



MICROCHIP

Regional Training Centers

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

MCC v3.0.3

指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

前言與心得

本實作手冊是改版是源自於 MCC v2.25.2 改寫成 MCC v3.0.3 的版本。雖然都稱為 MCC 但從 MCC 開始問世，我使用過 v2.0、v2.10 及 v2.25.2 等版本來撰寫 PIC18F45K22 的 MCC 教材。這期間只有一個感覺是：MCC 每個版本都不盡相容，使用新的 MCC 版本載入不同版本 MCC 所設定的專案時，幾乎每個週邊 (CIP) 的設定都要重新檢視並設定。

尤其是這本實作手冊雖是說由 MCC v2.25.2 轉成 MCC v3.0.3 的設定，但這轉換在 MCC v3.0 根本看不到原先 MCC v2.25.2 所設定的週邊，甚至於連 MCC 的設定檔名都不一樣。

在 MCC v2.xx 版本設定檔為 "device_config.mcc"，在 MCC v3.0 的設定檔為 "MyConfig.MC3"。所以不要期待使用 MCC v3.0 可以使用 MCC 舊版的設定。

不管是哪一個版本的 MCC 實際上使用起來都有 Bug 的存在。在使用過程中我也將所遇到的 Bug 儘量寫出來，至於那些沒遇到的 Bug 我就知道了。我的使用心得是 MCC v2.21 以前的版本應該把它都丟進垃圾桶，使用 MCC v2.25.2 就非常好用，雖然仍有一些小蟲，但瑕不掩瑜這是一個不錯圖型化的設定軟體，大大輔助設計者縮短週邊的設定，尤其是面對 CIP 內錯綜複雜的內部連線，沒有 MCC 的幫忙真的要設定內部連線，單單看懂暫存器的設定就足以花上半天的時間來搞定，經驗上使用 MCC 只要一分鐘就設定完畢，有錯還可以立即更改重新編譯。

使用上 MCC v3.0 與 v2.25 最大的差異是：

- a) MCC v3.0 加入了一些新的應用程式庫(請暫時不要使用)，要等下一版 v3.15 才会有完整的支援。(希望下一版本可再加入 mTouch & USB 的 MCC 設定來擴充其應用)。
- b) MCC v3.0 多了暫存器的設定，可讓設計者做更底層的設定。
- c) MCC v3.0 加了一個警告訊息視窗可提示目前的設定的提示、警示及錯誤等訊息。
- d) MCC v3.0 在腳位設定視窗會一直出現周邊未使用的預定腳位，這常常與透過 PPS 的腳位設定出現腳位相互衝突的訊息及圖示，雖然不會影響腳位設定功能，但顯示有衝突總不是滋味。

MCC 出現主要是要取代先前傳統的周邊函數庫 (Peripheral Libraries) 的使用。新的周邊越來越多且內部連線日益複雜，透過圖型化的設定才能協助設計者簡化其周邊的設定並大大降低錯誤。

MCC 目前只支援 PIC16F1xxx、PIC18FxxKx 及 PIC24F 的元件。至於 dsPIC30F & dsPIC33F 系列，目前是沒有支援的。

PIC32 系列現在是由 MHC (MPLAB Harmony Configurator) 來支援。

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

Lab 0 - 低電壓燒錄模式設定	9
Lab 1 - 使用 Timer1 設定 LED 閃爍	14
Lab 2 - RGB 三色 LED 使用 CLC 的串列控制	18
Lab 3 - SMT Duty Cycle 或週期的量測	31
Lab 4 - ZCD 電壓零點偵測	38
Lab 5 - TMR2/HLT 單次觸發 PWM 的應用	42
Lab 6 - ZCD 及 HLT/TMR2 觸發 TRIAC	47
Lab 7 - Angular Timer Compare Mode	54
Lab 8 - ZCD 相位移動觸發 TRIAC	65

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 v3.20 (含) 或更新版本
- ◆ XC8 編譯器 v1.35 (含) 或更新版本 (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 v3.0.3 (請用此版本)
- ◆ 酷奇板(Curiosity) + PIC16F1619
- ◆ APP039 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 v3.0.3 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 v3.0.3 版本。

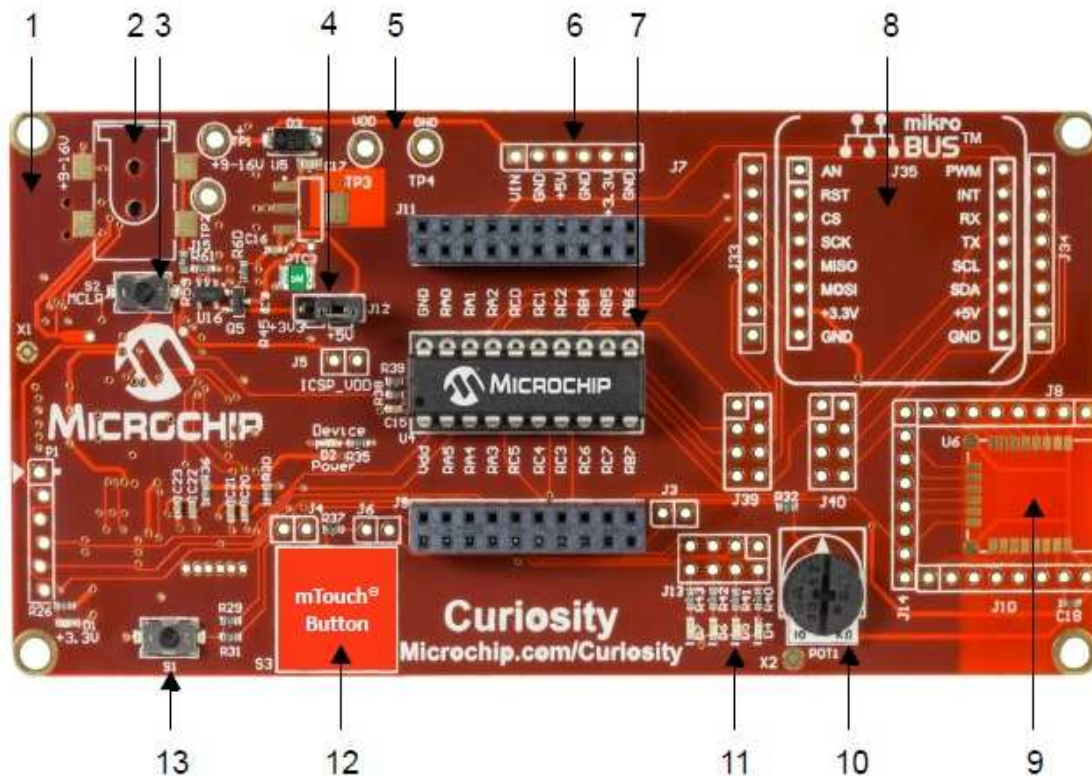
基本說明：

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

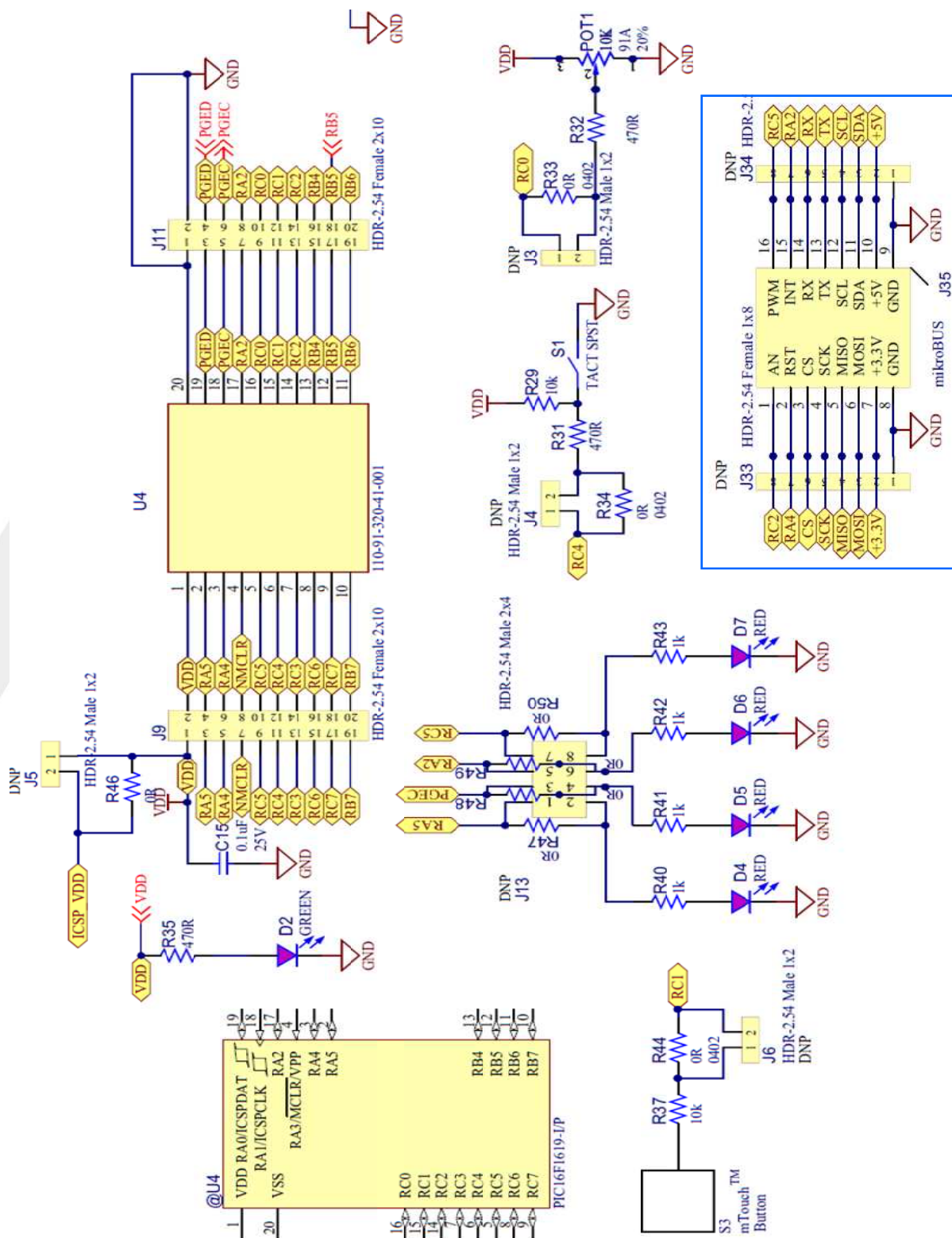
酷奇實驗板主要零件說明



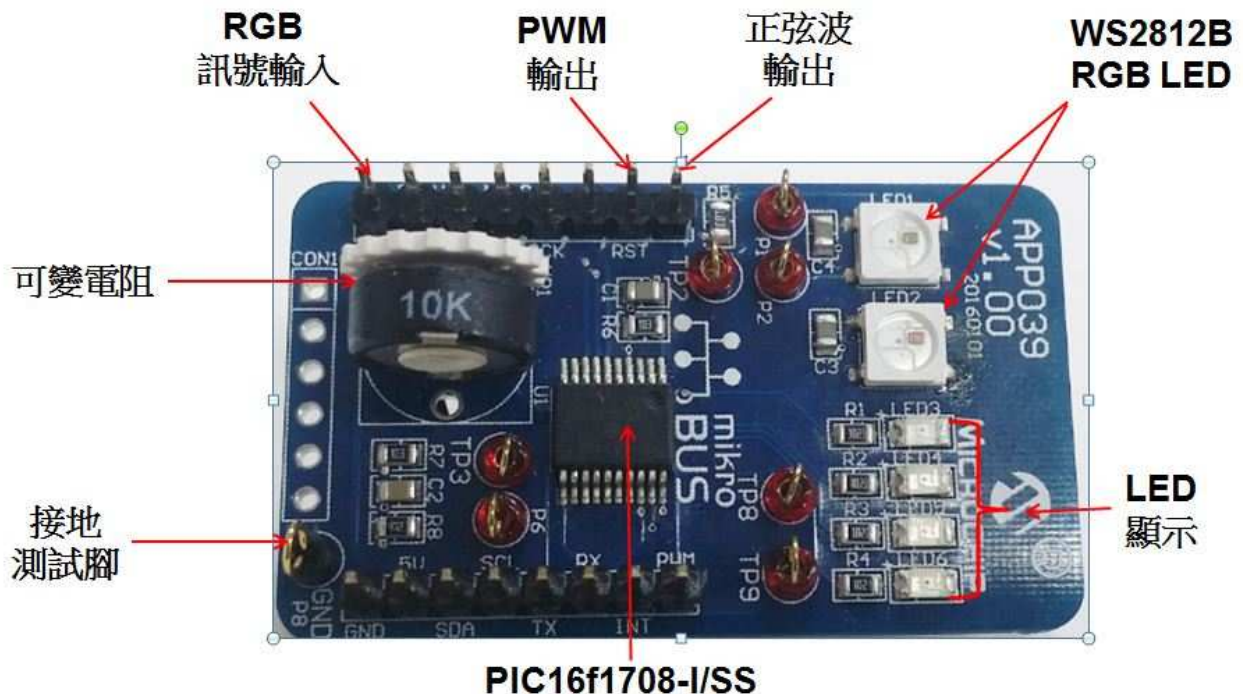
1. USB mini-B 連接器，供電及除錯（在背面）
2. 9V 電源接頭（未裝預留元件）
3. 主重置按鍵 (Reset)
4. 3.3V/5V 工作電壓選擇 (J12) (本實作請選擇 5V 供電)
5. 外接電源輸入接點（未焊接）
6. 外接擴充板排針接頭（未焊接）
7. PIC 插座（可適用於 8/14/20 接腳的 PIC）
8. mikroBUS 外接插卡連接器
9. RN4020 藍牙模組（未焊接）
10. 電位器 (RC0/AN4)
11. 4 顆 LED 顯示燈 (RA5, RA1/PGEC, RA2, RC5)
12. mTouch 觸控按鍵 (RC1)
13. 按鍵開關 (RC4)

指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

酷奇板的電路參考圖。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。

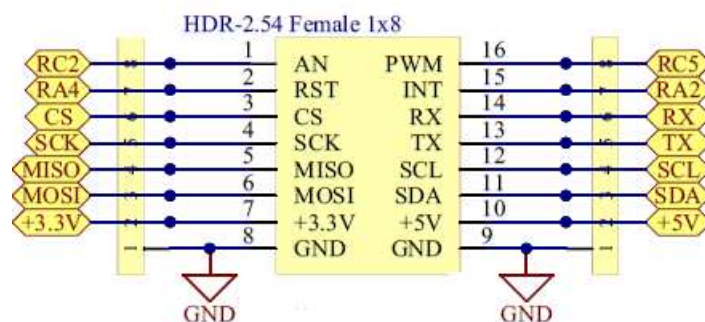


APP039 實驗子板圖示



上圖為本實驗所使用的 mikro BUS 接腳定義的子板 (APP023)。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。板子上也有兩顆 WS2812B RGB 三色 LED 做 CLC 的練習。

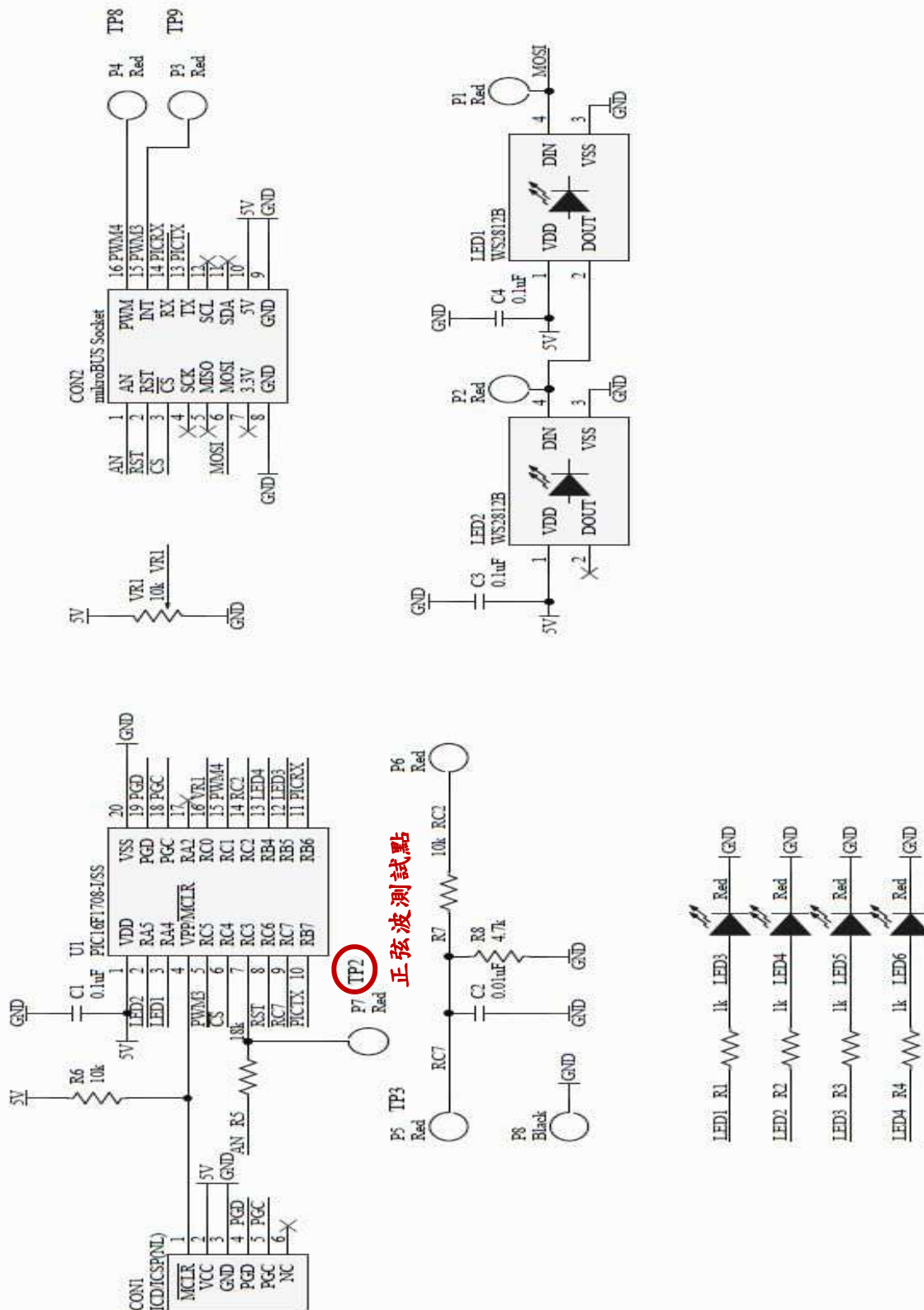
底下為 mikro BUS 的接腳，及接到 PIC16F1619 的對應腳位的功能。



子卡背板的排針腳位及接腳名稱

指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

子板 (APP039) 的線路圖:



Lab 0

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

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

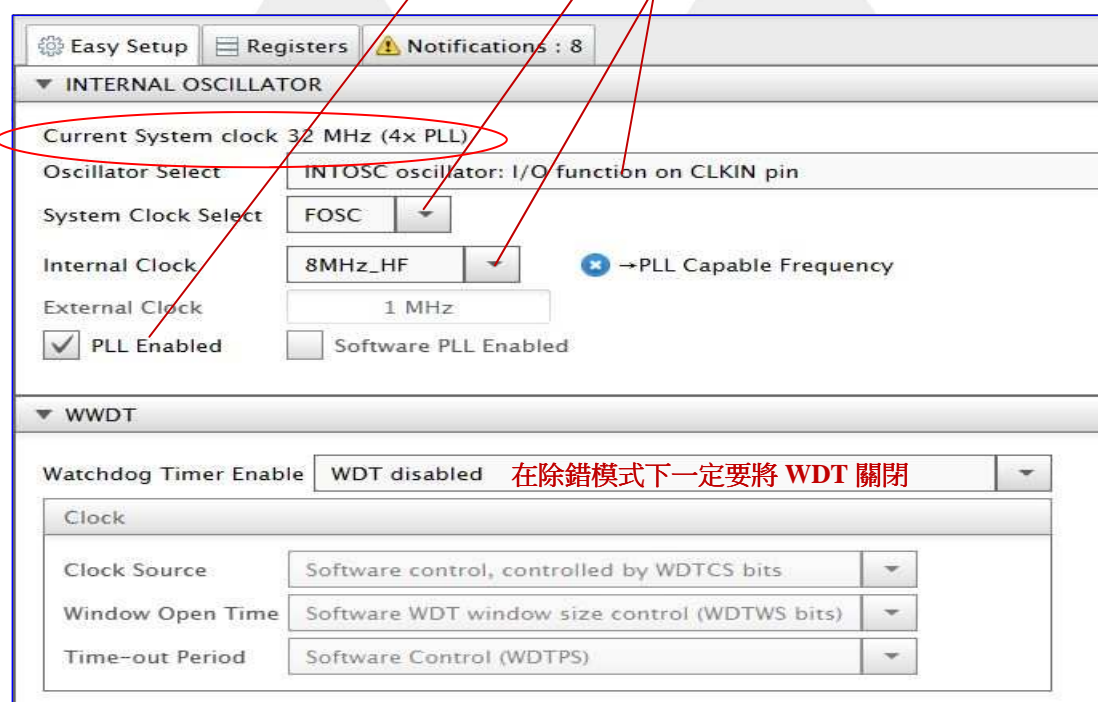
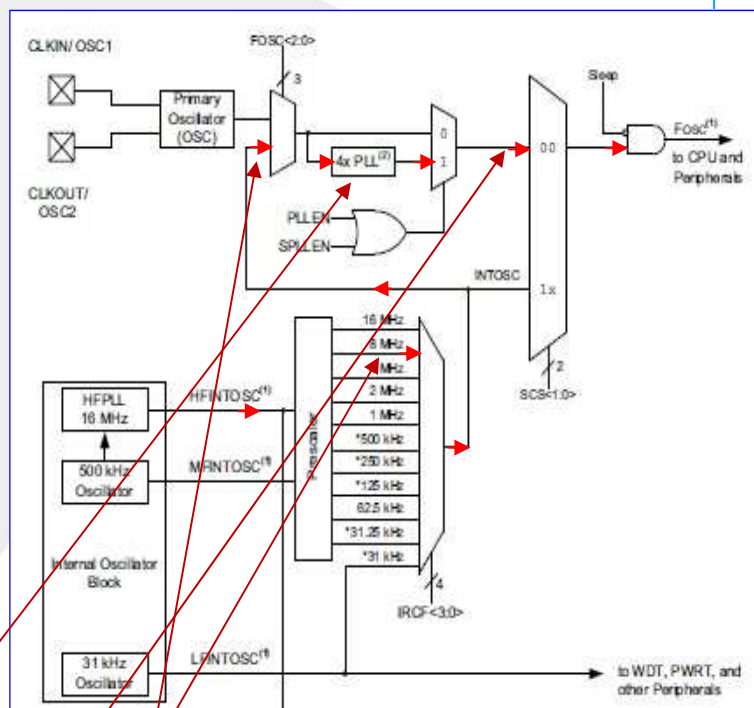
Lab 0: MCC v3.0 裡 Configuration Words 的設定

