CIP/MCC 實作指導手冊

使用獨立於核心外的周邊來提升性能

Core Independent Peripherals (CIPs)

Microchip Technology 台灣分公司
台北市民權東路三段四號12F
總機電話: 02-2508-8600

V1.20

CIP/MCC 實作指導手冊

使用獨立於核心外的周邊來提升性能

Core Independent Peripherals (CIPs)

實作內容

有關於本實作課程的一些系統需求與說明:

系統需求:

- MPLAB X IDE v3.05（含）或更新版本
- XC8 編譯器 v1.35（本版本如需 PIC18F 週邊函數庫的支援需另外下載安裝）或使用 v1.34（內含 PIC18F 函數庫）
- MPLAB MCC v2.25.2（請用此版本）
- 酷奇板(Curiosity) + PIC16F1619
- CIP V2B 子卡

為求實作中因使用軟體版本不同所產生的差異性，故本實作請務必使用 MCC v2.25.2 的版本。其它 MCC 版本會有些不相容的問題導致 MCC 設定項目的遺失造成功能上的錯誤。所以本實作強烈要求使用 MCC v2.25.2 版本。

基本說明:

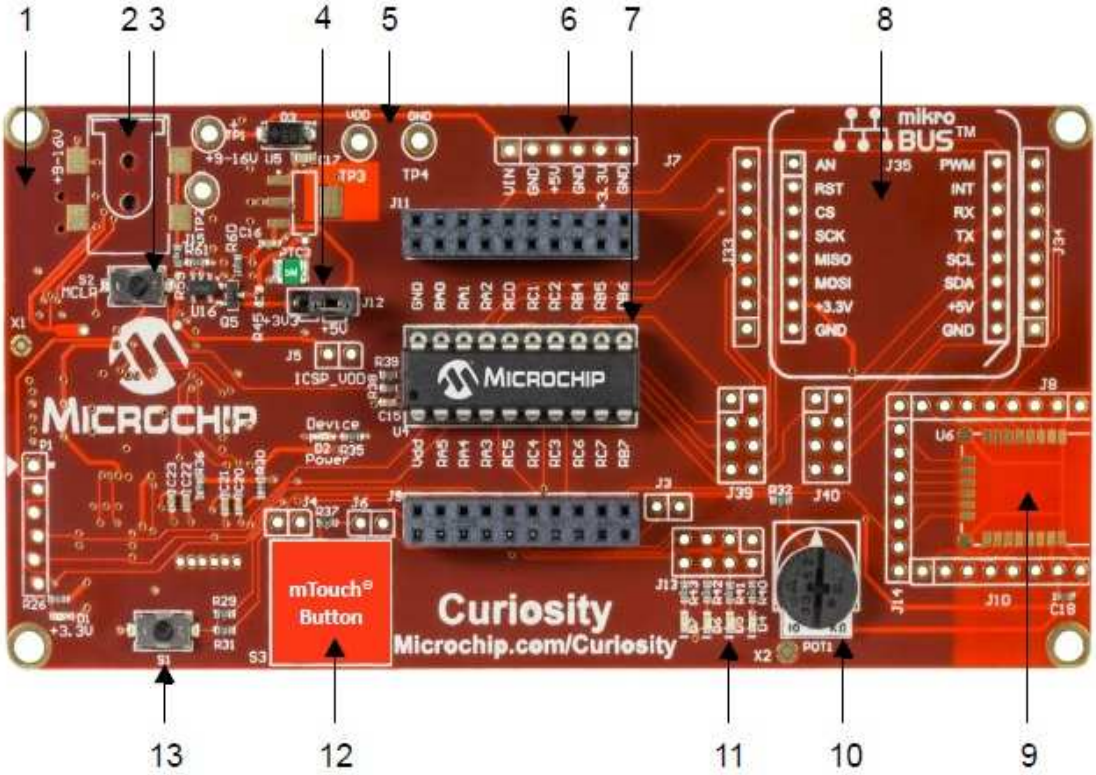
本實作課程不屬於基礎課程，為達到學習的目標建議參加此實作的學員需具備底下基本技能:

- 熟悉 XC8 編譯器及一定的 C 語言撰寫能力
- 熟悉 MPLAB X IDE 的使用及各項除錯功能用法
- 熟悉 MCC 基本周邊的設定及其周邊函數的使用
- 熟悉使用 PICkit3 或 ICD3 除錯工具
- 需具備電子電路硬體電路的設計知識

Microchip 台灣技術團隊與您共同創造未來

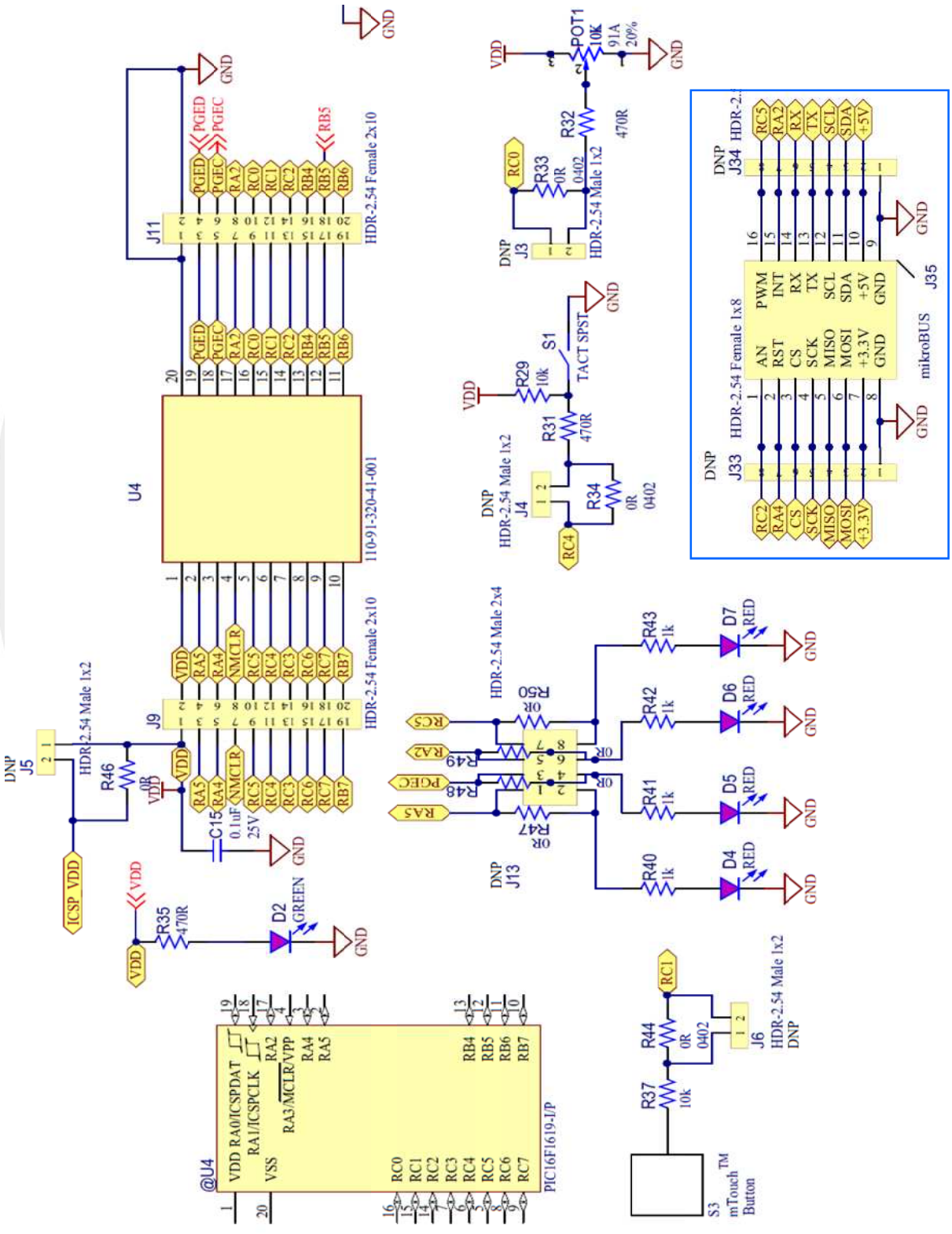
技術服務專線: 0800 717178

酷奇實驗板主要零件說明



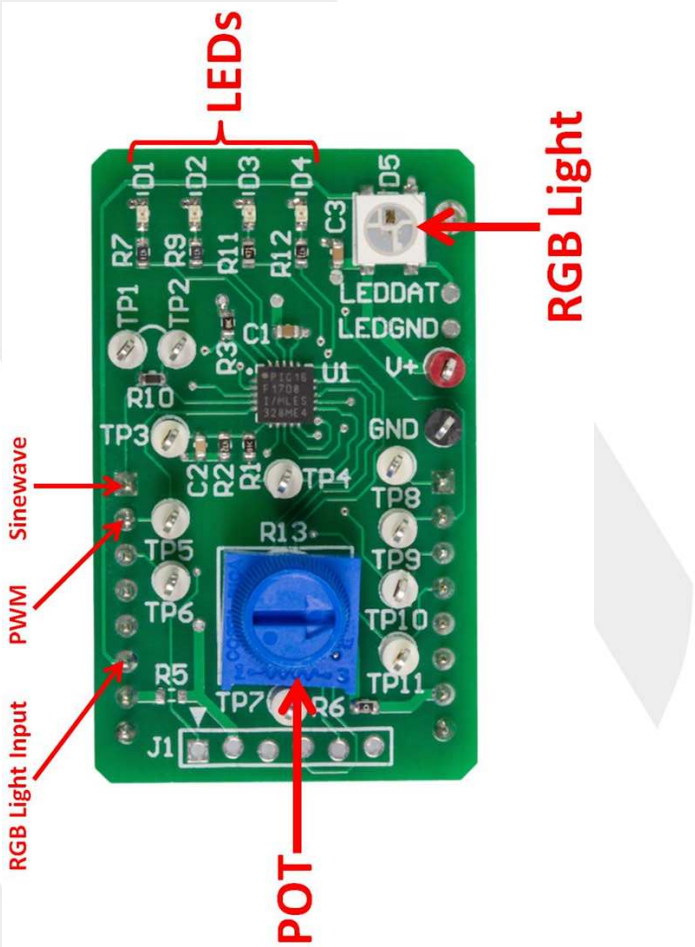
1. USB mini-B 連接器，供電及除錯（在背面）
2. 9V 電源接頭（未裝預留元件）
3. 主重置按鍵（Reset）
4. 3.3V/5V 工作電壓選擇（J12）
5. 外接電源輸入接點（未焊接）
6. 外接擴充板排針接頭（未焊接）
7. PIC 插座（可適用於 8/14/20 接腳的 PIC）
8. mikroBUS 外接插卡連接器
9. RN4020 藍牙模組（未焊接）
10. 電位器（RC0）
11. 4 顆 LED 顯示燈（RA5,RA1/PGEC, RA2,RC5）
12. mTouch 觸控按鍵（RC1）
13. 按鍵開關（RC4）

酷奇板的電路參考考圖。主要是讓讀者了解按鍵、LED 及 可變電阻的接線，在做實驗時比較容易上手。

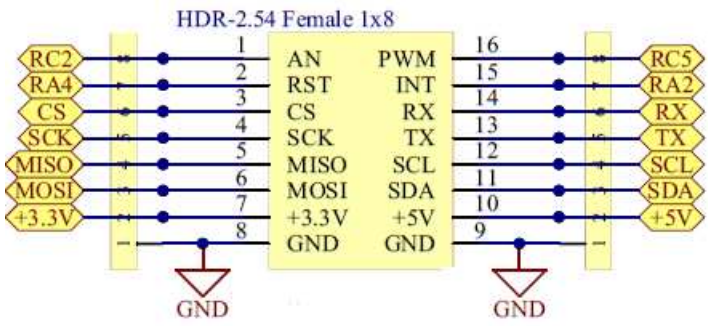


CIP V2B 實驗子板圖示

右圖為本實驗所使用的 mikro BUS 接腳定義的子板（CIP V2B）。此子板主要是提供正弦波（Sinewave）、PWM 並可調整可變電阻改變輸出的頻率供測試用。板子上也有一 RGB 三色 LED 做 CLC 的練習。

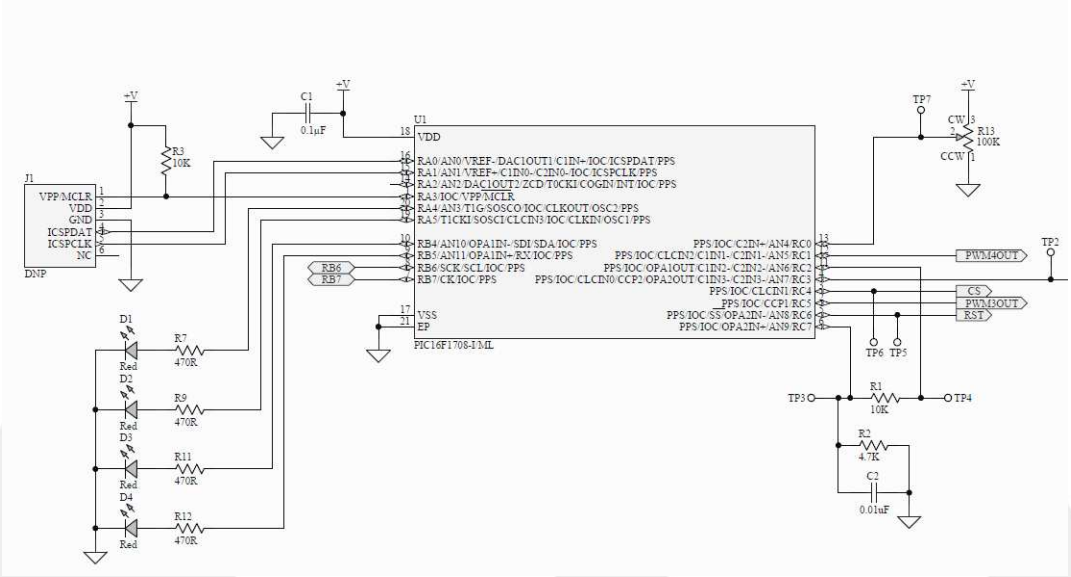


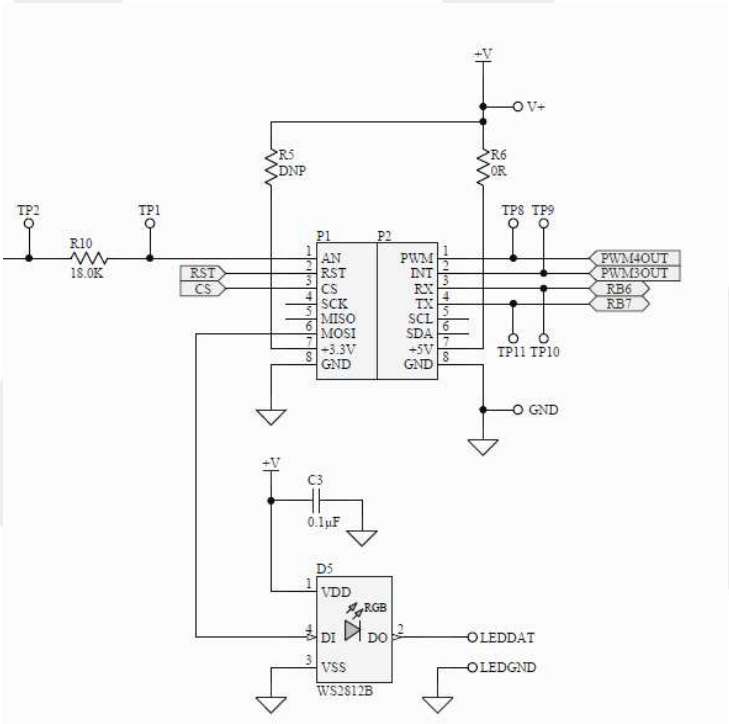
底下為 mikro BUS 的接腳，及接到 PIC16F1619 的對應腳位的功能。



子卡背板的排針腳位及接腳名稱

子板 (CIP V2B) 的線路圖:





Lab 0

Lab0 為基礎練習，主要是用來測試系統及 Curiosity 實驗板否可以正常工作:

1. MPLAB X IDE 正常
2. MPLAB XC 編譯
3. Curiosity 板連線正常
4. MCC 功能正常

在本實作課程裡，所有的實驗均使用 Lab0 的 Configuration 設定，使用 32MHz Fosc 的頻率

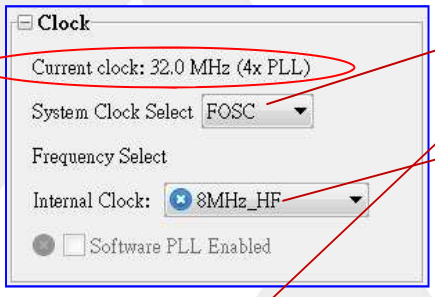
Lab 0: MCC 裡 Configuration Words 的設定

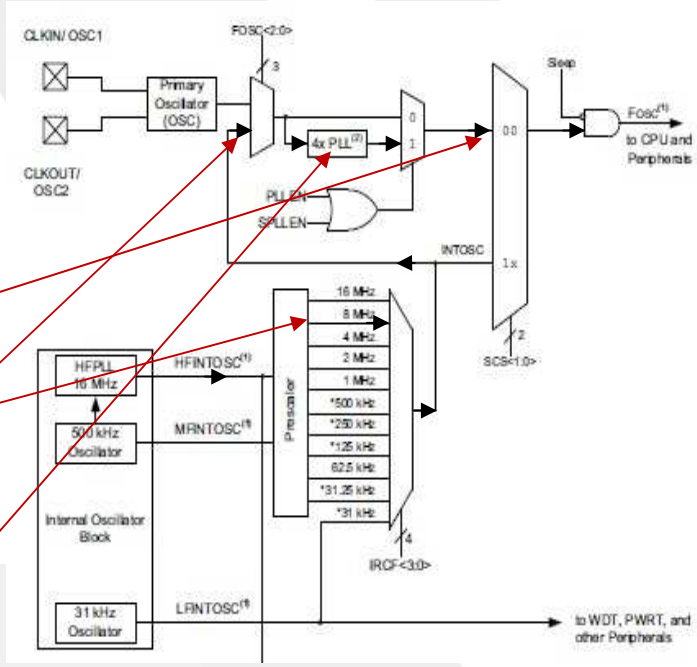
目標：

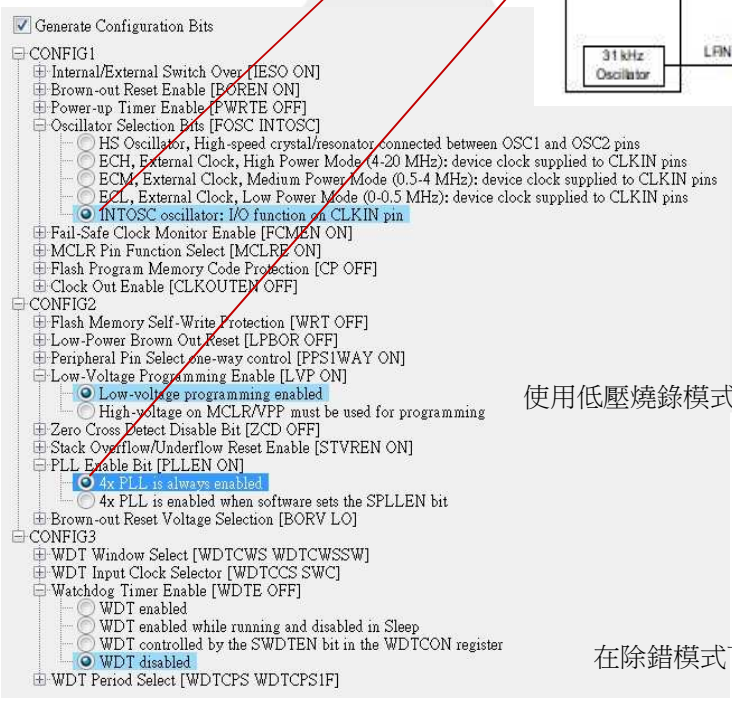
實驗中所使用的實驗板 (酷奇板) 內建除錯工具為 PICkit On Board (PKOB)，了解 Configuration 的設定並使用低電壓燒錄模式。

概述:

1. 酷奇板使用低電壓燒錄模式，需在 Config. Word 中做設定。
   - 開啟 Lab0 Low Voltage Programming.x
   - 在 Tools —>Embedded —> MPLAB Code Configurator 啟用 MCC
   - 點選 MCC 專案資源的 System”
     開啟 PIC16F1619 的 Clock 及 Config. Words 的設定項目
     ⇒ 看門狗計時器 (WDT) 需關閉
     ⇒ 使用 x4 PLL 倍頻輸出
     ⇒ 系統頻率: 32MHz



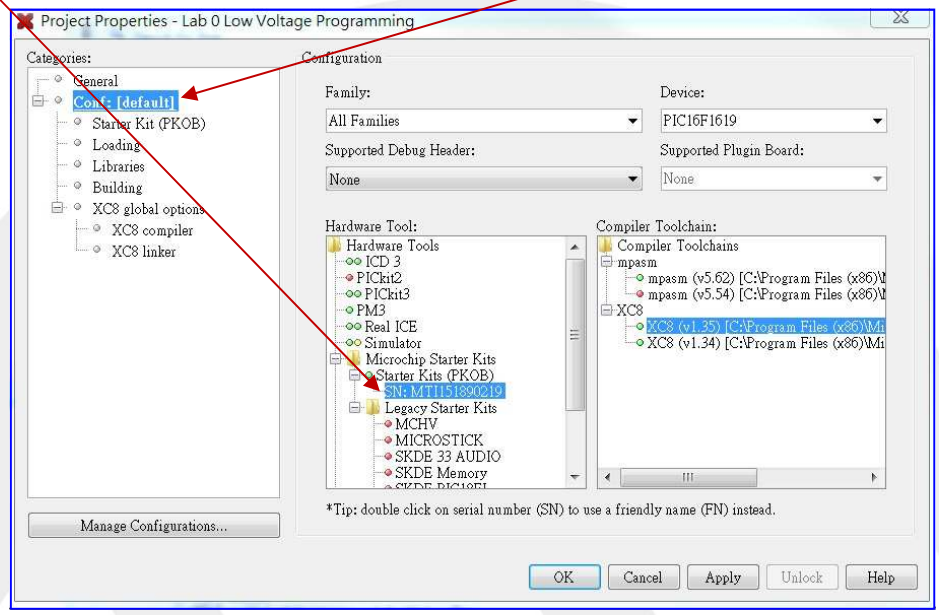




使用低壓燒錄模式，這不同於 PICkit3 的高壓燒錄模式

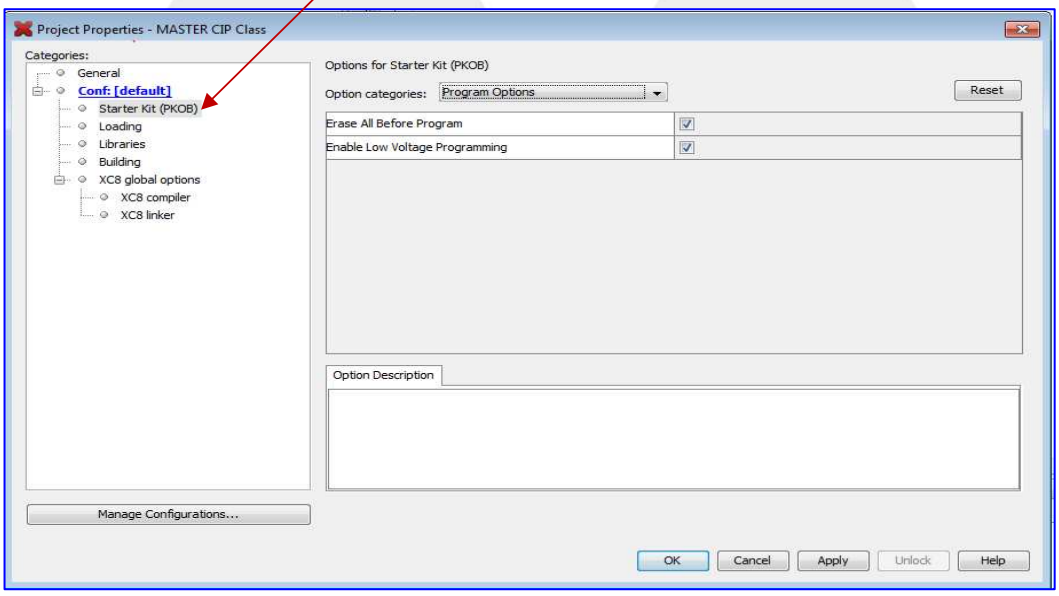
在除錯模式下一定要將 WDT 關閉

2. 請確定酷奇板的 USB 線已經接好。在專案裡選好 Lab0 Low Voltage Programming 的名稱按老鼠右鍵開啟內容 (Properties) 後，新帶出的視窗 Conf: [Default] 可以看到 PKOB 的序號。(序號每片實驗板都不一樣)



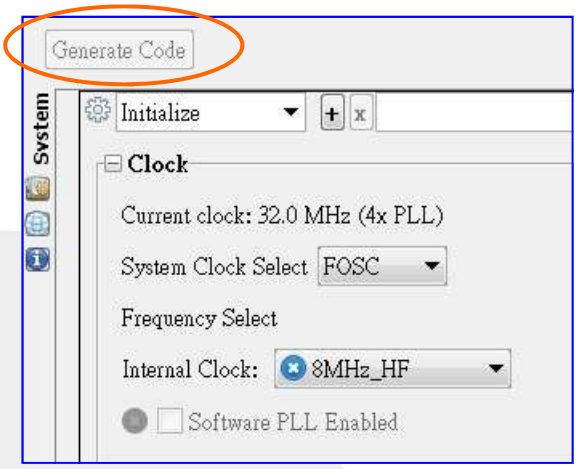
3. 一樣在 “Conf:[default]” 裡的 “Starter Kit (PKOB) ” 的選項目錄裡選擇 “Program Options” 後，請確定底下兩項都有勾選:

