MICROCHIP

Regional Training Centers

使用 MCC mTouch 的設定，快速完成觸控按鍵實作指導手冊

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

實驗手冊所使用到的提示圖例


目的: 這個實驗的目的


需求: 這個實驗需要那些工具


目標: 做什麼及期待什麼


過程: 要完成目標的步驟列表


警告：採取行動避開陷阱


訊息：有關該步驟的其它訊息


提示：您需要在該步驟中使用的函數或變數

第一章: mTouch 概述

第二章: APP046 mTouch 觸控電路板

第三章: mTouch Library 快速上手 (Lab1)

第一章: mTouch 概述

1.0 MCC 簡介

MCC 是 MPLAB X IDE 的外掛式模組(Plug-In) 。在專案下，只要啟用 mTouch 電容式觸控函數庫模組就可快速且輕鬆的設定觸控按鍵及接近感應所需的參數後，就可自動產生 C 語言的觸控函數庫的原始程式。在此之前如果要使用 PIC 做觸控按鍵最快的方式是使用 Microchip 所提供的 mTouch Library ，使用這 Library 主要的瓶頸是在於觸控參數的調整，及外界干擾的抑制。現在有了 MCC 的 mTouch 的設定，只要將各項需求與參數輸入即可快速完成觸控按鍵的功能。有了 MCC 的 mTouch 功能，在 PIC 的應用中加入觸控將是一件輕而易舉的工作。

mTouch 周邊的設定是使用 GUI (圖形化設定介面) 來完成以下的內容:

- 啟用各種 mTouch 觸控按鍵或接近感應功能
- 設定觸控按鍵各項參數

1.1 Microchip 的 1D 觸控介紹

觸控的方法有很多種，最具成本效益的方式幾乎都是採用電容變化為主的架構。在硬體的設計中普遍都會有雜散電容 (寄生電容) 的存在且這些電容量的變化都不大或呈現一個固定的電容值；就人體本身而言也是個電容器，藉由這個人體電容在距離及形狀上的改變，也就會造成整個電容量的改變。電容量的改變又會造成那些變化呢? 依照各種架構來看，這電容量的變化可以造成一個非穩態震盪器的頻率的改變，也可以造成電容並聯效應後的電壓改變，還有充、放電時間的改變等常見到物理現象。所以在頻率、電壓、時間上的改變所延伸出來觸控架構，在 Microchip 也做了些的分類:

- CSM : 偵測頻率的變化
- CTMU、QTouch： 充、放電時間的測量
- 軟體 CVD、HCVD、ADCC ：充、放電壓的測量

CSM (Capacitive Sensing Module) 的架構是利用非穩態震盪器的頻率輸出，測量在觸控時單位時間的頻率變化。優點是程式撰寫簡單，但容易受外界干擾是最大的隱憂。Microchip 已不再支援 CSM 架構的觸控，所以就此不再此討論。

CTMU (Charge Time Measurement Unit) 是使用定電流來對電容 (含內、外部)來充電，在一定的時間時停止充電，然後用 ADC 量測電容上的電壓 或 以定電流方式來對電容充電到一定的準位時，使用硬體的 Timer 測量其所需的時間。CTMU 可以測量到 nS (奈秒）的時間，也常應用在超音波流量計的應用。

剩下的是 CVD 的架構，也是本實驗手冊所要討論的應用主題。底下就這三種 CVD 觸控感測主要的比較介紹:

- 軟體 CVD　(所有內建 10-bit ADC 的 PIC 都可以用)
  - 優化取樣時間，縮短處理觸控按鍵處理時間，降低 RAM 使用可以在較小顆的 PIC 上實現觸控的需求。
  - 在 MCC 用使用 AFA (跳頻機制) 自動啟用 Timer2 及 Timer2 的中斷
- 硬體 CVD（HCVD）(元件請參考 Page9 的表 1-0)
  - 優化取樣時間，縮短處理觸控按鍵處理時間，降低 RAM 使用可以在較小顆的 PIC 實現觸控的需求。
  - 自動校準功能可使用內部 Cap 和採集時間
- ADCC (ADC 具累計功能) (元件請參考 Page9 的表 1-1)
  - 自動校準功能可彈性使用內部取樣電容和取樣時間
  - 自主性無阻擋 CVD 取樣
  - ADC 轉換後的數值可自動做累積、取平均、突發平均及低通濾波
  - 在 MCC 介面的設定中自動啟用 ADCC 的 ADTI 中斷功能 (ADTI，ADC 閾值比較中斷)

目前 mTouch 的的設定是依照所使用的元件設定去做自動元件偵測方式來選用 mTouch 的函數庫。底下將會以 CVD 為主軸來闡述其原理及使用。

1.2 CDV 的原理說明

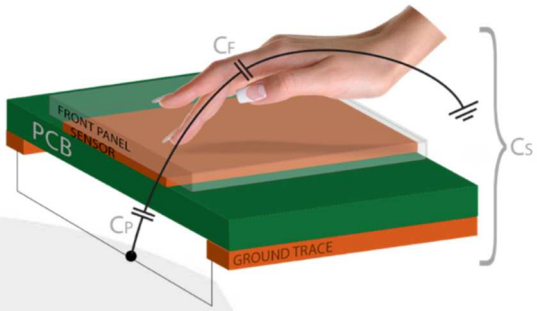
CVD (Capacitive Voltage Divider)　就字面上翻譯是"電容分壓器"。電容值代表水桶的底面積，電容大則底面積大，反之則底面積小；電荷是水桶內的水；電壓即是水位高度。大、小水桶底部使用水管連通，水流相互流動後，最終水位(電壓)就會平衡。水位 (電壓)的高低和底面積(電容)有關。例如: 大水桶是 (ADC 內建的取樣電容）平常先充電到滿水位 (充飽到達 Vdd 的準位)，小水桶則是（外界的寄生電容＋人體電容）平常都把水放乾 (即接地)。這時將大水桶與小水桶的連通管開啟，大水桶的水(電荷) 經連通管流向小水桶來取得一個平衡後，測量大水桶平衡後的水位(電壓)。

因為當人觸摸到接點時因人體的雜散電容加入使得小水桶變大，大水桶的水位就降的比較多。所以正確量測 ADC 取樣電容上的平衡電壓就可以知道按鍵是不是有被觸摸著。

此方法需要配備內部類比至數位轉換器(ADC)的微控制器，目前大多數 PIC 微控制器均符合此項要求。此方法使用 ADC 的內部取樣電容(C_{HOLD})以及電路板上感應點的電容來產生的電容分壓效應 (大、小水桶同水位高度)，分壓的高低則視感測器上的感應電容量而定。先使用 ADC 測量此分壓，再由軟體做 ADC 數值上的平均後的差異變化再進行觸控的判斷。

底下就軟體 CVD 的架構來說明其觸控的原理:

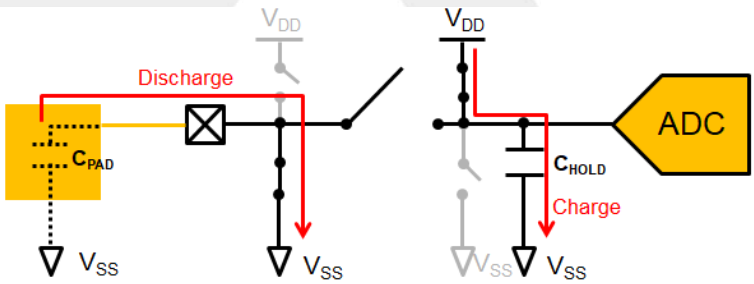
右圖為一標準的觸控點的電容分布簡易圖式。原本 PCB 的走線及觸控點本身就有寄生電容的存在；人體本身也有寄生電容的存在。當手指靠近或碰觸到觸控感應點(Touch Pad) 時，這時 Cp 與 C_F 就會因電容並聯電容量相加 Cs 就會變大。反之，觸控點沒有被觸摸也就沒有外在的影響會去改變其寄生電容量。CVD 的內涵就是測量這單位時間內電容上的電壓變化來判斷是否有觸摸或接近感應點。



CP: *觸控點的寄生電容*
CF: *手指的電容*
Cs: 觸控點的電容總和 (並聯效應)

步驟一: 由右圖可以很清楚的看出，取樣電容 (C_{HOLD}) 透過內部的設定將 C_{HOLD} 接到 VDD 將其充電到 VDD。

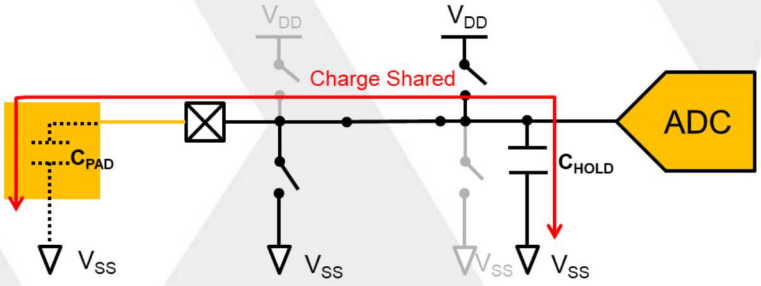
而外部的觸控點則是透過 I/O 腳接地將寄生電容的電荷先放光。



步驟一 ： 正向的預充電準備

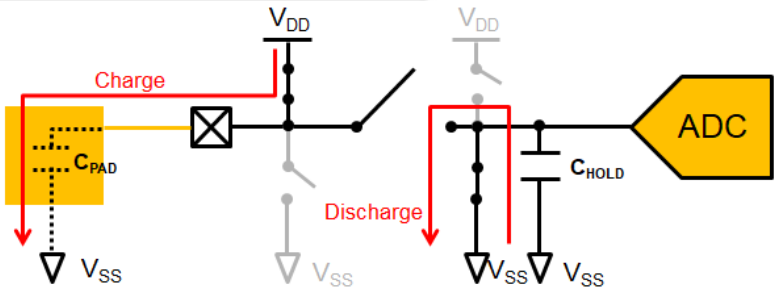
步驟二: 關閉外部觸控點的接地使其浮接，再來取樣電容 (C_{HOLD}) 透過 ADC 通道的設定直接連接到外部的觸控點，這時 C_{HOLD} 的電壓(荷)就會往外流向觸控點的寄生電容。因為外部的寄生容量很小，很快的這放電就會取得一個電荷的平衡電壓點。

啟動 ADC 做轉換就可以得到這時觸控按鍵的正向充放電的數值。



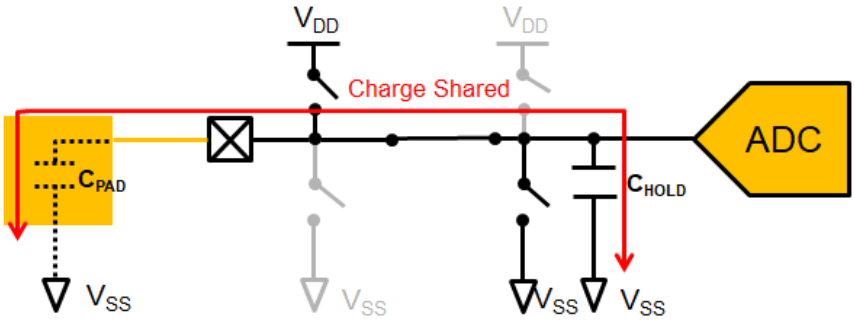
步驟二 ： CHOLD 開始放電取得電位的平衡並轉換

步驟三: 其實經由步驟一及步驟二的處理我們已經可以分辨出觸控按鍵的動作。但為加強按鍵對雜訊的抗干擾能力，在步驟三裡做了一次負向的充放電的動作。也就是將外面的寄生電容先充飽到 Vdd，C_{HOLD} 則透過內部的設定連接到地將電荷先放光。



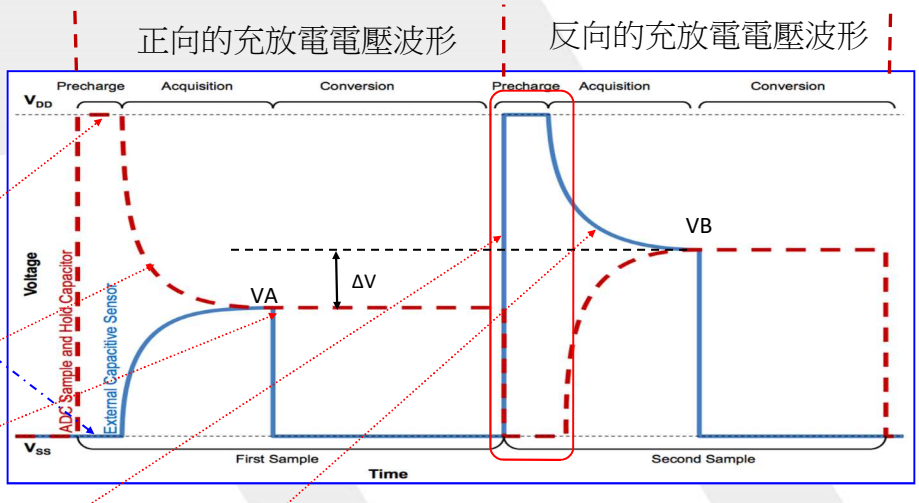
步驟三 ： CHOLD 負向充、放電的切換

步驟四: 當觸摸點的寄生電容 (C_{PAD}) 充飽電荷之後經由內部開關的切換連接到 C_{HOLD} 時，電荷將取的平衡。這時啟動 ADC 做轉換以取的反向平衡電壓數值。



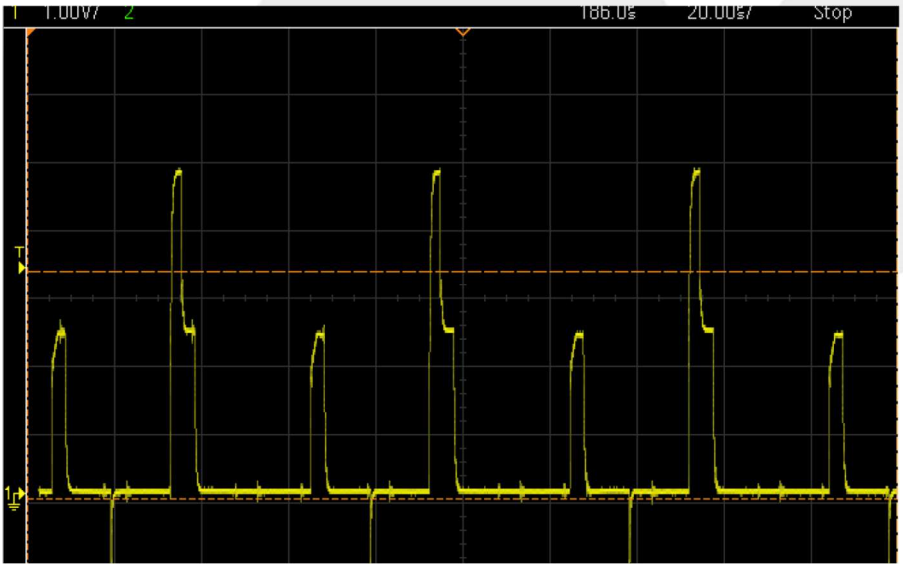
右圖很明顯地畫出正、負兩種充放電的電壓變化。

1. C_{HOLD} (紅色)充電到 VDD ，C_{PAD} (藍色) 放電到 Vss 階段。
2. C_{HOLD} 向 C_{PAD} 放電的階段
3. 大、小水庫取的平衡的階段，ADC 開始轉換。
4. 負向預充電開始，C_{PAD} 充電到 VDD，C_{HOLD} 接地將電荷放盡到 0 V。
5. C_{PAD} 向 C_{HOLD} 放電取的電荷上的平衡後進行 ADC 的轉換。



正向及負向 C_{PAD} 及 C_{HOLD} 的電壓變化

由於無法使用示波器測量內部取樣電容 C_{HOLD} 的電荷變化，在下圖我們用示波器紀錄了在 C_{PAD} 上在做軟體 CVD 轉換時的電壓變化。這波形與上圖藍色部分的波形吻合。當然這只是基本做 CVD 的波形顯示，但實際為避免外界雜訊的干擾我們還會再加入 Guard 的防護電場來加強觸控點 C_{PAD} 的抗雜訊。



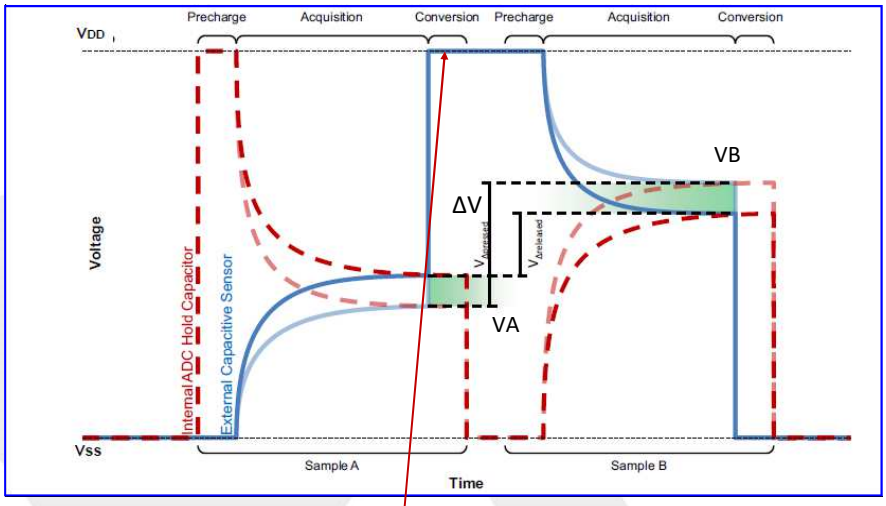
實際測量 Pad 的波形

關於 Guard 的原理及最重要的 PCB Layout 將會在後面的章節裡提到，這裡我們只是要先了解 CVD 動作原理。當然單一點取樣所得到的 CVD 轉換值是不穩定的。這些所取的到的數值是要經過計算及取平均的(濾波），這些 CVD 的軟體函數都會依我們的設定需求去完成這些複雜的演算法。

如何判斷按鍵有被觸摸呢? 在 CVD 的波形圖示裡，VB 點的轉換電壓值及 VA 轉換點的電壓值之間的差異電壓ΔV 就可以判斷出是否有按鍵。

CVD 公式 : ΔV = VB—VA

右圖所顯示的是有按鍵及沒有按鍵時的波形，同時也可以看出其ΔV 電壓的差異。藉由此種判斷觸控按鍵是否按下，這樣的偵測方式可以得到更大的差值，大大的提高了觸控的穩定。



由右圖也可以看出，為了更有效率緣故，在 ADC 對內部的取樣電容(CHold)做轉換時，軟體立即切換至 Pad 的寄生電容開始做預充電的動作，而不像上一頁的波形 Pad 先放電再充電到 Vdd。如此預充電的做法可加快觸控按鍵掃描的時間而不影響其精確度。

CVD 計算公式 = V Δpress—V ΔRelease

以上就是最簡單的 CVD 原理說明。看起來似乎只要有 ADC 就可以做 CVD 的量測。其實這也要看微控制器是否有強大的腳位設定功能可以使內部 ADC 取樣電容及外部的寄生電容充電到 Vdd 及放電到 Vss 的地電位。因 PIC 接腳使用上有很大的彈性設定，只要是有內建 10-bit ADC 的 PIC 不管是 8、16 或 32-bit 的 PIC 都可以實現軟體 CVD 的功能。

1-3 當然除了我們所講的以 MCU 為基礎的觸控外，Microchip 還有多種在 1D 的觸控元件與完整的觸控解決方案，底下將做個介紹：

- mTouch 的架構: 主要是以 PIC 的周邊配合軟體所設計出來的觸控，有前面介紹的 CVD、HCVD，CTMU 及 ADCC 等架構所建構出來的應用。
- Right Touch : ASIC 單元件觸控解決方案。這是最簡單的觸控應用，單顆元件透過 I2C 的連線即可。其最大的優點是擁有優越的抗雜訊能力及防水特性，很容易就可以將觸控功能加到原有的應用中。主要的元件編號有三大類： CAP1xxx、MTCH10x 及 AT42QTxxx。
- QTouch 的架構：以 Atmel 的 SAM (Cortex ARM) 及 AVR MCU 內建 PTC (Peripheral Touch Controller) 周邊的元件為主要支援的元件，配合 QTouch Library 來建構出一優秀且防水的觸控按鍵解決方案。

在本章節裡，我們只討論 mTouch 下使用 ：軟體 CVD，HCVD 及 ADCC 的架構來開發觸控按鍵的方法。至於 Right Touch Solutions 將不在這裡討論。針對 QTouch，我們會再有出版一篇專門探討 QTouch 的應用設計。

PIC12LF1552	PIC16LF1554/59	PIC16LF1566/67	PIC16F1512/13

表 1-0 ：目前有支援 HCVD 的元件，腳位從 8-pin (PIC12LF1552) 到 40-pin (PIC16F1567)

PIC16F1844x	PIC16F1885x	PIC16F1887x		
PIC16F1915x	PIC16F1917x	PIC16F1918x	PIC16F1919x	
PIC18FxxK22	PIC18FxxK40	PIC18FxxK42	PIC18FxxK83	PIC18FxxQ10

表 1-1 : 目前 8-bit 裡有支援 ADCC 功能的元件

如要更詳細的查詢可以使用 Microchip Advanced Part Selector 的工具來篩選所需功能的元件，MAPS 網站的連結如下: http://www.microchip.com/maps/

如果你所使用的元件如果沒有 HCVD 或 ADCC 的周邊，也可以使用軟體 CVD (SCVD) 的方式取完成觸控功能。只要是 PIC 具有 10-bit ADC 周邊都可以透過 MCC 裡的 mTouch 來設定就可以觸控的功能了。

本實驗手冊將使用三個不同的 8-bit PIC 來驗證觸控的應用設計:

SCVD : 使用 PIC16F15355-I/SP (8KW Flash, 28-pin)

HCVD : 使用 PIC16F1513-I/SP (4KW Flash, 28-pin) (Options)

ADCC : 使用 PIC16F18855-I/SP (8KW Flash, 28-pin)

1.3 Guard 的原理與防護驅動

一般在設計觸控感應點(Pad)的大小應該是一樣的，平均是用戶的手指按下（15x15 毫米）的尺寸。當然依照前面所講述的 CVD 方法來做，這觸控按鍵功能是可以達到的。但這只是一個基本而已，然實際去設計觸控按鍵還有很多事要考慮的，例如：觸控點之間相互的串擾 (Cross Talk) 現象、防潑水（Water Resistance）、外部的雜訊干擾及 ESD 防護等都要考慮的。

PAD-PAD 之間的串擾

串擾可能成為電容式觸控的挑戰，如果覆蓋層太厚或觸控感應器太多且緊密地放在一起這時候串擾現象就容易造成誤動作。且電場線因串擾因素造成無法集中因而降低的靈敏度。如果串擾去影響了一個應用程序中處理的問題，軟體必須比較兩個觸控感應器並確定那個觸控感應器被 "更多按下"，這增加了一個額外的步驟到解碼程序裡，這也增加可能性的錯誤判定。為降低串擾的現象我們可以加入 Guard 的防護設計來降低相互干擾的現象。

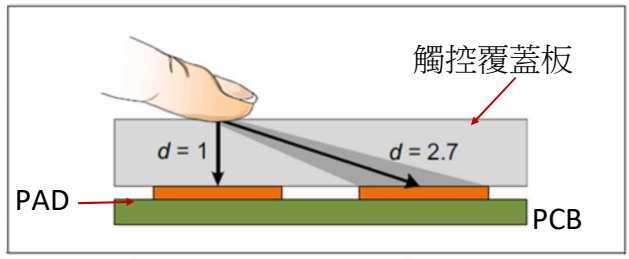


圖 1 顯示了手指的按壓的串擾

$$C = \varepsilon_r \varepsilon_0 \frac{A}{d}$$

Where:

ε_r = relative permittivity of the dielectric material

ε_0 = permittivity of space (8.854 x 10^{-12} F/m)

A = plate area in square meters (m^2)

d = distance between the plates in meters (m)

公式 1 ： 寄生電容的產生

位於目標觸控感應器周圍的其它觸控感應器，通過約 2-3 倍的厚度距離到周圍的觸控感應器，手指對周圍觸控感應器耦合的強度因儘量降低以降低彼此間的串擾。另一種思考這種關係的方式是：

在圖 1 及公式 1 裡，A 是觸控板 Pad 的面積，d 則是與 Pad 的距離。 在這裡先不管隔離的材質。由於手指接觸時你的鄰居會有 d=2.7 的互感，這會導致主觸控點的靈敏度下降，且來自 Pad 到 Pad 的耦合的串擾是另一種形式的干擾，這可能對設計產生負面影響。

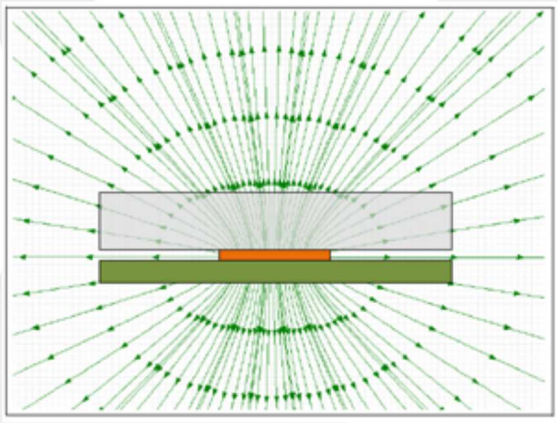


圖 2 單獨一個 Pad 時的電場線分布圖

如圖 2 所示電場線如何從電容式觸控感應器輻射出去。這些電場線影響相鄰觸控感應器的能力是基於它傳播的距離和材料。電場線方向主要是集中在正上方與正下方。側邊則比較少的。

如果這時有一相鄰的 Pad 時，這時電場線就會有極大的變化。如圖 3 所示，兩 Pad 之間相互影響在覆蓋物內電場線較少不容易造成互擾。當電場線穿出覆蓋物之後然後再透過覆蓋返回影響鄰近觸控感應器的串擾量將會大大增加。這在圖中顯示出作為強者與弱者間的區別耦合電場線。

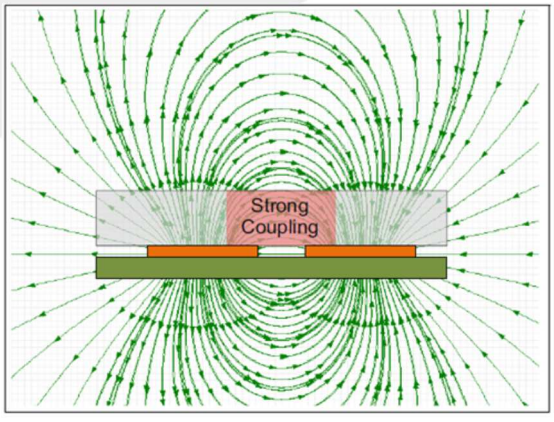


圖 3 兩相鄰的 Pad 的電場線分布圖

為降低這串擾的現象，底下有機個方法提出來討論：

對策 1 ：盡可能將觸控感應器距離拉開。理想的最小分隔是覆蓋物（壓克力）的 2-3 倍厚度的距離。

- 公式 1 中的距離 d，手指與觸控感應器保持較近的距離以減少觸控感應器串擾。
- 公式 2 中的觸控 Pad 寄生電容 C_{BASE} 是保持較低的容量，而手指的電容 CF 維持較大容量，這可增加的靈敏度 。
  CT = C_{BASE} + CF

$$C_T = C_{BASE} + C_F$$

Where:
C_T = total capacitance / measured value
C_{BASE} = sensor's base capacitance
C_F = finger's capacitance

公式 2 整體電容量

對策 2 ：覆蓋物（壓克力）開槽氣隙 (打洞)。

- 對於公式 1 中的覆蓋物的介電常數εr 介於兩者之間觸控感應器，如被降低到趨近於 “1” (空氣為 1)，這可以減少了觸控感應器之間的耦合，減少觸控感應器串擾。

對策 3 ：使用觸控感應器之間的保護(Guard)。

- 如圖 4 使用 Guard 的設計來建立一個低阻抗屏蔽於兩個觸控感應器之間。由於 Guard 在與被掃描的觸控感應器具有相同的電位，觸控感應器的電場線被集中在感應點的上、下方。

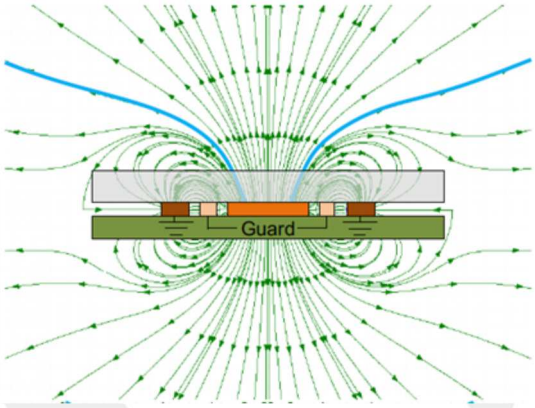


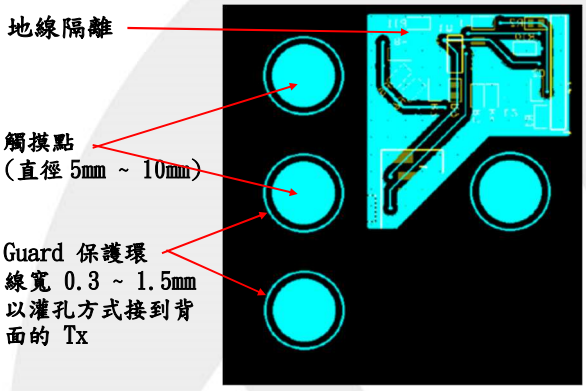
圖 4 使用 Guard 的隔離

1.4 主動防護驅動 (Active Guard Drives)

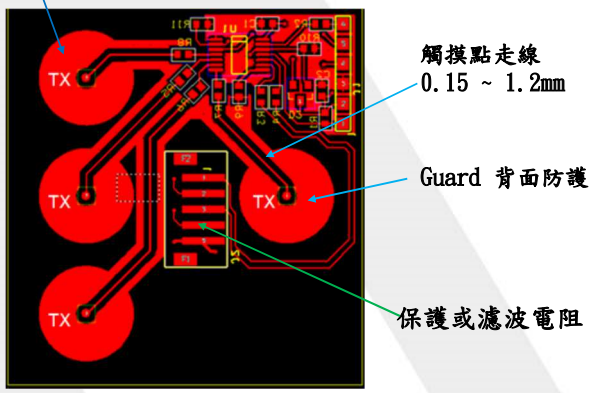
觸控感應點 (Pad) 的寄生電容量決定了感應器的靈敏度。Guard 究竟是如何驅動又是取決於的 mTouch 在何種時機掃描按鍵時同步來啟動 Guard 的輸出。但是 Guard 所形成的隔離電場不一定就可以完全的與原來的電場完全的匹配，一般透過簡單地 I/O 輸出做 Guard 的驅動，這觸控感應器大約會有 60-70%的防護效果。當然者也跟 PCB Layout 設計有很大的關係，底下有些經驗分享一下:

- 所有觸控感應器都可以共同使用一條 Guard 的輸出。mTouch 會以順序掃描方式進行感測。所以 Guard 在主動驅動防護時，觸控感應器在掃描的情況下是不會影響其它的感應器。
- 任何電源層或低阻抗的 PCB 走線應該考慮到對觸控感應點的保護。適當的在沒使用到的區域補鋪上、下層的地網，並增加灌孔以降低彼此之間阻抗達大最佳的抗干擾能力。但很重要的一點地網也不可以隨意鋪設的，考慮到觸控感應器的防水性功能，這地網的鋪線也要遵循規則的。我們將在下一章節裡討論到觸控按鍵的防水性時會在說明。
- 在 PCB Layout 的正面: 觸控感應點(Pad)的大小不建議小於 5mm 的直徑，理想的觸控點的 Layout 最好是大於 10mm 的直徑。
- 在 PCB Layout 的正面: 圍繞在觸控感應點周圍 Guard 環形線寬應該約 0.3 ~ 1.5mm 的寬度，並與觸控感應點約有 0.3 ~ 2mm 的距離。

- 在 PCB Layout 的背面: 從觸控感應器開始拉線到 PIC 之後接腳的這一段 PCB 走線也是很重要的，建議使用 0.15mm ~ 1.2mm 的線寬。Layout 時必須要用 Guard Line 將感應線及感應點包圍隔離，該 Guard Line 與觸控拉線之間的間距可以小到 0.4mm，最好以 Pad 的形狀在背面加入 Guard 防護 (如紅色 PCB 背面的 Tx Layout 方式)。
- 為降低來自感應點高頻雜訊干擾，在感應器拉線進入 PIC 接腳之前須串個 10K (建議值) 的電阻。這電阻最大的功能是: 一. 保護感應器的輸入腳避免 ESD 的損壞。二. 與內部電容形成一個低通濾波器可將高頻雜訊濾掉，增加訊號雜音比 (SNR) (濾波電容儘量靠近 PIC 的接腳，如此方可獲得較大的濾波效果)。



