MICROCHIP

Regional Training Centers

Section 6
Timer

MICROCHIP
Regional Training Centers

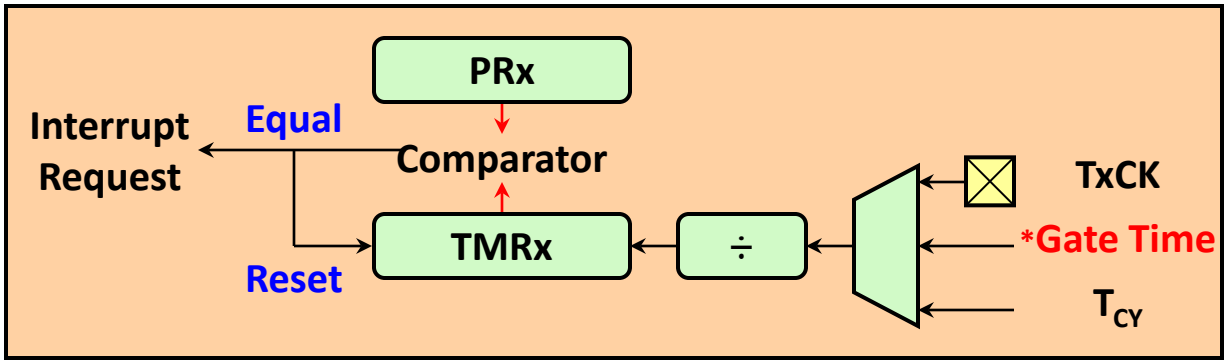
What's Timer ?

- Timer簡單的說就是一個會持續不斷的計算進入Timer中Clock數量的模組。
- Timer一般有兩種稱呼Timer或Counter。其實是一樣的東西。
  輸入的Clock時間未知:僅能得知計數到多少個Clock,稱為Counter。
  輸入的Clock時間已知:可以進一步換算出時間, 稱為Timer。
- 16 Bits MCU的Timer為遞增型,也就是會固定從"0"開始計數。
  模組可以設定計數到多少Clock後要通知CPU。
  例如:設定計數到1,000個Clock後, 通知CPU。
- 此處指的通知, 就是中斷旗標(Interrupt Flag) 。當數到期望值時, 中斷旗標會被模組設定為"1", 對CPU發出中斷請求(Interrupt Request)。
  - **如果此時中斷是除能的, 則必須由程式自行檢查事件是否發生。**
  - **如果此時中斷是致能的, 就會自動進入中斷服務常式。**

MICROCHIP
Regional Training Centers

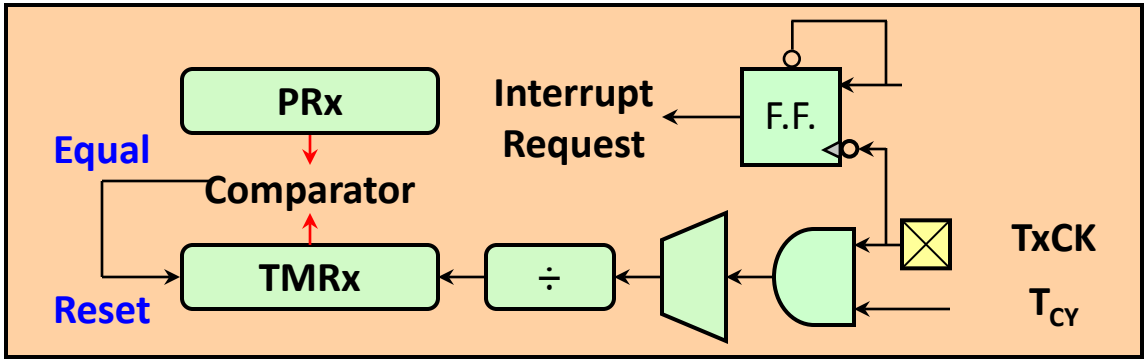
G.P. Timers's 計時/計數模式

- 16-Bits MCU的G.P Timers方塊圖, 如圖所示。
- Clock可以選擇內部或者由外部接腳輸入。經過預除器後, 送入TMRx。TMRx從"0"開始遞增, Comparator會不斷比較PRx與TMRx的值, 相同時發出中斷需求, 並且將TMRx歸零。
- 舉例來說,假設T_{CY}是1uS, 想要計算1mS的時間。則可將Clock設為T_{CY}, 預除器設為1:1, PRx設為1,000。
  如此一來, 每次數到1,000個Clock時(1mS), Timer就會清除TMRx, 設定中斷旗標, 發出中斷需求。
- 如果中斷有致能, 就會進入中斷服務常式。或者透過輪詢(Polling)中斷旗標得知。