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)，本練習的目的在了解 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。
 - ◆ 開啟專案 Lab0 Low Voltage Programming.x
 - ◆ 在 Tools —> Embedded —> MPLAB Code Configurator 啟用 MCC
 - ◆ 點選 MCC 專案資源的 System Module” 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



指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

由上圖的震盪器的設定可以得知，我們使用內建的 8MHz Internal RC 震盪器為主，其輸出經 x4 PLL 倍頻後以 32MHz 的頻率當作系統的 Fosc。

設完主震盪器後，記得還有其它的 Configuration Words 要做設定。點選“Registers”即可進入 Configuration Words 的設定。

Easy Setup Registers Notifications : 8

▼ System Module

▼ Register: CONFIG1 0xEE4

BOREN	Brown-out Reset enabled	▼
CLKOUTEN	CLKOUT function is disabled. I/O or oscillator function on the CLKOUT pin	▼
CP	Program memory code protection is disabled	▼
FCMEN	Fail-Safe Clock Monitor is disabled	▼
FOSC	INTOSC oscillator: I/O function on CLKIN pin	▼
IESO	Internal External Switch Over mode is disabled	▼
MCLR	MCLR/VPP pin function is MCLR	▼
PWRT	PWRT disabled	▼

▼ Register: CONFIG2 0x1F87

BORV	Brown-out Reset Voltage (Vbor), low trip point selected.	▼
LPBOR	Low-Power BOR is disabled	▼
LVP	Low-voltage programming enabled	▼
PLLEN	4x PLL is always enabled	▼
PPS1WAY	The PPSLOCK bit cannot be cleared once it is set by software	▼
STVREN	Stack Overflow or Underflow will cause a Reset	▼
WRT	Write protection off	▼
ZCD	ZCD disable. ZCD can be enabled by setting the ZCDSEN bit of ZCDCON	▼

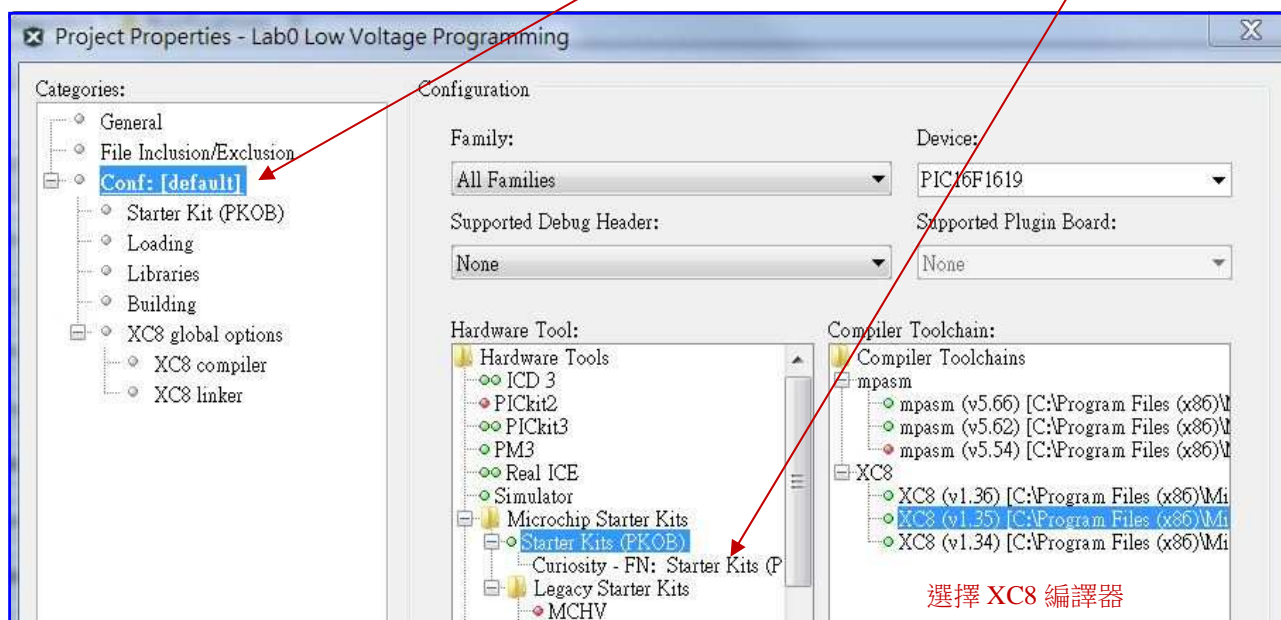
▼ Register: CONFIG3 0x3F1F

WDTCCS	Software control, controlled by WDTCS bits	▼
WDTCP	Software Control (WDTPS)	▼
WDTCS	Software WDT window size control (WDTWS bits)	▼
WDT	WDT disabled	▼

WDT 必須關閉

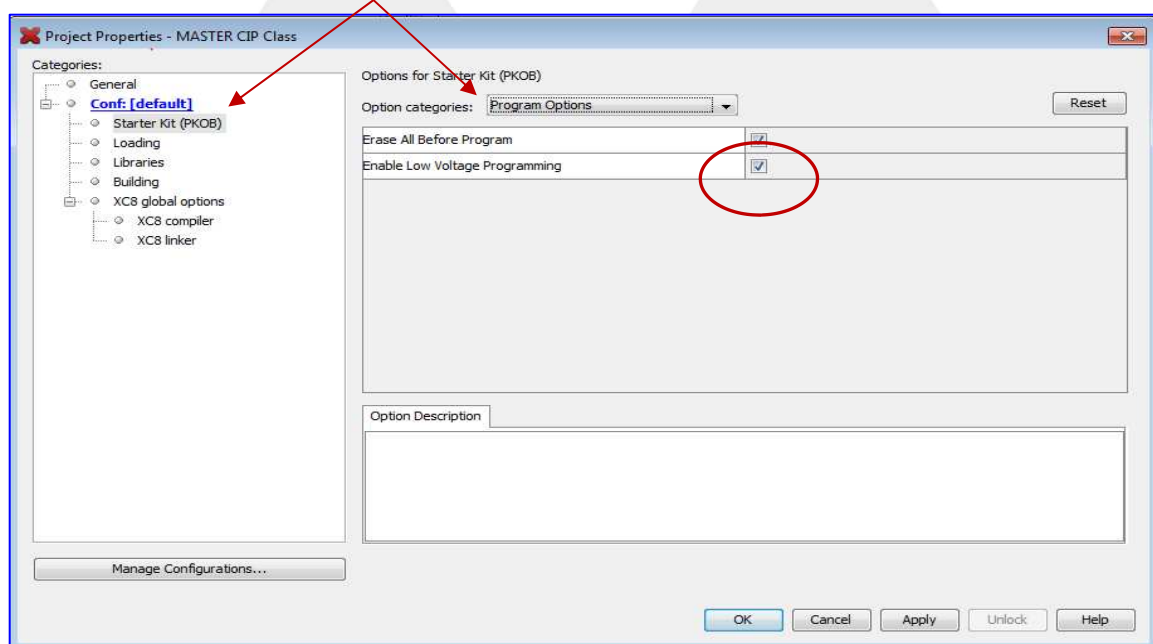
指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

2. 接下來將檢查酷奇板的連線問題。請確定酷奇板的 USB 線已經接好。在專案裡選擇 “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**。



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：

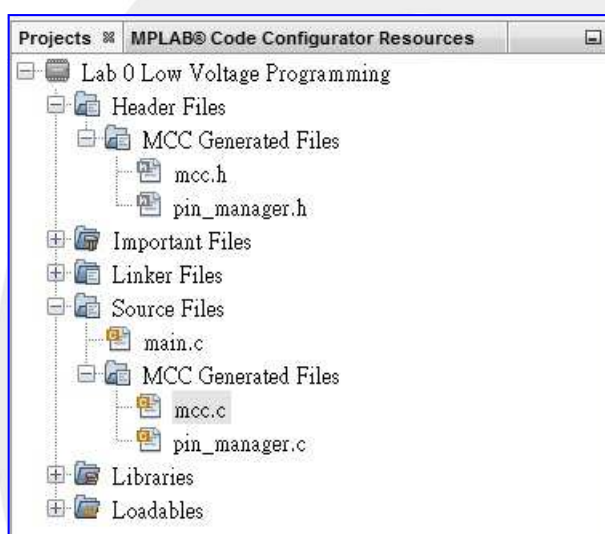
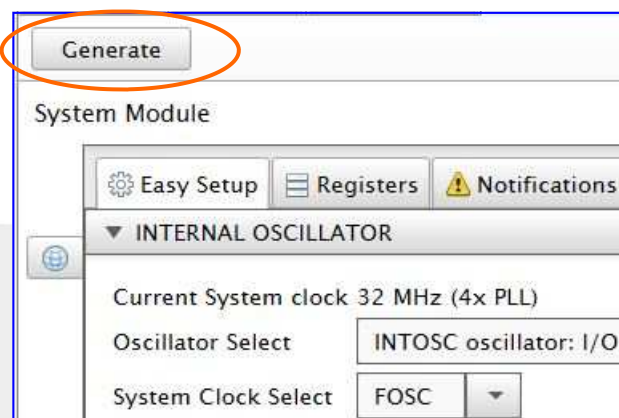
Erase All Before Program
Enable Low Voltage Programming



指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

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** 主程式 (可自動產生)
- ◆ **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 v3.0.3 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。

RTC教育訓練課程

Development Tools 相關課程

[XIDE MPLAB X IDE New!](#)

[XIDE MPLAB X IDE 中文版 New!](#)

8-Bits MCU 相關課程

[101ASP](#) PIC16F系列基礎課程 (原W100)

[102ASP](#) PIC18F系列基礎課程 (原W400)

[201ASP](#) PIC16F887周邊應用

[RELOCASM](#) Re-Locatable MPASM

[W201](#) PIC16F系列基礎課程

[W401 v3](#) MPLAB C18 C Compiler Workshop

[PIC18UBL v2](#) Bootloader with PIC18 New!

[TLS2118T v2.0](#) Getting Started with MPLAB C18 New!

[WAP002](#) PIC18F整合應用課程

[W402T v2.0](#) PIC18F整合應用課程 New!

[MCU1121T](#) HI-TECH PICC for PIC16F Series 2.0 New!

[W301](#) Advance PICC Application New!

[1601CLC](#) New Peripherals : CIP (CLC, NCO & CWG) New!

[XC8T v1.0](#) New!

[MCC201 v1.00](#) MPLAB Code Configurator New!

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 v3.0.3 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 Module” 的設定 (相同於 Lab 0 的設定)
4. 在 “Pin Module” 設定 GPIO (RA5 驅動 LED)
5. 在 Timer1 中斷裡的 Call Back() 加入 Toggle LED 的敘述

設定一般 I/O 的步驟: 在 Project Resources 點選 **“Pin Module”** 帶出 I/O 腳設定畫面。
點選右下角的腳位規劃圖示，將 RA5 的 Output 的鎖點一下即可變成 綠色的上鎖狀態。
其它週邊的腳位先不用管它。

Package:	PDIP20	Pin No:	19	18	17	4	3	2	13	12	11	10	16	15	14	7	6	5	8	9
			PORT A ▼					PORT B ▼					PORT C ▼							
Module	Function	Direction	0	1	2	3	4	5	4	5	6	7	0	1	2	3	4	5	6	7
OSC	CLKIN	input						🔒												
OSC	CLKOUT	output					🔒													
OSC	OSC1	input						🔒												
OSC	OSC2	input					🔒													
Pin Module	GPIO	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
Pin Module	GPIO	output	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
RESET	MCLR	input					🔒													

指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

再下來請到 Pin Module 的圖示介面，確定 RA5 為輸出腳，並將其 “Customer Name” 的輸入格裡輸入 腳位名稱 “LED”。

Pin Module

Easy Setup Notifications : 0

Selected Package : PDIP20

Pin N...	Module	Function	Custom Na...	Start Hi...	Analog	Output	WPU	OD	IOCP	IOCN
RA5	Pin Mo...	GPIO	LED	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

GPIO 已經設定完畢，接下來將設定 Timer1 來取得 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。

Prescaler 先設定最大除率 1:8 後即可看到 Timer 可設定週期的範圍，在這裡直接輸入”50mS” 做為 Timer1 的計時週期，並啟用中斷計時模式。

TMR1

Easy Setup Registers Notifications : 0

Hardware Settings

☒ Enable Timer

Timer Clock

Clock Source: FOSC/4

External Frequency: 32.768 kHz

Prescaler: 1:8

☒ Enable Synchronization

Timer Period

Timer Period: 1 us ≤ 50 ms ≤ 65.536 ms

Calculated Period: 50 ms

☐ Enable Gate

☐ Enable Gate Toggle Gate Signal Source: T1G

☐ Enable Gate Single-Pulse mode Gate Polarity: low

☒ Enable Timer Interrupt

☐ Enable Timer Gate Interrupt

Software Settings

Callback Function Rate: 5 x Time Period = 0.25 s

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。在這裡我們使用 Callback 的方式，也就是在 Timer1 的中斷裡會有一個計數器幫我們計數，直到發生五次的中斷才執行一次 Callback 函數並將此計數器歸零。

$$50\text{mS} \times 5 = 250\text{mS}$$

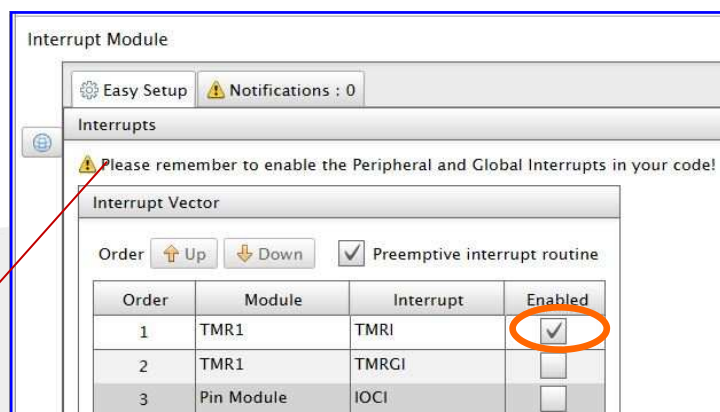
指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

因為 Timer1 有使用中斷功能，所以不要忘記在 “System” 下再開啟

“Interrupt Module” 來檢視一下中斷的設定。

右圖為 Interrupt Module 的設定，確定 Timer1 的中斷致能是有勾選的，同時也要注意其警告的內容說明。

Please remember to enable the Peripheral and Global Interrupt in your code!



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 PIE 及 GIE 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 LED 腳位的基本功能設定如右圖所示。

```
// 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。

