CAE空中教室 mTouch系列

mTouch 理論探討

說明 Capacitive Voltage Divider (CVD) 的工作原理與干擾的防護

A Leading Provider of Smart, Connected and Secure Embedded Control Solutions

Richard Yang
CAE Taiwan Team

Jan. 13, 2021

Agenda

- 電容式感應種類
- Capacitive Voltage Divider 的原理
- 保護/驅動屏蔽
- 防水機制
- APP-ESS18-2 mTouch mTouch evm Board
- 視頻展示
  - PIC16F18855 mTouch Demo 1

Microchip 基本的觸控方式

- 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

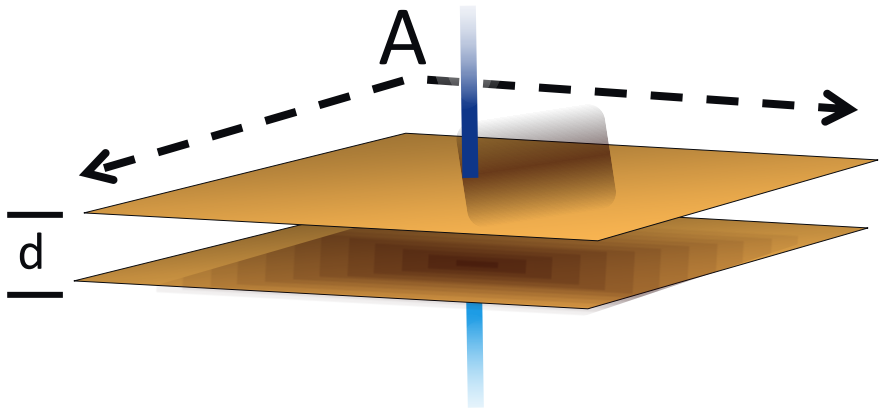
CVD 的原理說明

- 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，最後再測量大水桶平衡後的水位高度(即電壓)。

- 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。

常駐的電容？

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



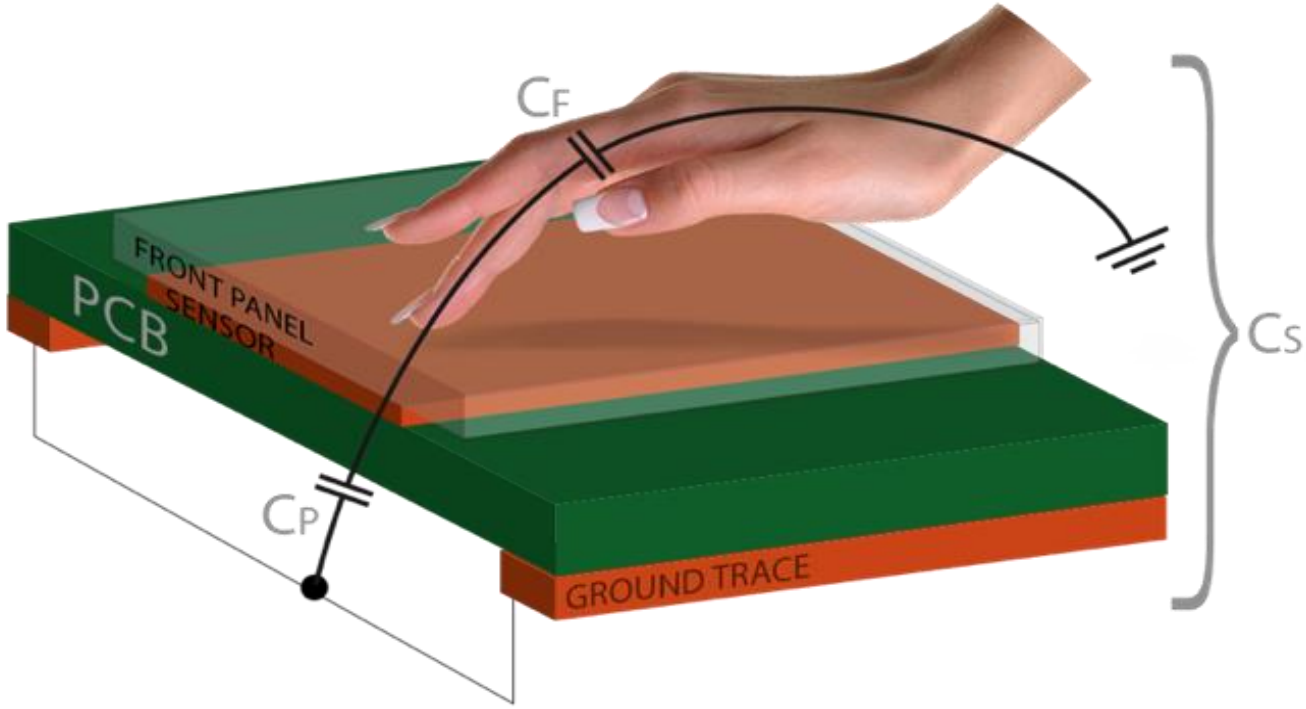
C	電容 (法拉)
ε_0	自由空間的介電常數 (8.854 Pico-Farad/meter)
ε_r	相對介電常數 (unit-less)
A	面積 (meters²)
d	兩個平板間的距離 (meters)

介質材料	ε_r
空氣	1
聚乙烯	2.25
聚苯乙烯	2.4 – 2.7
玻璃	4 – 10
玻璃纖維板	4.8
水	80

電容感測？

手指的觸摸所產生的感應電容

(**C_F**)



C_P: *觸控點的寄生電容*
C_F: *手指的感應電容*
C_S: 觸控點的電容總和 (並聯效應)

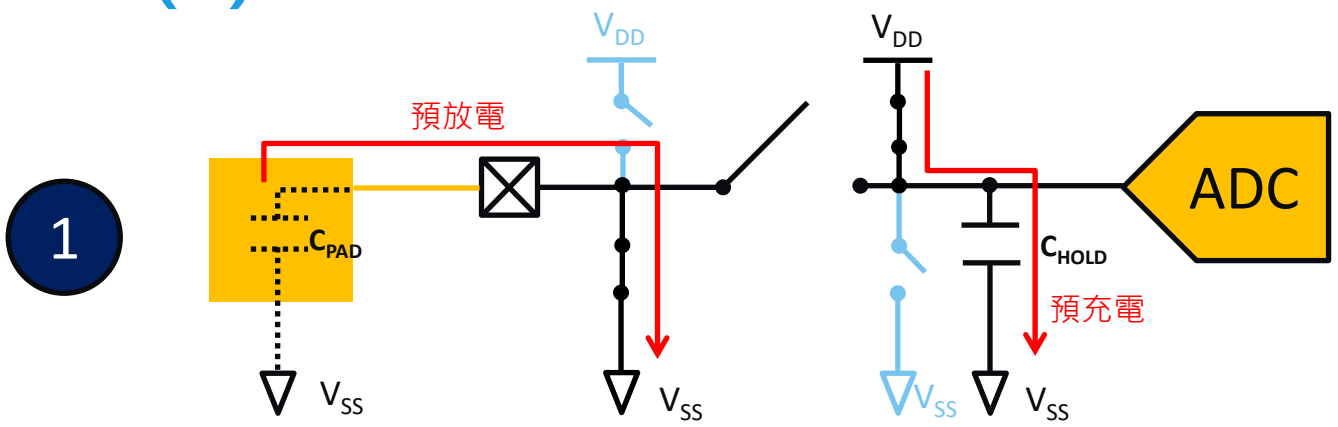
感應觸控點的總電容量 (C_S) = **C_P** + **C_F**

MICROCHIP

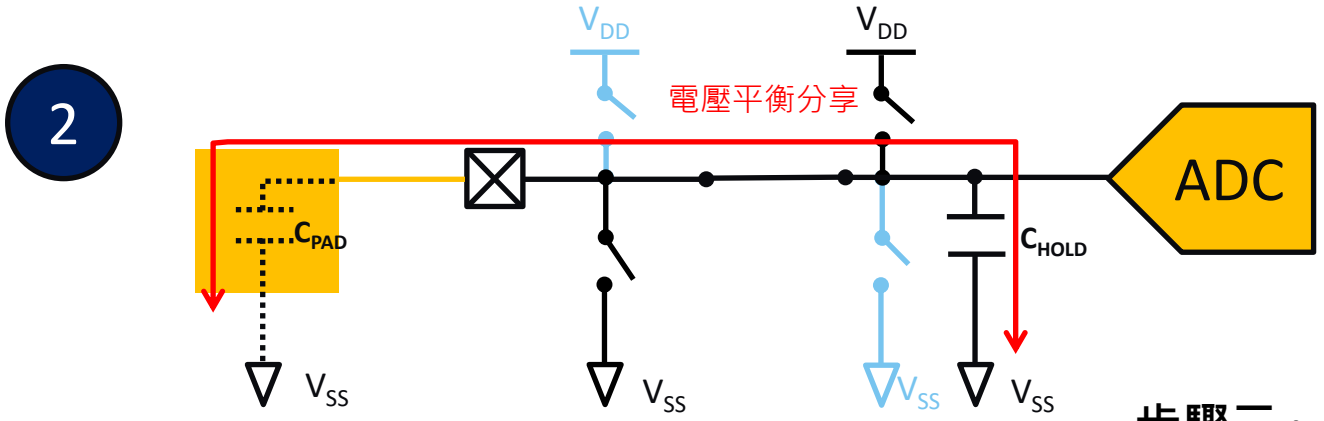
Capacitive Voltage Divider

正向轉換 (1)

