Tutorial: Timer Interrupt & PWM Output

I. Overview

In this lab, you will learn how to set up MCU Timers for Timer Interrupt and PWM output.

Objectives of this lab are learning how to

- Configure registers of Timers(TIMx)
- Generate Timer Interrupt
- Generate PWM signals

Preparation:

- You need to study the following registers: Timer(Advanced and General Purpose, TIMx) in 'STM Reference Manual'
- User defined APIs for GPIO and External Interrupt control

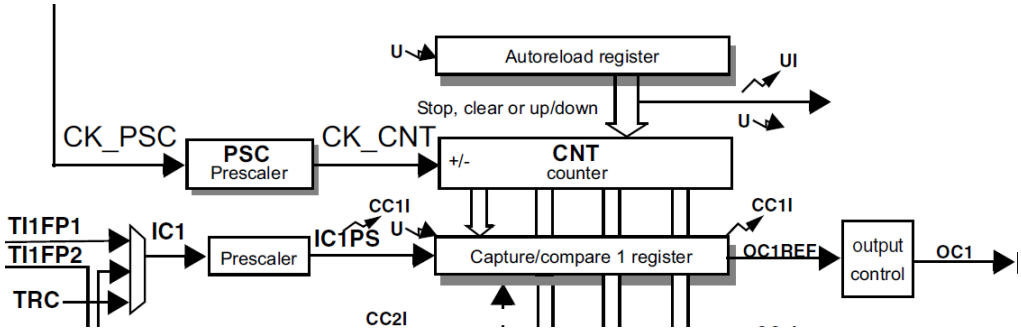
II. Tutorial – Timer Interrupt / No Output

