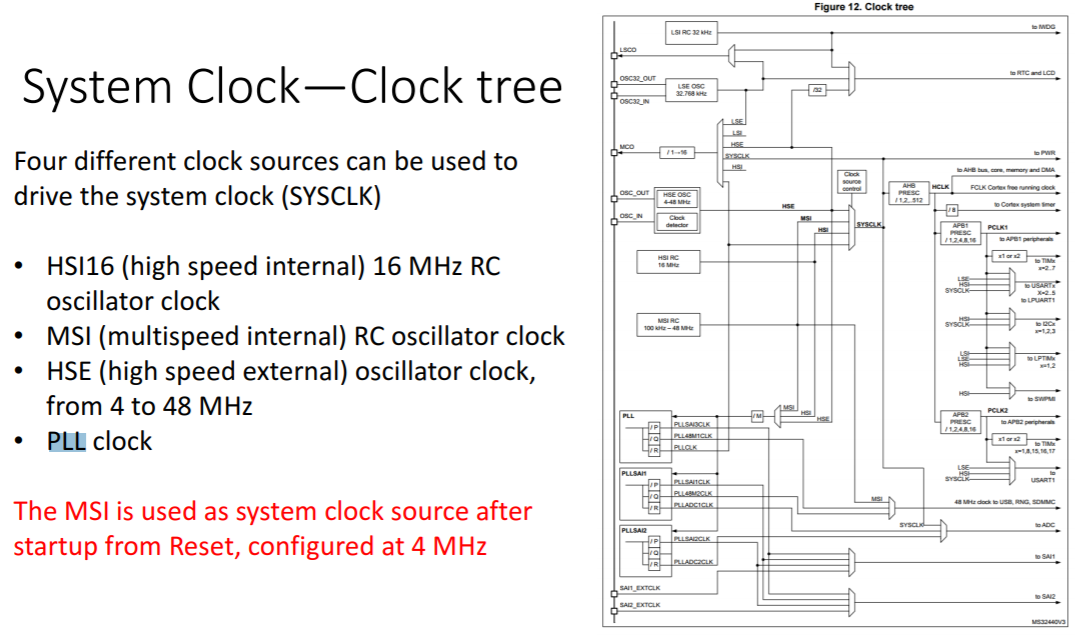
**Lab6 STM32 Clock and Timer**

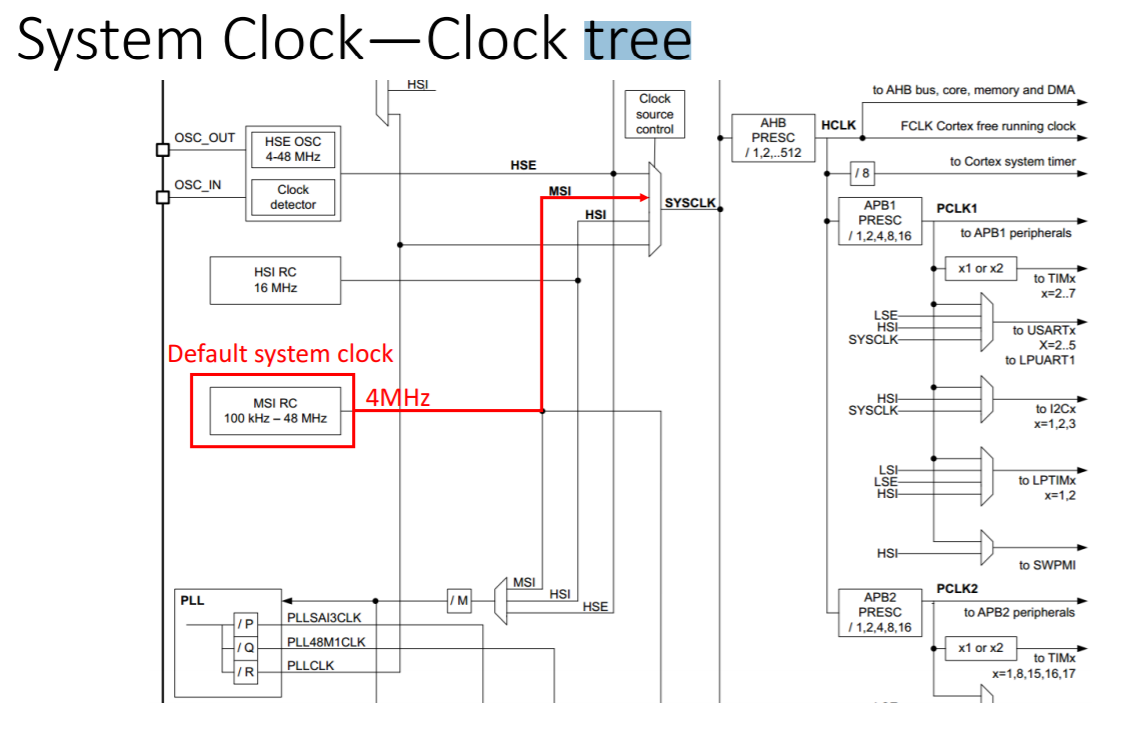
**1. Lab objectives 實驗目的**

* 瞭解 STM32 的各種 clock source 使用與修改
* 瞭解 STM32 的 timer 使用原理
* 瞭解 STM32 的 PWM 使用原理與應用

**2. Lab theory 實驗原理**

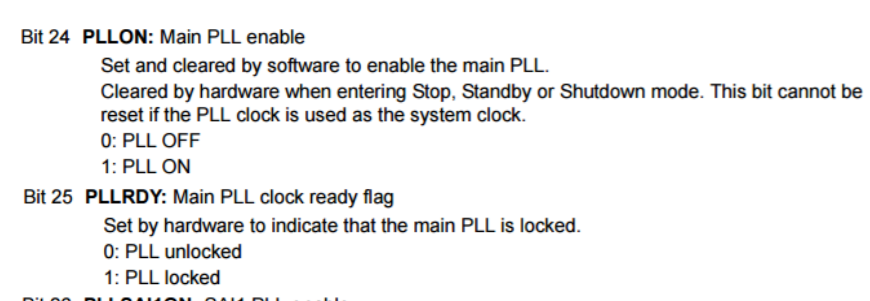
**2.1. Clock**





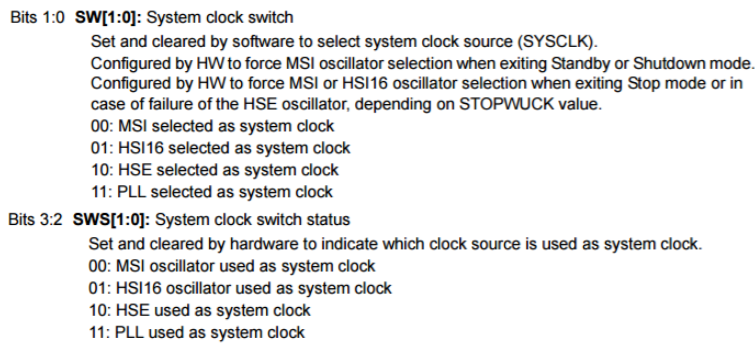
**RCC\_CR**

* XXXON:開啟哪一種系統clock，4選1(MSI,HSI,HSE,PLL)
* XXXRDY:狀態回報，是否已開啟完成
* 其中PLL還需要再開其他3個clock之一，當作參考源。

