**LAB: General Purpose Timer**

**- RC Servo Motor Control with PWM -**

**I. Overview**

In this lab, you will learn how to use general purpose timers of MCU to control PWM signals with user defined PWM period and duty ratio. Also, you will learn how to change the angle of an RC servo motor using the PWM signals.

Objectives of this lab are learning how to

* Read and configure registers of General-purpose timers(TIMx)
* Generate PWM signals
* Control angle of RC servo motor

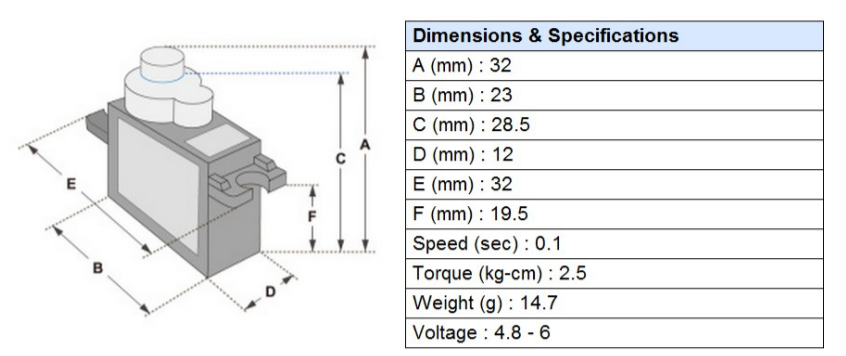
**Preparation**:

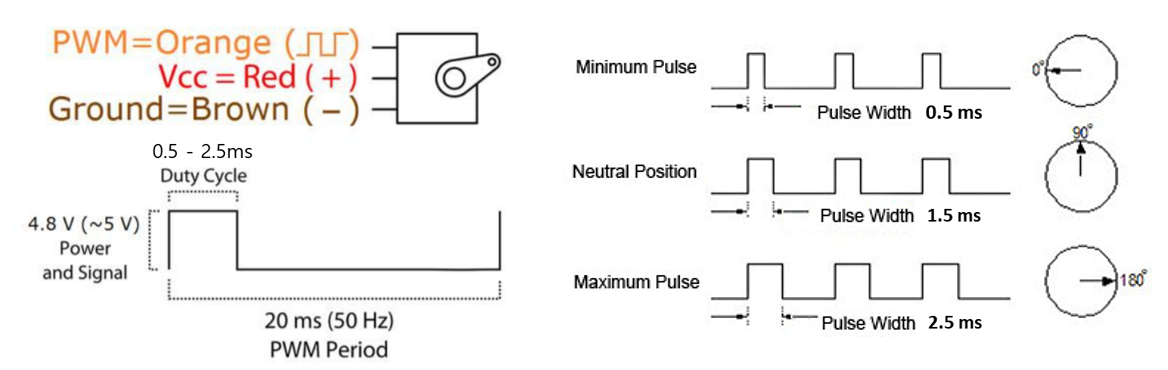
* You need to study the following registers: Timer(Advanced and General Purpose, TIMx) in ‘STM Reference Manual’
* User defined APIs for GPIO and External Interrupt control

**II. Pre-Lab**

**A. RC Servo Motor (SG90)**

* An RC servo motor is a tiny and light weight with high output power. It is used to control rotation angles, approximately 180 degrees (90 degrees in each direction).
* The angle of the motor can be controlled by the pulse width (duty ratio) of PWM signal. The PWM period should be set at 20ms or 50Hz. Refer to the data sheet of the RC servo motor for detailed specifications.