```
177 void TMR1_CallBack(void)
178 {
179     LED_Toggle();
180     if(TMR1_InterruptionHandler)
181     {
182         TMR1_InterruptionHandler();
183     }
184 }
```

在 Tmr1.c 加入這行敘述後，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 PKOB 因 X IDE 版本不同需要更新韌體，這時請耐心等待韌體的更新，更新其間不可斷線。

Lab 2

利用 CLC ， PWM ， SPI 周邊來控制
RGB (WS2812B) 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 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(WS2812B)。

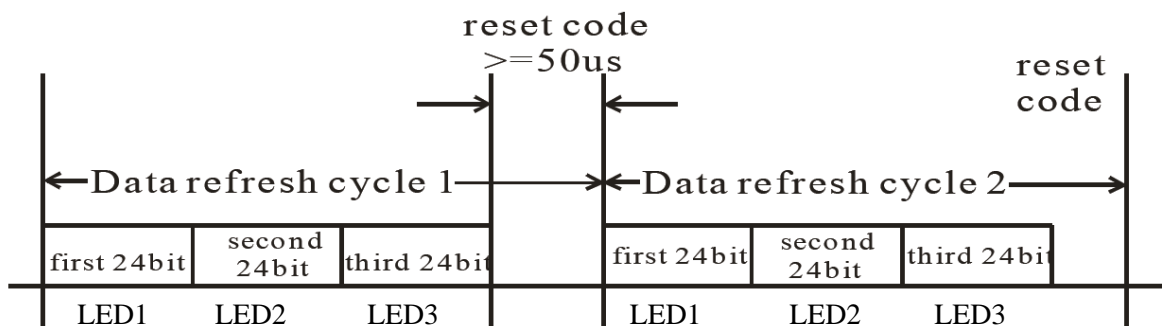
WS2812 系列 為一高速單線傳輸的 RGB LED，APP039 使用 4-pin 的 WS2812B 的元件，它可以完成 24-bit 的全彩顯示，並可作亮度控制。因為是使用 高速的 24-bit 串列編碼訊號，以 G(8-bit) R(8-bit) 及 B(8-bit) 組合而成。

WS2812B 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 WS2812B，所以可以做成一長串的 RGB LED 顯示條或 RGB LED 顯示板。

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

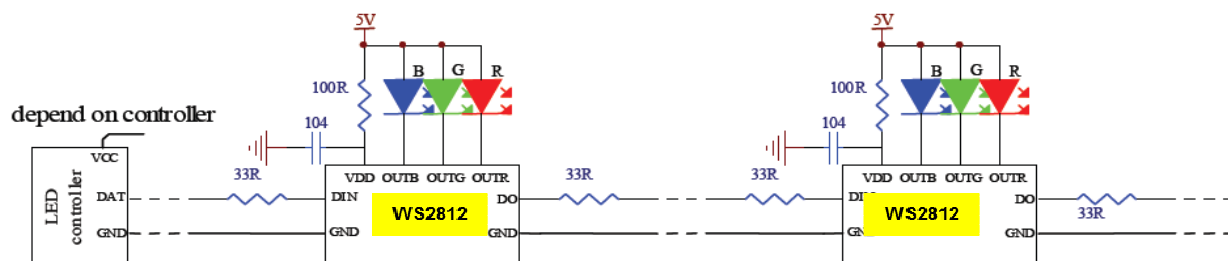
WS2812串接模式



由上圖所示，只要送出大於 50uS Low 訊號，WS2812B 就會重置並將顯示的訊號從 LED1 開始依序顯示。

指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

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

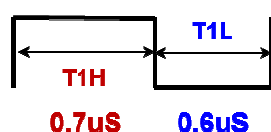
傳輸速度又是多少？依據 WS2812 資料手冊所示，其傳輸的速率不可以低於 400kps，建議使用 **800Kbps** 的速度，也就是傳輸 1 bit 的時間只有 1.25uS。

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。

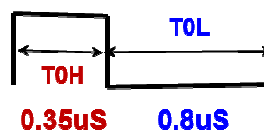
T1H 時間	位元 1 的編碼，Hi 週期所暫的時間	0.7uS	+ - 150nS
T1L 時間	位元 1 的編碼，Low 週期所暫的時間	0.6uS	+ - 150nS
T0H 時間	位元 0 的編碼，Hi 週期所暫的時間	0.35uS	+ - 150nS
T0L 時間	位元 0 的編碼，Low 週期所暫的時間	0.8uS	+ - 150nS
Reset	Low 電壓持續時間	大於 50uS 以上	

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圖所示。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

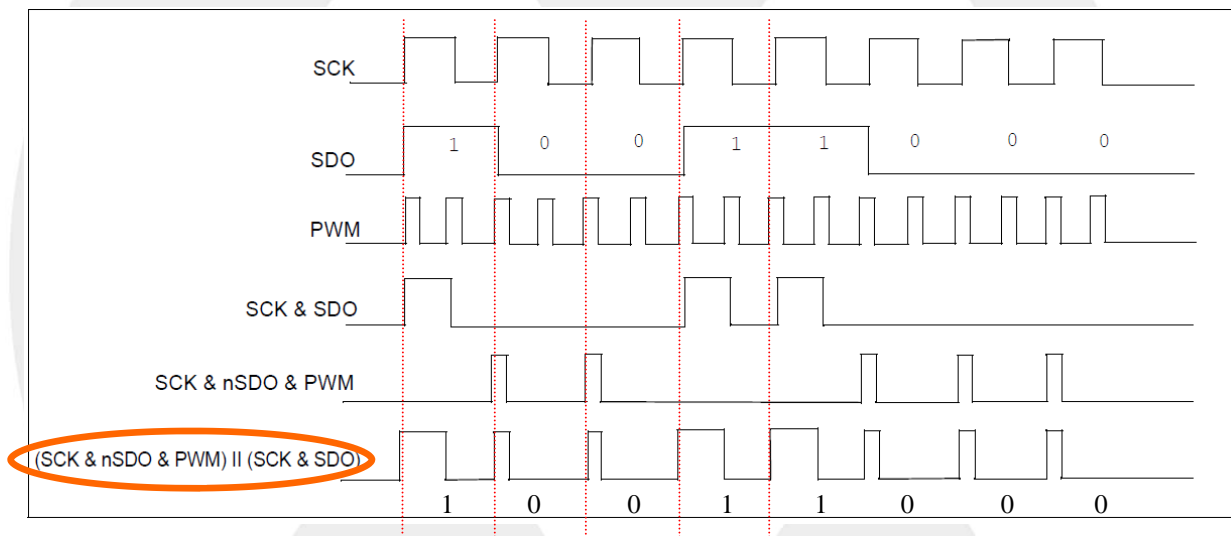


位元 0 的編碼

指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

看起來 WS2812 需求高速的單線式傳輸，最小的調變脈波寬度僅 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。

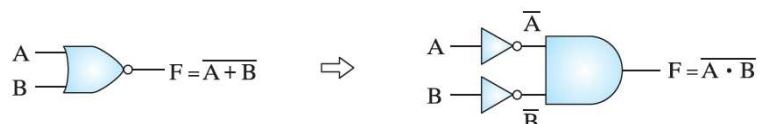
1. 基本上傳輸一個位元時間需 1.2uS，且一次以 8-bit 為一組，一次要傳送三個 Bytes 出去。所以考慮用 SPI 的傳輸模式。因為 SPI 傳輸速度可以達 10MHz 大大符合我們 800KHz 的傳輸速度。所以 SPI 的設定其 CLK 為 800KHz，其 CLK 的 Hi 週期為 0.75uS，Low 週期也為 0.75uS。
2. 至於最小的脈波 0.35uS 考慮用 PWM 固定來產生，此脈波主要是為了資料位元為 0 時的調變。所以設定 PWM Hi Duty 為 0.35uS，週期為 0.75uS，並與 SCK 的上升緣同步。
3. 到現在基本波都有了，在過來就要使用 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。



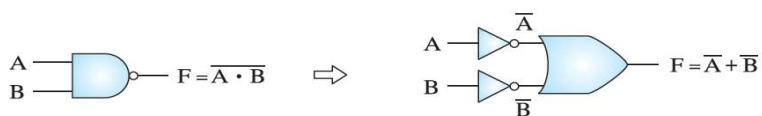
再去查看一下 CLC 的架構，發現 CLC 都是使用 OR Gate 做輸入的，與我們需要的 AND Gate 輸入需求是不一樣的，怎麼辦？

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: $\overline{A + B} = \overline{A} * \overline{B}$



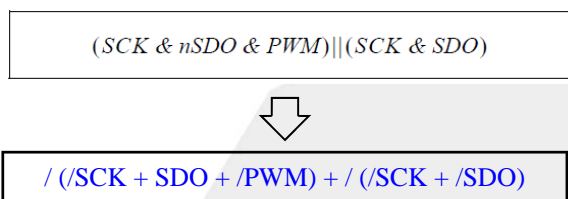
迪摩根第二定律: $\overline{A * B} = \overline{A} + \overline{B}$



指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

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:



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:

$$ABC = \overline{ABC} = \overline{A + B + C}$$

(轉換公式)

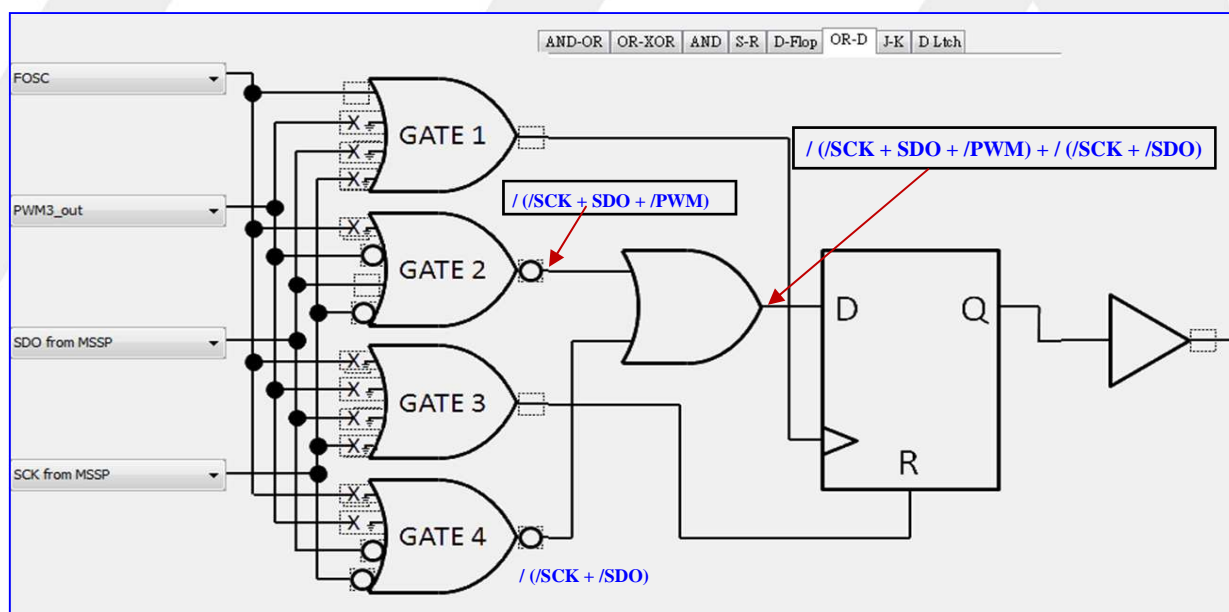
$$SCK \& /SDO \& PWM = / (/SCK + SDO + /PWM)$$

$$SCK * SDO = / (/SCK + /SDO)$$

輸出為兩項之和:

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

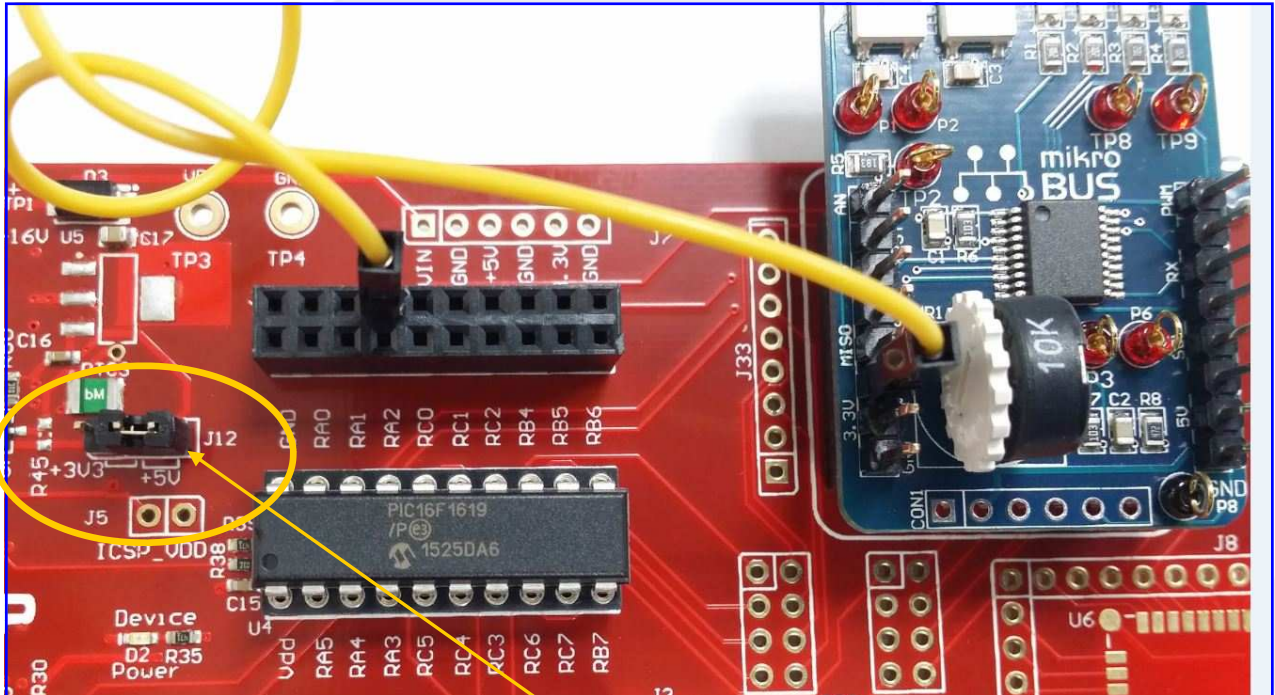
將上面的方程式實際選擇 OR-D 的 CLC 邏輯電路來設計，由下圖對照公式完全符合我們的需求。最後只要將輸出經過 D-Type 正反器在與 Fosc 取得同步過濾掉雜訊毛邊的訊號使輸出更加完美。



指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

接線的設定:

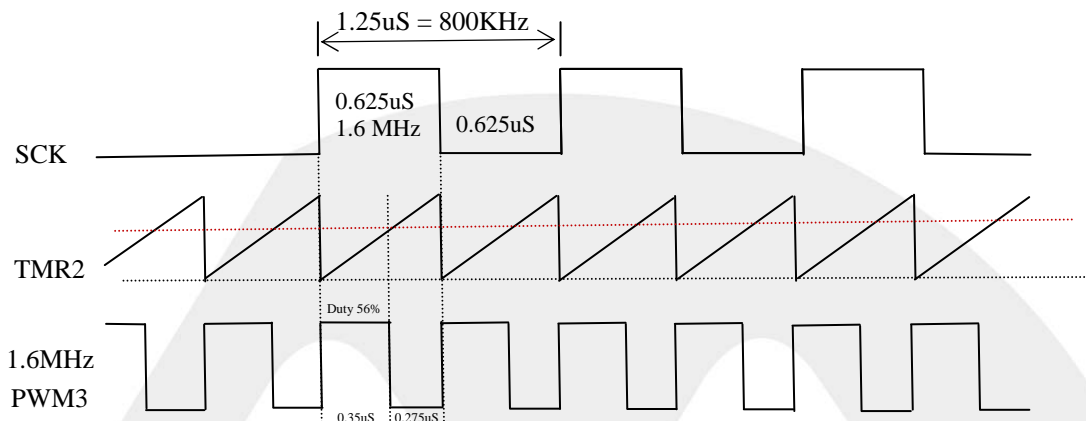
連接 J11 的 RA2 (CLC1) 的輸出訊號到 mikro BUS 子卡上的 MOSI pin (底下算來第三腳) 做為 WS2812B 的輸入訊號，其連接方式如下圖所示。



還有一點 WS2812B 的三色 LED 需使用 5V 的電壓，請確定 J12 的 Jumper 是選擇 5V 供電的位置。

指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

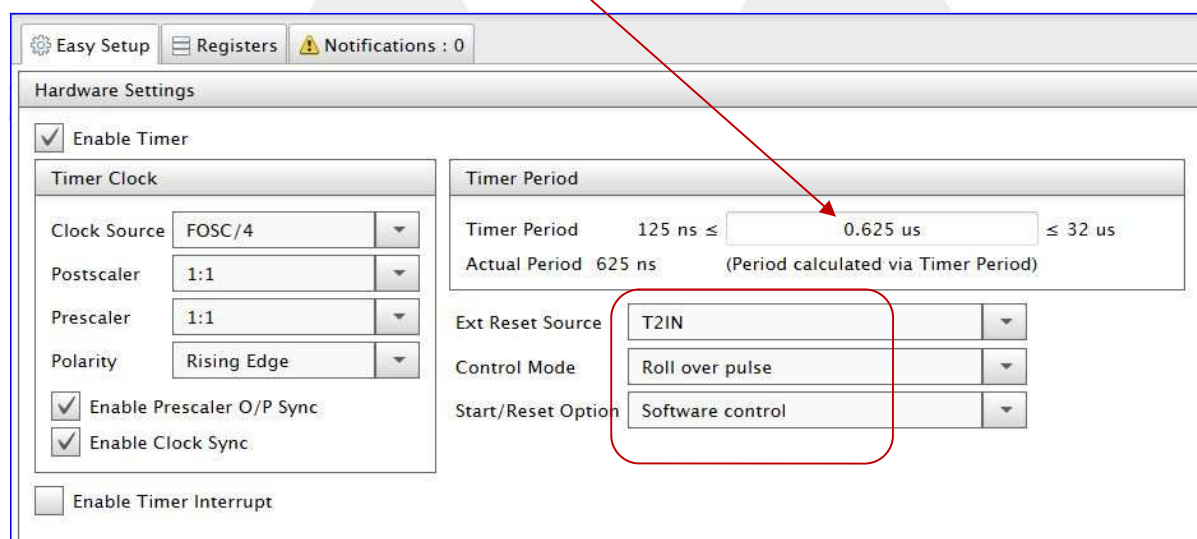
我們已經可以用 CLC 來組合所需的編碼來驅動 WS2818 RGB LED 了。接下來將繼續說明 Timer2，PWM 及 SPI SCK 的時脈關係的設定。首先看一下底下的相關的圖示：



由上圖可知整個時脈是由 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

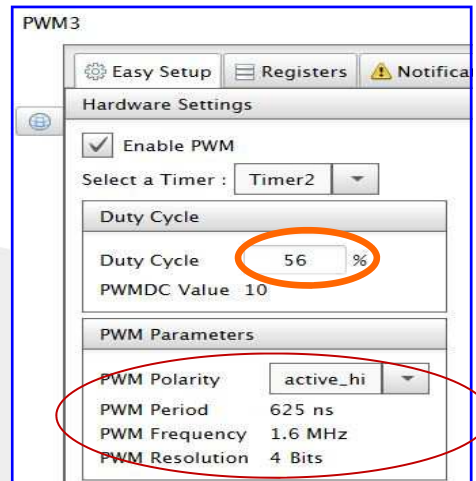
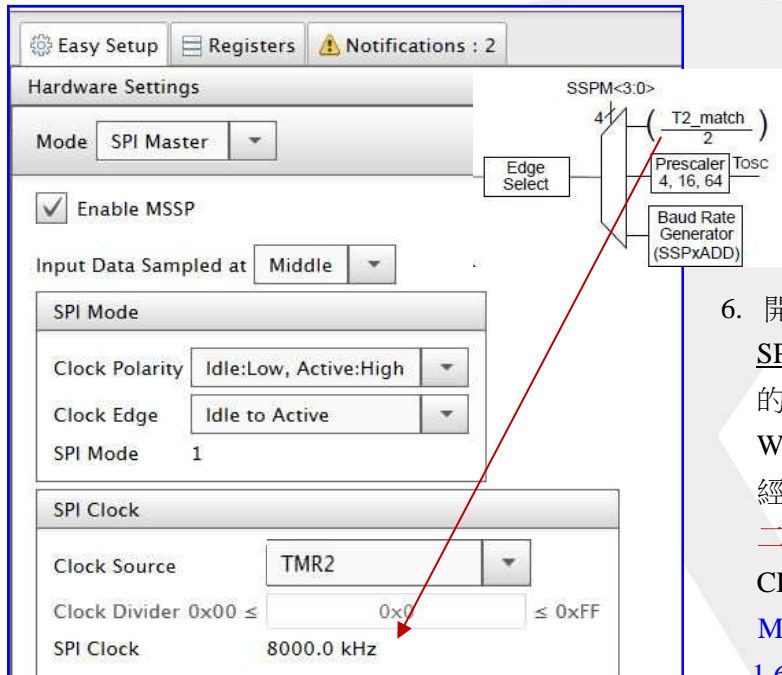
1. 開啟 MPLAB X IDE
2. 在 “File” 目錄下，開啟 LAB2_RGBLED.X 的專案
3. 在 “Tools” 目錄選擇 “Embedded “ 開啟 ” MPLAB Code Configurator” (MCC)
4. 開啟 TMR2: 的設定。設定週期 (0.625uS，1.6MHz)，請直接輸入 0.625uS。



5. 這 Timer2 所產生的 1.6MHz 是最基礎的參考頻率，這 1.6MHz 將做為底下 PWM3 設定的週期及 SPI 的串列輸出速度 (1.6MHz / 2 = 800KHz)

指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

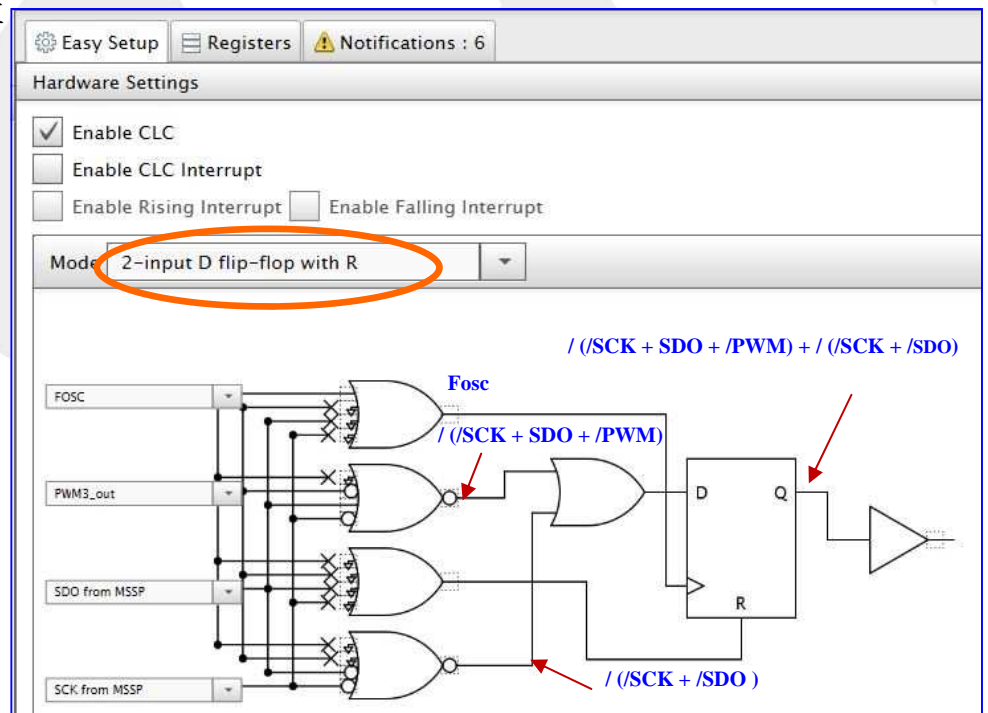
5. 開啟 PWM3 的設定。選擇 PWM3 的計時器為 Timer2。
所以 PMW 的週期為 0.625uS。週期需求為 0.35uS，所以
週期需求為 $0.625\mu\text{s} \times 56\% = 0.35\mu\text{s}$



6. 開啟 MSSP 設成 SPI Master 的模式。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。
MCC v3.0 在此 SPI Clock 會有錯誤計算成 1.6MHz 的速度。

7. 再來設定 CLC1 的邏輯輸出做為 WS2812B 的單線式控制訊號。選擇 CLC1 依照最後的輸出布林代數公式先設定輸入腳位及輸出腳位 (RA2)

$$\begin{aligned} & / (/(\text{SCK} + \text{SDO} + /(\text{PWM})) \\ & + \\ & / (/(\text{SCK} + /(\text{SDO})) \end{aligned}$$



指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

7. 最後在右上邊開啟 “Pin Manager” 選項, 設定 RA2 為 CLC1OUT (綠色上鎖)。選完之後要設定 RA2 的功能, 如下圖 “Pin Module” 的圖示, RA2 為 CLC1 的輸出腳請注意將 “Analog” 的選項空白並確定在 “Output” 的選項打勾。

The screenshot shows the Pin Manager interface. On the left, the "Pin Module" table lists various pins and their configurations. The RA2 pin is highlighted with an orange box. On the right, the "PORT A" table shows the configuration for pins 0 through 5. The RA2 pin is configured as CLC1OUT, output, and is locked (indicated by a green padlock icon).

Pin No.	Module	Function	Custom Na...	Start Hi...	Analog	Output	WPU
RA2	CLC1	CLC1OUT	RA2	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
RA5	CLC1	CLCIN3	RA5	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
RA5	TMR2	T2IN	RA5	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
RB4	MSSP	SDI	RB4	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
RB6	MSSP	SCK	RB6	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
RC1	CLC1	CLCIN2	RC1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
RC3	CLC1	CLCIN0	RC3	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
RC4	CLC1	CLCIN1	RC4	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Module	Function	Direction	0	1	2	3	4	5
CLC1	CLC1OUT	output	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
CLC1	CLCIN0	input	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
CLC1	CLCIN1	input	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
CLC1	CLCIN2	input	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
CLC1	CLCIN3	input	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
OSC	CLKIN	input	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
OSC	CLKOUT	output	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
OSC	OSC1	input	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

MCC 錯誤、警告及訊息提示功能: 在每個模組都會有所以使用中的模組訊息提示, 如下圖所示。基本上都要再三確認其設定, 尤其是出現 **Error** 及 **Warning** 的型態時一定要解決其衝突的。

Easy Setup Registers Notifications : 6			
Category	Module Name	Type	Description
	MSSP	INFO	MSSP uses TMR2 Module
	CLC1	INFO	CLC1 uses PWM3
	CLC1	INFO	CLC1 uses MSSP
	CLC1	INFO	CLC1 uses MSSP
	PWM3	INFO	PWM3 uses Timer2
	PWM3	HINT	Postscaler setting in Timer2 does not affect the calculated parameters in PWM module.

指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

8. 按下 MCC 的 Generate Code 來產生 MCC 所設定的週邊函數值初始設定。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= 0x00;                    // 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

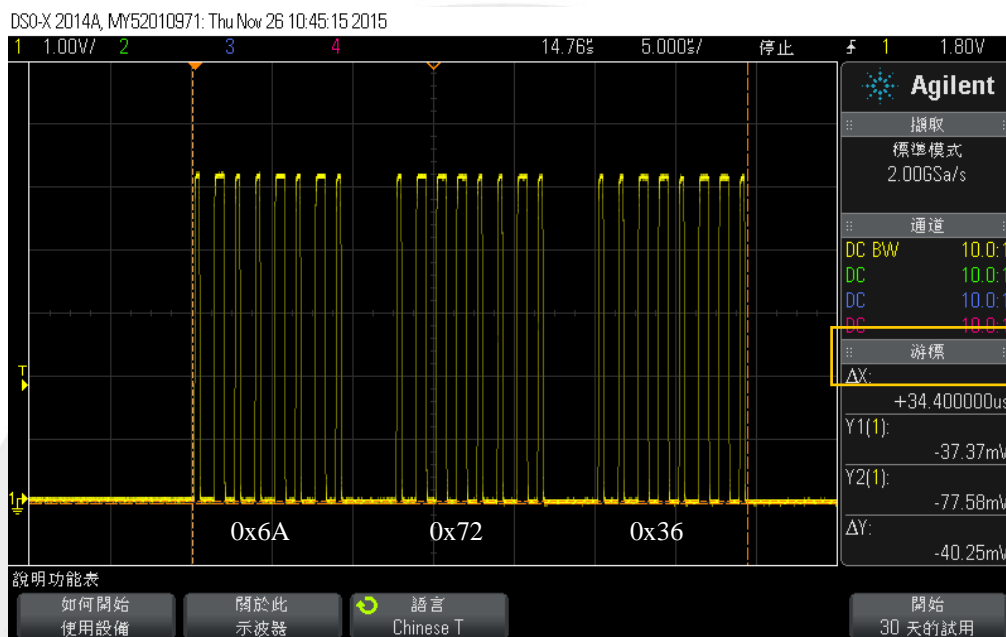
    Disp_Color.RGB.Green = (Disp_Color.RGB.Green +1) & 0x7F; // 改變顯示顏色
    Disp_Color.RGB.Red = (Disp_Color.RGB.Red +5) & 0x7F;
    Disp_Color.RGB.Blue = (Disp_Color.RGB.Blue -1) ;

    __delay_ms(150);                       // Reset WS2812B (CLC1OUT 持續 Low)
}
```

指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

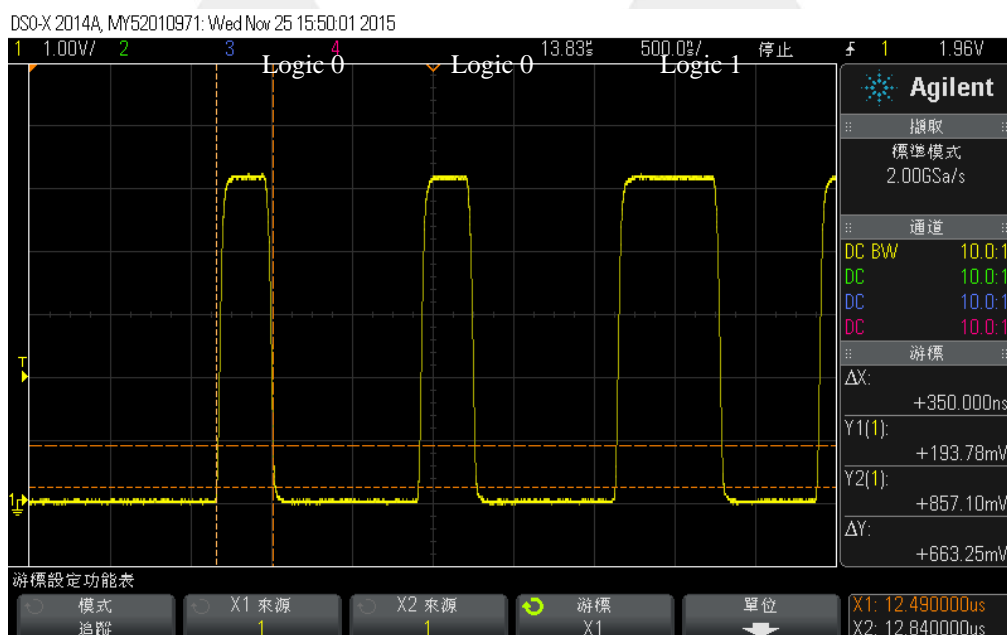
輸出訊號實測 (CLC1OUT) :