A. Timer Registers: TIMx

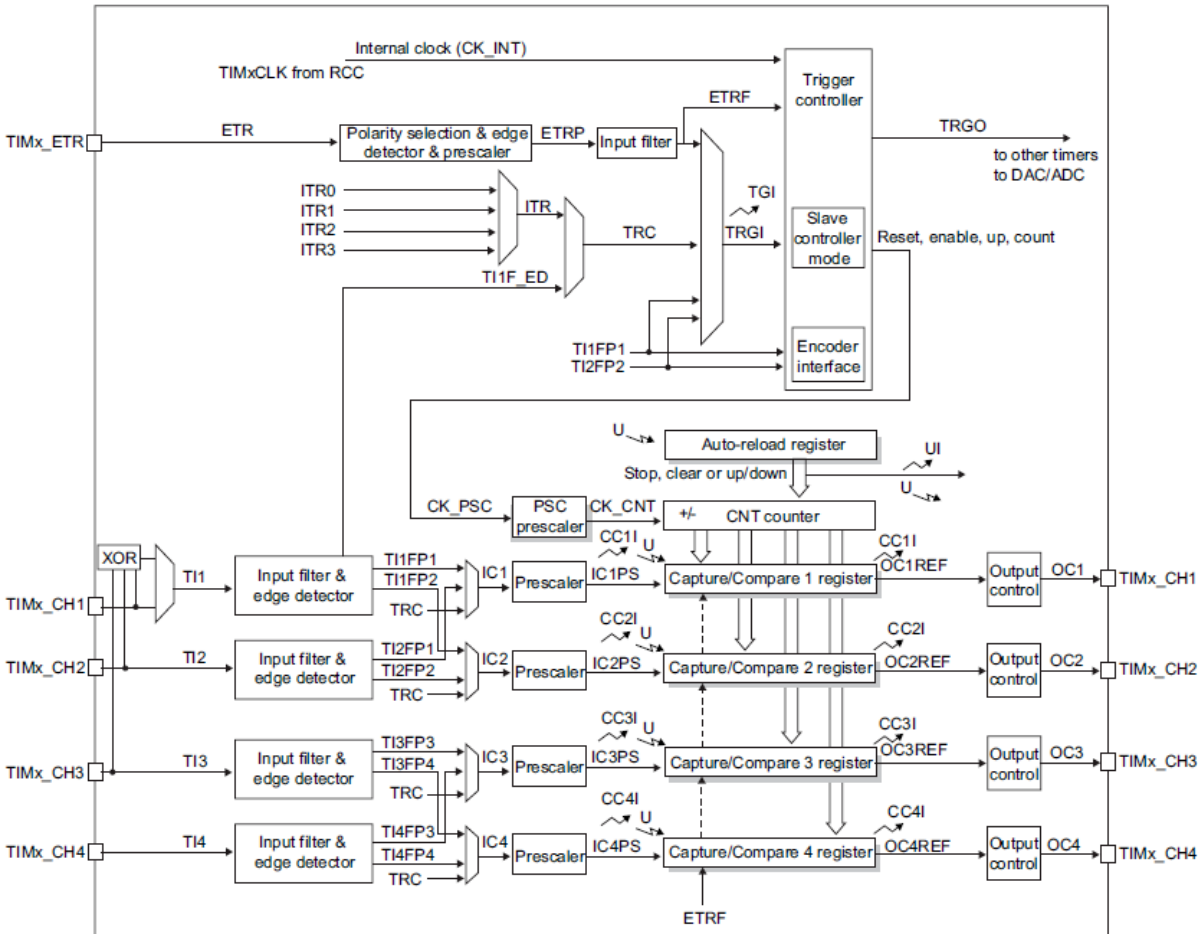
- List TIMx registers for this LAB

Class	Register Name	Description
TIMx	TIMx_CR1	TIMx control register 1
	TIMx_PSC	TIMx prescaler register
	TIMx_ARR	TIMx auto-reload register
	TIMx_DIER	TIMx DMA Interrupt Enable register
	TIMx_BDTR	TIM1(only) break and dead-time register

- Block Diagram for General Purpose Timer



- Block Diagram for General Purpose Timer



- TIM_TypeDef : <stm32f411xe.h>

```
typedef struct
{
  __IO uint32_t CR1;         /*!< TIM control register 1,              Address offset: 0x00 */
  __IO uint32_t CR2;         /*!< TIM control register 2,              Address offset: 0x04 */
  __IO uint32_t SMCR;        /*!< TIM slave mode control register,     Address offset: 0x08 */
  __IO uint32_t DIER;        /*!< TIM DMA/interrupt enable register,   Address offset: 0x0C */
  __IO uint32_t SR;          /*!< TIM status register,                 Address offset: 0x10 */
  __IO uint32_t EGR;         /*!< TIM event generation register,       Address offset: 0x14 */
  __IO uint32_t CCMR1;       /*!< TIM capture/compare mode register 1, Address offset: 0x18 */
  __IO uint32_t CCMR2;       /*!< TIM capture/compare mode register 2, Address offset: 0x1C */
  __IO uint32_t CCER;        /*!< TIM capture/compare enable register, Address offset: 0x20 */
  __IO uint32_t CNT;         /*!< TIM counter register,                Address offset: 0x24 */
  __IO uint32_t PSC;         /*!< TIM prescaler,                       Address offset: 0x28 */
  __IO uint32_t ARR;         /*!< TIM auto-reload register,            Address offset: 0x2C */
  __IO uint32_t RCR;         /*!< TIM repetition counter register,     Address offset: 0x30 */
  __IO uint32_t CCR1;        /*!< TIM capture/compare register 1,      Address offset: 0x34 */
  __IO uint32_t CCR2;        /*!< TIM capture/compare register 2,      Address offset: 0x38 */
  __IO uint32_t CCR3;        /*!< TIM capture/compare register 3,      Address offset: 0x3C */
  __IO uint32_t CCR4;        /*!< TIM capture/compare register 4,      Address offset: 0x40 */
  __IO uint32_t BDTR;        /*!< TIM break and dead-time register,    Address offset: 0x44 */
  __IO uint32_t DCR;         /*!< TIM DMA control register,            Address offset: 0x48 */
  __IO uint32_t DMAR;        /*!< TIM DMA address for full transfer,   Address offset: 0x4C */
  __IO uint32_t OR;          /*!< TIM option register,                 Address offset: 0x50 */
} TIM_TypeDef;
```

```
#define TIM2                ((TIM_TypeDef *) TIM2_BASE)
#define TIM3                ((TIM_TypeDef *) TIM3_BASE)
#define TIM4                ((TIM_TypeDef *) TIM4_BASE)
#define TIM5                ((TIM_TypeDef *) TIM5_BASE)

/*!< APB1 peripherals */
#define TIM2_BASE           (APB1PERIPH_BASE + 0x0000UL)
#define TIM3_BASE           (APB1PERIPH_BASE + 0x0400UL)
#define TIM4_BASE           (APB1PERIPH_BASE + 0x0800UL)
#define TIM5_BASE           (APB1PERIPH_BASE + 0x0C00UL)
```