PCB 正面 Layout 配置



PCB 背面 Layout 配置

圖 5 顯示了一個主動防護裝置如何塑造一個觸控感應器的電場線以增加其靈敏度。由下圖與之前單一觸控點的電場分布來比較，加入 Guard 後電場線變的較往內集中，往外擴散所造成交互感應 (Cross Talk) 的變少了，電場線較集中該感應點的靈敏度相對也就變高了。

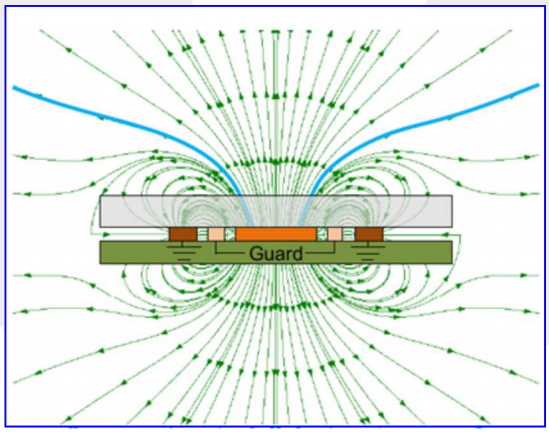
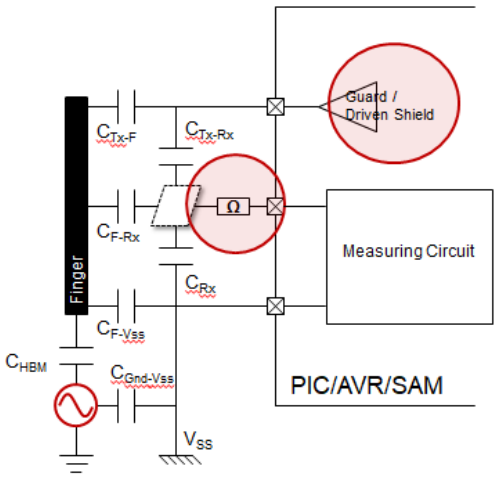


圖 5 加入 Guard 防護後的電力線分布圖

1.5 降低外部的干擾與濾波器

右圖為一個使用 Guard 輸出防護的觸控感應的等效圖示。我們先看 Pad 本身受手指電容量的影響，也受來自 Guard 的感應及雜訊、對地電容等影響形成一個複雜的電路環境。先從 Guard 的防護來說，當 mTouch 掃描到這 Pad 時，Guard 電場也就建立了，Guard 電場提供了一個正電場來隔離外界的雜訊，同時也因這正電場經手指 (CTX-F to CF-Rx) 到 Pad 再加上從 CTX_RX 的直接感應。所以 Pad 的量測值將會變動，如圖 b 所示 。



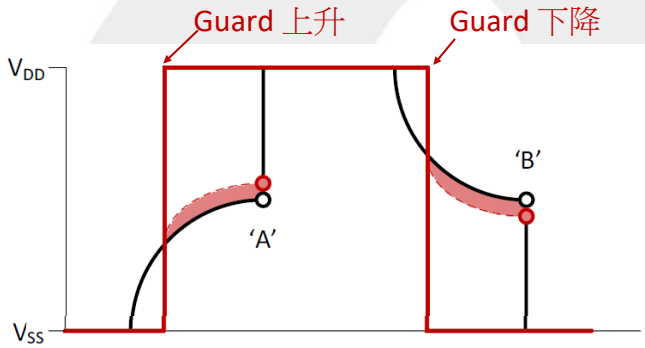
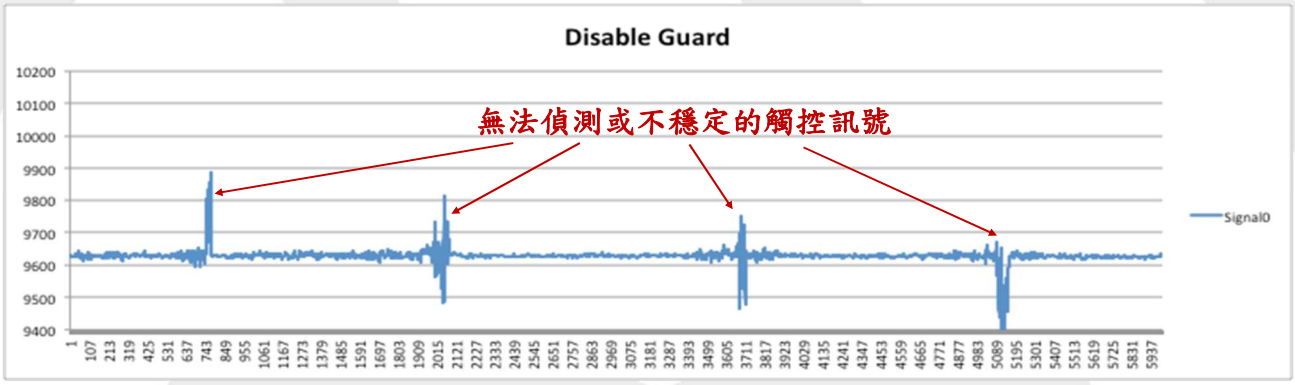
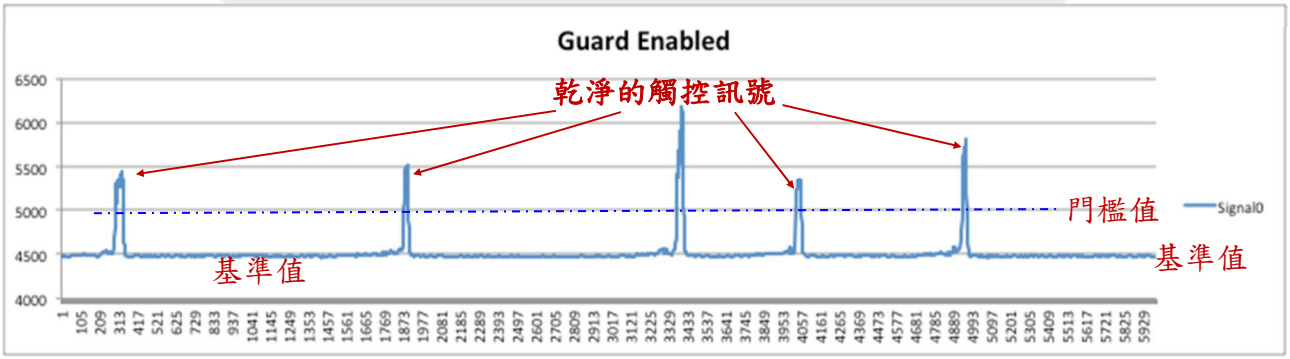


圖 b Guard 開啟與關閉時機

- 紅色線為 Guard 的輸出
- A 點為原先 ADC 轉換點位置，Guard 的電場會增加對 Pad 的感應，所以 ADC 轉換的電壓會升高。
- B 點為原先 ADC 轉換點位置，Guard 的電場在此會先消失不再感應到 Pad 上，所以 ADC 轉換的電壓會降低。



Guard 防護關閉時的訊號



Guard 防護啟用時的訊號

在如何加強觸控按鍵的穩定度在前面已討論了一些對策，最後我們要談到的是外在高頻雜訊的干擾與防治之道。Pad 露在外面沒有屏障物來降低干擾，再加上不算短的走線，這些都是感應天線對於外來的高頻極易感應進來干擾造成觸控按鍵的基準線的雜訊過大形成誤動作的情況發生。

再者 Pad 呈現一個高阻抗的輸入除了雜訊容易入侵外還有一個就是極易造成該腳位的 ESD 損壞。

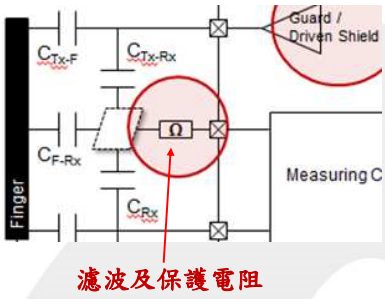


圖 6

圖 6 所示在 Pad 與 PIC 觸控感應輸入接腳之間串上一個 10K ohm (建議值) 的電阻即可發揮濾波及防護的功能。

再則從這電路的等效圖來看，PIC 接腳有一 5PF 的寄生電容量 (Cpin) 再加上 ADC 輸入時有一 8k ohm 的內阻與 10PF 的取樣電容所形成的一個 二階的 RC 低通濾波，來對輸入訊號進行低通濾波。

圖 7 是使用 10 ohm 的串聯電阻，可以看到頻率的 -3db 點頻率還是很高，莫約在 2.3MHz 左右。

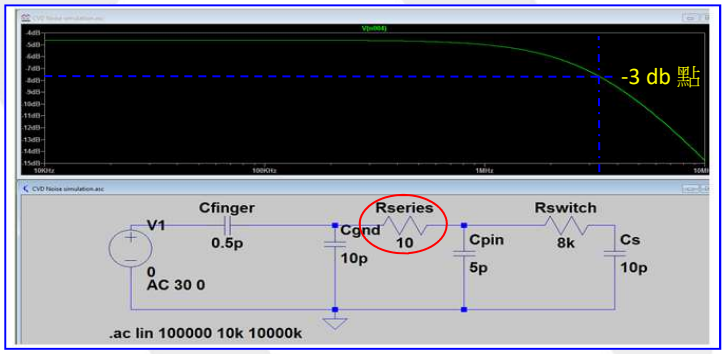


圖 7

圖 8 使用 10K ohm 的電阻，-3db 點將到 1.8MHz ，降低了雜訊的干擾。

有許你會問那這樣使用更高阻抗的電阻那輸入雜訊不就可以濾的更多嗎? 當然有一好就沒有二好，加大 RC 濾波常數這也會形成較慢的反應，取樣電容充電時間就需較久的時間。當然真正設計上的所需電阻值也是要視環境及外在的雜訊大小來決定，在這裡我們的建議值是 10k ohm 左右。

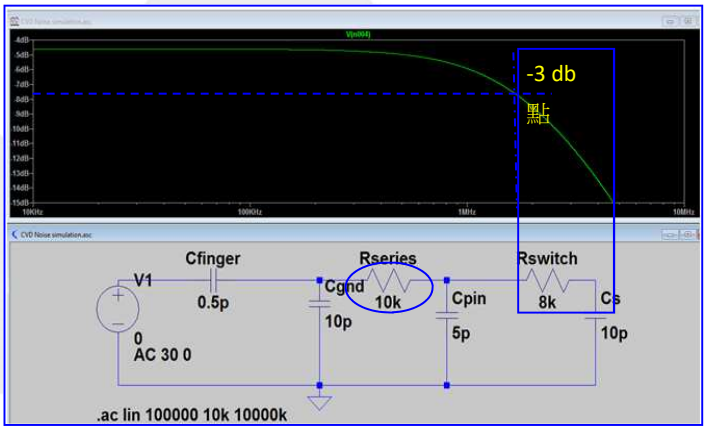
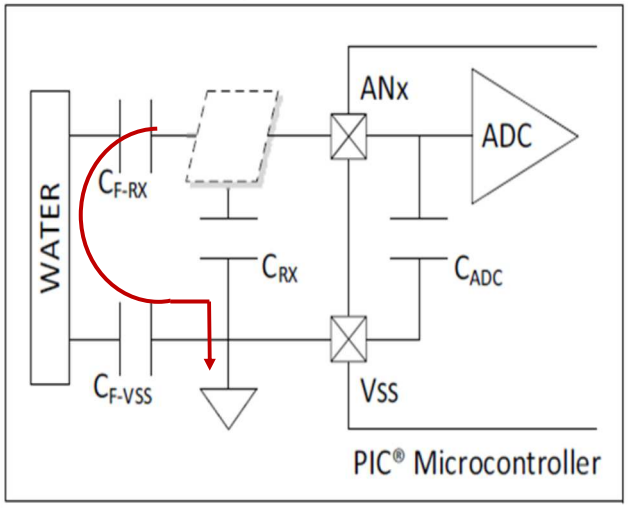


圖 8

1.6 觸控的防水性

通常討論到這防水議題就會牽扯出很多的理論，在這裡我們只做簡短的說明為什麼 mTouch 具有防水的功能。之前很多做觸控感應的工程師為降低雜訊干擾往往會在 Pad 周圍及底層佈滿了地線來降低外來的干擾；當然這是不錯的抗干擾方法之一，如果這是在乾燥的環境下是沒問題的。但是一但有水沾上或被潑水在感應 Pad 上時，觸控就有可能會有誤動作的現象發生。因為水與電路板的地線會形成一個寄生電容，Pad 沾到水後與地線之間的耦合就像加入一個並聯的電容，導致 Pad 整體電容量變大而造成誤動作。

水是個導電體，當 Pad 有水時，水將透過 CF-RX 和 CF-VSS 耦合到感應器和電路板的地線，這也相對於 CRX 是建立一個並聯電容分掉了 Pad 上 CRX 的電荷，Pad 測量到的電壓就會降低，類似手指觸碰到 Pad 的效果導致誤動作，如圖一所示。



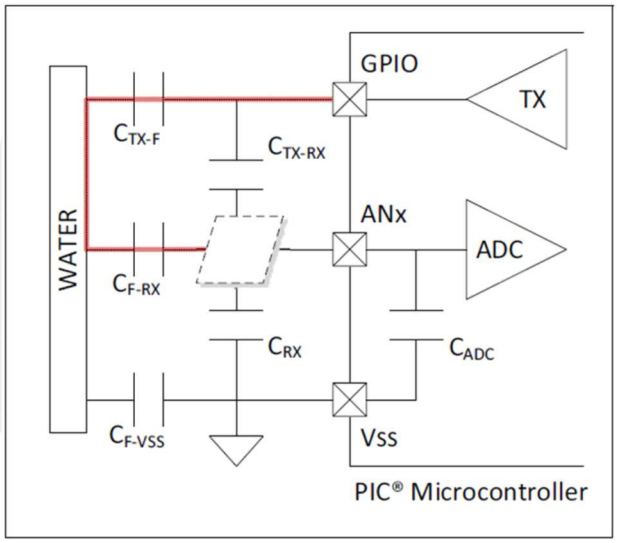
圖一

為了加強防水功能在 mTouch 設計裡加入了 Guard 的防護。除了先前所講的抗互擾等功能外，適當的使用 Guard 的輸出及 Layout 才可以達到防水功能。

如圖所示圖二所示: Guard TX 信號將透過 CTX-F 耦合到水中，但水的存在也會自 CTX-RX 拉走一些的充電電荷。此時，水中的電荷有兩種可能的路徑：透過 CF-RX 到感應器及 CF-VSS 路徑到 PCB 的地線。Guard 的訊號在水中會再經 CF-RX 的耦合重新導回一些電荷到感應器中，如此可降低 Pad 電荷的因水所造成的經 CF-VSS 到地的耦合所造成的電荷損失。

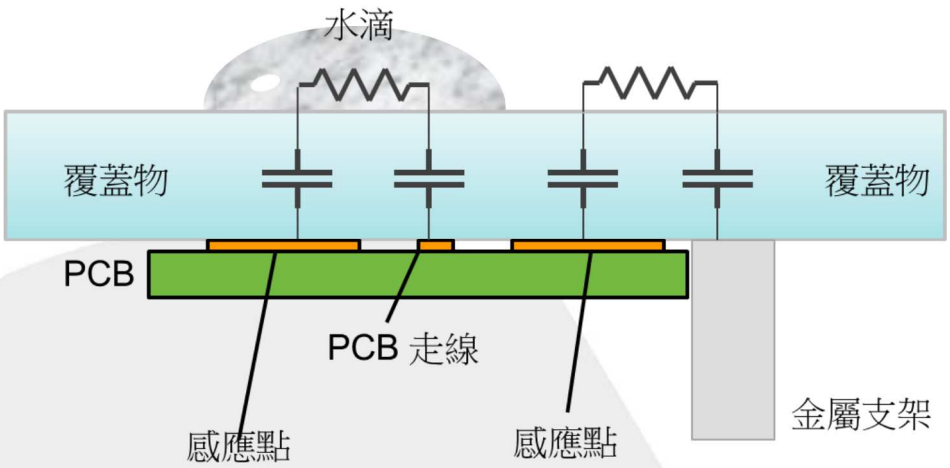
水多耦合經 CF-RX 耦合到 Pad 的電荷就多；水少 CF-RX 容量就小耦合到 Pad 的電荷就少。如此就可以維持 Pad 的基本電荷(電壓)而不因水而造成影響產生誤動作。

當然這只是個基本的說明，想了解更多的防水原理請參考: AN2098 mTouch® Sensing Technology Water Resistance



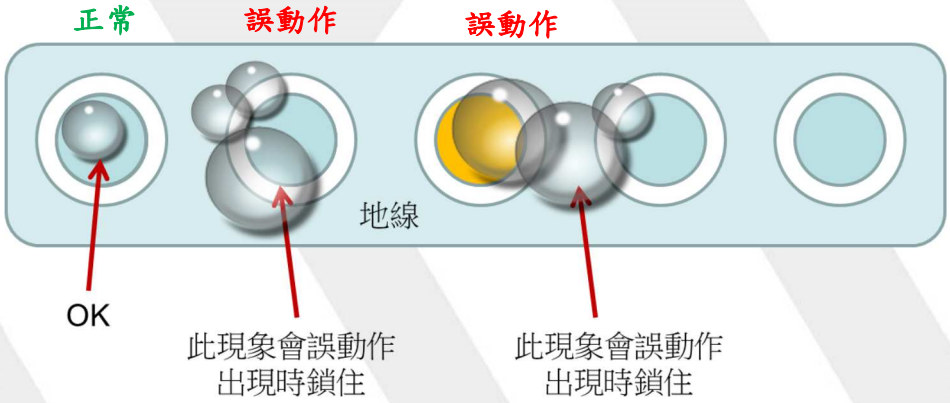
圖二

右圖為等效圖，水滴在感應點上會與地線及其它的走線形成一個並聯電容效應導致容量變大 (電壓變低) 產生有觸控輸入的誤動作。

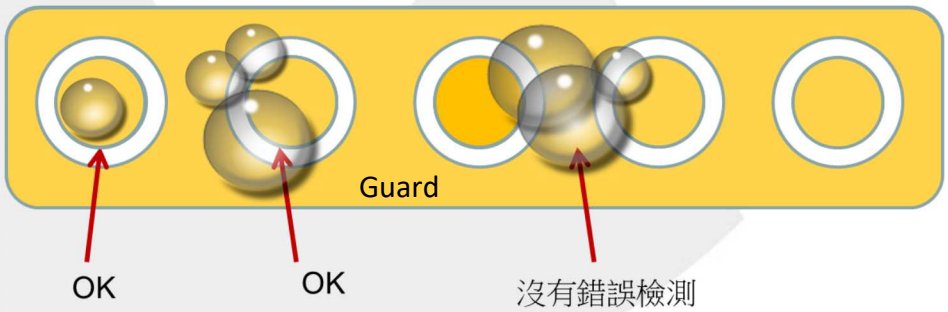


觸控感應點使用地線圍繞可以降低雜訊的干擾，但也會造成觸控無法防水滴(點)。

只要 Pad 與地線之間有水滴的耦合，Pad 上的電荷電壓就會被耦合到地線(電壓變小) 而發生誤動作。



使用保護環 (Guard Ring) 和所有其它電極都以與掃描電極相同的電位驅動。Guard 的輸出會填補因水滴所造成的電荷耦合到地線的損失，如此就大大降低因水滴所造成的誤動作。



經由上頁簡單的防水原理說明，但實際在硬體 PCB 設計上還是有要注意的事項，如下:

要求 1 ：

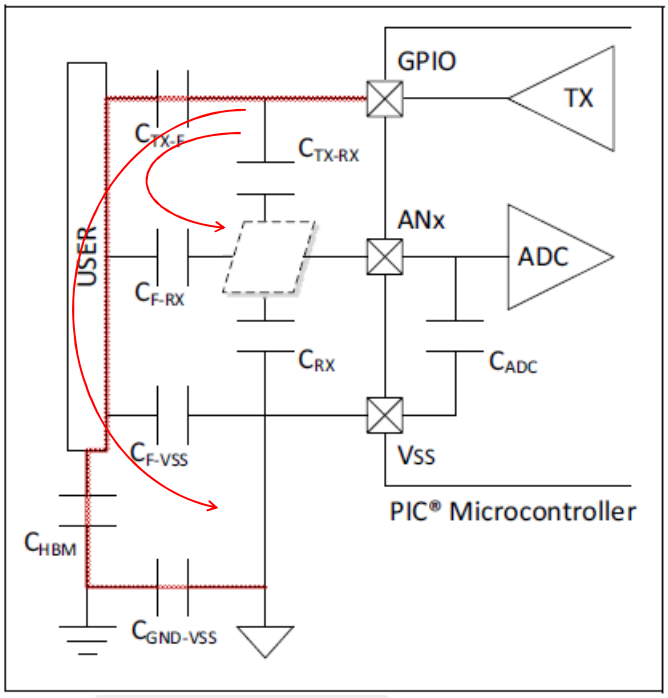
經由上頁簡單的防水原理說明，但實際在硬體 PCB 設計上還是有要注意的事項，如下:

要求 1 ：

當沒有手指觸摸時，“CTX-F 到 CF-RX” 路徑阻抗必須比 “CTX-F 到 CF-VSS” 的路徑具有較低的阻抗。TX 信號是主要且直接的耦合到感應器的訊號。

這導致了兩個設計注意事項：

- 請勿將 VSS (地線) 放置在感應器附近。相反的，建議在 Pad 感應器周圍及下方鋪設一個 Guard 的鋪網線屏蔽以降低地線的影響。也就是說: 感應器附近的水必須是耦合到感應器及 Guard，而不再是耦合到電路板的地線。
- 覆蓋在感應 Pad 的覆蓋材料不能導電，這是絕對的要求。



圖三

要求 2 ：

“CTX-F 到 CHBM (人體電容) 到 CGND-VSS” 的路徑的阻抗必須比路徑 “CTX-F 到 CF-RX”更低。人體必須能夠將 TX 信號從感應器上拉開並將其消散到電路板的地線。注意這時有兩個地的存在，一個是人體所接的大地及電路板的地。

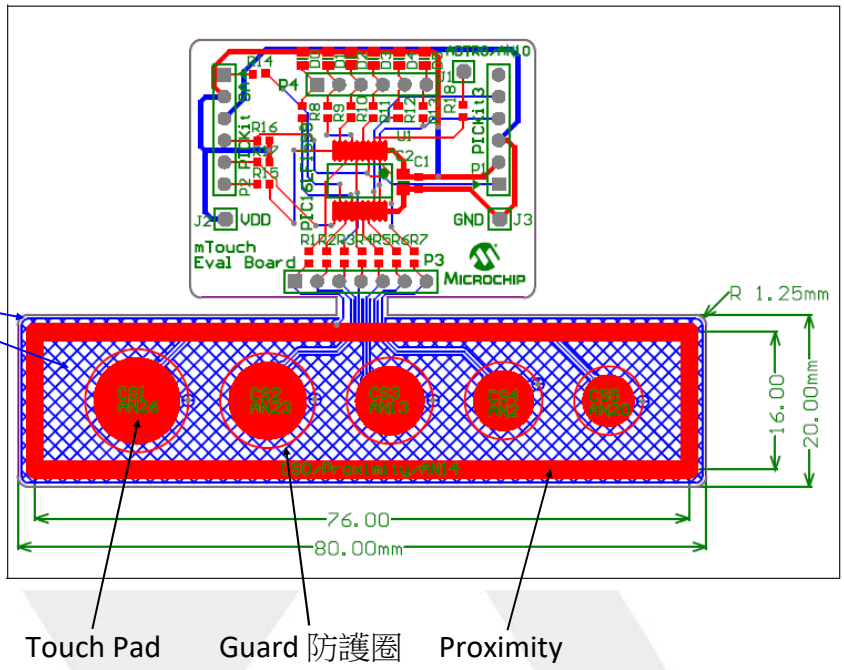
有幾個應用需要考慮：

- 當使用電池動力系統是最大的挑戰對於這個電路板的地耦合的要求。必須有一個機制允許人體的接地與電路板的地結合在一起。這可能意味著，類似於手機，用戶是需要拿起設備時須將用戶的地線耦合到用戶的所握的裝置來進行電路板的接地。(虛擬地)
- 非隔離電源是首選，它允許用戶和線路板共享一個地線。
- 用戶身體耦合到電路板的地線可以以各種方式實現。用車測試門把手已經顯示出用戶正在接近一輛車雖然不能共享汽車的地線，但可以通過汽車的車體將汽車的地線連接起來形成一個大面積的地來跟人體的大地的耦合，以滿足這一要求。
- 覆蓋層材料的厚度。如果材料太薄會削弱用戶接地的耦合。CTX-F 到 CF-RX 路徑可能會更強，打破 CTX-F 到 CHBM (人體電容) 到 CGND-VSS” 的路徑的阻抗必須比路徑 “CTX-F 到 CF-RX”更低需求。

1.7 防水的 PCB Layout

FIGURE A-1 是一個使用 PIC16LF1559 所做的防水的 PCB Layout 圖示。在這裡 Pad 及 Proximity 的背面 (藍色部分) 被鋪滿了網格式的 Guard TX 線。這取代了之前的鋪地網的觀念。使用 Guard 的鋪網格的硬體設計，如此就可以具有防水的功能。當然 mTouch 軟體的配合也是不可缺的。

FIGURE A-1: LAYOUT EXAMPLE – PIC16LF1559 LOW COST mTouch® EVALUATION BOARD



Touch Pad　　Guard 防護圈　　Proximity

1.8 結論

要做好一個觸控輸入裝置不是向簡單的任務，尤其是在抗干擾議題上極為棘手。在這裡我們做了抗干擾對策的一些心得:

I. 加入 Guard 的防護機制是很重要的

II. PCB Layout 對抑制干擾及互擾是很重要的，請遵循 Layout 的需求

III. 使用輸入串聯電阻來抑制輸入雜訊及提升 ESD 的保護

IV. 使用較大容量的取樣電容 (HCVD 或 ADCC) 硬體架構的觸控元件。

第二章: APP046 mTouch 觸控電路板

2.0 MCC 裡的 mTouch 是有版本的要求，請使用者確定所安裝的軟體版本。本練習所使用的軟體開發工具版本：

- MPLAB® X IDE v4.05 或更新版本
- XC8 compiler v 1.43 或更新版本
- MCC Version v3.36 或更新版本

2.1 本實作練習所使用的實驗板名稱為 APP046 mTouch ，該實驗板使用 28-pin DIP 包裝及 IC 插座，可適用於 Microchip PIC16F 及 PIC18F 所有 28-pin DIP 包裝的元件。在接下來的練習將使用三顆元件做 mTouch 的練習。

Lab1 : SCVD 的練習，使用 PIC16F15355 的元件。

Lab2 : SCVD 使用 CallBack 函數的練習，使用 PIC16F15355。

Lab3 : ADCC 的練習，使用 PIC16F18855 的元件。

Lab4 : mTouch 加入 Crypto 的驗證功能，使用 PIC16F18855 。

以上兩顆都是 28-pin 的元件，腳位相容且都有內建 Debug Module 方便除錯且都可以使用 3.3V 供電。在 APP-ESS18-2 (APP046) mTouch 實驗板上只要更換 IC 就可做不同的 mTouch 練習。底下列出這幾顆元件的主要周邊列表的參考:

PIC16F15355 (SCVD)

Device	Data Sheet Index	Program Flash Memory (KW)	Program Flash Memory (KB)	Storage Area Flash (B)	Data SRAM (bytes)	I/OPins	10-bit ADC	5-bit DAC	Comparator	8-bit/ (with HLT) Timer	16-bit Timer	Window Watchdog Timer	CCP/10-bit PWM	CWG	NCO	CLC	Zero-Cross Detect	Temperature Sensor	Memory Access Partition	Device Information Area	EUSART/ I2C-SPI	Peripheral Pin Select	Peripheral Module Disable	Debug (1)
PIC16(L)F15354	(A)	4	7	224	512	25	24	1	2	1	2	Y	2/4	1	1	4	Y	Y	Y	Y	2/2	Y	Y	I
PIC16(L)F15355	(A)	8	14	224	1024	25	24	1	2	1	2	Y	2/4	1	1	4	Y	Y	Y	Y	2/2	Y	Y	I

PIC16F15355 為 5 位數編碼的新元件，內建多種標準周邊及多樣的 CIP 為一全功能低單價的 MCU。該系列 從最少的 8-pin 包裝的 PIC12F15313 到最多腳位 48-pin 的 PIC16F15386 ，完整系列共有 13 個元件成員，屬於低價位泛用型的 MCU。

PIC16F18855 (ADCC)

Device	Data Sheet Index	Program Flash Memory (Words)	Program Flash Memory (KB)	EEPROM (bytes)	Data SRAM (bytes)	I/O Pins(1)	10-Bit ADC2 (ch)	5-Bit DAC	Comparator	8-Bit (with HLT)/ 16-Bit Timers	SMT	Windowed Watchdog Timer	CRC and Memory Scan	CCP/10-Bit PWM	Zero-Cross Detect	CWG	NCO	CLC	DSM	EUSART/I2C/SPI	Peripheral Pin Select	Peripheral Module Disable
PIC16(L)F18854	(1)	4096	7	256	512	25	24	1	2	3/4	2	Y	Y	5/2	Y	3	1	4	1	1/2	Y	Y
PIC16(L)F18855	(2)	8192	14	256	1024	25	24	1	2	3/4	2	Y	Y	5/2	Y	3	1	4	1	1/2	Y	Y

PIC16F18855 一樣為 5 位數編碼的新元件，內建多種標準周邊及多樣的 CIP 為一全功能低單價的 MCU。該系列 從最少的 28-pin 包裝的 PIC16F18854 到 40/44-pin 的 PIC16F1877 (56KB Flash) ，該系列元件主要是以高容量的記憶體及支援 ADCC (10-bit ADC + Computation) 的周邊。

PIC16F1513 (HCVD)

Device	Data Sheet Index	Program Memory Flash (words)	Data SRAM (bytes)	High Endurance Flash (bytes)	I/Os(2)	ADC		Timers (8/16-bit)	EUSART	MSSP (I2C/SPI)	CCP	Debug(1)	XLP
						10-bit (ch)	Advanced Control						
PIC16(L)F1512	(1)	2048	128	128	25	17	Y	2/1	1	1	2	I	Y
PIC16(L)F1513	(1)	4096	256	128	25	17	Y	2/1	1	1	2	I	Y

PIC16F1513 不算是新元件，這顆元件主要的訴求是低單價及超低功耗 eXtreme Low-Power (XLP) 的應用。在實驗中主要是使用它內建的 ADC Advanced Control 功能，也就是 HCVD 功能。

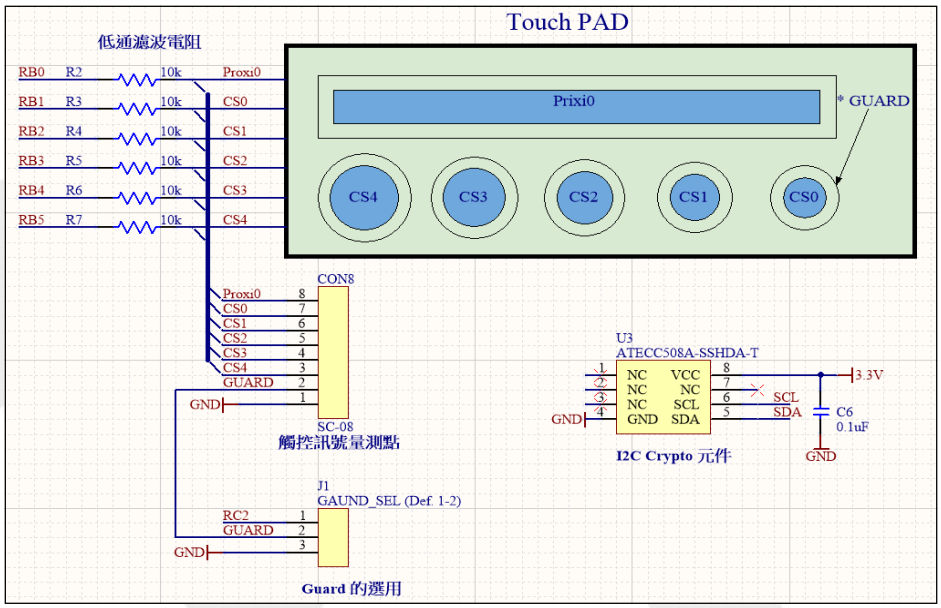
2-2 APP-ESS18-2 (APP046) mTouch 實驗板電路圖 (底下統稱 APP046 mTouch 實驗板)