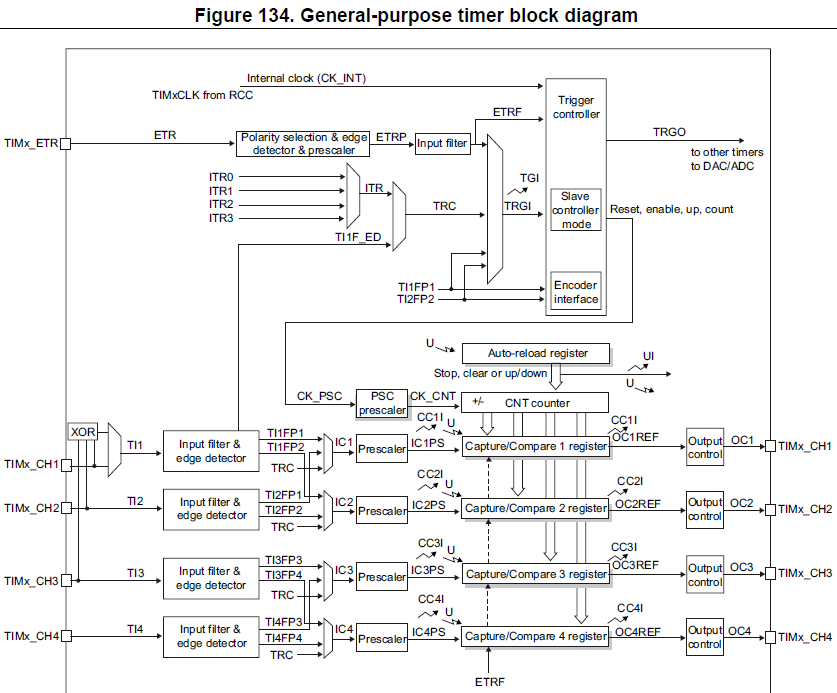
**Figure 1. Operation of Servo Motor**

**B. Timer Registers: TIMx\_**

* List TIMx registers for this LAB

|  |  |  |
| --- | --- | --- |
| Class | Register Name | Description |
| TIMx | TIMx \_CR1 | TIMx control register 1 |
|  | TIMx \_CCRn | TIMx capture/compare register for nth channel |
|  | TIMx \_CCMRn | TIMx capture/compare mode register for nth channel |
|  | TIMx \_PSC | TIMx prescaler register |
|  | TIMx \_ARR | TIMx auto-reload register |
|  | TIMx \_CCER | TIMx capture/compare output enable register |
|  | TIMx \_BDTR | TIM1(only) break and dead-time register |

* Block Diagram for General Purpose Timer



**C. Register Initialization**

Process of Timer(TIMx) register initiation

a) Timer Interrupt from Output compare or Over/Underflow

|  |
| --- |
| **GPIO Pin setting**  1. RCC setting for GPIO  2. AF(TIMx) mode selection for Pin 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. Set CompareCapture value (**TIMx\_CCRy->CCR**) y=channel  6. Set Timer Output mode (**TIMx\_CCMR 🡪OCyM**)  7. Enable Timer DMA/Interrupt. **(TIMx\_DIER🡪UIE)** or **(TIMx\_DIER🡪CCyE)**  8. Enable CompareCaptureOutput (**TIMx\_CCER🡪CCyE**)  9. Enable counter (**TIMx\_CR1🡪CEN**)  **NVIC setting**   * + 1. Enable TIMx Interrupt: NVIC\_EnableIRQ(TIMx\_IRQn)     2. Set interrupt Priority NVIC\_SetPriority(TIMx\_IRQn,2) |

b) Timer Setting for PWM output

|  |
| --- |
| **GPIO Pin setting**  1. Set RCC for GPIO  2. AF(TIMx) mode selection for Pin\_y in GPIOx  **PWM Timer setting:**  1. Enable Timer peripheral Clock (**RCC\_APB1ENR**)  2. Set Counting direction (**TIMx\_CR1🡪DIR**)  3. Set PWM Output mode (**TIMx\_CCMR 🡪OCnM**)  4. Set Timer clock Pre-scaler value (**TIMx**\_**PSC🡪PSC[15:0]**)  5. Set Auto-reload value (**TIMx\_ARR->ARR**)  6. Set CompareCapture value (**TIMx\_CCRn->CCR**) n=channel  7. Select Output Polarity (**TIMx\_CCER🡪CCyP**)  8. Enable CompareCaptureOutput (**TIMx\_CCER🡪CCyE**)  9. Enable counter (**TIMx\_CR1🡪CEN**) |

**1. GPIO Pin Initialization**

* Find GPIO pins for TIM2 channel 1 to 4 of MCU.
* Set the GPIO pin as Alternate function (AF) for TIM2 / No pull-up & No pull-down / High speed / Push-Pull

**2. TIM2 Initialization:** Output Compare mode

* **TIM2\_CR1:** Enable auto-reload preload / Up-counter / Enable counter

|  |
| --- |
| *Register map goes here* |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Register** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Mask** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Value** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

* **TIM1\_CCMRn:** Output compare mode as PWM mode1, Output compare preload enable

|  |
| --- |
| *Register map goes here* |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Register** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Mask** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Value** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

* **TIM2\_PSC**: Set prescaler value to make timer counter clock as 1MHz.

|  |
| --- |
| *Register map goes here* |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Register** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Mask** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Value** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

* **TIM2\_ARR:** Set the ARR value of up-counting mode. What should be ARR for 1kHz PWM frequency?

|  |
| --- |
| *Register map goes here* |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Register** | **31** | **30** | **29** | **28** | **27** | **26** | **25** | **24** | **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Mask** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Value** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

* **TIM2\_CCER:** Output as active high and C/C output enable

|  |
| --- |
| *Register map goes here* |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Register** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Mask** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Value** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |  | 1 | 1 |

* **TIM2\_CR1:** Enable auto-reload preload / Up-counter / Enable counter

|  |
| --- |
| *Register map goes here* |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Register** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Mask** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Value** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

* **TIM2\_CCR1:** What should be CCR value for 50% duty ratio?