- 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)

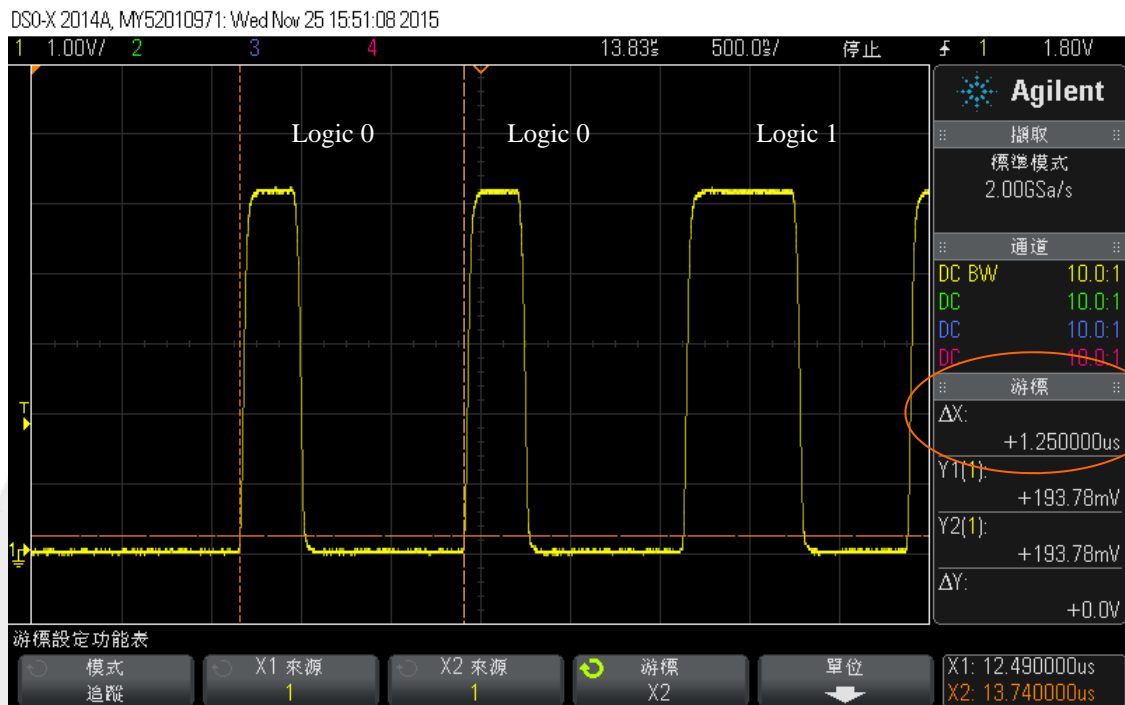
- 圖二為邏輯“0”編碼中 T0H 的脈波寬度為 0.35uS，T0L 為 0.90uS，符合 WS2812 的編碼時間。



(圖二)

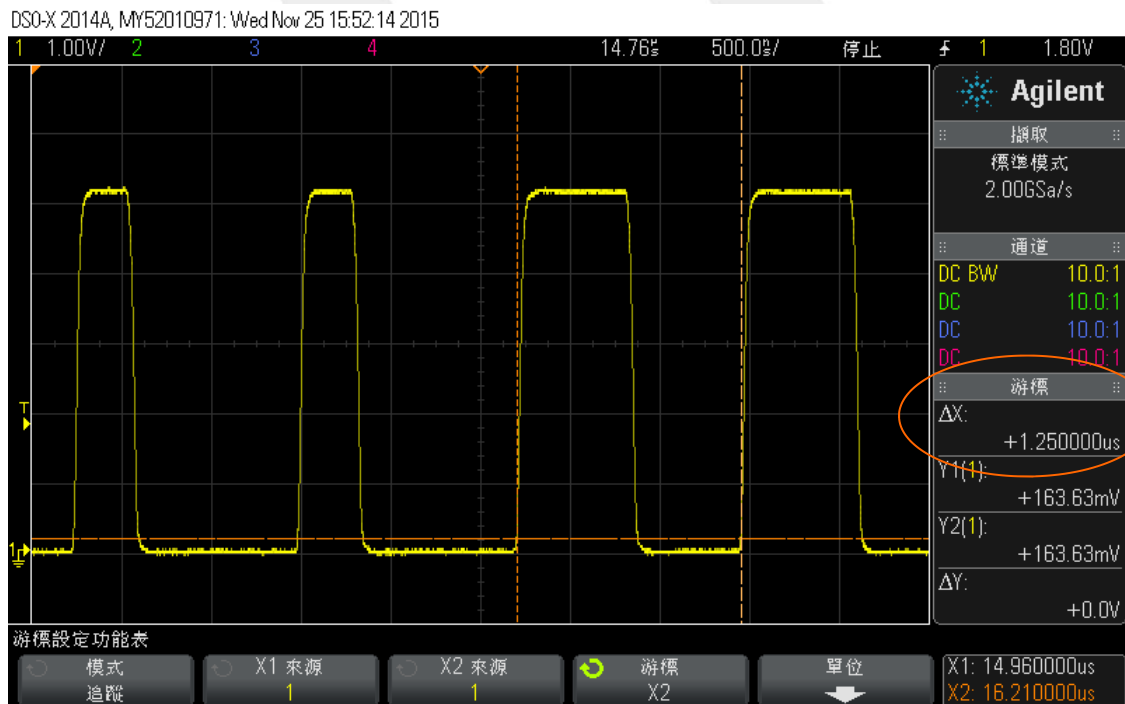
指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

- 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)

指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

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

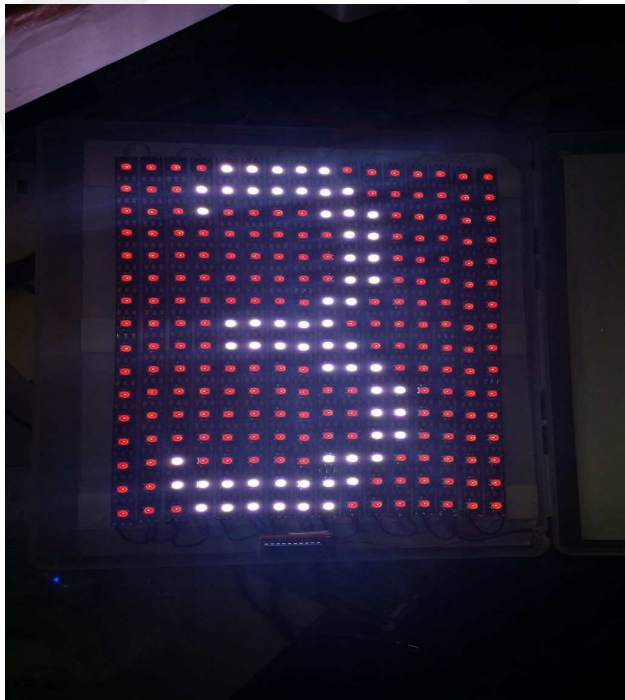
底下圖片為 WS2812 配合 CIP 所做出來的一些應用



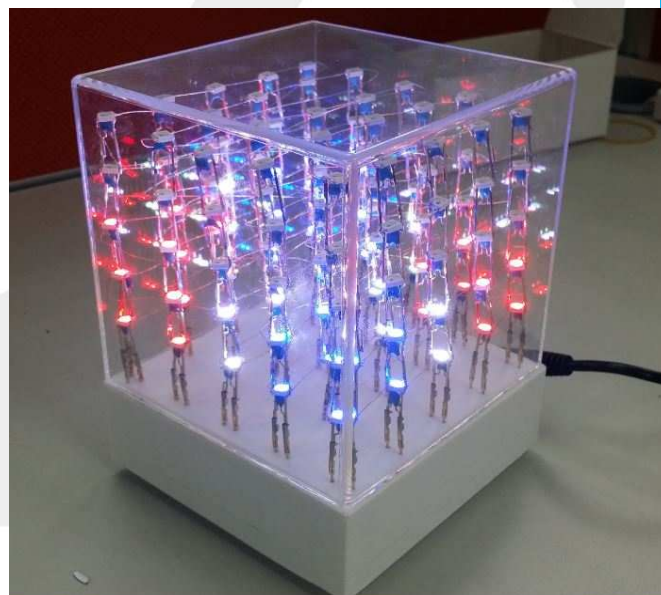
300 顆 RGB LED 顯示軟條



動態行人號誌顯示版



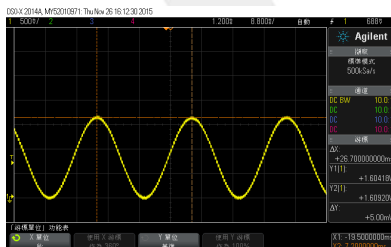
大型數字顯示板



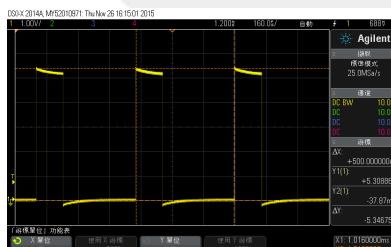
3D 立體顯示

Lab 3

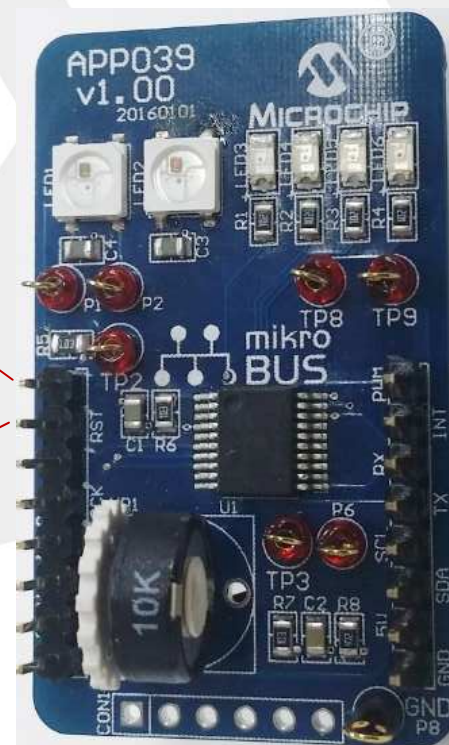
利用 SMT 周邊來測量子板上的 PWM 輸出的 Duty Cycle



ZCD 低頻正弦波輸出 (AN 腳)



可用 VR 調整 Duty 的 PWM 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%.

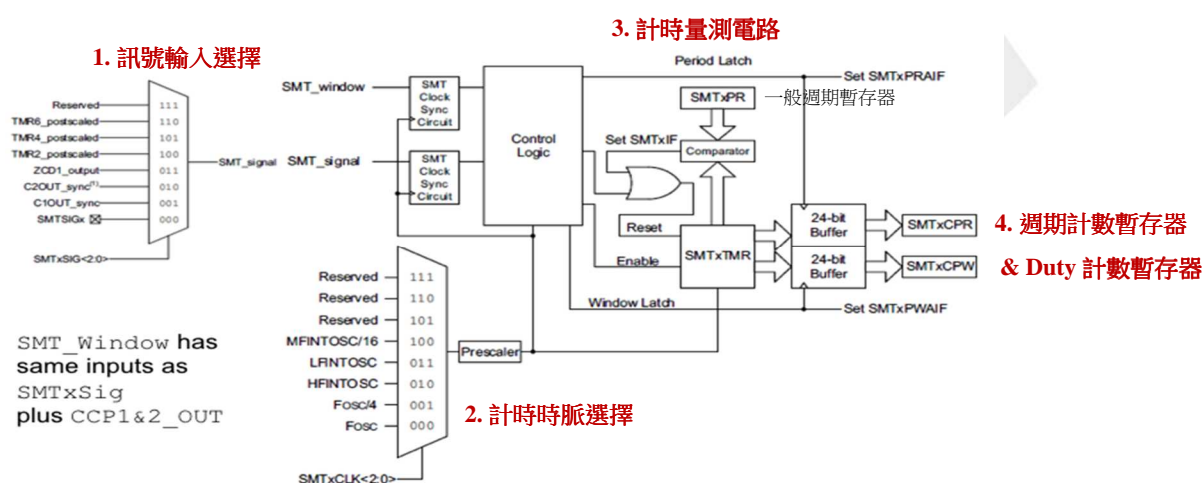
概述:

子卡已經寫好程式會自 RST 腳位送出一個可由可電電阻(VR1) 調整 Duty Cycle 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 VR1 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” 模組來測量 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:

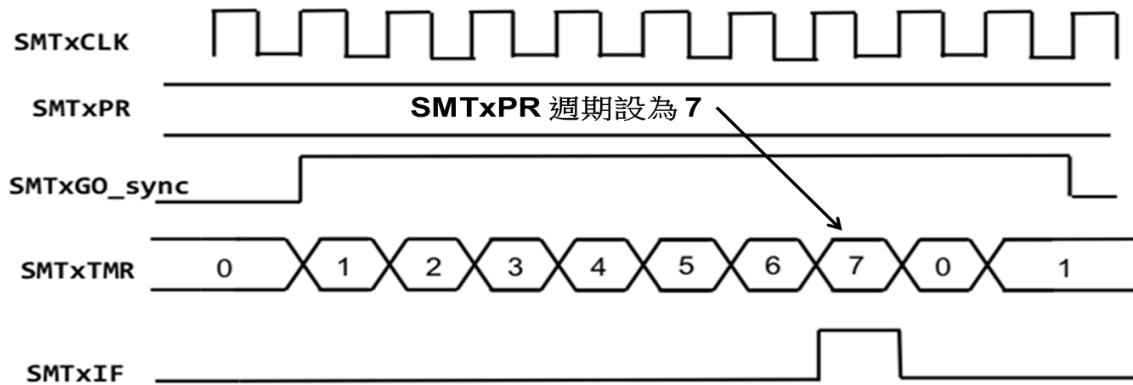
下圖為 SMT 方塊圖示，1. 訊號輸入多工選擇，可選擇待測量訊號。2. 計時時脈選擇器，選擇測量用的時脈來源。3. 待測訊號進到計時測量電路來測量輸入的時脈。4. 最後量測結果會送到 “SMTxCPR” 週期計數暫存器，及 “SMTxCPW” Duty 計數暫存器。



指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

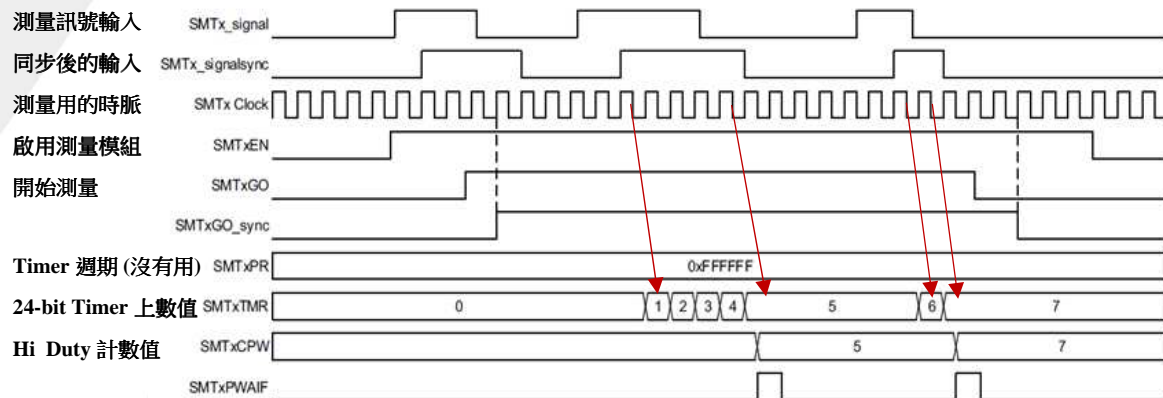
SMT 工作模式：

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)。這是最簡單適合長時間的計數應用。

2. Gated Timer Mode (Repeat Acquisition) : Lab3

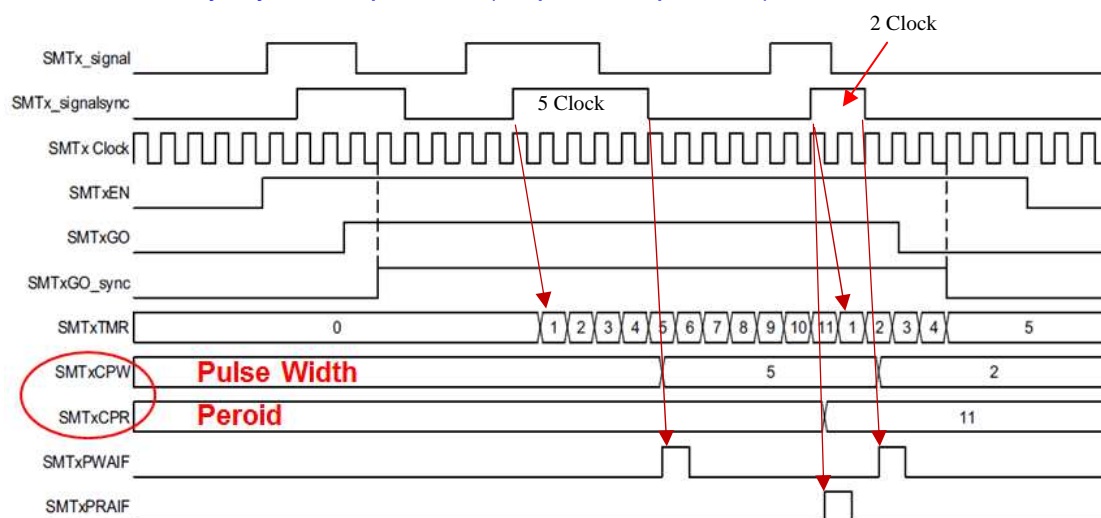


第一個 Duty 量測為 $5 - 0 = 5$ ，第二 Duty 量測為 $7 - 5 = 2$

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。

注意事項:

在做 SMT 練習時，原本在 MCC v2.25.2 使用 SMT1 來做練習是沒有問題的。但使用 MCC v3.0.3 時會出現 SMT1WIN 與 RA4 (D4) 腳位衝突的警告，且無法關閉 SMT1WIN 的輸入功能。這時可將 SMT1WIN 透過 PPS 設在沒有使用的 RC7。

使用除錯功能時，注意事項：

PIC16F1619 的除錯腳位在 RA0 (ICSPDAT) 及 RA1 (ICSPCLK)。因為 RA1 也接到 D5 (LED) 上。所以要使用除錯功能的話需暫時先將 RA1 的 LED 輸出設定移除，也就是不要動到 RA1 讓 RA1 可以做為除錯腳使用，待除錯完畢再恢復其應有的設定。

指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

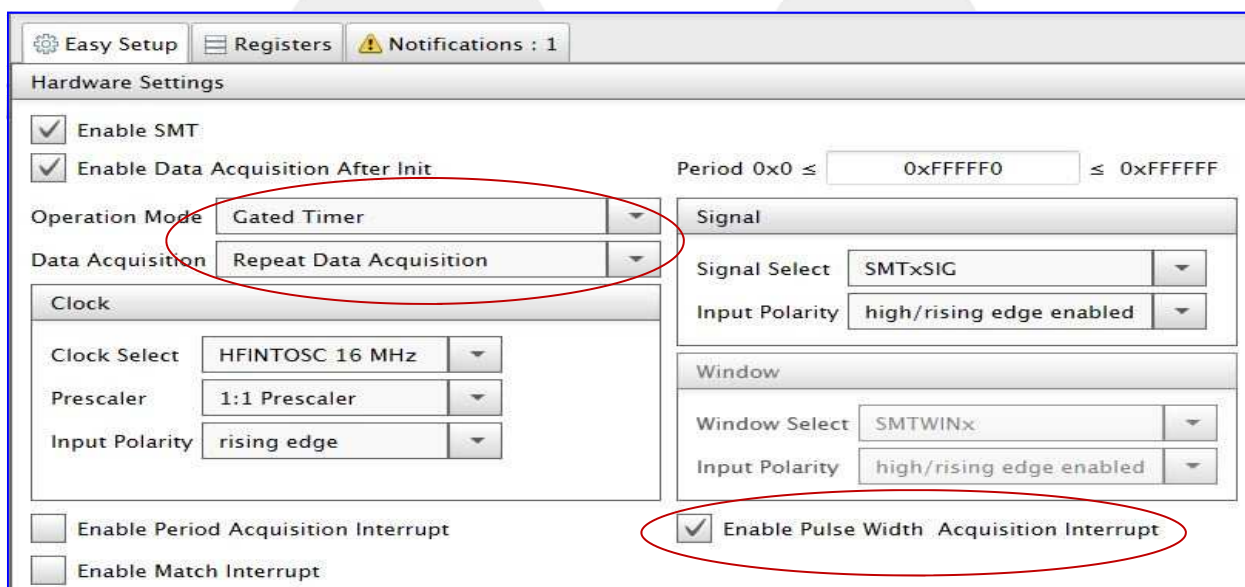
概述:

子卡已經寫好程式會自 RST 腳位送出一個可由可變電阻(VR1) 調整的 PWM 輸出訊號到 Curiosity 主板上的 RA4/SMT1SIG 腳位做為 SMT 的訊號輸入。調整 VR1 可以看到 D1 ~ D4 會依不同的 Duty 來點亮。(子卡的 PWM 輸出已經接到 PIC16F1619 的 RA4 的輸入腳，所以本練習無需再接線，直接可做 SMT1 的練習。)

同樣的，下一個 Lab 3-1 練習將使用 SMT 週邊的“**Period and Duty Cycle Acquisition**”模式，測量 duty cycle 後再去控制 Curiosity 板子上的 LEDs，讓顯示方式同子板的 LEDs。

Lab3 開始做實驗

1. 開啟 Lab3_SMT_GatedTMR_LED.X
2. 啟用 MCC
3. 設定 SMT1 工作模式為“**Gated Timer**”。資料獲取模式為“**Repeat Data Acquisition**”，並“**Enable Pulse Width Acquisition Interrupt**”
4. 量測的時脈使用內建的“**HFINTOSC 16MHz**”做為計數時脈。
5. 在 Gated Timer 的時脈圖可知，Duty 的計數是隨 SMT1TMR 往上數的，它並不會在新的 Duty 進來時歸零，所以要在中斷裡將 SMT1TMR 做歸零的動作，這樣才可以做為下一次新 Duty 的量測。
6. 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。



