AN4636 Application note

How to use LC sensors for gas or water metering

Introduction

This application note describes the LC sensor metering feature included in the STM32L073Z-EVAL, STM32L476RG-NUCLEO, and STM32U031R8-NUCLEO evaluation boards.

STM32L073xx, STM32L476xx, and STM32U0xxxx microcontrollers combine ultra-low power and performance, offering a complete set of analog and digital peripherals, 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 inductive reading of the rotation of a mechanical wheel.

The demonstration based on STM32L0 can be executed with a single USB cable (type A to B) connecting the host PC to the detection accessory PCB MB1199, both provided in the evaluation board package.

The demonstration based on STM32L4 can be executed with a single USB cable (type A to B) connecting the host PC to the STM32L476RG-NUCLEO, and a custom LC sensor board.

The demonstration based on STM32U0 can be executed with a single USB cable (type A to B) connecting the host PC to the STM32U031R8-NUCLEO, and a custom LC sensor board.

Firmware for each of the application examples is part of the demonstration software included, respectively, in the STM32CubeL0 and X-CUBE-LCSENSOR firmware packages.

Reference documents

- STM32L073Z-EVAL user manual (UM1878)
- STM32Cube embedded software for STM32L0 Series including HAL drivers, USB, File System, RTOS and Touch sensing (DB2318)
- STM32L476RG-NUCLEO user manual (UM1724)
- STM32Cube embedded software for STM32L4 Series including HAL drivers, USB, File System, RTOS and Touch sensing (DB2602)

These documents are available from STMicroelectronics web site www.st.com.

Contents

List of tables

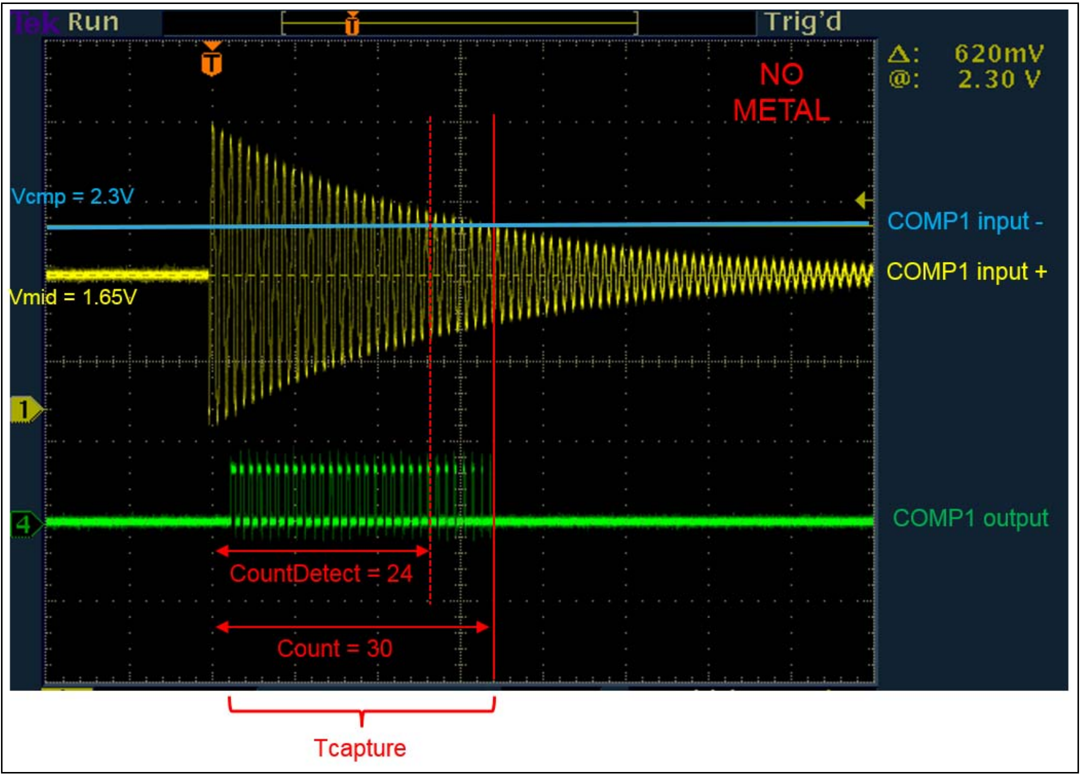
List of figures

1 LC sensor metering principle

LC sensors are driven by an excitation pulse managed by GPIOs (see *Figure 1*). Once the pulse is delivered, the GPIO is reconfigured from output push-pull to alternate function analog input. The LC oscillation has been generated.

The COMP peripheral collects the analog oscillations and compares them to a reference voltage (V_{cmp}), delivering a digital signal on the comparator output. The low power timer peripheral counts the pulses.

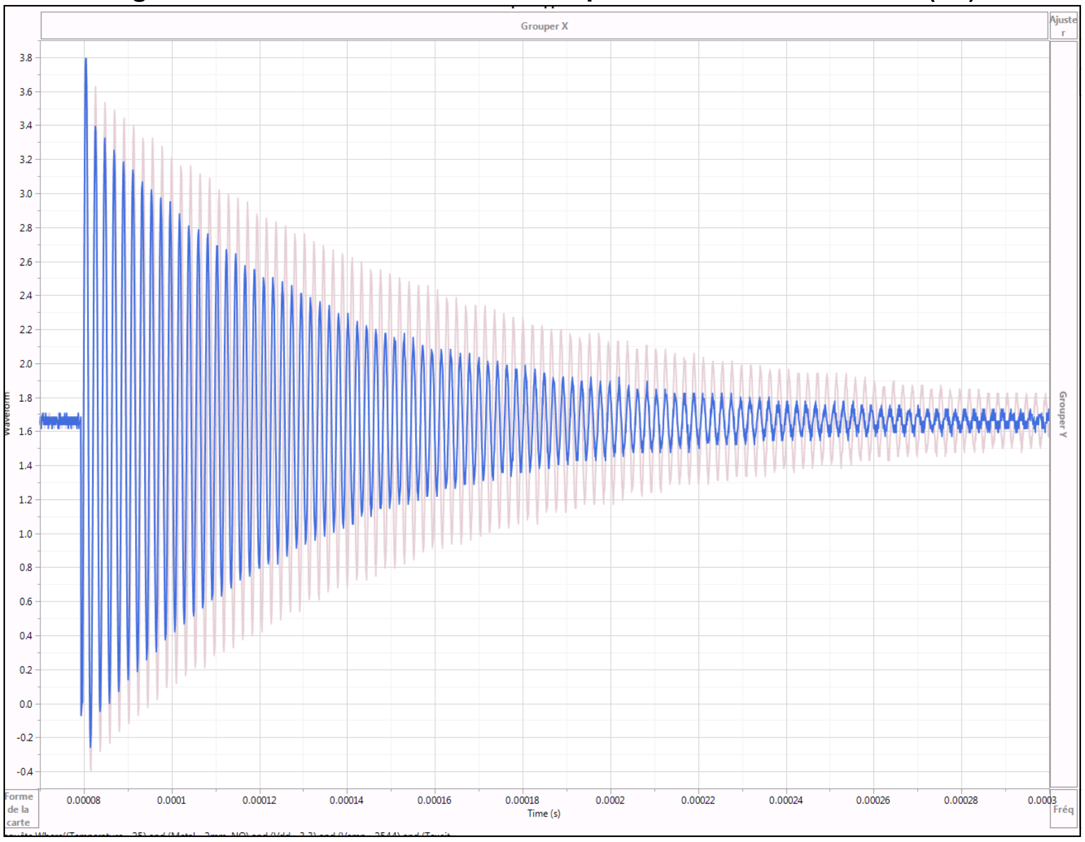
Figure 1. LC sensor oscillations example



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.

Figure 2 shows the oscillations in the two cases (metal in blue, while red refers to air).

Figure 2. LC sensor oscillations example with and without metal (air)



The detection is based on the comparison between the detected pulses in air and the ones counted in presence of metal. The count result is treated by software to define the metal or no metal state by comparison with a threshold count number.

Definitions

V_{mid}	Middle voltage (V_{DD} / 2) to energize LC sensor. It is delivered by an external operational amplifier for STM32L0 and by DAC peripheral output for STM32L4. To save power consumption, V_{mid} is disabled when the LC sensor is not activated.
V_{cmp}	Comparator threshold voltage, set above V_{mid} to provide a noise margin.
T_{excit}	Excitation pulse width to drive LC sensor in output push-pull (pull-up or pull-down).
$T_{capture}$	Time to perform the measurement. During this time, the comparator collects oscillations, compares them to V_{cmp}, and delivers pulses to the low power timer.
Count	Pulse count number generated by the LC sensor and conditioned by the comparator.

CountDetect	Detection threshold for the count number to define 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

The microcontrollers concerned by this application note are based on Arm® cores[a].

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

2 Application description - STM32L073Z-EVAL

2.1 Hardware required

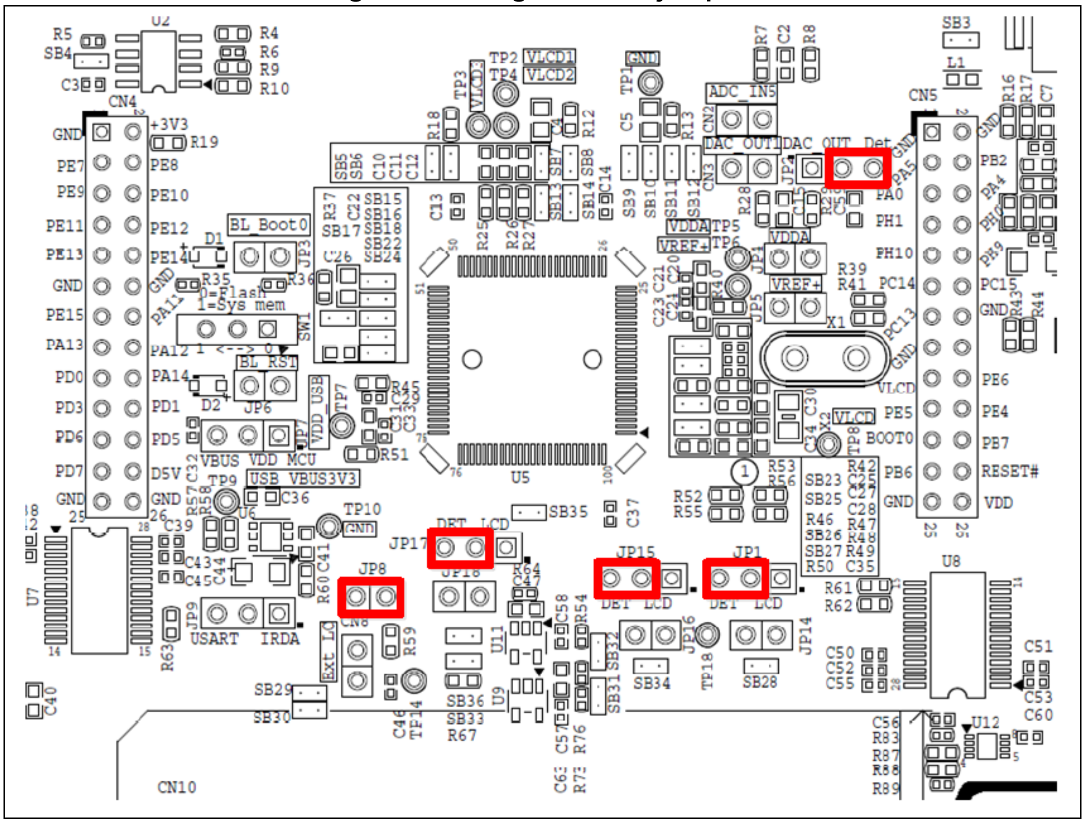
This demonstration uses STM32L073Z-EVAL evaluation board and the LC sensor metering section. The board can be connected by the USB cable to the host PC, and does not need any external power supply for this example (JP11 is fitted between 3-4 U5V_STLINK). A detection accessory board MB1199 is provided with the evaluation board and 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 (see *Section 2.4.1*).

2.2 Hardware settings

The SBx solder bridges must be in their initial factory configuration. Jumpers JP1, JP2, JP15, and JP17 must be set in the DET position (detection), and JP8 fitted as shown in *Figure 3*, where these jumpers are marked in red. If the user wants to measure the average current during the LC sensor metering in low-power mode (few µA range), an ammeter can be inserted on JP10 between pin 1 (VDD) and 2 (VDD-MCU).

Special attention must be paid to the measurement of pulsed currents (refer to *Section 2.5.2* for details). If some tests must be performed with an external LC sensor, SB29 and SB30 must be removed and the new LC cell connected to CN8 (external LC sensor).

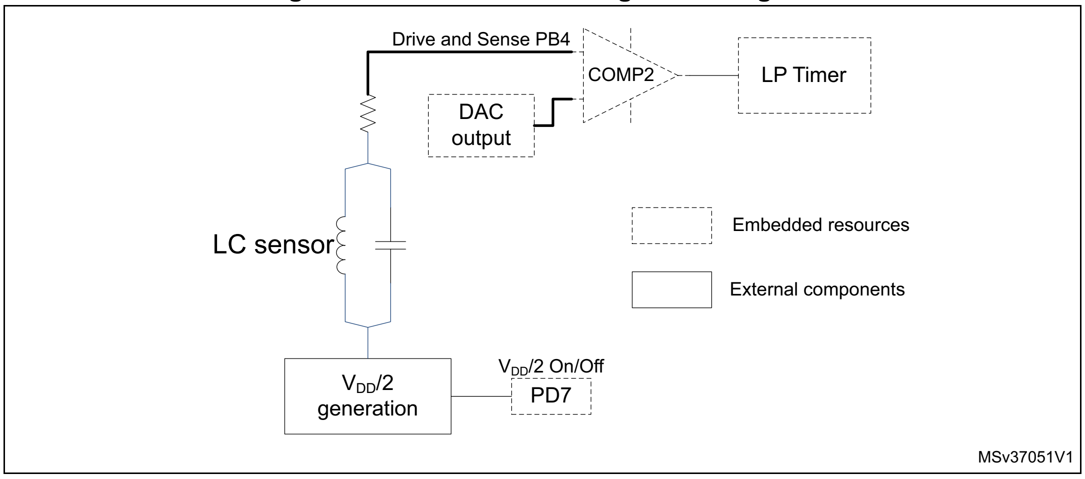
Figure 3. Configuration of jumpers



2.3 LC sensor metering block diagram

Figure 4 shows the LC sensor metering block diagram.

Figure 4. LC sensor metering block diagram



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

On the other side, the LC sensor is connected to PB4 pin through a serial resistor that limits the DC current needed to start oscillations. This current must be below the GPIO maximum current capability (20 mA). PB4 is used both as a GPIO drive pin and as sense pin connected to the comparator. When used as a drive pin, PB4 GPIO is set as an output pin and delivers a positive excitation pulse to the LC sensor, whereas it is set as the comparator positive input, to sense the oscillations signals as soon as they started. The COMP2 comparator is internally connected to the DAC, as an input threshold signal on its negative comparator input. The threshold signal is set approximately 100 mV above the V_{DD} / 2 bias voltage to provide a noise margin. Finally, the pulse train on comparator output is redirected to the low power timer input, to be counted.

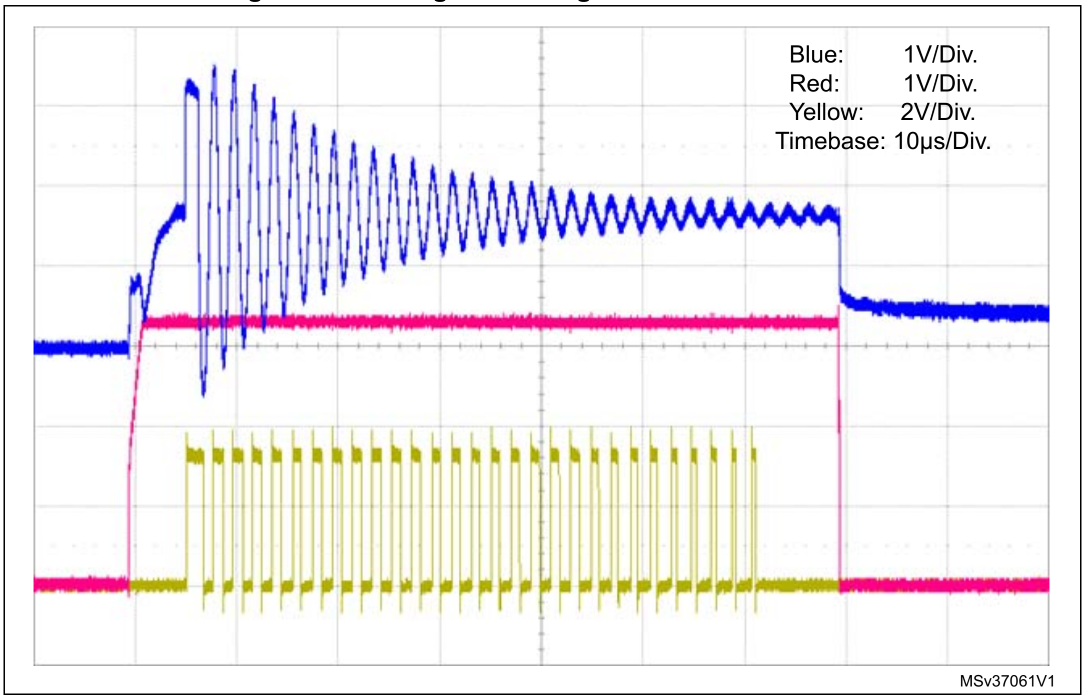
2.4 Application principles

2.4.1 Overview

As described above, the LC sensor is driven by an excitation pulse managed by PB4 GPIO. The pulse needs a minimum approximately width of 2 µs, so that the oscillations amplitude can reach and exceed the power signals rails (V_{DD} and GND). Once this pulse is delivered, the GPIO is reconfigured from output push-pull to alternate function analog input. The oscillations then appear on comparator non-inverting input, which is high-impedance.

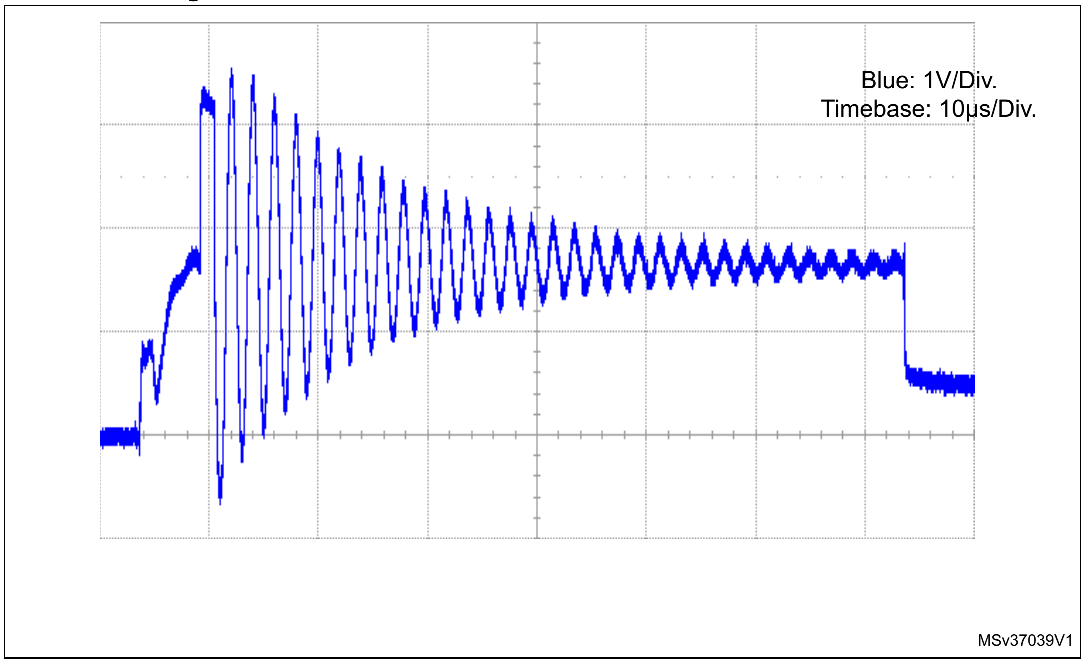
Figure 5 shows the main signals, including PB4 waveform (blue signal), V_{DD} / 2 controlled by PD7 (red signal) and COMP2 comparator output (yellow signal). V_{DD} / 2 is first turned ON (red signal), then the oscillations are generated by PB4 (blue signal) and collected by the LPTIM after being conditioned by the comparator (yellow signal).

Figure 5. Main signals during LC sensor activation



Once started, the amplitude of these oscillations decays exponentially until the signal comes back to V_{DD} / 2 bias, as shown in *Figure 6*. This behavior is obtained when the LC sensor is in air and no metal is present. This is what is called undamped oscillations.

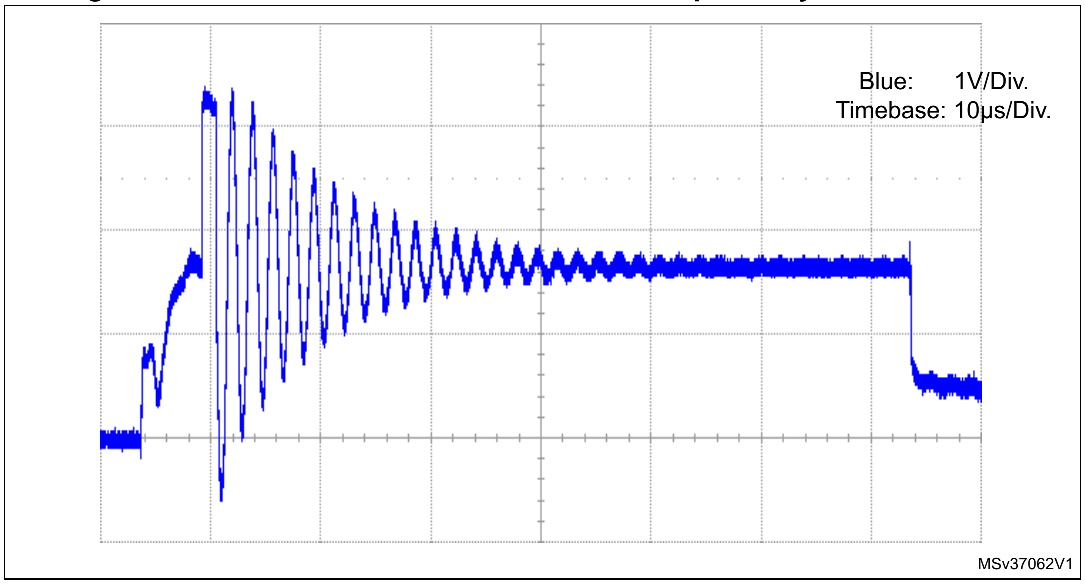
Figure 6. Oscillations collected with LC sensor in air on PB4



When a metal is in the proximity of the LC sensor, a part of the magnetic field emitted by the inductor is absorbed, as eddy currents and the energy of the inductor are lost in the metal target. In this case the oscillations observed on PB4 (measured on JP8) decay more rapidly and the number of collected pulses is lower than in the previous case (see *Figure 7*). The detection is based on the comparison between the detected pulses, obtained in air, and the ones counted in presence of metal.

During detection, both the amplitude and frequency of these oscillations change. The oscillations are damped, due to the presence of metal. The amplitude decreases with the proximity of metal and the frequency increases approximately 15%.

Figure 7. Oscillations collected with LC sensor in proximity of metal on PB4



The theoretical oscillations frequency is determined by the inductor and capacitor values:

$$f = \frac{1}{2\pi\sqrt{LC}}$$

In this example, L = 100 µH and C = 1 nF. The oscillations frequency is then approximately 503.3 kHz. These values have been selected according to the comparator input bandwidth and the current consumption needed to excite the LC sensor.

The damping time depends mainly upon the Q factor of the inductor. The higher Q, the lower the energy loss and the longer the damping time during the oscillation sequence.

2.4.2 LC sensor metering capture window

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

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

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

The sequence can be divided in four main steps:

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

Peripherals are initialized at the software start-up, then simply switched ON or clocked when the capture window occurs. Due to peripherals different stabilization times, a switching order is defined, to comply with the corresponding waiting times. For instance, COMP2 is initialized before DAC, due to a longer initialization time. Once the capture has been performed, the peripherals are switched off, to save power consumption. The Sleep on exit feature is used to turn the device into Stop mode as soon as the interrupt has been serviced and completed. Refer to *Section 3.2* for the description of flows.

2.5 Getting started with the demonstration

The LC sensor metering demonstration can be accessed by the main demonstration menu, including other applications examples. The starting menu shows some icons including the LC sensor demonstration located on the bottom left of the menu icons.

By the joystick (B3), the user can access the LC demonstration, by selecting the corresponding icon and by pressing the joystick center push-button.

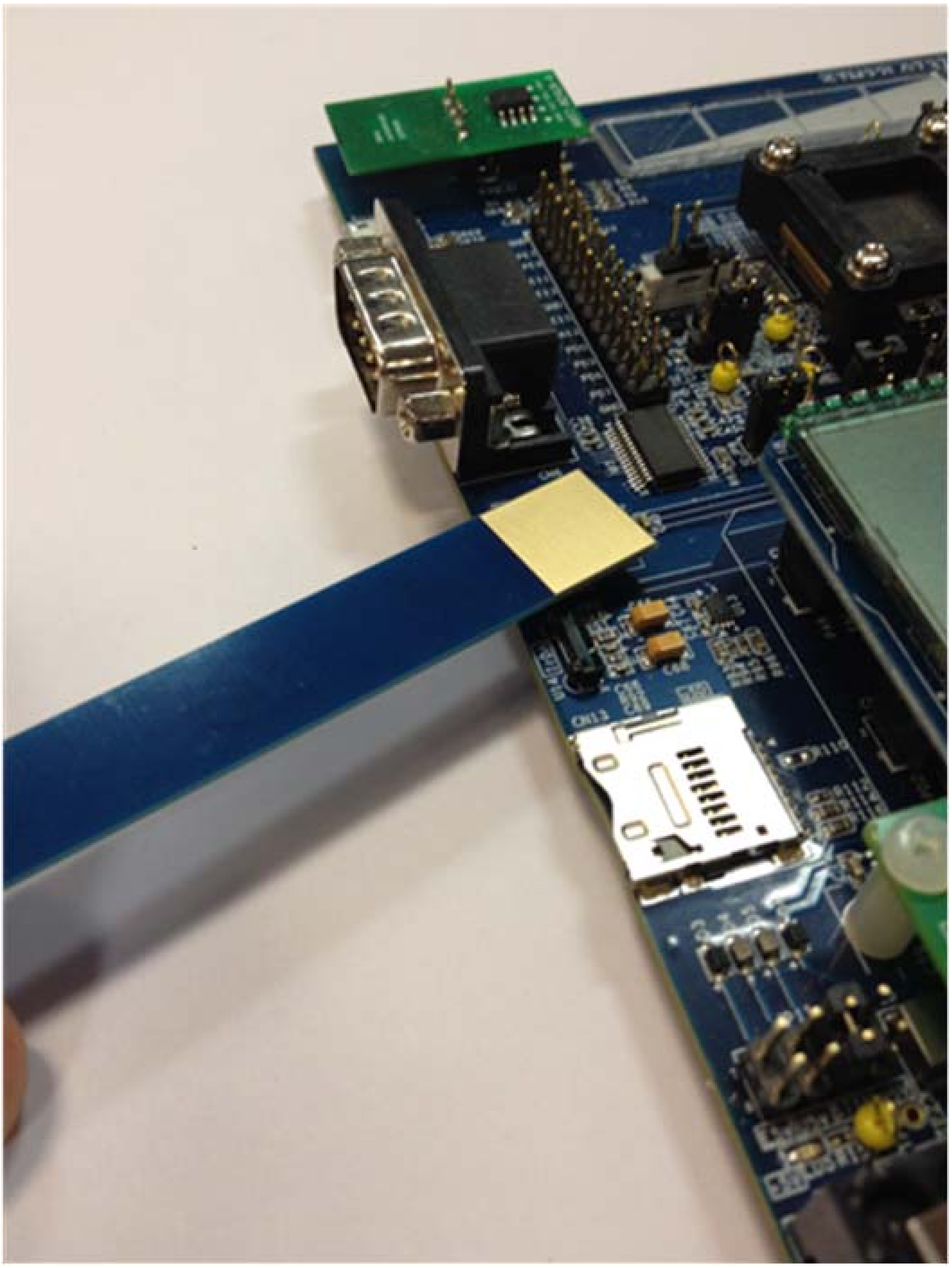
The LC sensor metering menu is then accessed. There are two available demonstrations:

- Mode Standard
- Mode Low Power

Mode Standard does not optimize the power consumption, and a graphical demonstration is available on the LCD TFT screen.

When using the detection accessory board MB1199, the metal strip must be positioned over the LC sensor as shown in *Figure 8*. Notice that the copper layer is oriented on the top side, leaving a dielectric gap of 0.8 mm between the LC sensor and the copper, to simulate the air gap between a rotating wheel and the detection coils.

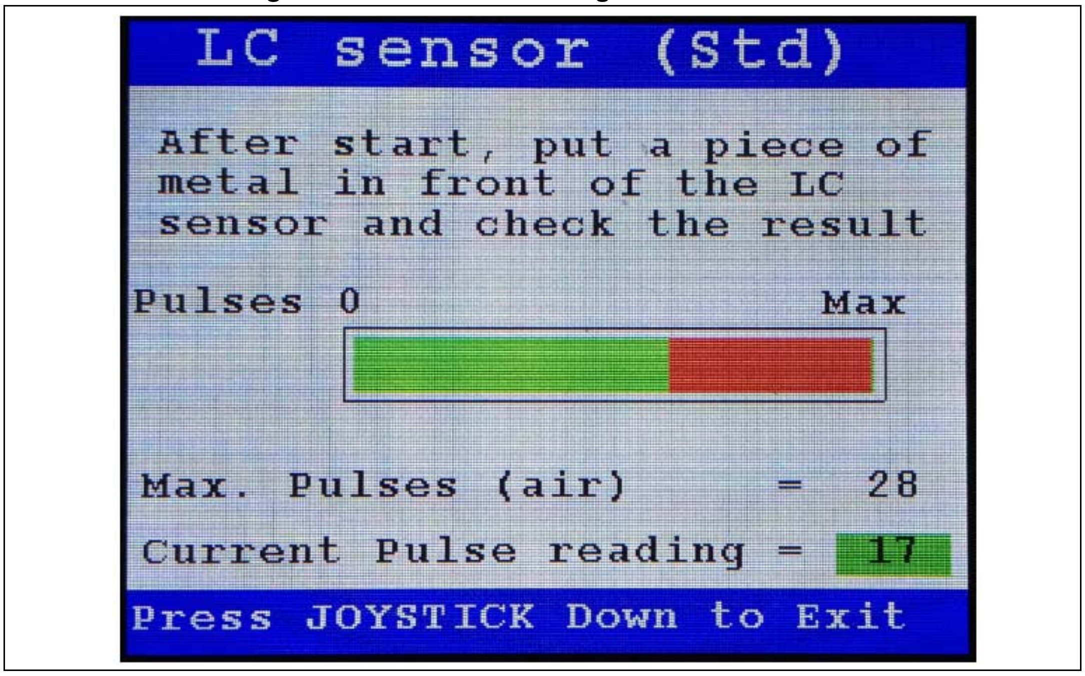
Figure 8. Using detection accessory board MB1199 with LC sensor metering



2.5.1 LC sensor metering in Standard mode

Entering in Standard mode, the menu shown in *Figure 9* is displayed. A bar graph shows the number of collected pulses (green area). The maximum graduation (Max. Pulses label) is reached when the LC sensor is in air (no metal). The missing pulses due to the proximity of the metal are displayed In the red area. The current pulse reading gives the number of pulses when a piece of metal is put above the LC sensor. If the damped oscillations reach 20% less of undamped oscillations in air, then the LED LD1 toggles as detection flag (green LED near B2 push-button) and the current pulse reading value is turned to green.

Figure 9. LC sensor metering in Standard mode



LC sensor (Std)

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

Pulses 0 Max

Max. Pulses (air) = 28

Current Pulse reading = 17

Press JOYSTICK Down to Exit

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

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

2.5.2 LC sensor metering in Low Power mode

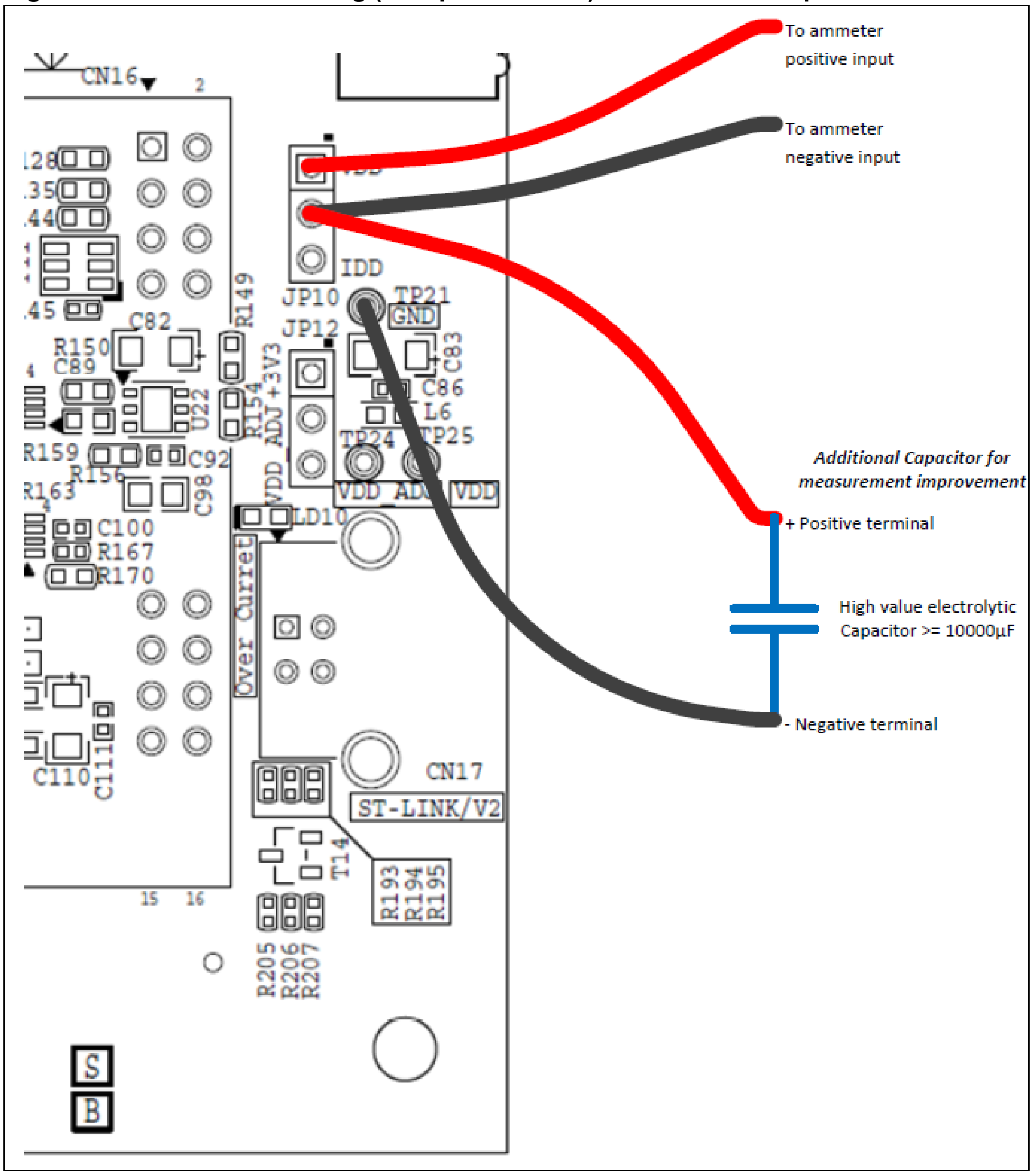
In this demonstration, the LCD TFT screen does not display any information from the LC sensor metering system. The goal is to evaluate the power consumption and the different signals, while the LC sensor feature is active to mimic a real application case.

To measure the power consumption on the V_{DD}-MCU signal, the JP10 jumper must be removed and an ammeter must be placed between pin 1 (+) and pin 2 (-) of JP10 connector (refer to *Figure 10* for the ammeter wiring).

The ammeter measures exclusively the current delivered by the V_{DD} power supply to the MCU. Knowing that the application changes from MCU running steps to MCU stop modes, the power consumption is not constant: when the microcontroller is in Run mode, the power consumption can be up to few mA, when it enters Stop mode (with RTC on LSE enabled), it drops to a maximum value of 1.2 µA (at 25°C).

The ammeter must be able to average the current value with a dynamic range from µA to mA, and extremely low duty cycles (80 µs Run time for 30 ms Stop duration). To ease the current average monitoring, a high value polarized electrolytic capacitor (~ 10 mF) can be placed with the positive terminal wired to the negative ammeter input and the negative terminal wired to the application ground (GND). This smooths the power consumption peaks. In these conditions, the average power consumption for the LC sensor metering Mode Low power is lower than 7 µA.

Figure 10. LC sensor metering (Low power mode): ammeter and capacitor connection

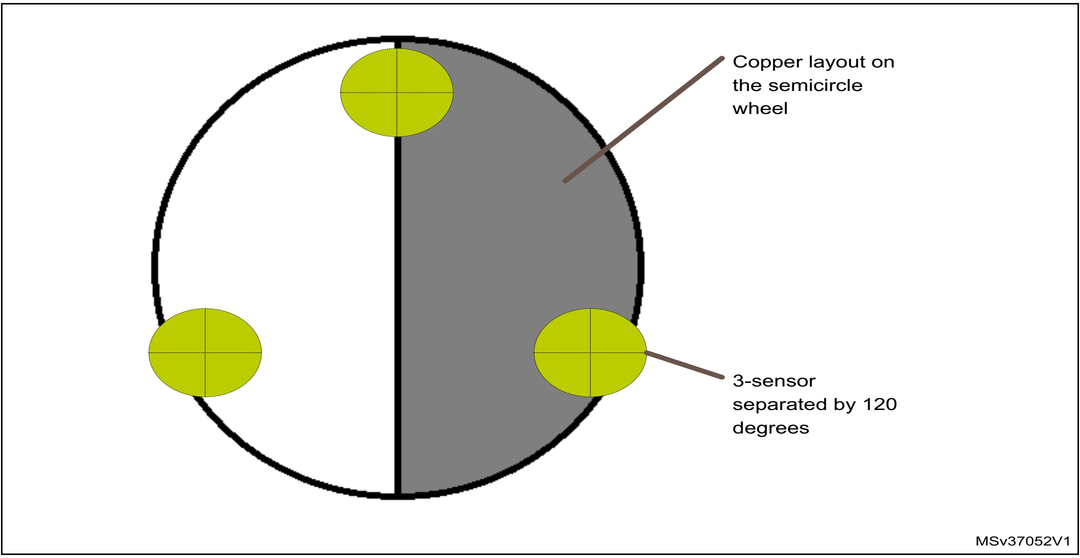

To ammeter positive input
To ammeter negative input
Additional Capacitor for measurement improvement
+ Positive terminal
High value electrolytic Capacitor >= 10000µF
- Negative terminal
CN16
IDD
JP10
JP12
TP21
GND
TP24
TP25
VDD_ADJ
VDD
CN17
ST-LINK/V2
LD10
Over Current

2.6 How to upgrade the LC sensor system up to three detectors

This demonstration shows an example of LC sensor metering, having only one LC network and it is intended to demonstrate the feasibility with the STM32L073x microcontroller. In gas and water meters, several LC sensors are used to detect the rotation of the wheel and to prevent errors that cannot be detected with 1-sensor systems.

