life.augmented

AN4865
Application note

Low-power timer (LPTIM) applicative use-cases on STM32 MCUs

Introduction

This application note describes the various modes and specific features of the low-power timer (LPTIM) embedded in the STM32 products. See *Table 1* for applicable products.

This application note also provides below three practical applicative use-cases of the LPTIM peripheral:

- Asynchronous pulse counter in STOP mode.
- PWM generator in STOP mode.
- Timeout wakeup mode.

These use-cases demonstrate the importance of the low-power timer and clarify the features that differ from the general-purpose timer.

This document and its associated firmware are available for download on STMicroelectronics website.

Table 1. Applicable products

Type	Product series and line
Microcontrollers	STM32L0 Series, STM32L4 Series, STM32F7 Series, STM32F410

Contents

List of tables

List of figures

1 Overview

Low-power techniques aim to reduce the MCU current consumption and extend the battery lifetime. This is done by optimizing the consumption during both run-time and idle-time. The low-power timer (LPTIM) specifically helps to reduce the power consumption while the system is in low-power mode.

The STM32 low-power timer allows the system to perform simple tasks while the power consumption is kept at an absolute minimum. It features a 16-bit auto-reload register to set the period and a 16-bit compare register to set the duty-cycle for a PWM waveform signal output on the timer.

The LPTIM can be used for timing and for output generation while the microcontroller is in low-power mode. It also features a wide diversity of clock sources allowing it to stay active in most power modes, except for the Standby and the Shutdown modes.

A flexible clock scheme allows the LPTIM to be used as a pulse counter by running with an external clock source. It also enables the LPTIM to wake up the system from low-power modes and to realize a “timeout function” with extremely low power consumption.

The low-power timer peripheral provides the basic functions of the STM32 general-purpose timers with the advantage of a very low power consumption. Additionally, when configured in asynchronous counting mode, the LPTIM keeps running even when no internal clock source is active.

The low-power timer peripheral remains active both in Sleep and in Stop mode, and it is able to wake up the microcontroller from these modes. Conversely, when in Standby or Shutdown mode, the timer is powered down and it must be completely reinitialized when the MCU exits any of these modes.

Table 2 summarizes the state of the LPTIM in different power modes.

Table 2. Effect of power modes on the LPTIM

Mode	Description
Run	Active.
Sleep	Active. Peripheral interrupts cause the device to exit Sleep mode.
Low-power run[(1)]	Active.
Low-power sleep[(1)]	Active. Peripheral interrupts cause the device to exit low-power sleep mode.
STOP1	Active. Peripheral registers content is kept. Peripheral may make the device exit STOP1 mode.
STOP2[(1)]	Active. Peripheral registers content is kept. Peripheral may make the device exit STOP2 mode.
Standby	Powered-down. The peripheral must be reinitialized after exiting Standby mode.
Shutdown[(1)]	Powered-down. The peripheral must be reinitialized after exiting Shutdown mode.

1. This mode is available only in the STM32L4 family.

Table 3 summarizes the availability of the LPTIM in different STM32 families.

Table 3. Simplified overview of LPTIM availability in STM32 products

-	STM32F410 Line	STM32F7 Series	STM32L0 Series	STM32L4 Series
Low-power timer	LPTIM1	LPTIM1	LPTIM1	LPTIM1
	-	-	-	LPTIM2

The low-power timer presents up to eight external trigger sources with a configurable polarity. The external trigger inputs embed digital filters to cancel-out possible faulty triggers in noisy operating environments. the LPTIM can be configured to run either in Continuous mode or in One-shot mode. The One-shot mode is used to generate pulse waveforms, while the Continuous mode is used to generate PWM waveforms.

The low-power timer features an Encoder mode. This function enables the LPTIM to interface with incremental quadrature encode sensors using the Input1 and Input2 of the peripheral. Both inputs feature a glitch-filtering circuitry.

The low-power timer can output different waveform types even when the microcontroller is in specific low-power modes whereas almost all internal clock sources are turned off. The LPTIMx_CMP and LPTIMx_ARR registers, in conjunction with the bit-fields ‘WAVE’ from the LPTIMx_CFGR register and the bit field ‘SNGSTRT’ from the LPTIMx_CR register, are used to control the output waveform.

The output waveform can be a typical PWM signal with its period and duty-cycle controlled by the LPTIMx_ARR and the LPTIMx_CMP registers respectively, or the output waveform can be a single pulse with the last output state defined by the configured waveform.

