MICROCHIP

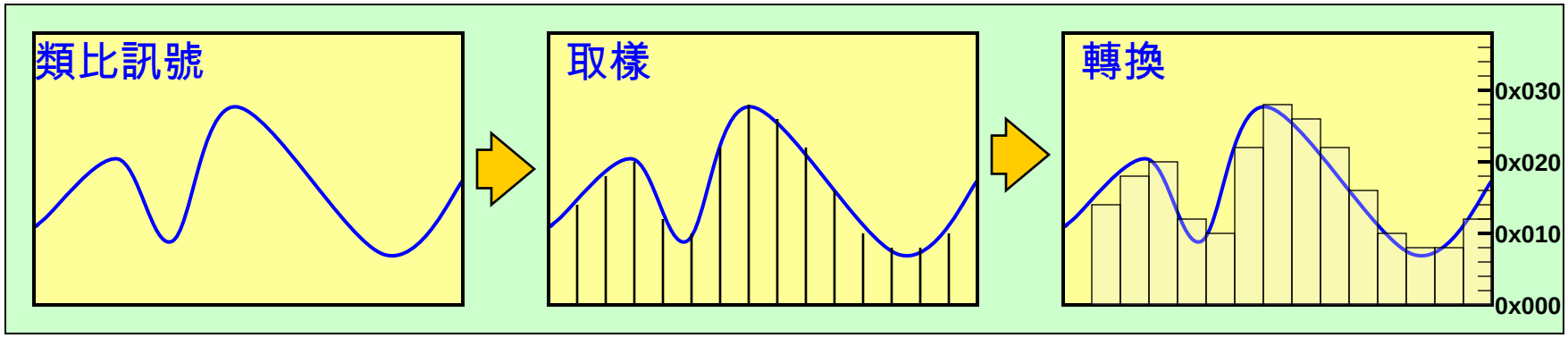
Regional Training Centers

Section 10
10 Bits ADC

MICROCHIP
Regional Training Centers

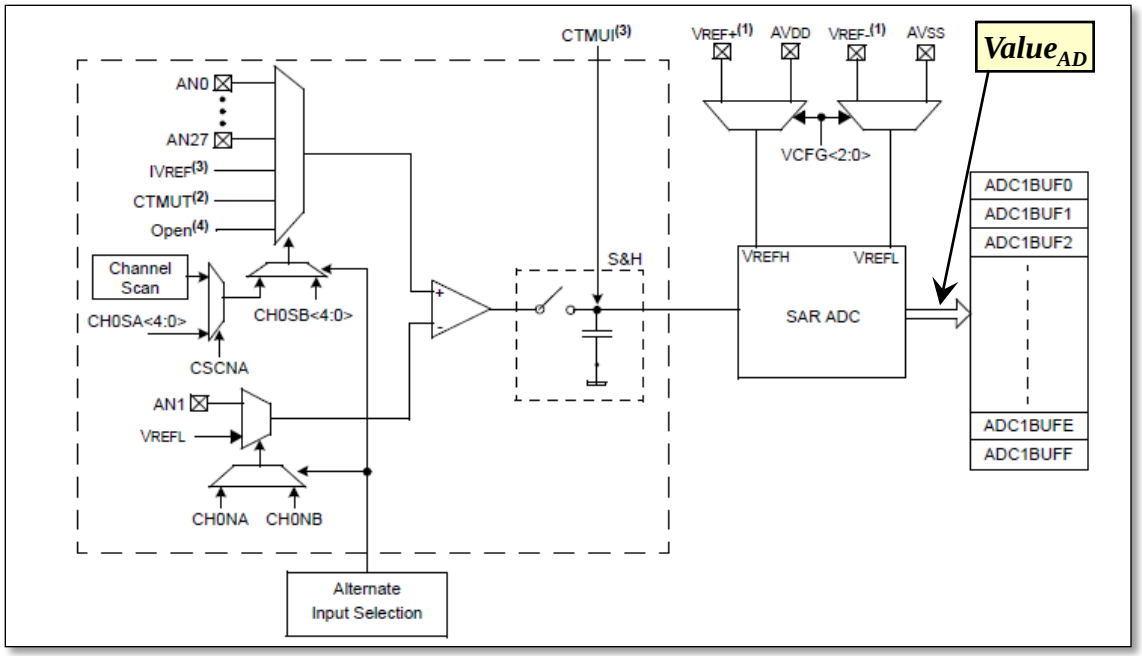
What's ADC ?