B. Register Initialization Process

Process of Timer(TIMx) register initiation: Timer Interrupt of Over/Underflow of Counter

System Clock setting

1. RCC setting (HSE/PLL)

Timer setting

1. Enable Timer Peripheral Clock	(**RCC_APB1ENR**)
2. Set Counting Direction	(**TIMx_CR1→DIR**)
3. Set Timer Clock Pre-scaler value	(**TIMx_PSC→PSC[15:0]**)
4. Set Auto-reload value	(**TIMx_ARR->ARR**)
5. Enable Timer DMA/Interrupt.	**(TIMx_DIER→UIE)**
6. Enable counter	(**TIMx_CR1→CEN**)

NVIC setting

1. Enable TIMx Interrupt:	NVIC_EnableIRQ(TIMx_IRQn)
2. Set interrupt Priority	NVIC_SetPriority(TIMx_IRQn,2)

C. Exercise

- Timer interrupt for every 1 sec.
- System CLK is PLL 84MHz for STM32F411RE
- Use Counter of TIM2: Up-counting, Timer2_CLK = 100 kHz, COUNT_CLK=1 kHz

1. Fill in the table

Port/Pin	Description	Register setting
RCC	PLL Initialization	RCC_PLL_init(); EC_SYS_CLK =EC_PLL= 84,000,000
	Enable Timer Peripheral Clock: TIM2	RCC->APB1ENR |=1≪0
TIM2	TIM2 counting direction: DIR0	TIM2->CR1 & = ~ (1≪0)
	Set Timer Clock Pre-scaler value 84MHz To 100kHz	TIM2->PSC _____
	Set Auto-reload value: With 100kHz, counting of 1kHz	TIM2->ARR _____
	Enable Timer DMA/Interrupt	TIM2->DIER_____
	Enable Counter	TIM2->CR1_____
NVIC	Set TIM2_IRQn with Priority 2, and enable	NVIC_SetPriority(_____, ___); NVIC_EnableIRQ(_____);

2. Firmware Programming

- Create a new project and name the project as '**Tutorial_TIMER_Interrupt'**.
- Create a new item called 'main.c'
- Then, using timer interrupt (" void TIM2_IRQHandler(void)"), make LED LD2 turn On and Off in every 1 sec: ON for 1sec, OFF for 1 sec etc.

```
void TIM2_IRQHandler(void){
    if ((TIM2->SR & TIM_SR_UIF) != 0) { //UIF pended,

      // your code goes here
      // your code goes here


    }
    TIM2->SR &= ~TIM_SR_UIF; // Clear UIF flag
}
```