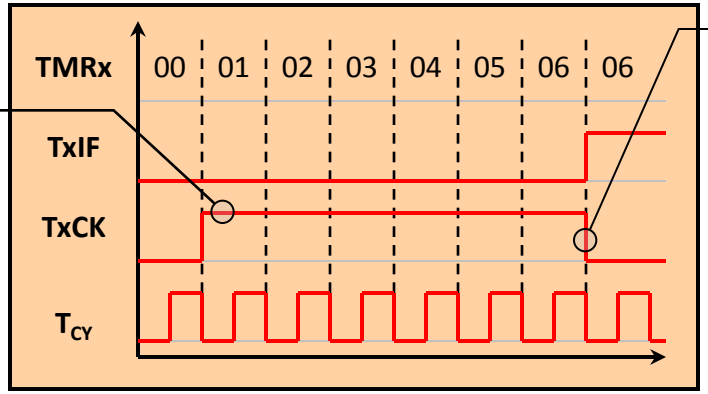


G.P. Timers's 閘控計時模式

- 16-Bits MCU的G.P Timers還有一個叫閘控計時(Gate Time)的特殊功能。可以用來計數外部輸入訊號的寬度。
- TxCK的輸入為 High時,T_{CY}可以被Timer計數,一直持續到TxCK的輸入由High變Low時停止。同時設定中斷旗標,發出中斷需求。



TxCK為High,AND Gate動作,T_{CY} 可進入TMRx。



負緣觸發正反器,設定中斷旗標,發出中斷需求。

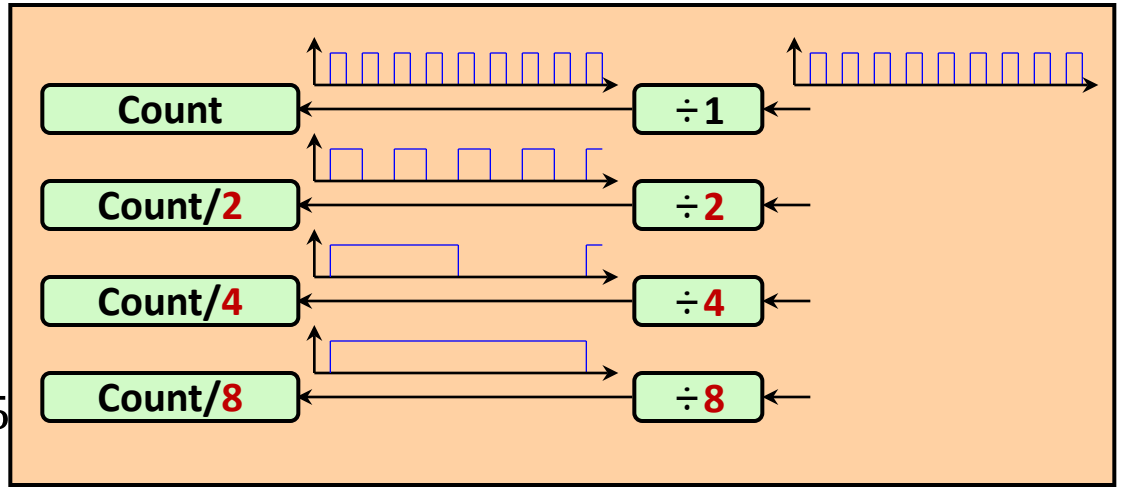
Prescaler & Postscaler

- 16-Bits MCU的G.P Timers是16Bits, 計數範圍0~65,535。如果需要的計數值超過時, 必須透過預除器修正。預除器可以用來擴大計數範圍。提高預除器比率, 減少計數值。
- 舉例來說, 如果T_{CY}為1uS, 要達成500mS的周期, PRx必須設定為500,000即500,000 * 1uS = 500mS。但此數值已經超過PRx能接受的範圍。此時可以透過預除器的設定, 減少計數值, 擴大計數範圍。