- 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

- PIC32MX450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

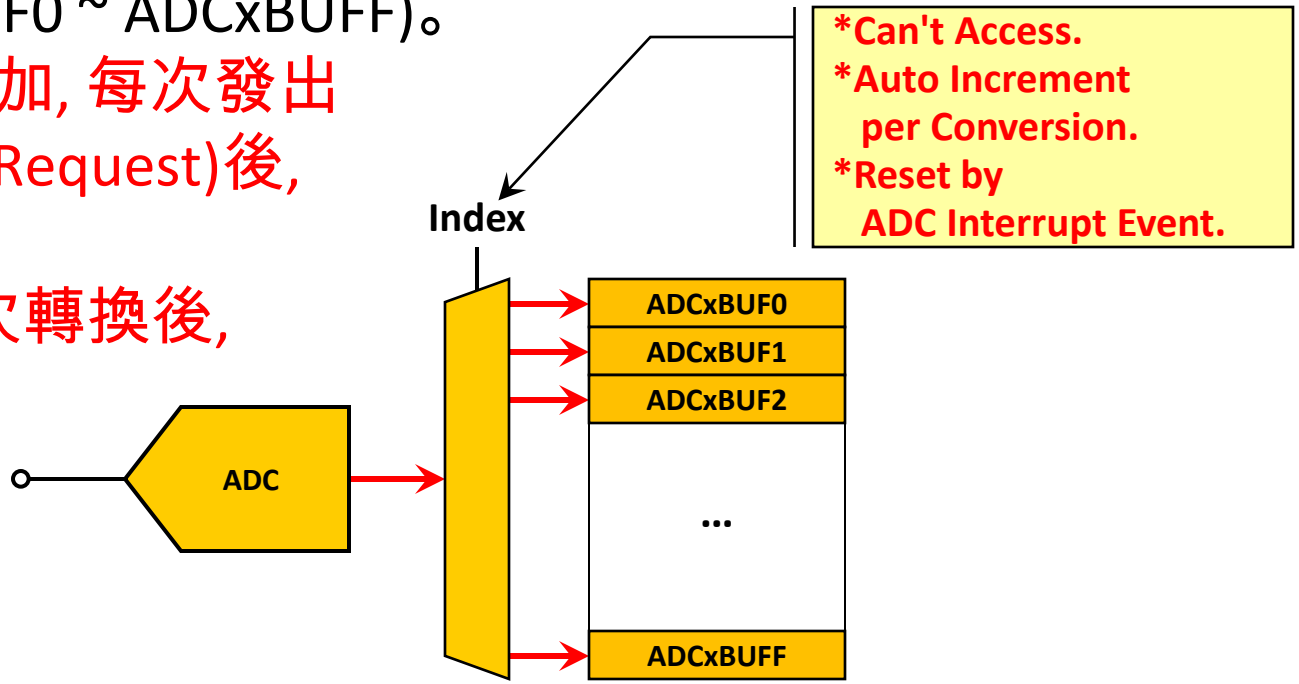


$$Value_{AD} = \frac{V_{AD} - V_{R-}}{V_{R+} - V_{R-}} \times 2^n$$

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

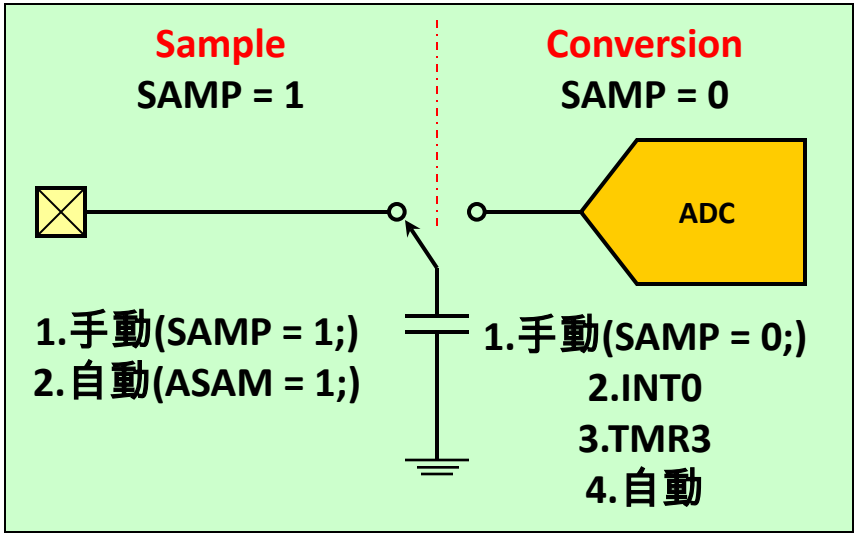
