STM32 — Timer/Counter

By Prof. Shiao-Li Tsao
NCTU CS 2016

Why?

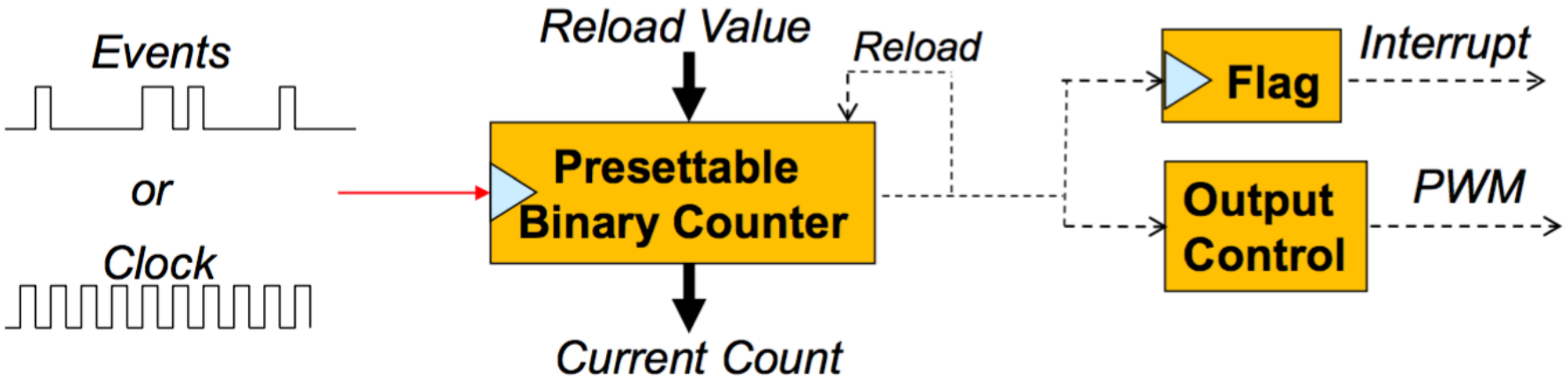
- Why we need counters?
- Why we need timers?
- Why timers/counters are together?
- Why so many different counters/timers?

Timing functions

- Periodically interrupt CPU to perform tasks
  - Sample sensor readings (temperature, pressure, etc.)
  - Generate music samples
- Provide accurate time delays
  - Instead of software loops
- Generate pulses or periodic waveforms
  - PWM signal for motor control
  - Strobe pulse for an external device
- Measure duration of an external event
  - Tachometer signal period to measure motor speed

Source: http://www.eng.auburn.edu/~nelson/courses/elec5260_6260/slides/timers1.pdf

Timer/Counter General Architecture



STM32 Timer/Counter

- Advanced Control Timer
  - TIM1 and TIM8
  - Input capture, output compare, PWM, one pulse mode
  - 16-bit auto-reload register
  - Additional control for driving motor or other devices
- General Purpose Timer
  - TIM2 to TIM5
  - Input capture, output compare, PWM, one pulse mode
  - 16-bit or 32-bit auto-reload register
- General Purpose Timer
  - TIM9 to TIM14
  - Input capture, output compare, PWM, one pulse mode
  - Only 16-bit auto-reload register
- Basic Timer (Simple timer)
  - TIM6 and TIM7
  - Can be generic counter and internally connected to DAC
  - 16-bit auto reload register
- 24 bit system timer(SysTick) – standard in all Cortex-M CPUs

6,7	16,17	15	3,4	2	1,8	Timer
×	×	×	×		×	16 Bits
				×		32 Bits
×	×	×	×	×	×	Up
			×	×	×	Down
			×	×	×	Up/Down
×	×	×	×	×	×	Auto-Reload
	×	×	×	×	×	Input Capture
	×	×	×	×	×	Output Compare
	×	×	×	×	×	Edge-aligned PWM
			×	×	×	Center-aligned PWM
	×	×	×	×	×	One-pulse mode output
	×	×			×	Complementary outputs with programmable dead-time
		×	×	×	×	Synchronization circuit to control the timer with external signals and to interconnect several timers together
	×	×			×	Repetition counter to update the timer registers only after a given number of cycles of the counter
	×	×			×	Break inputs to put the timer's output signals in a safe user selectable configuration
×	×	×	×	×	×	Interrupt/DMA generation
			×	×	×	Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
			×	×	×	Trigger input for external clock or cycle-by-cycle current management
×						Synchronization circuit to trigger the DAC

Timer type	Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/ compare channels	Complementary output	Max interface clock (MHz)	Max timer clock (MHz)
Advanced-control	TIM1, TIM8	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	Yes	84	168
General purpose	TIM2, TIM5	32-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	42	84
	TIM3, TIM4	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	42	84
	TIM9	16-bit	Up	Any integer between 1 and 65536	No	2	No	84	168
	TIM10, TIM11	16-bit	Up	Any integer between 1 and 65536	No	1	No	84	168
	TIM12	16-bit	Up	Any integer between 1 and 65536	No	2	No	42	84
	TIM13, TIM14	16-bit	Up	Any integer between 1 and 65536	No	1	No	42	84
Basic	TIM6, TIM7	16-bit	Up	Any integer between 1 and 65536	Yes	0	No	42	84

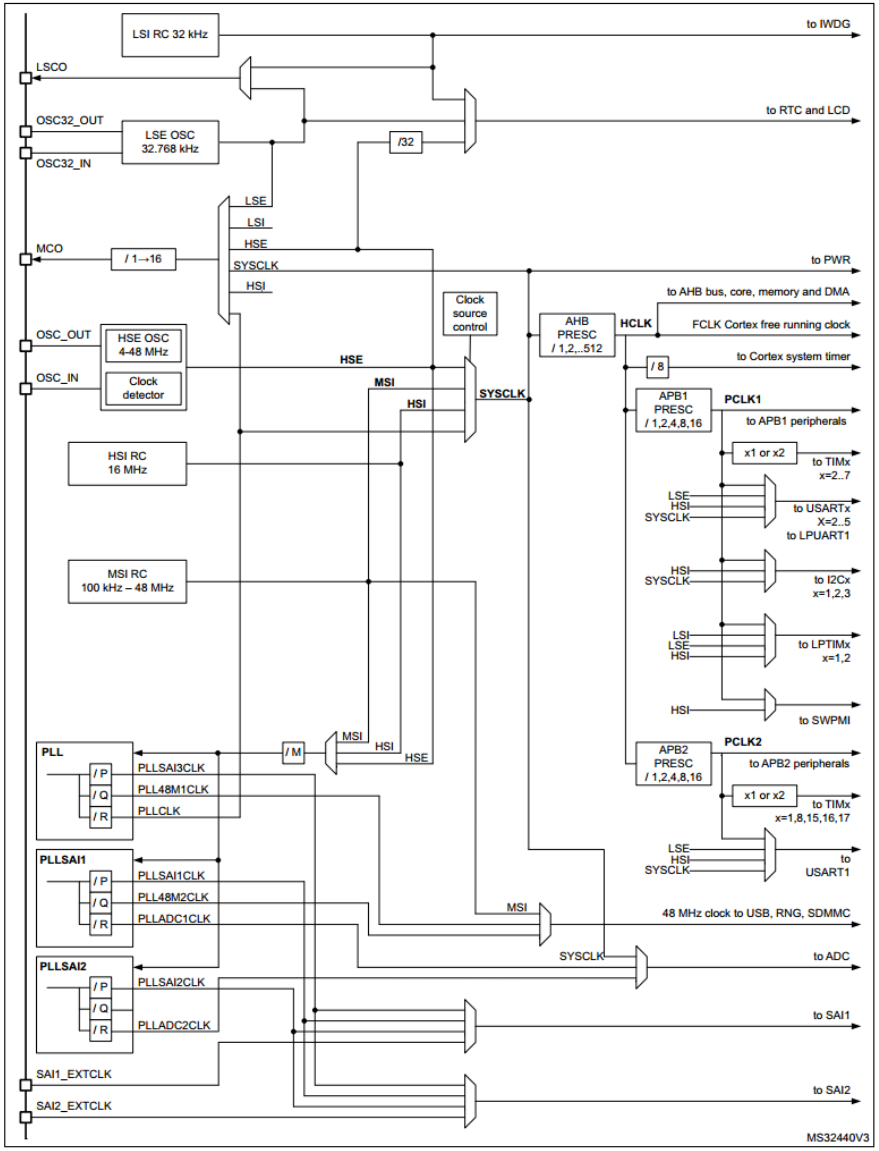
System Clock—Clock tree

Four different clock sources can be used to drive the system clock (SYSCLK)

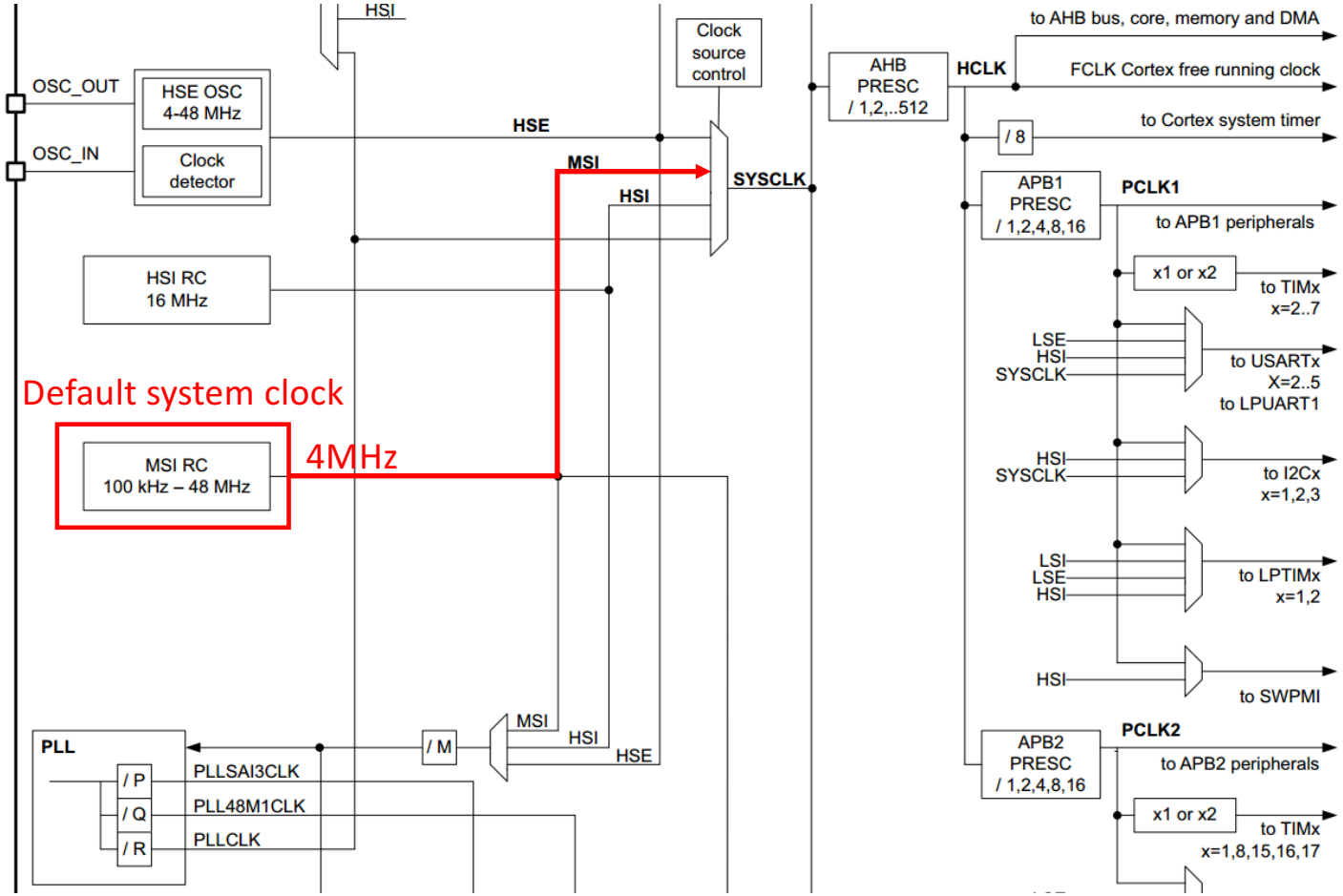
- HSI16 (high speed internal) 16 MHz RC oscillator clock
- MSI (multispeed internal) RC oscillator clock
- HSE (high speed external) oscillator clock, from 4 to 48 MHz
- PLL clock

The MSI is used as system clock source after startup from Reset, configured at 4 MHz

