

# MICROS 32 BITS STM – TIMERS

ROBINSON JIMENEZ MORENO

LUISA FERNANDA GARCIA VARGAS



UNIVERSIDAD MILITAR  
NUEVA GRANADA



# TIM2/TIM3/TIM4/TIM5 functional description

## Time-base unit

The main block of the programmable timer is a 16-bit/32-bit counter with its related auto-reload register. The counter can count up, down or both up and down but also down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter Register (TIMx\_CNT)
- Prescaler Register (TIMx\_PSC):
- Auto-Reload Register (TIMx\_ARR)

The counter is clocked by the prescaler output CK\_CNT, which is enabled only when the counter enable bit (CEN) in TIMx\_CR1 register is set,

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx\_CR1 register

### 5.3.14 RCC APB2 peripheral clock enable register (RCC\_APB2ENR)

Address offset: 0x44

Reset value: 0x0000 0000

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.	LTDC EN	Res.	Res.	SAI2EN	SAI1EN	SPI6EN	SPI5EN	Res.	TIM11 EN	TIM10 EN	TIM9 EN
					rw			rw	rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SYSCFG EN	SPI4 EN	SPI1 EN	SDMMC1 EN	ADC3 EN	ADC2 EN	ADC1 EN	Res.	Res.	USART6 EN	USART1 EN	Res.	Res.	TIM8 EN	TIM1 EN
	rw	rw	rw	rw	rw	rw	rw			rw	rw			rw	rw

Bit 0 **TIM1EN**: TIM1 clock enable

This bit is set and cleared by software.

0: TIM1 clock disabled

1: TIM1 clock enabled

### 5.3.13 RCC APB1 peripheral clock enable register (RCC\_APB1ENR)

Address offset: 0x40

Reset value: 0x0000 0000

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
UART8 EN	UART7 EN	DAC EN	PWR EN	CEC EN	CAN2 EN	CAN1 EN	I2C4 EN	I2C3 EN	I2C2 EN	I2C1 EN	UART5 EN	UART4 EN	USART 3 EN	USART 2 EN	SPDIFRX EN
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

[illegible]

## Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (CK\_INT)
- External clock mode1: external input pin (Tl<sub>x</sub>)
- External clock mode2: external trigger input (ETR)
- Internal trigger inputs (ITR<sub>x</sub>): using one timer as prescaler for another timer, for example, you can configure Timer 13 to act as a prescaler for Timer 2. Refer to : *Using one timer as prescaler for another timer on page 744* for more details.



Table 12. STM32F745xx and STM32F746xx alternate function mapping

Port		AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7	AF8	AF9	AF10	AF11	AF12	AF13	AF14	AF15
		SYS	TIM1/2	TIM3/4/5	TIM8/9/10/11/LPTIM1/CEC	I2C1/2/3/4/CEC	SPI1/2/3/4/5/6	SPI3/SAI1	SPI2/3/USART1/2/3/UART5/SPDIFRX	SAI2/USART6/UART4/5/7/8/SPDIFRX	CAN1/2/TIM12/13/14/QUADSPI/LCD	SAI2/QUADSPI/OTG1_FS	ETH/OTG1_FS	FMC/SDMMC1/OTG2_FS	DCMI	LCD	SYS
Port A	PA0	-	TIM2_C H1/TIM2_ETR	TIM5_C H1	TIM8_ET R	-	-	-	USART2_CTS	UART4_TX	-	SAI2_SD_B	ETH_MII_CRS	-	-	-	EVEN TOUT
	PA1	-	TIM2_C H2	TIM5_C H2	-	-	-	-	USART2_RTS	UART4_RX	QUADSP I_BK1_IO 3	SAI2_MC K_B	ETH_MII_RX_CLK/ ETH_RMII_REF_C LK	-	-	LCD_R2	EVEN TOUT
	PA2	-	TIM2_C H3	TIM5_C H3	TIM9_CH 1	-	-	-	USART2_TX	SAI2_SC K_B	-	-	ETH_MDI O	-	-	LCD_R1	EVEN TOUT
	PA3	-	TIM2_C H4	TIM5_C H4	TIM9_CH 2	-	-	-	USART2_RX	-	-	OTG_HS_ULPI_D0	ETH_MII_COL	-	-	LCD_B5	EVEN TOUT
	PA4	-	-	-	-	-	SPI1_NS S/I2S1_WS	SPI3_NS S/I2S3_WS	USART2_CK	-	-	-	-	OTG_HS_SOF	DCMI_H SYNC	LCD_VS YNC	EVEN TOUT
	PA5	-	TIM2_C H1/TIM2_ETR	-	TIM8_CH 1N	-	SPI1_SC K/I2S1_CK	-	-	-	-	OTG_HS_ULPI_CK	-	-	-	LCD_R4	EVEN TOUT
	PA6	-	TIM1_B KIN	TIM3_C H1	TIM8_BK1 N	-	SPI1_MI SO	-	-	-	TIM13_C H1	-	-	-	DCMI_PI XCLK	LCD_G2	EVEN TOUT
							SPI1_M						ETH_MII_				