PIC32's ADC Architecture

- 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)。



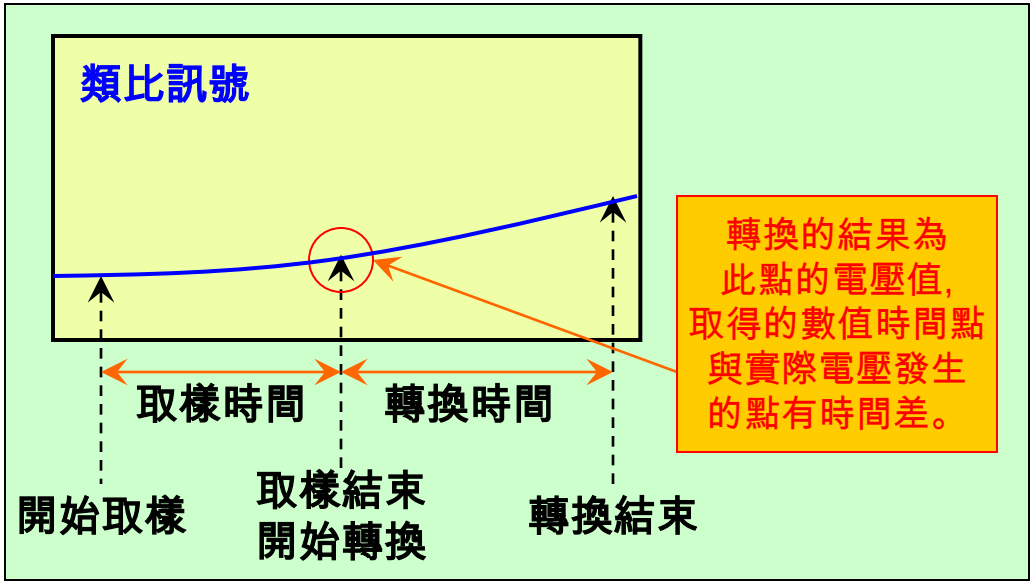
ADC Sample, Conv. Trigger Source

- 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

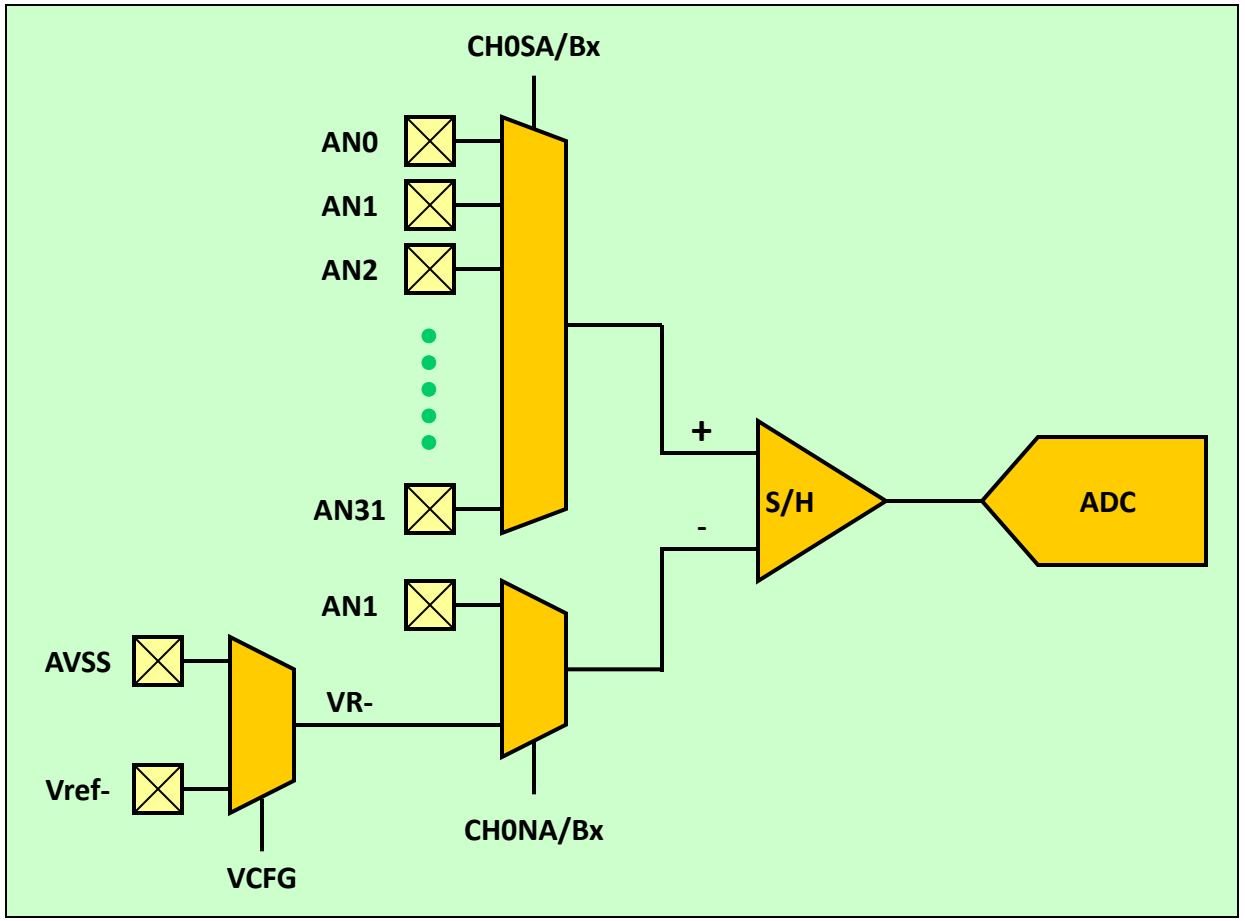
- SAR ADC作動分為"取樣"及"轉換"兩步驟,
  取樣時利用外部訊號對ADC內部的電容器充電,取得外部電壓值。
  轉換時依據取得的電壓值換算出結果。