Figure 12. Clock tree



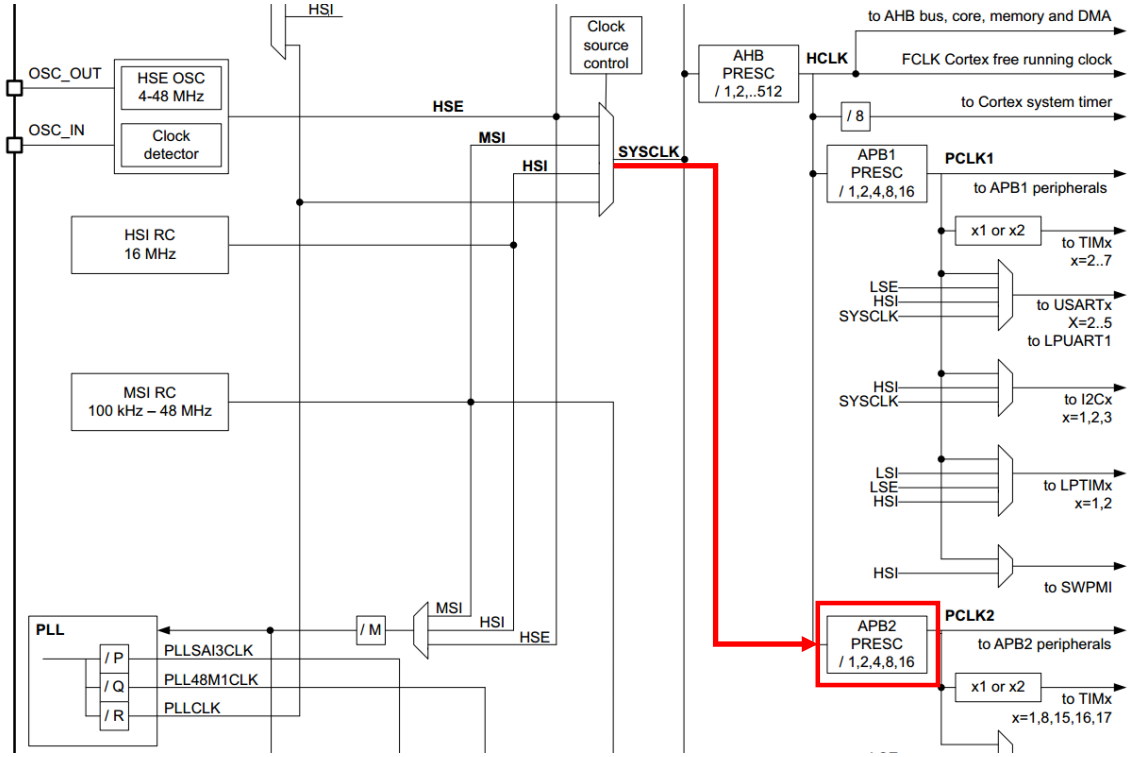
System Clock—Clock tree



Lab 1: Enable AHB2 Clock

```
.text
.global main
.equ RCC_AHB2ENR, 0x4002104C


main:
//Enable AHB2 clock
movs r0, #0x1
ldr r1, =RCC_AHB2ENR
str r0, [r1]
```

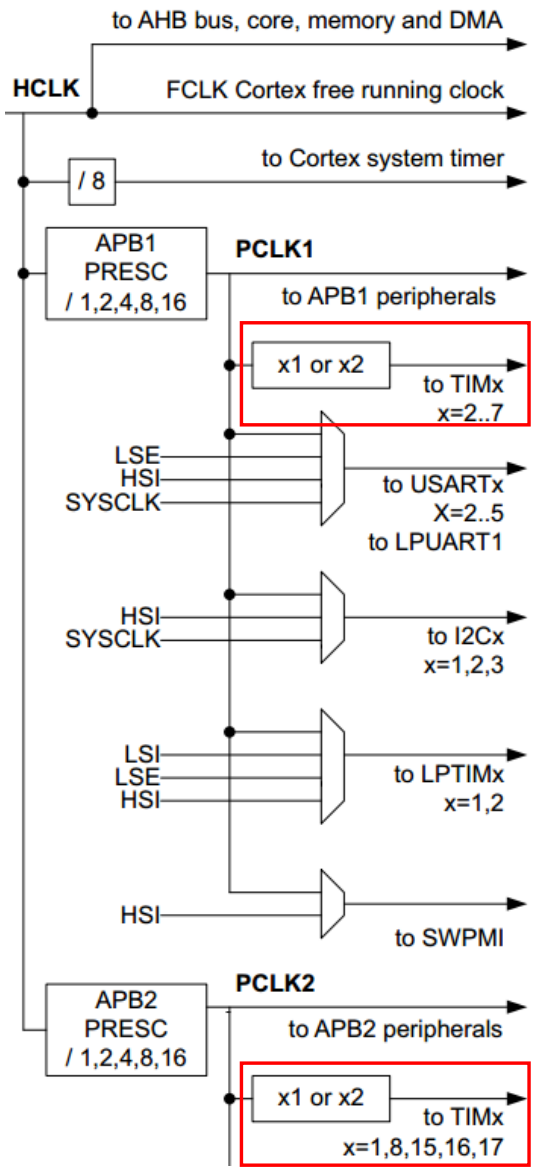


Timer

- The timer clock frequencies are automatically defined by hardware. There are two cases:

1. If the APB pre-scaler equals 1, the timer clock frequencies are set to the same frequency as that of the APB domain.

2. Otherwise, they are set to twice (×2) the frequency of the APB domain



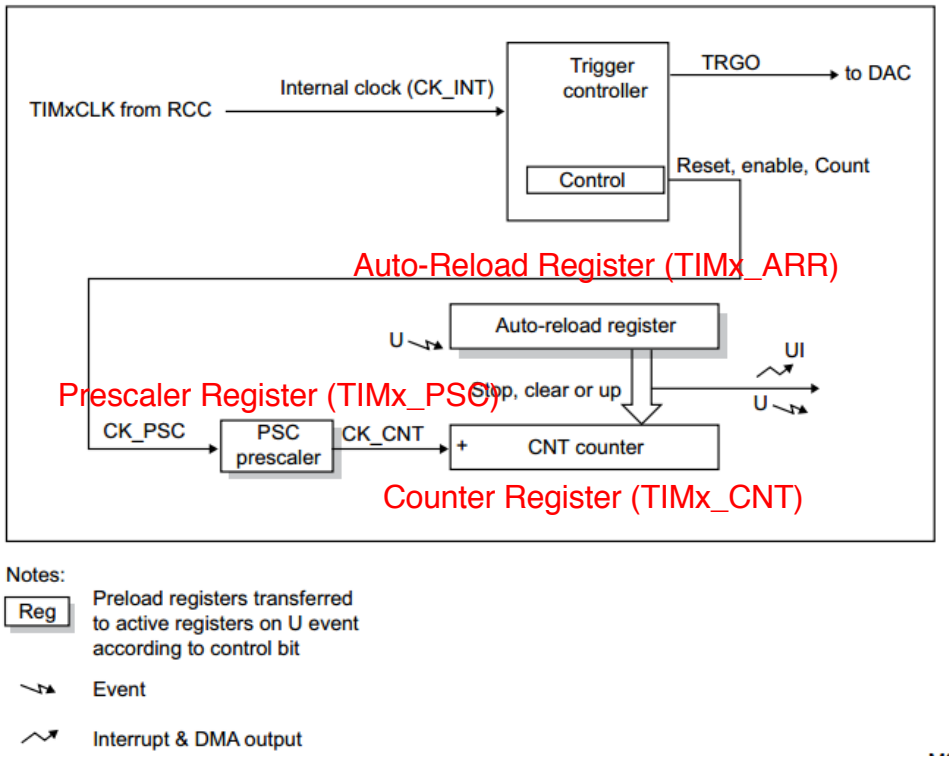
Basic Timers

- TIM6 and TIM7
- 16-bit auto-reload counter driven by a programmable pre-scaler
- Generic timers for time-base generation
- The timers are internally connected to the DAC and are able to drive it

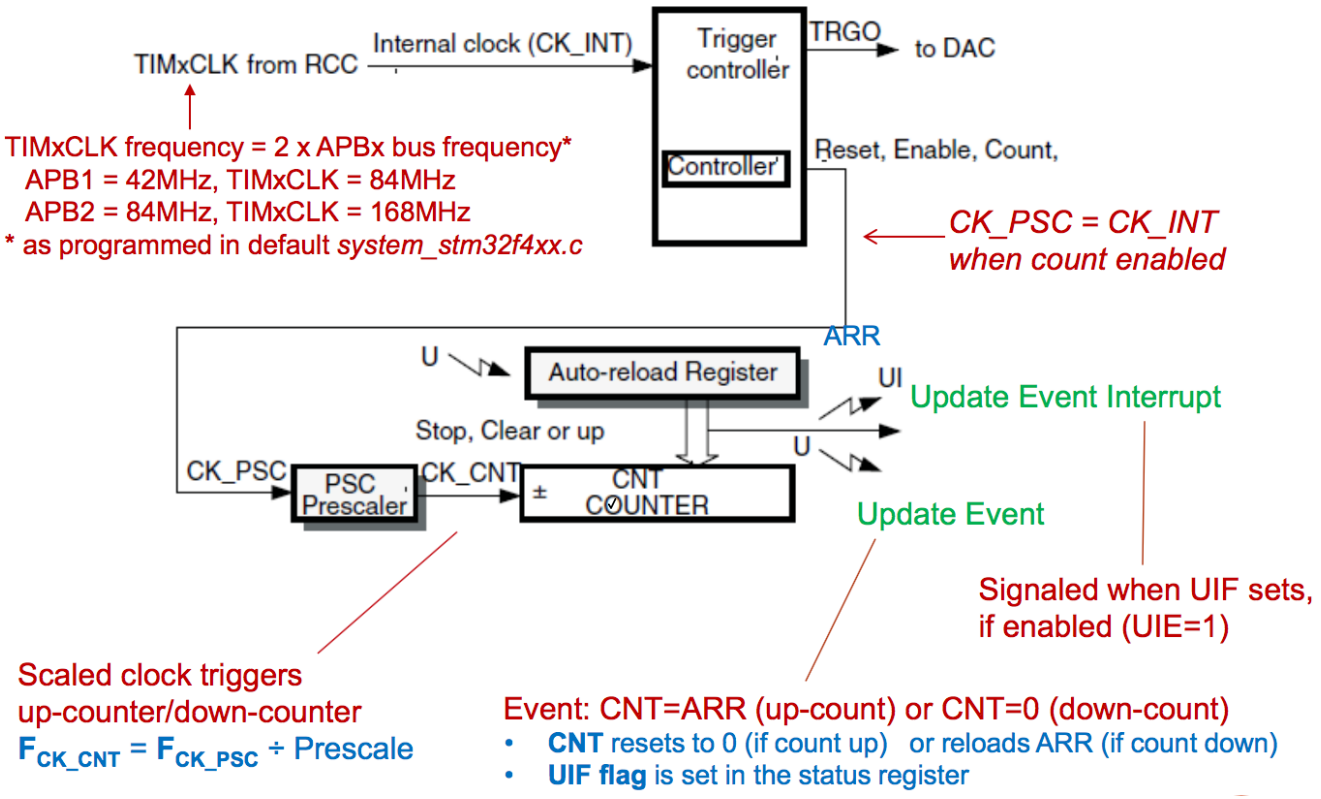
Basic timers –TIM6/TIM7

- 16-bit auto-reload up-counter
- 16-bit programmable pre-scaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Synchronization circuit to trigger the DAC
- Interrupt/DMA generation on the update event: counter overflow

Figure 330. Basic timer block diagram

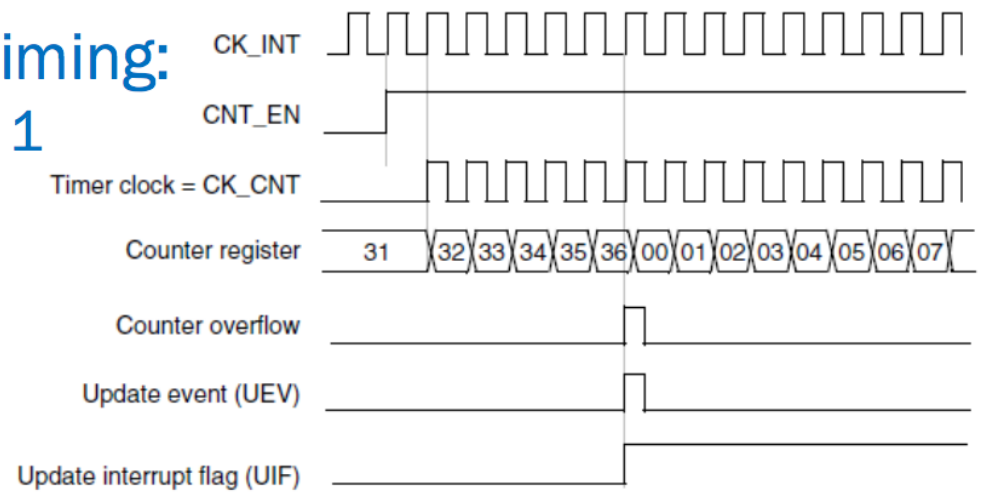


Basic timers –TIM6/TIM7

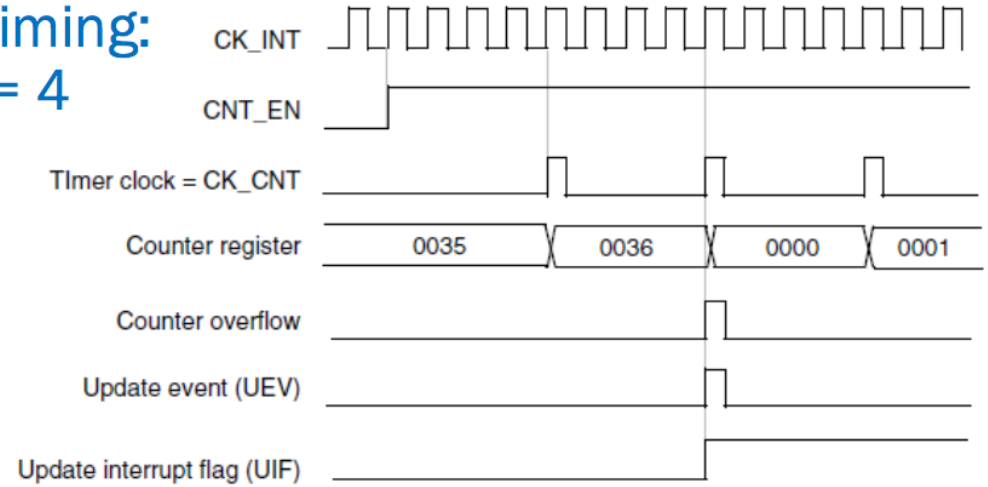


$$T_{EVENT} = \text{Prescale} \times \text{Count} \times T_{CK\_INT} = (PSC+1) \times (ARR+1) \times T_{CK\_INT}$$