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
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
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits  $2y+1:2y$  **MODER $y$ [1:0]**: Port x configuration bits ( $y = 0..15$ )

These bits are written by software to configure the I/O mode.

00: Input mode (reset state)

01: General purpose output mode

10: Alternate function mode

11: Analog mode

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFR7[3:0]				AFR6[3:0]				AFR5[3:0]				AFR4[3:0]			
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
AFR3[3:0]				AFR2[3:0]				AFR1[3:0]				AFR0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFRy[3:0]**: Alternate function selection for port x pin y (y = 0..7)

These bits are written by software to configure alternate function I/Os

AFSELy selection:

0000: AF0

0001: AF1

0010: AF2

0011: AF3

0100: AF4

0101: AF5

0110: AF6

0111: AF7

1000: AF8

1001: AF9

1010: AF10

1011: AF11

1100: AF12

1101: AF13

1110: AF14

1111: AF15



## 23.4.1 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

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.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD**: Clock division

This bit-field indicates the division ratio between the timer clock (CK\_INT) frequency and sampling clock used by the digital filters (ETR, Tlx),

00:  $t_{DTS} = t_{CK\_INT}$

01:  $t_{DTS} = 2 \times t_{CK\_INT}$

10:  $t_{DTS} = 4 \times t_{CK\_INT}$

11: Reserved

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

0: TIMx\_ARR register is not buffered

1: TIMx\_ARR register is buffered

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

0: TIMx\_ARR register is not buffered

1: TIMx\_ARR register is buffered

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 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.*

## 23.4.1 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 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.*

...

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.