PIC ADC 與 I/O 周邊具有特殊切換的設計這些內部的類比開關都可以用暫存器的設定來完成適當的切換



步驟一：正向的預充電準備



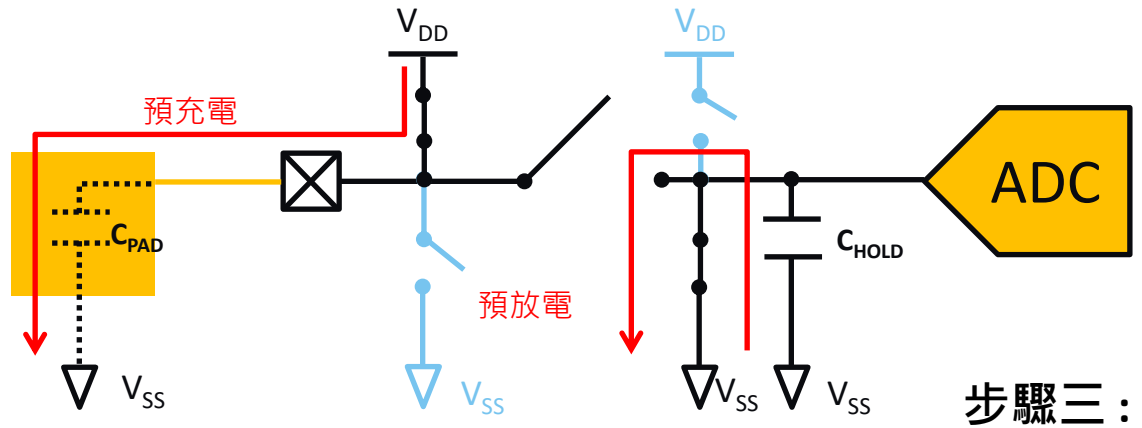
步驟二：CHOLD 開始放電取得電位的平衡並轉換

MICROCHIP

Capacitive Voltage Divider

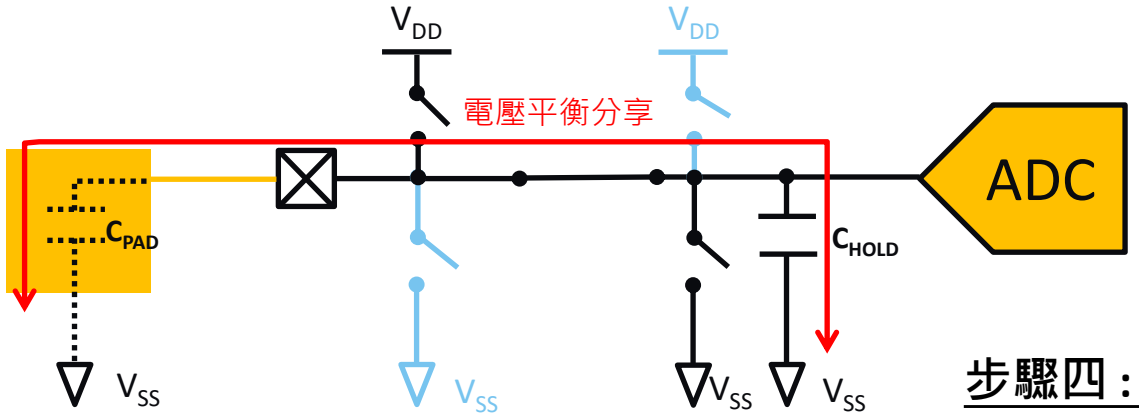
負向轉換 (2)

