AN4013
Application note

STM32 cross-series timer overview

Introduction

The purpose of this document is to:

- Present an overview of the timer peripherals for the STM32 product series itemized in *Table 1*.
- Describe the various modes and specific timer features, such as clock sources.
- Explain how to use the available modes and features.
- Explain how to compute the time base in each configuration.
- Describe the timer synchronization sequences and the advanced features for motor control applications, in addition to the general-purpose timer modes.

For each mode, the document provides typical configurations and implementation examples.

In the rest of this document (unless otherwise specified), the term STM32xx Series is used to refer to the product series listed in *Table 1*.

Table 1. Applicable products

Type	Product series
Microcontrollers	STM32F0 Series, STM32F1 Series, STM32F2 Series, STM32F3 Series, STM32F4 Series, STM32F7 Series, STM32G0 Series, STM32G4 Series, STM32H7 Series, STM32L0 Series, STM32L1 Series, STM32L4 Series, STM32WB Series

Contents

1 Overview 6

2 General-purpose timer modes 10

2.1 Clock input sources 10

2.1.1 Internal clock 10

2.1.2 External clock 10

2.2 Time base generator 11

2.3 Timer input capture mode 13

2.4 Timer in output compare mode 14

2.5 Timer in PWM mode 15

2.6 Timer in one pulse mode 16

2.7 Timer in asymmetric PWM mode 17

2.8 Timer in combined PWM mode 18

2.9 Retriggerable one pulse mode 20

3 Timer synchronization 22

3.1 Timer system link 22

3.2 Master configuration 22

3.3 Slave configuration 24

4 Advanced features for motor control 25

4.1 Signal generation 25

4.2 Combined three-phase PWM mode 27

4.3 Specific features for motor control applications 28

4.3.1 Complementary signal and deadtime feature 28

4.3.2 Break input 29

4.3.3 Locking mechanism 31

4.3.4 Specific features for feedback measurement 32

5 High-resolution timer applications 36

6 Low-power timer 37

6.1 Wakeup timer implementation 37

6.2 Pulse counter 38

7 Specific applications 39

7.1 Infrared application 39

7.2 3-phase AC and PMSM control motor 39

7.3 Six-step mode 39

8 Revision history 41

List of tables

Table 1. Applicable products . . . 1
Table 2. Simplified overview of timer availability in STM32Fx products . . . 7
Table 3. Simplified overview of timer availability in STM32Gx/Lx/WB products . . . 8
Table 4. Timer features overview . . . 9
Table 5. Advanced timer configurations . . . 26
Table 6. Behavior of timer outputs versus Break1 and Break2 inputs . . . 30
Table 7. Locking levels . . . 31
Table 8. Document revision history . . . 41

Figure 1. Asymmetric PWM mode versus center Aligned PWM mode 17
Figure 2. Combined PWM mode 19
Figure 3. Retriggerable OPM mode 21
Figure 4. Timer system link 22
Figure 5. Combined three-phase PWM 27
Figure 6. Two signals are generated with insertion of a deadtime 28
Figure 7. Position at X4 resolution 32
Figure 8. Position at X2 resolution 33
Figure 9. Output waveform of a typical Hall sensor 34
Figure 10. Commutation sequence 35

1 Overview

The STM32xx Series devices, based on the Arm® cores[a], have various built-in timers outlined as follows;

- **General-purpose timers** can be used by any application for: output comparison (timing and delay generation), one-pulse mode, input capture (for external signal frequency measurement), sensor interface (encoder, hall sensor);
- **Advanced timers**: these timers have the most features. In addition to general purpose functions, they include several features related to motor control and digital power conversion applications: three complementary signals with deadtime insertion and emergency shut-down input;
- One or two **channel timers**: used as general-purpose timers with a limited number of channels.
- One or two channel timers with **complementary output**: same as the previous timer type with an additional deadtime generator on one channel. In some situations, this feature allows a general purpose timer to be used where an additional advanced timer would be necessary.
- **Basic timers** are used either as timebase timers or for triggering the DAC peripheral These timers don't have any input/output capabilities;
- **Low-power timers** are simple general purpose timers and are able to operate in low-power modes.They are used to generate a wake-up event for example;
- **High-resolution timers** are specialized timer peripherals designed to drive power conversion in lighting and power source applications. They can also be used in other fields that require very fine timing resolution. AN4885 and AN4449 are practical examples of high-resolution timer use.

Table 2 and *Table 3* summarize the STM32 family timers.

Table 4 presents a general overview of timer features.

Timers are enhanced with more advanced features in newer devices. Besides minor changes not in scope of this overview, a significant update divides the STM32 family advanced motor control and general purpose timers. In this document STM32F0/F1/F2/F4 Series and STM32F37x devices are referred to as the "original series". Some of the features are not available for them and are identified as such.

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

Table 2. Simplified overview of timer availability in STM32Fx products

Timer type		STM32 F04x /F070x6 /F03x (excluding /F030x8 and /F030x)	STM32 F030xB /F030x8 /F05x /F09x /F07x (excluding F070x6)	STM32 F101 /F102 /F103 lines XL density (xF, xG)	STM32 F101 /F102 /F103 /F105 /F107 lines up to high-density (x4-xE)	STM32 F100 value line	STM32 F2 /F4 (excluding /F401, /F411, /F410)	STM32 F401 /F411 /F410	STM32 F30X /F3x8 (excluding /F378)	STM32 F37x	STM32 F334	STM32 F31x	STM32 F7 Series
Advanced		TIM1	TIM1	TIM1(1) TIM8(1)	TIM1(1) TIM8(1)	TIM1	TIM1 TIM8	TIM1	TIM1 TIM8(1) TIM20(1)	-	TIM1	TIM1 TIM8(1)	TIM1 TIM8
General purpose	32-bit	TIM2	TIM2	-	-	-	TIM2 TIM5	TIM2(1) TIM5	TIM2	TIM2 TIM5	TIM2	TIM2	TIM2 TIM5
	16-bit	TIM3	TIM3	TIM2 TIM3 TIM4 TIM5	TIM2 TIM3 TIM4(1) TIM5(1)	TIM2 TIM3 TIM4 TIM5(1)	TIM3 TIM4	TIM3(1) TIM4(1)	TIM3(1) TIM4(1) TIM19(1)	TIM3 TIM4 TIM19	TIM3	TIM3 TIM4	TIM3 TIM4
Basic		-	TIM6 TIM7(1)	TIM6 TIM7	TIM6(1) TIM7(1)	TIM6 TIM7	TIM6 TIM7	TIM6(1)	TIM6 TIM7(1)	TIM6 TIM7 TIM18	TIM6 TIM7	TIM6 TIM7(1)	TIM6 TIM7
1 channel		TIM14	TIM14	TIM10 TIM11 TIM13 TIM14	-	TIM13(1) TIM14(1)	TIM10 TIM11 TIM13 TIM14	TIM10(1) TIM11	-	TIM13 TIM14	-	-	TIM10 TIM11 TIM13 TIM14
2-channel		-	-	TIM9 TIM12	-	TIM12(1)	TIM9 TIM12	TIM9	-	TIM12	-	-	TIM9 TIM12
2-channel with complementary output		-	TIM15	-	-	TIM15	-	-	TIM15	TIM15	TIM15	TIM15	-
1-channel with complementary output		TIM16 TIM17	TIM16 TIM17	-	-	TIM16 TIM17	-	-	TIM16 TIM17	TIM16 TIM17	TIM16 TIM17	TIM16 TIM17	-
Low-power timer		-	-	-	-	-	-	LPTIM1(1)	-	-	-	-	LPTIM1
High-resolution timer		-	-	-	-	-	-	-	-	-	HRTIM	-	-

1. Not available on all products in the line. Check the datasheet for details.

Note: More recent versions of advanced timers present several new modes: asymmetric mode, combined mode, one retriggerable mode, combined 3 PWM mode and a second break input. These modes are not available in the STM32F0/F1/F2/F4 Series and STM32F37x device advanced control timers also known as the original series.

Table 3.Simplified overview of timer availability in STM32Gx/Lx/WB products

Timer type		STM32 G0 Series	STM32 G4 Series	STM32 H7 Series	STM32L0 5X /L06x /L07x /L08x lines	STM32 L0x0 Value Line	STM32 L0x3 /L0x2 /L0x1 lines	STM32 L1 Series	STM32 L4 Series	STM32 WB Series
Advanced		TIM1	TIM1 TIM8 TIM20(1)	TIM1 TIM8	-	-	-	-	TIM1 TIM8(1)	TIM1
General purpose	32-bit	TIM2	TIM2 TIM5(1)	TIM2 TIM5	-	-	-	TIM5(1)	TIM2 TIM5(1)	TIM2
	16-bit	TIM3	TIM3 TIM4	TIM3 TIM4	TIM2 TIM3(1)	TIM2	TIM2	TIM2 TIM3 TIM4	TIM3(1) TIM4(1)	-
Basic		TIM6 TIM7	TIM6 TIM7	TIM6 TIM7	TIM6 TIM7(1)	-	-	TIM6 TIM7	TIM6 TIM7	-
1 channel		TIM14	-	TIM13 TIM14	-	-	-	TIM10 TIM11	-	-
2-channel		-	-	TIM12	TIM21 TIM22	TIM21	TIM21 TIM22(1)	TIM9	-	-
2-channel with complementary output		TIM15	TIM15	TIM15	-	-	-	-	TIM15	-
1-channel with complementary output		TIM16 TIM17	TIM16 TIM17	TIM16 TIM17	-	-	-	-	TIM16 TIM17(1)	TIM16 TIM17
Low-power timer		LPTIM1 LPTIM2	LPTIM1 LPTIM2	LPTIM1 LPTIM2 LPTIM3 LPTIM4 LPTIM5	LPTIM1	-	LPTIM1	-	LPTIM1 LPTIM2	LPTIM1 LPTIM2
High-resolution timer		-	HRTIM1(1)	HRTIM1	-	-	-	-	-	-

1. Not available on all products in the line. Check the datasheet for details.

Note: More recent versions of advanced timers present several new modes: asymmetric mode, combined mode, one retriggerable mode, combined 3 PWM mode and a second break input. These modes are not available in the STM32F0/F1/F2/F4 Series and STM32F37x device advanced control timers also known as the original series.