The wheel features a semicircle with a copper layer and the other one with dielectric material. To detect the rotation sense, at least two sensors are needed and can be placed symmetrically. One LC sensor faces the copper layer, while the other is in front of the dielectric. To complete the system and offer more security and flexibility, three LC sensors are generally used to guarantee the operation in case one of them is defective. These sensors are spaced by 120 degrees as shown below.

Figure 11. 3-sensor LC metering arrangement

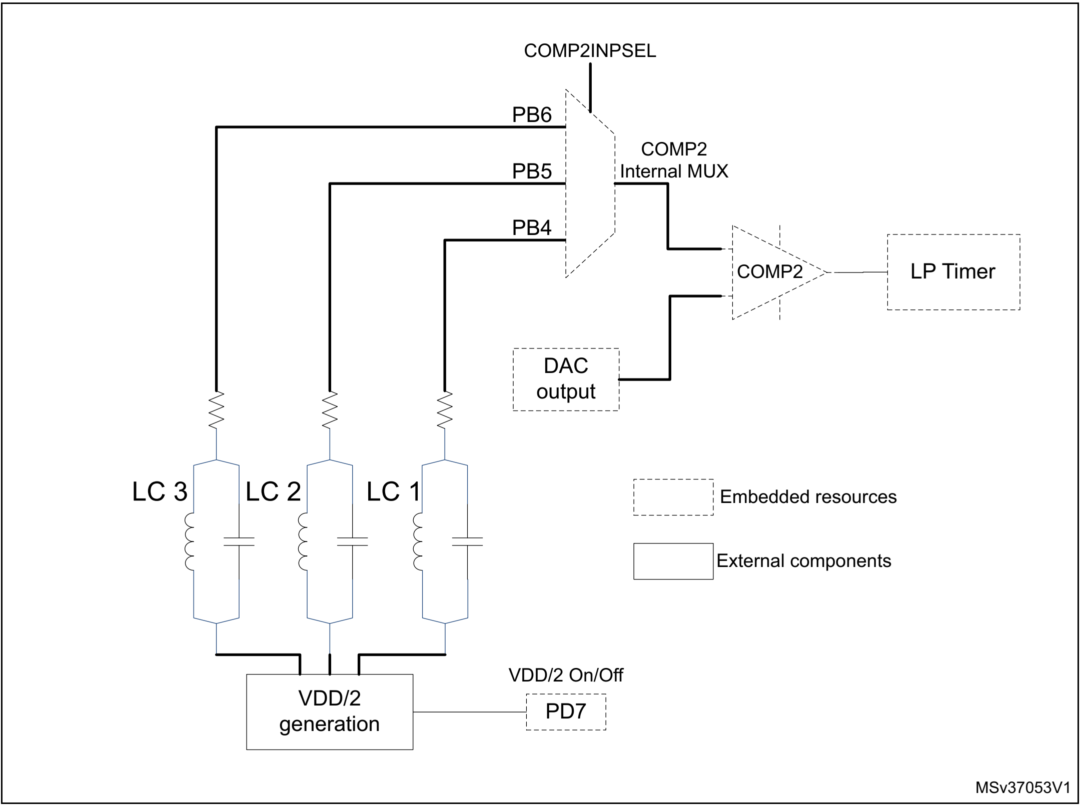

Copper layout on the semicircle wheel
3-sensor separated by 120 degrees
MSv37052V1

In our LC sensor detection system, based on the STM32L073 microcontroller, it is possible to extend the number of LC networks, as shown in *Figure 12*.

Three sensors are now available, and the selection can be done internally by the COMP2 positive input selection multiplexer. As the LC sensors must be enabled sequentially, only one V_{DD} / 2 generation circuit is needed and controlled by PD7. On the comparator side, PB4, PB5, and PB6 are activated as GPIO outputs, to drive the corresponding LC networks LC1, LC2, and LC3, and alternatively configured as positive comparator inputs.

Nevertheless, only one LC network can be scanned at a time. During the activation of PB4 for the LC1 network, PB5 and PB6 are set in high impedance pins. PB4 can be set as GPIO output, to excite the LC1 sensor, then immediately switched into COMP2 positive input to collect the generated pulses of the LC1 sensor. In the same way, while PB5 and LC2 are controlled, PB4 and PB6 are set in high impedance. Again, while PB6 and LC3 are controlled, PB4 and PB5 are set in high impedance.

Figure 12. 3-sensor LC metering system upgrade



The drawbacks are that all the sensors cannot be scanned at the same time and that the global power consumption of the system is multiplied by 3. If an 80 µs capture window is needed for one sensor, the data for the three sensors can be acquired in approximately 240 µs, still an adequate timing for low speed rotations wheels.

The sensors capture frequency (set here to 32 Hz) can be increased or decreased, according to the system desired response.

3 Firmware description - STM32L073Z-EVAL

3.1 STM32L073x peripherals used by the application

This application example uses the following STM32L073xx peripherals with the settings described below.

GPIOs

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

- PE4 set as output GPIO to drive the LED LD1.
- PD7 set as output GPIO to control the V_{DD} / 2 reference bias. This GPIO power supplies the operational amplifier, that is designed to generate the V_{DD} / 2 reference voltage. This allows switching ON and OFF this amplifier and save power consumption.
- PB4 is alternatively used as output GPIO, to drive the LC sensor and put in alternate function (positive comparator input), once the excitation pulse is delivered.

DAC

DAC_OUT1 is used to generate the input threshold voltage, needed by the comparator. Oscillations signals from LC sensor are referenced to V_{DD} / 2. The DAC peripheral allows to provide a preset threshold voltage value near V_{DD} / 2 but with a necessary positive margin to eliminate noise. For further developments, using the DAC as comparator threshold voltage, would allow to calibrate the threshold value according to different LC network responses. Typically, LC network responses and collected noise vary from one material to another, and the capability to adjust the comparison threshold, is an improvement compared to fixed threshold voltages.

COMP2

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

LPTIM

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

RTC

This peripheral is used to wake up the microcontroller at regular intervals once it has entered Stop mode. The RTC clock is connected to the LSE clock, whose frequency is 32.768 kHz. The wake-up frequency corresponds to the LC sensor capture frequency, set to 32 Hz for this example. This capture frequency can be increased or decreased, depending upon the wheel maximum rotating speed. Modifications to the capture frequency lead to a change of the average power consumption.

TIM

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

Interrupts

LC sensor metering Mode Standard and Mode Low Power are distinguished in the RTC IRQ handler process.

To save the maximum time and then power consumption in MCU Run mode, the RTC wake-up timer IRQ handler is directly processed inside the interrupt subroutine for the Mode Low Power, while a specific RTC wake up timer call back function is used for Mode Standard.

3.2 Description of flows in Low Power mode

The software for Low Power mode (including Stop mode entry procedure and wake-up processes, performing the LC sensor metering feature) is described in this section.

The global demonstration includes several examples related to many features, called modules, such as the lc_sensor_metering_app.c, pressure_app.c, thermometer_app.c, and others.

These examples are launched from a main.c file included in the Demo folder and gathering all these modules in a single demonstration project.

The focus is on the function called by the LC sensor metering demonstration for the low power mode, which is named LcSensorCountingDemoLowPower(). This function is the entry point for the Mode Low Power LC sensor example, once the user has chosen the LC sensor demonstration in the graphical menu, among the different examples and select Mode Low Power in the submenu.

A detailed description is available in this section for the StopEntry() function, which is intended to prepare the microcontroller for the lowest power consumption in Stop mode.

Finally, the RTC IRQ handler is described to explain the wake-up sequence that triggers the LC sensor measurement, while the microcontroller is in Run state (capture window).

3.2.1 LcSensorCountingDemoLowPower() function

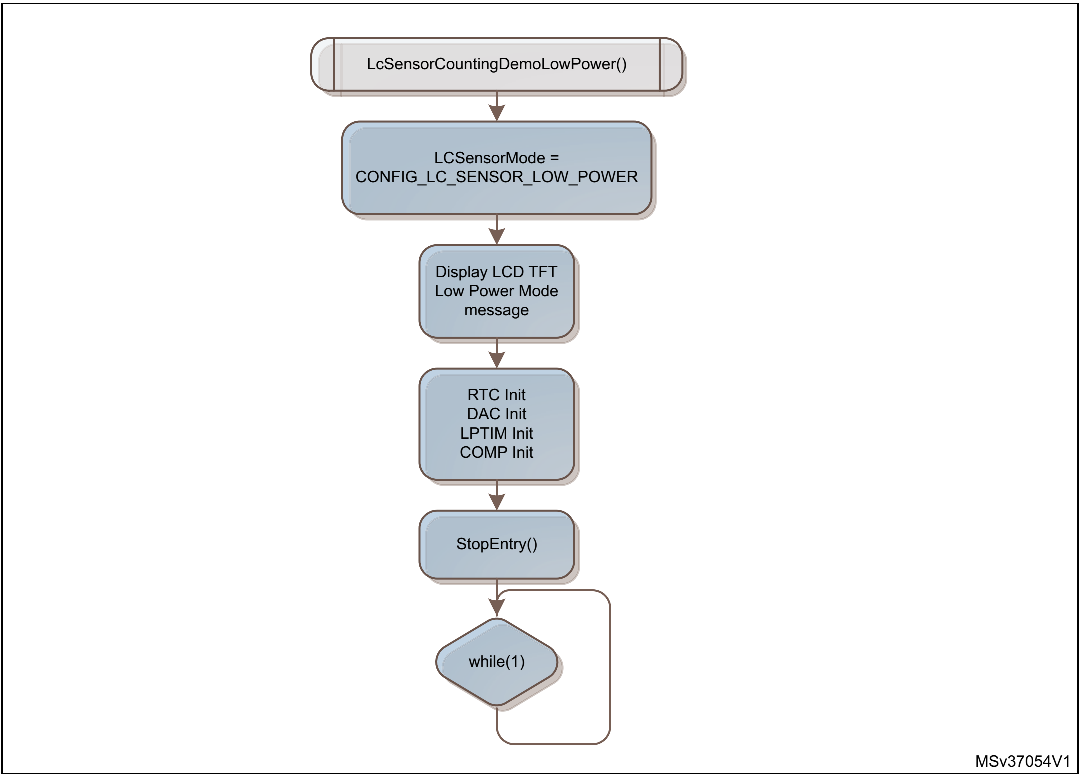
As soon as this function is fully executed, a RTC interrupt is created to wake up the MCU at regular intervals. This function is then called once and the application will later alternate Stop sequences and Run slots at the frequency rate, given by the RTC wake-up parameters. In this example, the microcontroller is woken up 32 times per second.

As shown in *Figure 13*, the function starts by setting the LCSensorMode variable to indicate in which mode the current demonstration is. The LCSensorMode variable is used to identify the Mode Low Power in the RTC_IRQHandler() and then to bypass callback functions to save time and power consumption (refer to firmware file stm32l0xx_it.c for details).

The LCD TFT display is updated to alert the user on the way to exit this Mode Low Power demonstration. A reset must be applied to exit this example.

RTC, DAC, LPTIM, and COMP peripherals are initialized with the settings described in *Section 3.1*. Then, the StopEntry() function is executed to prepare the device to enter Stop mode with specific settings, and the microcontroller enters an infinite loop, waiting for the RTC wake-up interrupt. All processes are performed in interrupt, using Sleep on exit mode, thus no activity is performed in main.c section. User can exit the demonstration by applying a reset sequence, pressing B1 (RESET button).

Figure 13. LCSensorCountingDemoLowPower() function

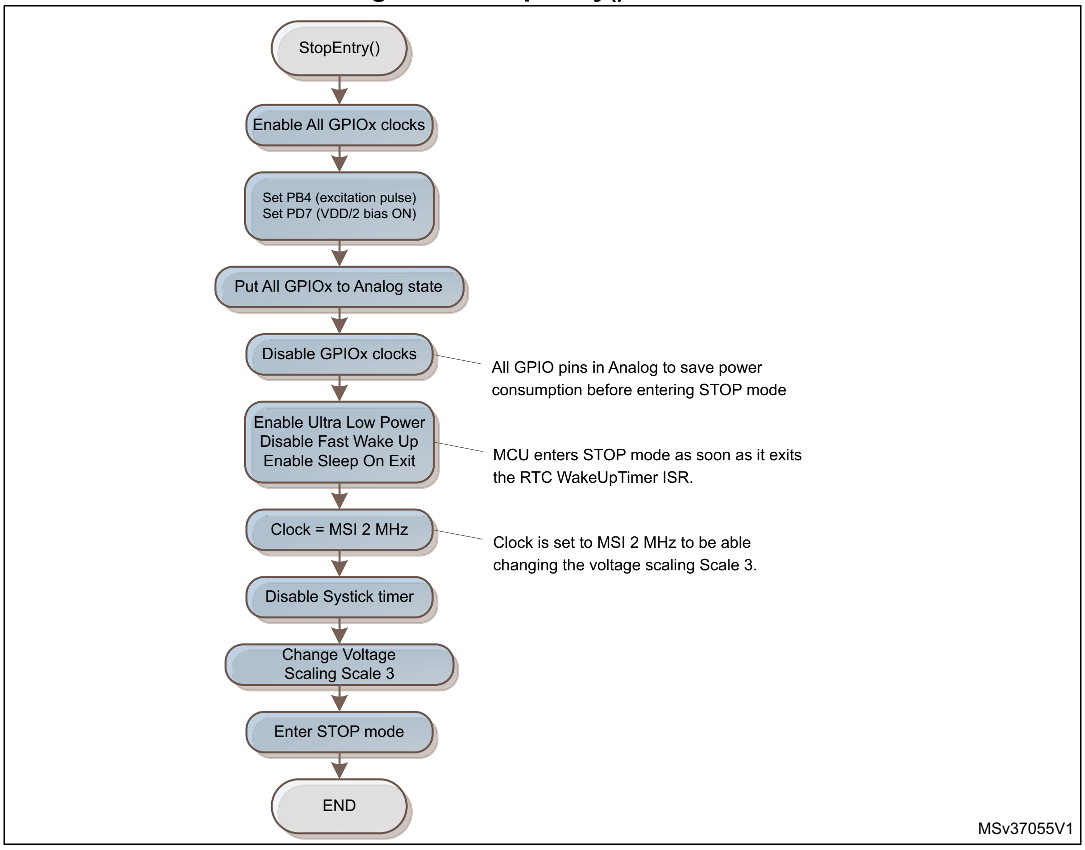

LcSensorCountingDemoLowPower()
LCSensorMode = CONFIG_LC_SENSOR_LOW_POWER
Display LCD TFT Low Power Mode message
RTC Init
DAC Init
LPTIM Init
COMP Init
StopEntry()
while(1)
MSv37054V1

3.2.2 StopEntry() function

The aim of this function is to prepare the device to enter Stop mode with parameters set to minimize the power consumption. It starts by enabling all GPIO clocks to perform GPIO registers modifications. Then PB4 and PD7 output data registers are preset to logical 1. All GPIO can now be placed in Analog state to minimize power consumption and the associated clocks disabled. The internal voltage reference is disabled by setting the ULP bit in PWR_CR register. Fast wake-up mode is disabled to wait for VREFINT ready, before waking up the MCU and using COMP2. The feature Sleep on Exit is used to force the MCU entering the Stop mode, as soon as the RTC wake-up has been serviced and executed. In that case, the StopEntry() function is executed once and the microcontroller returns in Stop mode after each interrupt.

The MSI clock is set to 2 MHz to minimize the power consumption. The voltage scaling is adjusted accordingly (range 3, 1.2 V), to further decrease dynamic consumption. The Systick timer is disabled for the same reason. This operation must be performed after the system clock changes. All wait states are disabled.

Figure 14. StopEntry() function

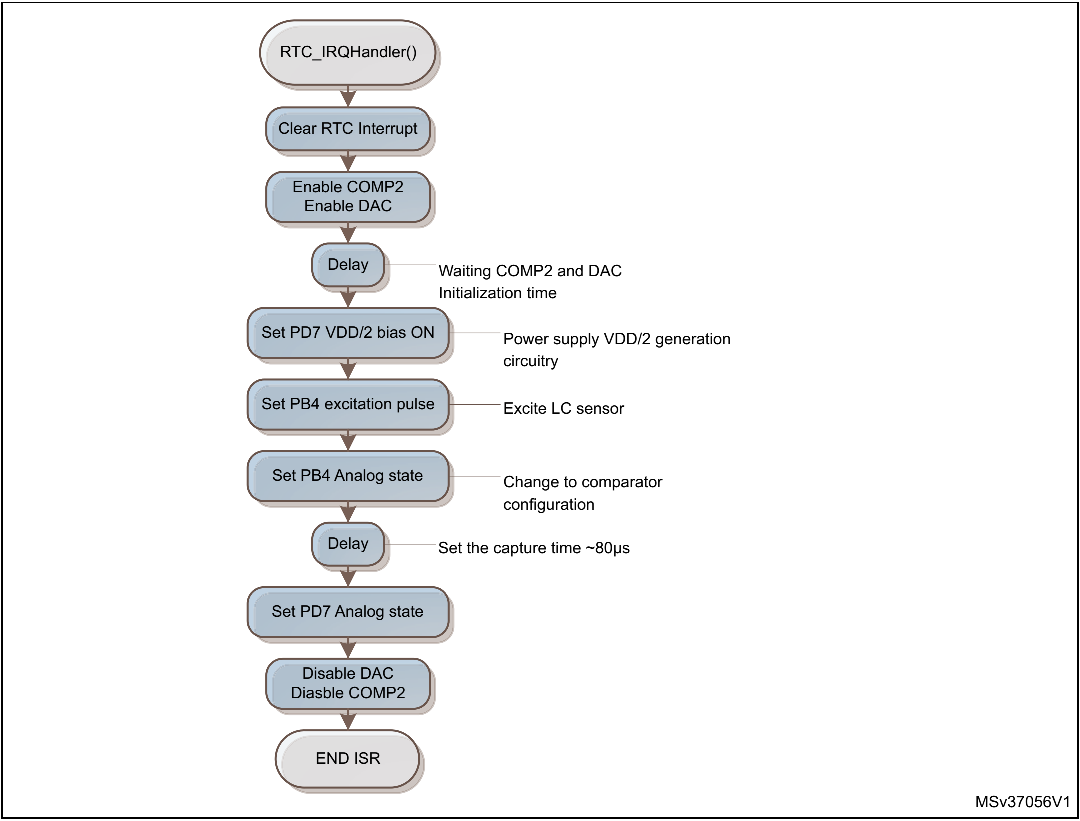

StopEntry()
Enable All GPIOx clocks
Set PB4 (excitation pulse)
Set PD7 (VDD/2 bias ON)
Put All GPIOx to Analog state
Disable GPIOx clocks
Enable Ultra Low Power
Disable Fast Wake Up
Enable Sleep On Exit
Clock = MSI 2 MHz
Disable Systick timer
Change Voltage Scaling Scale 3
Enter STOP mode
END
All GPIO pins in Analog to save power consumption before entering STOP mode
MCU enters STOP mode as soon as it exits the RTC WakeUpTimer ISR.
Clock is set to MSI 2 MHz to be able changing the voltage scaling Scale 3.
MSv37055V1

3.2.3 RTC_IRQHandler() interrupt subroutine

The RTC IRQ Handler manages the RTC interrupt triggered by the RTC wake-up timer. In this example, the interrupt occurs 32 times per second. To save time during the interrupt execution and since the device operates in Run mode, the interrupt subroutine is executed in the Low Power demonstration mode, without performing any call back function in the main program.

The interrupt flag is first cleared. Then COMP2 and DAC peripherals are switched on and a delay is inserted, to wait the peripherals stabilization times before they can be operational. The V_{DD} / 2 reference bias voltage is turned on by mean of PD7 GPIO and the excitation pulse is delivered by PB4 pin when first placed in GPIO output mode. As soon as the positive pulse is applied on the LC network, PB4 is switched into Analog state and acts as the non-inverting comparator input. Application waits for an 80 µs capture time, until all the pulses are collected, and switches off the V_{DD} / 2 reference bias voltage. DAC and COMP2 are disabled to minimize power consumption. The capture of the LC sensor oscillations has been performed during this interrupt.

Figure 15. RTC_IRQHandler() interrupt subroutine



4 Application description - STM32L476RG-NUCLEO

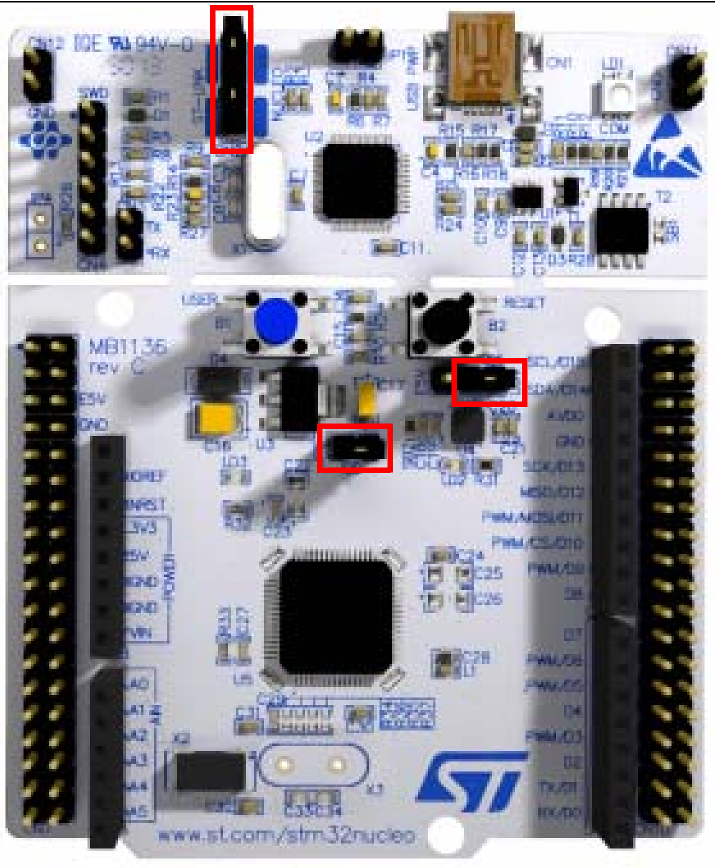
4.1 Hardware required

This demonstration uses the STM32L476RG-NUCLEO64 board and custom board for LC sensing with external components (see *Figure 18* and *Figure 19*). The 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.

4.2 Hardware settings

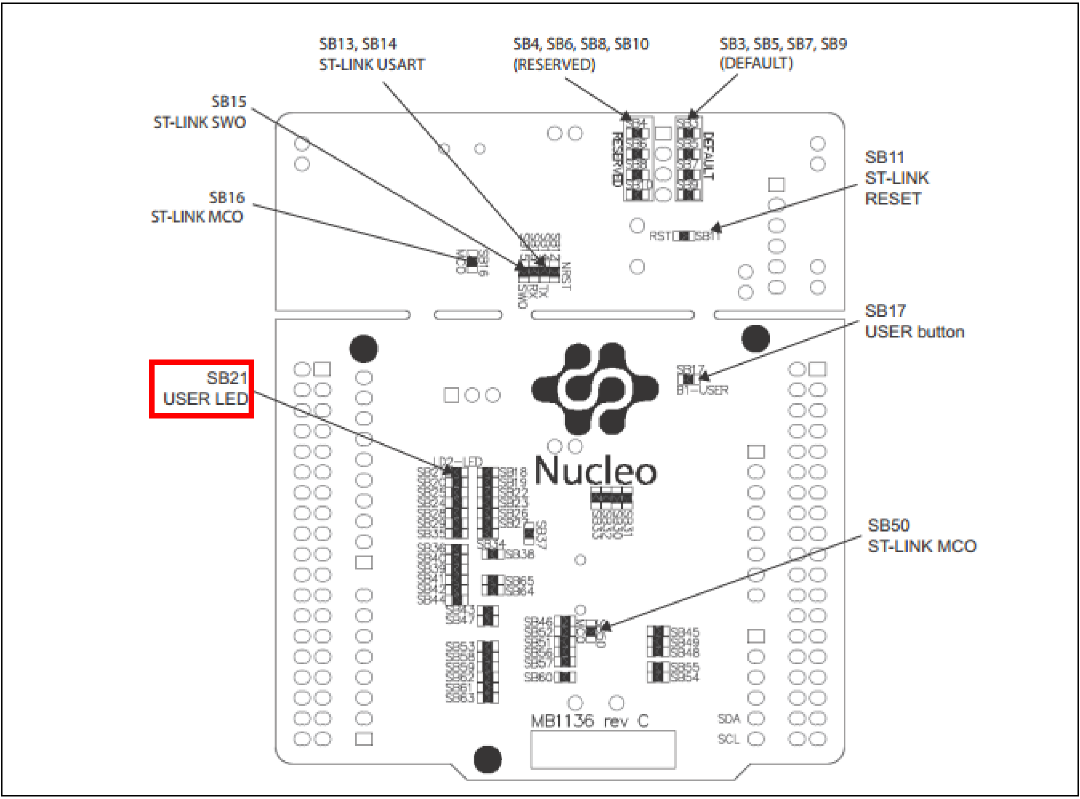
Jumpers (red boxes in *Figure 16*) must be set in their initial factory configuration.

Figure 16. Configuration of jumpers



The SBx solder bridges must be set in their initial factory configuration. DAC output channel (V_{mid}) can be routed to PA4 or PA5 pin. If PA4 is used, no change is required, if PA5 is used, SB21 must be disconnected.

Figure 17. SB21 USER LED disconnection (only if PA5 is used)



4.3 Hardware settings for LC sensor custom board

4.3.1 Requirements

A custom board with LC sensors must be developed and connected to the STM32L4 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 = 470 µH (e.g. Murata 11R474C)
  - Cs = 220 pF (e.g. Murata RPE5C2A221J2S1A03A)
- Reference capacitor: used to store energy for LC sensors oscillations.
  - Cref = 470 nF (e.g. Murata RPER71H474K2M1C03A)
- 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 (20 mA for STM32L4x6, refer to the product datasheet).
  - R = 150 Ω (for V_{DD} = 3 V)
- Protection diodes: external diodes must be connected to IO and VDDA, see *Excitation time: Texcit* to define if requested or not. The I/O type is _a with analog switch function

supplied by VDDA (see datasheet for details). For this example the diode used is STMicroelectronics BAT43.

Table 1. I/O configuration

LC I/O	Pin(s)	LC function
IO1	PA4 or PA5	Vmid
IO2	PC5	Sensor1
IO3	PB2	Sensor2
IO4	PB4	Sensor3
IO5	PB6	Sensor4

Figure 18. Schematic: one or two sensors

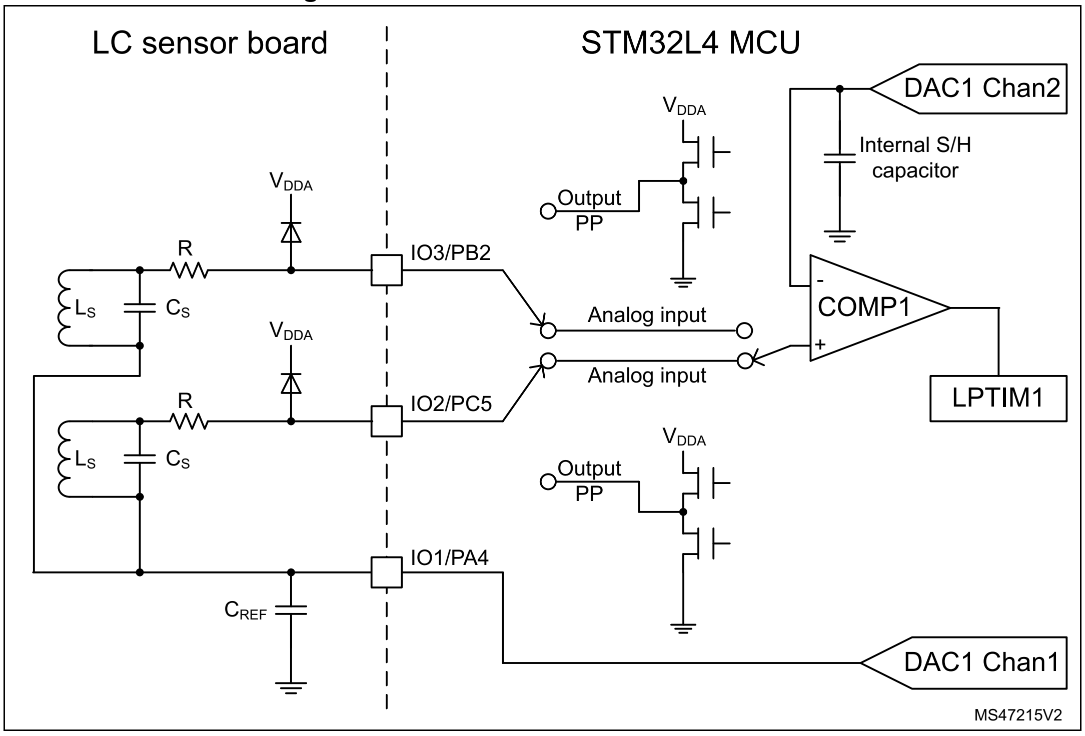
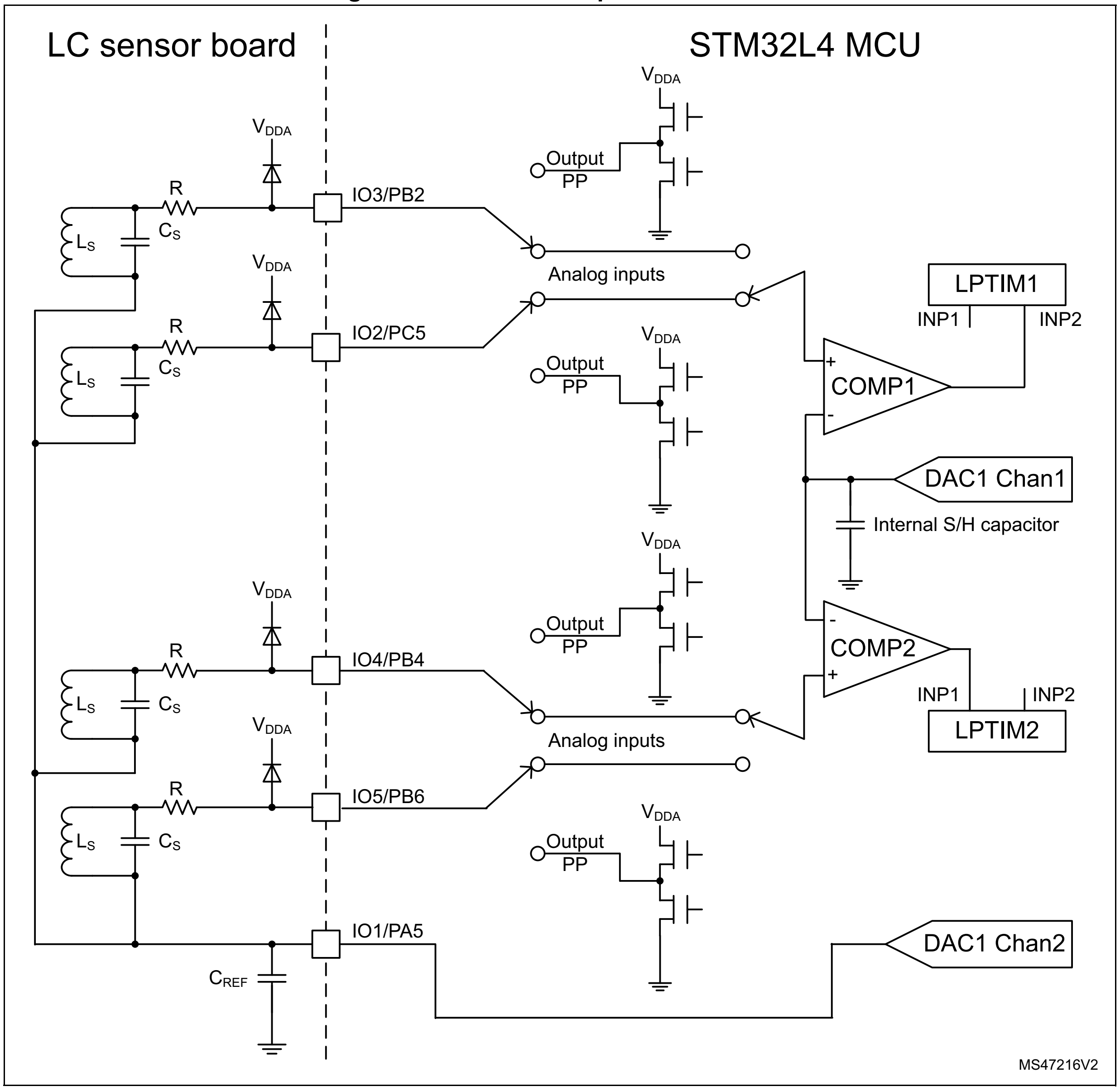


Figure 19. Schematic: up to four sensors



4.4 Application principles

4.4.1 Overview

The application is based on the same principle seen for the STM32L0 application, the main difference is that the V_{mid} level (V_{DD} / 2) is generated with STM32L4 DAC peripheral instead of an external circuitry.

The LC sensor is driven by an excitation pulse managed by GPIOs. Once the pulse is delivered, the GPIO is reconfigured from output push-pull to analog input. The oscillations appear on the non-inverting comparator and compared to V_{mid} level (V_{DD} / 2).

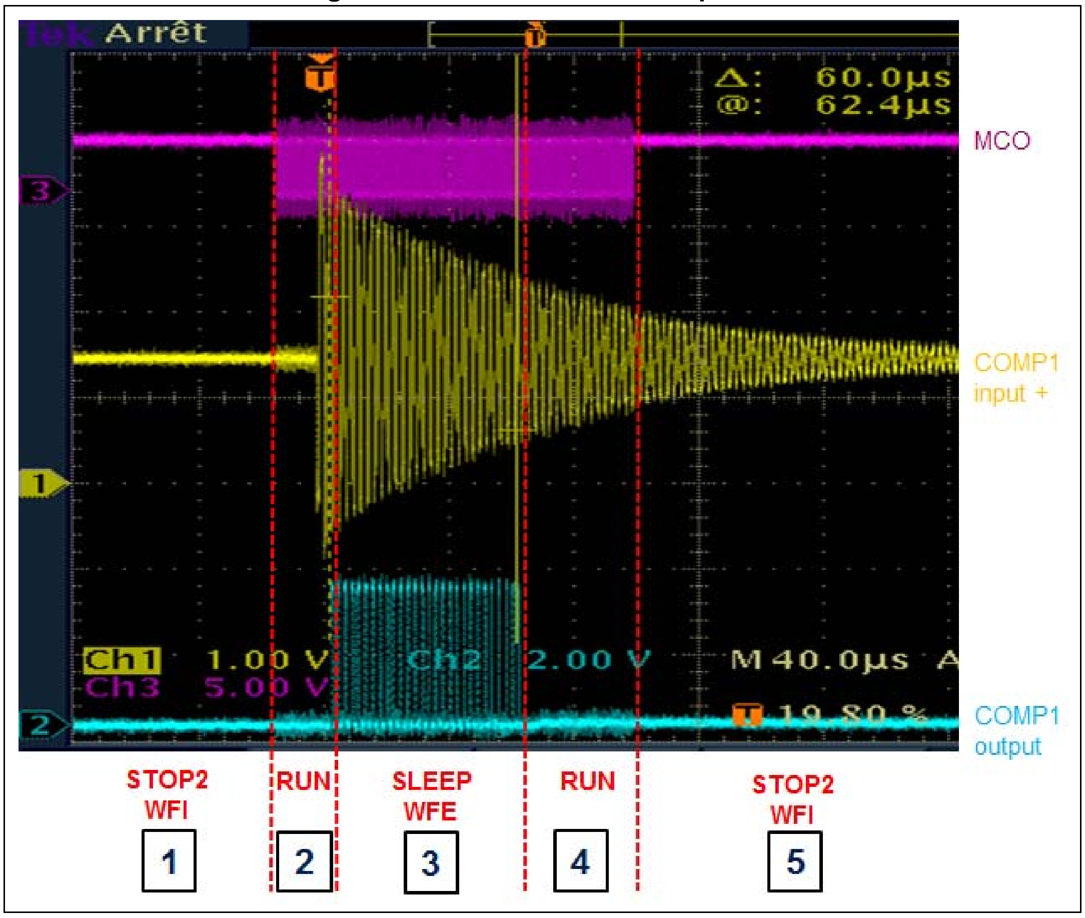
V_{mid} level is generated by the internal DAC and can be stopped after the measurement to reduce the power consumption.

Once started, the amplitude of these oscillations decays exponentially until the signal comes back to V_{mid} bias, as shown in *Figure 20*. This behavior is obtained when the LC sensor is in air and no metal is present.

4.4.2 LC sensor metering capture window

To reduce power consumption, the LC sensor metering is activated periodically and the microcontroller is put in the Stop2 mode (Low Power mode demonstration selected).

Figure 20. LC measurement sequence



For each measurement, as indicated in *Figure 20*, the following steps are executed:

1. The LC sensor metering is activated and the microcontroller is put in the Stop2 mode (Low Power mode).
2. The MCU is woken up and enables the peripherals used by application, with a waiting time inserted to let peripherals stabilize. One pulse timer is started and the MCU is put in wait for timer event (WFE), as described in *DAC configuration*. After that, the LC sensor excitation pulse is delivered (see *Excitation time: Texcit*). To perform the measurement, one pulse timer is started and the MCU is placed in WFE (refer to *Capture window: Tcapture*).
3. During the measurement phase the MCU is put in Sleep 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 timer event, the MCU returns in Run mode, processes the LPTIMER result and compares it with the detection threshold (CountDetect) to define if there is a Metal or No Metal state. The software state machine can process counter results at this level.
5. The MCU goes back in Stop2 mode and reaches the lowest power consumption.

4.4.3 Parameters to define

Some parameters must be defined according to the application requirements:

DAC configuration

Two channels are used to generate V_{mid} and V_{cmp} levels:

- V_{mid} for reference level to V_{DD} / 2: Channel 1 or Channel 2 connected to external pins, respectively PA4 or PA5.
- V_{cmp} for threshold comparator level: Channel 1 or Channel 2 connected to on chip comparator inverting input.

DAC peripheral is configured in Sample and Hold mode, and managed in wake-up interrupt. It is enabled at each LC measurement and disabled before returning in Stop2 mode.

