**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

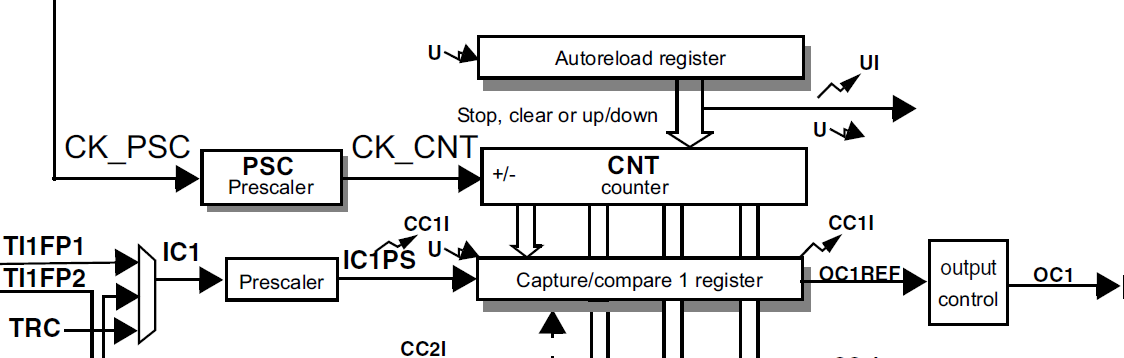
**II. Tutorial – Timer Interrupt / No Output**

**A. Timer Registers: TIMx**

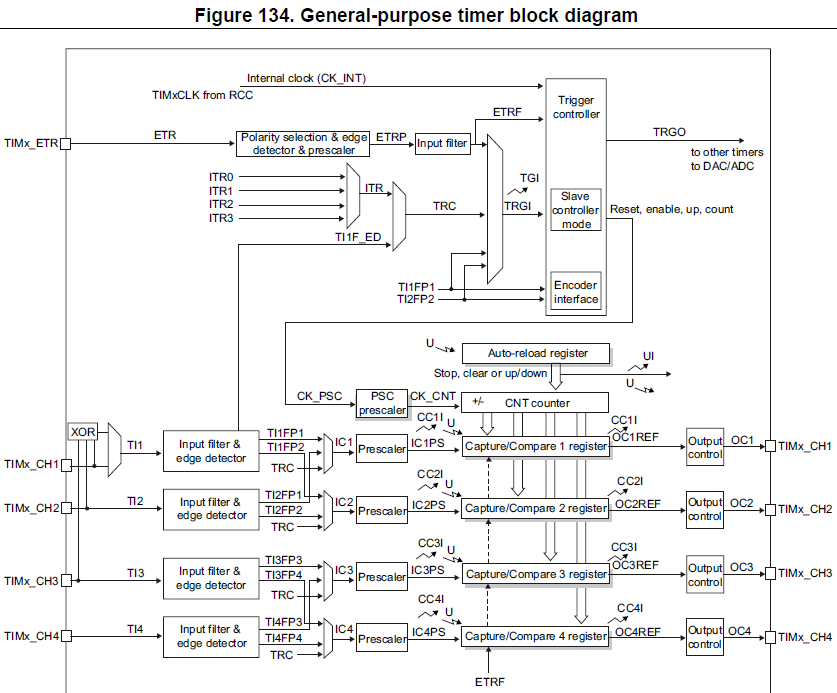
* List TIMx registers for this LAB

|  |  |  |
| --- | --- | --- |
| 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

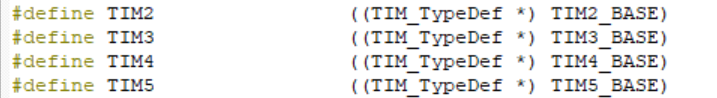


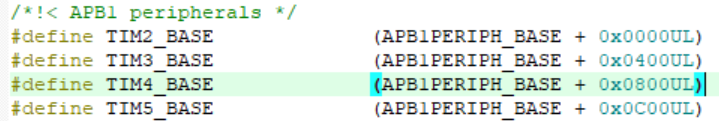
* Block Diagram for General Purpose Timer