Table 4. Timer features overview

Timer type	Counter resolution	Counter type	DMA	Channels	Output. channels	Synchronization	
						Master configuration	Slave configuration
Advanced Control	16 bit	Up, down and center aligned	Yes	4, 6(1)	3	Yes	Yes
General-purpose	16 bit 32 bit(2)	Up, down and center aligned	Yes	Up to 4	0	Yes	Yes
Basic	16 bit	Up	Yes	0	0	Yes	No
1-channel	16 bit	Up	No	1	0	Yes (OC signal)	No
2-channel	16 bit	Up(3)	No	2	0	Yes	Yes
1-channel with one complementary output	16 bit	Up	Yes	1	1	Yes (OC signal)	No
2-channel with one complementary output	16 bit	Up(4)	Yes	2	1	Yes	Yes
Low-power timer	16 bit	Up	No	1(5)	0	Yes (OC signal)	No
High-resolution timer	16 bit	Up	Yes	Up to 12(5)	Up to 6	Yes	Yes

1. With STM32L4/G4/F7 Series and STM32F30x/F3x8 lines, the advanced timers have 6 channels. The two extra channels are however not connected to GPIO (not available as output).
2. TIM2 and TIM5 are 32-bit counter resolution on some products and 16-bit on others. See *Table 2* and *Table 3* or product datasheet as reference.
3. On some devices, the counter type also supports Up, Down, Up/Down as exception (STM32L0 Series).
4. On some devices the counter type also supports Up, Down, Up/Down as exception (STM32WB Series).
5. Low-power timer and high-resolution timer do not have channels directly comparable with channels on regular timer peripherals. Indicated number is a “channel equivalent”.

2 General-purpose timer modes

General-purpose timers can be programmed to work in various different configurations. The following chapter is an introduction to the timer usage.

2.1 Clock input sources

The timer always needs a clock source. It can also be synchronized by several clocks simultaneously:

- Internal clock.
- External clock.
  - External mode1 (TI1 or TI2 pins)
  - External clock mode2 (ETR pin)
  - Internal trigger clock (ITRx).

2.1.1 Internal clock

By default, the timer is clocked by the internal clock provided by the RCC. To select this clock source, the TIMx_SMCR->SMS (if present) bits should be reset.

The RCC register then defines the internal clock source for the timer.

2.1.2 External clock

The external clock timer is divided in two categories:

- External clock connected to TI1 or TI2 pins
- External clock connected to ETR pin.

In these cases, the clock is provided by an external signal connected to TIx pins or ETR pin. The maximum external clock frequency should be verified.

In addition to all these clock sources, the timer should be clocked with the APBx clock.

The external clocks are not directly feeding the prescaler, but they are first synchronized with the APBx clock through dedicated logical blocks.

External clock mode1 (TI1 or TI2 pins)

In this mode, the external clock are be connected to the timer input TI1 pin or TI2 pin. To do this:

1. Configure the timers to use the TIx pin as input:
   a) Select the pin to be used by writing CCxS bits in the TIMx_CCMR1 register
   b) Select the polarity of the input:
   For the STM32F100/101/102/103/105/107 lines: write CCxP in the TIMx_CCER register to select the rising or the falling edge;
   For the other series and lines: write CCxP and CCxNP in the TIMx_CCER register to select the rising/falling edge, or both edges[(a)].
   c) Enable corresponding channel by setting the CCEx bit in the TIMx_CCER register.
2. Select the timer TIx as the trigger input source by writing TS bits in the TIMx_SMCR register
3. Select the external clock mode1 by writing SMS=111 in the TIMx_SMCR register.

External clock mode2 (ETR pin)

The external clock mode2 uses the ETR pin as timer input clock. To use this feature:

1. Select the external clock mode2 by writing ECE = 1 in the TIMx_SMCR register
2. Configure, if needed, the prescaler, the filter and the polarity by writing ETPS [1:0], ETF [3:0] and ETP in the TIMx_SMCR register.

Internal trigger clock (ITRx)

This is a particular timer synchronization mode. When using one timer as a prescaler for another timer, the first timer update event or output compare signal is used as a clock for the second one.

2.2 Time base generator

The timer can be used as a time base generator. Depending on the clock, prescaler and auto reload, repetition counter (if present) parameters, the 16-bit timer can generate an update event from a nanosecond to a few minutes. The 32-bit timers provide a wider range.

Example update event period

The update event period is calculated as follows:

Update_event = TIM_CLK/((PSC + 1)*(ARR + 1)*(RCR + 1))

Where: TIM_CLK = timer clock input

PSC = 16-bit prescaler register

ARR = 16/32-bit Autoreload register

RCR = 16-bit repetition counter

a. For the STM32F100/101/102/103/105/107 lines, polarity selection for both edges can be achieved by using TI1F_ED, but only for TI1 input.

TIM_CLK = 72 MHz

Prescaler = 1

Auto reload = 65535

No repetition counter RCR = 0

Update_event = 72*(10^6)/((1 + 1)*(65535 + 1)*(1))

Update_event = 549.3 Hz

Example external clock mode2

In this mode, the update event period is calculated as follows:

Update_event = ETR_CLK/((ETR_PSC)*(PSC + 1)*(ARR + 1)*(RCR + 1))

Where ETR_CLK = the external clock frequency connected to ETR pin.

ETR_CLK = 100 kHz

Prescaler = 1

ETPS - external trigger prescaler= 2

Autoreload = 255

Repetition counter = 2

Update_event= 100*(10^3)/((2)* (1+ 1)*((255 + 1)*(2 + 1))

Update_event = 21.7 Hz

Example external clock mode1

In this mode, the update event period is calculated as follows:

Update_event = TIx_CLK/((PSC + 1)*(ARR + 1)*(RCR +1))

Where TIx_CLK = the external clock frequency connected to TI1 pin or TI2 pin.

TIx_CLK = 50 kHz

Prescaler = 1

Auto reload = 255

Repetition counter = 2

Update_event = 50 000/((1+ 1)*((255 + 1)*(2 + 1))

Update_event = 32.55 Hz

Example Internal trigger clock (ITRx) mode1

In this mode, the update event period is calculated as follows:

Update_event = ITRx_CLK/((PSC + 1)*(ARR + 1)*(RCR + 1))

Where ITRx_CLK = the internal trigger frequency mapped to timer trigger input (TRGI)

ITRx_CLK = 8 kHz

Prescaler = 1

Auto reload = 255

Repetition counter = 1

Update_event = 8000/((1+ 1)*((255 + 1)*(1 + 1))

Update_event = 7.8 Hz

Depending on the counter mode, the update event is generated each:

- Overflow, if up counting mode is used: the DIR bit is reset in TIMx_CR1 register
- Underflow, if down counting mode is used: the DIR bit is set in TIMx_CR1 register
- Overflow and underflow, if center aligned mode is used: the CMS bits are different from zero.

The update event is generated also by:

- Software, if the UG (update generation) bit is set in TIM_EGR register
- Update generation through the slave mode controller.

As the buffered registers (ARR, PSC, CCRx) need an update event to be loaded with their preload values, set the URS (Update Request Source) to 1 to avoid the update flag each time these values are loaded. In this case, the update event is only generated if the counter overflow/underflow occurs.

The update event can also be disabled by setting the bit UDIS (update disable) in the CR1 register. In this case, the update event is not generated, and shadow registers (ARR, PSC, CCRx) keep their values. The counter and the prescaler are reinitialized if the UG bit is set, or if a hardware reset is received from the slave mode controller.

An interrupt or/and a DMA request can be generated when the UIE bit or/and UDE bit are set in the DIER register.

Most STM32Cube firmware packages include examples in Examples\TIM\TIM_TimeBase sub folders.

2.3 Timer input capture mode

The timer can be used in input capture mode to measure an external signal. Depending on timer clock, prescaler and timer resolution, the maximum measured period is deduced.

To use the timer in this mode:

1. Select the active input by setting the CCxS bits in CCMRx register. These bits should be different from zero, otherwise the CCRx register are be read only
2. Program the filter by writing the IC1F[3:0] bits in the CCMRx register, and the prescaler by writing the IC1PSC[1:0] if needed
3. Program the polarity by writing the CCxNP/CCxP bits to select between rising, falling or both edges.

The input capture module is used to capture the value of the counter after a transition is detected by the corresponding input channel. To get the external signal period, two consecutive captures are needed. The period is calculated by subtracting these two values:

Period = Capture(1) /(TIMx_CLK *(PSC+1)*(ICxPSC)*polarity_index(2))

The capture difference between two consecutive captures CCRx_tn and CCRx_tn+1:

- If CCRx_tn < CCRx_tn+1: capture = CCRx_tn+1 - CCRx_tn
- If CCRx_tn > CCRx_tn+1: capture = (ARR_max - CCRx_tn) + CCRx_tn+1.