III. Tutorial – PWM Output

A. Timer Registers: TIMx

- List TIMx registers: Timer Setting

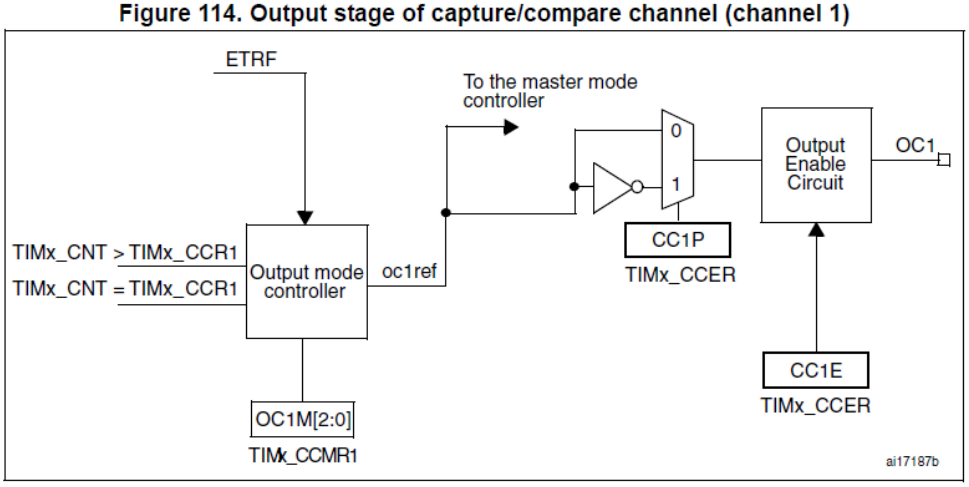
Class	Register Name	Description
TIMx	TIMx _CR1	TIMx control register 1
	TIMx _PSC	TIMx prescaler register
	TIMx _ARR	TIMx auto-reload register
	TIMx _DIER	TIMx DMA Interrupt Enable register
	TIMx _BDTR	TIM1(only) break and dead-time register

- List TIMx registers: PWM output

Class	Register Name	Description
TIMx	TIMx _CCMRn	TIMx capture/compare mode register for nth channel
	TIMx _CCRn	TIMx capture/compare register for nth channel
	TIMx _CCER	TIMx capture/compare output enable register
	TIMx _BDTR	TIM1(only) break and dead-time register

- Block Diagram for General Purpose Timer

Figure 114. Output stage of capture/compare channel (channel 1)



B. Register Initialization Process

Process of Timer(TIMx) PWM output register initiation

GPIO Pin setting

1. Set RCC for GPIO

2. AF(TIMx) mode selection for Pin_y in GPIOx

Timer setting:

1. Enable Timer peripheral Clock (**RCC_APB1ENR**)

2. Set Counting direction (**TIMx_CR1→DIR**)

3. Set Timer clock Pre-scaler value (**TIMx_PSC→PSC[15:0]**)

4. Set Auto-reload value (**TIMx_ARR->ARR**)

5. Enable Counter (**TIMx_CR1→CEN**)

PWM Out setting:

1. Set PWM Output mode (**TIMx_CCMR →OCnM**)

2. Set CompareCapture value (**TIMx_CCRn->CCR**)

3. Select Output Polarity (**TIMx_CCER→CCyP**)

4. Enable CompareCaptureOutput (**TIMx_CCER→CCyE**)

C. Exercise

PWM output of period 1kHz with duty ratio 50%. PWM Out on pin PA_5 (TIM2_CH1)

- Timer Clock
  - System CLK is PLL 84MHz for STM32F411RE
  - Use Counter of TIM2: Up-counting, Timer2_CLK = 100 kHz, COUNT_CLK=1 kHz
- PWM output: TIM2_CH1 / PA_5
  - There are several pins for TIM2CH1. We will use GPIO Pin5 (LD2) for tutorial
  - Set the GPIO pin as Alternate function (AF) for TIM2 / No pull-up & No pull-down / High speed / Push-Pull
  - PWM mode 1, Duty ratio 50%