3



步驟三：負向充、放電的準備

4

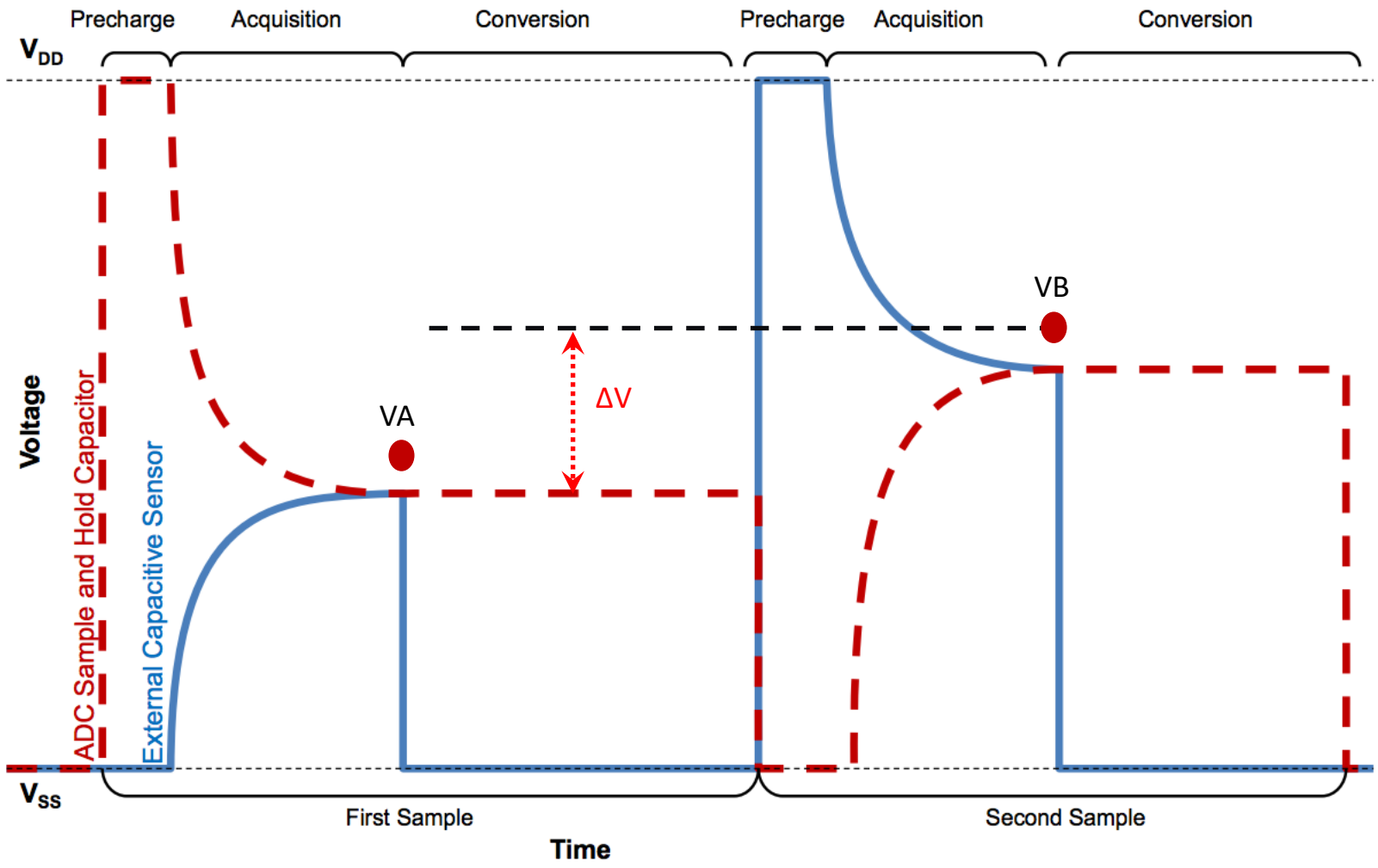


步驟四：CPAD 開始放電取得電位的平衡後轉換

MICROCHIP

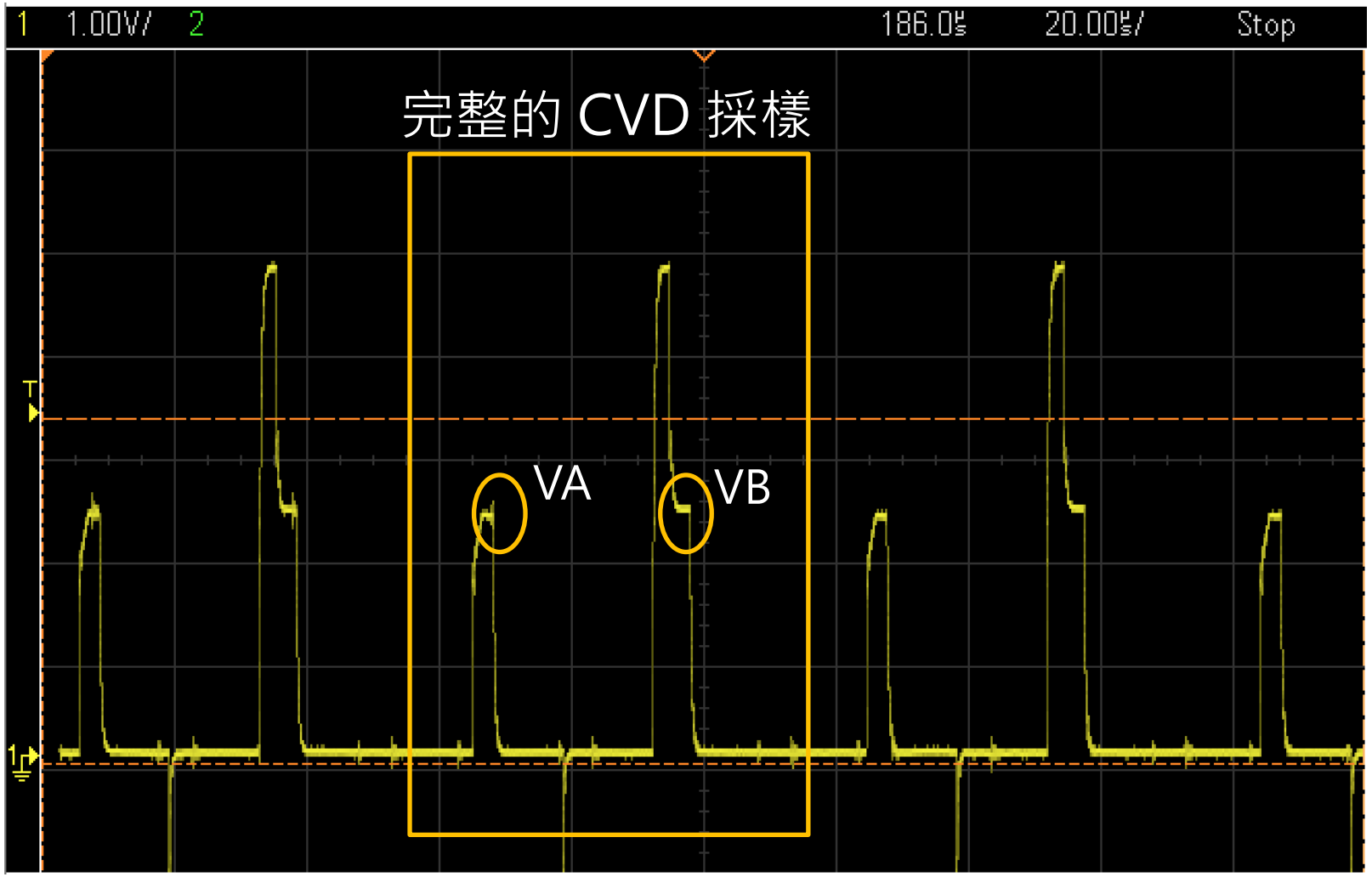
CVD : 外部 PAD 與內部 C_{HOLD} 的電壓變化

正向的充放電電壓波形　　負向的充放電電壓波形

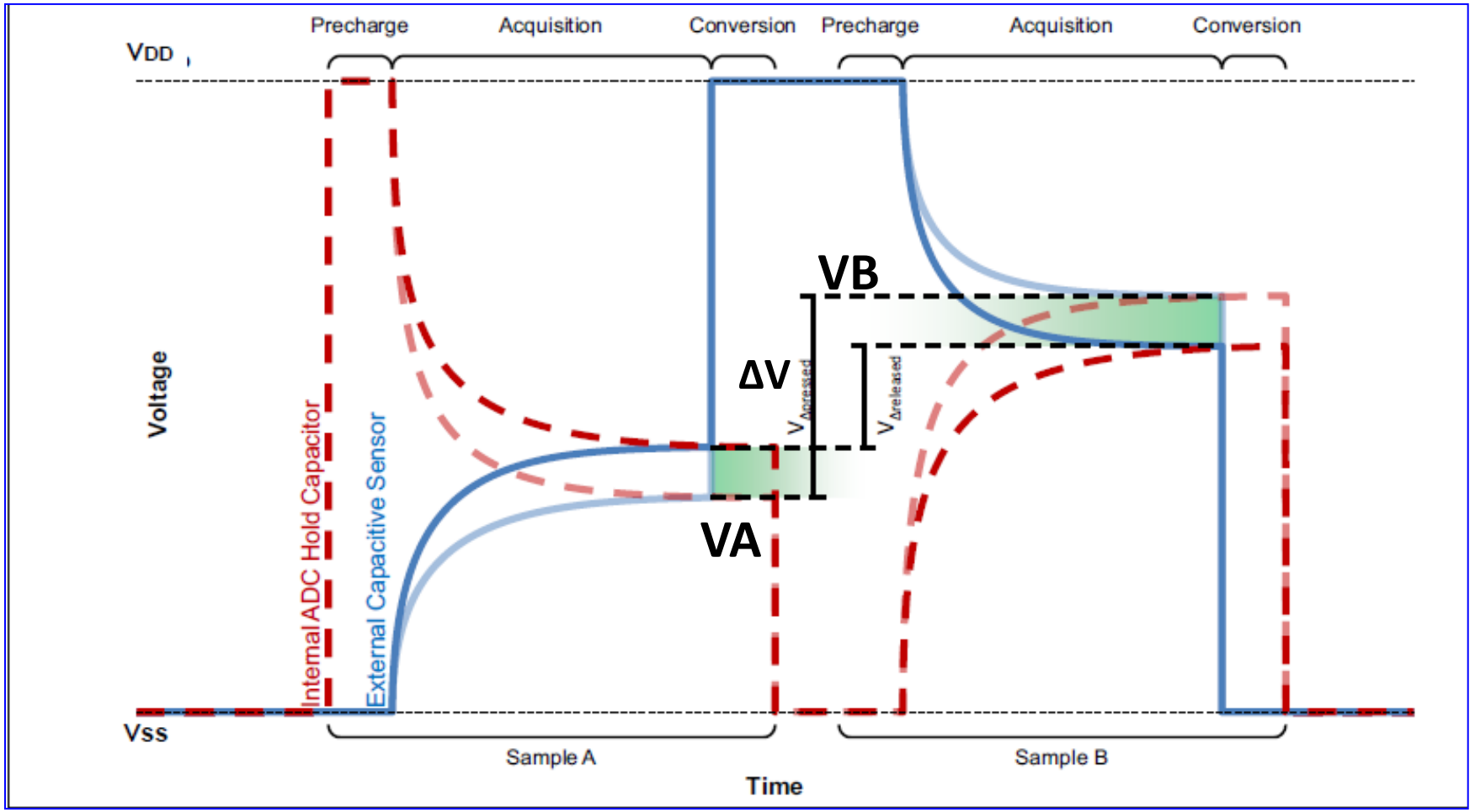


MICROCHIP

實測觸控點 PAD的 CVD 波形



取得觸控電壓的變化量



VB – VA 可以用嗎?

MCC 使用的計算方式 = V Δpress—V ΔRelease

MICROCHIP

PAD-PAD 之間的串擾

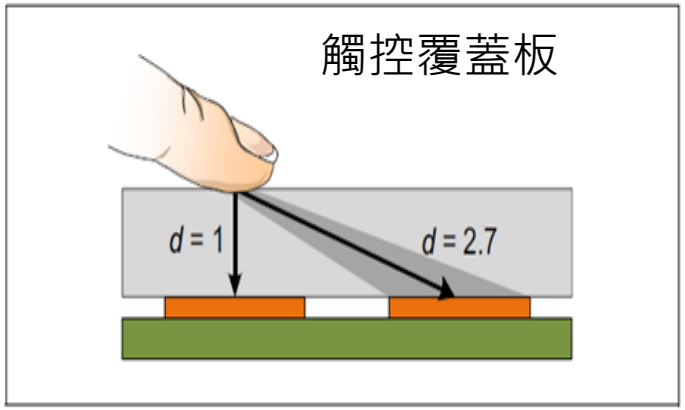


圖 1 顯示了手指按壓的串擾程度

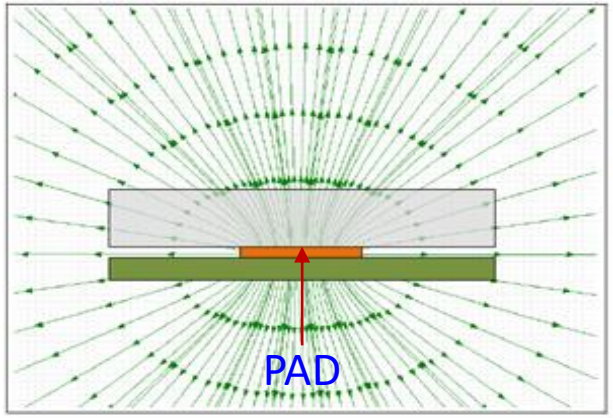


圖 2 單獨一個 Pad 時的電場線分布圖

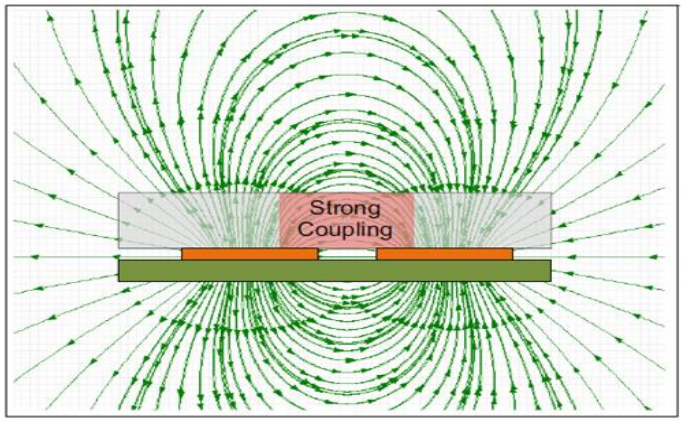


圖 3 兩相鄰的 Pad 的電場線分布圖

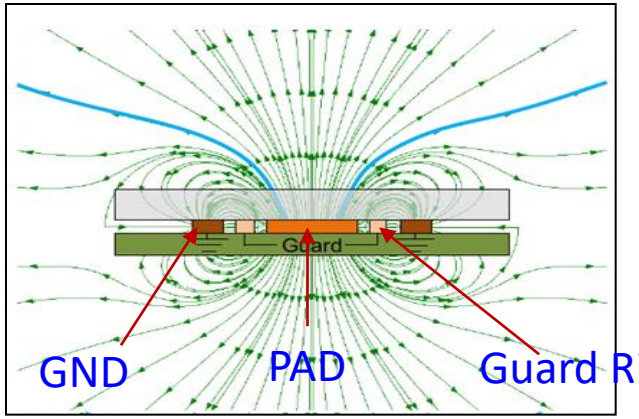
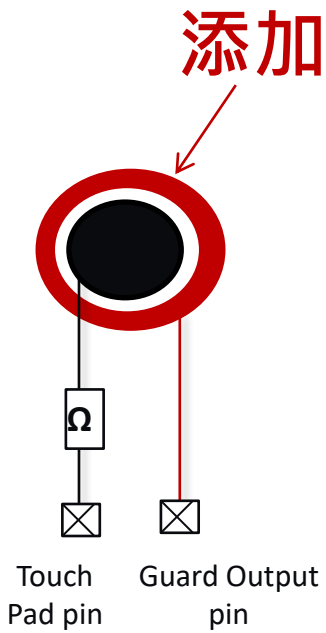
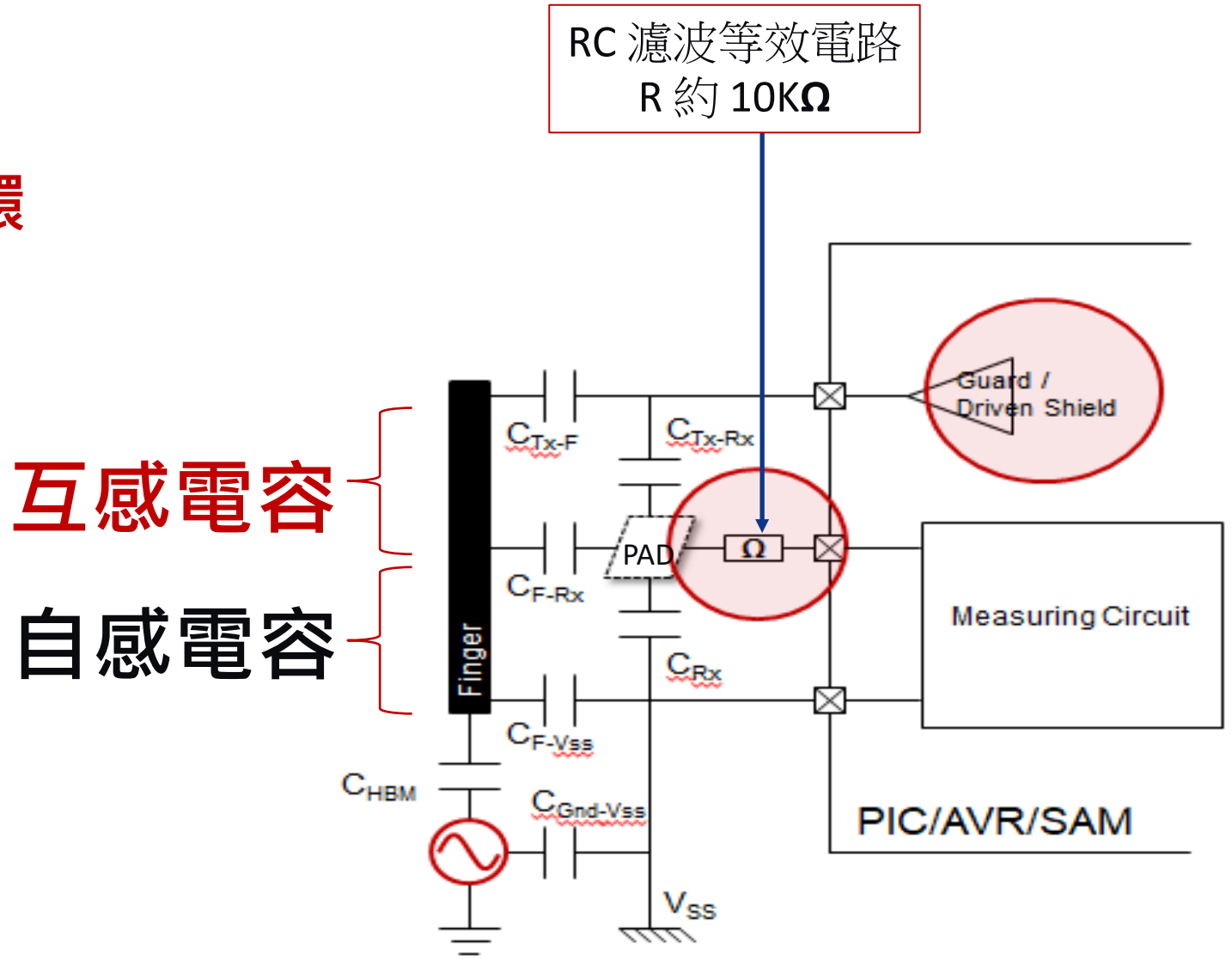