The polarity index is 1 if the rising or falling edge is used, and 2 if both edges are used.

Particular case

To facilitate the input capture measurement, the timer counter is reset after each rising edge detected on the timer input channel by:

- Selecting TIxFPx as the input trigger by setting the TS bits in the SMCR register
- Selecting the reset mode as the slave mode by configuring the SMS bits in the SMCR register.

Using this configuration, when an edge is detected, the counter is reset and the period of the external signal is automatically given by the value on the CCRx register. This method is used only with channel 1 or channel 2.

In this case, the input capture prescaler (ICPSC) is not considered in the period computation.

The period is computed as follows:

Period = CCRx /(TIMx_CLK *(PSC+1)* polarity_index(1))

The polarity index is 1 if rising or falling edge is used, and 2 if both edges are used.

Many STM32Cube firmware packages include examples in Examples\TIM\TIM_InputCapture sub folder.

2.4 Timer in output compare mode

To control an output waveform, or to indicate when a period of time has elapsed, the timer is used in one of the following output compare modes. The main difference between these modes is the output signal waveform.

- **Output compare timing**: The comparison between the output compare register CCRx and the counter CNT has no effect on the outputs. This mode is used to generate a timing base
- **Output compare active**: Set the channel output to active level on match. The OCxRef signal is forced high when the counter (CNT) matches the capture/compare register (CCRx)
- **Output compare inactive**: Set channel to inactive level on match. The OCxRef signal is forced low when the counter (CNT) matches the capture/compare register (CCRx);
- **Output compare toggle**: OCxRef toggles when the counter (CNT) matches the capture/compare register (CCRx)
- **Output compare forced active/inactive**: OCREF is forced high (active mode) or low (inactive mode) independently from counter value.

To configure the timer in one of these modes:

1. Select the clock source
2. Write the desired data in the ARR and CCRx registers
3. Configure the output mode:
   a) Select the output compare mode: timing / active / inactive / toggle
   b) In case of active, inactive and toggle modes, select the polarity by writing CCxP in CCER register
   c) Disable the preload feature for CCx by writing OCxPE in CCMRx register
   d) Enable the capture / compare output by writing CCxE in CCERx register.
4. Enable the counter by setting the CEN bit in the TIMx_CR1 register
5. Set the CCxIE / CCxDE bit if an interrupt / DMA request is to be generated.

Timer output compare timing / delay computation

CCx update rate = CK_CNT / TIMx_ARRx

CCx delay = CCRx / CK_CNT

- If internal clock: CK_CNT= CK_PSC / (PSC + 1;
- If external clock mode2: CK_CNT = CK_PSC / ((ETPS)*(PSC + 1))
- If external clock mode1: CK_CNT = CK_PSC / (PSC + 1):
  – if ETRF used as clock source: CK_PSC = ETR_CLK / ETPS
  – if TIxFPx used as clock source: CK_PSC = TIx_CLK / ICPS
  – if TI1F_ED (filtered edge detection) used as clock source: CK_PSC = TI1_ED_CLK
  – if ITRx (another timer) used as clock source: CK_PSC = ITRx_CLK.

For more details on using the timer in this mode, refer to the examples provided in the STM32Cube package libraries in Examples\TIM\TIM_OCToggle, \TIMxOCActive and \TIM_OCInactive subfolders.

2.5 Timer in PWM mode

The timer is able to generate PWM in edge-aligned mode or in center-aligned mode with a frequency determined by the value of the TIMx_ARR register, and a duty cycle determined by the value of the TIMx_CCRx register.

PWM mode 1

- In up-counting, channelx is active as long as CNT< CCRx, otherwise it is inactive
- In down-counting, channelx is inactive as long as CNT> CCRx, otherwise it is active.

PWM mode 2

- In up-counting, channelx is inactive as long as CNT < CCRx, otherwise it is active
- In down-counting, channelx is active as long as CNT > CCRx, otherwise it is inactive.

Note: Active when OCREF = 1, inactive when OCREF = 0.

To configure the timer in this mode:

1. Configure the output pin:
   a) Select the output mode by writing CCS bits in CCMRx register
   b) Select the polarity by writing the CCxP bit in CCER register.
2. Select the PWM mode (PWM1 or PWM2) by writing OCxM bits in CCMRx register
3. Program the period and the duty cycle respectively in ARR and CCRx registers
4. Set the preload bit in CCMRx register and the ARPE bit in the CR1 register
5. Select the counting mode:
   a) PWM edge-aligned mode: the counter must be configured up-counting or down-counting
   b) PWM center aligned mode: the counter mode must be center aligned counting mode (CMS bits different from '00').
6. Enable the capture compare
7. Enable the counter.

For more details on using the timer in this mode, refer to the STM32CubeF3 firmware package examples in the Examples\TIM\TIM_PWMOutput subfolders.

2.6 Timer in one pulse mode

One Pulse Mode (OPM) is a particular case of the input capture mode and the output compare mode. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

To configure the timer in this mode:

1. Configure the input pin and mode:
   a) Select the TIxFPx trigger to be used by writing CCxS bits in CCMRx register
   b) Select the polarity of the input pin by writing CCxP and CCxNP bits in CCER register
   c) Configure the TIxFPx trigger for the slave mode trigger by writing TS bits in SMCR register
   d) Select the trigger mode for the slave mode by writing SMS = 110 in SMCR register.
2. Configure the output pin and mode:
   a) Select the output polarity by writing CCyP bit in CCER register
   b) Select the output compare mode by writing OCyM bits in CCMRy register (PWM1 or PWM2 mode)
   c) Set the delay value by writing in CCRy register
   d) Set the auto reload value to have the desired pulse: pulse = TIMy_ARR - TIMy_CCRy.
3. Select the one pulse mode by setting the OPM bit in CR1 register, if only one pulse is to be generated. Otherwise this bit should be reset:

   Delay = CCRy/(TIMx_CLK/(PSC + 1))

   Pulse-Length= (ARR+1-CCRy)/(TIMx_CLK/(PSC+1)).

For more details on using the timer in this mode refer to the examples provided in the STM32Cube package in the Examples\TIM\TIM_OnePulse sub folder.

2.7 Timer in asymmetric PWM mode

This feature is not available in the original series. See *Section 1: Overview* for more details.

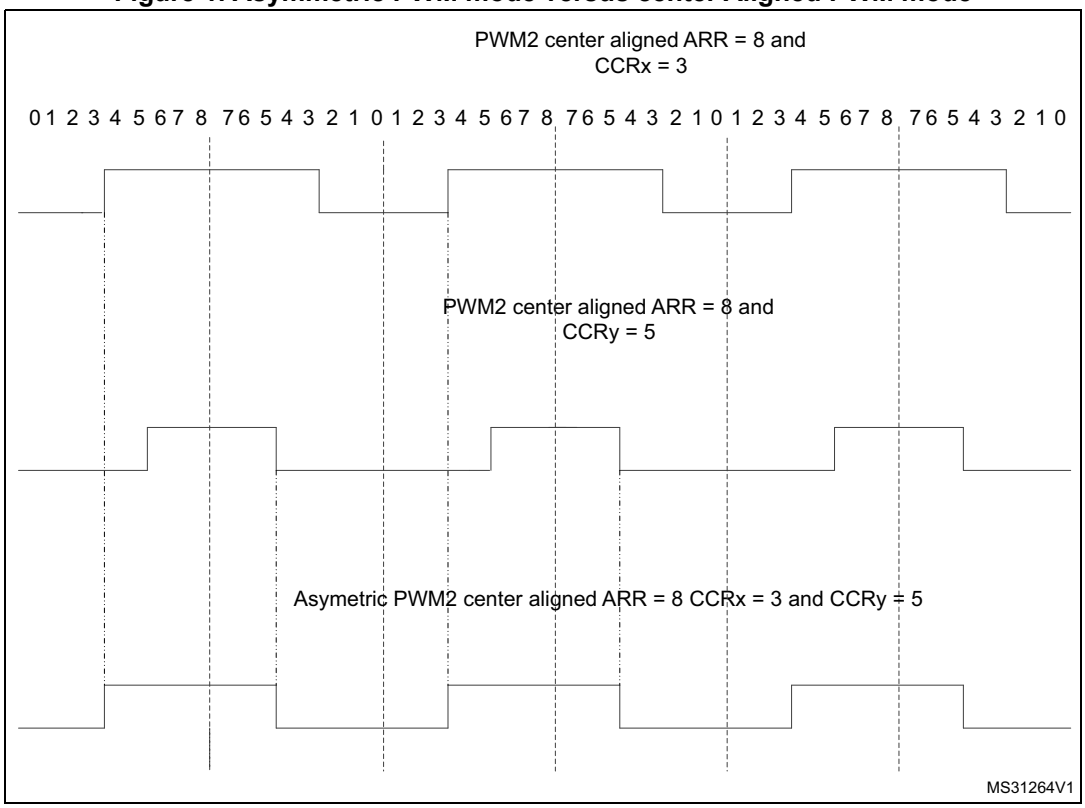
The asymmetric mode allows center-aligned PWM signals to be generated with a programmable phase shift.

