MICROCHIP

Regional Training Centers

Section 8
General Propose 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。當Clock的頻率未知時, 僅能得知計數到多少個Clock,稱為Counter。如果Clock已知, 就可以換算出時間, 稱為Timer。其實是一樣的東西。
- 計數值可能是遞增或遞減。PIC32的Timer僅支援遞增。也就是會從"0"開始計數。模組可以設定數多少時要通知CPU。例如: 設定數1,000個Clock後, 通知CPU。
- 此處指的通知,就是中斷旗標(Interrupt Flag) 。當數到設定值時, 中斷旗標會被設定, 對CPU發出中斷請求 (Interrupt Request)。如果此時中斷是致能的, 就會進入中斷服務常式。

MICROCHIP
Regional Training Centers

PIC32的Timers種類

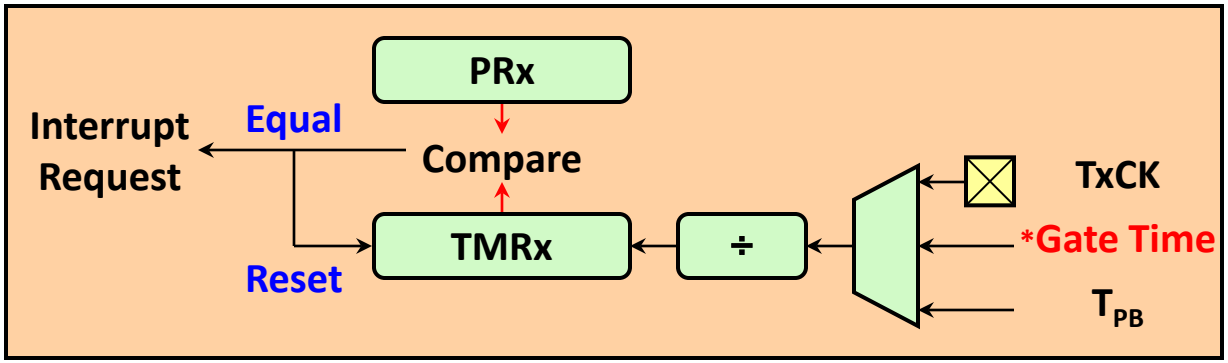
- PIC32有兩種不同樣式的Timer, Core Timer & General Propose Timer。在PIC32中, 擁有一組Core Timer及五組的General Propose Timer。

- Core Timers內建於CP0, 是一組32Bits的Timer。使用System Clock為時脈來源。是特定用途所使用的Timer, 一般不建議拿來使用。

- General Propose Timer是16Bits的Timer, 使用PB Clock為時脈來源。五組Timer的功能幾乎一模一樣。支援計時/計數模式或者閘控計時模式。

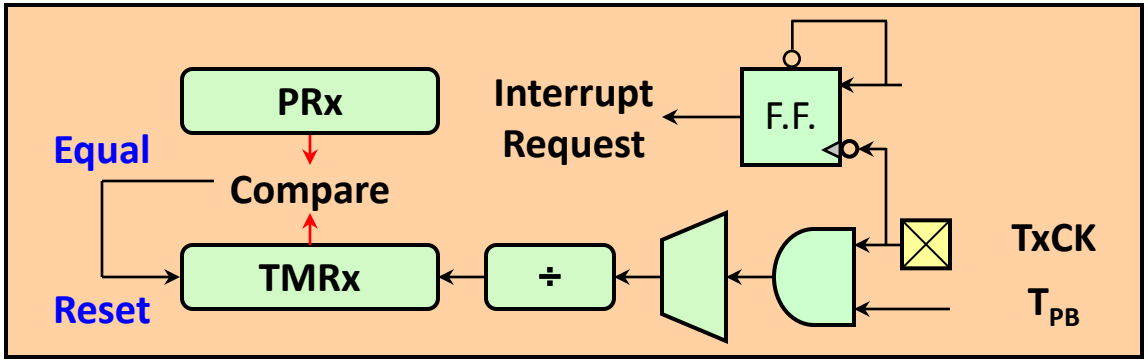
G.P. Timers's 計時/計數模式

- PIC32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, Compare會不斷比較PRx與TMRx的值, 相同時發出中斷需求, 並且將TMRx的值歸零。
- 舉例來說,假設T_{PB}是1mS, 想要計算1秒的時間。則可將Clock設為T_{PB}, 預除器設為除1, PRx設為1,000。如此一來, 每次數到1,000 Clock時(1秒), 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。