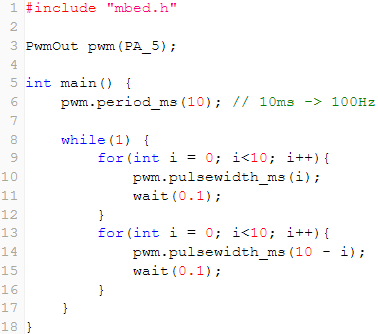
|  |
| --- |
| *Register map goes here* |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Register** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Mask** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Value** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**III. Tutorial**

**A. Mbed online compiler**

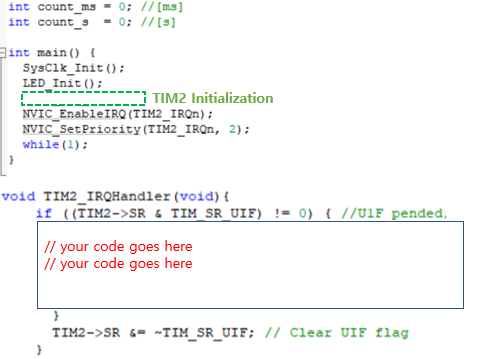
* Open <https://www.mbed.com/en/> and create new program ‘**Tutorial\_PWM**’.
* Write the following code on ‘main.cpp’. Check whether the platform is selected as ‘NUCLEO-F411RE’.



* Click on **Compile** button. Then, the binary files will be created and downloaded.
* Connect the MCU board to your PC. Copy and paste the downloaded binary files to NODE\_F411RE (E:) drive.
* Verify the performance. The brightness of LED should be changed.