基於 PIC 28-pin 封裝腳位相容的優點，不管是使用新的或舊版的 PIC16F 或 PIC18F 都可以使用 APP046 mTouch 來做練習的。電源的供應需使用外接 USB 來供應 5V 的電，內部再經 LDO TC2117-3.3 (3.3V 輸出，800mA) 降壓到 3.3V 後做為系統的工作電壓。

APP046 mTouch 實驗板上有一顆 PIC18LF25K50 的 MCU，其主要功能有二：一是模擬一個燒錄的資料夾，只要將 Hex 檔複製到此資料夾後該 Hex 檔就會自動燒錄的 PIC 裡；二是一個 USB Dongle 的功能， 將 MCU 所測量到的觸控資料經 UART 送給 USB 虛擬的通訊 Port 後將觸控資料做動態的顯示或使用 Data Visualizer 做觸控資料的分析 ，讓使用者可以針對觸控資料做參數上的調整。觸控的感應跟 PCB 走線的長短、觸控點的大小及觸控點上方所覆蓋的壓克力厚薄有很大的關係。透過即時資料的顯示，使用者可以輕鬆地調整出所需的參數以達成其觸控的應用需求。

底下為 APP046 mTouch CVD 觸控板電路圖，電路圖分成四塊模組，並一一說明其功能：

電路圖一：觸控面板、濾波電阻、觸控訊號測試點及加密認證



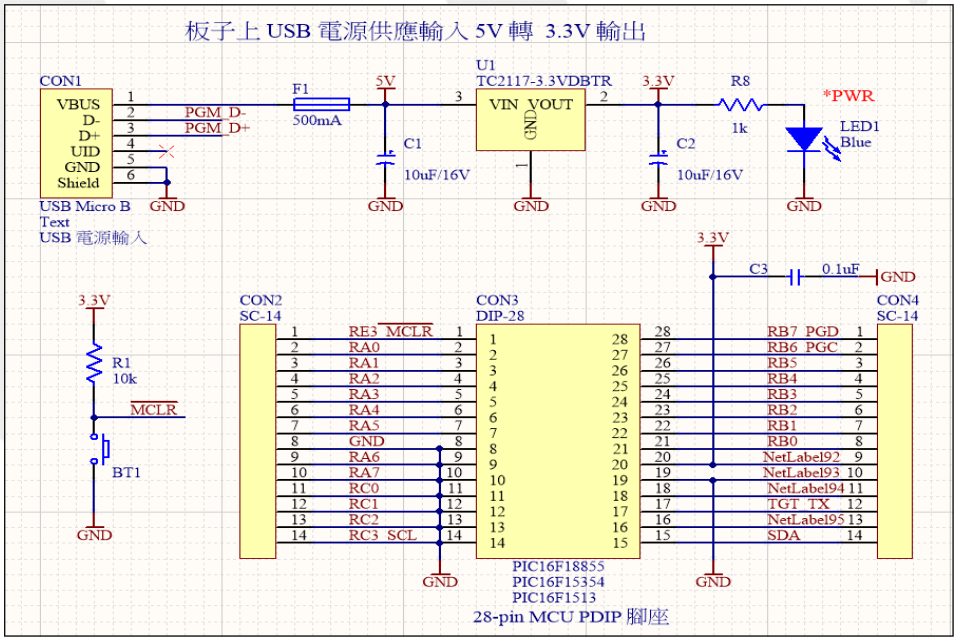
a. APP046 mTouch 實驗板有使用 Guard 的防護架構，因此 PCB 的走線有其規則限制。關於 Guard 的 Layout 如前面所述。

b. 電路圖裡每個觸控點 (CS0 ~ CS4, Proxi0) 都有串接一個 10K 歐姆的電阻，這是為降低雜訊干擾所加的電阻，由觸控點往內看時會該電阻與輸入端的寄生電容會形成一個二階的低通濾波器藉以降低一些高頻的雜訊的干擾。

c. J1 為 Guard 訊號的選擇。使用者藉由 J1 可以將 Guard 接給 RC2 來啟用 GUARD 的防護，也可以將其接地，以感受 GUARD 的防護威力、觸控靈敏度及防水性的差異。

d. ATCC508A 是一個安全認證元件，使用 I2C 介面與 PIC16Fxxxx 通訊。此為 option 功能，使用者可以透過 USB 來做認證的動作。

電路圖二：USB 供電穩壓，MCU 接腳功能



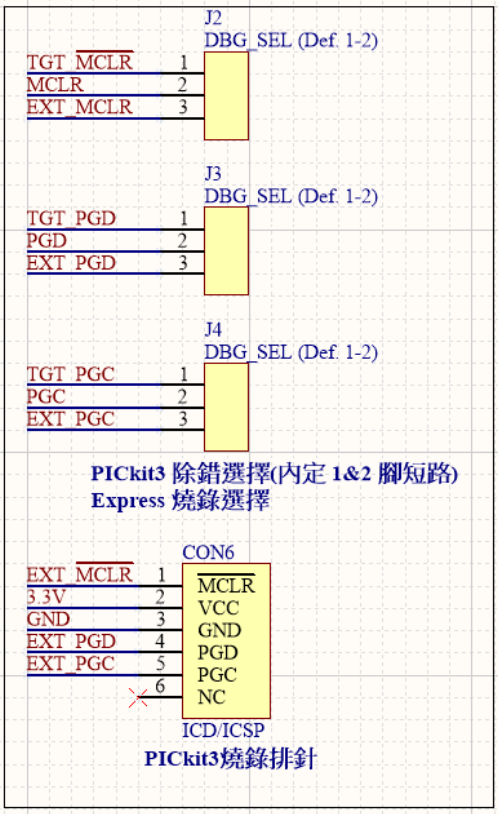
a. APP046 mTouch 實驗板的供電是來自 USB Micro 的腳座。USB 5V 電源經 LDO 穩壓成 3.3V 供給電路板使用。上電時，藍色 LED (LED1) 會點亮。

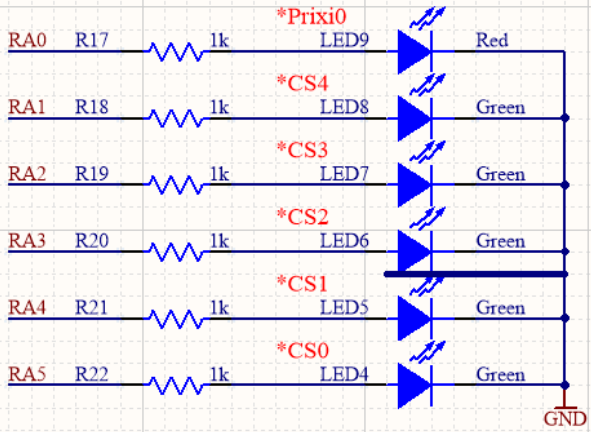
b. USB 的 D+ & D– 是接到 PIC18LF25K50，主要用來做 UART to USB 及檔案直接燒錄功能(後述)。

c. MCU 依使用者的需求，採用 28-pin PDIP 腳座 (CON3)。如電路圖的標示，在這裡可依不同的 mTouch 的功能來更換 PIC 做實驗與測試。 CON2 & CON4 為擴充排針，其功能也用文字標明以方便做測試。

電路圖三：PICKit3 與 PIC18LF25K50 燒錄切換、觸控狀態 LED 顯示

a. J2 ~ J4 為三個緊連的排針。主要是選擇燒錄是要用 PICKit3 (CON6) 或是要使用 PIC18LF25K50 所提供的燒錄功能。建議使用 PICKit3 (內定) 就可以使用除錯及燒錄功能。

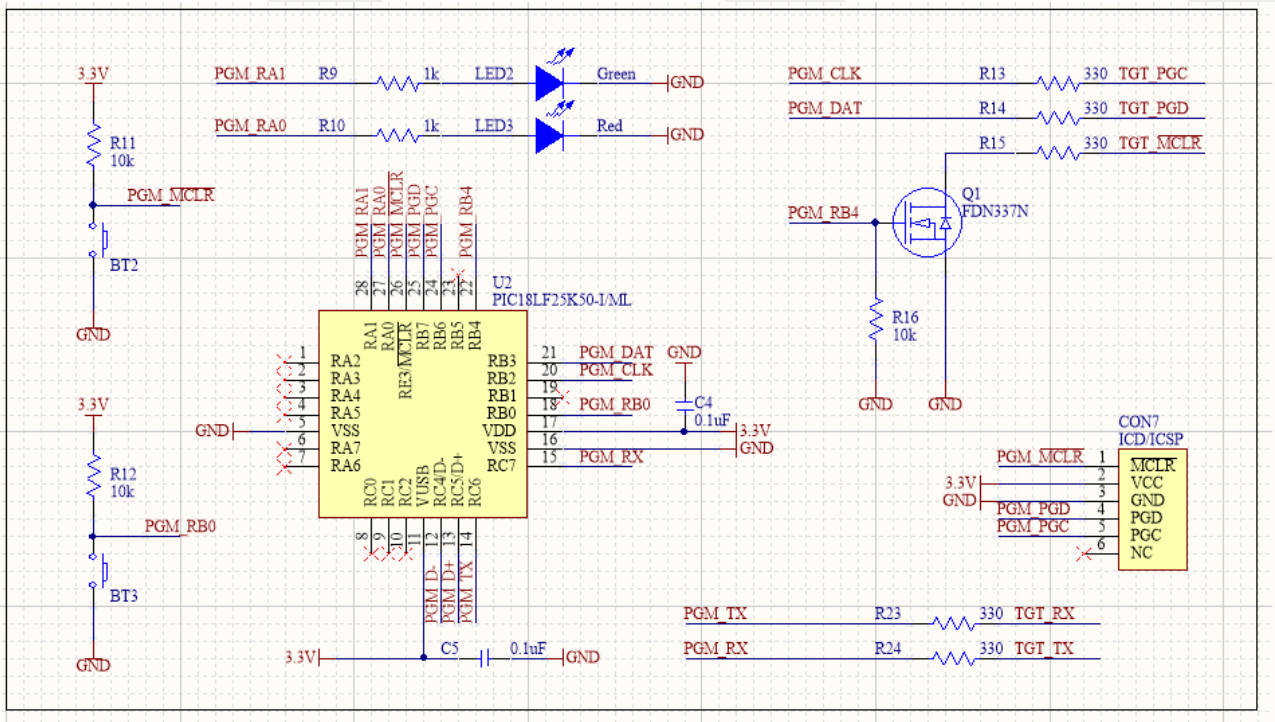




b. LED4 ~ LED8 為觸控顯示 LED。只要 CSx 有被觸摸，相對應的 LEDx 就會被點亮。若持續觸摸同一個 CSx，因程式會依背景做自動校正功能，相對應的 LEDx 會一段時經後會自動熄滅。

LED9 (Prixi0) 為近接距離感應，其亮度是 PWM 方式輸出。距離越近燈越亮方式呈現

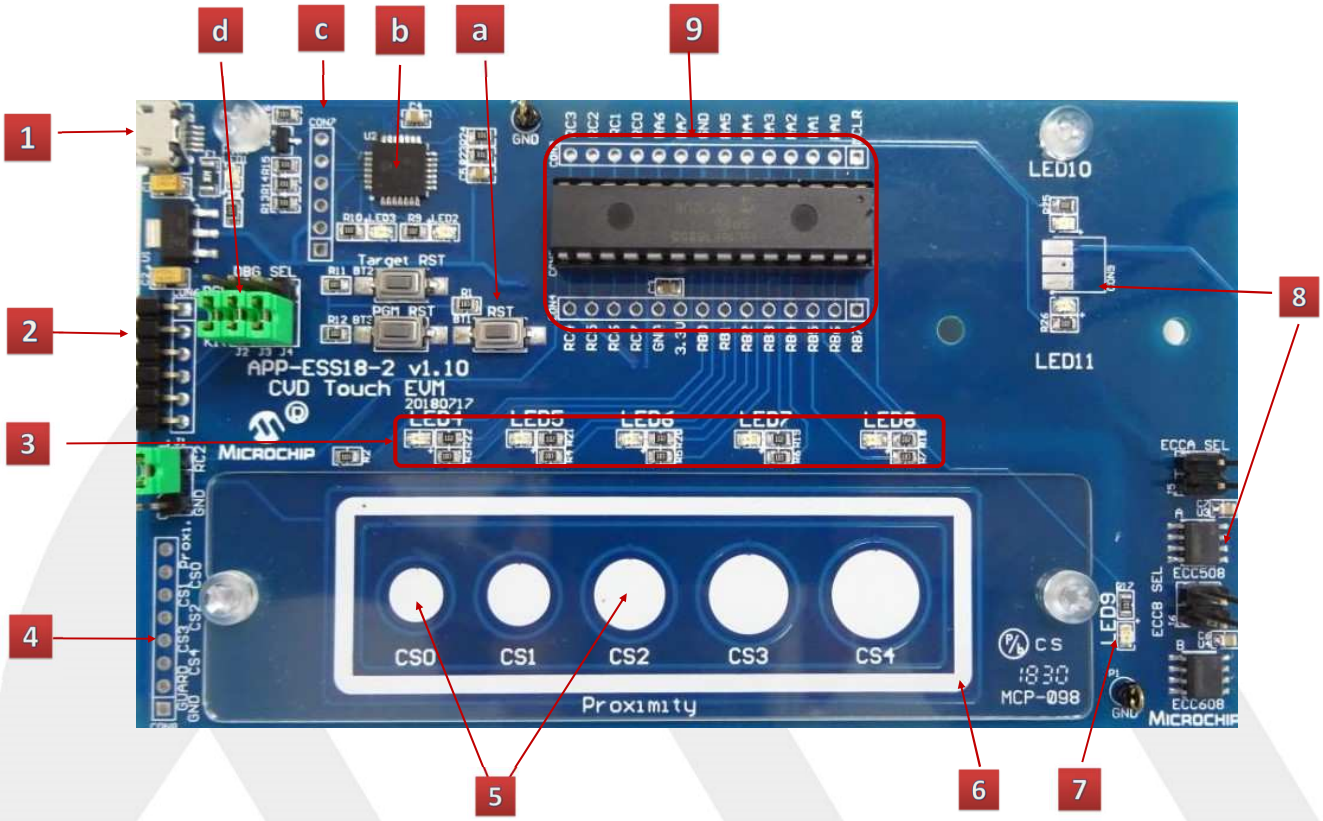
電路圖四：PIC18LF25K50 USB UART 及 On-Board 燒錄



a. CON7 為 PIC18LF25K50 標準的 ICSP 燒錄腳，用於更新 USB to UART 的韌體。

b. USB to UART 的腳位經電阻 R23 & R24 連接到 28-pin 的 PIC16F18855 的 UART 腳位。

2-3 APP046 mTouch 實驗板零件配置說明



1. Micro USB 接頭，提供 5V 系統供電，也做為 USB to UART 的轉換功能。
2. PICKit3 6-pin 排針式接腳，用於 28-pin PIC16F18855 之除錯及燒錄，須配合 J2 ~J4 使用。
3. 相對位置的觸控顯示 LED (CS0 ~ CS4) 及 Pad 輸入濾波電阻。
4. 觸控感測點的測量點 (Gnd, Gurad, CS4 ~ CS0, Proxi0)。
5. 五個大小不一樣的觸控感應點 (CS0 ~ CS4)，越大的感應點電容量改變效果較大就越靈敏。
6. Proximity 接近感應環。
7. Proximity 的顯示 LED，用以顯示接近的狀態。越靠近感應器 LED 越亮 (PWM 驅動方式)。
8. Crypto 加解密驗證區域。
9. PIC 28-pin 排針式接腳，用於 28-pin PIC16F18855 腳位訊號測量及擴充外接點。

a. PIC16F18855 Reset 按鍵。

b. PIC18LF25K50 UART to USB 功能轉換及 On-Board 燒錄。

c. PIC18LF25K50 ICSP 燒錄腳位，更換韌體用。

d. PICKit3 除錯燒錄及 On-Board 燒錄選擇。

第三章: mTouch Library 快速上手 (Lab1)

3.0 練習前的準備

MCC 的使用是有需求的，請確認底下的軟體是否已安裝完畢且功能正常:

- MPLAB® X IDE v4.05 或更新版本
- XC8 compiler v 1.43 或更新版本
- MCC Version v3.26 或更新版本

關於上述的軟體安裝請參考 “PIC101 X IDE & MCC & XC8 的綜合基礎課程” 裡的“MPLAB v3.35 & MCC v3.15 .軟體安裝 v2.0”裡的說明。安裝 MCC 時也許會需要更新 Java 到新版本，這時請到 Java 網站更新。

目前 mTouch v2.0 的版本上只支援 1D 的按鍵功能，如果需要滑條或滾輪的應用則會在下一版本支援，如現在就需要滑條或滾輪的應用可以使用 QTouch 的解決方案來完成。

3.1 練習一 (mTouch 16F15355 Polling.X) 所使用的元件是 PIC16F15355，這是一顆功能強大內建多樣周邊的元件。PIC16F15355 是加強型中階 PIC 的架構，28-pin 的腳位及 14K Bytes 的記憶體容量，具有內容豐富的 CIP 及低功耗的特點。

如果讀者想多了解這顆 PIC 的應用可以參考 Microchip 所出的教育訓練光碟裡的 8-bit 中文教育訓練教材 “ CIP102v2.00 進階獨立式週邊(Adv. CIP)的應用“，此教材以 MCC 為設計的基礎，解講 PIC16F1619 裡的 CIP 的設定及應用，是本不錯的 CIP 基礎教材。

PIC16F15355 Data Sheet : http://www.microchip.com/wwwproducts/en/pic16F15355

3-2　本練習使採用一步接一步的做法來完成的，只要讀者 “按圖施工、保證成功” 即可輕鬆完成基本的觸控按鍵。本練習採用 PIC16F15355 的元件，但也適用在其他的 PIC16F1xxx 系列的元件裡，所以當您能成功完成此練習，這以意味著您也可以擴展到其他的 PIC 觸控應用的需求。底下此專案範例項目包括以下內容：

- 五個名稱為 CS0 ~ CS4 的觸控按鍵輸入及五個觸控按鍵 LED4 ~ LED8 顯示
- 一個名稱為 Proxi0 的接近感測器及一個 LED9 顯示
- 啟用一隻輸出 I/O 腳做為防護訊號輸出 (Guard)
- 啟用 AFA (Automatic Frequency Adaptation) : 自動頻率調配，使用 Timer2 做跳頻產生器。
- 啟動 UART1 (115.2Kbps, N, 8, 1) 傳送觸控觸控資料到終端機顯示。

3-3　PIC16F15355 腳位的對應。這對應表格是很重要設定 MCC mTouch 時的參考表，依據 APP046 mTouch 的電路圖我們設定了 mTouch 所使用到的腳位。

	名稱代號	腳位名稱	功能
1.	Proxi0	RB0	Proximity Input
2.	CS0	RB1	Touch Pad 0 (Small)
3.	CS1	RB2	Touch Pad 1
4.	CS2	RB3	Touch Pad 2
5.	CS3	RB4	Touch Pad 3
6.	CS4	RB5	Touch Pad 4 (Large)
7.	Guard	RC2	Guard Output
8.	CS0_ LED	RA5	LED4
9.	CS1_LED	RA4	LED5
10.	CS2_LED	RA3	LED6
11.	CS3_LED	RA2	LED7
12.	CS4_LED	RA1	LED8
13.	Proxi0_LED	RA0	LED9
14.	TxD	RC6	UART Tx

PIC16F15355 腳位功能對照表

有了這張腳位功能對照表後就可以開啟動手實作練習一

動手做練習一

1 開啟 MPLAB® X 的專案，並使用 APP046 mTouch® 實驗板:

- 確定 DBG_SEL Jumper (J2, J3, J4) 接在 PICKit 的位置 (使用 PICKit3)
- 接上 PICKit3 到 CON6 ，做為燒錄及除錯用
- Micro USB 連線到 CON1，做為系統供電及觸控資料的分析
- 開啟 MPLAB X IDE
- 關閉所有的專案，確定沒有專案原始檔案區是空的

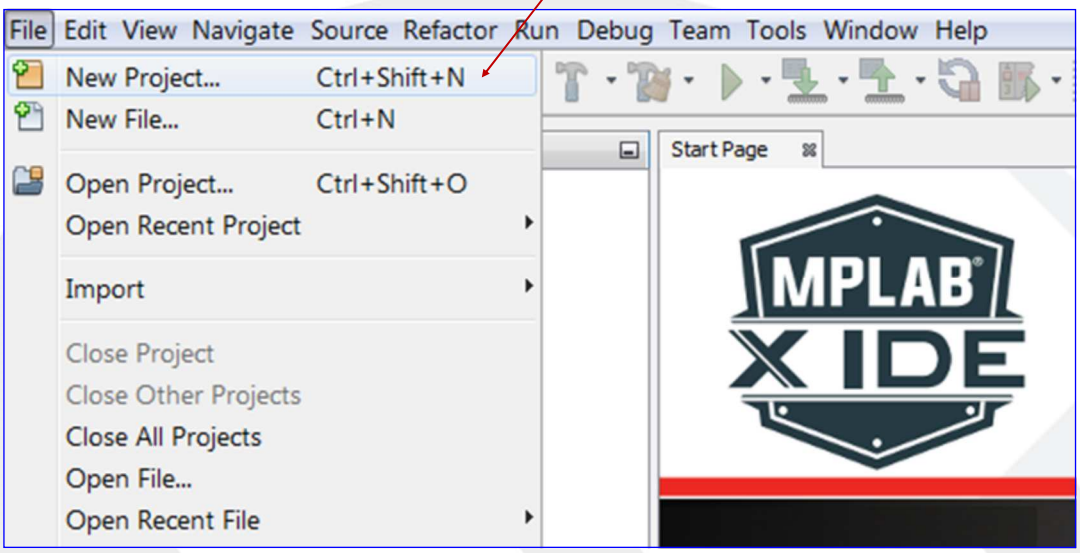
2 建立一個新的專案 （參考底下的的步驟，一步接一步地完成）

- 建立名稱為”mTouch 16F15355 Polling.X”的專案
- 元件選擇 PIC16F15355
- 專案目錄 : c:\mTouch CVD
- 檢視 MPLAB® X 已經建立了一個新的專案
- 檢視一下新專案下的 Source Files 是空的

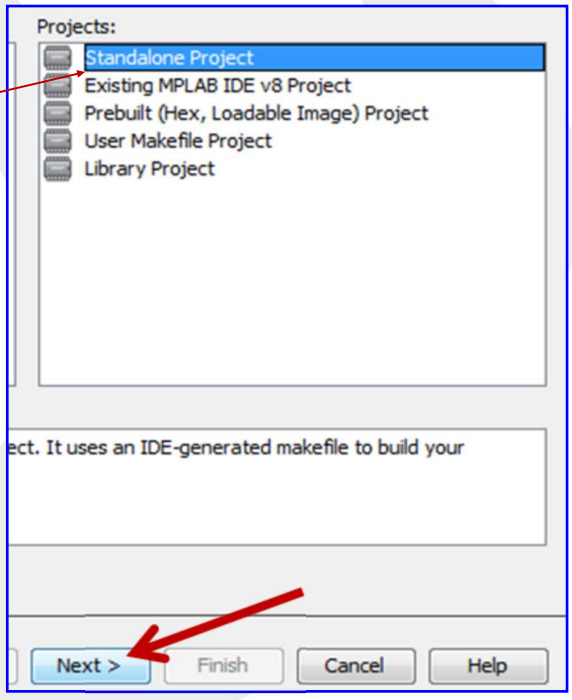
關於 MPLAB® X IDE 建立新的專案及更詳細的操作，請參考 Microchip 教育訓練光碟裡的教材 “PIC101 X IDE & MCC & XC8 的綜合基礎課程” 及 “ MPLAB X IDE 中文版 “ 的說明。

開始建立專案

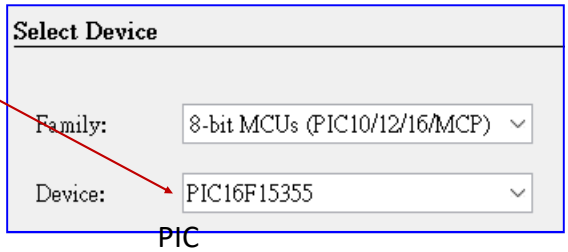
- MPLAB® X IDE 下在 “File” 選項，點選 “New Project” 或是透過 “專案精靈 “ (project wizard) 快速建立專案.



- 專案必須選擇 “獨立式專案” (Standalone Project) 後再點選 “Next”



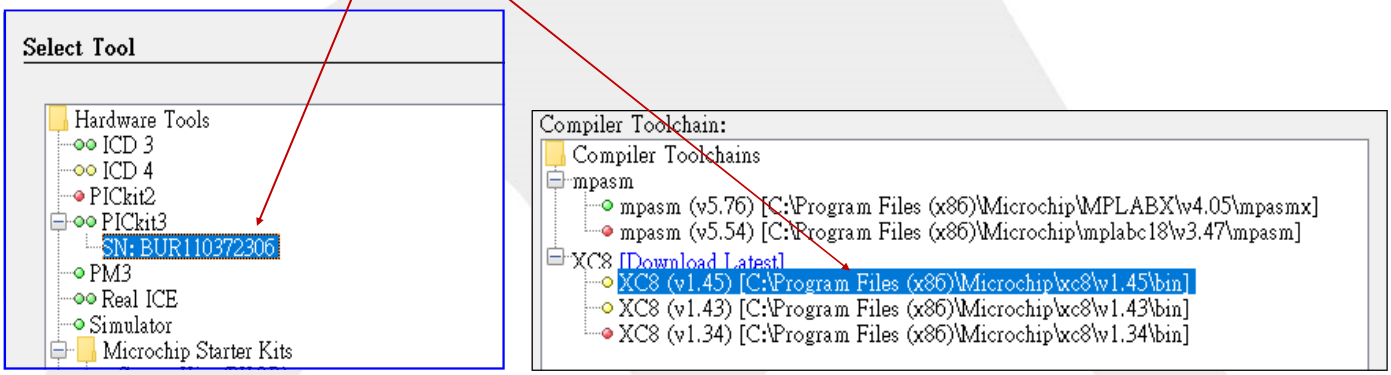
- 找出元件所屬的族群或直接輸入元件名稱 PIC16F15355
- 確認選擇的元件為 PIC16F15355 後再點選 “Next”
- 接下來則會顯示除錯所需使用到的轉換座，在本練習無須外接轉換座，請直接按下一步。



PIC

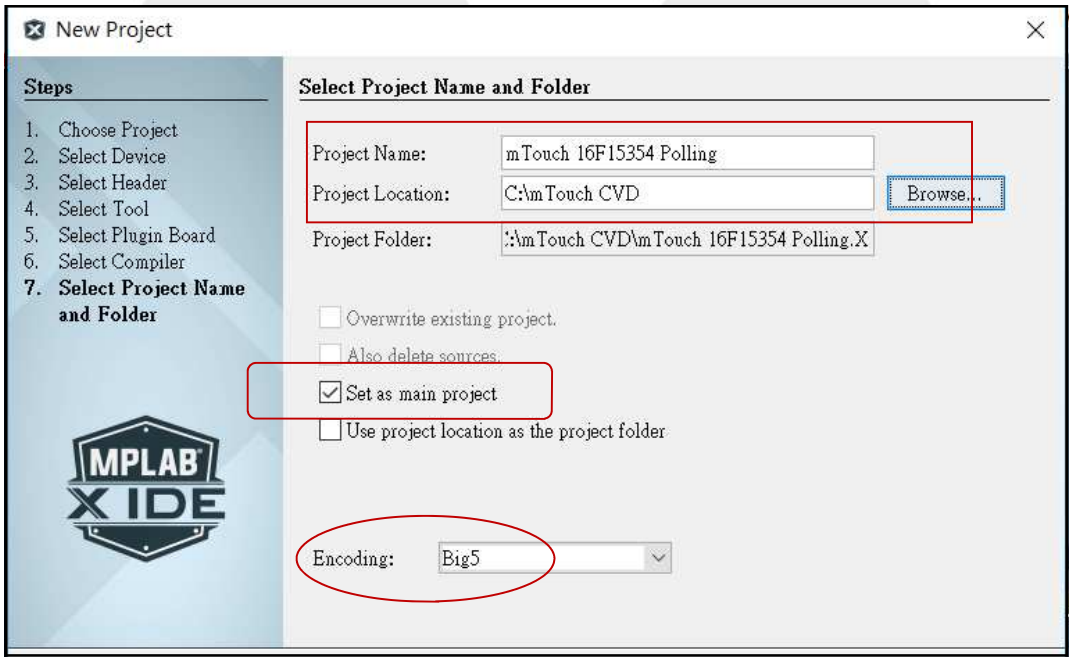
設定專案（繼續...）

- 在 “Select Tool” 選項裡。請先確定 PICKit3 已經插在 PC 的 USB 插槽上，選擇 **PICKit 3** 做為開發的工具。請點選 PICKit3 的序號後再按 “Next”
- 選用 XC8 (v1.43 或 v1.45) 編譯器後，按 “Next”



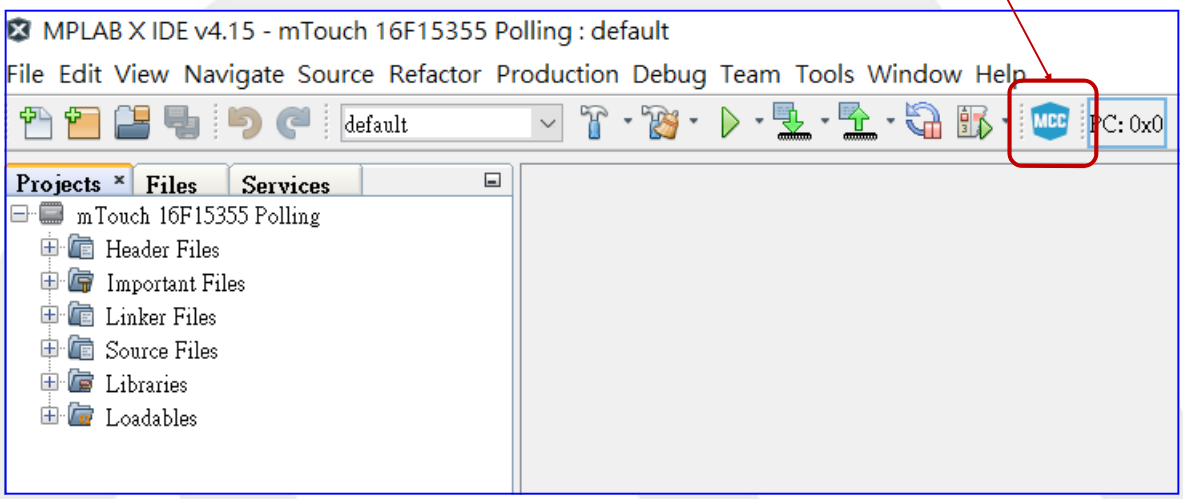
接下來是要設定專案的名稱及專案的路徑，記住不要使用中文的路徑。初學者建議使用書上的路徑來做練習。

- 如下圖所示，輸入 “mTouch 16F15354 Polling” 做為專案的名稱.
- 設定專案的檔案位置: c:\mTouch CVD，這時在 Project Folder 會顯示整個專案的的檔案路徑及專案名稱 (會在專案名稱後加上 .X 的附加檔名做為識別)
- 確認一下將該專案設成主專案，請勾選 “Set as main project”
- 如果程式裡要使用中文做說明，最後面的編碼 (Encoding) 需用 “Big5” 或 “UTF-8” 的編碼
- 最後按下 “Finish” 完成專案的設定