Reference voltage (V_{mid})

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

DAC V_{mid} (to program) = 4096 * (V_{mid} / V_{DD}) = 4096 / 2 = 2048

Comparator threshold voltage (V_{cmp})

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

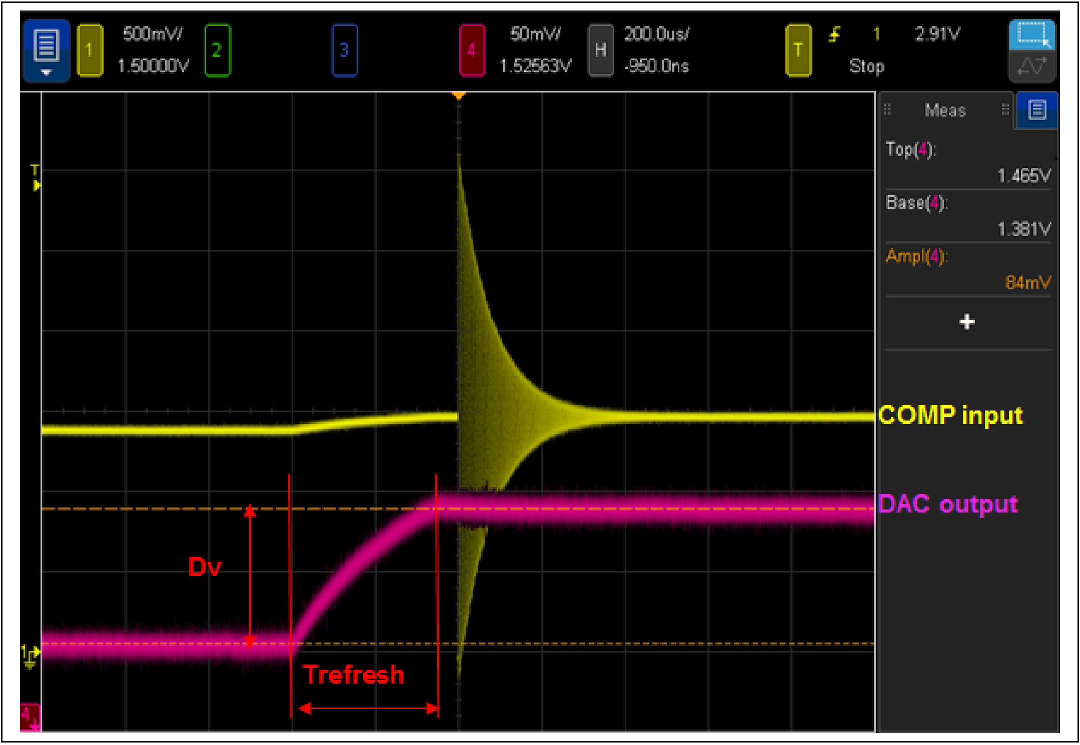
In this example: DAC output is configured to deliver V_{cmp} = V_{mid} + 750 mV

DAC V_{cmp} (to program) = 4096 * (V_{cmp} / V_{DD}) = 4096 / 3 V * 2.25 V = 3072 = 0xC00

DAC V_{mid} refresh time ($T_{refresh}$)

To save power consumption, the DAC output is disabled between two LC measurements. During this phase, the DAC output level is in analog state (high impedance) and the V_{mid} level is held by Cref capacitor. V_{mid} drop (Dv) depends upon the holding time (time between two measurements). Before each measurement, a DAC refresh time after DAC enabling is required to reach the expected V_{mid} value.

Figure 21. DAC output example

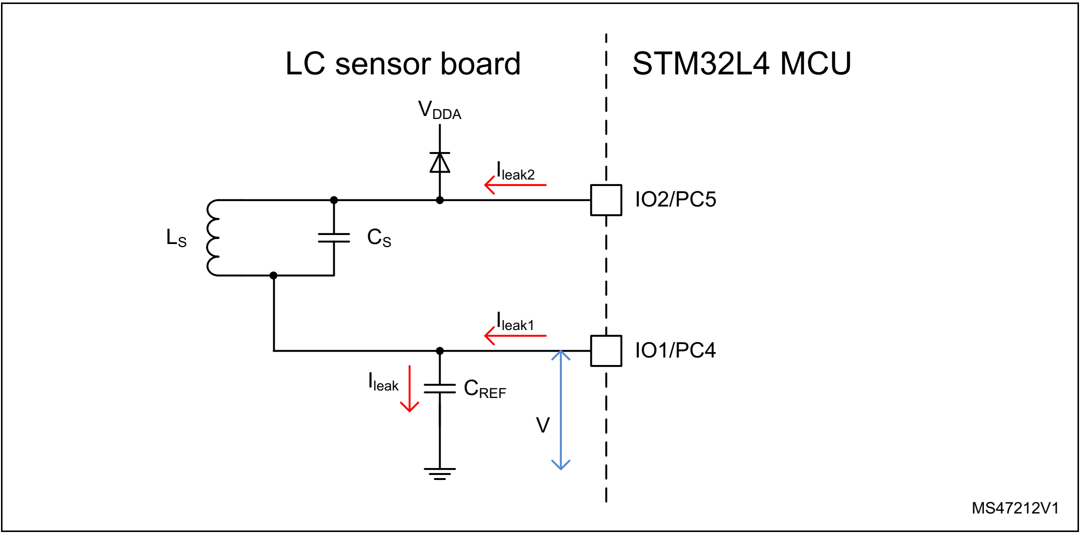


Dv computing

V_{mid} drop (Dv) during holding time (the time between two measurements) depends upon the leakage current (I_{leak} = 150 nA by I/O, as specified in the STM32L4x6 datasheet).

For each I/O I_{leak} = 150 nA (worst case on the I/O leakage on all the temperature range), hence $I_{leak} = I_{leak1} + I_{leak2} + I_{leak3} + ... = n * 150$ nA, where n is the total number of I/Os (see *Figure 22*, where currents are indicated in red, and voltage in blue).

Figure 22. Cref leakage current during hold time



The tolerated voltage drop during the hold phase (dv) is represented by the number of LSBs after the capacitor discharges with the output leakage current, that is Nlsb = (dv / V_{DD}) * 2^{12}.

To compute dv, use the relation Cref * (dv / dt) + I_{leak} = 0, so dv = - (I_{leak} * dt) / Cref.

For this example, a 10 mV error accuracy has been defined for power saving.

The settling back to the desired value with 10 mV error accuracy (14 LSB at 3 V) requires a DAC constant time equal to ln (Nlsb / 14).

$Lsb_{accuracy}$ = (V error accuracy) / lsb

lsb = V_{DD} / 2^{12} = 3 / 4096 = 0.00073 V

$T_{refresh}$ computing

The DAC V_{mid} refresh time is set to let the time to reach the value according to the desired accuracy error.

$T_{refresh}$ = TstabBON + (RBON * Cref) * ln (Nlsb / ($Lsb_{accuracy}$))

The parameters TstabBON and RBON are specified in the datasheet.

DAC refresh time is managed with timer peripheral (TIM6) configured in One Pulse mode. The HCLK frequency is decreased (divided by 64) during sleep to save power consumption.

$T_{refresh\_to\ program}$ = $T_{refresh\_seconds}$ * (Freq_MSI / 64)

Examples

For one sensor and Cref = 470 nF:

- Sampling time 500 Hz (hold time = 2 ms):
  dv = -(2 * 150 nA * 2 ms) / 470 nF ≈ -1.3 mV → Nlsb = 2
  - 1 mV accuracy error:
    $T_{refresh}$ = 237 µs, $T_{refresh\_to\ program}$ = 237 µs * 24 MHz / 64 = 89
  - 10 mV accuracy error:
    $T_{refresh\_to\ program}$ = 0, No more refresh time required because Dv < 10 mV
- Sampling time 10 Hz (hold time = 100 ms):
  dv = -(2*150nA * 100 ms) / 470 nF ≈ -64 mV → Nlsb = 87
  - 1 mV accuracy error:
    $T_{refresh}$ = 3.91 ms, $T_{refresh\_to\ program}$= 3.91 ms * 24 MHz / 64 = 1468
  - 10mV accuracy error:
    $T_{refresh}$ = 1.75 ms (10 mV accuracy error)
    $T_{refresh\_to\ program}$ = 1.75 ms * 24 MHz / 64 = 656

Note: Oscilloscope probes connected to LC sensors change significantly the system behavior because of their impedance.

Note: The MCU is in SLEEP mode during DAC Vmid refresh time. This time must be minimized to save power consumption and a compromise with Vmid accuracy must be struck. For this example an accuracy error of 10 mV has been used.

$ileak_{probe}$ = V / Rprobe, where Rprobe is the probe input resistance

ileak = ileak1+ ... + $ileak_{probe1}$ + $ileak_{probe2}$ + ...

Examples for one sensor, Cref = 470 nF and one probe single ended with Rprobe = 1 MΩ:

- Sampling time = 500 Hz (hold time = 2 ms):
  dv = -((2 * 150 nA + 1.5 V / 1 MΩ) * 2 ms) / 470 nF ≈ -7.7 mV, instead of -1.3 mV without probe
- Sampling time = 10 Hz (hold time = 100 ms):
  dv = -((2 * 150 nA + 1.5 V / 1 MΩ) * 100 ms) / 470 nF ≈ -383 mV, instead of -64 mV without probe

Excitation time: T_{excit}

LC sensor excitation can be done to V_{SS} (output push pull down) or to V_{DD} (output push pull up), the default value used in this example. The pulse width can be adjusted to set the oscillation amplitude (T_{excit}).

If the maximum oscillation amplitude exceeds maximum voltage allowed, an external diode must be added to protect the I/O. See the product datasheet for details. For this example Vin max = min (min (V_{DD}, V_{DDA},V_{DDIO2}, V_{DDUSB}, V_{LCD}) + 3.6 V, 5.5 V) = 5.5 V.

Figure 23. Excitation pulse managed by GPIOs - Example with T_{excit} to V_{SS} = 250 ns

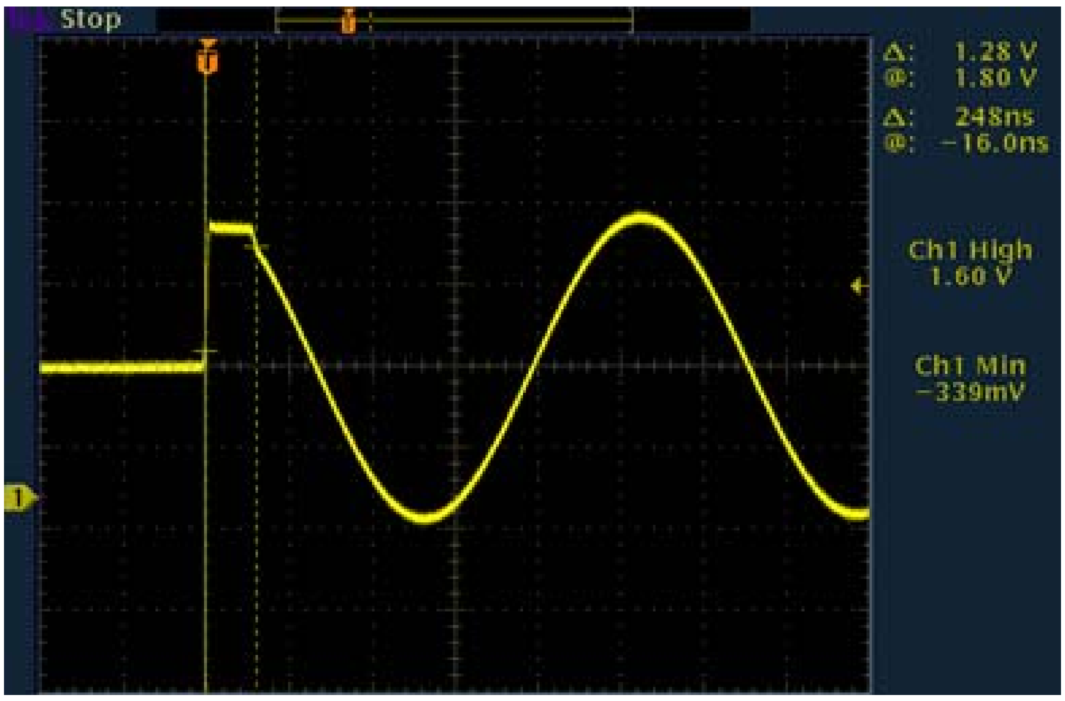
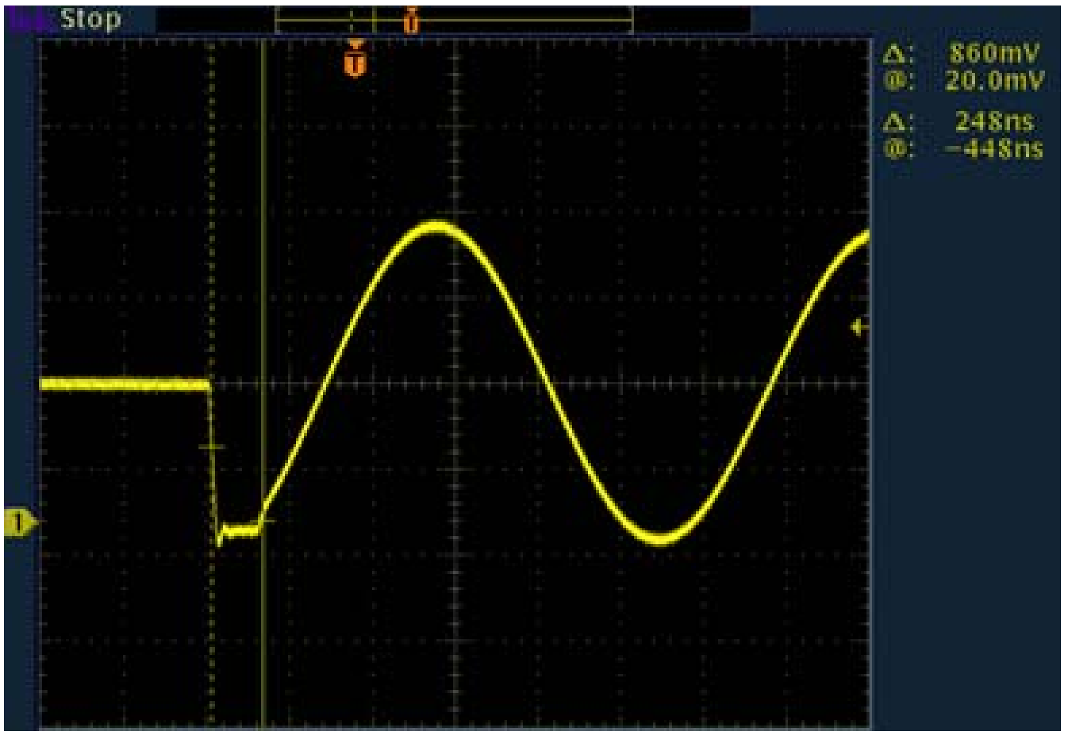


Figure 24. Excitation pulse managed by GPIOs - Example with T_{excit} to V_{DD} = 250 ns



Excitation is generated by assembler code and the size of excitation pulse is

$T_{excit\_to\ program} = T_{excit\_seconds}$ * (Freq_MSI) / (Instructions Cycles)

For this example $T_{excit\_to\ program}$ = 250 ns * 24 MHz / 3 = 2

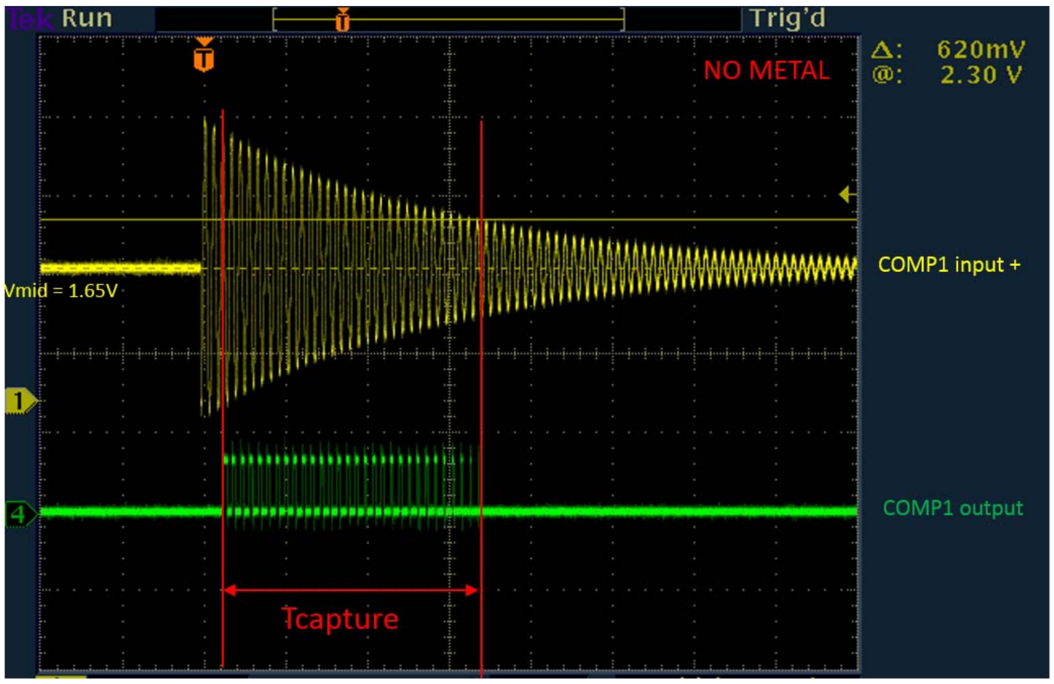
Capture window: $T_{capture}$

The collection of LC sensor oscillation and pulse count is performed during this time, managed with timer peripheral (TIM6) configured in One Pulse mode. The HCLK frequency is decreased during sleep (divided by 64) to save power consumption.

$T_{capture\_to\ program} = T_{capture\_seconds}$ * (Freq_MSI) / 64

For this example (see *Figure 22*) $T_{capture\_to\ program}$ = 50 μs * 24 MHz / 64 = 19.

Figure 25. $T_{capture}$ example



To reduce power consumption, the capture window can be minimized and adjusted to the maximum count number (without metal).

$T_{capture\_seconds}$ = (CountMax / Freq_Osc) * (1+ x), where x is the margin in percent (default value is 5%).

The theoretical oscillation frequency is determined as

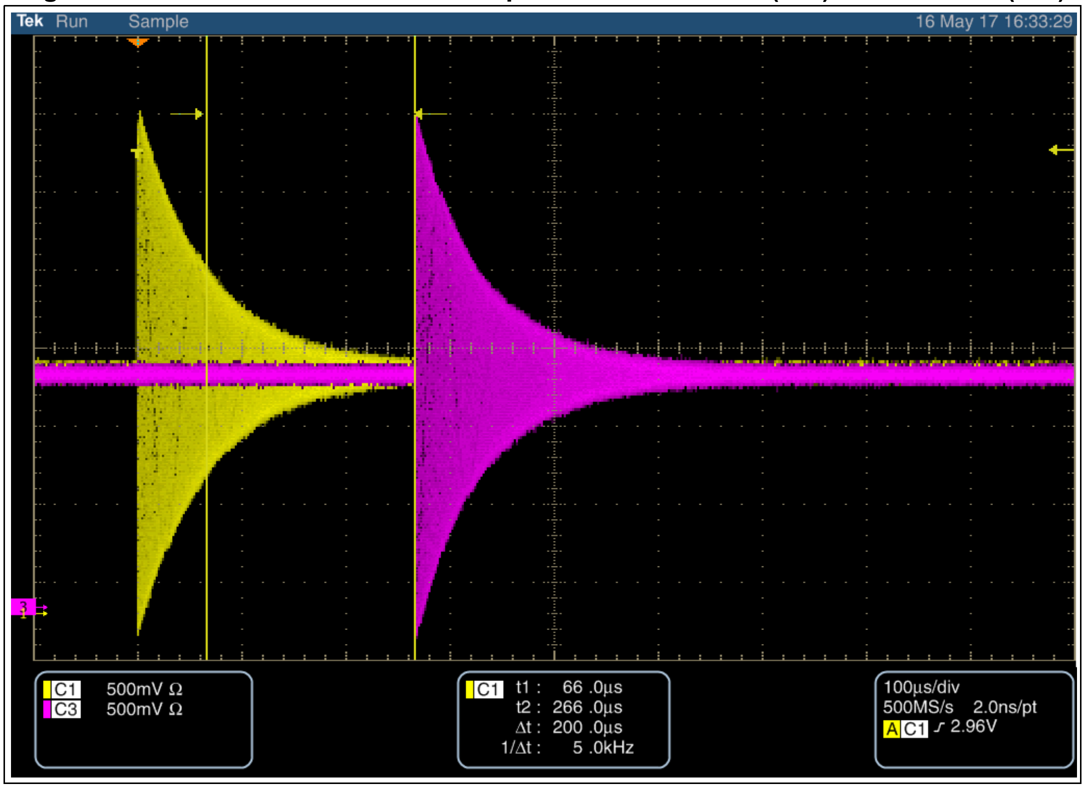
$$\text{Freqosc} = \frac{1}{2 \cdot \pi \cdot \sqrt{\text{Ls} \cdot \text{Cs}}}$$

In this example, Ls = 470 µH and Cs = 220 pF. The oscillations frequency is approximately 495 kHz.

Time between measurements

Measurements must be done sequentially to avoid cross-effects between the 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.

Figure 26. Time between sensors: 200 µs between Sensor1 (IO2) and Sensor3 (IO4)



TimeBetweenSensor is managed with timer peripheral (TIM6) configured in One Pulse mode. The HCLK frequency is decreased during sleep (divided by 64) to reduce power consumption.

$\text{TimeBetweenSensor}_{\text{to program}} = \text{TimeBetweenSensor}_{\text{seconds}} * (\text{Freq\_MSI}) / 64$

For this example $\text{TimeBetweenSensor}_{\text{to program}} = 200\ \mu s * 24\ \text{MHz} / 64 = 75$

Using up to four LC detectors

In the LC sensor detection system based on the STM32L4 microcontroller, it is possible to extend the number of LC networks, as already seen in *Figure 19*, and summarized in *Table 2*.

- One sensor

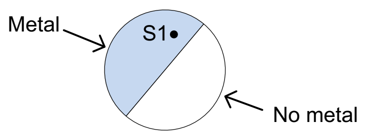
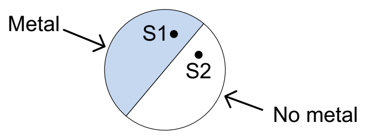
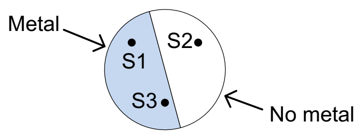
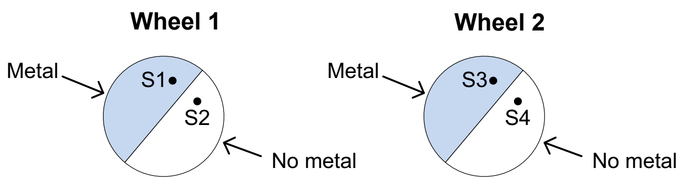
  This configuration makes it possible only to detect the metal presence or the wheel rotation
  - Two increments by rotation
- Two sensors

  This configuration makes it possible to detect both the rotation and the direction of the wheel
  - Sensors are placed around the wheel and spaced 90°
  - Up to four increments by rotation
  - Clockwise and anticlockwise counter
- Three sensors

  This configuration results in a higher precision (eight increments per rotation) and prevents some errors that cannot be detected with the two sensors system.
  - Sensors are placed around the wheel and spaced 120°
  - Up to eight increments per rotation
  - Needs two comparators and two low power timers.
- Four sensors

  This configuration is based on two wheels with two sensors per wheel, and features rotation and direction detection capability.
  - Sensors are placed on two wheels and are spaced 90°
  - Two comparators and two low power timers are required

Table 2. Sensor configurations supported by STM32L476RG-NUCLEO board

Configuration		Schematic
One sensor	One wheel	
Two sensors(1)	One wheel	
Three sensors(2)	One wheel	
Four sensors(3)	Two wheels	 Wheel 1 Wheel 2 Metal S1 S2 No metal Metal S3 S4 No metal

1. S1 and S2 are radially spaced 90°.
2. S1, S2 and S3 are radially spaced 120° from each other.
3. S1 and S2 are radially spaced 90°, and the same is true for S3 and S4.

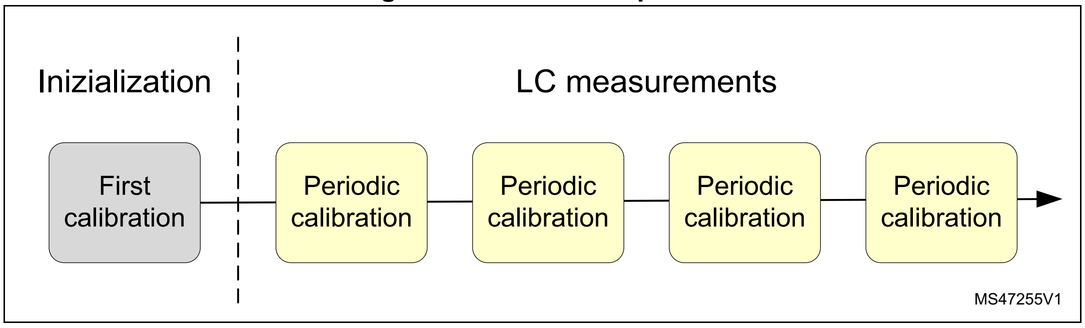
4.4.4 Calibration

The examples described in this section detail a method to calibrate the system and compensate LC sensors deviations. These examples must be tuned depending upon the application requirements.

In this demonstration, the system is calibrated in two steps:

- First calibration: performed during the system start (or factory calibration)
- Periodic calibration: performed during normal operation to compensate for small drifts due to external conditions modifications (temperature, humidity, voltage). Measurements are still carried out during this phase, so only small drifts can be compensated.

Figure 27. Calibration phases



For each calibration phase, two examples are presented, depending upon the application:

- Static method for basic detections: the calibration is done without metal in proximity of the LC sensor (air).
- Dynamic method, typically with rotating wheel: the calibration is done with metal and no metal transitions.

First calibration

The aim of this calibration is to adjust 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 (see *Section 5.2* for more details).

Static method

This calibration is done without metal and based on the maximum pulses count number.

The predefined pulses count number must be defined carefully:

- A small number involves a small count difference between metal and no metal state. The detection threshold is near maximum and minimum values, and instabilities can appear.
- A number too large involves MCU Run state and power consumption increases. Moreover, noise can appear with small oscillations amplitude.

The following parameters are automatically defined (by software) during this calibration:

- Vcmp

  The comparator threshold level (Vcmp) is adjusted to have a predefined maximum oscillations count number without metal in the proximity of LC sensors.
- CountDetect = f(CountMax)

  The detection threshold (metal / no metal) is based only on the maximum oscillations number: CountDetect = CountMax * x, where x = value in percent
- Tcapture= f(CountMax)

  To save power consumption, the capture time window is minimized and adjusted to the maximum pulses count number: Tcapture = (CountMax * PeriodOsc) * (1 + x), where
  – PeriodOsc = 2 π (Ls * Cs)$^{1/2}$
  – Ls = Sensor inductor value
  – Cs = Sensor capacitor value
  – x = margin in percent

Example

- Comparator threshold level setting (Vcmp)

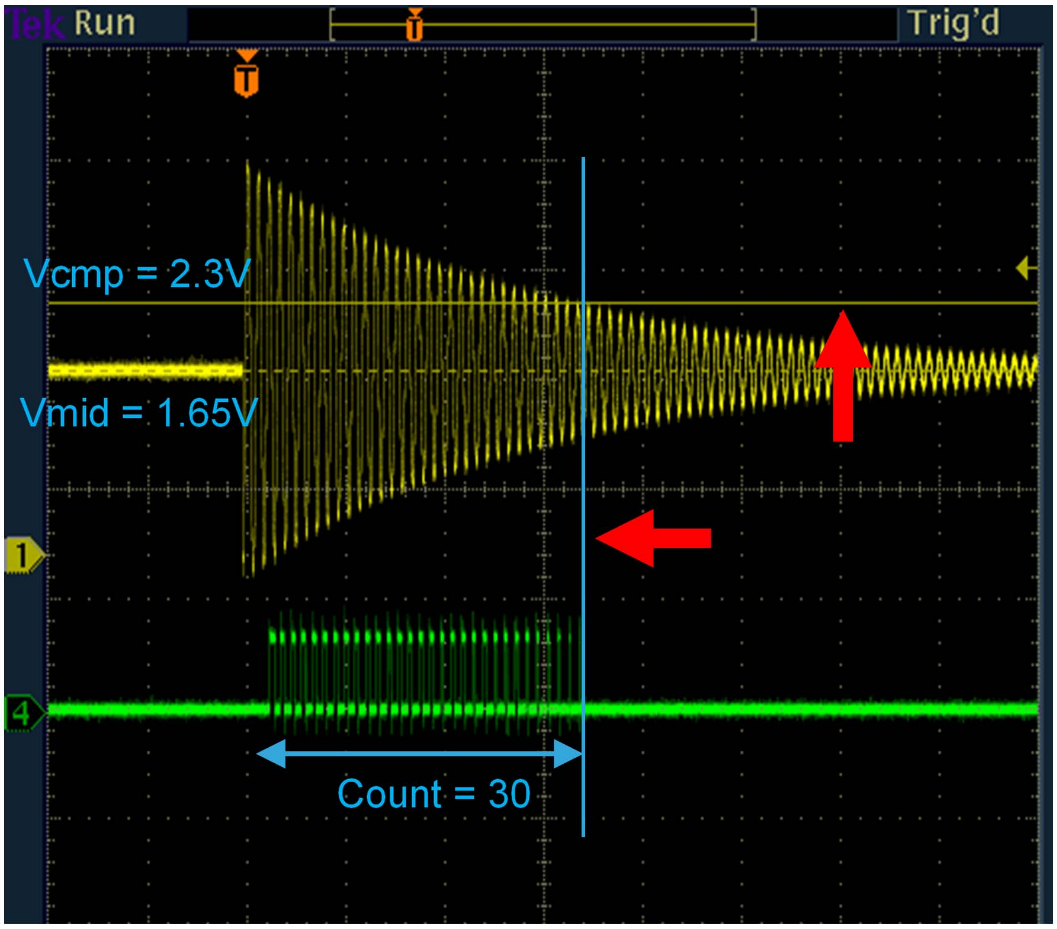
  Vcmp is increased from 1.9 to 3 V. For each step, the CountMax value is compared to the targeted pulse count number, increments are stopped when it is reached.

  In the example (see *Figure 28*) PulseCount = 30 when Vcmp = 2.3 V.
- Detection threshold count number

  CountDetect = CountMax * x = 24 (for example x = 80%)

  Tcapture (CountMax = 30, Ls = 470 μH, Cs = 220 pF, x = 5%) ≈ 64 μs

Figure 28. First calibration example for a predefined maximum count number of 30



Dynamic method

This calibration is done with metal and no metal states to have, respectively, minimum and maximum pulses count numbers.

The following parameters are automatically defined during this calibration:

- Vcmp

  The comparator threshold is incremented to find the best ratio between the minimum (metal) and the maximum (air) pulse count number. The goal is to have a minimum difference (Δ) in the pulse count to reach a satisfactory compromise between stability and power consumption.

- CountDetect = f(CountMax, CountMin)

  The detection threshold (metal / no metal) is based on the both maximum and minimum pulse count numbers.
- CountDetec = (CountMax + CountMin) / 2
- Tcapture= f(CountMax)

  To save power consumption, the capture time window is minimized to be adjusted to maximum pulse count number: Tcapture = (CountMax * PeriodOsc) * (1 + x), where
  - PeriodOsc = 2 π (Ls * Cs)$^{1/2}$
  - Ls = Sensor inductor value
  - Cs = Sensor capacitor value
  - x = margin in percent

Example

- Comparator threshold level (Vcmp)

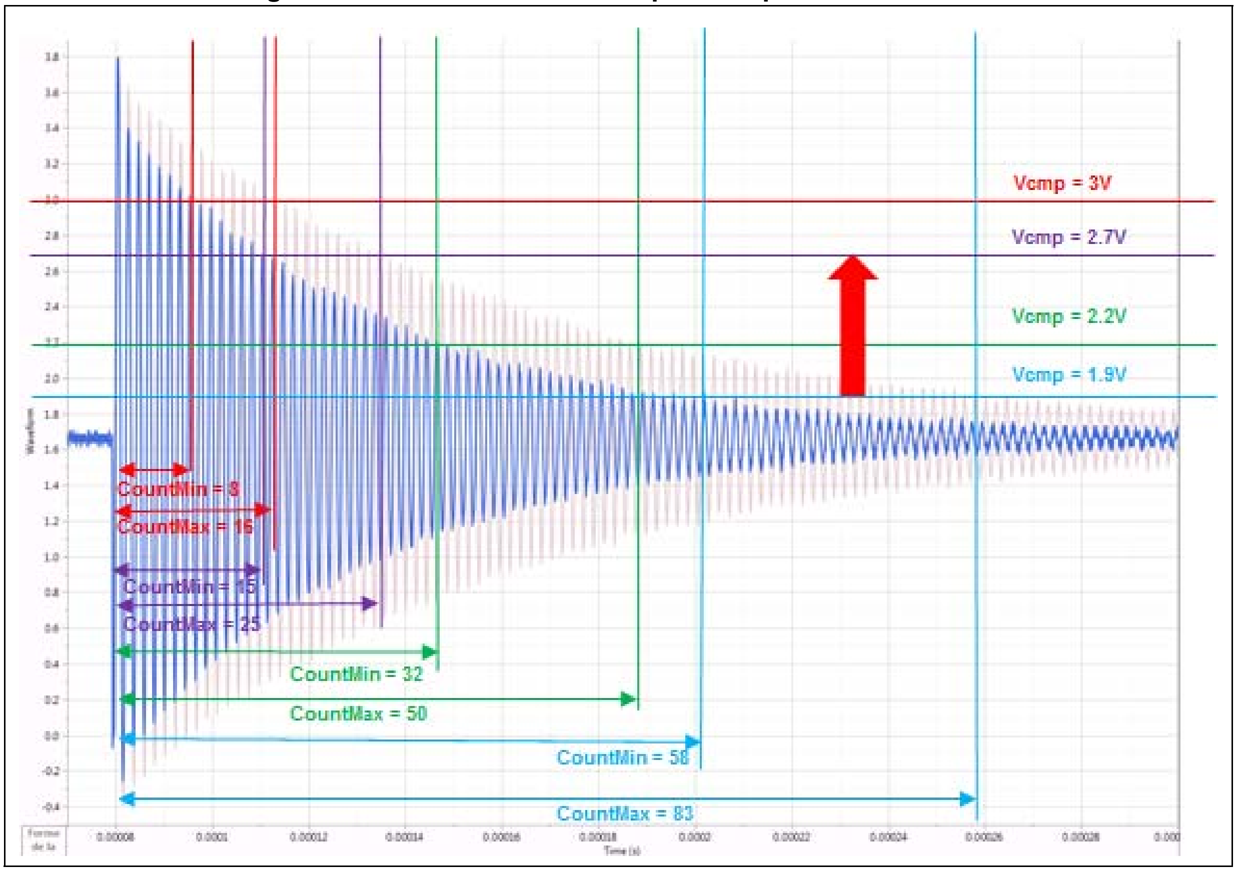
  Vcmp is increased from 1.9 to 3 V. For each step, the difference between CountMax and CountMin is computed. Increments are stopped when the minimum Δ to guarantee is reached.

  In this example (see *Figure 29*) PulseCountDeltaMin = CountMax - CountMin = 10 and Vcmp = 2.7 V.

  Detection threshold count number (CountMax = 25, CountMin = 15) CountDetec = 20

  Tcapture (CountMax = 25, Ls = 470 µH, Cs = 220 pF, x = 5%) ≈ 53 µs

Figure 29. First calibration example for a predefined Δ of 10



Periodic calibration

This makes possible to compensate small drifts due to external variations, such as temperature, voltage, humidity. During periodic calibrations, measurements are still ongoing.

The detection threshold is adjusted based on the calibration results (details are described in *HAL_RTCEx_AlarmBEventCallback () - Periodic calibration function*).

Static method

This calibration is done without metal in proximity of LC sensor. Only the maximum pulses count number is used. To check that calibration results are valid, no transitions should appear during this phase (if this requirement is not met calibration results must be ignored).

To eliminate noisy results that could appear, a maximum drift is allowed:

- CountMax - CountDrift < Calibration CountMax result < CountMax + CountDrift

The following parameters are automatically defined during this calibration:

- CountDetect = f(CountMax), same as in *Static method* of *First calibration*
- Tcapture = f(CountMax), same as in *Static method* of *First calibration*

Dynamic method

This calibration is done with a rotating wheel. Maximum and minimum pulses count number are used, so it is mandatory to have several transitions during this calibration phase. If not, calibration results must be ignored.

To eliminate noisy results that could appear, maximum drifts are allowed:

- CountMax - CountDrift < Calibration CountMax result < CountMax + CountDrift
- CountMin - CountDrift < Calibration CountMax result < CountMin + CountDrift

The following parameters are automatically defined during this calibration:

- CountDetect = f(CountMax), same as in *Dynamic method* of *First calibration*
- Tcapture = f(CountMax), same as in *Dynamic method* of *First calibration*

4.5 Getting started with the demonstration

There are four demonstrations available:

- Demo1: Basic counting demonstration with one sensor to use with the detection accessory board MB1199 or other metal accessory.
- Demo2: Counting demonstration with two sensors: rotation/direction detections to use with a rotating wheel.
- Demo3: Tachometer demonstration with two sensors: rotation/direction detections to use with a rotating wheel.
- Demo4: Counting demonstration with two by two LC sensors: rotation/direction detections to use with a rotating wheel.

Each demonstration can be run in standard mode (for debug) or in low power mode (for the final application and to perform consumption measurements).

Counter informations are sent periodically on USART2 port and can be displayed with UART terminal software on the host PC connected to the USB cable.

Warning: Thanks to the user button (Blue), the user can switch ON or OFF the USART2 communication to read LC sensors results (enabled by default). Before performing consumption measurements it is mandatory to disable the USART2 communication.

Terminal software configuration

- Port: STMicroelectronics STLink Virtual COM port
- Baud rate: 921600
- Data rate: 8 bit
- Parity: none
- Stop: 1 bit
- Flow control: none

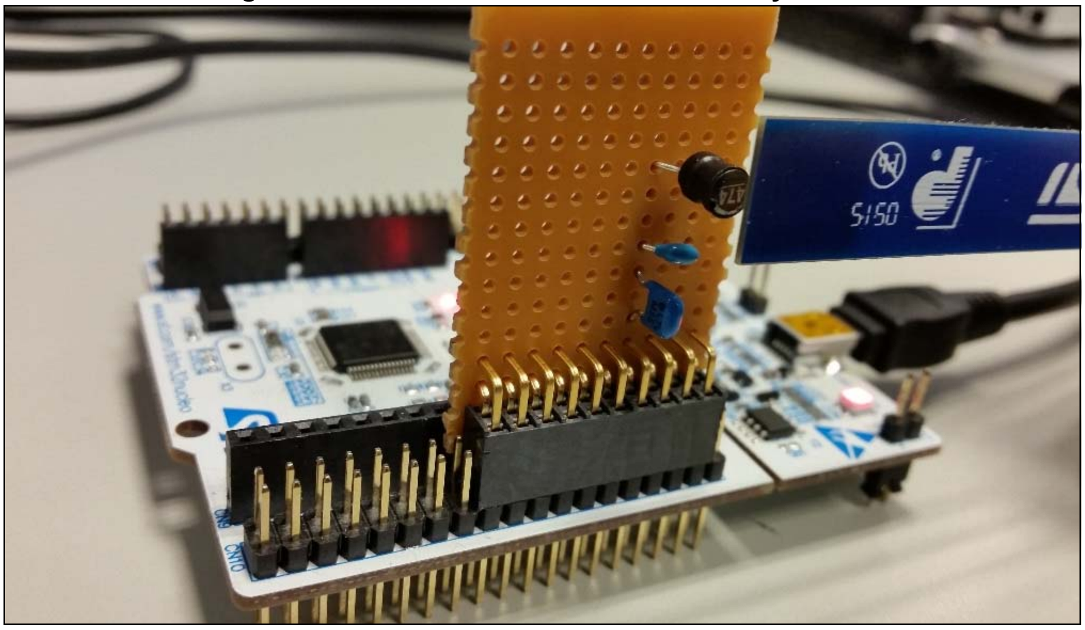
Note: *To connect debugger and store the software in memory, it could be necessary to press the reset button (Black) at the beginning of the connection. This is due to debug pins (SWD PA13-PA14) put in analog mode by the previous low power software execution.*

4.5.1 LC sensor metering basic counting demonstration with one sensor

This demonstration uses one sensor, IO1 and IO2 (one IO is needed for each sensor and another, DAC output, to supply all the sensors). This basic demonstration allows to check the presence of metal.