3. Fill in the table

Port/Pin	Description	Register setting
RCC	PLL Initialization	RCC_PLL_init(); EC_SYS_CLK =EC_PLL= 84,000,000
	Enable Timer Peripheral Clock: TIM2	RCC->APB1ENR |=1≪0
TIM2 (Timer setting)	TIM2 counting direction: DIR0	TIM2->CR1 & = ~ (1≪0)
	Set Timer Clock Pre-scaler value 84MHz To 100kHz	TIM2->PSC _____
	Set Auto-reload value: With 100kHz, counting of 1kHz	TIM2->ARR _____
	Enable Timer DMA/Interrupt	TIM2->DIER_____
	Enable Counter	TIM2->CR1_____
GPIOA	Set GPIOA pin 5 as Alternate Function mode: 10	GPIOA->MODER &=~_____ GPIOA->MODER |= _____
	GPIOA Alternate Function Selection: AF1(TIM2/TMI5) AFRL_5[3:0]=0001	GPIOA->AFR[0] _____
	AF Output as No-PUPD, Push-Pull,Very High Speed.	// write your HAL API _____ _____
TIM2 (PWM setting)	Enable auto-reload preload	TIM2_CR1_____
	Output compare mode as PWM mode1	TIM2_CCMRn_____
	Set CCR value for 50% duty ratio: (ARR+1)/2	TIM2_CCR1_____
	Output as active high	TIM2_CCER_____
	Enable Compare and Capture output	TIM2_CCER_____

4. Firmware Programming

- Create a new project and name the project as '**Tutorial_TIMER_PWMout**'.
- Give PWM output of 0%, 50%, 100% of 1kHz PWM
- Check the brightness of LD2 (PA5) for each different Duty ratio
- Change to 50% Duty ratio of 10kHz frequency PWM

```
17  #define LED_PIN   5
18
19  void setup(void);
20
21  int main(void) {
22    // Initialiization --------------------------------------------------------
23    RCC_PLL_init();                         // System Clock = 84MHz
24    SysTick_init();                         // for delay_ms()
25    GPIO_init(GPIOA, LED_PIN, EC_ALTE);     // GPIOA 5 ALTERNATE function
26    GPIO_ospeed(GPIOA, LED_PIN, EC_HIGH);   // GPIOA 5 HIGH SPEED
27
28    // TEMP: TIMER Register Initialiization -----------------------------------------------------
29    TIM_TypeDef *TIMx;
30    TIMx = TIM2;
31
32    // GPIO: ALTERNATIVE function setting
33    GPIOA->AFR[0]  =                                    // AF1 at PA5 = TIM2_CH1 (p.150)
34
35    // TIMER: PWM setting
36    RCC->APB1ENR |=                                     // Enable TIMER clock
37
38    TIMx->CR1 &=                                        // Direction Up-count
39
40    TIMx->PSC =                                         // f_cnt = 10kHz
41
42    TIMx->ARR =                                         // Auto-reload: Upcounting (0...ARR).
43
44    TIMx->CCMR1 &= ~TIM_CCMR1_OC1M;                     // Clear ouput compare mode bits for channel 1
45    TIMx->CCMR1 |=                                      // OC1M = 110 for PWM Mode 1 output on ch1
46    TIMx->CCMR1 |= TIM_CCMR1_OC1PE;                     // Output 1 preload enable (make CCR1 value changable)
47
48    TIMx->CCER &= ~TIM_CCER_CC1P;                       // select output polarity: active high
49    TIMx->CCER  |=                                      // Enable output for ch1
50    TIMx->CCR1  =                                       // Output Compare Register for channel 1
51
52    TIMx->CR1  |= TIM_CR1_CEN;                          // Enable counter
53
54
55    // Inifinite Loop ----------------------------------------------------------
56    while(1){
57        //Create the code to change the brightness of LED as 10kHZ (use "delay(100)")
58      }
59    }
60  // Initialiization
61  void setup(void)
62  {
63    RCC_PLL_init();                         // System Clock = 84MHz
64    SysTick_init();                         // for delay_ms()
65  }
66
```