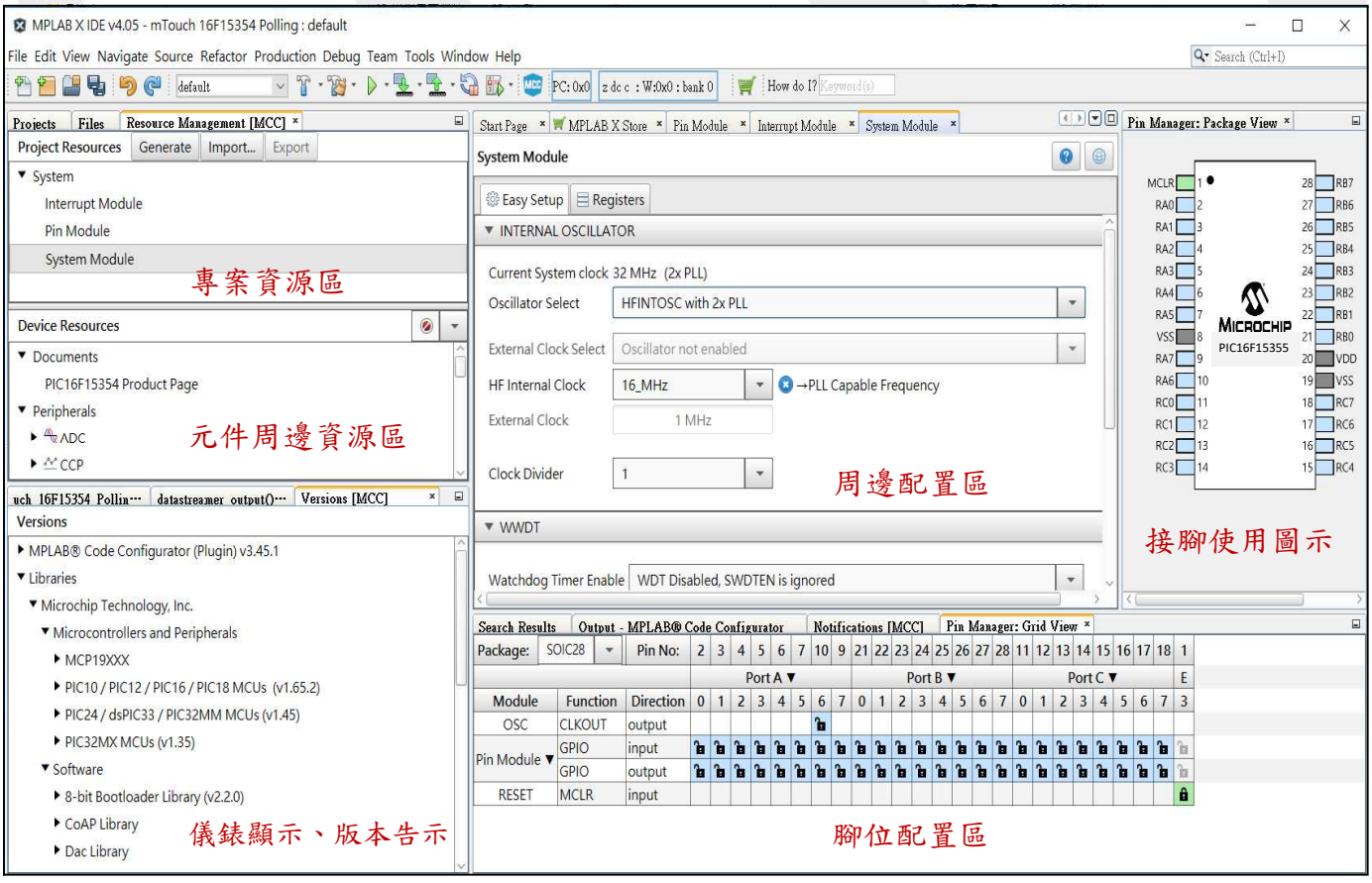


3 開啟 (MCC) MPLAB® Code Configurator 工具

- 專案建立完成後將可看到名稱為 “mTouch 16F15355 Polling” 的一的空白專案
- (使用 MCC 的工具，你必須事先安裝 MCC v3.xx 到 MPLAB X IDE)
- 啟用 MCC，按下 MCC 的圖示，或在 “Tools” 下點選 MCC 即可以啟動 MCC
- **MCC 的啟動需花一點時間開啟，請耐心稍等一下**。



- 啟用了 MCC 後，IDE 會開啟一些專屬 MCC 的視窗
- 在專案所屬的視窗會開啟一個 “MCC Resources “ 的視窗，如下圖之說明



有關此實作所需的基本技能知識可以參考底下的教育訓練光碟: (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。

4 開始使用 MCC，設定系統工作的頻率

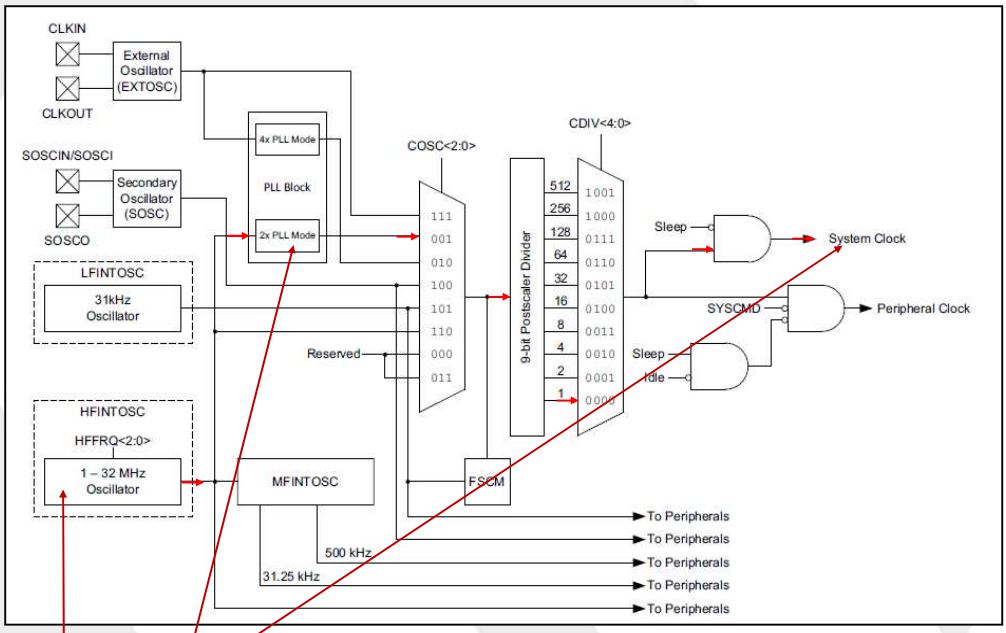
- 因為使用軟體 CVD 的方式做觸控按鍵，所消耗的系統需求 (System Performance) 會比較多。所以建議使用最高的執行速度 (32MHz)來完成。
- 選擇 “ HFINTOSC with x2 PLL” 內部震盪模式做為時脈來源，將 16MHz 經 x2 PLL 倍頻成 32MHz，將系統頻率輸出設為 32 MHz (Fosc)

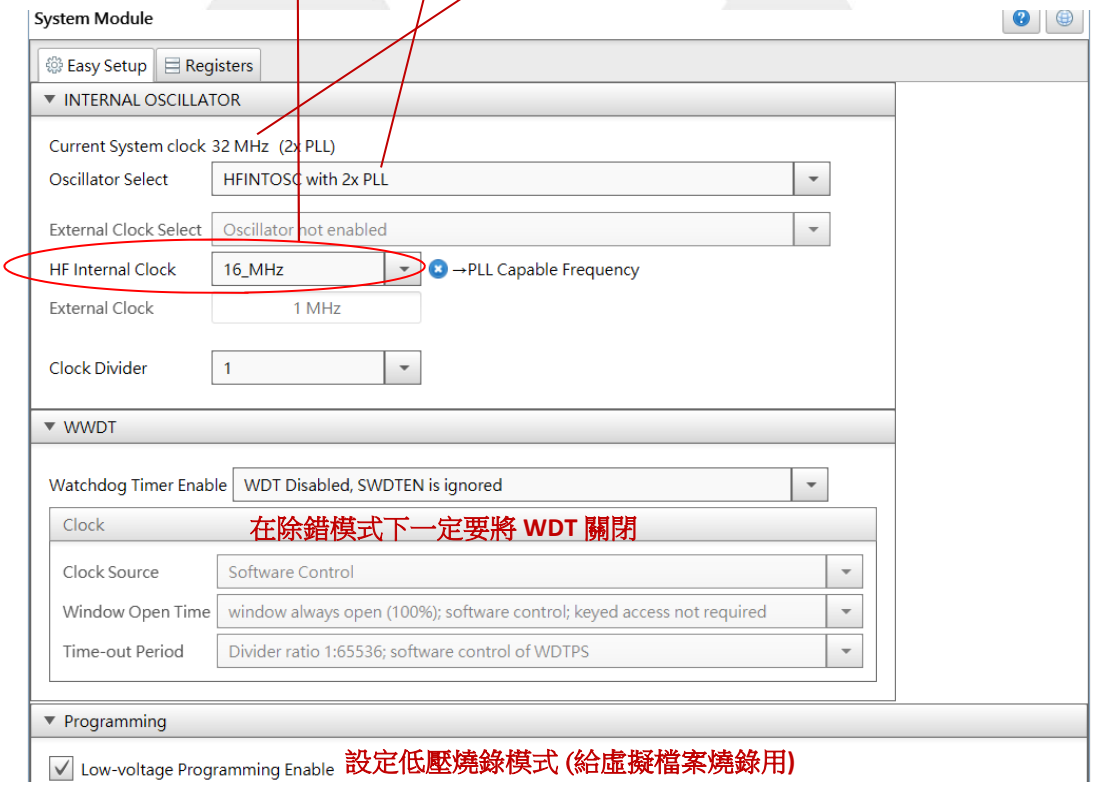
※ 點選 MCC 專案資源的 “System Module” 開啟 PIC16F15355 的 Clock 及 Config. Words 的設定項目

⇒ 看門狗計時器 (WDT) 需關閉

⇒ 使用內部振盪的 16MHz 輸出後經 x2 PLL 倍頻輸出

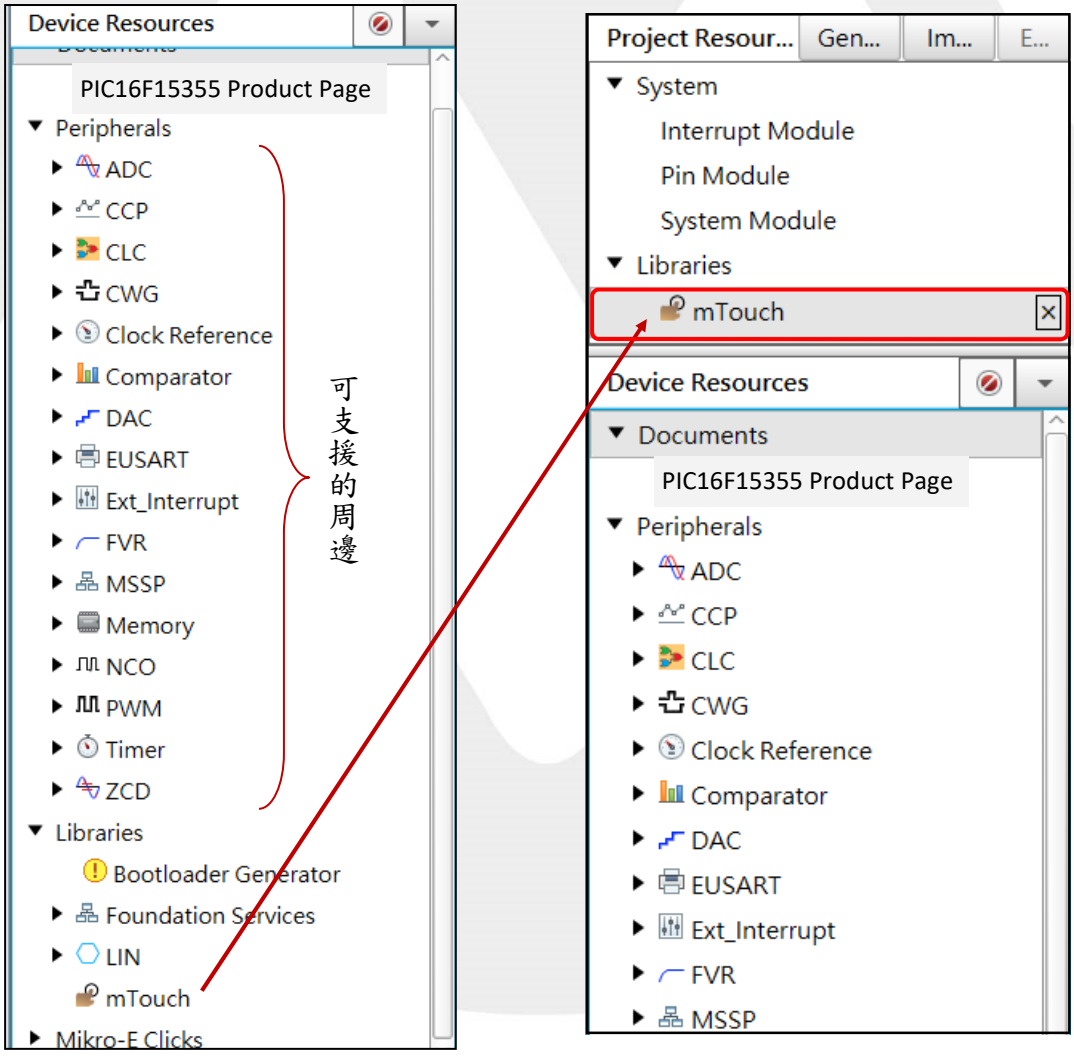
⇒ 系統頻率: 32MHz





5 載入 mTouch 模組

- 目前在 “Project Resources” 下並沒有 mTouch 模組的存在。必須到 “Device Resources” 下尋找，下拉找到 “mTouch” 的圖示後並在這圖示上點選兩次，這 “mTouch” 就會自動載入 “Project Resources “的視窗。這時只要再點選 mTouch 即可開啟 mTouch 的設定。
- 由此也可以看到 PIC16F5354 有多種標準周邊及 CIPs 周邊的支援。



6 設定 mTouch 使用的觸控的腳位

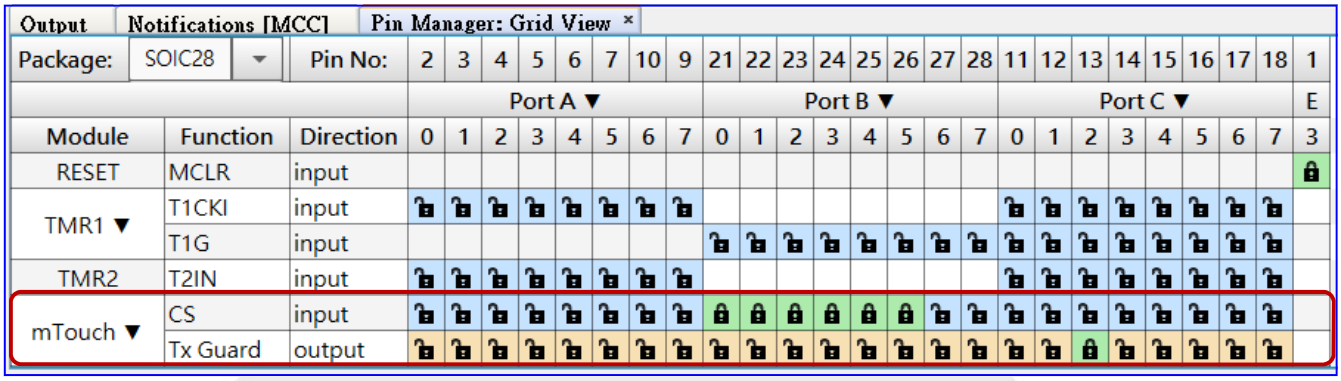
- 當加入了 mTouch 模組後。這時就需考慮觸控感應要使用那些腳位來輸入。沒有事先規劃觸控腳位 mTouch 模組是無法設定的。首先到 “Project Resources” 下點選 “Pin Module” 就可以看到 “Pin Manager: Grid View” 的設定視窗。

- 請依照底下的腳位功能對照表所使用到的觸控輸入腳位在 “Pin Manager: Gird View” 的選項裡勾選所需的觸控感應腳位及 Guard 腳位 (Item 1 ~ Item 7)

Item	名稱代號	腳位名稱	功能
1.	Proxi0	RB0	Proximity Input
2.	CS0	RB1	Touch Pad 0 (Small)
3.	CS1	RB2	Touch Pad 1
4.	CS2	RB3	Touch Pad 2
5.	CS3	RB4	Touch Pad 3
6.	CS4	RB5	Touch Pad 4 (Large)
7.	Guard	RC2	Guard Output
8.	CS0_ LED	RA5	LED4
9.	CS1_LED	RA4	LED5
10.	CS2_LED	RA3	LED6
11.	CS3_LED	RA2	LED7
12.	CS4_LED	RA1	LED8
13.	Proxi0_LED	RA0	LED9
14.	TxD	RC6	UART Tx

Page 23 的 PIC16F15355 腳位功能對照表

在這範例裡，使用 RB0 做為接近感測，RB1 ~ RB5 為五個觸控感測輸入腳。所以將 “Pin Manager: Gird View” 的 mTouch 選項裡的 CS 功能項目將 RB0 ~ RB5 由黃色的開鎖狀態點選成 綠色的上鎖狀態。由於我們在這裡指定了觸控按鍵的輸入腳位，屆時在使用 MCC 下的 mTouch 設定，就需要參考到這些可用的觸控感應輸入腳位。這時也可以在 “Pin Module” 的視窗看到 mTouch 感應器的腳位都連帶地開啟 ADC 的輸入功能。



Output | Notifications [MCC] | Pin Manager: Grid View

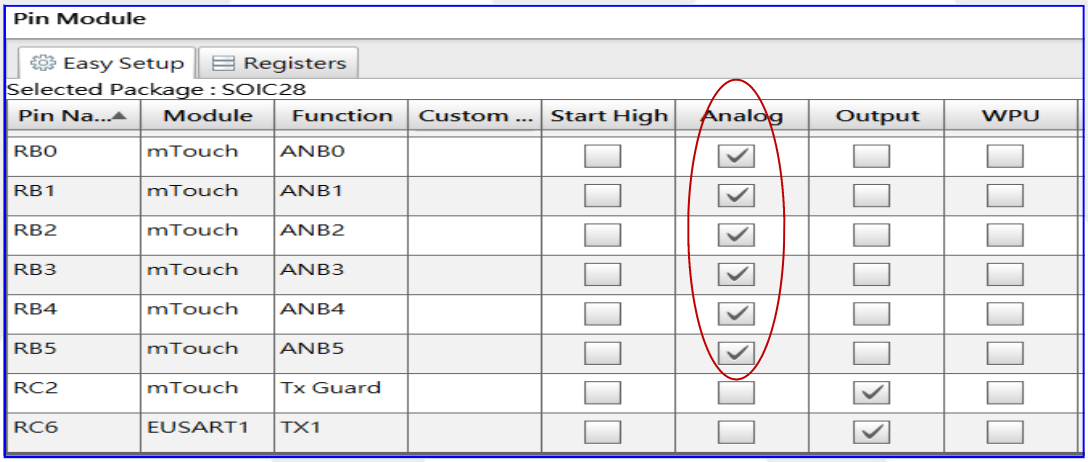
Package: SOIC28 | Pin No: 2 3 4 5 6 7 10 9 21 22 23 24 25 26 27 28 11 12 13 14 15 16 17 18 1

Port A ▼ | Port B ▼ | Port C ▼ | E

Module	Function	Direction	A0	A1	A2	A3	A4	A5	A6	A7	B0	B1	B2	B3	B4	B5	B6	B7	C0	C1	C2	C3	C4	C5	C6	C7	E3
RESET	MCLR	input																									🔒
TMR1 ▼	T1CKI	input	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓									🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	
	T1G	input									🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	
TMR2	T2IN	input	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓									🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	
mTouch ▼	CS	input	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔒	🔒	🔒	🔒	🔒	🔒	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	
	Tx Guard	output	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔒	🔓	🔓	🔓	🔓	🔓	

Pin Manager: Gird View 視窗

很重要的一點是 mTouch 的輸入腳必須使有 ADC 輸入功能的腳位，並要檢查一下 ADC 輸入功能一定要勾選。



Pin Module

Easy Setup | Registers

Selected Package : SOIC28

Pin Na...	Module	Function	Custom ...	Start High	Analog	Output	WPU
RB0	mTouch	ANB0		☐	☑	☐	☐
RB1	mTouch	ANB1		☐	☑	☐	☐
RB2	mTouch	ANB2		☐	☑	☐	☐
RB3	mTouch	ANB3		☐	☑	☐	☐
RB4	mTouch	ANB4		☐	☑	☐	☐
RB5	mTouch	ANB5		☐	☑	☐	☐
RC2	mTouch	Tx Guard		☐	☐	☑	☐
RC6	EUSART1	TX1		☐	☐	☑	☐

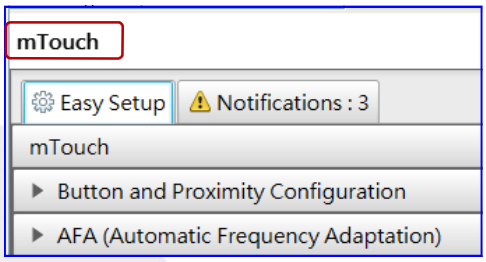
Pin Module 視窗

mTouch 模組有一個很重要的輸出訊號 Guard。這 Guard 在做觸控按鍵時會自動建立一個隔離電場，用來降低外在的干擾，這是很重要的訊號，使用時也須配合 PCB Layout 的配置法則來處理。依照上圖所示: Guatd 腳位是設定在 RC2 的，所以下 Pin Manager: Gird View 的設定裡請將 RC2 設成 Guard 的輸出。

圖示中，綠色上鎖 狀態的腳位表示已被其它周邊所使用了

7 開始設定 mTouch 模組

- mTouch 的設定，基本上會預先填入 Default 的參數，如沒有要修改的項目，請使用預設的參數值來設定。



- 點選在 “Project Resources” 的 “mTouch” 圖示即可開啟設定對話視窗，如右圖所示。
- ”Esay Setup” 為主要的設定選項。在這項目下有 “mTouch” 的設定項，在 mTouch 下又有兩的設定項目: 一為 “Button and Proximity Configuration” 及 “AFA” 的設定項目。

Button and Proximity Configuration : 觸控按鍵及近接感應的設定。

AFA (Automatic Frequency Adaptation) : 觸控按鍵跳頻掃描的選項，減少外界雜訊的干擾。

點選 “Button and Proximity Configuration” 的設定

這時會進入 “Common Button Settings” (通用按鍵設定區) (如右圖所示)，在此設定區內在分成五個項目的設定，如下說明:

1. 建立新的觸控按鍵
2. 觸控資料的使用方式
3. 按鍵的溢時計數
4. 負電容恢復
5. 基準線的過濾

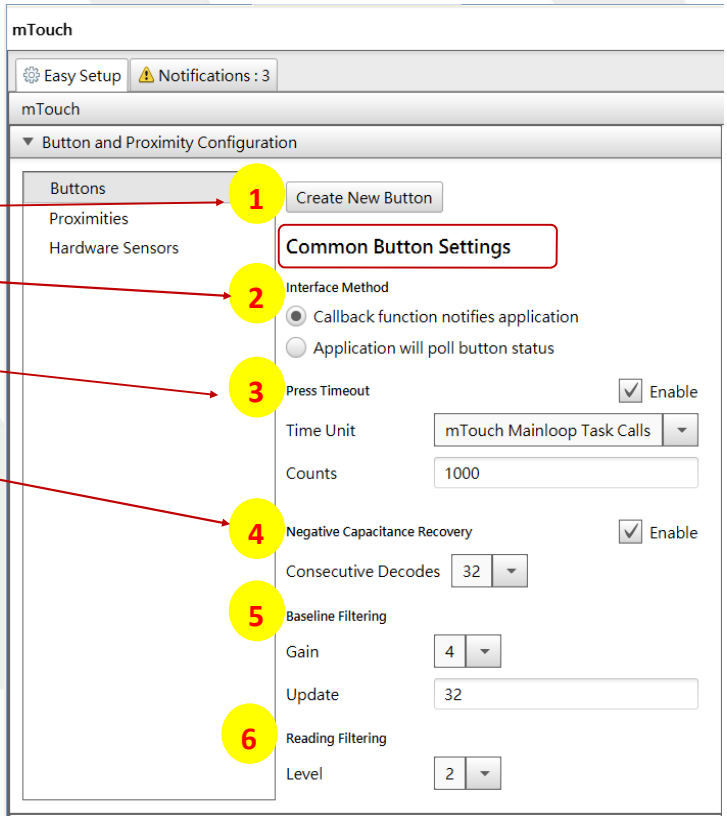


圖 2-1

底下將說明這五個設定項目的功能及其設定。

a. 建立新的觸控按鍵

“Create New Button” 通用按鍵設置區域中的選項，允許為 mTouch 項目建立或新增一個額外的觸控按鍵。當一個新的按鍵被建立時，內定的名稱將是 “Button #“，其中 # 將是當前未使用的最低號碼。

b. 觸控資料的使用方式

觸控所產生的的函數允許使用者的應用程式以底下的兩種方式獲得按鍵狀態:

- Callback function (回調函數) 必須開啟中斷功能
- Polling (輪詢方式)

回調函數方法比輪詢方法更有效率，但需要在觸控應用的主程式裡加入 callback 函數指標的宣告，這些函數也就會多加一些額外的程式空間。

輪詢模式皆可使用這在兩種選項，以提供簡易存取觸控按鍵的狀態

c. 按鍵的溢時計數 (致能選項)

觸控按鍵溢時的量測，是用來釋放被卡住按鍵的機制。如果一個按鍵被持續按下且超過連續計數器得設定值時，按鍵的狀態將被重置。

- Time Unit (單一選項)

  目前只有一個選項而已，也就是每當觸控按鍵被按下時，溢時計數器值會在主程式裡，每次主程式呼叫 MTOUCH_Service_Mainloop（）的函數時，內部的溢時計數器就會加一。

- Counts (溢時計數門檻值) 預設值為 1000

  設定溢時計數的門檻值。每個觸控按鍵的按下時的溢時計數器開始計數。當觸控按鍵的溢時計數器超過此計數門檻值時，觸控按鍵狀態將變為重置 (Reset)。

說明: 在此項目的預設值為 1000，也就是按鍵持續被按住時每當呼叫 MTOUCH_Service_Mainloop（）就會加一。當計數值累加超過 1000 時，按鍵狀態將會被釋放後再呈現按住的狀態。以 LED 方式來呈現就會看到 LED 先亮一陣子達到 1000 次後就開始熄滅再點亮，也就是呈現 LED 閃爍的現象

d. NEGATIVE CAPACITANCE RECOVERY (負電容恢復)

啟用此功能將允許快速恢復按鍵的基準線（如果為負壓電容被檢測到)。負壓電容意味著基準線的值大於按鍵的讀取值。

Consecutive Decodes (連續解碼)

連續解碼參數，指定連續解碼的數量次數（MTOUCH_Service_Mainloop（）的呼叫）在重設按鈕的狀態之前負壓電容程序必須檢測。

e. Baseline Filtering (基準線的平均)

觸控按鍵鈕的基準線會因 PCB Layout 線路的長短會造成每個觸控按鍵的基準線不同，基準線的值是與 Pad 寄生電容成正比的。觸控按鍵的走線越長或 Touch Pad 越大該觸控按鍵基準線的值就會比較大。基準線的值基本上是一個有條件的低通濾波器的輸出，這個低通濾波器的輸入是按鍵的原始讀數。這允許基準線跟踪並對諸如溫度和濕度的環境變化作出反應可自我校正基準線。一個按鍵進入按下狀態，基準線將停止更新，直到釋放。

Gain (增益)

濾波器增益參數決定了基準線濾波強度。使用較大的設定值時過濾器強度就變大，這也會造成較慢的基準線響應。

Update

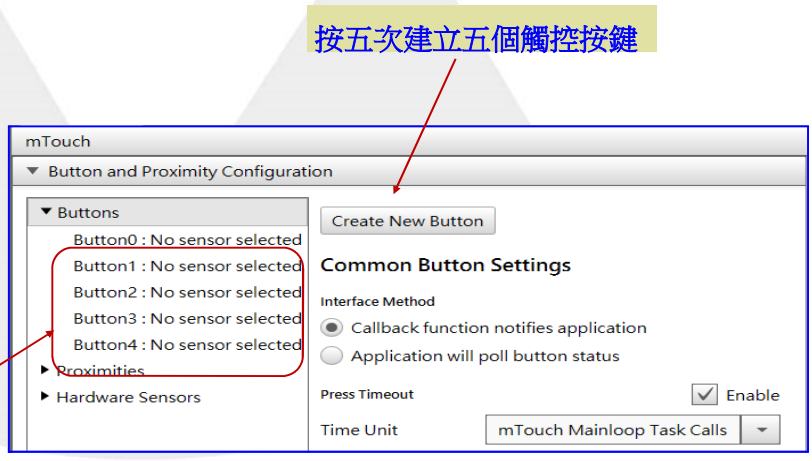
更新參數確定系統更新按鍵的頻率基準線。值越大，系統更新基準線的頻率越低。

f. Reading Filtering (讀取值的平均)

讀取觸控按鍵的讀值的取平均的次數 (雜訊過濾)，在這裡的內定值 Level 為 2。如果 Level > 2 的話取平均次數較多，雜訊干擾明顯變少但也犧牲了觸控按鍵響應的時間。除非是處在雜訊很大的場合可將 Level 提高，如果使一般使用的環境建議使用 Level 2 的設定。

8 建立五個觸控按鍵

接下來，開始建立觸控按鍵。在這裡有五個觸控按鍵需要設定，其步驟如下:

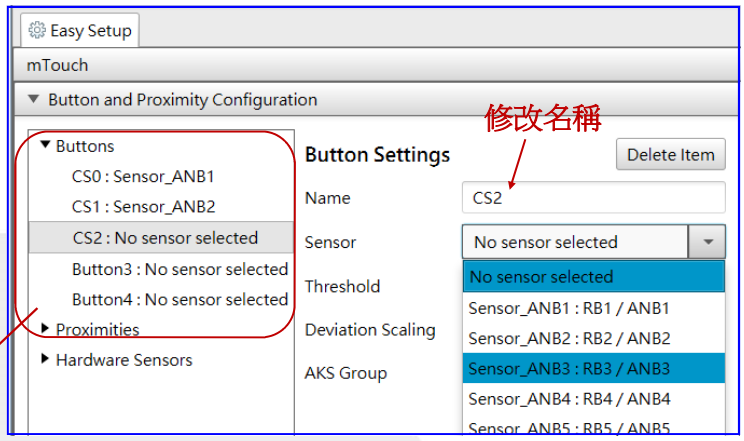


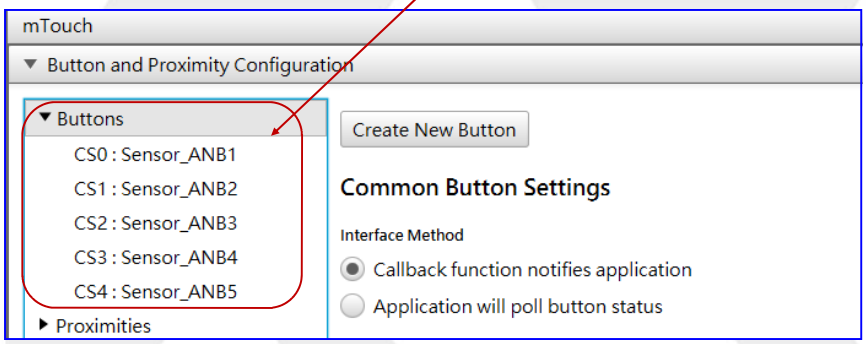
1. 按下 “Create New Button” 五次。每按一次即可產生一個觸控按鍵 Button 的設定項。
2. 先建立了五個按鍵，內定按鍵的名稱為 “Button0 ~ Button4” 共五個按鍵。
3. 接下來的步驟是賦予每個 Button 的使用名稱及所連接到的觸控感應接腳。

4. 點選 Button0 進到 Button0 的設定視窗。

5. 將內定的 Button0 名稱改成 CS0，並將 Sensor 選項裡設定連接到 RB0/ANB1 的腳位上。

6. 如下圖所示，一一將五個觸控按鍵設定完畢，名稱為 CS0 ~ CS4，分別連接到 RB1 ~ RB5 的腳位輸入。



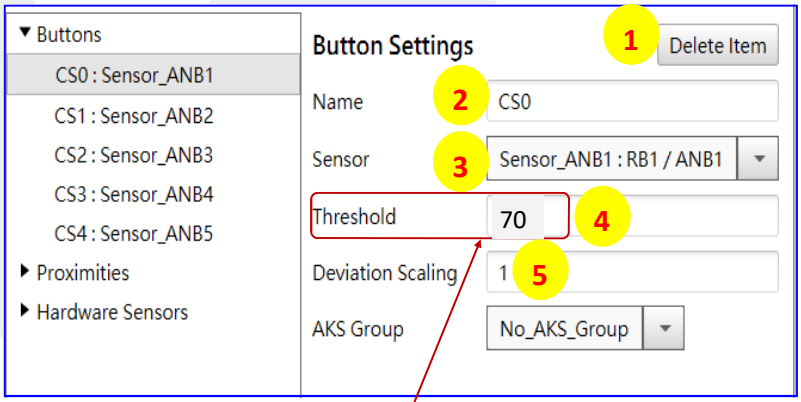


7. 確定完成五個觸控按鍵輸入並連接到適當的腳位上。

8. 接下來將設定每個觸控按鍵的參數。每個按鍵都有自己的獨立設置頁面。先點選 CS0 的按鍵設定視窗。

每個按鍵的設定由五個項目組成，如下圖所示：

1) 刪除按鍵 (Delete Item)
2) 按鍵名稱 (Name)
3) 按鍵的硬體觸控感測輸入腳位 (Sensor)
4) 按鍵偵測的的差異值 (Threshold)
5) 偏差縮放 (Deviation Scaling)