Counter timing:
Prescale = 1
ARR = 36



Counter timing:
Prescale = 4
ARR = 36



Control register

29.4.1 TIM6/TIM7 control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	UIF RE-MAP	Res	Res	Res	ARPE	Res	Res	Res	OPM	URS	UDIS	CEN
				rw				rw				rw	rw	rw	rw

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered.
1: TIMx_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generates an update interrupt or DMA request if enabled. These events can be:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable UEV event generation.
0: UEV enabled. The Update (UEV) event is generated by one of the following events:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.
1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC). However 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.

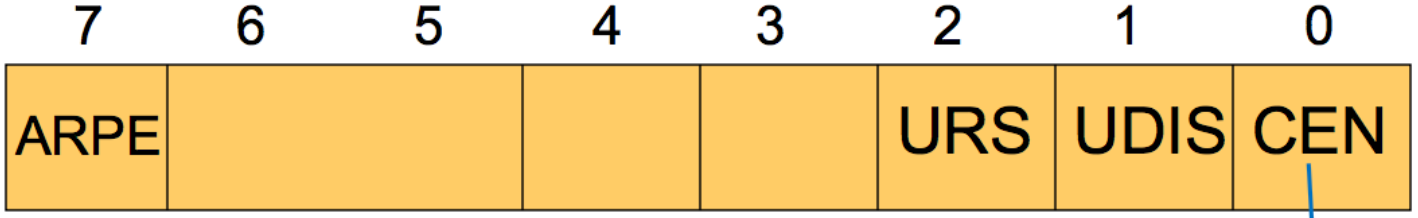
Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled

Note: Gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

CEN is cleared automatically in one-pulse mode, when an update event occurs.

Timer System Control Register 1

TIMx_CR1 (default = all 0's)



1 = enable, 0 = disable

CEN=1 to begin counting
(apply CK_INT to CK_PSC)

Examples:

```
TIM4->CR1 |= 0x01;      //Enable counting
TIM4->CR1 &= ~0x01;     //Disable counting
```

Other Options:

- UDIS = 0 enables update event to be generated (default)
- URS = 0 allows different events to generate update interrupt (default)
  1 restricts update interrupt to counter overflow/underflow
- ARPE = 0 allows new ARR value to take effect immediately (default)
  1 enables ARR buffer (new value held in buffer until next update event)

TIM2-4 and TIM9 include up/down direction and center-alignment controls

Status/Auto-reload register

29.4.8 TIM6/TIM7 auto-reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **ARR[15:0]**: Prescaler value

ARR is the value to be loaded into the actual auto-reload register.

Refer to *Section 29.3.1: Time-base unit on page 1014* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

29.4.4 TIM6/TIM7 status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	UIF
															rc_w0

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

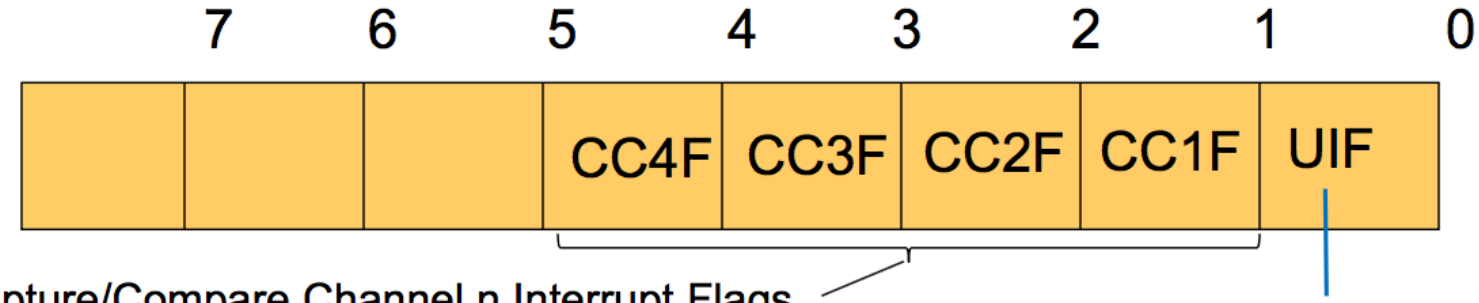
1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow or underflow regarding the repetition counter value and if UDIS = 0 in the TIMx_CR1 register.

– When CNT is reinitialized by software using the UG bit in the TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

Timer Status Register

TIMx_SR (reset value = all 0's)



Update Interrupt Flag

1 = update interrupt pending

0 = no update occurred

Set by hardware on update event

(CNT overflow)

Cleared by software

(write 0 to UIF bit)

Example: do actions if UIF=1

```
if (TIM4->SR & 0x01 == 0x01) {   //test UIF
    .. do some actions
   TIM4->SR &= ~0x01;  //clear UIF
}
```

Counter/Pre-scaler registers

29.4.6 TIM6/TIM7 counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved and read as 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

29.4.7 TIM6/TIM7 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency CK_CNT is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded into the active prescaler register at each update event.
(including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

Timer interrupt control register

29.4.3 TIM6/TIM7 DMA/Interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	UDE	Res	Res	Res	Res	Res	Res	Res	UIE
							rw								rw

Bits 15:9 Reserved, must be kept at reset value.

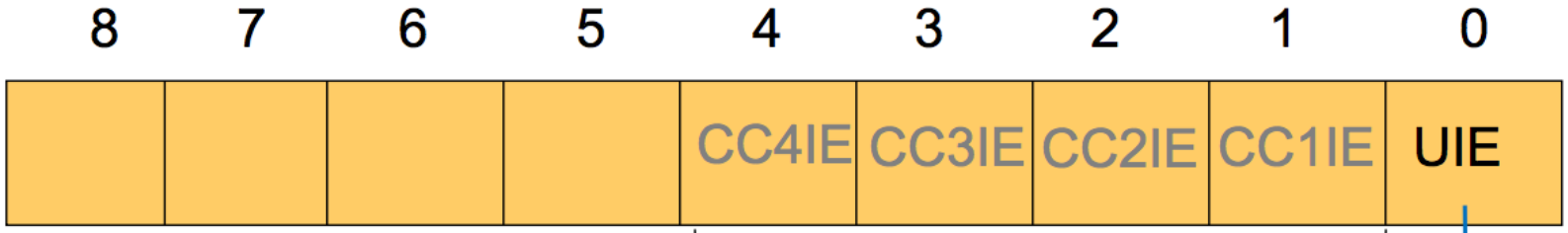
Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled.
1: Update DMA request enabled.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

Timer Interrupt Control Register

TIMx_DIER (default = all 0's)



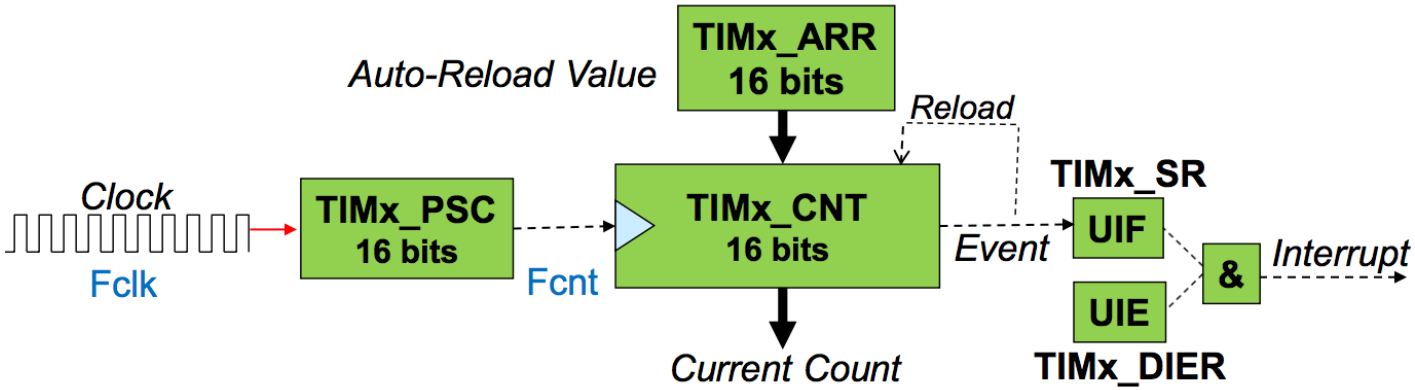
Capture/Compare n Interrupt Enable
(To be discussed later)

Update interrupt enable
1 = enable, 0 = disable
(interrupt if UIF=1 when UIE=1)

Examples:

```
TIM4->DIER |= 0x01;      //Enable interrupt
TIM4->DIER &= ~0x01;  //Disable interrupt
```

Timer



- Count-up mode overflow if **TIMx_CNT** reaches **TIMx_ARR**
  - On "overflow event", **UIF** flag is set and TIMx_CNT resets to 0.
  - If **UIE** = 1 (update interrupt enabled), interrupt signal is sent to NVIC
- **TIMx_PSC** prescale value multiplies input clock period (1 / Fclk) to produce counter clock period: $Tcnt = 1/Fcnt = (PSC+1)\times(1/Fclk)$
- Periodic time interval is the ARR (Auto-Reload Register) value times counter clock period:

$$Tout = (ARR+1)\times Tcnt = (ARR+1)\times(PSC+1)\times(1/Fclk)$$

Example: For 1 second time period, given Fclk = 16MHz:

$$Tout = (10000 \times 1600) \div 16000000 = 1 \text{ second}$$

Set ARR = 9999 and PSC = 1599 *(other combinations can also be used)*

Timer Example

```
Int main(){
    RCC->AHB2ENR |= RCC_AHB2ENR_GPIOAEN;//enable GPIOA clock
    init_GPIO();//init PA5 as output
    RCC->APB2ENR |= RCC_APB1ENR1_TIM6EN;//timer1 clock enable
    TIM6->PSC = (uint32_t)999;//Prescalser
    TIM6->ARR = (uint32_t)3999;//Reload value
    TIM6->EGR = TIM_EGR_UG;//Reinitialize the counter. CNT takes the auto-reload value.
    TIM6->CR1 |= TIM_CR1_CEN;//start timer
    while(1){
        int timerValue = TIM6->CNT;
        if (timerValue < 2000)
            GPIOA->BRR = (1<<5);
        if (timerValue >= 2000)
            GPIOA->BSRR = (1<<5);
    }
}
```

Timer clock source = 4MHz
➔4000000/(999+1) = 4000
➔每1/4000秒CNT會減1

CNT從4000數到0剛好需時1秒

Counter Example

```
void TIM3_Configuration(void)
{ GPIO_InitTypeDef GPIO_InitStructure;
  TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
  /* GPIOC clock enable */
  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC, ENABLE);
  /* TIM3 clock enable */
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);
  /* GPIOA Configuration: TIM3 CH1 (PC6) */
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF; // Input/Output controlled by peripheral
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
  GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP; // Button to ground expectation
  GPIO_Init(GPIOC, &GPIO_InitStructure);
    /* Connect TIM3 pins to AF */
  GPIO_PinAFConfig(GPIOC, GPIO_PinSource6, GPIO_AF_TIM3);
  TIM_TimeBaseStructure.TIM_Period = 65535;
  TIM_TimeBaseStructure.TIM_Prescaler = 0;
  TIM_TimeBaseStructure.TIM_ClockDivision = 0;
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
  TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);
  TIM_TIxExternalClockConfig(TIM3, TIM_TIxExternalCLK1Source_TI1, TIM_ICPolarity_Rising, 0);
  TIM_Cmd(TIM3, ENABLE);
}
```