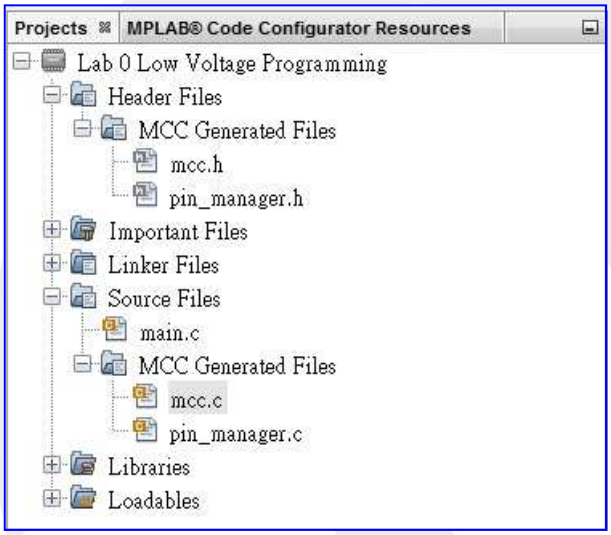
Erase All Before Program
Enable Low Voltage Programming



基本需求的 Config. Word 的設定已完成。接下來就只要按一下左上角的 “Generate Code” 來產生程式碼。

在產生程式碼的過程，MCC 會判斷有無 main( ) 函數的存在，它會主詢問是否要代為產生 main( ) 函數。在這裡我們將使用 MCC 所產生的 main( ) 加入目前的專案裡。





左圖為使用 MCC 所產生的基本檔案。當然這些是最基本的檔案需求，隨著各種周邊的加入 MCC 所產生的周邊檔案就會逐項的一一加入到專案裡。

基本檔案說明:

- mcc.h 及 pin_mamager.h 主要是做函數原型宣告，一些基本的定義。
- main.c MCC 自動產生的主程式
- mcc.c 設定 config. Word ，系統振盪的設定及系統初始設定

有關此實作所需的基本技能知識可以參考底下的中文教育訓練: (如下圖所示)

www.microchip.com.tw/Data_CD/

- MCC v2.5.2 的操作，如有不熟悉的話，建議參考教育訓練 “ MCC201 v1.00 MPLAB Code Configuration New!” 的教材。
- XC8 請參考教育訓練 “XC8T v1.0 New!” 的教材。
- MPLAB X IDE 的使用請參考 “XIDE MPLAB X IDE 中文版 New!” 的教材。

Lab 1

Lab1 是 MCC 基礎練習，試著使用 MCC 來設定 Timer1 的計時 0.25 秒，並在中斷裡將一 顆 LED 做轉態做測試。

Lab 1: 最基本的實作練習 (一定要了解)

利用 MCC 來設定 LED (RA5) 腳位功能，並使其每 0.25 秒閃爍一次。

目標：

使用 MCC v2.25.2 來設定：

1. 系統工作頻率為 32MHz 及 Config. Word。
2. RA5 設成 LED 的輸出驅動腳位。
3. 設定 Timer1 每 0.25 秒產生中斷一次。
4. 在 Timer1 的中斷裡 Toggle LED。

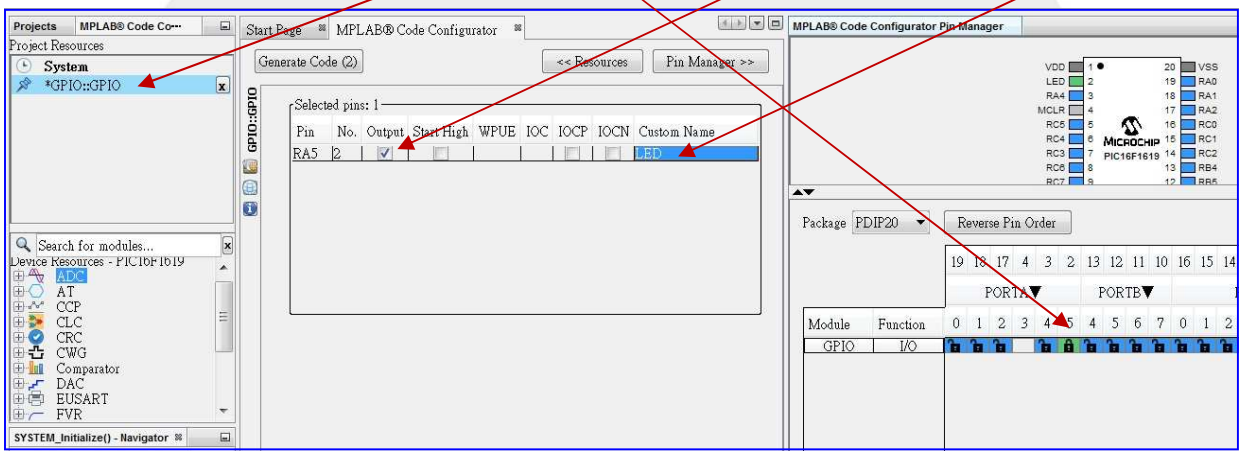
概述：

主程式 main( ) 只在開始實作 CIP 周邊的初始設定後，隨即進入while(1); 的永久迴圈。

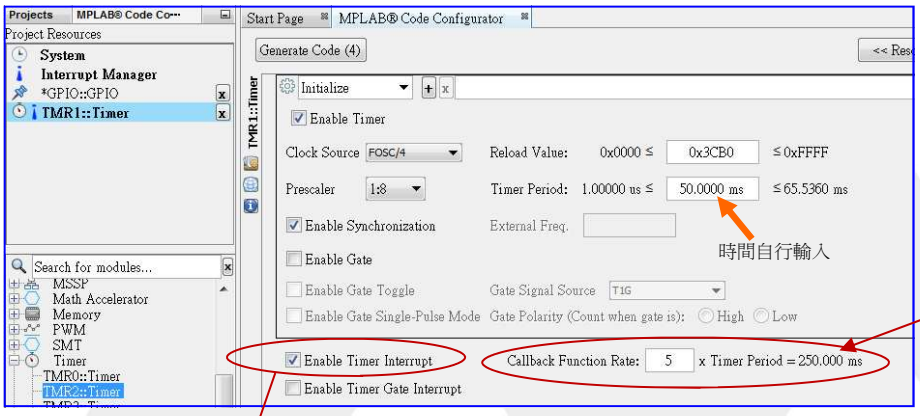
Lab1 開始做實驗

1. 開啟 MPLAB X IDE 並以酷奇板需求建立一個專案 “Lab1_Blink LED”
2. 啟動 MCC 工具。(Tools —> plugins —> 選取 MCC)
3. 先設定 “ System” 的設定 (相同於 Lab 0 的設定)
4. 設定 GPIO (RA5 驅動 LED)
5. 在 Timer1 中斷裡加入Toggle LED 的敘述

設定 GPIO 的步驟: 在 Project Resources 點選 GPIP:GPIO 帶出 I/O 腳設定畫面。點選右下角的腳位規劃圖示，將 RA5 點成綠色的上鎖狀態。接這勾選 RA5 為輸出腳，並將其腳位名稱設為 “LED” 。

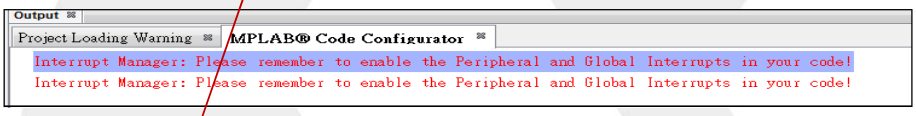


GPIO 已經設定完畢，接下來將設定 Timer1 來取的 250mS 的時間來閃爍 LED。
先啟動 Timer1 的設定，其相關的設定如下圖所示。



因為系統使用 32MHz 及使用 1:8 的預除器還是無法達到 250mS 的計時。所以使用軟體輔助計時來產生 250mS 的計時。也就是發生五次的中斷才執行一次 CallBack 函數。

Timer1 中斷計時方式，在其底下會有開啟中斷的提示 (如下圖) 提醒在 main( ) 裡要開啟 GIE & PEIE 的中斷控制。因為開啟 Timer1 的中斷只會設定 TMR1IE 而已。



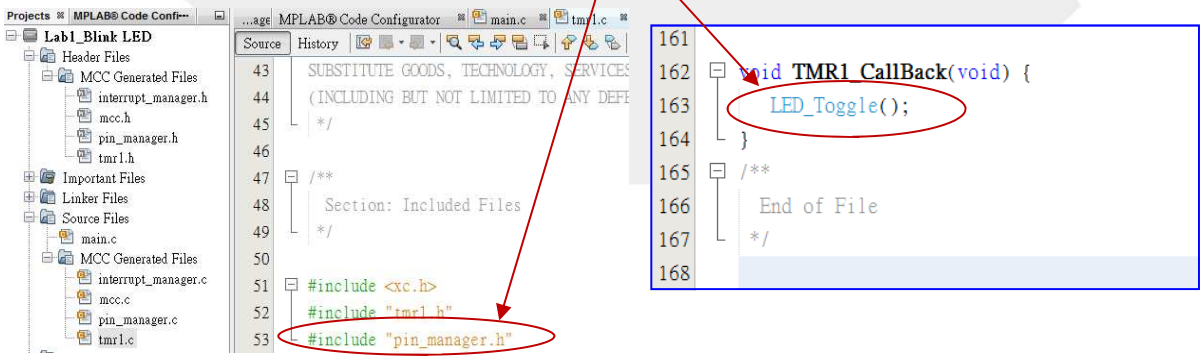

```
// Enable the Global Interrupts
INTERRUPT_GlobalInterruptEnable();
// Enable the Peripheral Interrupts
INTERRUPT_PeripheralInterruptEnable();
```

需在 main.c 函數裡，啟用 GIP 及 GPIE 設定

最後就是要修改 tmr1.c 裡的程式。之前有提到 Timer1 設定時有用到 Callback Function Rate 的設定。因為 Timer1 中斷已裡用軟體計數方式將時間變成 250mS 了，所以只要直接將 Toggle LED 功能加到 TMR1_CallBack ( ) 的函數裡。
因為我們在設定 GPIO 腳位時已經定義了 RA5 為 LED 的名稱，所以在 pin_manager.h 就會有該

```
// get/set LED aliases
#define LED_TRIS                TRISA5
#define LED_LAT                 LATA5
#define LED_PORT                RA5
#define LED_SetHigh()           do { LATA5 = 1; } while(0)
#define LED_SetLow()            do { LATA5 = 0; } while(0)
#define LED_Toggle()            do { LATA5 = ~LATA5; } while(0)
#define LED_GetValue()          RA5
#define LED_SetDigitalInput()   do { TRISA5 = 1; } while(0)
#define LED_SetDigitalOutput()  do { TRISA5 = 0; } while(0)
```

針對 LED 腳位的功能定義，在這裡只要將 LED_Toggle( ) 加到 TMR1_CallBack ( ) 的函數裡即可。
注意: 因為使用了該 GPIO 的功能，所以也要將 #include pin_manager.h 加到 Tmr1.c 裡。



在 Tmr1.c 加入這兩行敘述後，直接按燒錄圖示 待編譯及燒錄完成後就可以看到 LED (D4) 每隔 0.5 秒閃爍一次。注意: 也許 Start Kit 要更新韌體，這時請耐心等候韌體的更新，更新其間不可斷線。

Lab 2

利用 CLC ，PWM，SPI 周邊來控制 RGB (WS2812) 單線式串列 LED 的顯示

Lab 2:
使用 CLC 來控制高速的單線串列式 RGB LED

目標:

使用 CLC, PWM, and SPI 的組合來驅動高速的單線式 RGB LED .

概述：

這個設計範例將說明如何整合 SPI, CLC, and PWM 還產生一高速的串列驅動訊號來推動這 RGB 達到顏色及亮度的控制(WS2812).

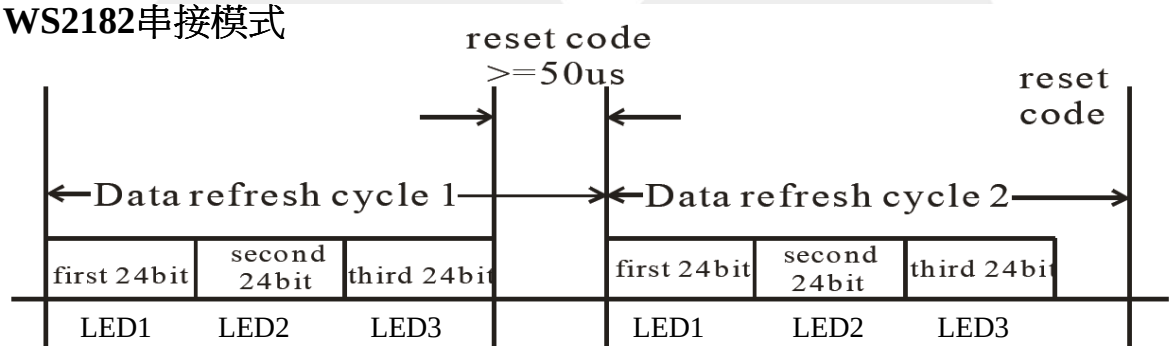
WS2812 為一高速單線傳輸的 RGB LED，它可以完成 24-bit 的全彩顯示，並可作亮度控制。因為是使用 高速的 24-bit 串列編碼訊號，以 G(8-bit) R(8-bit) 及 B(8-bit) 組合而成。

WS2812 24 位元傳輸方式 : 綠，紅，藍

G7, G6, G5, G4, G3, G2, G1, G0, R7, R6, R5, R4, R3, R2, R1, R0, B7, B6, B5, B4, B3, B2, B1, B0

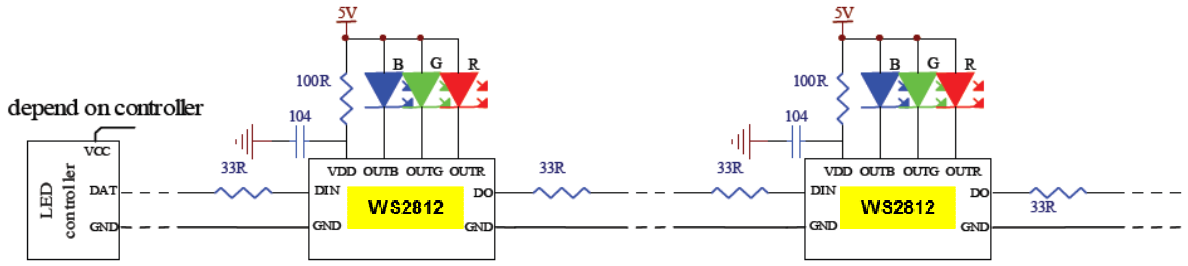
WS2812 雖為一單體的 RGB LED，但透過本身的串列輸出腳可以一直串接給下一個 WS2812，所以可以做成一長串的 RGB LED 顯示條或 RGB LED 顯示板。

WS2182串接模式



由上圖所示，只要送出大於 50uS Low 訊號，WS2812 就會重置並將顯示的訊號從 LED1 開始依序顯示。

那 WS2812 每一個位元訊號的編碼又是長得如何，為何不可以用一般 MCU I/O 腳送出串列訊號呢? 底下來看一下 WS2812 的傳輸速度及位元的編碼方式:



WS2812 串接方式

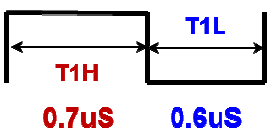
傳輸速度又使多少? 依據 WS2812 資料手冊所示，其傳輸的速率不可以低於 400kps 建議使用 800Kbps 的速度，也就是傳輸 1 bit 的時間只有 1.25uS。

WS2812 的通訊方式是每一位元採用單線式的 NZR 通訊模式，只要三條線 (VDD，GND，DIN) 即可串接多個 LED。當多串的 LED 串在一起時，為避免動態顯示的閃爍現象，WS2812 建議以 800Kbps 速度傳輸。底下為一個位元 Hi 及 Low 的編碼方式與時間。

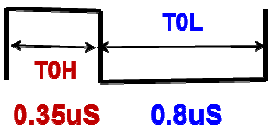
T1H 時間	位元 1 的編碼，Hi 週期所暫的時間	0.7uS	+ - 150nS
T1L 時間	位元 1 的編碼，Low 週期所暫的時間	0.6uS	+ - 150nS
T0H 時間	位元 0 的編碼，Hi 週期所暫的時間	0.35uS	+ - 150nS
T0L 時間	位元 0 的編碼，Low 週期所暫的時間	0.8uS	+ - 150nS
Reset	Low 電壓持續時間	大於 50uS 以上	

資料編碼傳輸時間 (T1H + T1L = 1.30uS，T0H + T0L = 1.25uS)

單一位元編碼的格式與時間如下同所示。除了單一位元傳輸時間為 1.25uS，又必須對位元做特殊的編碼方式方可傳輸出去。由下圖來看 T0H 的傳輸單位時間短到 0.35uS。這是一般 MCU 的 I/O 腳以傳統軟體驅動方式所做不到的時間控制。所以在本練習將透過 CIP 的組合來達到此高速的傳輸並解決其編碼問題。如此以一個小 PIC 微控制器就可以輕鬆控制 WS2812 的 LED 顯示串。



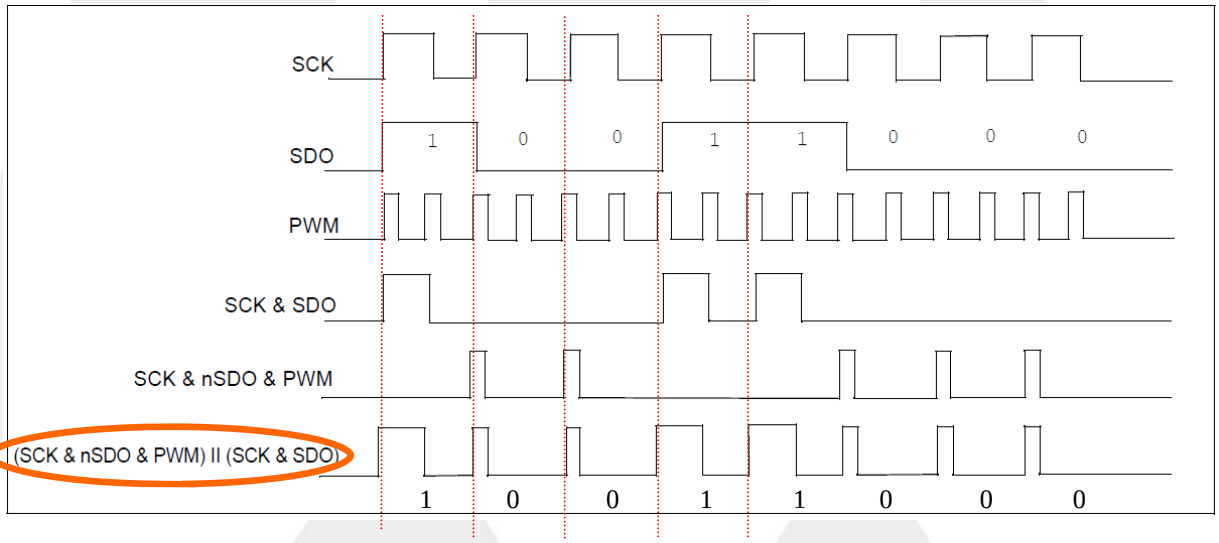
位元 1 的編碼



位元 0 的編碼

看起來 WS2812 需求高速的單線式傳輸，最小的的調變脈波寬度僅 0.35uS ，這絕對無法用現有的 PIC 以軟體驅動 I/O 來完成的。這時就像設計 ASIC 一樣，需要發揮一下硬體的設計長才方可達成此功能。

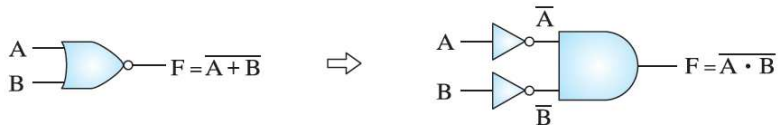
1. 基本上傳輸一個位元時間需 1.2uS，且一次以 8-bit 為一組，一次要傳送三個 Bytes 出去。所以考慮用 SPI 的傳輸模式。因為 SPI 傳輸速度可以達 10MHz 大大符合我們 800KHz 的傳輸速度。所以 SPI 的設定其 CLK 為 800KHz，其 CLK 的 Hi 週期為 0.75uS，Low 週期也為 0.75uS。
2. 至於最小的脈波 0.35uS 考慮用 PWM 固定來產生，此脈波主要是為了資料 位元為 0 時的調變。所以設定 PWM Hi Duty 為 0.35uS，週期為 0.75uS，並與 SCK 的上升緣同步。
3. 到現在基本波都有了，在過來就要使用 CLC 的組合邏輯功能透過積之和的運算(先 AND 在 OR 運算）基本少就可以產生資料的編碼。
4. 基本波的組合，將 SCK 與 SDO 做 AND 運算後的得到位元 1 的編碼。將 SCK， /SDO ， PWM 三個訊號做 AND 運算後的得到位元 0 的編碼。後將位元 1 與位元 0 的編碼輸出再做 OR 運算即可獲得正確的編碼輸出。



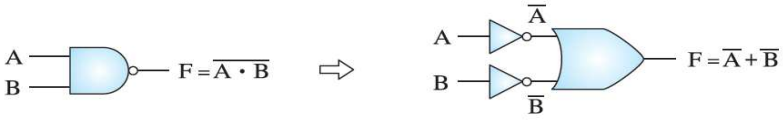
再去查看一下 CLC 的架構，發現 CLC 都是使用 OR Gate 做輸入的，與我們需要的 AND Gate 輸入需求是不一樣的，怎麼辦?

沒關係，在學校的邏輯電路都有學過布林代數 (Boolean Algebra) 與迪摩根定律 (De-Morgan's eorem) ，將以上的輸出方程式再做一下轉換就可以符合 OR Gate 端的輸入要求了。要怎樣做，首先還是要複習一下迪摩根定律: (如無法理解請自行用真值表來證明)

迪摩根第一定律： $\overline{A + B} = \overline{A} * \overline{B}$



迪摩根第二定律： $\overline{A * B} = \overline{A} + \overline{B}$



我們將上面的輸出波形的方程式經轉換後得到底下的公式來設定 CLC 的邏輯組合電路:

$$(SCK \ \& \ nSDO \ \& \ PWM)||(SCK \ \& \ SDO)$$

⇩

/ (/SCK + SDO + /PWM) + / (/SCK + /SDO)

以 OR 項為輸出的布林代數來規劃 CLC 電路

在右邊的布林代數在 CLC 電路裡需要的是 OR Gate 的輸入項，所以先將左邊的 SCK & / SDO & PWM 做迪摩根的轉換:

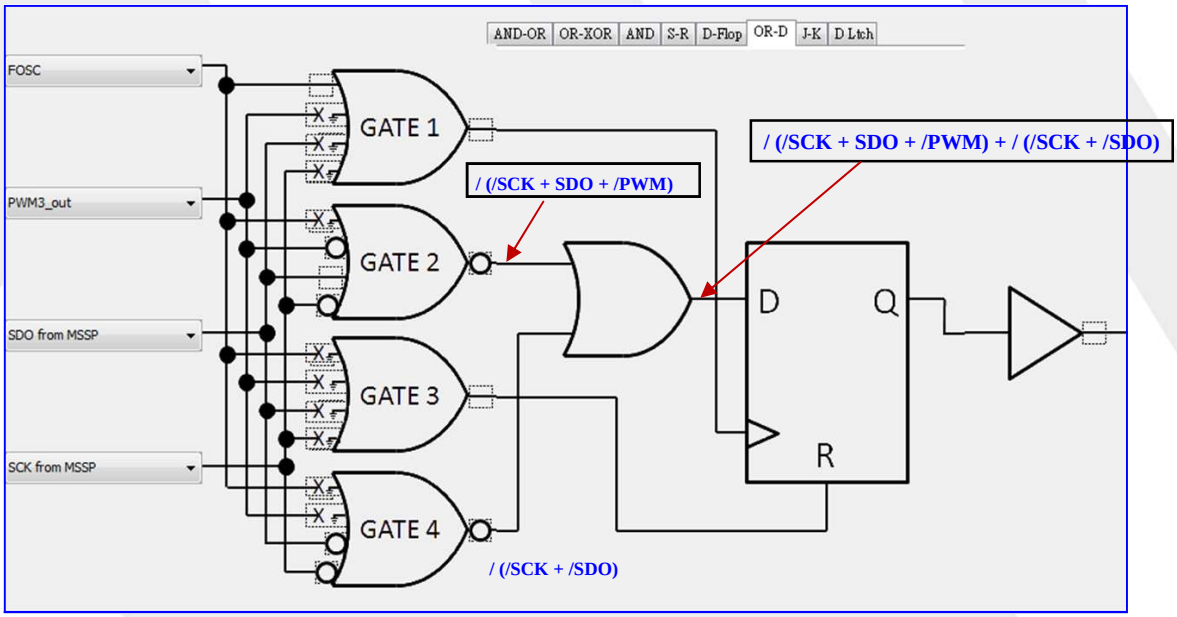
$$A\bar{B}C = \overline{\bar{A}BC} = \overline{\bar{A} + B + \bar{C}}$$

（轉換公式）

SCK & /SDO & PWM = /(/SCK + SDO + /PWM)
SCK * SDO = / (/SCK + /SDO)

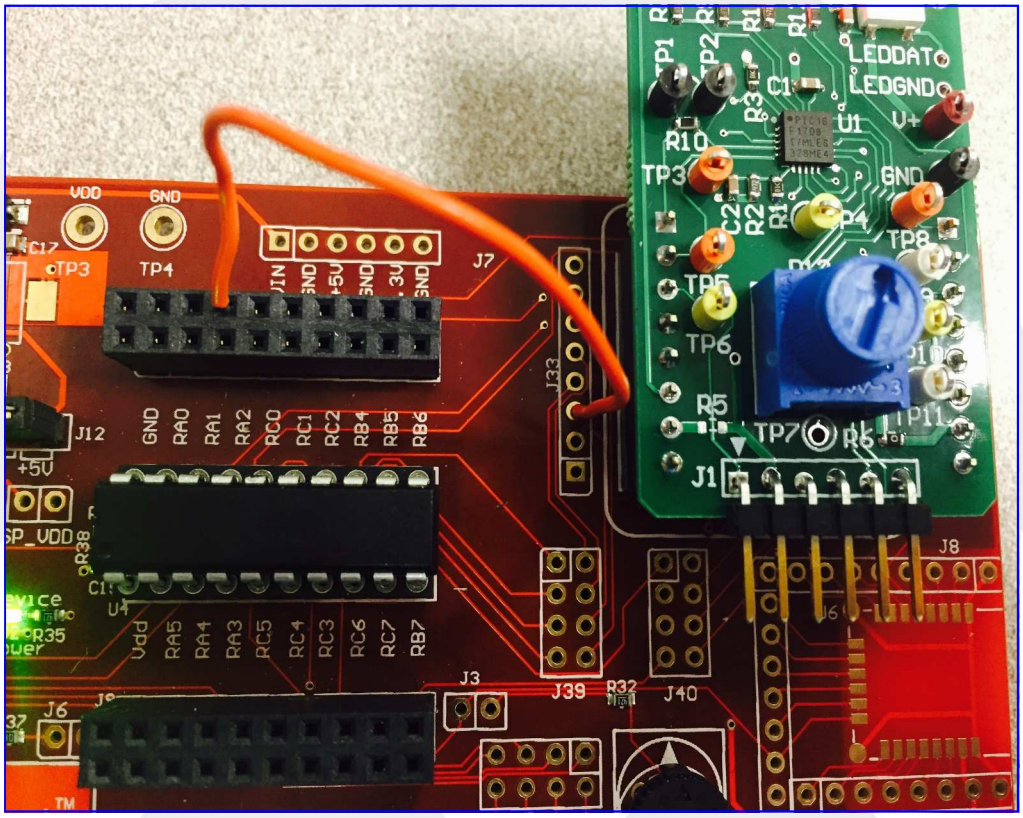
輸出為兩項之和:
/ (/SCK + SDO + /PWM) +/(/SCK + /SDO)