If the output waveforms are not equal, the SetOnce mode is configured. The polarity of the low-power timer output is controlled through the ‘WAVPOL’ bit-field in the LPTIMx_CFGR register.

2 Low-power timer clock sources

The LPTIM is a peripheral with two clock domains: the APB (advanced peripheral bus) clock domain and the kernel clock domain. The APB clock domain contains the APB interface and the core functions of the peripheral such as the registers and signals connected to the CPU (typically the interrupt request).

The kernel clock domain can be clocked by the APB clock source or by other internal clock sources including the LSE, LSI and HSI sources. It can also be clocked from an external clock source through the input (Input1) of the timer.

The LPTIM main feature is its ability to keep running in low-power modes when almost all clock sources are turned off. It also features a very flexible clocking scheme enabling the features below:

- It can be clocked from on-chip clock sources such as LSE, LSI, HSI, or APB clocks (refer to the reference manual of the product for more details).
- It can be clocked from an external clock source through the input “Input1”.

The flexible clocking scheme is used for building “Pulse Counter” applications and it is a key function for metering applications such as gas-meters. *Table 4* summarizes the LPTIM clock sources in different power modes.

Table 4. LPTIM clock source on different power mode

-	HSI	LSI	LSE	PCLK1	External clock
Run	x	x	x	x	x
Sleep	x	x	x	-	x
Low-power run(1)	x	x	x	x	x
Low-power sleep(1)	x	x	x	-	x
STOP1	-	x	x	-	x
STOP2(1)	-	x	x	-	x

1. This mode is available only in the STM32L4 family.

3 Low-power timer peripheral versus general-purpose timer peripheral

The main advantages of the LPTIM compared to any other timer peripheral available in the STM32 MCUs are:

- its ability to continue working even in the Stop mode
- its ability to generate events waking up the MCU from the Stop mode.

The general-purpose timer is active in Run and Sleep mode, while it is frozen in Stop mode. In Stop mode, both the general-purpose timer state and the register contents are preserved; the timer directly resumes operations when the MCU is woken up.

Depending on the selected clock source, the power consumption can be substantially lowered with LPTIM compared to a general-purpose timer.

Table 5 summarizes the difference between the timer TIM peripheral and the low-power timer LPTIM peripheral during different power modes.

Table 5. TIM versus LPTIM on the working mode

Peripheral	Run	Sleep	Low-power run(1)	Low-power sleep(1)	STOP1	STOP2
TIMx	x	x	x	-	-	-
LPTIMx	x	x	x	x	x	x

1. This mode is available only in the STM32L4 family.

4 Synchronization block

The low-power timer in the STM32 microcontrollers can be configured in two modes:

- Asynchronous mode: the LPTIM is not clocked by an internal clock signal.
- Synchronous mode: the LPTIM is clocked by an internal clock signal.

In Synchronous mode, the external signal is synchronized with a clock signal different from the one used by the LPTIM peripheral.

The synchronization circuit used to synchronize the external signals is connected to the LPTIM inputs. This circuit is mainly composed of cascaded D flip-flops that are clocked by the LPTIM core clock signal. This synchronization circuit introduces a delay of at least two LPTIM core clock cycles; for this reason, the internal clock signal frequency should be at least two times higher than the external clock signal frequency.

In the case that both edges are configured to be the active ones, the internal clock signal frequency should be at least four times higher than the external clock signal frequency. This delay of two counter clocks is also needed after enabling the LPTIM and before the LPTIM starts counting.

In Asynchronous mode, the timer can operate with any external clock (below 15 MHz), while the MCU system clock is stopped.

