MICROCHIP

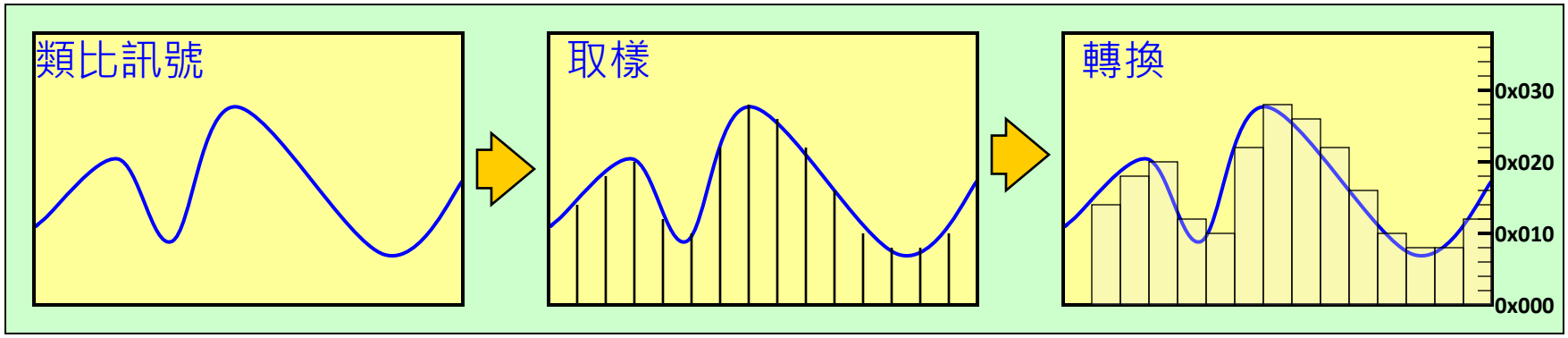
Regional Training Centers

Section 10
10 Bits High Speed 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。

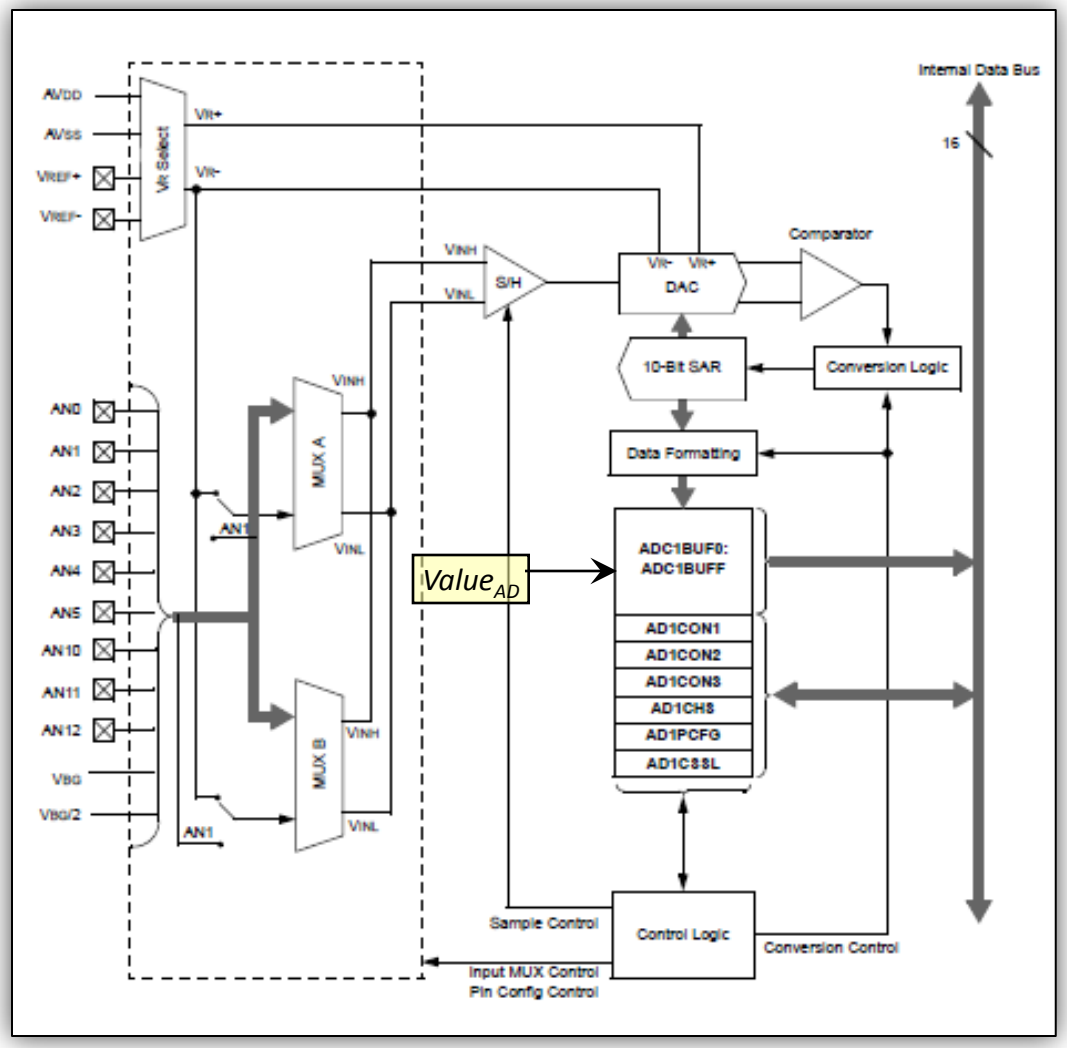


PIC24F ADC Architecture

- PIC24F16KA102具有一組採用SAR(連續近似法)的ADC。搭配13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

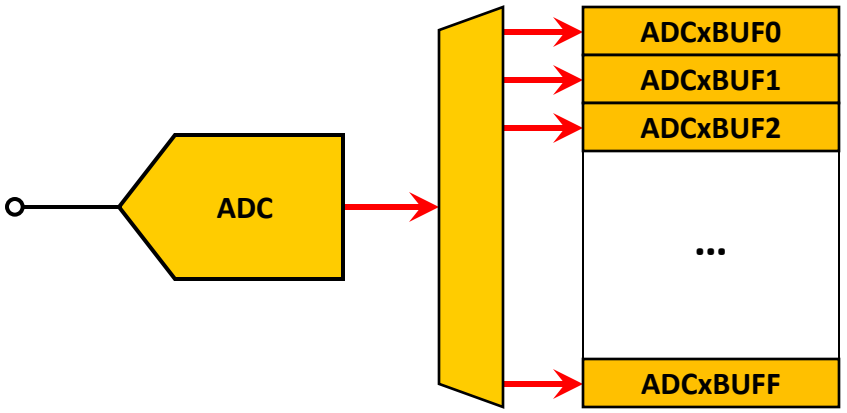
$$Value_{AD} = \frac{V_{AD} - V_{R-}}{V_{R+} - V_{R-}} \times 2^n$$