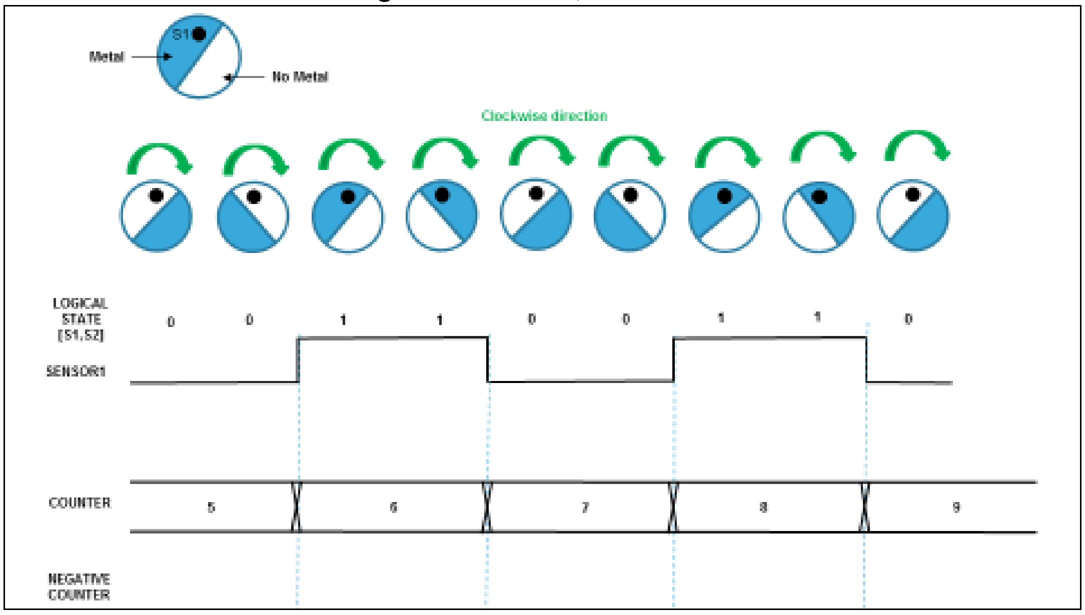
When using the detection accessory board MB1199, the metal strip must be positioned over the LC sensor as shown in *Figure 30*.

Figure 30. LC sensor with MB1199 accessory board



When using one wheel, this basic demonstration makes it possible to collect the number of turns with two increments for each rotation, as indicated in *Figure 31*.

Figure 31. Demo1, one sensor



The counter is updated at each metal / no metal transition, but it is not possible to differentiate between clockwise and anticlockwise rotations.

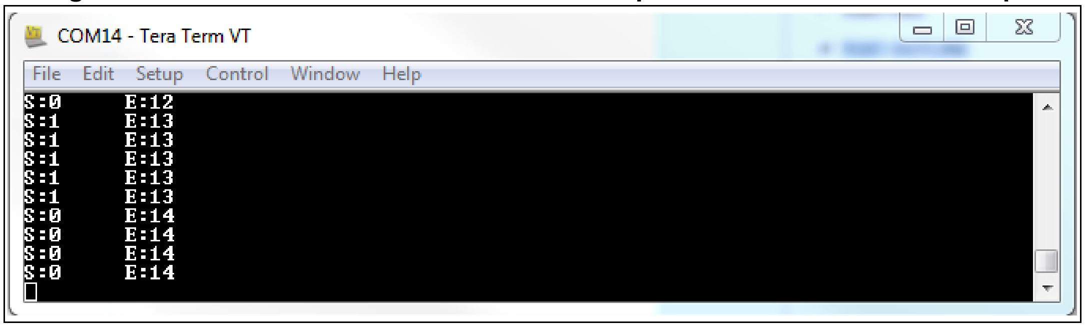
FW configuration

To enable this demonstration, LC_SENSOR_DEMO 1 define must be selected in lc_sensor_metering.h file.

LcConfig and LcConfigSensorX variables, set in LcSensorConfig () function, must be defined according to application (sampling rate, DAC refresh, V_{cmp}, T_{excit}, $T_{capture}$).

Each two seconds, the transitions number (LcStatus.EdgeCount) and the current sensor status (LcSensor1.status) are updated.

Figure 32. Demo1 - USART communication output - Terminal software example



```
S:0     E:12
S:1     E:13
S:1     E:13
S:1     E:13
S:1     E:13
S:1     E:13
S:0     E:14
S:0     E:14
S:0     E:14
S:0     E:14
```

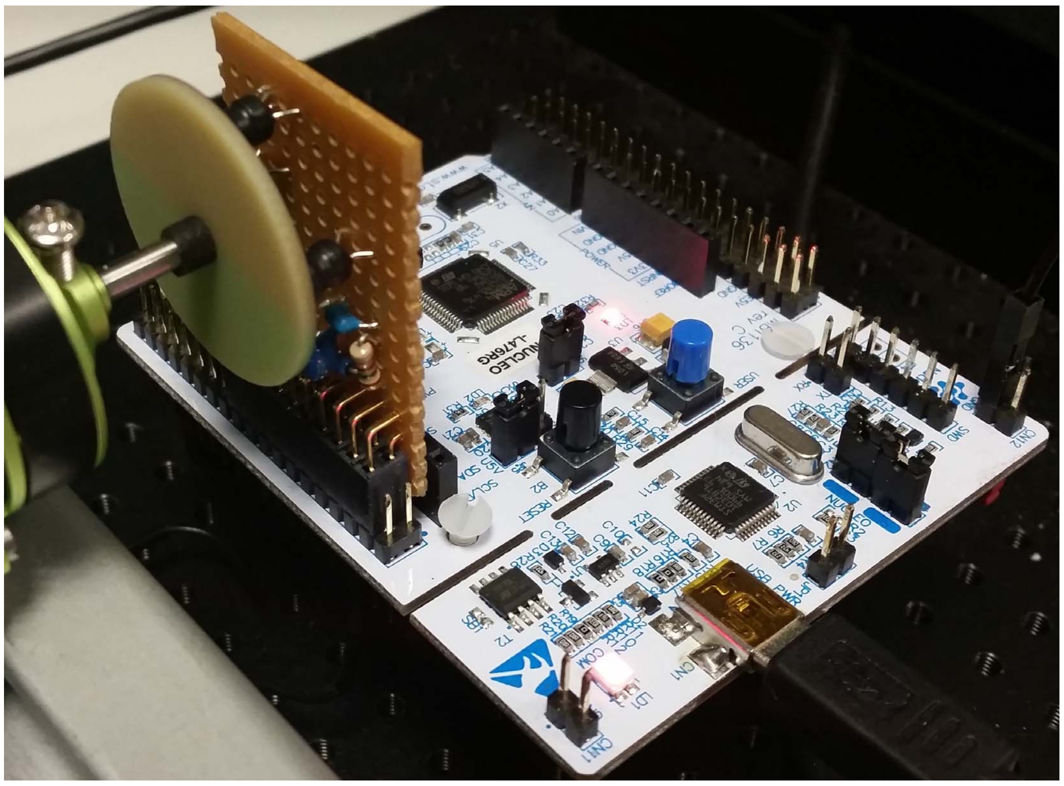
In *Figure 32* S is Sensor1 status (0 = no metal, 1 = metal), and E indicates count number.

4.5.2 LC sensor metering counting demonstration with two sensors

This demonstration (*Figure 33*) uses two LC sensors (IO1, IO2 and IO3).

If the calibration is disabled or in static mode, no metal must be present over the LC sensors during start of the system. In the case of dynamic calibration selection, the wheel must be in rotation.

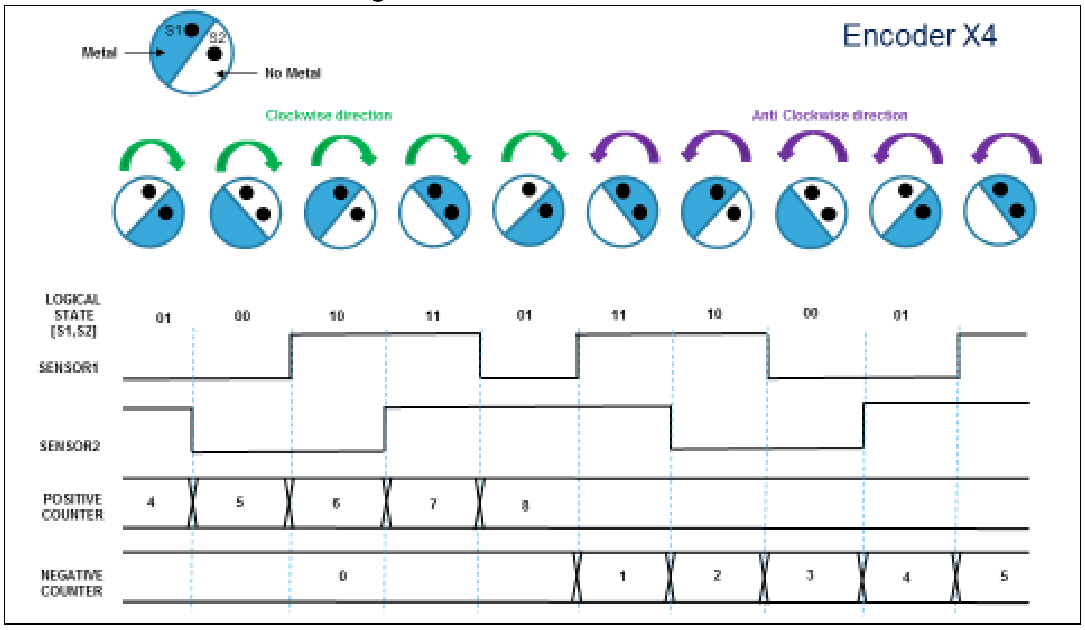
Figure 33. Hardware for the two sensors demonstration



It is possible to determine both the wheel rotation and its direction. Two counters are updated, one for the clockwise rotation and the other one for the anticlockwise rotation.

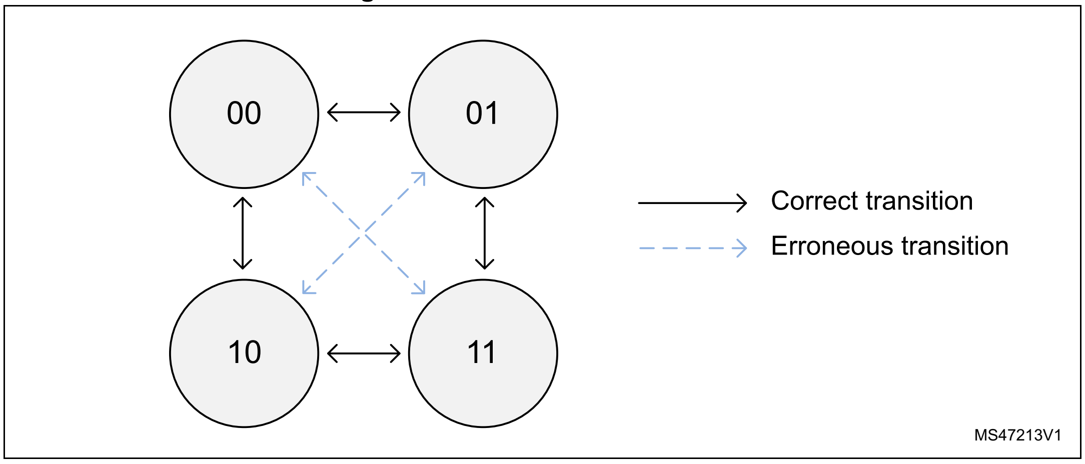
- Rotation and direction detection
- Clockwise counter and anticlockwise counter (negative or positive variation defined by previous sensor state)
- Four increments by rotation
- Sensors are spaced 90°

Figure 34. Demo2, two sensors



To interpret the results, a basic 4-state machine (*Figure 35*) has been put in place.

Figure 35. Basic 4-state machine



Erroneous transitions are detected with the LcStatus.Errors variable.

FW configuration

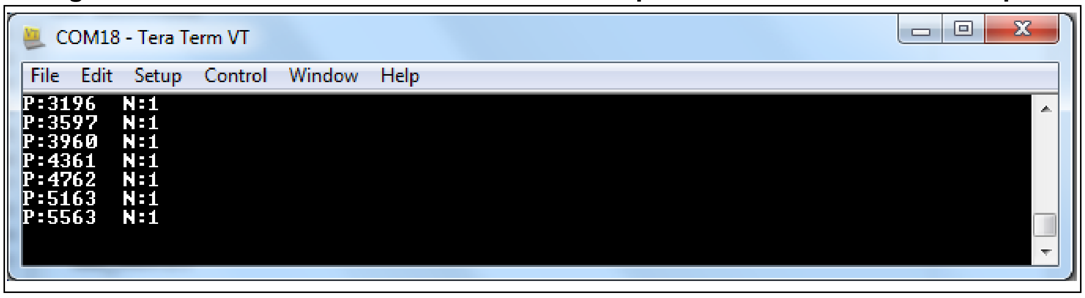
To enable this demonstration, LC_SENSOR_DEMO 2 define must be selected in the lc_sensor_metering.h file.

LcConfig and LcConfigSensorX variables, set in LcSensorConfig () function, must be defined according to application (sampling rate, DAC refresh, V_{cmp}, T_{excit}, $T_{capture}$, TimeBetweenSensors).

Outputs: based on X4 encoder result, two counters are updated with four increments per rotation:

- LcStatus.EdgeCountPos: Positive counter
- LcStatus.EdgeCountNeg: Negative counter

Figure 36. Demo2 - USART communication output - Terminal software example



```
P:3196  N:1
P:3597  N:1
P:3960  N:1
P:4361  N:1
P:4762  N:1
P:5163  N:1
P:5563  N:1
```

In *Figure 36*, P stands for positive counter and N for negative counter.

4.5.3 LC sensor metering tachometer demonstration with two sensors

This demonstration uses two LC sensors, IO1, IO2 and IO3, and the same configuration of Demo2. It allows to collect the rotation speed and direction. For a clockwise direction the rotation per minute (RPM) number is positive, it is negative for anticlockwise rotation.

If the calibration is disabled or in static mode, no metal must be present over the LC sensors during start of the system. In the case of dynamic calibration selection, the wheel must be in rotation.

FW configuration

To enable this demonstration, LC_SENSOR_DEMO 3 define must be selected in lc_sensor_metering.h file.

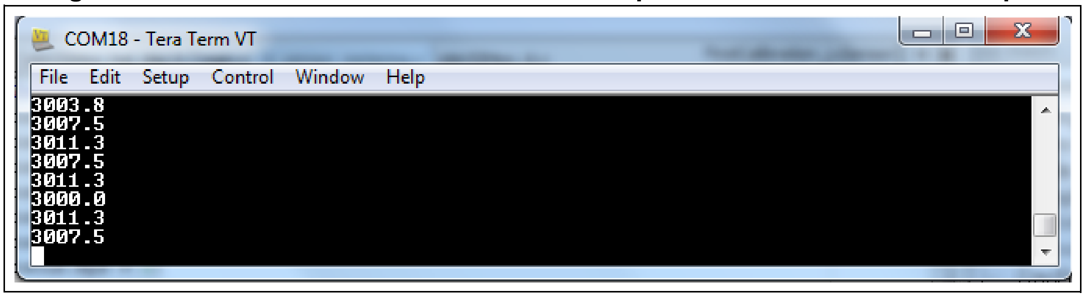
LcConfig and LcConfigSensorX variables, set in LcSensorConfig () function, must be defined according to application (sampling rate, DAC refresh, V_{cmp}, T_{excit}, $T_{capture}$, TimeBetweenSensors).

Outputs

The rotation per minute value is updated:

- LcStatus.RPM: Rotation per minutes

Figure 37. Demo3 - USART communication output - Terminal software example



```
3003.8
3007.5
3011.3
3007.5
3011.3
3000.0
3011.3
3007.5
```

4.5.4 LC sensor metering counting demonstration with four sensors

This demonstration uses four (two by two) LC sensors, IO1, IO2, IO3, IO4 and IO5. It is possible to know both the rotation and the direction of two wheels. It is the duplication of Demo2 configuration.

If the calibration is disabled or in static mode, no metal must be present over the LC sensors during start of the system. In the case of dynamic calibration selection, the wheel must be in rotation.

To interpret results, the basic 4-state machine used for Demo2 has been duplicated. Four counters are updated.

FW configuration

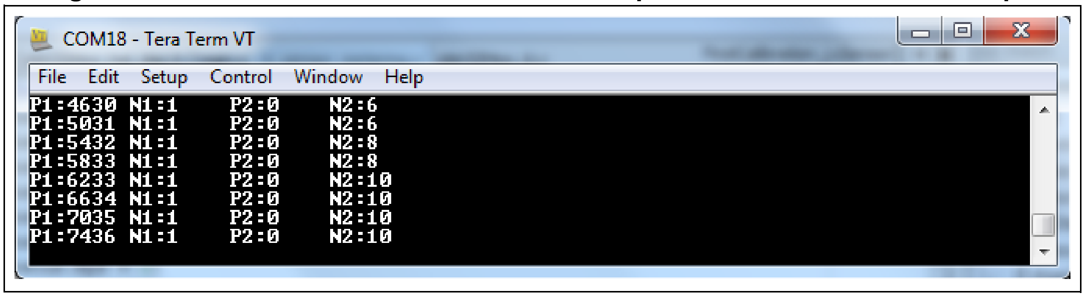
To enable this demonstration, LC_SENSOR_DEMO 4 define must be selected in lc_sensor_metering.h file.

LcConfig and LcConfigSensorX variables, set in LcSensorConfig () function, must be defined according to application (sampling rate, DAC refresh, V_{cmp}, T_{excit}, $T_{capture}$, TimeBetweenSensors).

Outputs: four counters are updated with four increments per rotation

- LcStatus.EdgeCountPos: Wheel 1 positive counter
- LcStatus.EdgeCountNeg: Wheel 1 negative counter
- LcStatus.EdgeCountPos2: Wheel 2 positive counter
- LcStatus.EdgeCountNeg2: Wheel 2 negative counter

Figure 38. Demo4 - USART communication output - Terminal software example



```
P1:4630 N1:1     P2:0     N2:6
P1:5031 N1:1     P2:0     N2:6
P1:5432 N1:1     P2:0     N2:8
P1:5833 N1:1     P2:0     N2:8
P1:6233 N1:1     P2:0     N2:10
P1:6634 N1:1     P2:0     N2:10
P1:7035 N1:1     P2:0     N2:10
P1:7436 N1:1     P2:0     N2:10
```

In *Figure 38*, P1 and N1 refer to Wheel 1 (positive and negative counters, respectively), and P2 and N2 to Wheel 2 (positive and negative counters, respectively).

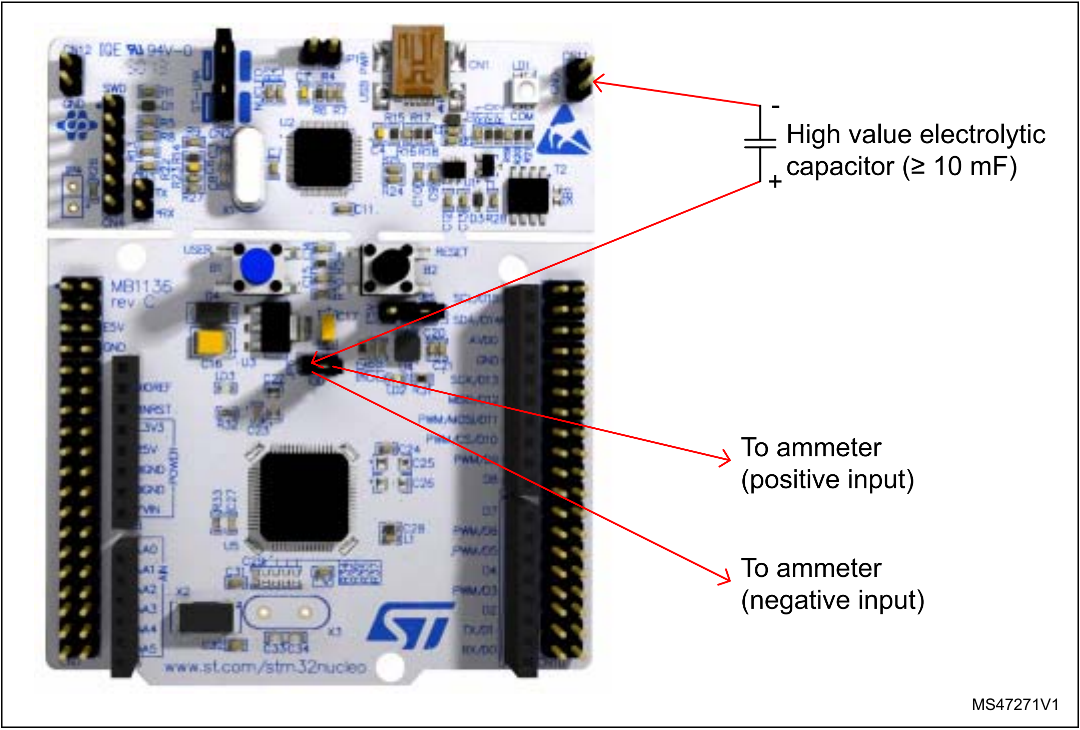
4.6 LC sensor metering Mode Low Power

To perform consumption measurements, it is necessary to check that application is running in low power mode:

- LC_SENSOR_LOW_POWER mode (enabled by default), selection is done by SW
- USART communication is disabled (enabled by default), can be switched off with blue user button (B1) or can be disabled by SW

To measure the average current during the LC sensor metering in low-power mode (a few µA), an ammeter can be inserted on JP6 between pin 1 (VDD_MCU) and pin 2 (VDD 3.3V). Special attention must be taken to measure pulsed current (see *Section 2.5.2*).

Figure 39. Connections for the measurement of power consumption


-
+
High value electrolytic capacitor (≥ 10 mF)
To ammeter (positive input)
To ammeter (negative input)
MS47271V1

5 Firmware description - STM32L476RG-NUCLEO

5.1 STM32L476x peripherals used by the application

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

GPIOs

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

- PA5 or PA4 as V_{mid} generator (DAC output) at V_{DD} / 2 reference voltage (IO1)
- PC5, PB2, PB4 and PC6 are alternatively used as output GPIO to drive the LC sensors and put in alternate function (positive comparator inputs COMP1 and COMP2) once the excitation pulse is delivered
- PC5 to drive the LC sensor 1 (IO2)
- PB2 to drive the LC sensor 2 (IO3)
- PB4 to drive the LC sensor 3 (IO4)
- PB6 to drive the LC sensor 4 (IO5)

DAC

DAC_OUT1 and DAC_OUT2 are used to generate

- V_{DD} / 2 reference voltage (V_{mid})
- The input threshold voltage needed by the comparator. This reference voltage allows to provide a preset threshold voltage value near V_{DD} / 2 but with a necessary positive margin to eliminate noise. It allows to calibrate the threshold value according to different LC network responses and environment variations.

DAC outputs are configured in Sample and Hold mode and enabled/disabled in RTC_WakeUp interrupt.

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 and this results in digital pulses delivered on the comparator output. This signal will be then transmitted to the LPTIM.

COMP1 is enabled only if sensor 1 or sensor 2 are used.

COMP2 is enabled only if sensor 3 or sensor 4 are used.

Comparators are connected to the LC sensor network.

LPTIM1 and/or LPTIM2

LP 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. This avoids a time consuming LPTIM reset.

LPTIM1 is enabled only if COMP1 is enabled.

LPTIM2 is enabled only if COMP2 is enabled.

RTC

This peripheral is used to wake up the microcontroller at regular intervals once it has entered Stop mode. The RTC clock is connected to the LSE clock, whose frequency is 32.768 kHz.

- The wake-up timer is used to perform LC measurements. The wake-up frequency corresponds to the LC sensor capture frequency and is set to 500 Hz for this example. This capture frequency can be increased or decreased depending upon the rotating wheel maximum speed. Modifications on the capture frequency lead to a change of the average power consumption.
- Alarm A is used to perform communications with user or others peripherals to interpret LC sensors results. For this demo, results are send by USART2 communication.
- Alarm B is used to perform periodic calibrations to compensate thermal and voltage deviations.
- Tamper1 is used to enable or disable Alarm A to reduce 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.

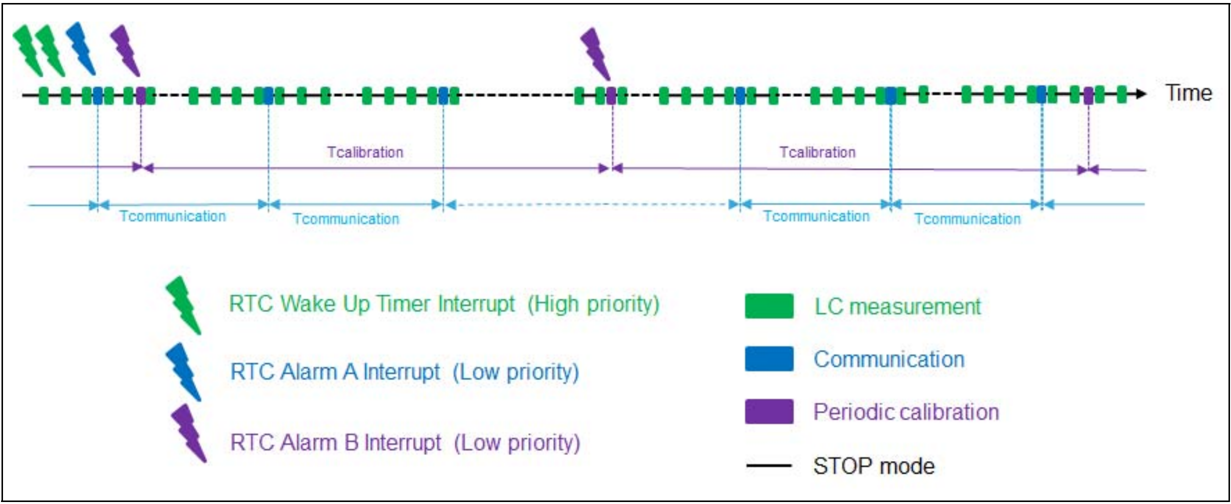
Interrupts

To save the maximum time and power consumption in MCU Run mode, the RTC wake-up timer IRQ handler is directly processed inside the interrupt subroutine.

RTC alarms interrupts are managed by alarm event call backs (see *Figure 40*).

Alarm interrupts have a lower priority compared to wake-up timer interrupts. The frequency depends upon the application, it can range from several seconds to several hours for user communications (Alarm A), and from several minutes to several hours for periodic calibration (Alarm B).

Figure 40. Interrupts profile



TAMP_STAMP_IRQ is managed by Tamper1 event call back. It allows to enable or disable AlarmA interrupt (used for communication).

LC measurements still working during communication and calibration phase. So, Alarm A and Alarm B Execution needs to be performed between two measurements (between two wake-up interrupts) and it is mandatory to control their execution time and to have a lowest priority compared to that of the wake-up timer.

5.2 Description of flows

This software demonstration allows the user to manage from one to four sensors sequentially. Hardware configuration is done in lc_sensor_metering.h file by define instructions.

Two hardware configurations have been implemented:

- BOARD_CONFIG 1 define, to manage up to four sensors
  - VMID on PA4 (DAC_OUT1)
  - COMP1 and COMP2 inverting inputs connected to DAC_OUT2
- BOARD_CONFIG 2 define, to manage up to four sensors
  - VMID on PA5 (DAC_OUT2)
  - COMP1 and COMP2 inverting inputs connected to DAC_OUT1

Four demonstrations are available:

- LC_SENSOR_DEMO 1: Basic counting demonstration with one sensor
- LC_SENSOR_DEMO 2: Counting demonstration with two sensors - Rotation and direction
- LC_SENSOR_DEMO 3: Tachometer demonstration with two sensors - Speed and direction
- LC_SENSOR_DEMO 4: Counting demonstration with two by two sensors - Rotation and direction

These examples are launched from a main.c file. All LC sensor metering functions are in lc_sensor_metering.c file and called by main.c.

Initialization

Configuration function LcSensorConfig ():

- Fills all structures required to set LC sensor configuration

Demo function LcSensorDemo ():

- Initializes all peripherals
- Starts counting demonstration

Initialization function LcSensorInit ():

- Allows to set all variables used by application.

StopEntry () function: Called in LOW_POWER mode

- Prepares MCU to enter in Stop mode

Calibration

Calibration functions:

- LC_Calibration (): First calibration performed at the power up.
- HAL_RTCEx_AlarmBEventCallback (): Periodic calibration function.

Interrupts

- RTC_WKUP_IRQHandler () function to manage LC measurements:
  - LC sensors excitation
  - Process sensor state
  - Execute state machine
  - Process periodic calibration if required.
- HAL_RTCEx_Tamper1EventCallback () function to enable and disable AlarmA by user push button (BLUE)
- HAL_RTC_AlarmAEventCallback () function to communicate with user by USART or with other peripherals.
- HAL_RTCEx_AlarmBEventCallback (): function to perform periodic calibration.

Files and functions description

Lc_sensor_metering.h

Hardware configuration

- BOARD_CONFIG 1: to manage boards with V_{mid} on PA4 pin
- BOARD_CONFIG 2: to manage boards with V_{mid} on PA5 pin

Demonstration selection

- LC_SENSOR_DEMO 1: basic counting demonstration with one sensor
- LC_SENSOR_DEMO 2: counting demonstration with two sensors and rotation/direction detection
- LC_SENSOR_DEMO 3: tachometer demonstration with two sensors and speed/direction detection
- LC_SENSOR_DEMO 4: counting demonstration with two by two sensors

USART2 communication

- USART2_ENABLE 0: disabled
- USART2_ENABLE 1: enabled

LC excitation mode selection:

- LC_EXCIT_MODE 0: LC sensor excitation to V_{SS}
- LC_EXCIT_MODE 1: LC sensor excitation to V_{DDA}

This file contains all macros used to manage LC sensor oscillations:

- LC_EXIT (): to do LC excitation (ASM code)
  - GPIO in output state
  - GPIO in analog state
- LC_WAIT (): to insert a waiting time for peripherals stabilization
  - Set TIM6 in single shot

  - Decrease HCLK during sleep
  - Start TIM6
  - Wait For Event (time to stabilize peripherals)
  - Go back to full HCLK speed
- LC_MEASURE (): to perform LC measurements
  - Set TIM6 in single shot
  - Decrease HCLK during sleep
  - Disable DAC output V_{mid}
  - Start TIM6
  - Wait For Event (time to do measurements)
  - Go back to full HCLK speed

LcSensorConfig () function

This function allows the user to configure the application.

Two structures are available to configure and manage LcSensor application:

1. LcConfig to configure application

- Mode: LC_SENSOR_STD (for debug only) or LC_SENSOR_LOW_POWER (default)
- Sampling: value in Hz (up to 500, default being 100)
- DacVmidRefreshTime: Trefresh value or 0 to disable (default)
- DacVmid (default value is 0x800)
- DacVcmp (default value is 0xC00)
- TimeBetweenSensor (default value is 75)
- FirstCal
  - Status: CAL_ENABLED or CAL_DISABLED (default)
  - Mode: STATIC or DYNAMIC (default)
  - Measures: minimum measurement number during calibration phase (default is 50)
  - DacVcmpMax: starting value for V_{cmp} calibration (default value is 0xE00)
  - DacVcmpMin: stop value for V_{cmp} calibration (default value is 0x850)
  - DacVcmpStep: step value for V_{cmp} calibration (default value is 10)
  - PulseCount: targeted pulse number (default is 20), used for STATIC calibration
  - PulseCountDeltaMin: set the difference between min and max pulse count (default is 20), used for DYNAMIC calibration
- PeriodicCal
  - Status: CAL_ENABLED or CAL_DISABLED (default)
  - Mode: STATIC or DYNAMIC (default)
  - CountDrift: Valid periodic calibration values only if the drift does not exceed this value (default is 5)

2. LcConfigSensorX to configure sensors (X = 1, 2, 3, or 4)

- Ls: Ls value in H (default value is 470 µH)
- Cs: Cs value in F (default value is 220 pF)
- CountDetectPercent: detection threshold in percent, not used if dynamic calibration is enabled

- Texcit: excitation time, minimum value is 1 (default value is 2)
- $T_{capture}$: capture time (default value is 19), value updated automatically when calibration is enabled

LcSensorInit () function

This function allows the user to initialize structures used by application with default values. It must be called before Wake-up interrupt enabling (called in LcSensorDemo and StopEntry functions if low power mode is enabled).

Two structures are used to store results updated at each measurements (sensors status, counter status, errors …).

1. LcSensorX to store individual sensor results
   - Status: NO_METAL (0) or METAL (1)
   - PreviousStatus: NO_METAL (0) or METAL (1)
2. LcStatus to store LC application status

- MeasuresCount: measurements counter
- EdgeCountPos: positive edge counter
- EdgeCountNeg: negative edge counter
- Errors: error counter
- Rpm: rotations per minute
- Rps: rotations per seconds
- WakeUpCounter: counter value loaded in RTC wake-up timer
- Sampling: the real sampling value based on WakeUpCounter value

One structure to store GPIOs state for a fast I/Os modifications during LC sensors excitation. It allows to minimize instructions number.

- LcIO store and modify GPIOs registers
  - LcIO.IOx_GPIO_GROUP: Group of I/O used
  - LcIO.IOx_OUTPUT_PP: pin in output state in Push Pull
  - LcIO.IOx_ANALOG: pin in analog state

LcSensorDemo () function

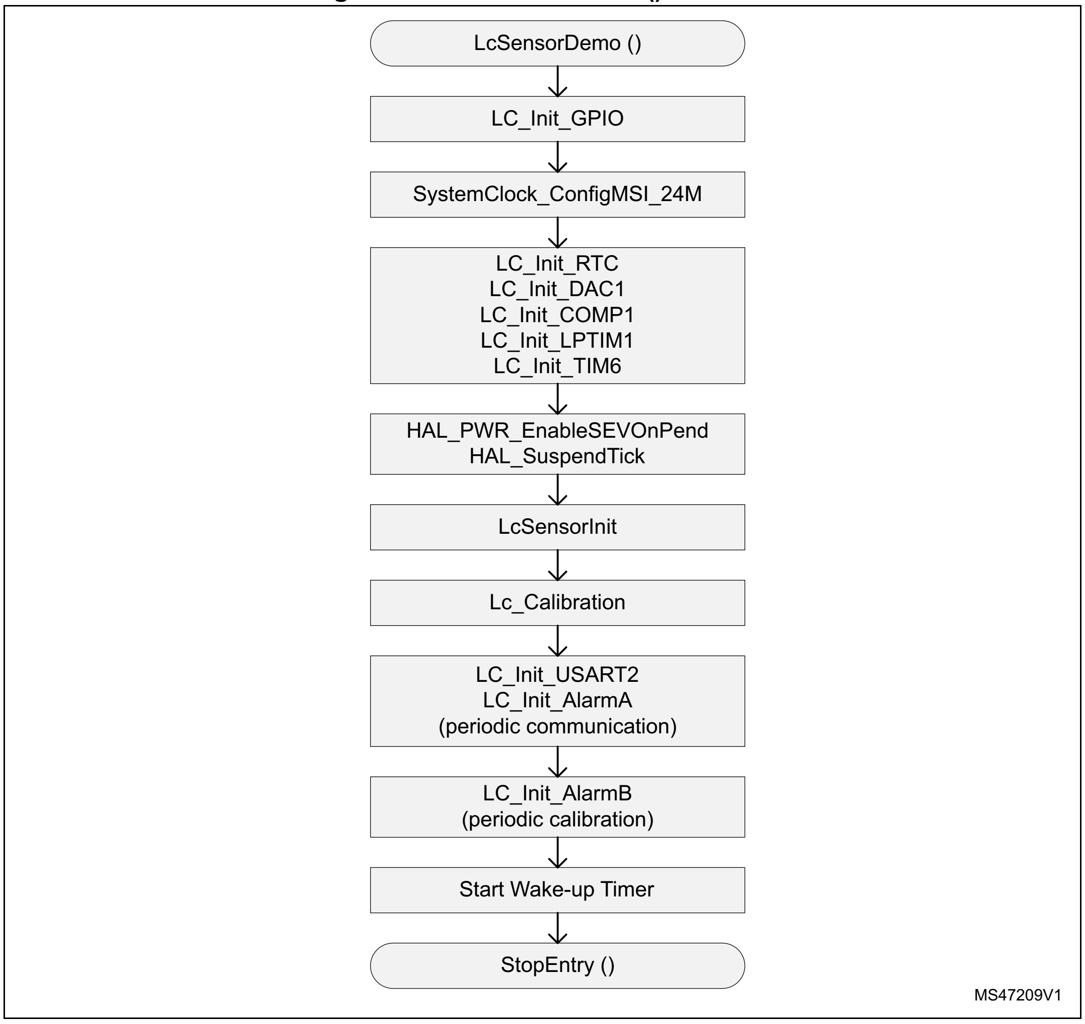
As soon as this function is fully executed, an RTC interrupt is created to wake up the MCU at regular intervals. This function is then called once and the application will later alternate Stop sequences and Run slots at the frequency rate given by the RTC wake-up parameters. In this example, the microcontroller is woken up 500 times per second.

GPIOs are initialized and system clock is configured.

RTC, DAC, LPTIM, TIM and COMP peripherals are initialized with the settings described in *Section 5.1*.

In this example, USART and RTC Alarm A are initialized for periodic communication.RTC Alarm B is initialized if periodic calibration is enabled.Then, the StopEntry() function is executed to prepare the device to enter Stop mode with specific settings and the microcontroller placed in an infinite loop waiting for the RTC wake up interrupt. All processes are indeed performed in interrupt, using sleep on exit mode, thus nothing to do in main. User can exit the Low Power demonstration applying a reset sequence (pressing the B2 (RESET) button).

Figure 41. LcSensorDemo () function



StopEntry () function

The aim of this function is to prepare the device to enter Stop2 mode with parameters set to minimize the device power consumption.

All GPIOs are placed in analog state to minimize power consumption and the associated clocks are disabled.

SysTick timer is disabled to decrease further dynamic consumption.

Voltage scaling is set to RANGE2 to optimize the power consumption.

