Channel NVIC Configuration

DMA Channel Attributes Configuration

Enable Channel as Privileged: Non-Privileged

3. After the code generation, disable DAC and COMP1/2. Three nodes are created and added to the same queue named *LC_sensor_app_Q*. A first *Stop_DAC* node is placed after *Write_Pin_8_Desc* node. *Stop_COMP1* and *Stop_COMP2* nodes are placed after *Stop_DAC* node.
4. At this step, all the LPBAM LC sensors are built in the STM32CubeMX with LPBAM interface. To check the presence of metal, store the LPTIM counter value after each LC sensor oscillation.
5. Create four nodes that are added to the same queue named *LC_sensor_app_Q*. This is to store the LPTIM counter for four sensors.

Note: These nodes are to be created automatically with the STM32CubeMX with LPBAM interface once the feature `ADV_LPBAM_LPTIM_GetCounterValue_SetFullQ()` *is implemented.*

5 Demonstration with four sensors

5.1 Data collection

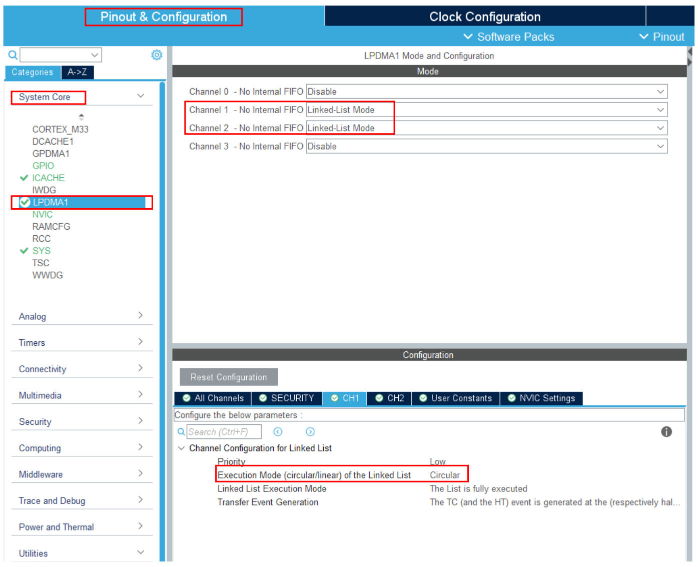
At this step, the LPTIM counter values are stored in a 16-bit buffers. Two LPDMA channels (CH1 and CH2) are configured to concatenate two buffers of 16 bits into 32 bits.

The two buffers Sensor and Sensor2 are stored in a 32-bit buffer named Mem_s1s2[]. The two other buffers Sensor3 and Sensor4 are stored in a 32-bit buffer named Mem_s3s4[].

Configure LPDMA CH1 and CH2 in linked-list mode with the following steps:

1. Go to the standard view of the STM32CubeMX tool. Go to *Pinout & Configuration*, then *System Core*, and select *LPDMA1*. Select *Linked-List Mode* for Chanel 1 and Channel 2.
   Under *Configuration*, set the Linked-List Execution Mode as *Circular* for CH1 and CH2.

Figure 46. LPDMA CH1 and CH2 configuration



Channel 0 - No Internal FIFO: Disable
Channel 1 - No Internal FIFO: Linked-List Mode
Channel 2 - No Internal FIFO: Linked-List Mode
Channel 3 - No Internal FIFO: Disable

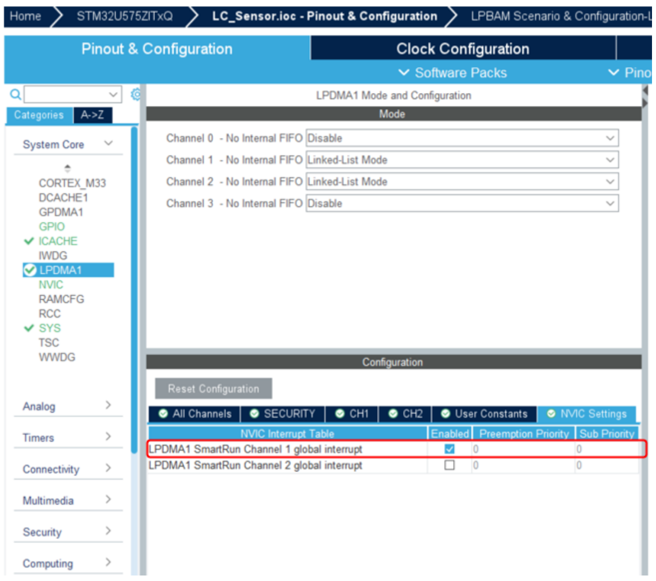
Configure the below parameters :

Channel Configuration for Linked List

Priority	Low
Execution Mode (circular/linear) of the Linked List	Circular
Linked List Execution Mode	The List is fully executed
Transfer Event Generation	The TC (and the HT) event is generated at the (respectively hal...

2. Enable LPDMA global interrupt for Channel1 as shown in the figure below.

Figure 47. Enable LPDMA global interrupt for CH1

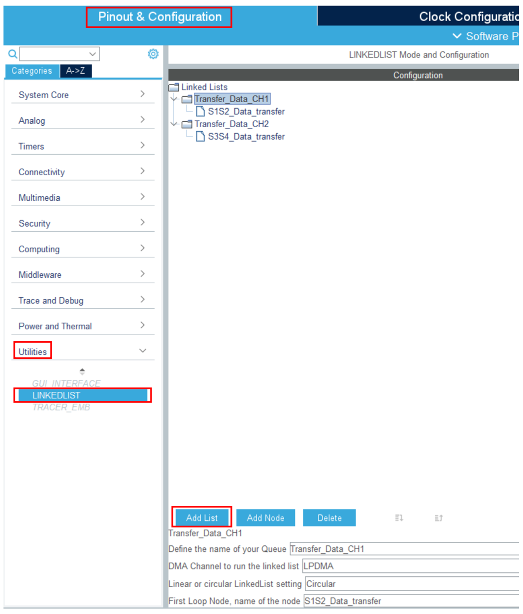


Channel 0 - No Internal FIFO: Disable
Channel 1 - No Internal FIFO: Linked-List Mode
Channel 2 - No Internal FIFO: Linked-List Mode
Channel 3 - No Internal FIFO: Disable

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
LPDMA1 SmartRun Channel 1 global interrupt	✓	0	0
LPDMA1 SmartRun Channel 2 global interrupt	☐	0	0

3. Create and configure a linked-list for each channel. Go to *Pinout & Configuration*, then *Utilities*, and select *LINKEDLIST*. Click on *Add List*. Now, define the name of the list and select *LPDMA* to run the linked-list and configure it in circular mode.
   Configure the linked-list for CH1 named *Transfer_Data_CH1*. Then add a new list for CH2 named *Transfer_Data_CH2* in the same way as CH1.

Figure 48. Linked-list configuration

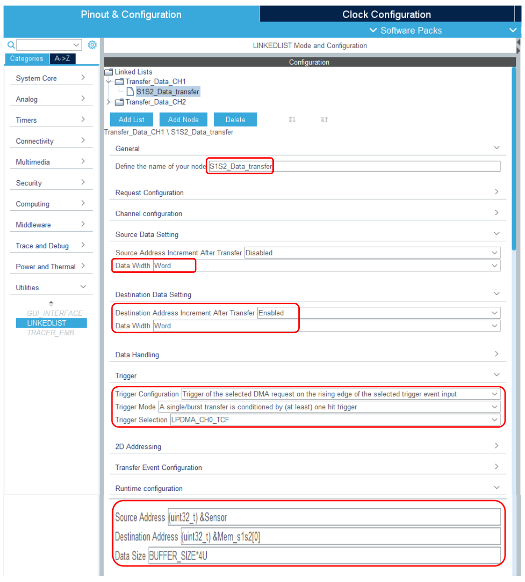


Linked Lists
Transfer_Data_CH1
S1S2_Data_transfer
Transfer_Data_CH2
S3S4_Data_transfer

Transfer_Data_CH1
Define the name of your Queue: Transfer_Data_CH1
DMA Channel to run the linked list: LPDMA
Linear or circular LinkedList setting: Circular
First Loop Node, name of the node: S1S2_Data_transfer

4. Define the name of the created node to *S1S2_Data_transfer*, then configure it as shown in Figure 49.
   The *S1S2_Data_transfer* node performs the transfer of the number of oscillations for Sensor1 and Sensor2 at each LC sensor acquisition.