For a dedicated channel, the phase shift and the pulse length are programmed using the two TIMx_CCRx register (TIMx_CCR1 and TIMx_CCR2 or TIMx_CCR3 and TIMx_CCR4), the frequency is determined by the value of the TIMx_ARR register. So the asymmetric PWM mode can be selected independently on two channels by programming the OCxM bits in TIMx_CCMRx register:

- OCxM = 1110 to use the asymmetric PWM1, in this mode the output reference has the same behavior as in PWM1 mode. When the counter is counting up the output reference is identical to OC1/3REF, when the counter is down counting, the output reference is identical to OC2/4REF
- OCxM = 1111 to use the Asymmetric PWM2, in this mode the output reference has the same behavior as in PWM2 mode. When the counter is counting up the output reference is identical to OC1/3REF, when the counter is down counting, the output reference is identical to OC2/4REF.

The following figure summarizes the asymmetric behavior versus the center aligned PWM mode:

Figure 1. Asymmetric PWM mode versus center Aligned PWM mode



To configure the timer in this mode:

1. Configure the output pin:
   a) Select the output mode by writing CCS bits in CCMRx register
   b) Select the polarity by writing the CCxP bit in CCER register.
2. Select the Asymmetric PWM mode (Asymmetric PWM1 or Asymmetric PWM2) by writing OCxM bits in CCMRx register
3. Program the period, the pulse length and the phase shift respectively in ARR, CCRx and CCRy registers
4. Select the counting mode: the Asymmetric PWM mode is working only with center aligned mode: the counter mode must be center aligned counting mode (CMS bits different from '00')
5. Enable the capture compare
6. Enable the counter.

STM32CubeF3 firmware package includes examples in the following directories:

- Projects\STM32F3-Discovery\Examples\TIM\TIM_Asymetric
- Projects\STM32303E_EVAL\Examples\TIM\TIM_Asymetric
- Projects\STM32303C_EVAL\Examples\TIM\TIM_Asymetric.

2.8 Timer in combined PWM mode

This feature is not available in the original series. See *Section 1: Overview* for more details.

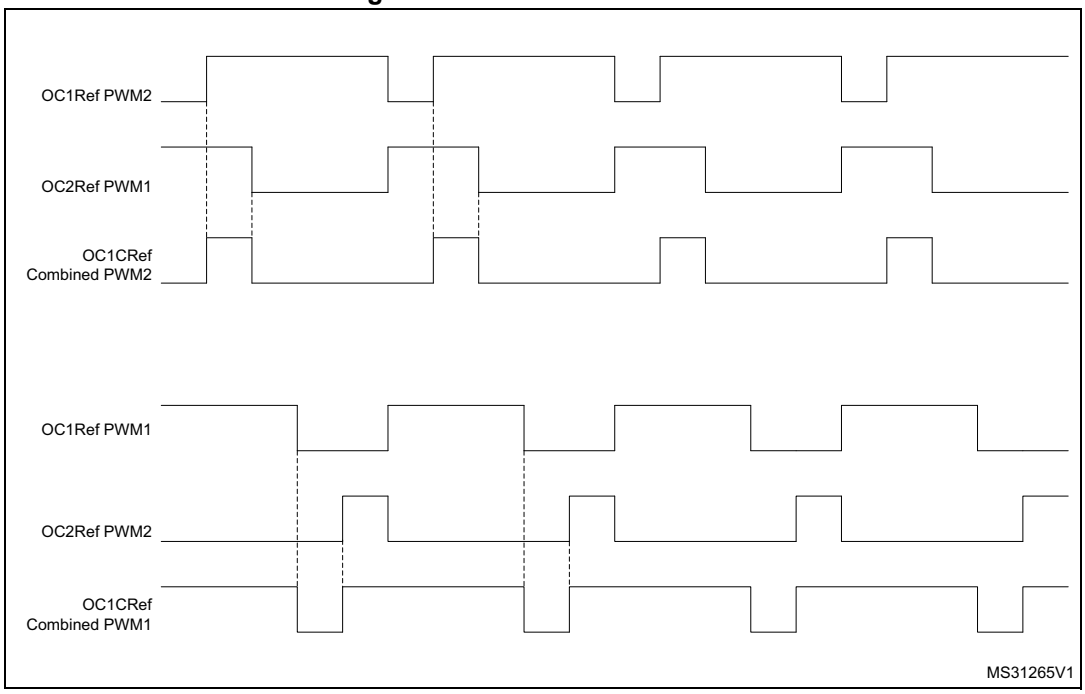
The combined mode allows edge or center aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. To generate a combined signal, the TIMx_CCRx and TIMx_CCRy must be used to program the delay and the phase shift. The frequency is determined by the value of the TIMx_ARR register.

The resulting signal (combined signal) is made of an OR or AND logical combination of two reference PWMs. So the combined PWM mode can be selected independently on two channels by programming the OCxM bits in TIMx_CCMRx register:

- OCxM = 1100 to use the Combined PWM1, in this case the combined output reference has the same behavior as in PWM mode 1. The combined output reference is the logical OR between OC1/3REF and OC2/4REF
- OCxM = 1101 to use the Combined PWM2,in this case the combined output reference has the same behavior as in PWM mode 2. The combined output reference is the logical AND between OC1/2REF and OC2/4REF.

The following figures resume the combined mode:

Figure 2. Combined PWM mode



To configure the timer in this mode:

1. Configure the output pin:
   a) Select the output mode by writing CCS bits in CCMRx register;
   b) Select the polarity by writing the CCxP bit in CCER register.
2. Select the Combined PWM mode (Combined PWM1 or Combined PWM2) by writing OCxM bits in CCMRx register;
3. Program the period, the delay and the phase shift respectively in ARR, CCRx and CCRy registers;
4. Select the counting mode:
   a) Edge-aligned mode: the counter must be configured up-counting or down-counting;
   b) Center aligned mode: the counter mode must be center aligned counting mode (CMS bits different from '00').
5. Enable the capture compare;
6. Enable the counter.

2.9 Retriggerable one pulse mode

This feature is not available in the original series. See *Section 1: Overview* for more details.

The retriggerable one pulse mode is a one pulse mode with these additional characteristics:

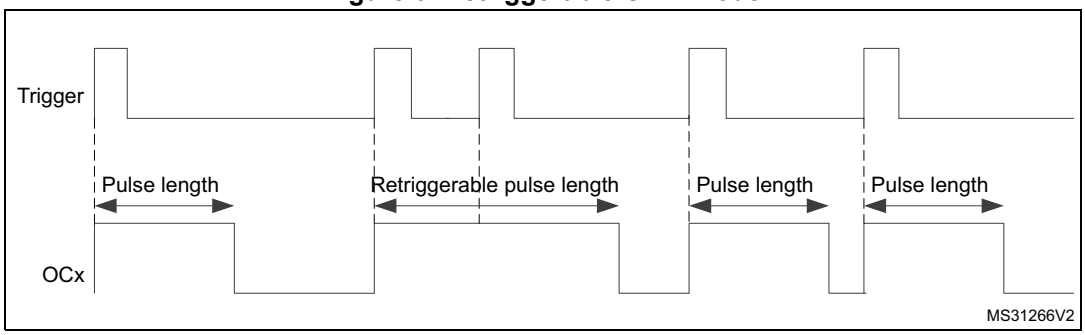
- The pulse starts as soon as the trigger occurs (no programmable delay);
- The pulse is extended if a new trigger occurs before the previous one is completed.

If the counter is configured in up-counting mode, the corresponding CCRx must be set to 0, in this case the pulse length is determined by ARR register. If the timer is configured in down-counting mode, the ARR must be set to 0 in this case the pulse length is determined by CCRx register. As for the OPM mode, there are two retriggerable one pulse mode, retriggerable OPM mode 1 and retriggerable OPM mode 2:

- Retriggerable OPM mode 1 is selected by setting the OCXM bits to 1000:
  - In up-counting mode, channel is inactive until a trigger event is detected (on TRGI signal), the comparison is performed like in PWM mode 1, then the channel becomes inactive again at the next update;
  - In down-counting mode, channel is active until a trigger event is detected (on TRGI signal), the comparison is performed like in PWM mode 1, then the channel becomes active again at the next update.
- Retriggerable OPM mode 2 i s selected by setting the OCXM bits to 1001:
  - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal), the comparison is performed like in PWM mode 2, then the channel becomes inactive again at the next update;
  - In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal), the comparison is performed like in PWM mode 1, then the channel becomes inactive again at the next update.

Figure 3 presents an example of the retriggerable OPM mode.

Figure 3. Retriggerable OPM mode



To configure the timer in this mode:

1. Configure the input pin and mode:
   a) Select the TIxFPx trigger to be used by writing CCxS bits in CCMRx register;
   b) Select the polarity of the input pin by writing CCxP and CCxNP bits in CCER register;
   c) Configure the TIxFPx trigger for the slave mode trigger by writing TS bits in SMCR register;
   d) Select the Combined Reset + trigger mode for the slave mode by writing SMS = 1000 in SMCR register;
2. Configure the output pin and mode:
   a) Select the output polarity by writing CCyP bit in CCER register;
   b) Select the output compare mode by writing OCyM bits in CCMRy register (Retriggerable OPM mode 1 or retriggerable OPM mode 2);
   c) Set the Pulse length value by writing in CCRy register if the counter is down-counting or by writing in the ARR if the counter is up-counting.