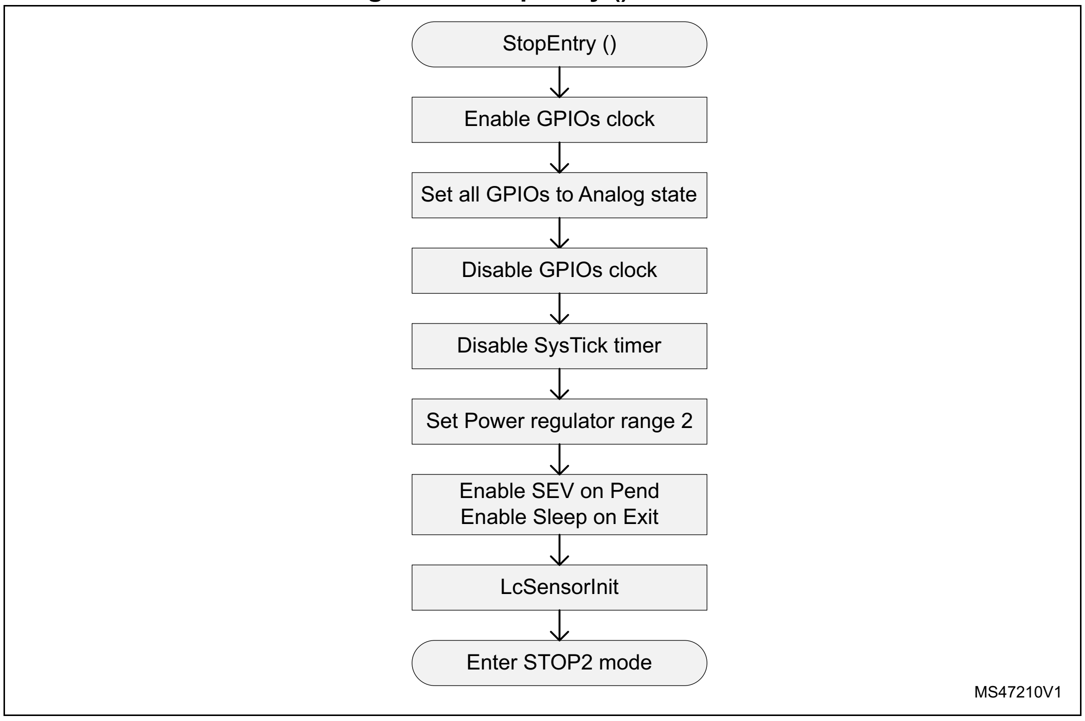
Set SEVONPEND bit, this causes WFE to wake up when an interrupt moves from inactive to pending (to catch TIM6 event).

Sleep on exit is used to force the MCU entering the Stop mode as soon as the RTC wake-up has been serviced and executed. In that case, the StopEntry () function is executed once and the microcontroller returns in Stop mode after each interrupt.

Lc variables are initialized with LcSensorInit ().

Enter in Stop2 mode and wait for interrupt.

Figure 42. StopEntry () function



RTC_WKUP_IRQHandler () interrupt subroutine

The RTC Wake Up IRQ Handler manages the RTC interrupt triggered by the RTC wake-up timer. To save time during the interrupt execution, and since the device operates in Run mode, the interrupt subroutine is executed in the Low Power demonstration mode without performing any call back function in the main program.

The interrupt flag is first cleared.

Then COMP2 and DAC peripherals are switched on and a delay is inserted to wait the peripherals stabilization times before they can be operational.

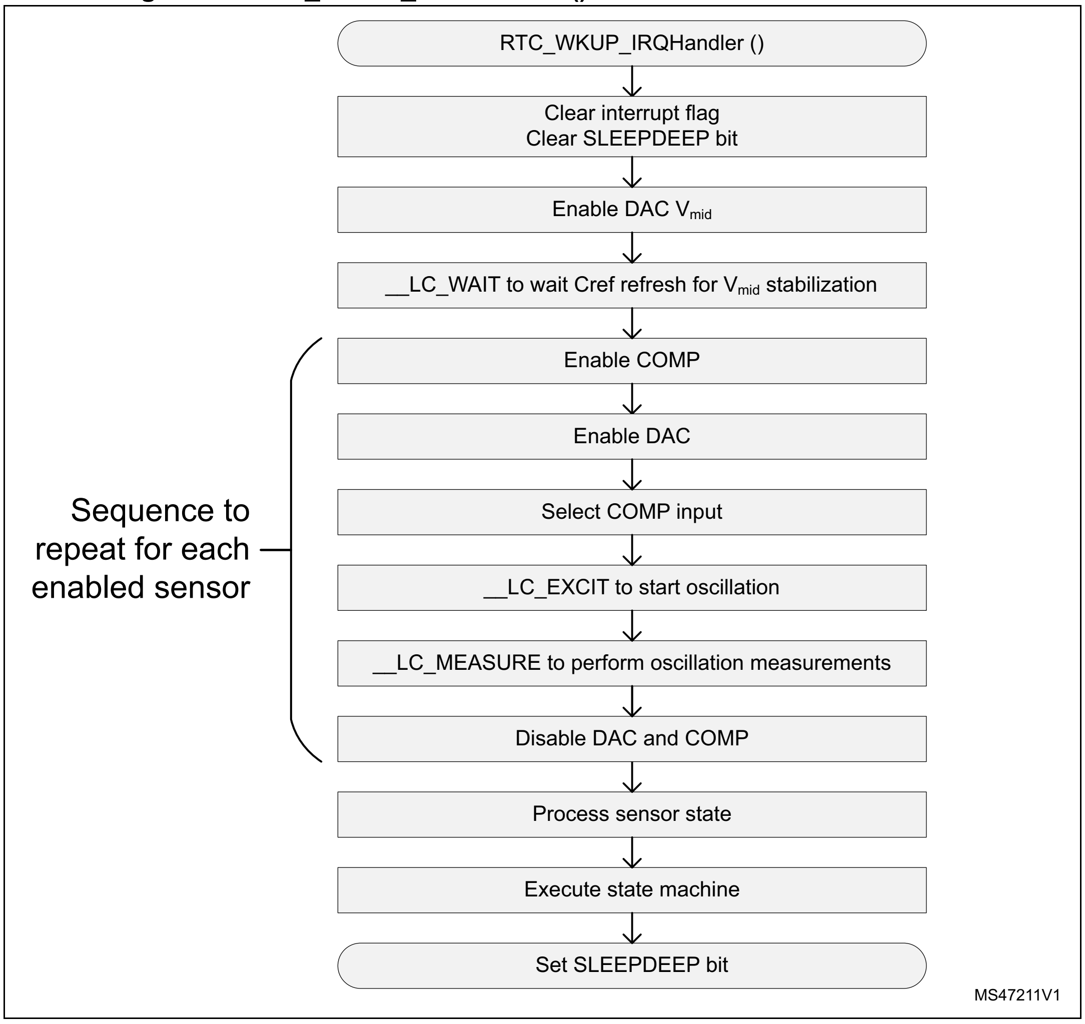
The excitation pulse is delivered by IO pin placed in GPIO output mode. As soon as the pulse is applied on the LC network, IO is switched into Analog state and acts as the non-inverting comparator input.

Application waits for a capture time in SLEEP mode until all the pulses are collected

If more than one sensor is enabled, a waiting time in SLEEP mode is added to let a stabilization time between the two measurements. Another excitation pulse is delivered on the next sensor with a new capture time in SLEEP mode until all the pulses are collected.

At the end of measurements, switched off the V_{mid} reference bias voltage. DAC and COMP2 are disabled to minimize power consumption. The capture of the LC sensor oscillations has been performed during this interrupt.

Figure 43. RTC_WKUP_IRQHandler () subroutine for one LC sensor



LC_calibration () function

This function is executed only if LcConfig.FirstCal.Status variable is set to CAL_ENABLED (CAL_DISABLED by default).

The aim of this function is to perform a calibration at the power up.

Maximum and minimum pulse counts, capture time and detection threshold are updated during this calibration according to components and external conditions.

Two examples are available, STATIC or DYNAMIC

- Static method example: there is no metal in the proximity of LC sensor, so only the maximum pulse count is used to define the detection threshold count number and capture time for measurements.
- Dynamic method example: a wheel is in rotation with metal and no metal transitions. The maximum and minimum pulse counts are used to define the detection threshold count number and capture time for measurements.

Figure 44. LC_calibration () function STATIC method

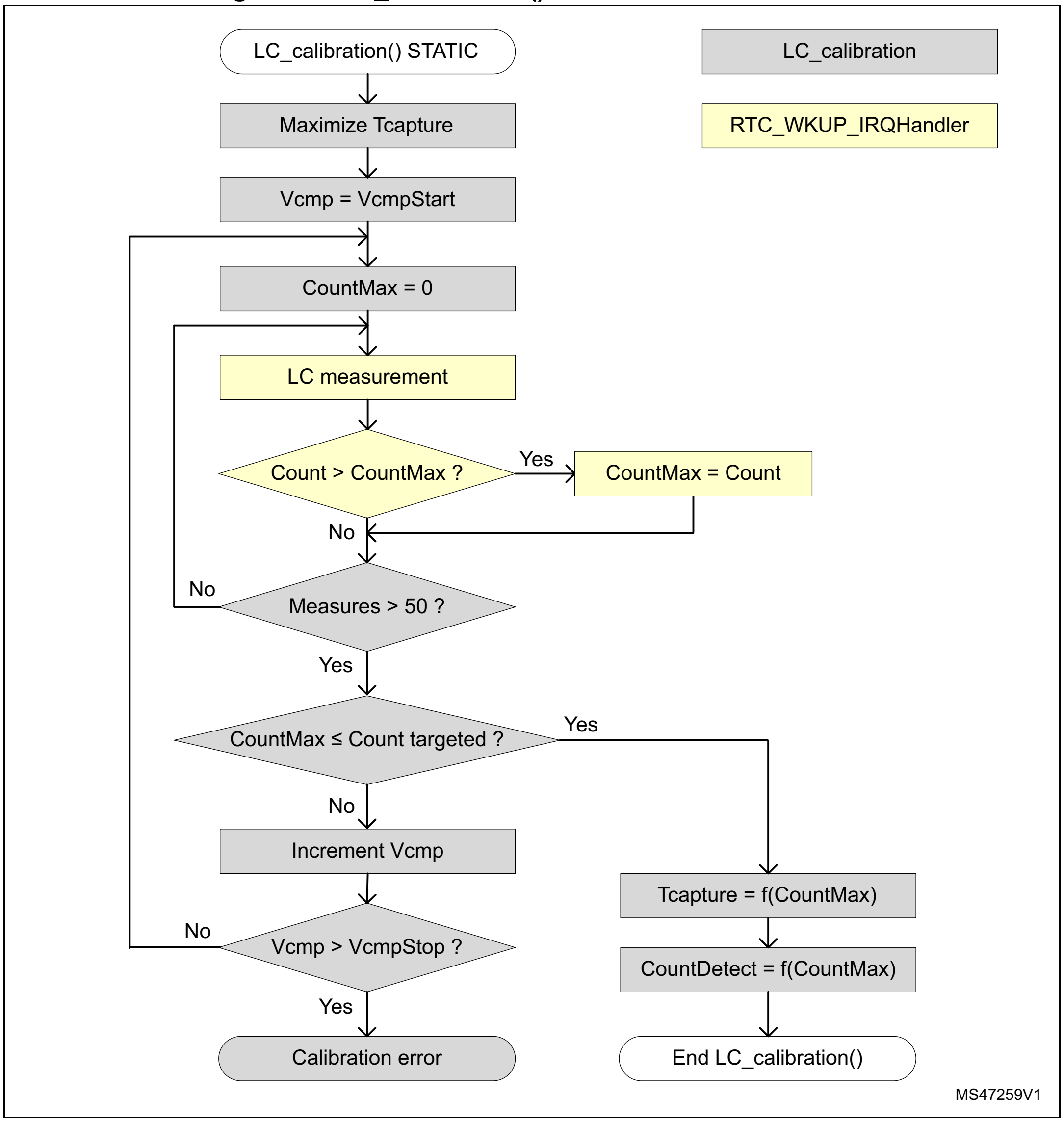
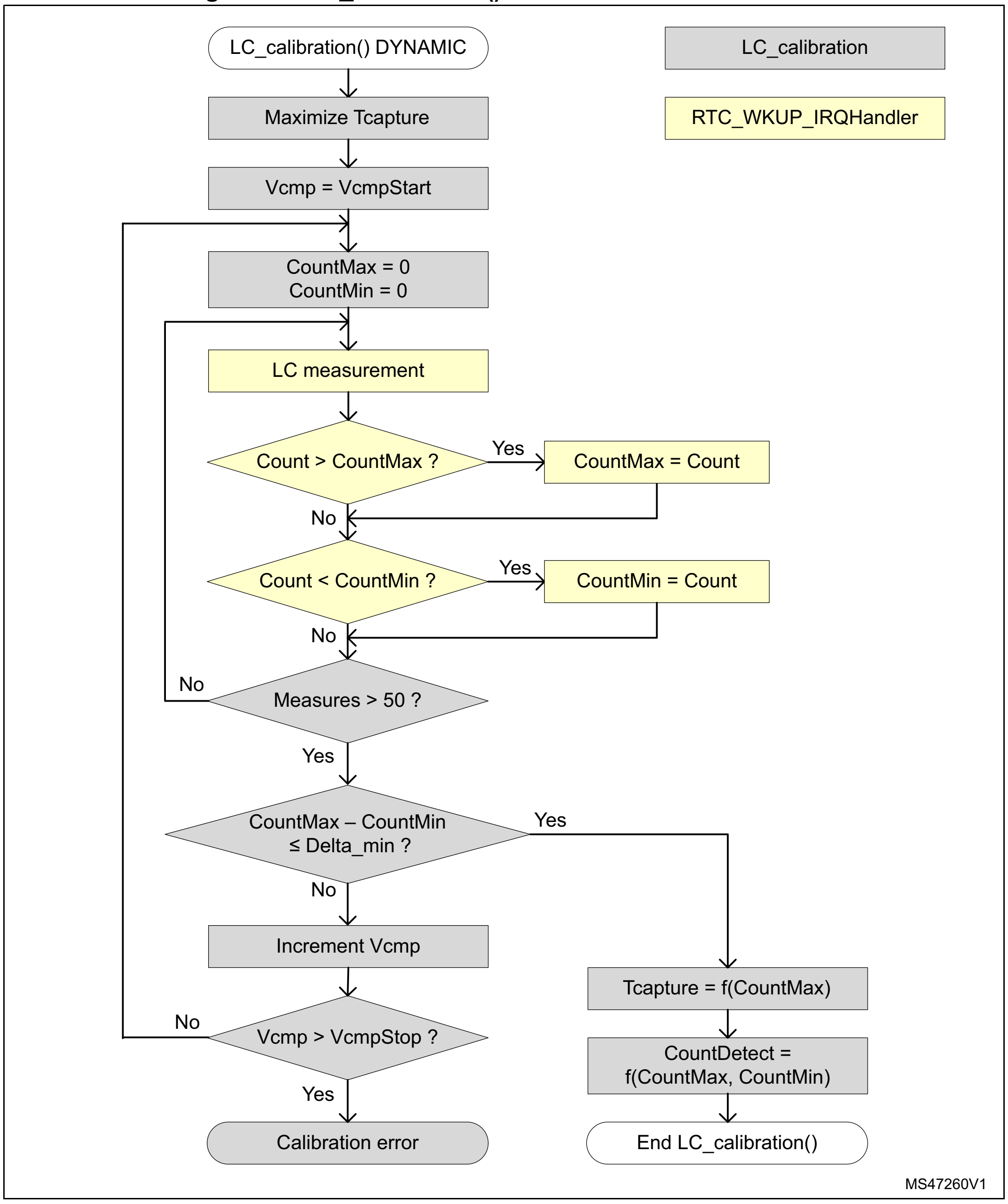


Figure 45. LC_calibration () function DYNAMIC method



HAL_RTCEx_AlarmBEventCallback () - Periodic calibration function

This function is executed only if LcConfig.PeriodicCal.Status variable is set to CAL_ENABLED (CAL_DISABLED by default).

All calibration operations are managed in Alarm interrupt and must be done between two LC measurements.

Calibration measurements are performed during standard measurements in WakeUp timer interrupt handler.

The periodic calibration (*Figure 46*) is performed in three steps:

1. Calibration initialization, executed in HAL_RTCEx_AlarmBEventCallback()

   Capture time is maximized to catch all pulses count and calibration variables are initialized for each sensor.
   – PeriodicCal.SensorXCountMax = 0, with X = 1, 2, 3, or 4
   – PeriodicCal.SensorXCountMin = 0xFFFF, with X = 1, 2, 3, or 4

   After that, the alarm is rearmed to ensure the time needed (few seconds) to perform calibration measurements.
2. Calibration measurements, executed in RTC_WKUP_IRQHandler()

   These measurements are performed in a routine executed during normal operation in WakeUp timer interrupt handler.
3. Calibration results analysis, and value update, executed in HAL_RTCEx_AlarmBEventCallback()

   The flow depends upon the selected method.

Figure 46. Periodic calibration

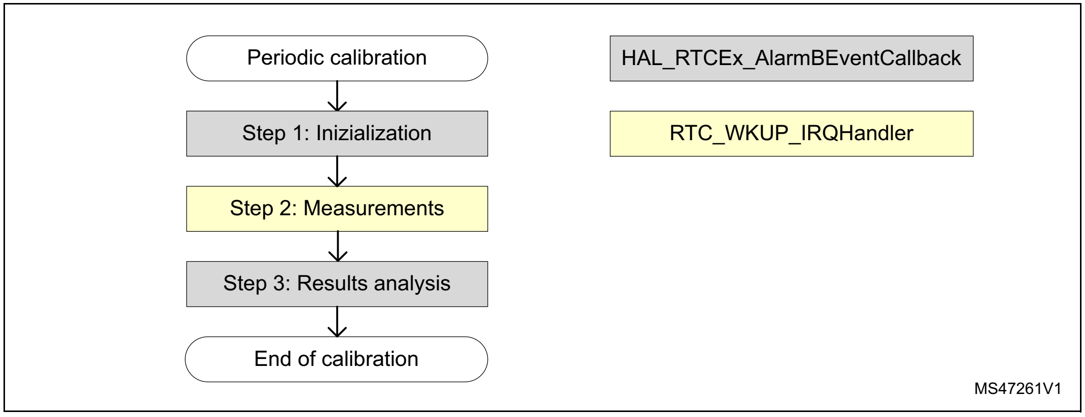


Figure 47. Periodic calibration interrupt profile

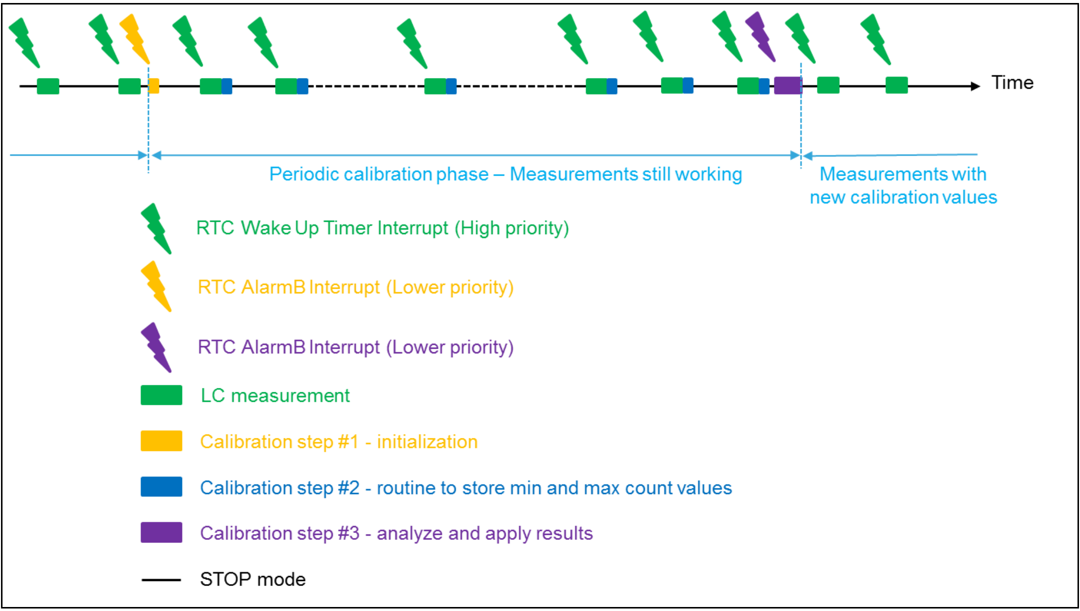
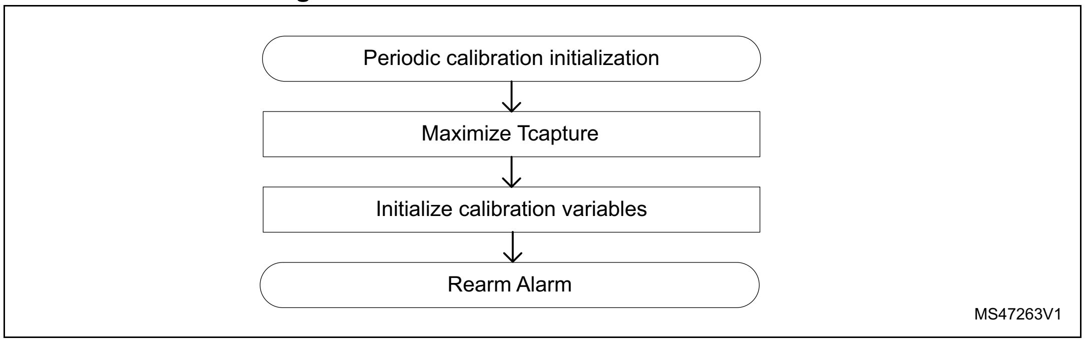

Time
Periodic calibration phase – Measurements still working
Measurements with new calibration values
RTC Wake Up Timer Interrupt (High priority)
RTC AlarmB Interrupt (Lower priority)
RTC AlarmB Interrupt (Lower priority)
LC measurement
Calibration step #1 - initialization
Calibration step #2 - routine to store min and max count values
Calibration step #3 - analyze and apply results
STOP mode

Figure 48. Periodic calibration initialization


Periodic calibration initialization
Maximize Tcapture
Initialize calibration variables
Rearm Alarm
MS47263V1

STATIC method example

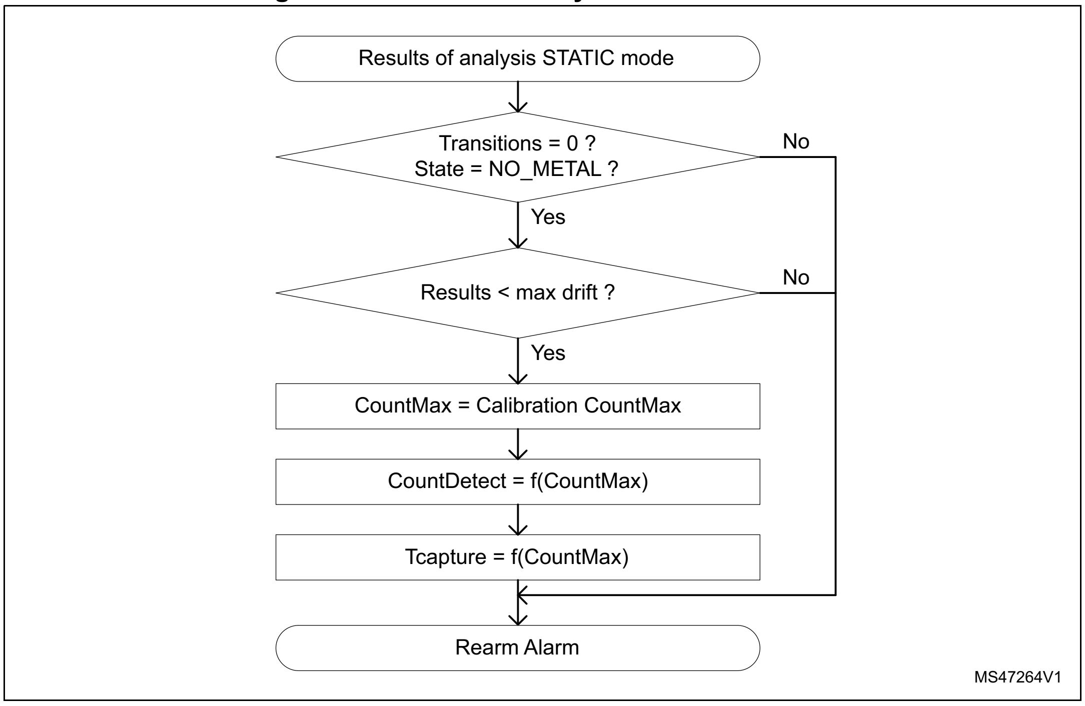
To check that calibration results are valid, the following points must be respected:

- No transitions during calibration phase
- Sensor state is no metal
- To eliminate noisy results, a maximum drift is allowed

After having checked the results, the new detection threshold counter and the new capture time are computed.

After that, the alarm is rearmed for the next periodic calibration.

Figure 49. Results of analysis in STATIC mode



DYNAMIC method example

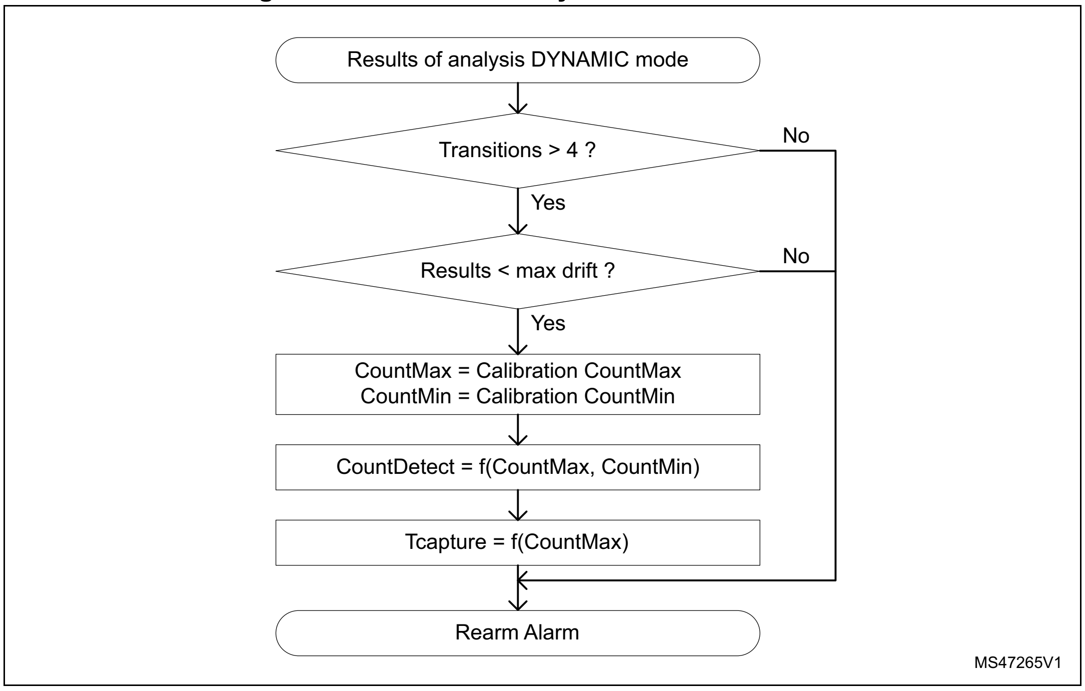
To check that calibration results are valid, the following point must be respected:

- Transitions number must be non null to catch metal and no metal states (a minimum of two rotations with four transitions)
- To eliminate noisy results, a maximum drift is allowed

After results validation, new detection threshold counter and new capture time are computed.

After that, the alarm is rearmed for the next periodic calibration.

Figure 50. Results of analysis in DYNAMIC mode


Results of analysis DYNAMIC mode
Transitions > 4 ?
No
Yes
Results < max drift ?
No
Yes
CountMax = Calibration CountMax
CountMin = Calibration CountMin
CountDetect = f(CountMax, CountMin)
Tcapture = f(CountMax)
Rearm Alarm
MS47265V1

6 Power consumption - STM32L476RG-NUCLEO

As shown by *Figure 51* and *Figure 52*, the power consumption depends upon the application: the main parameters are the number of sensors, the inductance and capacitor values, and the sampling rate.

Figure 51. STM32L073Z - One sensor - I_{DD} (average) vs. sampling rate (V_{DD} = 3.3 V)

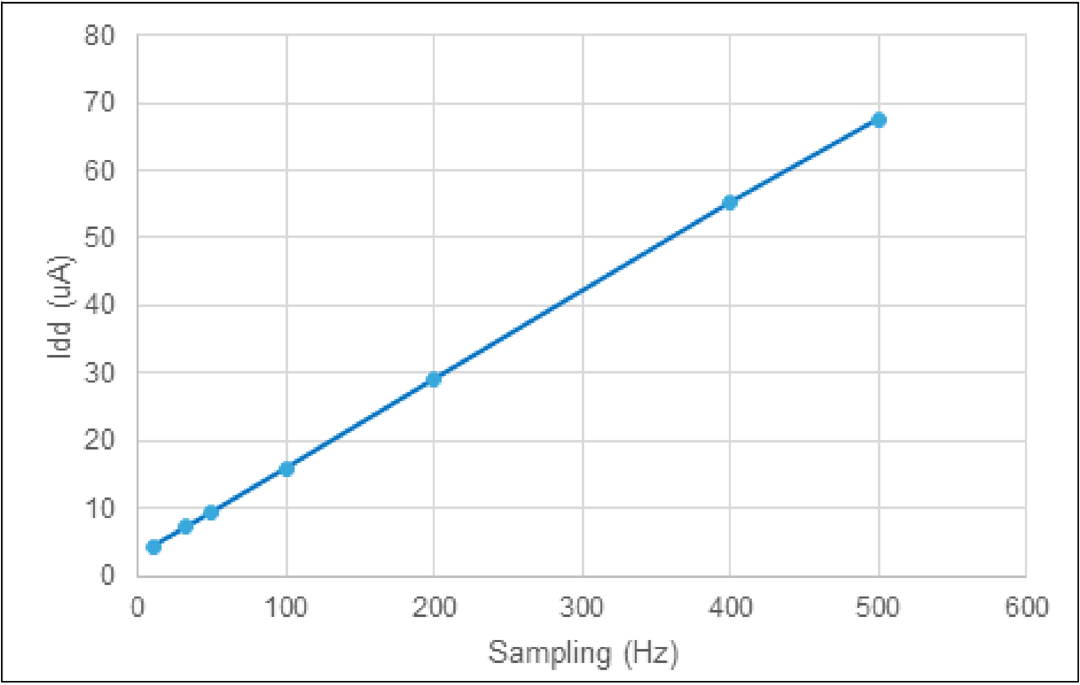
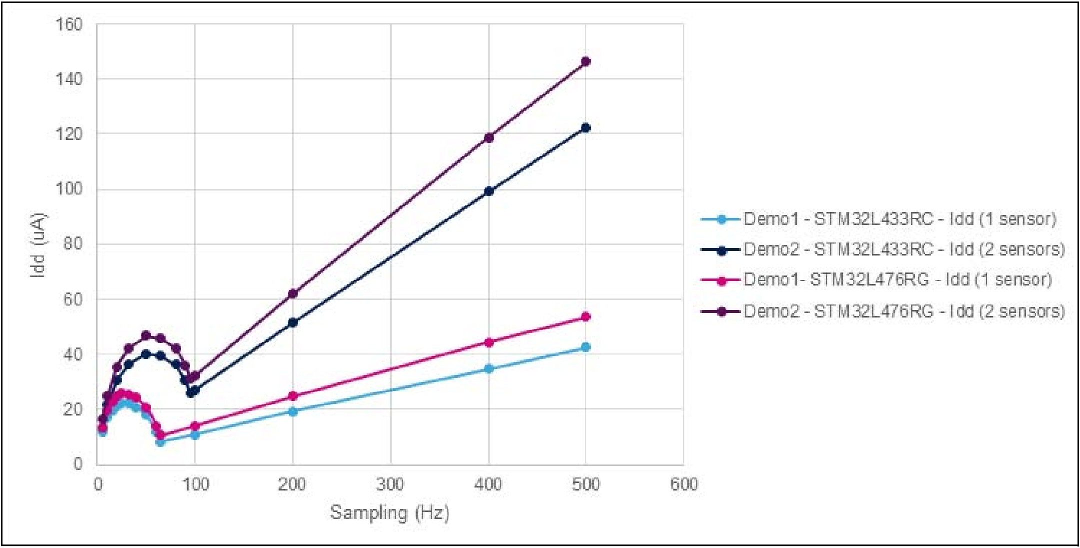


Figure 52. STM32L476RG and STM32L433RC - One or two sensors I_{DD} (average) vs. sampling rate (V_{DD} = 3.3 V)



For the STM32L4 demonstration, the power consumption is higher for low sampling rates due to the DAC Vmid refresh time. Indeed, Vmid drop increases with low frequencies and must be compensated by a power consuming refresh time, see *DAC Vmid refresh time (Trefresh)*.

For the demonstration with Cref = 470 nF, a refresh time must be added for sampling rates below 100 Hz (two sensors). If the application requests such sampling rates it is worth trying with 100 Hz anyway, to reduce power consumption.

7 Application description - STM32U031R8-NUCLEO

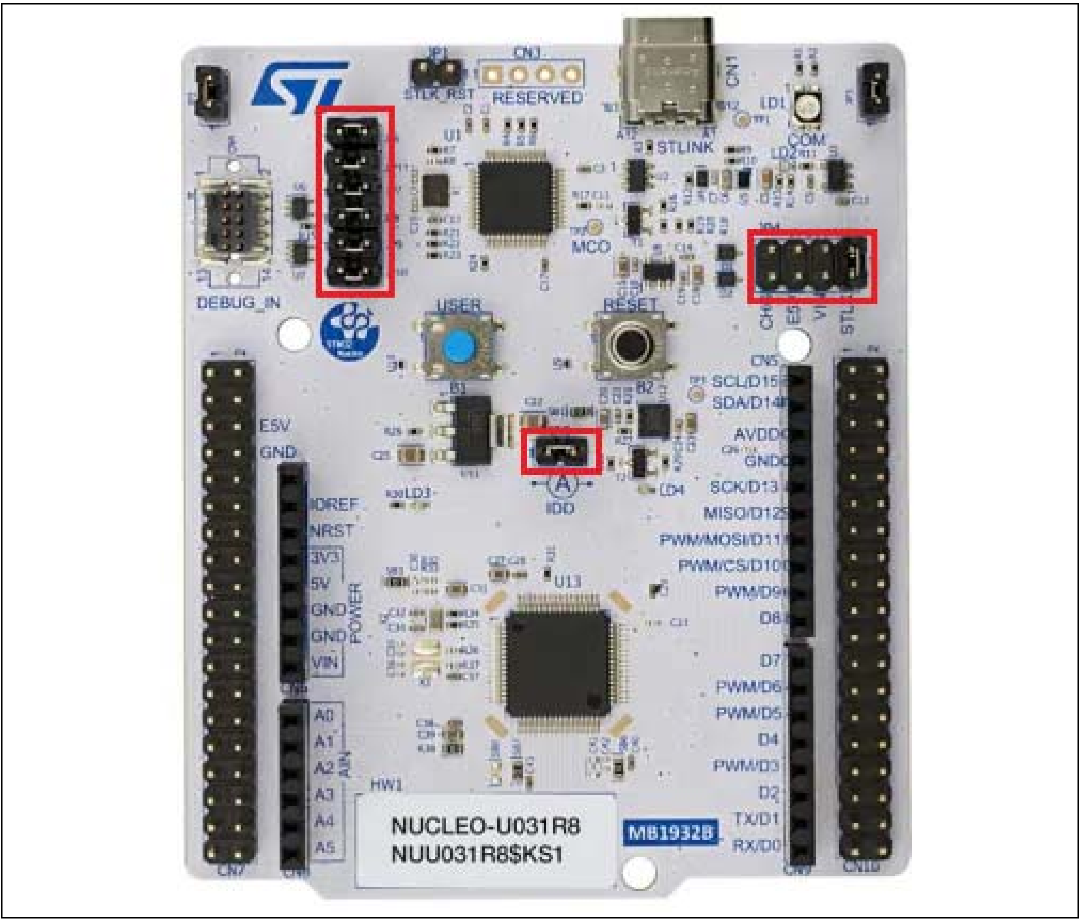
7.1 Hardware required

This demonstration uses STM32U031R8-NUCLEO board and custom board for LC sensing with external components (see *Figure 53*). The board can be connected to the USB cable of the host PC. To simulate the presence of metal over the LC sensor there is used rotating wheel with half copper layer metal side and half no-metal side.

7.2 Hardware settings

Jumpers (red boxes in *Figure 53*) should be set in their initial factory configuration.

Figure 53. Configuration of jumpers



7.3 Hardware settings for LC sensor custom board

7.3.1 Requirements

A custom board with LC sensors must be developed and connected to the STM32U0 I/Os.

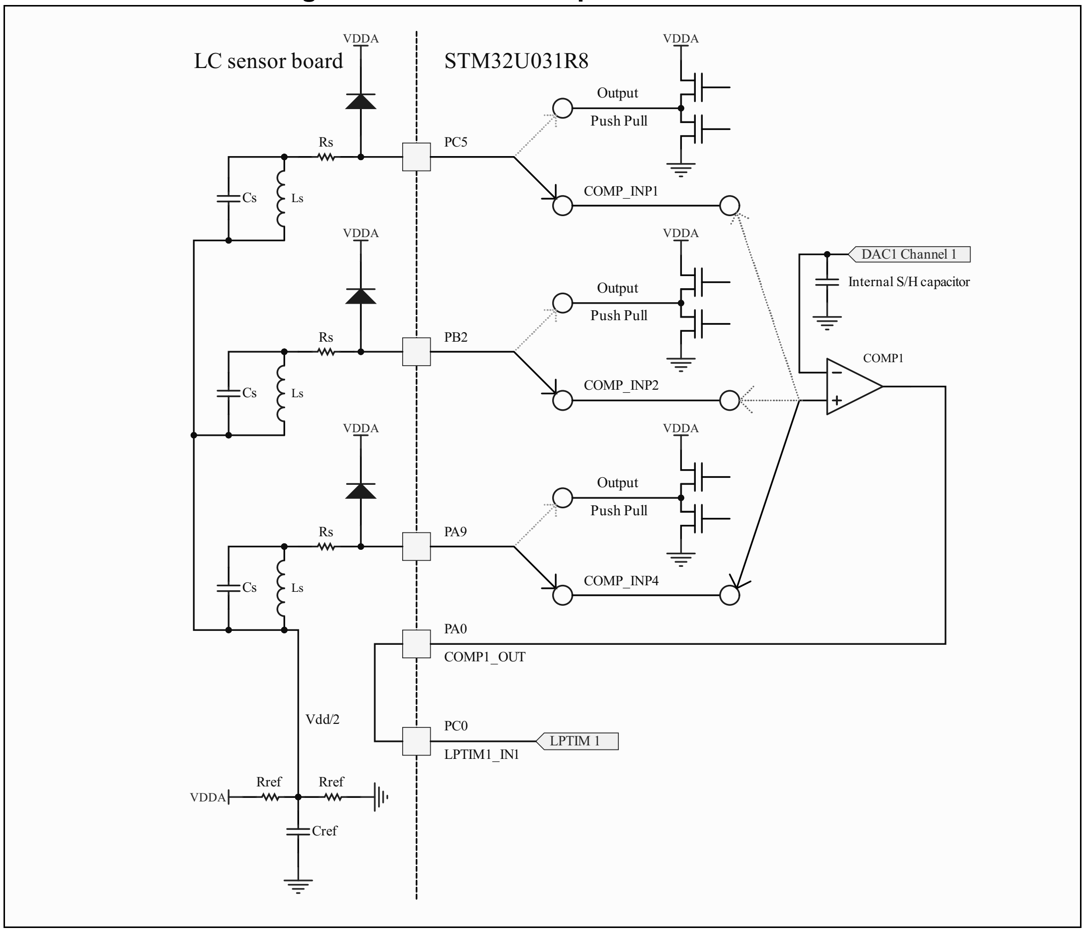
External components must be selected to minimize deviations due to external conditions (such as temperature, humidity), the characteristics of the components used for this demonstration are detailed below.