將上面的方程式實際選擇 OR-D 的 CLC 邏輯電路來設計，由下圖對照公式完全符合我們的需求。最後只要將輸出經過 D-Type 正反器在與 Fosc 取得同步過濾掉雜訊毛邊的訊號使輸出更加完美。

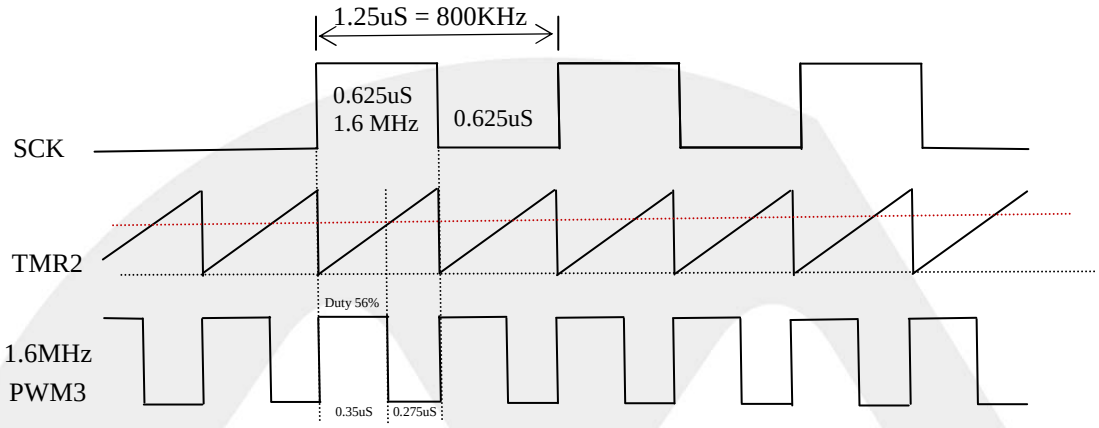


接線的設定:

連接 RA2 的輸出訊號到 mikro BUS 子卡上的 MOSI pin (J33) 做為 WS2812 的輸入訊號，其連接方式如下圖所示。



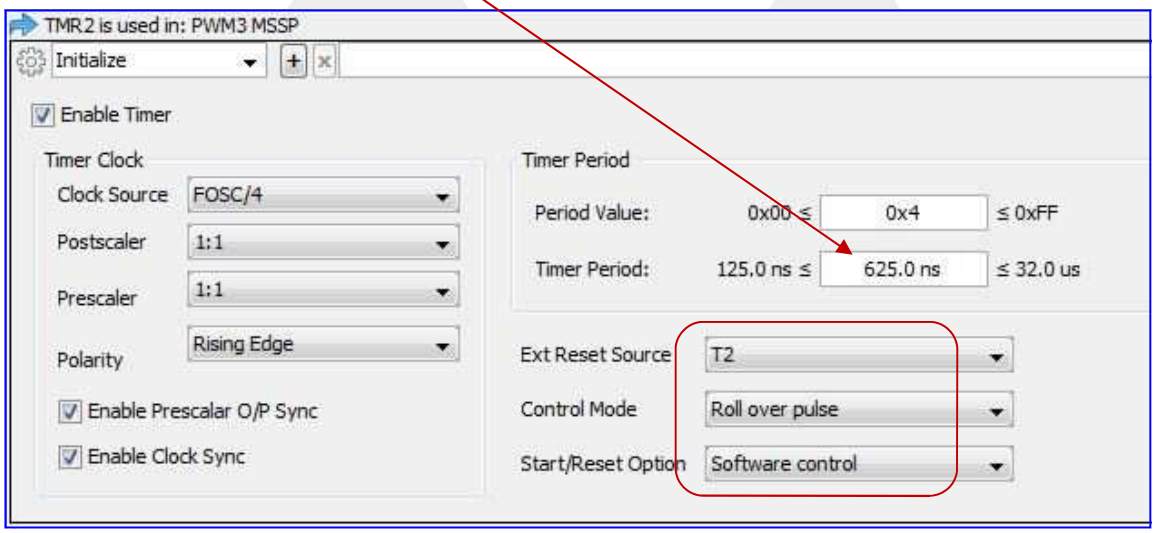
我們以之可以用 CLC 來組合所需的編碼來驅動 WS2818 RGB LED 了。接下來將繼續說明 Timer2，PWM 及 SPI SCK 的時脈關係的設定。首先看一下底下的相關的圖示:



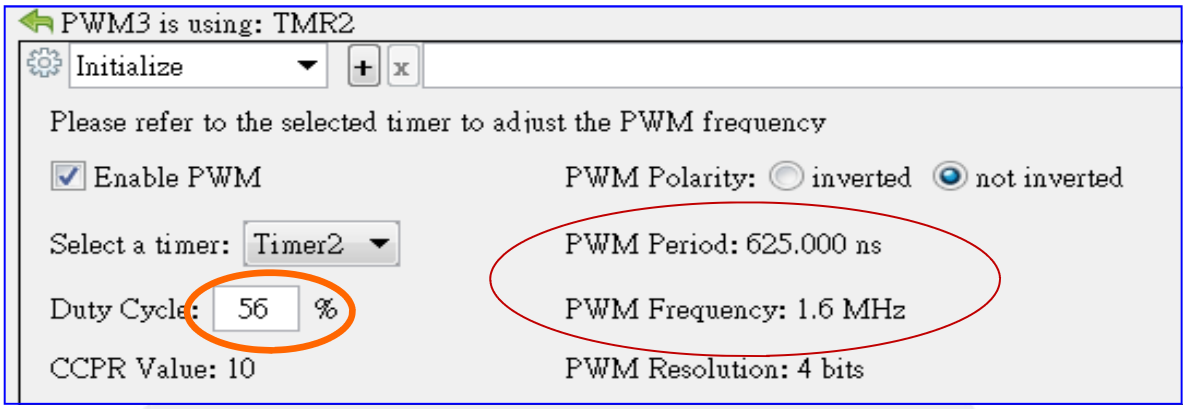
由上圖可之整個時脈是由 Timer2 來控制的。Timer2 規劃成 1.6MHz 後給 PWM3 (0.625 uS Period) 及 SPI 的 SCK 其速率一樣來自 Timer2 為 800KHz。

Lab2 開始做實驗

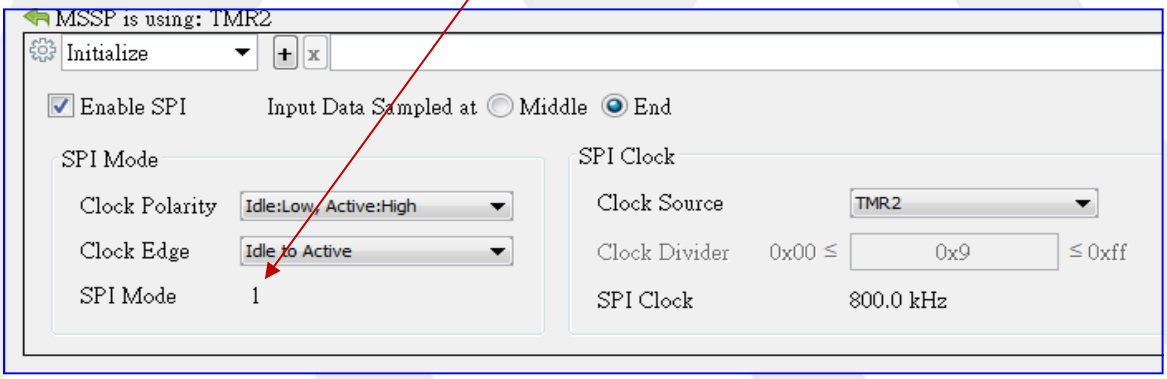
1. 開啟 MPLAB X IDE
2. 在 “File” 目錄下，開啟 LAB2_RGBLED.X 的專案
3. 在 “Tools” 目錄選擇 “Embedded “ 開啟” MPLAB Code Configurator” (MCC)
4. 開啟 TMR2::Timer 的設定。週期 (0.625uS，1.6MHz)



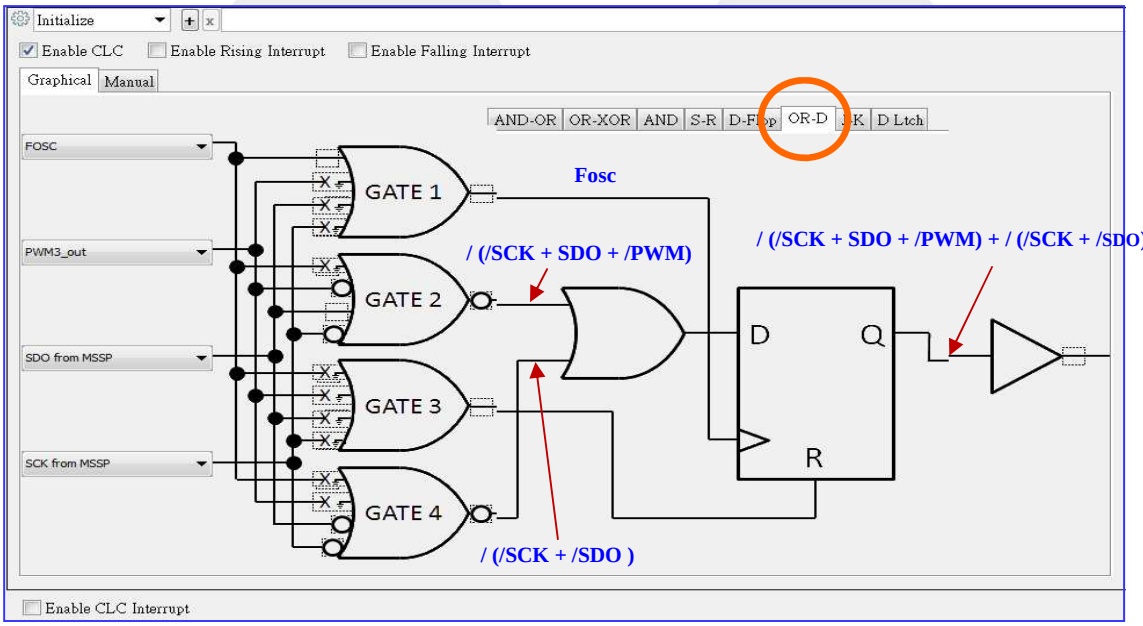
5. 開啟 PWM3::PWM 的設定。選擇 PWM3 的計時器為 Timer2。所以 PMW 的週期為 0.625uS。週期需求為 0.35uS，所以週期需求為 **0.35uS / 0.625uS = 56%**



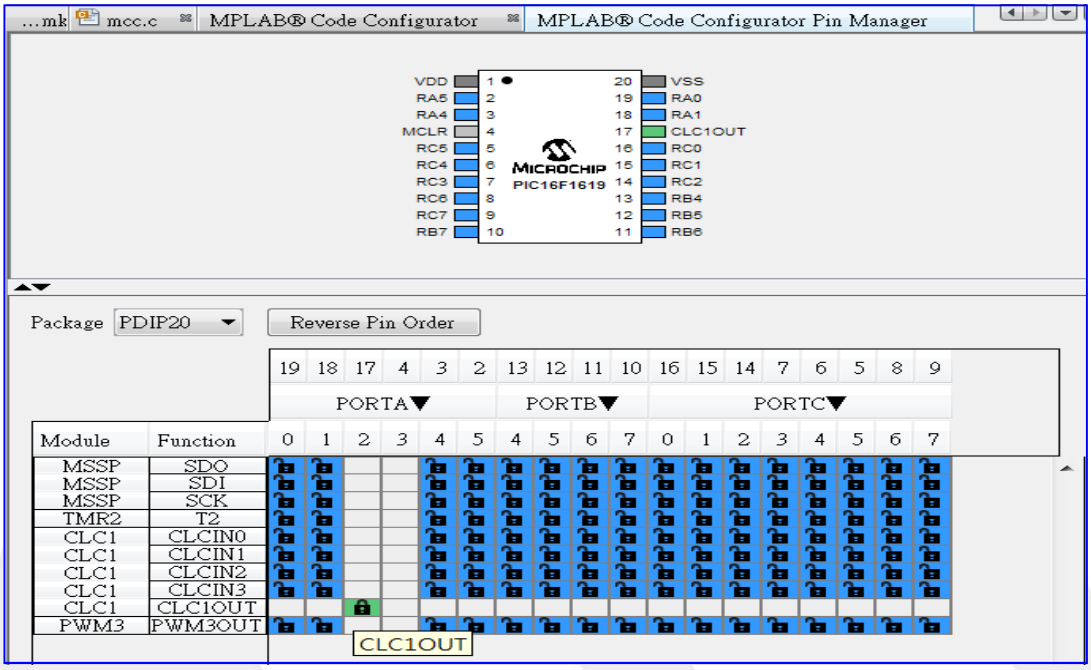
6. 開啟 MSSP::SPI Master 的設定。 設定 SPI Mode : 1, 也就是 SCK 平常是在 Low 的狀態 (即超過 50uS 的 Low 時就會 Reset WS2812)。SPI 使用 TMR2 1.6MHz 的時脈經 TMR2/2 (此為 SPI 使用 TMR2 的內定除二電路) 的輸入後就可取得與 PWM 同步的 CLK 800KHz 的傳輸速率。



7. 再來設定 CLC 的邏輯輸出做為 WS2912 的單線式控制訊號。選擇 CLC::CLC 依照最後的輸出布林代數 / (/SCK + SDO + /PWM) + / (/SCK + /SDO) 來設定 CLC 的連接



7. 最後在右上邊開啟 “Pin Manager” 選項, 設定 RA2 為 CLC1OUT。



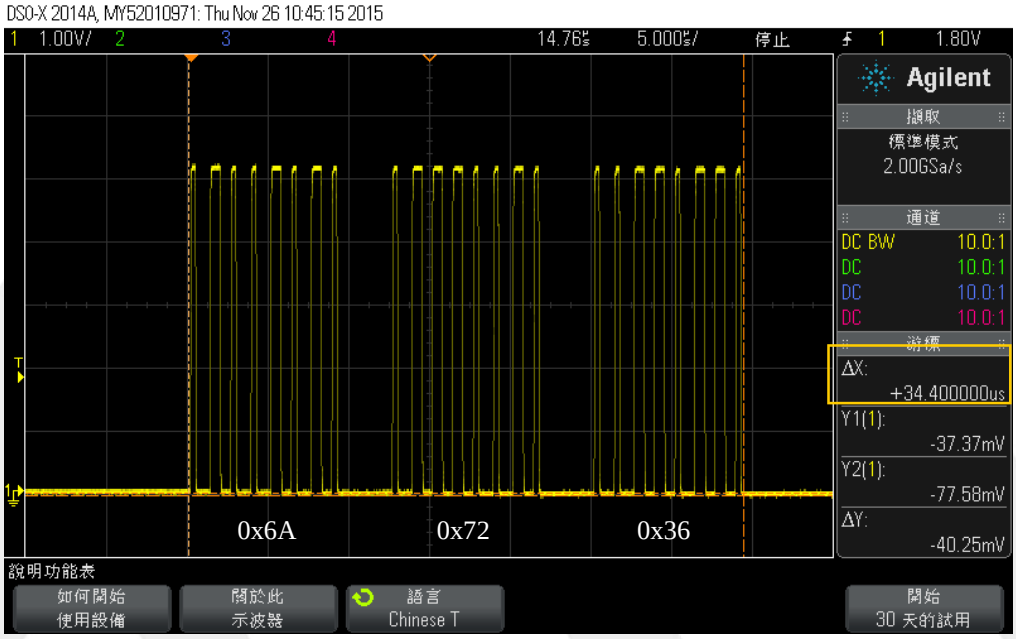
8. 按下 MCC 的 Generate Code 來產生 MCC 所設定的週邊函數即初始設定。最後檢查一下 Project 下的 main.c 。查看一下 SPI 是如何送出 24-bit 的控制訊號，並試著修改輸出的顏色亮度。

9. 底下為 main.c 的部分程式:

```
union{                                  //  宣告 24-bit 三色的資料及 8-bit 各單色資料
        uint24_t Color;
        struct {
                unsigned char Green;
                unsigned char Red;
                unsigned char Blue;
        } RGB;
} Disp_Color;

Disp_Color.Color= 0x808080;             // 初始顏色的設定
 __delay_ms(50);                        // Reset WS2812

while (1) {
        // 送出 24-bit SPI 資料
        SPI_Exchange8bit(Disp_Color.RGB.Green);      // 綠
        SPI_Exchange8bit(Disp_Color.RGB.Red);        // 紅
        SPI_Exchange8bit(Disp_Color.RGB.Blue);       // 藍

        Disp_Color.RGB.Green = (Disp_Color.RGB.Green +1) & 0x7F;    // 改變顯示顏色
        Disp_Color.RGB.Red = (Disp_Color.RGB.Red +5) & 0x7F;
        Disp_Color.RGB.Blue = (Disp_Color.RGB.Blue -1) & 0x7F;

      __delay_ms(50);                   // Reset WS2812  (CLC1OUT 持續 Low)
 }
```

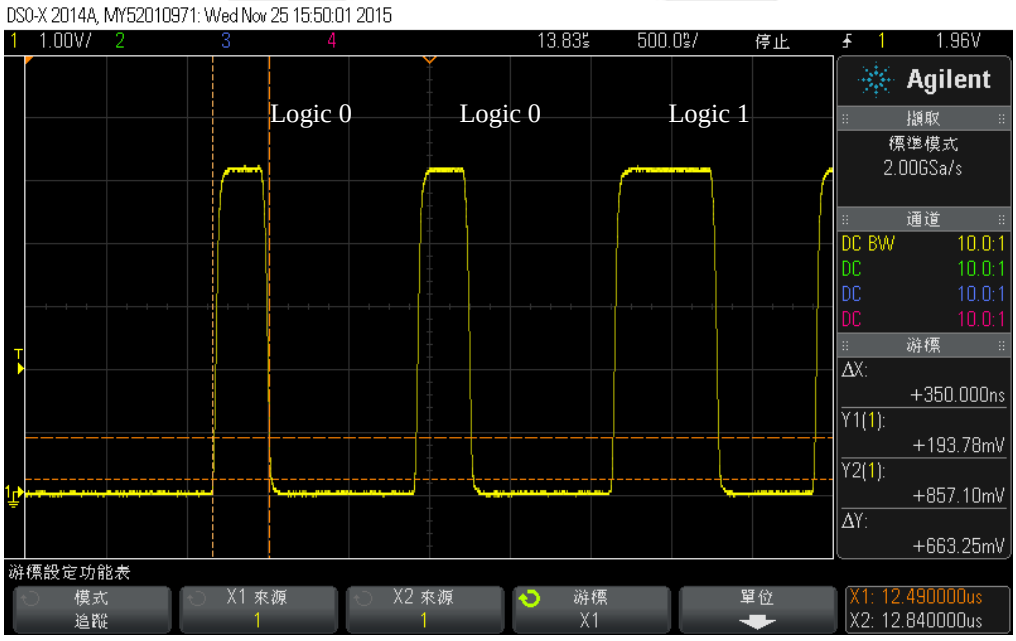
輸出訊號實測 (CLC1OUT) :

- 圖一為實際在示波器所抓到的 24-bit 輸出訊號 (RA2/CLC1OUT)。24-bit 傳輸時間為 34.4uS，所傳送的資料為 01001010, 01110010, 00110110 (綠 0x6A ，紅 0x72，藍 0x36)



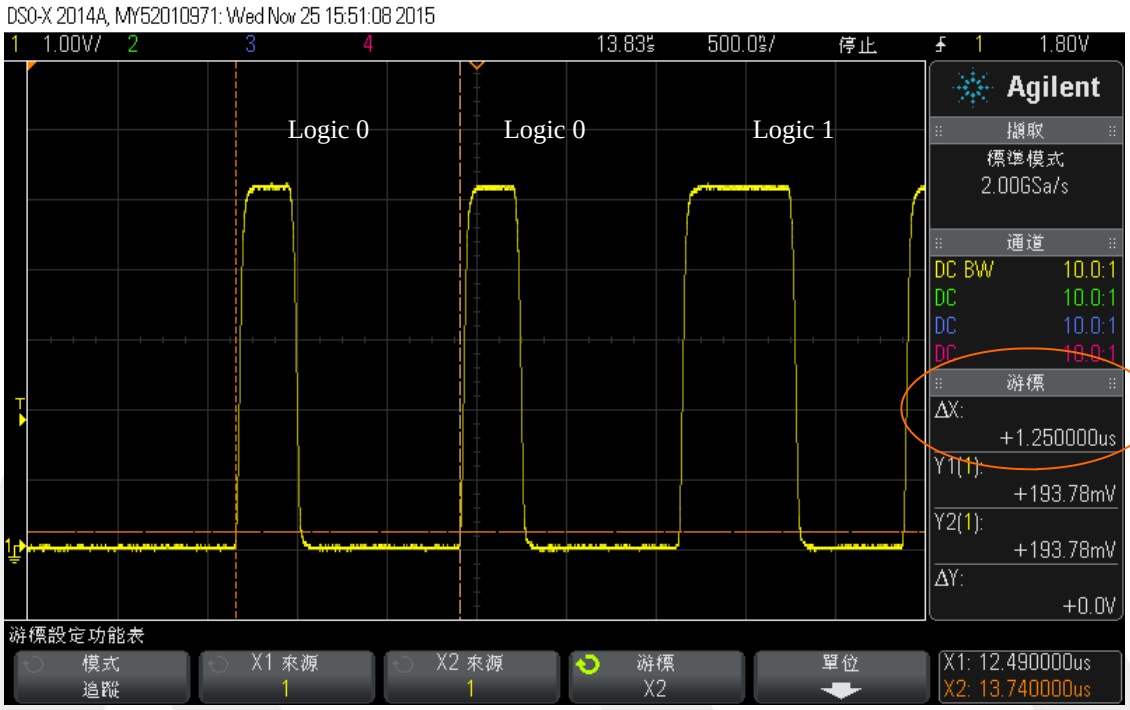
(圖一)

- 圖二為邏輯 “0” 編碼中 T0H 的脈波寬度為 0.35uS，T0L 為 0.90uS，符合 WS2812 的編碼時間。



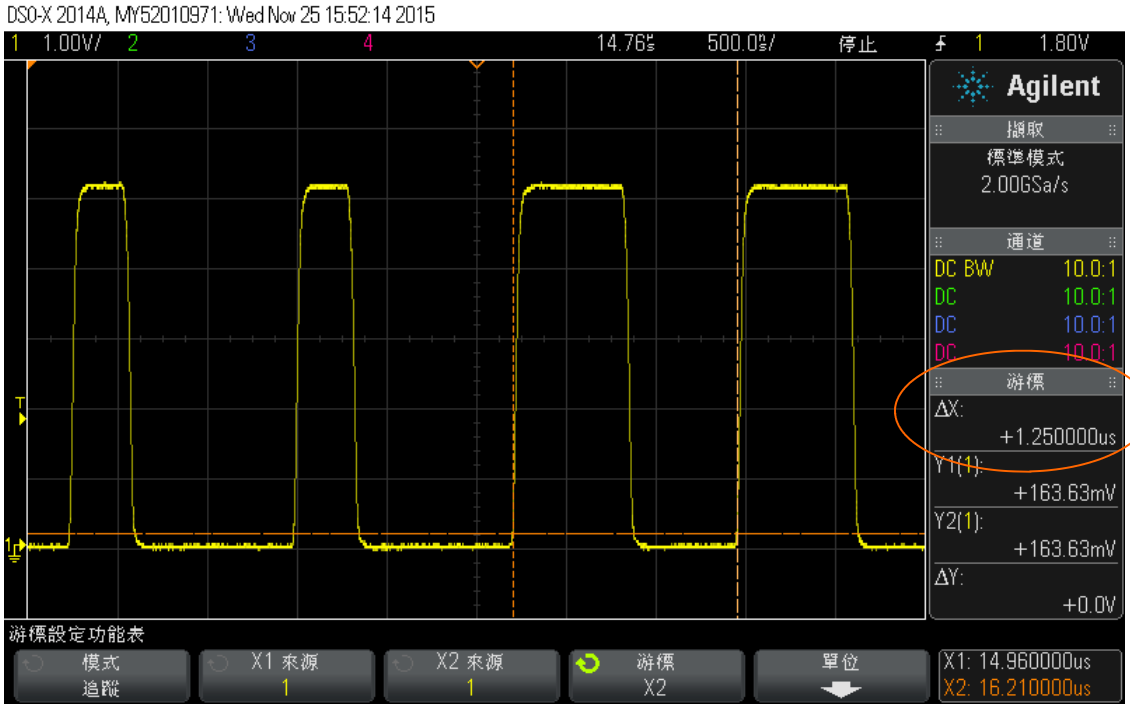
(圖二)

- 圖三所量測的是實際的 Logic 0 的編碼輸出及時間，示波器所測得的時間為 1.25uS。



(圖三)

- 圖四所量測的是實際的 Logic 1 的編碼輸出及時間，示波器所測得的時間為 1.25uS。

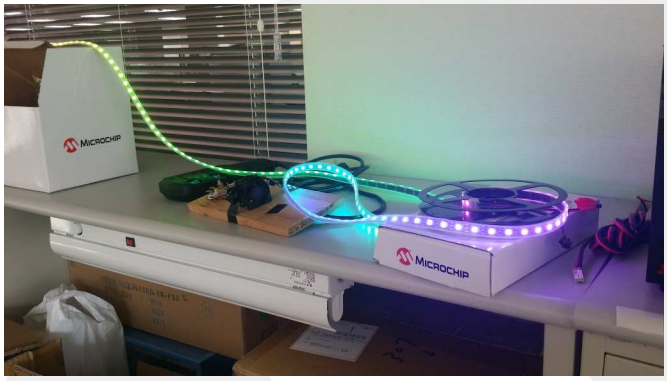


(圖四)