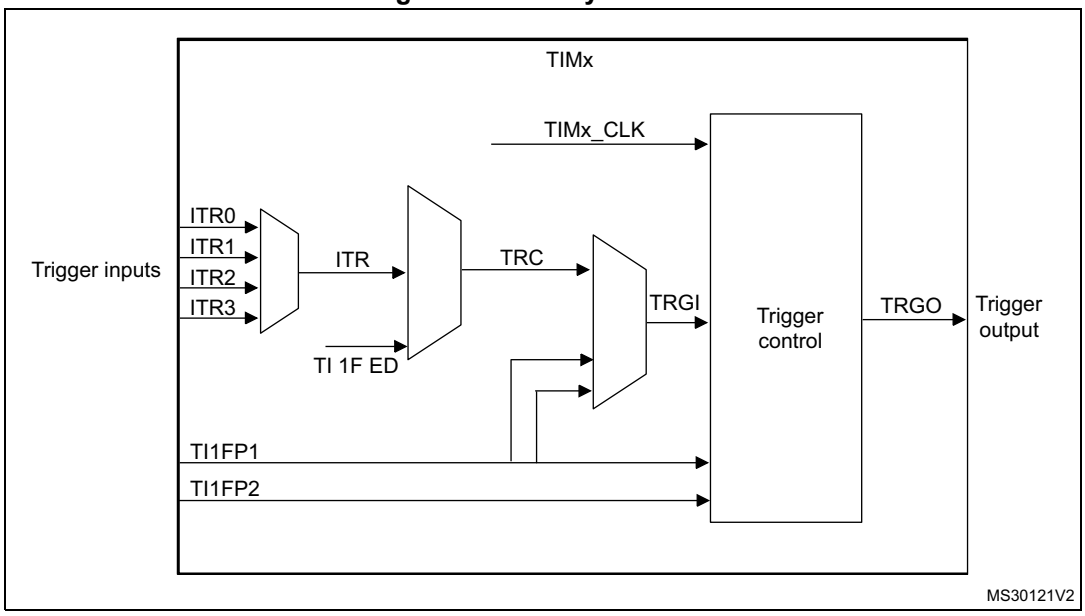
For more details on using the timer in this mode, refer to the examples provided in the STM32F30x standard peripheral libraries, in the /Project/STM32F30x_StdPeriph_Examples/ TIM/Retriggerable OPM folder.

3 Timer synchronization

3.1 Timer system link

STM32xx Series timers are linked together internally for timer synchronization or chaining. Each timer has several internal input and output triggers. These signals allow timer interconnection. *Figure 4* illustrates the time interconnection.

Figure 4. Timer system link



3.2 Master configuration

When a timer is selected as a master timer, the corresponding trigger output signal is used by the slave internal trigger (when configured). The trigger output can be selected from the following list:

- **Reset**: the UG bit from the TIMx_EGR register is used as a trigger output (TRGO);
- **Enable**: the counter enable signal is used as a trigger output (TRGO) to start several timers at the same time, or to control a window in which a slave timer is enabled;
- **Update**: the update event is selected as trigger output (TRGO). For example, a master timer can be used as a prescaler for a slave timer;
- **Compare pulse**: the trigger output sends a positive pulse when the CC1IF flag is to be set (even if it was already high) as soon as a capture or a compare match occurs;
- **OC1Ref**: OC1REF signal is used as trigger output (TRGO);
- **OC2Ref**: OC2REF signal is used as trigger output (TRGO);
- **OC3Ref**: OC3REF signal is used as trigger output (TRGO);
- **OC4Ref**: OC4REF signal is used as trigger output (TRGO).

To configure a timer in master mode:

1. Configure the timer
2. Select the trigger output to be used, by writing the MSM (Master /Slave Mode selection) bits in TIMx_CR2 register
3. Enable the MSM (Master/Slave Mode) bit in the SMCR register to allow a perfect synchronization between the current timer and its slaves (through TRGO).

On selected devices such as STM32G4 Series for example, the advanced-control timer can generate two trigger outputs: TRGO as described above and TRGO2 (used for TIM and ADC synchronization) which can be selected from the following list:

- Reset - the UG bit from the EGR register is used as trigger output (TRGO2)
- Enable - the Counter Enable signal CNT_EN is used as trigger output (TRGO2). It is useful to start several timers simultaneously or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logical AND between CEN control bit and the trigger input when configured in gated mode
- Update - The update event is selected as trigger output (TRGO2). For instance a master timer can then be used as a prescaler for a slave timer
- Compare Pulse - The trigger output sends a positive pulse when the CC1IF flag is set high (even if it was already set high), as soon as a capture or a compare match occurs
- Compare - OC1REF signal is used as trigger output (TRGO2)
- Compare - OC2REF signal is used as trigger output (TRGO2)
- Compare - OC3REF signal is used as trigger output (TRGO2)
- Compare - OC4REF signal is used as trigger output (TRGO2)
- Compare - OC5REF signal is used as trigger output (TRGO2)
- Compare - OC6REF signal is used as trigger output (TRGO2)
- Compare Pulse - OC4REF rising or falling edges generate pulses on TRGO2
- Compare Pulse - OC6REF rising or falling edges generate pulses on TRGO2
- Compare Pulse - OC4REF rising or OC6REF rising edges generate pulses on TRGO2
- Compare Pulse - OC4REF rising or OC6REF falling edges generate pulses on TRGO2
- Compare Pulse - OC5REF rising or OC6REF rising edges generate pulses on TRGO2
- Compare Pulse - OC5REF rising or OC6REF falling edges generate pulses on TRGO2.

3.3 Slave configuration

The slave timer is connected to the master timer through the input trigger. Each ITRx is connected internally to another timer, and this connection is specific for each STM32Fx/Gx/Hx/Lx//Wx series product as stated on the first page.

The slave mode can be:

- **Reset mode**: rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers
- **Gated mode**: the counter clock is enabled when TRGI is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both counter start and stop are controlled
- **Trigger mode**: the counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the counter start is controlled
- **External clock mode** 1: rising edges of the selected trigger TRGI clock the counter
- **Combined reset + trigger mode**: rising edge of the selected TRGI reinitializes the counter, generates an update of the registers and starts the counter. This feature is not available in the original series. See *Section 1: Overview* for more details.

To configure a timer in slave mode:

1. Select the slave mode to be used by writing SMS (slave mode selection) bits in SMCR register
2. Select the internal trigger to be used by writing TS (trigger selection) bits in SMCR register.

For more details on using the timer in this mode, refer to the examples provided in the STM32Cube package:

`Examples\TIM\TIM_CascadeSynchro,\TIM_ExtTriggerSynchro\TIM_Synchronization and \TIM_ParallelSynchro folders.`

4 Advanced features for motor control

4.1 Signal generation

The STM32Fx/Gx/Hx/Lx//Wx Series timer can output two complementary signals and manage the on/ off state switching of the outputs.

The complementary signals OCx and OCxN are activated by a combination of several control bits: the CCxE and CCxNE and the MOE, OISx, OISxN, OSSI and OSSR bits.

The main output enable (MOE) bit is reset once a break input becomes active. It is set by software or automatically based on the Automatic Output Enable (AOE) bit. When the MOE bit is reset, the OCx and OCxN outputs are disabled or forced to idle state (OISx OISxN), depending on whether the OSSI bit is set or not.

Note: The MOE bit is valid only on the channels that are configured as output.

The Off-State Selection for Run mode (OSSR) bit is used when MOE=1 to determine the pin output when the channel output is not enabled. When this bit is set, OCx and OCxN outputs are set to their inactive level as soon as their complementary bits CCxE=1 or CCxNE=1. The output is still controlled by the timer.

The Off-State Selection for Idle mode (OSSI) bit is used when MOE=0 due to a break event or by a software write, on channels configured as outputs. When this bit is set, OCx and OCxN outputs are first forced with their inactive level, then forced to their idle level after the deadtime. The timer maintains its control over the output.

Table 5: Advanced timer configurations explains the possible configurations of the advanced timer.

Table 5. Advanced timer configurations

Control bits					Output state		Typical use
MOE	**OSSI**	**OSSR**	**OCxE**	**OCxNE**	**OCx output state**	**OCxN output state**	
1	x	0	0	0	Output disable	Output disable	General purpose
	x	0	0	1	Output disable	OCxREF + polarity	
	x	0	1	0	OCxREF + polarity	Output disable	
	x	0	1	1	OCxREF + polarity + Deadtime	(not OCxREF) + polarity + Deadtime	Motor control (sinewave)
	x	1	0	0	Output disabled	Output disabled	Motor control (6-steps)
	x	1	0	1	Off-state	OCxREF + polarity	
	x	1	1	0	OCxREF + polarity	Off-state	
	x	1	1	1	OCxREF + polarity + deadtime	(not OCxREF) + polarity + deadtime	Motor control (sinewave)
MOE	**OSSI**	**OSSR**	**OCxE**	**OCxNE**	**OCx output state**	**OCxN output state**	**Comments**
0	0	x	0	0	Output disable		Outputs disconnected from I/O ports
	0	x	0	1			
	0	x	1	0			
	0	x	1	1			
	1	x	0	0	Off-state (outputs are first forced with their inactive level then forced to their idle level after the deadtime.)		All PWMs OFF (low Z for safe stop)
	1	x	0	1			
	1	x	1	0			
	1	x	1	1			

Note: 1 Deadtime insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit.

2 When only OCxN is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP=0 then OCxN=OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1) OCx becomes active when OCxREF is high, whereas OCxN is complemented and becomes active when OCxREF is low.

4.2 Combined three-phase PWM mode

This feature is not available in the original series. See*Section 1: Overview* for more details.