- 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 = 160 µH
  - Cs = 200 pF
- Reference capacitor: used to store energy for LC sensors oscillations.
  - Cref = 1 µF
- Reference resistor divider: used to create V_{DD} / 2 reference voltage.
  - Rref = 270 kΩ
- 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 (20 mA for STM32U0xxxx, refer to the product datasheet).
  - R = 150 Ω (for V_{DD} = 3 V)
- Protection diodes: external diodes must be connected to IO and VDDA, see *Excitation time: Texcit* to define if requested or not. For this example the diode is 1N4148.

Table 3. I/O configuration

Pin	LC function	Type
PA0	COMP1_OUT	Output
PC0	LPTIM1_IN1	Input
PC5	Sensor1	Input (COMP_INP1) / output
PB2	Sensor2	Input (COMP_INP2) / output
PA9	Sensor3	Input (COMP_INP4) / output
PA5	LED	Output
PA2	UART2_TX	Output
PA3	UART2_RX	Input

Figure 54. Schematic: up to three sensors



7.4 Application principles

7.4.1 Overview

The application is based on the same principle seen for the STM32L4 application, the main difference is that the V_{mid} level (V_{DD} / 2) is generated with external circuitry.

The LC sensor is driven by an excitation pulse managed by GPIOs. Once the pulse is delivered, the GPIO is reconfigured from output push-pull to analog input. The oscillations appear on the non-inverting comparator and compared to V_{mid} level (V_{DD} / 2).

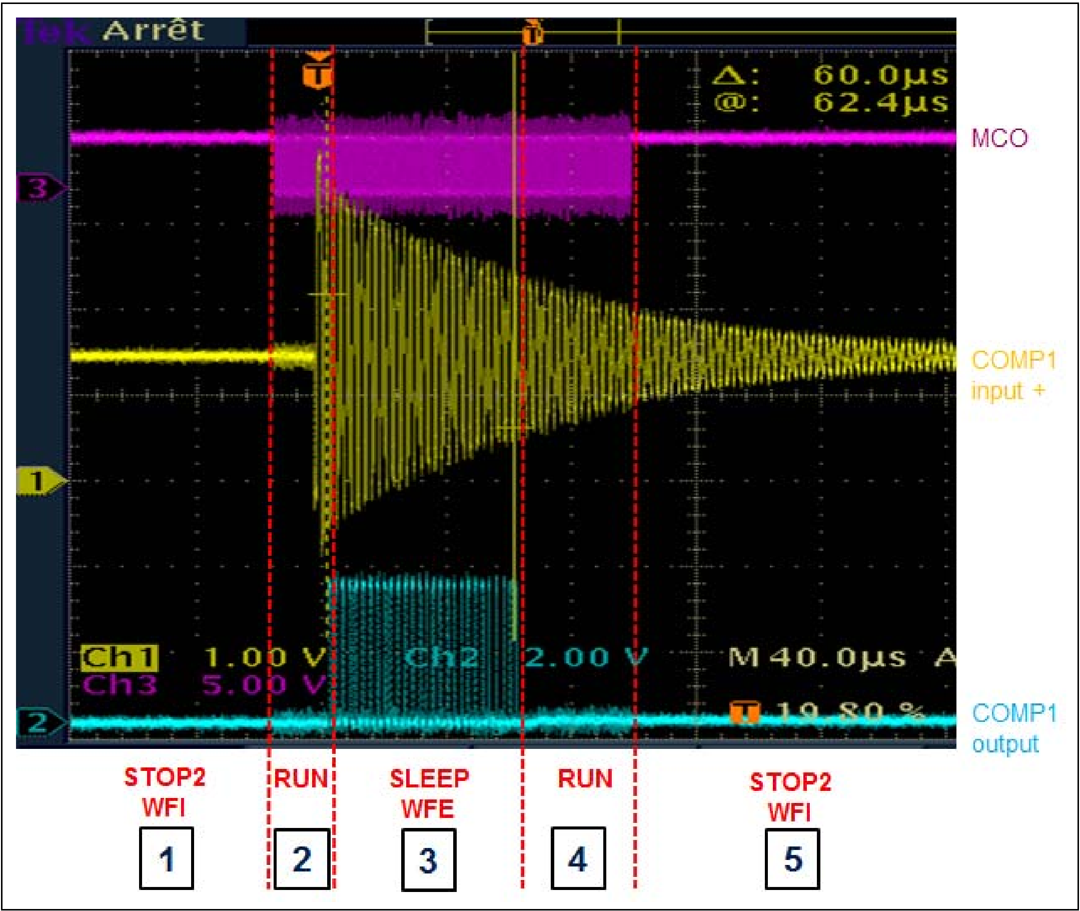
V_{mid} level is generated by the external resistor divider. The value of each resistors is high enough to reduce power consumption, and low enough to ensure precise V_{mid} level during measurements.

Once started, the amplitude of these oscillations decays exponentially until the signal comes back to V_{mid} bias, as shown in *Figure 55*. This behavior is obtained when the LC sensor is in air and when no metal is present.

7.4.2 LC sensor metering capture window

To save power consumption, the LC sensor metering is activated periodically and the microcontroller is put in Stop2 mode.

Figure 55. LC measurement sequence



For each measurement, as indicated in *Figure 55*, the following steps are executed:

1. The LC sensor metering is activated and the microcontroller is put in the Stop2 mode (low power mode).
2. The MCU is woken-up and enables the peripherals used by application. After that, the LC sensor excitation pulse is delivered (see *Excitation time: Texcit*). To perform the measurement one pulse timer is started and MCU is placed in WFE (refer to *Capture window: Tcapture*).
3. During the measurement phase the MCU is put in sleep 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 timer event, the MCU returns in Run mode, processes the LPTIMER result and compares it with the detection threshold (CountDetect) to define if there is a Metal or No Metal state. The software state machine can process counter results at this level.
5. The MCU goes back in Stop2 mode and reaches the lowest power consumption.

7.4.3 Parameters to define

Some parameters must be defined according to the application requirements:

DAC configuration

One channel is used to generate V_{cmp} level:

- V_{cmp} for threshold comparator level: Channel 1 connected to on chip comparator inverting input.

DAC peripheral is configured in Sample and Hold mode and managed in wake-up interrupt. It is enabled at each LC measurement and disabled before returning in Stop2 mode.

Comparator threshold voltage (V_{cmp})

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

In this example: DAC output is configured to deliver $V_{cmp} = V_{mid} + 364$ mV

DAC V_{cmp} (to program) = 4096 * (V_{cmp} / V_{DD}) = 4096 * (2.01 V / 3.3 V) = 2500

Excitation time: T_{excit}

LC sensor excitation can be done to V_{SS} (output push pull down) or to V_{DD} (output push pull up), the default value used in this example. The pulse width can be adjusted to set the oscillation amplitude (T_{excit})

If the maximum oscillation amplitude exceeds the maximum allowed voltage, an external diode must be added to protect the I/O. See the product datasheet for details. For this example Vin max = min (min (V_{DD}, V_{DDA}) + 3.6 V, 5.5 V) = 5.5 V.

Figure 56. Excitation pulse managed by GPIOs - Example with T_{excit} to V_{SS} = 250 ns

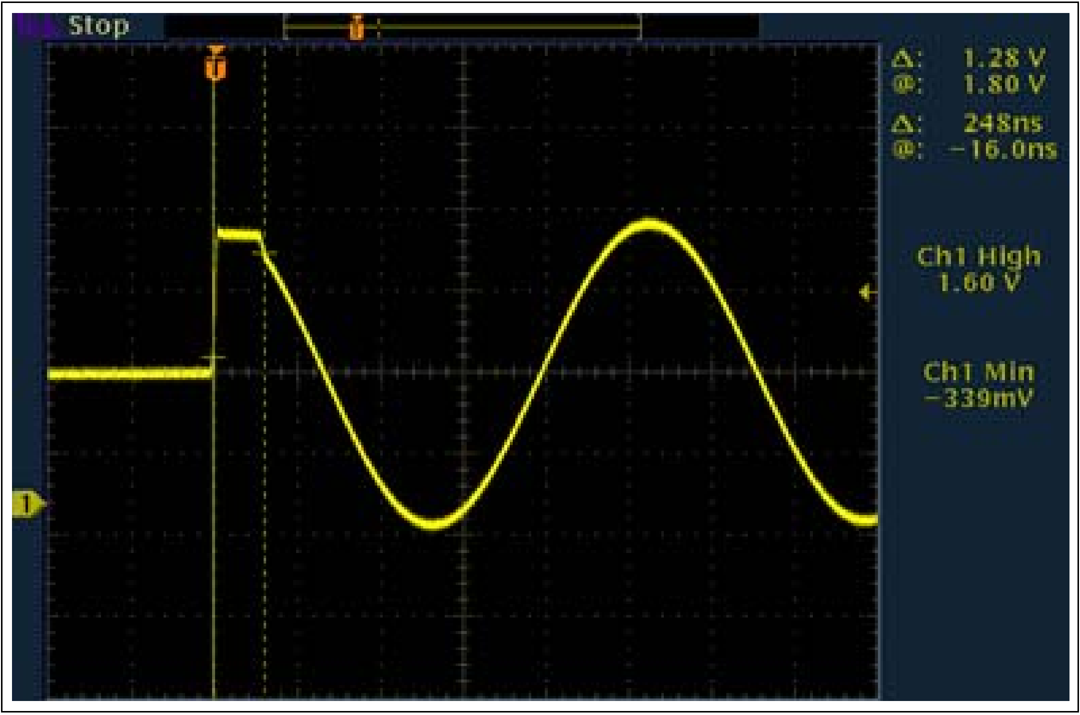
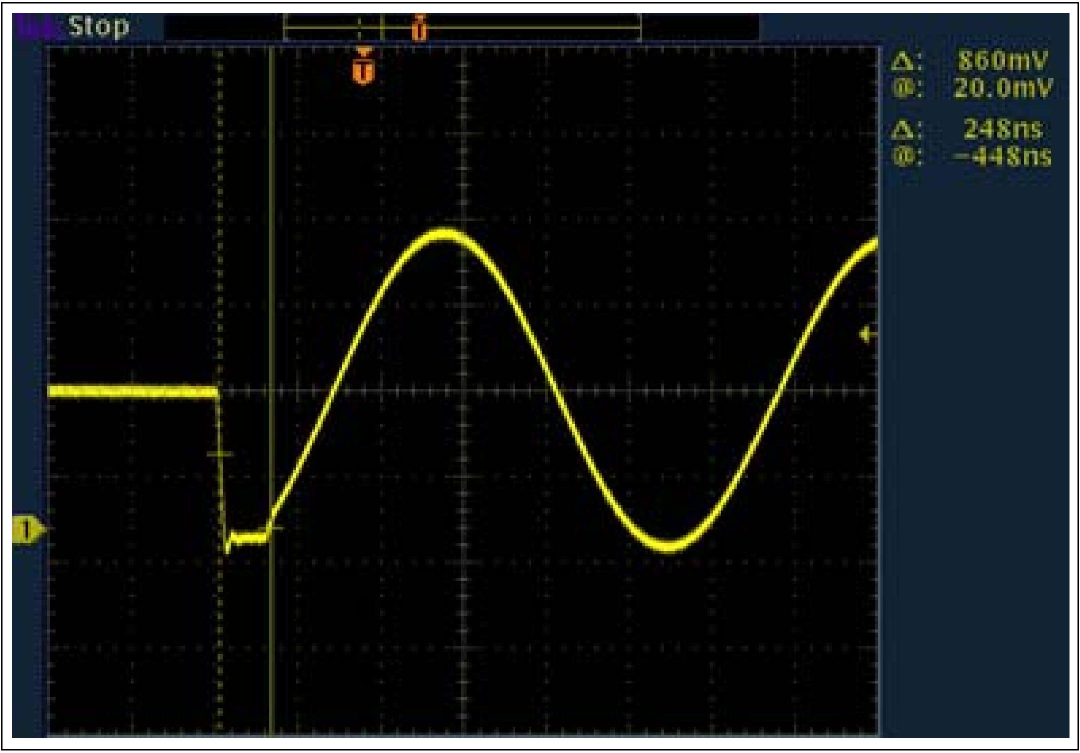


Figure 57. Excitation pulse managed by GPIOs - Example with T_{excit} to V_{DD} = 250 ns



The excitation is programmed as direct register access to obtain excitation for x CPU cycles. The size of excitation pulse is $T_{excit\_seconds}$ = (3 * ($T_{excit\_to\ program}$ + 1)) / Freq_MSI.

For this example $T_{excit\_to\ program}$ = ((250 ns * 24 MHz) / 3) - 1 = 1

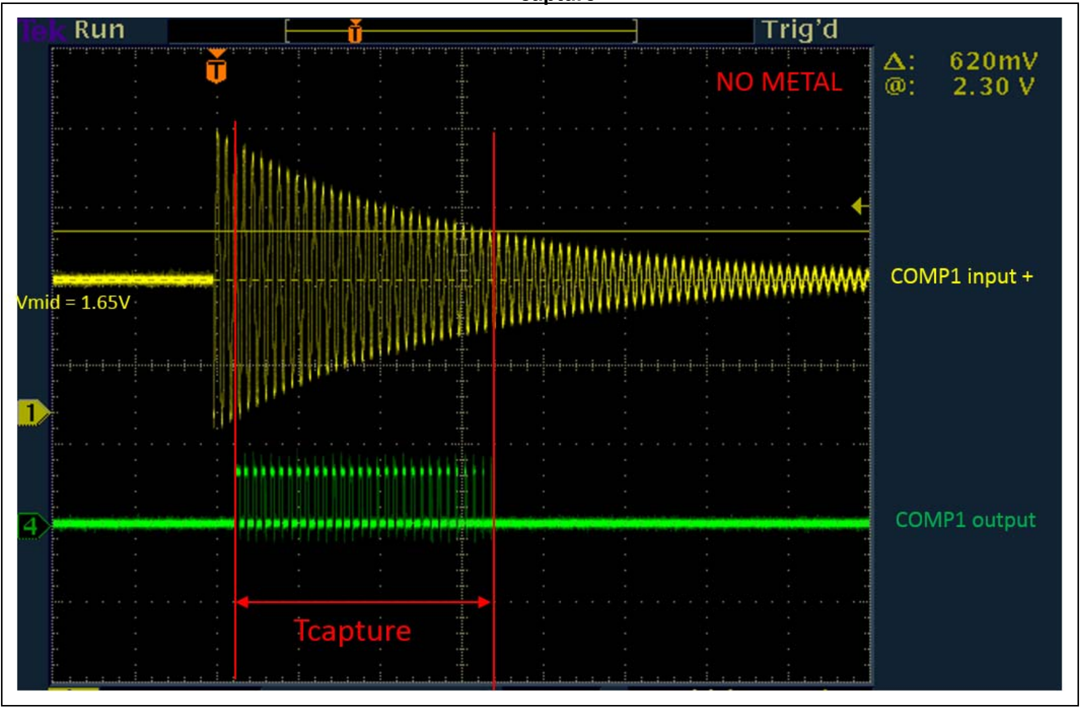
Capture window: $T_{capture}$

The collection of LC sensor oscillation and pulse count is performed during this time, managed with timer peripheral (TIM6) configured in One Pulse mode.

$T_{capture\_to\ program}$ = ($T_{capture\_seconds}$ - 23.5 µs) * Freq_MSI

For this example (see *Figure 58*) $T_{capture\_to\ program}$ = (50 - 23.5) µs * 24 MHz = 636.

Figure 58. $T_{capture}$ example



To reduce power consumption, the capture window can be minimized and adjusted to the maximum count number (without metal).

$T_{capture\_seconds}$ = (CountMax / Freq_Osc) * (1 + x), where x is the margin in percent (default value is 5%).

The theoretical oscillation frequency is determined as

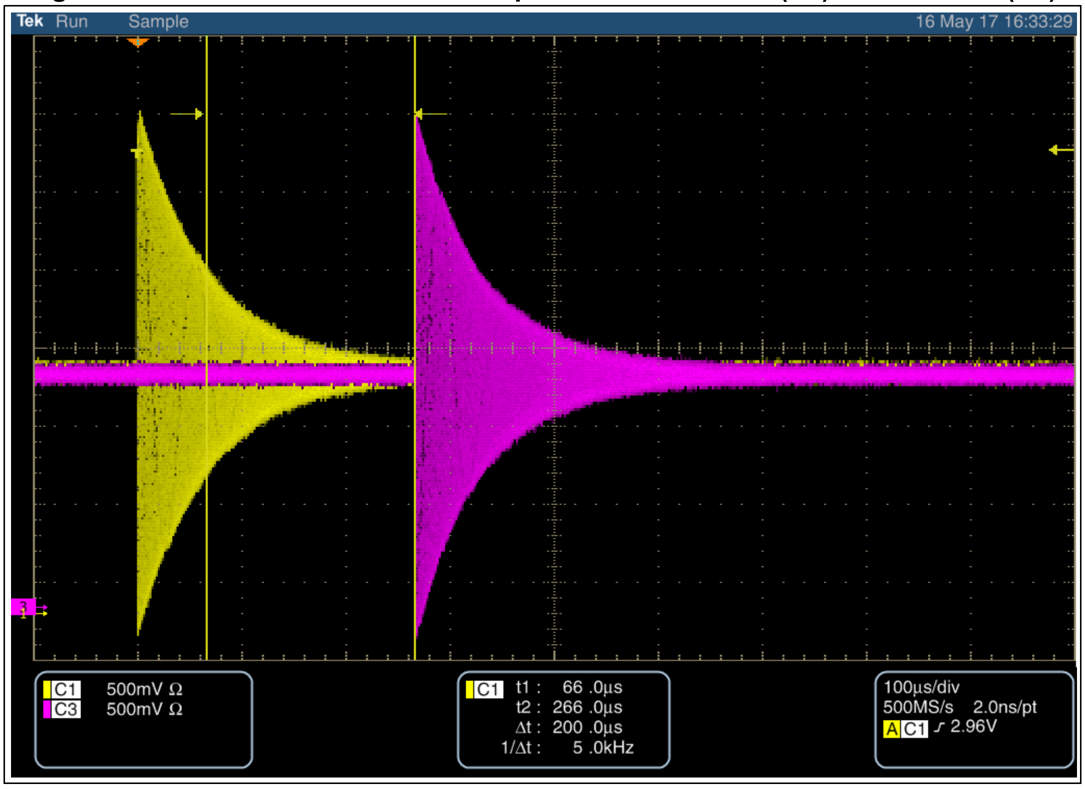
$$\text{Freqosc} = \frac{1}{2 \cdot \pi \cdot \sqrt{\text{Ls} \cdot \text{Cs}}}$$

In this example, the value of the inductor Ls is 160 µH and that of the capacitor Cs is 200 pF. The oscillations frequency is then approximately 890 kHz.

Time between measurements

Measurements must be done sequentially to avoid cross-effects between the 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.

Figure 59. Time between sensors: 200 µs between Sensor1 (C1) and Sensor3 (C3)



TimeBetweenSensor is managed with timer peripheral (TIM6) configured in One Pulse mode.

$\text{TimeBetweenSensor}_{\text{to program}}$ = ($\text{TimeBetweenSensor}_{\text{seconds}}$ -23.5 µs) * Freq_MSI

For this example $\text{TimeBetweenSensor}_{\text{to program}}$ = (200 -23.5) µs * 24 MHz = 4236

Using up to three LC detectors

In the LC sensor detection system based on the STM32U0 MCU, it is possible to extend the number of LC networks, as already seen in *Figure 54*, and summarized in *Table 4*.

- One sensor

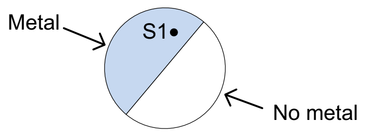
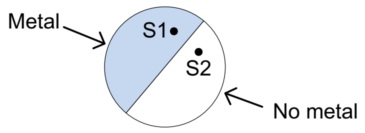
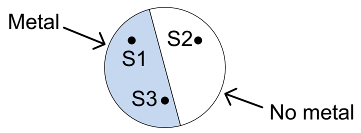
  This configuration makes possible only to detect the metal presence or the wheel rotation
  - Two increments by rotation
- Two sensors

  This configuration makes possible to detect both the rotation and the direction of the wheel
  - Sensors are placed around the wheel and spaced 90°
  - Up to four increments by rotation
  - Clockwise and anticlockwise counter
  - Error prevention

- Three sensors

  This configuration results in a higher precision (six increments per rotation) and prevents some errors.
  - Sensors are placed around the wheel and spaced 120°
  - Up to six increments per rotation
  - Clockwise and anticlockwise counter

Table 4. Sensor configurations supported by STM32U031R8-NUCLEO board

Configuration		Schematic
One sensor	One wheel	
Two sensors[1]		
Three sensors[2]		

1. S1 and S2 are radially spaced 90°.
2. S1, S2, and S3 are radially spaced 120° from each other.

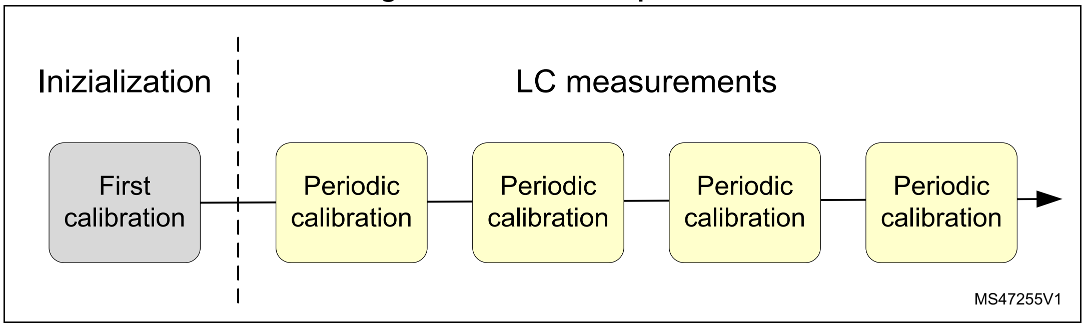
7.4.4 Calibration

The examples described in this section detail a method to calibrate the system and compensate LC sensors deviations. They must be tuned according to the application requirements.

In this demonstration, the system is calibrated in two steps:

- First calibration: performed during the system start (or factory calibration)
- Periodic calibration: performed during normal operation to compensate for small drifts due to external conditions modifications (temperature, humidity, voltage). Measurements are still carried out during this phase, so only small drifts can be compensated.

Figure 60. Calibration phases



For each calibration phase, two examples are presented, depending upon the application:

- Static method for basic detection: the calibration is done without metal in proximity of the LC sensor (air).
- Dynamic method, typically with rotating wheel: the calibration is done with metal and no metal transitions.

First calibration

The aim of this calibration is to adjust 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 (see *Section 5.2* for more details).

Static method

This calibration is done without metal and based on the maximum pulses count number.

The predefined pulses count number must be defined carefully:

- A small number involves a small count difference between metal and no metal state. The detection threshold is near maximum and minimum values, and instabilities can appear.
- A number too large involves MCU Run state, and power consumption increases. Moreover, noise can appear with small oscillations amplitude.

The following parameters are automatically defined (by software) during this calibration:

- Vcmp

  The comparator threshold level (Vcmp) is adjusted to have a predefined maximum oscillations count number without metal in the proximity of LC sensors.
- CountDetect = f(CountMax)

  The detection threshold (metal / no metal) is based only on the maximum oscillations number: CountDetect = CountMax * x, where x = value in percent
- Tcapture= f(CountMax)

  To save power consumption, the capture time window is minimized and adjusted to the maximum pulses count number: Tcapture = (CountMax / Freq_osc) * (1 + x), where
  - Freq_Osc = 1 / (2 π (Ls * Cs)$^{1/2}$)
  - Ls = Sensor inductor value
  - Cs = Sensor capacitor value
  - x = margin in percent

Example

- Comparator threshold level setting (Vcmp)

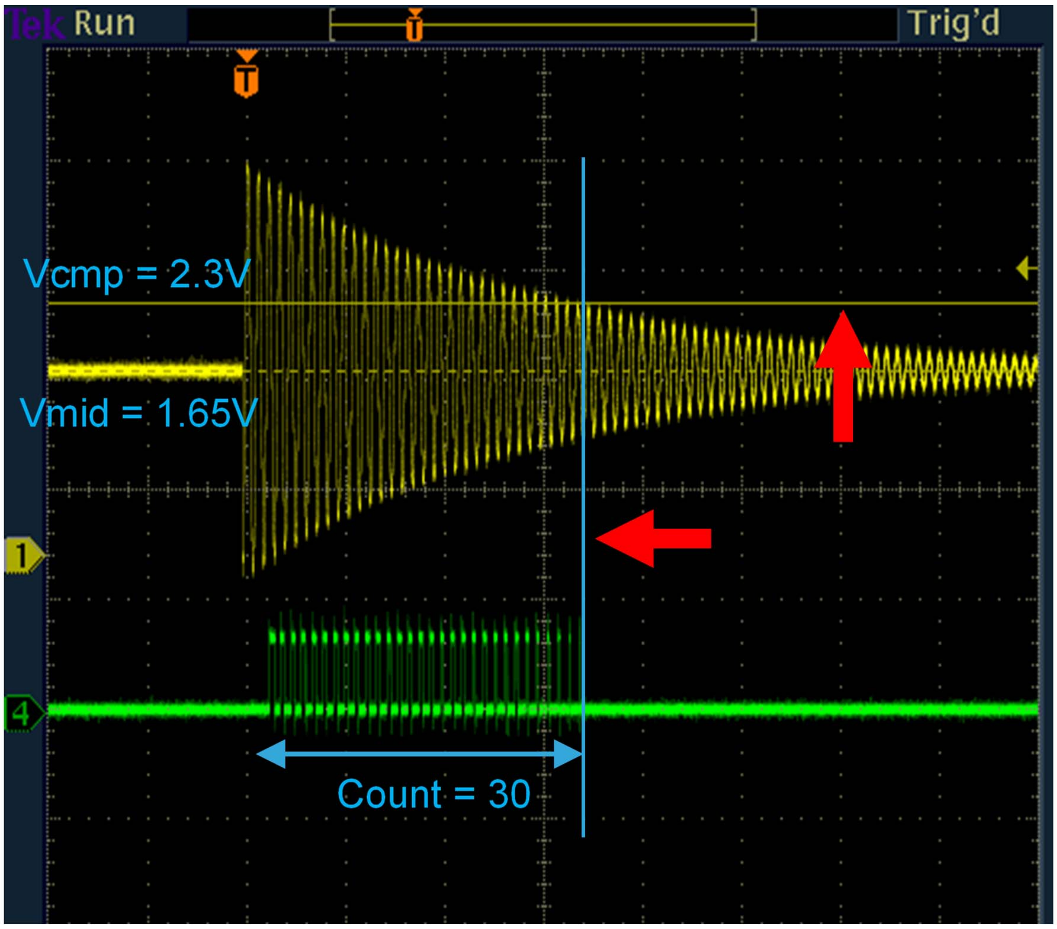
  Vcmp is increased from 1.9 to 2.5 V. For each step, the CountMax value is compared to the targeted pulse count number, increments are stopped when it is reached.

  In the example (see *Figure 61*) PulseCount = 30 when Vcmp = 2.3 V.
- Detection threshold count number

  CountDetect = CountMax * x = 24 (for example x = 80%)

  Tcapture (CountMax = 30, Ls = 160 µH, Cs = 200 pF) ≈ 34 µs

Figure 61. First calibration example for a predefined maximum count number of 30



Dynamic method

This calibration is done with metal and no metal states to have, respectively, minimum and maximum pulses count numbers for the predefined comparator threshold.

The Vcmp must be defined carefully:

- Vcmp close to V_{DD} / 2 involves MCU Run state, and power consumption increases. Moreover, noise can appear with small oscillations amplitude.
- Vcmp close to V_{DD} (or GND) involves a small count difference between metal and no metal states. The detection threshold is near maximum and minimum values, and instabilities can appear.

Minimum number of measurements during calibration:

- During the calibration measurements change state on the LC sensor between metal and no metal state at least once to reach minimal and maximal pulse count. There is a small difference (Δ) between the minimal and maximal pulse count to guarantee the occurrence of a transition.

The following parameters are automatically defined during this calibration:

- CountDetect = f(CountMax, CountMin)

  The detection threshold (metal / no metal) is based on the both maximum and minimum pulse count number. The goal is to have a minimum difference (Δ) in the pulse count minimum and maximum to reach a satisfactory distinction between metal and no metal state.
- CountDetec = (CountMax + CountMin) / 2
- Tcapture= f(CountMax)

  To reduce power consumption, the capture time window is minimized to be adjusted to maximum pulse count number: Tcapture = (CountMax / Freq_osc) * (1 + x), where
  - Freq_Osc = 1 / (2 π (Ls * Cs)$^{1/2}$)
  - Ls = Sensor inductor value
  - Cs = Sensor capacitor value
  - x = margin in percent

Example

- Comparator threshold level (Vcmp)

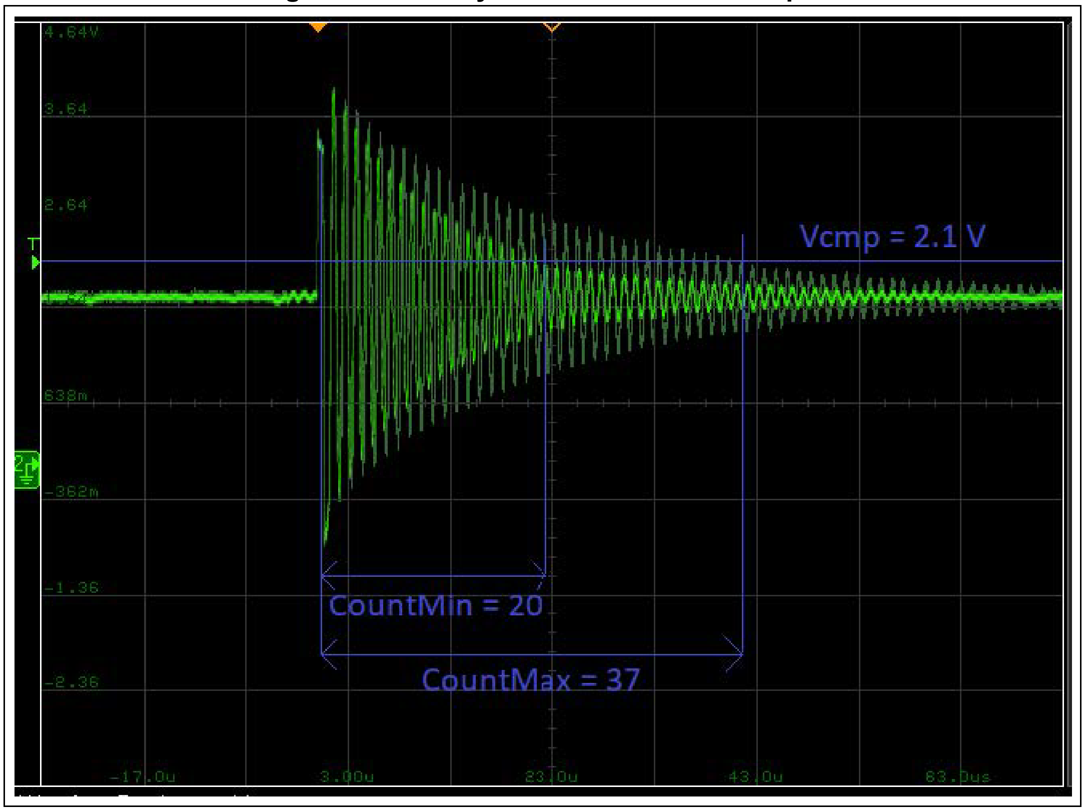
  Vcmp is set to 2.1 V. The minimal number of measurements are done and they are stopped when minimum Δ between minimal and maximal count detect is reached.

  In this example (*Figure 62*) PulseCountDeltaMin (Δ) = 10 < CountMax - CountMin = 17 and Vcmp = 2.1 V.

  Detection threshold count number (CountMax = 37, CountMin = 20) CountDetec = 28

  Tcapture (CountMax = 37, Ls = 160 µH, Cs = 200 pF) ≈ 42 µs

Figure 62. First dynamic calibration example



Periodic calibration

This makes possible to compensate small drifts due to external variations, such as temperature, voltage, humidity. During periodic calibrations, measurements are still ongoing.

The detection threshold is adjusted based on the calibration results (details are described in *HAL_RTCEx_AlarmBEventCallback () - Periodic calibration function*).

Static method

This calibration is done without metal in proximity of the LC sensor. Only the maximum pulse count number is used. To check that calibration results are valid, no transitions should appear during this phase (if this requirement is not met, ignore the calibration results).

To eliminate noisy results that could appear, a maximum drift is allowed:

- CountMax - CountDrift < Calibration CountMax result < CountMax + CountDrift

The following parameters are automatically defined during this calibration:

- CountDetect = f(CountMax), same as in *Static method* of *First calibration*
- Tcapture = f(CountMax), same as in *Static method* of *First calibration*

Dynamic method

This calibration is done with a rotating wheel. Maximum and minimum pulses count number are used, so it is mandatory to have several transitions during this calibration phase. If not, calibration results must be ignored.

To eliminate noisy results that could appear, maximum drifts are allowed:

- CountMax - CountDrift < Calibration CountMax result < CountMax + CountDrift
- CountMin - CountDrift < Calibration CountMin result < CountMin + CountDrift

The following parameters are automatically defined during this calibration:

- CountDetect = f(CountMax, CountMin), same as in *Dynamic method* of *First calibration*
- Tcapture = f(CountMax), same as in *Dynamic method* of *First calibration*

7.5 Getting started with the demonstration

There are four demonstrations available:

- Demo1: basic counting demonstration with one sensor to use with metal accessories.
- Demo2: counting demonstration with two sensors: rotation/direction detection to use with a rotating wheel.
- Demo3: tachometer demonstration with two sensors: rotation/direction detection to use with a rotating wheel.
- Demo4: tachometer demonstration with three sensors: rotation/direction detection and error states detection to use with a rotating wheel.

Each demonstration can be run in standard mode (for debug) or in low power mode (for the final application and to perform consumption measurements).

Counter information is sent periodically on USART2 port and can be displayed with UART terminal software on the host PC connected to the USB cable.

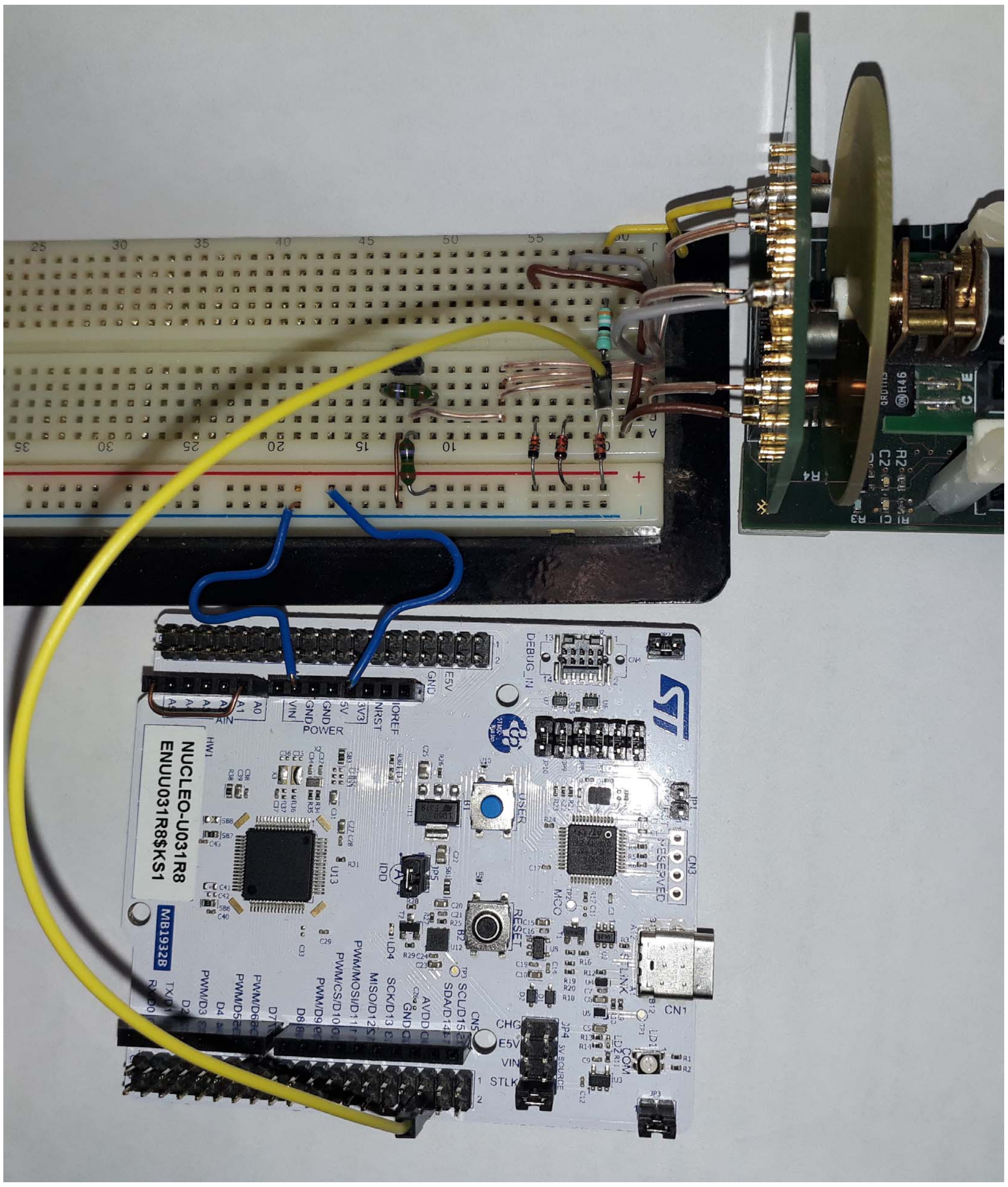
Terminal software configuration

- Port: STMicroelectronics STLink Virtual COM port
- Baud rate: 115200
- Data rate: 8 bit
- Parity: none
- Stop: 1 bit
- Flow control: none

7.5.1 LC sensor metering basic counting demonstration with one sensor

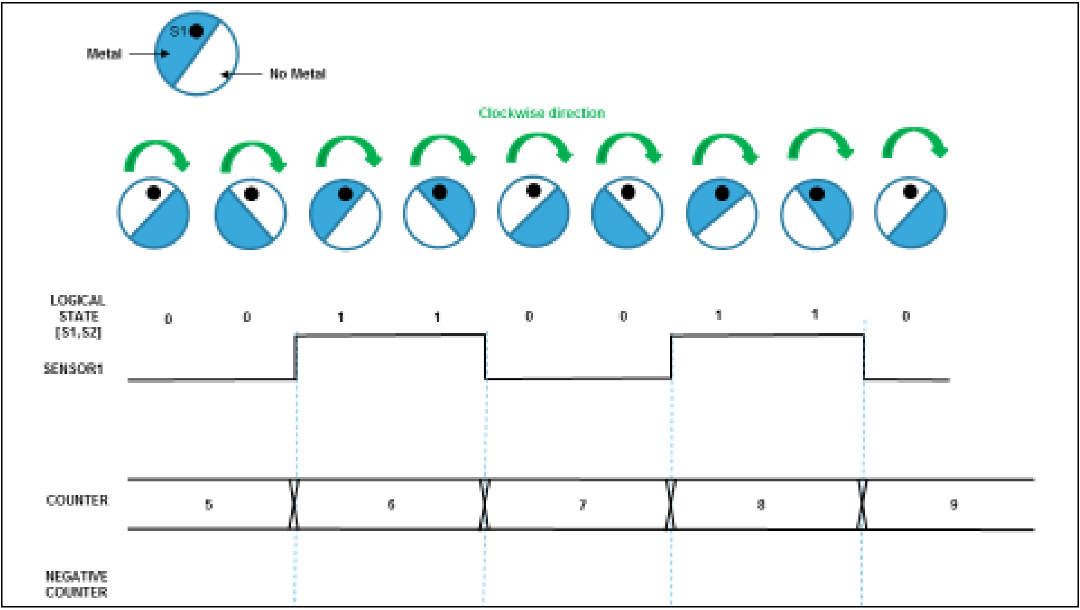
This demonstration uses one sensor with GPIO pins PC5, PA0, and PC0 (one GPIO is needed for each sensor and another two for the external connection of comparator output and low power timer input). This basic demonstration allows to check the presence of metal.