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



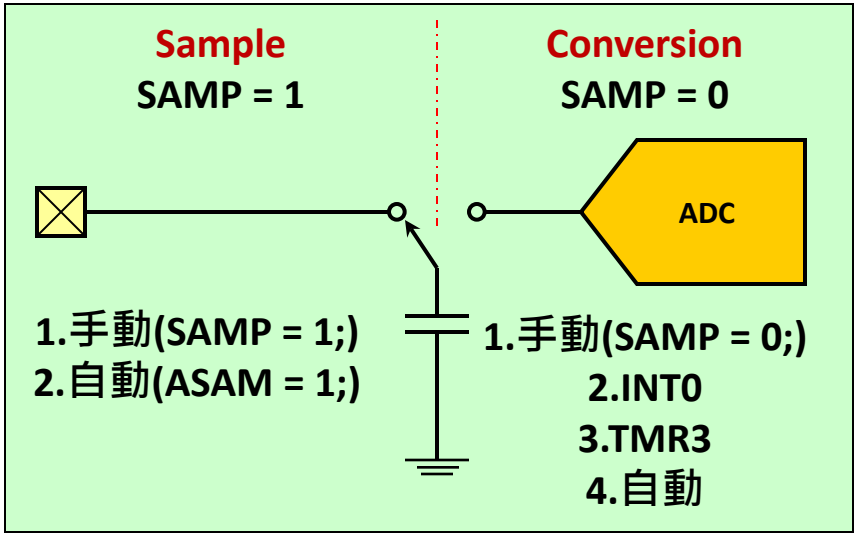
ADC Data Buffer

- ADC有16個Buffer用以儲存轉換結果(ADCxBUF0 ~ ADCxBUFF), 資料格式共有四種可選擇(有/無號, 小數/整數) 。
- **轉換結果固定從ADCxBUF0開始依序存放。SMPI(Sample Per Interrupt)可以決定幾次轉換後, 發出中斷需求(Interrupt Request) 。每次中斷後, 轉換結果會再次從ADCxBUF0開始存放。**



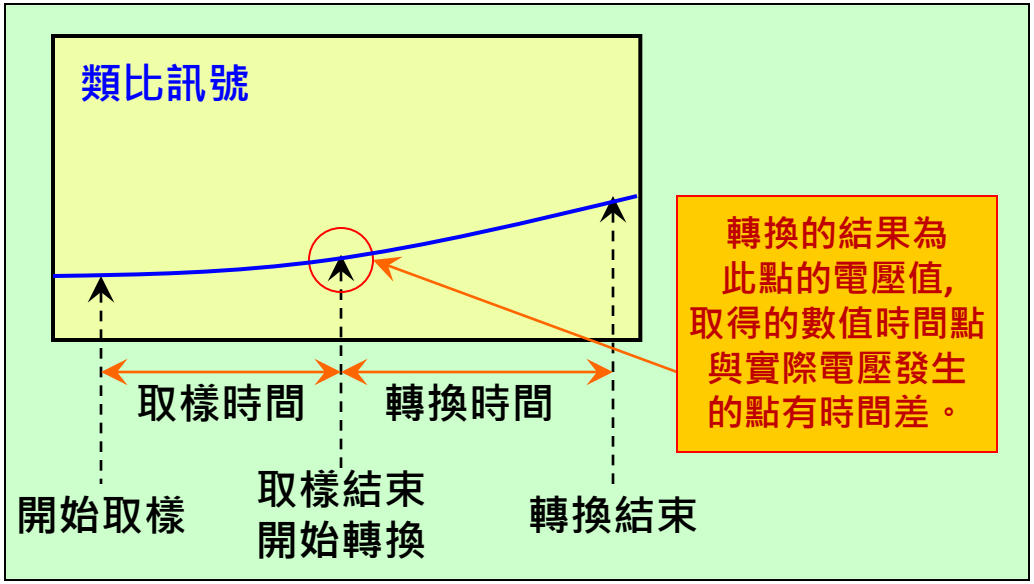
ADC Sample, Conv. Trigger Source

- 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有多個”轉換”觸發源: 手動, 自動, 外部中斷, Timer3 Match, etc..。



Sample and Conversion Sequence

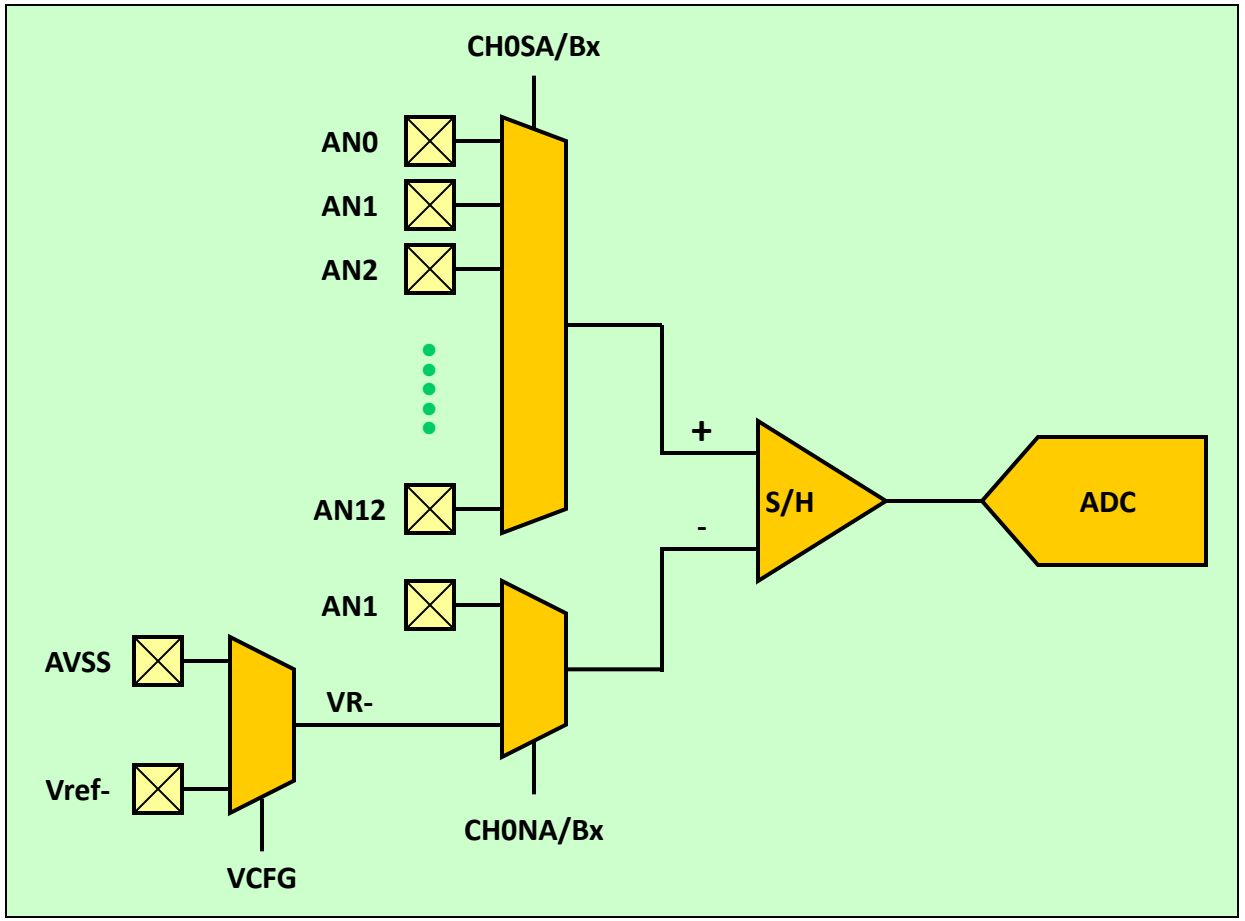
- SAR ADC作動分為”取樣”及”轉換”兩步驟。取樣是利用外部訊號對ADC內部的電容器充電,取得外部電壓值。轉換時則依據取得的電壓值換算出編碼值。



- 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。
- **PIC24F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$12T_{AD}$($1\ T_{AD}$ =75nS) 。**