General-purpose timers—TIM2/TIM3/TIM4/TIM5

- They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare and PWM)
- 16-bit (TIM3, TIM4) or 32-bit (TIM2 and TIM5) up, down, up/down auto-reload counter
- 16-bit programmable pre-scaler
- Up to 4 independent channels for:
  - Input capture, Output compare, PWM generation(Edge- and Center-aligned modes), One-pulse mode output
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software internal/external trigger), Trigger event (counter start, stop, initialization or count by internal/external trigger), Input capture, Output compare

Figure 249. General-purpose timer block diagram

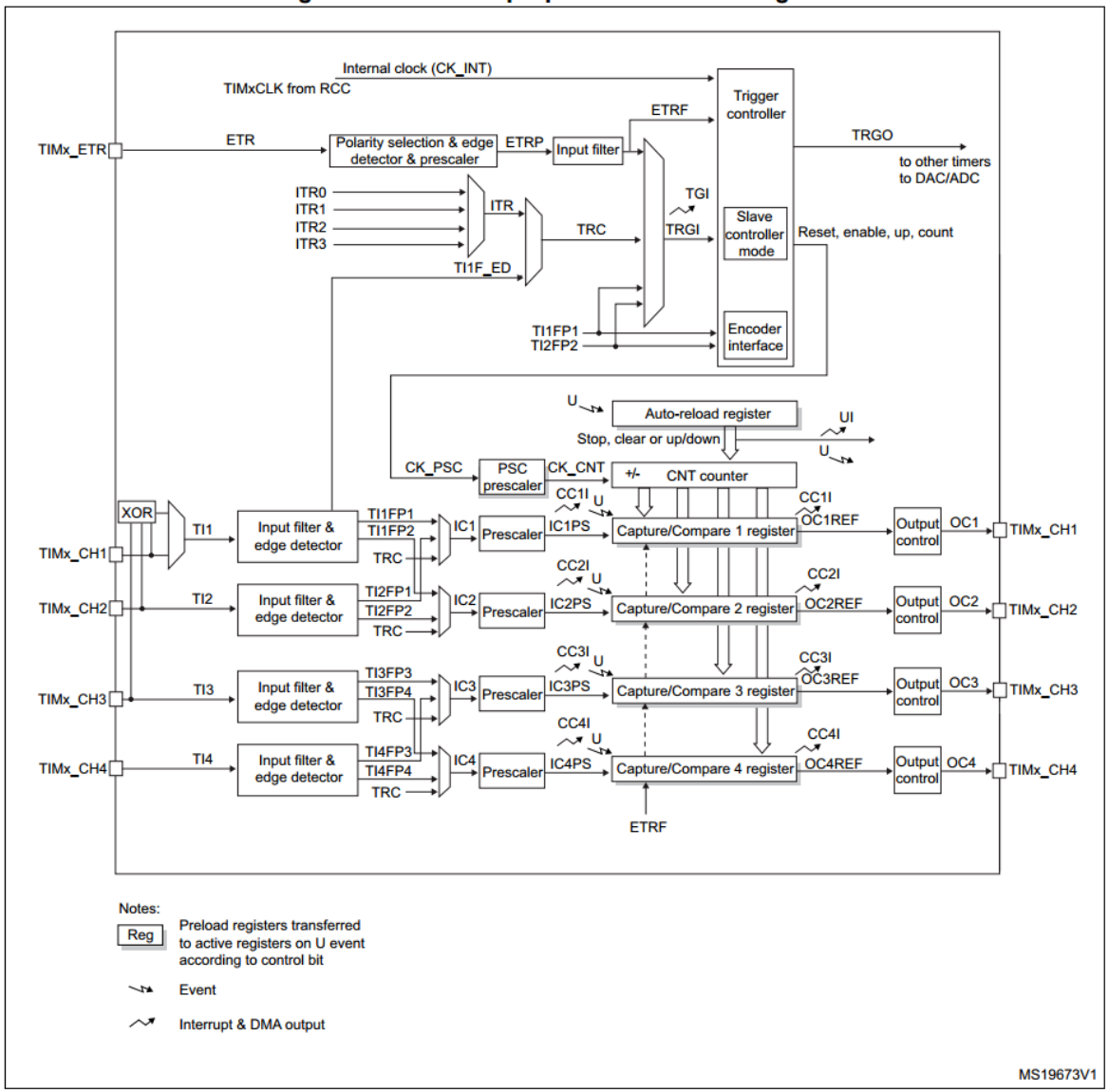


Figure 298. TIM15 block diagram

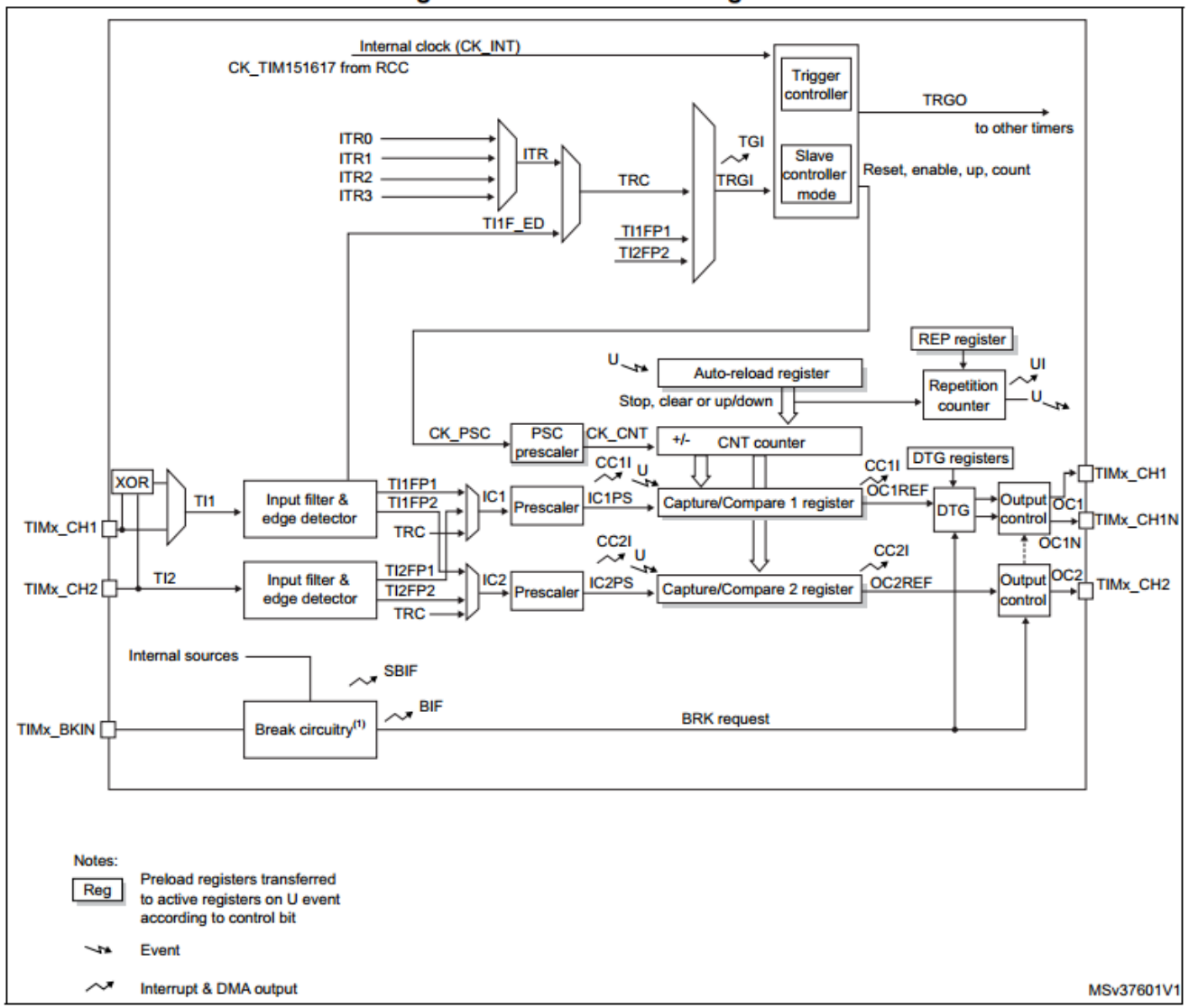
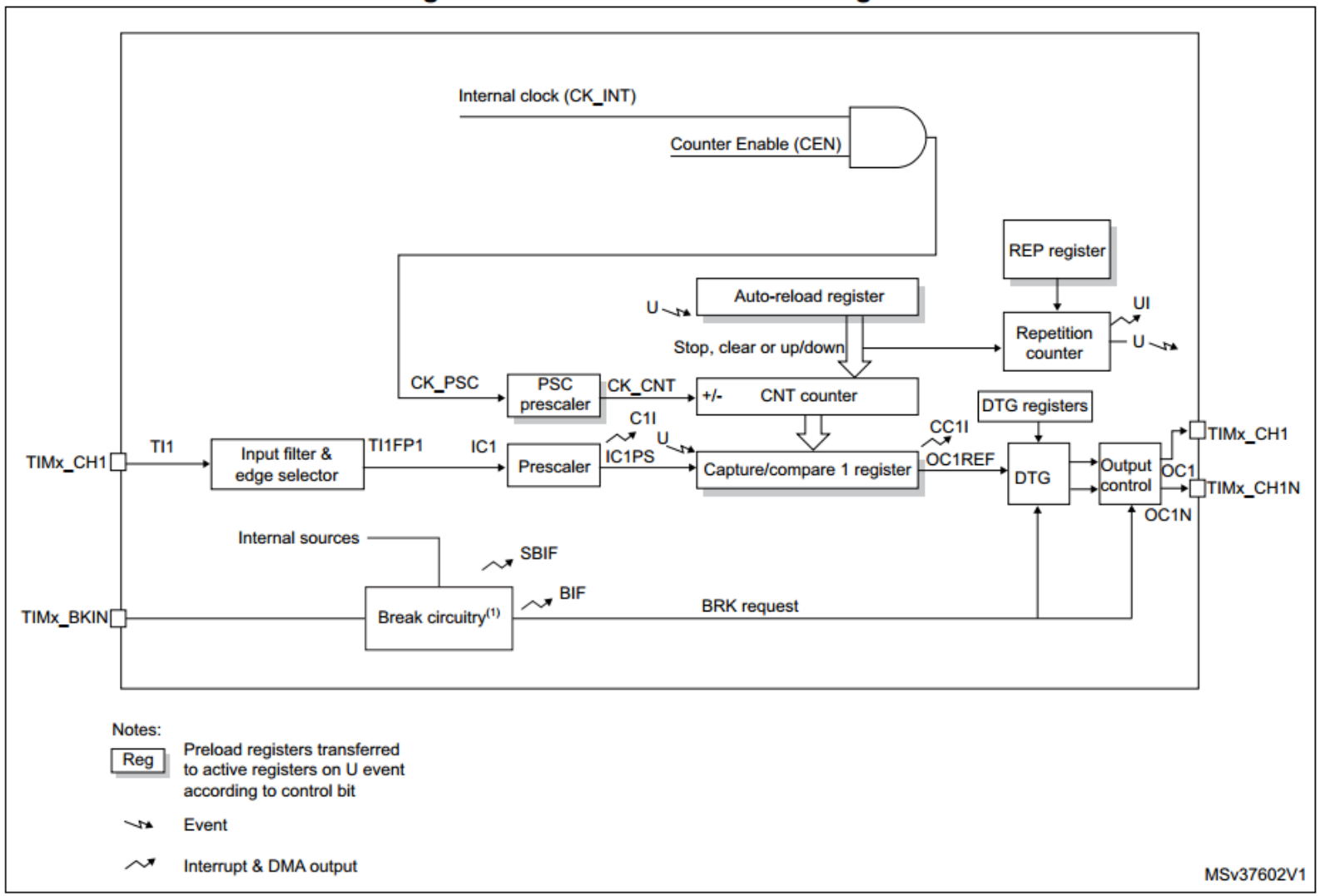
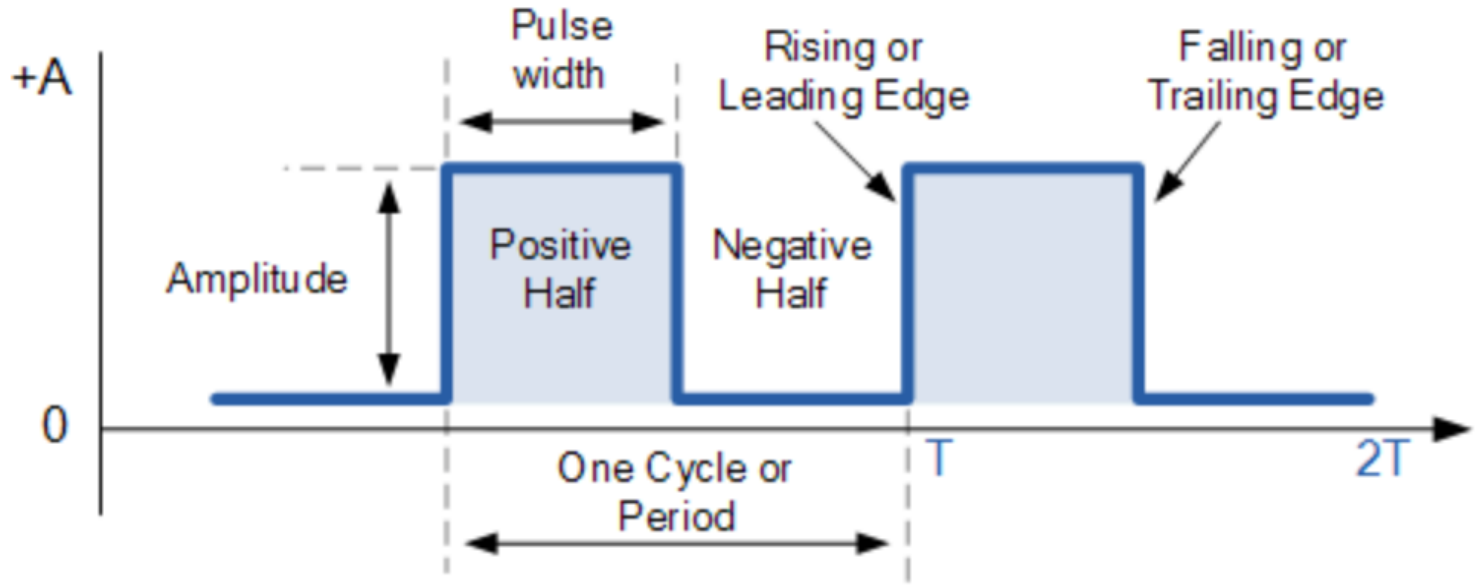