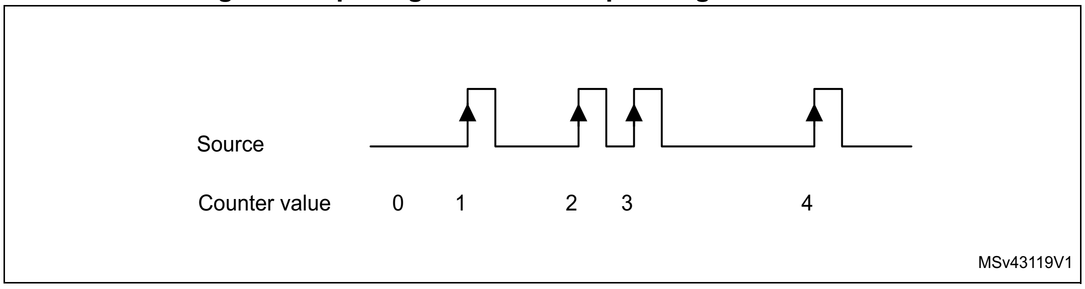
5 Use-cases

5.1 Asynchronous pulse counter in STOP2 mode

The pulse counting method for flow measurement consists of converting the kinetic energy from rotation into electrical digital signals in the form of pulses. The pulses density vary in accordance to the measured flow: the faster the rotation, the higher the number of pulses, varying in accordance with the measured flow.

In typical applications the microcontroller counts the number of pulses, but it is not wanted to wake it up from Stop mode at every pulse. In this kind of cases, the LPTIM configured as pulse counter is very useful: it uses the Asynchronous mode and it is clocked by the pulses, which do not have a regular pace. *Figure 1* shows the input signal and the corresponding counter value.

Figure 1. Input signal and corresponding counter value



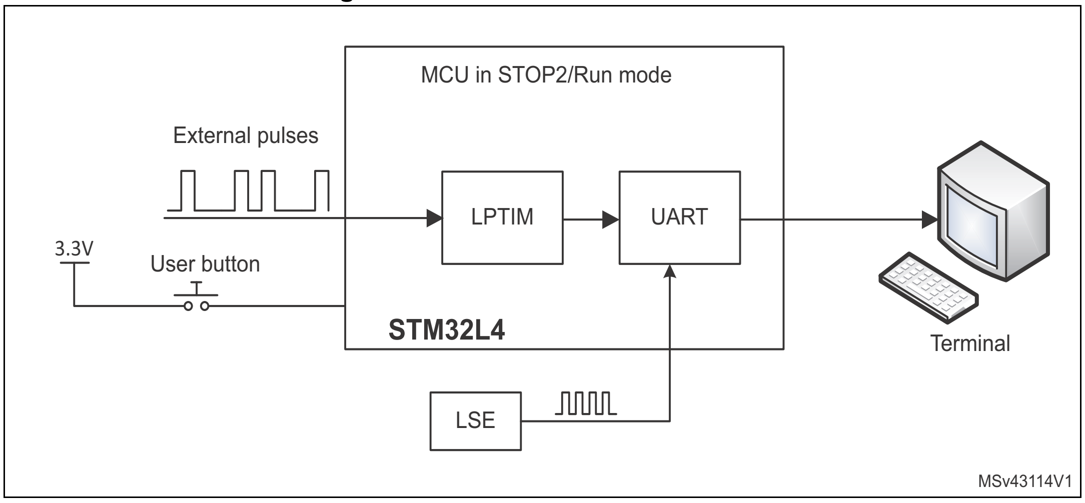
A flow meter can be divided in three parts:

- A transducer that convert the physical quantity to measure into electrical signal.
- A data treatment logic that count and stores.
- A display to show the accumulated count.

This application aims to count the pulses coming from a pulse source and to display them on a terminal by using a serial communication UART. The used parameters are: the pulse counter is based on the LPTIM1 in Asynchronous mode and the MCU is in STOP2 power mode. Those parameters have been chosen to show the LPTIM features in low-power mode and the UART is clocked by the LSE to guarantee a low power consumption.

The transmission of the pulse count by the UART is performed in the interrupt service routine (ISR) of the push-button interrupt. *Figure 2* describes the synoptic of the pulse counter.

Figure 2. Pulse counter architecture



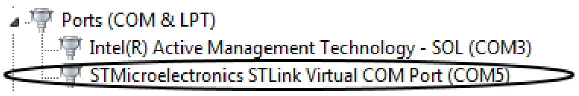
The hardware environment used for this example development is the NUCLEO-L476RG.

To establish the serial communication, the STM32 NUCLEO board already embeds the ST-LINK debug and programming probe which features an auxiliary UART interface exploited by this example to interface with the PC and to offer a virtual COM port interface. This application does not require any additional hardware and only requires a mini-USB cable for data transmission.