其中差異值 (Threshold) 及 Deviation Scaling 會因觸控感應點的大小，PCB 拉線的長短的不同而有所改變。在本練習裡因為 CS0 的觸控點最小，也意味著期感應電容的變化會比較小所以在這裡設定 CS0 的差異值為 70。CS4 的觸控點最大 (110) 其感應變化最大。實驗裡將 CS0 的差異值設成 70，其餘的差異值設定為 CS1 = 80, CS2 = 90, CS3 = 100。如此可削減觸控按鍵的寄生電容大小的影響。

底下就這觸控按鍵設定項裡的五種設定做些說明:。

1) 刪除按鍵

將本項目的按鍵將從 mTouch 項目中刪除。當太多的按鍵被加入時，可以選定的哪個按鍵項要被被刪除。

2) 按鍵名稱

此輸入名稱允許用戶修改按鍵的名稱。一個例子: 如一個媒體控制面板項目上的 “Play” 按鍵。該按鍵可以從原先內定的 “Button0” 重先命名為 “Play”。本練習使用 CS0 ~ CS4 做為觸控按鍵的名稱。

3) 按鍵的硬體觸控感測輸入腳位

選擇所要使用的硬體觸控感應輸入的腳位。硬體感測器可以設成觸控按鍵或是接近感測器。例如，如果用戶想要在同一物理感測器上實現接近度檢測和觸控檢測，這個感應器就可以同時為一個按鍵與接近感測器的設定。

4) 按鍵偵測的的差異值(Threshold)

觸控所測量到的信號差異值被定義為一個按鈕的讀值減去基準線值所得到的變化值。當觸控感應的訊號其變化值超過此差異值時就會承認有觸控按鍵輸入。差異值輸入允許用戶自行設定，其範圍從 1 ~ 127。差異值越大，較不容易被干擾但相對的感應距離就要比較近些；使用者可以依其應用場合及感應點的大小、觸控距離來調整此差異值。一般的使用差異值建議設定在 40 ~110 之間的值，以獲得完整的動態範圍。

5) 偏差縮放

將測量到信號值與基準線相減後所的到的變化值，因變化值動態範圍很大，所以將其值從 signed int 縮減到 signed char 的資料型別。因為有時因感應的輸入訊號值過大，如果縮小後的變化值仍大於 127，則該值將被限制在 127（0x7F）。偏差縮放是將變化值設定有多少位元將會被右移 (除法) 其設定值從 0 ~ 9 位元的右移。所以位移值越小，按鍵就越敏感。

假設 CS0 門檻值 = 60，CS1 門檻值 = 70。

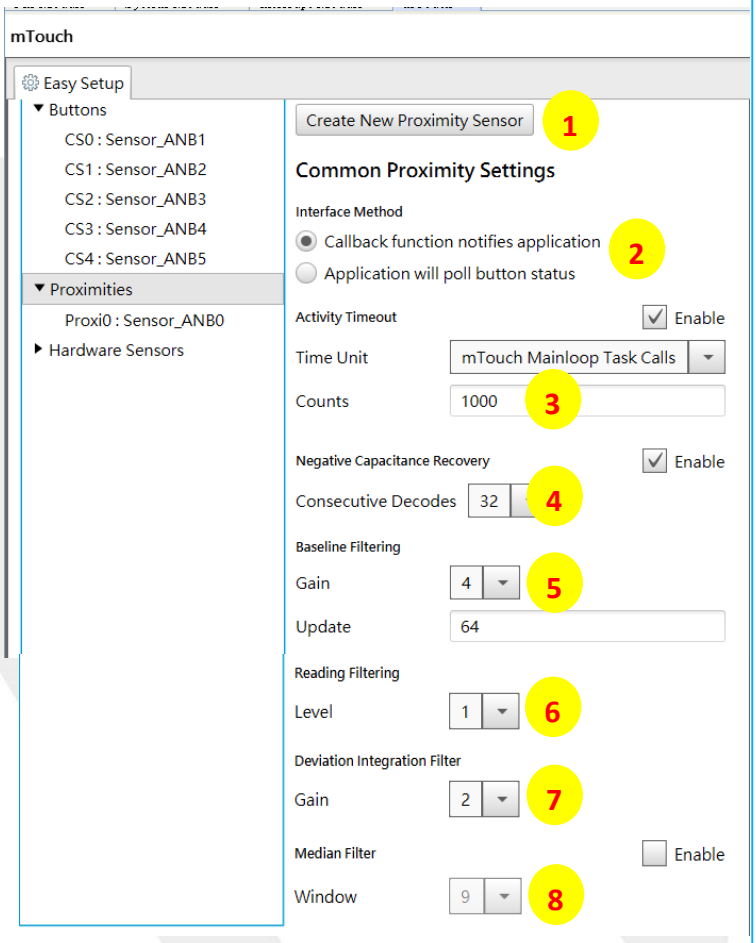
下圖所量測到的實際讀取值 (Reading0) 減去基準線(Baseline0) 所得到的變化值為 65，這變化大於所 CS0 所設定的門檻值 (Threshold) 60。所以在按鍵狀態 (Button Stat) 顯示 1，表示此觸控按鍵被判定是按下的狀態。第二個按鍵 Reading1 的變化為 9 並沒有超過 CS1 的門檻 (Threshold) 的設定值 70，所以顯示按鍵狀態為 0 (放開模式)

Button Data							
Reading0	8823	Baseline0	8692	Deviation0	65	Button Stat	1
Reading1	8048	Baseline1	8029	Deviation1	9	Button Stat	0

9 加入一個接近感應 “Proximity” 的功能

右圖所示為: 共同觸控感應感測器設置區域 (Common Proximity Setting) , 是由五個小設定區域所組成：

1. 建立新的接近觸控感應
2. 觸控資料的使用方式
3. 按鍵的溢時計數 = 1000
4. 負電容恢復 = 32
5. 基準線的過濾 ，Gain = 4, Update = 64
6. 讀取值的過濾，Level = 1
7. Deviation Integration Filter (偏差積分濾波) = 2
8. 中階濾波 (Disable)



底下只針對第一項目做說明，其他 2 ~ 5 項目請參考前頁面的說明。

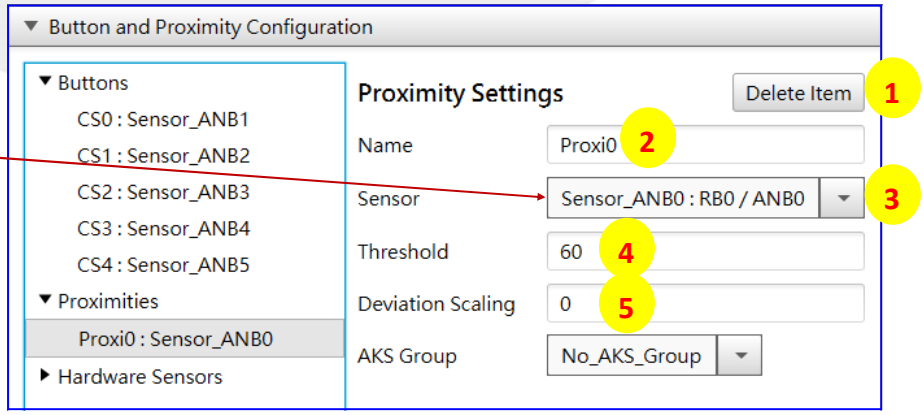
建立新的接近觸控感應

在 “通用接近感測器設置” 區域中建立新的接近感測器。可允許為 mTouch 項目建立一個附加的接近感測器。當一個新的接近感測器被創立後，內定的名稱將是 Proximity ＃，其中＃將是目前沒有使用的最低的數字。

10 接近感應 “Proximity” 裡的設定

mTouch 項目的每個接近感測器都會有自己的獨特設置頁面(如下圖所示)。該頁面也是由五個項目組成：

1. 從項目中刪除接近感測器
2. 接近感測器的名稱
3. 接近感測器連接的硬體腳位
4. 接近感測器的門檻值 = 60
5. 偏差縮放 = 0 (感度較大，接近感應距離較遠，= 1 時感測距離少一半)。

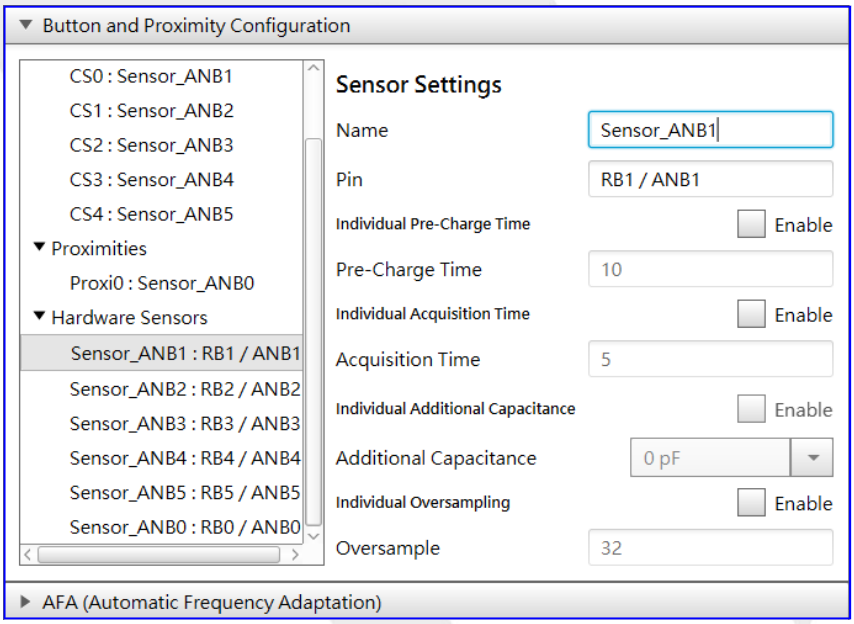


依硬體電路的需求，我們建立了一個名稱為 Proxi 0 的近接按鍵的設定並使用 RB0/AN0 的腳位。

到目前為止，我們已經建立了 CS0 ~ CS4 的觸控按鍵及 Proxi0 的近接偵測感應。

11 個別的觸控感應器硬體的設定（使用內定值設定）

每個硬體的觸控感應都有獨立的設定頁面。可單獨設定硬體觸控感應器的名稱及顯示該感測器硬體所連接的接腳。下列章節提供了有關這兩個組件的更多細節的說明，如下圖所示。



1. 名稱 (Name)

該輸入欄允許用戶修改所選硬體感測器的名稱。用戶可能想要加入包裝接腳的腳位以便於參考硬體電路。

2. 連接腳位 (PIN)

該項顯示該硬體感測器連接到哪個接腳。

3. 個別的取樣電容預充電時間 (Disable)

這個取樣電容預充電時間是在 Hardware Sensors (設定 Guard 時) 統一決定要設多少預充電的時間。

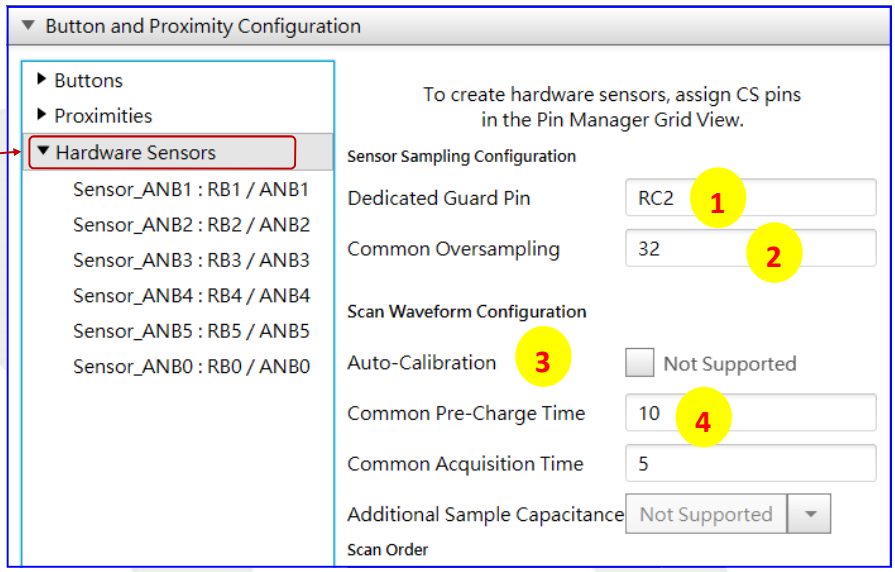
4. 個別的取樣時間 (Disable)

取樣時間也是統一由 Hardware Sensors 裡設定。

使用的是內建 10-bit SAR ADC 做電壓充放電的轉換，所以可以啟動對內部 ADC 取樣電容的預充電，及設定取樣時間。以上兩項所採取的時間單位是 Tad。在本實驗裡將不啟用這兩種設定。

12 Guard 防護輸出的設定

在所有的觸控感應及近接感應輸入都設定完後這時還有一項很重要的設定就是 Guard 的防護輸出的啟用設定。

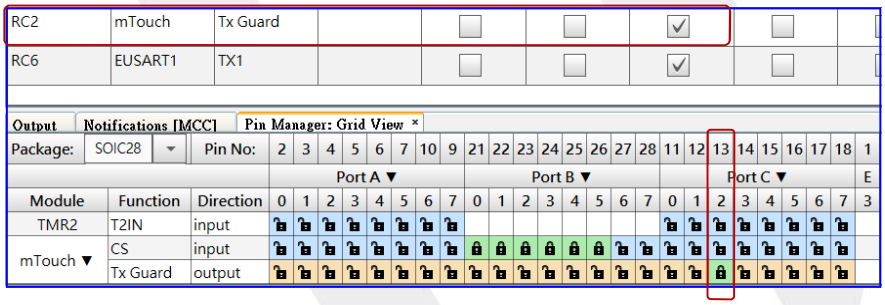


點選 “Hardware Sensors” 的選項後會出現 Guard 的設定畫面。

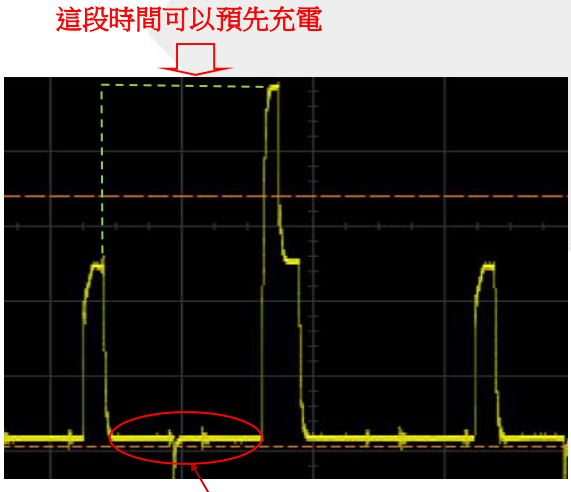
1. Dedicated Guard Pin : 決定 Guard 的腳位在這裡選用 RC2，這時檢視一下 “Pin Module” 的 RC2 是否有設成 I/O 輸出模式並關閉 Analog 輸入功能。

2. Common Oversampling : 共用的過度取樣

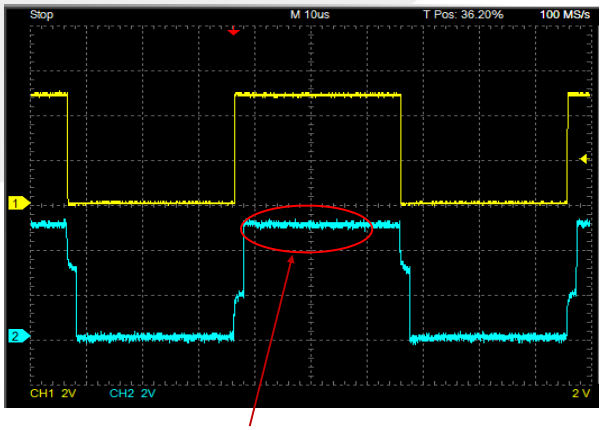
3. Auto Calibration: 有一很大的關鍵就是 Auto-Calibration (自動校正功能) 在這裡是 “不支援” 的。也就是說軟體 CVD 不支援此自動校正功能，因為校正功能要花掉很多 CPU 執行的效能，所以在此並不支援。但有 HCVD 及 ADCC 的硬體周邊元件就有支援此自動校正功能，執行 mTouch 功能相對就穩定多了。



4. Common Pre-Charge Time : 共用的預充電時間，單位是 Tad。使用這個選項可以縮短 mTouch 的整體時間，當正相偵測完畢他不必將 Pad 的放掉而是直接接到 Vdd 做充電後再做負向的偵測。



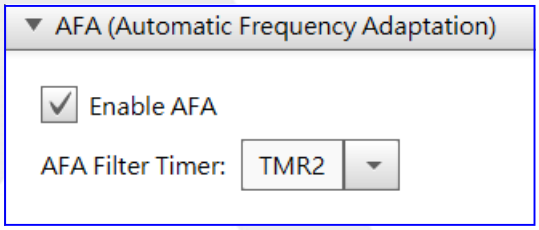
沒有預充電的波形輸出



啟用預充電的波形輸出

13 觸控按鍵掃描的跳頻設定

自動頻率調配（AFA）部分（下圖所示）允許用戶啟用先進的抗雜訊演算法。自動頻率調配程序不斷跟踪每個感測器上的雜訊量，並選擇一個新的智能掃描頻率。這個程序需要一個 8 位元的計時器來精確控制掃描頻率。每當自動頻率調配程序啟用，mTouch 將擁有最高優先所有權來使用這個 8-bit 的 Timer。以下說明提供了更多的細節 AFA 兩個組成部分。



Enable AFA

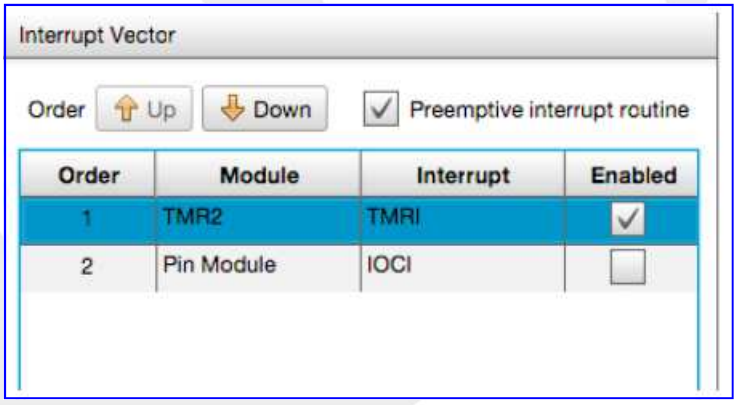
該選項決定是否包含自動頻率調配邏輯。

注意: 當啟用 AFA 功能時， AFA 需要額外的 Flash 和 RAM 記憶體容量，所以這個功能可能不適用於 RAM 有限的控制器。

AFA Filter Timer (AFA 濾波器定時器)

這個選擇決定了那個可用的 8-bit Timer 將被用於自動頻率調配邏輯。當選用了 AFA 功能所使用到的 8-bit Timer 將會是由 mTouch 函數所擁有。這意味著此 8-bit Timer 將不會顯示在可使用的周邊模組選項裡。在這裡啟用了 AFA 功能，並使用到 Timer2 作為頻率調配，這時要在中斷管理器裡須啟用 Timer2 的中斷功能。

在“Project Resources” 專案的選項裡，點選 “Interrupt Vector” 的選項即會出現右圖，勾選 TMR2 的中斷致能選項。



14 UART 的設定

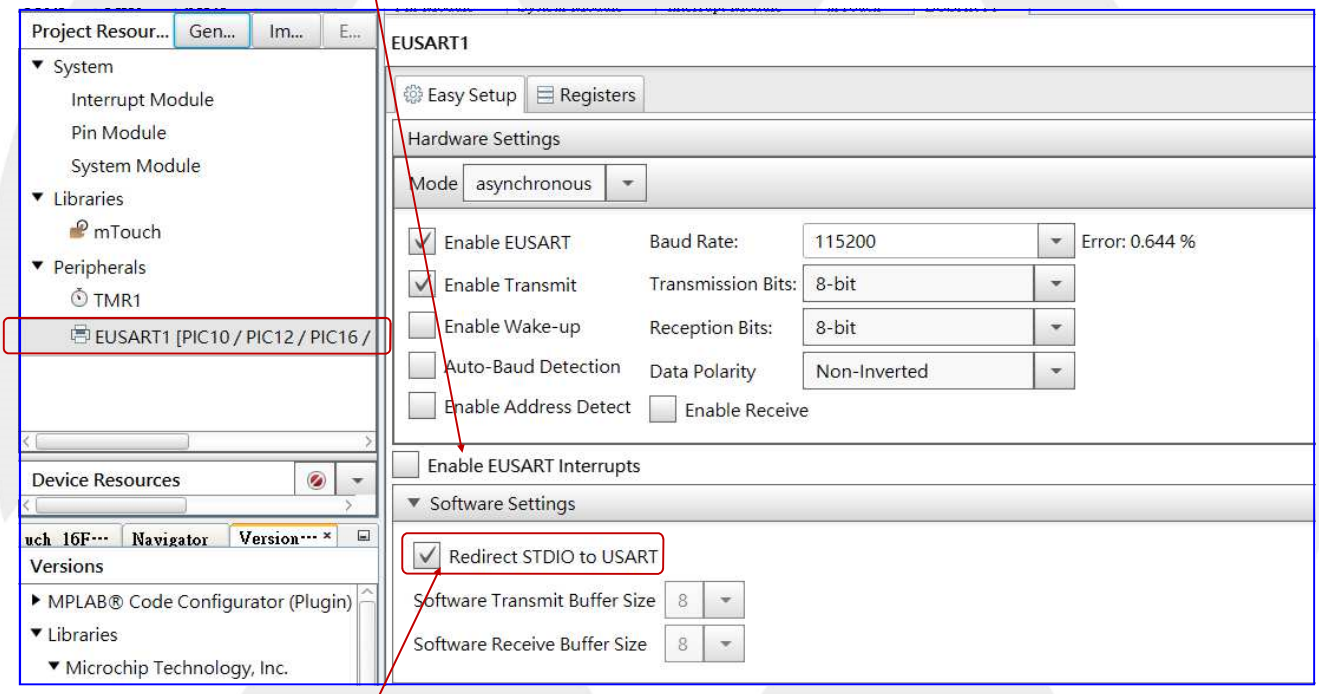
為了能監看觸控的資料並依實際的設計做 mTouch 的參數調整，所以我們使用 UART 將內的觸控資料送到一個 UART to USB 的裝置裡，透過虛擬通埠 (COM n) 由終端機程式 Tera Term 來顯示觸控的資料。

通訊協定為: 19200bps, 8-bit Data, Non parity, 1 stop

TxD 接腳 : RC6 (不使用中斷傳送模式)

RxD 接腳 : 沒使用到

UART 的設定視窗



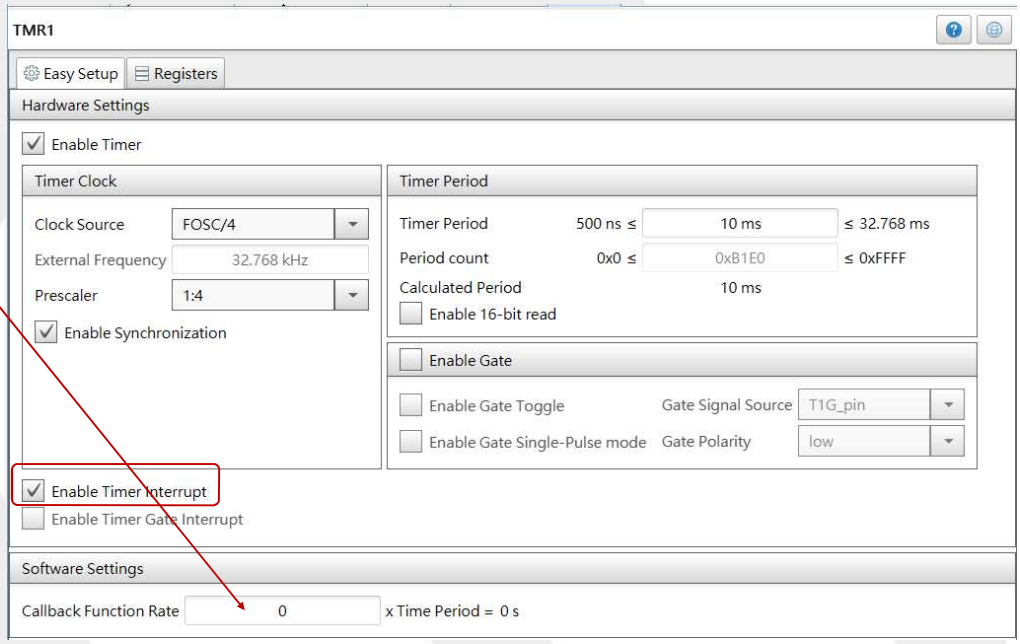
首先在 “Project Resources“ 加入 EUSART 的周邊模組後，開啟 UART 的設定視窗。如上圖的勾選及設定，在這裡我們不使用中斷傳輸方式 (當然你也可以啟用中斷傳輸模式)，主要的一項設定是勾選 Redirect STUDIO to USART 的選項，其意思就是將 UART 的輸出轉送到 STDIO 的標準輸出，也就是轉向到 printf( ) 這個函數裡。這樣我們就可以輕鬆使用 printf( ) 函數的功能來列印資料到 UART。還有一個函數是 sprint( ) 函數也常用到，sprint( ) 與 printf( ) 功能相似，sprint( ) 是將資料列印到 RAM 裡後再去出來使用。

15 Timer1 的設定

會使用到 Timer1 的功能主要是做為 UART 傳送的時間。因為使用終端機來監看觸控的資料，所以要控制每行傳送的時間及終端機畫面捲動的時間，在這裡的設定是間格 500mS 送出一行的顯示資料到終端機。然 Timer1 在 32MHz 的 Fosc 下是無法直接使用 500mS 的延遲。在這裡 Timer1 是設定 10mS 的計時，並啟用 Timer1 的中斷。

在這裡沒有使用到 Timer1 的 “Callback Function Rate “的功能。

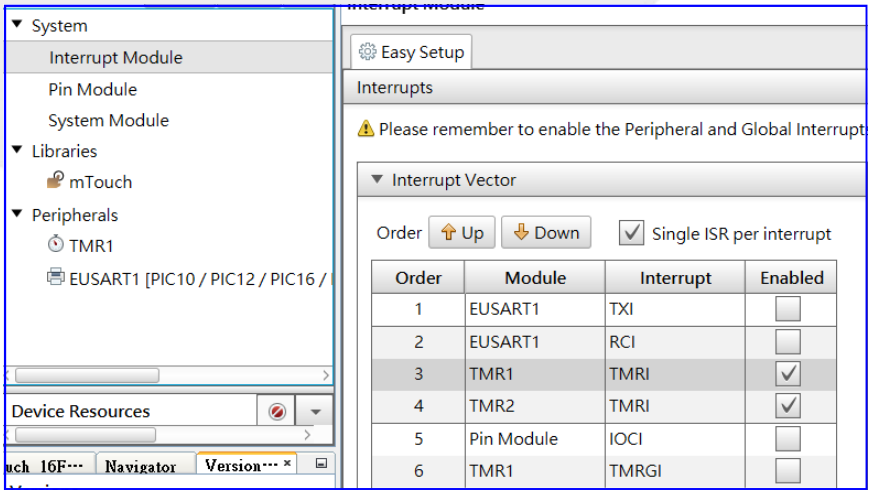
主要是 Timer1 只負責產生 10mS 的中斷計時，在中斷裡在使用中斷函數轉移的方式 (Callback) 交由主程式來做 500mS 的計時。有關 Interrupt Function Callback 的使用方式將在下一個練習做說明。



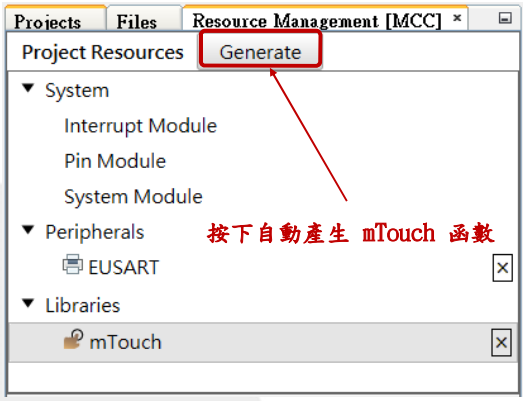
最後別忘了中斷視窗的設定。 開啟 “Interrupt Module” 在本練習只有兩項周邊會使用到中斷: 一是 Timer2 的 AFA 功能，二是 Timer1 的 10mS 計時功能。請在 Interrupt 設定視窗勾選 TMR1 & TMR2。

記住: 雖然我們在這裡勾選了啟用 TMR1 & TMR2 的中斷，這只是說 MCC 會依中斷使用方式來產生 TMR1 & TMR2 的函數。但 MCC 所產生的 main.c 裡 Global Interrupt (GIE) & Peripheral Interrupt (PEIE) 的內定功能是關閉的。所以說，只要有用到中斷就要在 main.c 的程式裡開啟中斷功能。

辛苦了，到目前為止該用到的周邊我們都設定完畢了!