Figure 63. Hardware configuration for one sensor demonstration



When using one wheel, this basic demonstration makes it possible to collect the number of turns with two increments for each rotation, as indicated in *Figure 64*.

Figure 64. Demo1, one sensor



The counter is updated at each metal / no metal transition, but it is not possible to differentiate between clockwise and anticlockwise rotations.

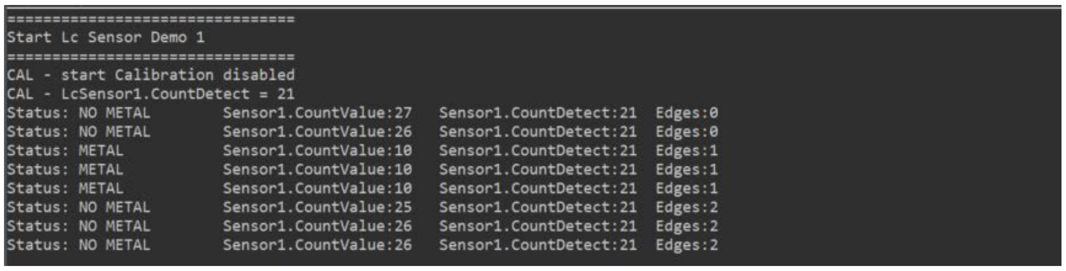
FW configuration

To enable this demonstration, LC_SENSOR_DEMO 1 define must be selected in lc_sensor_metering.h file.

LcConfig and LcConfigSensorX variables, set in LcSensorConfig () function, must be defined according to application (sampling rate, DAC refresh, V_{cmp}, T_{excit}, $T_{capture}$)

Each two seconds, the transition number (LcStatus.EdgeCount) and the current sensor status (LcSensor1.status) are updated. The last counter measurement (LcSensor1.CountValue) and the actual count detection limit (LcSensor1.CountDetect) are shown as well, see *Figure 65*).

Figure 65. Demo1 - USART communication output - Terminal software example



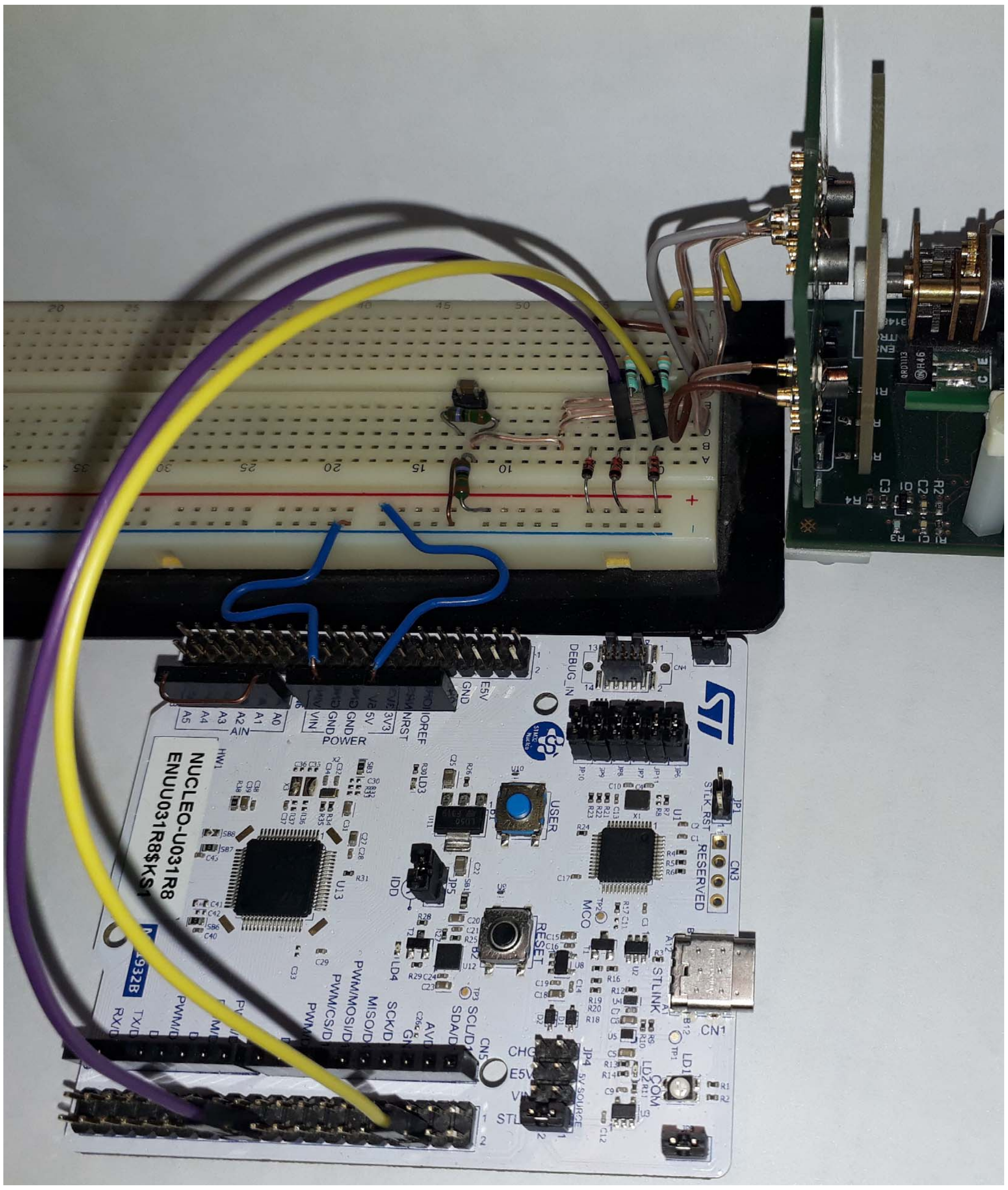
```
================================
Start Lc Sensor Demo 1
================================
CAL - start Calibration disabled
CAL - LcSensor1.CountDetect = 21
Status: NO METAL        Sensor1.CountValue:27   Sensor1.CountDetect:21  Edges:0
Status: NO METAL        Sensor1.CountValue:26   Sensor1.CountDetect:21  Edges:0
Status: METAL           Sensor1.CountValue:10   Sensor1.CountDetect:21  Edges:1
Status: METAL           Sensor1.CountValue:10   Sensor1.CountDetect:21  Edges:1
Status: METAL           Sensor1.CountValue:10   Sensor1.CountDetect:21  Edges:1
Status: NO METAL        Sensor1.CountValue:25   Sensor1.CountDetect:21  Edges:2
Status: NO METAL        Sensor1.CountValue:26   Sensor1.CountDetect:21  Edges:2
Status: NO METAL        Sensor1.CountValue:26   Sensor1.CountDetect:21  Edges:2
```

7.5.2 LC sensor metering counting demonstration with two sensors

This demonstration (*Figure 66*) uses two LC sensors with GPIO pins PC5, PB2, PA0, PC0 (one IO is needed for each sensor, and another two for the external connection of comparator output and low power timer input).

If the calibration is disabled or in static mode, no metal must be present over the sensors during the start of the system. In the case of dynamic calibration, the wheel must be in rotation.

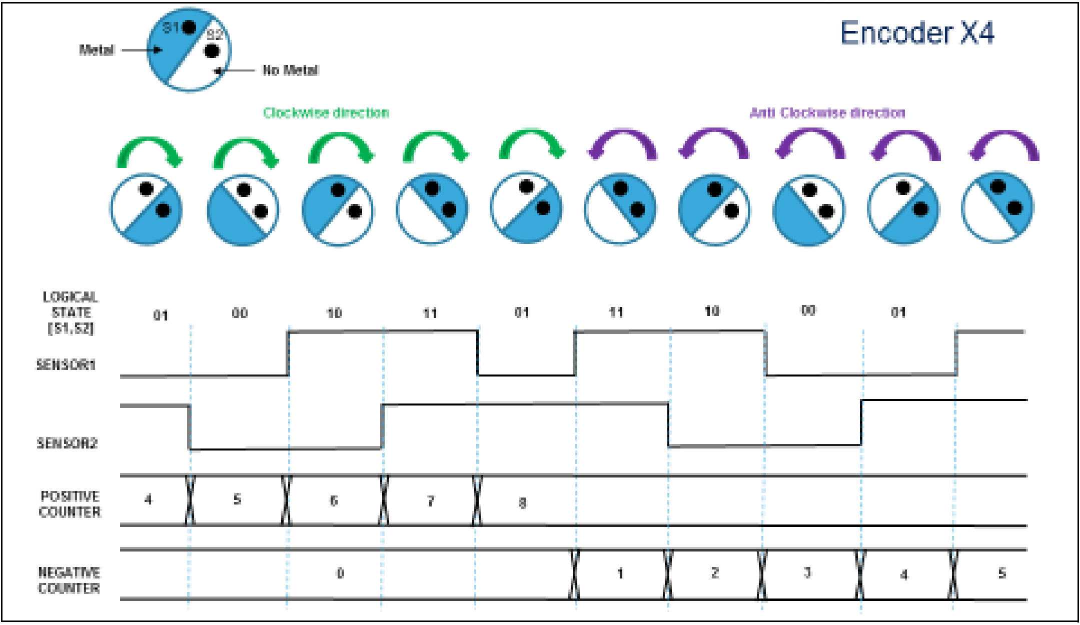
Figure 66. Hardware for the two sensors demonstration



It is possible to determine both the wheel rotation and its direction. Two counters are updated, one for the clockwise rotation and the other one for the anticlockwise rotation.

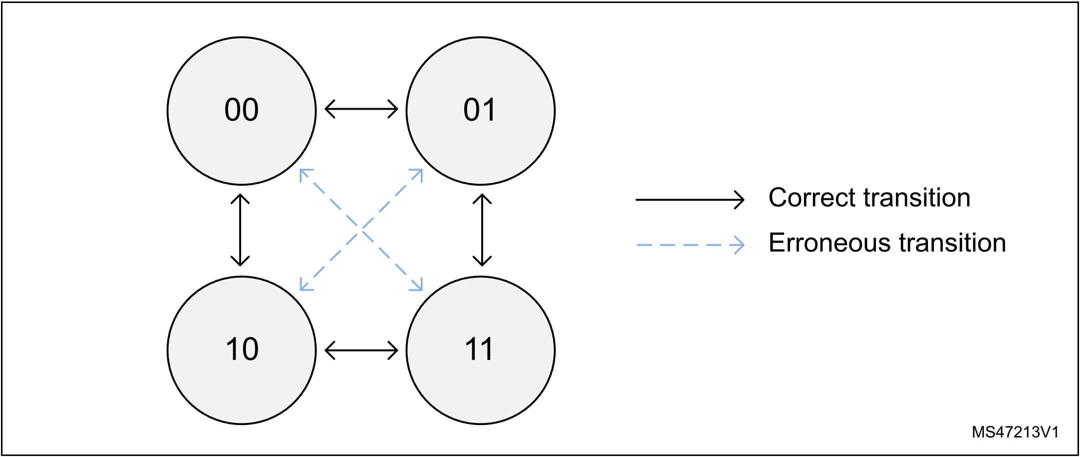
- Rotation and direction detection
- Clockwise counter and anticlockwise counter (negative or positive variation defined by previous sensor state)
- Four increments by rotation
- Sensors are spaced 90°

Figure 67. Demo2, two sensors



To interpret the results, a basic 4-state machine (*Figure 68*) has been put in place.

Figure 68. Basic 4-state machine



Erroneous transitions are detected with the LcStatus.Errors variable.

FW configuration

To enable this demonstration, LC_SENSOR_DEMO 2 define must be selected in the lc_sensor_metering.h file.

LcConfig and LcConfigSensorX variables, set in LcSensorConfig () function, must be defined according to application (sampling rate, DAC refresh, V_{cmp}, T_{excit}, $T_{capture}$, TimeBetweenSensors).

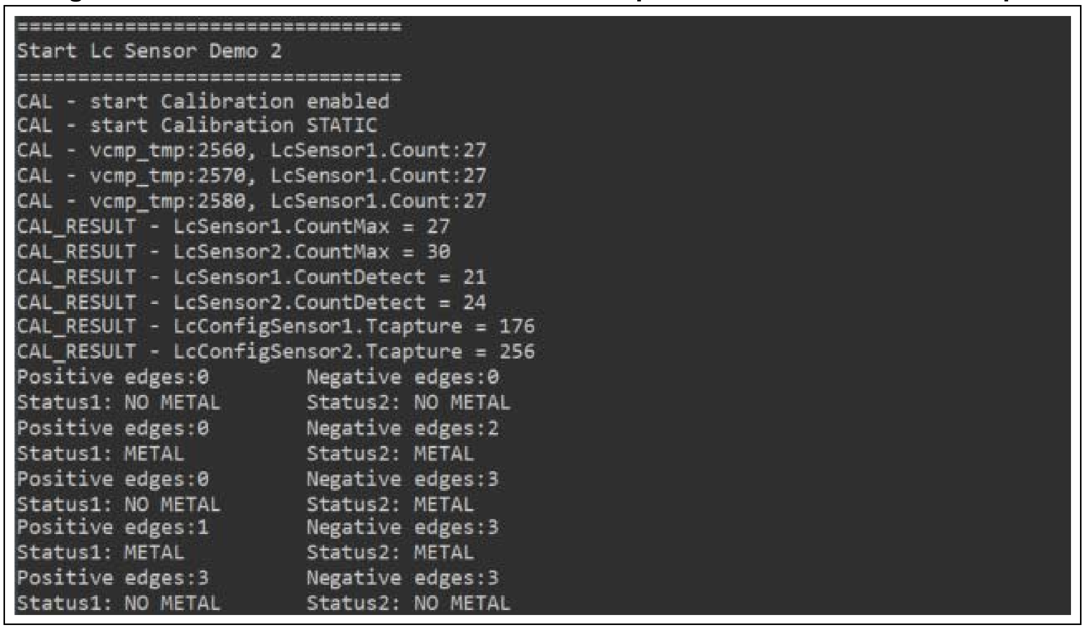
Outputs:

- LcSensor1.Status: Metal/No metal
- LcSensor2.Status: Metal/No metal

Based on X4 encoder result, two counters are updated with four increments per rotation:

- LcStatus.EdgeCountPos: positive counter
- LcStatus.EdgeCountNeg: negative counter

Figure 69. Demo2 - USART communication output - Terminal software example



```
================================
Start Lc Sensor Demo 2
================================
CAL - start Calibration enabled
CAL - start Calibration STATIC
CAL - vcmp_tmp:2560, LcSensor1.Count:27
CAL - vcmp_tmp:2570, LcSensor1.Count:27
CAL - vcmp_tmp:2580, LcSensor1.Count:27
CAL_RESULT - LcSensor1.CountMax = 27
CAL_RESULT - LcSensor2.CountMax = 30
CAL_RESULT - LcSensor1.CountDetect = 21
CAL_RESULT - LcSensor2.CountDetect = 24
CAL_RESULT - LcConfigSensor1.Tcapture = 176
CAL_RESULT - LcConfigSensor2.Tcapture = 256
Positive edges:0        Negative edges:0
Status1: NO METAL       Status2: NO METAL
Positive edges:0        Negative edges:2
Status1: METAL          Status2: METAL
Positive edges:0        Negative edges:3
Status1: NO METAL       Status2: METAL
Positive edges:1        Negative edges:3
Status1: METAL          Status2: METAL
Positive edges:3        Negative edges:3
Status1: NO METAL       Status2: NO METAL
```

In *Figure 69*, P stands for positive counter and N for negative counter.

7.5.3 LC sensor metering tachometer demonstration with two sensors

This demonstration uses two LC sensors, IO1, IO2 and IO3, and the same configuration of Demo2. It allows to collect the rotation speed and direction. For a clockwise direction the rotation per minute (RPM) number is positive, it is negative for anticlockwise rotation.

If the calibration is disabled or in static mode, no metal must be present over the sensors during the start of the system. In the case of dynamic calibration selection, the wheel must be in rotation.

FW configuration

To enable this demonstration, LC_SENSOR_DEMO 3 define must be selected in lc_sensor_metering.h file.

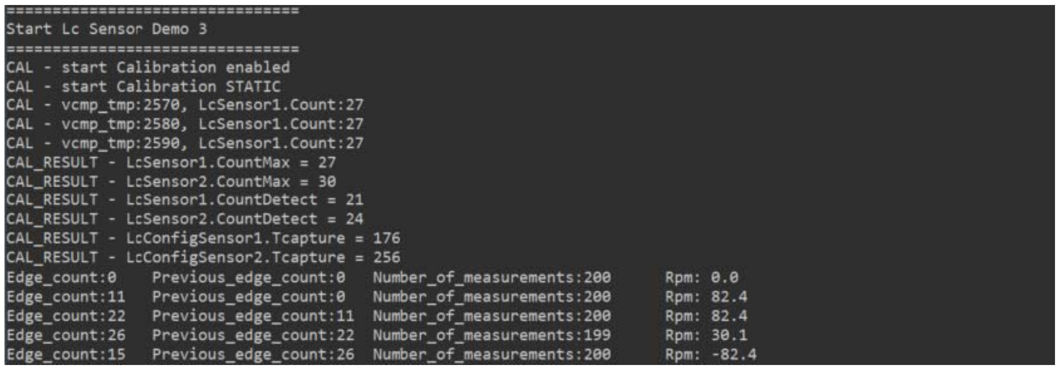
LcConfig and LcConfigSensorX variables, set in LcSensorConfig () function, must be defined according to application (sampling rate, DAC refresh, V_{cmp}, T_{excit}, $T_{capture}$, TimeBetweenSensors).

Outputs

The rotation per minute value is updated:

- LcStatus.EdgeCount: number of counted edges from both sensors
- LcStatus.PerviousEdgeCount: number of counted edges from both sensors before last RPM computing
- LcStatus.MeasuresCount: number of measurements since last RPM computing
- LcStatus.Rpm: rotations per minute

Figure 70. Demo3 - USART communication output - Terminal software example

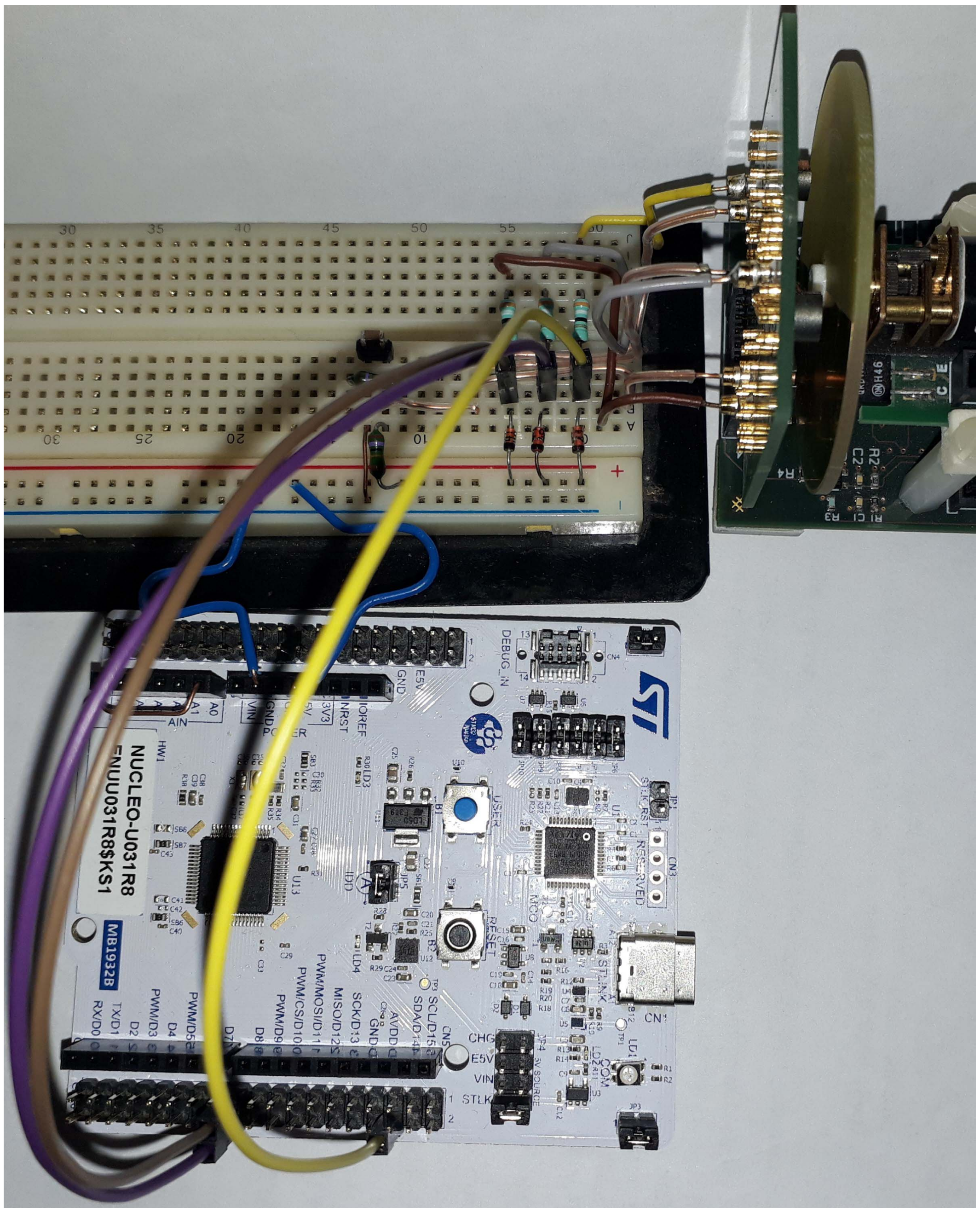


```
================================
Start Lc Sensor Demo 3
================================
CAL - start Calibration enabled
CAL - start Calibration STATIC
CAL - vcmp_tmp:2570, LcSensor1.Count:27
CAL - vcmp_tmp:2580, LcSensor1.Count:27
CAL - vcmp_tmp:2590, LcSensor1.Count:27
CAL_RESULT - LcSensor1.CountMax = 27
CAL_RESULT - LcSensor2.CountMax = 30
CAL_RESULT - LcSensor1.CountDetect = 21
CAL_RESULT - LcSensor2.CountDetect = 24
CAL_RESULT - LcConfigSensor1.Tcapture = 176
CAL_RESULT - LcConfigSensor2.Tcapture = 256
Edge_count:0    Previous_edge_count:0   Number_of_measurements:200      Rpm: 0.0
Edge_count:11   Previous_edge_count:0   Number_of_measurements:200      Rpm: 82.4
Edge_count:22   Previous_edge_count:11  Number_of_measurements:200      Rpm: 82.4
Edge_count:26   Previous_edge_count:22  Number_of_measurements:199      Rpm: 30.1
Edge_count:15   Previous_edge_count:26  Number_of_measurements:200      Rpm: -82.4
```

7.5.4 LC sensor metering tachometer demonstration with three sensors

This demonstration uses three LC sensors with GPIO pins PC5, PB2, PA9, PA0, PC0 (one IO for each sensor, another two for the external connection of comparator output and low power timer input). It allows to collect the rotation speed and the direction. For a clockwise direction the rotation per minute (RPM) number is positive, it is negative for anticlockwise rotation. Error detection is displayed.

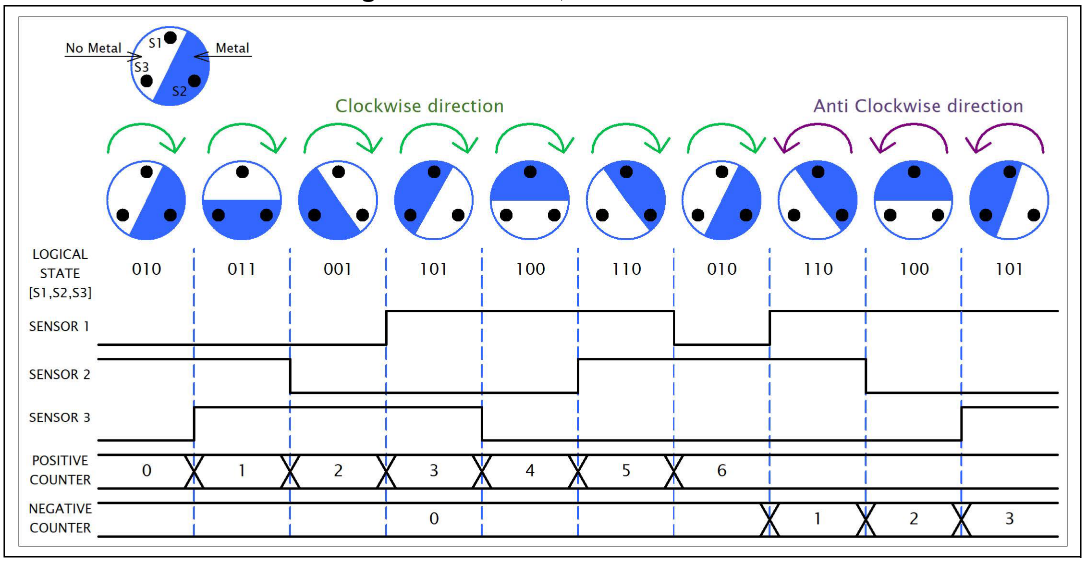
Figure 71. Hardware for the three sensors demonstration



It is possible to determine both the wheel rotation and its direction. Two counters are updated, one for the clockwise rotation and the other one for the anticlockwise rotation.

- Rotation and direction detection
- Clockwise and anticlockwise counter (negative or positive variation defined by previous sensor state)
- Six increments by rotation
- Sensors are spaced 120°
- Error counter is increased, if more than one sensor registers transition between Metal and No Metal state in the same time

Figure 72. Demo4, three sensors



FW configuration

To enable this demonstration, LC_SENSOR_DEMO 4 define must be selected in lc_sensor_metering.h file.

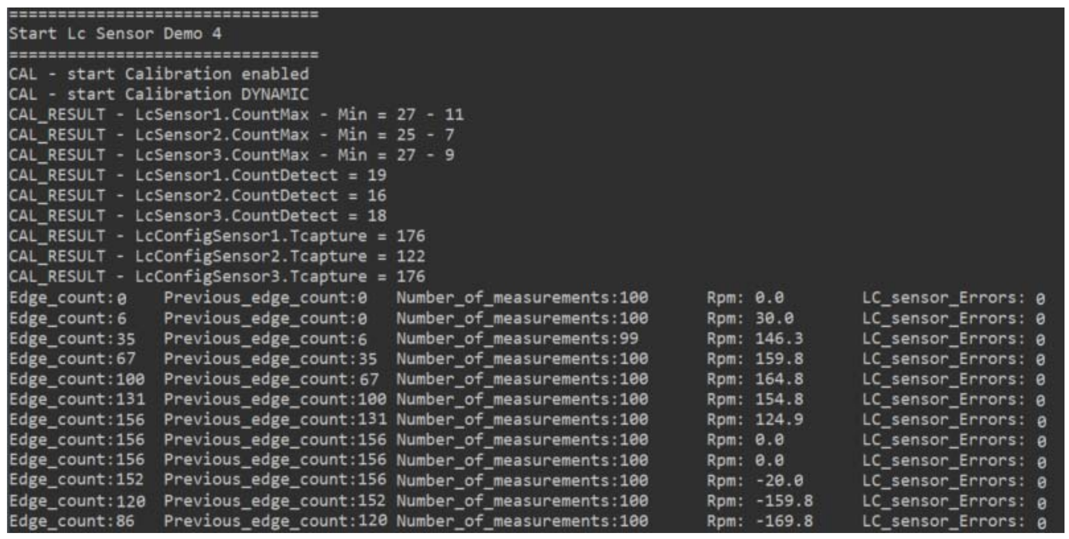
LcConfig and LcConfigSensorX variables, set in LcSensorConfig () function, must be defined according to application (sampling rate, V_{cmp}, T_{excit}, $T_{capture}$, TimeBetweenSensors).

Outputs

The rotation per minute value is updated:

- LcStatus.EdgeCount: number of counted edges from both sensors
- LcStatus.PerviousEdgeCount: number of counted edges from both sensors before last RPM computing
- LcStatus.MeasuresCount: number of measurements since last RPM computing
- LcStatus.Rpm: rotations per minute
- LcStatus.Errors: number of incorrect transitions (more than one sensor registers transitions in the same time)

Figure 73. Demo4 - USART communication output - Terminal software example



```
================================
Start Lc Sensor Demo 4
================================
CAL - start Calibration enabled
CAL - start Calibration DYNAMIC
CAL_RESULT - LcSensor1.CountMax - Min = 27 - 11
CAL_RESULT - LcSensor2.CountMax - Min = 25 - 7
CAL_RESULT - LcSensor3.CountMax - Min = 27 - 9
CAL_RESULT - LcSensor1.CountDetect = 19
CAL_RESULT - LcSensor2.CountDetect = 16
CAL_RESULT - LcSensor3.CountDetect = 18
CAL_RESULT - LcConfigSensor1.Tcapture = 176
CAL_RESULT - LcConfigSensor2.Tcapture = 122
CAL_RESULT - LcConfigSensor3.Tcapture = 176
Edge_count:0    Previous_edge_count:0   Number_of_measurements:100      Rpm: 0.0        LC_sensor_Errors: 0
Edge_count:6    Previous_edge_count:0   Number_of_measurements:100      Rpm: 30.0       LC_sensor_Errors: 0
Edge_count:35   Previous_edge_count:6   Number_of_measurements:99       Rpm: 146.3      LC_sensor_Errors: 0
Edge_count:67   Previous_edge_count:35  Number_of_measurements:100      Rpm: 159.8      LC_sensor_Errors: 0
Edge_count:100  Previous_edge_count:67  Number_of_measurements:100      Rpm: 164.8      LC_sensor_Errors: 0
Edge_count:131  Previous_edge_count:100 Number_of_measurements:100      Rpm: 154.8      LC_sensor_Errors: 0
Edge_count:156  Previous_edge_count:131 Number_of_measurements:100      Rpm: 124.9      LC_sensor_Errors: 0
Edge_count:156  Previous_edge_count:156 Number_of_measurements:100      Rpm: 0.0        LC_sensor_Errors: 0
Edge_count:156  Previous_edge_count:156 Number_of_measurements:100      Rpm: 0.0        LC_sensor_Errors: 0
Edge_count:152  Previous_edge_count:156 Number_of_measurements:100      Rpm: -20.0      LC_sensor_Errors: 0
Edge_count:120  Previous_edge_count:152 Number_of_measurements:100      Rpm: -159.8     LC_sensor_Errors: 0
Edge_count:86   Previous_edge_count:120 Number_of_measurements:100      Rpm: -169.8     LC_sensor_Errors: 0
```

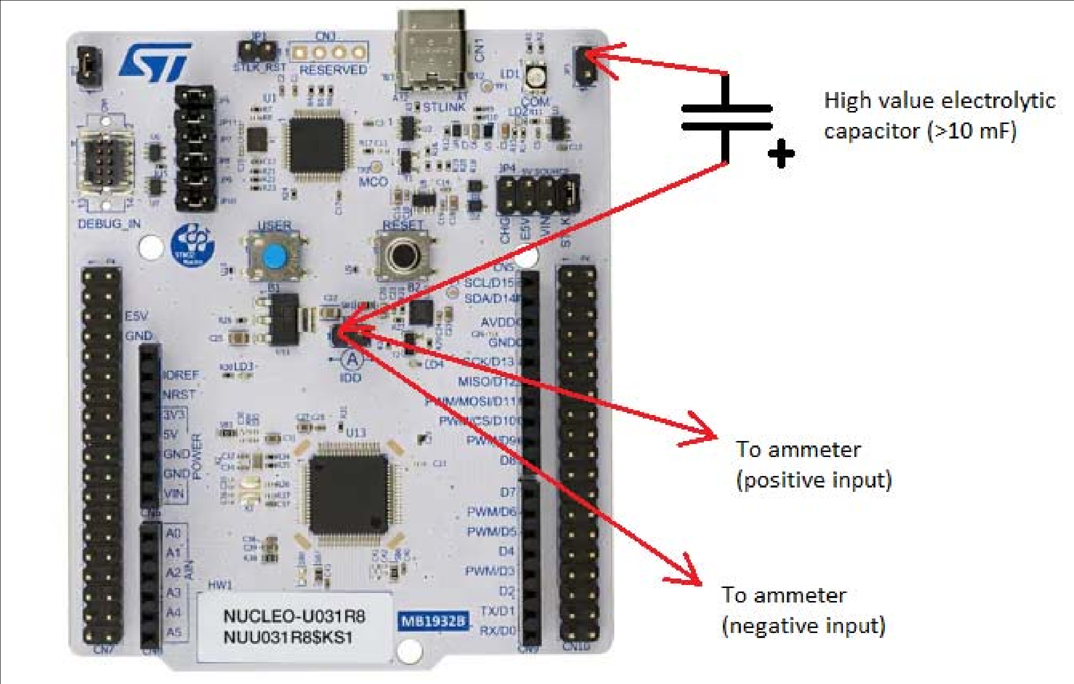
7.5.5 LC sensor metering counting in low power mode

To perform consumption measurements, check that the application is running in low power mode:

- USART communication is disabled (it is enabled by default)
- User LED is disabled (it is disabled by default), selection is done by SW

To measure the average current during the LC sensor metering in low-power mode (a few μA), an ammeter can be inserted on JP5 between pin 1 (VDD_MCU) and pin 2 (VDD 3.3V). Special attention must be taken to measure pulsed currents (see *Section 2.5.2*).

Figure 74. Connections for the measurement of power consumption


High value electrolytic capacitor (>10 mF)
+
To ammeter (positive input)
To ammeter (negative input)
NUCLEO-U031R8
NUU031R8$KS1
MB1932B
IDD

8 Firmware description - STM32U031R8-NUCLEO

8.1 STM32U031 peripherals used by the application

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

GPIO

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

- PA0 comparator output, for the external connection between comparator and low power timer
- PC0 low power timer input, for the external connection between comparator and low power timer
- PC5 to drive the LC sensor 1
- PB2 to drive the LC sensor 2
- PA9 to drive the LC sensor 3
- PA5 user LED, can be disabled in software
- PA2 USART transmitter, can be disabled in software
- PA3 USART receiver, can be disabled in software

DAC

DAC_OUT1 is used to generate the input threshold voltage needed by the comparator. This reference voltage provides a preset threshold voltage value near V_{DD} / 2, but with a necessary positive margin to eliminate noise. It allows to calibrate the threshold value according to different LC network responses and environment variations.

DAC outputs are configured in Sample and Hold mode and enabled/disabled in RTC_WakeUp interrupt.

COMP1

The comparator is connected to LC sensors. As soon as the excitation pulse has been delivered, the COMP1 positive input collects analog oscillations referenced to V_{DD} / 2 bias, while its negative input is fed by the DAC threshold voltage. The LC sensor oscillations are conditioned and this results in digital pulses delivered on the comparator output. This signal is transmitted by external connection to the LPTIM.

LPTIM1

The low power timer is 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 from the previous one to get the number of collected pulses for the corresponding measurement. This avoids a time consuming reset.

RTC

This peripheral is used to wake up the microcontroller at regular intervals once the MCU has entered STOP mode. The RTC clock is connected to the LSE clock which frequency is 32.768 kHz.

The WakeUpTimer is used to perform LC measurements. The wake-up frequency corresponds to the LC sensor capture frequency and is set to 50 Hz for this example. This frequency can be increased or decreased, depending upon the rotating wheel maximum speed. Modifications of the capture frequency change the average power consumption.

Alarm A is used to perform communications with user or others peripherals to interpret LC sensors results. For this demo, results are send by USART2 communication.

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.

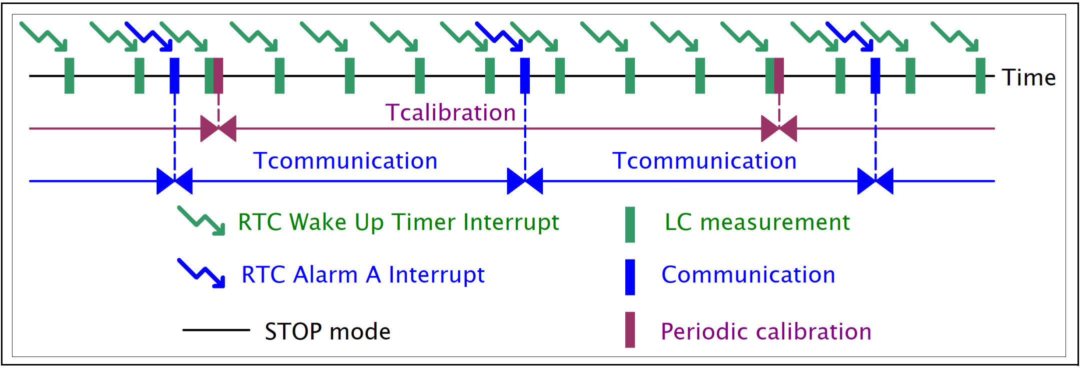
Interrupts

To save the maximum time and power consumption in MCU RUN mode, the RTC Wake up Timer IRQ handler is directly processed inside the interrupt subroutine.

Both RTC Wake Up Timer and RTC Alarm A interrupts are managed by common RTC call back. Periodic calibration timing is done by software timer in LC measurement interrupt (see *Figure 75*).

Alarm and Wake Up interrupt must have the same priority because they are executed by the same IRQ handler. Frequency is application dependent, it can range from several seconds to several hours for user communications (Alarm A).

Figure 75. Interrupt sequence



LC measurements are ongoing during communication and calibration phases, hence Alarm A execution must be performed between two measurements (between two wake-up interrupts), whose execution time must be controlled.

8.2 Description of flows

This software demonstration allows the user to manage from one to three sensors sequentially. Hardware configuration is done in “lc_sensor_metering.h” file by define instructions.

Four demonstrations are available:

- LC_SENSOR_DEMO 1: Basic counting demonstration with one sensor
- LC_SENSOR_DEMO 2: Counting demonstration with two sensors – Rotation and
- direction
- LC_SENSOR_DEMO 3: Tachometer demonstration with two sensors – Speed and
- direction
- LC_SENSOR_DEMO 4: Tachometer demonstration with three sensors – Speed, direction and error state counter

These examples are launched from a “main.c” file, used also for the configuration of used peripherals. LC sensor metering functions are in “lc_sensor_metering.c” file and called by “main.c”.