Figure 49. Node configuration for linked-list_CH1



Linked Lists
Transfer_Data_CH1
S1S2_Data_transfer
Transfer_Data_CH2

Transfer_Data_CH1 \ S1S2_Data_transfer

General
Define the name of your node: S1S2_Data_transfer

Request Configuration
Channel configuration

Source Data Setting
Source Address Increment After Transfer: Disabled
Data Width: Word

Destination Data Setting
Destination Address Increment After Transfer: Enabled
Data Width: Word

Data Handling

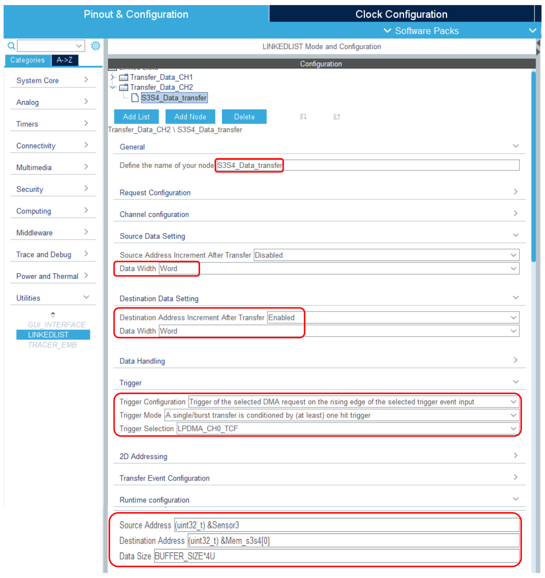
Trigger
Trigger Configuration: Trigger of the selected DMA request on the rising edge of the selected trigger event input
Trigger Mode: A single/burst transfer is conditioned by (at least) one hit trigger
Trigger Selection: LPDMA_CH0_TCF

2D Addressing
Transfer Event Configuration

Runtime configuration
Source Address: (uint32_t) &Sensor
Destination Address: (uint32_t) &Mem_s1s2[0]
Data Size: BUFFER_SIZE*4U

5. Define the name of the created node to *S3S4_Data_transfer*, then configure it as shown in Figure 50. The *S3S4_Data_Transfer* node performs the transfer of the number of oscillations for Sensor3 and Sensor4 at each LC sensor acquisition.

Figure 50. Node configuration for linked-list_CH2



Transfer_Data_CH2 \ S3S4_Data_transfer

General
Define the name of your node S3S4_Data_transfer

Source Data Setting
Source Address Increment After Transfer Disabled
Data Width Word

Destination Data Setting
Destination Address Increment After Transfer Enabled
Data Width Word

Trigger
Trigger Configuration Trigger of the selected DMA request on the rising edge of the selected trigger event input
Trigger Mode A single/burst transfer is conditioned by (at least) one hit trigger
Trigger Selection LPDMA_CH0_TCF

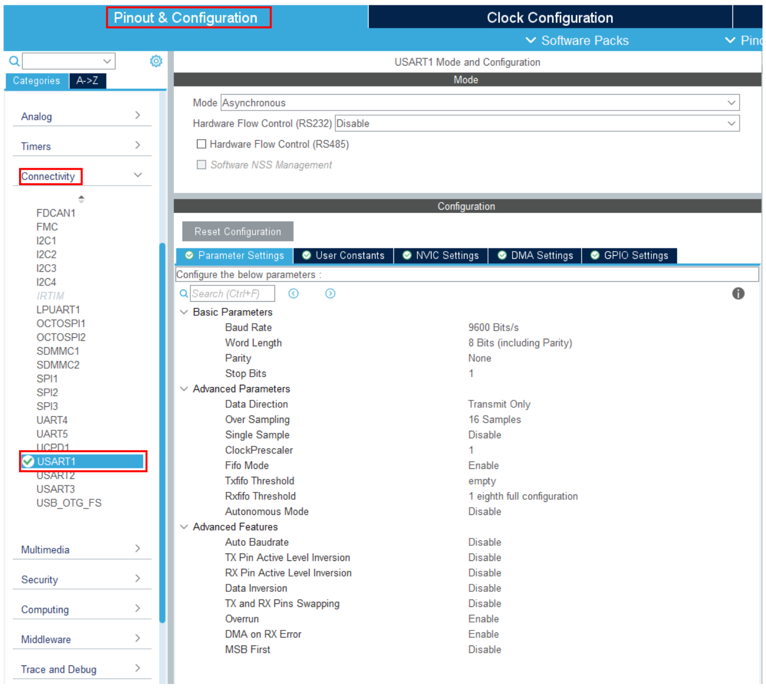
Runtime configuration
Source Address (uint32_t) &Sensor3
Destination Address (uint32_t) &Mem_s3s4[0]
Data Size BUFFER_SIZE*4U

5.2 USART configuration

To configure USART1:

1. Go to the standard view of the STM32CubeMX tool. Go to *Pinout & Configuration* and select *Connectivity*. Click on *USART1* and configure it as shown in Figure 51.

Figure 51. USART1 configuration



Mode Asynchronous
Hardware Flow Control (RS232) Disable
☐ Hardware Flow Control (RS485)
☐ *Software NSS Management*

Basic Parameters	
Baud Rate	9600 Bits/s
Word Length	8 Bits (including Parity)
Parity	None
Stop Bits	1
Advanced Parameters	
Data Direction	Transmit Only
Over Sampling	16 Samples
Single Sample	Disable
ClockPrescaler	1
Fifo Mode	Enable
Txfifo Threshold	empty
Rxfifo Threshold	1 eighth full configuration
Autonomous Mode	Disable
Advanced Features	
Auto Baudrate	Disable
TX Pin Active Level Inversion	Disable
RX Pin Active Level Inversion	Disable
Data Inversion	Disable
TX and RX Pins Swapping	Disable
Overrun	Enable
DMA on RX Error	Enable
MSB First	Disable

Note: *When the system enters Stop 2 mode, the USART2 is automatically deactivated. The configuration is retained in SRAM4 to send data after wakeup from Stop 2 mode.*

Define Baudrate = 9600

5.3 Code generation

At this point, the main and the LPBAM applications are ready to be generated.

Click on *GENERATE CODE*.

5.4 Data processing

When the buffers Mem_s1s2 and Mem_s3s4 are full, the LPDMA_CH1 TC interrupt is generated. The CPU performs the metal state check using the two buffers Mem_s1s2 and Mem_s3s4.

In the static void `TransferComplete(DMA_HandleTypeDef *hdma)` function of the `main.c` file, the CPU:

- Extracts the number of pulses for each sensor from the Mem_s1s2 and Mem_s3s4 buffers.
- Compares the number of pulses with the detection threshold (CountDetect) to define if there is a metal or no metal state.

6 How to call APIs

According to the project needs, the user must call the necessary APIs for each peripheral.