Appendix

1. Pin Configuration of NUCLE-F411RE

Figure 19. NUCLEO-F411RE

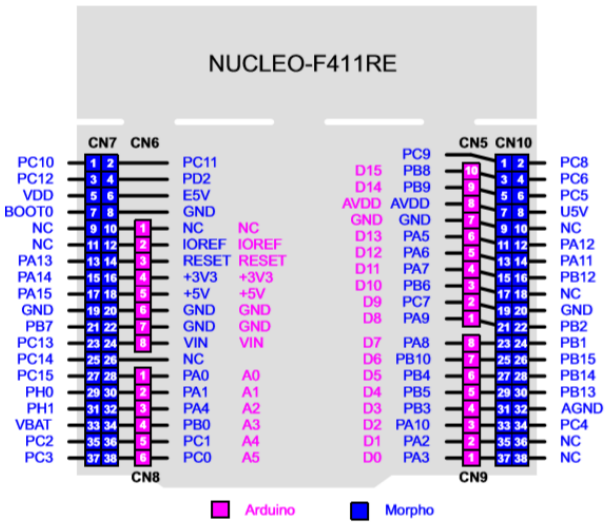


Table 29. ST morpho connector on NUCLEO-F401RE, NUCLEO-F411RE, NUCLEO-F446RE

CN7 odd pins		CN7 even pins		CN10 odd pins		CN10 even pins	
Pin	Name	Name	Pin	Pin	Name	Name	Pin
1	PC10	PC11	2	1	PC9	PC8	2
3	PC12	PD2	4	3	PB8	PC6	4
5	VDD	E5V	6	5	PB9	PC5	6
7	BOOT0(1)	GND	8	7	AVDD	U5V(2)	8
9	-	-	10	9	GND	-	10
11	-	IOREF	12	11	PA5	PA12	12
13	PA13(3)	RESET	14	13	PA6	PA11	14
15	PA14(3)	+3.3V	16	15	PA7	PB12	16
17	PA15	+5V	18	17	PB6	-	18
19	GND	GND	20	19	PC7	GND	20
21	PB7	GND	22	21	PA9	PB2	22
23	PC13	VIN	24	23	PA8	PB1	24
25	PC14	-	26	25	PB10	PB15	26
27	PC15	PA0	28	27	PB4	PB14	28
29	PH0	PA1	30	29	PB5	PB13	30
31	PH1	PA4	32	31	PB3	AGND	32
33	VBAT	PB0	34	33	PA10	PC4	34
35	PC2	PC1 or PB9(4)	36	35	PA2	-	36
37	PC3	PC0 or PB8(4)	38	37	PA3	-	38

1. Default state of BOOT0 is 0. It can be set to 1 when a jumper is on pin5-7 of CN7.Two unused jumpers are available on CN11 and CN12 (bottom side of the board).
2. U5V is 5 V power from ST-LINK/V2-1 USB connector and it rises before +5V.
3. PA13 and PA14 share with SWD signals connected to ST-LINK/V2-1, it is not recommend to use them as IO pins if ST-LINK part is not cut.
4. Refer to *Table 10: Solder bridges* for details.

2. Timer GPIO pinout for STM32f411

Timer PinOut Map

Advanced Timer

Timer	Channel	Port	Pin
1	1	A	8
	1N	A	7
		B	13
	2	A	9
	2N	B	0
		B	14
	3	A	10
	3N	B	1
		B	15
	4		
	4N		

General Purpose Timer

Timer	Channel	Port	Pin
2	1	A	0
		A	5
		A	15
	2	A	1
		B	3
	3	B	10
	4		
3	1	A	6
		B	4
		C	6
	2	B	5
		C	7
	3	C	8
	4	C	9
4	1	B	6
	2	B	7
	3	B	8
	4	B	9