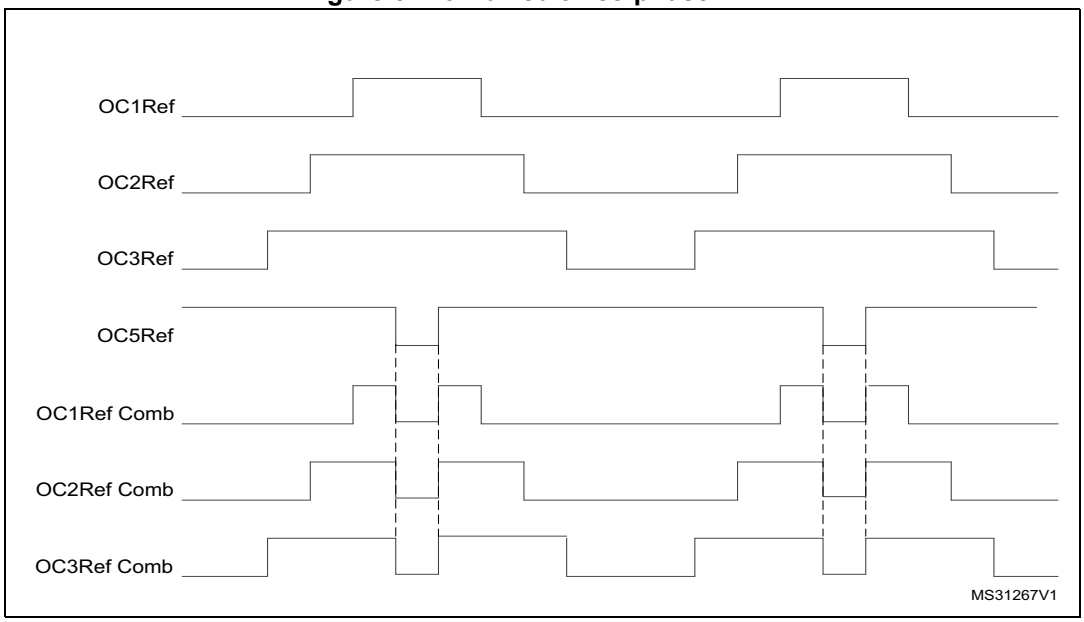
The combined three-phase mode allows the generation of one to three center-aligned PWM signals with a single programmable signal ANDed in the middle of the pulses. The configuration is helpful for shunt resistor current sensing applications. Refer to UM1052 for further reading on this topic.

Using the 3-bits GC5C[3:1] in the TIMx_CCR5, each channel of the TIM can be a combination between the original signal and the OC5Ref signal:

- If GC5C1 is set, OC1 output is controlled by TIMx_CCR1 and TIMx_CCR5
- If GC5C2 is set, OC1 output is controlled by TIMx_CCR2 and TIMx_CCR5
- If GC5C3 is set, OC1 output is controlled by TIMx_CCR3 and TIMx_CCR5.

The *Figure 5* below illustrates an example of this mode:

Figure 5. Combined three-phase PWM



To configure the timer in this mode:

1. Configure the output pin:
   a) Select the output mode by writing CCS bits in TIMx_CCMRx register
   b) Select the polarity by writing the CCxP bit in TIMx_CCER register.
2. Configure the used channel (1, 2 or/and 3) in PWM mode:
   a) Configure the frequency, the duty cycle and the polarity
   b) Select the PWM 1 or 2.
3. Configure the Channel 5 in PWM mode with the desired parameter (duty cycle)
4. Select the Combined PWM mode by programming the GC5Cx bits
5. Select the Center aligned mode as counting mode
6. Enable the capture compare
7. Enable the counter.

For more details on using the timer in this mode refer to the examples provided in the STM32CubeF3 firmware package in the Examples

\TIM\TIM_Combined sub folder.

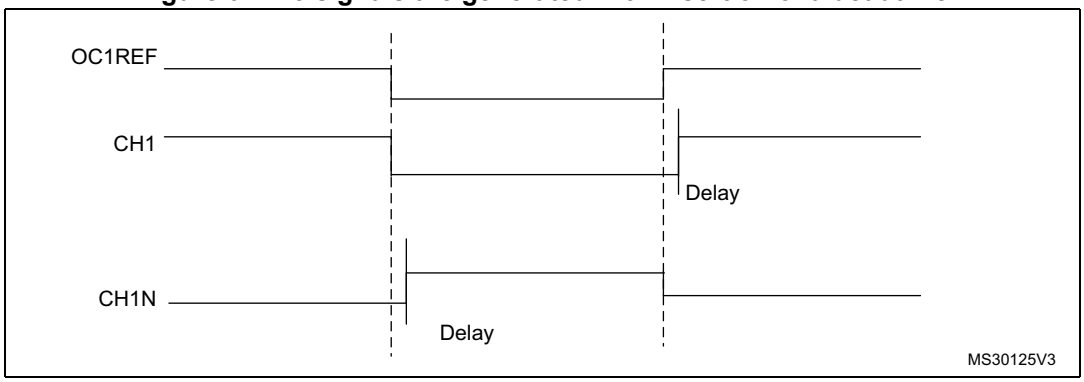
4.3 Specific features for motor control applications

4.3.1 Complementary signal and deadtime feature

The STM32xx Series advanced timers can generate up to three complementary outputs with the insertion of deadtime.

To use the complementary signal for one channel, set the two output compare enable bits of this channel and its complementary (OCxE and OCxNE) channel. If the deadtime bits are not zero, the two signals are generated with the insertion of a deadtime as illustrated in *Figure 6*:

Figure 6. Two signals are generated with insertion of a deadtime



Note: The deadtime parameter is computed using the DTG[7:0] bits and the deadtime clock (Tdtg).

The deadtime clock is computed as follows:

Tdtg = TDTS, if DTG[7] = 0
Tdtg = 2 x TDTS, if DTG[6] = 0
Tdtg = 8 x TDTS, if DTG[5] = 0
Tdtg = 16 x TDTS, if DTG[7:5] = 111

Where: TDTS = TCK_INT, if CKD[1:0] = 00
TDTS = 2 x TCK_INT, if CKD[1:0] = 01
TDTS = 4 x TCK_INT, if CKD[1:0] = 10

Note: TCK_INT is the internal clock timer.

The deadtime delay is computed using the following formula:

deadtime = DTG[7:0]x Tdtg, if DTG[7] = 0
deadtime = (64+DTG[5:0]) x Tdtg, if DTG[6] = 0
deadtime = (32+DTG[4:0]) x Tdtg, if DTG[5] = 0
deadtime = (32+DTG[4:0]) x Tdtg, if DTG[7:5] = 111

For more details on using the timer in this mode refer to the examples provided in the STM32Cube package examples in the following directories:

- Examples\TIM\TIM_Complementary Signals
- Examples\TIM\TIM_Combined.

Note: At the time of writing this AN, the complementary signals example is only available for STM32F0, STM32F1, STM32F3, STM32F4, STM32F7 and STM32H7 Series microcontrollers; the combined example only for STM32F3 and STM32H7 Series microcontrollers.

4.3.2 Break input

The break input is an emergency input in the motor control application. The break function protects power switches driven by PWM signals generated with the advanced timers. The break input is usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the TIM outputs and forces them to a predefined safe state.

The break event is generated by:

- The BRK input that has a programmable polarity and an enable bit BKE
- The CSS (Clock Security System)
- Software, by setting the BG bit in the EGR register

When a break event occurs:

- The MOE bit (main output enable) is cleared
- The break status flag is set and an interrupt request can be generated
- Each output channel is driven with the level programmed in the OISx bit.

Revised break inputs

Break inputs Break1 and Break2 are not available on the original series.

The break can be generated by any of the two BRK inputs which have:

- a programmable polarity (BKPx bit in the TIMx_BDTR Register)
- a programmable enable bit (BKEx in the TIMx_BDTR Register)
- a programmable filter (BKxF[3:0] bits in the TIMx_BDTR Register) to avoid spurious events.

Table 6 presents the two break inputs priorities.

Table 6. Behavior of timer outputs versus Break1 and Break2 inputs

Break input 1	Break input 2	OCxN output	OCx output
Active	Inactive	ON after deadtime insertion	OFF
Inactive	Active	OFF	OFF
Active	-	ON after deadtime insertion	OFF

4.3.3 Locking mechanism

The advanced timer registers and bits can be protected or locked in order to safeguard the application using the locking mechanism by programming the LOCK bits in the BDTR register. There are three locking levels illustrated in *Table 7*.

Table 7. Locking levels

Level 1		LOCK Level 2(1)		LOCK Level 3(2)	
Register	**Bits**	**Register**	**Bits**	**Register**	**Bits**
CR2	OISx	**CR2**	OISx	**CR2**	OISx
	OISxN		OISxN		OISxN
BDTR	DTG[7:0]	**BDTR**	DTG[7:0]	**BDTR**	DTG[7:0]
	BKE		BKE		BKE
	BKP		BKP		BKP
	AOE		AOE		AOE
	BK2E(3)		OSSR		OSSR
	BK2P(3)		OSSI		OSSI
	BKF[3:0](3)	**CCER**	CCxP	**CCER**	CCxP
	BK2F[3:0](3)		CCxNP		CCxNP
-	-	-	-	**CCMRx**	OCxM
-	-	-	-		OCxPE

1. LOCK Level 2 = LOCK Level 1 + CC polarity bits (CCxP/CCxNP bits in TIMx_CCER).
2. LOCK Level 3 = LOCK Level 2 + CC control bits (OCxM and OCxPE.
3. Bits present in STM32L4/F7 Series and STM32F30x/F3x8 lines.

Note: The LOCK bits can only be written once after the reset. Once the BDTR register has been written, its content is frozen until the next reset.

4.3.4 Specific features for feedback measurement

Encoder modes

The incremental quadrature encoder is a type of sensor used in motor-control applications to measure the angular position and the rotation direction.