6.1 lpbam_lc_sensor_scenario_config.c file

1. Go to the `lpbam_lc_sensor_scenario_config.c` file, then in `MX_COMP1_Init and MX_COMP2_Init`. Add `__HAL_COMP_ENABLE()` after the COMP1/2 initialization.

```
/* USER CODE BEGIN COMP2_Init 2 */
__HAL_COMP_ENABLE(&hcomp2);
/* USER CODE END COMP2_Init 2 */
```

```
/* USER CODE BEGIN COMP1_Init 2 */
__HAL_COMP_ENABLE(&hcomp1);
/* USER CODE END COMP1_Init 2 */
```

2. Configure DAC_DHR register for CH1 and CH2.
   Go to `MX_DAC1_Init()` function and add `HAL_DAC_SetValue()` after DAC initialization as follows:

```
/* USER CODE BEGIN DAC1_Init 2 */
/* Set DAC VMID DHR register */
if (HAL_DAC_SetValue(&hdac1, DAC_CHANNEL_1, DAC_ALIGN_12B_R, 0x800) != HAL_OK)
{
  /* Setting value Error */
  Error_Handler();
}

/* Set DAC COMP THRESHOLD DHR register */
if (HAL_DAC_SetValue(&hdac1, DAC_CHANNEL_2, DAC_ALIGN_12B_R, 0xC00) != HAL_OK)
{
  /* Setting value Error */
  Error_Handler();
}
/* USER CODE END DAC1_Init 2 */
```

3. Start the LPTIM counter for LPTIM1/4.
   Go to `MX_LPTIM1_Init() and MX_LPTIM4_Init()` functions and add `HAL_LPTIM_Counter_Start()` as follows:

```
/* USER CODE BEGIN LPTIM1_Init 2 */
/* Start counting */
if (HAL_LPTIM_Counter_Start(&hlptim1) != HAL_OK)
{
  Error_Handler();
}
/* USER CODE END LPTIM1_Init 2 *
/* USER CODE BEGIN LPTIM4_Init 2 */
/* Start counting */
if (HAL_LPTIM_Counter_Start(&hlptim4) != HAL_OK)
{
  Error_Handler();
}
/* USER CODE END LPTIM4_Init 2 */
```

6.2 lpbam_lc_sensor_scenario_build.c file

1. Go to `lpbam_lc_sensor_scenario_build.c` file and add the declaration of all buffers used in the LPBAM application.

```
 /* External variables ---------------------------------------------------*/
 /* USER CODE BEGIN EV */
 DMA_NodeTypeDef Start_DAC_Node;
 DMA_NodeTypeDef Stop_DAC_Node;
 DMA_NodeTypeDef STOP_COMP1_Node;
 DMA_NodeTypeDef STOP_COMP2_Node;
 DMA_NodeTypeDef LPTIM4_ReadCNT_S1;
 DMA_NodeTypeDef LPTIM4_ReadCNT_S2;
 DMA_NodeTypeDef LPTIM1_ReadCNT_S3;
 DMA_NodeTypeDef LPTIM1_ReadCNT_S4;
uint32_t Enable_dac      = DAC_ON;
uint32_t Disable_dac     = DAC_OFF;
uint32_t Disable_comp    = COMP_OFF;
uint32_t Data_size       = SIZE;
 #if defined ( __ICCARM__ )
 #pragma location=SRAM4_ADRESS_STORE_S1
 uint16_t Sensor;
 #pragma location=SRAM4_ADRESS_STORE_S2
 uint16_t Sensor2;
 #pragma location=SRAM4_ADRESS_STORE_S3
 uint16_t Sensor3;
 #pragma location=SRAM4_ADRESS_STORE_S4
 uint16_t Sensor4;
 #endif
 #if defined ( __ARMCC_VERSION )||( __GNUC__ )
 uint16_t  Sensor __attribute__((section(".Sensor")));
 uint16_t  Sensor2 __attribute__((section(".Sensor2")));
 uint16_t  Sensor3 __attribute__((section(".Sensor3")));
 uint16_t  Sensor4 __attribute__((section(".Sensor4")));
 #endif
/* USER CODE END EV */

/* USER CODE BEGIN PFP */
HAL_StatusTypeDef MX_LC_sensor_app_Q_Config(void);
HAL_StatusTypeDef MX_Transfer_Data_CH1_Config(void);
HAL_StatusTypeDef MX_Transfer_Data_CH2_Config(void);
/* USER CODE END PFP */
```

2. In `void MX_LC_sensor_Scenario_Build(void)` function, call functions:

```
  /* USER CODE BEGIN LC_sensor_Scenario_Build 1 */
  MX_LC_sensor_app_Q_Config();
  MX_Transfer_Data_CH1_Config();
  MX_Transfer_Data_CH2_Config();
  /* USER CODE END LC_sensor_Scenario_Build 1 */
```

3. In the *USER CODE* section, add this function to build manually the missing features in the STM32CubeMX with LPBAM interface.

```
  /* USER CODE BEGIN LC_sensor_Scenario_Build */
  HAL_StatusTypeDef MX_LC_sensor_app_Q_Config(void)
  {
  HAL_StatusTypeDef ret = HAL_OK;
  /* DMA node configuration declaration */
  DMA_NodeConfTypeDef pNodeConfig;

  /* Set node configuration ################################################*/
  pNodeConfig.NodeType = DMA_LPDMA_LINEAR_NODE;
  pNodeConfig.Init.Request = DMA_REQUEST_SW;
  pNodeConfig.Init.BlkHWRequest = DMA_BREQ_SINGLE_BURST;
  pNodeConfig.Init.Direction = DMA_MEMORY_TO_MEMORY;
  pNodeConfig.Init.SrcInc = DMA_SINC_FIXED;
  pNodeConfig.Init.DestInc = DMA_DINC_FIXED;
  pNodeConfig.Init.SrcDataWidth = DMA_SRC_DATAWIDTH_WORD;
  pNodeConfig.Init.DestDataWidth = DMA_DEST_DATAWIDTH_WORD;
  pNodeConfig.Init.TransferEventMode = DMA_TCEM_LAST_LL_ITEM_TRANSFER;
  pNodeConfig.TriggerConfig.TriggerMode = DMA_TRIGM_BLOCK_TRANSFER;
  pNodeConfig.TriggerConfig.TriggerPolarity = DMA_TRIG_POLARITY_FALLING;
  pNodeConfig.TriggerConfig.TriggerSelection = LPDMA1_TRIGGER_LPTIM3_CH1;
  pNodeConfig.DataHandlingConfig.DataAlignment = DMA_DATA_RIGHTALIGN_ZEROPADDED;
  pNodeConfig.SrcAddress = (uint32_t) &Enable_dac;
  pNodeConfig.DstAddress = (uint32_t) &DAC1->CR;
  pNodeConfig.DataSize = Data_size * 4U;

  /* Build Start_DAC_Node Node */
  ret |= HAL_DMAEx_List_BuildNode(&pNodeConfig, &Start_DAC_Node);

  /* Insert Start_DAC_Node to Queue */
ret |=
HAL_DMAEx_List_InsertNode(&LC_sensor_app_Q,&(LC_sensor_app_Q_Conversion_Config_1_Desc.pNodes[1U]),
&Start_DAC_Node);

  /* Set node configuration ################################################*/
  pNodeConfig.SrcAddress = (uint32_t) &Disable_dac;

  /* Build Stop_DAC_Node Node */
  ret |= HAL_DMAEx_List_BuildNode(&pNodeConfig, &Stop_DAC_Node);

  /* Insert Stop_DAC_Node to Queue */
ret |= HAL_DMAEx_List_InsertNode(&LC_sensor_app_Q,
&(LC_sensor_app_Q_Write_Pin_8_Desc.pNodes[0U]),&Stop_DAC_Node);

  /* Set node configuration ################################################*/
  pNodeConfig.TriggerConfig.TriggerPolarity = DMA_TRIG_POLARITY_MASKED;
  pNodeConfig.SrcAddress = (uint32_t) &Disable_comp;
  pNodeConfig.DstAddress = (uint32_t) &COMP1->CSR;

  /* Build STOP_COMP1_Node Node */
  ret |= HAL_DMAEx_List_BuildNode(&pNodeConfig, &STOP_COMP1_Node);

  /* Insert STOP_COMP1_Node to Queue */
  ret |= HAL_DMAEx_List_InsertNode_Tail(&LC_sensor_app_Q, &STOP_COMP1_Node);

  /* Set node configuration ################################################*/
  pNodeConfig.Init.TransferEventMode = DMA_TCEM_BLOCK_TRANSFER;
  pNodeConfig.DstAddress = (uint32_t) &COMP2->CSR;

  /* Build STOP_COMP2_Node Node */
  ret |= HAL_DMAEx_List_BuildNode(&pNodeConfig, &STOP_COMP2_Node);

  /* Insert STOP_COMP2_Node to Queue */
  ret |= HAL_DMAEx_List_InsertNode_Tail(&LC_sensor_app_Q, &STOP_COMP2_Node);

  pNodeConfig.Init.TransferEventMode = DMA_TCEM_LAST_LL_ITEM_TRANSFER;
  pNodeConfig.Init.SrcDataWidth = DMA_SRC_DATAWIDTH_HALFWORD;
  pNodeConfig.Init.DestDataWidth = DMA_DEST_DATAWIDTH_HALFWORD;
  pNodeConfig.SrcAddress = (uint32_t) &LPTIM4->CNT;
  pNodeConfig.DstAddress = (uint32_t) &Sensor;
  pNodeConfig.DataSize = Data_size * 2U;

  /* Build LPTIM4_ReadCNT_S1 Node */
  ret |= HAL_DMAEx_List_BuildNode(&pNodeConfig, &LPTIM4_ReadCNT_S1);

  /* Insert LPTIM4_ReadCNT_S1 to Queue */
  ret |=
HAL_DMAEx_List_InsertNode(&LC_sensor_app_Q,&(LC_sensor_app_Q_Start_1_Desc.pNodes[0U]),
&LPTIM4_ReadCNT_S1);

  /* Set node configuration ################################################*/
  pNodeConfig.DstAddress = (uint32_t) &Sensor2;

  /* Build LPTIM4_ReadCNT_S2 Node */
  ret |= HAL_DMAEx_List_BuildNode(&pNodeConfig, &LPTIM4_ReadCNT_S2);

  /* Insert LPTIM4_ReadCNT_S2 to Queue */
  ret |=
HAL_DMAEx_List_InsertNode(&LC_sensor_app_Q,&(LC_sensor_app_Q_Start_2_Desc.pNodes[0U]),
&LPTIM4_ReadCNT_S2);

  /* Set node configuration ################################################*/
  pNodeConfig.SrcAddress = (uint32_t) &LPTIM1->CNT;
  pNodeConfig.DstAddress = (uint32_t) &Sensor3;

  /* Build LPTIM1_ReadCNT_S3 Node */
  ret |= HAL_DMAEx_List_BuildNode(&pNodeConfig, &LPTIM1_ReadCNT_S3);

  /* Insert LPTIM1_ReadCNT_S3 to Queue */
  ret |=
HAL_DMAEx_List_InsertNode(&LC_sensor_app_Q,&(LC_sensor_app_Q_Start_3_Desc.pNodes[0U]),
&LPTIM1_ReadCNT_S3);

  /* Set node configuration ################################################*/
  pNodeConfig.DstAddress = (uint32_t) &Sensor4;

  /* Build LPTIM1_ReadCNT_S4 Node */
  ret |= HAL_DMAEx_List_BuildNode(&pNodeConfig, &LPTIM1_ReadCNT_S4);

  /* Insert LPTIM1_ReadCNT_S4 to Queue */
  ret |=
HAL_DMAEx_List_InsertNode(&LC_sensor_app_Q,&(LC_sensor_app_Q_Start_4_Desc.pNodes[0U]),
&LPTIM1_ReadCNT_S4);

  return ret;
  }
  /* USER CODE END LC_sensor_Scenario_Build */
```