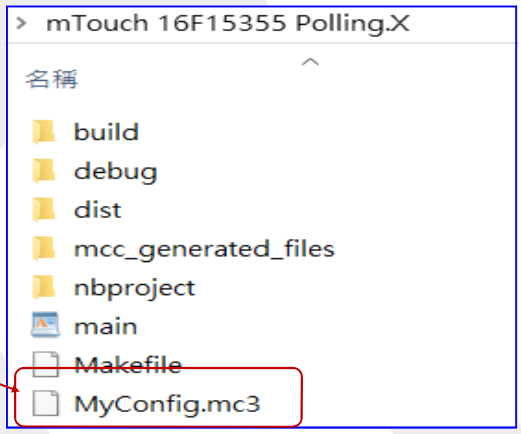


16 產生 mTouch 的函數



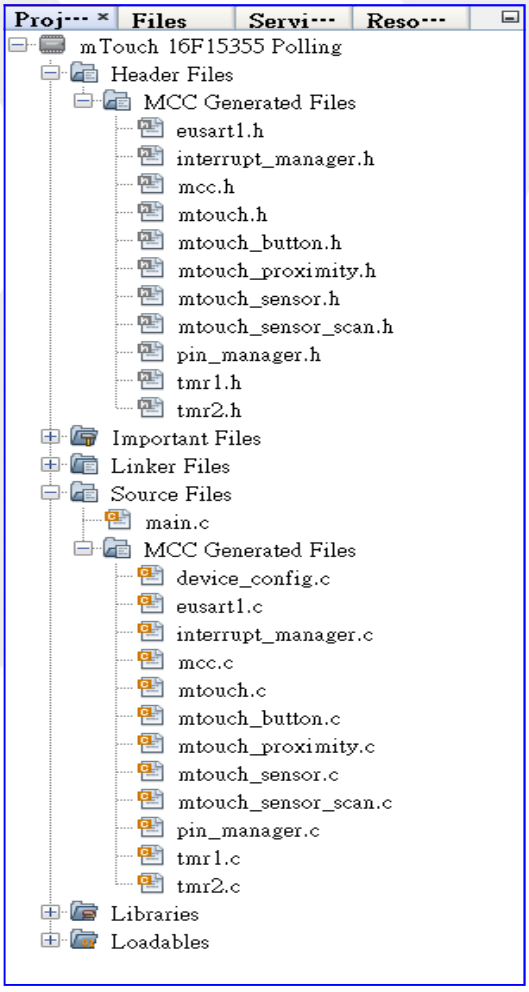
到目前為止，我們已經完成 mTouch 的設定項，這時回到 “Project Resources” 的視窗。請按下 “Generate” 就可以自動產生 MCC 所設定的周邊函數並加到專案裡。

注意: 每次修改過 MCC 的資料，就必須重新做 Code Generate 的動作。



MCC 也會將所有的設定存成一個名為 “**MyConfig.mc3**” 的設定檔，這檔案位置在 \mtouch\mTouch pic16F15355 Polling.X 下的 Project 的工作區裡。

這個專案除了 main.c 的主函數外，他也加入了在 MCC 裡所使用到的周邊函數庫並一一分類顯示如右圖所示。使用者可以找出相對應的周邊函數的原始程式來研讀其使用方式。當然的周邊函數也有相對應的標頭檔 (H Files)。



原則上由 MCC 所產生出來的周邊函數是不用再修改的，當然您要進去改成所需要的功能也是可以的。而 main.c 是我們所要使用的應用程式的主程式，接下來的章節將介紹如何改寫 main.c 實現觸控按鍵的功能。

第四章: 實現 PIC16F15355 觸控按鍵的應用

完成了 PIC16F15355 mTouch 在 MCC 的設定後也產生了周邊函數原始程式，接下來要討論的是如何使用這些觸控函數來撰寫應用程式。在這裡我們先暫不討論 MCC 所產生的周邊函數的內容。

還記的在設定 mTouch 時，在共同按鍵設定 (Common Button Setting) 選項時有兩個選項:

1. Callback function notifies application - 使用 mTouch 中斷方式，將主程式的函數指標傳給 callback 函數，這樣主程式就處理 callback 的資料，而不用進到 mTouch 的函數裡去做修改或轉移。使用 callback 的方法會是比較即時極有效率的方式 (下一個練習中說明)。
2. Application will poll button status - 主程式使用 polling 的方式來處理 mTouch 的資料。

如果設定使用 Callback 方式才會有 Callback 函數的產生，但同時也會有 Poll Button Status 的功能一起產生。所以兩種設定都可以使用 Polling 方式來寫成是。因為使用 Polling 方式比較簡單也容易說明，所以我們會先在練習一討論 Polling 的處理方式。

A. Polling 方式　　mTouch 16F15355 Polling.X 專案下的 main.c

```
/* 本範例程式使用最基本的詢問 (Polling) 的方式來展示 mTouch Software CVD 的功能。程式裡，每間隔 500mS 會在終端機顯示 Proxi0 、觸控按鍵 CS0 及 CS4 的基本資料，讓設計者可以調整出最佳的使用設定。本程式只是做示範用顯示出其基本的觸控按鍵功能供使用者作為設計上的參考使用 */
  Printf("%c[2J",0x1b);     // Clear Terminal Screen， 送出"Esc+[+2+J"
  Delay_Count = 50;
  Int I;
  while (1)
  {
    // 功能: 每間隔 500mS 送出 Touch 的資料到終端機顯示，其中時間的計算是交給 Timer1 的中斷函數來計算，中斷 時間為 10mS
    // UART 的通訊協定為: 115200bps, N, 8, 1
    //
    if (Delay_Count == 0) {
      printf("Proxi0= %3d  %5d    | ", MTOUCH_Proximity_Deviation_Get(Proxi0), MTOUCH_Proximity_Reading_Get(Proxi0));
      printf("CS0= %5d  %3d", MTOUCH_Button_Reading_Get(0), MTOUCH_Button_Deviation_Get(0));
      printf("     | ");
      printf("CS4= %5d  %3d", MTOUCH_Button_Reading_Get(4), MTOUCH_Button_Deviation_Get(4));
      printf("\r\n");
      Delay_Count = 50;              // 設定 500mS 的顯示時間 (50 x 10mS)
    }
```

應用程式為容易監看及調整 mTouch 各觸控按鍵的數值及參設，在這裡使用 printf( ) 將一些需監看的資料利用 UART 送給 (使用 115200, N, 8, 1 的協定) 一個 UART 轉 USB 的裝置 (PIC18LF25K50) 後，利用虛擬的 COM Port 傳給 Tera Team 的終端機顯示。底下為 mTouch 主要監看的的函數:

MTOUCH_Button_Reading_Get(n) 函數—取的目前按鍵 n 的讀值(-32768 ~ 32767)

MTOUCH_Button_Baseline_Get(n) 函數—取的目前按鍵 n 的基準線讀值(-32768 ~ 32767)

MTOUCH_Button_Deviation_Get(n) 函數—取的目前按鍵 n 的差異值(0 ~ 127)，即在各個 Button Setting 設定下的 的門檻值(Threshold) ，該值越大感應距離越近(最高 127)，值較小(< 40)感應距離較遠，但也較容易受干擾。

同樣的要監看接近感應的數值可以使用 MTOUCH_Proximity_Deviation_Get(n)，只要將上列函數的 Button 改成 Proximity 即可。以上的 mTouch 所提供的讀取函數可以在 mtouch_buttton.c 裡可以看到詳細的程式內容，可以使用 <Ctrl + Mouse Lift Key> 進入函數內層來追蹤。

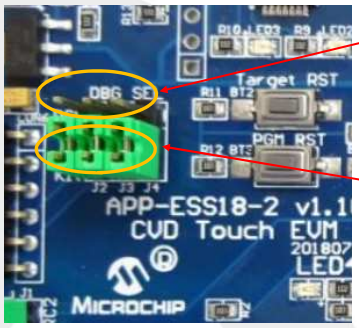
接下來，看一下 while(1) 迴圈下的 mTouch 判斷的種程式。記得所們針對每個按鍵所設定的門檻值嗎? 這門檻值可以針對每一個觸控按鍵的大小及拉線的長短來做調整，使每個觸控按鍵的觸摸感覺都一樣。MTOUCH_Button_Deviation_Get(CS0) >= Threshold 的判斷結果會直接反映到該觸控按鍵狀態的讀取函數 MTOUCH_Button_isPressed(CS0)) ，如為 1 表示 CS0 正被觸摸中。為 0 時，表示出控按鍵沒被觸摸。使用者也可以依實際環境的需要將這段按鍵判斷式改成動態的判斷行為。

```
 if (MTOUCH_Service_Mainloop());          // 叫用所有設定及啟用的 mTouch 按鍵的主循環服務函數
{
   if (MTOUCH_Button_isPressed(CS0))      // 檢查 CS0 按鍵是否按下?
            LED4_SetHigh();               // CS0 被按下了，點亮 LED4
   else
            LED4_SetLow();                // CS0 是放開的，熄滅 LED4

   if (MTOUCH_Button_isPressed(CS1))      // 檢查 CS1 按鍵是否按下?
            LED5_SetHigh();               // CS1 被按下了，點亮 LED5
   else
            LED5_SetLow();                // CS1 是放開的，熄滅 LED5
   if (MTOUCH_Button_isPressed(CS2))      // 檢查 CS2 按鍵是否按下?
            :
            :
}
{
   if (MTOUCH_Proximity_isActivated(Proxi0))  // 檢測接近感應的距離
            LED9_SetHigh();               // 距離低於門檻值，點亮 LED9
   else
            LED9_SetLow();                // 距離內沒有感應到，熄滅 LED9
}
```

PICKit3 程式碼的燒錄執行:

1. 使用 PICKit3 的除錯燒錄器。使用 PICKit3 是最便宜且快速的開發工具，先將 APP046 mTouch 板子上的 DBG_SEL (J2, J3, J4) 往下跳接至 PICKit3 燒錄的位置。如此，就可與一般 PICKit3 的除錯與燒錄的動作一樣。

2. 使用虛擬檔案的燒錄方式。板上的 PIC18LF25–K50 可以支援三種功能: USB to UART 及 虛擬檔案燒錄器。使用虛擬的檔案燒錄時必須將 DBG_SEL (J2, J3, J4) 三個 Jumper 短路到上方的 PGM 位置。 且需使用 Low Voltage Program-ming (LVP Mode) 方式燒錄



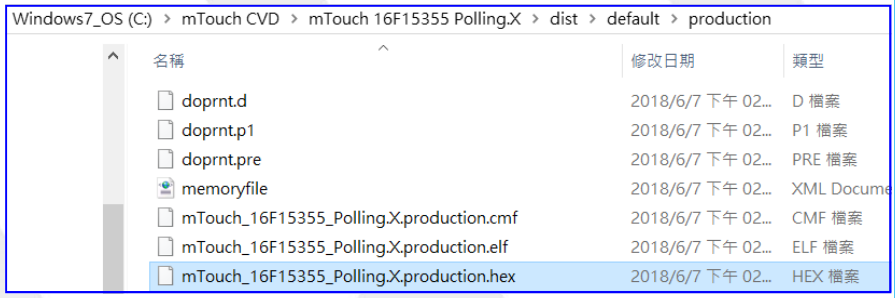
虛擬檔案的燒錄方法:

A. 燒錄檔案使用的是 Hex 格式的檔案，專案 “mTouch 16F15355 Polling.X” 經編譯成功後會產生一個 Hex 檔，尋找一下該 Hex 檔在哪?

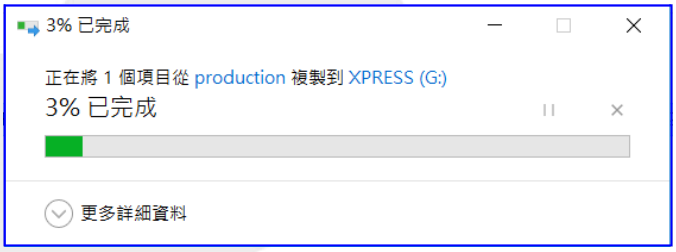
B. 開啟檔案管理員，檢視是否有多一個 XPRESS(F:) 的儲存裝置。

C. 將 “mTouch 16F15355 Poll-ing-production.hex” 貼入到 XPRESS(F:) 的目錄下立即出現一個複製的視窗並進行燒錄動作。待視窗結束關閉後，Hex 檔就易經完成燒錄，這時就可以做 mTouch 的測試了。

D. 程式有問題，回到 MPLAB X IDE 下重新修改、再編譯後燒錄來找出問題。當然也可以使用 PICKit3 來進行除錯的工作。



成功編譯後所產生的 Hex 檔案

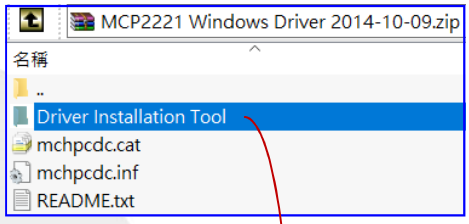


檔案傳輸與燒錄中

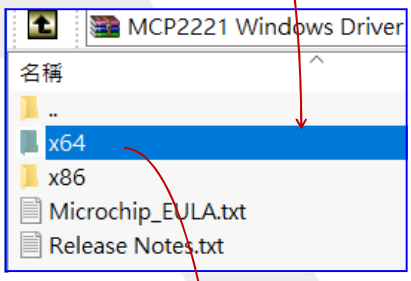
使用終端機監看程式 (Tera Term):

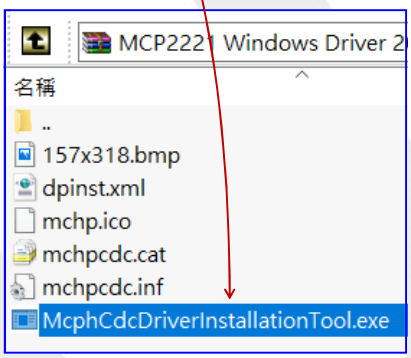
1. 建立虛擬的 Com Port :

Tera Term 是一個開放源免費的軟體，可實現的終端模擬器（通信）程序。它模擬不同類型的終端機從 DEC VT100 到 DEC VT382 的終端機，使用者可自行上網下載。



因為現行的電腦幾乎都沒有傳統的 COM Port (RS-232)，取而代之的幾乎都是 USB Port。所以我們也使用了一個 USB to UART 的轉換器 (PIC18LF25K50, U2) 將觸控資料送給 Tera Term 顯示。因為 Tera Term 是要 Com Port 的支援，所以第一次將 USB 連接到此板子時會需要安裝 CDC 的驅動程式。驅動程式可以在 www.microchip.com/mcp2200 的網頁內找到: MCP2200/MCP2221 Windows Driver & Installer 後下載並解壓縮。如果你是使用 64-bit 的 Windows 作業系統請到 x64 的目錄下執行，其它請到 x86 下的目錄下 “McphCdcDriverInstallationTool.exe” 來安裝此 CDC 的驅動程式。



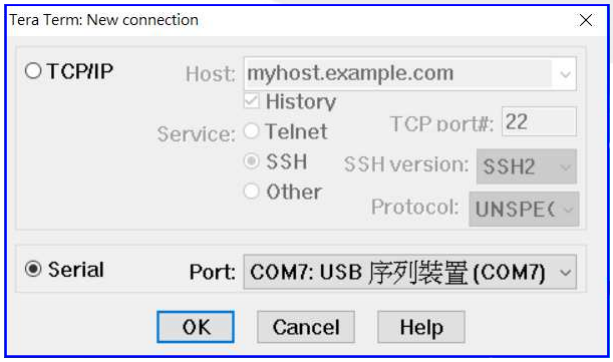


CDC 驅動程式安裝完畢後，這時的動作就是將 Tera Term 掛到 CDC 驅動程式所模擬的 Com Port。請一定要先連接 USB 線到 mTouch APP046 的實驗板後才可以開啟電腦的 “裝置管理員“。

(使用 W10 系統，請用老鼠右鍵點選左下角 Windows 的圖示 即可看到裝置管理員的選項。檢查一下裝置管理員下的 “COM 和 LPT” 下 ”USB 序列裝置” 是占用哪一個 COM Port，右側圖內顯示的是使用 COM7 當作虛擬的 Com Port (請記住此 Com Port 號碼)。

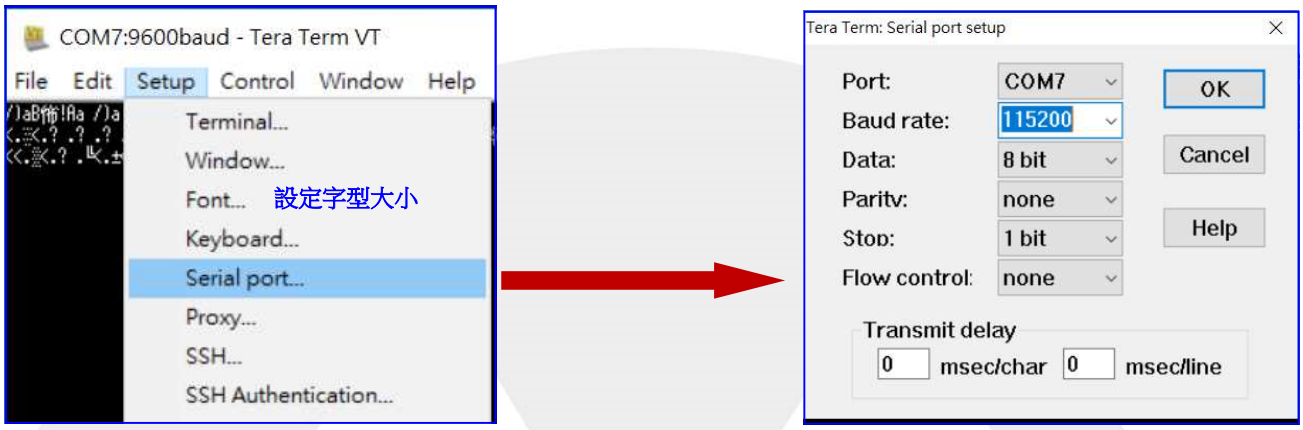
2. 設定 Tera Term 的工作:

啟動 Tera Term 後，選取 Serial Port 並選擇適當的 “USB 序列裝置的 COMx (這裡連接的是 COM7，這連接的號碼不一定會相同的)。



接下來須設定通訊協定及字型大小。

選擇 Serial Port ，在 COM7 底下設定 115200bps, 8-bit Data, None Parity and 1 Stop bit 的通訊協定。如果終端機顯示的字型大小，只要到 Font 的選項就可以調整所需的字型及大小。



到目前為止，Tera Term 已經做好顯示的準備，現在可以回到 MPLAB X IDE 下準備遍及燒錄程式。在這裡建議第一次編譯時，請使用 Build All 的方式來編譯。之後就直接使用編譯及燒錄圖示。

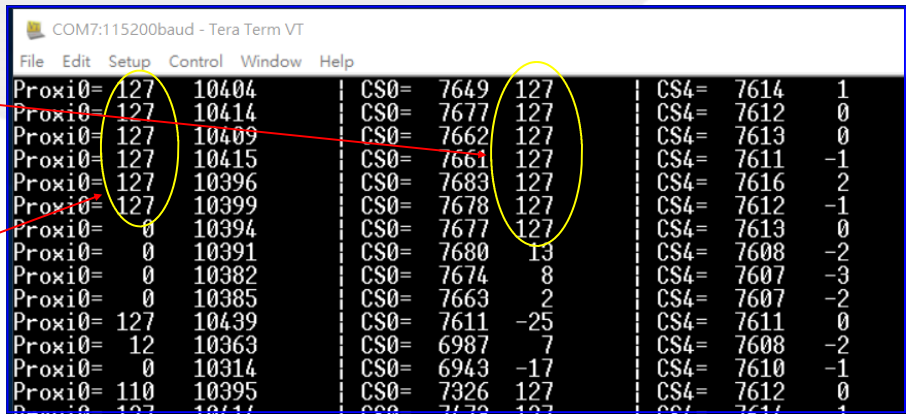
3. 程式的執行

專案的 main.c 修改完畢後，經編譯成功並透過 PICkit3 燒錄程式到 PIC16F15355 執行，由 UART 經 USB 送給 Tera Term 顯示觸控的資料。從右圖所示: CS0 在沒有觸摸的狀態下其 差異值莫約為 0 附近。當手去碰觸 CS0 的感應點時，其數值升高到最大的差異值 127。

因 CS0 的門檻值 (Threshold) 設定在 70 左右，所以板子上相對應的 LED4 也跟著一起亮。CS4 一直都沒有被觸摸到，所以其差異值一值都穩定的維持在 0 的附近。

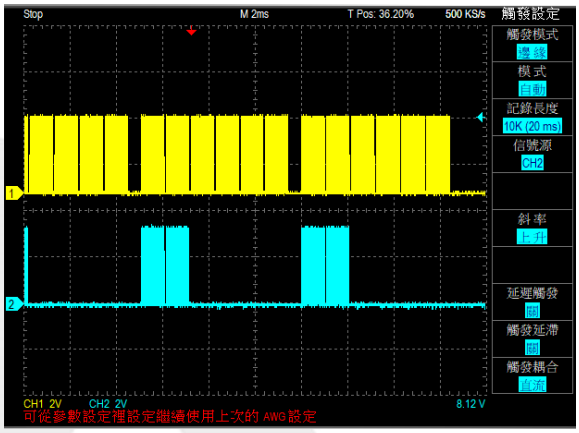
相對地的接近感應器的差異值是設成 60 (約 6 cm 的感度距離)

右圖終端機的資料顯示，當 CS0 被觸碰時期變化值增加到 127 這超過的差異值 70 的設定，LED4 被點亮。因為要觸摸 CS0，手指也會觸動到近接的感應器使 Proxi0 的變化值也升到 127。但 CS4 的變化值始終穩定的維持在 0 附近的變化。

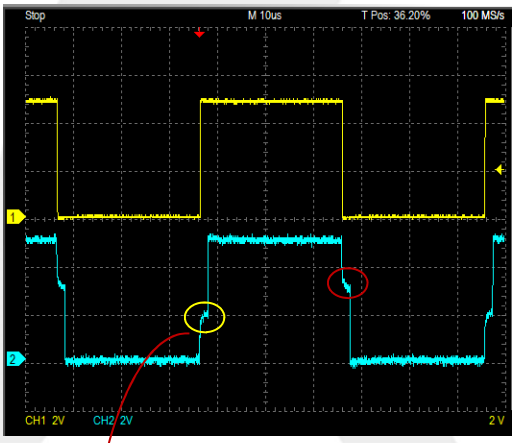


接著我們使用示波器來觀察實際軟體 CVD 的輸出波形。示波器的 CH1 是測量 Guard (RC2) 的輸出訊號，CH2 是測量 CS0 (RB1) 的觸控腳的波形。

右圖一可以看出總共有六個觸控按鍵的輸入正在被掃描中… Proxi0 -> CS0 -> CS1 -> .. ->CS4 周而復始的掃描。第一個波形群是測量 CS0 共量測兩次，每次的測量都會送出 16 次後開始取平均值。

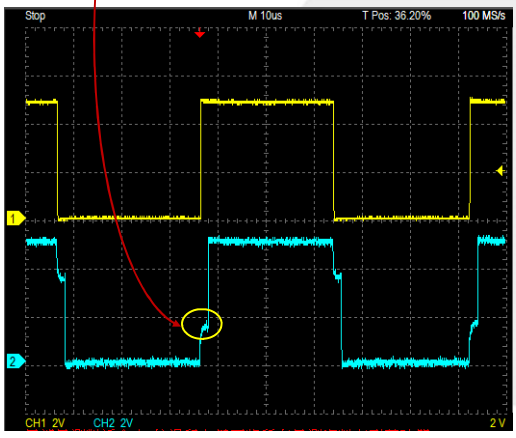


圖一



圖二

圖二是放大第一個觸控感測 CS0 的測量。可以明顯地看到先建立 Guard 的電場後，即開始做正向的觸控值的電壓量測，在 Guard 下降緣後立即做負向的電壓測量。這樣的作法可大大降低外界的雜訊干擾。



圖三

圖三是 CS0 是在被觸摸的情形下的波形，可以比較圖二的測量點的電位。因為有觸摸關係寄生電容變大，電荷取的平衡後的的電壓在正向的觸控值變小了；而負向的電壓值升高了。基於這種一降低、一升高的變化，使用 ADC 做量測時做差異值的演算後取平均的比較，就可得知觸控按鍵的狀態了。

B. 使用 Timer1 callback 函數方式

在練習一的程式 (mTouch 16F15355 Polling.X) 有使用到 Timer1 的 Callback 的中斷回調功能來做 500mS 的計時。Callback 的中斷回調功能對熟悉 C 的工程師是駕輕就熟的工作，可是對初學 C 的工程師就是件辛苦的事了。MCC 或 HMC 所產生的周邊函數很多都具有 Callback 的功能，對於 MCC 所產生的周邊函數我們也是儘量的不去修改或增加程式到 MCC 所產生的程式裡，如此可在重新產生新的 MCC C Code 時可避免忘記修改或復原造成功能執行的錯誤。也就是說不用去修改 MCC 的 Code，只要修改自己的主程式即可。

底下就所使用的 Timer1 Callback 方式做個說明:

1. 查看 Timer1 的 tmr1.h 裡的函數原型宣告

   void TMR1_SetInterruptHandler(void (* InterruptHandler)(void)); 這是一個用來註冊在中斷期間所要回調函數的函數, (以函數指標型式傳入)

2. 接下來要做的是: 撰寫一個在主程式下的一個中斷處理函數 (MyEvent)。

```
void MyEvent(void) {
if (Delay_Count !=0) Delay_Count--;    // 終端機顯示計時器，每 10mS 遞減
}
```

   程式功能非常簡單，每次 Timer1 發生中斷就會到 MyEvent 函數來執行，將目前的 Delay_Count 減一。

3. 加入該函數 MyEvent( ) 的原型宣告及 Delay_Count 的變數宣告。要注意的是 Delay_Count 是在中斷期間所使用到的變數，所以不可忘記要加上 volatile 的宣告。

```
int volatile Delay_Count; // Timer1 使用到的變數
void MyEvent(void);
```

4. 最後的工作就是將 MyEvent 的函數指標註冊到前面所提的 TMR1_SetInterruptHandler(void (* InterruptHandler)(void))裡。只要在程式一開始執行後就可以加入 TMR1_SetInterruptHandler (MyEvent); 的註冊工作。

以上的工作一一完成後，Timer1 Callback 功能就可以實現了。

C. 使用 mTouch 的 callback 函數方式 : Lab2 (mTouch 16F15355 Callback.X)

使用 callback 函數的方式比較複雜，也會增加程式的大小，但無可否認的是使用 callback 方式反應速度更快具有較高的處理效率。Callback 方式為讓使用者不用去詳看 mTouch 所產生的函數即可使用觸控的資料。在這裡只要將你要處理觸控按鍵功能的函數指標傳給 callback 即可。

可是查詢這使用方式只有在 mtouch.h 的檔案裡有提到怎樣使用 callback 功能。底下是摘錄其說明的內容:

> Use the 'Set Callback' API to provide the function pointer to your application's function:
> void MTOUCH_Button_SetPressedCallback (void (*callback)(enum mtouch_button_names button));
> void MTOUCH_Button_SetNotPressedCallback (void (*callback)(enum mtouch_button_names button));

以上這短短的使用說明，我想軟體高手一看便知如何使用。但普羅大眾看了這段使用說明後應滿肚疑惑不知如何下手? 沒關係，這裡將一一做說明。

首先看底下的原型宣告。

```
void MTOUCH_Button_SetPressedCallback  (void (*callback)(enum mtouch_button_names button));
```

A. MTOUCH_Button_SetPressedCallback (函數指標) : 這是在 mTouch 的 callback 主執行函數。要讓這函數可以跳到主程式去執行 觸控按鍵的動作，也為了程式的可攜性。在這裡只需傳入在主程式裡處理觸控按鍵的函數指標。

B. *callback : 主程式裡處理觸控按鍵的函數指標。如此 callback 跳到這應用函數來執行。這裡傳入的參數是一個指標，也就是該應用程式的執行位址。

C. enum mtouch_button_names ：列舉宣告。觸控按鍵的列舉宣告。在這裡可以看到宣告了兩個觸控按鍵；定義的名稱為 CS0 & CS1。

button : 傳入主程式裡處理觸控按鍵的函數的變數，其內容為 (CS0 或 0) 或 (CS1 或 1) 。

在 mtouch_button.h 下的宣告

```
#define MTOUCH_BUTTONS 2
  enum mtouch_button_names
  {
    CS0 = 0,
    CS1 = 1
  };
```

接下來使用 callback 的方式來撰寫觸控的應用程式。這裡有三個步驟須完成的：

1. 觸控應用函數的原型宣告。
2. 在進入 while（1）迴圈之前註冊 callback 函數，當任何按鈕/接近處於觸摸或釋放狀態時，相對應的觸控應用函數將會以指標方式傳入被 callback 呼叫 。
3. 觸控應用函數的撰寫。

底下就以 mTouch 16F15355 Callback.X 的專案的 main.c 做個說明：

mTouch 16F15355 Callback.X 專案下的 main.c

```
#include "mcc_generated_files/mcc.h"
#include <stdio.h>
void processButtonTouch (enum mtouch_button_names);   // callback 的觸控按下函數雛型宣告
Void processButtonRelease(enum mtouch_button_names);   // callback 的觸控放開函數雛型宣告
int  volatile Delay_Count;

/*                      Main application                      */
void main(void)
{
   // initialize the device
   SYSTEM_Initialize();

   // When using interrupts, you need to set the Global and
   // Peripheral Interrupt Enable bits
   // Use the following macros to:

   // Enable the Global Interrupts
   INTERRUPT_GlobalInterruptEnable();

   // Enable the Peripheral Interrupts
   INTERRUPT_PeripheralInterruptEnable();

 // 將應用端的 processButtonTouch() 函數的指標傳給 mTtouch 函數庫裡的 MOTUCH_Button_SetPressedCallback() 函數。
 //  當 mTouch 相對應的中斷發生之後即會呼叫 processButtonTouch() 的函數執行。
  MTOUCH_Button_SetPressedCallback   (processButtonTouch);

  // 將應用端的 processButtonRelease() 函數的指標傳給 mTtouch 函數庫裡的 MOTUCH_Button_SetNotProcessedCallback()
  //  函數。
  // 當 mTouch 相對應的中斷發生之後即會呼叫 processButtonRelease() 的函數執行。
   MTOUCH_Button_SetNotPressedCallback(processButtonRelease);

   printf("%c[2J",0x1b);      // Clear Terminal Screen ， 送出"Esc+[+2+J"

   Delay_Count = 50;
   int i;
```

1

2

Interrupt Vector

Order Up Down Preemptive interrupt routine

Order	Module	Interrupt	Enabled
1	EUSART	TXI	☐
2	EUSART	RCI	☐
3	TMR2	TMRI	☑
4	TMR1	TMRI	☑
5	TMR1	TMRGI	☐
6	Pin Module	IOCI	☐

```
while (1)
  {
    // 功能: 每間隔 250mS 送出 Touch 的資料到終端機顯示，其中時間的計算是交給 Timer1 的中斷函數來計算，中斷時
    //       間為 10mS
     if (Delay_Count == 0)
    {
       printf("CS0= %5d   %3d", MTOUCH_Button_Reading_Get(0),MTOUCH_Button_Deviation_Get(0));
       printf("    ");
       printf("CS1= %5d   %3d",MTOUCH_Button_Reading_Get(1),MTOUCH_Button_Deviation_Get(1));
       printf("\r\n");
       Delay_Count = 25;
    }

    (MTOUCH_Service_Mainloop());          // 叫用所有設定及啟用的 mTouch 按鍵的主循環服務函數
     //
     // 將應用程式寫在這裡
     //
  }
}
void processButtonTouch (enum mtouch_button_names bt)  // 應用端所建立的 Callback 按鍵按下時的轉移函數，傳入按鍵
                                                       // 列舉型別變數 bt
{
  switch (bt)                  // 判斷是哪一個觸控按鍵動作了?
  {
    case CS0: LED4_SetHigh();     // CS0 被按下，點亮 LED4
    break;
    case CS1: LED5_SetHigh();     // CS1 被按下，點亮 LED5
    break;
        :
    default: break;
  }
}

void processButtonRelease(enum mtouch_button_names bt)  // 應用端所建立的 Callback 按鍵放開時的轉移函數，傳入按鍵
                                                        // 的列舉型別變數 bt
{
  switch (bt)                  // 判斷是哪一個觸控按鍵放開了?
  {
    case CS0: LED4_SetLow();      // CS0 已放開，熄滅 LED4
    break;
    case CS1: LED5_SetLow();      // CS1 已放開，熄滅 LED5
    break;
        :
    default: break;
  }
}
```

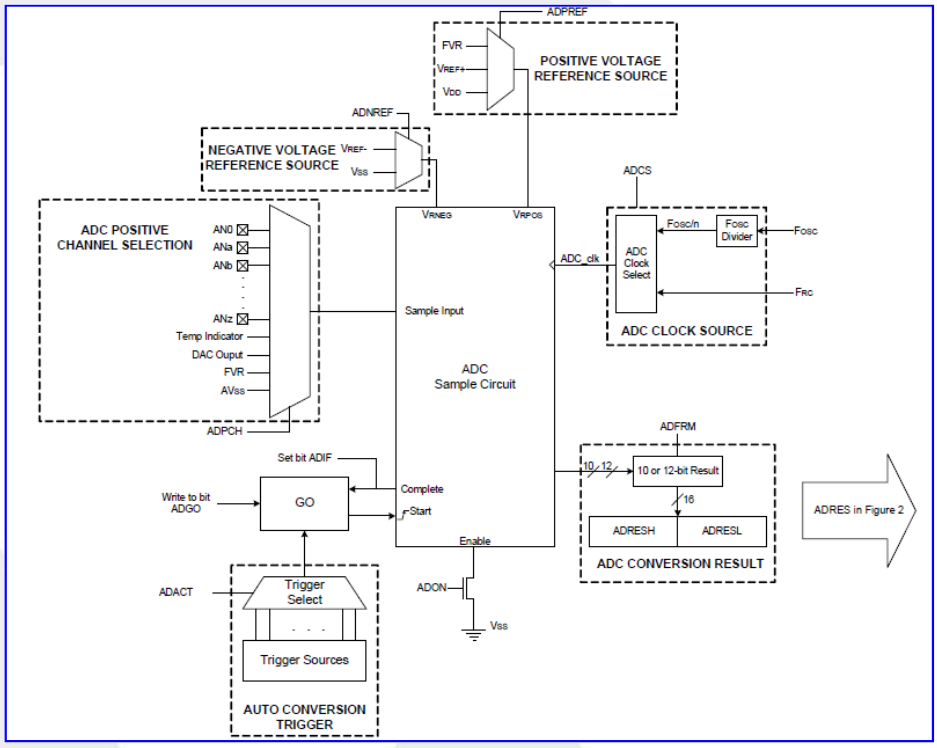
3

第五章: 使用 ADCC 硬體周邊的觸控按鍵應用

前言:

為了實現更簡單、更穩定的觸控按鍵應用，Microchip 推出了一個新的類比置數位轉換的周邊並帶有硬體的計算功能的周邊，一般簡稱 ADCC 或 ADC^2 (Analog-to-Digital Converter with Computation)。這新周邊 ADCC 主要有兩個區塊: 一是強化功能後的 ADC ，及後段的計算處理負責轉換完成的後段資料處理(如: 自動處裡輸入資料的平均，濾波器的計算，過取樣及比較)。ADC 部分功能仍為一般的 10-bit 解析度，但後段的計算模組功能就可以大大降低 MCU 的執行耗損，增加整體處件的優越表現。

右圖為 PIC16F18855 的 ADC 轉換模組的部分。看上去幾乎每一隻 I/O 接腳都可以規劃成類比 ADC 的輸入腳。參考電壓設定，ADC clock 的來源設定基本上都沒有特殊的變化。但在自動觸發轉換 (Trigger Convert) 增加到 10 幾個觸發來源的選擇，凡舉所有的 Timer，CLC ，OC 等都可以觸發 ADC 來做轉換。