- 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) 。

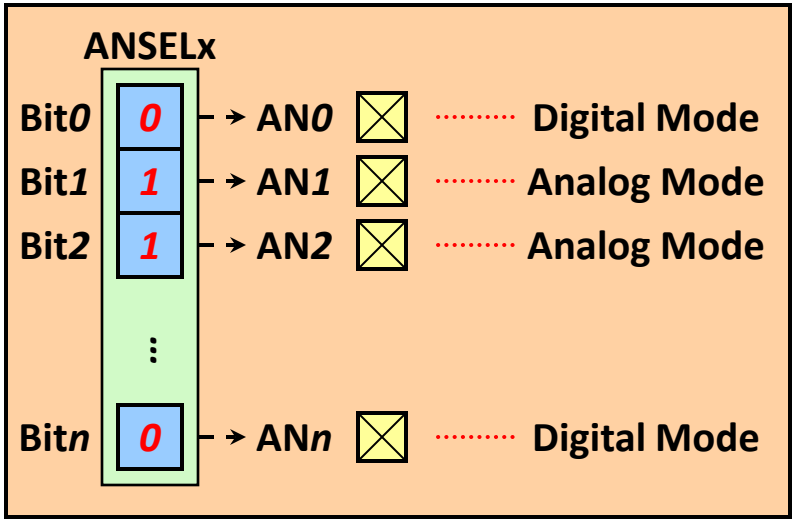
ADC Channel Select

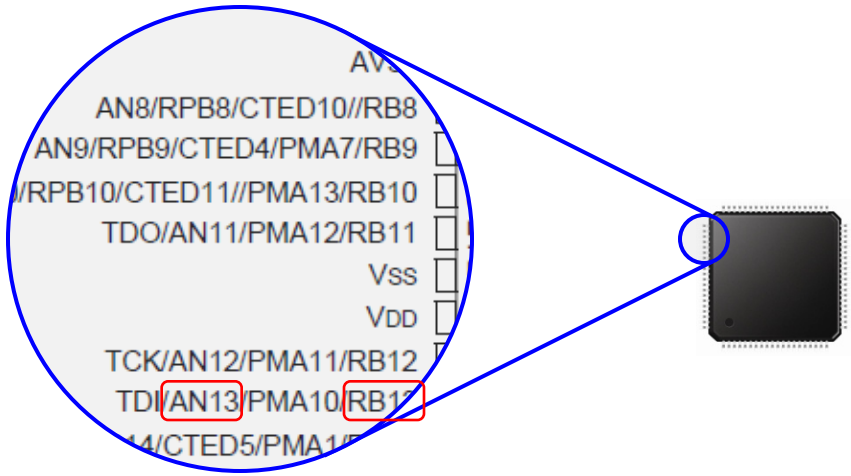
- ADC的正端輸入可以選擇AN0~AN31。負端輸入則可選擇連接VR-(V_{REF-}或AV_{SS})。
- 負端輸入也可選擇連接到AN1,讓兩訊號相減後,再送入ADC(單端差動模式)。
- 單端差動,類似”差動”的概念,但兩者有相同的地(GND)。



Analog or Digital Mode

- 複習一下, 前面提過, 有些 IO接腳(Rx*n*)的功能跟類比輸入(AN*n*)共用的。但MCU在Reset/Power On後的預設值為Analog Mode, 無法做為數位輸出入使用。所以如果碰到這類接腳, 在使用Digital Mode IO模式時必須先取消Analog Mode。反過來要當作ADC的輸入時, 則要設定為Analog Mode才能正常動作。
- PIC32MX450F2456H透過ANSELx來設定
  Ex: ANSELBbits.ANSB13 = 1; //設定為Analog Mode(AN13 Enable)