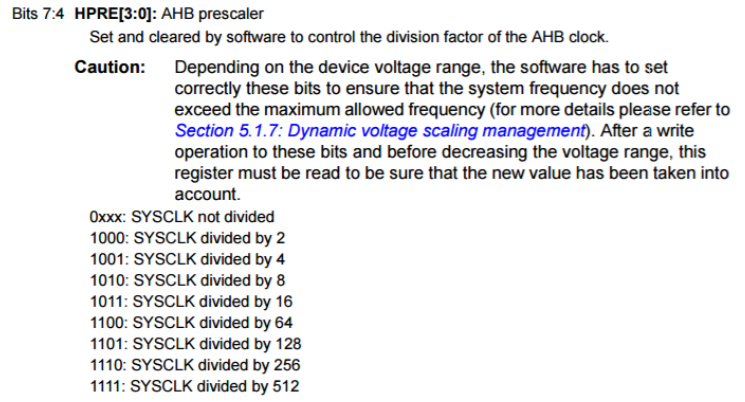


**RCC\_CFGR**

* SW:指定系統clock是用哪一種config，4選1
* SWS:狀態回報，是否已指定完成



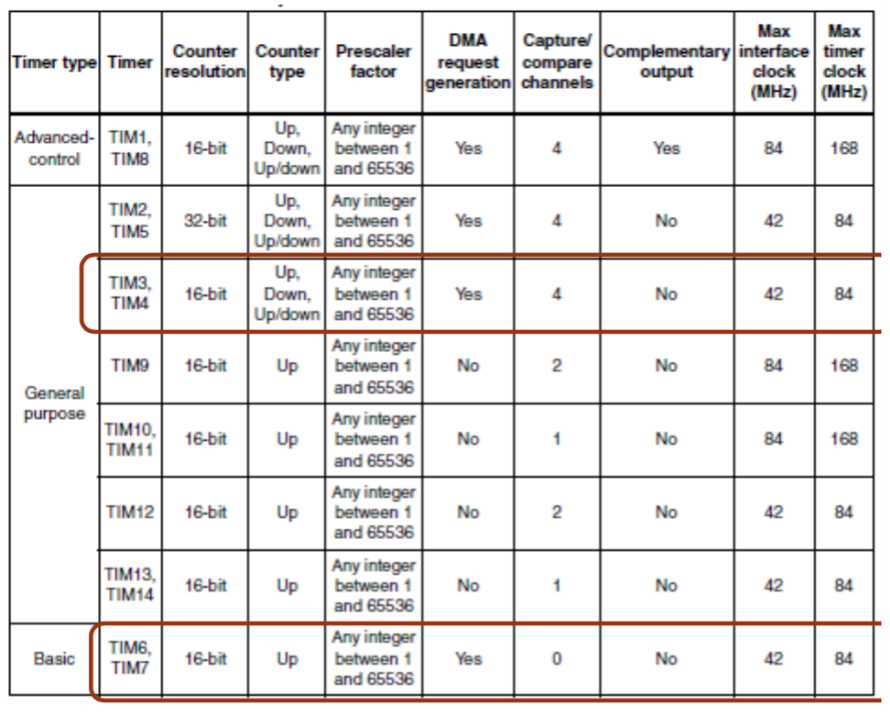
* HPRE: prescaler，選擇倍率



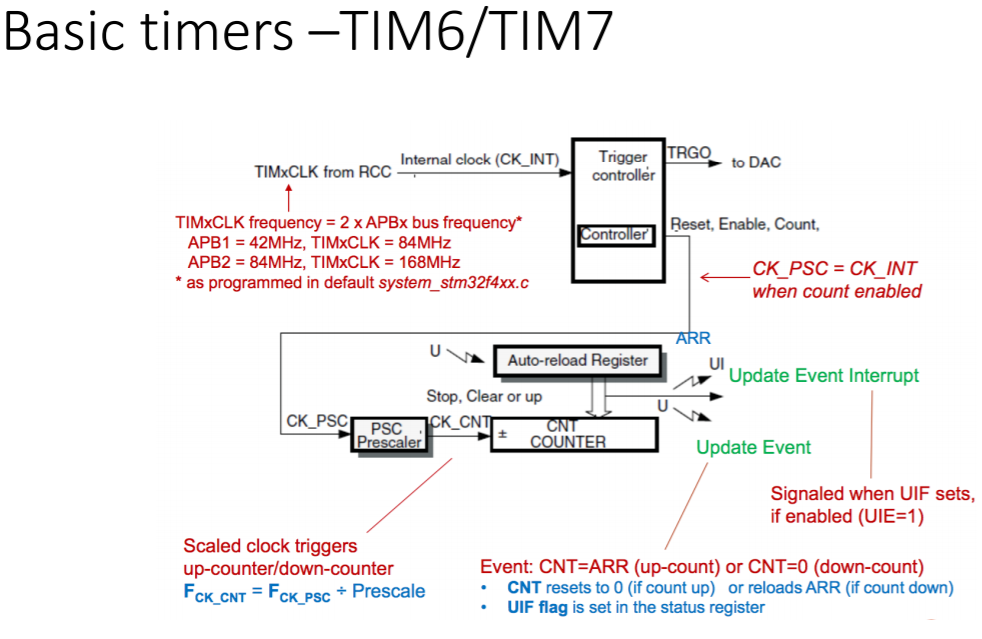
**如何開啟一個clock:**

1. RCC->CR四選一個，並等待RDY
2. RCC->CFGR 的SW一樣四選一，並等待SWS。
3. 設定RCC->CFGR 的HPRE，如果要更動倍率的話。

**2.2. Timer and Counter：Timer就是利用counter數經過幾次clock後，做出一些反應的設計。Timer有很多種：**

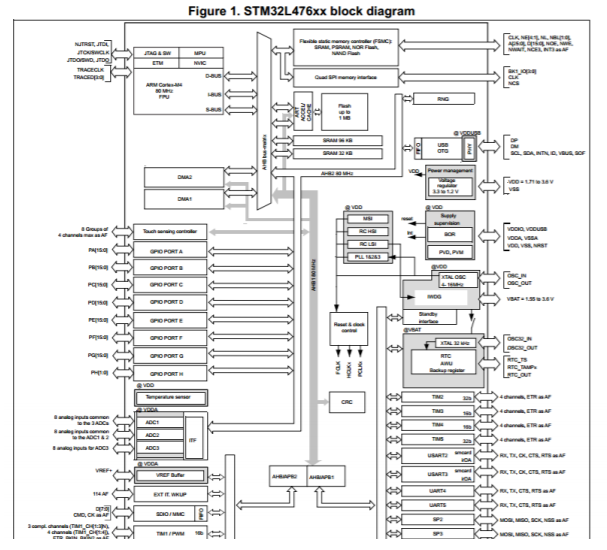


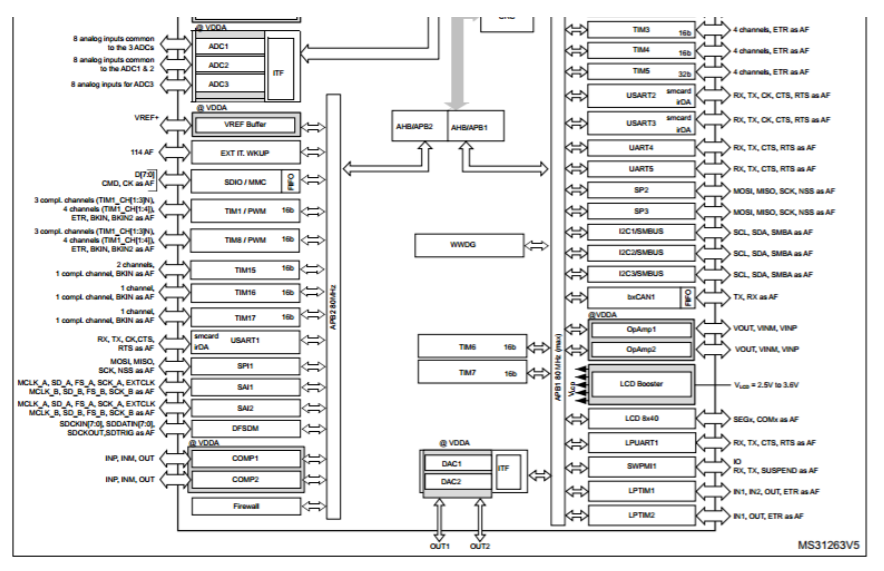
**最基本的Timer 6/7也是需要設定CR、ARR、PSC。**



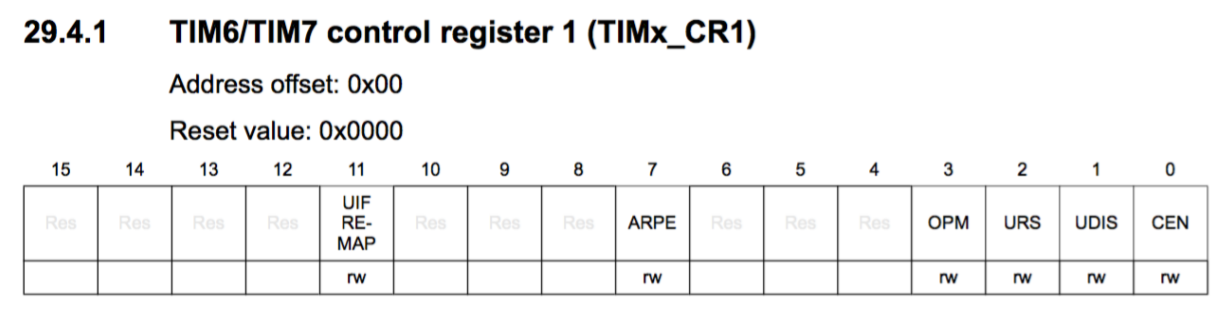
**3種clock狀態：原本板子clock (CK\_INT)、 開啟enable後clock (CK\_PSC)，最後除頻後，變成你用的所參考的counter (CK\_CNT)。**