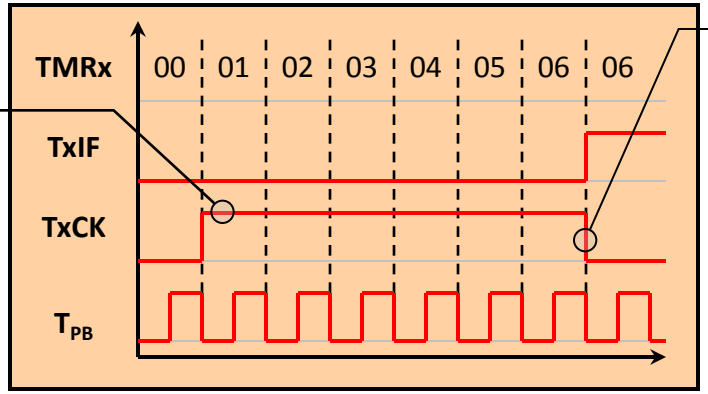


G.P. Timers's 閘控計時模式

- PIC32的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_{PB}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_{PB}可進入TMRx。



負緣觸發正反器,設定中斷旗標,發出中斷需求。

MICROCHIP
Regional Training Centers

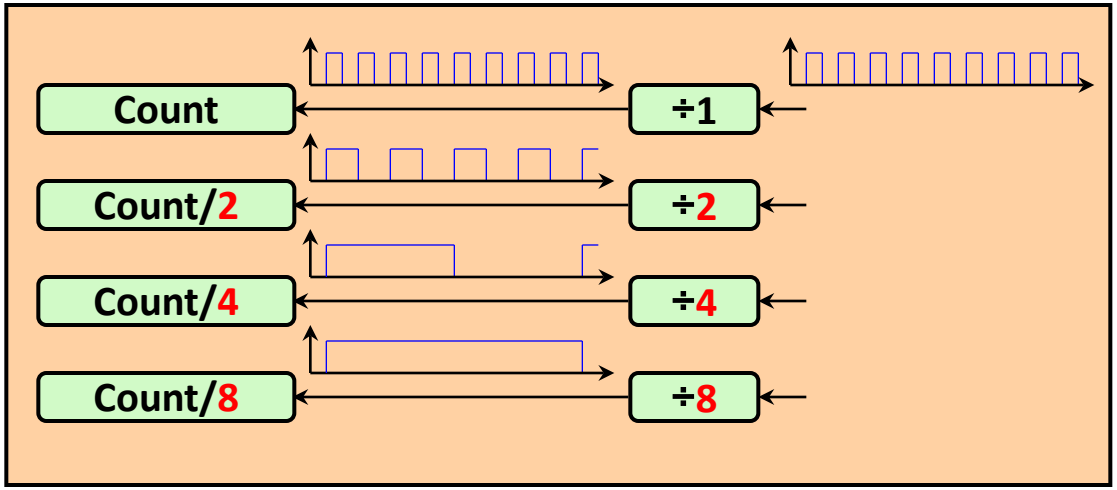
Prescaler

- PIC32MX 的Timer是16Bits, 計數範圍0~65,535。TMRx, PRx都不可超過。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PB Clock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。