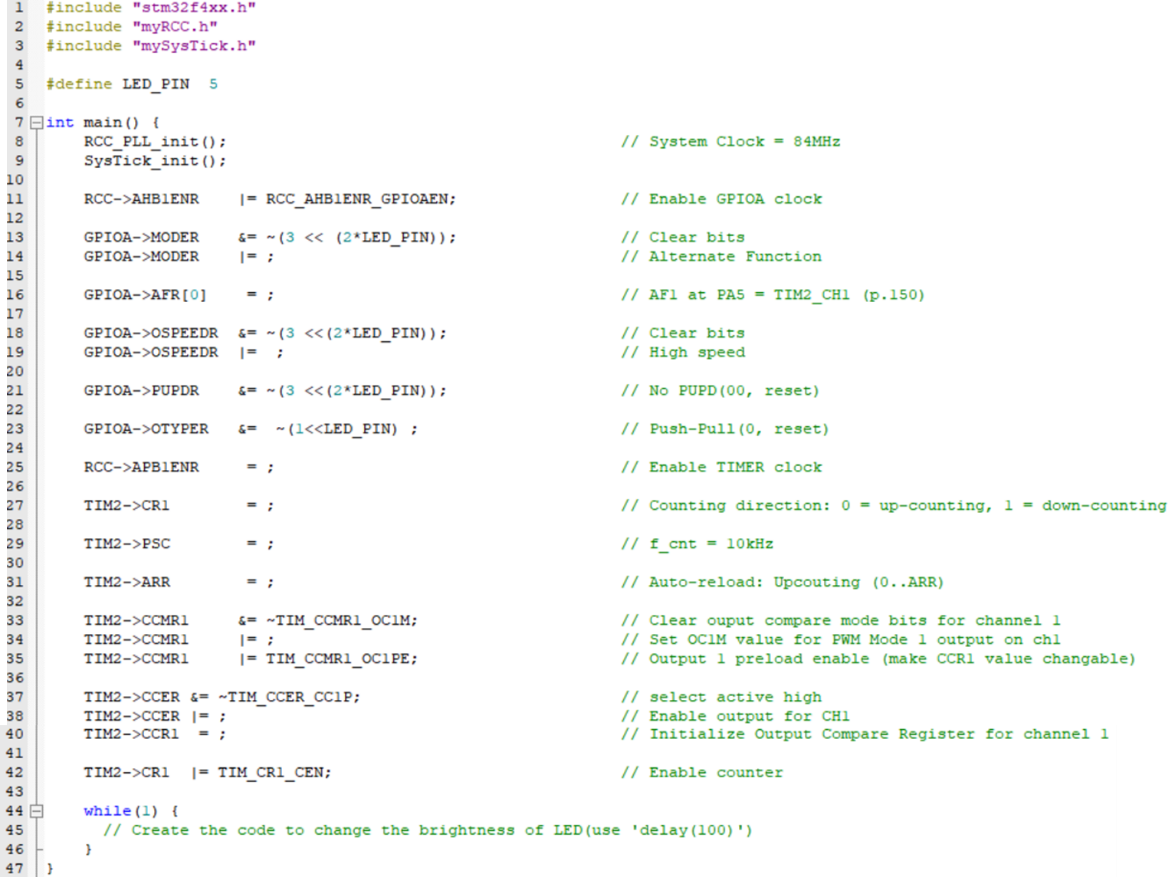
**B. Periodic LED Toggle using Timer**

* You can use general purpose timer(TIMx) interrupt to measure specific time interval or give periodic interrupt. It is different from SysTick.
* Create a program to toggle an LED in every second. Refer to the following code.



**C. PWM Register tutorial**

* Open the program ‘Keil uVision5’ and create a new project. Name the project as ‘**Tutorial**\_ **PWM\_LED**’. Refer to ‘Tutorial0\_Project Start’ if you need help.
* Create a new item called ‘main.c’ and copy-paste the given source code on ‘**main.c**’.

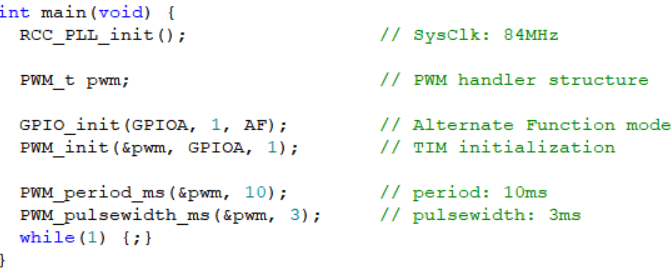


* Compile(F7) and flash(F8) the source code on to the MCU board.
* Verify the program by checking if the brightness of LED changes.

**IV. Exercise/Demo**

1. **Create your own functions for PWM initialization and control**

* Create your own function to initialize the clock setting, and timer register for pwm mode.
* Below are examples of the functions. Attach your codes in Appendix with sufficient comments
* Sample code)



|  |  |  |
| --- | --- | --- |
| **Source File** | **Example Code** | **Description** |
| myGPIO.h  myGPIO.c | #define AF 0x2  // modify GPIO\_init() to allow AF mode | Initialization for GPIO mode for Alternate Function (TIMx) |
| myPWM.h,  myPWM.c | typedef struct {  GPIO\_TypeDef \*port;  int pin;  TIM\_TypeDef \*timer;  int ch;  } PWM\_t; | *PWM\_t* is a structure type for initializing GPIO port, pin and the number of timer, channel. You can use this variable as a handler of PWM signal. |
| void PWM\_pinmap(PWM\_t \*pwm); | Match the GPIO port and pin number with the timer number and channel number. Refer **Appendix 2. Timer Pinout** |
| void PWM\_init(PWM\_t pwm, GPIO\_TypeDef \*Port, int pin) | Timer Initialization and enable.  Default: 83MHz source clk, 1MHz counter clock, 50% duty (PSC: 83, ARR: 999) |
| void PWM\_period\_ms(PWM\_t pwm, float period\_ms) | float period \_ms=0.01~10,000 |
| void PWM\_pulsewidth\_ms(PWM\_t pwm, float pulsewidth\_ms) | float pulsewidth\_ms=0.001~10,000 |
| void PWM\_duty (PWM\_t pwm, float duty) | float duty: 0.0~1.0 |

1. **Create a program to generate PWM signals**
2. 1kHz frequency PWM with 25% duty ratio and 50% duty ratio
3. 10kHz frequency PWM with 25% duty ratio and 50% duty ratio
4. Check the PWM signals with oscilloscope and attach the images of each signal
5. **Create a program to control a RC servo motor.**

* Make a simple program that changes the angle of the RC servo motor by pressing a push button.
* Use Port A Pin 1 as PWM output pin. What is the number of timer and channel on PA1?
* Initialize the PWM timer (period/duty ratio etc) as instructed in the datasheet of the motor.
* Increase the angle of RC servor motor from 0° to 180° each time you push the button. After reaching 180°, decrease the angle to 0°. Divide 180° into 10 intervals.
* Button input should be configured as External Interrupt. Use your own functions in the previous lab.
* You need to observe how the PWM signal output is generated as input button is pushed, using oscilloscope.