**Timer大多定義在AHB1APB1下，記得去Enable。**

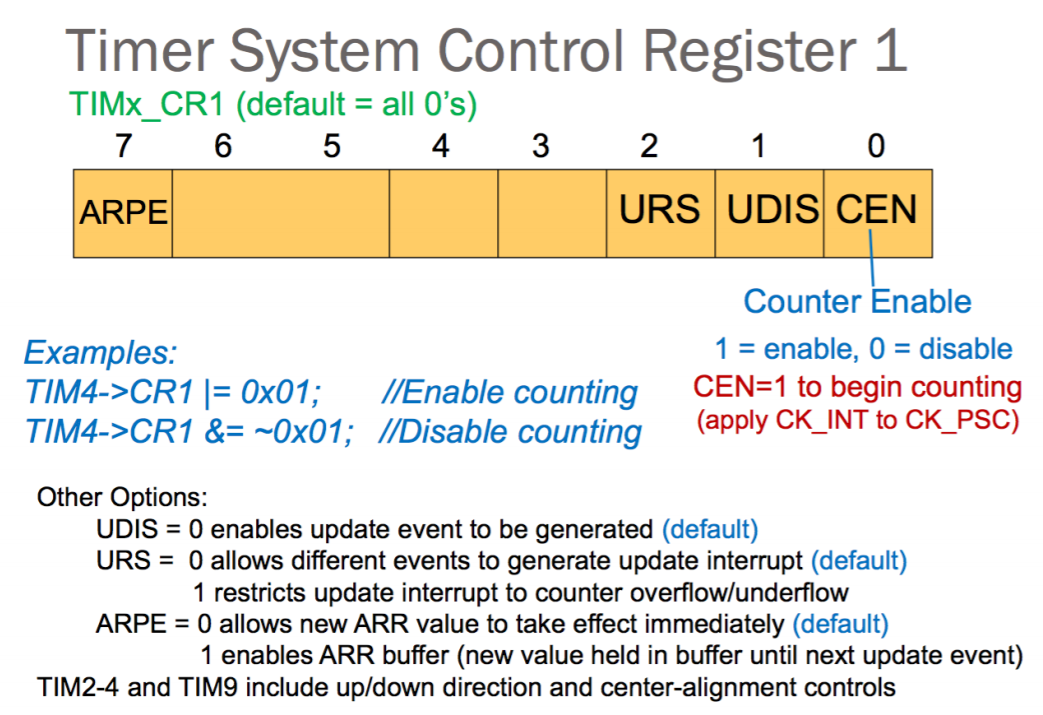




* **TIMx\_CR1 (timer control reg)：Timer設定控制**



**Example:**

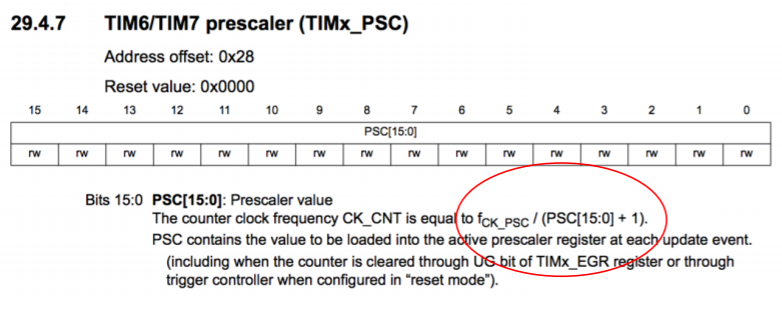


\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

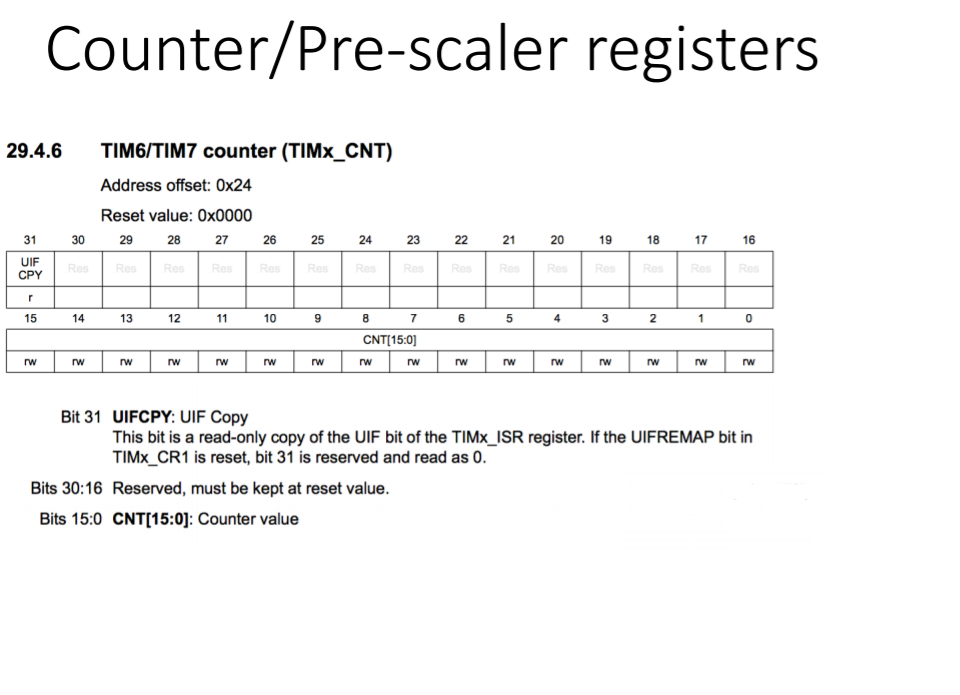
* **TIMx\_ARR (auto reload reg)：**看數到多少要給event。



* **TIMx\_PSC (Pre-scaler reg)：把原時脈做除頻or增頻(直接給值)**