## 23.4.6 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).

## 23.4.10 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

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



## 23.4.11 TIMx prescaler (TIMx\_PSC)

Address offset: 0x28

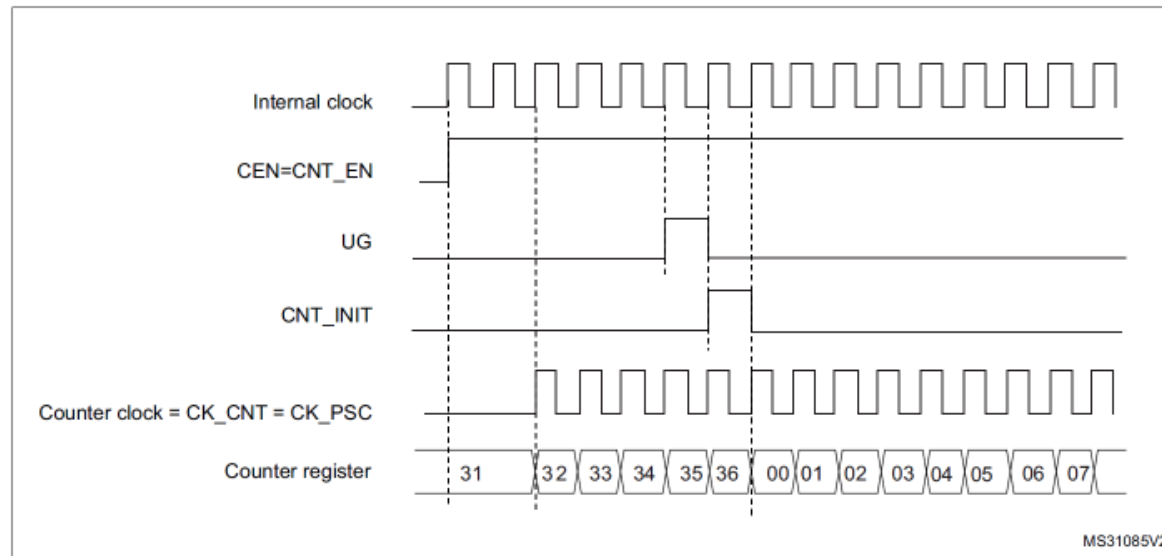
Reset value: 0x0000

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

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”).



## Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx\_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 150. Counter timing diagram with prescaler division change from 1 to 2

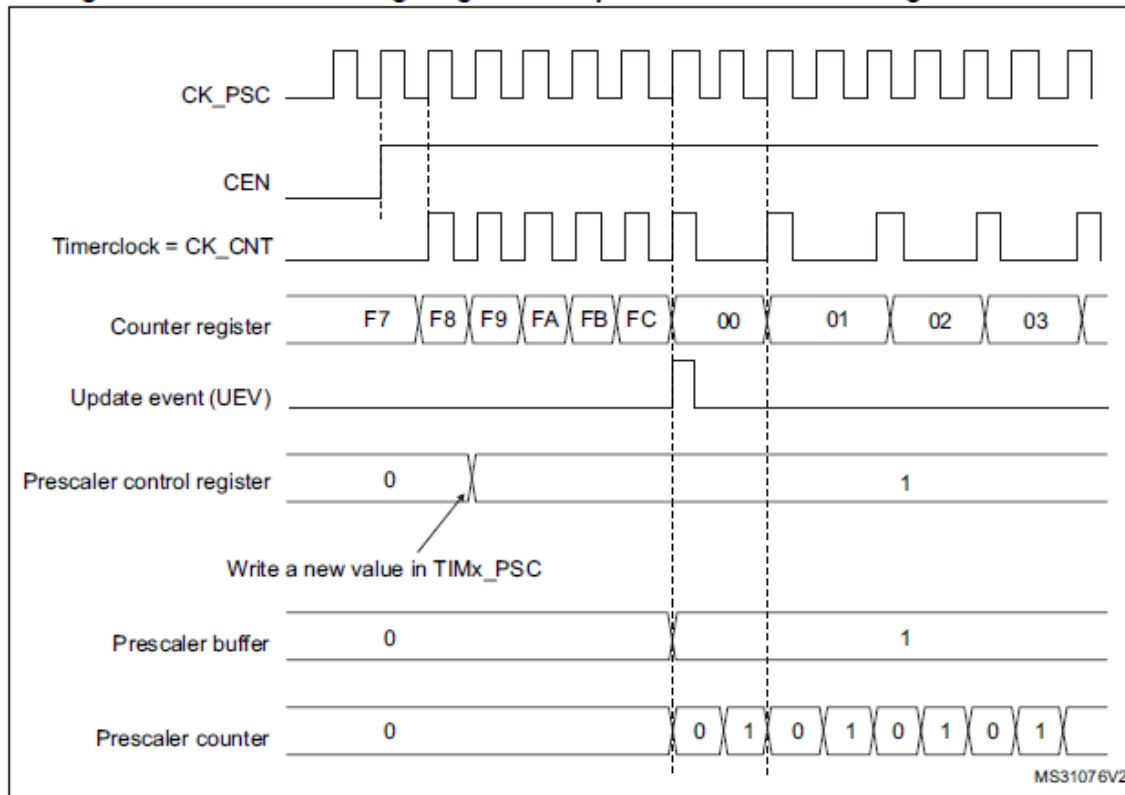
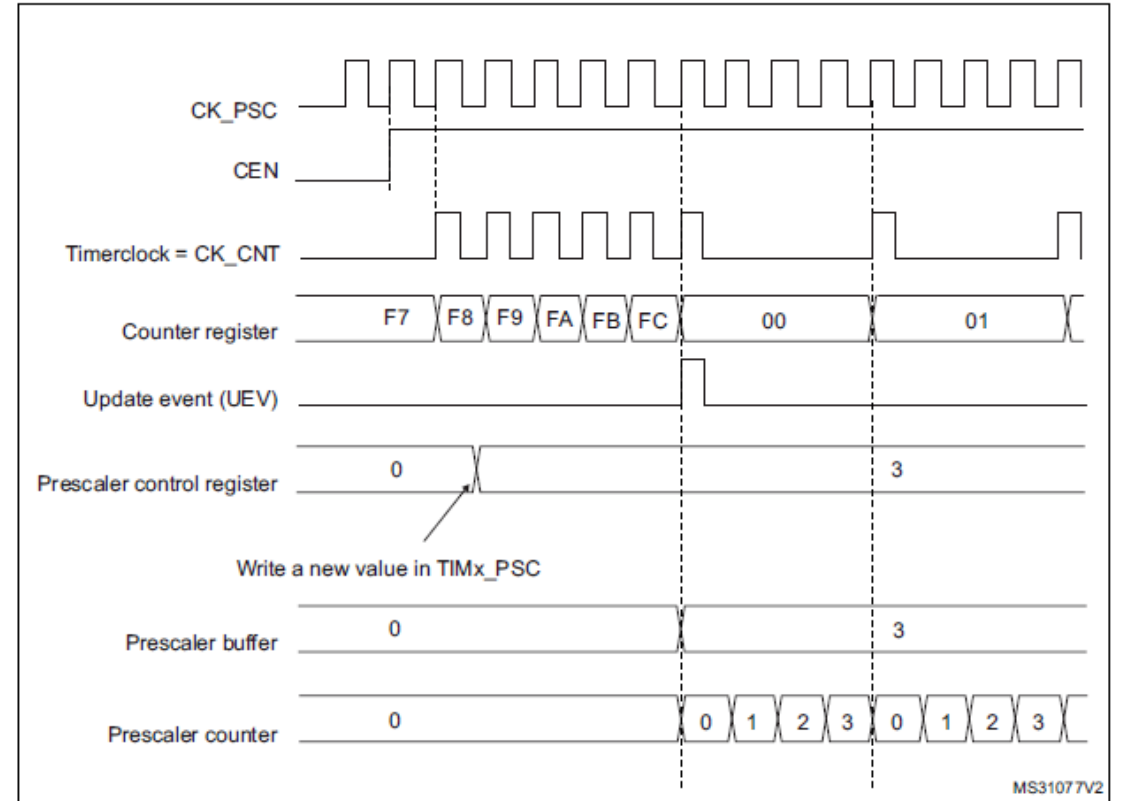