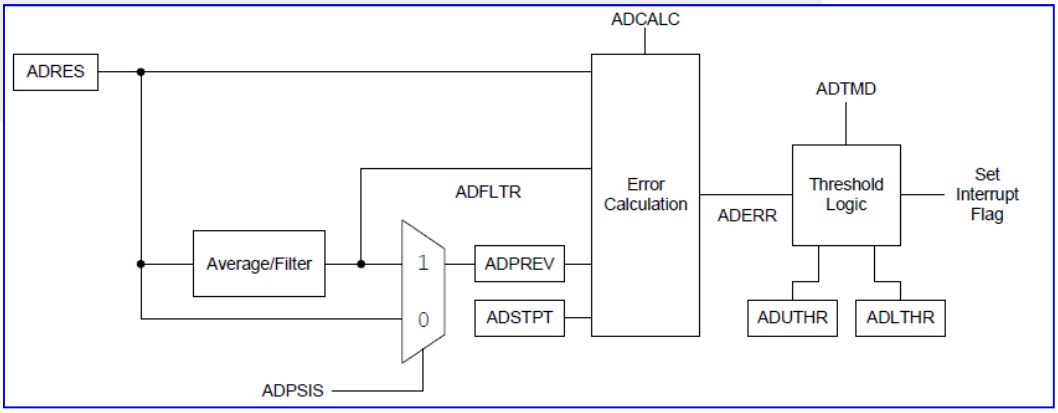


PIC16F18855 的 10-bit ADC 方塊圖

右圖是 PIC16F18855 的 ADC 計算部分的方塊圖。在這個計算功能模式可支援底下幾種計算模式:

1. 基本模式
2. 累積模式
3. 平均模式
4. 突發平均模式
5. 低通濾波器模式

基本模式下就是一個傳統 ADC 操作。不會使用到累積器功能，可視為一般 10-bit 的 ADC 傳統轉換功能。



PIC16F18855 的 ADC 計算方塊圖

第二種模式為: 累積模式。這是一種常用的取平均模式。將轉換後輸出的 10-bit 或 12-bit 的數值透過一個 ADCC 的暫存器累加到 16-bit 或 18-bit ，其輸出可以做平均或濾波後輸出。

接下透過一個圖表來說明這累積模式的功能。首先我們還是要知道一些內部暫存器的設定功能。

ANCNT : 累積次數的計數器 (0 ~ 256)。

ADCRS : 將累積器的數值右移幾個位元。如圖 ADCRS = 3，表向右移 3 bit 的設定 (除 8)。

ADRES : ADC 轉換後的輸出暫存器。

ADFLTR : ADC 低通濾波暫存器。

下圖表，為累積模式的數值輸出的說明表。總共做了 8 次的轉換，ADCNT 顯示轉換的計數值 = 8。ADCRS = 3 表示累積後的數值向右移 3 bit 也就是除 8 後輸出。

範例圖表中，ADC 輸入電壓為 2.5V，ADC 轉換輸出為 514 (ADRES 暫存器)。累積器的值是將 ADRES 的輸出加總起來 (514, 1026, 1542 ….) 而 ADFLTR 低通濾波器則會將 ADACC 的值右移 3 bit 後輸出。

514 >> 3 = 64,　1028 >>3 = 128 直到累計轉換到 8 次後，低通濾波輸出恢復正常。

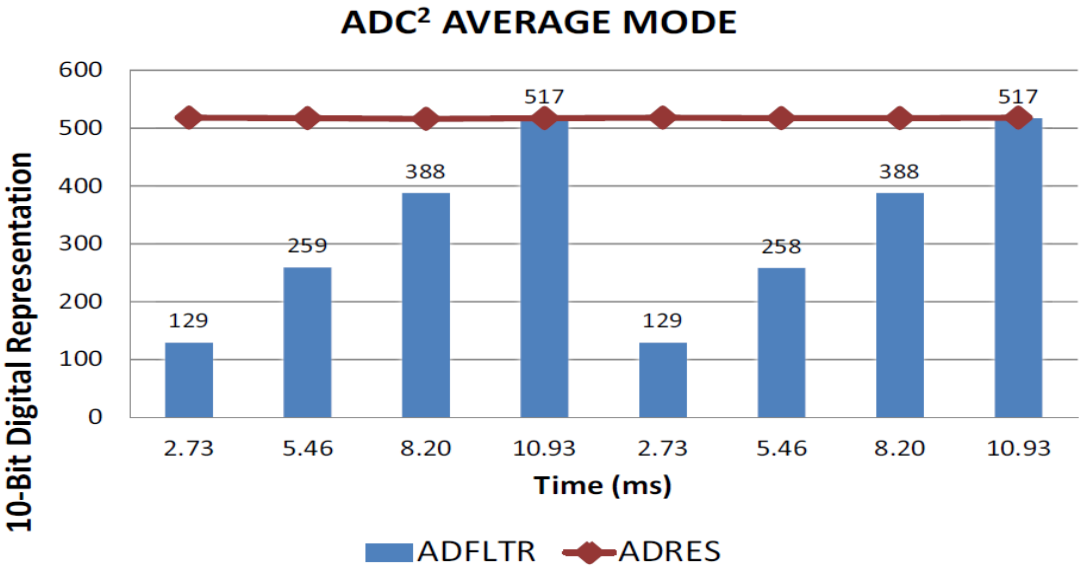
ADCNT	ADCRS	VINPUT	10-bit ADRES	16-bit ADACC	16 bit ADFLTR
1	3	2.5V	514	514	64
2	3	2.5V	514	1028	128
3	3	2.5V	514	1542	192
4	3	2.5V	514	2056	257
5	3	2.5V	514	2570	321
6	3	2.5V	514	3084	385
7	3	2.5V	514	3598	449
8	3	2.5V	514	4112	514

累積器模式的範例說明表

接下來要介紹的是平均模式(Average Mode)，在這裡會使用到 ADRPT (ADC Repeat Setting Reg.) 暫存器的設定資料平均的重複計數值。如下表所示: ADCRS 設定累積器 ADACC 向右移 2 bit 後輸出到 ADFLTR 暫存器。ADRPT 設定重複值為 4 ，也就是累積器累加四次後歸零再重新累加，這時只要取出 ADFLTR 的值就是經四次平均後的值。

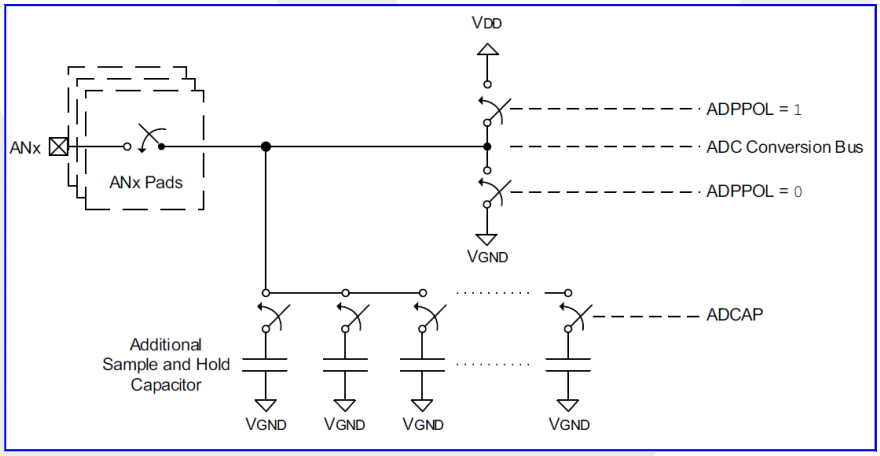
ADACC 經四次累加後期值為 2069，經平均後 ADFLTR 輸出為 517。

ADCRS	ADRPT	SAMPLE	ADCNT	10-bit ADRES	ADACC	ADFLTR
2	4	0	1	518	518	129
2	4	1	2	518	1036	259
2	4	2	3	516	1552	388
2	4	3	4	517	2069	517
2	4	4	1	518	518	129
2	4	5	2	517	1035	258
2	4	6	3	517	1552	388
2	4	7	4	518	2070	517
2	4	8	1	517	517	129
2	4	9	2	516	1033	258



在這裡只介紹三種模式 Basic, Accumulator 及 Averaging Mode，另外兩種 Burst Average Mode & Low-Pass Filter Mode 請參考技術手冊: TB3146 Analog-to-Digital Converter with Computation Technical Brief

接下來要說明的是 ADCC 周邊針對 CVD 所加強的功能。傳統的 ADC 其內部的取樣電容容量是固定的，但在 ADCC 模組裡是使用多個電容器的並聯選擇作為分壓器的元件電路。圖 3 顯示了取樣電路的方塊圖。新設計的 ADCC 硬體 CVD 取樣電容部分是可以依外部寄生電容量來對取樣電容的大小作選擇的。右圖為硬體 CVD 的取樣方塊圖，其中取樣電容可以透過 ADCAP 暫存器從基本的 Chold 容量 10pF 的基本容量開始可以額外增加 1pF ~ 31pF 的容量設定。此外 Chold 也可以透過 ADPPOL 位元來設定其充放電的動作。



ADCC 為 CVD 所設計的取樣電容器

在本實驗裡，我們使用新一代的元件 PIC18F18855 。這個號稱全功能的微控制處理器有 28-pin 到 44-pin 的腳位，內建的周邊如下表所示:

PIC16(L)F188XX Family Types

Device	Data Sheet Index	Program Flash Memory (Words)	Program Flash Memory (KB)	EEPROM (bytes)	Data SRAM (bytes)	I/O Pins(1)	10-Bit ADC2 (ch)	5-Bit DAC	Comparator	8-Bit (with HLT)/ 16-Bit Timers	SMT	Windowed Watchdog Timer	CRC and Memory Scan	CCP/10-Bit PWM	Zero-Cross Detect	CWG	NCO	CLC	DSM	EUSART/I2C/SPI	Peripheral Pin Select	Peripheral Module Disable
PIC16(L)F18854	(1)	4096	7	256	512	25	24	1	2	3/4	2	Y	Y	5/2	Y	3	1	4	1	1/2	Y	Y
PIC16(L)F18855	(2)	8192	14	256	1024	25	24	1	2	3/4	2	Y	Y	5/2	Y	3	1	4	1	1/2	Y	Y
PIC16(L)F18856	(3)	16384	28	256	2048	25	24	1	2	3/4	2	Y	Y	5/2	Y	3	1	4	1	1/2	Y	Y
PIC16(L)F18857	(4)	32768	56	256	4096	25	24	1	2	3/4	2	Y	Y	5/2	Y	3	1	4	1	1/2	Y	Y
PIC16(L)F18875	(2)	8192	14	256	1024	36	35	1	2	3/4	2	Y	Y	5/2	Y	3	1	4	1	1/2	Y	Y
PIC16(L)F18876	(3)	16384	28	256	2048	36	35	1	2	3/4	2	Y	Y	5/2	Y	3	1	4	1	1/2	Y	Y
PIC16(L)F18877	(4)	32768	56	256	4096	36	35	1	2	3/4	2	Y	Y	5/2	Y	3	1	4	1	1/2	Y	Y

記憶體大小及封裝腳位的選擇也有很大的彈性。在這個 Lab 3 的觸控實驗裡，我們會使用的周邊有 ADC² (ADCC) 、EUSART、Timer2、 PPS 及 I/O 腳。

在使用這顆元件之前，先介紹一下這顆 PIC16F18855 元件的主要功能:

- 14KB (8KW) 快閃程式空間，1KB RAM，256 Bytes EEPROM
- 工作電壓範圍 : 1.8V ~ 3.6V (PIC16LF18855)，2.3V~5.5V (PIC16F18855)
- 消耗電流 : 32uA/MHz @ 1.8V, typical，Sleep mode: 50 nA @ 1.8V, typical
- 類比周邊: 10-bit ADC with Computation，兩組電壓比較器，5-bit DAC 轉換器，參考電壓輸出
- 豐富的數位周邊如下 :

- Four Configurable Logic Cells (CLC)
- Three Complementary Waveform Generators (CWG)
- Five Capture/Compare/PWM (CCP) module:
  - 16-bit resolution for Capture/Compare modes
  - 10-bit resolution for PWM mode
- Two 10-bit PWMs
- Numerically Controlled Oscillator (NCO)
- Two Signal Measurement Timers (SMT)
- Cyclical Redundancy Check (CRC/SCAN)
- Communication:
  - EUSART, RS-232, RS-485, LIN compatible
  - Two SPI
  - Two I2C, SMBus, PMBus™ compatible
- Up to 36 I/O Pins
- Peripheral Pin Select (PPS)
- Data Signal Modulator (DSM)

練習三

使用 PIC16F18855 的 ADCC 模組來完成一個穩定的觸控應用

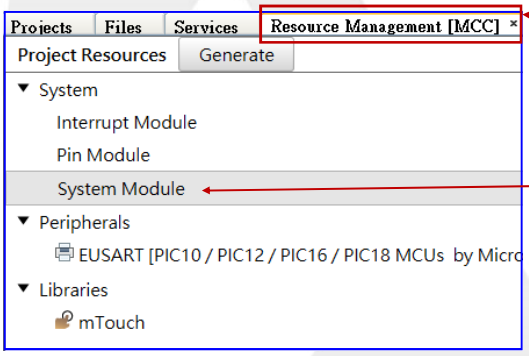
步驟

1 開啟 MPLAB® X 的專案，並使用 APP046 mTouch® 實驗板：

- 開啟 MPLAB X IDE
- 關閉目前被開啟的專案
- 開啟現有專案: ..\mtouch CVD\PIC16F18855 CVD Lab.X 或自行建立一個新的專案

2

接下來的步驟，請自行開啟一個使用 PIC16F18855 元件的專案(請參考 Lab1 的練習來開啟新的專案，並啟用 MCC 的工具 (這時請稍微等一下 MCC 的開啟)。

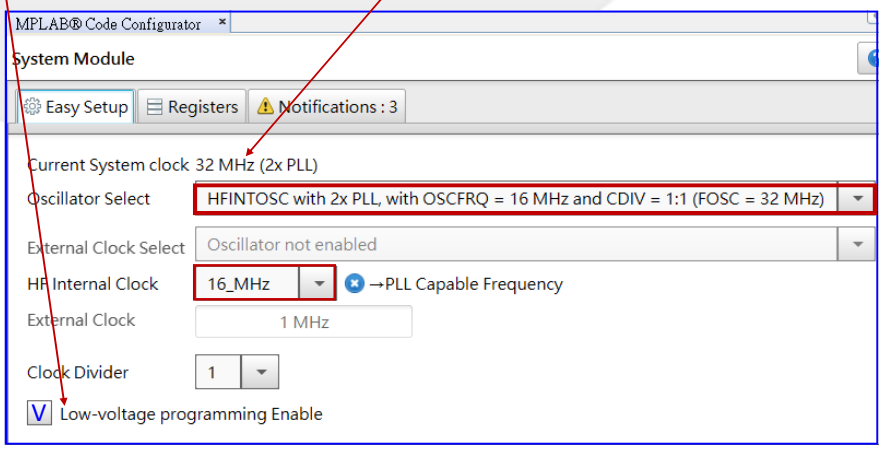


圖一: Project Resources 視窗

待 MCC 開啟完畢後，點選一下 “Resource Management [MCC] “ 的選卡項後，即會出現 “Project Resources” 的視窗。當然，首要設定的是 “System Module “的部分，這裡面主要設定工作的時脈頻率及 Configuration Words 的工作設定。

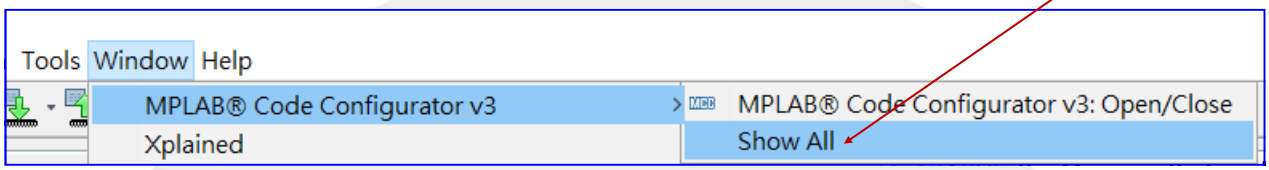
為了執行上可以發揮最大的效能，請將系統工作頻率設定為最高工作頻率 32MHz。在這裡使用 16 MHz 的內部 RC 震盪 (HFINTOSC) 配合使用 x 2 PLL 倍頻到 32MHz。因為本練習使用 PICKit3 或 ICD3 為開發工具，在此請開啟 “Low-Voltage Programming” 的功能使用一般低壓模式的 ICSP 燒錄模式。

此外，還有一些功能在開發及除錯階段是需要被關閉的，WDT (看門狗計時器) & Code Protection (程式碼保護)。除了在 “Easy Setup” 選項設定外，”Registers” 的設定選項有更多的 “Configuration Words” 的設定，如有需要變更其它設定可以在 “Registers” 裡做更詳細的設定。



3 建立使用的腳位

建立使用的腳位，依電路圖所示在 “Pin Manager: Grid [MCC]” 的視窗下選擇所要使用的腳位。如果看不到腳位選用視窗，請到主選項裡的 “Window —> MPLAB Code Configuration —> Show All” 開啟腳位選用視窗。



我們要參考一下 APP046 mTouch 的電路圖，並列出在 Lab3 裡所需使用到的 PIC16F18855 的腳位規劃，如底下的三張表格所示 (腳位的設定及名稱與 Lab1 相同)。

Touch Pad 名稱定義	I/O Pin	功 能
Proximity	RB0	Proximity Input
CS0	RB1	Touch_Pad 0
CS1	RB2	Touch_Pad 1
CS2	RB3	Touch_Pad 2
CS3	RB4	Touch_Pad 3
CS4	RB5	Touch_Pad 4

LED	I/O Pin	功 能
CS0_LED	RA5	LED4
CS1_LED	RA4	LED5
CS2_LED	RA3	LED6
CS3_LED	RA2	LED7
CS4_LED	RA1	LED8
Proxi0_LED	RA0	LED9

注意事項:

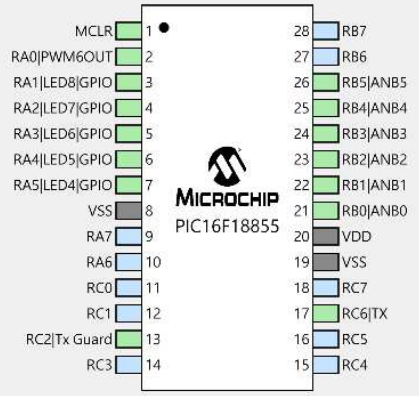
一般為求工作上的穩定，請習慣上將沒用到的腳位 ==> 關閉 ADC 輸入功能，設成輸出模式並輸出 Low 電位，避免有輸入腳的浮接以降低干擾。

其它設定	腳位	功 能
TxD	RC6	UART Tx
RxD	RC7	UART Rx
Guard	RC2	防護輸出
ICSP_CLK	RB6	燒錄/除錯
ICSP_DAT	RB7	燒錄/除錯
MCLR/Vpp	RE3	重置/燒錄高壓

點選 “Pin Manager: Grid [MCC]” 視窗先建立所要使用到腳位，下圖為本練習依據上頁的腳位規劃表所做的腳位設定。一定要先規劃好觸控感應所使用到的腳位後才可以做 mTouch 的設定。

Package:	PDIP28	Pin No:	2	3	4	5	6	7	10	9	21	22	23	24	25	26	27	28	11	12	13	14	15	16	17	18	1
			Port A ▼								Port B ▼								Port C ▼								E...
Module	Function	Direction	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	3
EUSART ▼	RX	input																									
	TX	output																									
OSC	CLKOUT	output																									
PWM6	PWM6OUT	output																									
Pin Module ▼	GPIO	input																									
	GPIO	output																									
RESET	MCLR	input																									
TMR2	T2IN	input																									
TMR4	T4IN	input																									
mTouch ▼	CS	input																									
	Tx Guard	output																									

規劃所使用的腳位

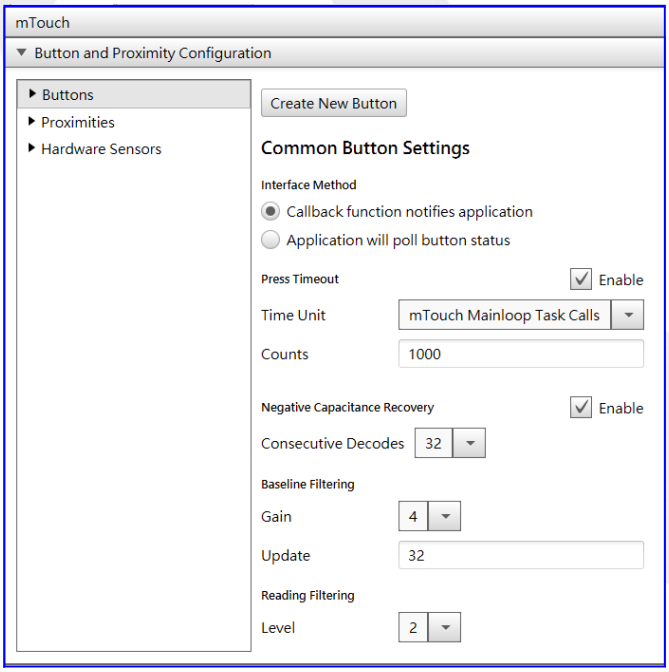


使用腳位顯示圖

3 設定 mTouch 的項目

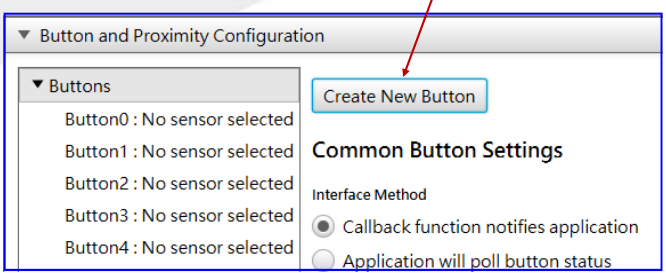
mTouch 的設定有三大項要設定，這些設定前面幾個練習都做過，這裡就請按照底下的圖示做設定輸入:

1. Buttons 的設定



觸控按鍵共同設定視窗

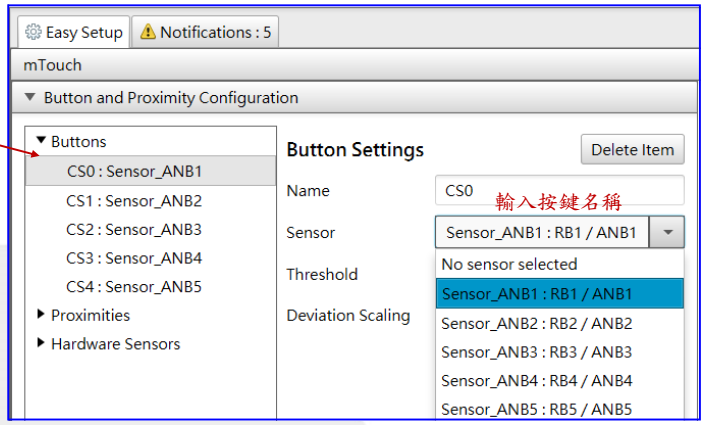
- 勾選使用 callback function 方式
- 設定觸控按鍵的溢時次數，在這裡設定 1000 次的迴圈檢測
- 其餘輸入項目請參照左圖的參數來輸入
- 輸入完畢後，按下 “Create New Button” 5 次，建立 5 個觸控按鍵的輸入，如下圖所示:



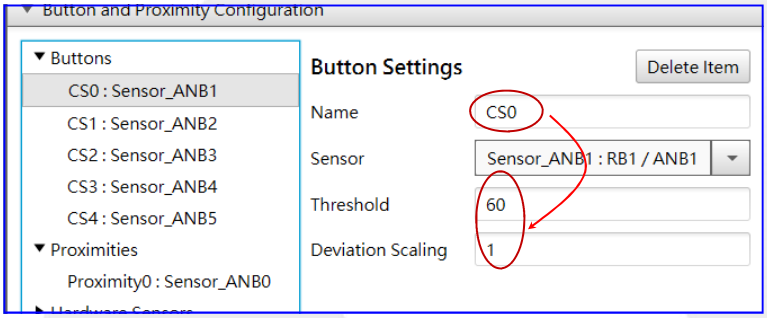
建立新的觸控按鍵

接下來開始設定觸控按鍵的名稱及連接的腳位。

- 點選每一個 “Buttons” 下的選項
- 在 “Button Setting” 視窗下，輸入該觸控按鍵的名稱(內定的名稱 Button_n) 及所使用的接腳。
- 在這裡總共有 5 個觸控感應輸入需指定輸入的腳位 (RB1 ~ RB5)。



指定觸控按鍵的接腳



單一觸控按鍵的設定值

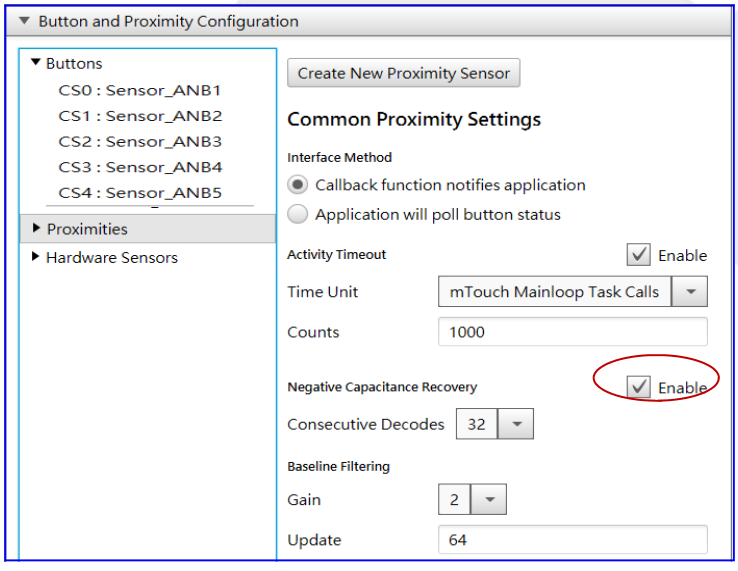
接下來的設定極為重要，每一隻腳的 PCB 拉線或或因感應 Pad 的大小、形狀也許會有所不同，所以所感應到的觸控讀值也就會不同。在這個設定項裡，允許每個觸控感應都可以單獨設定其感度的門檻值 (Threshold) 及差異值的縮放。為配合感應 Pad 的大小不一，在這裡的設定也都不同，請先依範例的值來設定。

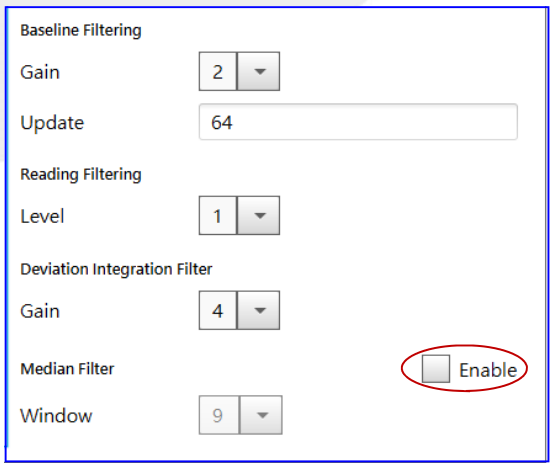
Threshold :在這裡將門檻值 (Threshold) 一 Pad 的直徑大小從 100(CS0) ~ 110(CS4) (最大值為 127)。這 Threshold 就是前面所談的 ΔV 的差異值 (目前的讀值 – 基準值)。Threshold 值較小感應零敏度較大，感應距離較遠。使用者可依實際的覆蓋物(例: 壓克力) 的厚度來調整 Threshold 的值。

Deviation Scaling : (差異值的縮放) 是在做 ΔV 數值的向右移位的運算，設定 1 就會將 ΔV 的值向右移一個位元。在這裡就使用 1 個位元 (除二) 的設定。

2. Proximity 的設定

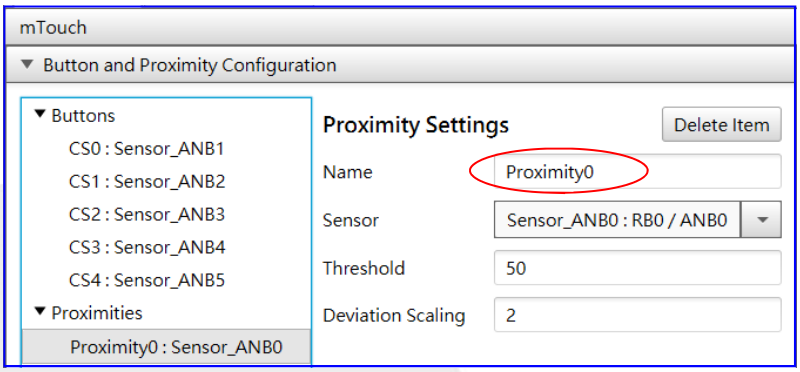
APP046 mTouch 的實驗板有支援一個接近感應 (Proximity) 的設計。在 mTouch 中這也需要單獨設定的，在 Proximity 的共同輸入設定畫面中，請使用底下圖表裡的參數值作為設定的輸入。





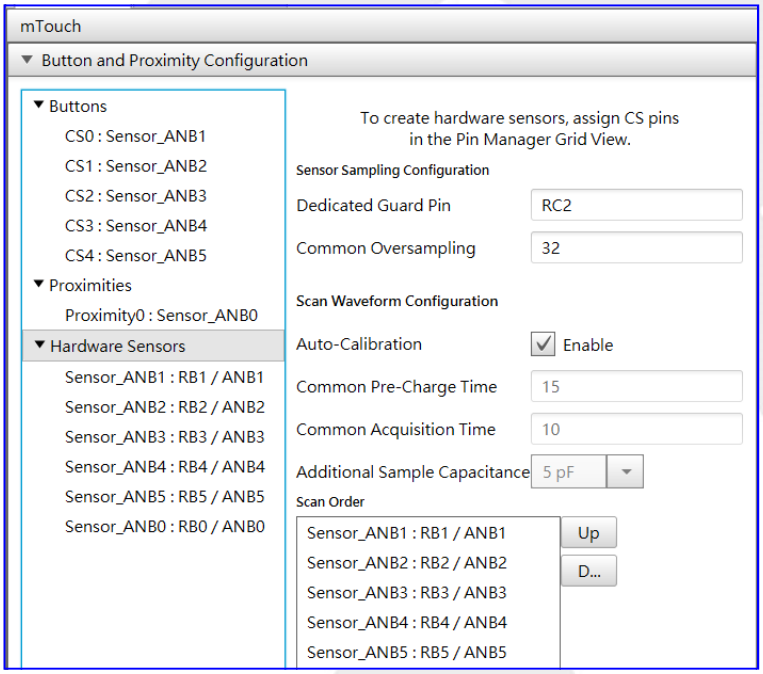
接下來要設定 Proximity 的名稱、接腳及門檻值。如下圖所示: 名稱使用內定的 “Proximity0” 名稱、接近感應腳位設在 RB0/ANB0。

因為是接近感應其實主要的是距離數值的反應，在這裡加入 Threshold = 50 的設定，並將差異縮放 = 2 以較低的感度來做接近的偵測。



3. Hardware Sensors 的設定

點選 “Hardware Sensors” 的設應畫面如下所示。這一個設定項是 “設定 Guard 的防護輸出腳位”。依照之前的腳位安排 Pin Manager: Gird [MCC] 的 Guard 的輸出腳位是 RC2。



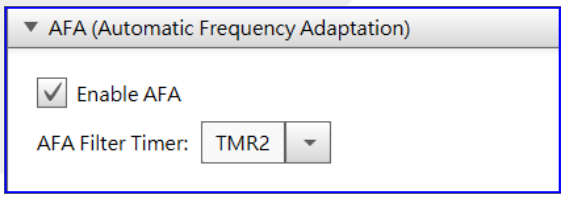
設定採樣中所有感應器的採樣次數，在此的預設值為 32。

啟用自動校準功能，硬體感應器的共用參數將由硬體的演算法自動校準。

如要調整預充電，採樣時間及增加 Chold 的容量，需先關閉自動取樣 (Auto-Calibration) 功能後才可以修改。如果增加了 Chold 的容量 (基本容量 10pF) 也要將預充電及取樣時間將長一點。

至於 Hardware Sensors 底下所管轄的 Sensor_ANB0 ~ Sensor_ANB5 則無須在特別的設定，全部使用 Default 的設定即可。

畫面在往下拉一下就可以看到一個 AFA (Automatic Frequency Adaptation) 的設定畫面。開啟這項功能，並使用 Timer2 做為跳頻產生器的計頻器。使用跳頻技術，可以降低外界雜訊的干擾。在這裡使用了 Timer2 後，在 Project Resources 裡就不會有 Timer2 的設定資源，因為 mTouch 已使用 Timer2 了，且 mTouch 有最高的優先使用 Timer2 的權限。