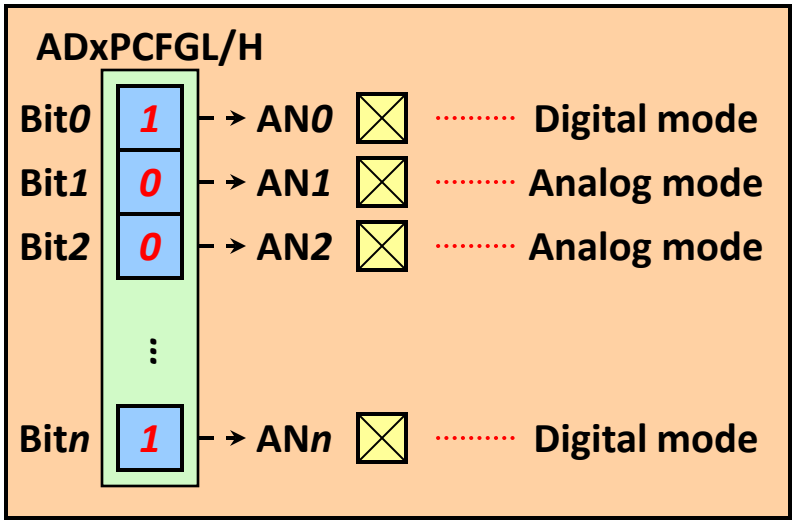
ADC Channel Select

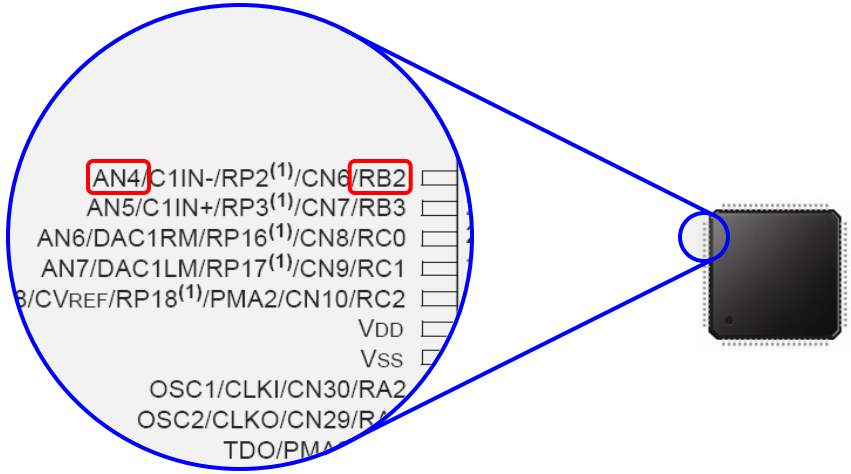
- ADC的正端輸入可以選擇AN0~AN12。負端輸入則可選擇連接VR-(V_{REF-}或AV_{SS})。
- 負端輸入也可選擇連接到AN1, 讓兩訊號相減後, 再送入ADC (單端差動模式)。
- 單端差動, 類似“差動”的概念, 但兩者有相同的地(GND)。



Analog Mode for GPIO

- 回憶下IO章節, 類比輸入(AN*n*)跟GPIO接腳(R*xn*)是共用的。
  所以我們必須設定接腳為Analog Mode, AD才能正常動作。
- PIC24F16KA102透過AD1PCFG來設定。
  1為Digital mode, 0為Analog mode。





MICROCHIP
Regional Training Centers

MCC's ADC Function

- **MCC中的幾個ADCx常用函數:**

void ADC1_Initialize(void);

// 啟用ADC1, 設定ADC1工作模式。

void ADC1_ChannelSelect(ADC1_CHANNEL channel);

// 手動切換ADC1的取樣通道。

void ADC1_Start(void);

// 手動觸發取樣(SAMP <- 1)。

void ADC1_Stop(void);

// 手動觸發轉換(SAMP <- 0) 。

uint16_t ADC1_ConversionResultGet(void);

// 讀取ADC1轉換的結果(ADC1BUF0)

uint16_t ADC1_ConversionResultBufferGet(uint16_t *buffer);

// 讀取ADC1轉換的結果(ADC1BUF0 ~ ADC1BUFn -> buffer[ ])

Lab9 ADC Single CH Manually

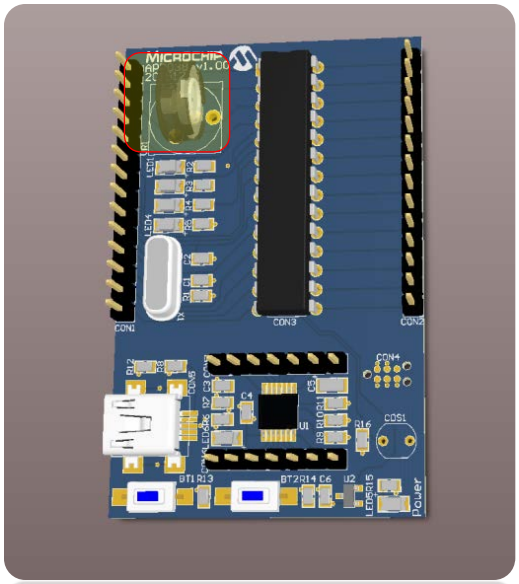
目 標

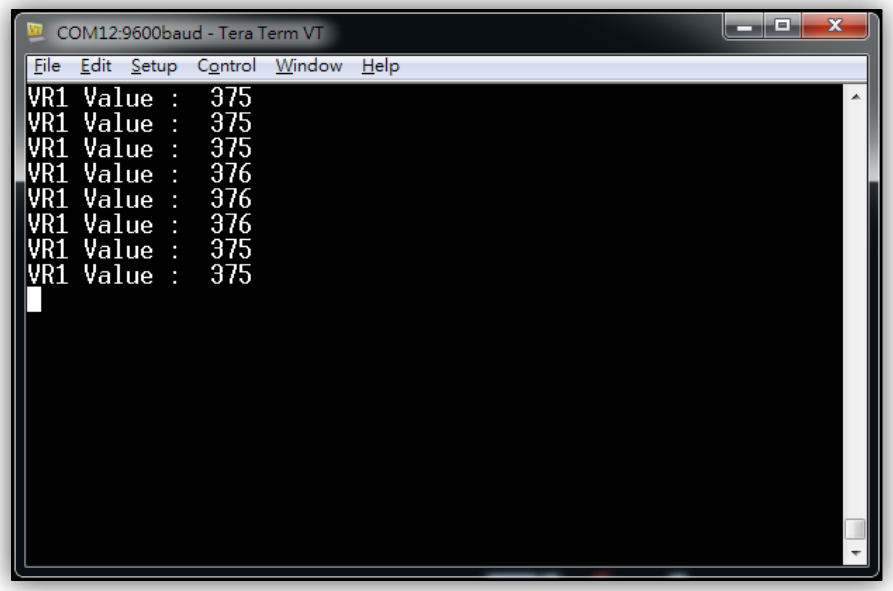
- 嘗試透過MCC的設定, 在Lab9架構上加入**ADC1**的設定。利用AN0取得VR1的電壓值, 並顯示在LCD Module上。
- **ADC1**的工作模式設定為, 自動觸發取樣, 手動觸發轉換, 類比通道設定為AN0, 無號整數格式。AN0修改別名為Channel_VR1。
- 利用原先的Timer1, 手動觸發ADC動作(ADC1_Stop( )), 再利用Polling ADC Flag(**_AD1IF**)的方式確認ADC是否轉換完成。最後將數值取出, 透過UART1傳送至PC顯示。顯示在終端機畫面上。