Figure 299. TIM16/TIM17 block diagram

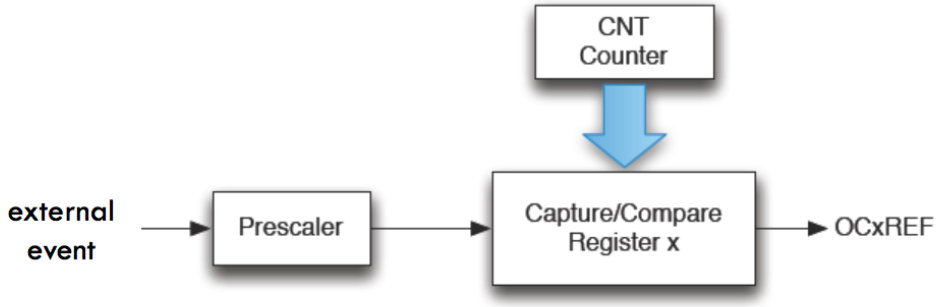


Input Capture

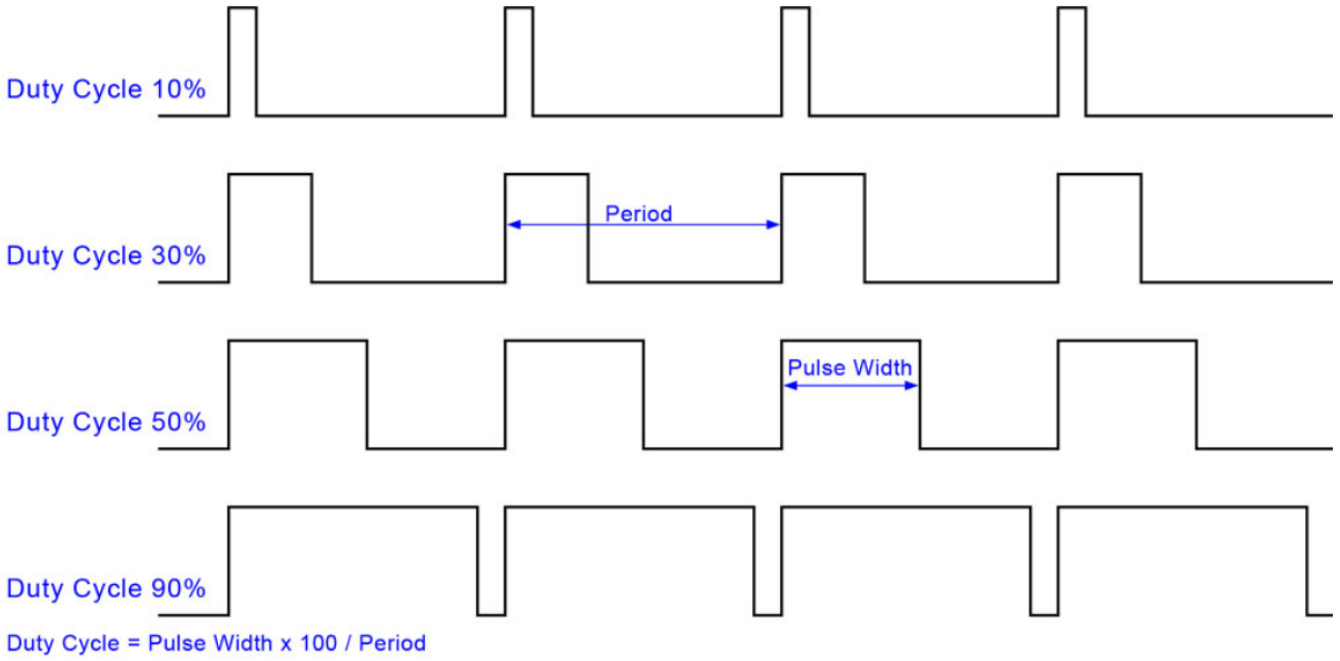


Output Compare

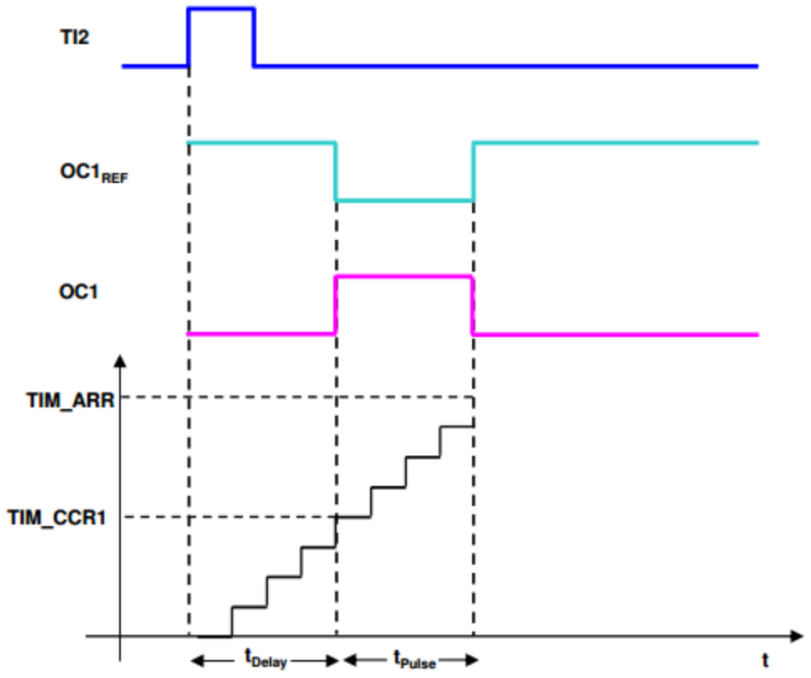
- Many timers extend the basic module with the addition of counter channels. The "x" refers to the channel.
- With this modest additional hardware, an output can be generated whenever the count register reaches a specific value **or** the counter register can be captured when a specific input event occurs (possibly a prescaled input clock).



PWM generation



One-pulse mode output



Advanced-control timers—TIM1/TIM8

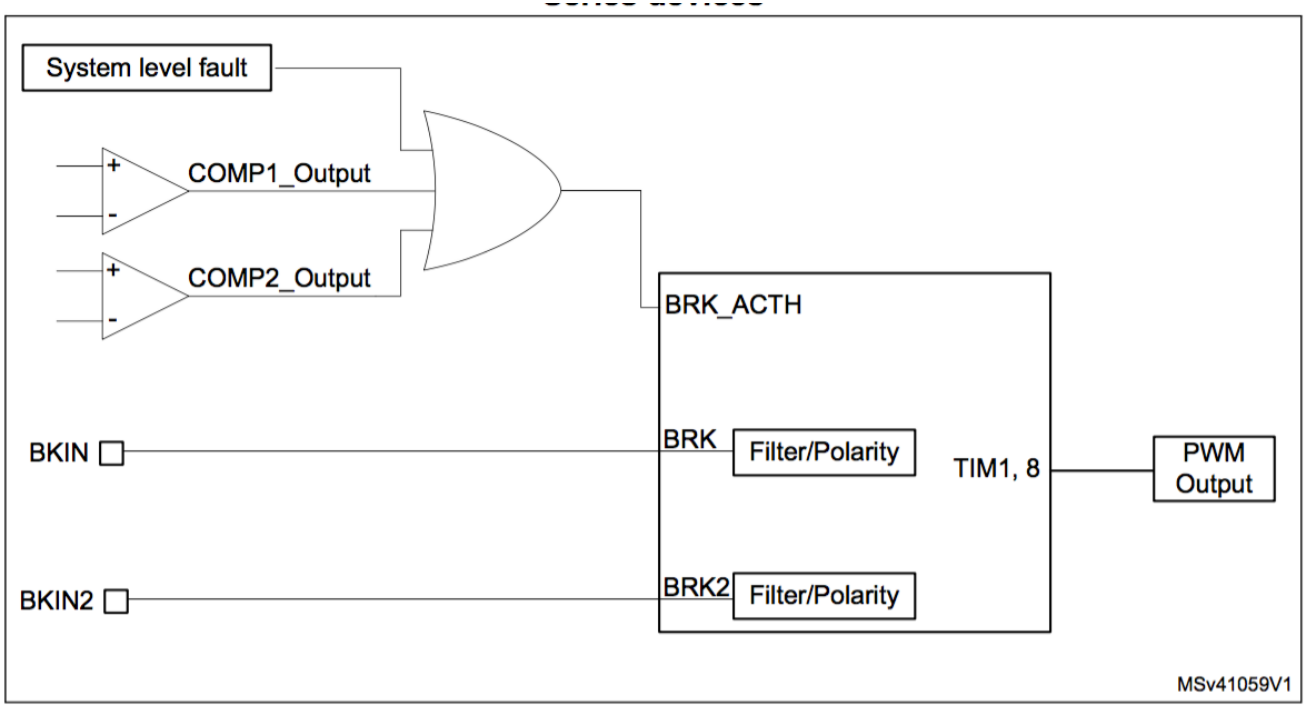
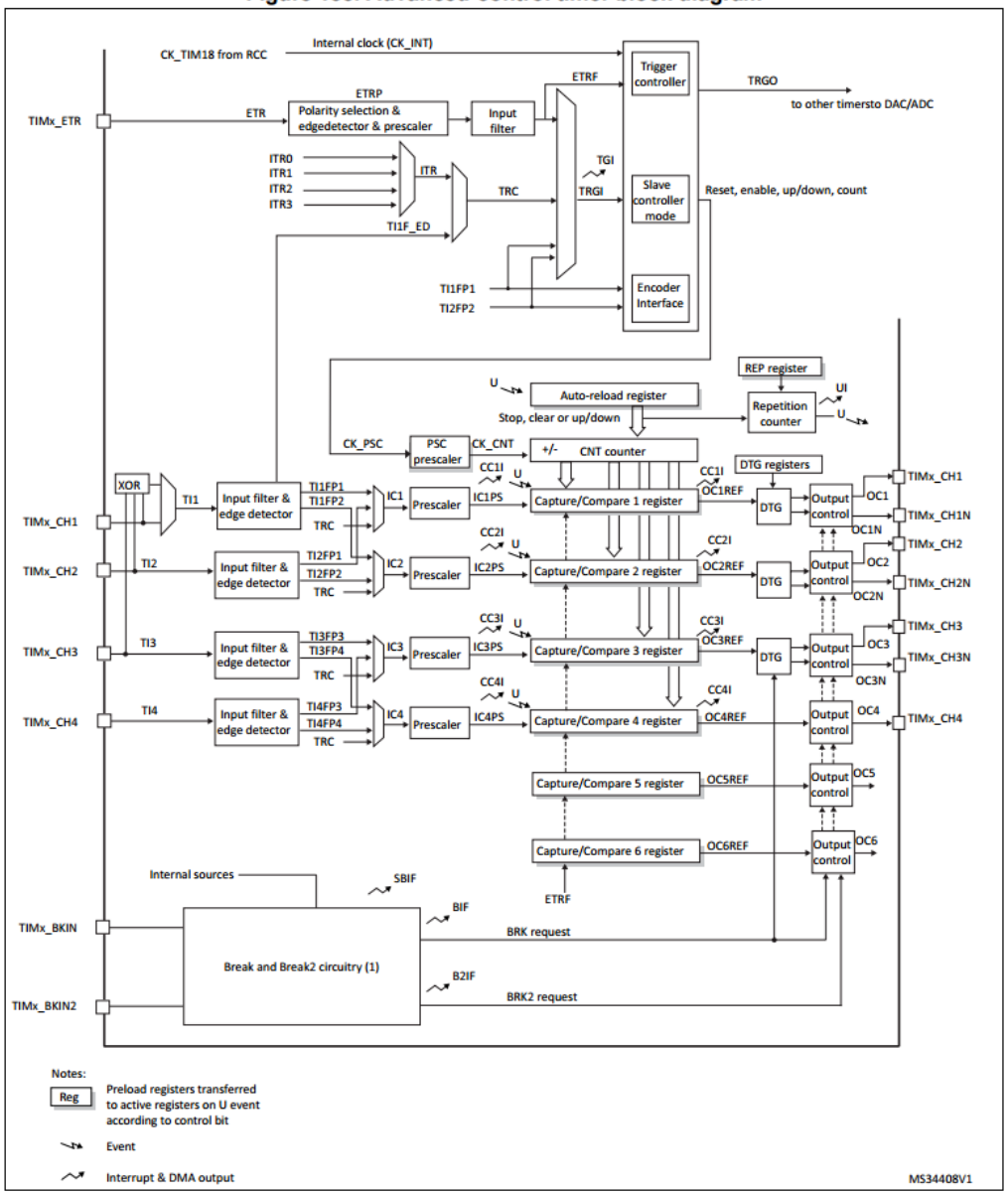