$$(2^n - 1) \le PRx = \frac{Peroid}{Clock \times Scale}$$

✗ $PRx = \frac{500mS}{0.25uS \times 1}, PRx > 65535$

✓ $PRx = \frac{500mS}{0.25uS \times 64}, PRx < 65535$



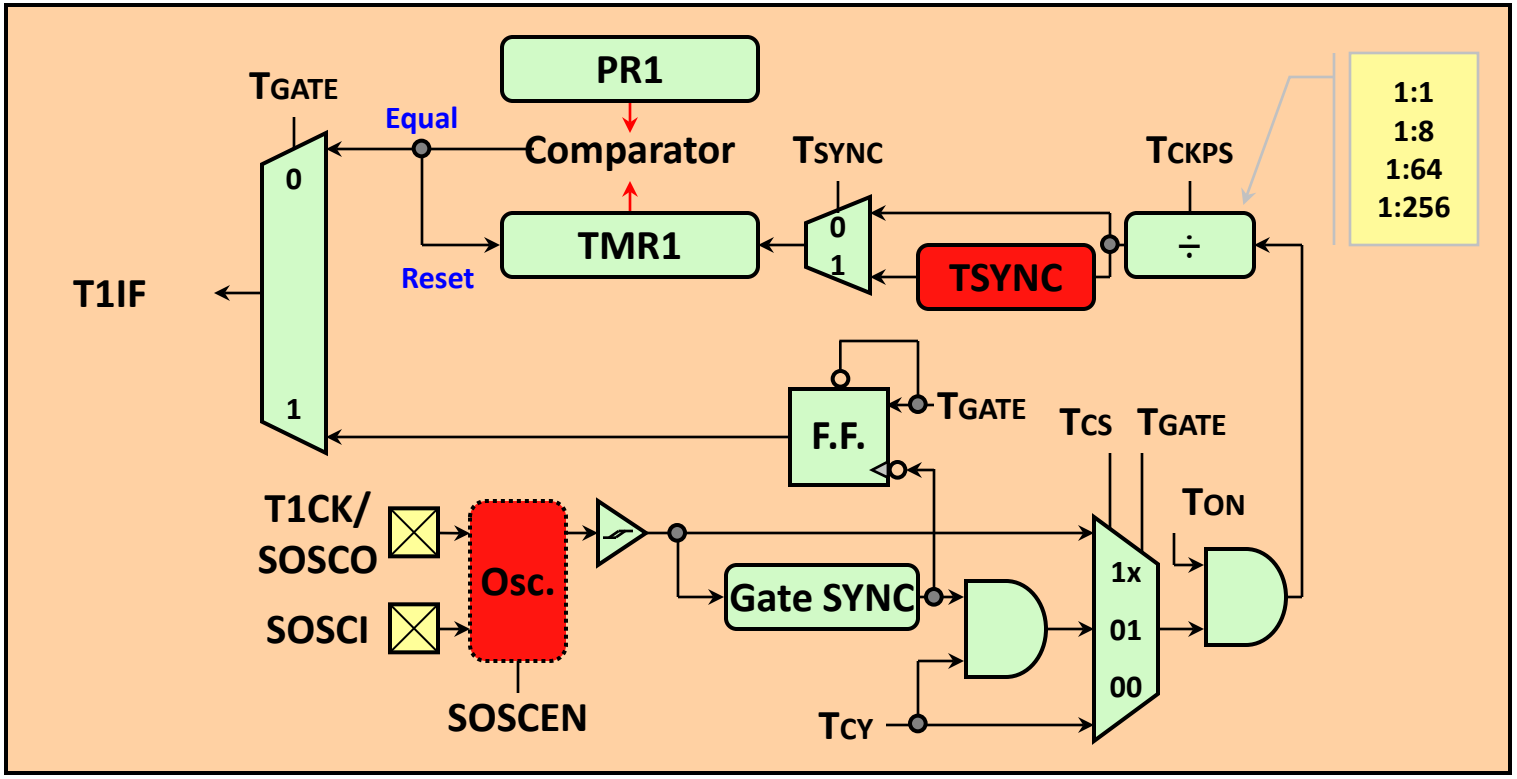
16-Bits Timers Introduction

- PIC24F16KA102具有3個General Purpose Timers/Counters。這3個Timers/Counters的功能與操作方式, 只有少許差異。
- Timer是16-Bits,所以計數範圍0~65,535。TMRx, PRx都不可超過。
- 預除器可以用來擴大計數範圍。舉例來說,如果預計TMRx要數1,000。代表必須有1,000個Clock進入TMRx。假設一個Clock 是1uS來看,則預計計數到1mS。
  此時如果把預除器設為除8(1:8),則變成Clock每8個才會有一個進入TMRx,意即TMRx數到1,000時,實際的Clock已經產生了8,000個,經過了8mS。