圖 4 加入 Guard 防護後的電力線分布圖

加入防護 / 屏蔽





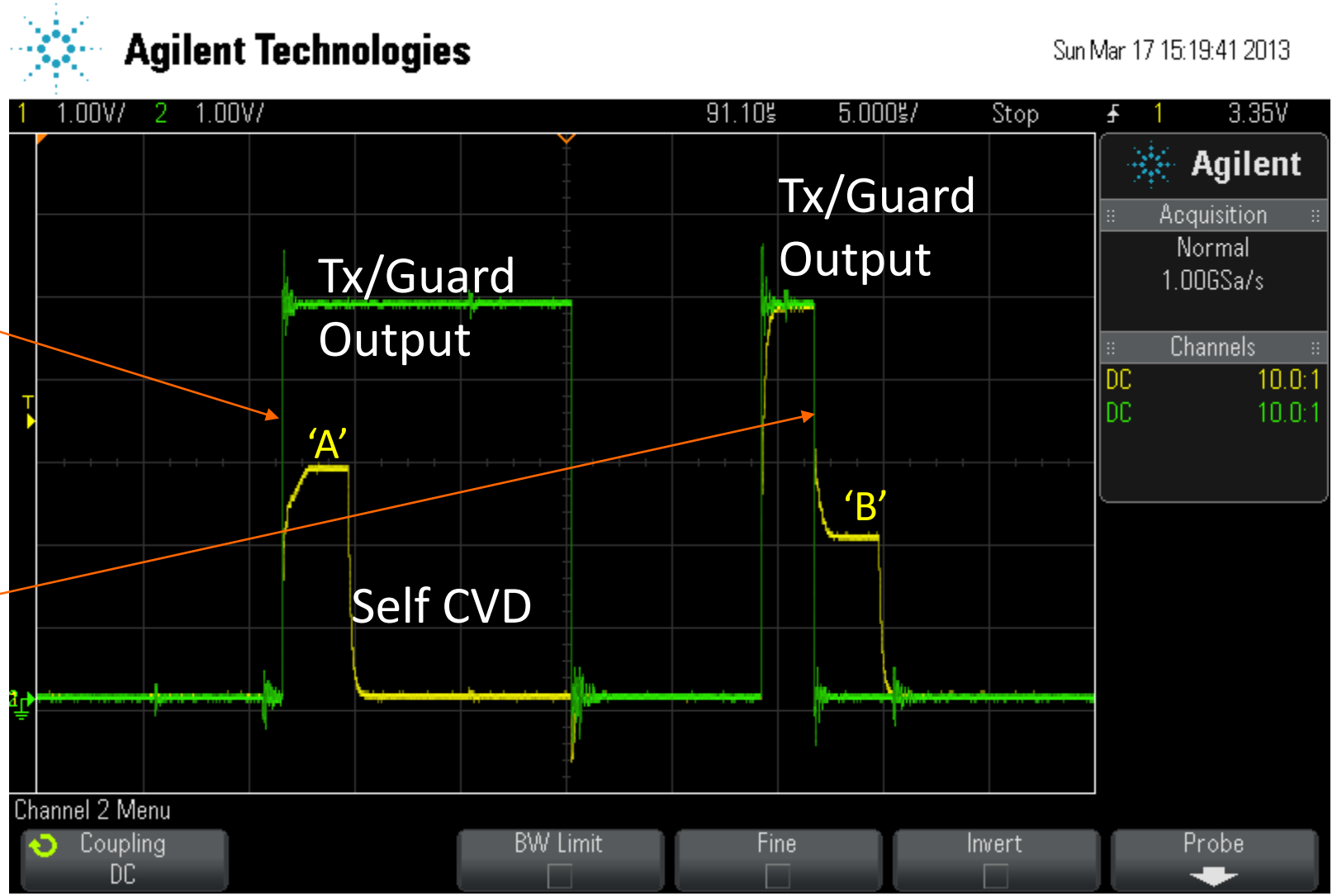
MICROCHIP

混合式的自/互感掃描

Guard 的啟動時機：

- 在正向轉換時 Chold 開始向 Pad 充電時，Guard 送出 Low → Hi 的訊號向Pad 提供正電場，'A' 的電壓會提高。
- 在負向轉換時，Pad 向 Chold 充電時，此時 Guard 須送出由 Hi → Low 的電位，此時 'B' 的電壓會些微下降。

Guard 訊號與觸控掃描是同步的。



綠色為 Guard Ring 的輸出波型　　黃色部分為外部觸控感應點的電壓波型

MICROCHIP

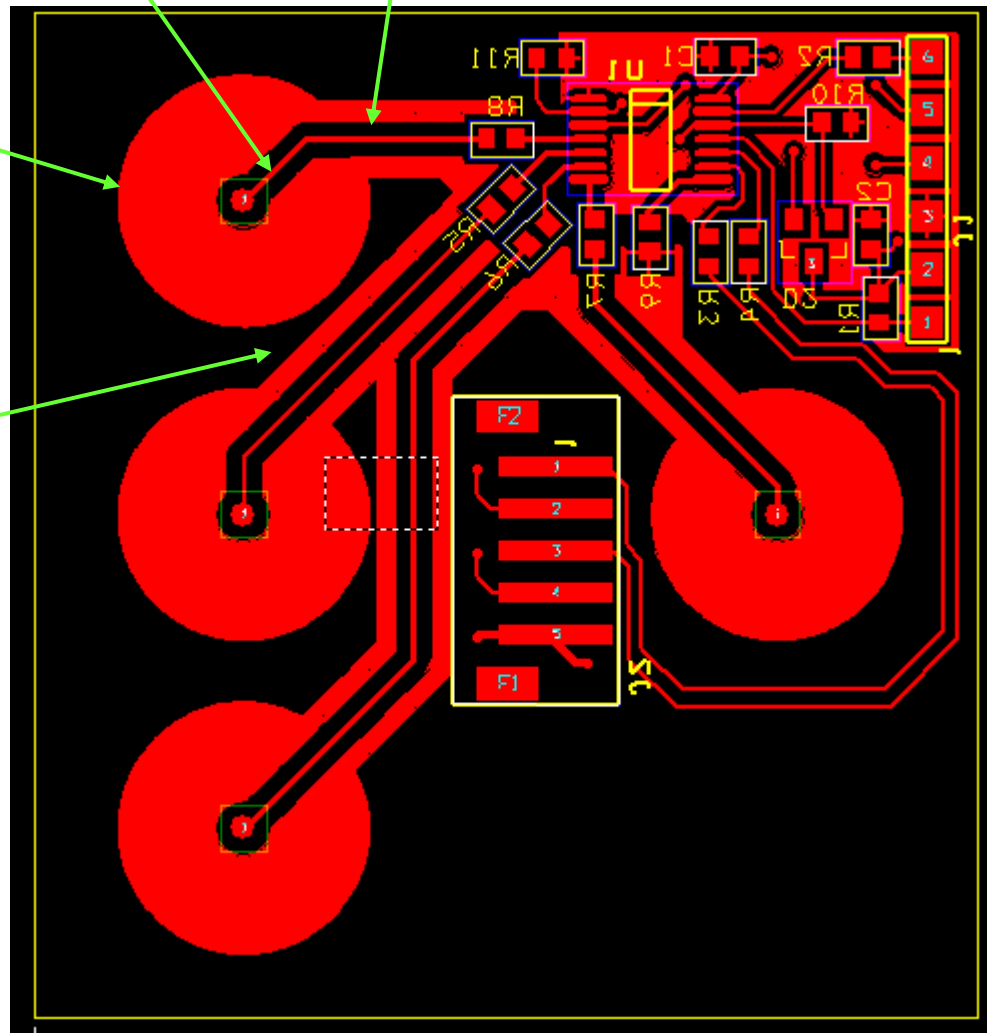
PCB 背面的防護配置布局:

感應點的連線: 0.15-1.2mm

0.2-0.4mm 的隔離間隔

防護驅動: 與正面的觸控感應點相同的直徑

驅動線徑 :
0.2 - 0.5mm



MICROCHIP

PCB 正面的防護配置布局

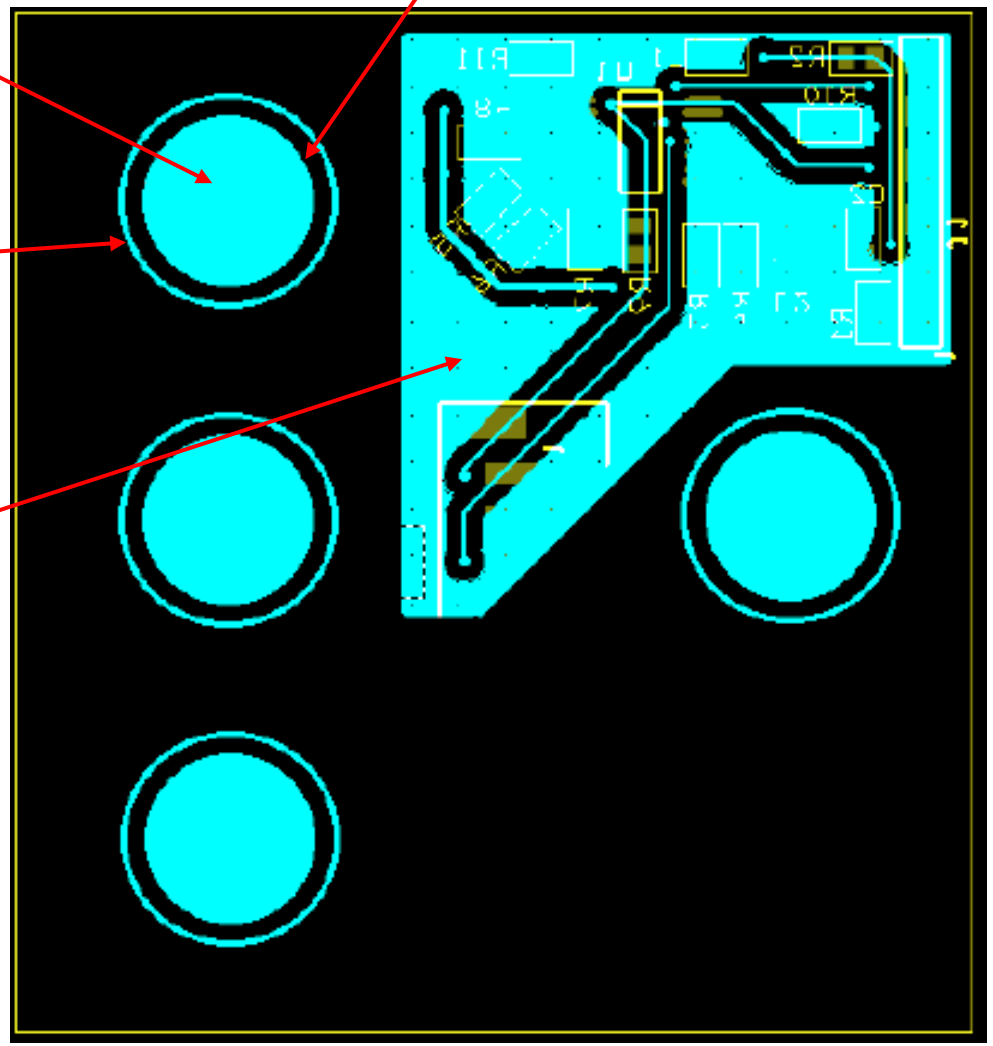
0.3-2mm 感測點和防護環之間的隔離

觸控感應點
直徑 >= 5mm

Guard 防護環寬度約
0.3 – 1.5mm 以灌孔
方式連接到背面

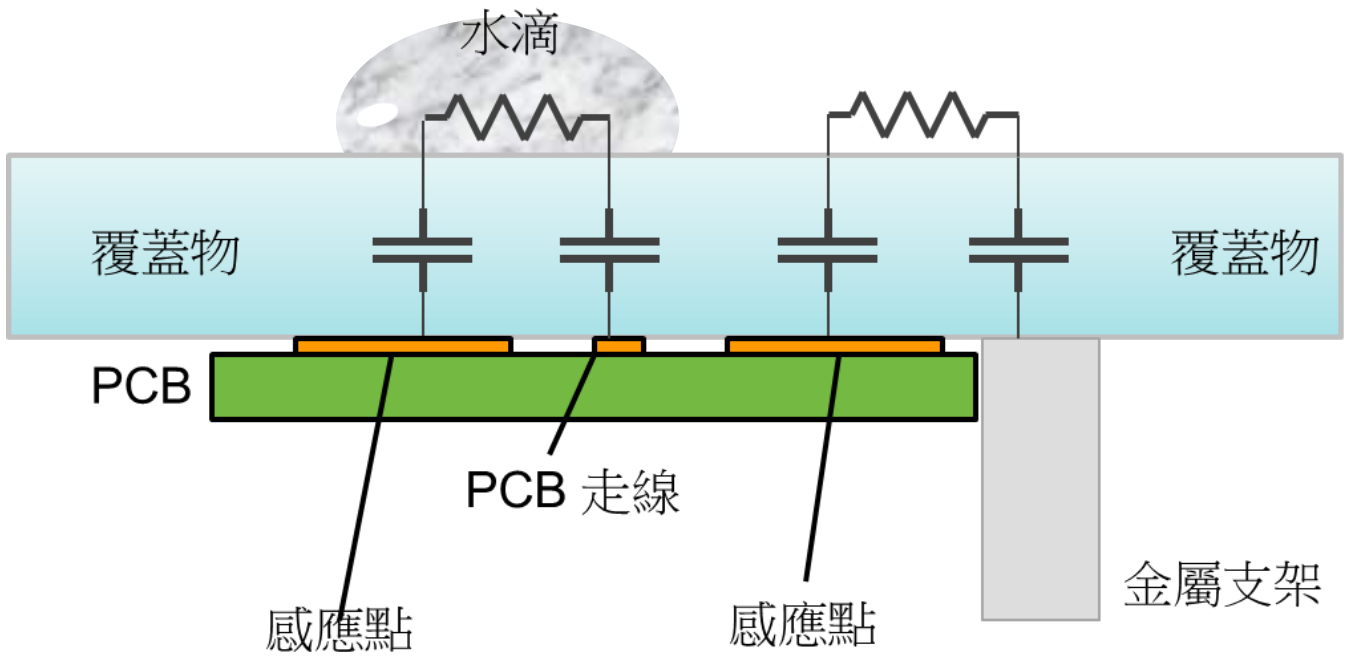
GND 地網隔離
(可降低雜訊的干擾)