- 該如何開始？

Lab9
ADC Single CH Manually Result

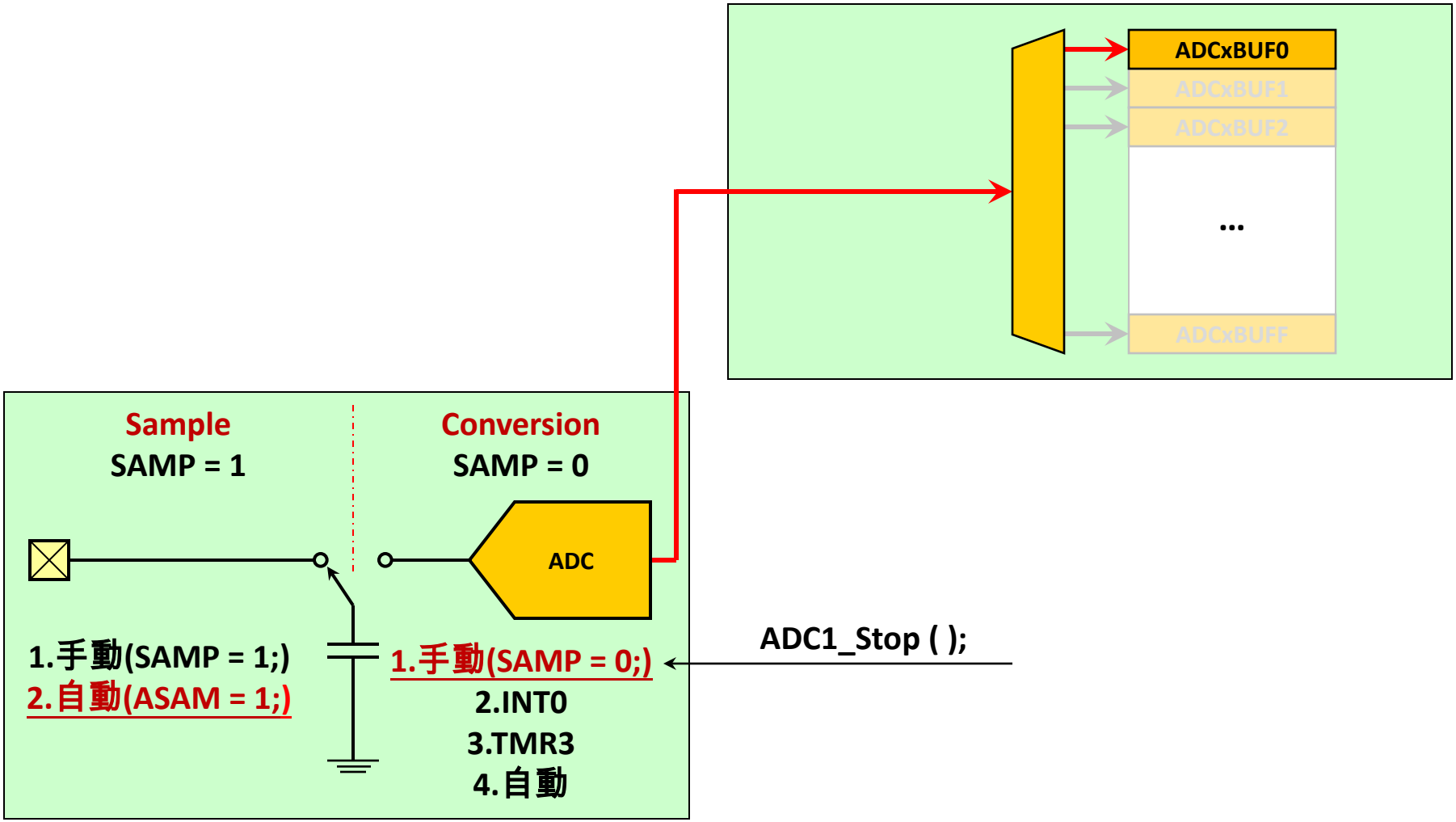
- 開啟既有專案(C:\Exercises\Exams\Lab9 ADC Single CH Manually.X)
- 接著開啟MCC, 新增**ADC1**資源, 根據題目要求完成設定。





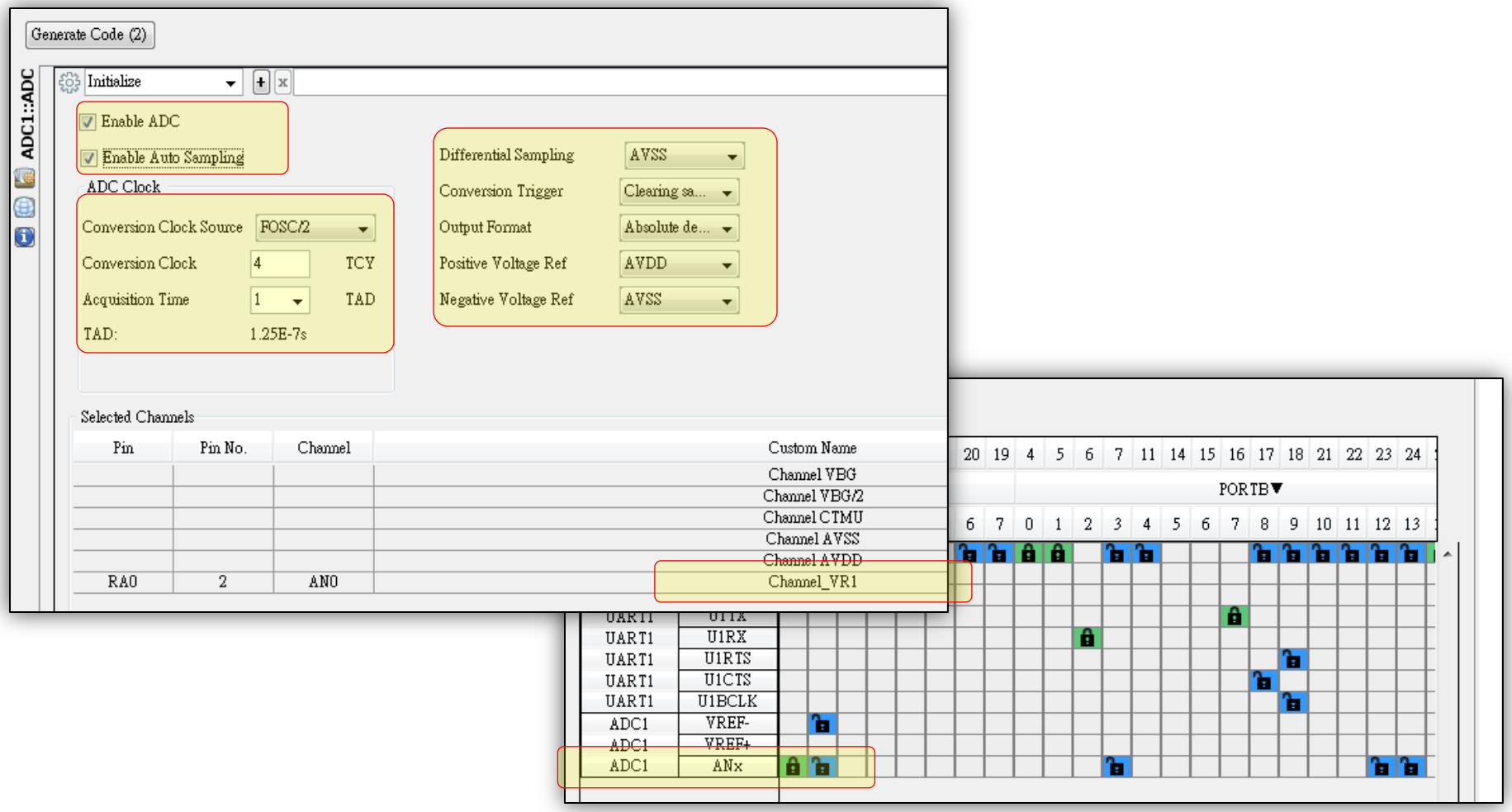
Lab9 ADC Single CH Manually

Block Diagram



Lab9 ADC Single CH Manually

MCC's Setting



MICROCHIP
Regional Training Centers

Lab9 ADC Single CH Manually Code Example

main.c

```
...
ADC1_ChannelSelect(ADC1_CHANNEL_VR1);
...

while (1)
{
   if (T2Flag)
   {
       T2Flag = 0;
       ...
       ADC1_Stop();
   }

   if (_AD1IF)
   {
      _AD1IF = 0;
      sprintf((char *) UARTString, "VR1 Value : %4d\r\n",
            ADC1_ConversionResultGet( ) );
      UART1_WriteBuffer(UARTString, strlen(UARTString));
   }
   UART1_TasksTransmit( ) ;
}
```