Figure 188. Advanced-control timer block diagram



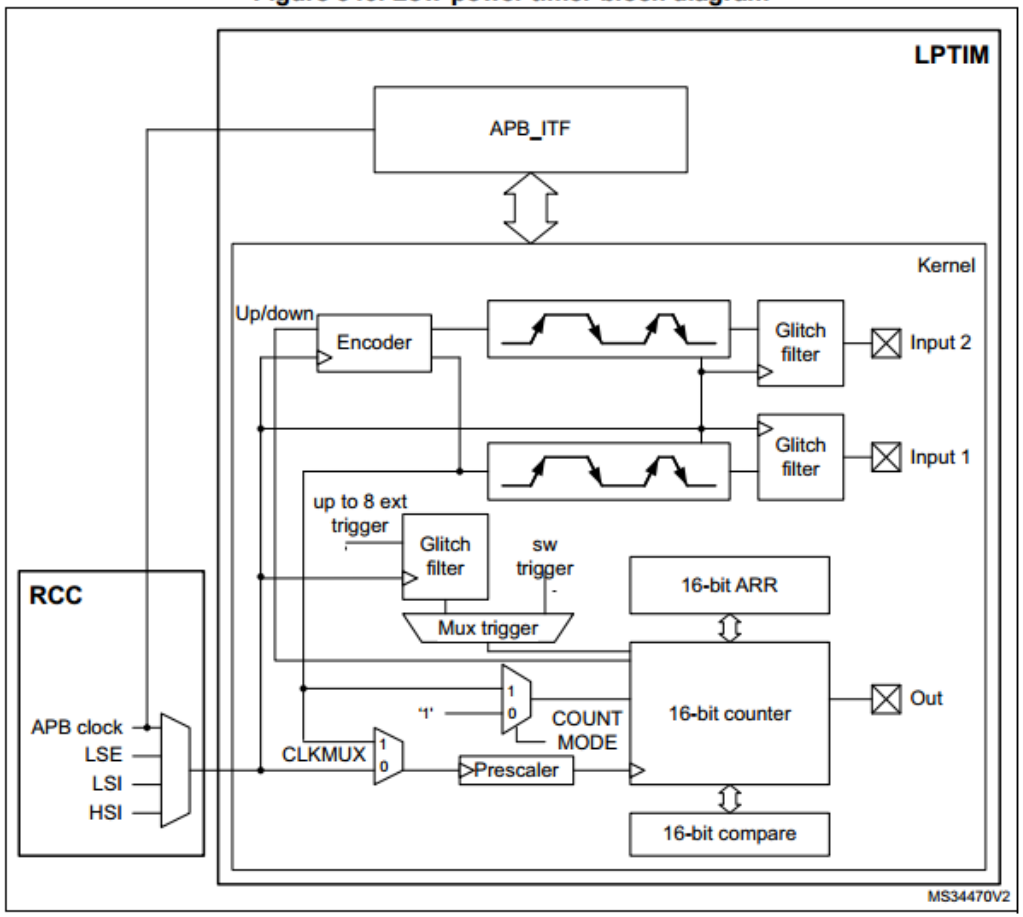
Low-power timer

- 16 bit upcounter
- 3-bit prescaler with 8 possible dividing factor (1,2,4,8,16,32,64,128)
- Selectable clock
  - Internal clock sources: LSE, LSI, HSI16 or APB clock
  - External clock source over ULPTIM input (working with no LP oscillator running, used by Pulse Counter application)
- 16 bit ARR autoreload register
- 16 bit compare register
- Continuous/one shot mode
- Selectable software/hardware input trigger
- Programmable Digital Glitch filter
- Configurable output: Pulse, PWM
- Configurable I/O polarity
- Encoder mode

Low-power timer

- The LPTIM is a 16-bit timer that benefits from the ultimate developments in power consumption reduction. Thanks to its diversity of clock sources, the LPTIM is able to keep running in all power modes except for Standby mode. Given its capability to run even with no internal clock source, the LPTIM can be used as a “Pulse Counter” which can be useful in some applications. Also, the LPTIM capability to wake up the system from low-power modes, makes it suitable to realize “Timeout functions” with extremely low power consumption.

Figure 340. Low-power timer block diagram



RCC memory address

Table 1. STM32L4x6 memory map and peripheral register boundary addresses (continued)

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
AHB1	0x4002 4000 - 0x4002 43FF	1 KB	TSC	*Section 23.6.11: TSC register map*
	0x4002 3400 - 0x4002 3FFF	1 KB	Reserved	-
	0x4002 3000 - 0x4002 33FF	1 KB	CRC	*Section 13.4.6: CRC register map*
	0x4002 2400 - 0x4002 2FFF	3 KB	Reserved	-
	0x4002 2000 - 0x4002 23FF	1 KB	FLASH registers	*Section 3.7.17: FLASH register map*
	0x4002 1400 - 0x4002 1FFF	3 KB	Reserved	-
	0x4002 1000 - 0x4002 13FF	1 KB	RCC	*Section 6.4.31: RCC register map*
	0x4002 0800 - 0x4002 0FFF	2 KB	Reserved	-
	0x4002 0400 - 0x4002 07FF	1 KB	DMA2	*Section 10.5.9: DMA register map*
	0x4002 0000 - 0x4002 03FF	1 KB	DMA1	*Section 10.5.9: DMA register map*

RCC registers—CR

6.4.1 Clock control register (RCC_CR)

Address offset: 0x00

Reset value: 0x0000 0063. HSEBYP is not affected by reset.

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	PLL SAI2 RDY	PLL SAI2 ON	PLL SAI1 RDY	PLL SAI1 ON	PLL RDY	PLLON	Res.	Res.	Res.	Res.	CSS ON	HSE BYP	HSE RDY	HSE ON
		r	rw	r	rw	r	rw					rs	rw	r	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSI ASFS	HSI RDY	HSI KERON	HSION	MSIRANGE[3:0]				MSI RGSEL	MSI PLLEN	MSI RDY	MSION
				rw	r	rw	rw	rw	rw	rw	rw	rs	rw	r	rw

- 1➔Clock ON, 0➔Clock OFF
- 1➔Clock Ready, 0➔Clock not ready

RCC_CR– MSI

Bit 0 **MSION:** MSI clock enable

This bit is set and cleared by software.

Cleared by hardware to stop the MSI oscillator when entering Stop, Standby or Shutdown mode.

Set by hardware to force the MSI oscillator ON when exiting Standby or Shutdown mode.

Set by hardware to force the MSI oscillator ON when STOPWUCK=0 when exiting from Stop modes, or in case of a failure of the HSE oscillator

Set by hardware when used directly or indirectly as system clock.

0: MSI oscillator OFF

Bit 1 **MSIRDY:** MSI clock ready flag

This bit is set by hardware to indicate that the MSI oscillator is stable.

0: MSI oscillator not ready

1: MSI oscillator ready

Note: Once the MSION bit is cleared, MSIRDY goes low after 6 MSI clock cycles.

Bit 2 **MSIPLLEN:** MSI clock PLL enable

Set and cleared by software to enable/ disable the PLL part of the MSI clock source.

MSIPLLEN must be enabled after LSE is enabled (LSEON enabled) and ready (LSERDY set by hardware).There is a hardware protection to avoid enabling MSIPLLEN if LSE is not ready.

This bit is cleared by hardware when LSE is disabled (LSEON = 0) or when the Clock Security System on LSE detects a LSE failure (refer to RCC_CSR register).

0: MSI PLL OFF

1: MSI PLL ON

Bit 3 **MSIRGSEL:** MSI clock range selection

Set by software to select the MSI clock range with MSIRANGE[3:0]. Write 0 has no effect.

After a standby or a reset MSIRGSEL is at 0 and the MSI range value is provided by MSISRANGE in CSR register.

0: MSI Range is provided by MSISRANGE[3:0] in RCC_CSR register

1: MSI Range is provided by MSIRANGE[3:0] in the RCC_CR register

Bits 7:4 **MSIRANGE[3:0]:** MSI clock ranges

These bits are configured by software to choose the frequency range of MSI when MSIRGSEL is set.12 frequency ranges are available:

0000: range 0 around 100 kHz

0001: range 1 around 200 kHz

0010: range 2 around 400 kHz

0011: range 3 around 800 kHz

0100: range 4 around 1M Hz

0101: range 5 around 2 MHz

0110: range 6 around 4 MHz (reset value)

0111: range 7 around 8 MHz

1000: range 8 around 16 MHz

1001: range 9 around 24 MHz

1010: range 10 around 32 MHz

1011: range 11 around 48 MHz

others: not allowed (hardware write protection)

Note: Warning: MSIRANGE can be modified when MSI is OFF (MSION=0) or when MSI is ready (MSIRDY=1). MSIRANGE must NOT be modified when MSI is ON and NOT ready (MSION=1 and MSIRDY=0)

RCC_CR– HSI

Bit 8 **HSION:** HSI16 clock enable

Set and cleared by software.

Cleared by hardware to stop the HSI16 oscillator when entering Stop, Standby or Shutdown mode.

Set by hardware to force the HSI16 oscillator ON when STOPWUCK=1 or HSIASFS = 1 when leaving Stop modes, or in case of failure of the HSE crystal oscillator.

This bit is set by hardware if the HSI16 is used directly or indirectly as system clock.

0: HSI16 oscillator OFF

1: HSI16 oscillator ON

Bit 9 **HSIKERON:** HSI16 always enable for peripheral kernels.

Set and cleared by software to force HSI16 ON even in Stop modes. The HSI16 can only feed USARTs and I^2Cs peripherals configured with HSI16 as kernel clock. Keeping the HSI16 ON in Stop mode allows to avoid slowing down the communication speed because of the HSI16 startup time. This bit has no effect on HSION value.

0: No effect on HSI16 oscillator.

1: HSI16 oscillator is forced ON even in Stop mode.

Bit 10 **HSIRDY:** HSI16 clock ready flag

Set by hardware to indicate that HSI16 oscillator is stable. This bit is set only when HSI16 is enabled by software by setting HSION.

0: HSI16 oscillator not ready

1: HSI16 oscillator ready

Note: Once the HSION bit is cleared, HSIRDY goes low after 6 HSI16 clock cycles.

Bit 11 **HSIASFS:** HSI16 automatic start from Stop

Set and cleared by software. When the system wakeup clock is MSI, this bit is used to wakeup the HSI16 is parallel of the system wakeup.

0: HSI16 oscillator is not enabled by hardware when exiting Stop mode with MSI as wakeup clock.

1: HSI16 oscillator is enabled by hardware when exiting Stop mode with MSI as wakeup clock.

RCC_CR– HSE

Bit 16 **HSEON:** HSE clock enable

Set and cleared by software.

Cleared by hardware to stop the HSE oscillator when entering Stop, Standby or Shutdown mode. This bit cannot be reset if the HSE oscillator is used directly or indirectly as the system clock.

0: HSE oscillator OFF

1: HSE oscillator ON

Bit 17 **HSERDY:** HSE clock ready flag

Set by hardware to indicate that the HSE oscillator is stable.

0: HSE oscillator not ready

1: HSE oscillator ready

Note: Once the HSEON bit is cleared, HSERDY goes low after 6 HSE clock cycles.

Bit 18 **HSEBYP:** HSE crystal oscillator bypass

Set and cleared by software to bypass the oscillator with an external clock. The external clock must be enabled with the HSEON bit set, to be used by the device. The HSEBYP bit can be written only if the HSE oscillator is disabled.

0: HSE crystal oscillator not bypassed

1: HSE crystal oscillator bypassed with external clock

Bit 19 **CSSON:** Clock security system enable

Set by software to enable the clock security system. When CSSON is set, the clock detector is enabled by hardware when the HSE oscillator is ready, and disabled by hardware if a HSE clock failure is detected. This bit is set only and is cleared by reset.

0: Clock security system OFF (clock detector OFF)

1: Clock security system ON (Clock detector ON if the HSE oscillator is stable, OFF if not).