**V. Questions**

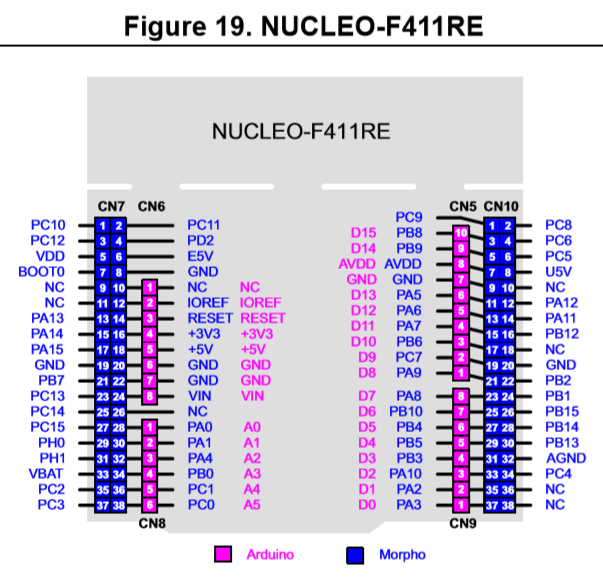
1. Derive a simple logic to calculate for CRR and ARR values to generate *x*Hz and *y*% duty ratio of PWM. How can you read the values of input clock frequency and PSC?

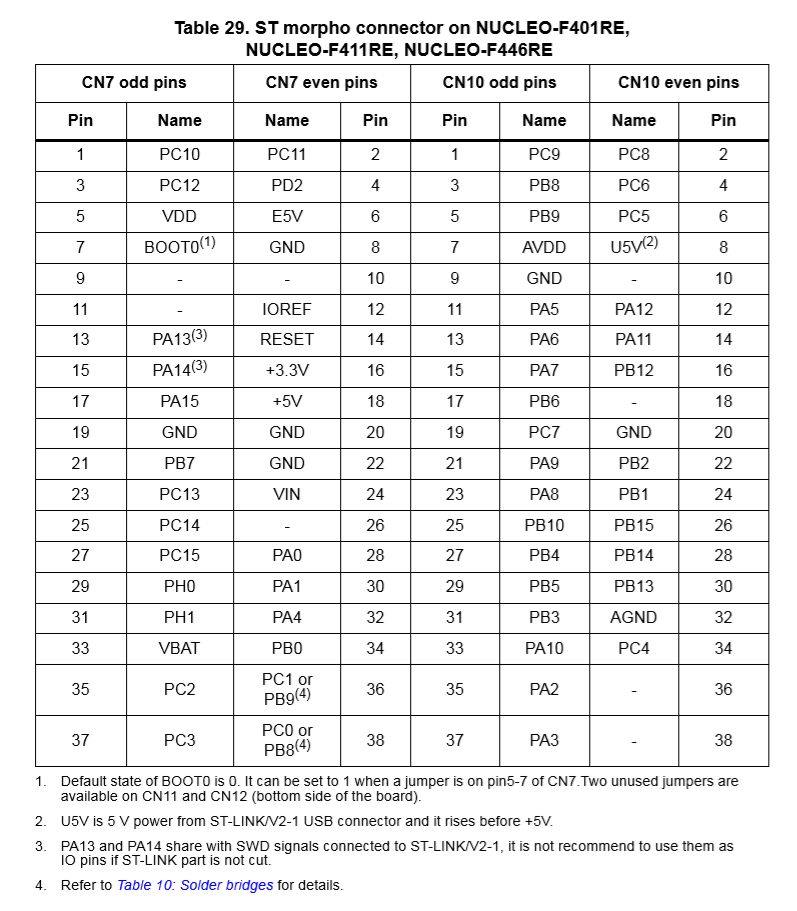
2. What is the smallest and highest PWM frequency that can be generated for Q1?

3. What is the major difference of advanced timer and general purpose timer? What is the master and slave control unit in Timer?

**Appendix**

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



****

1. **Timer GPIO pinout for STM32f411**

