MICROCHIP

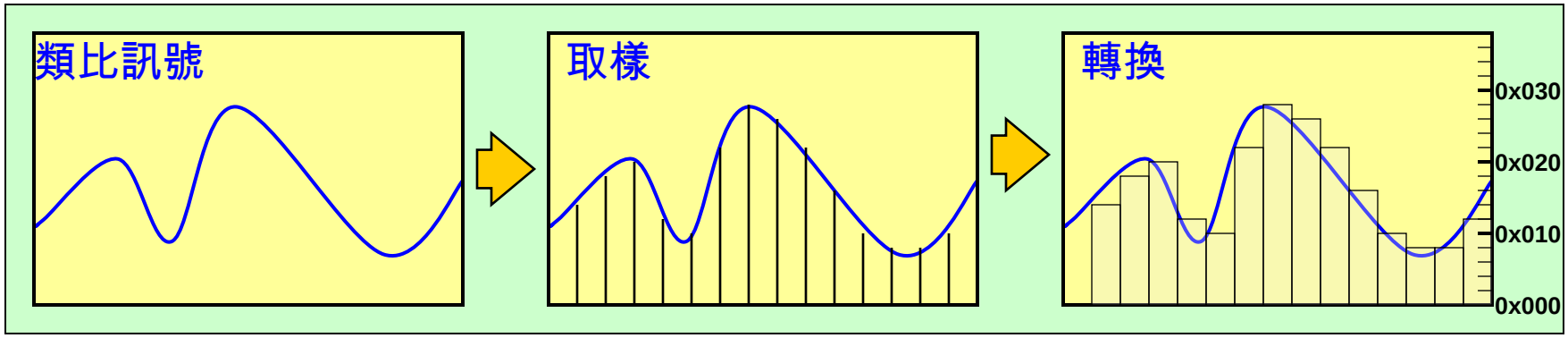
Regional Training Centers

Section 9
10 Bits ADC

MICROCHIP
Regional Training Centers

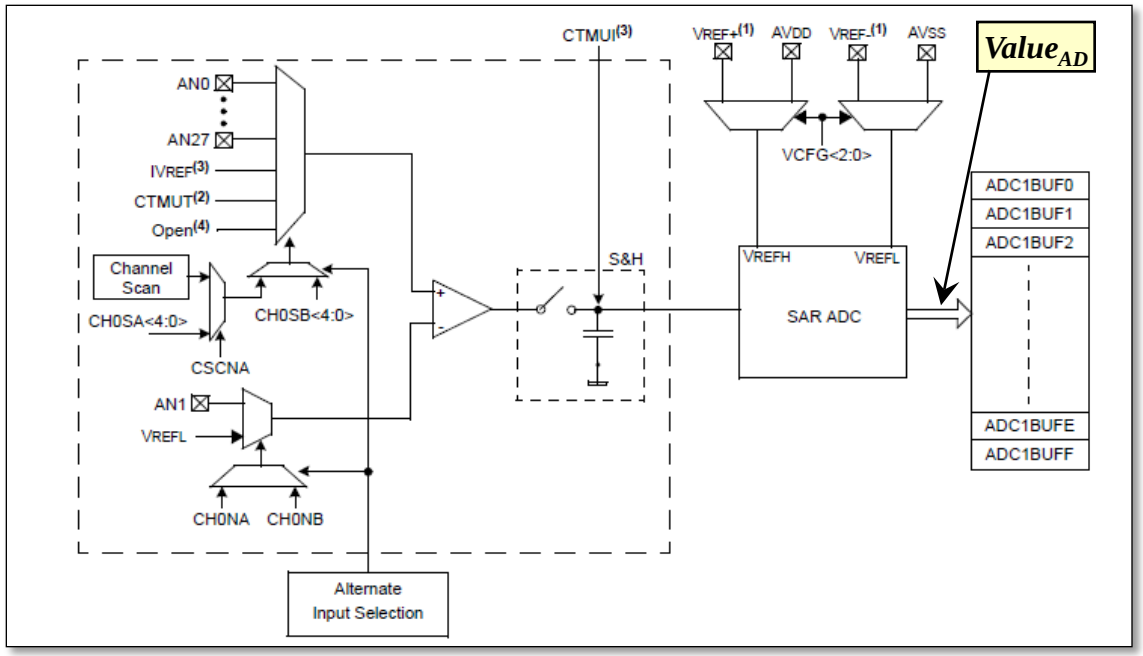
What's ADC ?

- ADC：Analog Digital Conversion,類比數位轉換器。
  一個可將類比訊號轉換成數位資料的模組。
- ADC的轉換過程,可以分為兩個步驟,如下圖所示,首先對類比信號進行"取樣",利用外部的類比訊號對ADC內部的小電容充電,已取得外部類比訊號的複本,接著再將獲得的資料加以"轉換",獲得量化後的數位資料。



PIC32's ADC Architecture

- PIC32MX470具有一組採用SAR(連續近似法)的10-Bits ADC。搭配32對1之類比多工器, 達成多通道轉換功能。
- 具兩組多工器, 可交替使用, 多工器A支援通道掃描轉換功能。
- 類比信號轉換結果為



$$Value_{AD} = \frac{V_{AD} - V_{R-}}{V_{R+} - V_{R-}} \times 2^n$$

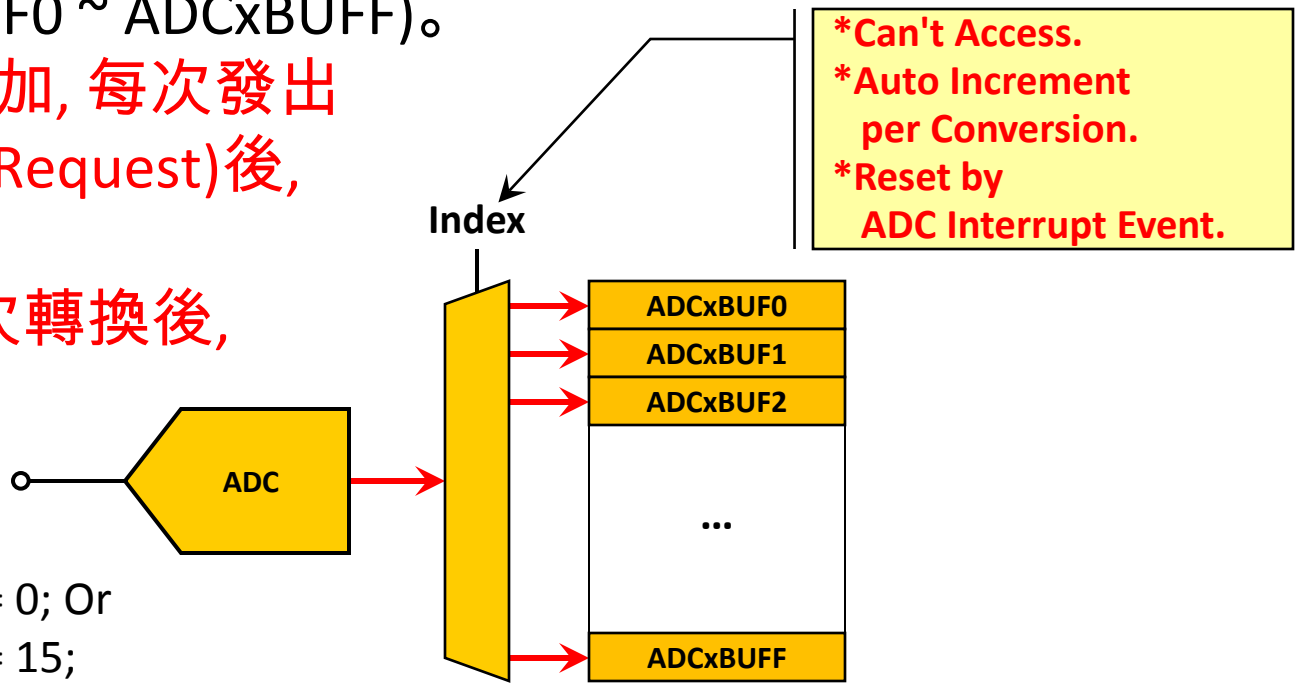
- V_{R+}, V_{R-}可使用V_{REF+}, V_{REF-}或AV_{DD}, AV_{SS}。