Lab2-1 是延伸的練習，藉由修改 main.c 的程式來達到 G、R、B 及白色的呼吸燈顯示範例。

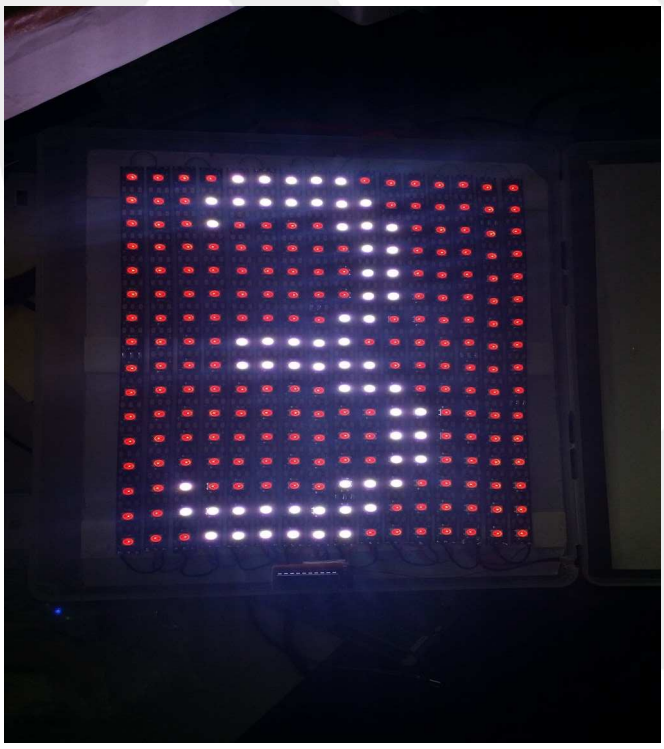
1. 開啟 Lab2-2_RGBLED.X
2. 直接編譯及燒錄，看看 LED 的變化。

底下圖片為 WS2812 配合 CIP 所做出來的一些應用

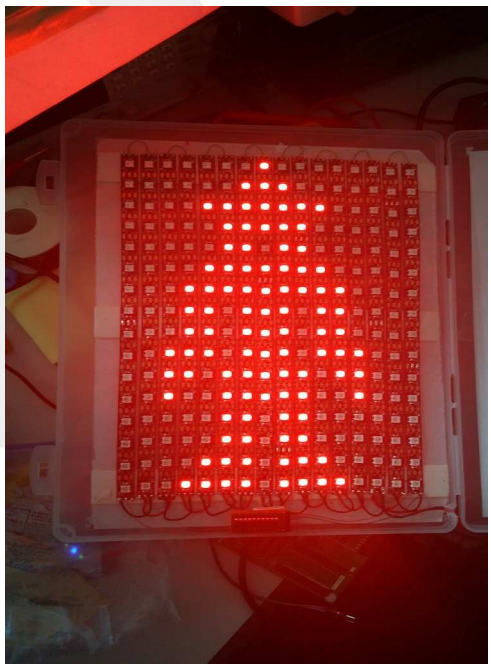


300 顆 RGB LED 顯示軟條

動態行人號誌顯示版



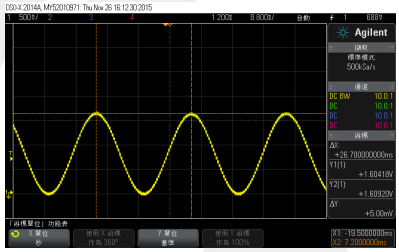
大型數字顯示板



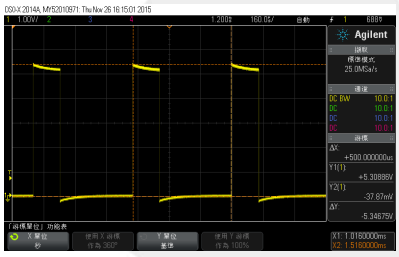
行人號誌顯示版

Lab 3

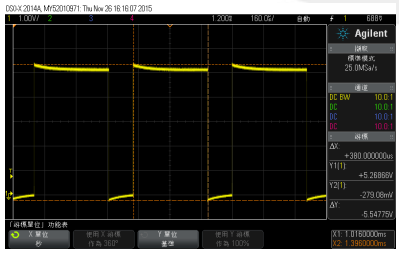
利用 SMT 周邊來測量子板上的 PWM 輸出的 Duty Cycle



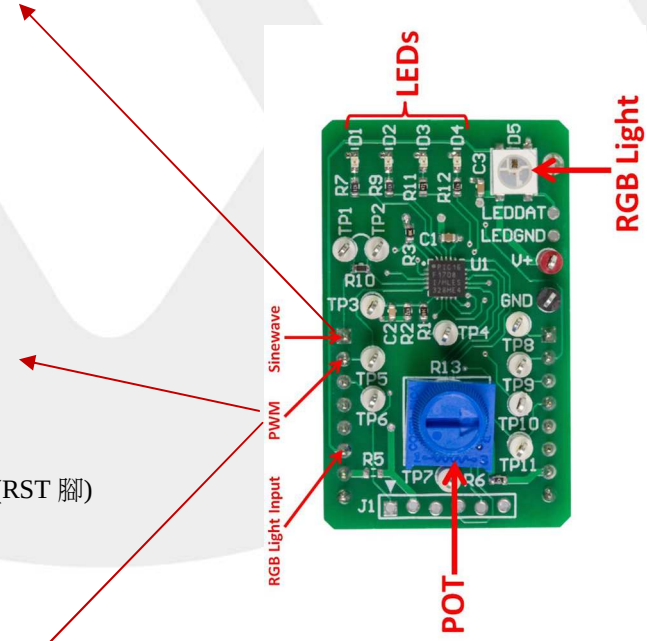
ZCD 37Hz 正弦波輸出 (AN 腳)



2mS 週期的PWM 25% Duty 輸出 (RST 腳)



2mS 週期的PWM 75% Duty 輸出 (RST 腳)



Lab 3:
SMT Duty Cycle 或週期的量測

目標：

確定 Curiosity 板子上的 LEDs (D4 ~ D7) 的顯示與子板上的 LEDs (D1 ~ D3) 所顯示的位置一樣。

D1 亮： < 25%， D2 亮 : 26%~50%， D3 亮 : 51% ~ 75%， D4 亮 : >76%.

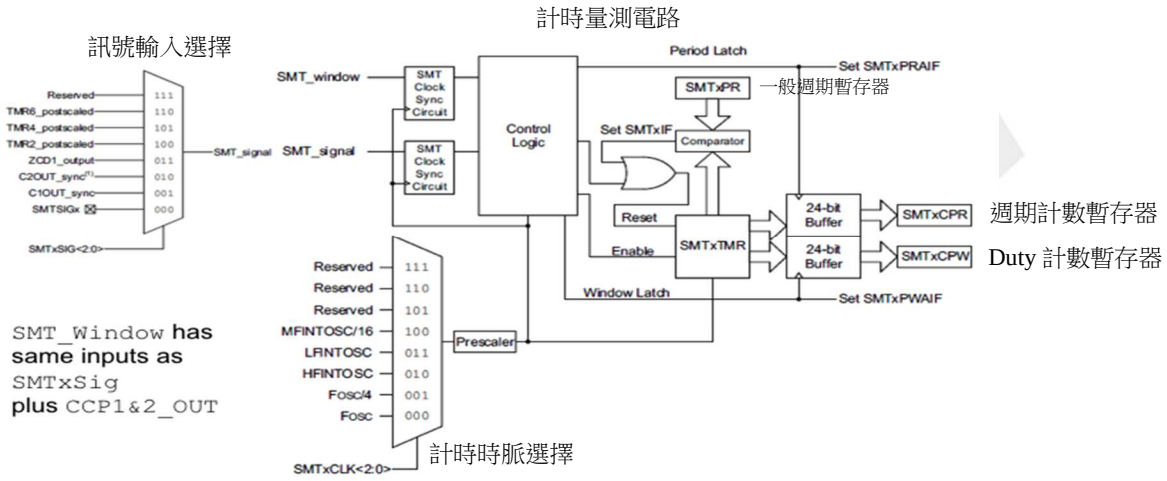
概述:

子卡已經寫好程式會自 RST 腳位送出一個可由可電電阻(R13) 調整的 PWM 輸出訊號到 Curiosity 母板上的 AN4/SMT1SIG 腳位做為 SMT 的訊號輸入。調整 R13 可以看到 D1 ~ D4 會依不同的 Duty 來點亮。

同樣的，本練習一樣使用 SMT 週邊的 “Period and Duty Cycle Acquisition” 模式來測量 duty cycle 後控制 Curiosity 板子上的 LEDs 的顯示方式同子板的 LEDs。

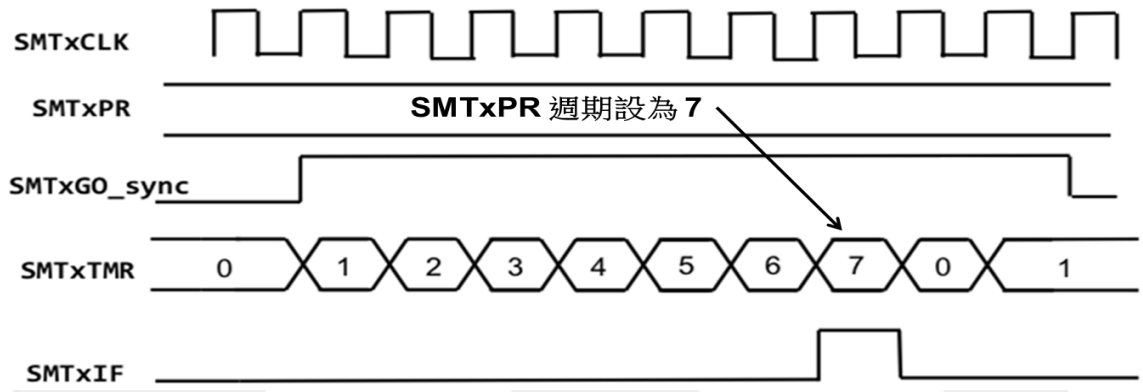
SMT 動做說明:

下圖為 SMT 方塊圖示，訊號輸入多工選擇，選擇待測量訊號。計時時脈選擇，選擇測量的時脈來源。最後量測結果會送到 “SMTxCPR” 週期計數暫存器，及 “SMTxCPW” Duty 計數暫存器。



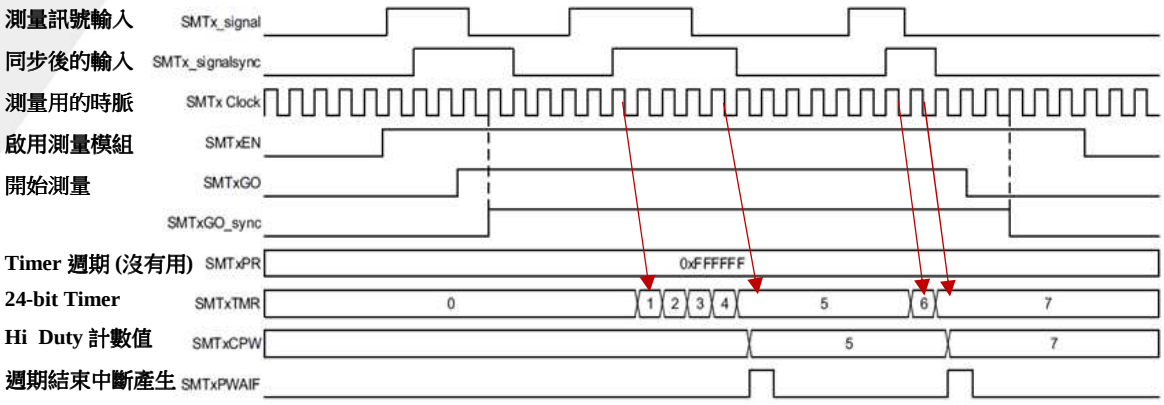
SMT 工作模式：

1. 一般 24-bit 比較器模式計時器 (24-bit Timer)



就如一般比較式的 Timer 一樣，設定好週期 (SMTxPR) 就可以重複計時並產生中斷 (SMTxIF)。這是最簡單的應用。

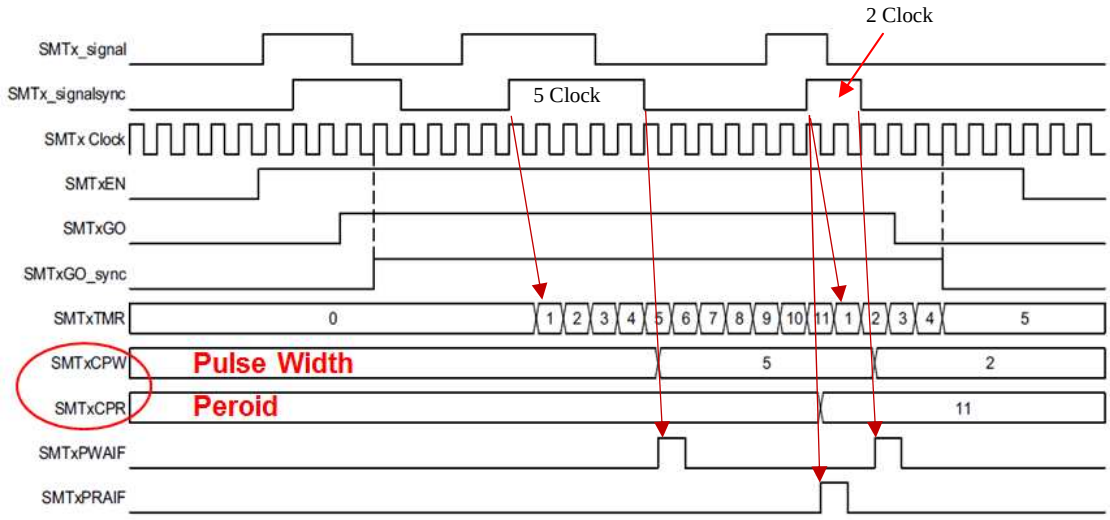
2. Gated Timer Mode (Repeat Acquisition) : Lab3



第一個 Duty 量測為 5 - 0 = 5，第二 Duty 量測為 7 - 5 = 2

由上圖所示，以 Gate Time Mode 來看，SMTxCPW (Duty 計數暫存器) 在啟動量測 (SMTxGO =1) 就會累計輸入訊號的 Hi Duty 的計數，Low Duty 就會停止計數，等到下一個 Hi Duty 進來時繼續計數。每次在輸入脈波下下緣時會產生中斷。如果在中斷裡將 SMTxTMR 歸零則可重新測量輸入的 Duty。
所以，了解測量模式示很重要的，設定時最好能先了解一下 Data Sheet 裡所做的說明。

3. Period and Duty Cycle Acquisition (Repeat Acquisition)： Lab3-1



此種模式下可以對輸入的脈波輕鬆的同時測得 Duty 及 Period。由上圖看出當 SMTxGO = 1 時開始啟用測量功能；第一個 Hi Duty 量測出 5 的計數且在下緣時產生中斷 (SMTxPWAIF)，Period 繼續計數直到下一個 Hi Duty 進來，此時新的上緣會觸發中斷 (SMTxPEAIF) Period 計數值為 11。

上圖也明顯看出在 Repeat 模式下，每次 Duty 暫存器 (SMTxCPW) 所抓的 Duty 都是新計數值，以上圖為例：第一個輸入脈衝的 Duty 是 5，其週期為 11。第二個輸入的 Duty = 2 ，因為 SMTGO = 0 所以不再計數了。

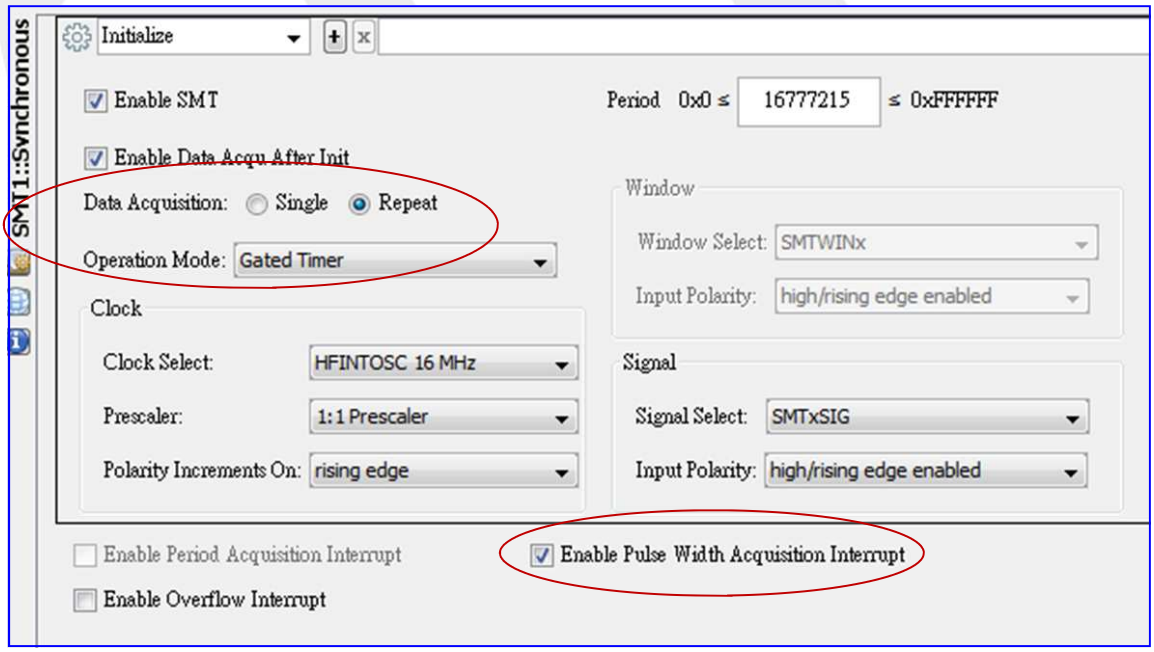
概述:

子卡已經寫好程式會自 RST 腳位送出一個可由可電電阻(R13) 調整的 PWM 輸出訊號到 Curiosity 母板上的 AN4/SMT1SIG 腳位做為 SMT 的訊號輸入。調整 R13 可以看到 D1 ~ D4 會依不同的 Duty 來點亮。

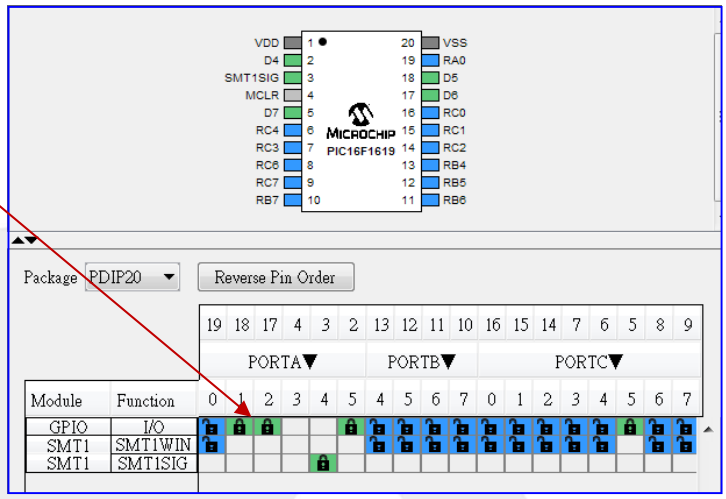
同樣的，本練習一樣使用 SMT 週邊的 “Period and Duty Cycle Acquisition” 模式來測量 duty cycle 後控制 Curiosity 板子上的 LEDs 的顯示方式同子板的 LEDs。

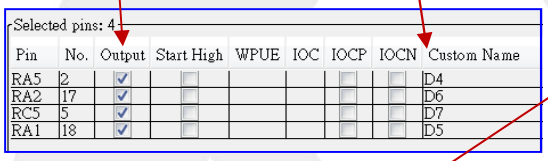
Lab3 開始做實驗

1. 開啟 Lab3_SMT_GatedTMR_LED.X
2. 啟用 MCC
3. 設定 SMT 工作模式為 “Gated Timer” 及啟用資料獲取模式為 “Repeat”，並 “ Enable Pulse Width Acquisition Interrupt”
4. 輸入脈波的 Duty-Cycle 的計數會存放在 SMT1CPW 暫存器，在中斷裡做 Duty 的比較並依比較結果設定 LEDs 的顯示。
5. 在 Gated Timer 的時脈圖可知，Duty 的計數是隨 SMT1TMR 往上加的，它並不會在新的 Duty 進來時歸零，所以要在中斷裡將 SMT1TMR 歸零以做下次新 Duty 的計數。



6. 關於腳位的設定如右圖所示，先選擇 RA1, RA2, RA5 & RC5 做為 LED 的輸出腳 (GPIO 綠色上鎖的部分)。接下了在 GPIO::GPIO 的設定項裡設為輸出腳位及賦與 Customer Name。MCC 會自動設定 LEDs 的腳位定義，也會啟動 Peripheral Pin Select 做 SMT1SIG 的腳位設定 (在此為 RA4 做為 SMT1SIG 量測訊號輸入腳)。





8. 因中斷有用到 LEDs 的腳位定義，所以需加入 pin_manager.h

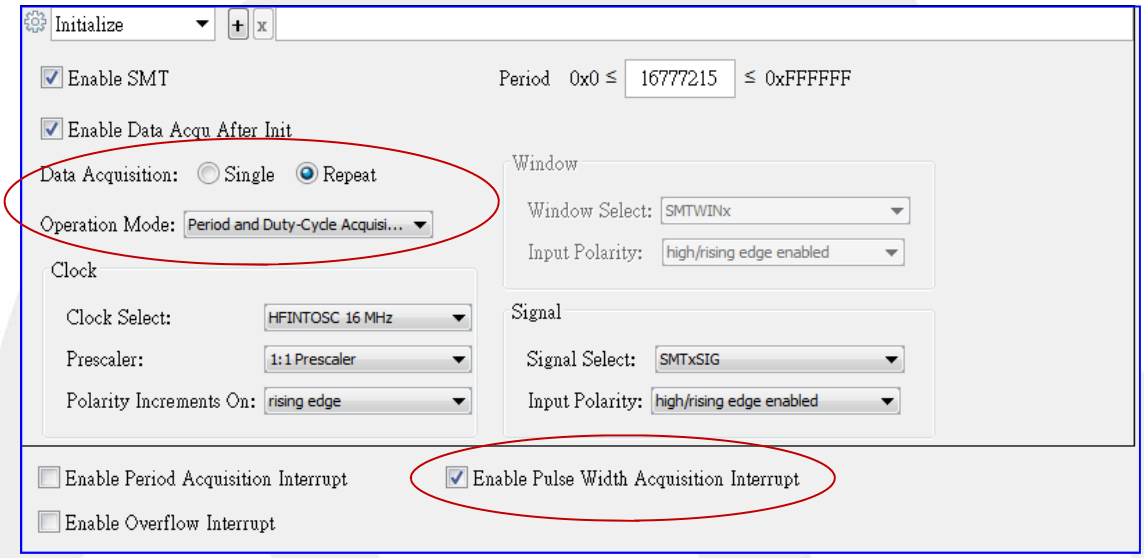
```
#include <xc.h>
#include "smt1.h"
#include "pin_manager.h"
```

9. Duty 是在 SMT1PWAIF 的中斷來處理 Duty 的比較並控制 LEDs 的顯示。最後要離開中斷時在將 SMT1TMR 歸零，做下一個 Duty 的計數準備。

```
void SMT1_PW_ACQ_ISR(void) {

   // Clear SMT1 pulse width acquisition interrupt flag bit.
   PIR4bits.SMT1PWAIF = 0;
    if(SMT1CPW > 6000)
   {
      D4_SetHigh();
      D5_SetLow();
      D6_SetLow();
      D7_SetLow();
   }
   else if(SMT1CPW > 4000)
   {
      D4_SetLow();
      D5_SetHigh();
      D6_SetLow();
      D7_SetLow();
   }
   else if(SMT1CPW > 2000)
   {
      D4_SetLow();
      D5_SetLow();
      D6_SetHigh();
      D7_SetLow();
   }
   else
   {
      D4_SetLow();
      D5_SetLow();
      D6_SetLow();
      D7_SetHigh();
   }
   SMT1TMR = 0;       // SMT1 Timer = 0;
}
```

Lab3-1 開始做實驗

1. 開啟 Lab3-1_SMT_Grt_Duty.X
2. 啟用 MCC
3. 設定 SMT 工作模式為 “Period and Duty Cycle Acquisition” 及啟用資料獲取模式為 “Repeat”，並啟用 “Pulse Width Acquisition Interrupt”



4. 程式改用主程式來判斷 Duty 來顯示 LEDs。在中斷程式裡加入一個用 bit 所宣告的旗號 SMT_GO_Flag，主程式 main.c 來判斷此旗號是否設定來決定是否執行 Duty 的比

main.c

```
bit SMT_GO_Flag;

while (1)
  {
    if (SMT_GO_Flag)
    {
      SMT_GO_Flag = 0;

      if(SMT1CPW > 6000)
      {
        D4_SetHigh();
        D5_SetHigh();
        D6_SetHigh();
        D7_SetHigh();
      }
      else if(SMT1CPW > 4000)
      {
        D4_SetLow();
        D5_SetHigh();
        D6_SetHigh();
        D7_SetHigh();
      }
      :
      :
```

在 smt1.h 裡加入的位元旗號宣告：

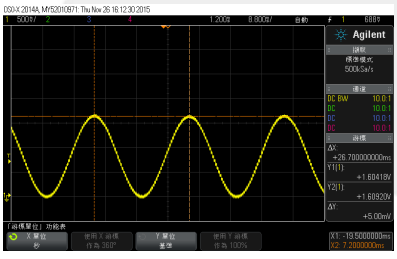
```
extern bit SMT_GO_Flag;
```

在 smt1,c 程式裡加入處理旗號ㄐ：

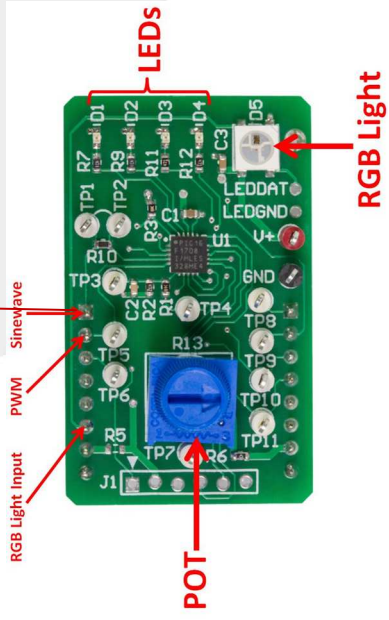
```
void SMT1_PW_ACQ_ISR(void) {

    // Clear SMT1 pulse width acquisition
interrupt flag bit.

    PIR4bits.SMT1PWAIF = 0;
    SMT_GO_Flag =1;

}
```