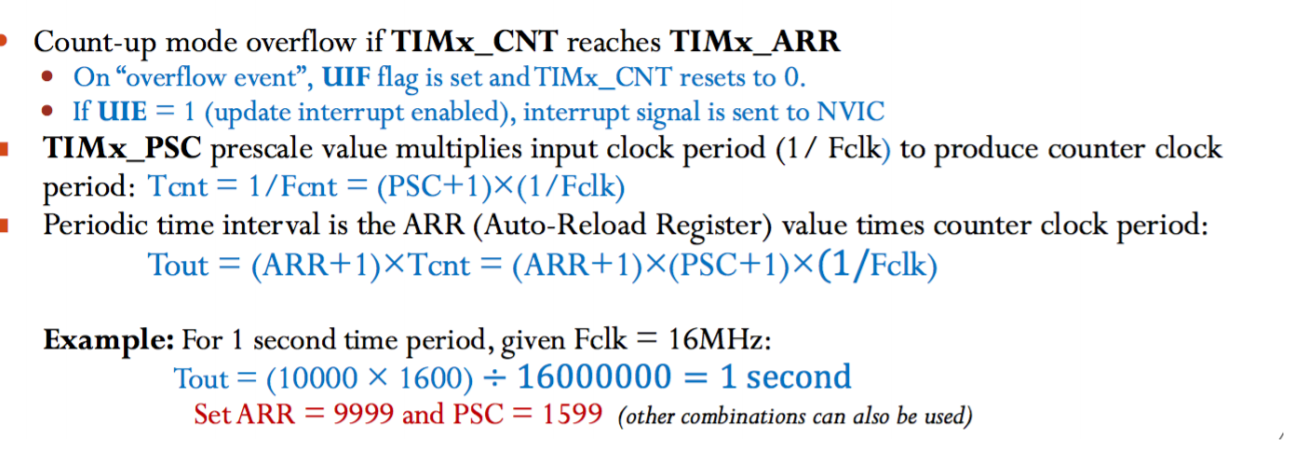


* **TIMx\_CNT: 只是讓你知道現在數到哪而已。**



**如何開啟一個timer:**

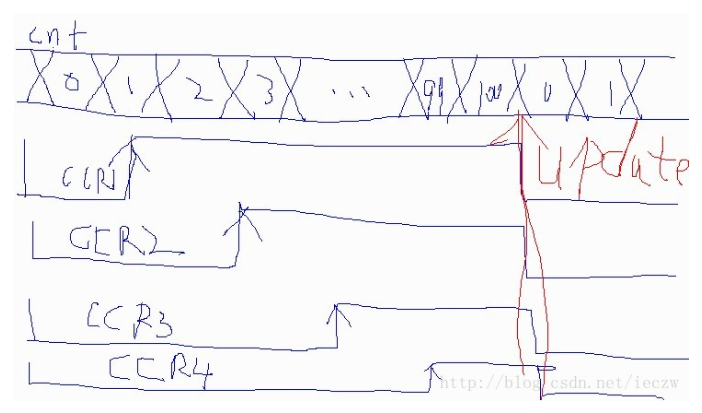
1. RCC的APB1ENR1設定成TIMxEN，打開TIMx。
2. 設定TIMx的PSC值，他決定多少cycle加一次counter
3. 設定TIMx的ARR值，他決定timer加到多少cycle會重數



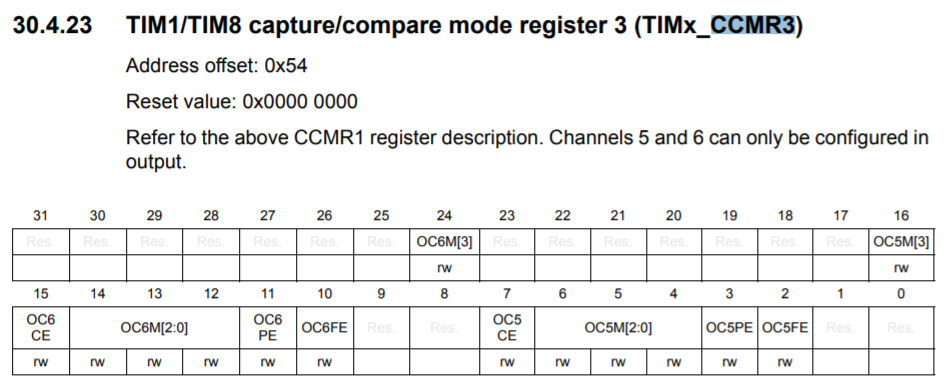
1. timer->EGR = TIM\_EGR\_UG :重新初始化
2. Start: TIMx->CR1 |= TIM\_CR1\_CEN; 設定TIMx的CR1變成CEN