A COM port monitor software application on the PC can be used to:

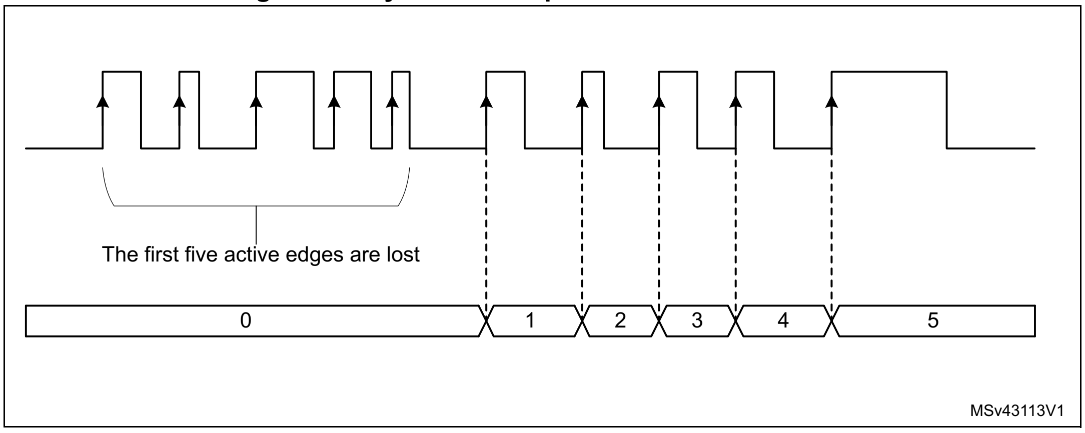
- display the messages coming from the microcontroller via the serial link
- allow sending data via the serial link to the microcontroller.

Figure 3. STMicroelectronics virtual COM port device manager


Ports (COM & LPT)
Intel(R) Active Management Technology - SOL (COM3)
STMicroelectronics STLink Virtual COM Port (COM5)

In Asynchronous mode, the external pulses injected on the LPTIM external Input1 are also used to clock the LPTIM that looses the first five active edges. This effect is illustrated in *Figure 4*. The LPTIM counter can be updated by either following a rising edge or by following a falling edge, and not both at the same time because this mode is not supported in Asynchronous mode.

Figure 4. Asynchronous pulse counter increment

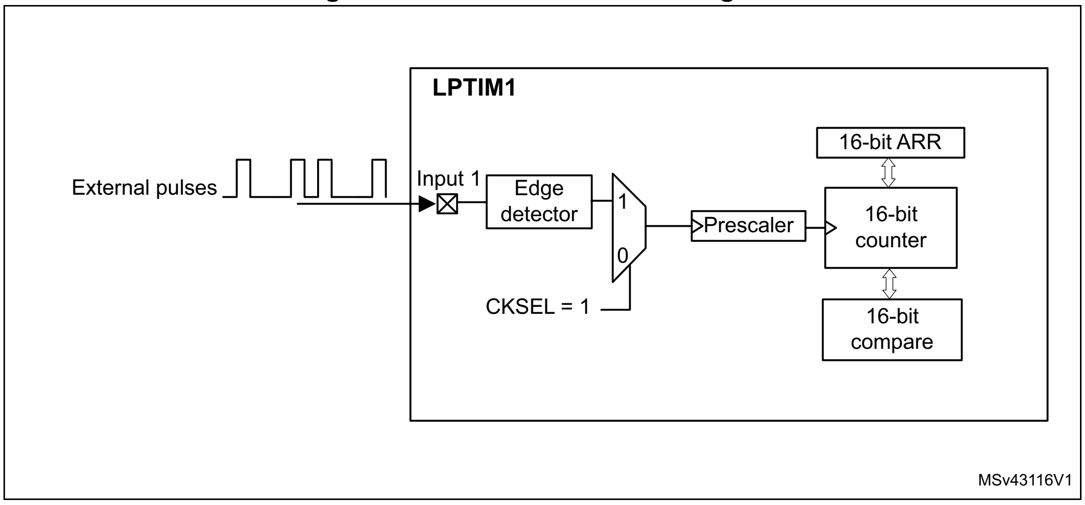


To apply this configuration:

- set the Control bit CKSEL in the LPTIMX_CFGR register.
- use CKPOL bits to configure the active edges.

Figure 5 shows the block diagram for the asynchronous pulse counter.

