MICROCHIP

Regional Training Centers

MICROCHIP

Regional Training Centers

串列通訊模組

SPI

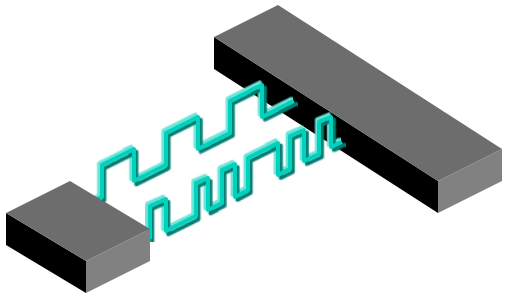
I^2C

UART

同步串列通訊(一)
Master Synchronous Serial Port (MSSP)

可工作於SPI或 I²C 兩種模式之一

- **SPI (Serial Peripheral Interface) 模式**
  - 可程式化速率設定
  - 最高速率 (@ 40 MHz) :
    - Master 10.0 Mbps
    - Slave 4.29 Mbps



  - 可程式化設定接收、發送的時脈(clock)動作極性
  - 支援兩種傳輸模式 Microwire™ 和 Motorola SPI

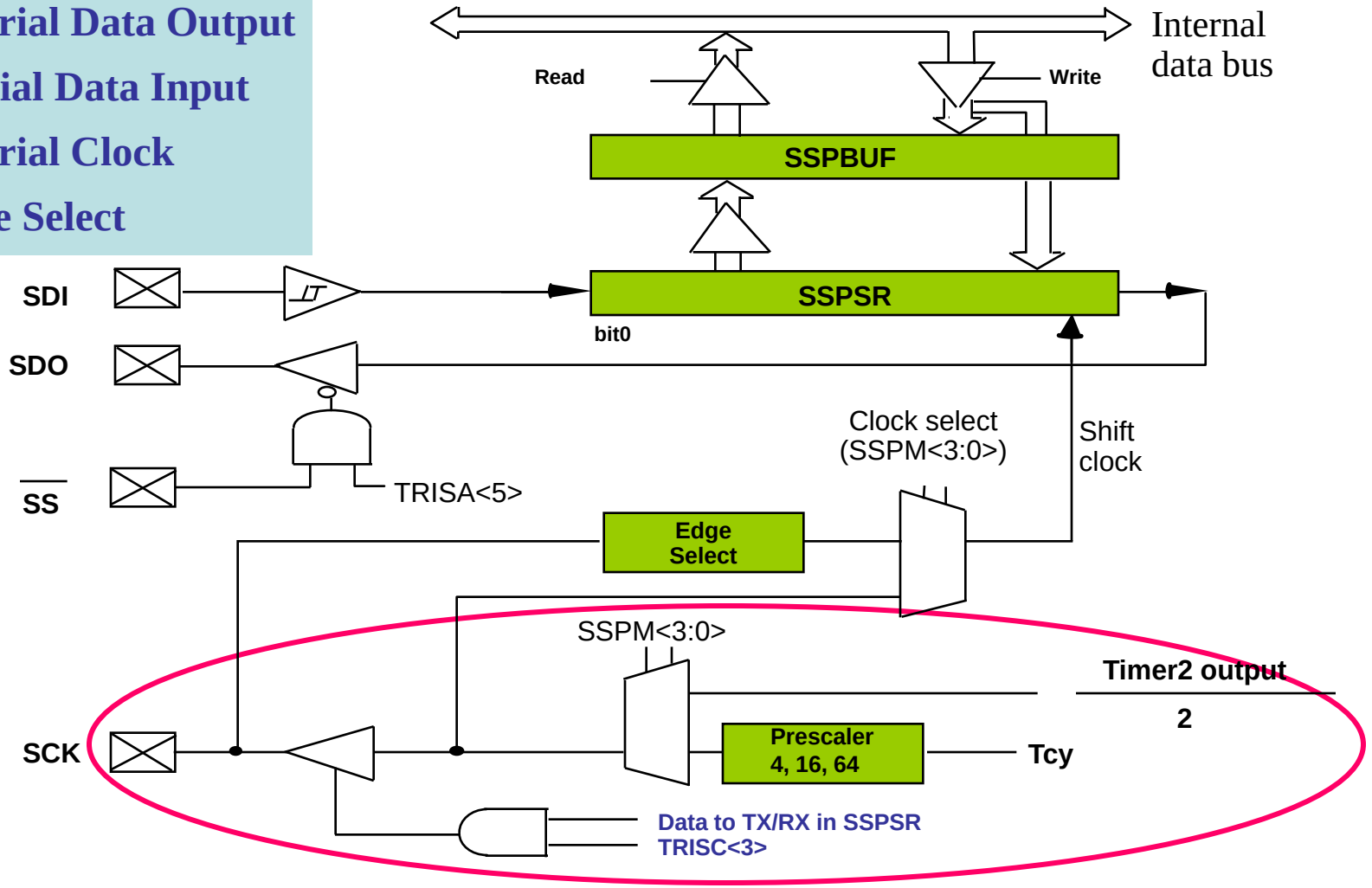
SPI 方塊圖

SDO: Serial Data Output

SDI: Serial Data Input

SCK: Serial Clock

$\overline{SS}$: Slave Select



Master Mode: SCK is the clock output pin

SPI 函數庫

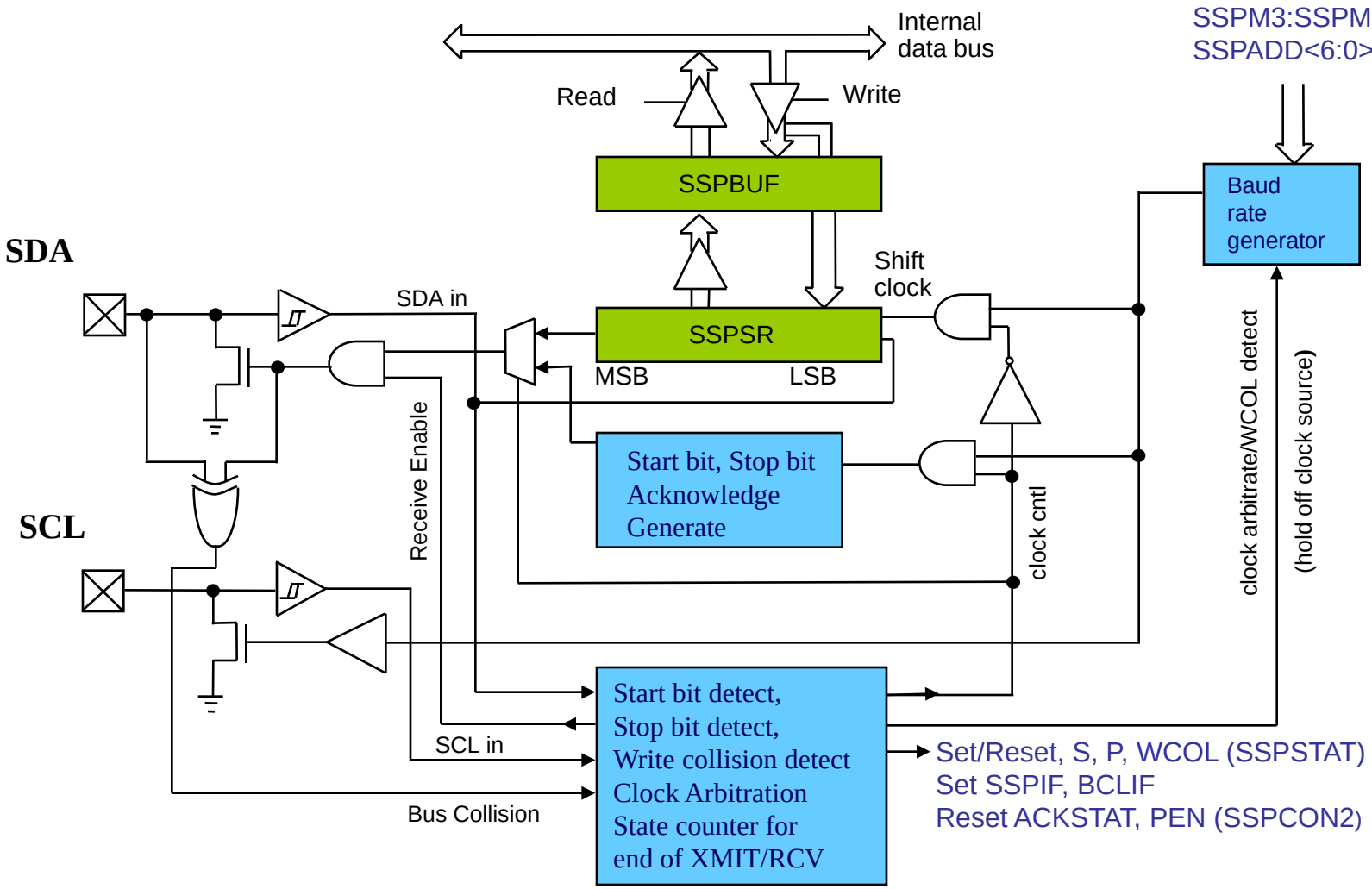
- OpenSPI( ) 設定 SPI工作模式
- DataRdySPI( ) 是否有接收資料在SSPBUF暫存器?
- getsSPI( ) 從SPI讀取一字串
- putsSPI( ) 寫一字串到SPI
- ReadSPI( ) 從SPI讀取一字元
- WriteSPI( ) 寫一字元到SPI
- CloseSPI( ) 關閉 SPI 介面