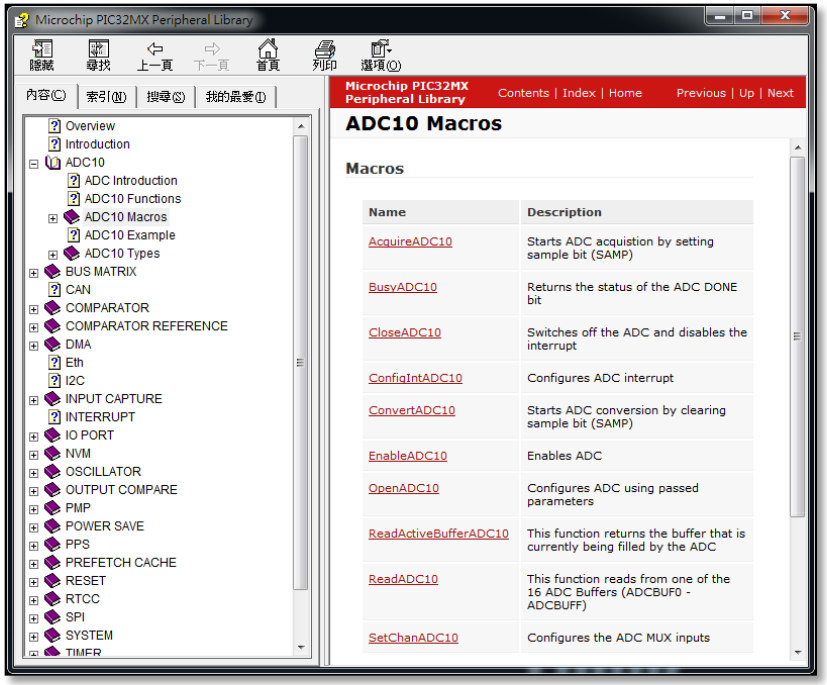




XC32 ADC Function & Macro

- 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_AN11_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_AN1 |
             ADC_CH0_NEG_SAMPLEA_NVREF |
             ADC_CH0_POS_SAMPLEB_AN0 |
             ADC_CH0_NEG_SAMPLEB_NVREF );
```

MICROCHIP
Regional Training Centers

LCD Module Functions

- 程式中已預先提供給各位LCD Module的Function,可以直接使用(APP026-3_LCM.c)。
- 提供以下幾個Function:

  void LCM_Init( ) //初始化LCD Module。
  int LCM_IsBusy( ) //讀取LCD Module的 BUSY Flag並傳回狀態。
  1:Busy,0:Not Busy。
  void LCM_PutASCII(unsigned char) // 輸出字元。
  Void LCM_SetCursor(char X , char Y) // 設定遊標位置。