- 1uS * 2 = 2uS
- 250,000 * 2uS = 500mS
- 1uS * 4 = 4uS
- 125,000 * 4uS = 500mS

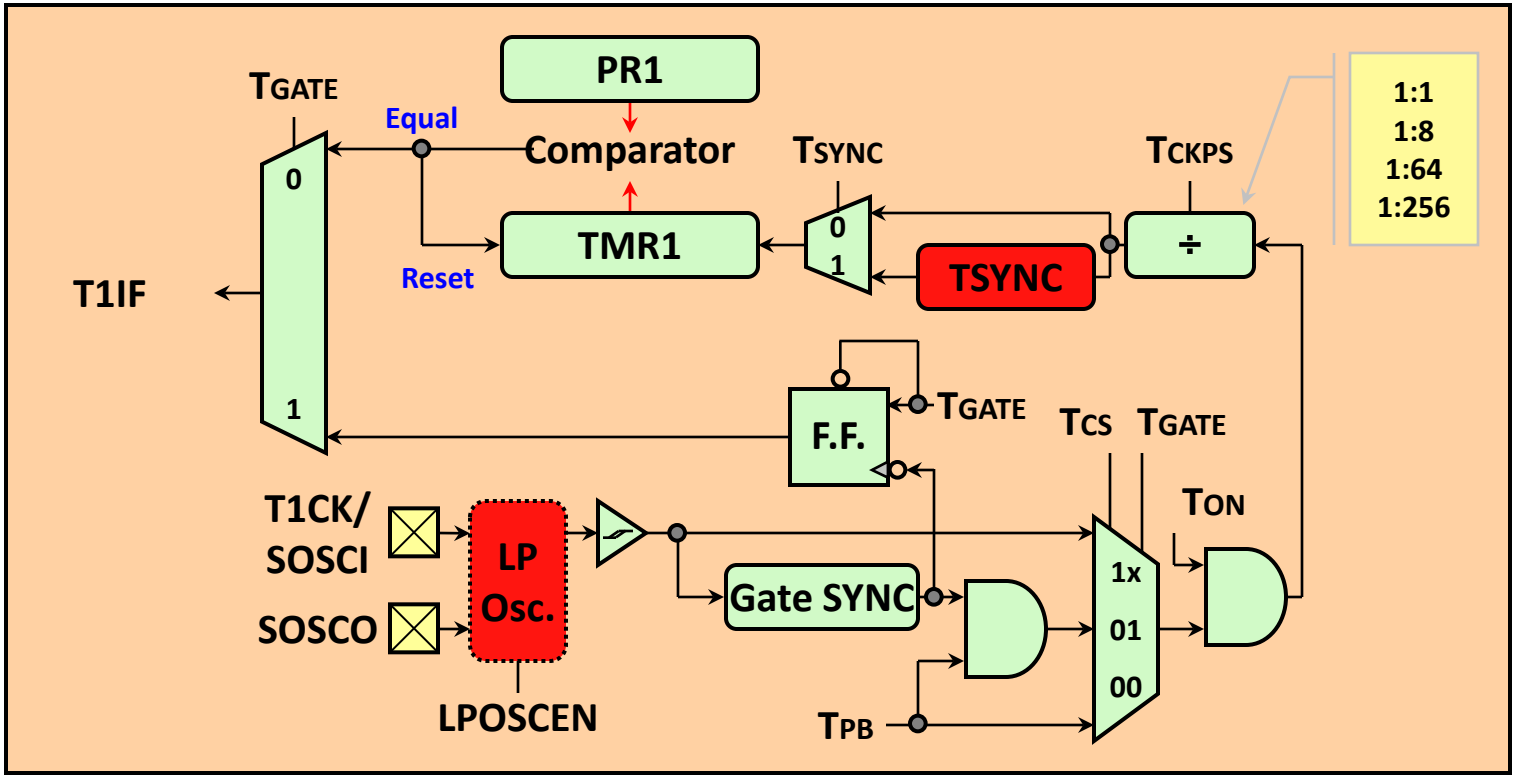
1uS * 8 = 8uS

62,500 * 8uS = 500mS