- **3個Timer的實際差異在哪？**

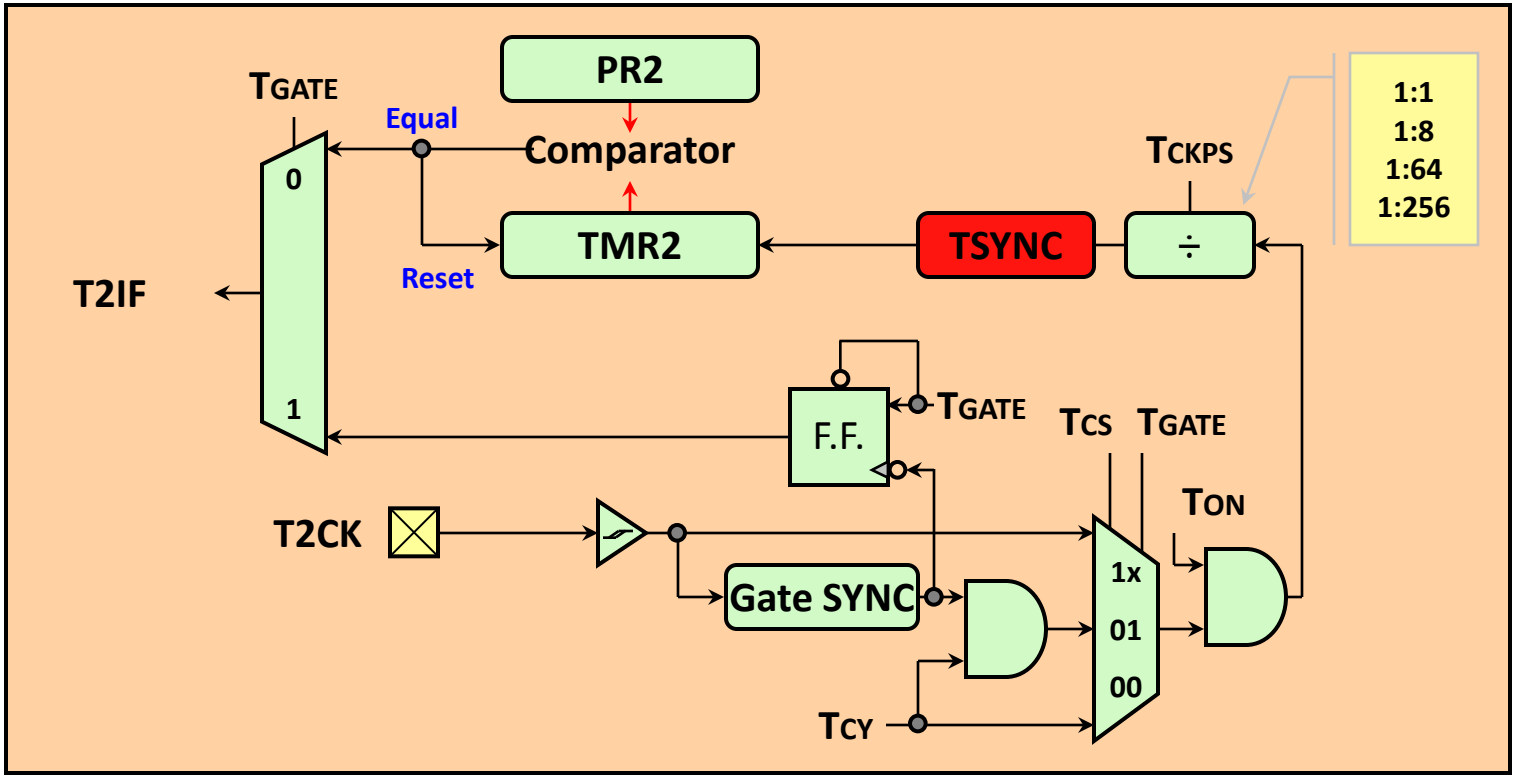
Timer1

- Timer1方塊圖如下。Timer1的特點是內建晶體振盪電路, 可以連接外部的Low Power Crystal, 如:32.768 KHz,做為RTCC的時脈輸入, 輸入訊號可選擇同步與否, 頻率不可大於N x ½F_{CY}。