Lab 4

利用 ZCD 周邊來測量子板上所輸入的弦波偵測出電壓零交越點



ZCD 37Hz 正弦波輸出 (AN 腳)



Lab 4:
零點交越電壓偵測 (ZCD)

目標:

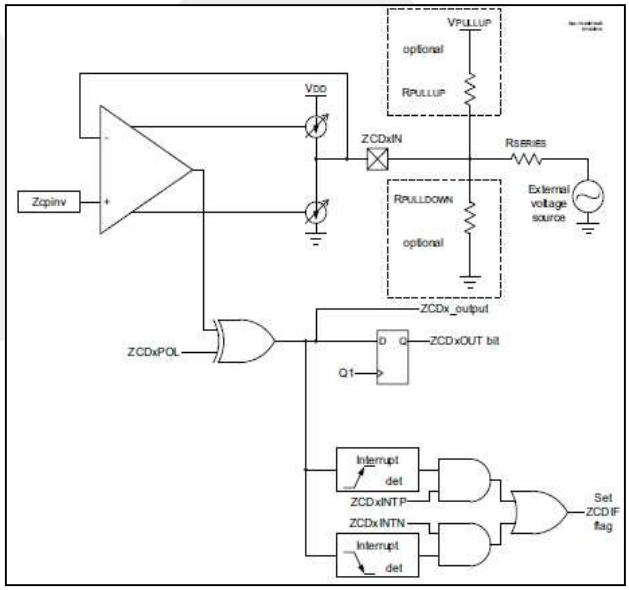
利用 ZCD 週邊來測量輸入訊號的零電壓交越點，測得上升的交越點時輸出 Hi 訊號及測得下降的交越點時輸出 Low。

概述:

零點交越電壓偵測 (ZCD) 模組可輕鬆的來偵測交流信號 (AC) 跨越地電位 (Ground) 的偵測。不管是由正電壓下降至負電壓或是由負電壓升到正電壓都可以偵測。ZCD 內建一個 0.75V 的參考電壓，所以只要輸入電壓在此電壓 (0.75V) 交越即可被偵測出來。
ZCD 通常用來偵測交流電源，主要的應用有:

1. 交流電源的週期
2. 長時間對交流的精確量測
3. 調光用的精確相位測量
4. 低 EMI 切換週期控制

ZCD 輸入需要一限流電組以防止模組的損壞，通常此一限定電流需小於 350uA。由右圖 ZCD 方塊圖所示，輸入訊號經電組輸入至 ZCDx_IN 腳位候會被內部的參考電壓 (Zcpinv) 經比較器與輸入電壓比較後會一直被箝制在 0.75V， 即 ZCDxIN 會一直維持在 +0.75V 的直流準位上，比較器的輸出經極性控制後輸出 ZCDxOUT。

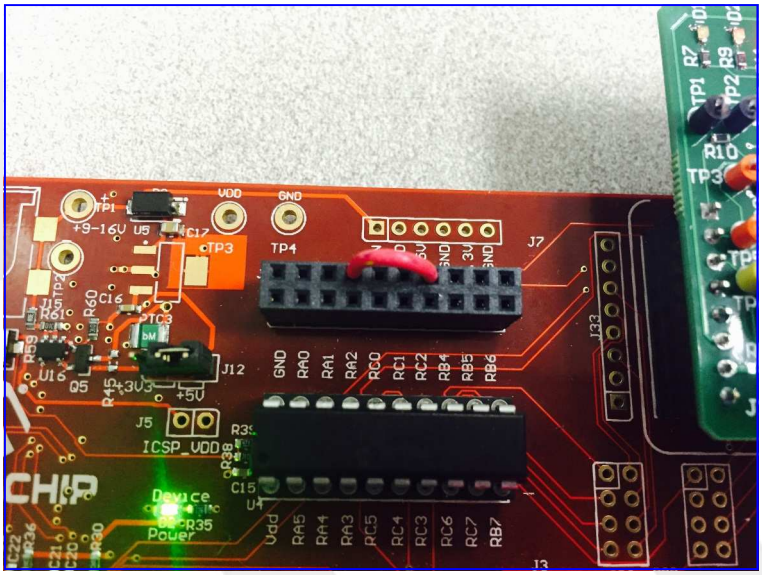


實驗的接線：

子板上的弦波輸出 TP3 經串聯電阻(18K ohm)後接到 RC2 輸出一正弦波。

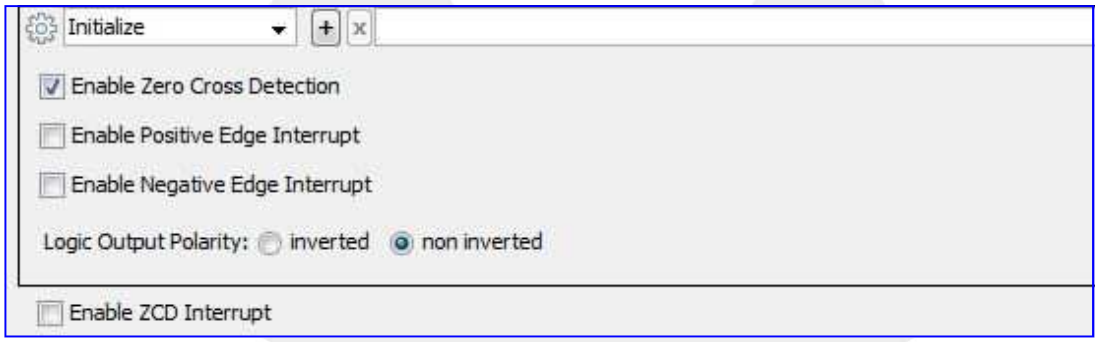
RC2 為正弦波輸出
RA2 規劃為 : ZCD1IN
RA5 規劃為 : ZCDOUT

所以需跳接一條 RC2 到 RA2 的跳線。

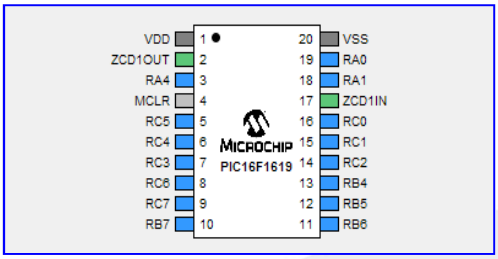


Lab4 開始做實驗

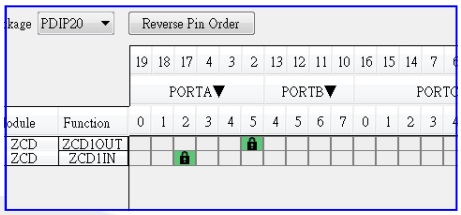
1. 開啟 Lab4 _ZCD_Basic.X
2. 啟用 MCC
3. 設定 ZCD::ZCD 模組功能，如下設定所示： 只要啟動 ZCD 並將其輸出不做反向。



4. 由於 ZCD1 的腳位輸入 ZCD1IN 是固定在 RA2 的腳位，但 ZCD1OUT 的腳位是可以經由 PPS 做設定的，在此實驗我們設定 ZCD1OUT pin 為 RA5 (與 D4 同腳位)。
5. 注意事項： 在實驗中 MCC 2.25.2 中 ZCD 所設定的腳位並不會隨 MCC 的開啟就將 Pin Manager 之前的設定載入進來。這應該是 MCC 的小”八哥鳥”(Bug) 的問題。**所以記得開啟 Lab4 時需重設一下 Pin Manager 下的 ZCD1IN (RA2) 及 ZCDOUT (RA5)。**

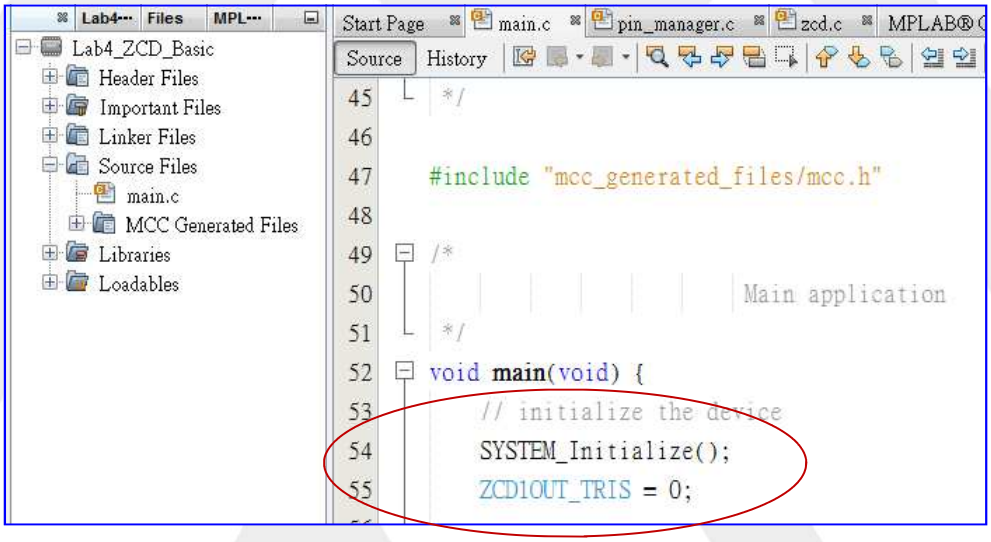


ZCD 腳位設定圖



這兩隻腳在重新載入 MCC 時會消失，需重新設定後實驗才會正確。

6. 設定完後即可產生 MCC 設定程式。
7. 開啟 main.c 加入一行程式 ZCD1OUT_TRIA = 0; 將 ZCD1OUT 設成輸出腳位。

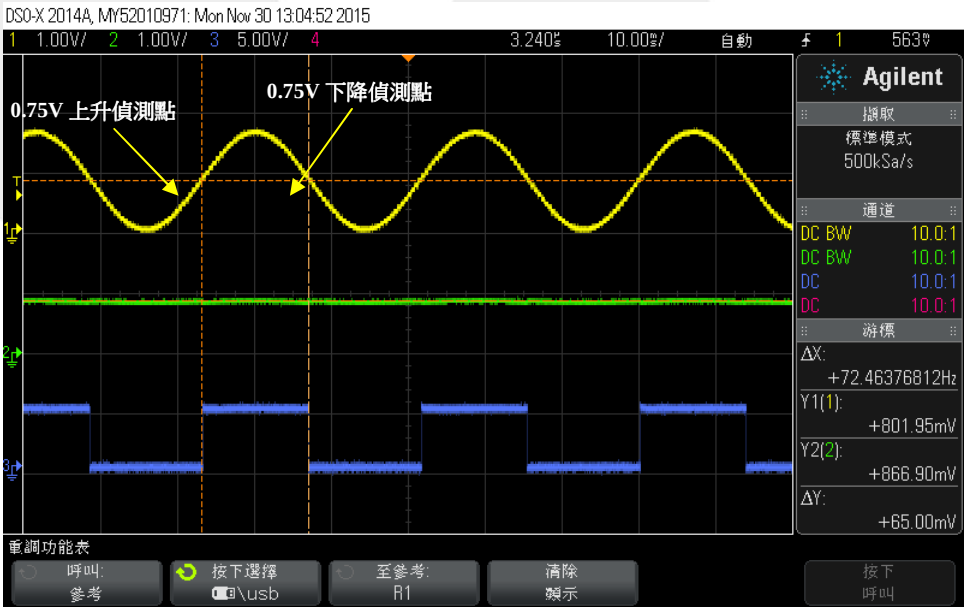


實測訊號輸出：

1. 輸入 TP3 的正弦波，轉換點在 0.75V。

2. RA2 (ZCD1IN) 輸入腳波形，以被箝制在 0.75V 的電位。

3. RA5 (ZCD1OUT) 輸出波形，其上緣雨下降緣點都發生在 0.75V 的轉換電壓點。



Lab 5

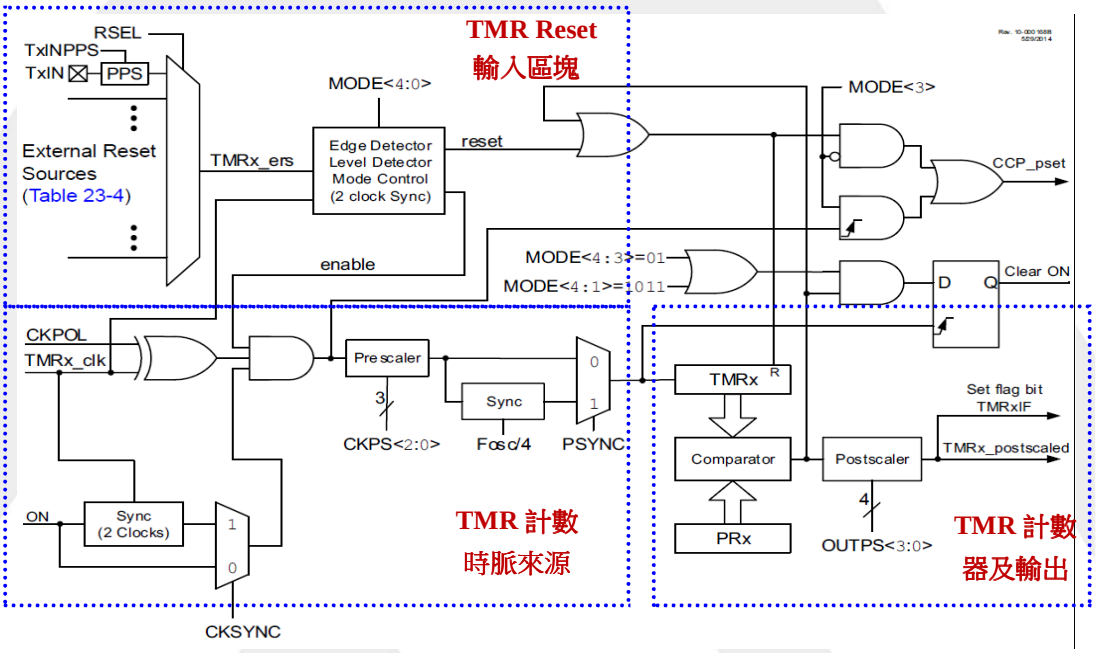
利用 ZCD 偵測上升弦波交越點後，啟動 TMR2/HLT 的 Mono-Stable 控制 PWM3 輸出 1.6mS 的脈波

以上的動作無需主程式的執行，經設定後 CIP 會自行運作

在進入本實驗之前我們先要了解一下新的 HLT/Timer (Timer with Hardware Limited Timer) 的功能。例如 HLT/TMR2 可以當作是標準的 Timer2 加上 HLT 的功能。也就是因為加入 HLT 的功能讓整個 Timer2 變的複雜許多但功能也變的強大。除了支援傳統 Timer2 比較器模式的計時功能外，還增加了下列的功能:

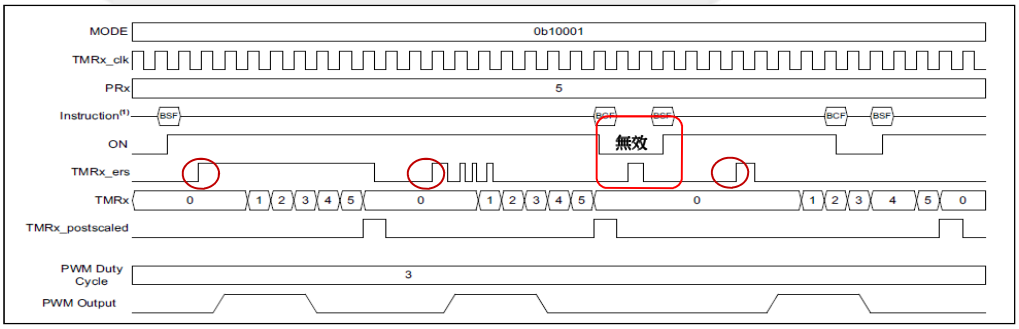
1. Free Running Period (Roll Over Pulse) (自由運行計數模式)
2. One-Shot (單次觸發計數模式)
3. Mono-Stable (單穩態計數模式)

底下是 TMR2/4/6 + HLT 的方塊圖，主要是由三個區塊組成。



1. TMR Reset 輸入區塊: 此區藉由多項週邊的輸入來控制 Timer 的計數。主要週邊選擇有 PWMx，CLCx，TMRx，CCPx，CxOUT，ZCD1 等。在本實驗將選擇 ZCD1 做為計數的控制訊號。
2. TMR 計數時脈來源: 同樣的也是眾多可選的時脈輸入，在這裡我們選擇 Fosc/4 做為Timer2 的計時時脈輸入。
3. TMR 計數器與輸出: 更傳統的 Timer2 一樣，輸出可產生中斷或經後除器再輸出

實驗中的 TMR2 +HLT 將會做為 PWM 的週期產生器，只不過該 PWM 週期並不是採用自由運行計數模式在運作，而是採用非穩態運行模式，也就是有人敲你一下你就動做一次的方式。ZCD 的偵測就是這敲擊的動作 (如下圖裡的 TMR2_ers) 這時 TMR2 開始計數直到 TMR2 = PR2(週期設定暫存器) TMR2 歸零，直到下一個 TMR2_ers 再次觸



Rising Edge-Trigger Mono-Stable Mode Timing (MODE= 100001)

Lab 5:
HLT/Timer2 的應用

目標：

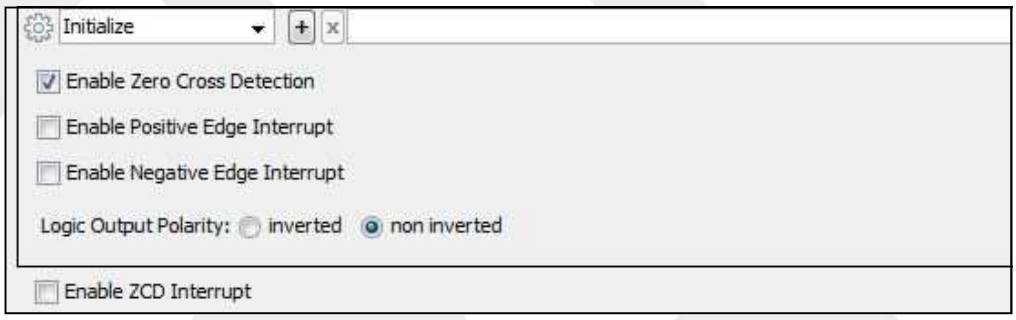
了解 HLT/TMR2 的設定與使用。利用 ZCD 偵測到上緣的零交越點時觸發 TMR2 及 PWM3 來產生一固定的 1.6mS 的脈波

概述:

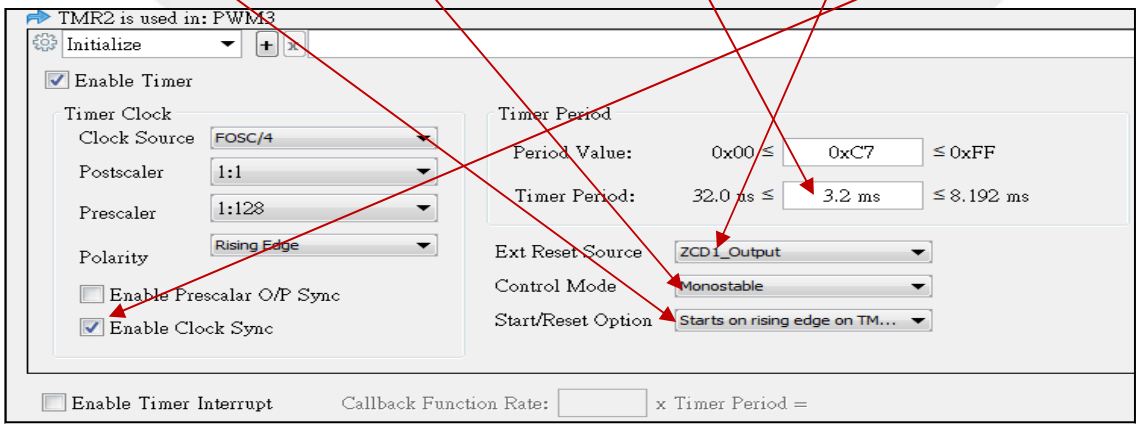
因為使用的是觸發模式，所以會設定 HLT/TMR2 在 “Monostable” 模式下做設定。也就是 ZCD 每觸發一次 THR2 就計數一次，PWM 也同步產生 1.6 mS 的脈波。

Lab5 開始做實驗

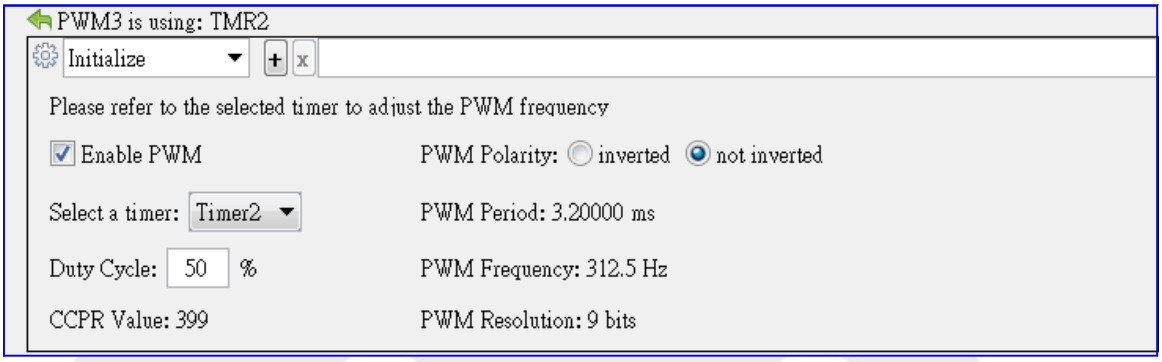
1. 開啟 Lab5 _ZCD_HLT_PWM.X
2. 啟用 MCC
3. 設定 ZCD



4. 設定 Timer2 為 “Monostable” 模式。外部重置的觸發源為 ZCD1_Output，計數的啟動是在上升緣時啟動。此時 Timer2 的週期設為 3.2mS (手動輸入)，並啟用時脈同步功能。



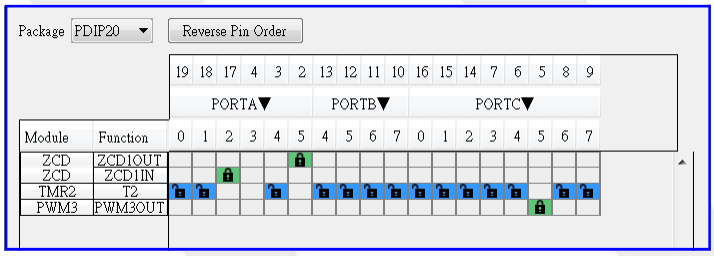
5. 再過來就是設定 PWM3 了。設定 PWM3 的週期計時器為 TMR2，如此 PWM3 的週期為 3.2ms，如果要輸出 1.6mS 的 Duty Cycle 就要設定 1.6mS / 3.2mS = 50%



6. 最後再設定一下腳位的輸出 (ZCD1OUT & PWM3OUT) 。如同上 Lab5 ZCD 的實驗一樣，重新載入專案時，MCC 裡的 Pin Manager 的設定會遺失 ZCD 的腳位設定 (ZCD1IN & ZCD1OUT)。請重新加入腳位之設定如右圖所示:

ZCD1_OUT 設在 RA5

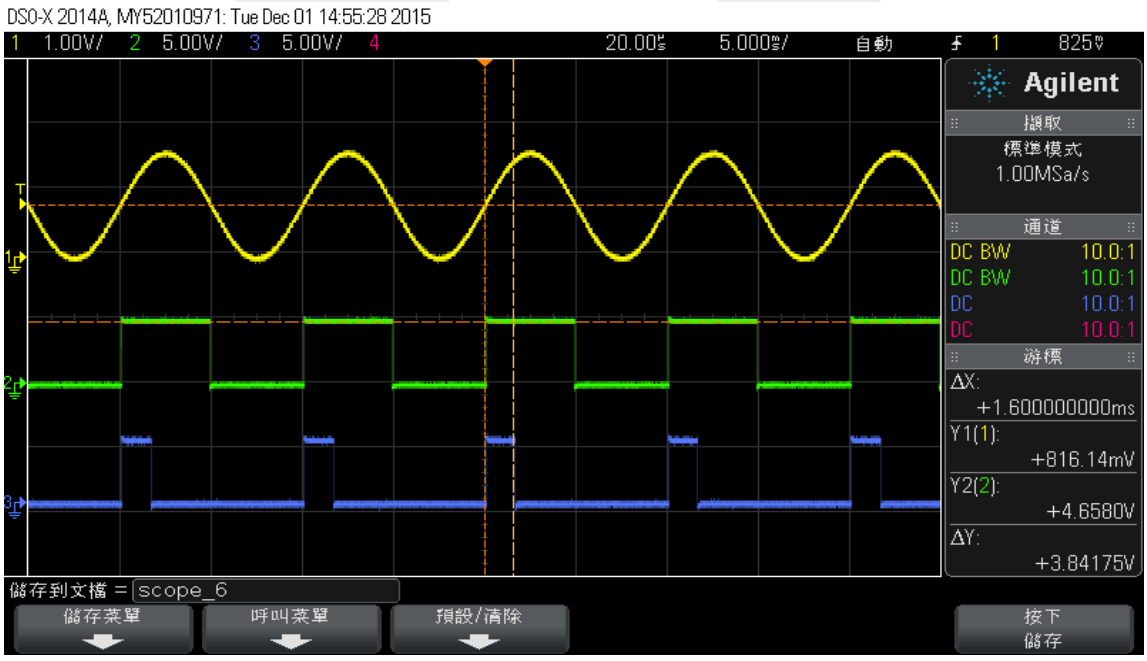
ZCD1IN 設在 RA2



實際實驗的測量：

1: TP3 的輸入的正弦波　　2.: ZCD1OUT (RA5) 的偵測轉態輸出

3. PWM3 (RC5) 的 1.6mS 固定寬度的脈波輸出



Lab 6

1. 利用 ZCD 偵測上升弦波交越點
2. 觸發 TMR2/HLT 的 Mono-Stable 輸出單次的 1.28 mS 的脈波
3. 在此 1.28mS 寬的時間框下啟動以 TMR4 的基礎的 PWM3 輸出 16 個 Duty Cycle 30% 的脈衝