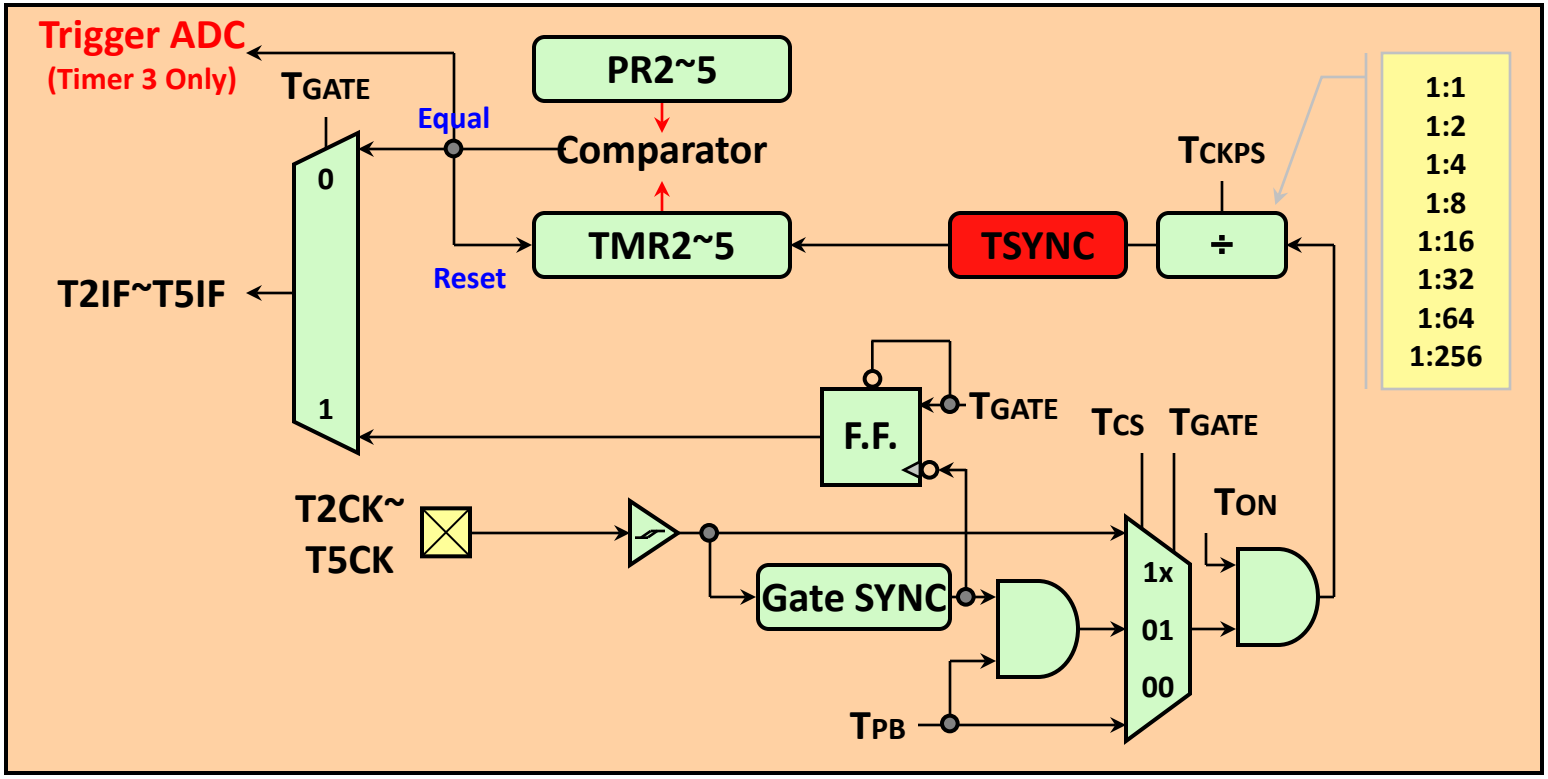
Type A Timer (Timer1)

- Timer1方塊圖如下。Timer1最大的特點,是可選擇同步與否,與內建晶體振盪電路,可以直接連接外部的Low Power Crystal, 如:32.768K Hz,做RTCC。