* TIM\_TypeDef : <stm32f411xe.h>







**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

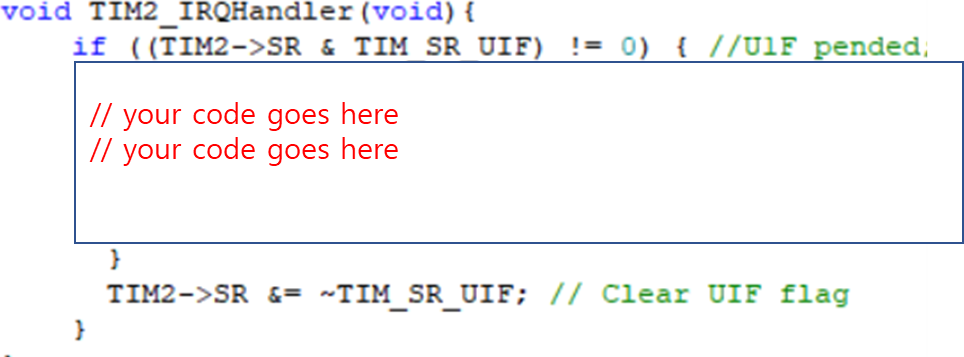
- **Use Counter of TIM2: Up-counting**, **Timer2\_CLK = 100 kHz**, **COUNT\_CLK=1 kHz**

1. **Fill in the table**

|  |  |  |
| --- | --- | --- |
| **Port/Pin** | **Description** | **Register setting** |
| RCC | PLL Initialization | RCC\_PLL\_init();  *EC\_SYS\_CLK =EC\_PLL= 84,000,000Hz* |
|  | Enable Timer Peripheral Clock: TIM2 | RCC->APB1ENR |=1≪0 |
| TIM2 | TIM2 counting direction: DIR0 | TIM2->CR1 & = ~ (1≪4) |
|  | Set Timer Clock Pre-scaler value  84MHz To 100kHz | TIM2->PSC = 840-1 |
|  | Set Auto-reload value:  With 100kHz, counting of 1kHz | TIM2->ARR = 100-1 |
|  | Enable Timer DMA/Interrupt | TIM2->DIER |= 1<<0  *// TIM\_DIER\_UIE* |
|  | Enable Counter | TIM2->CR1 |= 1<<0  *// TIM\_CR1\_CEN* |
| NVIC | Set TIM2\_IRQn with Priority 2, and enable | NVIC\_SetPriority(TIM2\_IRQn,2);  NVIC\_EnableIRQ(TIM2\_IRQn); |

1. **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.



**III. Tutorial – PWM Output**

**A. Timer Registers: TIMx**

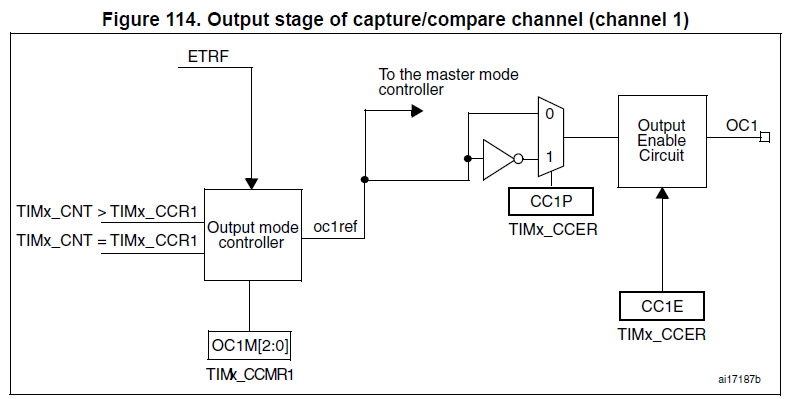
* List TIMx registers: Timer Setting

|  |  |  |
| --- | --- | --- |
| 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



**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%

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  (Timer setting) | TIM2 counting direction: DIR0 | TIM2->CR1 & = ~ (1≪0) |
|  | Set Timer Clock Pre-scaler value  84MHz To 100kHz | TIM2->PSC = 840-1 |
|  | Set Auto-reload value:  With 100kHz, counting of 1kHz | TIM2->ARR = 100-1 |
|  | Enable Timer DMA/Interrupt | TIM2->DIER |= 1<<0 |
|  | Enable Counter | TIM2->CR1 |= 1<<0 |
| GPIOA | Set GPIOA pin 5 as Alternate Function mode: 10 | GPIOA->MODER &=~ 3<<Y\*5  GPIOA->MODER |= 2<<Y\*5 |
|  | GPIOA Alternate Function Selection: AF1(TIM2/TMI5)  AFRL\_5[3:0]=0001 | GPIOA->AFR[0] = (pin%4)<<(4\*LED\_PIN); |
|  | 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/2 | TIM2\_CCR1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | Output as active high | TIM2\_CCER\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | Enable Compare and Capture output | TIM2\_CCER\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