6.3 linked_list.c file

In the `linked_list.c` file, define the source and destination address for the LPDMA1 CH1 and CH2 transfer.

```
/* Private define -----------------------------------------------------------*/
/* USER CODE BEGIN PD */
#if defined ( __ICCARM__ )
#pragma location=SRAM4_ADRESS_STORE_S1S2
uint32_t Mem_s1s2[BUFFER_SIZE];
#pragma location=SRAM4_ADRESS_STORE_S3S4
uint32_t Mem_s3s4[BUFFER_SIZE];
#endif
#if defined ( __ARMCC_VERSION )||( __GNUC__ )
uint32_t  Mem_s1s2[BUFFER_SIZE] __attribute__((section(".Mem_s1s2")));
uint32_t  Mem_s3s4[BUFFER_SIZE] __attribute__((section(".Mem_s3s4")));
#endif
extern uint16_t Sensor;
extern uint16_t Sensor3;
/* USER CODE END PD */
```

6.4 Main.c/h files

1. Go to the `main.h` file and add these declarations in the USER CODE section named *USER CODE BEGIN Private defines* and *USER CODE BEGIN EFP*:

```
/* Exported functions prototypes ---------------------------------------------*/
/* USER CODE BEGIN EFP */
void LC_Calibration_IT(void);
/* USER CODE END EFP */

/* Private defines -----------------------------------------------------------*/
/* USER CODE BEGIN Private defines */
#define LPTIM_MAX_PERIOD 0xFFFF
#define SRAM4_ADRESS_STORE_S1 0x28000000
#define SRAM4_ADRESS_STORE_S2 0x28000002
#define SRAM4_ADRESS_STORE_S3 0x28000004
#define SRAM4_ADRESS_STORE_S4 0x28000006
#define SRAM4_ADRESS_STORE_S4 0x28000006
#define SRAM4_ADRESS_STORE_S1S2 0x28001200
#define SRAM4_ADRESS_STORE_S3S4 0x28002900

#define SRAM2_ADRESS_STORE_S1 0x20030000
#define SRAM2_ADRESS_STORE_S2 0x20031700
#define SRAM2_ADRESS_STORE_S3 0x20032E00
#define SRAM2_ADRESS_STORE_S4 0x20035000
#define DAC_ON      (uint32_t)0x00010001
#define DAC_OFF     (uint32_t)0x00000000
#define COMP_OFF    (uint32_t)0x000C0000
#define SIZE (uint32_t)1U
#define BUFFER_SIZE 0x65
#define TXBUFFERSIZE(__BUFFER__) (sizeof(__BUFFER__) / sizeof(*(__BUFFER__)) - 1)
/* USER CODE END Private defines */
```

2. In the `main.c` file, include the necessary `.h` files and add the private define as follows:

```
/* USER CODE BEGIN Includes */
#include "lpbam_lc_sensor.h"
#include "linked_list.h"
#include "LC_Calibration.h"
#include "string.h"
#include "stdio.h"
/* USER CODE END Includes */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
#define UART_TIMEOUT 0xFFFF
#define METAL 1
#define NO_METAL 0
#define USART_ENABLE 1
/*#define DEBUG_CONFIGURATION*/
/* USER CODE END PD */
```

3. Define the private variable used in the `main.c` file as follows:

```
/* USER CODE BEGIN PV */
extern DMA_QListTypeDef Transfer_Data_CH1;
extern DMA_QListTypeDef Transfer_Data_CH2;
extern uint32_t Mem_s1s2[BUFFER_SIZE]  ;
extern uint32_t Mem_s3s4[BUFFER_SIZE]  ;
/* Buffers used to Store LPTIM pulses each period until SRAM4 is satureted */
uint8_t aMsgBuffer_tx[30];/* Buffer used to send message by USART2 */
#if defined ( __ICCARM__ )
#pragma location = SRAM2_ADRESS_STORE_S1
uint32_t Sensor1_Pulses[BUFFER_SIZE-1];
#pragma location=SRAM2_ADRESS_STORE_S2
uint32_t Sensor2_Pulses[BUFFER_SIZE-1];
#pragma location=SRAM2_ADRESS_STORE_S3
uint32_t Sensor3_Pulses[BUFFER_SIZE-1];
#pragma location=SRAM2_ADRESS_STORE_S4
uint32_t Sensor4_Pulses[BUFFER_SIZE-1];
#endif
#if defined ( __ARMCC_VERSION )||( __GNUC__ )
uint32_t  Sensor1_Pulses[BUFFER_SIZE-1] __attribute__((section(".Sensor1_Pulses")));
uint32_t  Sensor2_Pulses[BUFFER_SIZE-1] __attribute__((section(".Sensor2_Pulses")));
uint32_t  Sensor3_Pulses[BUFFER_SIZE-1] __attribute__((section(".Sensor3_Pulses")));
uint32_t  Sensor4_Pulses[BUFFER_SIZE-1] __attribute__((section(".Sensor4_Pulses")));
#endif
/* USER CODE END PV */
```