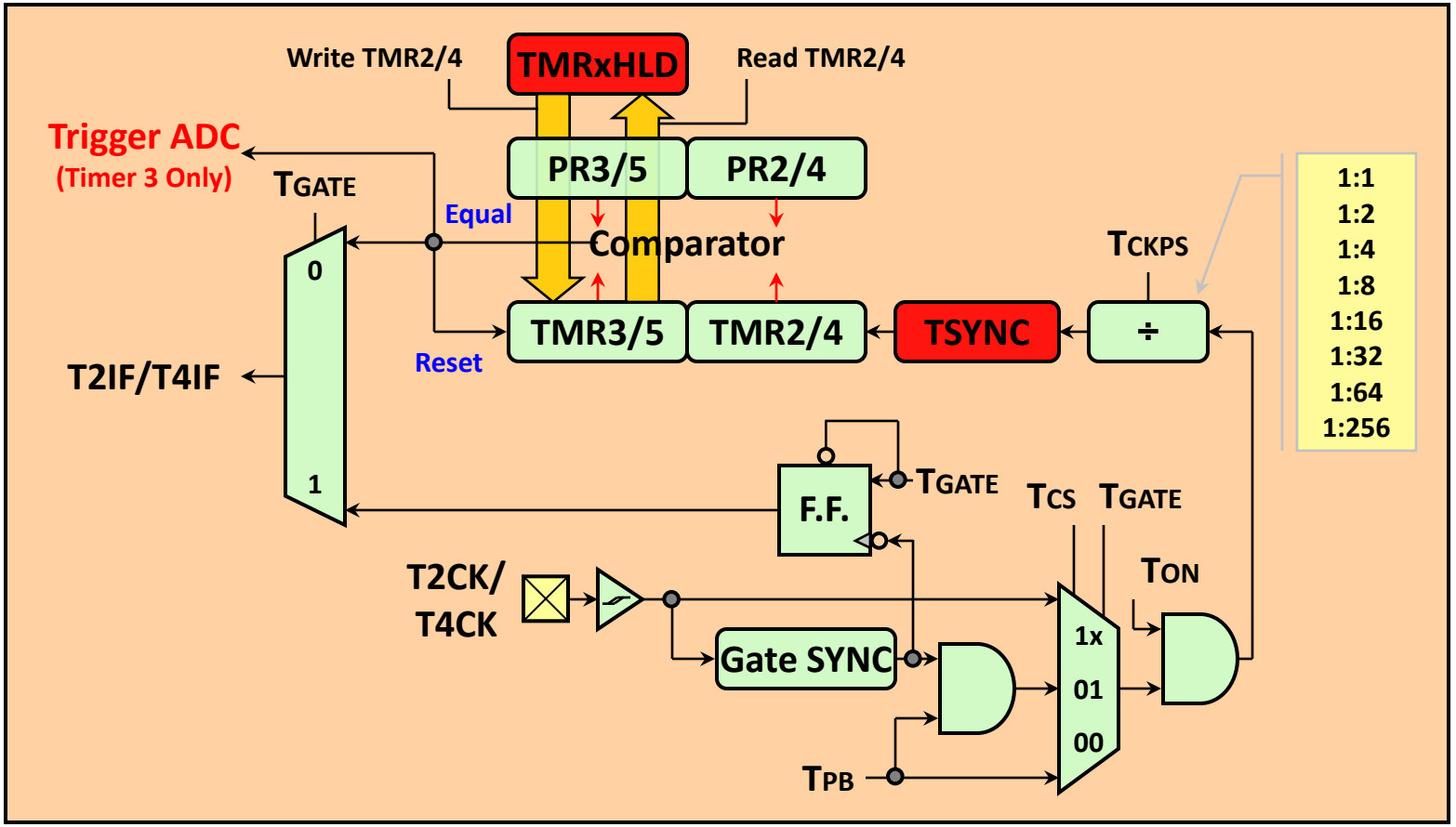
Type B Timer (Timer2~5)

- Timer2~5方塊圖如下。Timer2~5強制與系統時脈同步, 在Idle模式下無法工作。如果要計數外部訊號時,必須連接有明確正負緣狀態的方波,不可以直接連接Crystal。