MICROCHIP
Regional Training Centers

PIC32's ADC Architecture

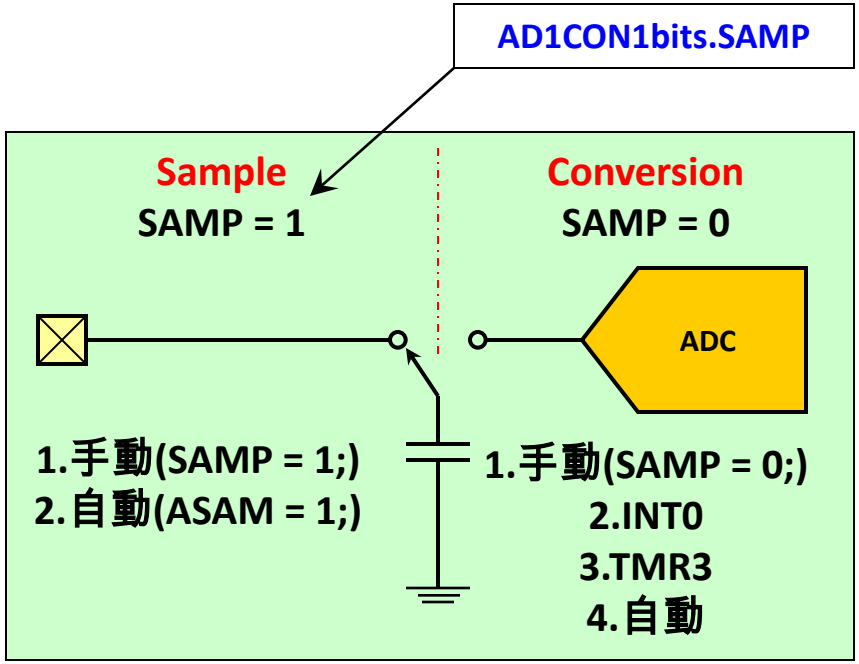
- ADC內建Buffer(ADCxBUF0 ~ ADCxBUFF, 共16個), 用以儲存轉換所得之結果。資料格式共有八種可選擇(有/無號, 小數/整數, 32/16 Bits) 。
- Buffer帶有Index, Index用來指定要將轉換後結果存入哪個Buffer (ADCxBUF0 ~ ADCxBUFF)。
- Index會自動累加, 每次發出中斷(Interrupt Request)後, Index會歸零。
- ADC可設定幾次轉換後, 發出中斷需求(1 ~ 16次)。

For Example:
AD1CON2bits.SMPI = 0; Or
AD1CON2bits.SMPI = 15;



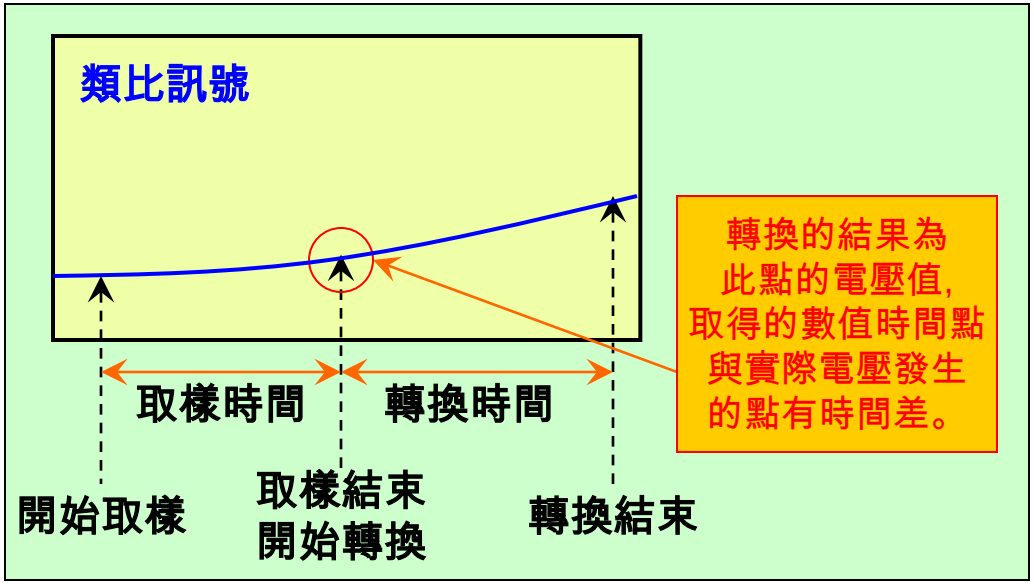
ADC Sample, Conv. Trigger Source

- ADC的轉換其實一點都不複雜, 所有動作其實都圍繞在SAMP位元(A/D Sample Enable bit), SAMP=1時進行取樣, SAMP=0則進行轉換。
- ADC的動作必須先進行取樣, 再進行轉換。
- ADC有2個”取樣“觸發源: 自動,手動。
- ADC有4個”轉換“觸發源: 手動,自動,外部中斷或Timer3 Match。