4. Define the private function prototypes.

```
/* USER CODE BEGIN PFP */
static void Enter_Stop2_Mode(void);
static void TransferComplete(DMA_HandleTypeDef *hdma);
void LC_Calibration_IT(void);
/* USER CODE END PFP */
```

Add the following function in `static void MX_LPDMA1_Init(void)` function to configure LPDMA the register callback.

```
  /* USER CODE BEGIN LPDMA1_Init 2 */
 HAL_DMA_RegisterCallback(&handle_LPDMA1_Channel1, HAL_DMA_XFER_CPLT_CB_ID,
TransferComplete);
  /* USER CODE END LPDMA1_Init 2 */
```

In the *USER CODE BEGIN Init* section of the `main()` function, in the `main.c` file, add the following function to enable the backup access and reset the RTC peripheral:

```
  /* USER CODE BEGIN Init */
  /* Reset the RTC peripheral and the RTC clock source selection */
  HAL_PWR_EnableBkUpAccess();
  __HAL_RCC_BACKUPRESET_FORCE();
  __HAL_RCC_BACKUPRESET_RELEASE();
  /* USER CODE END Init */
```

In the *USER CODE BEGIN 4* section, add the following function to enter system to Stop 2 mode.

```
  /* USER CODE BEGIN 4 */
  /**
  * @brief Enter Stop2 mode and checks whether the system was in Stop2 or not.
  * @param None
  * @retval None
  */
  static void Enter_Stop2_Mode(void)
  {
  /* Enter the system to STOP2 mode */
  __HAL_RCC_PWR_CLK_ENABLE();
  HAL_PWREx_EnterSTOP2Mode(PWR_STOPENTRY_WFI);

  /* Check that the system was resumed from stop 2 */
  if (__HAL_PWR_GET_FLAG(PWR_FLAG_STOPF) == 0U)
  {
    Error_Handler();
  }
  /* Clear stop flag */
  __HAL_PWR_CLEAR_FLAG(PWR_FLAG_STOPF);
  /* Check that stop flag is cleared */
  if (__HAL_PWR_GET_FLAG(PWR_FLAG_STOPF) != 0U)
  {
    Error_Handler();
  }
  }
```

In the `MX_RTC_Init()` function, in the `main.c` file, add the following function inside *USER CODE BEGIN RTC_Init 2* section to disable the wakeup timer:

```
  /* USER CODE BEGIN RTC_Init 2 */
  if (HAL_RTCEx_DeactivateWakeUpTimer(&hrtc) != HAL_OK)
  {
    /* DesactivateWakeUpTimer Error */
    Error_Handler();
  }
  /* USER CODE END RTC_Init 2 */
```

In the `MX_TIM6_Init()` function, in the `main.c` file, add the following function inside *USER CODE BEGIN TIM6_Init 2* section to enable TIM6 update interrupt:

```
  /* USER CODE BEGIN TIM6_Init 2 */
  /* Enable update interrupts to use them as events for sleep mode*/
    TIM6->DIER |= TIM_IT_UPDATE;

  TIM6->SR = ~(TIM_FLAG_UPDATE);
  NVIC_ClearPendingIRQ(TIM6_IRQn);
  /* USER CODE END TIM6_Init 2 */
```

In the same *USER CODE BEGIN 4* section, add the `TransferComplete()` function that performs the data processing to check the metal state every ten seconds.

```
/**
  * @brief LPDMA TC interrupt to Check Metal state and send it using USART.
  * @param hdma
  * @retval None
  */
static void TransferComplete(DMA_HandleTypeDef *hdma)
{
if (hdma->Instance  == LPDMA1_Channel1)
{
  uint8_t s1_status=0,s2_status=0,s3_status=0,s4_status=0;
  uint16_t *ptr_mem_s1s2=NULL,*ptr_mem_s3s4 =NULL;
  __IO uint32_t mem_indice=0;

  for( mem_indice=1;mem_indice<BUFFER_SIZE-1;mem_indice++)
  {
    ptr_mem_s1s2=(uint16_t*)&Mem_s1s2[mem_indice];
#ifdef SENSOR_1_ENABLED
    /* Compute pulses number For Sensor1 */
    if ((*(ptr_mem_s1s2+1) > *ptr_mem_s1s2))
    {
      Sensor1_Pulses[mem_indice]= *(ptr_mem_s1s2+1) - *ptr_mem_s1s2;
    }
    else
    {
      Sensor1_Pulses[mem_indice]= *(ptr_mem_s1s2+1) + (LPTIM_MAX_PERIOD-
*ptr_mem_s1s2);
    }
#endif /* SENSOR_1_ENABLED */

#ifdef SENSOR_2_ENABLED
    /* Compute pulses number For Sensor2 */
    if (*(ptr_mem_s1s2+2) > *(ptr_mem_s1s2+1))
    {
      Sensor2_Pulses[mem_indice]= (*(ptr_mem_s1s2+2) - *(ptr_mem_s1s2+1))-1;
    }
    else
    {
      Sensor2_Pulses[mem_indice]= (*(ptr_mem_s1s2+2) + (LPTIM_MAX_PERIOD-
*(ptr_mem_s1s2+1)))-1;
    }
#endif /* SENSOR_2_ENABLED */

    ptr_mem_s3s4=(uint16_t*)&Mem_s3s4[mem_indice];
#ifdef SENSOR_3_ENABLED
    /* Compute pulses number For Sensor3 */
    if ((*(ptr_mem_s3s4+1) > *ptr_mem_s3s4))
    {
      Sensor3_Pulses[mem_indice]= *(ptr_mem_s3s4+1) - *ptr_mem_s3s4;
    }
    else
    {
      Sensor3_Pulses[mem_indice]= *(ptr_mem_s3s4+1) + (LPTIM_MAX_PERIOD-
*ptr_mem_s3s4);
    }
#endif /* SENSOR_3_ENABLED */

#ifdef SENSOR_4_ENABLED
   /* Compute pulses number For Sensor4 */
    if (*(ptr_mem_s3s4+2) > *(ptr_mem_s3s4+1))
    {
      Sensor4_Pulses[mem_indice]= (*(ptr_mem_s3s4+2) - *(ptr_mem_s3s4+1))-1;
    }
    else
    {
      Sensor4_Pulses[mem_indice]= (*(ptr_mem_s3s4+2) + (LPTIM_MAX_PERIOD-
*(ptr_mem_s3s4+1)))-1;
    }
  }
#endif /* SENSOR_4_ENABLED */
  mem_indice=0;
  for( mem_indice=1;mem_indice<BUFFER_SIZE-2;mem_indice++)
  {
#ifdef SENSOR_1_ENABLED
    if (Sensor1_Pulses[mem_indice] <LcSensor1.CountDetect)
    {
      /* If oscillations are damped more than Count_detect then update Sensor 1 status
*/
      s1_status = METAL;   /* METAL */
    }
    else
    {
      s1_status = NO_METAL;  /*NO_METAL */
    }
#endif /* SENSOR_1_ENABLED */
#ifdef SENSOR_2_ENABLED
    if (Sensor2_Pulses[mem_indice] <LcSensor2.CountDetect)
    {
      /* If oscillations are damped more than Count_detect then update Sensor 2 status
*/
      s2_status = METAL;  /* METAL */
    }
    else
    {
      s2_status = NO_METAL;  /*NO_METAL */
    }
#endif /* SENSOR_2_ENABLED */
#ifdef SENSOR_3_ENABLED
    if (Sensor3_Pulses[mem_indice] <LcSensor3.CountDetect)
    {
      /* If oscillations are damped more than Count_detect then update Sensor 3 status
*/
      s3_status = METAL;  /* METAL */
    }
    else
    {
      s3_status = NO_METAL;  /*NO_METAL */
    }
#endif /* SENSOR_3_ENABLED */
#ifdef SENSOR_4_ENABLED
    if (Sensor4_Pulses[mem_indice] < LcSensor4.CountDetect)
    {
      /* If oscillations are damped more than Count_detect then update Sensor 4 status
*/
      s4_status = METAL;  /* METAL */
    }
    else
    {
      s4_status = NO_METAL;  /*NO_METAL */
    }
#endif /* SENSOR_4_ENABLED */
#if defined(SENSOR_1_ENABLED) || defined(SENSOR_2_ENABLED) ||
defined(SENSOR_3_ENABLED) || defined(SENSOR_4_ENABLED)
    if ((s1_status != NO_METAL)||(s2_status != NO_METAL) || (s3_status != NO_METAL) ||
(s4_status != NO_METAL))
    {
      sprintf((char*)aMsgBuffer_tx, "S1:%d\tS2:%d\tS3:%d\tS4:%d\r\n", s1_status,
s2_status,s3_status,s4_status);
#if USART_ENABLE
      /* USART2 Transmit */
      HAL_UART_Transmit(&huart1, (uint8_t *)aMsgBuffer_tx,
TXBUFFERSIZE(aMsgBuffer_tx), UART_TIMEOUT );
#endif
    }
#endif
  }
  /* Set Enable bit for COMP1/2 before entering STOP2 to generate COMP_OUT*/
  SET_BIT(COMP1->CSR, COMP_CSR_EN);
  SET_BIT(COMP2->CSR, COMP_CSR_EN);
 }
}
```