LED 可以位於觸控感應點的中間，以挖孔的方式做顯示。



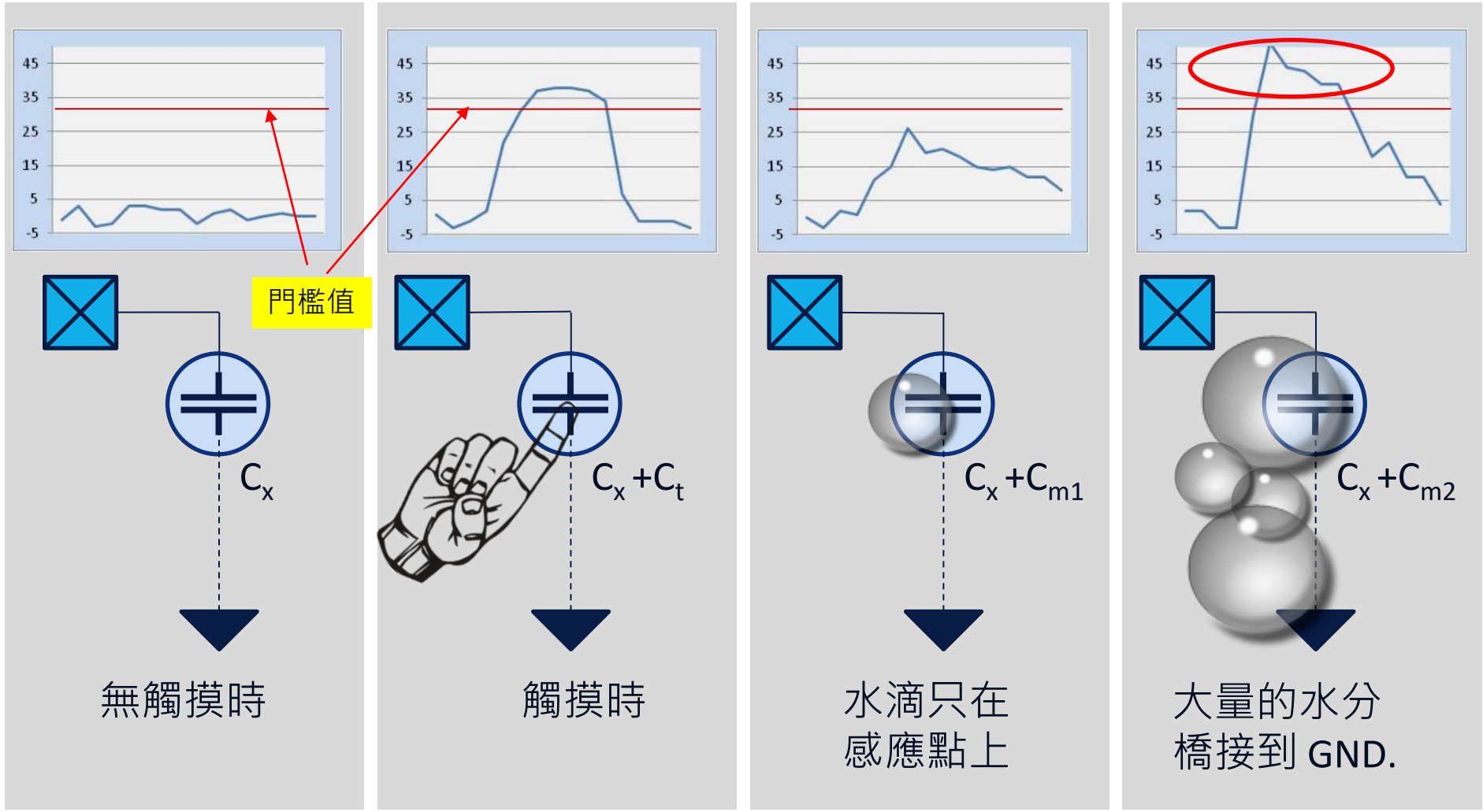
非純水是一種電容感應路徑

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) 的現象，易造成一種有觸控輸入的誤動作。



MICROCHIP

負載和水所造成的自感電容的變化

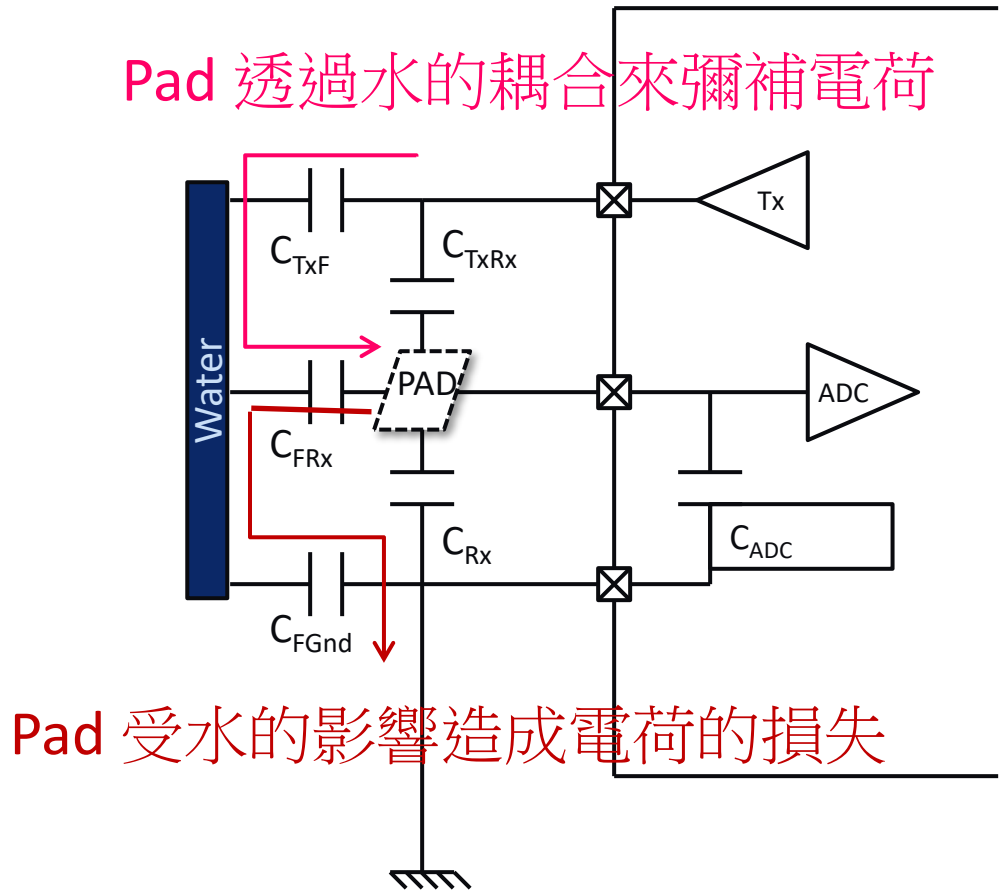


MICROCHIP

互感電容

更多的電荷回饋到感應器

減少感應器電荷的的損失

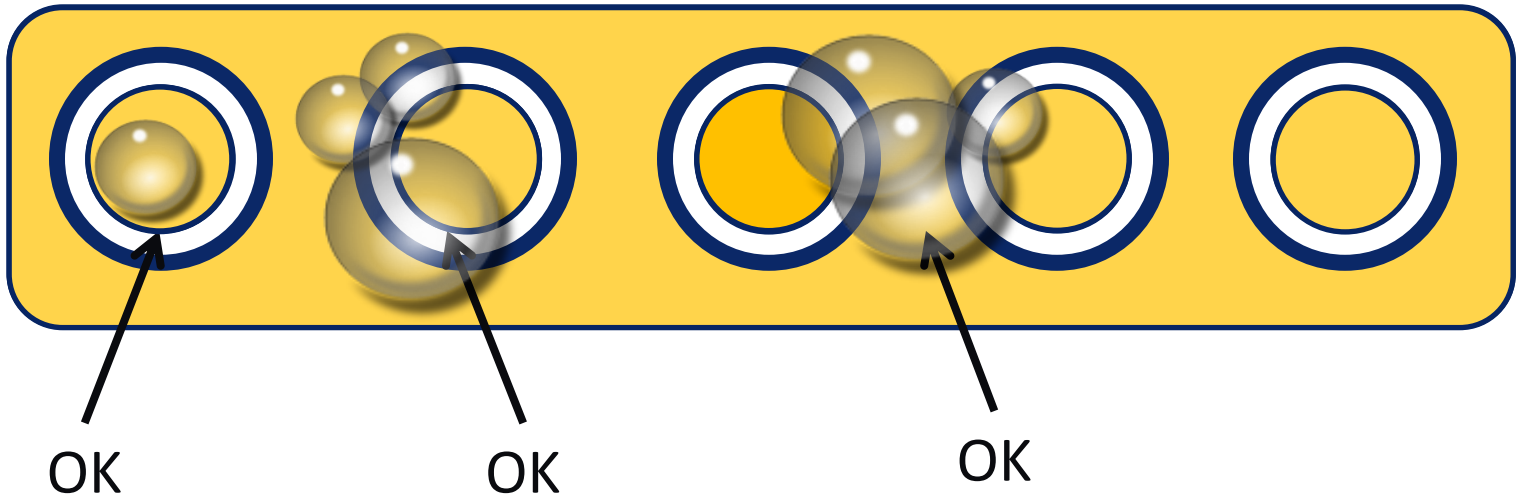


MICROCHIP

主動防護盾 (Active Guard Shield)

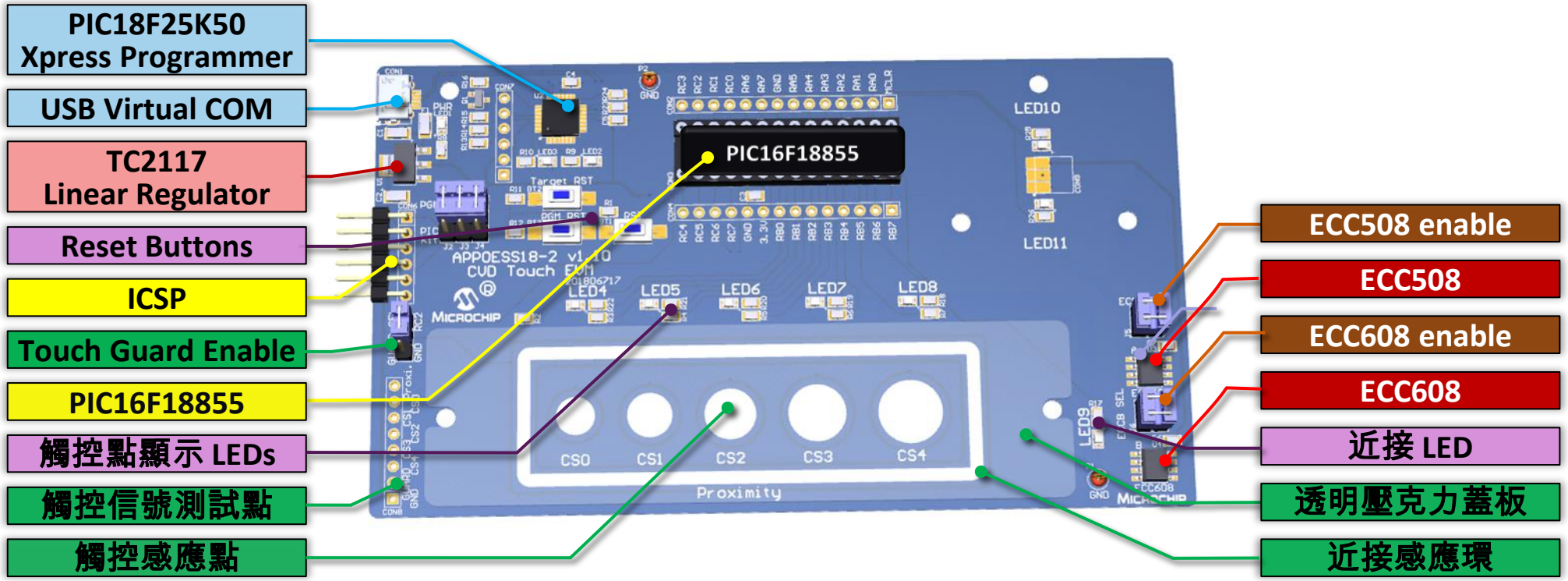
Microchip Style

- 使用保護環 (Guard Ring)方式與其它電極使用相同的電位來驅動掃描
- 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。



MICROCHIP

APP-ESS18-2 mTouch EVB 介紹



APP-ESS18-2 mTouch EVB

中文實做手冊

- 快速、簡單實現 mTouch 的設計
- 範例說明: PIC16F15355 PIC16F18
- 循序漸進的設計指引方式

MICROCHIP

探討 mTouch 16F15355 Callback.X 的專案

- **mTouch 中文實作手冊練習有:**
  - 最基本的範例 : mTouch 16F15355 Polling.X
  - 使用 Callback 的範例 : mTouch 16F15355 Callback.X
  - 使用 Data Visualizer 的範例 : PIC16F18855 CVD Lab3.X

- **課程將介紹**
  - 主要的 MCC 設定項與 mTouch 的關係
  - 基本 MCC Callback 函數的使用說明
  - mTouch 16F15355 Callback.X 程式架構
  - Demo 1 : APP-ESS18-2 mTouch 實驗板的簡易燒錄與執行
  - Demo 2 : Tera Term 終端機模擬程式與虛擬 COM Port 的連接
  - Demo 3 : 檢查並調整 mTouch 的參數
  - Demo 4 : 以 Data Visualizer 分析 mTouch 資料: PIC16F18855 CVD Lab3.X

MCC 設定項與 mTouch 的關係

以 mTouch 16F15355 Polling.X 專案為例

- **很重要的一點 : CVD 是使用 ADC 測量平衡後的電壓值，所以觸控感應點的配置一定要選擇有 ADC 輸入功能的腳位**
  - 本範例使用 RB0 ~ RB5 做為 mTouch 的輸入 (參考 Page 35)
- **mTouch 的主要設定是在 “Easy Setup” 的選卡項**
  - Button and Proximity Configuration (參考 Page 34 ~ Page 42)
  - Guard 防護設定輸出 (參考 Page 43)
  - AFT (自動頻率調配) 按鍵掃描跳頻設定 (參考 Page 44)
  - AFT 內定使用 Timer2 並會開啟中斷
- **UART (Procotal : 115200bps, N,8,1)**
  - 設定觸控資料的輸出，透過 UAB to UART 的虛擬通訊來顯示資料 (參考 Page 45)
  - 需使用到 Tera Team 終端機模擬程式來顯示
- **Timer1 (使用中斷)**
  - 提供 500mS 的時間做送出終端機的顯示資料 (參考 Page 46)
  - 使用 Callback 方式轉移中斷控制權給 Timer1 的一般函數使用

基本的 Callback 方式使用

以 mTouch 16F15355 Polling.X 專案為例

Callback 採用函數指標方式做執行權的轉移

1. **MCC 在 Timer1 的原型宣告 (Prototype)**
   - void TMR1_SetInterruptHandler(void (* InterruptHandler) (void)); Tmr1.h
2. **MCC 在 Timer1 的 “函數指標” 原型宣告**
   - void (*TMR1_InterruptHandler)(void); Tmr1.c
3. **使用時，主程式需將要執行的函數註冊**
   - TMR1_SetInterruptHandler(MyEvent);
   - 透過函數位址的轉移 MyEven( ) 將取代 TMR1_InterruptHandler( )
   - MyEven( ) 函數就可以獨立執行中斷功能，計數 500mS

如此: 透過函數指標的宣告，在中斷裡就可以用指標方式將執行權轉移到一般的函數來執行

Timer1 的中斷處理函式 (Tmr1.c)

```
void TMR1_ISR(void)
{
   // Clear the TMR1 interrupt flag
   PIR4bits.TMR1IF = 0;
   TMR1_WriteTimer (timer1ReloadVal);

   if (TMR1_InterruptHandler)
   {
      TMR1_InterruptHandler();
   }
}

void TMR1_SetInterruptHandler(void (*
InterruptHandler)(void)){
   TMR1_InterruptHandler = InterruptHandler;
}
```