指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

- 關於腳位的設定如右圖所示，先在 **Pin Module** 中的 **output** 選項裡，選擇 **RA1, RA2, RA5 & RC5** 做為 LED 的輸出腳 (**GPIO 綠色上鎖的部分**) (注意: **SMT1WIN** 透過 PPS 設定在 **RC7** 以避免 **Default RA5** 的腳位衝突)。
- 接下了在 **Pin Module** 的設定項裡設為輸出腳位及賦予 **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)。

Pin Module

Easy Setup Notifications: 0

Selected Package: PDIP20

Pin N...	Module	Function	Custom Na...	Start Hi...	Analog	Output	WPU
RA1	Pin Mo...	GPIO	D5	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
RA2	Pin Mo...	GPIO	D6	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
RA4	SMT1	SMT1SIG	RA4	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
RA5	Pin Mo...	GPIO	D4	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
RC5	Pin Mo...	GPIO	D7	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
RC7	SMT1	SMT1WIN	RC7	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Package: PDIP20 Pin No: 19 18 17 4 3 2 13 12 11 10 16 15 14 7 6 5 8 9

Module	Function	Direction	PORT A ▼					PORT B ▼				PORT C ▼						
			0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
OSC	CLKIN	input																
OSC	CLKOUT	output																
OSC	OSC1	input																
OSC	OSC2	input																
Pin Module	GPIO	input																
Pin Module	GPIO	output																
RESET	MCLR	input																
SMT1	SMT1SIG	input																
SMT1	SMT1WIN	input																

注意: 有時在修改腳位設定時，自定的 **Customer Name** (如 D5 ~ D7) 會不預期的變為預定的名稱，這點在做 **C=Generate**

- 因中斷有用到 LEDs 的腳位定義，所以需在 **smt1.c** 裡加入 **pin_manager.h**

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

- 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。

```
smt1.c

void SMT1_PW_ACQ_ISR(void) {

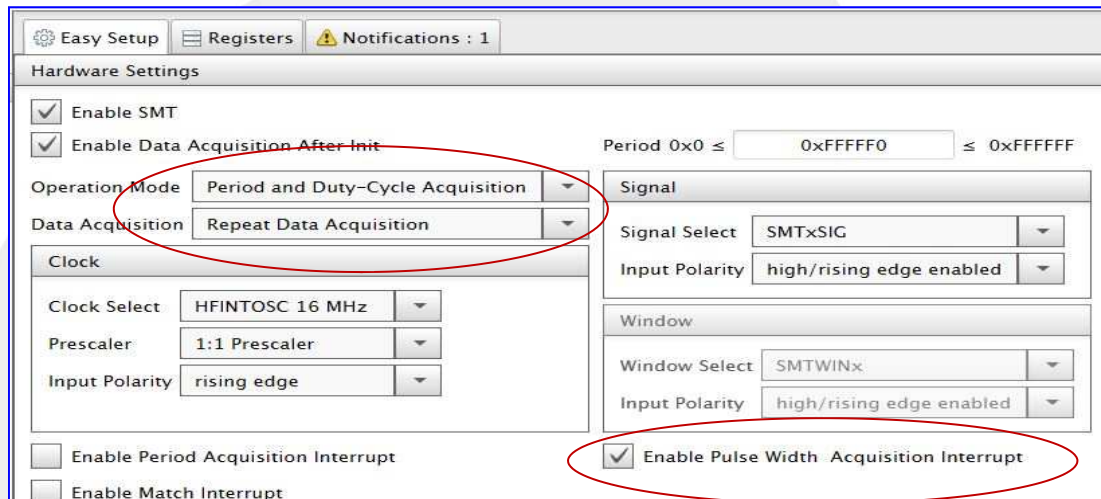
    // Clear SMT1 pulse width acquisition interrupt flag bit.
    PIR4bits.SMT1PWAIF = 0;

    if(SMT1CPW > 6000)
    {
        D4_SetHigh();
        D5_SetLow();
        D6_SetLow();
        D7_SetLow();
    }
    else if(SMT1CPW > 4000)
    {
        D4_SetLow();
        D5_SetHigh();
        D6_SetLow();
        D7_SetLow();
    }
    :
    :
    {
        D4_SetLow();
        D5_SetLow();
        D6_SetLow();
        D7_SetHigh();
    }
    SMT1TMR = 0;    // SMT1 Timer = 0;
}
```

指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

Lab3-1 開始做實驗

1. 開啟 Lab3-1_SMT_Grt_Duty.X
2. 啟用 MCC
3. 設定 SMT1 工作模式為 “Period and Duty Cycle Acquisition” 及啟用資料獲取模式為 “Repeat Data Acquisition”，並啟用 “Pulse Width Acquisition Interrupt”



4. I/O 腳位的設定及名稱與 Lab3 相同。
5. 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

```
bit SMT_GO_Flag;

while (1)
{
    if (SMT_GO_Flag)           main.c
    {
        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：

```
void SMT1_PW_ACQ_ISR(void) {

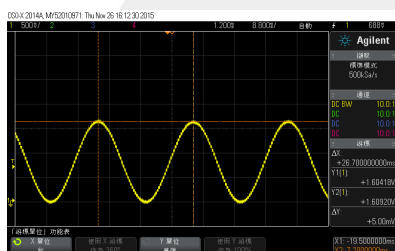
    // Clear SMT1 pulse width acquisition
    interrupt flag bit.

    PIR4bits.SMT1PWAIF = 0;
    SMT_GO_Flag = 1;