Then, in the same *USER CODE BEGIN 4* section, add the `LC_Calibration_IT()` function that performs the LC sensor Calibration interrupt subroutine.

```
/**
  * @brief  LC Calibration interrupt function.
  * @param  hrtc None
  * @retval None;
  */
void LC_Calibration_IT(void)
{
  /* Clear SLEEPDEEP bit of Cortex System Control Register */
  CLEAR_BIT(SCB->SCR, ((uint32_t)SCB_SCR_SLEEPDEEP_Msk));

  /* Refresh DAC1 reference power supply from Analog state to Output  and Refresh DAC1
to feed COMP threshold*/
  SET_BIT(DAC1->CR, DAC_CR_LC_EN_VMID_EN_VCMP);

  /* Enable the selected comparator */
#if defined(SENSOR_1_ENABLED) || defined(SENSOR_2_ENABLED)
  SET_BIT(COMP1->CSR, COMP_CSR_EN);
#endif /* defined(SENSOR_1_ENABLED) || defined(SENSOR_2_ENABLED) */

#if defined(SENSOR_3_ENABLED) || defined(SENSOR_4_ENABLED)
  SET_BIT(COMP2->CSR, COMP_CSR_EN);
#endif /* defined(SENSOR_3_ENABLED) || defined(SENSOR_4_ENABLED) */

  /* Waiting time for DAC VMID sampling: Start TIM6 counter in single-shot, decrease
HCLK during sleep and go back to full speed after TIM6 event*/
  if (LcConfig.DacVmidRefreshTime > 12)
  {
    LC_WAIT(LcConfig.DacVmidRefreshTime);
  }

#ifdef SENSOR_1_ENABLED

  /* Refresh DAC1 channel1 reference power supply from Analog state to Output  and
Refresh DAC1 channel2 to feed COMP threshold*/
  SET_BIT(DAC1->CR, DAC_CR_LC_EN_VMID_EN_VCMP);

  /* COMP_NONINVERTINGINPUT_IO1 selected */
  MODIFY_REG(COMP1->CSR, COMP_CSR_INPSEL,COMP_CSR_INPSEL_0);

  /* LPTIM1 counter */
   LcSensor1.CountTmp1 = LPTIM4->CNT;

  /* Set and Reset LPGPIO0 and start oscillations on sensor 1  */
  LC_EXCIT(GPIO_PIN_0);

  /* Start TIM6 counter in single-shot, disable DAC, decrease HCLK during sleep and go
back to full speed after TIM6 event*/
  LC_MEASURE(LcConfigSensor1.Tcapture);

  /* LPTIM1 counter */
  LcSensor1.CountTmp2 = LPTIM4->CNT;

#if defined(SENSOR_2_ENABLED) || defined(SENSOR_3_ENABLED) || defined(SENSOR_4_ENABLED)
  /* Waiting time between 2 LC measurements: Start TIM6 counter in single-shot,
decrease HCLK during sleep and go back to full speed after TIM6 event*/
```

```
  LC_WAIT(LcConfig.TimeBetweenSensor);
#endif /* defined(SENSOR_2_ENABLED) || defined(SENSOR_3_ENABLED) ||
defined(SENSOR_4_ENABLED) */
#endif /* SENSOR_1_ENABLED */

#ifdef SENSOR_2_ENABLED
   /* Refresh DAC1 channel1 reference power supply from Analog state to Output  and
Refresh DAC1 channel2 to feed COMP threshold*/

  /* COMP_NONINVERTINGINPUT_IO2 selected */
  MODIFY_REG(COMP1->CSR, COMP_CSR_INPSEL,COMP_CSR_INPSEL_1);

  /* LPTIM1 counter */
  LcSensor2.CountTmp1 = LPTIM4->CNT;

  /* Set and Reset LPGPIO1 and start oscillations on sensor 2  */
  LC_EXCIT(GPIO_PIN_1);

  /* Start TIM6 counter in single-shot, disable DAC, decrease HCLK during sleep and go
back to full speed after TIM6 event*/
  LC_MEASURE(LcConfigSensor2.Tcapture);

  /* LPTIM1 counter */
  LcSensor2.CountTmp2 = LPTIM4->CNT;

#if defined(SENSOR_3_ENABLED) || defined(SENSOR_4_ENABLED)
  /* Waiting time between 2 LC measurements: Start TIM6 counter in single-shot,
decrease HCLK during sleep and go back to full speed after TIM6 event*/
  LC_WAIT(LcConfig.TimeBetweenSensor);
#endif /* defined(SENSOR_3_ENABLED) || defined(SENSOR_4_ENABLED) */
#endif /* SENSOR_2_ENABLED */

#ifdef SENSOR_3_ENABLED

  /* COMP_NONINVERTINGINPUT_IO1 selected */
  CLEAR_BIT(COMP2->CSR, COMP_CSR_INPSEL);

  /* DAC COMP threshold can be stopped very soon (internal S/H) */
  CLEAR_BIT(DAC1->CR, DAC_CR_LC_EN_VCMP);

  /* LPTIM1 counter */
  LcSensor3.CountTmp1 = LPTIM1->CNT;

  /* Set and Reset LPGPIO2 and start oscillations on sensor 3  */
  LC_EXCIT(GPIO_PIN_2);

  /* Start TIM6 counter in single-shot, disable DAC, decrease HCLK during sleep and go
back to full speed after TIM6 event*/
  LC_MEASURE(LcConfigSensor3.Tcapture);

  /* LPTIM1 counter */
  LcSensor3.CountTmp2 = LPTIM1->CNT;

#ifdef SENSOR_4_ENABLED
  /* Waiting time between 2 LC measurements: Start TIM6 counter in single-shot,
decrease HCLK during sleep and go back to full speed after TIM6 event*/
  LC_WAIT(LcConfig.TimeBetweenSensor);
#endif /* SENSOR_4_ENABLED */
#endif /* SENSOR_3_ENABLED */

#ifdef SENSOR_4_ENABLED

  /* COMP_NONINVERTINGINPUT_IO2 selected */
  SET_BIT(COMP2->CSR, COMP_CSR_INPSEL_0);

  /* LPTIM1 counter */
  LcSensor4.CountTmp1 = LPTIM1->CNT;

  /* Set and Reset and Reset LPGPIO3 and start oscillations on sensor 4  */
  LC_EXCIT(GPIO_PIN_3);
  /* Start TIM6 counter in single-shot, disable DAC, decrease HCLK during sleep and go
back to full speed after TIM6 event*/
  LC_MEASURE(LcConfigSensor4.Tcapture);

  /* LPTIM1 counter */
  LcSensor4.CountTmp2 = LPTIM1->CNT;
#endif /* SENSOR_4_ENABLED */

  /* Power down the DAC1 (Holding capacitor on Vmid pin)*/
  CLEAR_BIT(DAC1->CR, DAC_CR_LC_EN_VMID_EN_VCMP);

  /* Power down the comparator */
#if defined(SENSOR_1_ENABLED) || defined(SENSOR_2_ENABLED)
  CLEAR_BIT(COMP1->CSR, COMP_CSR_EN);
#endif /* defined(SENSOR_1_ENABLED) || defined(SENSOR_2_ENABLED) */
#if defined(SENSOR_3_ENABLED) || defined(SENSOR_4_ENABLED)
  CLEAR_BIT(COMP2->CSR, COMP_CSR_EN);
#endif /* defined(SENSOR_3_ENABLED) || defined(SENSOR_4_ENABLED) */

  LcStatus.MeasuresCount++;
    /* Compute pulses number */
  if (LcSensor1.CountTmp1 <= LcSensor1.CountTmp2)
  {
    LcSensor1.CountValue = LcSensor1.CountTmp2 - LcSensor1.CountTmp1;
  }
  else
  {
    LcSensor1.CountValue = LcSensor1.CountTmp2 + (LPTIM_MAX_PERIOD -
LcSensor1.CountTmp1);
  }

  /* Periodic calibration if required */
  if (LcStatus.CalStatus.State == CAL_ON_GOING)
  {
    if (LcSensor1.CountValue > LcStatus.CalStatus.Sensor1CountMax)
    {
      LcStatus.CalStatus.Sensor1CountMax = LcSensor1.CountValue;
    }
    if (LcSensor1.CountValue < LcStatus.CalStatus.Sensor1CountMin)
    {
      LcStatus.CalStatus.Sensor1CountMin = LcSensor1.CountValue;
    }
  }
    /* Compute pulses number */
  if (LcSensor2.CountTmp1 <= LcSensor2.CountTmp2)
  {
    LcSensor2.CountValue = LcSensor2.CountTmp2 - LcSensor2.CountTmp1;
  }
  else
  {
    LcSensor2.CountValue = LcSensor2.CountTmp2 + (LPTIM_MAX_PERIOD -
LcSensor2.CountTmp1);
  }

  /* Periodic calibration if required */
  if (LcStatus.CalStatus.State == CAL_ON_GOING)
  {
    if (LcSensor2.CountValue > LcStatus.CalStatus.Sensor2CountMax)
    {
      LcStatus.CalStatus.Sensor2CountMax = LcSensor2.CountValue;
    }
    if (LcSensor2.CountValue < LcStatus.CalStatus.Sensor2CountMin)
    {
      LcStatus.CalStatus.Sensor2CountMin = LcSensor2.CountValue;
    }
  }
  /* Compute pulses number */
  if (LcSensor3.CountTmp1 <= LcSensor3.CountTmp2)
  {
    LcSensor3.CountValue = LcSensor3.CountTmp2 - LcSensor3.CountTmp1;
  }
  else
  {
    LcSensor3.CountValue = LcSensor3.CountTmp2 + (LPTIM_MAX_PERIOD - 
LcSensor3.CountTmp1);
  }
  /* Periodic calibration if required */
  if (LcStatus.CalStatus.State == CAL_ON_GOING)
  {
    if (LcSensor3.CountValue > LcStatus.CalStatus.Sensor3CountMax)
    {
      LcStatus.CalStatus.Sensor3CountMax = LcSensor3.CountValue;
    }
    if (LcSensor3.CountValue < LcStatus.CalStatus.Sensor3CountMin)
    {
      LcStatus.CalStatus.Sensor3CountMin = LcSensor3.CountValue;
    }
  }

  /* Compute pulses number */
  if (LcSensor4.CountTmp1 <= LcSensor4.CountTmp2)
  {
    LcSensor4.CountValue = LcSensor4.CountTmp2 - LcSensor4.CountTmp1;
  }
  else
  {
    LcSensor4.CountValue = LcSensor4.CountTmp2 + (LPTIM_MAX_PERIOD - 
LcSensor4.CountTmp1);
  }

  /* Periodic calibration if required */
  if (LcStatus.CalStatus.State == CAL_ON_GOING)
  {
    if (LcSensor4.CountValue > LcStatus.CalStatus.Sensor4CountMax)
    {
      LcStatus.CalStatus.Sensor4CountMax = LcSensor4.CountValue;
    }
    if (LcSensor4.CountValue < LcStatus.CalStatus.Sensor4CountMin)
    {
      LcStatus.CalStatus.Sensor4CountMin = LcSensor4.CountValue;
    }
  }
  /* Set SLEEPDEEP bit of Cortex System Control Register */
  SET_BIT(SCB->SCR, ((uint32_t)SCB_SCR_SLEEPDEEP_Msk));
}
/* USER CODE END 4 */
```