以上的動作無需主程式的執行，經設定後 CIP 會自行運作

Lab 6:

當 AC 輸入到零點交越時利用 CIP 來產生連續 16 個觸發脈衝來讓 TRIAC 導通

目標:

基於前面 ZCD 與 HLT/TMR2 的組合，在將這組合駛驅動 PWM3 以產生一 16 個 Duty Cycle 為 30% 的連續脈衝

概述：

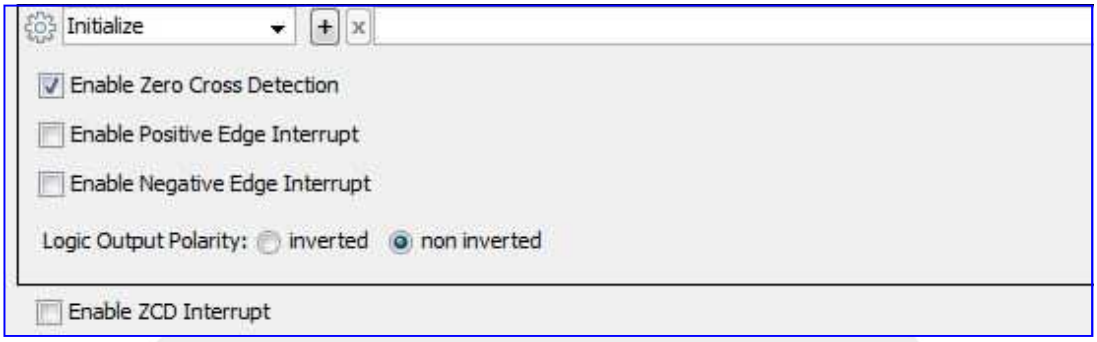
本實驗範例是利用 CIP 做 TRIAC 的連續導通的觸發，有時TRIAC 在電壓較低或觸發訊號的能量較弱時是不容易被觸發導通的，施與連續的觸發脈衝是不錯的做法。本實驗並沒有提及觸發角度的延遲做法，但實際可利用 ZCD 交越點配合 Timer 的延遲即可輕易完成。

本實驗將使用到的 CIP 的週邊有 ZCD、HLT/TMR2、HLT/TMR4、PWM4 及當做除錯輸出訊號用的 CLC。底下將一一說明其設定及工作需求。

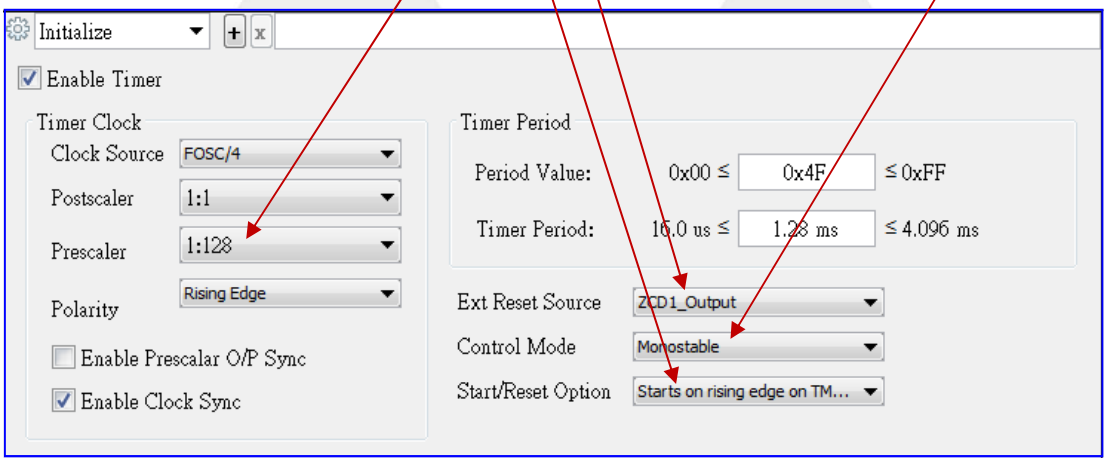
1. 實驗起起點是利用 ZCD 取得到一個上緣交越的觸發訊號為開始，接這這觸發訊號會利用內部連線的設定接給 HLT/TMR2 做一個單穩態計時器 (Monostable Mode) 的輸出，這的單穩態只要被 ZCD 觸發一次就會產生 1.28mS 的時間框。
2. 這個 1.28mS 的時間框再接到以 TMR4 的 PWM3 來產生 80uS 週期的 PWM 輸出。因為這是一個同步的動做且TMR4 是設定成 Free Run Mode 的工作模式，所以在 1.28mS 的時間框下就會做 16 次的 Free Run Over 的計數，也就是說 PWM3 會輸出 16 個 PWM 訊號 (1.28mS / 80uS = 16 次)。
3. PWM3 的輸出將設成反向輸出，即 Duty On 時為 Low，設定成 70% Duty Cycle 的輸出實際的輸出會是 30% 的 Hi 輸出。

Lab6 開始做實驗

1. 開啟 Lab7_ZCD_TRIAC.X
2. 啟用 MCC
3. 設定 ZCD::ZCD 模組功能，如下設定所示：只要啟動 ZCD 並將其輸出不做反向。

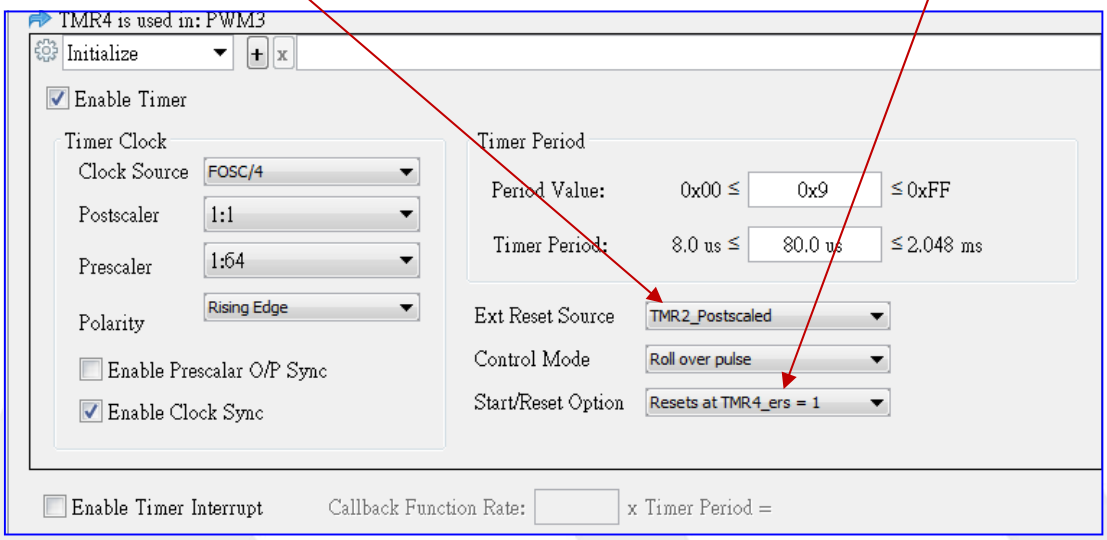


4. 由於 ZCD1 的腳位輸入 ZCD1IN 是固定在 RA2 的腳位，但 ZCD1OUT 的腳位是可以經由 PPS 做設定的，在此實驗我們設定 ZCD1OUT pin 為 RA5 (與 LED D4 同腳位)。
5. 注意事項：在實驗中 MCC 2.52.2 中 ZCD 所設定的腳位並不會隨 MCC 的開起就將之前的 Pin Manager 設定載入進來。這應該是 MCC 的小”八哥鳥”的問題。所以記得開啟 Lab4 時需重設一下 Pin Manager 下的 ZCD1IN (RA2) 及 ZCDOUT (RA5)。
6. 接下來將設定 HLT/TMR2。將 ZCD 的輸出當做是 TMR2 Reset 的輸入來源，也就一次 ZCD 的輸入將觸發 TMR2 一次並產生一次的 1.28mS 計時框。時脈仍是使用 Fosc=32MHz 的最高速。透過 1:128 的 Prescaler 降頻後送進 TMR2。
7. TMR2 的外部 Reset 來源為 ZCD1_Output，TMR2 工作模式為 “Monostable”，並設定啟動的時機為 “Starts on rising edge on TMR2_ers” 。使用手動方式修改 Timer Period 為 1.28mS，最後要勾選 “Enable Clock Sync” 的選項。



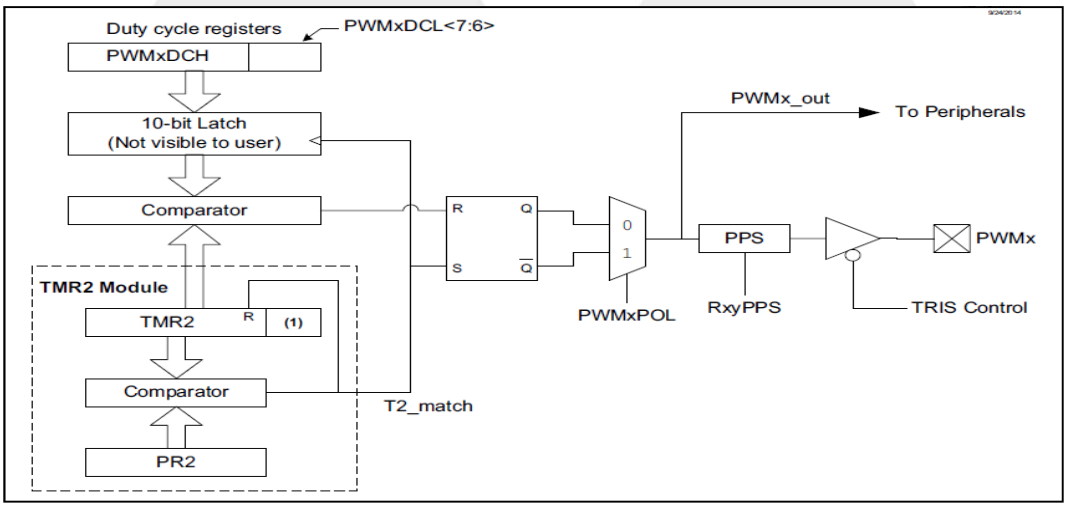
8. 設定好了 TMR2 的 1.28mS 時間框後，再過來就是要怎樣在這時間框內輸出 16 的脈波，1.28mS/16 = 80uS，所以要設定一個以 80uS 為週期的 PWM 輸出。因為 PWM 需要一個計時器來當作計時來源，所以選用 TMR4 當做 Period 的產生來源。且 TMR4

在這 1.28mS 的時間框裡需自由運轉計數所以需設定成 “Roll Over Pulse” 模式，啟動條件為 “TMR2_Postscaled” 的輸入 (1.28mS) 時所產生的 “Reset at TMR4_era=1” 的控制方式。

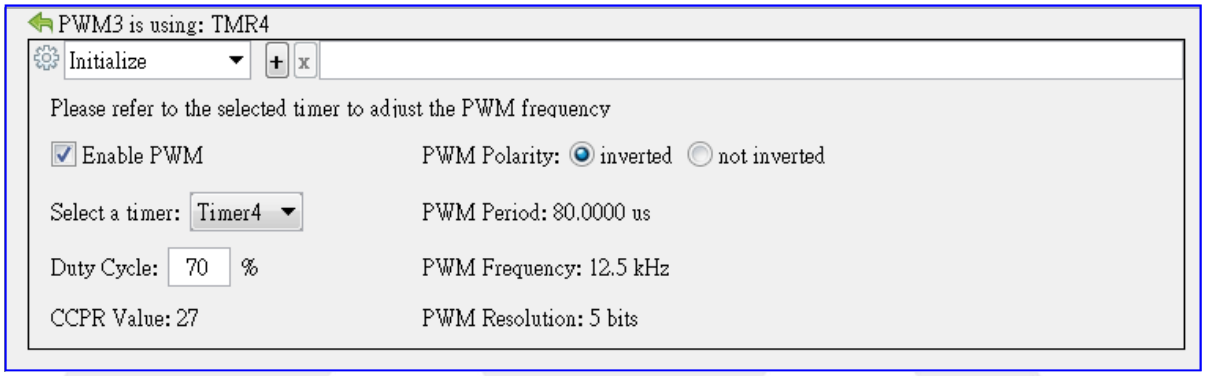


8. CIP 的 PWM 的模組設計為一專用的 PWM 控制，如下圖所示。週期由 Timer 2 /4/6 的 PRx (8-bit) 來對 Duty (10-bit) 做 1:4 的控制，Duty Cycle 的解析度可以有 10-bit 。PWM 輸出的 Duty Cycle 如右圖公式所示。因為要在 1.28mS 輸出 16 個脈衝，所以 PWM 的週期為 80uS，也就是 TMR4 的週期要設定在 80uS。

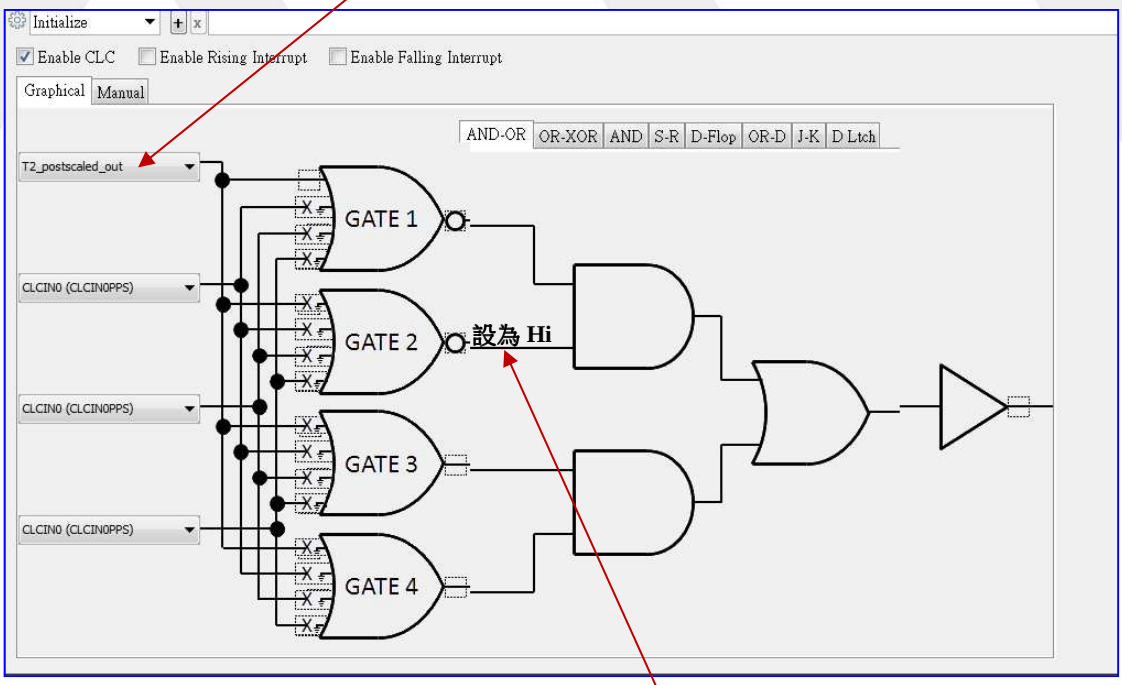
$$Duty\ Cycle\ Ratio = \frac{(PWMxDCH:PWMxDCL<7:6>)}{4(PR2+1)}$$



下圖即為 PWM3 的設定，週期是 TMR4 的 PR4 所設定的 80uS，PWM4 採反相輸出並設定負相 70% 的 Duty，經反相後為正 30% 的 Duty Cycle 輸出。

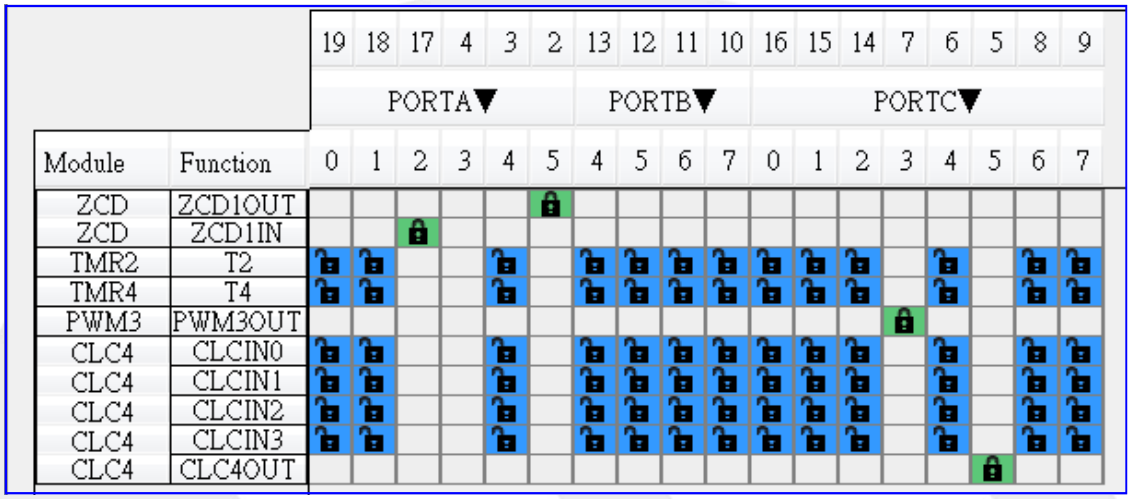


9. 因為 TMR2 的 Postscaled 輸出 (1.28mS) 並無輸出腳可供量測，雖然也用中斷模式觀測但需耗用一些軟體功能。因為 TMR2_Postscaled 訊號可以在內部連接給 CLC 的輸入端，在這裡我們就可以利用 CLC 將這 TMR2_Postscaled 的訊號送到 CLC 只定的輸出腳位上以供量測。底下的圖示就是設定 CLC4 的方式:



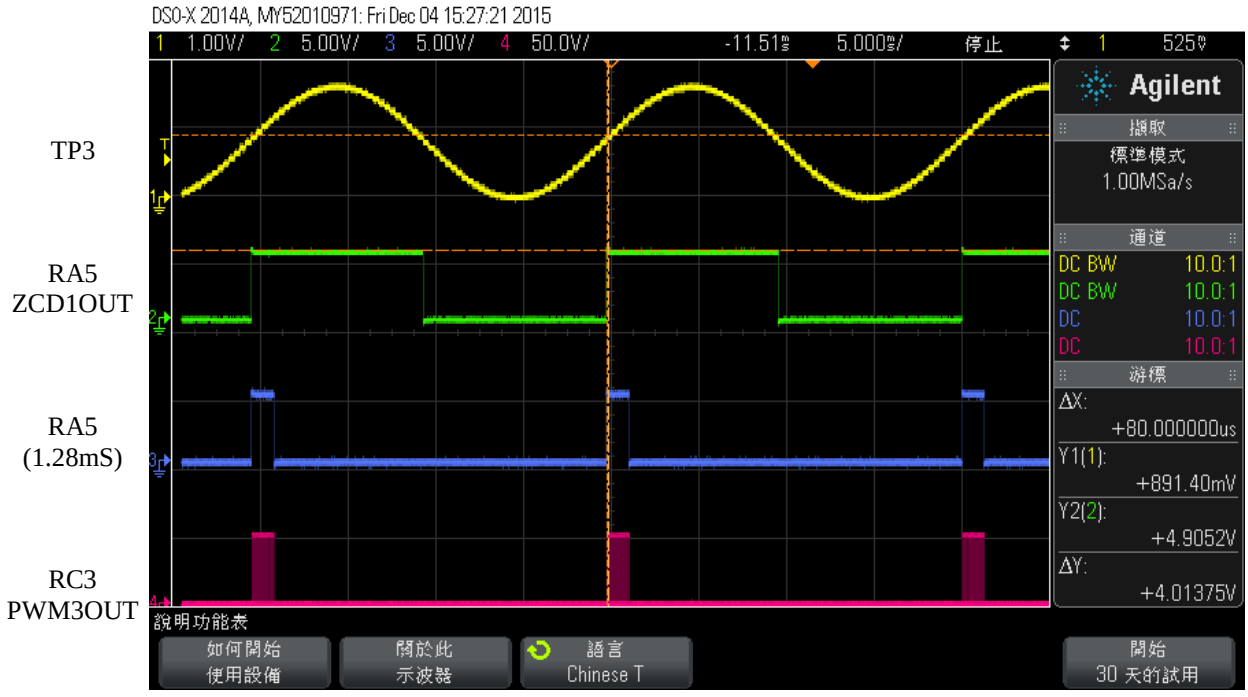
我們只用到一個輸入項 (T2_postscaled_out) 連接給 CLC4 的輸入端，注意，因為在 AND 運算時必須讓未使用的腳位恆為 1 ，所以在GATE2 加個反相輸出來啟動 AND Gate。

10. 最後來檢視一下輸出腳的設定。如前面所提的 MCC v2.25.2 的 Bug，我們需重設一下 ZCD1IN (RA2) 及 ZCD1OUT (RA5)。如上所述 TMR2_Postscaled 的輸出經 CLC4 反相後會送到 CLC4OUT (RC5) 做觀察用，以及最後的脈衝輸出在 PWM3OUT (RC3) 腳位。

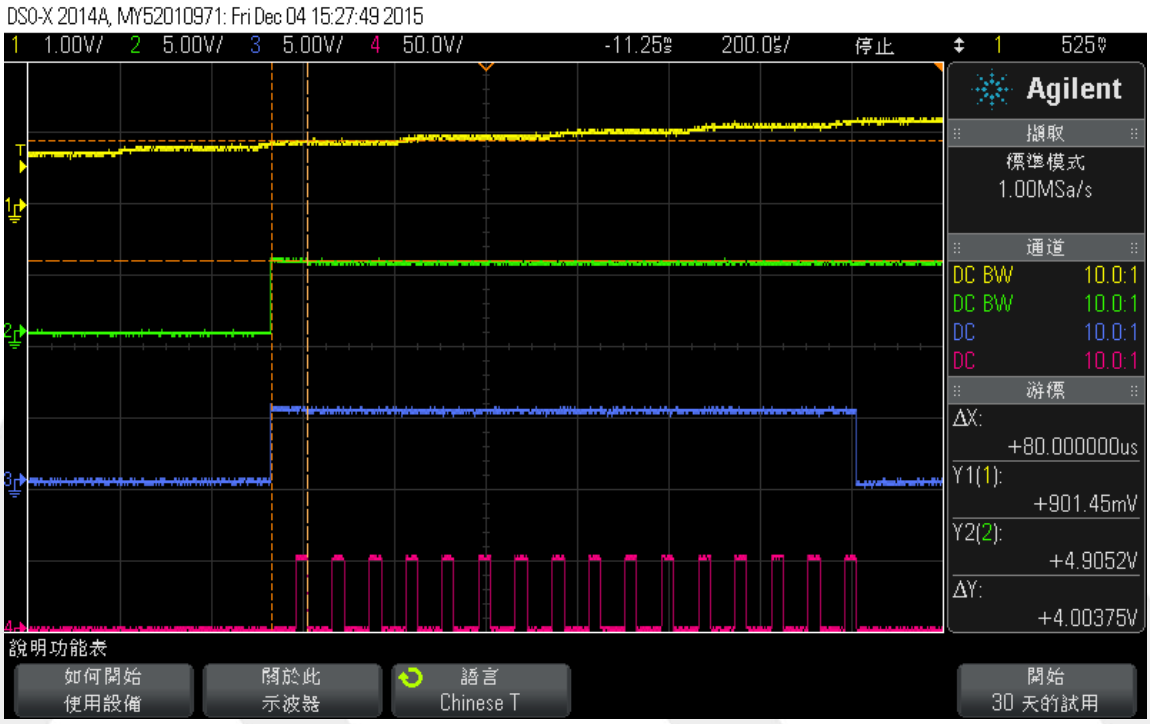


		19	18	17	4	3	2	13	12	11	10	16	15	14	7	6	5	8	9
		PORTA▼						PORTB▼				PORTC▼							
Module	Function	0	1	2	3	4	5	4	5	6	7	0	1	2	3	4	5	6	7
ZCD	ZCD1OUT						🔒												
ZCD	ZCD1IN			🔒															
TMR2	T2	🔓	🔓			🔓		🔓	🔓	🔓	🔓	🔓	🔓	🔓		🔓		🔓	🔓
TMR4	T4	🔓	🔓			🔓		🔓	🔓	🔓	🔓	🔓	🔓	🔓		🔓		🔓	🔓
PWM3	PWM3OUT														🔒				
CLC4	CLCIN0	🔓	🔓			🔓		🔓	🔓	🔓	🔓	🔓	🔓	🔓		🔓		🔓	🔓
CLC4	CLCIN1	🔓	🔓			🔓		🔓	🔓	🔓	🔓	🔓	🔓	🔓		🔓		🔓	🔓
CLC4	CLCIN2	🔓	🔓			🔓		🔓	🔓	🔓	🔓	🔓	🔓	🔓		🔓		🔓	🔓
CLC4	CLCIN3	🔓	🔓			🔓		🔓	🔓	🔓	🔓	🔓	🔓	🔓		🔓		🔓	🔓
CLC4	CLC4OUT																🔒		

11. 最後在 MPLAB X IDE 下，先產生 MCC 的週邊函數後再編譯及燒錄。下圖為實際測量本實驗的結果: 1. 為子板上 TP3 的正弦波輸入。 2. 為 ZCD 交越點的輸出。 3. 為 TMR2_postscaled 的 1.28mS 的時間框輸出。 4. 最後所得到的 16 個 PWM 脈衝輸出。

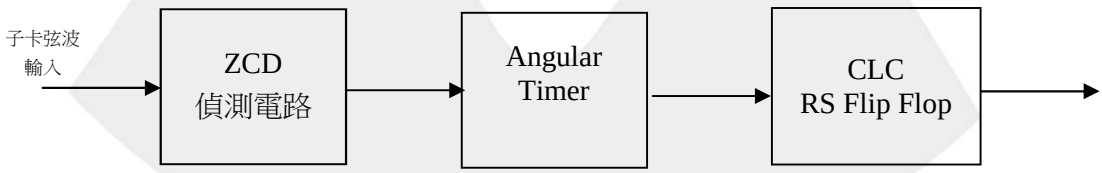


將上面的波型展開放到後可以看到 (如下圖所示) 在 1.28mS 的時間框裡所送出的 80uS 週期的 PWM 及 30% 的 High Duty。



Lab 7

1. 利用 ZCD 偵測上升弦波交越點
2. 將 ZCD 上升緣做為 Angular Timer 的 0^{O} 起點直到下一上升緣的到來
3. 設定兩比較器的比較時間 (C1 < C2)
4. 將兩比較器輸出送至 CLC 內的 RS 正反器做角度(相位)的輸出