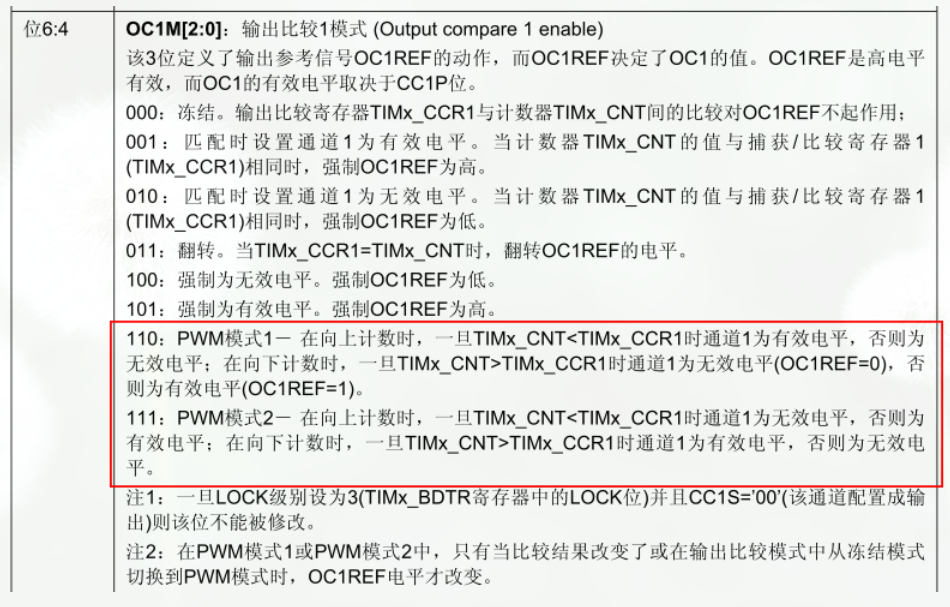
**如何設定一個PWM:**

1. **CCR1~4:一個timer能分出4個不同duty cycle的channel。CCR的值須與所用timer的ARR配合。**



1. **CCMR1~2: 控制輸出波的設定。TIMx\_CCMR1 控制CH1 和CH2，而TIMx\_CCMR2 控制CH3 和CH4**

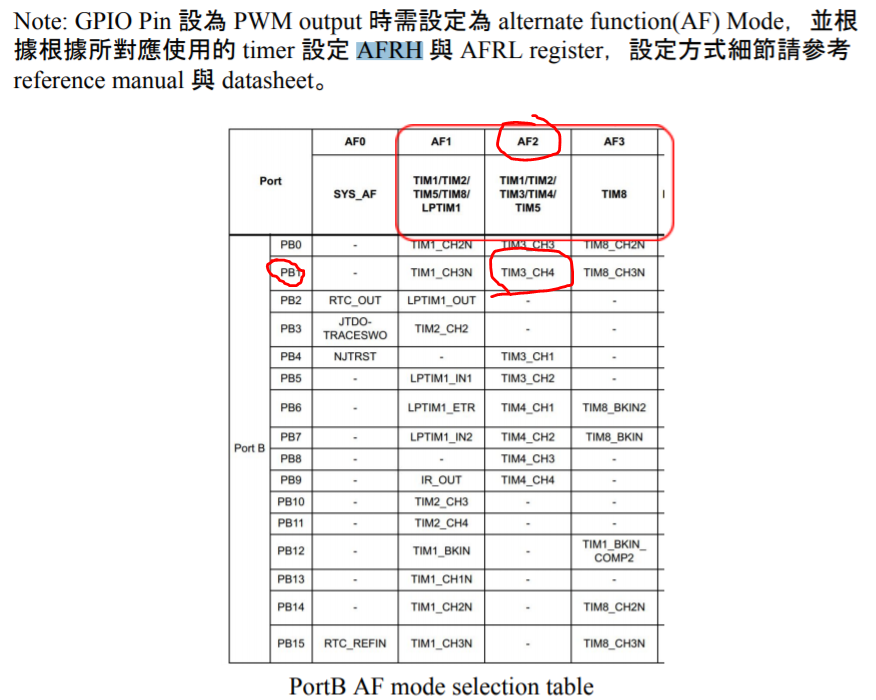




1. **CCER: 開啟輸出波(CCRx就選CCxE改)**

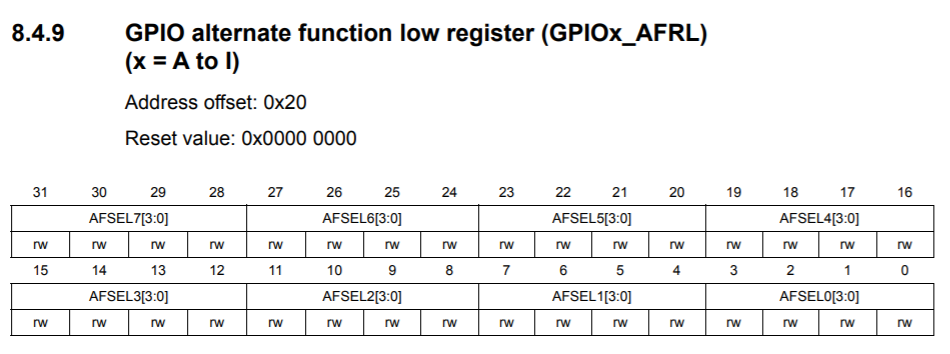


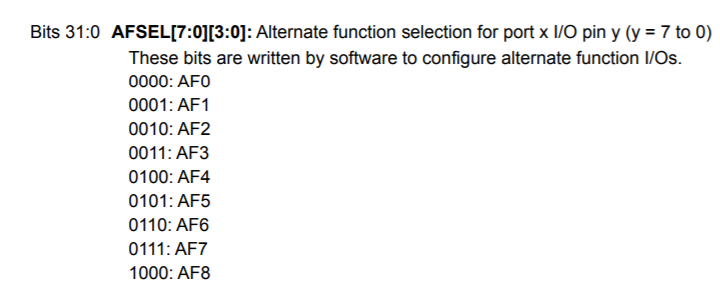
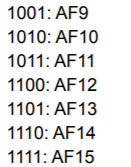
1. **AF Mode: (ex: output為PB1時，綁定AF2加上TIM3的CH4)**



**註：For each GPIO pins 0 through 7 are set in AFR[0] which is called GPIOx\_AFRL. For GPIO pins 8 through 15 set the corresponding**