Sample and Conversion Sequence

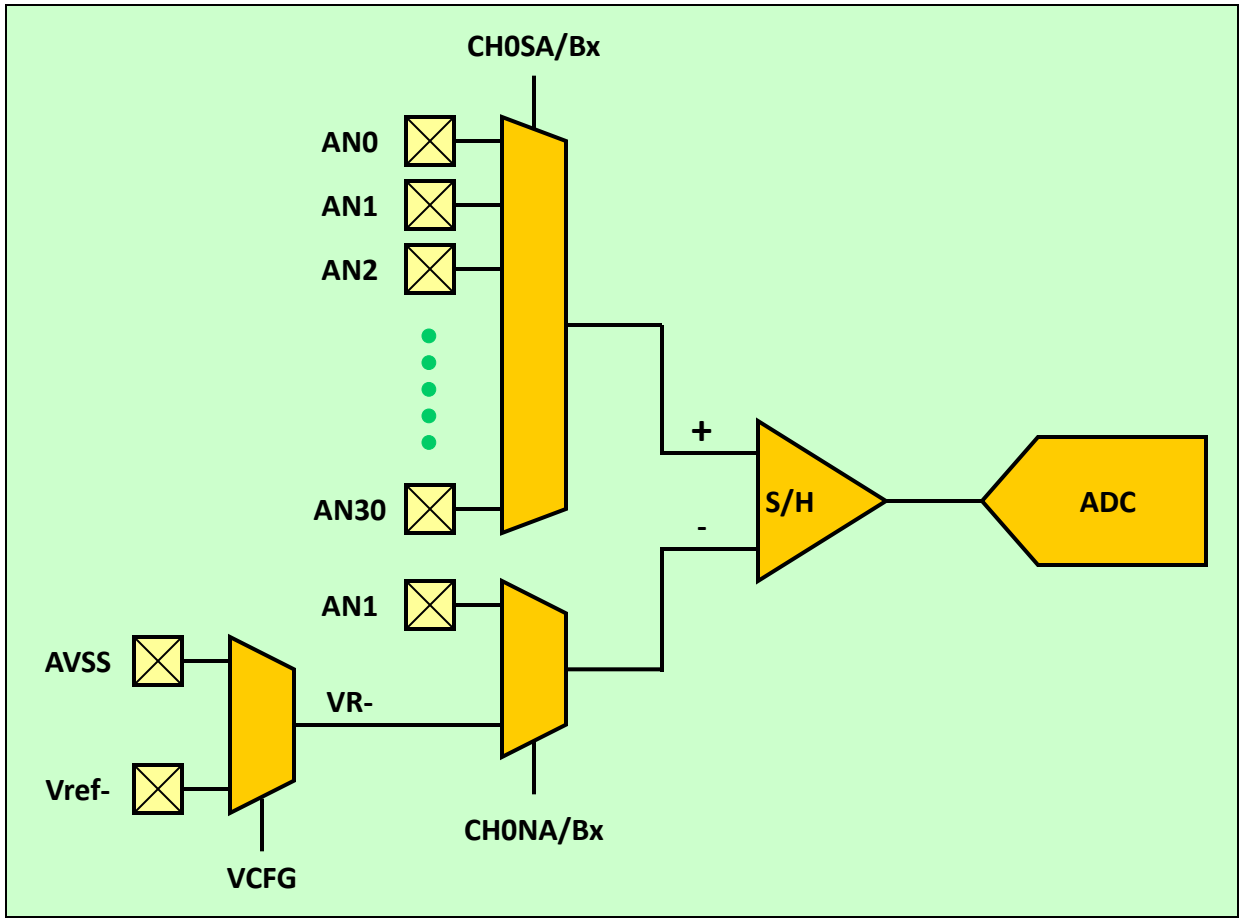
- SAR ADC作動分為"取樣"及"轉換"兩步驟,
  取樣時利用外部訊號對ADC內部的電容器充電,取得外部電壓值。
  轉換時依據取得的電壓值換算出結果。



- ADC的取樣時間與轉換時間都有最短需求時間的規範,設計時必須滿足才能確保轉換結果正確。時間規範可查詢Datasheet電氣特性章節。
- PIC32MX系列的取樣時間必須大於$1T_{AD}$(詳細規範請參考Datasheet),轉換時間通常需大於$12T_{AD}$($1\ T_{AD}$ =65nS) 。

ADC Channel Select

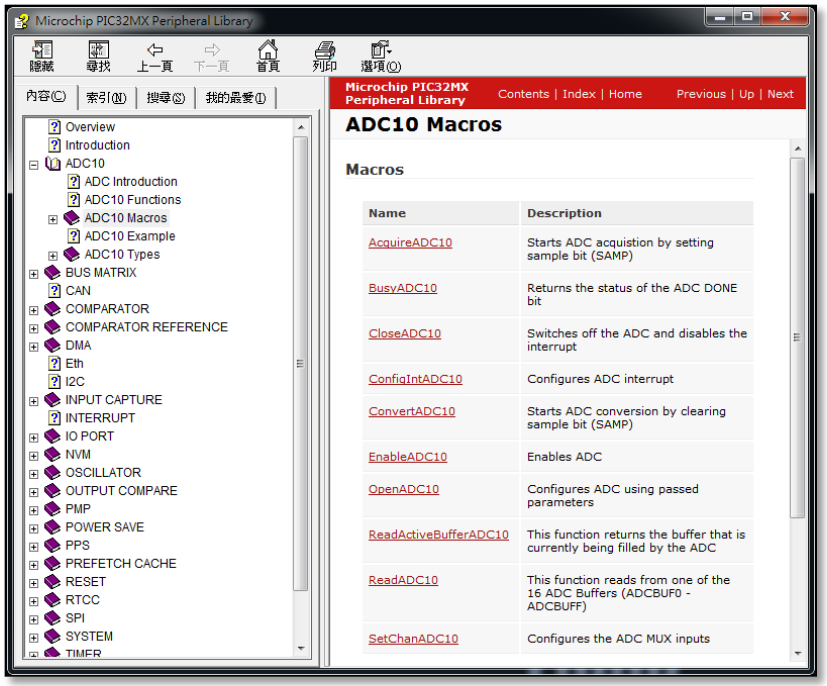
- ADC的正端輸入可以選擇AN0~AN30。負端輸入則可選擇連接VR-(V_{REF-}或AV_{SS})。
- 負端輸入也可選擇連接到AN1,讓兩訊號相減後,再送入ADC(單端差動模式)。
- 單端差動,類似”差動”的概念,但兩者有相同的地(GND)。



XC32 ADC Function & Macro

- 常用的ADC函數

```
OpenADC10( );                    // 啟用ADC,設定ADC工作模式。
SetChanADC10( );                 //手動切換ADCx的取樣通道。
ConvertADC10( );                 //手動觸發轉換( SAMP -> 0 ) 。
BusyADC10( );                    // 測試ADC是否忙碌中?
ReadADC10( );
  // 自AD Buffer讀取AD轉換後的結果。
ConfigIntADC10( );
  // 設定AD的中斷優先權,中斷啟用。
...
```



MICROCHIP
Regional Training Centers

ADC Open Example

- ADC10的初始化範例:

void OpenADC10(unsigned int *config1* , unsigned int *config2* ,
unsigned int *config3* ,
unsigned int *configport*,
unsigned int *configscan*);

config1 , *config2* , *config3* : ADC10的工作模式,
configport : 設定類比通道的運作模式(Digital Mode / Analog Mode),
configscan : 設定開啟通到掃描模式時, 要掃瞄的通道。

Ex:

```
OpenADC10(ADC_MODULE_ON | ADC_IDLE_CONTINUE | ADC_FORMAT_INTG16 |
          ADC_CLK_MANUAL | ADC_AUTO_SAMPLING_ON | ADC_SAMP_OFF,
          ADC_VREF_AVDD_AVSS | ADC_OFFSET_CAL_DISABLE | ADC_SCAN_OFF |
          ADC_SAMPLES_PER_INT_1 | ADC_ALT_BUF_OFF | ADC_ALT_INPUT_OFF,
          ADC_SAMPLE_TIME_31 | ADC_CONV_CLK_SYSTEM | ADC_CONV_CLK_32Tcy,
          ENABLE_AN19_ANA ,
          SKIP_SCAN_ALL );
```

MICROCHIP
Regional Training Centers

ADC Select Channel Example

- ADC1手動切換取樣通道的範例:

  void SetChanADC10(unsigned int *channel0*);

  channel0 : 設定Channel0的多工器A及多工器B的正負端輸入,

  Ex:

```
SetChanADC10(ADC_CH0_POS_SAMPLEA_AN19 |
             ADC_CH0_NEG_SAMPLEA_NVREF |
             ADC_CH0_POS_SAMPLEB_AN0 |
             ADC_CH0_NEG_SAMPLEB_NVREF );
```