Figure 5. Pulse counter block diagram



Firmware description

- System clock:
  - System clock: HSI.
  - LPTIM1 source clock: external signal.
  - UART2 source clock: LSE.
- LPTIM1:
  - Source clock: external source (LPTIM1_input1).
  - Prescaler = 1.
  - Clock polarity: rising edge.
  - Trigger source: software.
- UART2:
  - Source clock: LSE.
  - Baudrate = 1200.
  - STOP bits = 1.
  - Parity: none.
  - Mode: Tx.
  - Hardware flow control: none.

5.2 PWM generator in STOP2 mode

One of the main advantages of the low-power timer of the STM32 microcontrollers is its ability to work in Stop mode with the possibility to generate several different waveforms.

The PWM in STOP2 mode allows a signal to be generated:

- With a frequency determined by the value of the LPTIMx_ARR register.
- With a duty cycle determined by the value of the LPTIMx_CMP register.

There are two ways to update the values of the LPTIMx_ARR and the LPTIMx_CMP registers, and they are controlled by the PRELOAD bit included inside the LPTIM_CFGR register:

- PRELOAD = ‘0’: the LPTIMx_ARR and the LPTIMx_CMP registers are updated immediately after the write access.
- PRELOAD = ‘1’: the LPTIMx_ARR and the LPTIMx_CMP registers are updated at the end of the running period of the timer.

If PRELOAD bit is set to ‘1’, the update of the register is done at the end of the running period of the timer. Note the write access to the concerned register must be done before the last clock cycle of the period, otherwise the update will be postponed to the next period.

If PRELOAD bit is set to ‘0’, there is a short delay between the write access to the concerned register and the actual update of its value, this means that the waveform at the LPTIM output is directly impacted by the selected PRELOAD mode.

The LPTIM output depends on the continuous comparison between the LPTIM counter value and the LPTIM register LPTIMx_CMP value. Writing directly to the channel register in the middle of a PWM period may generate spurious waveforms. The synchronization of the clock source of the LPTIM with the APB clock domain creates some latency after the write access in the compare register LPTIM_CMP.

Any write access during this latency period must be avoided because it leads to unpredictable results. This latency is the delay between a *write* in one of the register (LPTIMx_ARR or LPTIMx_CMP) and the setting of the corresponding flag (ARROK or CMPOK).

The resynchronization time of the LPTIM core clock with the ABP clock is part of the delay. The delay includes 2 x APB_CLK + 3 x LPTIM_CLK, with LPTIM_CLK being the core clock of the LPTIM; then the user must add 2 x APB_CLK for ARROK flag to become ‘1’, knowing that this write access must be made after the LPTIM enabling.

The change in the registers LPTIMx_ARR or LPTIM_CMP is applied only when their flag is set to inform the application that the APB bus operation has been successfully completed. When the ARROK or the CMPOK flag is set, an interrupt is generated to the MCU; thanks to this, the MCU is not polling to check the completion of the write operation.

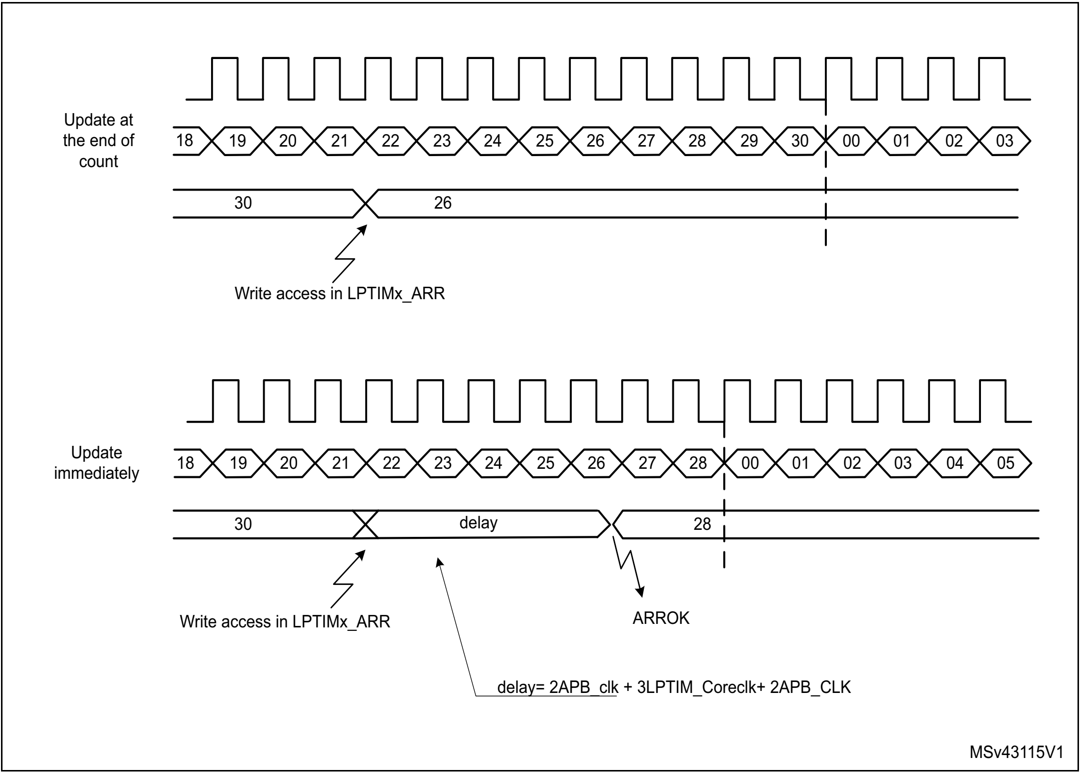
Example:

- the System clock = 180 MHz and the LPTIM_CLK = LSI = 32 KHz
- the MCU cycle = 5.55 ns and the LPTIM_CLK cycle = 31.25 µs
- The delay = 2 x APB_CLK + 3 x LPTIM_CLK + 2 x APB_CLK =
  2 x 5.55 ns + 3 x 31.25 µs + 2 x 5.55 ns = 93.77µs = 16 896 MCU cycles

If these two flags do not exist, the MCU wastes 16 896 cycles to test the update of the register status.

Figure 6 summarizes the preload mechanism for the LPTIM_ARR register.

Figure 6. Counter timing diagram with and without update immediately


Update at the end of count
18
19
20
21
22
23
24
25
26
27
28
29
30
00
01
02
03
30
26
Write access in LPTIMx_ARR
Update immediately
18
19
20
21
22
23
24
25
26
27
28
00
01
02
03
04
05
30
delay
28
Write access in LPTIMx_ARR
ARROK
delay= 2APB_clk + 3LPTIM_Coreclk+ 2APB_CLK
MSv43115V1

In this example, the LPTIM1 is selected because it is available in STOP2 mode.

To run the LPTIM, the chosen source clock is the LSI. The PWM output waveform is configured through:

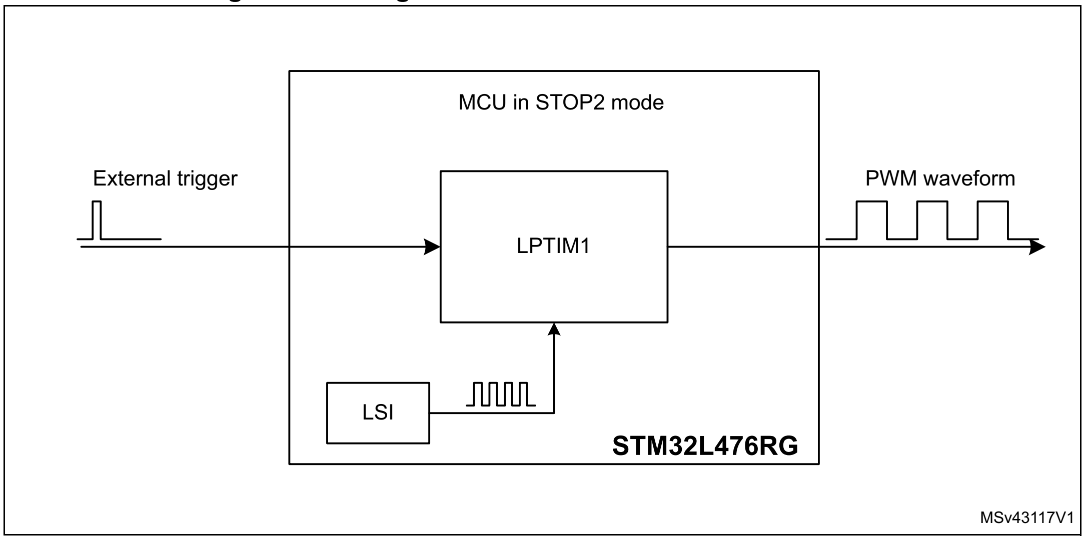
- the WAVE bit: reset to '0'
- the CNTSTRT bit: set
- the WAVEPOL bit: controls the output polarity.