以上的動作無需主程式的執行，
經設定後 CIP 會自行運作

Lab 7:
了解 **Angular Timer** 在比較器模式的應用

目標：

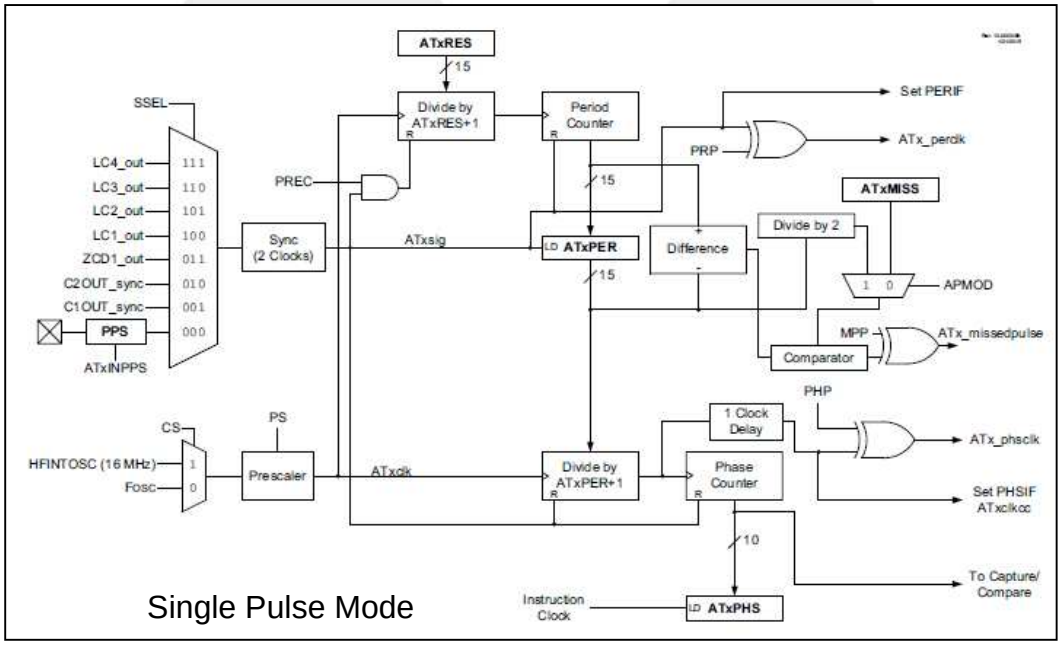
使用 MCC 來設定 Angular Timer 來做角度的偵測，實驗中將在 90 度時輸出 Hi 訊號及在 180 度時將輸出清除為 0 。

概述:

Angular Timer (角度計時器) 是一硬體電路用來測量輸入訊號的相位並利用比較器來輸出所設定腳度時的輸出。如此一來就可以在連續且重複的訊號中正確的找出特定的角度的位置。角度計時器可應用的範圍很廣，因位是硬體所完成的功能無需軟體的介入具有更加的彈性。應用上像: 鐘擺掃描的定位及角度偵測，伺服定位，倒單擺，機器人角度定位，自動駕駛…等。

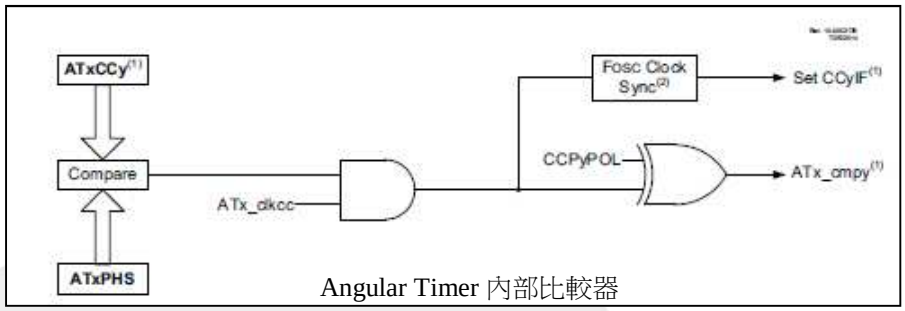
在本實驗 (Lab7) 是將 ZCD 的上升零電的偵測做為 Angular Timer 的量測訊號輸入，測量時所使用的計數時脈來自內部的 16MHz 震盪頻率。使用兩個相位(角度)比較器的輸出再去控制 CLC 的 RS 正反器的輸出來得到角度(相位)的偵測。

Angular Timer 有兩種工作模式 Single/Multi Pulse Mode。 首先看一下 Angular Timer 的在 Single Pulse Mode 的方塊圖，這方塊圖分成兩塊，第一部分是輸入訊號的計數及角度計數，第二部分是比較器。每個 Angular Timer 有三個比較器。



Single Pulse Mode

檢測角度 AT1CCY 的設定值與相位計數器 AT1PHS 相互比較後輸出到 AT1cmpy 或設定中斷輸出 CCyIF



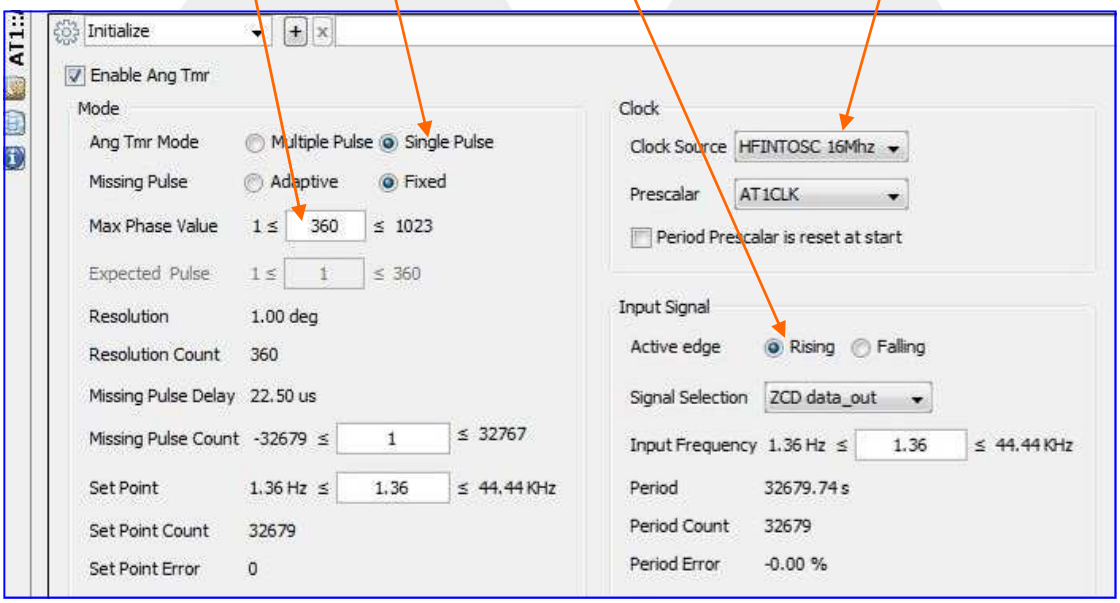
Angular Timer 內部比較器

Lab7 開始做實驗

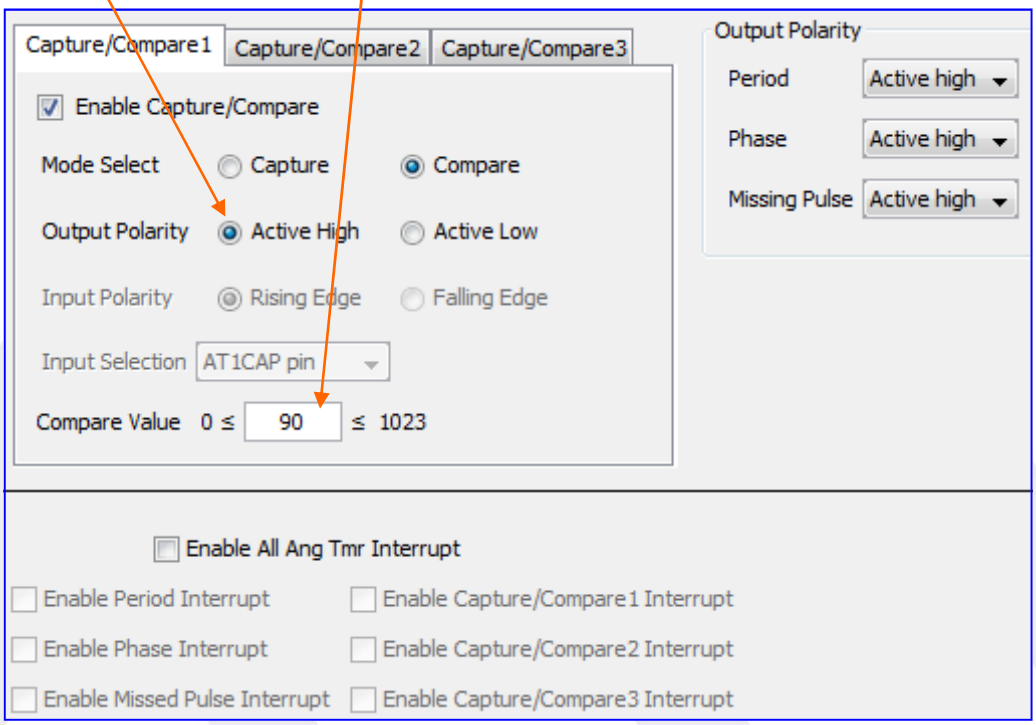
1. 開啟 Lab7_Ang_TMR_ZCD.X
2. 啟用 MCC
3. 設定 ZCD::ZCD 模組功能，如前面實驗的設定圖示：只要啟動 ZCD 並將其輸出不做反向。 注意新開啟的專案中，MCC 的 ZCD 設定需重新指定一下 RA5 為 ZCD1PUT 觀察腳位，也時需在 main.c 的程式裡加入 ZCD1OUT_TRIS = 0;

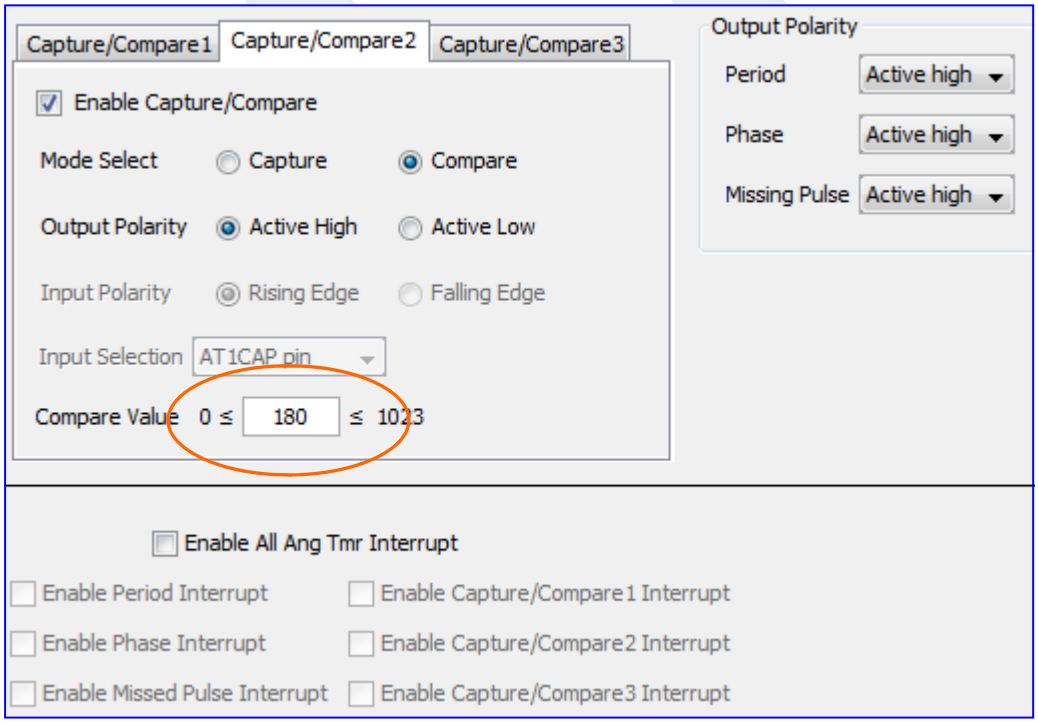
```
52  void main(void) {
53      // initialize the device
54      SYSTEM_Initialize();
55      ZCD1OUT_TRIS=0;
56
```

4. Angular Timer (AT) 總共有兩個圖示需設定，第一部分是 AT 的主體部分: 在這裡需將其設定為固定的單脈衝輸入模式 (Single Pulse)。接下來設定最大的相位角的值，在這裡我們採用 360 並得到 1 度的解析度，這 360 度也就是 ZCD 上緣電壓偵測輸入到下一次 ZCD 輸入的範圍。在計數相位的時脈選用內部高速 16MHz 的震盪，預除器採 1:1 的輸入比率設定。輸入訊號設定使用 ZCD 的上緣觸發來啟動 AT。

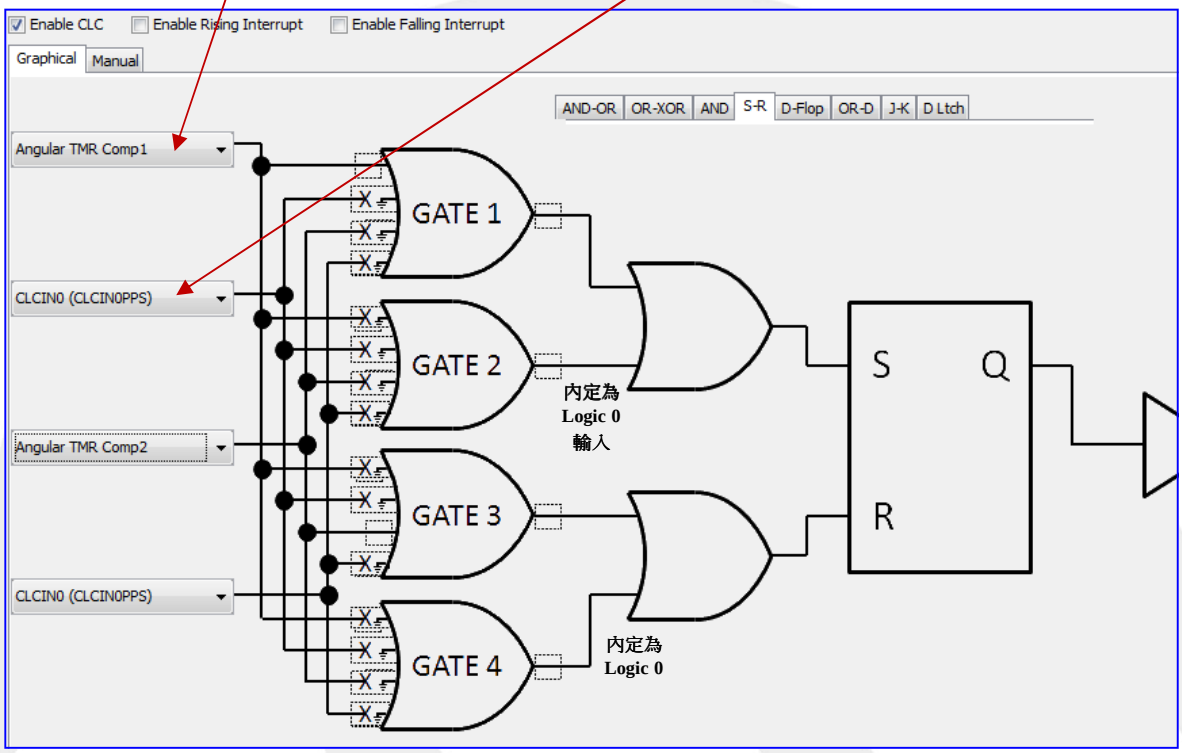


5. 接下來要設定的是 AT 的比較器。在這裡我們使用比較模式 (Compare)，比較相符時的輸出動做為 Hi。因為本實驗需求是兩個角度的偵測所一要啟用兩個比較器。第一個比較器要設定角度為 90 度時輸出；第二個比較器要設定角度為 180 度時輸出。

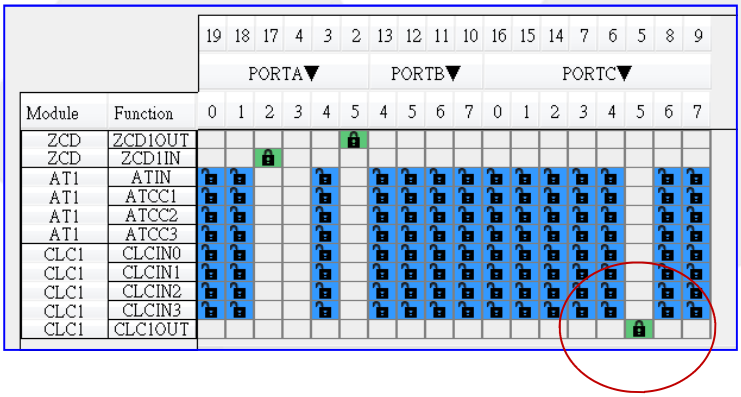




6. 接下來就要將這兩個比較器的輸出送到 CLC 模組做輸出。我們的目的是利用這兩個訊到接到 RS 正反器做 Set 及 Clear 的動作，所以需要設定 CLC 為 SR 栓鎖。如下圖所示: 將 AT Comp1 輸入連到正反器的 Set 端；ATComp2 連到 Reset 端。CLC 的輸出端設定 (CLC1OUT) RC5。



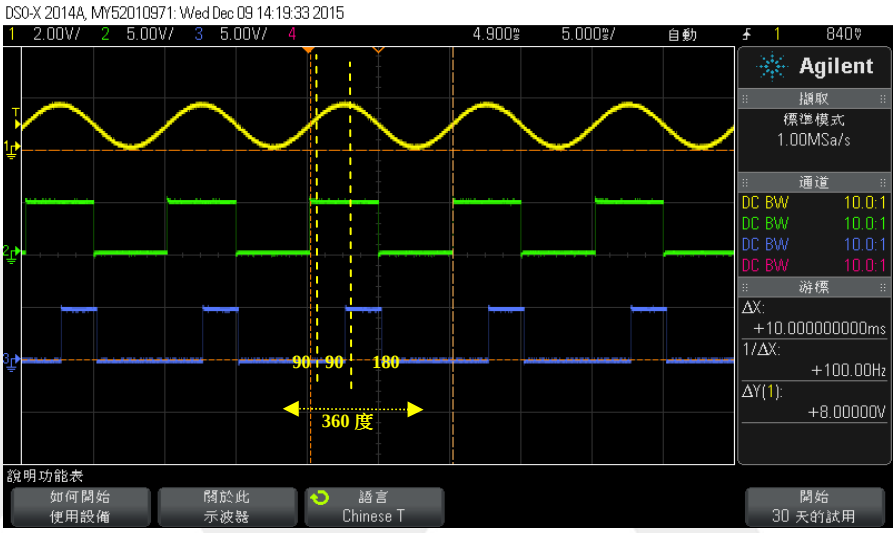
7. 最後到 Pin Manager 的腳位設定裡，透過 PPS 來設定 CLC1OUT 到 RC5 的腳位。



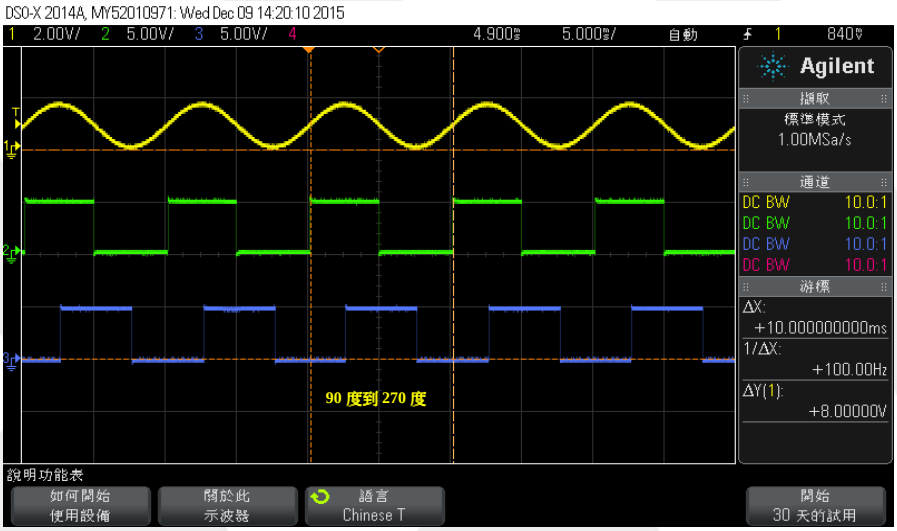
8. 啟用 MCC Code Generator 產生 CIP 的函數後，在 main.c 加入 ZCD1OUT_TRIS=0; 的程式後，值接編譯及燒錄，用示波器量測一下個點的波形。

右圖為設定 90 度及 180 度的輸出。

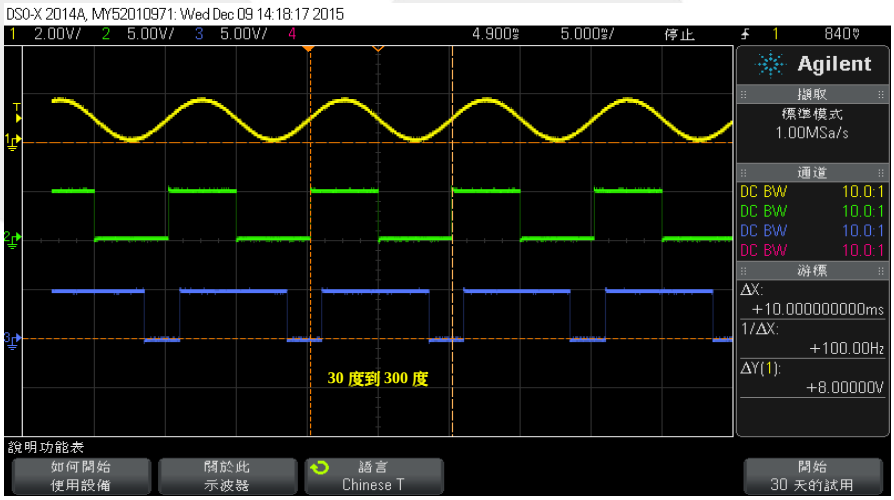
1. 為弦波輸入到 ZCD (TP3)
2. ZCD 的準位檢測輸出 (RA5)
3. CLC1OUT (RC5)
   注意看一下與正弦波的角度變化點，是否落在弦波的 90 度及 180 度。



右圖為設定 90 度及 270 度的輸出。

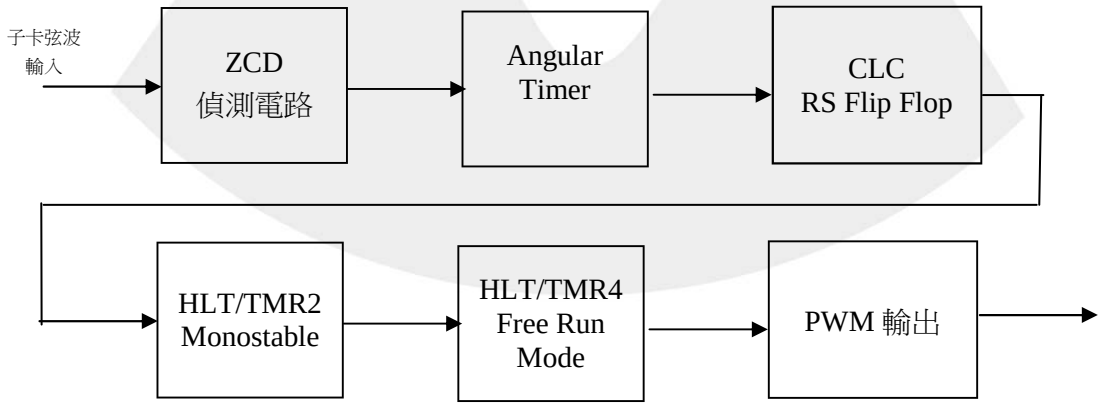


右圖為設定 30 度及 300 度的輸出。



Lab 8

1. 修改前面的 Lab7 加入對 TRIAC 觸發的應用
2. 將 ZCD 的上、下緣偵測做為 HLT/TMR2 的時間框
3. 在時間框內做 5 個 PWM 脈衝的輸出



Lab 8:
利用 ZCD 輸入做 TRIAC 的正負觸發

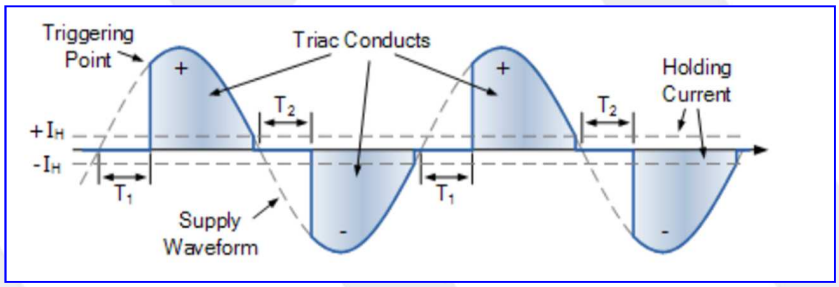
目標：

綜合練習使用 CIP 模組做 TRIAC 正負週的角度觸發應用

概述：

這實驗整合了前面幾個實驗，並連貫起來做成 TRIAC 角度控制的觸發(點火)的應用。

我們都知道 TRIAC 在交流時的觸發需在正週期觸發一次，也要再負週期時觸發一次，這樣才可以使 TRIAC 正負週期都可以導通達到功率輸出的控制 (如 AC 燈泡明亮控制)。