MICROCHIP
Regional Training Centers

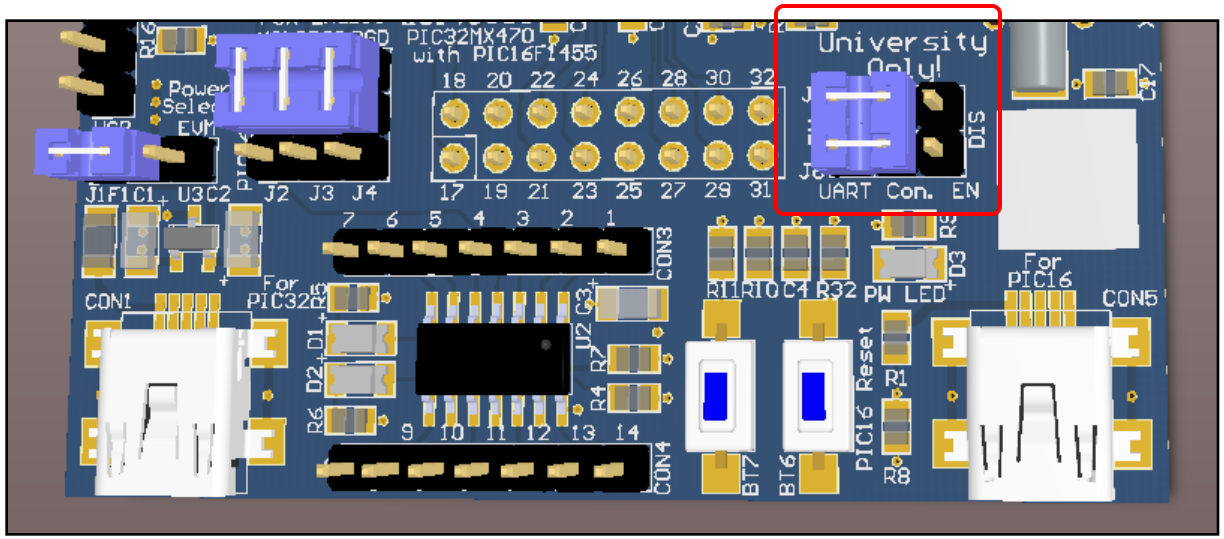
How to Display
Additional Information ?

- APP028-1並沒有提供太多的顯示介面。但PIC16F1455內建有USB的Module, 因此我們利用USB Module實作USB CDC類別的USB to UART Bridge(USB Serial Emulator)。
- PIC32可以透過UART將資料送給PIC16F1455, 再經由USB Serial Emulator送給PC。
- PC上則可以透過超級終端機軟體來顯示資訊。
- PIC32上相關的初始化程式已經完成, 只需使用printf( )函式, 就可以將資訊透過USB Serial Emulator送給PC。
- For Example

```
printf( "Hello, This is Microcihp PIC32 Demo\r\n" );
```

USB CDC Emulator

- Tx腳為Pin6(RC4), Rx腳為Pin5(RC5)。D1, D2分別會在傳送與接收資料時閃爍, 指示目前資料傳輸的情形。
- 記得要連接右側的USB Connector 才能使用USB to UART Bridge(USB Serial Emulator)的功能。
- 還要確認J5, J6的位置設定在EN, 連接PIC232與PIC16的UART, 才能傳遞資料。

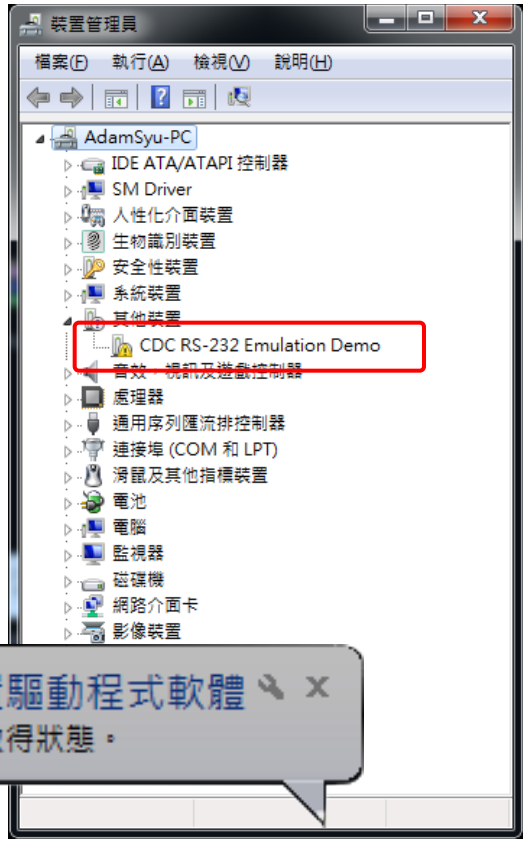


Driver Install for USB CDC Emulator

- USB CDC Emulator的韌體已經預先燒錄。
- 使用Mini USB Cable連接CON5, 此時PC會出現”找到新硬體”的訊息, 接著請安裝該裝置的驅動程式,

  (Path:\USB Serial Emulator Driver\mchpcdc.inf)

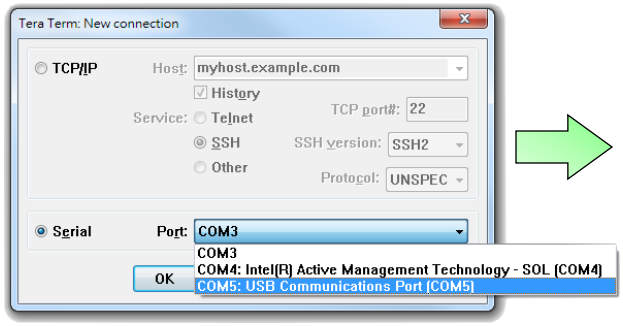
  安裝後會多出一個COM Port,我們將透過這個COM Port來與PIC32的UART溝通。

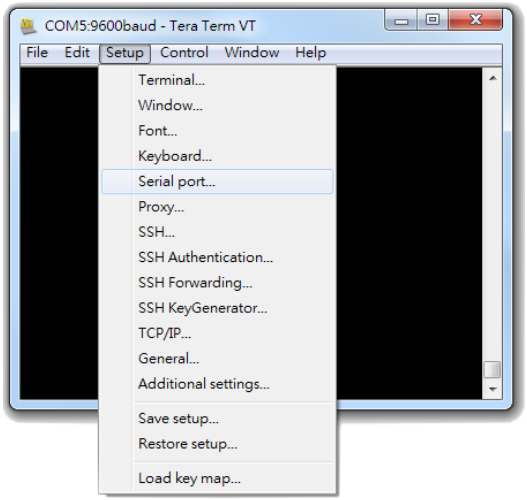


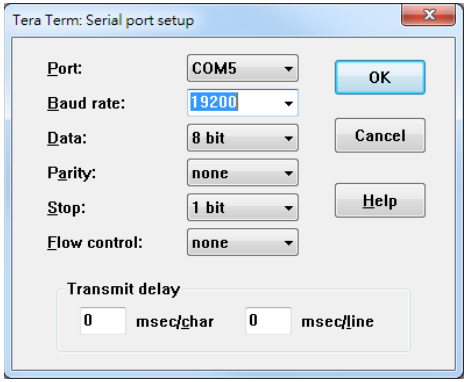
Setup Terminal

Tera Term

- 由於Windows XP後的OS不再提供超級終端機, 如果使用Windows 7, 8等OS, 必須先安裝替代軟體, 才可以監控COM Port的狀態。
- 請先安裝Tera Term, 然後開啟然後Tera Term, 指定COM Port, 設定為**9600 , N , 8 , 1**。
  (Path:\Tools\ teraterm-x.xx.exe)