The LPTIM1 is started after the detection of an active edge on the LPTIM1_ETR pin.

Before the MCU enters the low-power mode, the LPTIM1 must be configured by loading the period and pulse values to the LPTIM1_ARR and LPTIM1_CMP registers, and then by enabling LPTIM. To start the LPTIM1, a pulse must be applied on the LPTIM1_ETR even if the MCU is in Stop mode.

The architecture of this example is described in *Figure 7*.

Figure 7. PWM generator in STOP2 mode architecture



Firmware description

- System clock:
  - System clock: HSI.
  - LPTIM1 source clock: LSI.
- LPTIM1:
  - Source clock: internal source (LSI).
  - Prescaler = 1.
  - Trigger source: LPTIM1_ETR pin.
  - Trigger polarity: rising edge.
  - Polarity output: high.

5.3 Timeout wakeup mode

The LPTIM can be configured to periodically wake up the MCU from the Sleep or the Stop mode, for example to refresh a display or to read a sensor. The main purpose of this mode is to wake up the MCU only in case of abnormal operation or malfunction (like a watchdog).

As long as the periodic trigger keeps coming, the MCU stays in ultra-low power mode; if no pulse is generated when it is expected, the MCU wakes up to identify the reason.

The clock source can be either the LSE, the LSI oscillator or an external clock source. An external clock source can be generated externally and injected to the LPTIM input1 whenever the LPTIM is already configured to use it (CKSEL is appropriately configured).

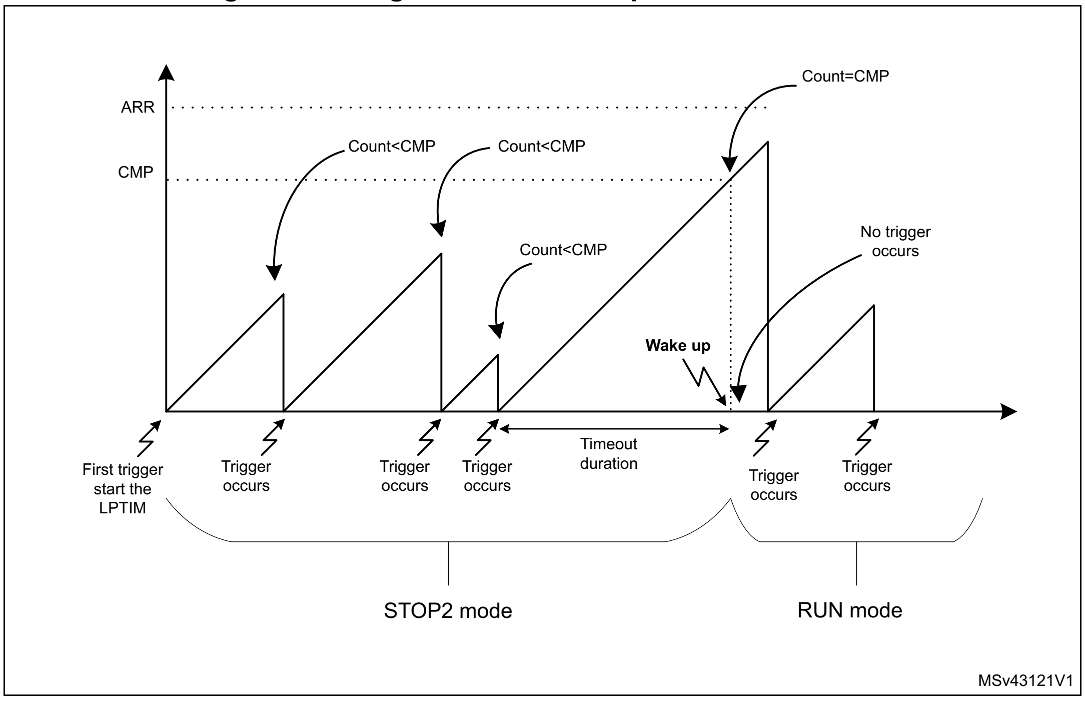
This LPTIM feature is controlled through the TIMOUT bit in the LPTIMx_CFGR register which allows the trigger signal to reset the LPTIM counter by detection of an active edge when the timer is already started.