**4 bits for the desired pin in AFR[1] which is called GPIOx\_AFRH**

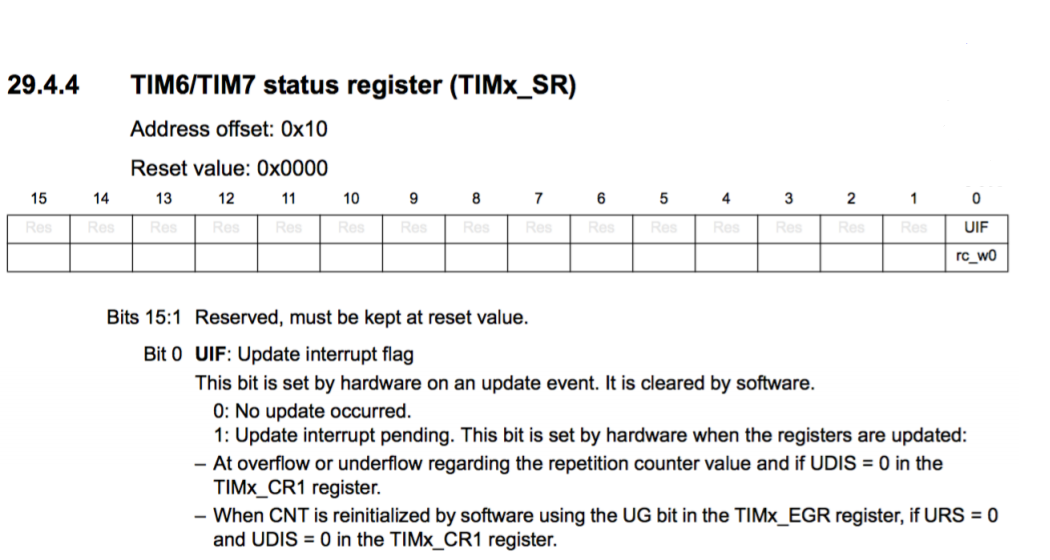




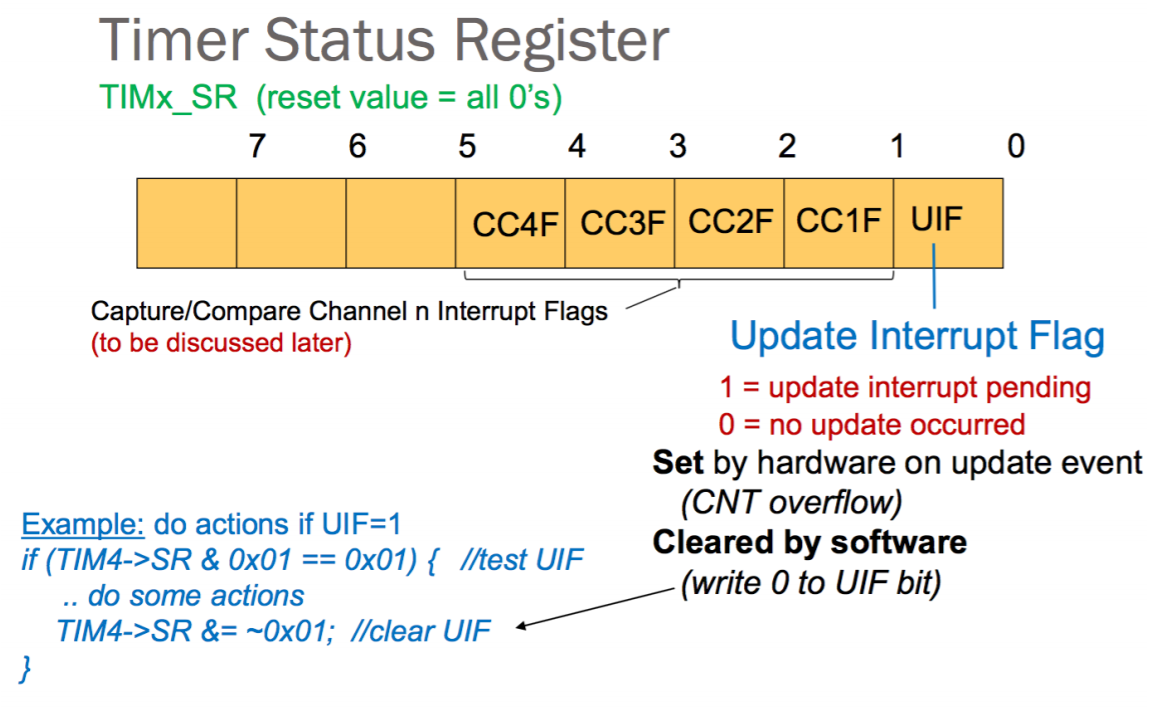
**GPIOB->AFR[0] = 0x2; 因為PB3要開AF2**

其他**Interrupt用：**

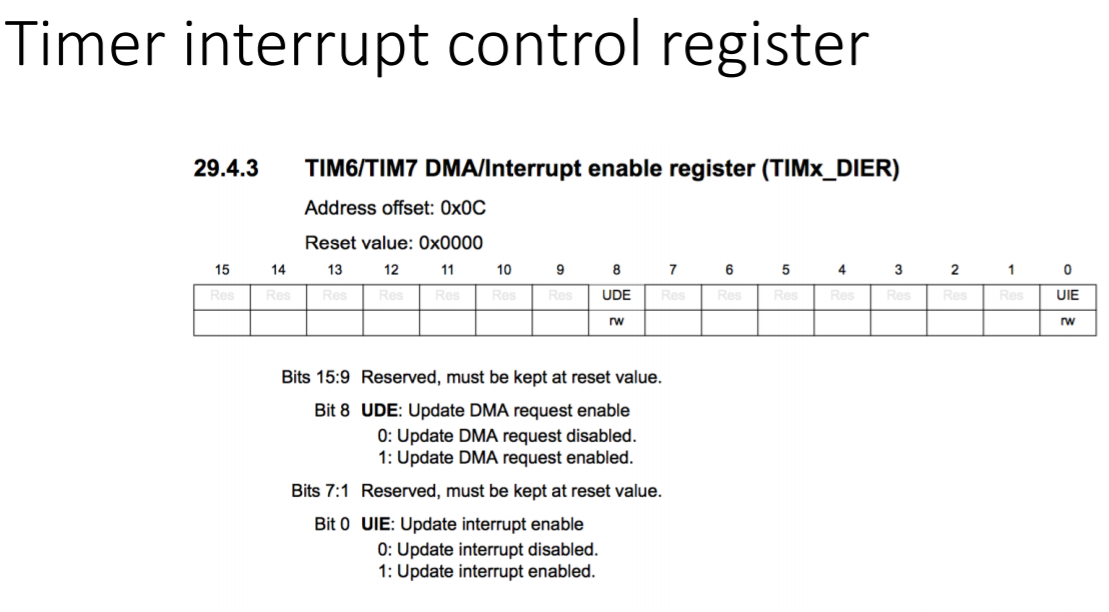
* **TIMx\_SR (status reg**)：**Interrupt**用。