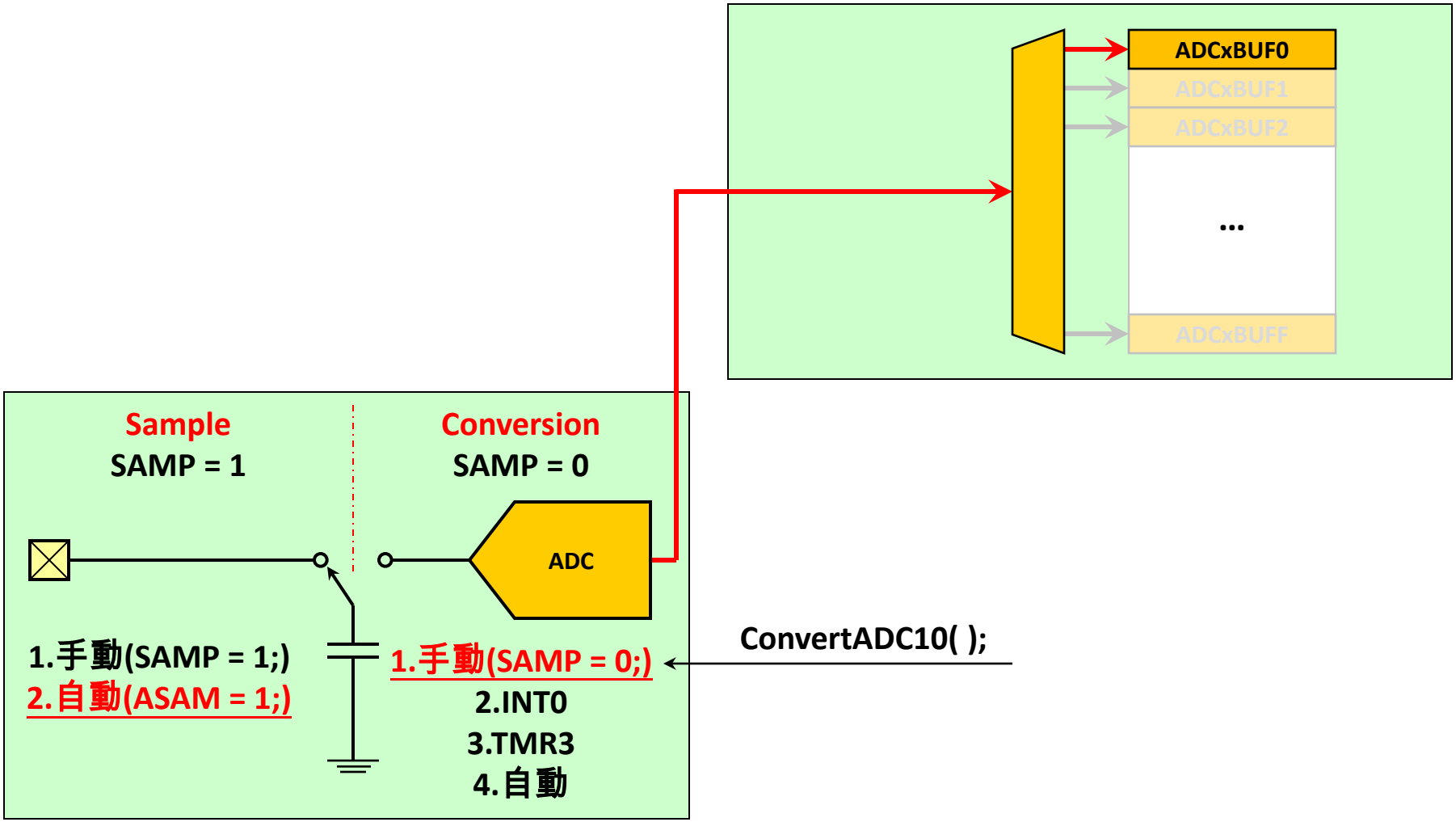
MICROCHIP
Regional Training Centers

Lab6 ADC Single CH Manually

- 在Lab5的程式基礎上, 嘗試加入ADC的程式片斷。利用ADC10來取得VR1的類比電壓值, 並透過printf( )轉換的結果顯示在PC上。
  ADC的工作模式設定為, 自動取樣, 手動轉換, 類比通道設定為AN19(VR1), 16Bits無號整數格式。
- ADC的設定比Timer複雜很多,設定上要更細心,先閱讀說明文件,了解ADC Function的使用方法。至少必須了解OpenADC10( ),SetChanADC10( ), ConvertADC10( ); BusyADC10( ),ReadADC10( ),用法。
- 使用Bootloader將程式燒錄進APP028-1。觀察程式執行的情況。驗證看看是否可以PC上看到VR1的值(0~1023)。

Lab6 ADC Single CH Manually

Block Diagram



MICROCHIP
Regional Training Centers

Lab6 ADC Single CH Manually *Step1*

- 程式ADC的工作模式, 通道選擇要如何設定?

  觀察OpenADC10( )與SetChanADC10( )裡面缺漏的參數, 然後查閱文件中有關ADC的說明, 將缺漏的部分補齊。

- 如何觸發ADC開始轉換?

  ADC被設定為手動觸發轉換模式, 所以必須自行呼叫來觸發ADC開始轉換ConvertADC10( )。
  程式中利用Timer1產生100mS的中斷事件, 請在Timer中斷中加入ConvertADC10( ), 讓ADC每100mS會進行一次轉換。

Lab6 ADC Single CH Manually *Step2*

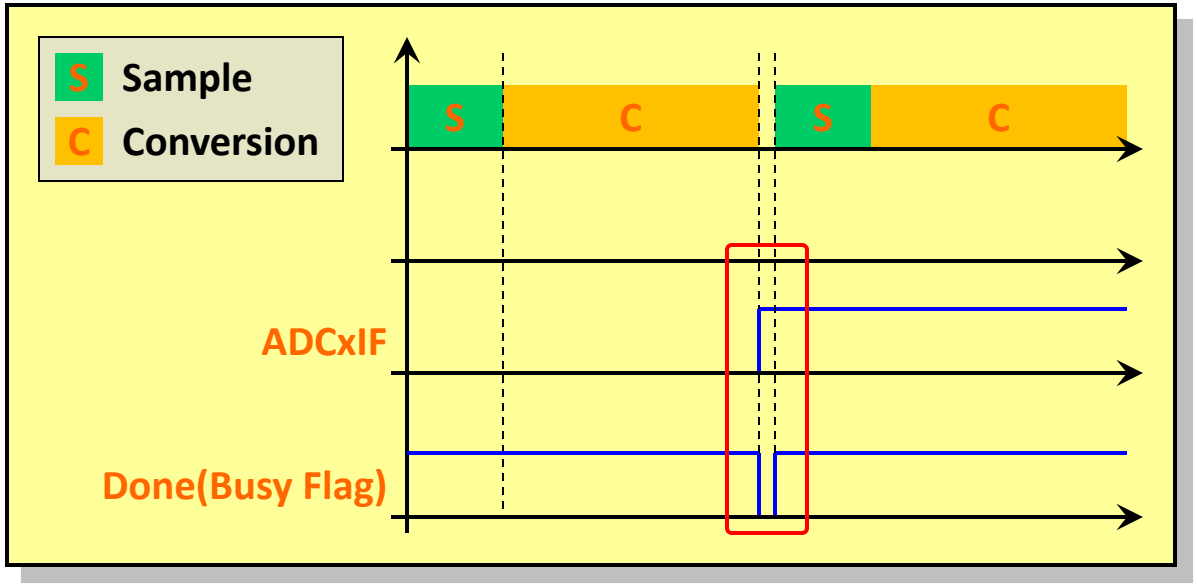
- 如何取得ADC轉換的結果?