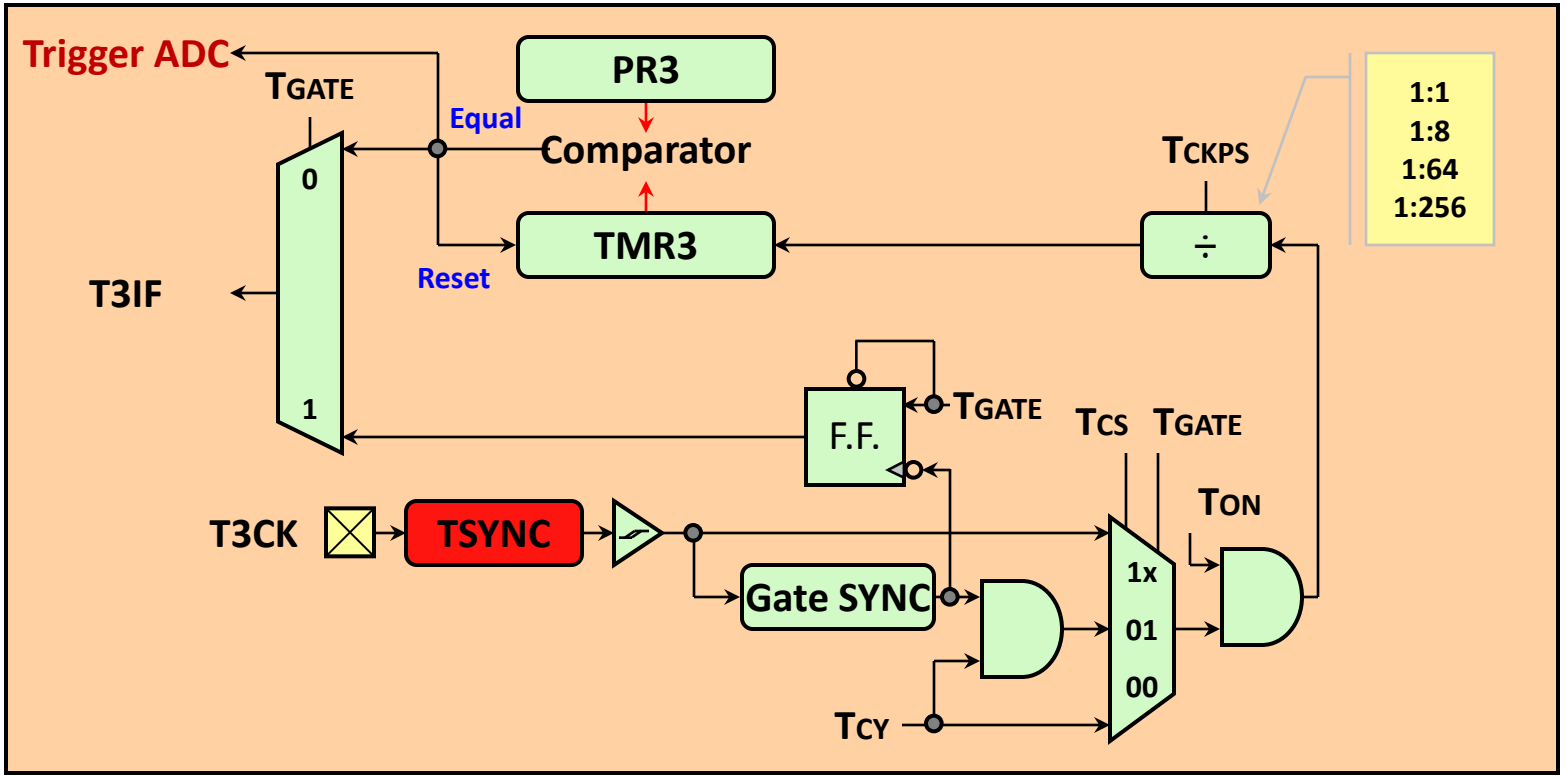
Timer 2

- Timer2方塊圖如下。Timer2與Timer1的差異,就是沒內建震盪器電路, 並且強制與系統時脈同步, 外部訊號的最高頻率不可大於N x ½F_{CY}。