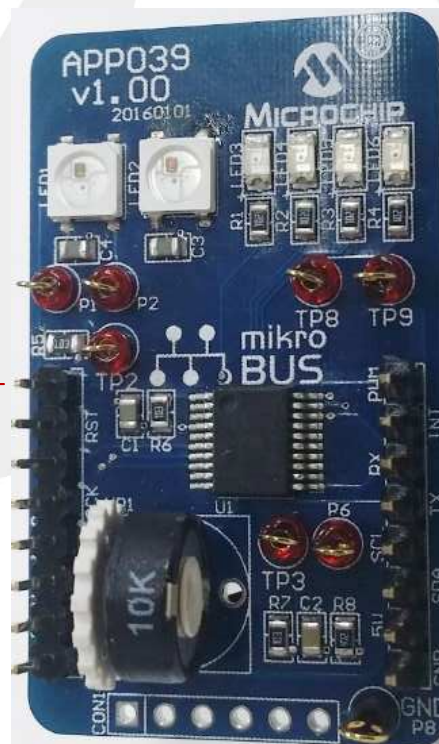
}
```

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。

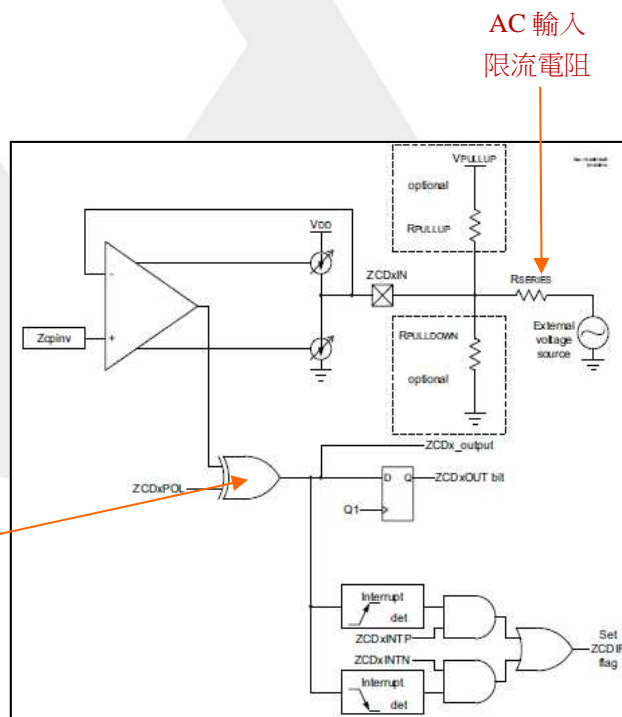
概述:

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**限流電阻**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，輸入訊號經電阻輸入至 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 (XOR Gate) 後輸出 ZCDxOUT。



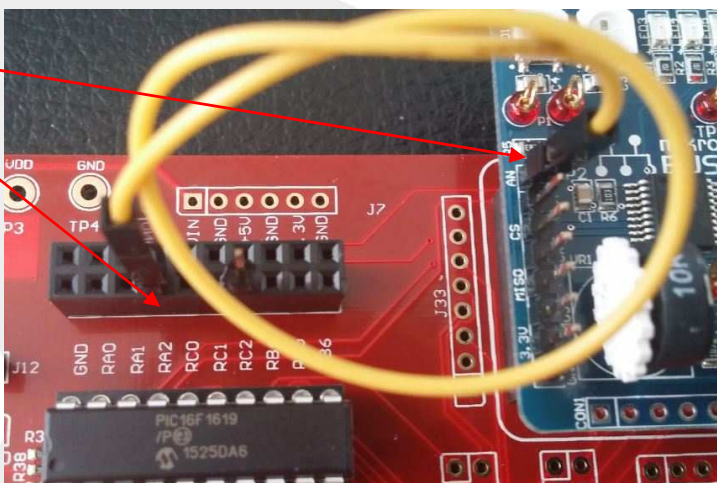
指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

實驗的接線：

子板上的弦波輸出 TP2 經串聯電阻(18K ohm, R5) 後接到 AN 腳輸出一正弦波。
所以因負載關係，你不一定就可以在 RA2 量到正弦波的波型，因為它可能被 ZCD 的電壓給吃掉了。所以量測正弦波請在子卡上的 TP2 量取。

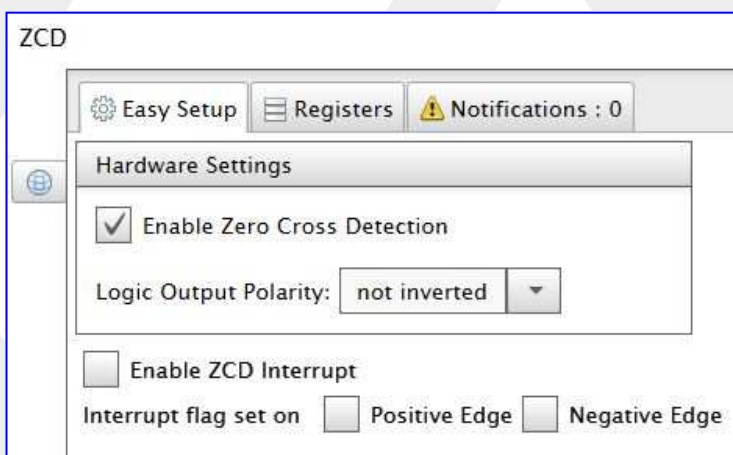
- ◆ AN 腳為子卡上的正弦波輸出
- ◆ RA2 規劃為：ZCD1IN 輸入。
- ◆ RA5 規劃為：ZCDOUT 輸出腳

所以需跳接一條 AN 到 RA2 的跳線。(如右圖所示)



Lab4 開始做實驗

1. 開啟 Lab4 _ZCD_Basic.X
2. 啟用 MCC
3. 設定 ZCD 模組功能是非常簡單的。如右圖的設定所示：只要啟動 ZCD 並將其輸出不做反向。



Pin Module

Easy Setup

Notifications : 0

Selected Package : UQFN20

Pin N...▲	Module	Function	Custom Na...	Start Hi...	Analog	Output	WPU
RA2	ZCD	ZCDIN	RA2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
RA5	ZCD	ZCD10...	RA5	<input type="checkbox"/>		<input checked="" type="checkbox"/>	<input type="checkbox"/>

Package: UQFN20

Pin No: 16 15 14 1 20 19 10

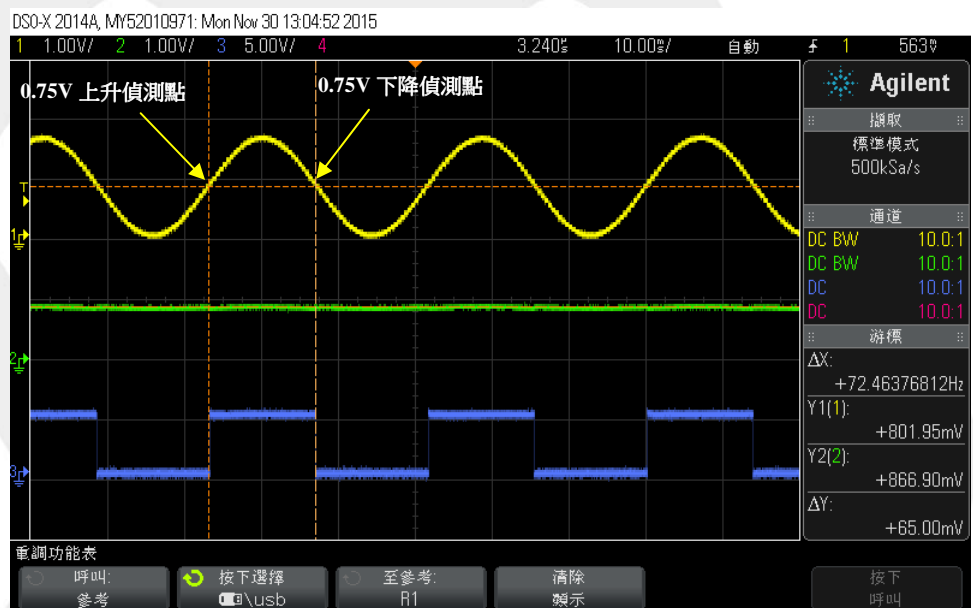
PORT A ▼

Module	Function	Direction	0	1	2	3	4	5	4
OSC	CLKIN	input							
OSC	CLKOUT	output							
OSC	OSC1	input							
OSC	OSC2	input							
Pin Module	GPIO	input							
Pin Module	GPIO	output							
RESET	MCLR	input							
ZCD	ZCD1OUT	output							
ZCD	ZCDIN	input							

指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

4. 由於 ZCD1 的腳位輸入 ZCD1IN 是固定在 RA2 的腳位。因為 ZCD 模組是屬於類比輸入的模組，所以在設定 RA2 時，需將 RA2 設成 Analog 輸入功能。(如上圖所示)
5. ZCD1OUT 的腳位必須經由 PPS 做設定的，在此實驗我們設定 ZCD1OUT 的輸出導向在 RA5 的腳位做輸出 (與 D4 同腳位)。
6. 設定完後即可產生 MCC 設定程式，產生的成是在這裡不做任何的修改直接就可以編譯、燒錄、執行。

1. 輸入 TP2 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，其上緣及下降緣點都發生在 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



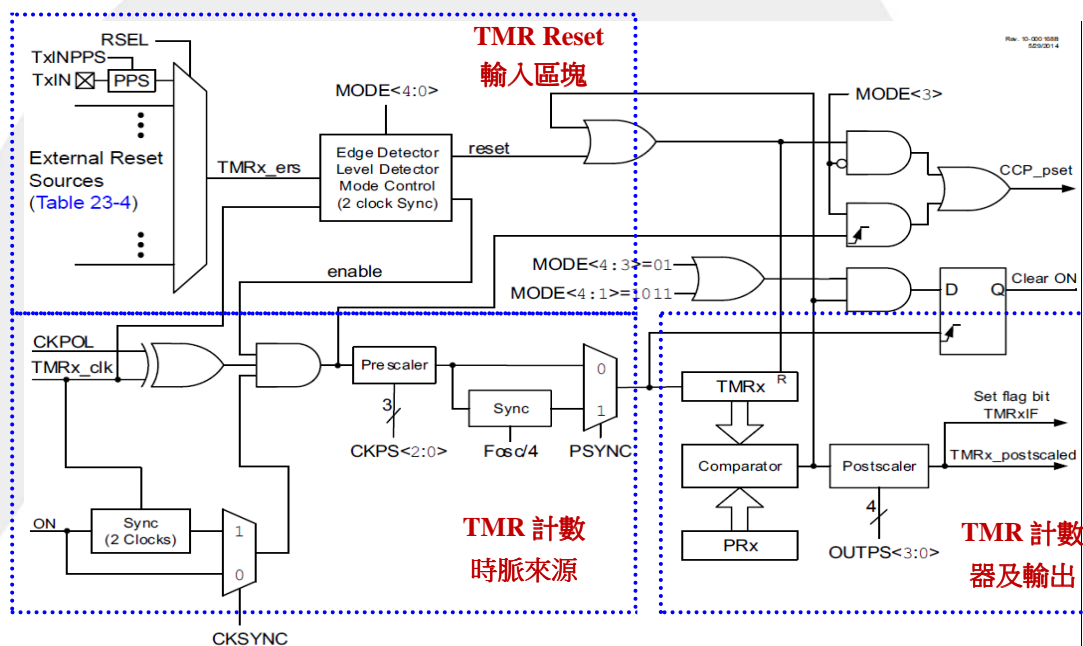
以上的動作無需主程式的執行，
經設定後 CIP 會自行運作

指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

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：

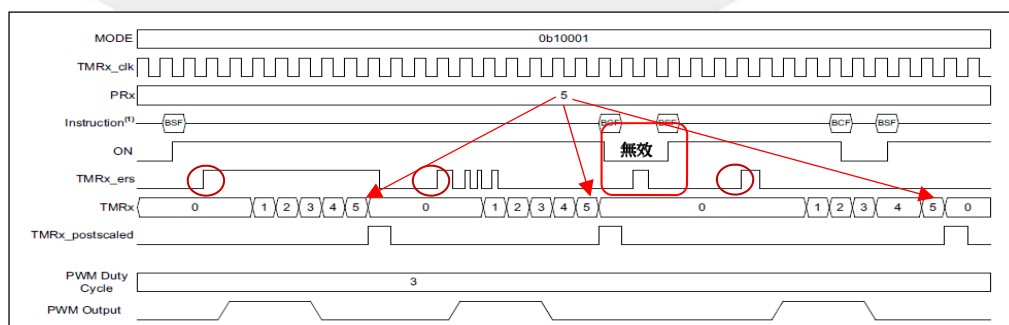
1. Free Running Period (Roll Over Pulse) (自由運行計數模式)
2. One-Shot (單次觸發計數模式)
3. Mono-Stable (單穩態計數模式)

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: 跟傳統的 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作動一次的方式。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 再次觸發計數。



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：

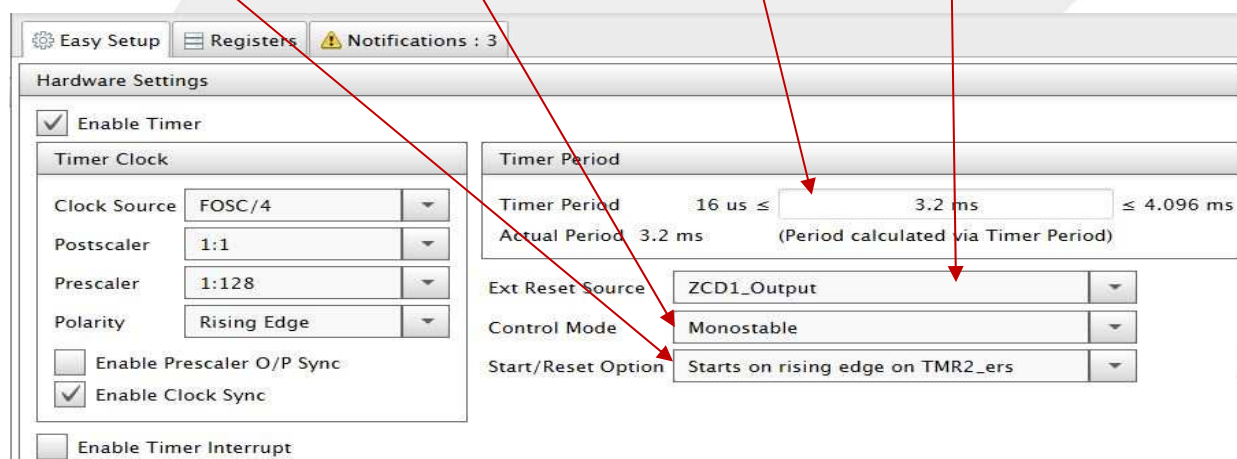
因為使用的是觸發模式，所以會設定 HLT/TMR2 在 “Monostable” 模式下做設定。也就是 ZCD 每觸發一次 TMR2 就計數一次，PWM 也同步產生 1.6 mS 的脈波。

Lab5 開始做實驗 (子卡上 AN 到 RA2 的跳線接了沒?)

1. 開啟 Lab5_ZCD_HLT_PWM.X
2. 啟用 MCC
3. 設定 ZCD，設定如右圖所示



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。



指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

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 (TMR2 所設定的)，如果要輸出 1.6mS 的 Duty Cycle 就要設定 $1.6\text{mS} / 3.2\text{mS} = 50\%$ 。

注意: 有時輸入 Duty 50% 時，底下的 PWMDC Value 不一定會出現 399 的值。此時請多試幾次或到 “Registers” 的設定項裡，直接輸入如右圖的輸入值：

Register: PWM3DCH 0x63
DC 0x63

Register: PWM3DCL 0xC0
DC 0x3

Hardware Settings

☒ Enable PWM

Select a Timer : Timer2

Duty Cycle

Duty Cycle 50 %

PWMDC Value 399

PWM Parameters

PWM Polarity active_hi

PWM Period 3.2 ms

PWM Frequency 312.5 Hz


PWM Resolution 9 Bits


6. 最後再設定一下腳位的輸出 (ZCD1OUT & PWM3OUT)。

ZCD1OUT 的輸出腳位設定是為了方便使用示波器量測 ZCD 的輸出訊號，預設的 ZCD1OUT 腳位是在 RA5，但這會與 T2IN 衝突的。這裡將 T2IN 利用 PPS 將腳位設在 RB4。該練習的腳位設定是: ZCD1IN 設在 RA2。ZCD1_OUT 設在 RA5。PWM3OUT 設在 RC5。

Package:	PDIP20	Pin No:	19	18	17	4	3	2	13	12	11	10	16	15	14	7	6	5	8	9
			PORT A▼							PORT B▼				PORT C▼						
Module	Function	Direction	0	1	2	3	4	5	4	5	6	7	0	1	2	3	4	5	6	7
PWM3	PWM3OUT	output	🔒	🔒	🔒		🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
Pin Module	GPIO	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
Pin Module	GPIO	output	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
RESET	MCLR	input				🔒														
TMR2	T2IN	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
ZCD	ZCD1OUT	output	🔒	🔒	🔒		🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
ZCD	ZCDIN	input			🔒															

Pin Module

 Easy Setup

 Notifications : 3

Selected Package : PDIP20

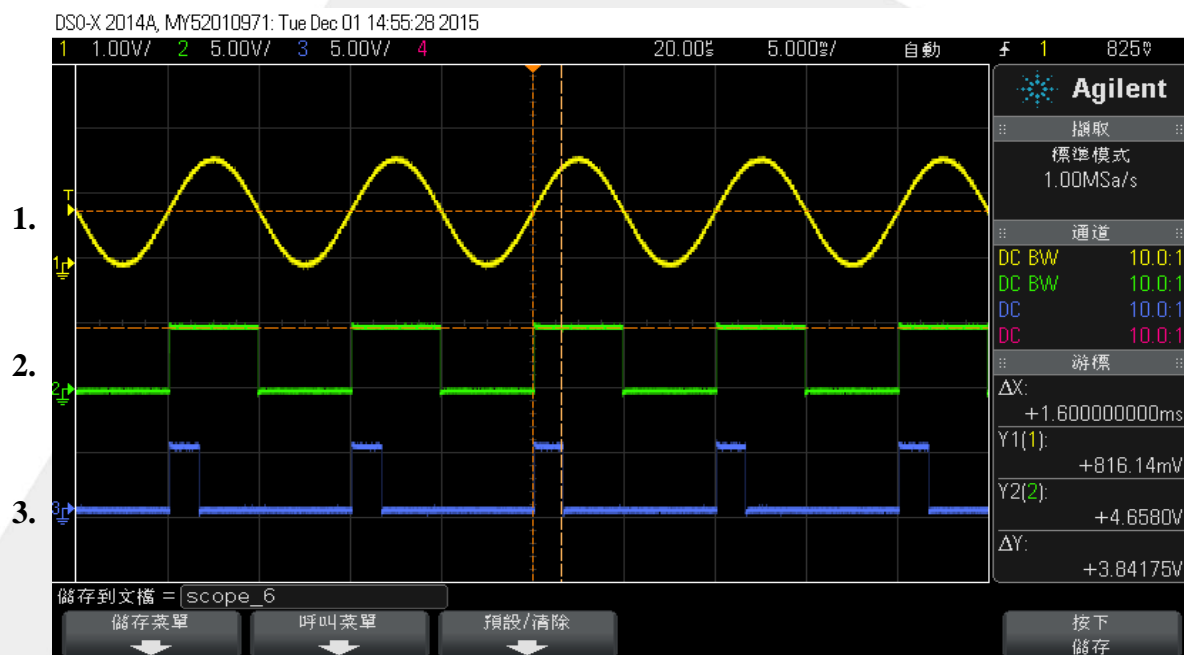
Pin N...▲	Module	Function	Custom Na...	Start Hi...	Analog	Output	WPU
RA2	ZCD	ZCDIN	RA2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
RA5	ZCD	ZCD1O...	RA5	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
RB4	TMR2	T2IN	RB4	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
RC5	PWM3	PWM3...	RC5	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

T2IN 雖沒使用，但已設成 Digital Input 的功能，為避免輸入腳浮接，在這裡啟用 WPU (Internal Weak Pull-Up)。以避免外部雜訊的干擾。

指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

實際實驗的測量：

1. 子卡TP2 的輸入的正弦波
2. ZCD1OUT (RA5) 的偵測 AC 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，再將這組合驅動 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。

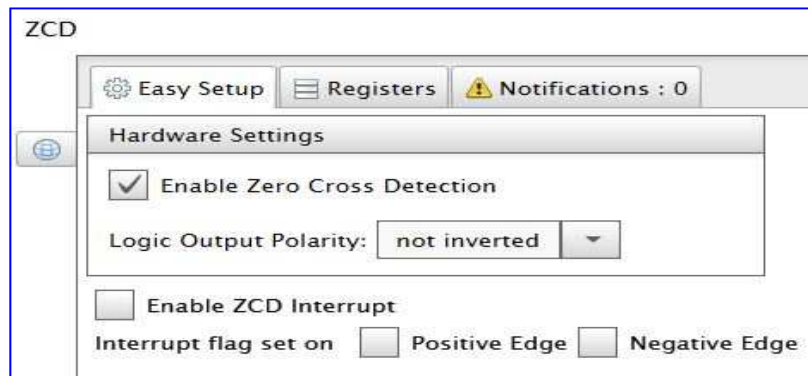
本實驗將使用到的 CIP 的週邊有 ZCD、HLT/TMR2、HLT/TMR4、PWM4 及當做除錯輸出訊號用的 CLC。底下將一一說明其設定及工作需求。

1. 實驗起點是利用 ZCD 取得到一個上緣交越的觸發訊號為開始，接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作且 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 ($1.28\text{mS} / 80\text{uS} = 16$ 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。

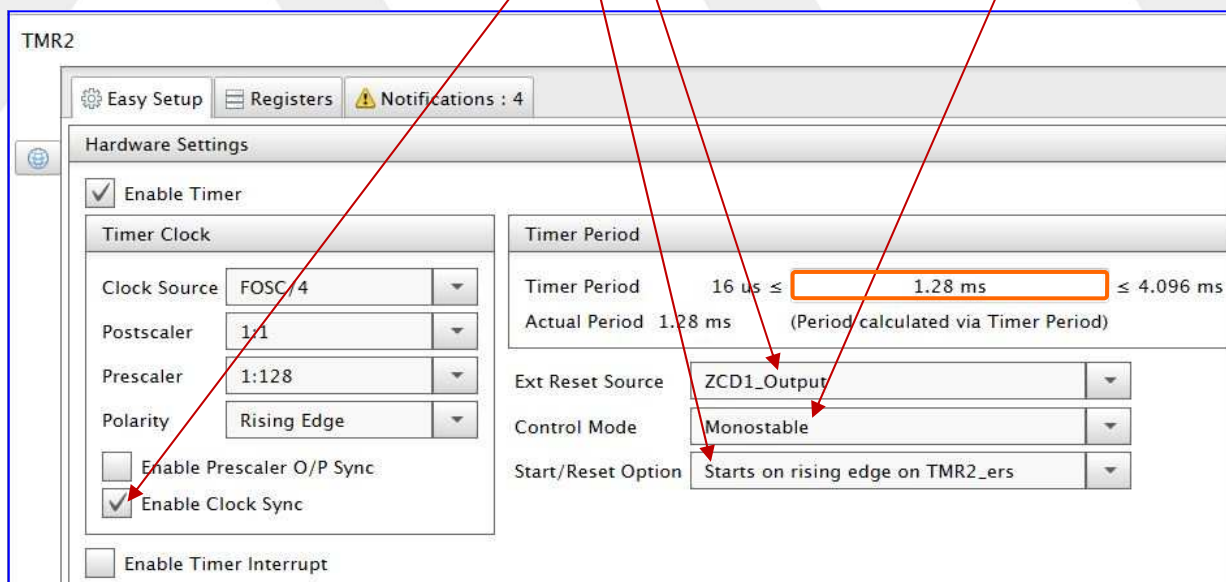
Lab6 開始做實驗 (子卡上 AN 到 RA2 的跳線接了沒?)

1. 開啟 Lab6_ZCD_TRIAC.X
2. 啟用 MCC
3. 設定 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。

指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習



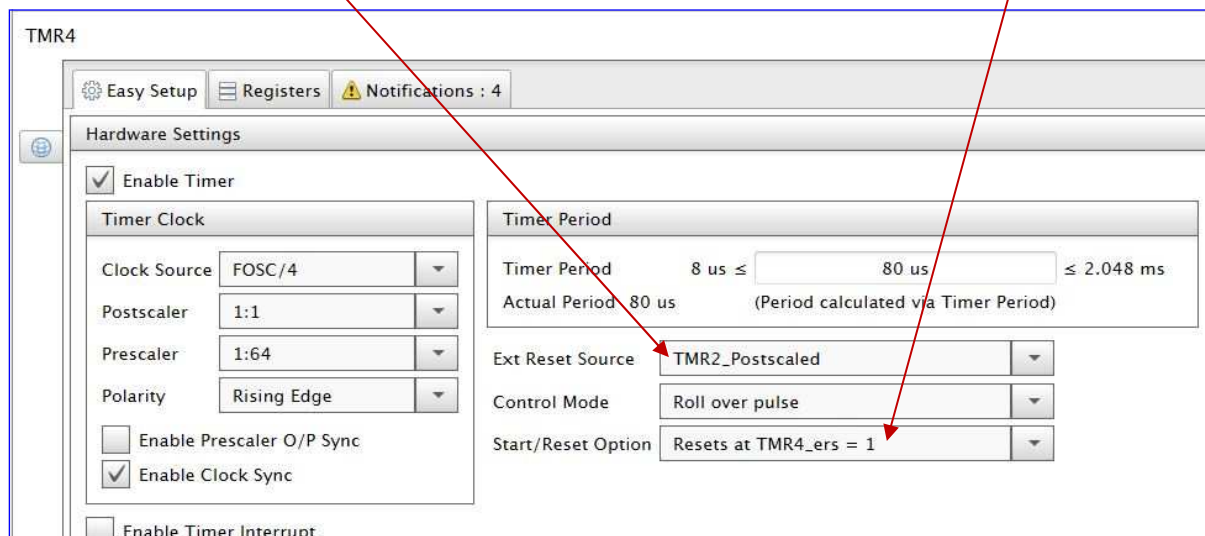
- 由於 ZCD1 的腳位輸入 ZCD1IN 是固定在 RA2 的腳位，但 ZCD1OUT 的腳位是可以經由 PPS 做設定的，在此實驗我們設定 ZCD1OUT pin 經 PPS 設在 RA5 (與 LED D4 同腳位)。
- 接下來將設定 HLT/TMR2。將 ZCD 的輸出當做是 TMR2 Reset 的輸入來源，也就一次 ZCD 的輸入將觸發 TMR2 一次並產生一次的 1.28mS 計時框。時脈仍是使用 $F_{osc}=32\text{MHz}$ 的最高速。透過 1:128 的 Prescaler 降頻後送進 TMR2。
- TMR2 的外部 Reset 來源為 ZCD1 Output，TMR2 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。



- 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， $1.28\text{mS}/16 = 80\mu\text{S}$ ，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

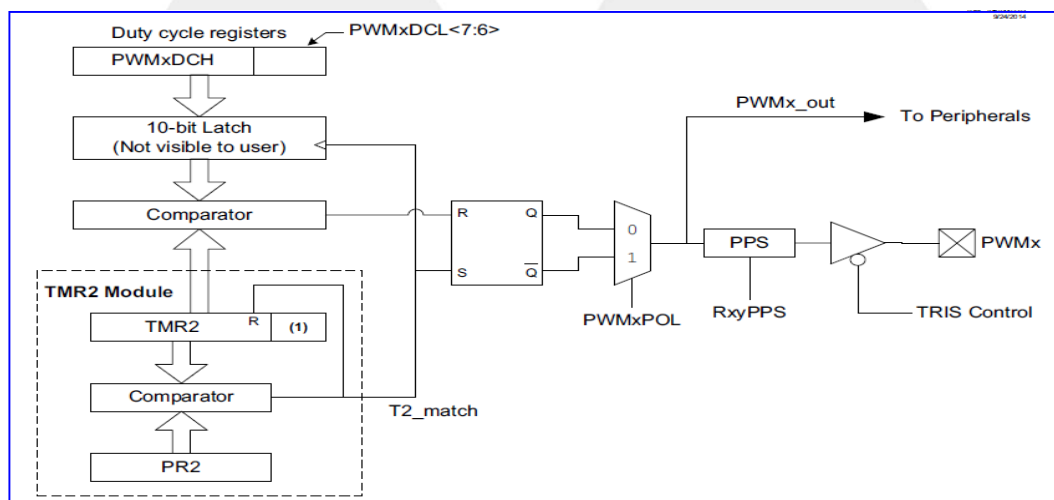
指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

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。



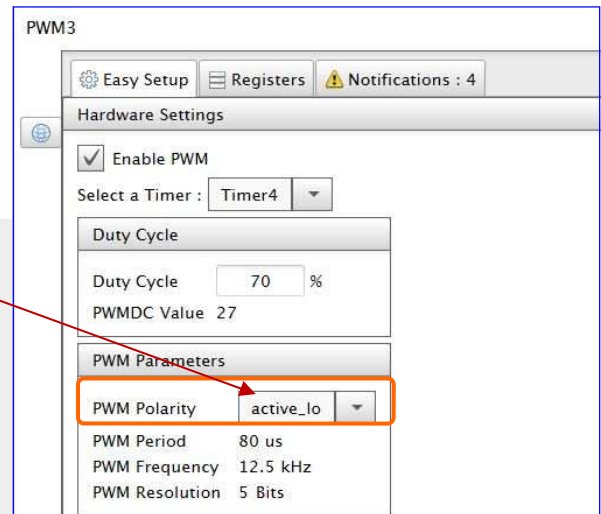
8. CIP 的 PWM 的模組設計為一專用的 PWM 控制，如下圖所示。週期由 Timer 2 /4/6 的 PRx (8-bit) 來對 Duty (10-bit) 做 1:4 的控制，Duty Cycle 的解析度可以有 10-bit。PWM 輸出的 Duty Cycle 如右圖公式所示。因為要在 1.28ms 輸出 16 個脈衝，所以 PWM 的週期為 80uS，也就是 TMR4 的週期要設定在 80uS。

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

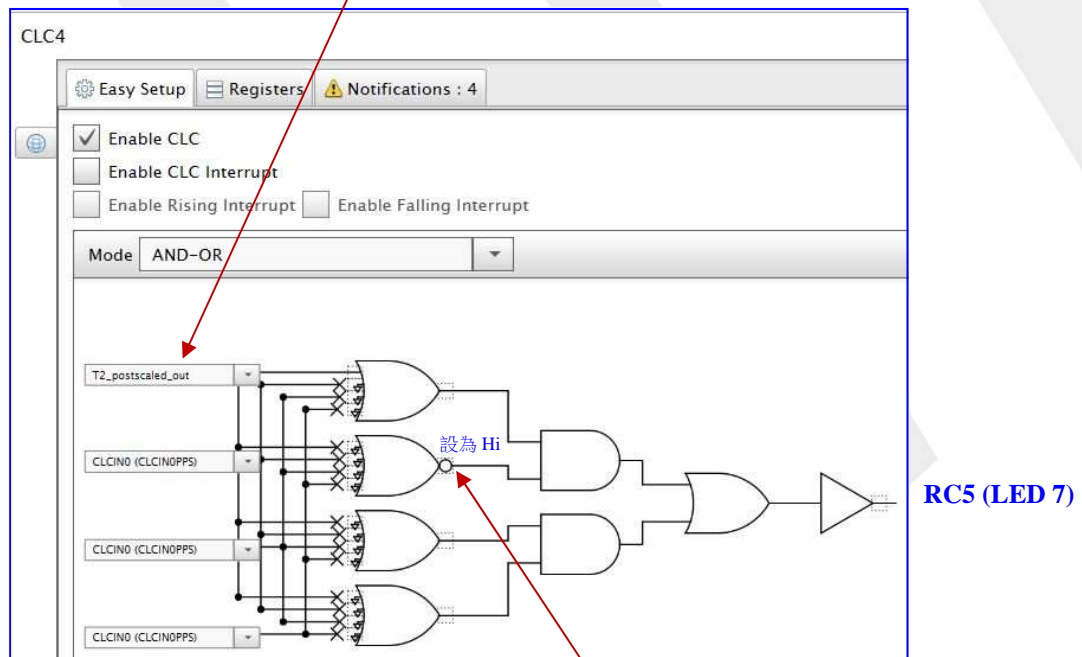


指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

右圖即為 PWM3 的設定，使用的計時週期是 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 CLC4 的輸入端，在這裡我們就可以利用 CLC4 將這 TMR2_Postscaled 的訊號送到 CLC4 指定的輸出腳位 (RC5) 之上以供量測。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。

指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

10. 最後來檢視一下輸出腳的設定。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。

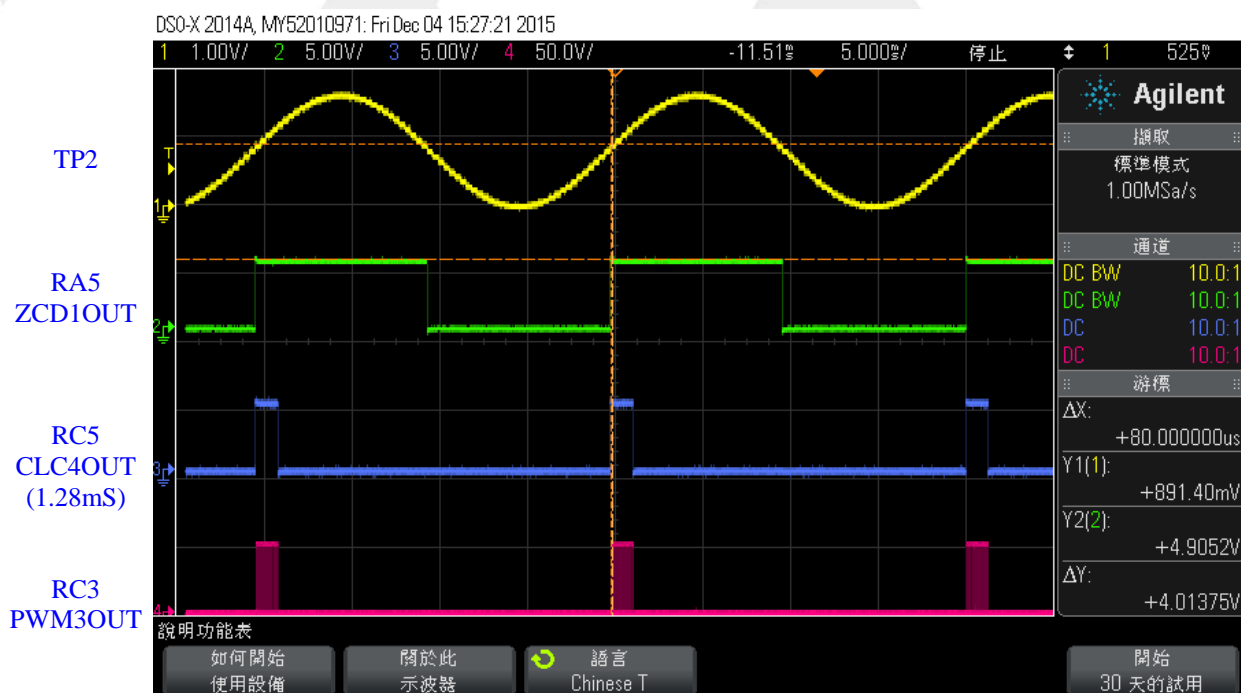
Pin N...	Module	Function	Custom Na...	Start Hi...	Analog	Output	WPU
RA2	ZCD	ZCDIN	RA2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
RA5	ZCD	ZCD10...	RA5	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
RC3	PWM3	PWM3...	RC3	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
RC5	CLC4	CLC4OUT	RC5	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
RC7	CLC4	CLCIN1	RC7	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
RC7	CLC4	CLCIN3	RC7	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
RC7	CLC4	CLCIN0	RC7	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
RC7	CLC4	CLCIN2	RC7	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
RC7	TMR2	T2IN	RC7	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
RC7	TMR4	T4IN	RC7	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Package:	PDIP20	Pin No:	19	18	17	4	3	2	13	12	11	10	16	15	14	7	6	5	8	9
Module	Function	Direction	0	1	2	3	4	5	4	5	6	7	0	1	2	3	4	5	6	7
CLC4	CLC4OUT	output																		
CLC4	CLCIN0	input																		
CLC4	CLCIN1	input																		
CLC4	CLCIN3	input																		
OSC	CLKIN	input																		
OSC	OSC2	input																		
PWM3	PWM3OUT	output																		
Pin Module	GPIO	input																		
Pin Module	GPIO	output																		
RESET	MCLR	input																		
TMR2	T2IN	input																		
TMR4	T4IN	input																		
ZCD	ZCD1OUT	output																		
ZCD	ZCDIN	input																		

注意: 有些週邊的腳位並沒有使用，在這裡統一透過 PPS 設到 RC7。因為未使用所以沒有問題。但這些未使用到的腳位在 MCC v2.25.2 卻不會顯示出來。

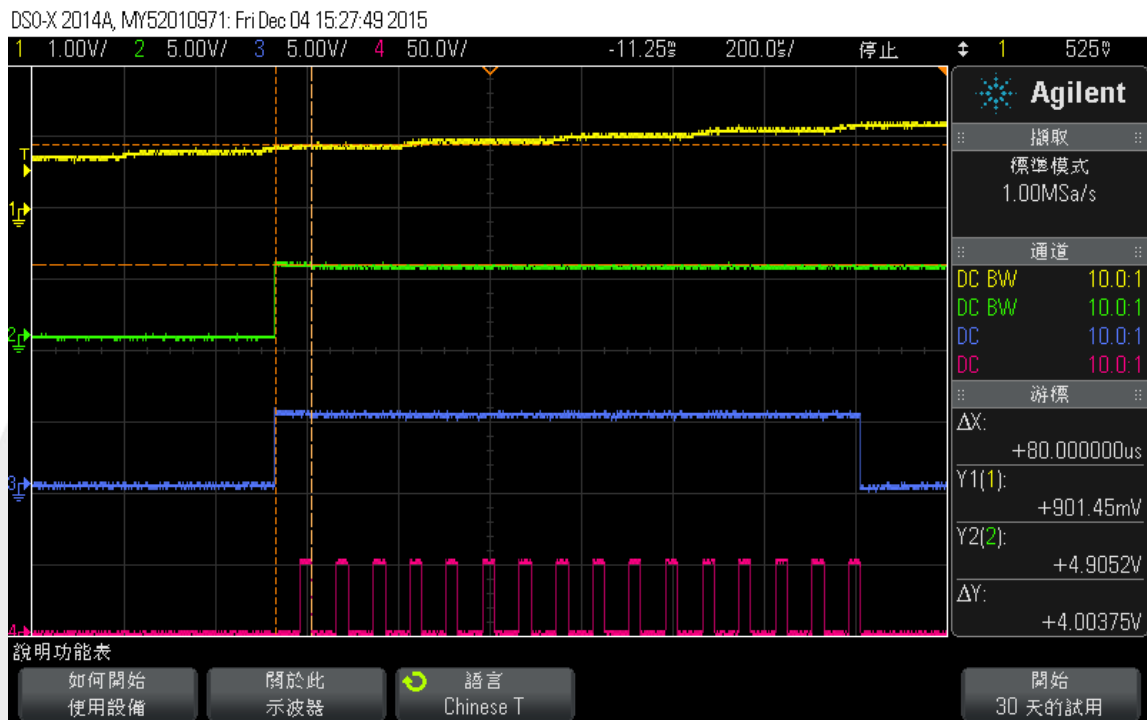
11. 最後在 MPLAB X IDE 下，先產生 MCC 的週邊函數後再編譯及燒錄。下圖為實際測量本實驗的結果:

1. 為子板上 TP2 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。



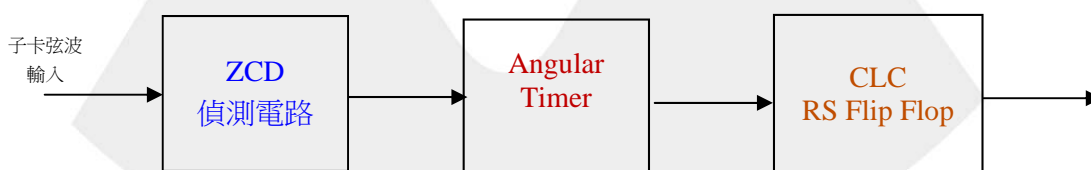
指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

將上面的波型展開放大後可以看到 (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° 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：

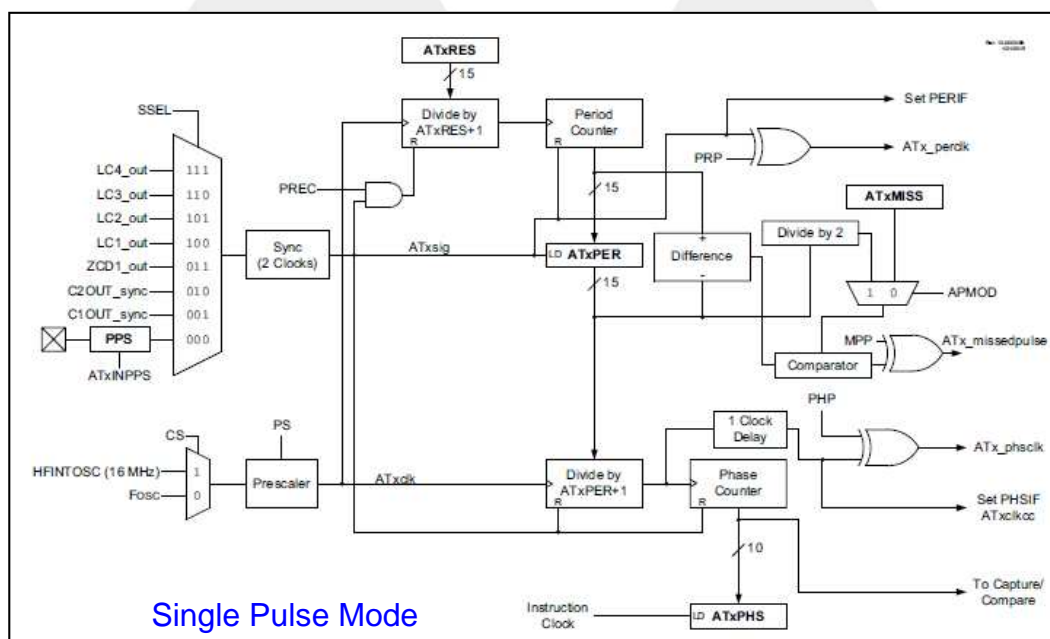
使用 MCC 來設定 Angular Timer 來做角度的偵測，實驗中將在 90 度時輸出 Hi 訊號及在 180 度時將輸出清除為 0。

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角度時的輸出。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，因為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。

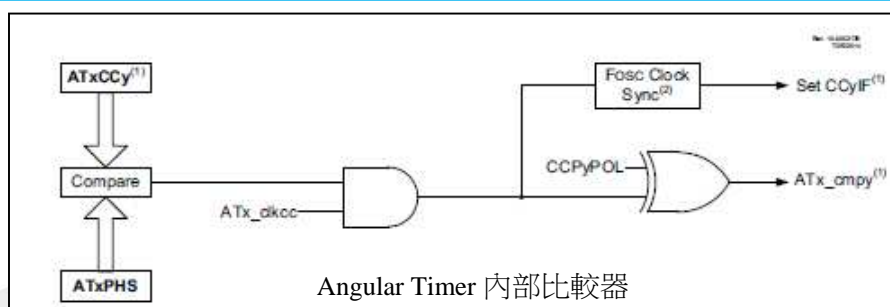
在本實驗 (Lab7) 是將 ZCD 的上升零點的偵測做為 Angular Timer 的量測訊號輸入，測量時所使用的計數時脈來自內部的 16MHz 震盪頻率。使用兩個相位(角度)比較器的輸出再去控制 CLC 的 RS 正反器的輸出來得到角度(相位)的偵測。

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。



指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

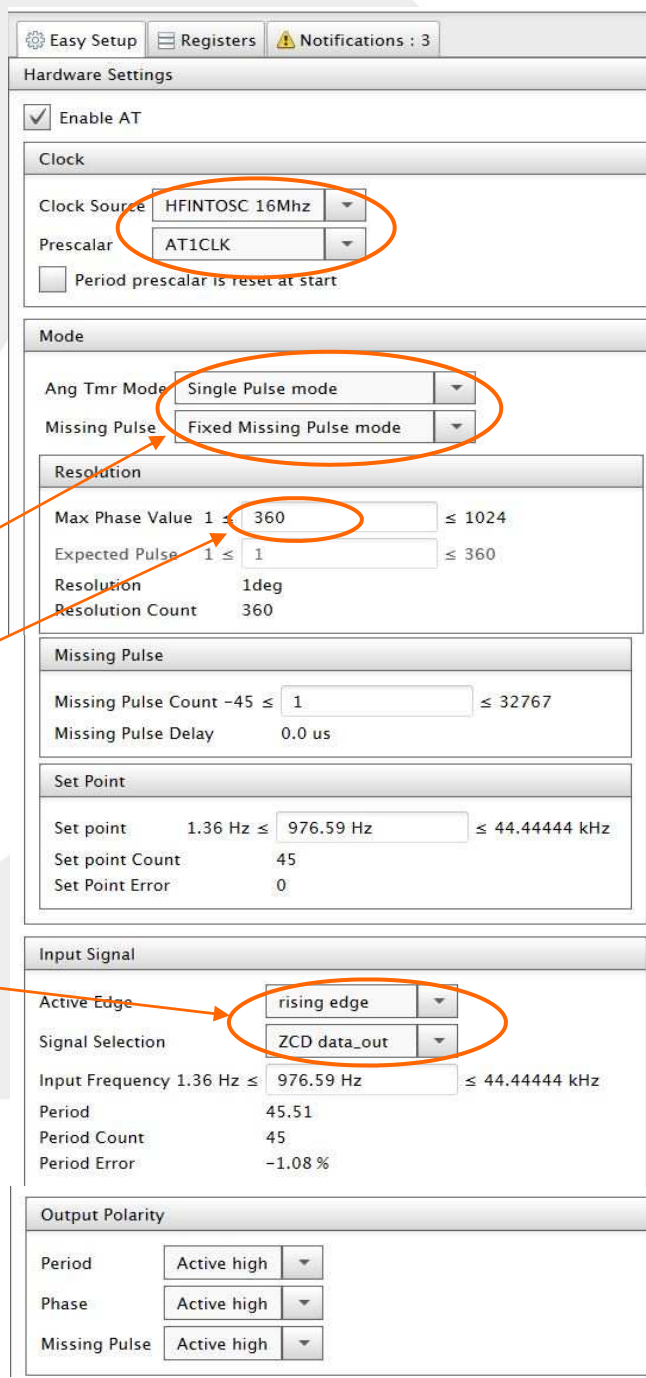
檢測角度 AT1CCY 的設定值與相位計數器 AT1PHS 相互比較後輸出到 AT1cmpy 或設定中斷輸出 CCyIF



Lab7 開始做實驗

1. 開啟 Lab7_Ang_TMR_ZCD.X
2. 啟用 MCC
3. 設定 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。(在這裡就不再貼 ZCD 的設定圖出來)
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 (如右圖所示)：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。