Inizialization

- Main function main(): initializes all peripherals
- Configuration function LcSensorConfig (): fills all structures required to set LC sensor configuration
- Demo function LcSensorDemo (): starts counting demonstration
- Initialization function LcSensorInit (): allows to set all variables used by application.
- StopEntry () function: prepares MCU to enter in Stop2 modes

Calibration functions

- LC_Calibration (): first calibration performed at the power up
- PeriodicCalibration(): periodic calibration function

Interrupts

- RTC_WKUP_IRQHandler () function to manage LC measurements:
  - LC sensors excitation
  - Process sensor state
  - Execute state machine
  - Process periodic calibration if required
- HAL_RTC_AlarmAEventCallback () function to communicate with user by USART

Demonstration selection

- LC_SENSOR_DEMO 1: Basic counting demonstration with one sensor
- LC_SENSOR_DEMO 2: Counting demonstration with two sensors and rotation/direction detection
- LC_SENSOR_DEMO 3: Tachometer demonstration with two sensors and speed/direction detection
- LC_SENSOR_DEMO 4: Tachometer demonstration with two sensors and speed/direction detection, error state counter
- Communication with PC terminal
  - USART2_ENABLE 0: USART disabled
  - USART2_ENABLE 1: USART enabled
- LC excitation mode selection

  - LC_EXCIT_MODE 0: LC sensor excitation to V_{SS}
  - LC_EXCIT_MODE 1: LC sensor excitation to V_{DDA}
- Enabled LED to signal sensor1 status
  - LC_LED_ENABLE 0: LED disabled
  - LC_LED_ENABLE 1: LED enabled
- Enabled debugger (GPIOA used by debugger does not switch to analog state):
  - LC_DEBUG_ENABLE 0: Debugger disabled
  - LC_DEBUG_ENABLE 1: Debugger enabled

This file contains all macros used to manage LC sensor oscillations:

- __LC_WAIT (): to insert a waiting time for peripherals stabilization
  - Set TIM6 in single shot
  - Start TIM6
  - Wait For Event (time to stabilize peripherals)

LcSensorConfig () function

This function is used to configure the application. Two structures are available to configure and manage LcSensor application:

1. *LcConfig* to configure application
   - Sampling: value in Hz (up to 500, default is 50)
   - TimeBetweenSensor (default value is 4236)
   - DacVcmp (default value is 2500)
   - FirstCal
     - Status: CAL_ENABLED (default) or CAL_DISABLED
     - Mode: STATIC (default) or DYNAMIC
     - Measures: minimum measurements number to perform during calibration phase (default is 75)
     - DacVcmpMax: starting value for Vcmp calibration (default value is 3103)
     - DacVcmpMin: stop value for Vcmp calibration (default value is 2450)
     - DacVcmpStep: step value for Vcmp calibration (default value is 10)
     - PulseCount: number of targeted pulses (default is 27), used for STATIC calibration
     - PulseCountDeltaMin: sets the needed difference e between min and max pulse count (default is 10), used for DYNAMIC calibration
   - PeriodicCal
     - Status: CAL_ENABLED or CAL_DISABLED (default)
     - Mode: STATIC or DYNAMIC (default)
     - CalibrationPeriod: number of measurements between each calibration (default is 1000)
     - CalibrationMeasurements: number of measurements during calibration (default is 50)
     - CountDrift: valid periodic calibration values only if the drift does not exceed this value (default is 8)

2. *LcConfigSensorX* to configure sensors (X = 1, 2, or 3)
   - FreqOsc: Theoretic LC oscillations frequency (default 890000)
   - CountDetectPercent: Detection threshold in percent ex: 0.8 – Not used if dynamic calibration is enabled
   - Texcit: Excitation time, minimum value is 1 (default value is 1)
   - Tcapture: Capture time (default value is 636), value updated automatically when calibration is enabled

LcSensorinit () function

This function is used to initialize structures used by the application with default values. It must be called before wake-up interrupt enable (called in *LcSensorDemo ()* and *StopEntry ()* functions.

Two structures are used to store results updated at each measurements (such as sensors status, counter status, errors):

1. *LcStatus* to store LC application status
   - MeasuresCount: measurement counter
   - EdgeCount: edge counter
   - EdgeCountPos: positive edge counter
   - EdgeCountNeg: negative edge counter
   - Errors: error counter
   - Rpm: rotation per minute value
   - Rps: rotation per seconds value
   - WakeUpCounter: counter value loaded in RTC WakeUp timer
   - Sampling: real sampling value based on WakeUpCounter value
2. *LcSensorX* to store individual sensor results
   - Status: NO_METAL (0) or METAL (1)
   - PreviousStatus: NO_METAL (0) or METAL (1)

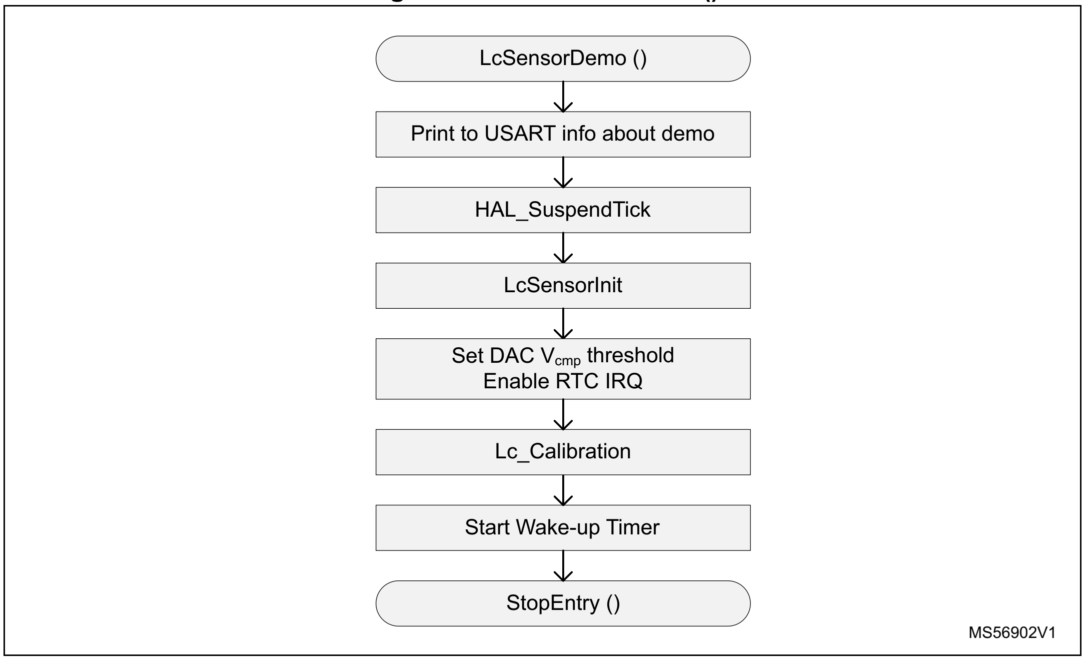
LcSensorDemo () function

As soon as this function is fully executed, an RTC interrupt is created to wake up the microcontroller at regular intervals. This function is then called once and the application alternates STOP sequences and RUN slots at the frequency rate given by the RTC wake-up parameters. In this example, the microcontroller is woken up 50 times per second.

If USART is enabled, information about actual demo is printed to USART and needed settings before calibration are done and next the calibration function is executed.

Then, the StopEntry() function is executed to prepare the device to enter STOP mode with specific settings, and the microcontroller is put in an infinite loop, waiting for the RTC wake-up interrupt. All processes are performed in interrupt, using sleep on exit mode, thus nothing to do in main.

Figure 76. LcSensorDemo ()



StopEntry () function

The aim of this function is to prepare the device to enter STOP2 mode with parameters set to minimize the device power consumption.

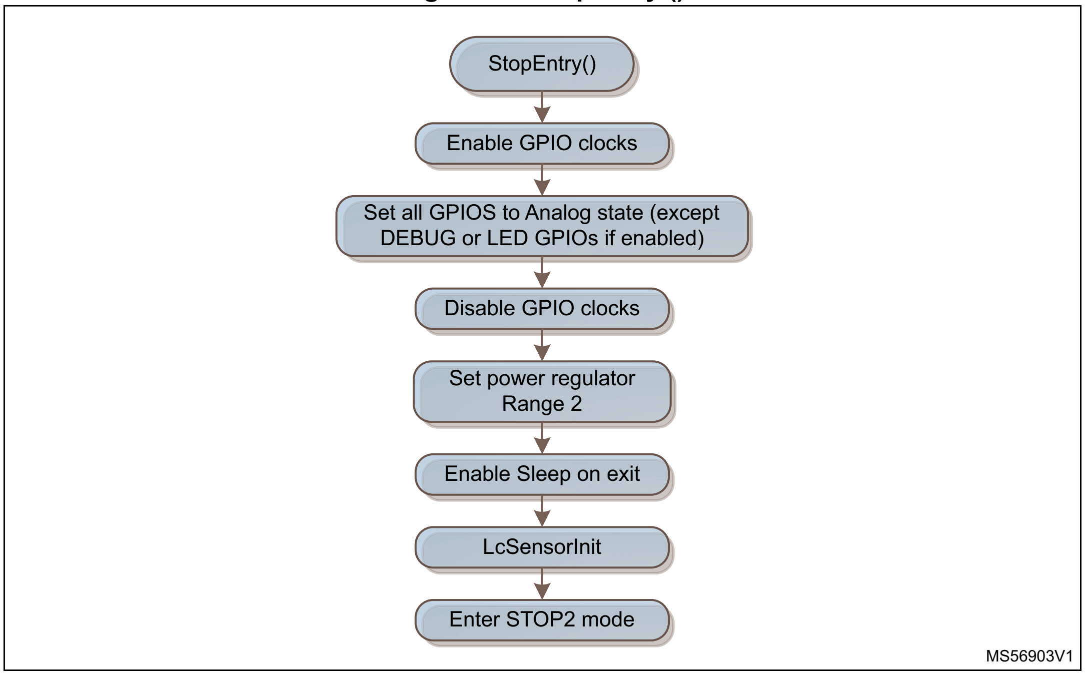
If DEBUG or LED are not enabled, all GPIOs are put in Analog state to minimize power consumption, and the associated clocks are disabled.

Voltage scaling is set to Range 2 to optimize the power consumption.

Sleep on Exit is used to force the MCU enter the STOP mode as soon as the RTC wake-up has been serviced and executed. In that case, the StopEntry () function is executed once, and the microcontroller returns in STOP mode after each interrupt.

Lc variables are initialized with LcSensorInit (), the MCU enters in Stop2 mode, and waits for an interrupt.

Figure 77. StopEntry ()



RTC_WKUP_IRQHandler () interrupt subroutine

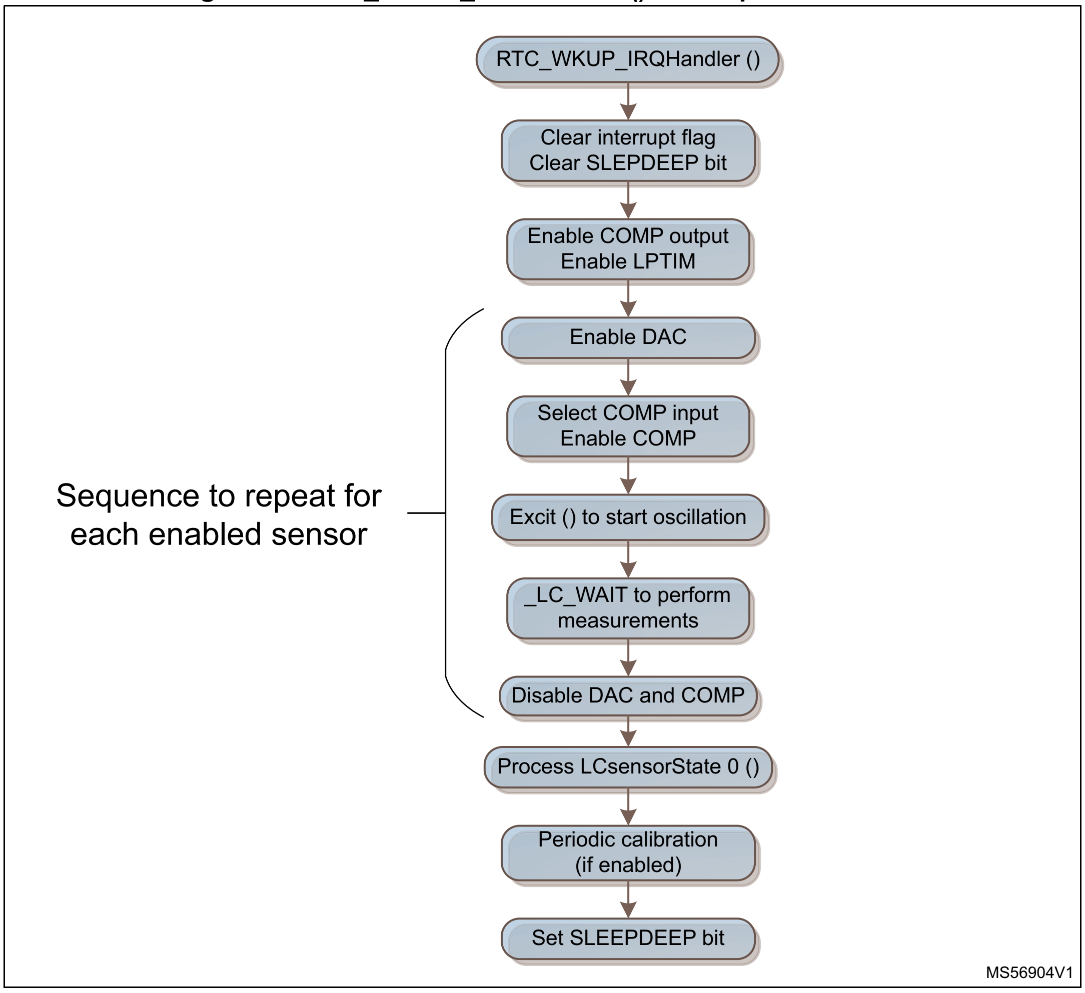
The RTC Wake Up IRQ Handler manages the RTC interrupt triggered by the RTC Wake Up timer. To save time during the interrupt execution and since the device operates in RUN mode, the interrupt subroutine is executed in the Low Power mode, without performing any call back function in the main program.

The interrupt flag is cleared, then COMP and DAC peripherals are switched on, and a delay is inserted to wait for the peripherals to be operational. The excitation pulse is delivered by IO pin placed in GPIO output mode. As soon as the pulse is applied on the LC network, IO switches into Analog state, and acts as the non-inverting comparator input.

The application waits for a capture time in SLEEP mode until all the pulses are collected. If more than one sensor is enabled, a waiting time in SLEEP mode is added to let a stabilization time between the two measurements. Another excitation pulse is delivered on the next sensor with a new capture time in SLEEP mode until all the pulses are collected.

At the end of measurements, DAC and COMP are disabled to minimize power consumption. The capture of the LC sensor oscillations has been performed during this interrupt.

Figure 78. RTC_WKUP_IRQHandler () interrupt subroutine



ProcessLCsensorState () function

Includes processing of all measured values for each sensors. LC sensor state (METAL / NO METAL) and transitions between these states are specified.

Excit () function

This function excites LC sensor pin to V_{SS} or V_{DD} (depending upon LC_EXCIT_MODE) to measure delay time and then turn LC sensor pin back to analog mode.

This function provides time critical operation, therefore all variables are declared as register.

LC_calibration () function

This function is executed only if LcConfig.FirstCal.Status variable is set to CAL_ENABLED. The aim of this function is to perform a calibration at the power up.

Maximum and minimum pulse counts, capture time and detection threshold are updated during this calibration according to components and external conditions.

Two examples are available, “STATIC” or “DYNAMIC”

1. Static method example (see *Figure 79*)

   There is no metal in the proximity of LC sensor, so only the maximum pulse count is used to define the detection threshold count number and capture time for measurements.

2. Dynamic method example (see *Figure 80*)

   A wheel is in rotation with metal and no metal transitions. The maximum and minimum pulse counts are used to define the detection threshold count number and capture time for measurements.

Figure 79. LC_calibration () function - Static method

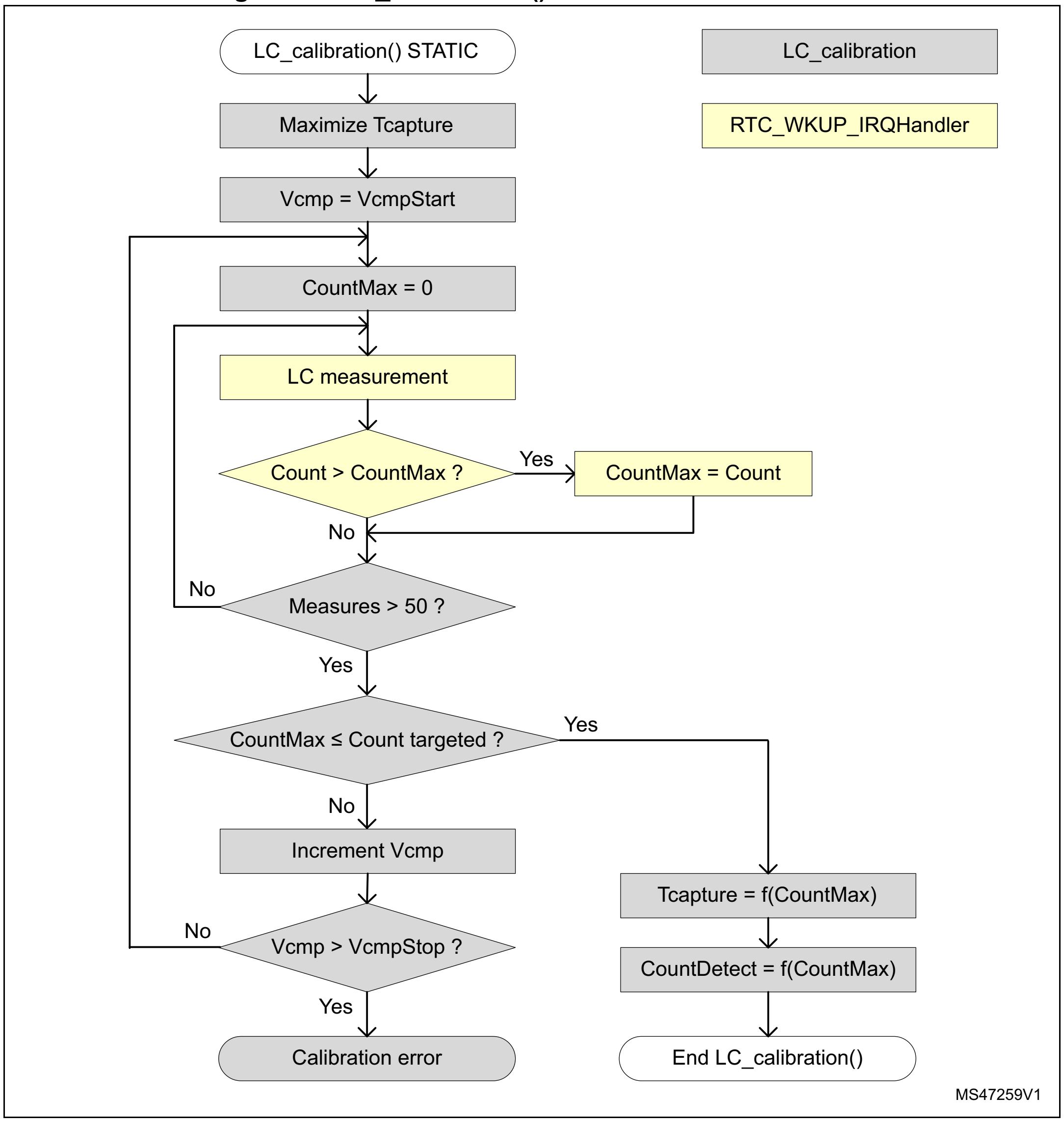
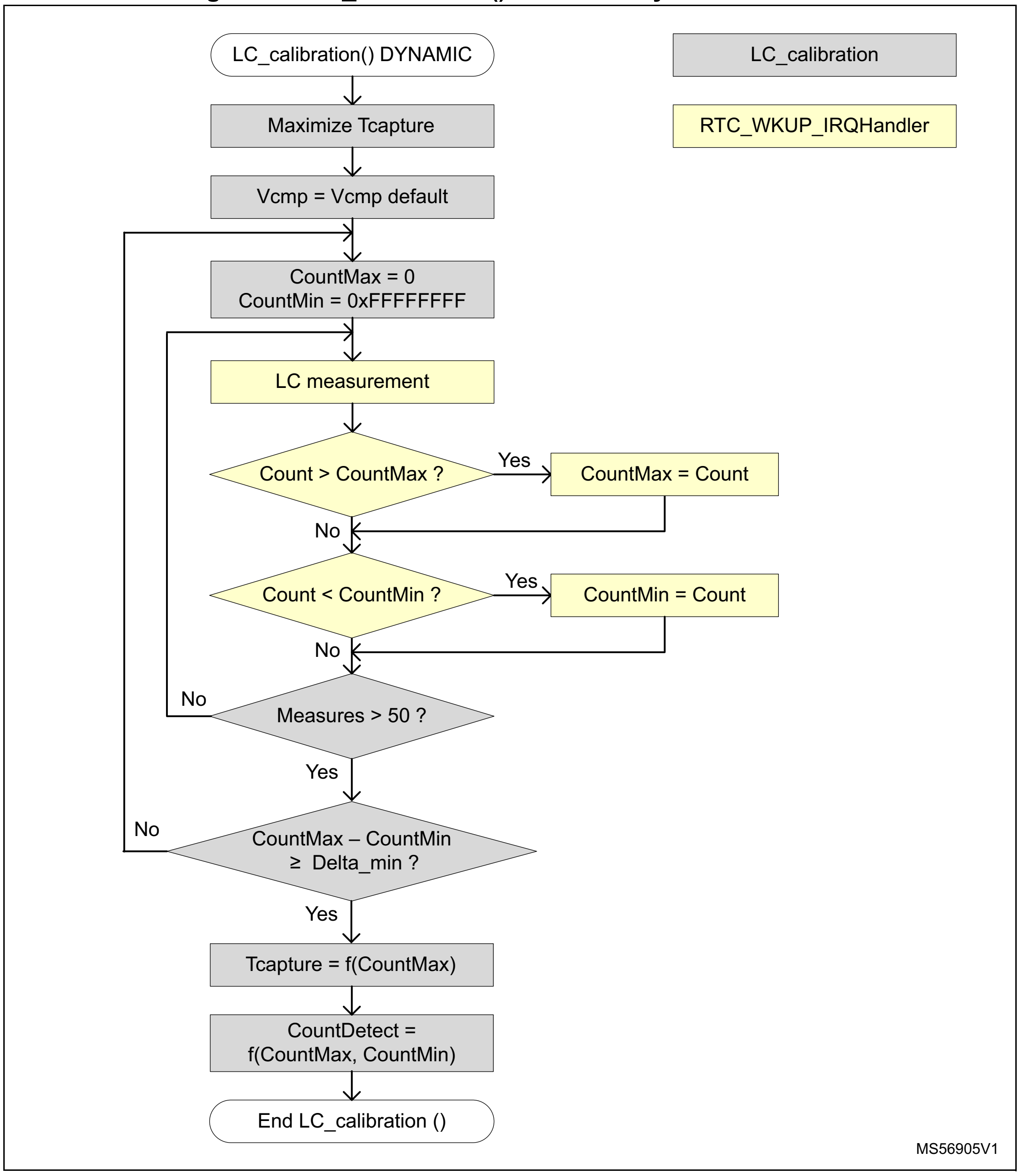


Figure 80. LC_calibration () function - Dynamic method



PeriodicCalibration () function

This function is executed only if LcConfig.PeriodicCal.Status variable is set to CAL_ENABLED.

All calibration operations are managed in WakeUp timer interrupt handler and must be done between two LC measurements.

Calibration measurements are performed during standard measurements in WakeUp timer interrupt handler.

The periodic calibration is performed in three steps:

1. Calibration initialization, executed in RTC_WKUP_IRQHandler() after Periodic calibration counter overflows period threshold. Capture time is maximized to catch all pulses count and calibration variables are initialized for each sensor.
   - PeriodicCal.SensorXCountMax = 0, with X = 1, 2, or 3
   - PeriodicCal.SensorXCountMin = 0xFFFFFFFF, with X = 1, 2, or 3

   After that, the threshold of Periodic calibration counter is reset to perform few calibration measurements.
2. Calibration measurements, executed in RTC_WKUP_IRQHandler()

   These measurements are performed in a routine executed during normal operation in WakeUp timer interrupt handler.
3. Calibration results analysis, and value update, executed in RTC_WKUP_IRQHandler() in function PeriodicCalibration()

The flow depends upon the selected method.

Figure 81. PeriodicCalibration () function

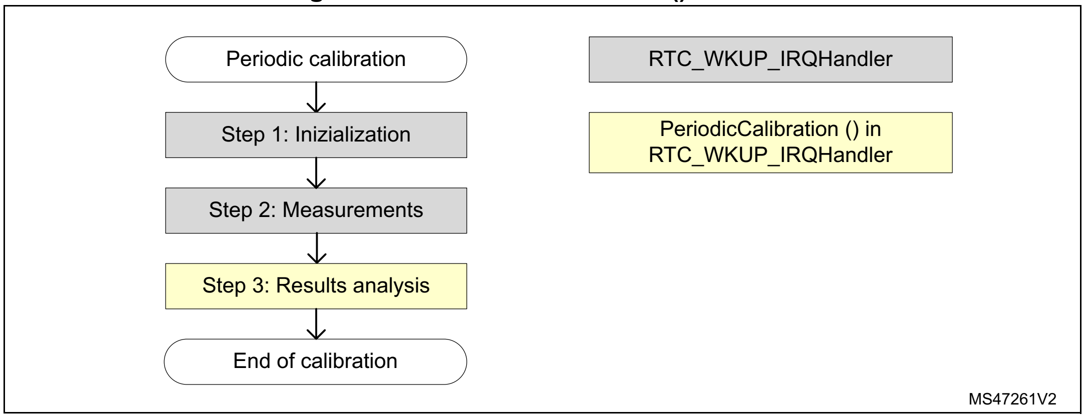
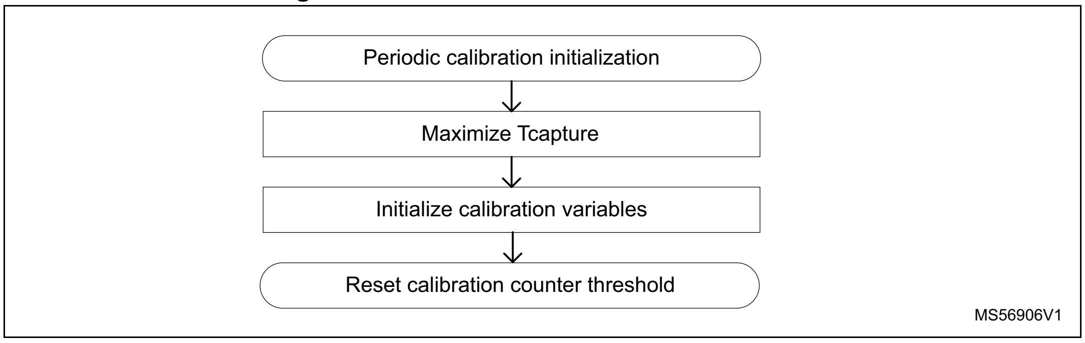


Figure 82. Periodic calibration initialization


Periodic calibration initialization
Maximize Tcapture
Initialize calibration variables
Reset calibration counter threshold
MS56906V1

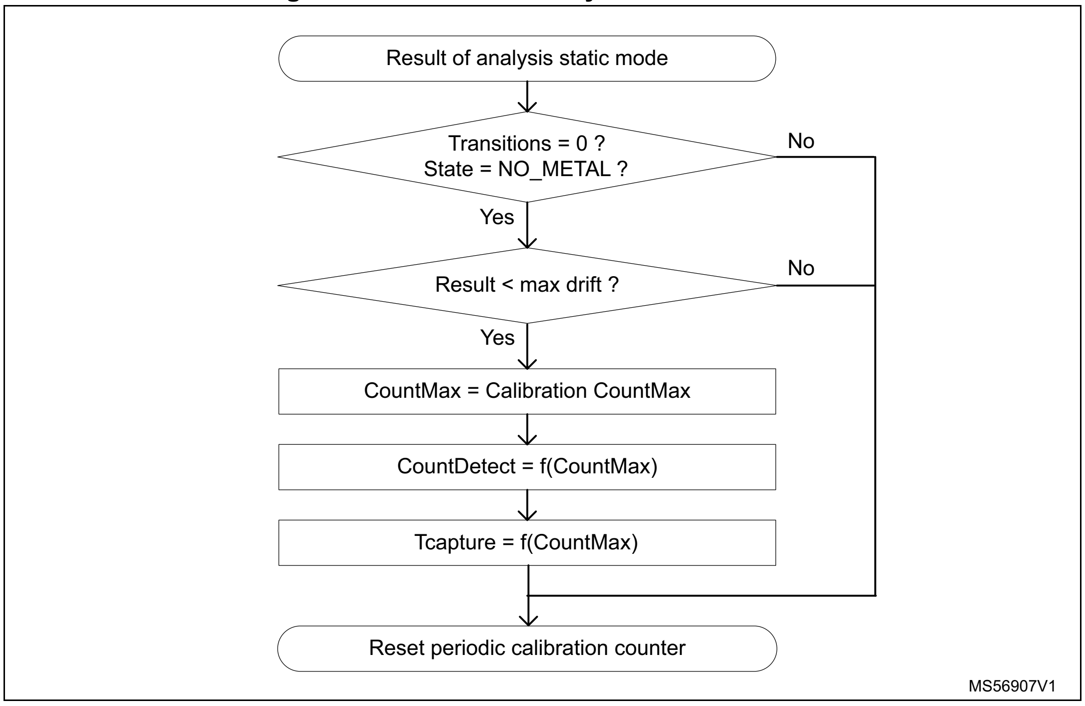
Static method example

To check that calibration results are valid, the following points must be respected:

- No transitions during calibration phase
- Sensor state is no metal

To eliminate noisy results, a maximum drift is allowed. After having checked the results, the new detection threshold counter and the new capture time are computed. After that, the Periodic calibration counter is reset for the next periodic calibration.

Figure 83. Results of analysis in static mode


Result of analysis static mode
Transitions = 0 ?
State = NO_METAL ?
No
Yes
Result < max drift ?
No
Yes
CountMax = Calibration CountMax
CountDetect = f(CountMax)
Tcapture = f(CountMax)
Reset periodic calibration counter
MS56907V1

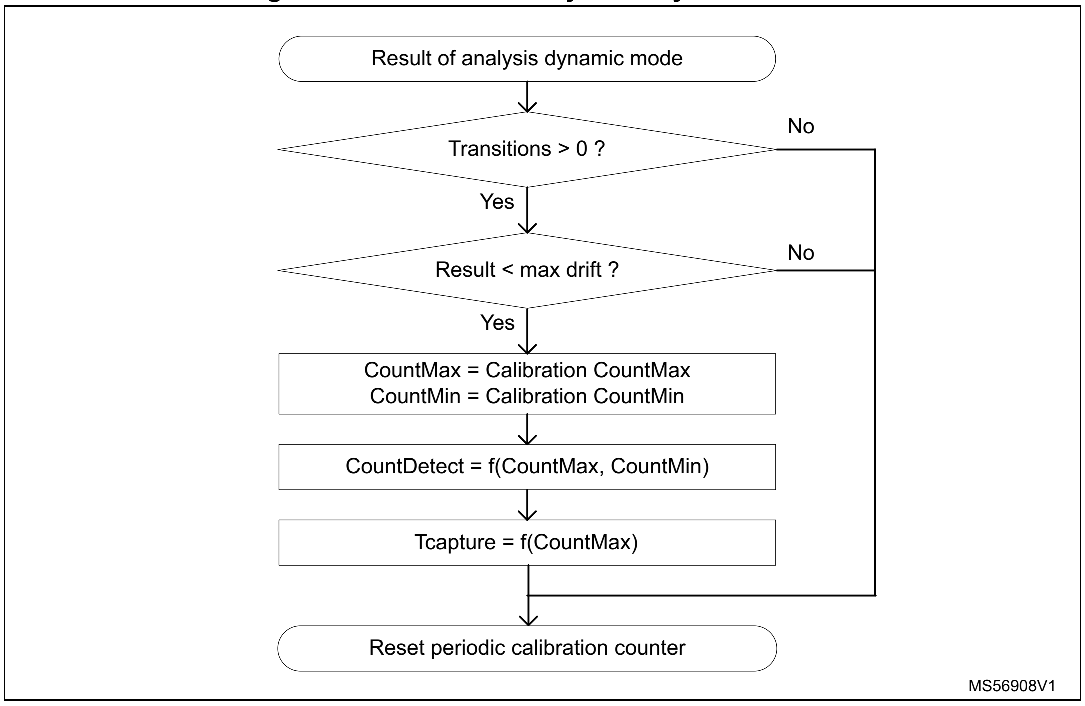
Dynamic method example

To check that calibration results are valid, the number of transitions must be non null to catch metal, and no metal states for each used sensor.

To eliminate noisy results, a maximum drift is allowed

After results validation, a new detection threshold counter and a new capture time are computed. After that, the Periodic calibration counter is reset for the next periodic calibration.

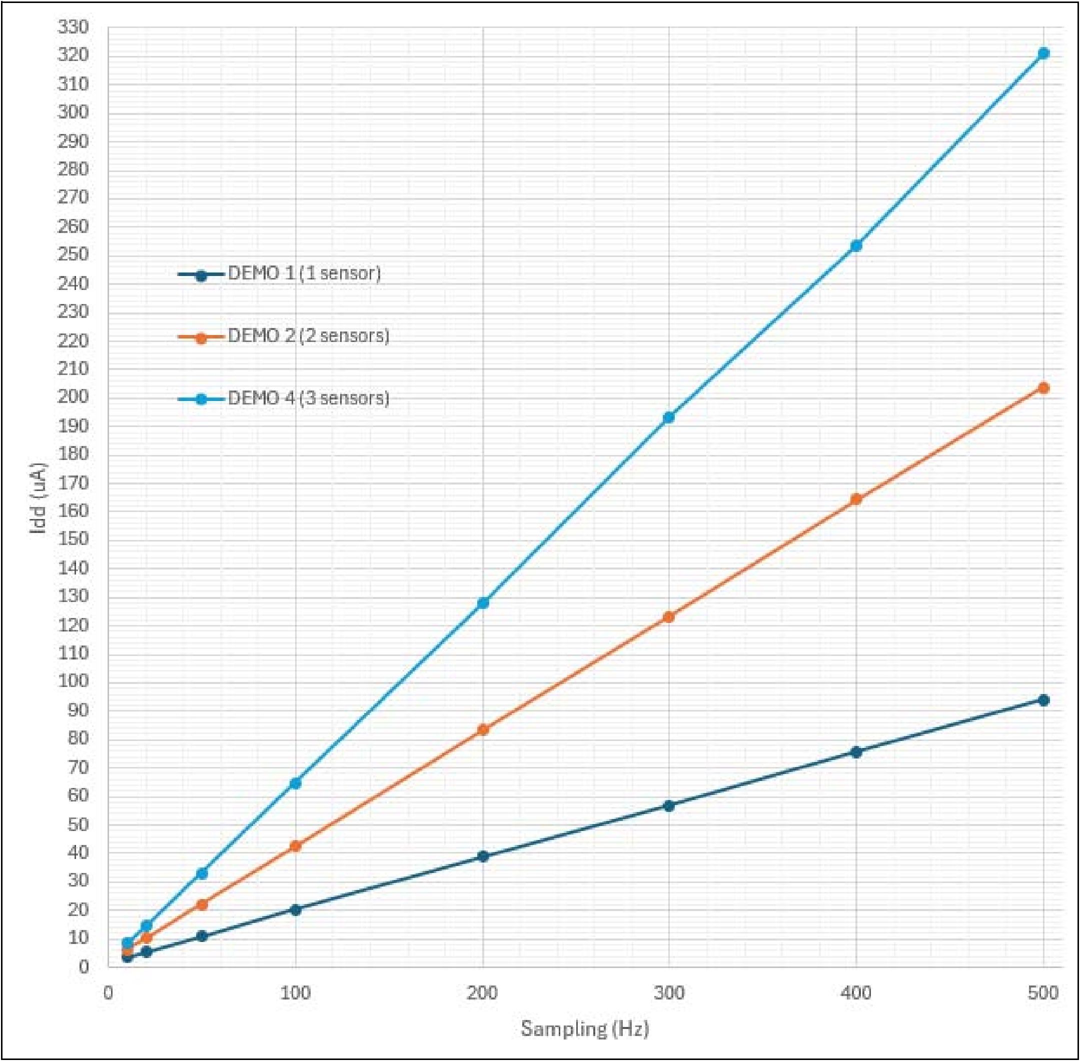
Figure 84. Results of analysis in dynamic mode


Result of analysis dynamic mode
Transitions > 0 ?
No
Yes
Result < max drift ?
No
Yes
CountMax = Calibration CountMax
CountMin = Calibration CountMin
CountDetect = f(CountMax, CountMin)
Tcapture = f(CountMax)
Reset periodic calibration counter
MS56908V1

9 Power consumption - STM32U031R8-NUCLEO

As shown in *Figure 85*. the power consumption depends upon the application. The main parameters to consider are the number of sensors, the inductance, the capacitor values, and the sampling rate. Vmid source power consumption, which in this example is added by a resistor divider (2 x 270 kΩ) with continuous consumption 6.1 μA, is not included.

Figure 85. STM32U031R8 - I_{DD} (average) vs. sampling rate (V_{DD} = 3.3 V)



10 Conclusion

Digital flow meters are gradually replacing systems based on mechanical switches and obsolete technologies. They are more accurate and bring much more flexibility and security.

The examples based on STM32L073Z-EVAL, STM32L476RG-NUCLEO, and STM32U031R8-NUCLEO evaluation boards describe the design of a 3-sensor LC metering system that can be used in gas or water meter applications. They show reliable approach of the LC sensor metering feature, and indicate possible further developments, based on these principles.

This LC sensor metering demonstrations use microcontroller peripherals, and highlight how they can be configured to operate in low power mode, to minimize power consumption. This is a key factor for battery-powered gas or water meter applications.

The methodology described in this document can be easily reused to target other developments, which can specifically fulfill other user needs.

11 Revision history

Table 5. Document revision history

Date	Revision	Changes
09-Apr-2015	1	Initial release.
26-Sep-2017	2	Introduced STM32L476RG-NUCLEO board and X-CUBE-LCSENSOR firmware, hence updated document title and *Introduction*, and added *Section 4: Application description - STM32L476RG-NUCLEO*, *Section 5: Firmware description - STM32L476RG-NUCLEO*, and their subsections. Added *Section 1: LC sensor metering principle* and *Section 6: Power consumption - STM32L476RG-NUCLEO*. Updated *Definitions*, *Section 2: Application description - STM32L073Z-EVAL*, *Section 3: Firmware description - STM32L073Z-EVAL* and *Section 10: Conclusion*.
10-Jun-2025	3	Updated document title and *Introduction*. Introduced STM32U031R8-NUCLEO board, hence added *Section 7: Application description - STM32U031R8-NUCLEO*, *Section 8: Firmware description - STM32U031R8-NUCLEO*, *Section 9: Power consumption - STM32U031R8-NUCLEO*, and their subsections. Updated *Figure 81: PeriodicCalibration () function*. Minor text edits across the whole document.