MICROCHIP
Regional Training Centers

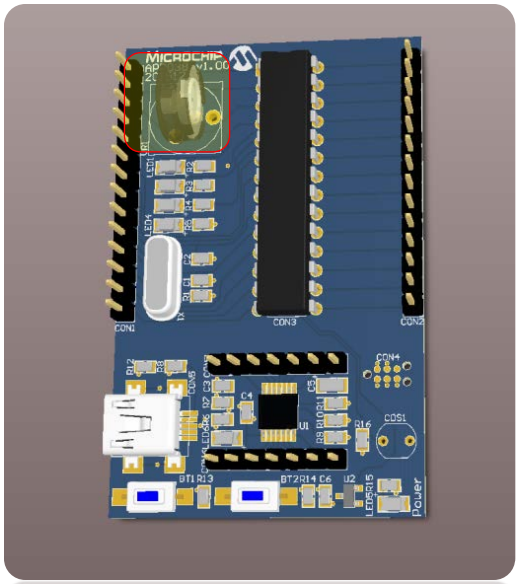
Lab10 ADC Single CH Timer3 Trigger

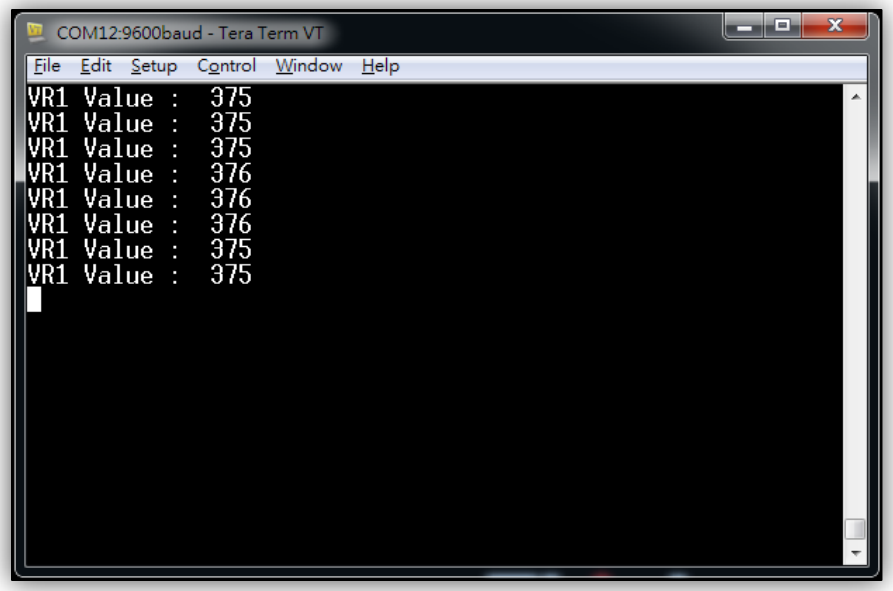
目 標

- 嘗試透過MCC的設定, 在Lab10架構上修改**ADC1**的設定。
- 修改**ADC1**設定手動觸發轉換 -> Timer3觸發轉換。
- 設定**Timer3**的週期為50mS。
- 刪除原先由Timer1手動觸發ADC的程式片段, ADC1轉換會改由Timer3自動觸發, 50mS一次。
- 再利用Polling ADC Flag(**_AD1IF**)的方式確認ADC是否轉換完成。最後將數值取出, 透過UART1傳送至PC顯示。顯示在終端機畫面上。

Lab10
ADC Single CH Timer3 Trigger Result

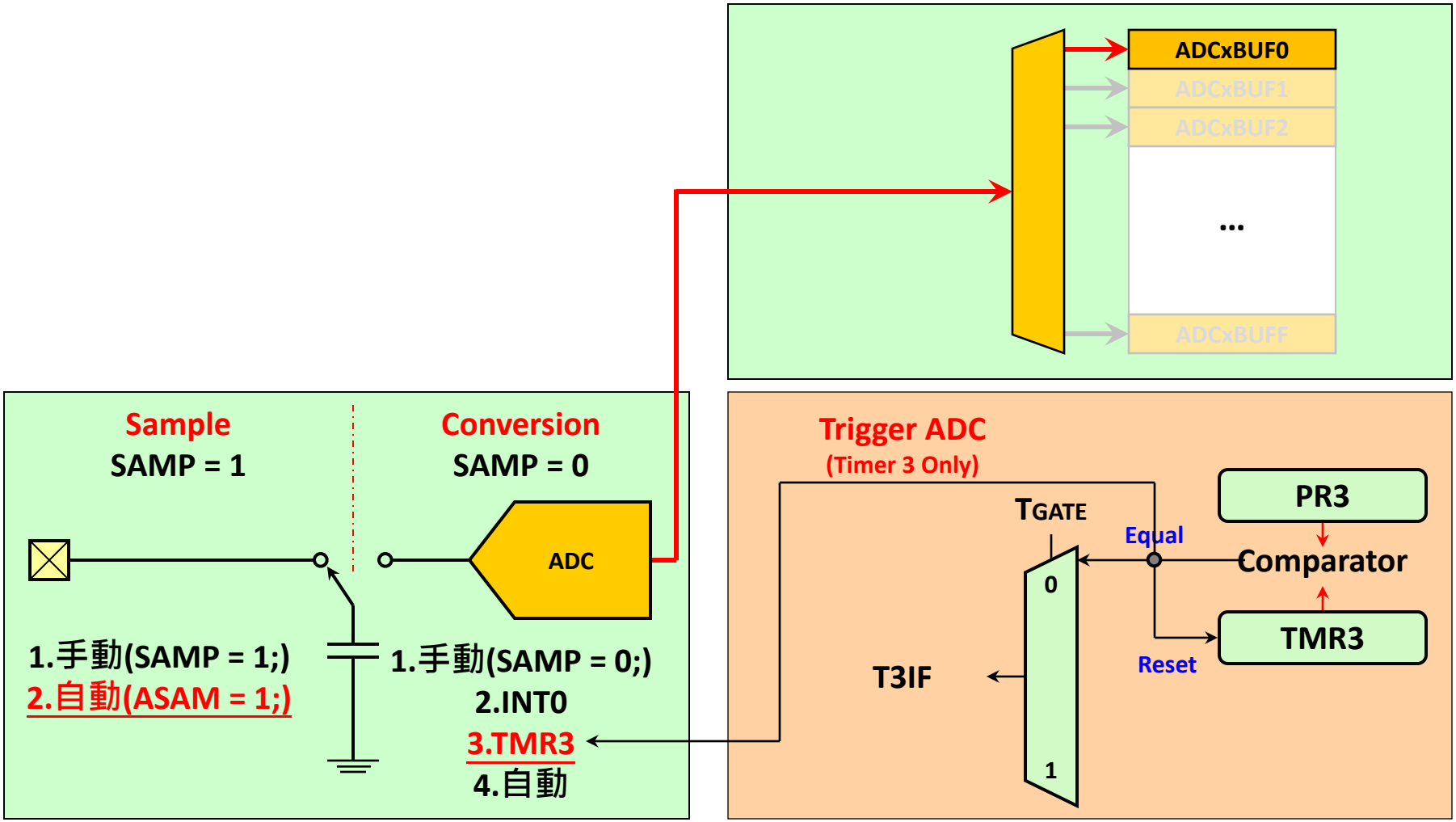
- 開啟既有專案(C:\Exercises\Exams\Lab10 ADC Single CH Timer3 Trigger.X)
- 接著開啟MCC, 修改**ADC1**資源, 根據題目要求完成設定。
- 顯示結果是相同的, 但觸發轉換源已經變更了(TMR3)。