在做部分項目設定時，曾發生過去點選修改部分設定卻沒有反應的情形。如更改 Clock Source & Signal Selection for the ZCD data_out 時無法變更。這時請嘗試到 Registers 去設看看，再回到此畫面再更改看看。



指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

5. 有關一圈的相位角度解析，在上圖是設一圈 (ZCD 上緣到一下個 ZCD 上緣為一圈) 為 360 度，解析的角度為 1 度。如果需要更高的解析度可以如右圖所示: 設成一圈為 720 度，這時解析度就可達 0.5 度。

Resolution	
Max Phase Value	1 ≤ 720 ≤ 1024
Expected Pulse	1 ≤ 1 ≤ 720
Resolution	0.5deg
Resolution Count	720

6. 接下來要設定的是 AT 的比較器。在上頁 AT 的輸入訊號是 ZCD 的檢測方波輸出，其零角度的起點就是 ZCD 的上升緣直到下一個 ZCD 上升緣。在這裡我們使用比較模式 (Compare)，比較相符時的輸出動作為 Hi。因為本實驗需求是兩個角度的偵測所以要啟用兩個比較器。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。

ZCD 的輸入就會在 90 度時由第一比較器輸出 Low 到 Hi 的輸出。

ZCD 的輸入就會在 180 度時由第二比較器輸出 Low 到 Hi 的輸出。

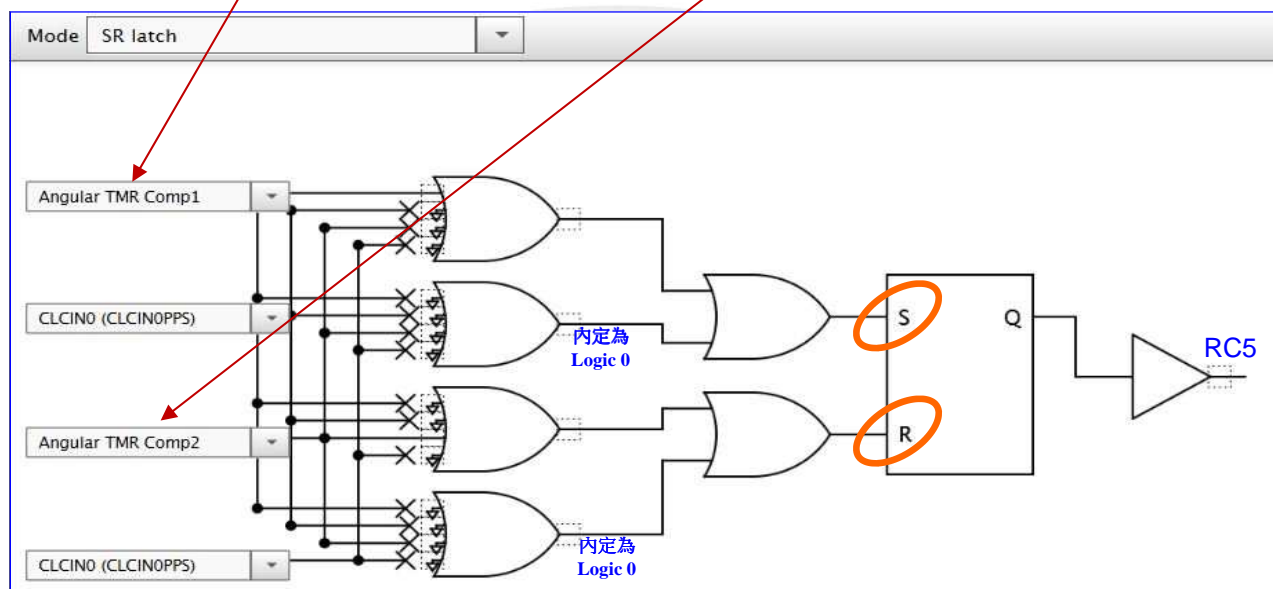
Capture/Compare1	Capture/Compare2	Capture/Compare3
<input checked="" type="checkbox"/> Enable Capture/Compare		
Mode Select: compare mode		
Output Polarity: Active high		
Input Polarity: rising edge		
Input Selection: AT1CC1 pin		
Compare Value 0 ≤ 90 ≤ 1023		

Capture/Compare1	Capture/Compare2	Capture/Compare3
<input checked="" type="checkbox"/> Enable Capture/Compare		
Mode Select: compare mode		
Output Polarity: Active high		
Input Polarity: rising edge		
Input Selection: AT1CC2 pin		
Compare Value 0 ≤ 180 ≤ 1023		

因為前面所設定的最大角度為 360 度，所以要注意在這裡比較器的比較角度值不可超過 360 度。

指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

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號接到 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。至於沒用到的腳位透過 PPS 通通設為 RC7。

Pin N...	Module	Function	Custom Na...	Start Hi...	Analog	Output	WPU
RA2	ZCD	ZCDIN	RA2		<input checked="" type="checkbox"/>		
RA5	ZCD	ZCD10...	RA5			<input checked="" type="checkbox"/>	
RC5	CLC1	CLC1OUT	RC5			<input checked="" type="checkbox"/>	
RC7	CLC1	CLCIN0	RC7				<input checked="" type="checkbox"/>
RC7	CLC1	CLCIN1	RC7				<input checked="" type="checkbox"/>
RC7	CLC1	CLCIN2	RC7				<input checked="" type="checkbox"/>
RC7	CLC1	CLCIN3	RC7				<input checked="" type="checkbox"/>
RC7	AT	ATCC1	RC7				<input checked="" type="checkbox"/>
RC7	AT	ATCC2	RC7				<input checked="" type="checkbox"/>
RC7	AT	ATCC3	RC7				<input checked="" type="checkbox"/>
RC7	AT	ATIN	RC7				<input checked="" type="checkbox"/>

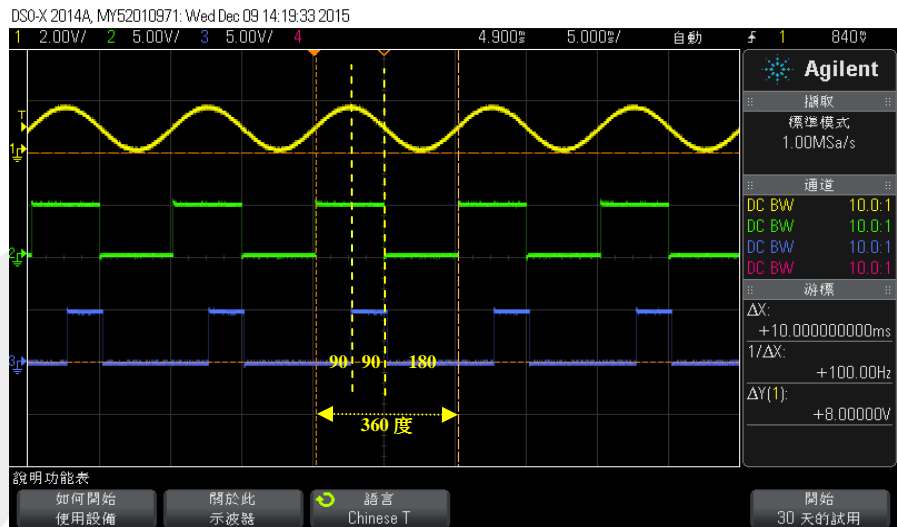
Package:	PDIP20	Pin No:	19	18	17	4	3	2	13	12	11	10	16	15	14	7	6	5	8	9
Module	Function	Direction	0	1	2	3	4	5	4	5	6	7	0	1	2	3	4	5	6	7
AT	ATCC1	input	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
AT	ATCC2	input	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
AT	ATCC3	input	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
AT	ATIN	input	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
CLC1	CLC1OUT	output	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
CLC1	CLCIN0	input	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
CLC1	CLCIN1	input	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
CLC1	CLCIN2	input	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
CLC1	CLCIN3	input	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
OSC	CLKIN	input						<input checked="" type="checkbox"/>												
OSC	CLKOUT	output					<input checked="" type="checkbox"/>													
OSC	OSC1	input					<input checked="" type="checkbox"/>													
OSC	OSC2	input					<input checked="" type="checkbox"/>													
Pin Module	GPIO	input	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Pin Module	GPIO	output	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
RESET	MCLR	input				<input checked="" type="checkbox"/>														
ZCD	ZCD1OUT	output	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
ZCD	ZCDIN	input		<input checked="" type="checkbox"/>																

8. 啟用 MCC Code Generator 產生 CIP 的函數後，直接編譯及燒錄，用示波器量測一下 **TP2**、**RA5**、**RC5** 的波形。

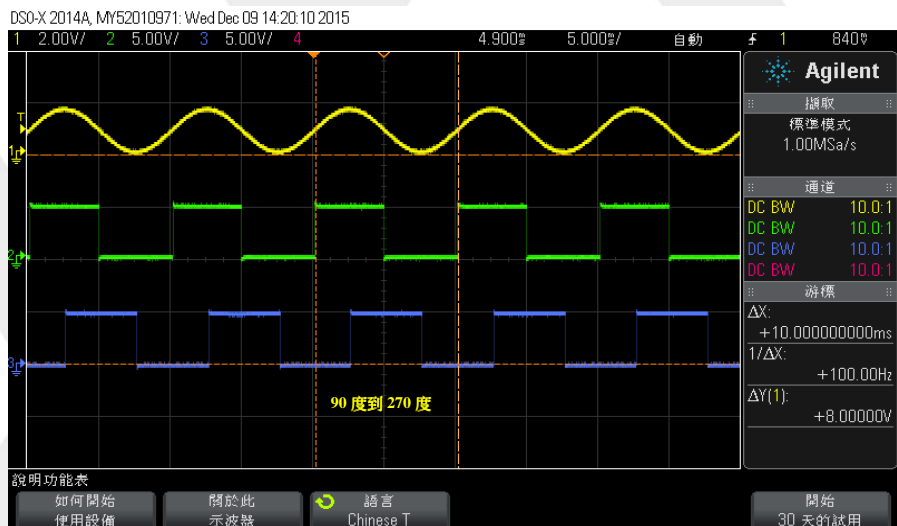
指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

右圖為設定 90 度及 180 度的輸出。

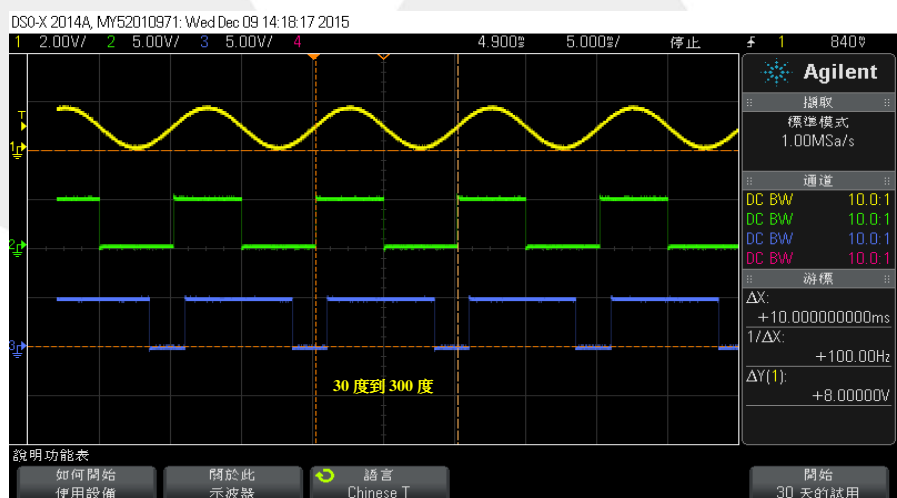
1. 為弦波輸入到 ZCD (TP2)
2. ZCD 的準位檢測輸出 (RA5)
3. CLC1OUT (RC5) 注意看一下與正弦波的角度變化點，是否落在弦波的 90 度及 180 度。



右圖為設定 90 度及 270 度的輸出。

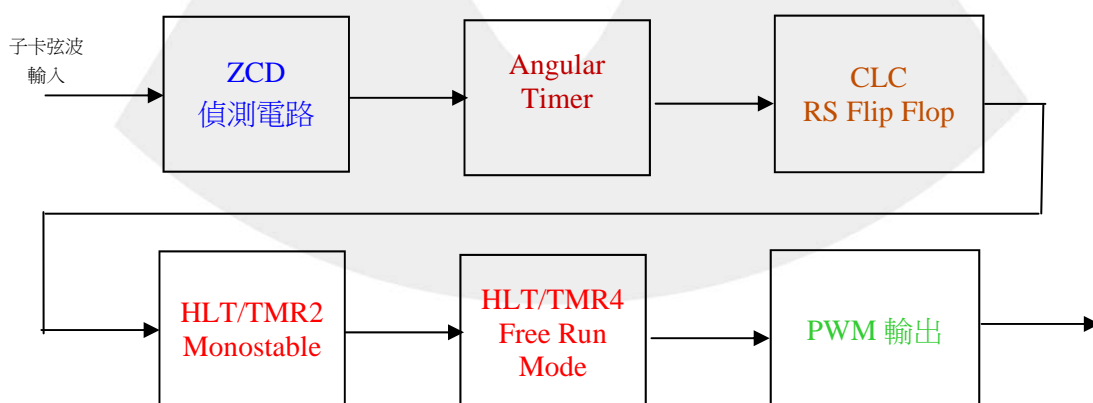


右圖為設定 30 度及 300 度的輸出。



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

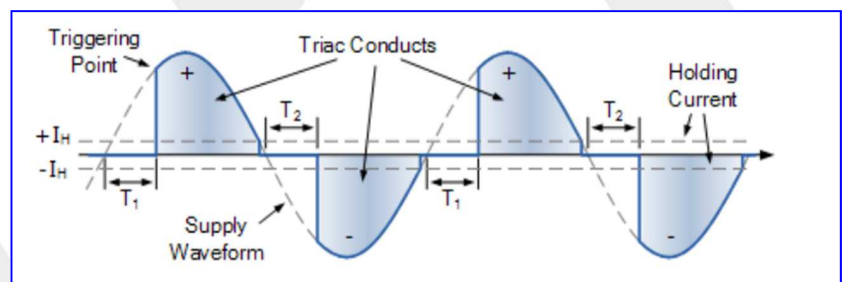
目標：

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，也要在負週期時觸發一次，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。

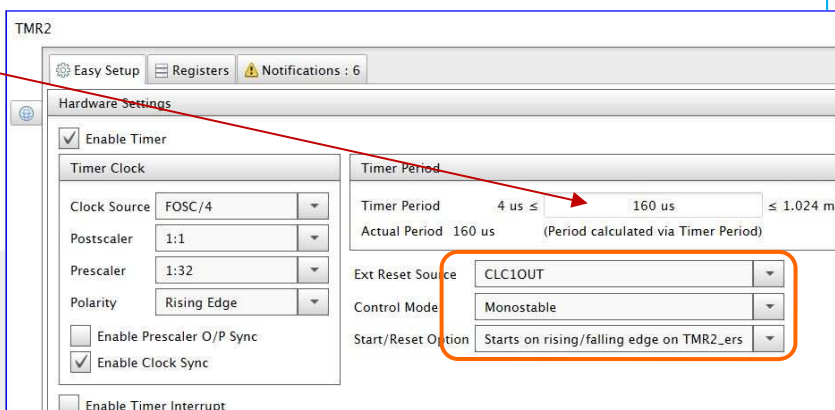
Lab8 開始做實驗

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)，解析度為 1 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

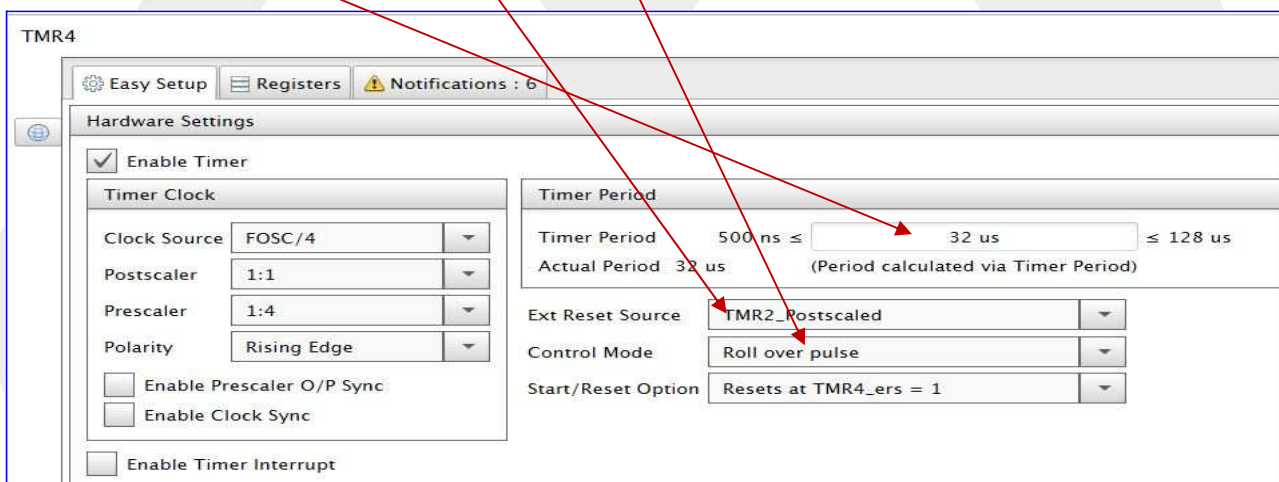
指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

框為 **160uS**。

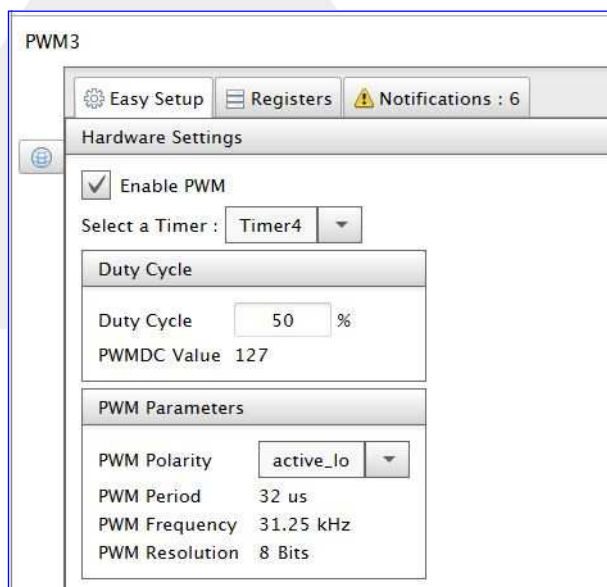
也就是說 AC 輸入的正弦波 0 ~ 180 度時會有一個延遲 30 度的觸發 TMR2 來產生的 160uS 的時間框；在負半週 180 ~ 360 度時也會有一個 210 度 160uS 的時間框。
(角度來自 AT 的比較器設定)



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，自動計時到 32uS 時自動歸零再重新計數，這總共會動作運轉 5 次後停止，因為這時候 160 uS 的時間框已到，TMR4 即會關閉。



8. 設定好了 TMR4 的 32uS 週期，接下來要設定 PWM3 模組。Duty Cycle 設為 50% 輸出，並將 **PWM 反相輸出**。因為 TMR4 在這 160uS 的時間框裡總共會計數五次，每次都會讓 PWM3 產生一個 PWM 的輸出。所以經此設定後我們可以得到五的 PWM 的輸出。



指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

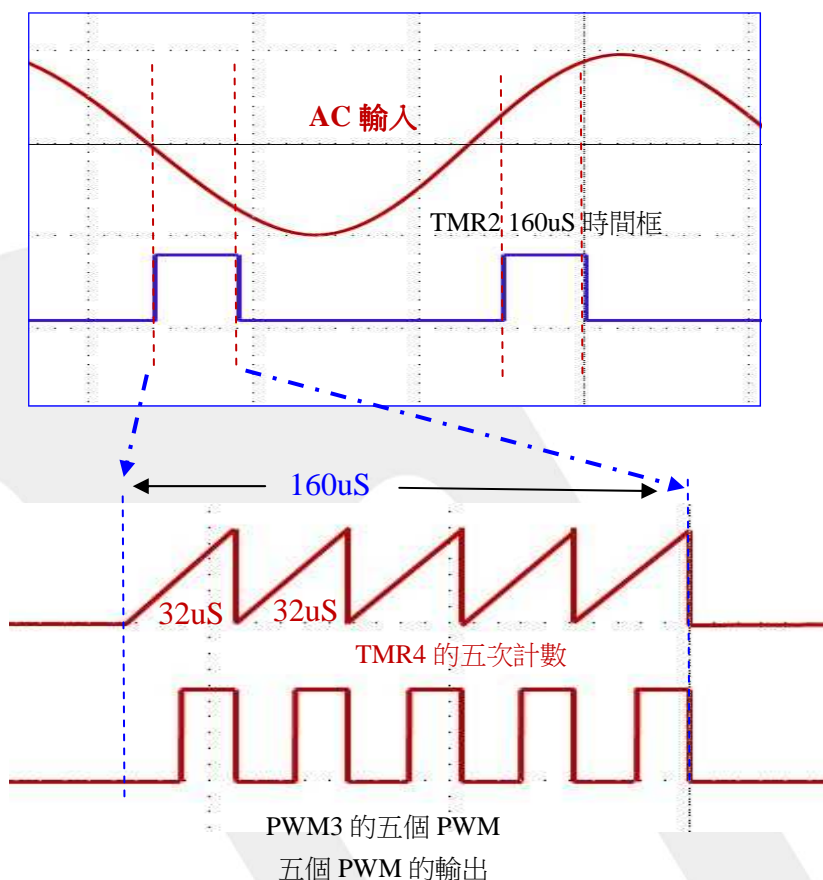
右圖為 AC 輸入經過 ZCD 偵測電路後經過 AT 的比較器後送進 CLC1 做角度觸發並輸出一方波。

此方波的上升/下降緣會再驅動 TMR2 來產生兩個 160uS 的時間框。

TMR2 的 160uS 時間框會再去驅動 TMR4 的計時，以 32uS 的計時單位做為 PWM3 的 PWM 50% 的 Duty 輸出。

- 最後確認一下腳位的設定。
在本實驗中自開始的 ZCD 輸入及 ZCD 輸出 (CLC1OUT 腳位是用來除錯測試用的) 這一連續的訊號連接都是在 IC 內部連接的，直到最終的 PWM3 的腳位輸出。這也就是 CIP 的內部透過暫存器的連線設定可以大大的減少外部的接腳，降低硬體的成本及尺寸，這可稱之為“麻雀雖小，五臟俱全”。

CLC1OUT (RC5) 可以清楚的觀測到延遲 30 度及 210 度時的 RS 的輸出訊號。PWM3OUT (RC3) 是送給 TRIAC 的五個連續 PWM 觸發訊號。



Pin N...	Module	Function	Custom Na...	Start Hi...	Analog	Output	WPU
RA2	ZCD	ZCDIN	RA2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
RA5	ZCD	ZCD10...	RA5	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
RC3	PWM3	PWM3...	RC3	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
RC5	CLC1	CLC1OUT	RC5	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
RC7	CLC1	CLCIN2	RC7	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
RC7	CLC1	CLCIN1	RC7	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Package:	PDIP20		Pin No:	19	18	17	4	3	2	13	12	11	10	16	15	14	7	6	5	8	9
				PORT A ▼					PORT B ▼					PORT C ▼							
Module	Function	Direction	0	1	2	3	4	5	4	5	6	7	0	1	2	3	4	5	6	7	
PWM3	PWM3OUT	output																			
Pin Module	GPIO	input																			
Pin Module	GPIO	output																			
RESET	MCLR	input																			
TMR2	T2IN	input																			
TMR4	T4IN	input																			
ZCD	ZCD1OUT	output																			
ZCD	ZCDIN	input																			

指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

實驗量測的輸出波形：

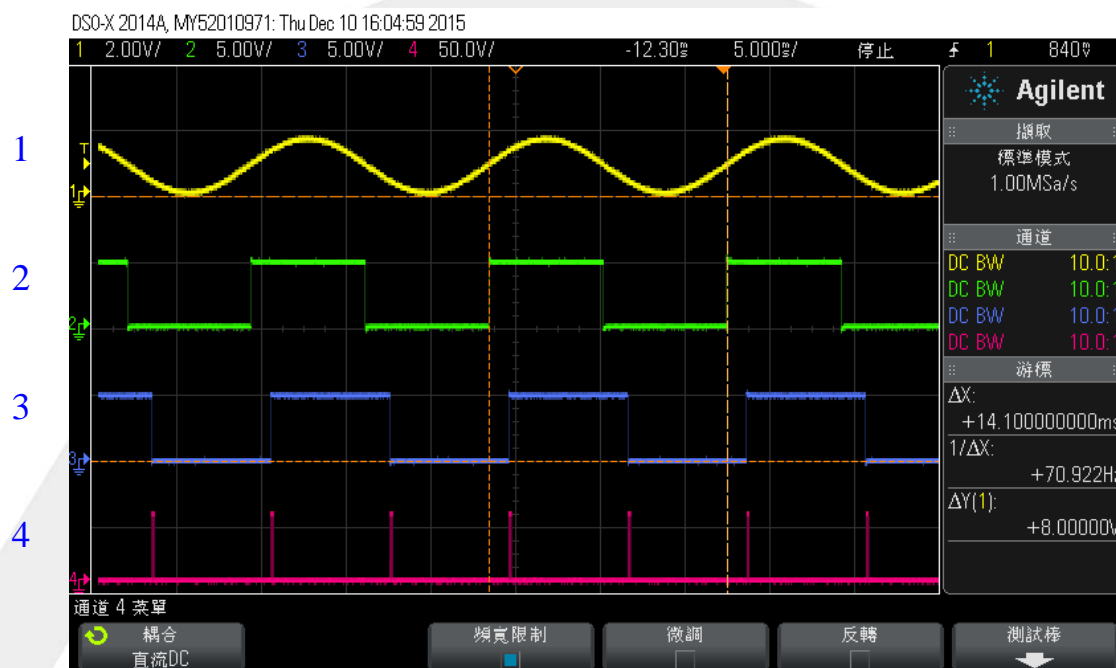
在下圖的示波器可看到 4 個波形：

1 為 TP2 輸入的正弦波。

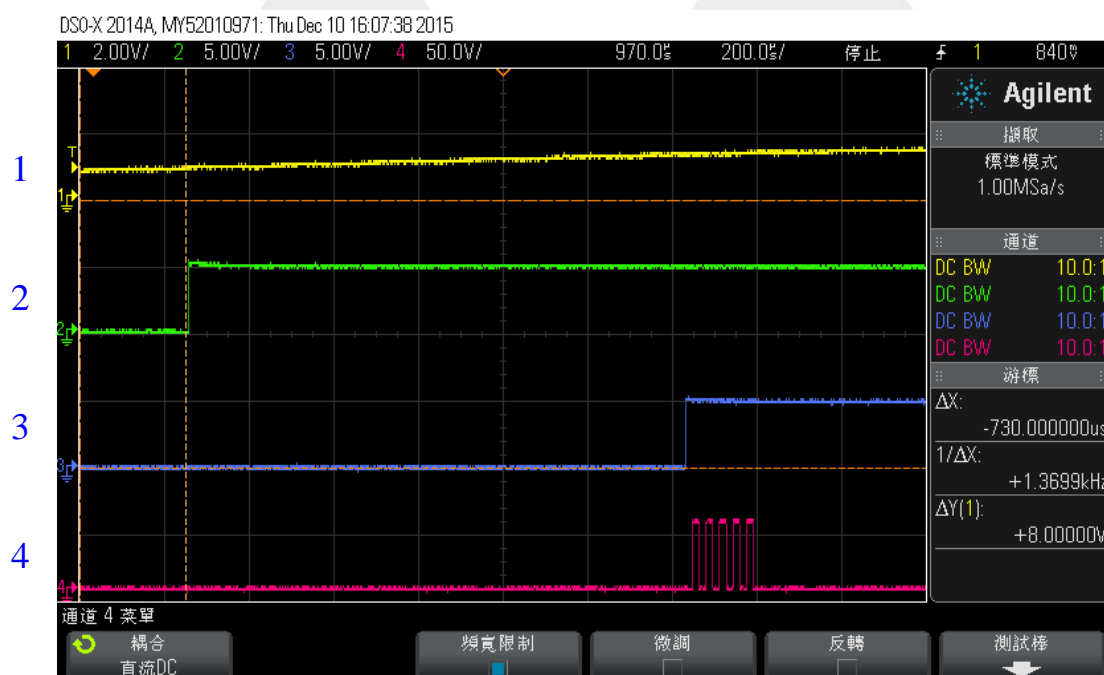
2 為 ZCD (RA5)的輸出。

3 為經過角度比較後的 CLC1OUT(RC5) 輸出。

4 為 PWM (RC3)觸發輸出

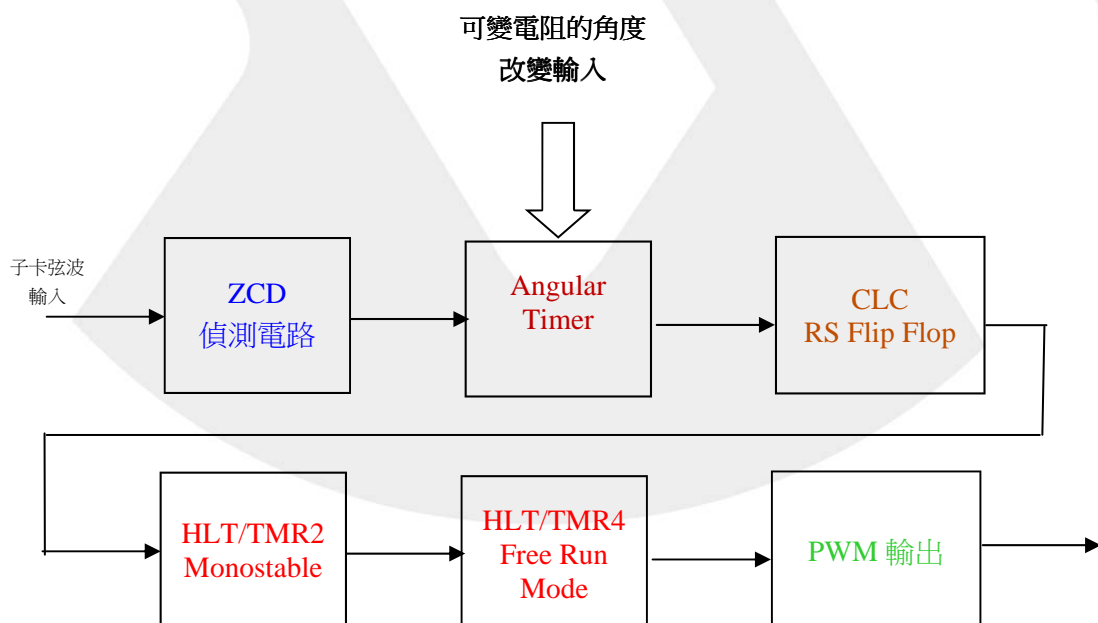


下圖為放大 PWM 的脈衝圖，圖中可明顯看到 5 個 Duty 50% 的脈波輸出。



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 的正負觸發可變角度的控制

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 ADC 的設定，由 Curiosity 板電路得知，可變電阻是接到 AN4/RC0 所以需先在腳位設定方塊裡設定 RC0 為 Analog Input 功能 (AN4)，綠色上鎖。
5. 在 Pin Module 的 ADC 設定項中使用內定的 “channel_AN4” 的名稱，該名稱會再被定義在 “pin_mamager.h” 裡以供程式撰寫者使用。

指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

Package:	PDIP20	<div><div></div></div>	Pin No:	19	18	17	4	3	2	13	12	11	10	16	15	14	7	6	5	8	9
				PORT A▼					PORT B▼				PORT C▼								
Module	Function	Direction	0	1	2	3	4	5	4	5	6	7	0	1	2	3	4	5	6	7	
ADC	ANx	input																			
ADC	VREF+	input																			

設定 RC0 為類比輸入功能

Easy Setup Notifications : 6							
Selected Package : PDIP20							
Pin N...	Module	Function	Custom Name	Start Hi...	Analog	Output	WPU
RA2	ZCD	ZCDIN	RA2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
RA5	ZCD	ZCD10...	RA5	<input type="checkbox"/>		<input checked="" type="checkbox"/>	<input type="checkbox"/>
RC0	ADC	AN4	channel_AN4	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
RC3	PWM3	PWM3...	RC3	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

ADC 使用預定的名稱。

6. ADC Module 的設定非常簡單，除了之前的 ADC 輸入腳的規劃 (RC0/AN4)，再下來就是 ADC 模組的主體。我們並沒有使用在最高的轉換速度，ADC Clock 選用內建的 RC 震盪器，1 Tad 的時間為 1.7uS，整個轉換時間為 19.55uS。

ADC 採向左靠齊方式，也就是說只取用 ADRESH 的 8-bit 結果，捨棄最低的 Lsb 2-Bits。因為控制角度只有 0~180 度之間，所以 ADRESH 的 8-bit 數值就足夠來控制了。

轉換觸發方式採用手動方式觸發轉換，這觸發動作是寫在在 TMR1 的 Callback() 裡。

參考電壓為 VDD
及轉換輸出向左靠齊

ADC

Easy Setup Registers Notifications : 6

Hardware Settings

☒ Enable ADC

ADC Clock

Clock Source

Frc

1 TAD

1.7 us

Sampling Frequency

51.1509 kHz

Conversion Time

= 11.5 * TAD = 19.55 us

Result Alignment:

left

Positive Reference

VDD

Auto-conversion Trigger

no_auto_trigger

☐ Enable ADC Interrupt

Selected Channels

Pin	Channel	Custom Name
Internal Channel	DAC	channel_DAC
Internal Channel	FVR	channel_FVR
Internal Channel	Temp	channel_Temp
RC0	AN4	channel_AN4

使用 channel_AN4 做為 AN4 的名稱定義

7. 最後加入 Timer1 的設定。目前 Lab 8 已經使用了 TMR2 及 TMR4 兩個 Timer 了。在此啟用 Timer1 做為 50mS 的計時器提供 ADC 做角度的更新。也就是說：點火的角度的更新是在 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。

指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

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

The screenshot shows the TMR1 configuration window. The 'Hardware Settings' tab is selected. In the 'Timer Clock' section, 'Clock Source' is 'FOSC/4', 'External Frequency' is '32.768 kHz', and 'Prescaler' is '1:8'. 'Enable Synchronization' is checked. In the 'Timer Period' section, 'Timer Period' is '50 ms' and 'Calculated Period' is '50 ms'. In the 'Enable Gate' section, 'Enable Gate Toggle' and 'Enable Gate Single-Pulse mode' are unchecked. 'Gate Signal Source' is 'T1G' and 'Gate Polarity' is 'low'. In the 'Enable Timer Interrupt' section, 'Enable Timer Interrupt' is checked and 'Enable Timer Gate Interrupt' is unchecked. In the 'Software Settings' section, 'Callback Function Rate' is '1', resulting in 'x Time Period = 50 ms'.

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 ADC 的開啟及固定要轉換 Channel 的輸入