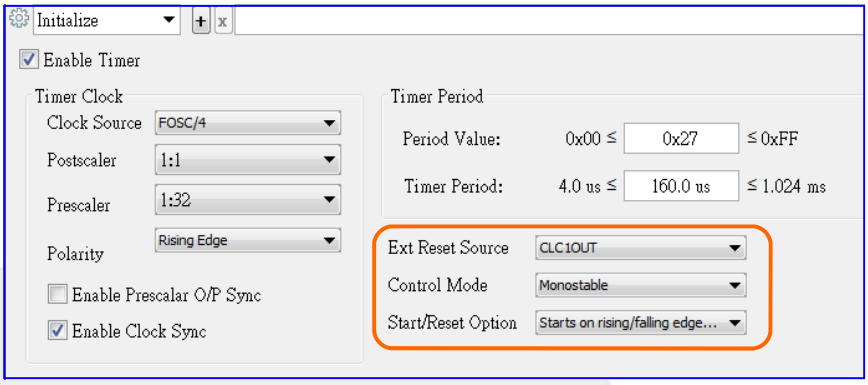
本練習是控制 TRIAC 在正半週的 30 度時發出 5 個 PWM 來觸發，在負半週 的 30 度 (即 210 度角) 時再發出 5 個 PWM 在觸發一次。

所以從 ZCD —> AT —> CLC —> HLT/TMR2 —> HLT/TMR4 —> PWM3 等一連串需做設定的 CIP 週邊。

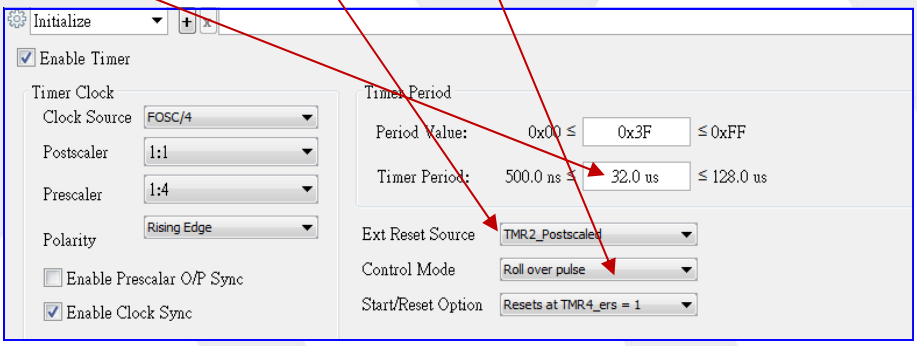
Lab8 開始做實驗

1. 開啟 Lab8_AT_TRIAC_ZCD.X
2. 啟用 MCC
3. 如 Lab7 一樣開啟 ZCD 的設定。再開啟 AT 的設定，設最大相位為 360 (Max Phase Value) ，1 度的解析度。比較器 1 設定為 30 度，比較器 2 設為 210 度。
4. 接下來要設定的是 CLC 模組，如同 Lab7 一樣，連接比較器 1 & 2 到 RS 正反器，CLC1OUT 的輸出設定在 RC5 以便示波器的量測。
5. HLT/TMR2 的設定: TMR2 的輸入觸發源為 ZCD1OUT 因為要有正、負緣的觸發所以要設定啟動模式為 “Start on rising/Falling edge on TMR2_ers” 。TMR2 的時間

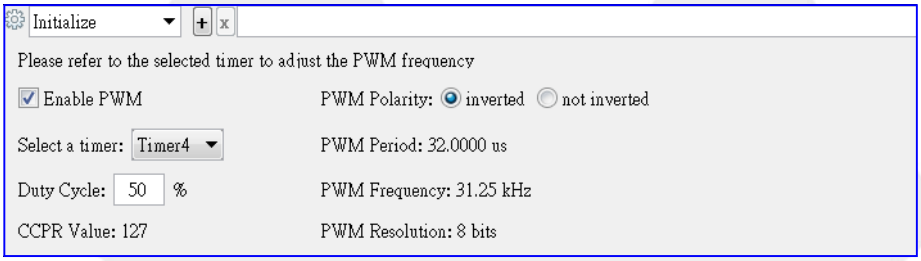
框為 160uS，也就是說正弦波 0 ~ 180 度時會有一個延遲 30 度的 160uS 的時間框，在 180 ~ 360 度時也會有一個 210 度 160uS 的時間框。



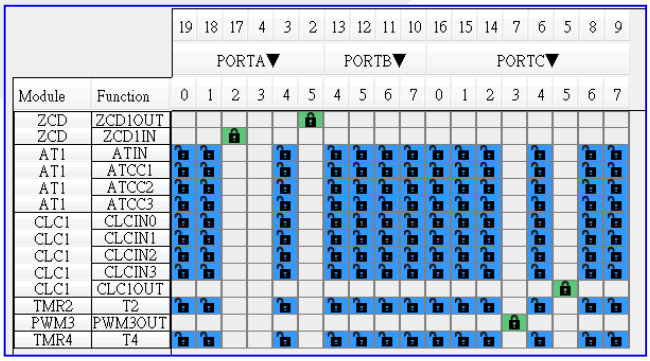
7. 上述的 TMR2 將做為一個弦波兩個時間框的輸出。將輸出做為啟動/關閉 HLT/TMR4 的控制訊號，因為在一個 160uS 的時間框內需要產生 5 個 PWM 輸出，所以 TMR4 所設定的週期為 32uS 並使用 Roll Over Pulse 模式運轉。



8. 設定好了 PWM 的 32uS 週期，接下來要設定 PWM3 模組。Duty Cycle 設為 50% 輸出，並將 PWM 輸出反相輸出。

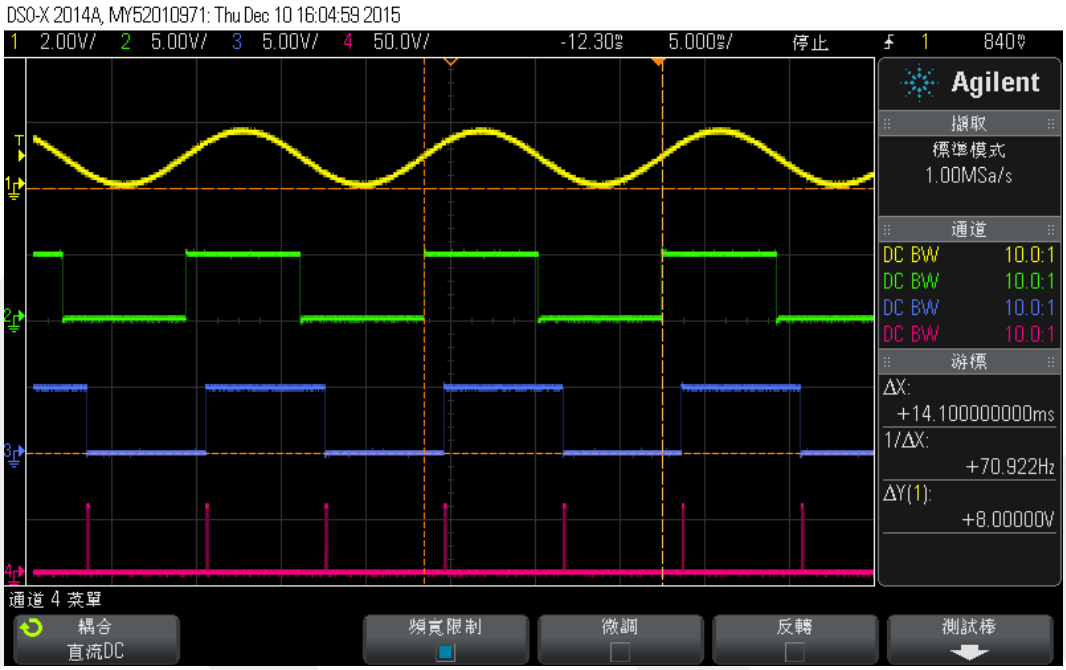


9. 最後確認一下腳位的設定，其中 CLC1OUT 腳位是用來除錯測試用的，可以清楚的觀測到延遲 30 度及 210度時的 RS 的輸出訊號。PWM3OUT 是送給 TRIAC 的觸發訊號。

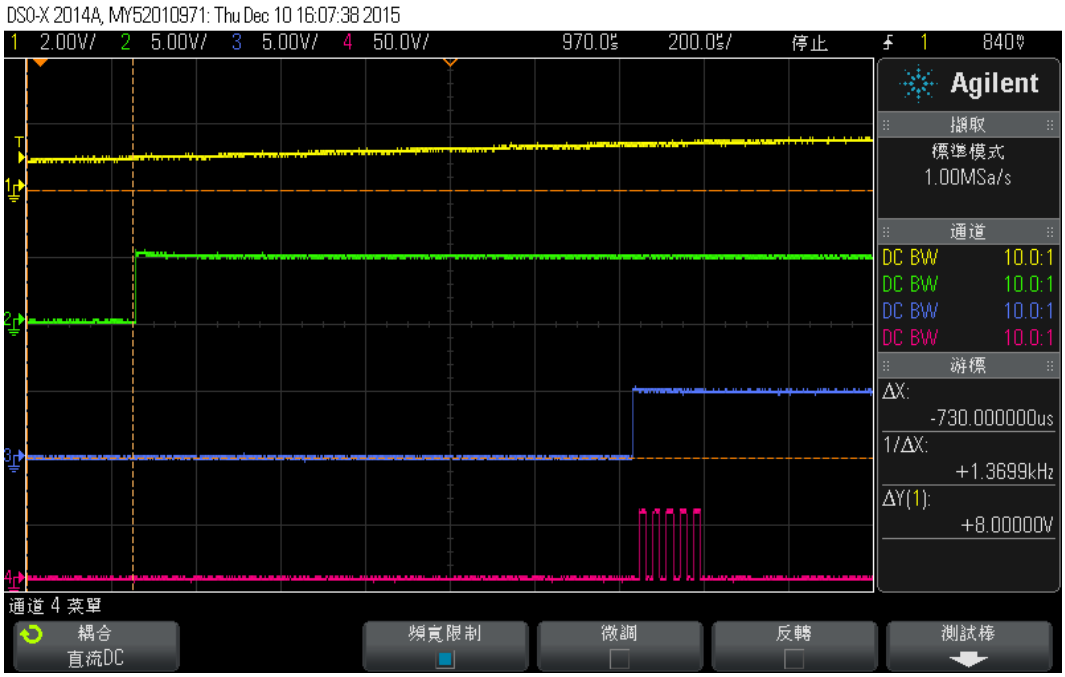


實驗量測的輸出波形：

在下圖的示波器可看到 4 個波形，1 為輸入的正弦波。　2 為 ZCD 的輸出。
3 為經過角度比較後的 CLC1OUT(RC5) 輸出。　4 為 PWM 觸發輸出

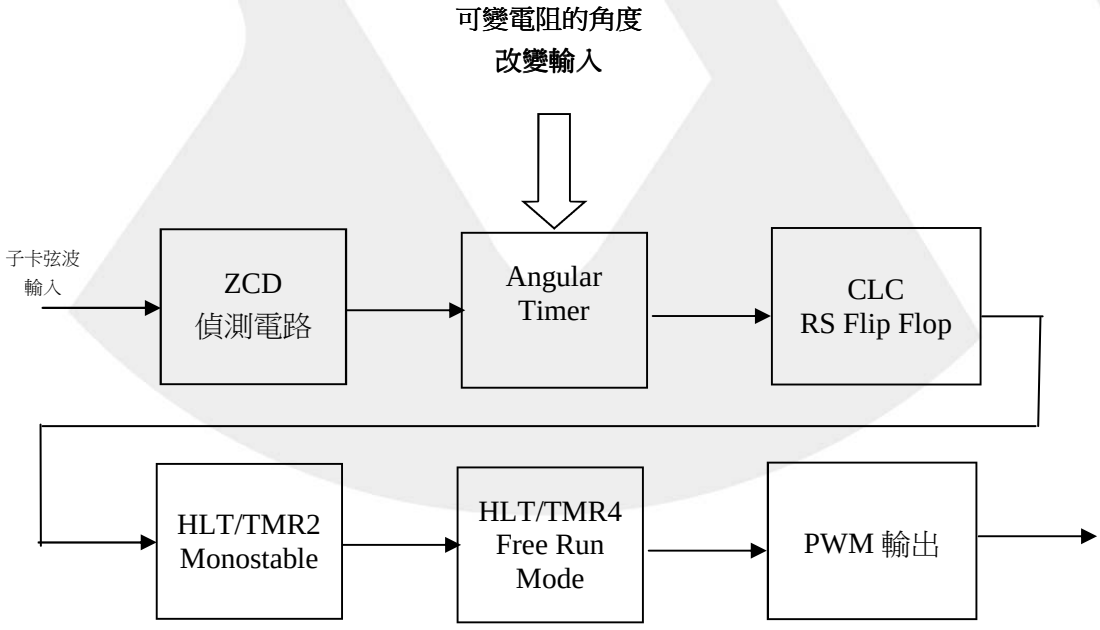


下圖為放大 PWM 的脈衝圖，圖中可明顯看到 5 個 Duty 50% 的脈波輸出。



Lab 8-1

延續 Lab8 的設定，加入 ADC 可變電阻的輸入，藉此 AD 的轉換值來調整 TRIAC 觸發角度從 10 度 ~ 170 度及負半週的 190 度 ~ 350 度。



Lab 8-1:
利用可變電阻的 AD 轉換值做為 TRIAC 的正負觸發角度的控制

目標：

這是結合 Lab 8 的綜合練習，由原先固定角度的 TRIAC 的觸發改成利用可變電阻來改變觸發的角度。在正半週時可變電阻可控制 10 度 ~ 170 度的觸發及負半週 190 度 ~ 350 度的觸發。

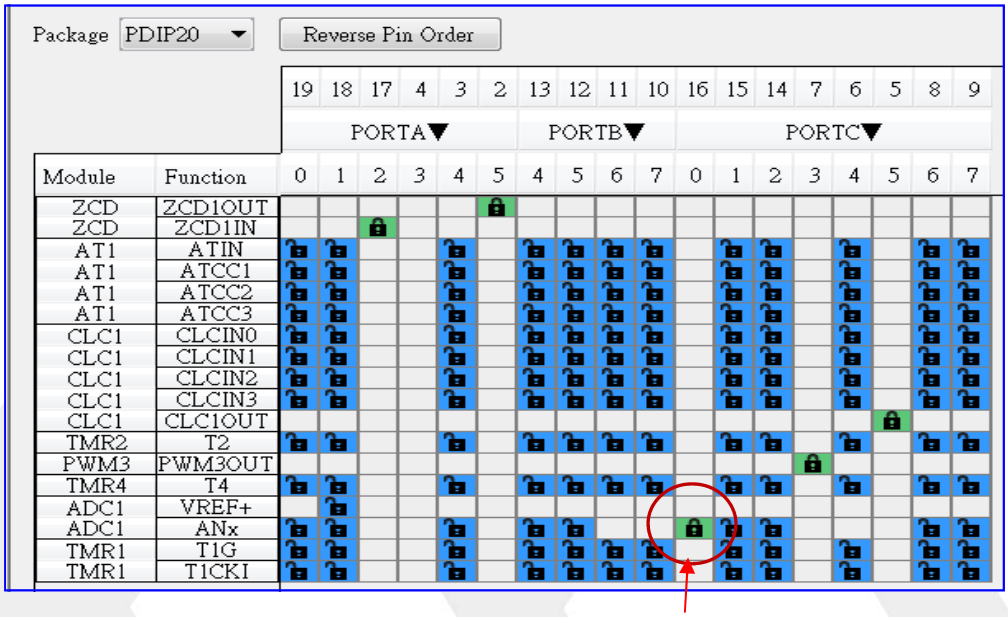
概述: 這是一個應用的延伸，當 Lab 8 的 CIP 的設定後可以得到 PWM 觸發 TRIAC 的脈衝訊號，但這也只是個固定的角度(相位)的觸發。這裡就想用可變電阻當做改變角度的輸入來源，想像一下如何來改變相位的觸發。
在 Lab 8 時，我們有去設定 Angular Timer 的比較器做為一個正弦波的相位比較點。如果我們將 ADC 的轉換值經過角度限制 (10 ~ 170 度 & 190 ~ 350 度) 及相位計算後是否就可以輕鬆的控制點火的角度呢!
到底要如何設定 AT 的比較器的值呢? 查一下 Data Sheet 裡可以知道:

AT1CC1H : AT1CC1L (16-bit 設計的第一個比較器)
AT1CC2H : AT1CC2L (16-bit 設計的第二個比較器)

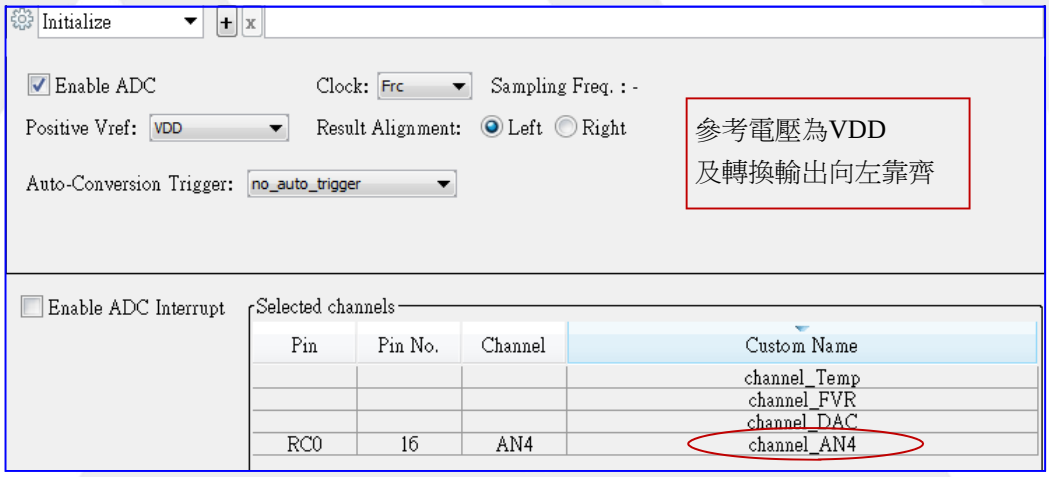
所以將 ADC 轉換值經修正後填入第一個比較器，將 ADC 值加上 180 度後再修正後填入第二個比較器。這樣就可以同步調整正、負半週的角度觸發。

Lab 8-1 開始做實驗

1. 開啟 Lab8-1_AT_ADC_TRIAC.X
2. 啟用 MCC
3. 同 Lab 8 的設定，請再加入 ADC 及 Timer1 的設定
4. 開啟 ADC1::ADC 的設定，由 Curiosity 板電路得知，可變電阻是接到 AN4/RC0 所以需先在 Pin Manager 設定 RC0 為 Analog Input 功能 (ANx)
5. 在 ADC Select Channels 的設定項中使用內定的 “channel_AN4” 的名稱，該名稱會在被定義在 “pin_mamager.h” 裡以供程式撰寫者使用。



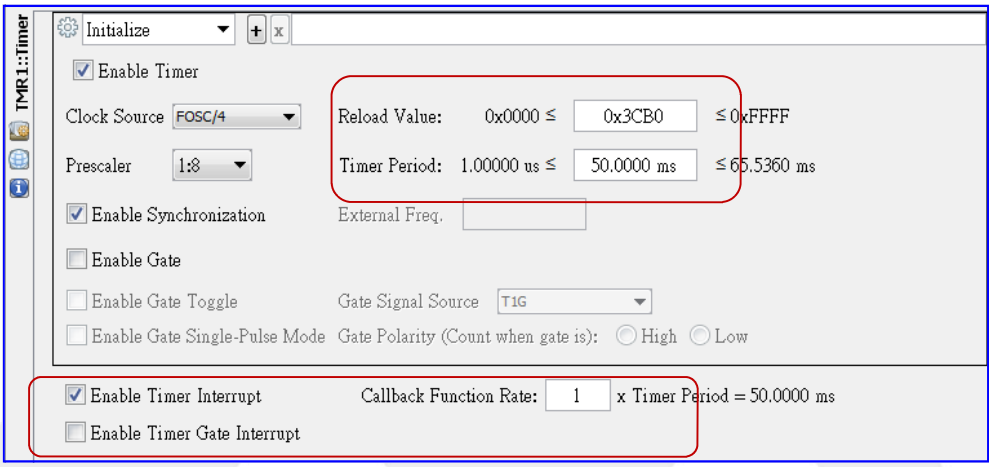
設定 RC0 為類比輸入功能腳



使用 channel_AN4 做為 AN4 的名稱定義

6. 最後加入 Timer1 的設定。目前 Lab 8 以經使用了 TMR2 及 TMR4 兩個 Timer 了。在此啟用 Timer1 做為 50mS 的計時器做為 ADC 做角度的更新。也就是說：點火的角度更新是在 Timer1 50mS 的中斷裡做轉換及更新的。主程式仍只有 while(1); 的永久迴圈。

底下為 Timer1 的設定圖示，設定使用中斷計時模式並加入 Callback 函數。



7. 程式說明：因為有啟用中斷所以需在 main.c 主程式開啟中斷功能，如下圖所示:

```
// Enable the Global Interrupts
INTERRUPT_GlobalInterruptEnable();

// Enable the Peripheral Interrupts
INTERRUPT_PeripheralInterruptEnable();
```

8. main.c 的開始要做了 ADC 的開啟及固定要轉換 Channel 的輸入

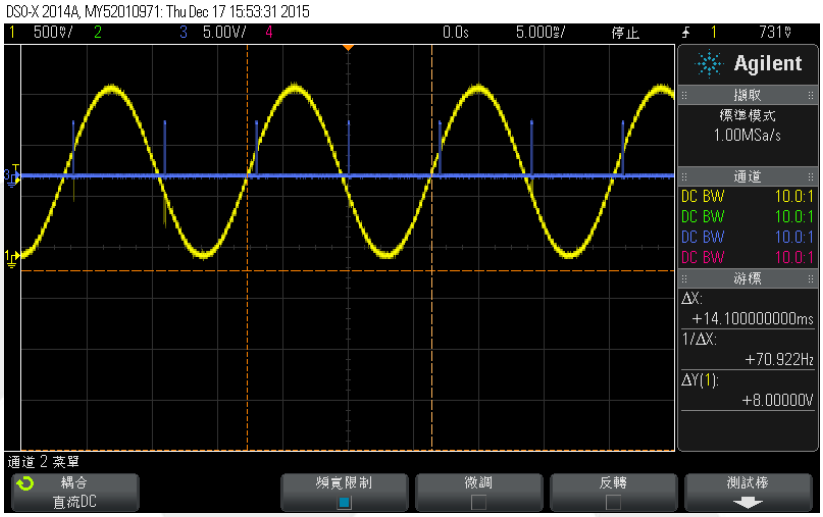
```
vid main(void) {
// initialize the device
SYSTEM_Initialize();          // 執行 MCC 所產生的週邊函數初始設定
ZCD1OUT_TRIS = 0;             // 設定 ZCD1 為輸出腳位以方便測量
ADCON0bits.ADON = 1;          // 啟動 ADC
ADCON0bits.CHS = channel_AN4; // 設定輸入的 Channel 為 AN4/RC5 的輸入
```

9. Timer1 的 Calledback (tmr1.c) 的說明:

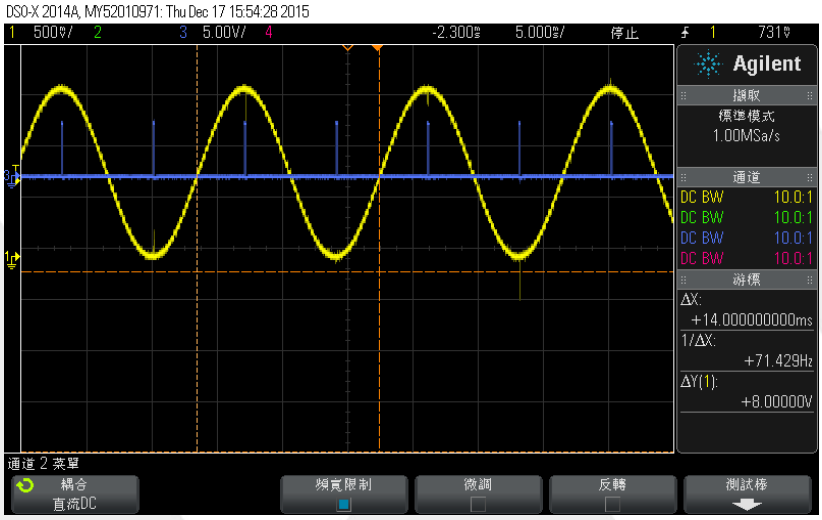
```
void TMR1_CallBack(void)
 {
     ADCON0bits.GO = 1;                    // AD 開始轉換
     While (ADCON0bits.GO);                // 轉換完成?
     If (ADRESH < 2) ADRESH = 10;          // 正半週角度控制在 10 ~ 170 度之間
     If (ADRESH > 170) ADRESH = 170;
     AT1CC1L = ADRESH;                     // 比較器一的控制
     AT1CC2L = (unsigned char)(ADRESH + 180); // 負半週角度控制在 190 ~ 350 度
     AT1CC2H = (unsigned char)((ADRESH + 180) >> 8);
```

實際實驗所量測的波形:

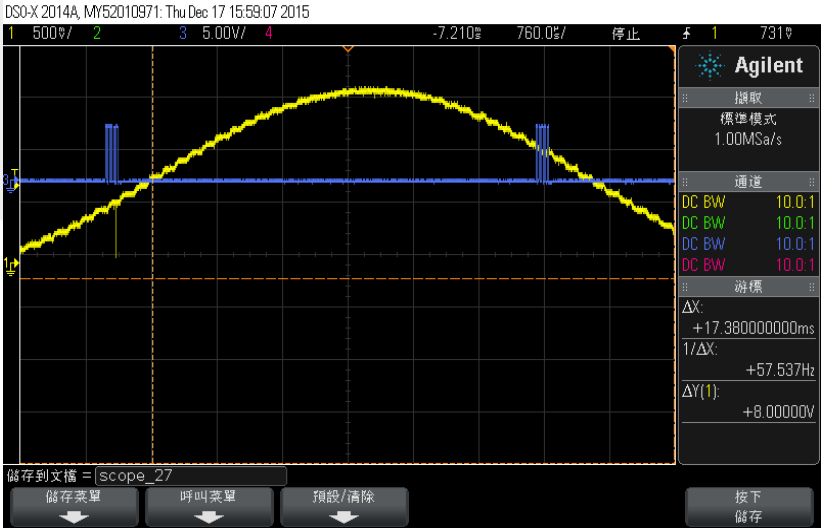
相位 10 度及 190 度的觸發



相位 90 度及 270 度的觸發



五個 PWM 觸發脈衝的展開



課程結束!

如有問題請撥打技術服務專線:
0800-717718

Microchip Technology 台灣分公司
台北市民權東路三段四號12F
總機電話: 02-2508-8600