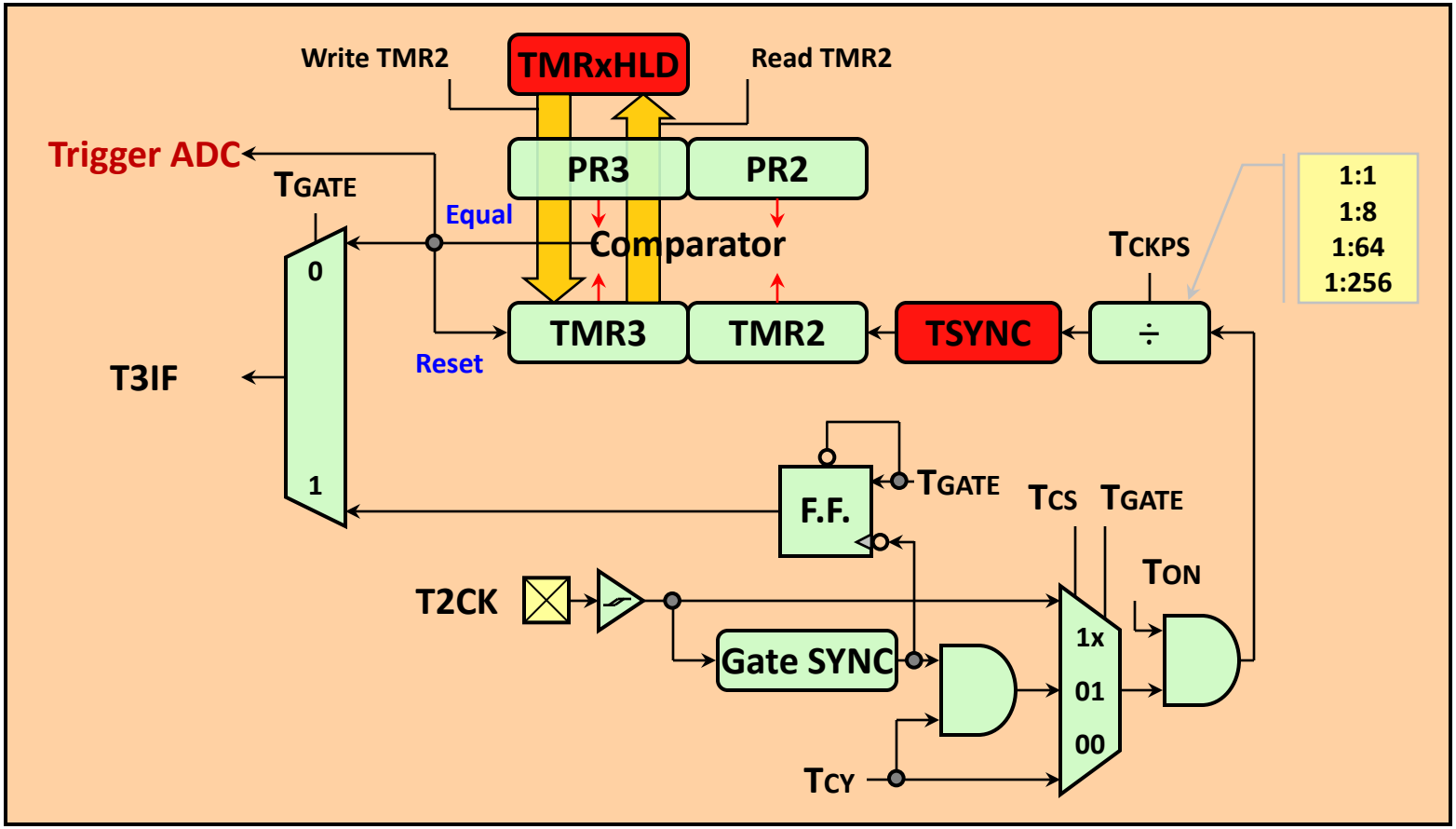
Timer 3

- Timer3方塊圖如下。 Timer3與Timer2的差異, 在於同步的點不同。Timer3外部訊號的最高頻率，不可以大於 ½F_{CY}。Timer3還可以作為ADC觸發轉換的觸發源。



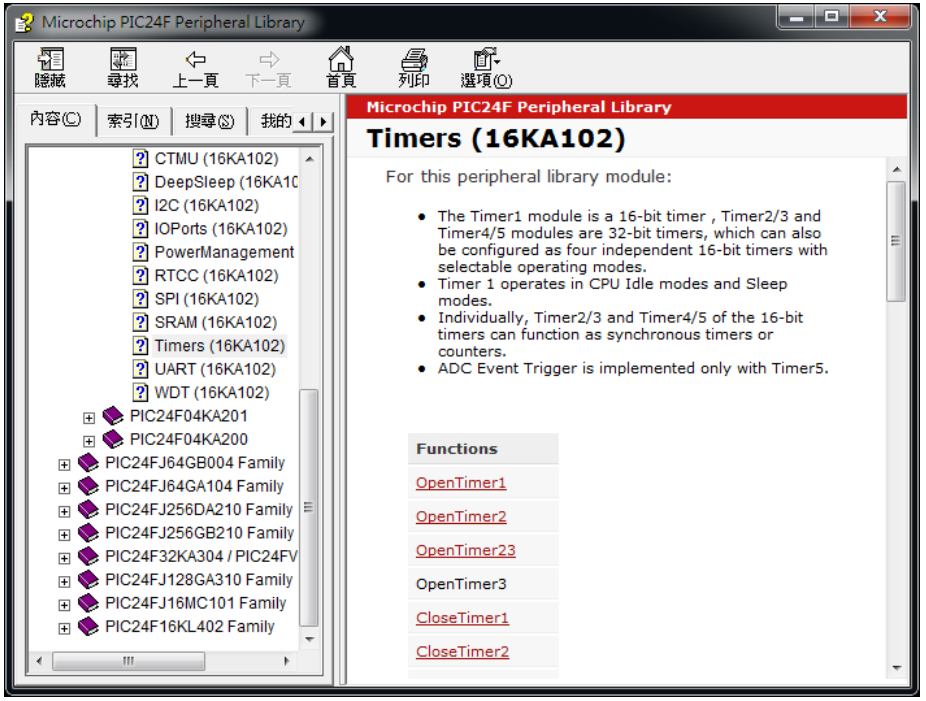
32-Bits Timer

- Timer23也可以組合成32-Bits的Timer。



MPLAB XC16對Timer的支援

- MPLAB XC16提供許多Timer Function可供使用。
  在C:\ProgramFiles\Microchip\mplabc xc16\vx.x\docs\periph_lib\ Microchip PIC24F Peripheral Library.chm 檔案中可以找到詳細的使用說明。
  OpenTimerx( );
  WriteTimerx( );
  ConfigIntTimerx( );
  …
- **以往我們都必須先閱讀每個Function的功能與使用方法,然後一個一個的寫到程式中。現在透過MCC可以減少許多繁複的初始化流程。**