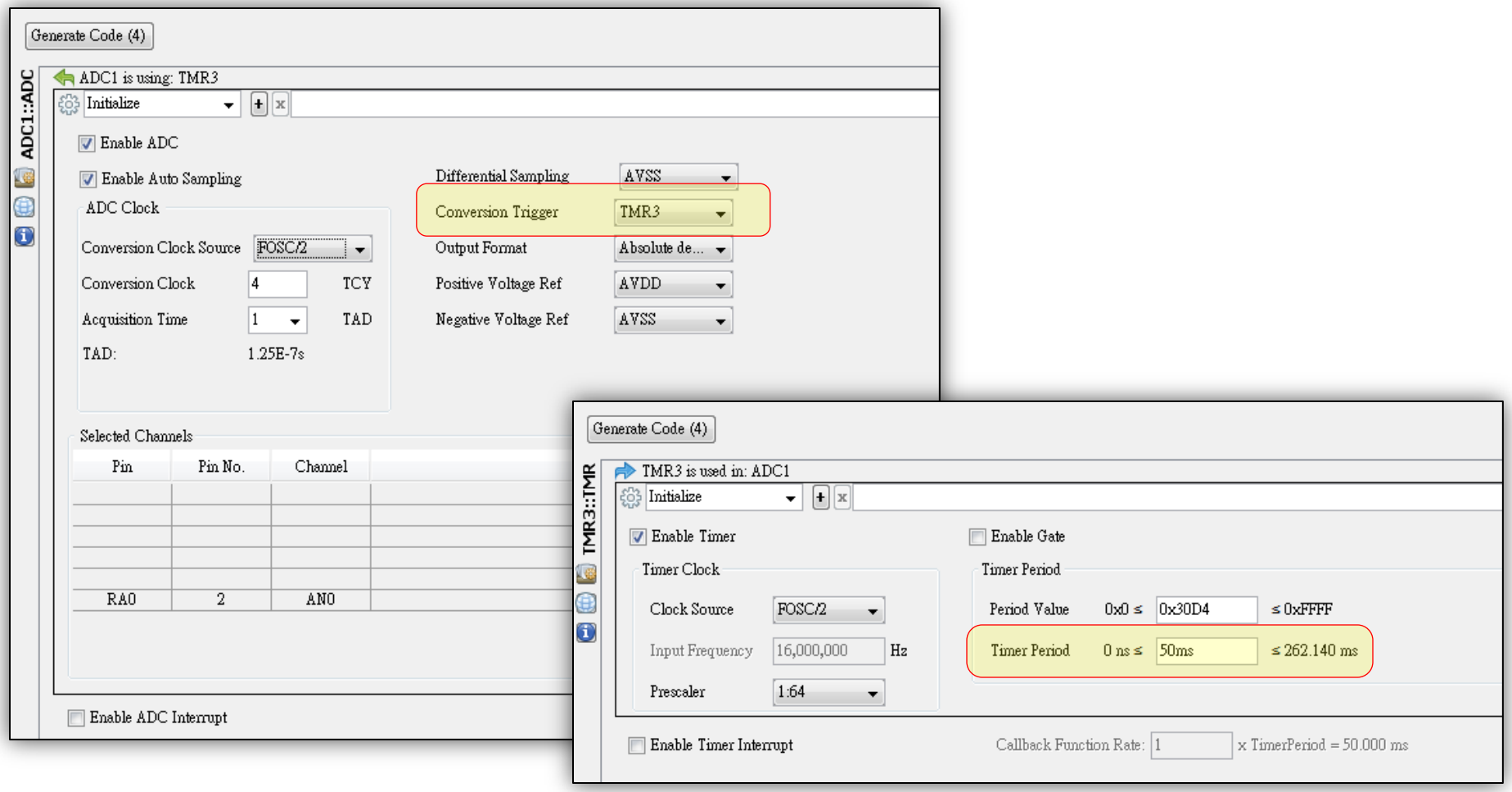
Lab10 ADC Single CH Timer3 Trigger

Block Diagram



Lab10 ADC Single CH Timer3 Trigger

MCC's Setting



MICROCHIP
Regional Training Centers

Lab10 ADC Single CH Timer3 Trigger

Code Example

main.c

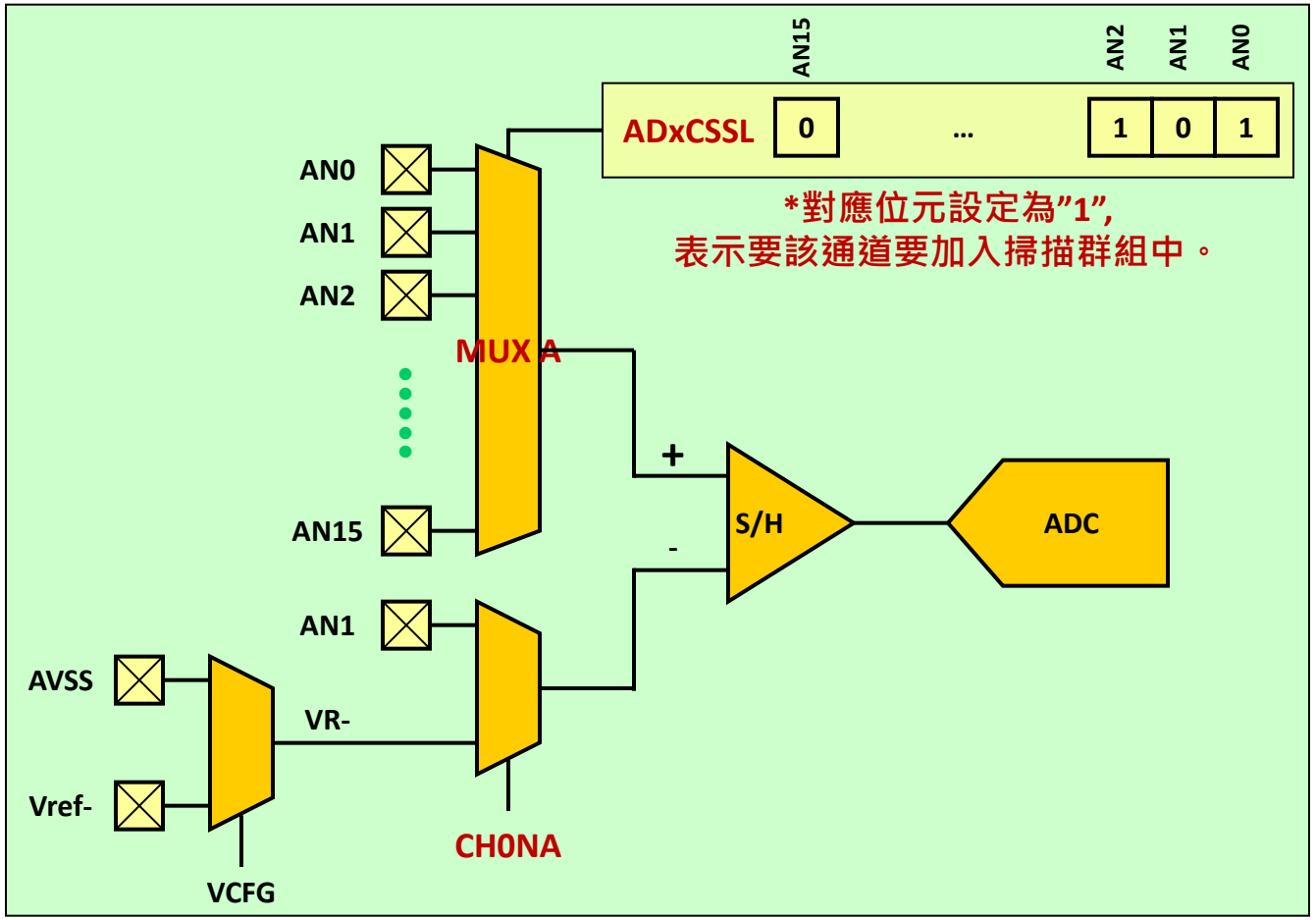
```
...
ADC1_ChannelSelect(ADC1_VR1);
...

while (1)
{
   if (T2Flag)
   {
       T2Flag = 0;
       ADC1_Stop();
   }

   if (_AD1IF)
   {
       _AD1IF = 0;
       sprintf((char *) UARTString, "VR1 Value : %4d\r\n",
             ADC1_ConversionResultGet());
       UART1_WriteBuffer(UARTString, strlen(UARTString));
   }
   UART1_TasksTransmit( ) ;
}
```