Figure 151. Counter timing diagram with prescaler division change from 1 to 4



## 23.4.12 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.

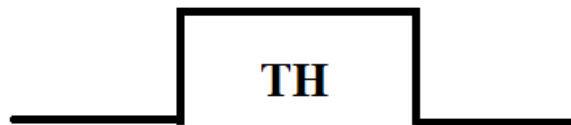
Refer to the [Section 23.3.1: Time-base unit on page 706](#) for more details about ARR update and behavior.

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

```

1  #include "stm32f7xx.h"
2
3  int main(void) {
4      RCC->AHB1ENR=0X6;
5      RCC->APB1ENR=0X1;
6      GPIOC->MODER=0;
7      GPIOB->MODER|=1;
8      TIM2->CR1=0X1;
9      TIM2->ARR=9000;
10     TIM2->PSC=20000;
11     GPIOB->ODR=0;
12
13     while(true) {
14         if((GPIOC->IDR & 0X2000)==0X2000) {
15             TIM2->EGR=1;
16             while(TIM2->CNT<1600) {GPIOB->ODR=1;}
17             GPIOB->ODR=0;
18         }
19     }
20 }
21

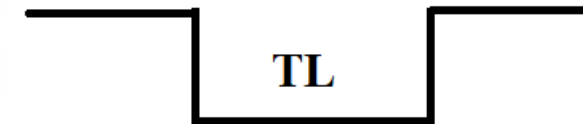
```



```

1  #include "stm32f7xx.h"
2
3  int main(void) {
4      RCC->AHB1ENR=0X6;
5      RCC->APB1ENR=0X1;
6      GPIOC->MODER=0;
7      GPIOB->MODER|=1;
8      TIM2->CR1=0X1;
9      TIM2->ARR=9000;
10     TIM2->PSC=20000;
11     GPIOB->ODR=0;
12
13     while(true) {
14         if((GPIOC->IDR & 0X2000)==0X2000) {
15             TIM2->EGR=1;
16             while(TIM2->CNT<1600) {GPIOB->ODR=0;}
17             GPIOB->ODR=1;
18         }
19     }
20 }
21