  Void LCM_PutROMString(const unsigned char *String) // 輸出const 字串。
  Void LCM_PutRAMString(unsigned char *String) //輸出字串。
  Void LCM_PutHex(unsigned char Hex) //將變數轉為Hex輸出。
  void LCM_PutNumber(unsigned int Number , unsigned char Digit);
  // 輸出整數。

- 提供原始碼,如果需要其它的功能,可以自行修改。

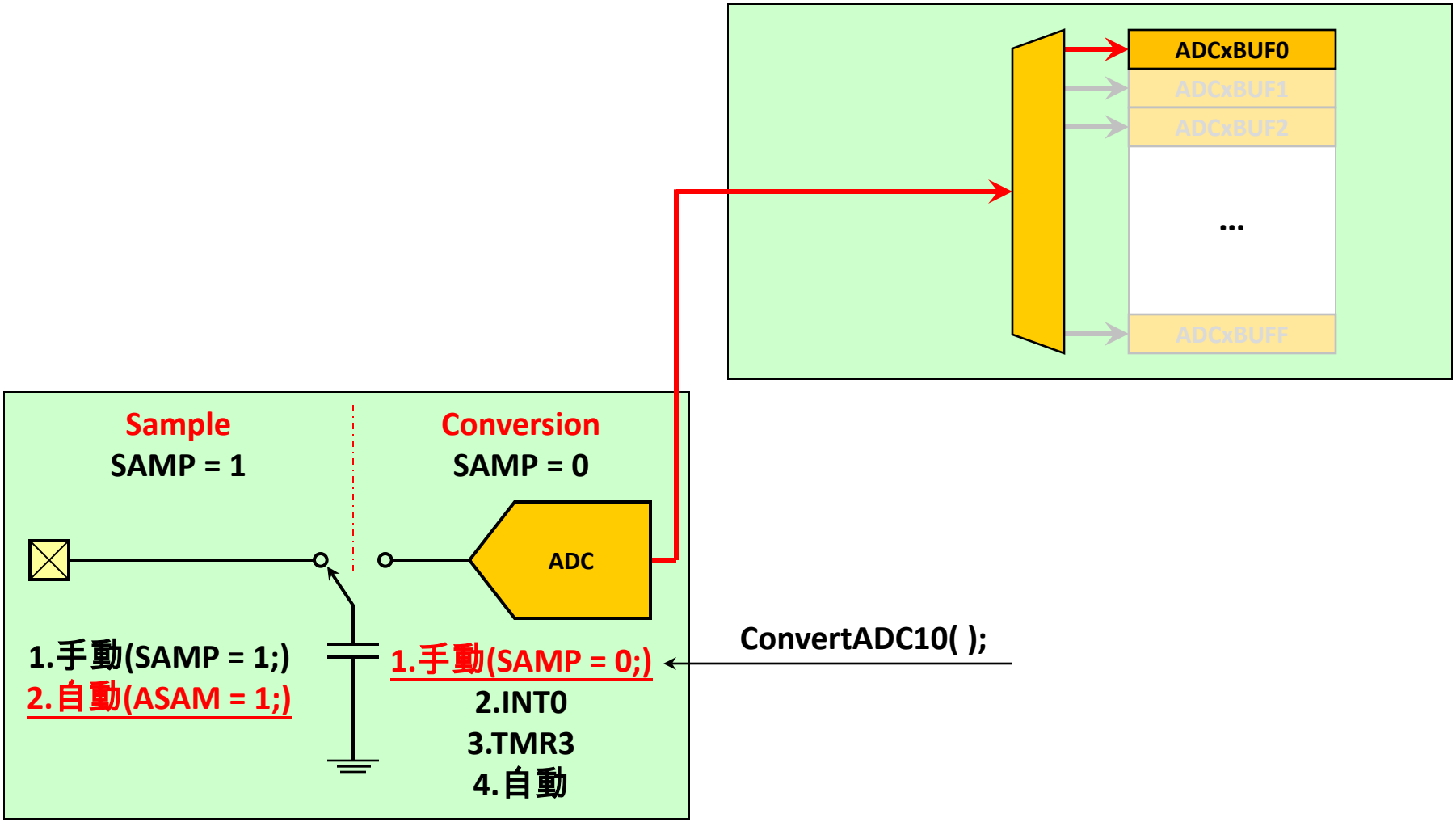
MICROCHIP
Regional Training Centers

Lab8 ADC Single CH Manually

- 在Lab7的程式基礎上, 嘗試加入ADC的程式片斷。利用ADC10來取得VR的類比電壓值, 並顯示在LCD Module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AN11, 16Bits無號整數格式, 不開啟通道掃描。
- 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的始用方法。至少必須了解
  OpenADC10( ),SetChanADC10( ), ConvertADC10( );
  BusyADC10( ),ReadADC10( ),用法。