Now, go to the `main()` function and call LPBAM generated APIs to initialize, build, link, and start the LPBAM application inside the user section named *USER CODE BEGIN 2*.
The call sequence of LPBAM application generated APIs is unique. It is mandatory to follow it to make any LPBAM application running.

```
  /* USER CODE BEGIN 2 */
  #ifdef DEBUG_CONFIGURATION
  /* following line Enables debug interface */
  HAL_DBGMCU_EnableDBGStopMode();
  #else
  /* following line disables debug interface */
  HAL_DBGMCU_DisableDBGStopMode();
  #endif
  /* LPBAM LC_sensor application init */
  MX_LC_sensor_Init();

 /* LPBAM LC_sensor application LC_sensor sceanrio init */
  MX_LC_sensor_Scenario_Init();

  /* LC sensor Calibration */
  LcSensorDemo();
```

```
/* LPBAM LC_sensor application LC_sensor sceanrio build */
MX_LC_sensor_Scenario_Build();

/* LPBAM LC_sensor application LC_sensor sceanrio link */
MX_LC_sensor_Scenario_Link(&handle_LPDMA1_Channel0);
HAL_DMAEx_List_LinkQ(&handle_LPDMA1_Channel1,&Transfer_Data_CH1);
HAL_DMAEx_List_LinkQ(&handle_LPDMA1_Channel2,&Transfer_Data_CH2);

/* LPBAM LC_sensor application LC_sensor sceanrio start */
MX_LC_sensor_Scenario_Start(&handle_LPDMA1_Channel0);

HAL_DMAEx_List_Start(&handle_LPDMA1_Channel1);
__HAL_DMA_ENABLE_IT(&handle_LPDMA1_Channel1,DMA_IT_TC );
HAL_DMAEx_List_Start(&handle_LPDMA1_Channel2);
/* USER CODE END 2 */
```

At this step, the LPBAM application is operating. Then the lowest power mode must be entered to ensure the functional aspects. For this application and device, it is Stop 2 low-power mode.

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
  /* USER CODE END WHILE */
  /* USER CODE BEGIN 3 */
 Enter_Stop2_Mode();
}
/* USER CODE END 3
```

6.5 STM32U5xx_it.c file

Go to the `RTC_IRQHandler()` function in the `STM32U5xx_it.c` file, and call the `LC_Calibration_IT()` function in the user section named *USER CODE BEGIN RTC_IRQn 1*.