ISR 中斷執行函數指標指向 → User 的函數位址 (MyEven)

mTouch 的 Callback 使用

以 mTouch 16F15355 Callback.X 專案為例

- 使用 mTouch Callback 功能的方式，請參考 Page 55 ~ 57 的說明

- 如同前面使用 Timer1 的 Callback 的設定方式一樣，只不過 mTouch 的 Callback 使用方式較為複雜

mTouch 所產生的四種 Callback 函數的原型宣告

- 底下兩項為處理按鍵的 Callback 函數
- void MTOUCH_Button_SetPressedCallback (void (*callback)(enum mtouch_button_names button));
- void MTOUCH_Button_SetNotPressedCallback(void (*callback)(enum mtouch_button_names button));

- 底下兩項為處理近接感應的 Callback 函數
- void MTOUCH_Proximity_SetActivatedCallback (void (*callback)(enum mtouch_proximity_names prox));
- void MTOUCH_Proximity_SetNotActivatedCallback(void (*callback)(enum mtouch_proximity_names prox));

mTouch Callback 函數使用說明 (參考 Page 55)

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

1. MTOUCH_Button_SetPressedCallback (函數指標) : 這是在 mTouch 的 callback 主執行函數。在這裡只需傳入在主程式裡處理觸控按鍵的函數指標。
2. *callback : 主程式裡處理觸控按鍵的函數指標。如此指標型態的 callback 跳到這應用函數來執行。這裡傳入的參數是一個指標位址，也就是該應用程式的執行位址。
3. 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。
4. 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
   };
```

在 main.c 使用 mTouch Callball 的步驟

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。

   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

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

   MTOUCH_Button_SetPressedCallback (processButtonTouch);

   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_SetNotPressedCallback(processButtonRelease);

   將應用端的 **processButtonRelease()** 函數指標傳給 **mTtouch** 函數庫裡的 **MOTUCH_Button_SetNotProcessedCallback()** 函數。

3. 觸控應用函數的撰寫

   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**

   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**

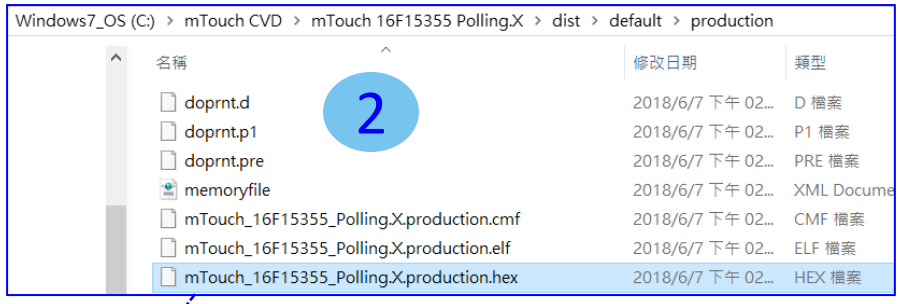
APP-ESS18-2 mTouch 實驗板的簡易燒錄與除錯 (參考 Page 50)

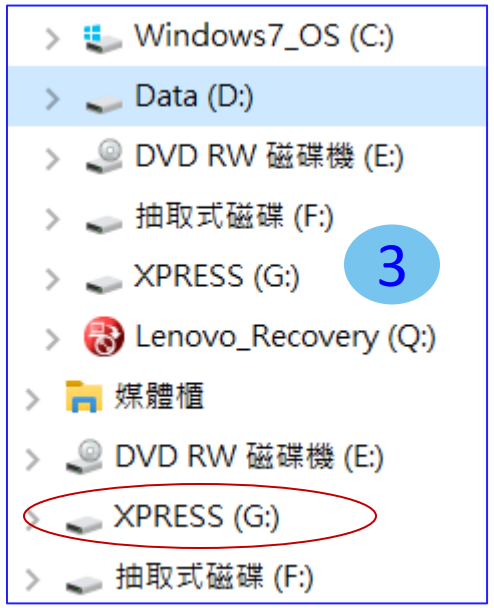
- **APP-ESS18-2 mTouch 實驗板有兩個燒錄介面**
  - 以標準的開發工具，SNAP、PK3、PK4 連接到 6-pin 排針接頭 (CN6)
  - 利用 PIC18LF25K50 所提供的虛擬 USB to UART 的燒錄
  - 注意: 實驗板上有一 DBG_SEL 的選擇，如要使用虛擬燒錄請將三個排針短路到上方的 PGM，如要使用除錯工具請短路到下方的 PICkit

Demo 1 : APP-ESS18-2 mTouch 實驗板的簡易燒錄與執行

使用實驗板所提供的 USB 燒錄

1. 將 Micro USB Cable 連接電腦的 Type A USB 及實驗板的 Micro USB
2. 開啟 “**mTouch 16F15355 Polling.X**” 專案，直接編譯成功後會產生一個**Hex** 檔
3. 開啟檔案管理員，檢視是否有多一個 **XPRESS(F:)** 的儲存裝置。