```



main.cpp

```
1  #include "stm32f7xx.h"
2
3  int main(void) {
4      RCC->AHB1ENR=0X6;
5      RCC->APB1ENR=0X1;
6      GPIOC->MODER=0;
7      GPIOB->MODER|=1;
8      TIM2->CR1=0X1;
9      TIM2->ARR=9000;
10     TIM2->PSC=20000;
11     GPIOB->ODR=0;
12
13     while(true) {
14         if((GPIOC->IDR & 0X2000)==0X2000) {
15             TIM2->EGR=1;
16             while(TIM2->CNT<1600) {GPIOB->ODR=1;}
17             GPIOB->ODR=0;
18         }
19     }
20 }
21
```

```
1  #include "stm32f7xx.h"
2
3  int main(void) {
4      RCC->AHB1ENR=0X6;
5      RCC->APB1ENR=0X1;
6      GPIOC->MODER=0;
7      GPIOB->MODER|=1;
8      TIM2->CR1=0X1;
9      TIM2->ARR=9000;
10     TIM2->PSC=20000;
11     GPIOB->ODR=0;
12
13     while(true) {GPIOB->ODR=0;
14         if((GPIOC->IDR & 0X2000)==0X2000) {
15             TIM2->EGR=1;
16             while( TIM2->CNT < 1600 ){}
17             while( TIM2->CNT > 1600 & TIM2->CNT < 3200) {GPIOB->ODR=1;}
18         }
19     }
20 }
21
22
```