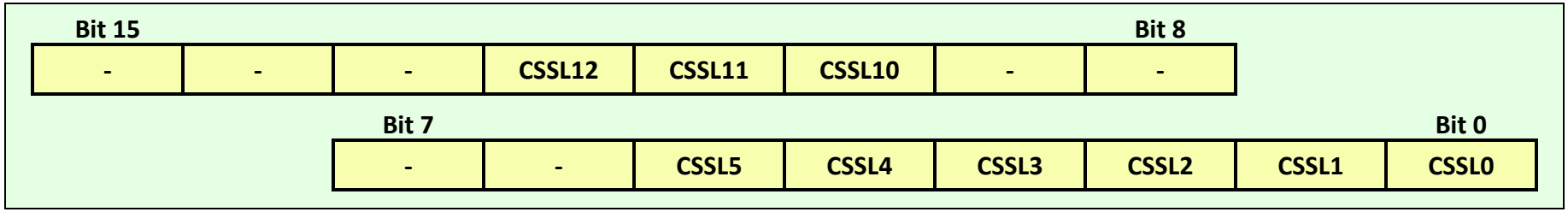
Auto Channels Scan

- 多工器A可支援自動通道掃描模式, 透過設定(CSCNA = 1), 可開啟通道掃描。
- 開啟自動掃描後, ADC會自動的逐次切換通道, 再進行AD取樣及轉換。
- 搭配SMPI的設定, 可以規劃轉換結果在Buffer中的存放方式。



Channel Scan Select

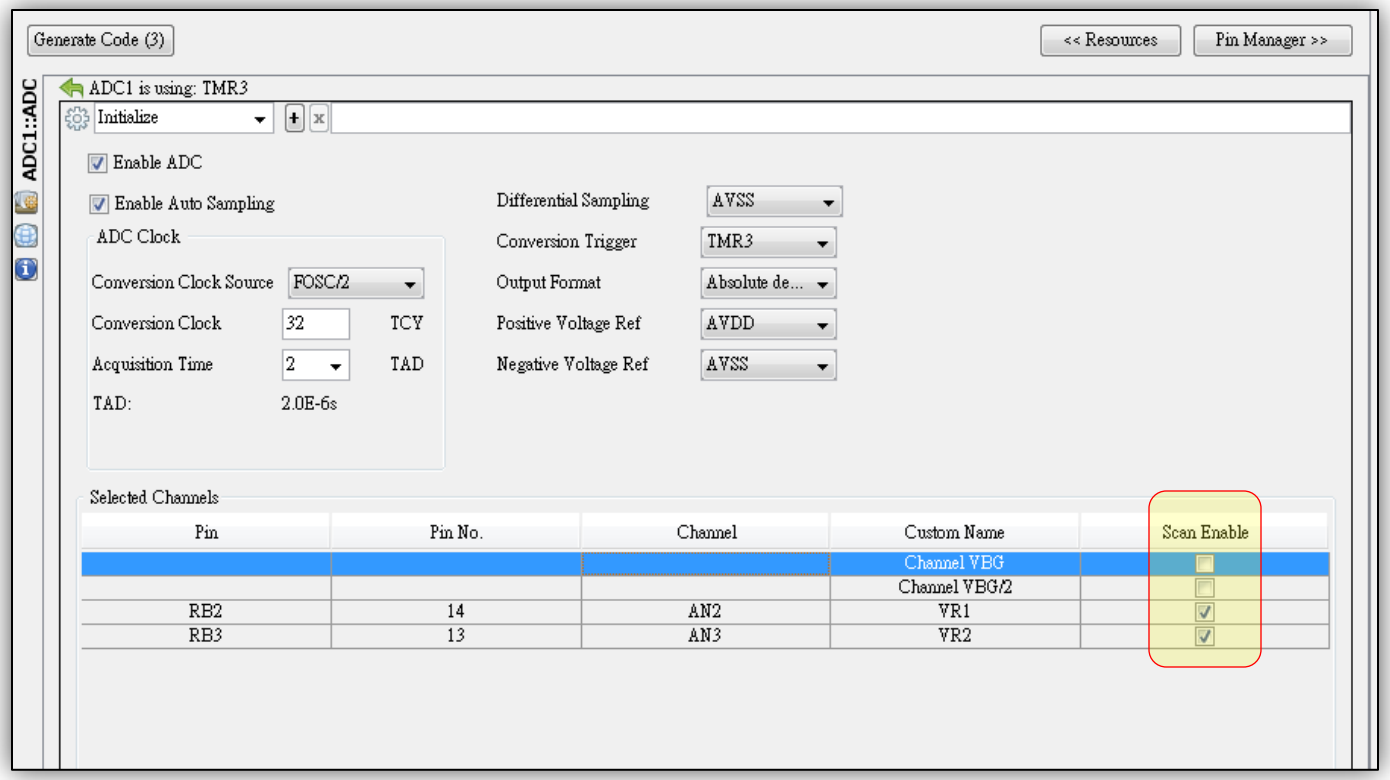
ADxCSSL



- 除了開啟通掃描模式外(ADxCON2bits.CSCNA = 1)。還必須指定哪些通道要進行掃描。指定方式就是設定ADxCSSL暫存器。被設定為”1”的通道,就會列入掃描。
- **ADC切換通道時, 會依通道編號, 由小至大自行切換通道, 轉換結果。**

MCC' Channel's Scan Support

- 在MCC中, 只需要在ADC的建構區域中, 將Select Channels後的Scan打勾✅, 就可以開啟Channel Scan功能, 並指定該通道加入Scan List。



Lab11 ADC Multi CH
Timer3 Trigger Interrupt

目 標

- 嘗試透過MCC的設定, 在Lab11架構上修改**ADC1**的設定, 將**ADC1**改成中斷架構, 中斷優先權修改為”3”。並且開啟ADC Channel Scan, 將VR1, CDS以及Temperature Sensor的數值透過掃描模式轉換。
- 搭配SMPI的設定, 規劃轉換結果在Buffer中的存放方式。
- 接著利用Timer定期的將VR1, CDS以及Temperature Sensor的數值取出，再將數值取出, 透過UART1傳送至PC顯示。顯示在終端機畫面上。

MICROCHIP
Regional Training Centers

MCC's Bug Fix for ADC Scan

- MCC v2.25.2版產生的ADC相關函數, 目前有Bug, 導致掃描功能與取值函數功能不正常。目前必須手動修改, 才能正常動作。
- **ADC1_Initialize( )中, 請加入紅色部分程式碼**
  AD1CON2bits.SMPI = 2; // 數值 = 掃描通道數 – 1, (3 – 1)
- **ADC1_Initialize( )中, 請修改紅色部分程式碼**
  adc1_obj.intSample = AD1CON2bits.SMPI **+ 1**; // 修正取值次數不正確問題

adc1.c

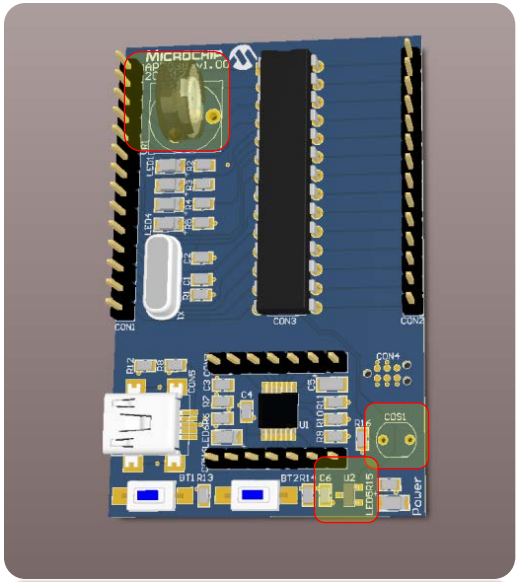
```
void ADC1_Initialize(void)
{
    AD1CON1 = 0x8044;
    AD1CON2 = 0x0400;
    AD1CON3 = 0x021F;
    AD1CHS = 0x0000;
    AD1CSSL = 0x000C;
    AD1CSSH = 0x0000;
    AD1CON2bits.SMPI = 2;
    adc1_obj.intSample = AD1CON2bits.SMPI;
    IEC0bits.AD1IE = 1;
}
```