4. 將 “**mTouch 16F15355 Polling-production.hex**” 拉到 **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。





Demo 2 : Tera Term 終端機模擬程式與虛擬 COM Port 的連接

說明如何設定 Tera Term 終端機及連線

使用 UART 觀察 mTouch 的按鍵輸出資料

- 設定 UART 的通訊格式 : 115200bps, N, 8 ,
- 在 main.c 下的顯示程式

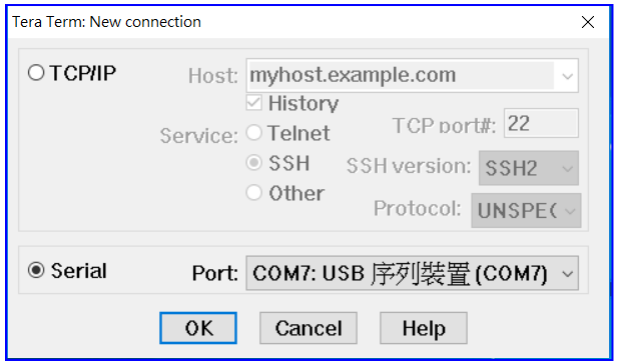
```
if (Delay_Count == 0) {
        printf("Proxi0= %3d   %5d      | ", MTOUCH_Proximity_Deviation_Get(Proxi0),
        MTOUCH_Proximity_Reading_Get(Proxi0));
        printf("CS0= %5d  %3d", MTOUCH_Button_Reading_Get(0), MTOUCH_Button_Deviation_Get(0));
        printf("      | ");
        printf("CS4= %5d  %3d", MTOUCH_Button_Reading_Get(4), MTOUCH_Button_Deviation_Get(4));
        printf("\r\n");                    // 換行與在第一位置開始顯示
        Delay_Count = 50;                  // 設定 500mS 的顯示時間  (50 x 10mS)
    }
```

主要是觀察觸控按鍵的差異值 (Deviation) 為主 (差值的範圍 0 ~ 127 之間，接近 0 的數值是沒有觸控的)

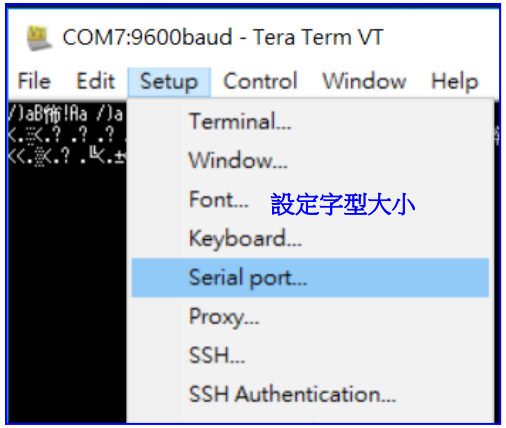
啟用 Tera Team 終端機模擬程式 (參考 Page 51 ~ 52)

- 安裝虛擬通訊 PIC18LF25K50 的 CDC USB 驅動程式
- 請上網 www.microchip.com/mcp2200 網頁下載:
  - MCP2200/MCP2221 Windows Driver & Installer
- 安裝方式請參考 **Page 51** 的步驟

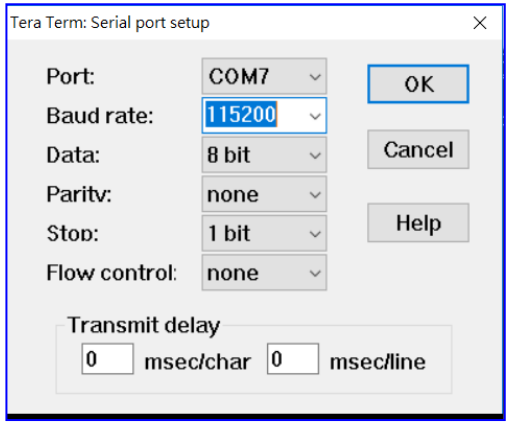
- 虛擬 COM Port 的建立
- 開啟 Tera Team 並設定



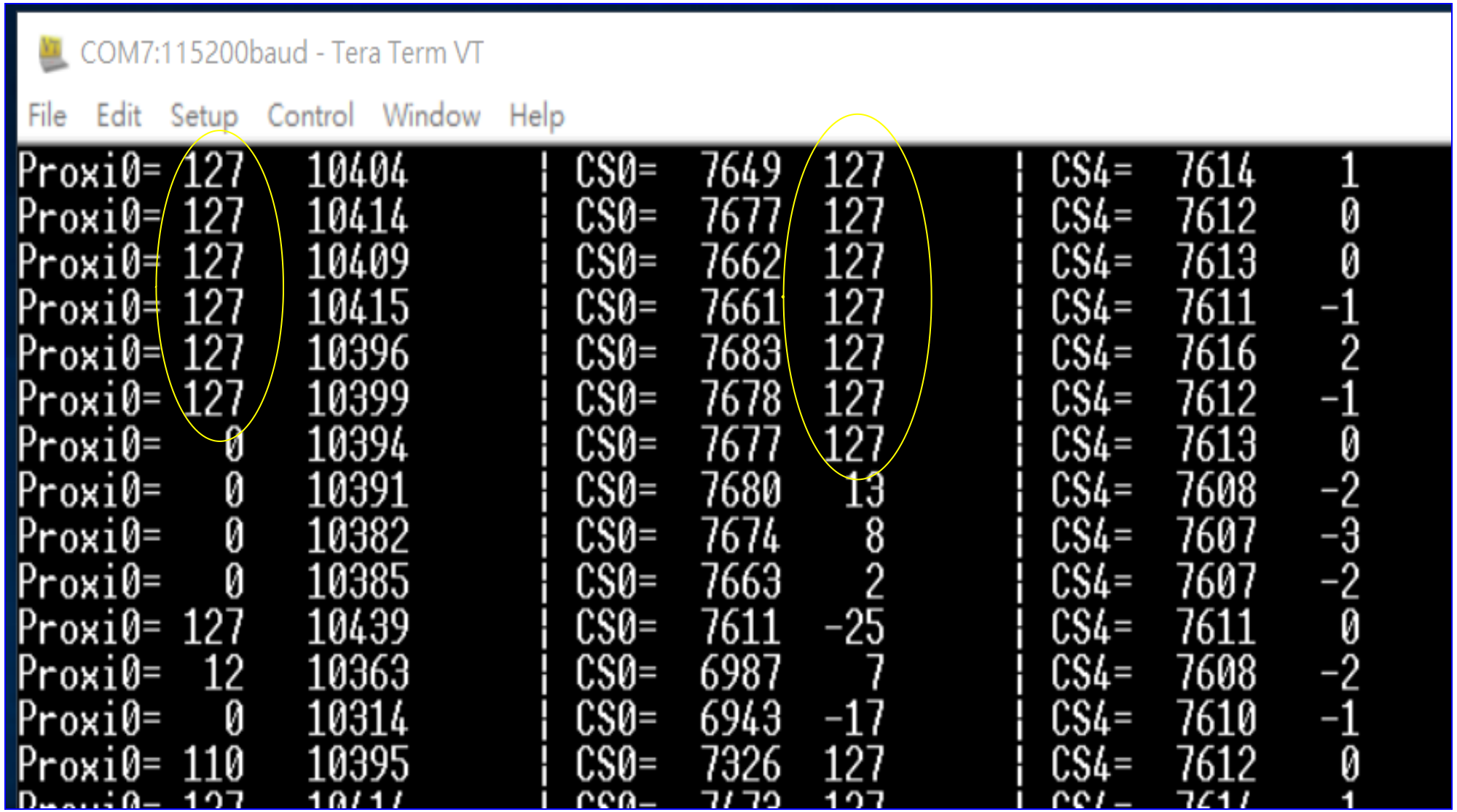
設定Com Port 為 COM7



在 Setup 選項設定通訊協定



Demo 2 : Tera Term 終端機模擬程式與虛擬 COM Port 的連接



Demo 3：檢查並調整 mTouch 的參數

在 Tera Term 終端機視窗下，
檢視 Proximity (近接感應) 的數值變化及觸控按鍵 CS0 的變化

Demo 4 : 以 Data Visualizer 分析 mTouch 資料 PIC16F18855 CVD Lab3.X

如何設定及使用 Data Visualizer
資料顯示及圖形顯示

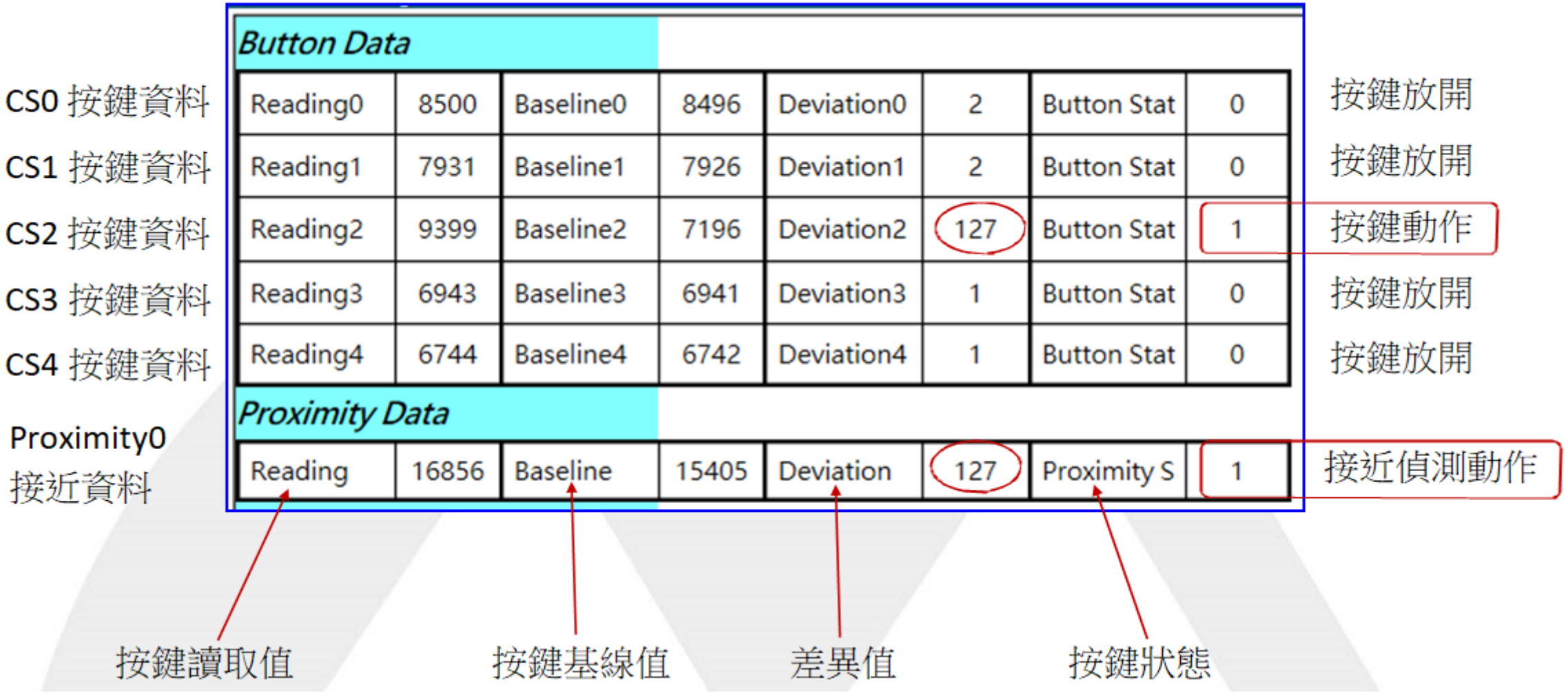
Data Visualizer (數據視覺化)

- 源自 Atmel Studio 7
- 現有 Standalone 版本可以使用

- 因受限於這次所提供的 UART 傳送資料格式限制，本範例示範限用
  - Standalone Data Visualizer v1.15.651
  - 下載: https://gallery.microchip.com/packages/AtmelDataVisualizerInstaller-Standalone/
- Data Visualizer 的軟體必須要安裝後再執行
  - (AtmelDataVisualizerInstaller-2.15.651.exe)

- 關於 Data Visualizer 的使用請參考手冊的 Page72
  - 在 “裝置管理員” 找出 “USB 序列裝置 (COMxx)”
  - 連接並設定資料路徑取出資料傳輸格式
  - 自動搜索傳輸速率並連線

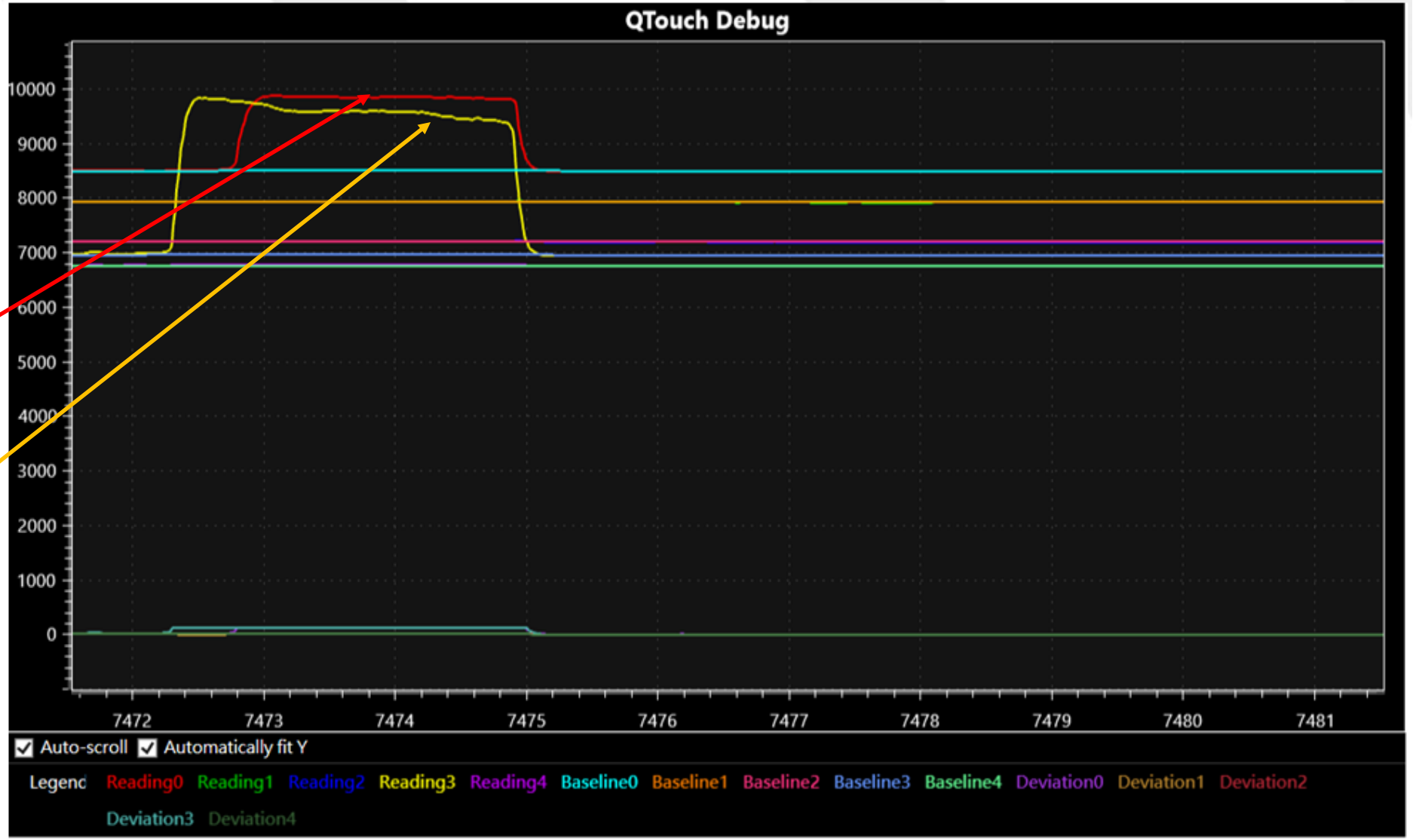
Data Visualizer 的觸控按鍵資料顯示



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

Data Visualizer 的觸控按鍵圖形顯示

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。



Thank You !!

Trademarks

MICROCHIP