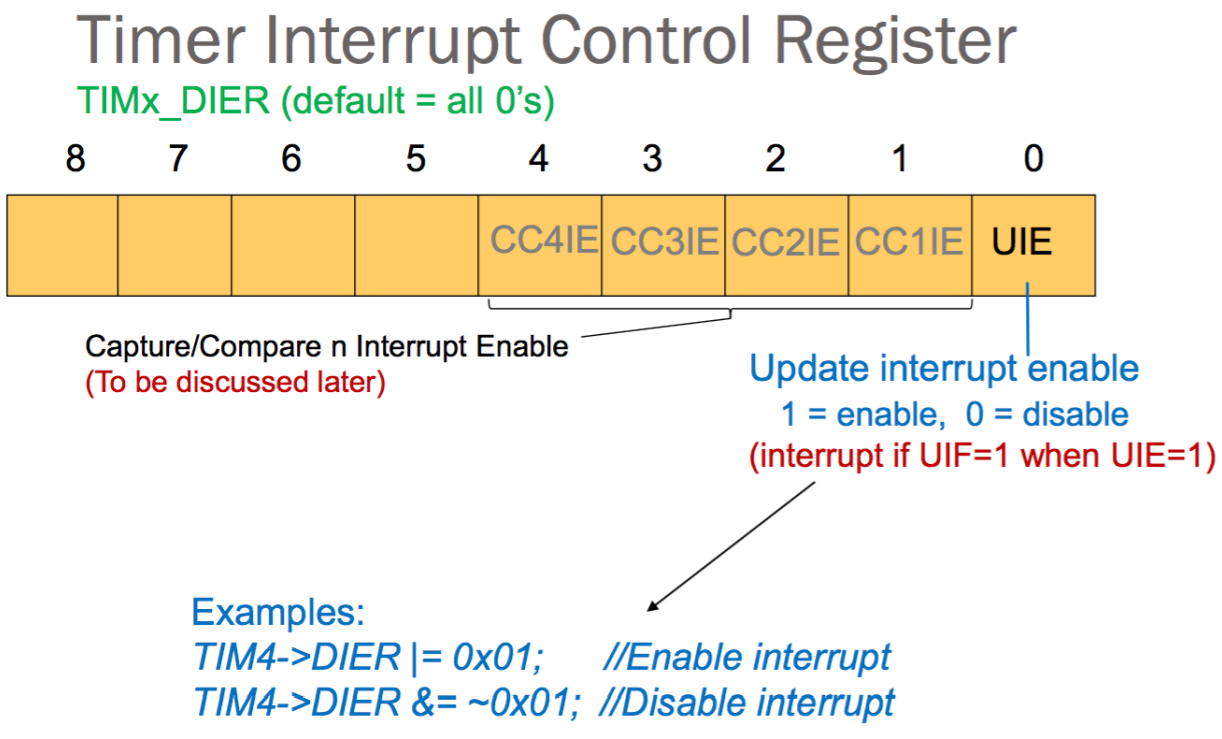
**Example:**



* **TIMx\_DIER (interrupt enable reg**)：**Interrupt**用。



**Example:**



**3. Steps 實驗步驟**

* 1. **Lab6.1: Modify system initial clock (20%)**

**實驗所要進行的目的是指定pll為system clk，並以msi作為pll參考的依據。**

**如同前一個lab，stm32的元件大部分都以stucter存在header中以直接使用。**

* **先開啟msi，和選定要開啟的range，開啟的方式是將enable與cr進行or。**

**(在進行開啟的動作後，應在while迴圈放入msirdy製作閘門，去檢查動作是否完成。**

* **在調整pll相關設定前，要先將pll disable，包括cfgr的sw設定。**
* **選擇pll參考的clk，在這邊選擇的是msi，選擇好後利用SET\_REG去將值放入 RCC\_PLLCFGR\_PLLSRC。**
* **根據頻率去調整係數，將GPIOB的ODR作為當前頻率的mode**

**(0b1 -> 1MHz, 0b11 -> 6MHz …)**

* **除1MHz特別將msirange改為4外，其他頻率皆可利用pllr/(pllm\*plln)推得**
* **pll設定結束後，將pllcfgr和pll enable，再將sw寫入**

**以上為pll和msi相關設定過程。除此之外，還須加上gpio設定、delay和button偵測的部分。這部分可從先前lab複製。**

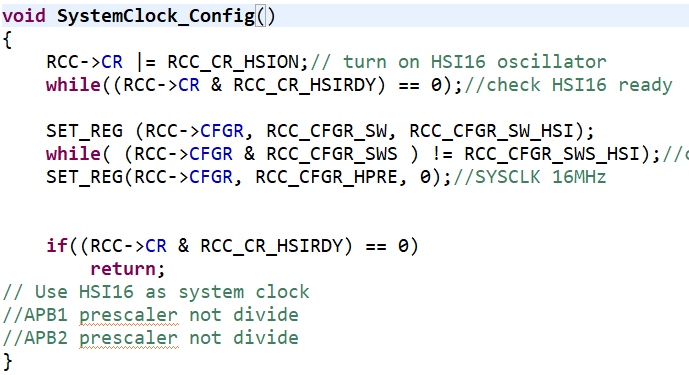
**GPIO方面，開啟ABC，A作為output操控led，B作為output以供 c檔內的flag，C則作為input接受button。**

**考量到高頻率時，程式執行過快，在一次push的過程便變化太多次，在每次push後加入一個小delay的動作(這個delay會根據當前頻率的不同，調整為不同的cycle數。**

* 1. **Lab6.2: Timer 計時器完成以下的 main.c 中的 Timer\_init() 與 Timer\_start(); 並使用 STM32 timer實做一個計時器會從 0 上數(Upcounting) TIME\_SEC 秒的時間。顯示到小數點以下第二位，結束時 7-SEG LED 停留在 TIME\_SEC 的數字。請使用 polling 的方式取得 timer CNT register 值並換算成時間顯示到 7-SEG LED 上。**
  2. **≤ TIME\_SEC ≤ 10000.00 (超過範圍請直接顯示0.00)**

1. **設定Clock (RCC\_CR\_xxxON, RCC\_CFGR\_SW\_xxx, RCC\_CFGR\_HPRE)**

* **RCC底下設定System Clock (預設: MSI = 4Hz)：**
  + **CR選擇clock來源(HSI16)**
  + **等待設定完成(HSIRDY)**
  + **CFGR切換到該clock來源的設定(SW\_HSI)**
  + **等待切換完成(SWS\_HSI)**
  + **如果需要，設定System Clock prescaler**

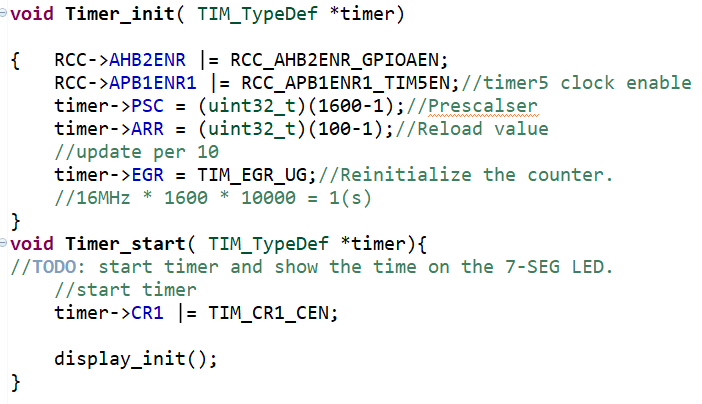


1. **設定Timer (APB1ENR1\_TIMxEN, TIMx\_PSC, TIMx\_ARR, TIMx\_EGR\_UG, TIMx\_CR1\_EN)**

* **Timer底下設定：**
  + **開啟APB1(x1) 或APB2(x2)**
  + **設定PSC (prescalar)，多少時脈加一次counter**
  + **設定ARR，counter加到多少時Reload**
  + **HIS (16 Hz) \* (1 / PSC ) \* ( 1 / ARR ) = 0.01秒**