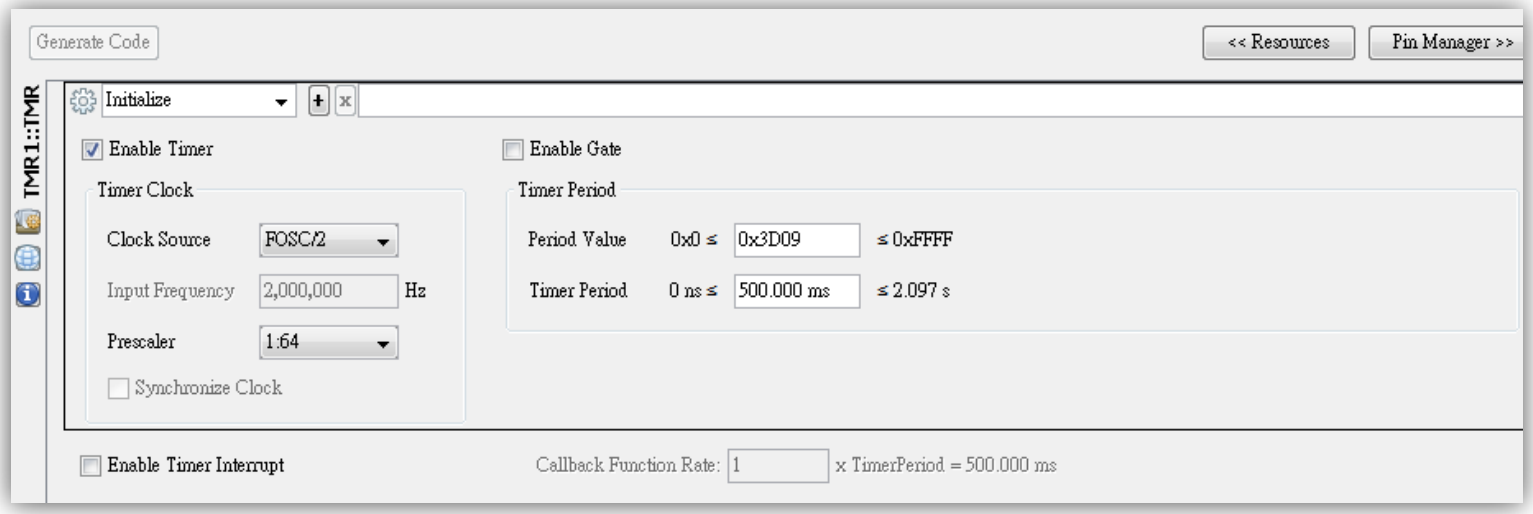
Timer Initialization

- 以前我們要自己在程式中, 加入各項的初始化程式碼

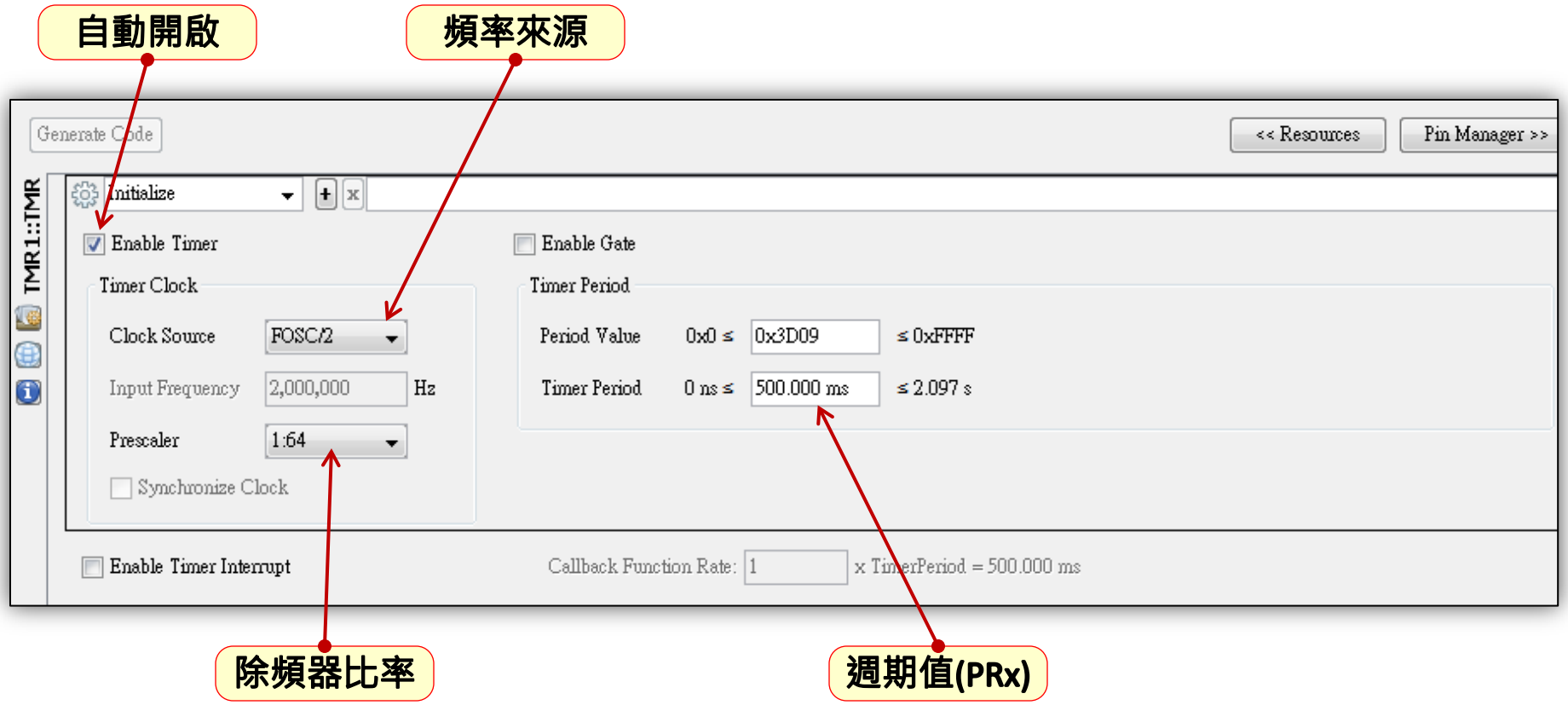
```
OpenTimer1( T1_ON & T1_GATE_OFF & T1_PS_1_256 & T1_SOURCE_INT , 1000 );

if( IFS0bits.T1IF == 1 )
{
   IFS0bits.T1IF = 0;
   // ...
}
```

- 現在, 透過MCC就可以達到相同的結果



MCC's Timer Setting



Lab3 – Timer1 Polling

目 標

- 嘗試透過MCC的設定, 加入**Timer1**的初始化以及控制程式。
- 在Lab2的程式基礎上,嘗試改用**Timer1**來取代原先的軟體Delay。控制LED。讓LED1 ~ LED4可以不斷的轉態(Toggle, 1->0->1->...)。每次Toggle的間隔設定為500 mS。

- 該如何開始?