32-Bits Timer

- Timer23, Timer45可以組合成32-Bits的Timer。



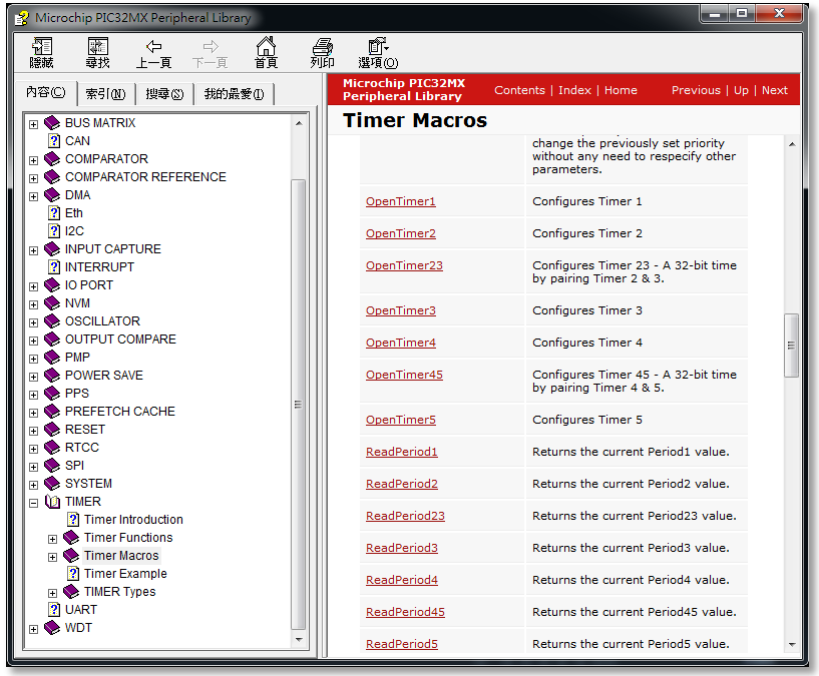
XC32 Timer Function & Macro

- MPLAB XC32提供許多Timer Function可供使用。

```
ConfigIntTimerx( );      // 致能Timerx中斷,並設定中斷優先權。
INTClearFlag( INT_Tx ); // 清除Timerx中斷旗標
INTGetFlag( INT_Tx );   // 取得Timerx中斷旗標
OpenTimerx( );           // 啟用Timerx, 設定Timerx工作模式。
ReadTimerx( );           // 讀取TMRx。
WriteTimerx( );          // 寫入TMRx。
CloseTimerx( );          // 關閉Timerx。
```

- 32-Bits Timer Function。

```
OpenTimerxy( );
ReadTimerxy( );
WriteTimerxy( );
CloseTimerxy( );
```



MICROCHIP
Regional Training Centers

Timer1 Example

- Timer1的初始化範例:

  void OpenTimer1(unsigned int *config* , unsigned int *period*);

  config:Timer的工作模式, *period*:PRx的值。

  Ex:

```
OpenTimer1( T1_ON | T1_IDLE_CON | T1_GATE_OFF | T1_PS_1_256 |
                T1_SYNC_EXT_OFF | T1_SOURCE_INT , 1000 );
```

- Timer1的初始化後,可以利用Polling T1IF(Timer的中斷旗標)的方式,來得知Timer是否計數到預期值或者開啟中斷來得知。

  Ex:

```
if( INTGetFlag( INT_T1 ) )
{
    INTClearFlag( INT_T1 ) );
    ....
}
```

- 相關的說明請參考Microchip PIC32MX Peripheral Library.chm 。

MICROCHIP
Regional Training Centers

Lab5 - Timer Polling

- 在Lab4的程式基礎上,嘗試改用Timer1來取代原先的軟體Delay。控制RB13可以可以不斷的轉態(Toggle, 1->0->1->…) 。 每次Toggle的時間間隔設一樣設為500 mS 。
- 系統時脈動態調整為
  - 80MHz (Primary OSC. With PLL)
  - Sys. Clock / 8 = PB Clock 。
- 透過Bootloader將程式實際燒錄到MCU中, 用肉眼觀察看看程式執行的情況。

Lab5 - Timer Polling Step

- 程式要如何改才能用Timer取代軟體Delay的功能？

  刪除掉原先的Delay副程式，並將Timer1透過OpenTimer1 Function設定完成。
  在主程式中,Polling T1IF來得知Timer是否計數到預期值。T1IF必須手動清除, 因此Polling=1後, 必須利用程式將其清除為”0”。
  建議使用INTGetFlag(INT_T1);及INTClearFlag(INT_T1); 。

- Timer要如何設定？

  設定時脈來源為內部 (T_{PB} = ((1 / 80MHz) / 8) 。然後試著計算PR1及預除器該填入多少,才能達到500mS。

MICROCHIP
Regional Training Centers

Timer Period (PRx)的計算?

- PRx有沒有更容易了解計算的方式?
  自己算太累了,請Compiler幫我們算。

```
#define SystemFrequency 80000000L
#define PBFrequency SystemFrequency / 8
#define Timer1Tick ( ( PBFrequency / 256 ) / Timer1TogglesPerSec )
#define Timer1TogglesPerSec 10
OpenTimer1( … & T1_PS_1_256 & … , Timer1Tick );
```

SystemFrequency : 系統頻率(80MHz)。
PBFrequency : SystemFrequency / 8 (PB Clock跟Clock比率) 。
PBFrequency / 256 (預除器的設定) :
取得進入Timer1頻率。換個方式想, 就是計算一秒內進入Timer1的Clock個數。
(SystemFrequency / 256) / Timer1TogglesPerSec:一秒鐘有m個Clock, 那0.5秒就是m/2, 0.1秒就是m/10。