```
/* USER CODE END RTC_IRQn 0 */
  /* USER CODE BEGIN RTC_IRQn 1 */
  LC_Calibration_IT();
/* USER CODE END RTC_IRQn 1 */
```

Note: Add the `LC_Calibration.c/h` files (which are located in the X-Cube-LCSENSOR project) under User/Core folder in the workspace of the project.

At this point, the project implementation is finalized. The user needs to build and run the project with zero error.

For this application, the LPDMA channel used can access only SRAM4 in the same power domain. Adjust linker ROM and RAM section addresses according to the chosen toolchain (EWARM, MDK-ARM, or STM32CubeIDE) as following:

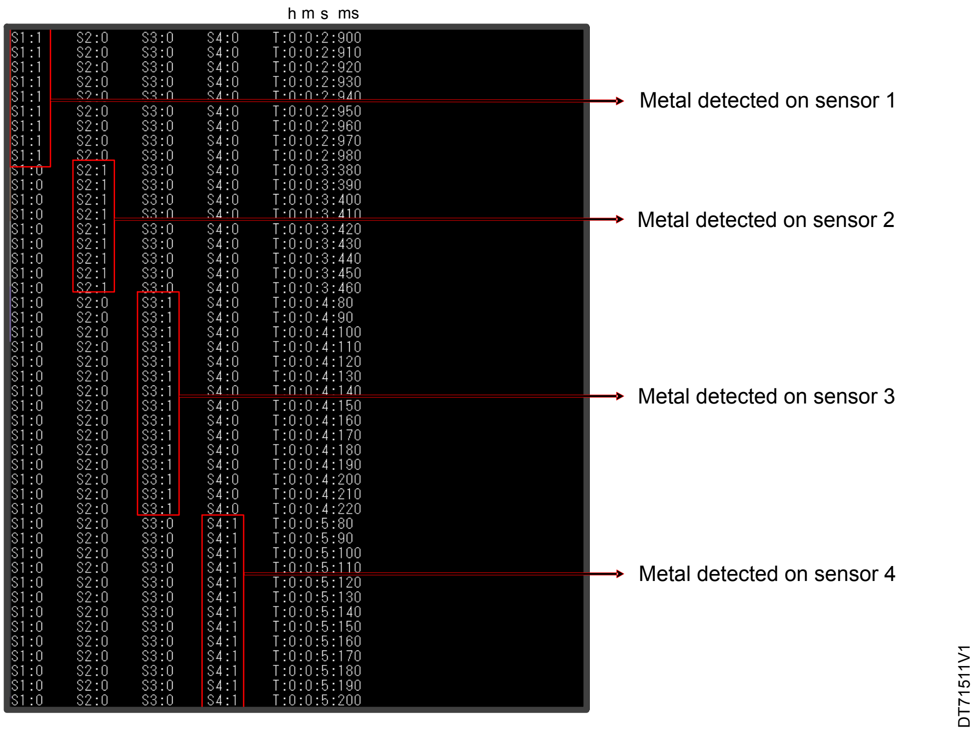
SRAM4
RAM_start = 0x28000000
RAM_end = 0x28003FFF
FLASH bank 1
ROM_start = 0x08000000
ROM_end = 0x0801FFFF

7 LC sensor metering counting demonstration with four sensors

To enable this demonstration, *LC_SENSOR_DEMO 4* define must be selected in `LC_Calibration.h` file.

Counter information are sent on USART1 port when there is a metal near to the sensor and can be displayed with UART terminal software on the host PC connected to the USB cable.

Figure 52. Demonstration with four sensors


h m s ms
Metal detected on sensor 1
Metal detected on sensor 2
Metal detected on sensor 3
Metal detected on sensor 4
DT71511V1

```
S1:1    S2:0    S3:0    S4:0    T:0:0:2:900
S1:1    S2:0    S3:0    S4:0    T:0:0:2:910
S1:1    S2:0    S3:0    S4:0    T:0:0:2:920
S1:1    S2:0    S3:0    S4:0    T:0:0:2:930
S1:1    S2:0    S3:0    S4:0    T:0:0:2:940
S1:1    S2:0    S3:0    S4:0    T:0:0:2:950
S1:1    S2:0    S3:0    S4:0    T:0:0:2:960
S1:1    S2:0    S3:0    S4:0    T:0:0:2:970
S1:1    S2:0    S3:0    S4:0    T:0:0:2:980
S1:0    S2:1    S3:0    S4:0    T:0:0:3:380
S1:0    S2:1    S3:0    S4:0    T:0:0:3:390
S1:0    S2:1    S3:0    S4:0    T:0:0:3:400
S1:0    S2:1    S3:0    S4:0    T:0:0:3:410
S1:0    S2:1    S3:0    S4:0    T:0:0:3:420
S1:0    S2:1    S3:0    S4:0    T:0:0:3:430
S1:0    S2:1    S3:0    S4:0    T:0:0:3:440
S1:0    S2:1    S3:0    S4:0    T:0:0:3:450
S1:0    S2:1    S3:0    S4:0    T:0:0:3:460
S1:0    S2:0    S3:1    S4:0    T:0:0:4:80
S1:0    S2:0    S3:1    S4:0    T:0:0:4:90
S1:0    S2:0    S3:1    S4:0    T:0:0:4:100
S1:0    S2:0    S3:1    S4:0    T:0:0:4:110
S1:0    S2:0    S3:1    S4:0    T:0:0:4:120
S1:0    S2:0    S3:1    S4:0    T:0:0:4:130
S1:0    S2:0    S3:1    S4:0    T:0:0:4:140
S1:0    S2:0    S3:1    S4:0    T:0:0:4:150
S1:0    S2:0    S3:1    S4:0    T:0:0:4:160
S1:0    S2:0    S3:1    S4:0    T:0:0:4:170
S1:0    S2:0    S3:1    S4:0    T:0:0:4:180
S1:0    S2:0    S3:1    S4:0    T:0:0:4:190
S1:0    S2:0    S3:1    S4:0    T:0:0:4:200
S1:0    S2:0    S3:1    S4:0    T:0:0:4:210
S1:0    S2:0    S3:1    S4:0    T:0:0:4:220
S1:0    S2:0    S3:0    S4:1    T:0:0:5:80
S1:0    S2:0    S3:0    S4:1    T:0:0:5:90
S1:0    S2:0    S3:0    S4:1    T:0:0:5:100
S1:0    S2:0    S3:0    S4:1    T:0:0:5:110
S1:0    S2:0    S3:0    S4:1    T:0:0:5:120
S1:0    S2:0    S3:0    S4:1    T:0:0:5:130
S1:0    S2:0    S3:0    S4:1    T:0:0:5:140
S1:0    S2:0    S3:0    S4:1    T:0:0:5:150
S1:0    S2:0    S3:0    S4:1    T:0:0:5:160
S1:0    S2:0    S3:0    S4:1    T:0:0:5:170
S1:0    S2:0    S3:0    S4:1    T:0:0:5:180
S1:0    S2:0    S3:0    S4:1    T:0:0:5:190
S1:0    S2:0    S3:0    S4:1    T:0:0:5:200
```

Note: Refer to document [1] for more demonstration with one, two, or three sensors.

8 Power consumption

To measure the power consumption of the application, the STM32 PowerShield plug-and-play solution is used to supply the Nucleo board.

Table 5. Four sensors I_{DD} (average) vs. sampling rate (V_{DD} = 3.3 V)

Sampling period	LPBAM: all in Stop 2 mode without metal check	LPBAM: all in Stop 2 mode with buffered data using LPDMA TC interrupt each 10 s	LPBAM: all in Stop 2 mode with RTC wakeup interrupt (with metal state check)	Legacy: LC sensor in Sleep mode, then enter Stop 2 mode
10 Hz	7.4 µA	8 µA	8.2 µA	12 µA
32 Hz	11 µA	11.4 µA	14 µA	23 µA
50 Hz	18 µA	19 µA	21 µA	31 µA
100 Hz	26 µA	27 µA	33 µA	55 µA

Table 6. Three sensors IDD (average) vs. sampling rate (V_{DD} = 3.3 V)

Sampling period	LPBAM: all in Stop 2 mode without metal check	LPBAM: all in Stop 2 mode with buffered data (8 Kbytes) using LPDMA TC interrupt each 10 s	LPBAM: all in Stop 2 mode with RTC wakeup interrupt (with metal state check)
32 Hz	9.7 µA	10.5 µA	12.5 µA

Note: If there is an overconsumption during the current measurement, configure HCLK as AdcDacClockSelection clock instead of MSIK clock in the `MX_DAC1_MspInit()` function of the `lpbam_lc_scensor_scenario_config file`.

Revision history

Table 7. **Document revision history**

Date	Version	Changes
02-Dec-2022	1	Initial release.

Contents

List of tables

List of figures