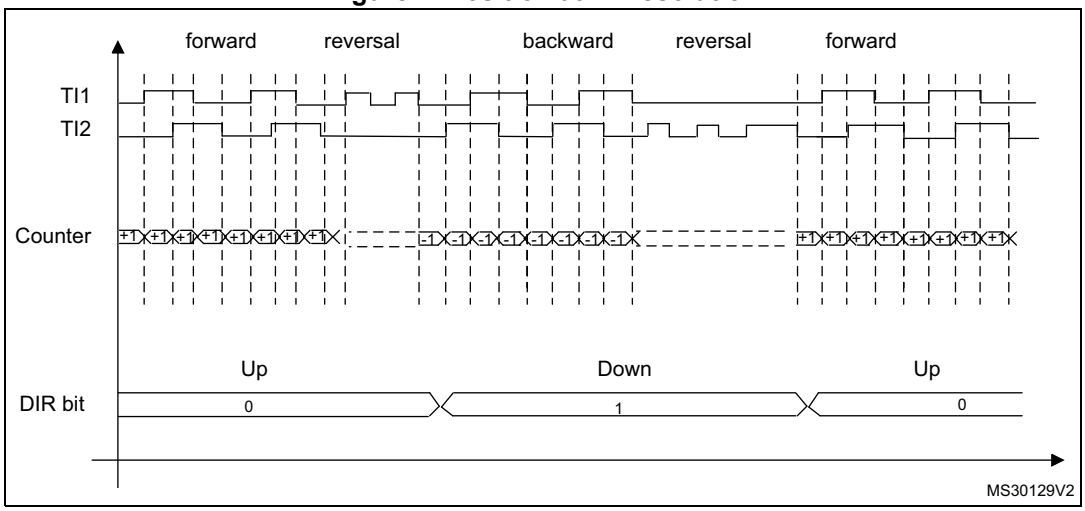
In general, the incremental quadrature encoder generates three signals: phase A, phase B and index.

The direction of the motor depends on whether Phase A leads Phase B, or Phase B leads Phase A. A third channel, Index pulse, occurs once per revolution and is used as a reference to measure an absolute position.

The Phase A and B output signals are connected to the encoder interface to compute the frequency and then determine the velocity and the position. Velocity and position information can be measured at X2 or X4 resolution. *Figure 7: Position at X4 resolution* and *Figure 8: Position at X2 resolution* explain the encoder interface function.

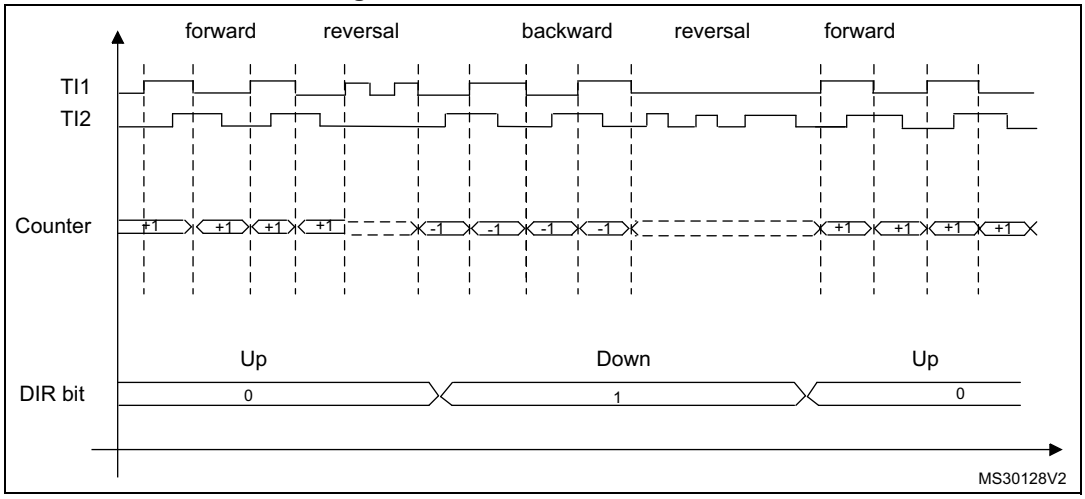
The timer's counter is incremented or decremented for each transition on both inputs TI1 and TI2.

Figure 7. Position at X4 resolution



The timer's counter is incremented or decremented for each transition on the selected input TI1 or TI2.

Figure 8. Position at X2 resolution



Note: In case of X2 resolution the counter can also be incremented on the TI1 edge.

In STM32 timer encoder interface mode, the encoder mode3 corresponds to the X4 resolution. In this mode, the counter counts up/down on both TI1 and TI2 edges.

The X2 resolution is selected when encoder mode 1 or mode 2 is selected, that is, the counter counts up/down on TI2 edge depending on the TI1 level, or the counter counts up/down on TI1 edge depending on TI2 level.

How to use the encoder interface

An external incremental quadrature encoder can be connected directly to the MCU without external interface logic. The third encoder output (index) which indicates the mechanical zero position, may be connected to an external interrupt input and triggers a counter reset.

The output signal of the incremental encoder is filtered by the STM32 timer input filter block to reject all noise sources that typically occur in motor systems. This filter is described in *Section 2.3: Timer input capture mode on page 13*.

TIM configuration in encoder mode

1. Select and configure the timer input:

- Input selection:
  - TI1 connected to TI1FP1 CC1S='01' in TIMx_CCMR1 register;
  - TI2 connected to TI2FP2 CC2S='01' in TIMx_CCMR1 register.
- Input polarity:
  - CC1P='0' and CC1NP='0'(CCER register, TI1FP1 non-inverted, TI1FP1=TI1);
  - CC2P='0' and CC2NP='0'(CCER register, TI1FP2 non-inverted, TI1FP2= TI2).

2. Select the encoder mode:

- Encoder mode1 (resolution X2 on TI2): SMS='001' in TIMx_SMCR register;
- Encoder mode2 (resolution X2 on TI1): SMS='010' in TIMx_SMCR register;
- Encoder mode3 (resolution X4 on TI1 and TI2): SMS='011' in TIMx_SMCR register.

3. Enable the timer counter:

- Set the counter enable bit, CEN='1' in TIMx_CR1 register.

Hall sensor

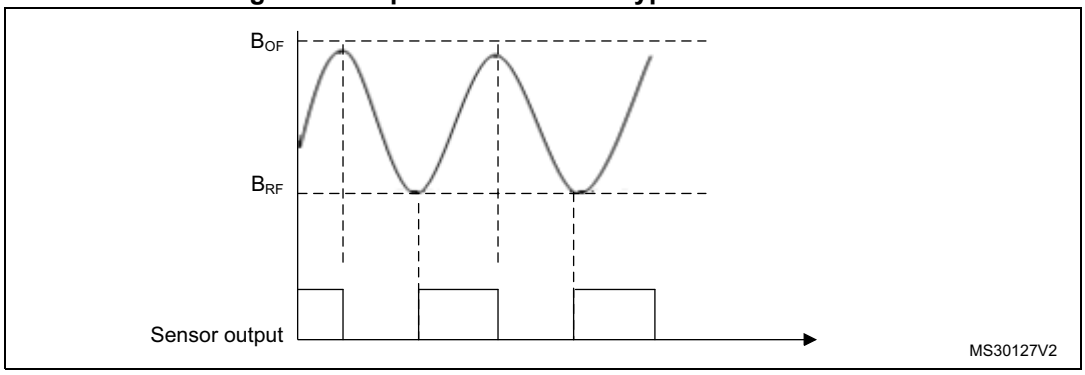
The Hall sensor is a type of sensor based on the Hall effect: when a conductor is placed in a magnetic field, a voltage is be generated perpendicular to both the current and the magnetic field.

There are four types of Hall sensor IC devices that provide a digital output: unipolar switches, bipolar switches, omni polar switches, and latches. The main difference between them is the output waveforms (pulse duration).

The digital Hall sensor provides a digital output in relation to the magnetic field which it is exposed. When the magnetic field increases and is greater than the B_{RP} (magnetic field release point value), the output is set to ON. When the magnetic field decreases and is lower than the B_{OP} (magnetic field operate point value) the output is set to OFF.

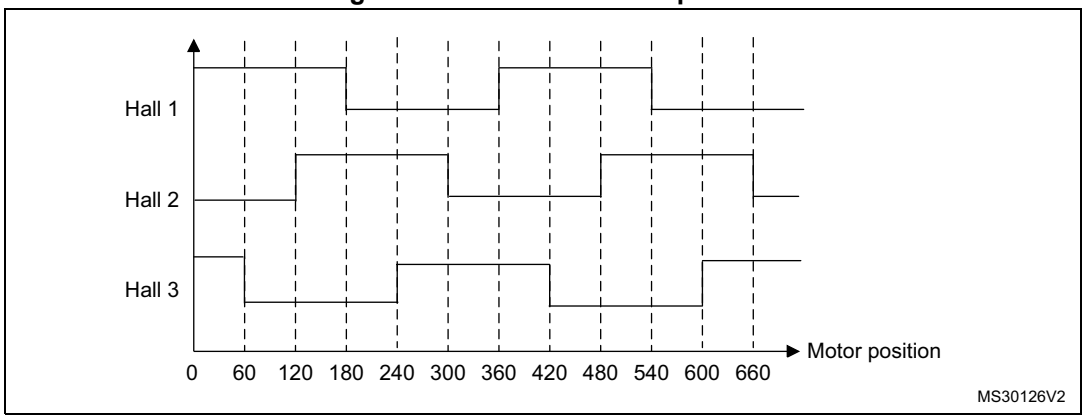
Figure 9 presents the output waveform of a typical Hall sensor.