RCC_CR– PLL

Bit 24 **PLLON:** Main PLL enable

Set and cleared by software to enable the main PLL.

Cleared by hardware when entering Stop, Standby or Shutdown mode. This bit cannot be reset if the PLL clock is used as the system clock.

0: PLL OFF

1: PLL ON

Bit 25 **PLLRDY:** Main PLL clock ready flag

Set by hardware to indicate that the main PLL is locked.

0: PLL unlocked

1: PLL locked

Bit 26 **PLLSAI1ON:** SAI1 PLL enable

Set and cleared by software to enable PLLSAI1.

Cleared by hardware when entering Stop, Standby or Shutdown mode.

0: PLLSAI1 OFF

1: PLLSAI1 ON

Bit 27 **PLLSAI1RDY:** SAI1 PLL clock ready flag

Set by hardware to indicate that the PLLSAI1 is locked.

0: PLLSAI1 unlocked

1: PLLSAI1 locked

Bit 28 **PLLSAI2ON:** SAI2 PLL enable

Set and cleared by software to enable PLLSAI2.

Cleared by hardware when entering Stop, Standby or Shutdown mod

0: PLLSAI2 OFF

1: PLLSAI2 ON

Bit 29 **PLLSAI2RDY:** SAI2 PLL clock ready flag

Set by hardware to indicate that the PLLSAI2 is locked.

0: PLLSAI2 unlocked

1: PLLSAI2 locked

RCC registers—CFGR

6.4.3 Clock configuration register (RCC_CFGR)

Address offset: 0x08

Reset value: 0x0000 0000

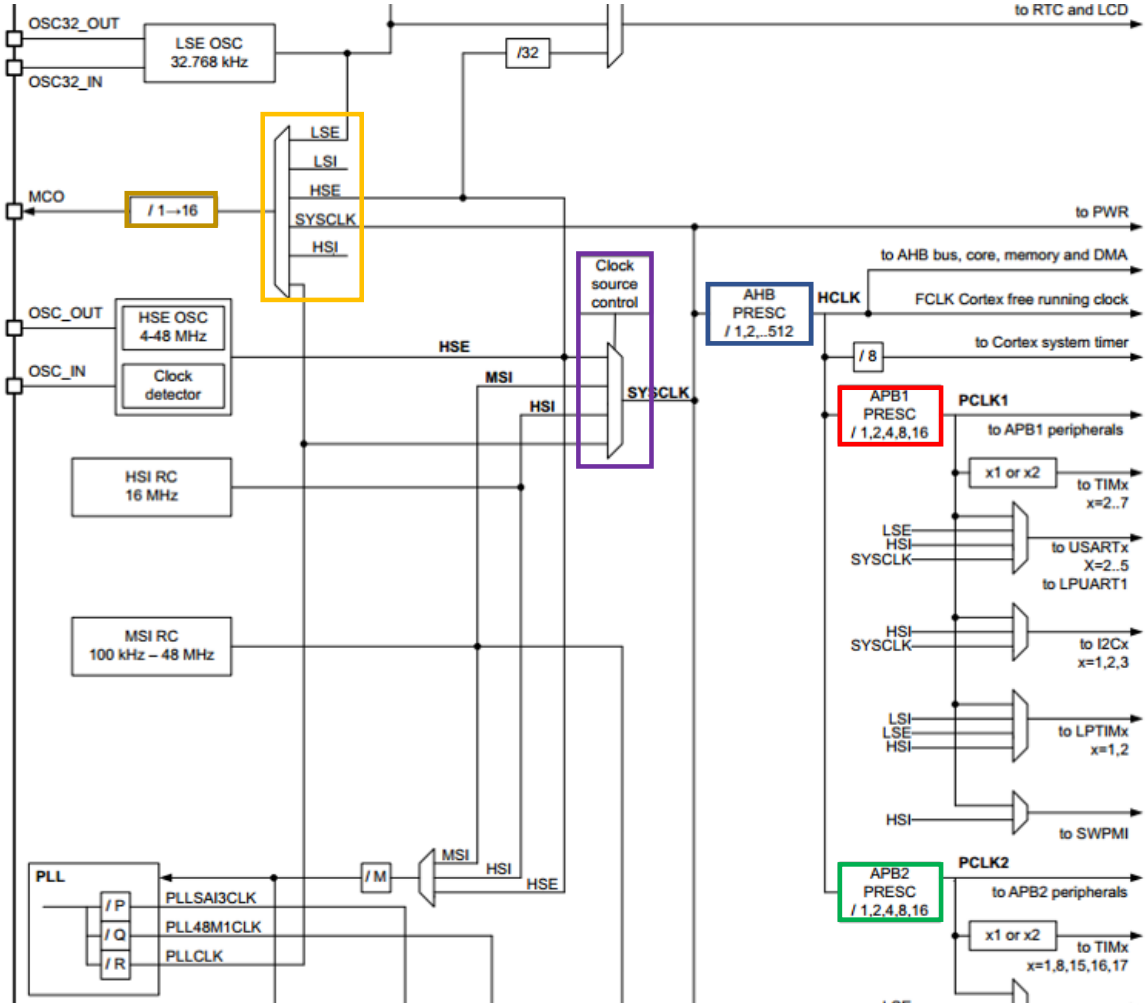
Access: 0 ≤ wait state ≤ 2, word, half-word and byte access

1 or 2 wait states inserted only if the access occurs during clock source switch.

From 0 to 15 wait states inserted if the access occurs when the APB or AHB prescalers values update is on going.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	MCOPRE[2:0]			Res.	MCOSEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw	rw	rw		rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP WUCK	Res.	PPRE2[2:0]			PPRE1[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw

RCC_CFGR



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	MCOPRE[2:0]			Res.	MCOSEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw	rw	rw		rw	rw	rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP WUCK	Res.	PPRE2[2:0]			PPRE1[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw

RCC_CFGR

Bits 1:0 **SW[1:0]:** System clock switch

Set and cleared by software to select system clock source (SYSCLK).
Configured by HW to force MSI oscillator selection when exiting Standby or Shutdown mode.
Configured by HW to force MSI or HSI16 oscillator selection when exiting Stop mode or in case of failure of the HSE oscillator, depending on STOPWUCK value.
00: MSI selected as system clock
01: HSI16 selected as system clock
10: HSE selected as system clock
11: PLL selected as system clock

Bits 3:2 **SWS[1:0]:** System clock switch status

Set and cleared by hardware to indicate which clock source is used as system clock.
00: MSI oscillator used as system clock
01: HSI16 oscillator used as system clock
10: HSE used as system clock
11: PLL used as system clock

Bits 7:4 **HPRE[3:0]:** AHB prescaler

Set and cleared by software to control the division factor of the AHB clock.

Caution: Depending on the device voltage range, the software has to set correctly these bits to ensure that the system frequency does not exceed the maximum allowed frequency (for more details please refer to *Section 5.1.7: Dynamic voltage scaling management*). After a write operation to these bits and before decreasing the voltage range, this register must be read to be sure that the new value has been taken into account.

0xxx: SYSCLK not divided
1000: SYSCLK divided by 2
1001: SYSCLK divided by 4
1010: SYSCLK divided by 8
1011: SYSCLK divided by 16
1100: SYSCLK divided by 64
1101: SYSCLK divided by 128
1110: SYSCLK divided by 256
1111: SYSCLK divided by 512

Bits 10:8 **PPRE1[2:0]:**APB low-speed prescaler (APB1)

Set and cleared by software to control the division factor of the APB1 clock (PCLK1).
0xx: HCLK not divided
100: HCLK divided by 2
101: HCLK divided by 4
110: HCLK divided by 8
111: HCLK divided by 16

Bits 13:11 **PPRE2[2:0]:** APB high-speed prescaler (APB2)

Set and cleared by software to control the division factor of the APB2 clock (PCLK2).
0xx: HCLK not divided
100: HCLK divided by 2
101: HCLK divided by 4
110: HCLK divided by 8
111: HCLK divided by 16

Bit 15 **STOPWUCK:** Wakeup from Stop and CSS backup clock selection

Set and cleared by software to select the system clock used when exiting Stop mode.
The selected clock is also used as emergency clock for the Clock Security System on HSE.
Warning: STOPWUCK must not be modified when the Clock Security System is enabled by HSECSSON in RCC_CR register and the system clock is HSE (SWS="10") or a switch on HSE is requested (SW="10").
0: MSI oscillator selected as wakeup from stop clock and CSS backup clock.
1: HSI16 oscillator selected as wakeup from stop clock and CSS backup clock

Bits 26:24 **MCOSEL[2:0]:** Microcontroller clock output

Set and cleared by software.
000: MCO output disabled, no clock on MCO
001: SYSCLK system clock selected
010: MSI clock selected.
011: HSI16 clock selected.
100: HSE clock selected
101: Main PLL clock selected
110: LSI clock selected
111: LSE clock selected

Note: This clock output may have some truncated cycles at startup or during MCO clock source switching.

Bits 30:28 **MCOPRE[2:0]:** Microcontroller clock output prescaler

These bits are set and cleared by software.
It is highly recommended to change this prescaler before MCO output is enabled.
000: MCO is divided by 1
001: MCO is divided by 2
010: MCO is divided by 4
011: MCO is divided by 8
100: MCO is divided by 16
Others: not allowed

Flash Read Access Latency

To correctly read data from Flash memory, the number of wait states (LATENCY) must be correctly programmed in the *Flash access control register (FLASH_ACR)* according to the frequency of the CPU clock (HCLK) and the internal voltage range of the device V_{CORE}. Refer to *Section 5.1.7: Dynamic voltage scaling management*. *Table 8* shows the correspondence between wait states and CPU clock frequency.

Table 8. Number of wait states according to CPU clock (HCLK) frequency

Wait states (WS) (LATENCY)	HCLK (MHz)	
	V_{CORE} Range 1	V_{CORE} Range 2
0 WS (1 CPU cycles)	≤ 16	≤ 6
1 WS (2 CPU cycles)	≤ 32	≤ 12
2 WS (3 CPU cycles)	≤ 48	≤ 18
3 WS (4 CPU cycles)	≤ 64	≤ 26
4 WS (5 CPU cycles)	≤ 80	≤ 26

After reset, the CPU clock frequency is 4 MHz and 0 wait state (WS) is configured in the FLASH_ACR register.

Flash Read Access Latency

When changing the CPU frequency, the following software sequences must be applied in order to tune the number of wait states needed to access the Flash memory

Increasing the CPU frequency:

1. Program the new number of wait states to the LATENCY bits in the *Flash access control register (FLASH_ACR)*.
2. Check that the new number of wait states is taken into account to access the Flash memory by reading the FLASH_ACR register.
3. Modify the CPU clock source by writing the SW bits in the RCC_CFGR register.
4. If needed, modify the CPU clock prescaler by writing the HPRE bits in RCC_CFGR.
5. Check that the new CPU clock source or/and the new CPU clock prescaler value is/are taken into account by reading the clock source status (SWS bits) or/and the AHB prescaler value (HPRE bits), respectively, in the RCC_CFGR register.

Decreasing the CPU frequency:

1. Modify the CPU clock source by writing the SW bits in the RCC_CFGR register.
2. If needed, modify the CPU clock prescaler by writing the HPRE bits in RCC_CFGR.
3. Check that the new CPU clock source or/and the new CPU clock prescaler value is/are taken into account by reading the clock source status (SWS bits) or/and the AHB prescaler value (HPRE bits), respectively, in the RCC_CFGR register.
4. Program the new number of wait states to the LATENCY bits in *Flash access control register (FLASH_ACR)*.
5. Check that the new number of wait states is used to access the Flash memory by reading the FLASH_ACR register.

FLASH ACR

3.7.1 Flash access control register (FLASH_ACR)

Address offset: 0x00

Reset value: 0x0000 0600

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SLEEP_PD	RUN_PD	DCRST	ICRST	DCEN	ICEN	PRFTEN	Res.	Res.	Res.	Res.	Res.	LATENCY[2:0]		
	rw	rw	rw	rw	rw	rw	rw						rw	rw	rw

Bits 2:0 **LATENCY[2:0]**: Latency

These bits represent the ratio of the SYSCLK (system clock) period to the Flash access time.