TH





```
#include <stdio.h>
#include "stm32f7xx.h"

int main(void){
    RCC -> AHB1ENR = 0X2; //PUERTO B
    RCC -> APB1ENR = 0X1; //TIMER 2
    GPIOB -> MODER = 0X10004001; //COLOCAR EN SALIDA PARA ENCENDER
    GPIOB -> OTYPER = 0X0; //PUSH PULL
    GPIOB -> OSPEEDR = 0X10004001; //VELOCIDAD MEDIA
    GPIOB -> PUPDR = 0X10004001; //PULL UP
    TIM2 -> CR1 = 0X1; //CONTADOR HABILITADO, DIVISION X1
    TIM2 -> ARR = 0xFFFFFFFF; //VALOR DE RESET
    TIM2 -> PSC = 20000; //PRE-ESCALER DE TIEMPO

    while(1){

        if (TIM2 -> CNT < 10){
            GPIOB -> ODR = 0X0080;
        }
        else if (TIM2 -> CNT > 200 & TIM2 -> CNT < 500){
            GPIOB -> ODR = 0X0001;
        }
        else if (TIM2 -> CNT > 500 & TIM2 -> CNT < 700 ){
            GPIOB -> ODR = 0X4000;
        }
        else if (TIM2 -> CNT > 900){
            TIM2 -> EGR = 1;
        }
    }
}
```

```
#include <stdio.h>
#include "stm32f7xx.h"

int main(void){
    RCC -> AHB1ENR = 0X2; //PUERTO B
    RCC -> APB1ENR = 0X1; //TIMER 2
    GPIOB -> MODER = 0X10004001; //COLOCAR EN SALIDA PARA ENCENDER LOS LEDS
    GPIOB -> OTYPER = 0X0; //PUSH PULL
    GPIOB -> OSPEEDR = 0X10004001; //VELOCIDAD MEDIA
    GPIOB -> PUPDR = 0X10004001; //PULL UP

    TIM2 -> CR1 = 0X1; //CONTADOR HABILITADO, DIVISION X1
    TIM2 -> ARR = 800; //VALOR DE RESET
    TIM2 -> PSC = 20000; //PRE-ESCALER DE TIEMPO

    while(1){
        if (TIM2 -> CNT < 10){
            GPIOB -> ODR = 0X0080;
        }
        else if (TIM2 -> CNT > 200 & TIM2 -> CNT < 500){
            GPIOB -> ODR = 0X0001;
        }
        else if (TIM2 -> CNT > 500){
            GPIOB -> ODR = 0X4000;
        }
    }
}
```

```

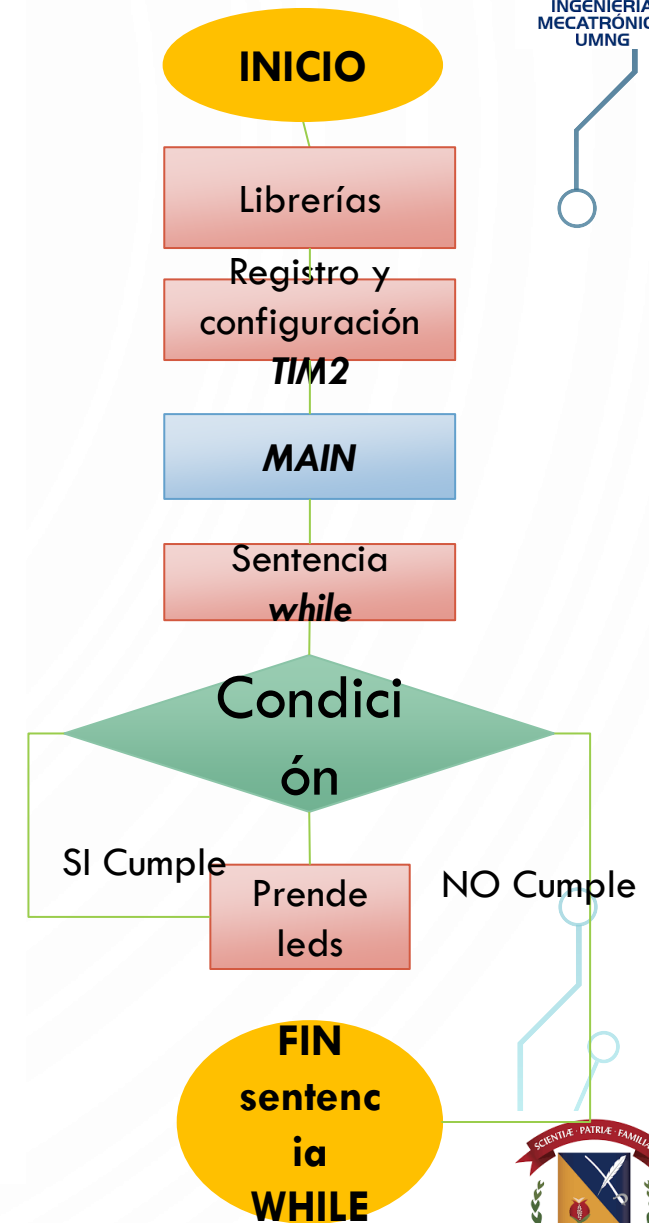
////////////////////////////////////

#include <stdio.h>
#include "stm32f7xx.h"

int main(void) {
    int temp=0;
    RCC -> AHB1ENR = 0X6; //PUERTO B
    RCC -> APB1ENR = 0X1; //TIMER 2
    GPIOC -> MODER = 0;
    GPIOB -> MODER = 0X10004001; //COLOCAR EN SALIDA PARA ENCENDER LOS LEDS
    GPIOB -> OTYPER = 0X0; //PUSH PULL
    GPIOB -> OSPEEDR = 0X10004001; //VELOCIDAD MEDIA
    GPIOB -> PUPDR = 0X10004001; //PULL UP
    TIM2 -> CR1 = 0X1; //CONTADOR HABILITADO, DIVISION X1
    TIM2 -> ARR = 0X500; //VALOR DE RESET
    TIM2 -> PSC = 20000; //PRE-ESCALER DE TIEMPO

    while(1) {
        if ((GPIOC-> IDR & 0X2000)== 0X2000){
            temp= TIM2 -> CNT;
            if (temp <10){
                GPIOB -> ODR = 0X0080;    }
            else if (temp > 500 & temp <700 ){
                GPIOB -> ODR = 0X4000;    }
        }
    }
}