The duration of the timeout is a function of the LPTIM frequency count and the LPTIMx_CMP value.

$$\text{Timeout} = \frac{\text{CMP} + 1}{\text{LPTIM clock}}$$

If the duration between two active edges triggers is smaller than the timeout, then the LPTIM resets and restarts the counter with the MCU still in Sleep or Stop mode, else the MCU is awoken by the compare match event. The timeout timing is presented in *Figure 8*.

Figure 8. Timing of timeout wakeup from STOP2 mode



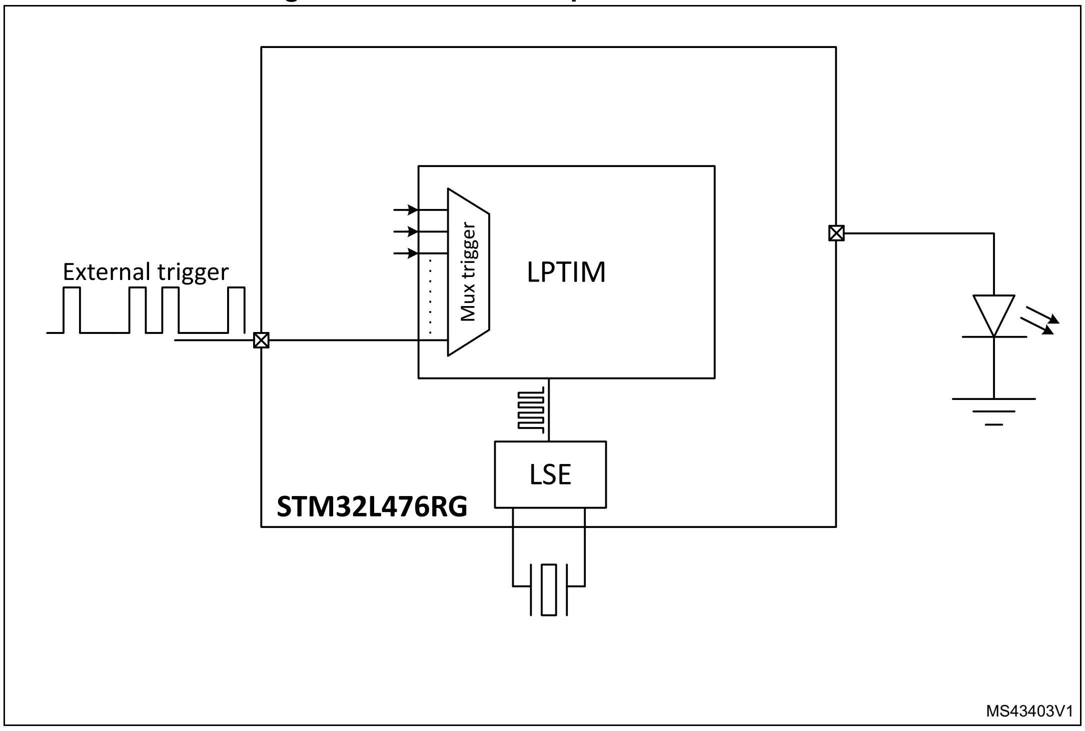
The trigger signal can be selected from several sources by the TRIGSEL bits, examples of available trigger sources are: GPIO, RTC alarm, RTC_TAMP and COMP_OUT. The TRIGSEL bits are used only when TRIGEN[1:0] is different from '00'.

In this example, the LPTIM1 is selected because it is available in the STOP2 mode. The LSE has been chosen as source clock to run this LPTIM. The Timeout mode is configured through the TIMOUT bit in the LPTIM1_CFGR register. The LPTIM1 is started after detection of the first active edge on the LPTIM1_ETR.

Before entering the low-power mode, the period and the pulse must be loaded to the LPTIM1_ARR and LPTIM1_CMP registers to set the timeout duration, then the LPTIM1 must be enabled.

The architecture of this example is described in *Figure 9*.

Figure 9. Timeout wakeup mode architecture



Firmware description

- System clock:
  - System clock: HSI.
  - LPTIM1 source clock: LSE.
- LPTIM1:
  - Source clock: internal source (LSE).
  - Prescaler = 1.
  - Trigger source: LPTIM1_ETR pin.
  - Trigger polarity: rising edge.
  - Counter source: internal source (LSE).

6 Revision history

Table 6. Document revision history

Date	Revision	Changes
18-Aug-2016	1	Initial release.