000: Zero wait state

001: One wait state

010: Two wait sates

011: Three wait sates

100: Four wait sates

others: reserved

RCC_APB1ENR1

Enable timer

6.4.19 APB1 peripheral clock enable register 1 (RCC_APB1ENR1)

Address: 0x58

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM1 EN	OPAMP EN	DAC1 EN	PWR EN	Res.	Res.	CAN1 EN	Res.	I2C3 EN	I2C2 EN	I2C1 EN	UART5 EN	UART4 EN	USART3 EN	USART2 EN	Res.
rw	rw	rw	rw			rw		rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 EN	SPI2 EN	Res.	Res.	WWD GEN	Res.	LCD EN	Res.	Res.	Res.	TIM7 EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2 EN
rw	rw			rs		rw				rw	rw	rw	rw	rw	rw

Bit 5 **TIM7EN:** TIM7 timer clock enable
Set and cleared by software.
0: TIM7 clock disabled
1: TIM7 clock enabled

Bit 4 **TIM6EN:** TIM6 timer clock enable
Set and cleared by software.
0: TIM6 clock disabled
1: TIM6 clock enabled

Bit 3 **TIM5EN:** TIM5 timer clock enable
Set and cleared by software.
0: TIM5 clock disabled
1: TIM5 clock enabled

Bit 2 **TIM4EN:** TIM4 timer clock enable
Set and cleared by software.
0: TIM4 clock disabled
1: TIM4 clock enabled

Bit 1 **TIM3EN:** TIM3 timer clock enable
Set and cleared by software.
0: TIM3 clock disabled
1: TIM3 clock enabled

Bit 0 **TIM2EN:** TIM2 timer clock enable
Set and cleared by software.
0: TIM2 clock disabled
1: TIM2 clock enabled

RCC_APB2ENR

6.4.21 APB2 peripheral clock enable register (RCC_APB2ENR)

Address: 0x60

Reset value: 0x0000 0000

Access: word, half-word and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFSDM1EN	Res.	SAI2EN	SAI1EN	Res.	Res.	TIM17EN	TIM16EN	TIM15EN
							rw		rw	rw			rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1EN	TIM8EN	SPI1EN	TIM1EN	SDMMC1EN	Res.	Res.	FWEN	Res.	Res.	Res.	Res.	Res.	Res.	SYSCFGEN
	rw	rw	rw	rw	rw			rs							rw

Bit 22 **SAI2EN:** SAI2 clock enable
Set and cleared by software.
0: SAI2 clock disabled
1: SAI2 clock enabled

Bit 21 **SAI1EN:** SAI1 clock enable
Set and cleared by software.
0: SAI1 clock disabled
1: SAI1 clock enabled

Bit 18 **TIM17EN:** TIM17 timer clock enable
Set and cleared by software.
0: TIM17 timer clock disabled
1: TIM17 timer clock enabled

Bit 17 **TIM16EN:** TIM16 timer clock enable
Set and cleared by software.
0: TIM16 timer clock disabled
1: TIM16 timer clock enabled

Bit 16 **TIM15EN:** TIM15 timer clock enable
Set and cleared by software.
0: TIM15 timer clock disabled
1: TIM15 timer clock enabled

Bit 13 **TIM8EN:** TIM8 timer clock enable
Set and cleared by software.
0: TIM8 timer clock disabled
1: TIM8 timer clock enabled

Bit 11 **TIM1EN:** TIM1 timer clock enable
Set and cleared by software.
0: TIM1 timer clock disabled
1: TIM1P timer clock enabled

Timer memory map address

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
APB2	0x4001 6400 - 0x4001 FFFF	39 KB	Reserved	-
	0x4001 6000 - 0x4000 63FF	1 KB	DFSDM1	*Section 21.8: DFSDM register map*
	0x4001 5C00 - 0x4000 5FFF	1 KB	Reserved	-
	0x4001 5800 - 0x4001 5BFF	1 KB	SAI2	*Section 39.5.10: SAI register map*
	0x4001 5400 - 0x4001 57FF	1 KB	SAI1	*Section 39.5.10: SAI register map*
	0x4001 4C00 - 0x4000 53FF	2 KB	Reserved	-
APB2	0x4001 4800 - 0x4001 4BFF	1 KB	TIM17	*Section 28.6.20: TIM16/TIM17 register map*
	0x4001 4400 - 0x4001 47FF	1 KB	TIM16	*Section 28.6.20: TIM16/TIM17 register map*
	0x4001 4000 - 0x4001 43FF	1 KB	TIM15	*Section 28.6.20: TIM16/TIM17 register map*
	0x4001 3C00 - 0x4001 3FFF	1 KB	Reserved	-
	0x4001 3800 - 0x4001 3BFF	1 KB	USART1	*Section 36.8.12: USART register map*
	0x4001 3400 - 0x4001 37FF	1 KB	TIM8	*Section 26.4.31: TIM8 register map*
	0x4001 3000 - 0x4001 33FF	1 KB	SPI1	*Section 38.6.8: SPI register map*
	0x4001 2C00 - 0x4001 2FFF	1 KB	TIM1	*Section 26.4.30: TIM1 register map*
	0x4001 2800 - 0x4001 2BFF	1 KB	SDMMC1	*Section 41.8.16: SDMMC register map*

Timer memory map address

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
APB1	0x4000 9800 - 0x4000 FFFF	26 KB	Reserved	-
	0x4000 9400 - 0x4000 97FF	1 KB	LPTIM2	*Section 30.7.11: LPTIM register map*
	0x4000 8C00 - 0x4000 93FF	2 KB	Reserved	-
	0x4000 8800 - 0x4000 8BFF	1 KB	SWPMI1	*Section 40.6.10: SWPMI register map and reset value table*
	0x4000 8400 - 0x4000 87FF	1 KB	Reserved	-
	0x4000 8000 - 0x4000 83FF	1 KB	LPUART1	*Section 37.7.10: LPUART register map*
	0x4000 7C00 - 0x4000 7FFF	1 KB	LPTIM1	*Section 30.7.11: LPTIM register map*
	0x4000 7800 - 0x4000 7BFF	1 KB	OPAMP	*Section 20.5.7: OPAMP register map*
	...			
	0x4000 1800 - 0x4000 2400	3 KB	Reserved	-
	0x4000 1400 - 0x4000 17FF	1 KB	TIM7	*Section 29.4.9: TIM6/TIM7 register map*
	0x4000 1000 - 0x4000 13FF	1 KB	TIM6	*Section 29.4.9: TIM6/TIM7 register map*
	0x4000 0C00- 0x4000 0FFF	1 KB	TIM5	*Section 27.4.23: TIMx register map*
	0x4000 0800 - 0x4000 0BFF	1 KB	TIM4	*Section 27.4.23: TIMx register map*
	0x4000 0400 - 0x4000 07FF	1 KB	TIM3	*Section 27.4.23: TIMx register map*
	0x4000 0000 - 0x4000 03FF	1 KB	TIM2	*Section 27.4.23: TIMx register map*

TIMx_CR1

27.4.1 TIMx control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIF RE-MAP	Res.	CKD[1:0]		ARPE	CMS		DIR	OPM	URS	UDIS	CEN
				rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled
Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.
CEN is cleared automatically in one-pulse mode, when an update event occurs.

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable UEV event generation.
0: UEV enabled. The Update (UEV) event is generated by one of the following events:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.
1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However 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.

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 3 **OPM**: One pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 4 **DIR**: Direction
0: Counter used as upcounter
1: Counter used as downcounter
Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

TIMx_CR1

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 9:8 **CKD[1:0]**: Clock division

This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (ETR, TIx),

00: $t_{DTS}=t_{CK\_INT}$
01: $t_{DTS}=2*t_{CK\_INT}$
10: $t_{DTS}=4*t_{CK\_INT}$
11: Reserved, do not program this value

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

TIMx_PSC

27.4.11 TIMx prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency CK_CNT is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

TIMx_ARR

27.4.12 TIMx auto-reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARR[31:16] (depending on timers)															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **ARR[31:16]**: High auto-reload value (on TIM2 and TIM5)

Bits 15:0 **ARR[15:0]**: Low Auto-reload Prescaler value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 27.3.1: Time-base unit on page 861* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

TIMx_CNT

27.4.10 TIMx counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNT[31] or UIFCPY	CNT[30:16] (depending on timers)														
rw or r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Value depends on IUFREMAP in TIMx_CR1.

If UIFREMAP = 0

CNT[31]: Most significant bit of counter value (on TIM2 and TIM5)

Reserved on other timers

If UIFREMAP = 1

UIFCPY: UIF Copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register

Bits 30:16 **CNT[30:16]**: Most significant part counter value (on TIM2 and TIM5)

Bits 15:0 **CNT[15:0]**: Least significant part of counter value

TIMx_EGR—UG

27.4.6 TIMx event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

3.1 Modify system initial clock

- 請呼叫assembly 版本的GPIO_init、FCLK_4MHz_delay_1s。
- 修改SYSCLK的clock source以及相關的pre-scaler使得CPU frequency為8MHz。
- 觀察修改前後LED燈閃爍的頻率。

```c
extern void GPIO_init();
extern void FCLK_4MHz_delay_1s();
void SystemClock_Config(){
    //TODO: Change the SYSCLK source and set the
corresponding Prescaler value.
}

int main(){
        SystemClock_Config();
        GPIO_init();
        while(1){
                GPIOA->BSRR = (1<<5);
                FCLK_4MHz_delay_1s();
                GPIOA->BRR = (1<<5);
                FCLK_4MHz_delay_1s();
        }
}
```

```c
void SystemClock_Config(void)
{
  RCC->CR |= RCC_CR_HSION;// turn on HSI16 oscillator
  while((RCC->CR & RCC_CR_HSIRDY) == 0);//check HSI16 ready
  SET_REG(RCC->CFGR, RCC_CFGR_HPRE, 8<<4);//SYSCLK divide by 2. SYSCLK = 16MHz/2 = 8Mhz
  if((RCC->CR & RCC_CR_HSIRDY) == 0)
    return;
  // Use HSI16 as system clock
  //APB1 prescaler not divide
  //APB2 prescaler not divide
}
```

3.2 倒數計時器

- 使用STM32 timer實做一個計時器會倒數TIME_SEC秒的時間。顯示到小數點以下第二位，倒數結束後7-SEG LED停留在0.00的狀態。(建議使用擁用比較高counter resolution 的TIM2~TIM5 timer)，請使用polling的方式取得 timer CNT register值並換算成倒數的時間顯示到7-SEG LED上。

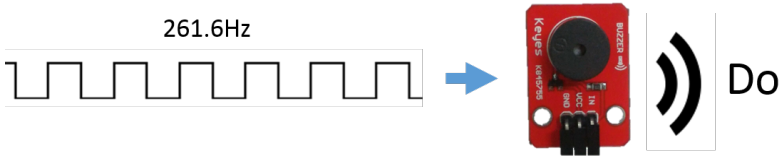
```
void InitializeTimer()
{
    RCC->APB1ENR1 |= RCC_APB1ENR1_TIM2EN;
    SET_REG(TIM2->CR1, TIM_CR1_DIR , TIM_COUNTERMODE_DOWN);//down counter
    TIM2->ARR = (uint32_t)TIM_ARR_VAL;//Reload value
    TIM2->PSC = (uint32_t)TIME_SEC*(MSI_DEFAULT_FREQ/TIM_ARR_VAL);//Prescalser
    TIM2->EGR = TIM_EGR_UG;//Reinitialize the counte
}
```

```
void start_timer(){
        TIM2->CR1 |= TIM_CR1_CEN;//start timer
        int pre_val = TIM_ARR_VAL;
        while(1){
                int timerValue = TIM2->CNT;//polling the counter value
                if(pre_val < timerValue){//check if times up
                        TIM2->CR1 &= ~TIM_CR1_CEN;
                        return;
                }
                pre_val = timerValue;
                int dis_val = TIME_SEC*100*timerValue/TIM_ARR_VAL;//convert counter value to time(seconds)

                Display_f(dis_val, 2);//display the time on the 7-SEG LED
        }
}
```

3.3. Music keypad

- 蜂鳴器分為有源(自激式)蜂鳴器和無源(他激式)蜂鳴器。有源蜂鳴器將驅動電路直接設計到蜂鳴器中，因此只需提供直流電壓就可以發出聲音，但其缺點是聲音的頻率無法更改。無源蜂鳴器外部需提供震盪波形才會發出聲音，其聲音的頻率就是輸入波的頻率。我們這次LAB使用的是無源蜂鳴器。



```
void InitializeTimer(uint32_t presc)
{
        RCC->APB1ENR1 |= RCC_APB1ENR1_TIM2EN;
        SET_REG(TIM2->CR1, TIM_CR1_DIR | TIM_CR1_CMS,
TIM_COUNTERMODE_DOWN);// Edge-aligned mode, down counter
    TIM2->ARR = (uint32_t)100;//Reload value
    TIM2->PSC = (uint32_t)presc;//Prescaler
    TIM2->EGR = TIM_EGR_UG;//Reinitialize the counter
}
```

```
while(1){
        for(col=0; col<4; col++){
                GPIOB->ODR = (GPIOB->ODR & ~0xF)|(1<<col);
                for(row = 0; row < 4; row++)
                        if((GPIOB->IDR) & (1<<(4+row))){
                        if(Table[col][row]!=0){
                        press_stat = PRESS;
                        if(befoer_val != -1)
                                TIM2->CR1 |= TIM_CR1_CEN;
                                if(befoer_val != Table[col][row]){
                                press_val = Table[col][row];
                                TIM2->PSC = Table[col][row];
                                }}}}
                if(press_stat == PRESS)
                        befoer_val = press_val;
                else{
                        TIM2->CR1 &= ~TIM_CR1_CEN;
                        befoer_val = -1;}
        press_stat = NOT_PRESS;
        int timerValue = TIM2->CNT;
        if (timerValue < 50)
                GPIOA->BRR = (1<<5);
        if (timerValue >= 50)
                GPIOA->BSRR = (1<<5);}
```

Reference

- http://www.st.com/resource/en/reference_manual/dm00083560.pdf