```
void main(void) {
// initialize the device
SYSTEM_Initialize();      // 執行 MCC 所產生的週邊函數初始設定
```

```
ADCON0bits.ADON = 1;      // 啟用 ADC
```

```
ADCON0bits.CHS = channel_AN4; // 設定輸入的 Channel 為 AN4/RC5 的輸入
```

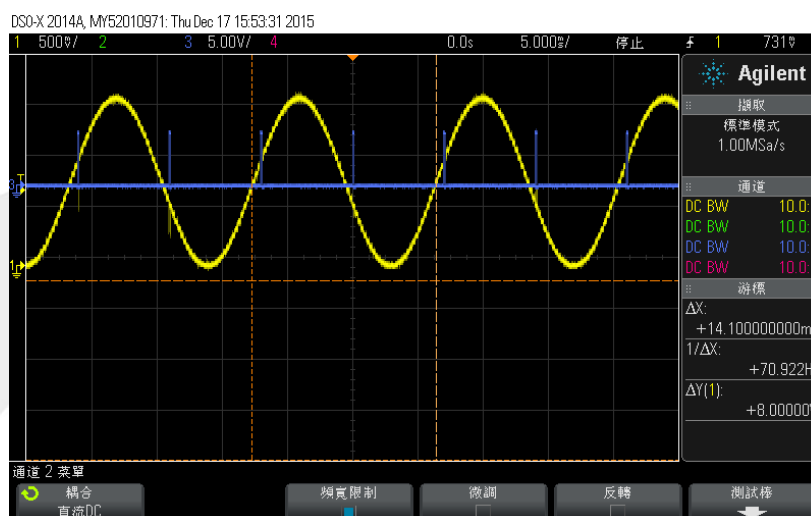
9. Timer1 的 Callback (tmr1.c) 的說明：

```
void TMR1_CallBack(void)
{
    ADCON0bits.GO = 1;          // AD 開始轉換
    While (ADCON0bits.GO);      // 轉換完成？
    If (ADRESH < 10) 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 (正半周)
                                // 比較器二的控制(負半周)
    AT1CC2L = (unsigned char)(ADRESH + 180); // 負半週角度控制在 190 ~ 350 度
    AT1CC2H = (unsigned char)((ADRESH + 180) >> 8);
}
```

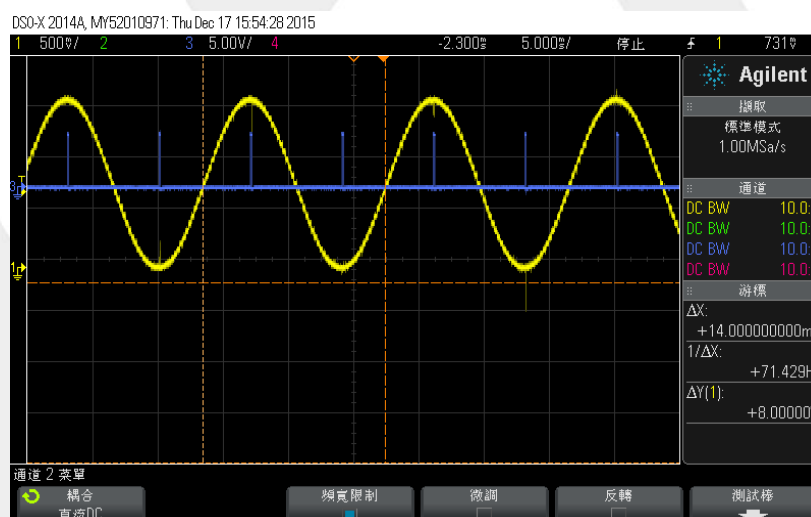
指導手冊 - 使用酷奇板 (Curiosity) 實作 MCC/CIP 練習

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