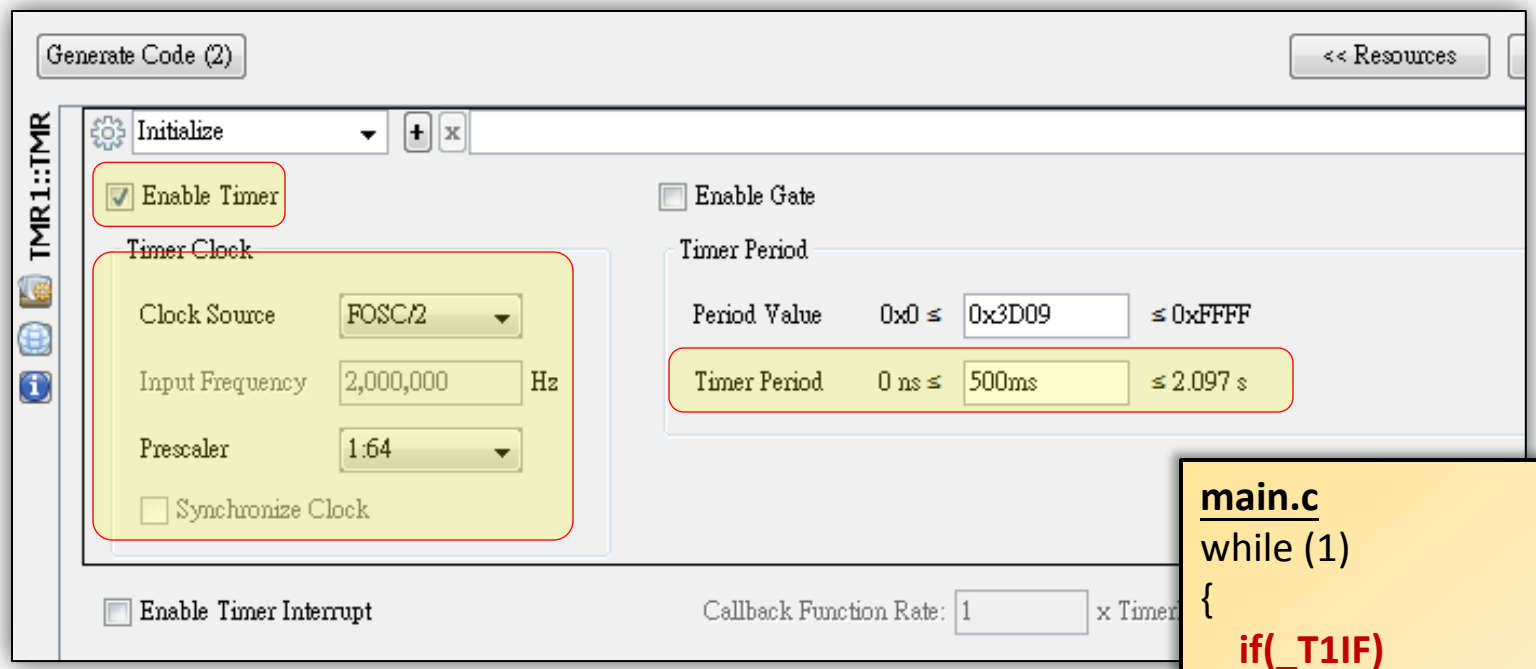
MICROCHIP
Regional Training Centers

Lab3 - Timer Polling Step

- 開啟既有專案(C:\Exercises\Exams\Lab3 Timer1 Polling.x)
- 接著開啟開啟MCC, 新增**Timer1**資源, 依照題目要求設定。
- 觀察MCC產生的函數, 並選用適合的函數來配合題目需求。
- 程式要如何改才能用Timer1取代軟體Delay的功能？
  刪除掉原先的Delay副程式，並透過MCC完成Timer1設定。
  在主程式中,利用Polling Timer的Interrupt Flag(**_T1IF**)來確認Timer1是否計數到預期值。
  Interrupt Flag必須手動清除, 因此確認到 _T1IF==1後, 必須利用程式將其清除為"0" 。
  E.g. : if(_T1IF == 1) { _T1IF = 0; …; }

Lab3 - Timer Polling

MCC's Setting & Code Example



main.c

```
while (1)
{
   if(_T1IF)
   {
      _T1IF = 0;
      // LEDs Toggle;
   }
}
```