  可以透過ReadADC10(*n*)取得ADCBUF*n*裡面的資料。
  讀取資料前必須先判斷
  ADC中斷旗標(INTGetFlag(INT_AD1);)或Busy Flag (BusyADC10( );)來確定ADC已經完成轉換, 不然無法取得正確的結果。

- Interrupt Flag跟

  Busy Flag的差異?

  當ADC連續動作時,
  前次轉換完成後,
  會馬上接續取樣。
  此動作, 會使
  Busy Flag被設置,
  有可能來不及判斷。

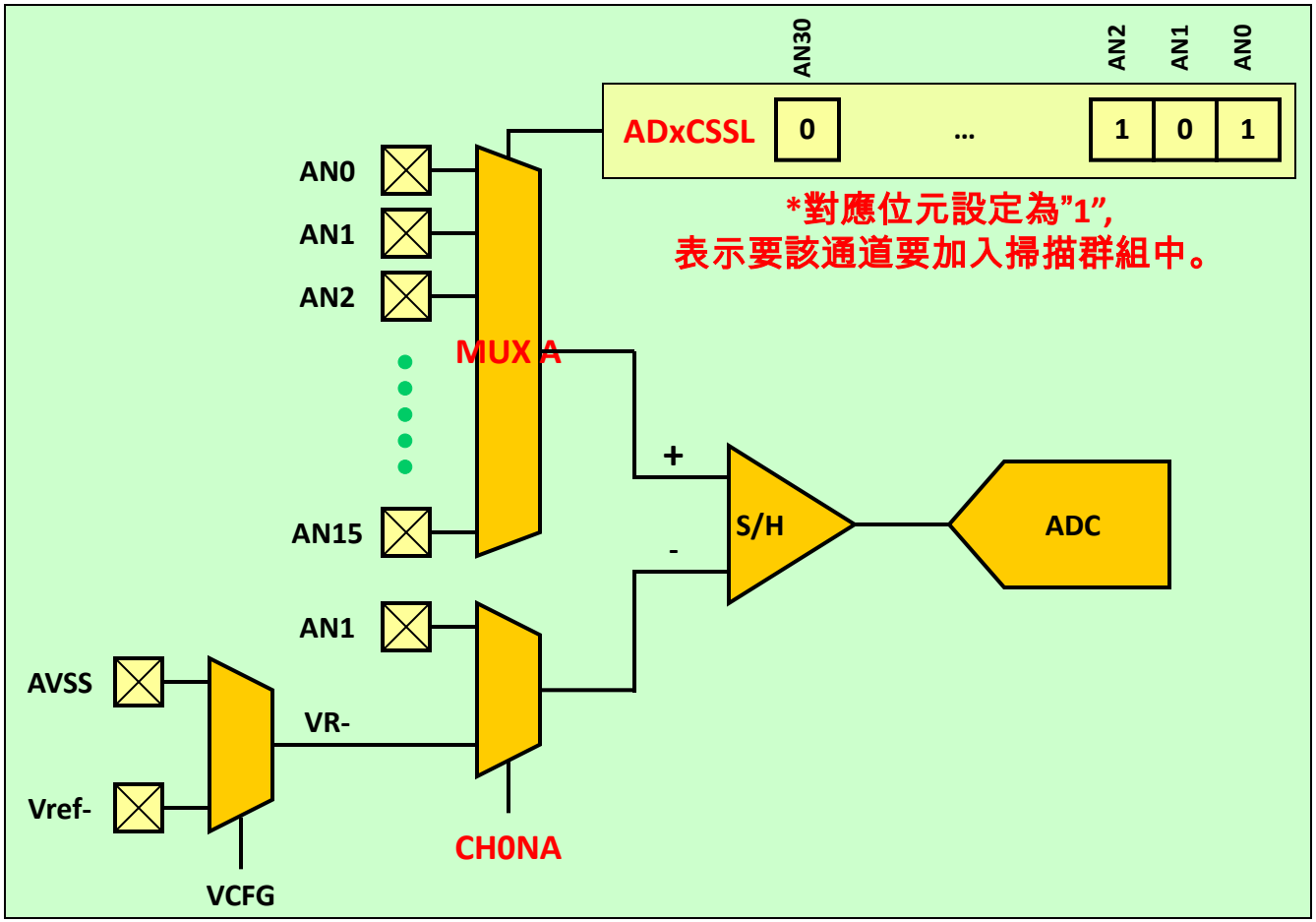


MICROCHIP
Regional Training Centers

Option!

Channel Scan

- PIC32 ADC支援自動通道掃描,開啟通道掃描模式ADC會自動切換通道,取得類比電壓,進行轉換。
- 搭配轉換次數(SMPI<3:0>)的設定可以規劃轉換資料存在Buffer中的方式。



MICROCHIP
Regional Training Centers

Option!

Channel Scan Initial

- 要啟用Channel Scan, 可以在OpenADC10中設定ADC_SCAN_ON, 並且指定要掃描的通道:

```
OpenADC10(ADC_MODULE_ON | ADC_IDLE_CONTINUE | ADC_FORMAT_INTG16 |
        ADC_CLK_MANUAL | ADC_AUTO_SAMPLING_ON | ADC_SAMP_OFF,
        ADC_VREF_AVDD_AVSS | ADC_OFFSET_CAL_DISABLE | ADC_SCAN_ON |
        ADC_SAMPLES_PER_INT_2 | ADC_ALT_BUF_OFF | ADC_ALT_INPUT_OFF,
        ADC_SAMPLE_TIME_31 | ADC_CONV_CLK_SYSTEM |
        ADC_CONV_CLK_32Tcy,
        ENABLE_AN19_ANA | ENABLE_AN18_ANA,
```

指定不想掃描的通道。

```
SKIP_SCAN_AN0 | SKIP_SCAN_AN1 | SKIP_SCAN_AN2 | SKIP_SCAN_AN3 |
SKIP_SCAN_AN4 | SKIP_SCAN_AN5 | SKIP_SCAN_AN6 | SKIP_SCAN_AN7 |
SKIP_SCAN_AN8 | SKIP_SCAN_AN9 | SKIP_SCAN_AN10 | SKIP_SCAN_AN11 |
SKIP_SCAN_AN12 | SKIP_SCAN_AN13 | SKIP_SCAN_AN14 | SKIP_SCAN_AN15 |
SKIP_SCAN_AN16 | SKIP_SCAN_AN17 |
SKIP_SCAN_AN20 | SKIP_SCAN_AN21 | SKIP_SCAN_AN22 | SKIP_SCAN_AN23 |
SKIP_SCAN_AN24 | SKIP_SCAN_AN25 | SKIP_SCAN_AN26 | SKIP_SCAN_AN27 |
SKIP_SCAN_AN28 | SKIP_SCAN_AN29 | SKIP_SCAN_AN30);
```

MICROCHIP
Regional Training Centers

Option!