- 透過Bootloader將程式實際燒錄到MCU中, 用肉眼觀察看看程式執行的情況。正確設定完成後,可以LCD Module上看到VR1的值(0~1023)。

Lab8 ADC Single CH Manually

Block Diagram



Lab8 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。

Lab8 ADC Single CH Manually

Step2

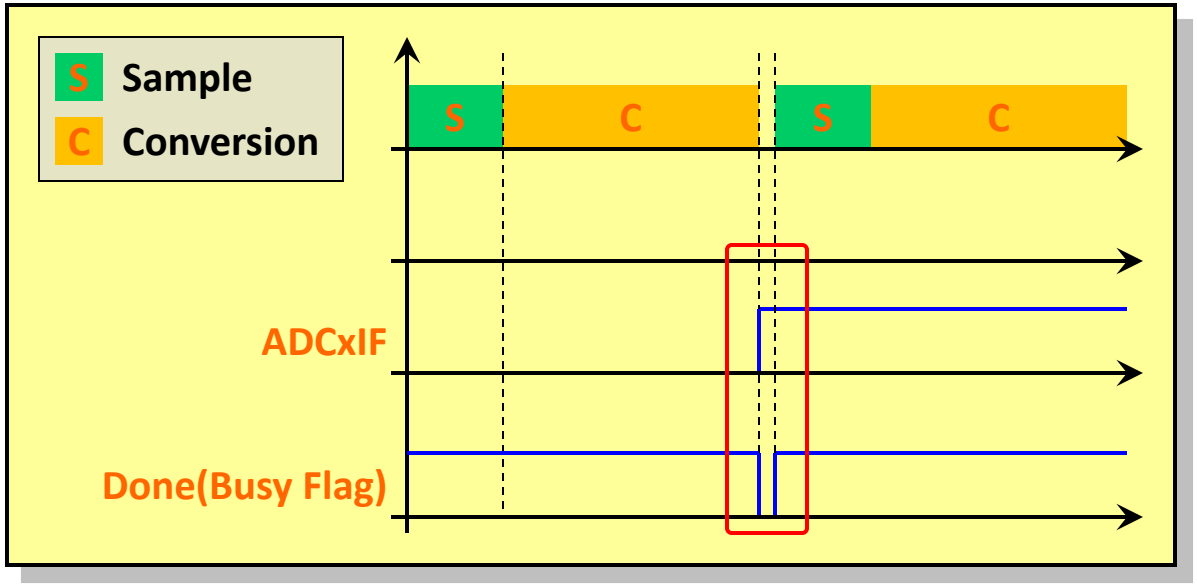
- 如何取得ADC轉換的結果?

  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。



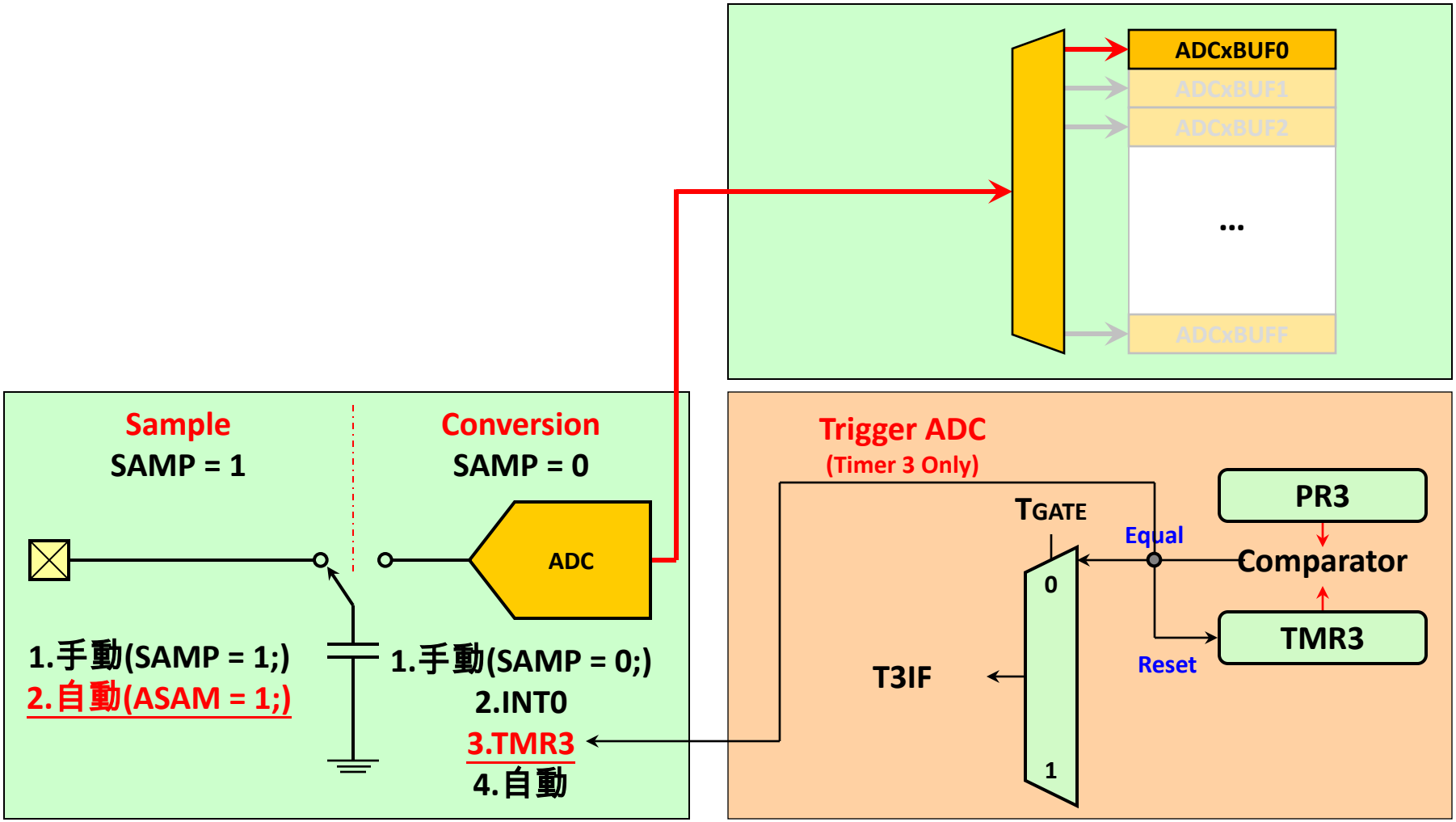
MICROCHIP
Regional Training Centers

Lab9 ADC Single CH Timer3 Trigger

- 利用Lab8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20mS, 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。
- 透過Bootloader將程式實際燒錄到MCU中, 用肉眼觀察看看程式執行的情況。 正確設定完成後,可以LCD Module上看到VR1的值(0~1023)。