1. **Firmware Programming**

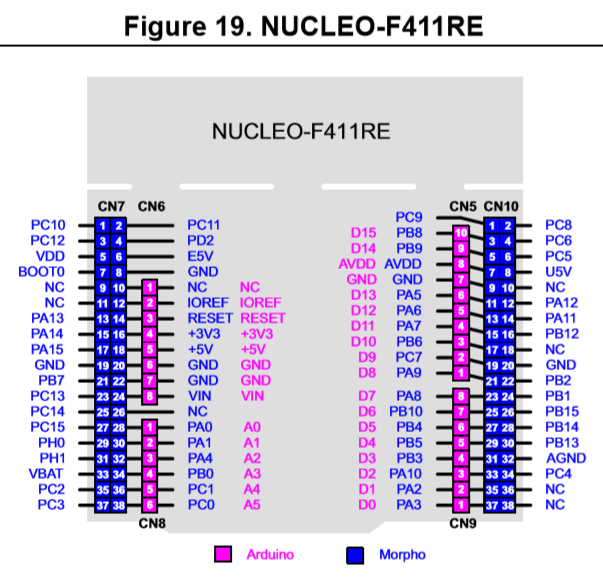
* Create a new project and name the project as ‘**Tutorial\_TIMER\_PWMout’**.
* Give PWM output of 0%, 50%, 100% of 1kHz PWM
* Check the brightness of LD2 (PA5) for each different Duty ratio
* Change to 50% Duty ratio of 10kHz frequency PWM

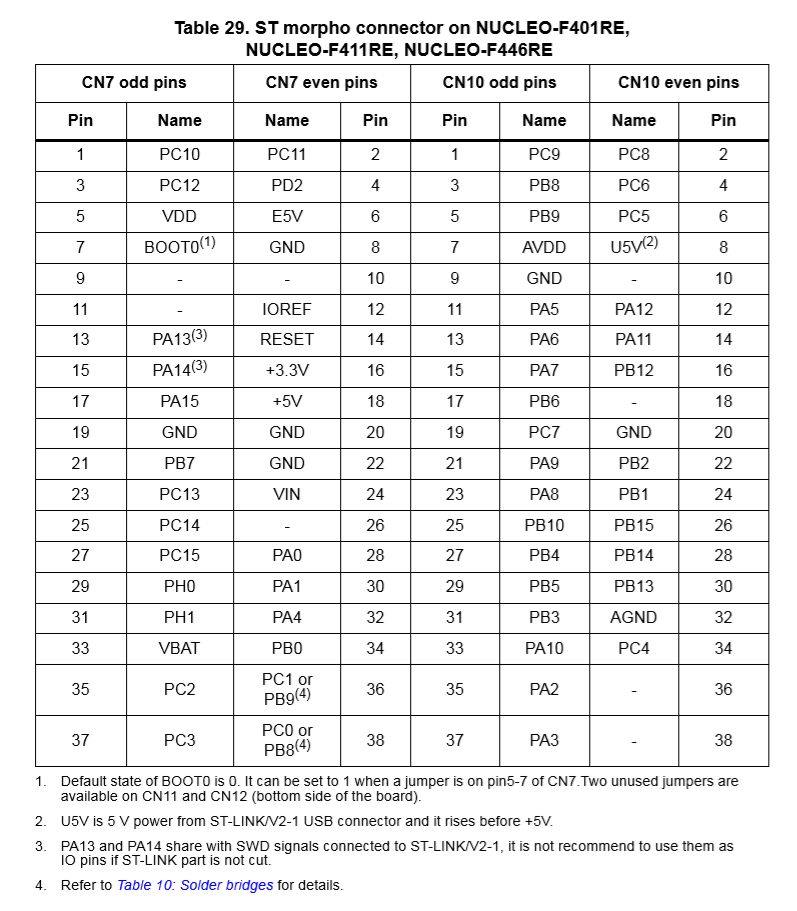
텍스트이(가) 표시된 사진

자동 생성된 설명

**Appendix**

1. **Pin Configuration of NUCLE-F411RE**



****

1. **Timer GPIO pinout for STM32f411**