Figure 9. Output waveform of a typical Hall sensor



Generally, the Hall sensor is used in the three-phase motor control. *Figure 10* presents the commutation sequence.

Figure 10. Commutation sequence



How to use the Hall sensor interface

The STM32 timers can interface with the Hall effect sensors via the standard inputs (CH1, CH2 and CH3). Setting TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins TIMx_CH1, TIMx_CH2 and TIMx_CH3.

The slave mode controller is configured in reset mode; the slave input is TI1F_ED. Thus, each time one of the 3 inputs toggles, the counter restarts counting from 0. This creates a time base triggered by any change on the Hall inputs.

Channel 1 is configured as an input capture mode and the capture signal is TRC. The captured value, which corresponds to the time, elapsed between 2 changes on the inputs, gives information on the motor speed.

TIM configuration in Hall sensor interface mode

1. Configure 3 timer inputs ORed to the TI1 input channel by writing the TI1S bit in TIMx_CR2 register to '1';
2. Program the time base: write the TIMx_ARR to the max value (the counter must be cleared by the TI1 change. Set the prescaler to a period longer than the time between 2 changes on the sensors;
3. Program channel 1 in capture mode (TRC selected): write the CC1S bits in the TIMx_CCMR1 register to '01'. The user can also program the digital filter if needed.

5 High-resolution timer applications

The high-resolution timer was designed specifically to control power conversion systems in lighting systems switch mode power supply. Even though it really excels in this role, it can of course be used in other applications with high-resolution timer requirements.

The HRTIM features up to 10 outputs which can be configured in various coupled and autonomous modes using five timing units tied to a common master for synchronization purposes. The synchronization with other timers is also facilitated. The HRTIM is strongly tied to ADCs and fault inputs for feedback purposes.

For more information about the high-resolution timer, read the reference manual of the particular MCU line.

Application related information can be found in *"High brightness LED dimming using the STM32F334 Discovery kit"* (AN4885) and *"Buck-boost converter using the STM32F334 Discovery kit"* (AN4449).

For practical examples check following STM32Cube package sub folders:

- Projects\STM32F3348-Discovery\Examples\HRTIM_BasicPWM
- Projects\STM32F3348-Discovery\Examples\HRTIM_BuckBoost
- Projects\STM32F3348-Discovery\Examples\HRTIM_BuckSyncRect
- Projects\STM32F3348-Discovery\Examples\HRTIM_DualBuck
- Projects\STM32F3348-Discovery\Examples\HRTIM_LLC_HalfBridge
- Projects\STM32F3348-Discovery\Examples\HRTIM_Multiphase
- Projects\STM32F3348-Discovery\Examples\HRTIM_Snippets
- Projects\STM32F3348-Discovery\Examples\HRTIM_TM_PFC.

6 Low-power timer

The main difference and advantage of the LPTIM compared to any other timer peripheral in STM32 microcontroller family is the ability to continue working even in stop mode and trigger events that will wake the MCU up from the stop mode. Depending on the selected clock source, the runtime power consumption can be substantially lower compared to a general purpose timer. While it can perform a similar job to the general purpose timer, let's focus on the task for which it is designed for.

6.1 Wakeup timer implementation

The LPTIM can be configured to periodically wake up the MCU from stop mode, for example to refresh a display or to read a sensor. For this purpose it needs to be configured to use a clock source that remains functional in stop mode. It can be an LSE, an LSI oscillator or an external clock source. An external clock source is fed to the LPTIM Input1 which has configured to use it (CKSEL is appropriately configured).

To configure the timer in this mode:

1. Configure a clock source;
2. Code the interrupt handler, callback function and enable the LPTIM interrupt;
3. Setup the LPTIM peripheral:
   a) Clock source selection;
   b) Set timing range using prescaler;
   c) Set trigger event (software or external signal).
4. Enable and start the timer;
5. Go to stop mode.

For more details on using the timer in this mode, refer to examples provided in the STM32Cube package in the Examples\LPTIM\LPTIM_Timeout sub folder.

6.2 Pulse counter

In some applications the microcontroller needs to keep track of some external events, but it is not desirable to wake it up from stop mode each time. In this case, the LPTIM is configured as pulse counter. Use the timer period/compare setting to set the number of events required to wake the microcontroller.

To configure the timer in this mode:

1. Configure a clock source;
2. Code the interrupt handler callback function and enable the LPTIM interrupt;
3. Setup the LPTIM timer peripheral:
   a) Clock source and counter source selection. Only input1 can be used as a clock source;
   b) Typical configuration selection is immediate update mode and software trigger source.
4. Enable and start the timer;
5. Go to Stop mode.

For more details on using the timer in this mode, refer to the examples provided in the STM32Cube package in the Examples\LPTIM\LPTIM_PulseCounter sub folder.

7 Specific applications

7.1 Infrared application

The STM32 general-purpose timers can be used to emulate several infrared protocols. An example of this application type is given in the application note *Implementation of transmitters and receivers for infrared remote control protocols with STM32Cube (*AN4834*).*

This application note describes a software solution for implementing an RC5/SIRC receiver and transmitter using the STM32 general-purpose timers.

This solution uses a specialized feature called IRTIM for implementation of the transmitter part. IRTIM is available on STM32F0, STM32F3, STM32G0, STM32G4 and STM32L4 Series microcontroller.

7.2 3-phase AC and PMSM control motor

The STM32 advanced and general-purpose timers with ADC and DAC are used to control two types of 3-phase motor: AC induction motor and PMSM, with different current sensing methodologies:

- Isolated current sensing (also referred as sensor-less solution);
- Three shunt resistors;
- Single shunt resistor (ST patented solution).

The STM32 timers are used also in the feedback loop to interface with the different sensors used in the different rotor position feedback:

- Tachogenerator;
- Quadrature encoder;
- Hall sensors: 60° and 120° placement.

For more details, refer to: stm32_pmsm_foc_motorcontrol_fmlib.

7.3 Six-step mode

The six-step mode is a specific mode of STM32 advanced timers. When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE and CCxNE bits. The preload bits are transferred to the shadow bits at the COM (commutation event).

The user can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on TRGI rising edge).

An application example of the use of this mode is the control of the brush-less 3-phase DC motor (3-phase BLDC motor).

Configuring the timer to generate a six step signal to control a brush-less 3-phase DC motor (3-phase BLDC motor)

- Time base configuration: prescaler, period, clock source;
- Channels 1, 2, 3 and 4 configured in PWM mode;
- Set the capture compare preload control bit CCPC;
- Enable the commutation interrupt source;
- Use the system tick to generate time base;
- For each commutation event, the TIM configuration is updated for the next commutation event.

For more details on using the timer in this mode, refer to the examples provided in the STM32Cube package in the Examples\TIM\TIM_6Steps folder.

Note: This example is not available for STM32L1 Series.

8 Revision history

Table 8. Document revision history

Date	Revision	Changes
21-Feb-2012	1	Initial release.
22-Oct-2012	2	Added support for STM32F30x, STM32F31x, STM32F37x, STM32F38x.
12-Feb-2014	3	Added support for STM32F0 Series, STM32F358xC. Replaced “basic timers” by “general-purpose” timers in the whole document. Updated *Section 2.1.2: External clock*. Updated*Section 2.4: Timer in output compare mode*. Updated *Section 2.5: Timer in PWM mode*. Updated *Section 7.3: Six-step mode*.
28-Jan-2015	4	Extended the applicability to STM32F303xDxE. Updated: – *Table 1: Applicable products* – *Table 2: Simplified overview of timer availability in STM32Fx products* – The document title and introduction Added references to timer examples available in STM32CubeF3 firmware package where applicable.
15-Apr-2016	5	Extended coverage to STM32F7 Series, STM32L0 Series and STM32L4 Series. Added: – *Section 5: High-resolution timer applications*. – *Section 6: Low-power timer*. Updated: – *Table 2.: Simplified overview of timer availability in STM32Fx products*. – *Table 4: Timer features overview*. – *Section 2.4: Timer in output compare mode*.

Table 8. Document revision history (continued)

Date	Revision	Changes
28-Jul-2016	6	Updated *Section 2.2: Time base generator*: corrected values on the formulas for “update event” on the examples for *update event period* and *external clock mode2*.
08-Apr-2019	7	Added the STM32WB Series, STM32H7 Series, STM32G0 Series, STM32G4 Series in *Table 1: Applicable products*. Updated; – *Section 1: Overview* content. – *Table 2: Simplified overview of timer availability in STM32Fx products* footer information. – *Table 4: Timer features overview* content and foot note – *Section 2.1.1: Internal clock* – *Section 2.6: Timer in one pulse mode* Pulse-Length definition. – *Section 2.7: Timer in asymmetric PWM mode*, *Section 2.8: Timer in combined PWM mode* and *Section 2.9: Retriggerable one pulse mode* with the unsupported series. – *Section 3.2: Master configuration* with STM32G4 Series reference. – *Section 3.3: Slave configuration* removed specified references to STM32L4/F7 Series and STM32F30x/F3x8 lines in and updated with the unsupported series. – *Section 4.2: Combined three-phase PWM mode* updated the title and added unsupported series. - Note updated in *Section 4.3.1: Complementary signal and deadtime feature*. - *Section 4.3.2: Break input* Revised break inputs section for supported series. - *Section 7.1: Infrared application* included additional supported series.