Lab9 ADC Single CH Timer3 Trigger

Block Diagram



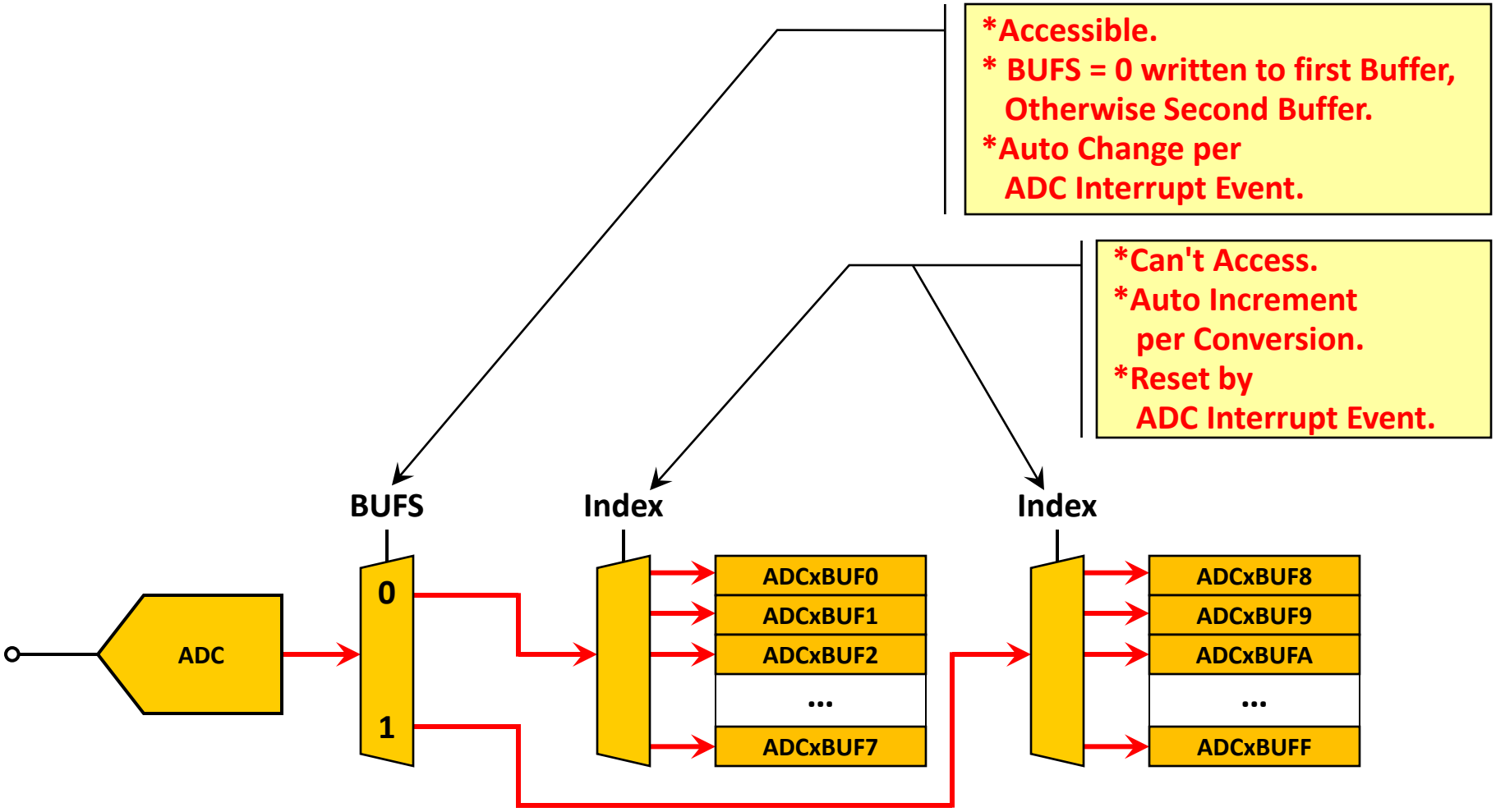
MICROCHIP
Regional Training Centers

Ping-Pong Buffer Mode (Buffer Fill Mode)

- 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 Mode (Buffer Fill Mode)



MICROCHIP
Regional Training Centers

Lab10 ADC Single CH Timer3 Trigger with Ping-Pong Buffer

- 嘗試在Lab9的程式基礎上, 修改ADC的工作模式。
- 設定為自動取樣與TMR3轉換模式, 並手動指定類比通道為AN11。開啟Ping-Pong Buffer Mode, 關閉通道掃描等功能。
- 設定Timer3每20mS, 觸發ADC轉換。
- 開啟ADC中斷並建立AD中斷服務常式。並於AD中斷服務常式中取得AD轉換數值。
- 由於開啟Ping-Pong Buffer Mode, 讀取Buffer時, 為了避免跟AD同時存取一個區塊,必須透過BUFS Bits來確認ADC目前存取的區塊。
- 透過Bootloader將程式實際燒錄到MCU中, 用肉眼觀察看看程式執行的情況。正確設定完成後,可以LCD Module上看到VR1的值(0~1023)。