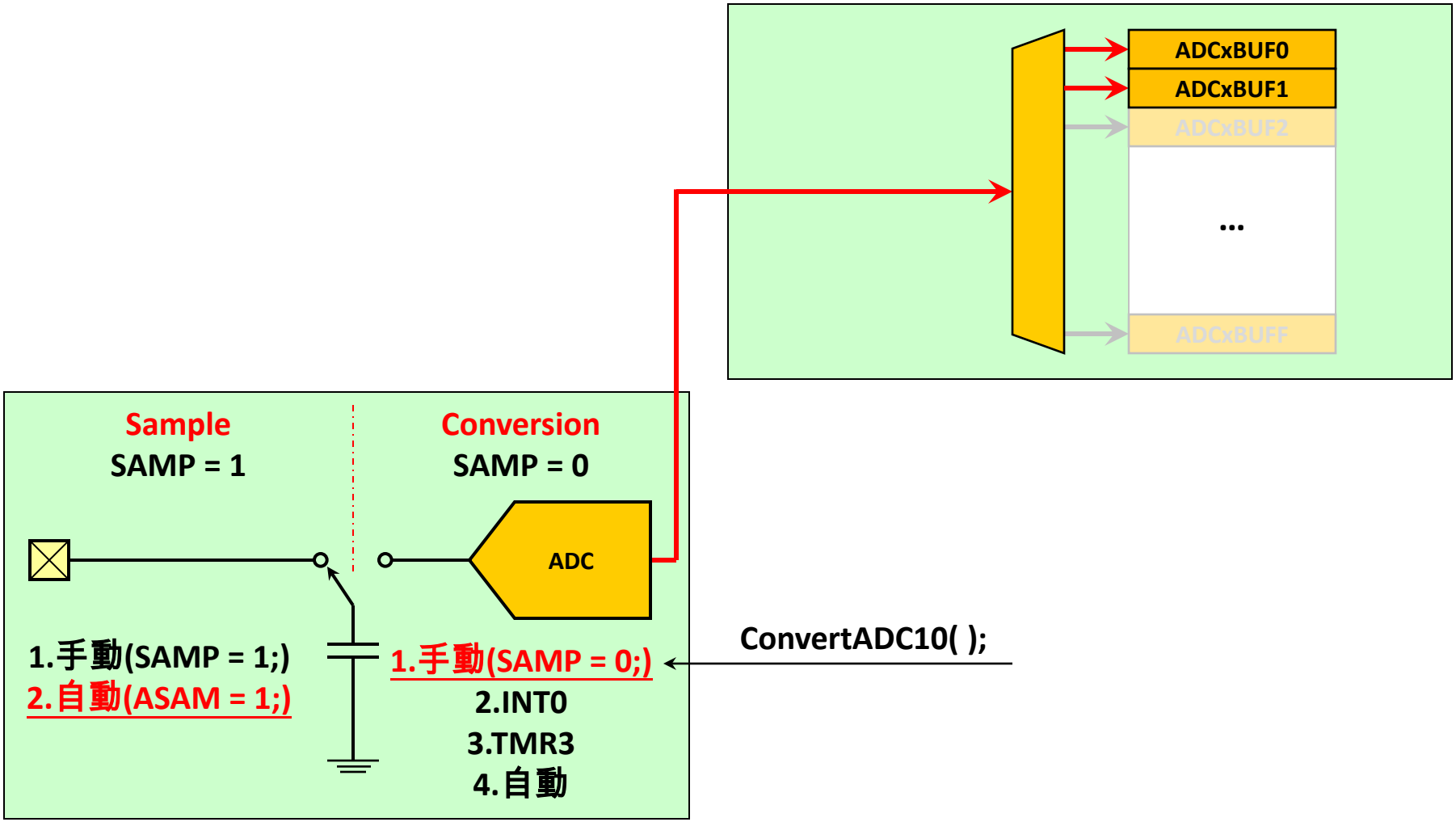
Lab7 ADC Multi CH Manually

- 在Lab6的程式基礎上, 嘗試開啟自動通道掃描功能。利用ADC取得VR1, VR2的類比電壓值, 並透過printf( )轉換的結果顯示在PC上。
  ADC的工作模式設定為, 自動取樣, 手動轉換, 16Bits無號整數格式, 開啟通道掃描(AN18, AN19)。
- 閱讀ADC Function的說明文件, 了解ADC Function的使用方法。了解如何使用OpenADCx( )開啟自動通道掃描。並了解如何在通道掃描模式下, 使用ReadADC10( )取得資料。
- 使用Bootloader將程式燒錄進APP028-1。觀察程式執行的情況。驗證看看是否可以PC上看到VR1, VR2的值(0~1023)。

Lab7 ADC Multi CH Manually

Option!

Block Diagram



MICROCHIP
Regional Training Centers

Option!