**Counter每0.01秒會數完並Reload 1次。**

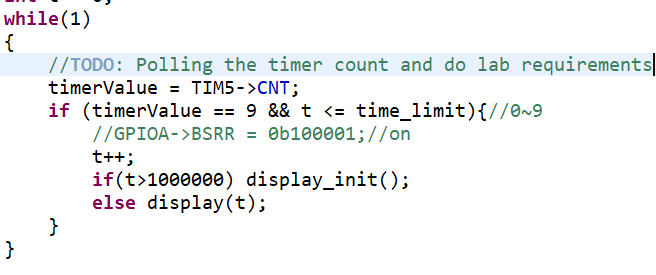
* + **(其他: DIR:倒數設定…等)**
  + **EGR: 重置Timer值**
  + **CR1設定CEN: 打開Timer**



* **利用polling TIMx->CNT取得counter值，用上次的lab顯示**

**只要在timer到達某一個特定值更新一下Max7219顯示值即可**

**(累加單位: 0.01秒，例如10秒時t = 1000 )**

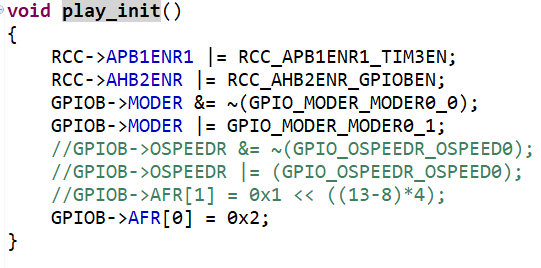


* 1. **Lab6.3: Music keypad**

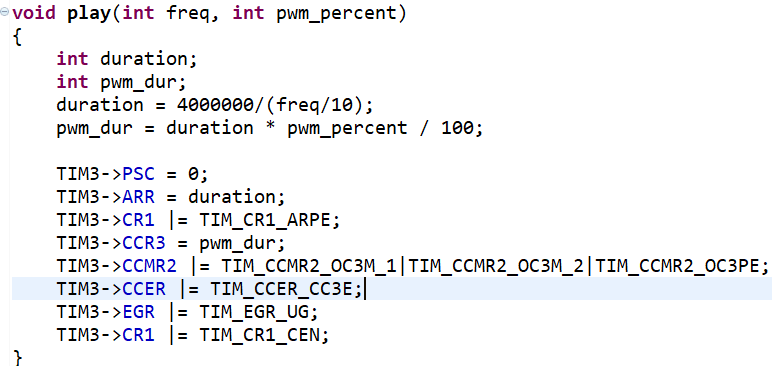
**請利用 timer 產生並輸出Duty cycle為 50% 的 PWM 訊號，並以 Lab5 中的keypad 為鍵盤，當使用者在按下不同 keypad 按鍵時產生特定頻率(參考下表)的PWM 方波給蜂鳴器，沒按鍵或按到沒功能的鍵時請不要發出聲音。**

**不同output port有不同的要求，例如pb1就要去設定TIM3底下的CCER, CCR4, CCMR2, 和GPIOB2底下的AFR[0]**

1. **設定PWM (GPIOx\_AFR[], TIMx\_CCER, TIMx\_CCMRn, TIMx\_CCRn)**

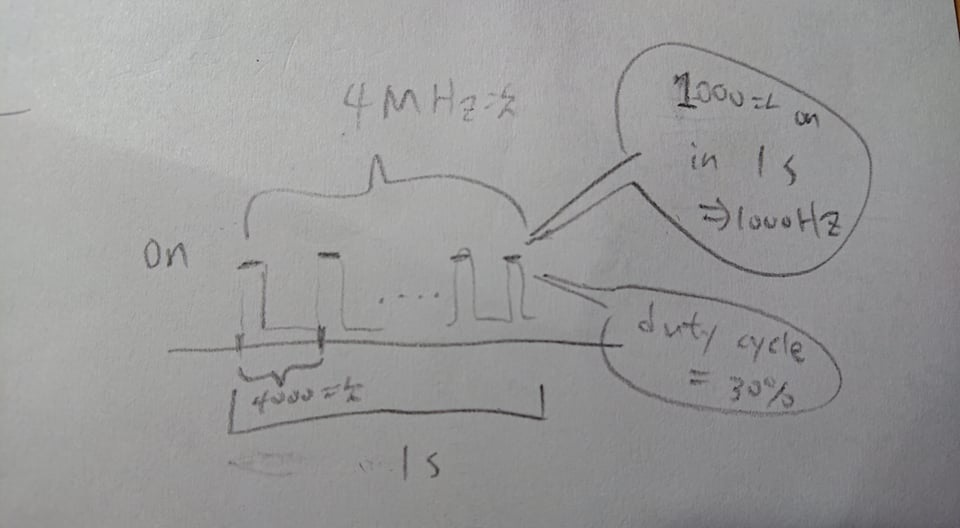
* **開啟APB1ENR1下的timer3 (TIM3EN)**
* **開啟output port(pb1)**
* **開啟AFRL的AF3**

1. **設定輸出PWM的Timer設定(比正常的多了CCER, CCMRn, CCRn要設**



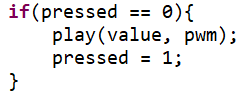
* **PSC=0，不要製造多餘的計算麻煩**
* **計算ARR和CCR3，使其為適當的值**

**例如在clock rate 4MHz下設定一個頻率1000Hz, duty cycle =30% 的PWM：設定ARR讓Timer每 4000(=4M/1000) 次就reload，再讓CCR3只有ARR數的過程中 (30) % 為高電位。**



* **CR1的ARPE區段設為1，使Timer會Auto reload & preload**
* **因為是用CCR3，所以要開啟CCER3**
* **CCR3的設定要在CCMR2改，去CCMR2把OC3M[2:0]設為PWM mode (110 or 111)**
* **其他就跟正常Timer一樣: EGR重置，CR1打開**

**值得注意的是對Timer的設定最好要用一個類似lock的東西保護，不要每次polling都去call setting function, 不然他會有類似啟動蜂鳴器會發出的雜音，而不是乾淨的聲音。**



* 1. **Lab6.4: Modify LED brightness**

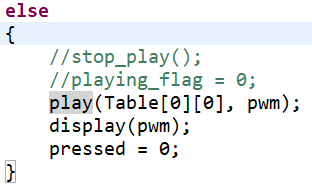
**在前一實驗中的 keypad 增加 2 個功能按鈕用以調整 PWM 輸出的 Duty cycle (範圍10%~90%，每按一次鍵調整5%)。使用 LED 去替換蜂鳴器。**

**設\*為加5%，#為減5%**

**把3.3改為不停輸出，只是按鍵從調頻率變成調duty cycle**

**一次按鈕只調一次：所以在沒進debounce時才改變duty cycle**





1. **Feedback 實驗心得或建議**

**Init pwn相關設定時，不能放在while loop讓他每次polling前都Init一次，似乎會來不及反應，導致沒有輸出聲音。但是之前的max7219\_init並不會這樣。可能是timer結構非常複雜吧**

**這次lab給的時間有點太短，因為太多新架構需要理解了**