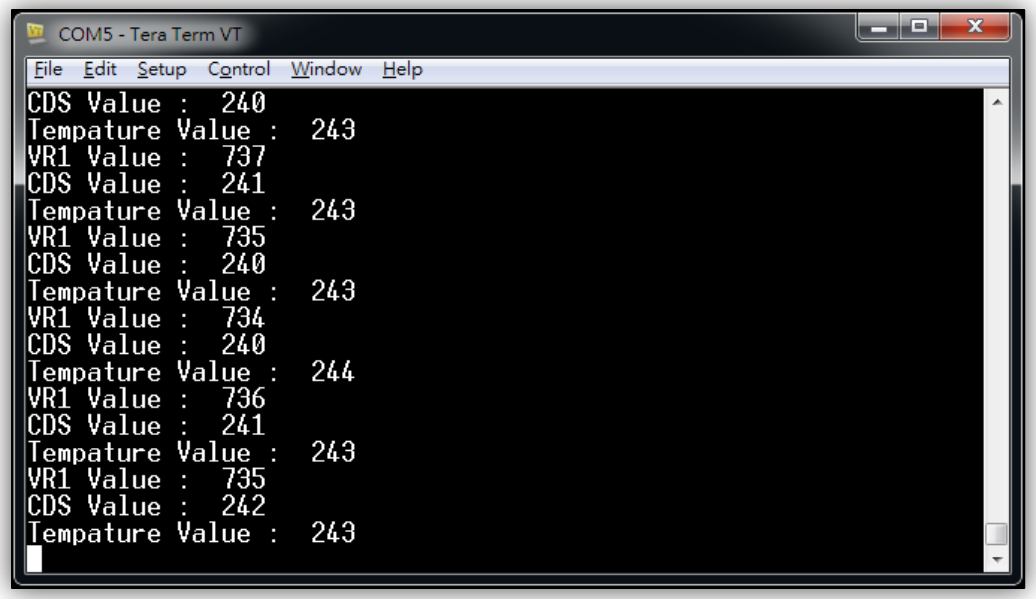
adc1.c

```
void ADC1_Initialize(void)
{
    AD1CON1 = 0x8044;
    AD1CON2 = 0x0400;
    AD1CON3 = 0x021F;
    AD1CHS = 0x0000;
    AD1CSSL = 0x000C;
    AD1CSSH = 0x0000;
    AD1CON2bits.SMPI = 2;
    adc1_obj.intSample = AD1CON2bits.SMPI + 1;
    IEC0bits.AD1IE = 1;
}
```

Lab11 ADC Multi CH Timer3 Trigger Interrupt

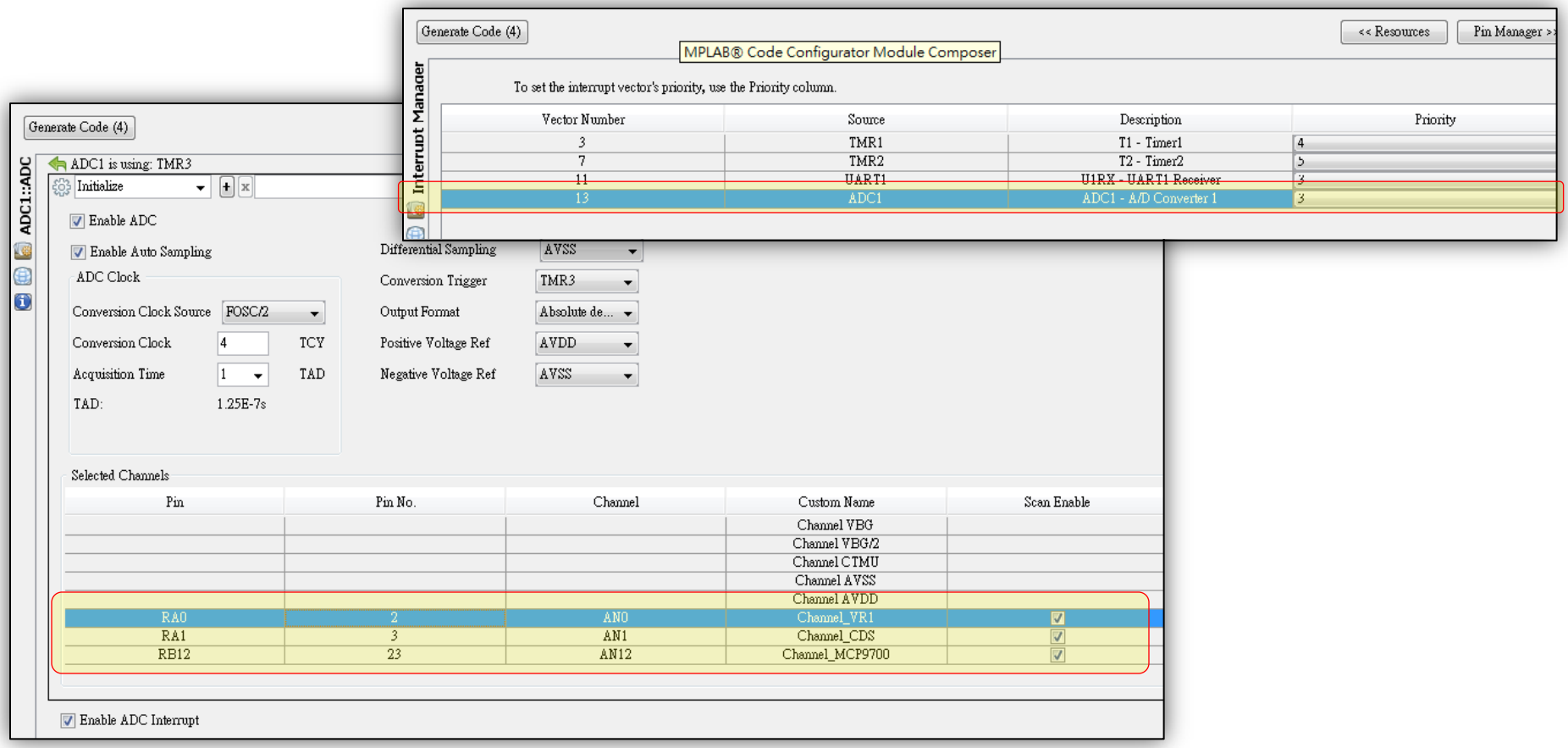
- 開啟既有專案
  (C:\Exercises\Exams\Lab11 ADC Multi CH Timer3 Trigger Interrupt.X)
- 接著開啟MCC, 修改**ADC1**資源, 根據題目要求完成設定。
- 顯示結果可以同時看到三個類比通道的資料。





Lab11 ADC MultiCH Timer3 Trigger Int.

MCC's Setting



Lab11 ADC MultiCH Timer3 Trigger Int.

Code Example

adc1.c

```
extern volatile unsigned char AD1Flag;
extern volatile unsigned int AD1Buffer[3];

void __attribute__((__interrupt__, auto_psv)) _ADC1Interrupt(void)
{
   // clear the ADC interrupt flag
   IFS0bits.AD1IF = false;

   AD1Flag = 1;
   ADC1_ConversionResultBufferGet
                                   ((uint16_t *) AD1Buffer);
}
```

main.c

```
volatile unsigned char AD1Flag = 0;
volatile unsigned int AD1Buffer[3];

while (1)
{
   …
   if (AD1Flag)
   {
      AD1Flag = 0;
      sprintf((char *) UARTString, "VR1 Value : %4d\r\n",
          AD1Buffer[0]);
      UART1_WriteBuffer(UARTString, strlen(UARTString));
      sprintf((char *) UARTString, “CDS Value : %4d\r\n",
          AD1Buffer[1]);
      UART1_WriteBuffer(UARTString, strlen(UARTString));
      …
   }
   UART1_TasksTransmit( ) ;
   …
}
```