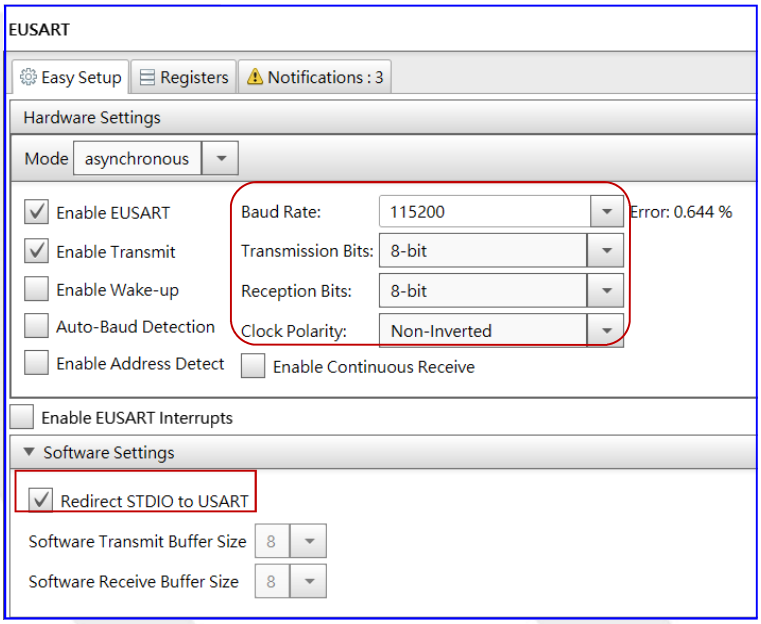


4 設定 EUSART (115200, N, 8, 1)

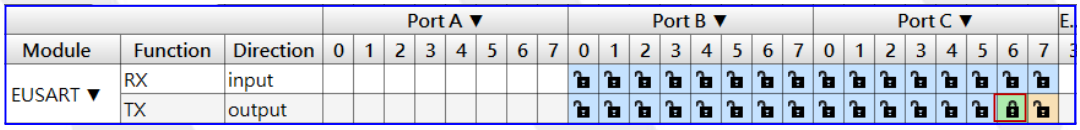
在這練習裡使用 EUSART 主要的目的是將觸控資料傳送出來到一個資料分析軟體 (Data Visualizer) 做觸控按鍵資料的顯示，讓使用者可以輕鬆地來調整 mTouch 的設定參數。

EUSART 有同步模式及非同步模式，在這裡需設為 "非同步模式傳輸"。通訊格式採用: 115200 bps，8-bit 的通訊資料長度，不做基、偶位元檢查，1 個結束位元。

在這個應用沒有用到接收，所以 "Enable Continuous Receive" 是關閉的。中斷也是關閉的。最重要的一點是因為有用到 printf( ) 做為 EUSART 的最終的輸出，所以要將 STDIO 指向 USART。最後，因為接收與傳送沒有使用到中斷，所以傳誦與接收的 Ring Buffer 都不需要啟用及設定的。



還有 USART 腳位的安排，請重新檢視一下 Pin Manager: Gird [MCC] 的視窗，確定 USART TxD 腳位已透過 PSS 設定在 RC6 的腳位，如右圖所示。



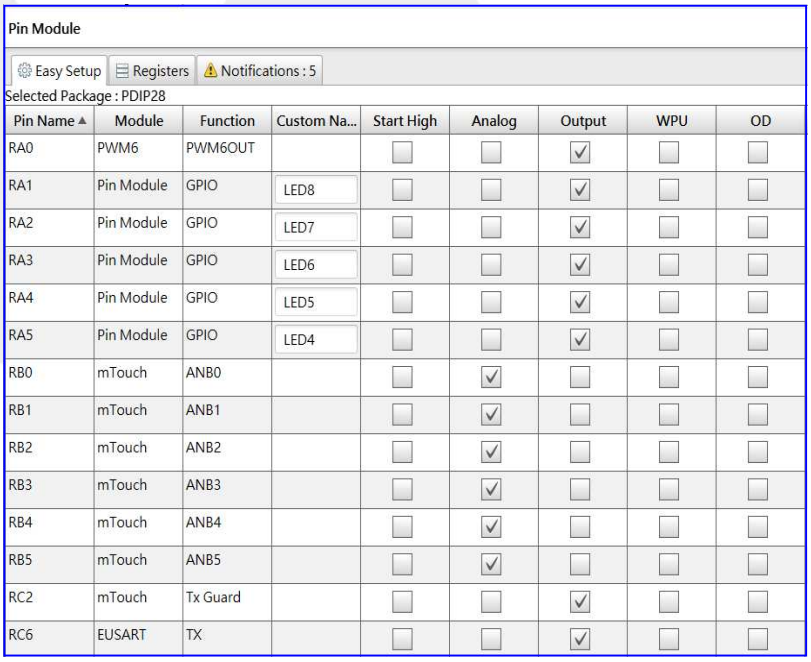
接下來是要規劃腳位的名稱及功能設定。請在 "Project Resources" 下的 "System" 開啟 "Pin Module" 視窗，如下圖所示:

Start Hi : 開機初始化完成後，該 Digital Out 腳位先送出 Hi 電位出去。

Analog : 關播或開啟該腳位的 ADC 類比輸入功能。在這裡只將 mTouch 要用到的 6 隻腳位 (RB0 ~ RB5) 勾選成類比輸入腳位。

Output : 純 Digital 輸出腳功能。如果這裡沒有勾選的話，那該腳位將設為 Digital Input 功能。在這裡 RA1 ~ RA5 為 LED 輸出腳、RA0 作為 PWM6 LED9 的輸出、RC2 為 Tx 輸出及 RC6 為 Tx Guard 的輸出。以上腳位都須設成輸出。

WPU : 是內部 Week Pull-Up 電阻的使用。



Pin Name	Module	Function	Custom Na...	Start High	Analog	Output	WPU	OD
RA0	PWM6	PWM6OUT		☐	☐	☑	☐	☐
RA1	Pin Module	GPIO	LED8	☐	☐	☑	☐	☐
RA2	Pin Module	GPIO	LED7	☐	☐	☑	☐	☐
RA3	Pin Module	GPIO	LED6	☐	☐	☑	☐	☐
RA4	Pin Module	GPIO	LED5	☐	☐	☑	☐	☐
RA5	Pin Module	GPIO	LED4	☐	☐	☑	☐	☐
RB0	mTouch	ANB0		☐	☑	☐	☐	☐
RB1	mTouch	ANB1		☐	☑	☐	☐	☐
RB2	mTouch	ANB2		☐	☑	☐	☐	☐
RB3	mTouch	ANB3		☐	☑	☐	☐	☐
RB4	mTouch	ANB4		☐	☑	☐	☐	☐
RB5	mTouch	ANB5		☐	☑	☐	☐	☐
RC2	mTouch	Tx Guard		☐	☐	☑	☐	☐
RC6	EUSART	TX		☐	☐	☑	☐	☐

一般而言 WPU 內部提升電阻其阻值約在 100K ~ 200K ohm 左右，主要是用在該腳位設為輸入模式時，可以透過 WPU 來設定使用內部的提升電阻。

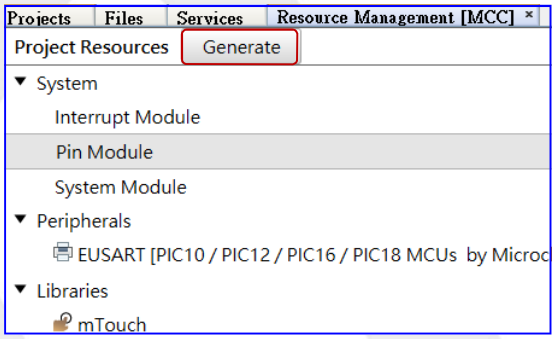
OD : 為 Open Drain (開集極) 的設定。一般使用在輸出的場合，外部須接一提升電阻後才能輸出 Hi 的準位。

Custom Name : 用戶自訂名稱。在這裡設定的 RA1~ RA5 為 LED8 ~ LED4 的名稱。在這裡的名稱指定經 MCC 的 "Code Generate" 後，可以在 "pin_manager.h" 檔裡看到定義後的巨集。

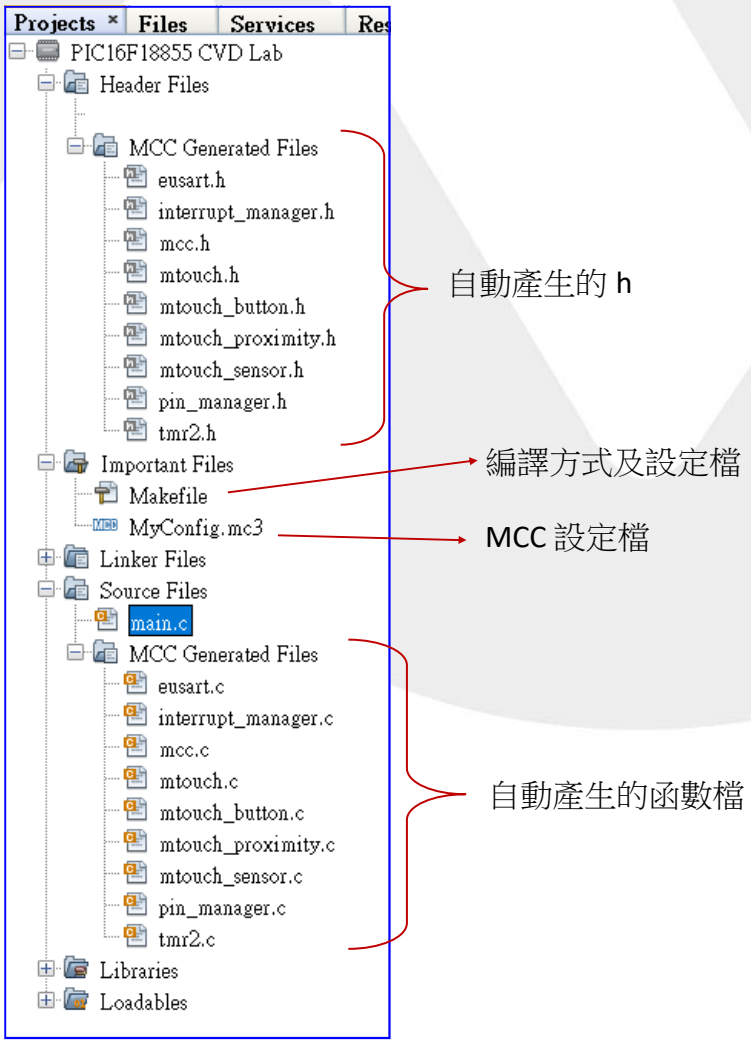
```
#define LED6_SetHigh()      do { LATAbits.LATA3 = 1; } while(0)
#define LED6_SetLow()       do { LATAbits.LATA3 = 0; } while(0)
#define LED6_Toggle()       do { LATAbits.LATA3 = ~LATAbits.LATA3; } while(0)
#define LED6_GetValue()     PORTAbits.RA3
```

5 加入應用程式與執行

至此，我們幾乎完成了 mTouch 的設定 (只剩下應用程式 main.c) 還沒搞定。視窗切到 "Resource Management [MCC] "，點選一下程式產生 "**Generate**" 按鈕後，Generate 按鈕顏色會立即變深，莫約幾秒後再變回原來的顏色後表示程式碼已經產生完畢。



開啟 Project 專案視窗，並做一些項目的展開後就可以看到所有自動產生的程式檔案，包括 main.c。



MCC 的周邊函數產生完後，再來就是撰寫 main.c 的應用程式了。當然 MCC 也會產生一個使用 MCC 函數庫的 main.c 檔，我們只要開啟 Project 裡的 main.c 來做修改。

1. 修改的第一步驟是: mTouch 有使用到中斷，所以要先開啟兩個中斷功能，一是 Global 中斷，二是 Peripheral 中斷。這兩個中斷只要在 main.c 裡將前面的註解(//) 移除即可。

```
52  void main(void)
53  {
54      // initialize the device
55      SYSTEM_Initialize();
56
57      // When using interrupts, you need to set the
58      // Use the following macros to:
59
60      // Enable the Global Interrupts
61      INTERRUPT_GlobalInterruptEnable();
62              移除中斷的註解
63      // Enable the Peripheral Interrupts
64      INTERRUPT_PeripheralInterruptEnable();
65
66      // Disable the Global Interrupts
67      //INTERRUPT_GlobalInterruptDisable();
```

```
72      while (1)
73      {
74          if (MTOUCH_Service_Mainloop())
75          {
```

2. 不管是使用 Polling 或是 Callbaclk 方式都要加 入這一行的敘述 if (MTOUCH_Service_Mainloop()) 來啟用 mTouch 的程式。

3. 再下來就是觸控應用程式的部分。在這裡們只說明簡單的使用方式，也就是只有觸摸到 Pad 時，相對應的 LED 會被點亮的應用而已。請將底下的程式加到 main.c 裡。

```
while (1) {

  if (MTOUCH_Service_Mainloop())
    datastreamer_output();
  {
    if (MTOUCH_Button_isPressed(0))
      LED4_SetHigh();
    else
      LED4_SetLow();
  }
  {
    if (MTOUCH_Button_isPressed(1))
      LED5_SetHigh();
    else
      LED5_SetLow();
  }
  {
    if (MTOUCH_Button_isPressed(2))
      LED6_SetHigh();
    else
      LED6_SetLow();
  }
```

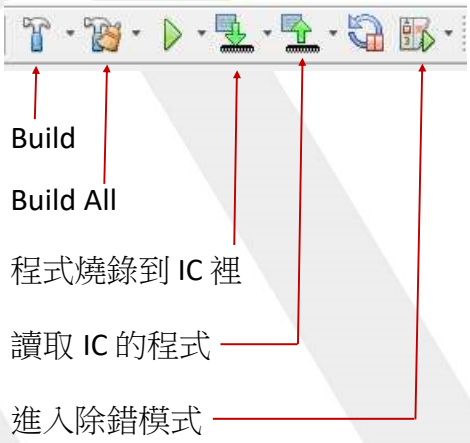
```
  { if (MTOUCH_Button_isPressed(CS3))
      LED7_SetHigh();
    else
      LED7_SetLow();
  }
  { if (MTOUCH_Button_isPressed(CS4))
      LED8_SetHigh();
    else
      LED8_SetLow();
  }
  { Proximity = MTOUCH_Proximity_Deviation_Get
(Proximity0); //接近感應改用 PWM 點亮 LED9 來呈現
    PWM6DCL = 0x00; // Duty = 0 , LED9 熄滅
    if (Proximity <= 5) PWM6DCH = 0 ; // 接近值 5 以
    Else PWM6DCH = Proximity <<1;   // 下 LED9 不亮
  }
```

到目前為止，算是完成了觸控感應的程式。按下 “Shift + F11” 就會開始編譯，而且可以的到沒有錯誤的編譯。(編譯過程中會有不少的警告或提醒，這些都不用裡它。)

只要看到綠色的顯示: BUILD SUCCESSFUL 就表示完成的編譯動作。

```
make[2]: Leaving directory 'D:/Test/PIC16F18855 CVD Lab.X'
make[1]: Leaving directory 'D:/Test/PIC16F18855 CVD Lab.X'

BUILD SUCCESSFUL (total time: 8s)
Loading code from D:/Test/PIC16F18855 CVD Lab.X/dist/default/production/PIC16F18855_CVD_Lab.X.production.hex...
Loading completed
```

4. 接下來就可以使用除錯模式或燒錄模式來驗證程式的功能。按下  的圖示，先編譯再燒錄，也許會因目前 PICKit3 的 F/W 版本是其它元件的，這是 X IDE 會執行 F/W 更新動作，這時就要花些時間等一下 F/W 的完成更新。最後會進入燒錄模式，等最後顯示出 “Programming/Verify Complete” 就完成了燒錄並直接執行程式。



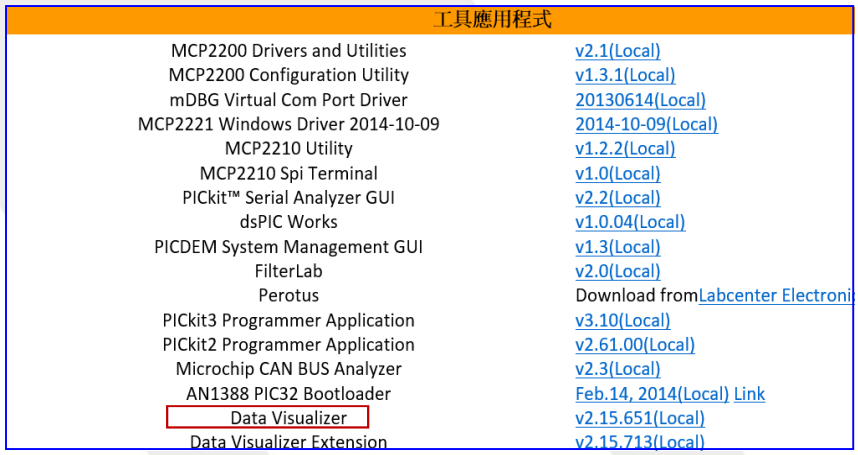

```
**************************************************

Connecting to MPLAB PICkit 3...

Currently loaded firmware on PICkit 3      檢查 PICKit3 連線
Firmware Suite Version.....01.51.06
Firmware type..............Enhanced Midrange

Target voltage detected
Target device PIC16F18855 found.           尋找目標元件
Device ID Revision = 2002
                                           清除 Flash & Con-
Device Erased...                           figuration Words

Programming...                             程式燒錄中….

The following memory area(s) will be programmed:
program memory: start address = 0x0, end address = 0x181f
configuration memory                       程式燒錄範圍
Programming/Verify complete                程式燒錄/比對成功
```

5. 最後請測試一下，觸摸 CS0 時對應的 LED4 是否被點亮。再一一測試其他的 CS Pad 看看。接近感應 Promixity0 的動作約在 12 公分的距離內 LED9 就會有由暗慢慢變亮的 PWM 動作。

6 使用 Data Visualizer 來觀察觸控資料

到目前為止，這觸控按鍵應該是可以工作了，但我還是想要能監控觸控的資料做為 mTouch 參數調整的依據。我們使用 Atmel 的 Data Visualizer 來做數據的觀測，當然 Data Visualizer 需自 Microchip 網站下載，這裡有兩個地方可以下載:

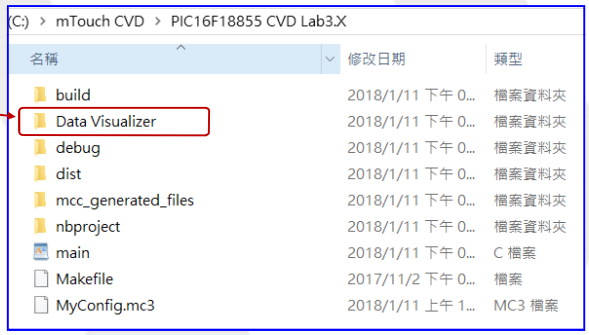
1. 美國官網: https://gallery.microchip.com/packages/AtmelDataVisualizerInstaller-Standalone/
2. 台灣官網下的教育訓練光碟 : http://www.microchip.com.tw/Data_CD/



Data Visualizer 有兩個版本，一是 Standalone (獨立版本)及 Extension 版本需外掛在 Studio 7 才能執行。

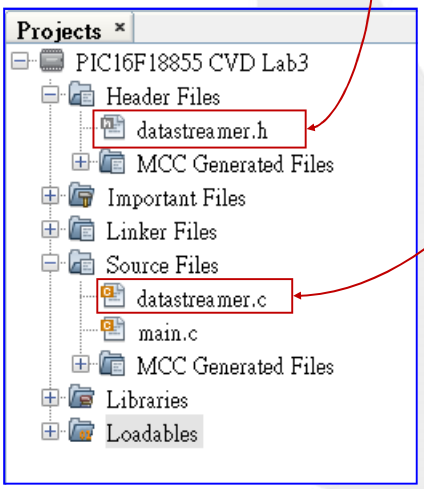
建議使用 Standalone v1.15.65 的版本(其他版本不相容)。美國官網搜尋一下”Data Visualizer” 就會有使用手冊。

步驟一 : 確定 Data Visualizer 設定用的檔案夾已經拷貝至 PIC16F18855 CVD Lab3.X 的專案下。



步驟一

步驟一: 要將 Data Visualizer 的標頭檔及程式檔 加到專案裡如步驟二的圖示。



步驟二

步驟三 : 加入標頭檔 datastreamer.h 到 main.c 裡。

```
#include "mcc_generated_files/mcc.h"
#include "Data Visualizer/datastreamer.h"
```

說明: “ xxxx” 是在目前專案下的目錄路徑下尋找。

步驟四 : 加入 datastreamer_output( ) 的函數到 main.c 的主迴圈裡。

```
While(1)
{
        if (MTOUCH_Service_Mainloop())
        datastreamer_output();
```

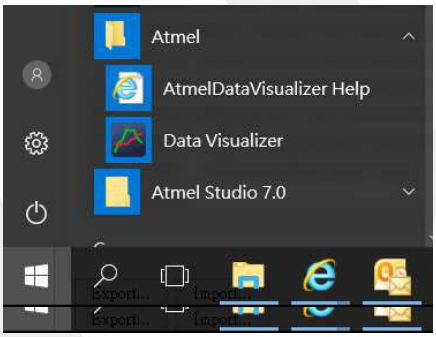
步驟五：編譯程式及燒錄。測試看看觸控按鍵是否正常動作。

開始使用 Data Visualizer 觀察資料

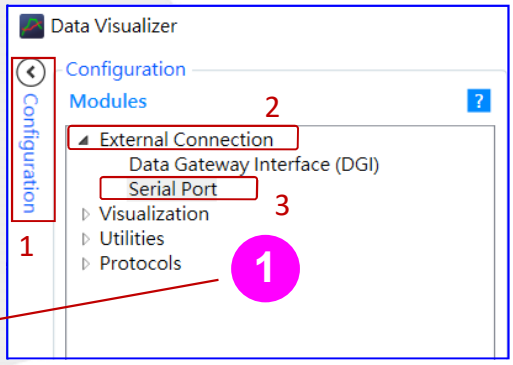
1. 連接板子上的 Micro USB 到電腦的 USB 接頭。
2. 因為我們是使用 UART to USB 的方式送出觀察資料的。首先要找出這 USB 模擬的 COM Port。在最下角 圖示 用老鼠左鍵開啟左邊的畫面。開啟裝置管理員的視窗。

3. 在裝置管理員視窗下，檢視 “連接埠 (COM 和 LPT)” 的連接項目。在這裡已經看到 “USB 序列裝置 (COM11)。這個虛擬通訊埠為動態安排的安裝越多的串列通訊裝置像藍芽、USB Dongle 等都會增加 COM Port 的數字。我的電腦目前是使用 COM11，但你的電腦會顯示別的 COM Port 數字。只要記住你所顯示的是哪一個 COM Port 就好了。

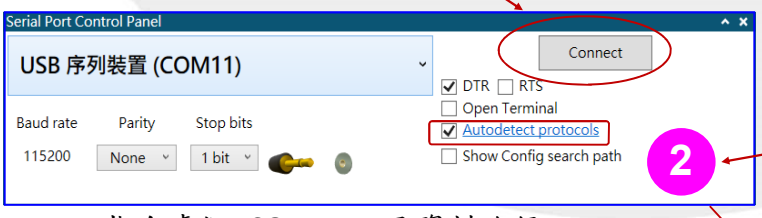
4. 啟動 Data Visualizer，按下 Windows 鍵 後找到 Atmel 的目錄，按下圖示 Data Visualizer 執行 Data Visualizer 或直接在桌面執行 Data Visualizer。



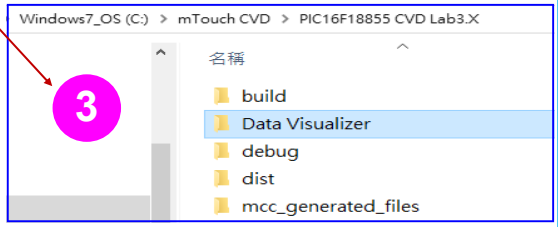
5. 底下開始做 Data Visualizer 的使用設定。
   - Configuration (1) —> External Connection(2) —> Serial Port (3)
   - Serial Port Control Panel 的視窗將會被開啟，如下圖所示，確定勾選了 Autodetect Protocols (自動通訊速度及通訊協定設定)後，再按下 “Connect” 按鍵圖示後 (第一次的使用)會出現要求輸入 Data Visualizer 檔案路徑的名稱，這時請將路徑帶到專案下的 “Data Visualizer” 目錄即可。



Data Visualizer 的啟用



指向虛擬 COM Port 及資料路徑



指向 Data Visualizer 路徑

6. Data Visualizer 開始自動搜索 Serial Port 直到 Baud 115200 時會連線成功。

7. 如果連線成功，這時可以看到按鍵資料視窗及圖形的顯示。先看一下按鍵資料的顯示視窗並做說明。

CS0 按鍵資料
CS1 按鍵資料
CS2 按鍵資料
CS3 按鍵資料
CS4 按鍵資料
Proximity0 接近資料

Button Data							
Reading0	8500	Baseline0	8496	Deviation0	2	Button Stat	0
Reading1	7931	Baseline1	7926	Deviation1	2	Button Stat	0
Reading2	9399	Baseline2	7196	Deviation2	127	Button Stat	1
Reading3	6943	Baseline3	6941	Deviation3	1	Button Stat	0
Reading4	6744	Baseline4	6742	Deviation4	1	Button Stat	0
Proximity Data							
Reading	16856	Baseline	15405	Deviation	127	Proximity S	1

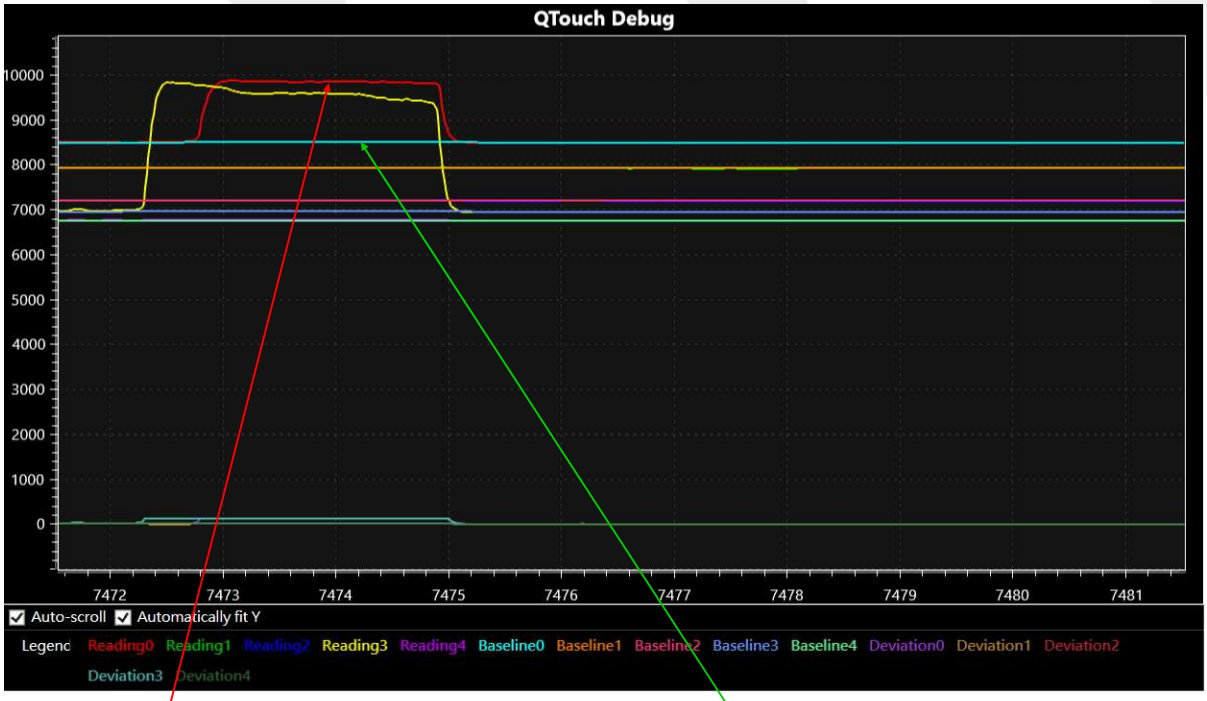
按鍵放開
按鍵放開
按鍵動作
按鍵放開
按鍵放開
接近偵測動作

按鍵讀取值　按鍵基線值　差異值　按鍵狀態

CS2 觸控按鍵的差異值設定為 80，此時的值為最大值 127，所以按鍵狀態是已按下的動作。

Promixity0 觸控按鍵的差異值設定為 50，此時的值為最大值 127，所以接近偵測是已偵測到。

8. 底下是依觸控資料所繪製的波形。



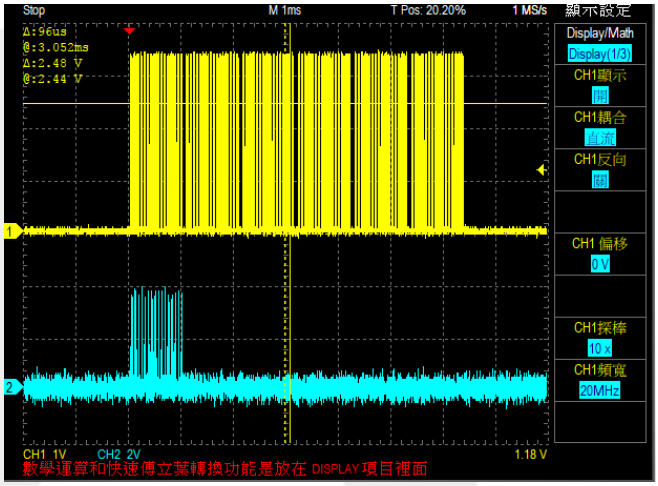
紅色波形顯示 CS0 觸控按鍵的變化，圖中可以看出原先 CS0 的基本值約為 8500 左右，觸摸 CS0 數值升到約 10000 左右。黃色波形為 CS4 的數值，數值約為 9800，因為 CS4 的 Pad 比 CS0 的 Pad 大，所以數值變化也比較大。

9. 由於 PIC16F168855 是使用 ADCC 周邊來做 CVC 的觸控功能，預充電的方是採用不同的演算法來完成，其輸出波形與 SCVD 長得不一樣，但其原理是一樣的。底下這 Lab3 完成後所測量到的波形，示波器的 CH1 是測量 Guard (RC2) 的輸出訊號，CH2 是測量 CS0 的觸控腳的波形。

右圖可以看出總共有六個觸控按鍵的輸入正在被掃描中 CH1（CS0 -> CS1 -> …. -> Proximity0)。CH2 波形群是測量 CS0 腳位上的量測波形。

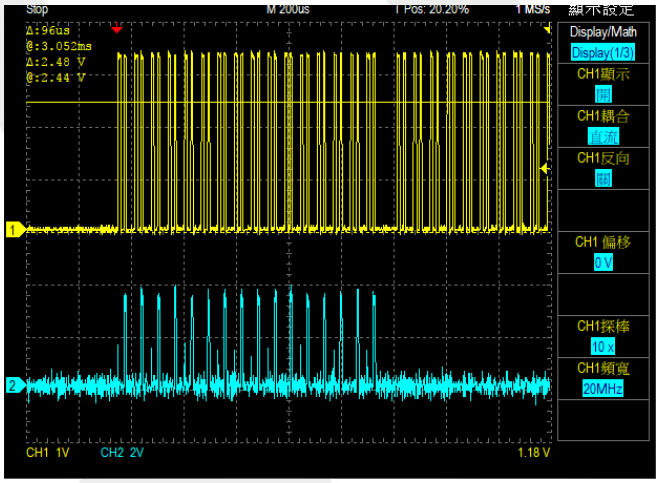
Guard (CH1)

CS0 (CH2)



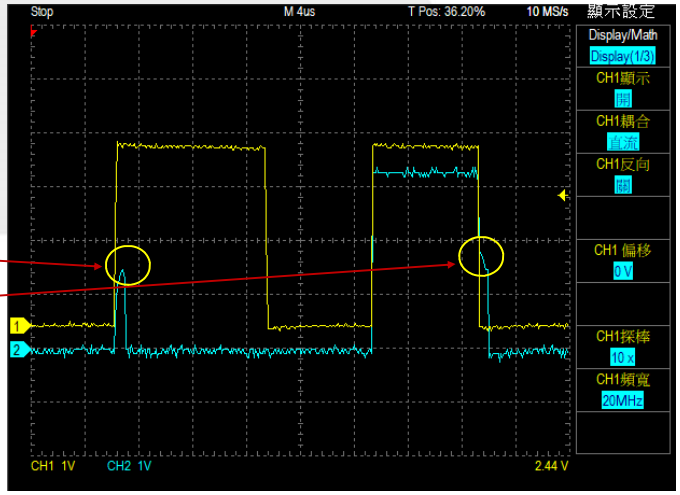
右圖是放大第一個觸控感測 CS0 的測量。CH2 是感測腳位 CS0 的波形，可以明顯看出每個 Channel 都會做了 16 次的量測後，再經內部的 Computation 做累積及平均的運算後輸出。

CS0



右圖是在放大 CH2 的第一個觸控感測 CS0 的測量。可以明顯地看到先建立 Guard 的電場一次做**正向的測量**。在第二次 guard 建立後在歸零後立即做**負向的測量**。

CS0



課程結束!

如有問題請撥打技術服務專線:

0800-717-718

Microchip Technology 台灣分公司
臺北市名權東路三段四號 12F
總機電話: 02-2508-8600