```



```
////////////////////////////////////  
  
#include <stdio.h>  
#include "stm32f7xx.h"  
  
int main(void){  
    int temp=0;  
    RCC -> AHB1ENR = 0X6; //PUERTO B  
    RCC -> APB1ENR = 0X1; //TIMER 2  
    GPIOC -> MODER = 0;  
    GPIOB -> MODER = 0X10004001; //COLOCAR EN SALIDA PARA ENCENDER LOS LEDS  
    GPIOB -> OTYPER = 0X0; //PUSH PULL  
    GPIOB -> OSPEEDR = 0X10004001; //VELOCIDAD MEDIA  
    GPIOB -> PUPDR = 0X10004001; //PULL UP  
    TIM2 -> CR1 = 0X1;  
    TIM2 -> ARR = 0X550000; //VALOR DE RESET  
    TIM2 -> PSC = 20000; //PRE-ESCALER DE TIEMPO  
  
    while(1){  
        if ((GPIOC-> IDR & 0X2000)== 0X2000){  
            TIM2 -> EGR = 1;  
            while ((GPIOC-> IDR & 0X2000)== 0X2000);  
            temp= (TIM2 -> CNT)/2;  
            if (temp <500){  
                GPIOB -> ODR = 0X0080;    }  
            else if (temp > 600 & temp <900 ){  
                GPIOB -> ODR = 0X4000;    }  
            else GPIOB -> ODR = 0X4081;  
        }  
    }  
}
```

#### 22.4.4 TIM1/TIM8 DMA/interrupt enable register (TIMx\_DIER)

Address offset: 0x0C

Reset value: 0x0000

[illegible]

Bit 0 **UIE**: Update interrupt enable

0: Update interrupt disabled

## 1: Update interrupt enabled

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 (update if repetition counter = 0) and if the UDIS=0 in the TIMx\_CR1 register.

- When CNT is reinitialized by software using the UG bit in TIMx\_EGR register, if URS=0 and UDIS=0 in the TIMx\_CR1 register.

- When CNT is reinitialized by a trigger event (refer to *Section 22.4.3: TIM1/TIM8 slave mode control register (TIMx\_SMCR)*), if URS=0 and UDIS=0 in the TIMx\_CR1 register.

#### 22.4.5 TIM1/TIM8 status register (TIMx\_SR)

Address offset: 0x10

Reset value: 0x0000 0000

[illegible]

# TIMER 1 SEG ON

```

1  #include <stdio.h>
2  #include "stm32f7xx.h"
3
4  int main(void){
5      RCC -> AHB1ENR = 0X2; //PUERTO B
6      RCC -> APB1ENR = 0X1; //TIMER 2
7      GPIOB -> MODER = 0X10004001; //COLOCAR EN SALIDA PARA ENCENDER LOS LEDS
8      GPIOB -> OTYPER = 0X0; //PUSH PULL
9      GPIOB -> OSPEEDR = 0X10004001; //VELOCIDAD MEDIA
10     GPIOB -> PUPDR = 0X10004001; //PULL UP
11
12     TIM2 -> CR1 = 0X1; //CONTADOR HABILITADO, DIVISION X1
13     TIM2 -> DIER = 0X1; //HABILITAR LA INTERRUPCION AL TERMINAR CADA CONTEO
14     TIM2 -> ARR = 800; //CALCULAR CON TIMER CALCULATOR
15     TIM2 -> PSC = 20000; //CALCULAR CON TIMER CALCULATOR 1HZ
16     NVIC_EnableIRQ(TIM2_IRQn);
17
18     while(1){
19
20     }
21 }
22
23 extern "C"{
24
25     void TIM2_IRQHandler(void)
26     {
27         TIM2->SR &= ~(1<<0);
28         GPIOB -> ODR ^= 0X4081; //INTERMITENCIA DE LOS LEDS
29     }
30
31 }

```