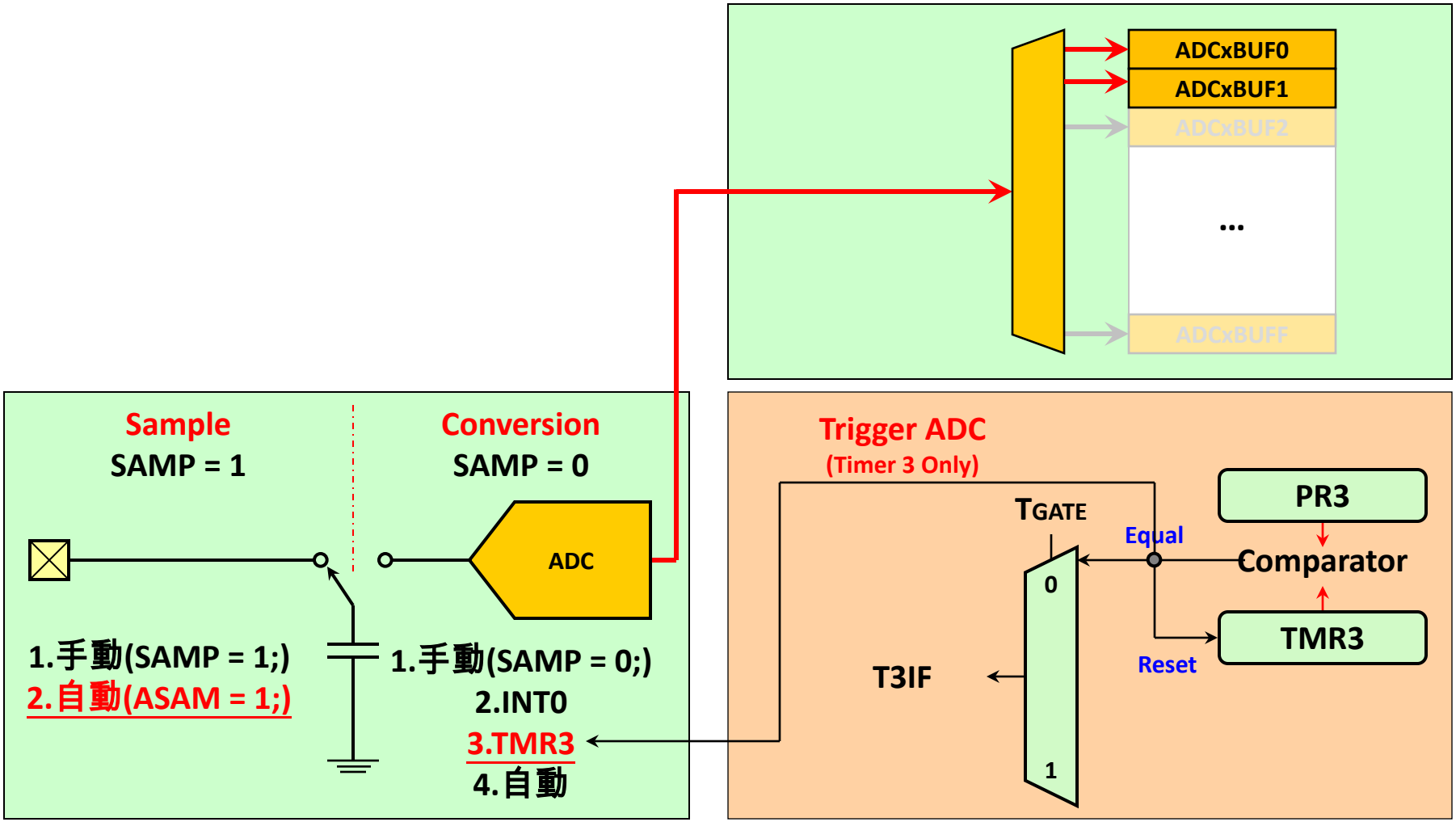
Lab8 ADC Single CH Timer3 Trigger

- 利用Lab7的程式基礎, 將ADC的轉換觸發來源由手動觸發改成TMR3觸發。 設定TMR3每50mS, 觸發ADC轉換。
- 建立ADC的中斷服務常式, 並開啟ADC的中斷, 設定優先權。 優先權設定為預設值”5” 。
- 將原先Polling AD1IF的模式, 改成由中斷服務常式完成。
- 閱讀ADC Function的說明文件, 了解ADC Function的始用方法。 了解如何使用OpenADCx( )改變”轉換”的觸發來源,跟 ConfigIntADCx( ); 的用法。
- 使用Bootloader將程式燒錄進APP028-1。 觀察程式執行的情況。 驗證看看是否可以PC上看到VR1, VR2的值(0~1023)。

Lab8 ADC Single CH Timer3 Trigger

Option!

Block Diagram



MICROCHIP
Regional Training Centers

Ping-Pong Buffer (Buffer Fill Mode)

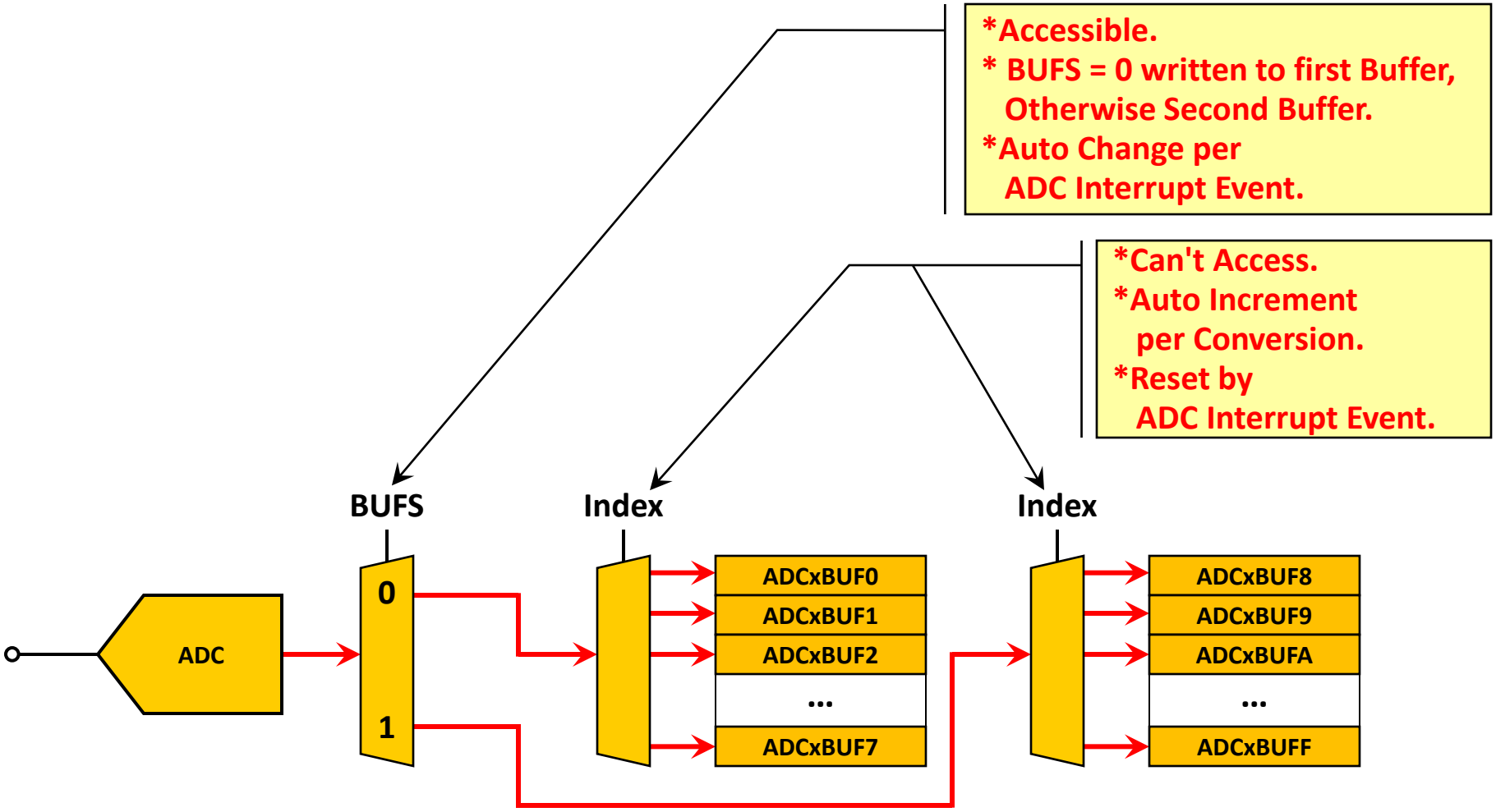
Option!

- ADC的Buffer在操作時, 有可能會發生ADC與CPU同時存取的狀況, 造成資料不一致或被破壞的情形。
- 通常的情形是, 讀取Buffer的速度太慢, CPU正準備讀取Buffer時, ADC也同時在將新的轉換結果存入Buffer中。
- ADC Buffer可以切割成兩塊各8的32Bits的Buffer,作為Ping-Pong Buffer使用 (ADCxBUF0~7,ADCxBUF8~F), 避免上述問題 。
- ADC在每次中斷時,會交替的存取兩塊區域,並透過BUFS (AD1CON2bits.BUFS)來標示目前存取的區域。CPU必須先檢查BUFS, 得知ADC的存取區塊, 然後自另一塊空間, 取得資料。
- BUFS=0時, 就從ADCxBUF8~F取得資料。BUFS=1時, 就改由ADCxBUF0~7取得資料。

MICROCHIP
Regional Training Centers

Ping-Pong Buffer (Buffer Fill Mode)

Option!



MICROCHIP
Regional Training Centers

Lab9 ADC Multi CH Timer3 Trigger with Ping-Pong Buffer

Option!

- 嘗試在Lab8的程式基礎上, 修改ADC的工作模式, 開啟Ping-Pong Buffer Mode。
- 由於開啟Ping-Pong Buffer Mode, 讀取Buffer時, 為了避免跟ADC同時存取一個區塊, 必須透過BUFS Bits(AD1CON2bits.BUFS)來確認ADC目前存取的區塊, 不合法的存取會導致系統出現例外事件。
- 使用Bootloader將程式燒錄進APP028-1。觀察程式執行的情況。驗證看看是否可以PC上看到VR1, VR2的值(0~1023)。

Lab9 ADC Multi CH Timer3 Trigger

Option!

P.P. Buffer Block Diagram