詳細 SPI 的函數操作功能及使用方式，請參閱
“C:\MCC18\doc\PIC18F Peripheral Library Help Document.chm”

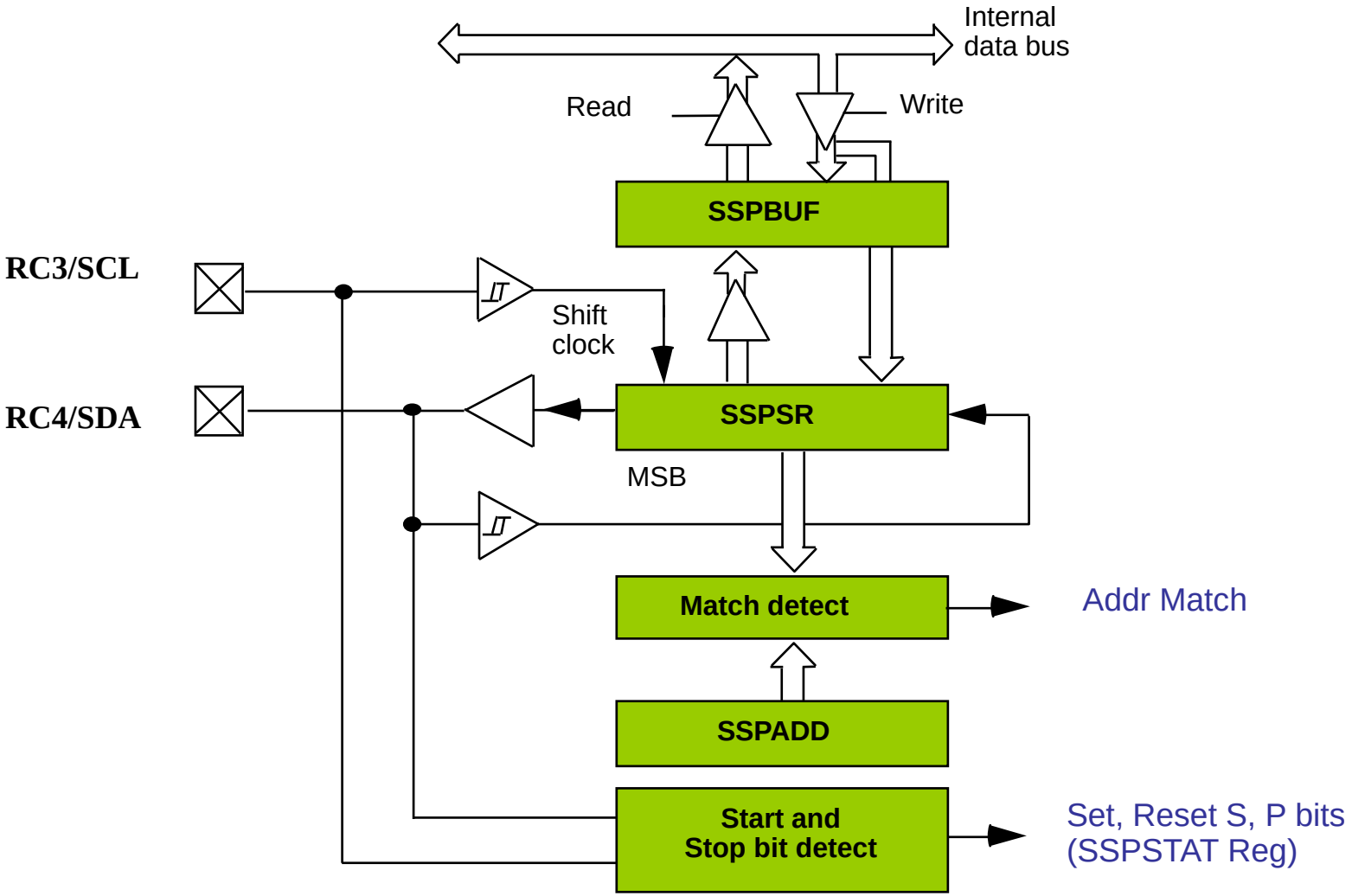
同步串列通訊(二)
Master Synchronous Serial Port (MSSP)

- **I²C (Inter-Integrated Circuit) 模式**
  - 全硬體化設計，支援下列三種I²C的操作模式
    - Master Mode
    - Multi-master Mode
    - Slave Mode
  - 支援 7-bit 或 10-bit 位址模式
  - 傳送及接收速度:100kHz， 400kHz ，1 MHz
  - 支援通用位址 (General call) 模式 “address=0x00”

I²C Master 模式方塊圖

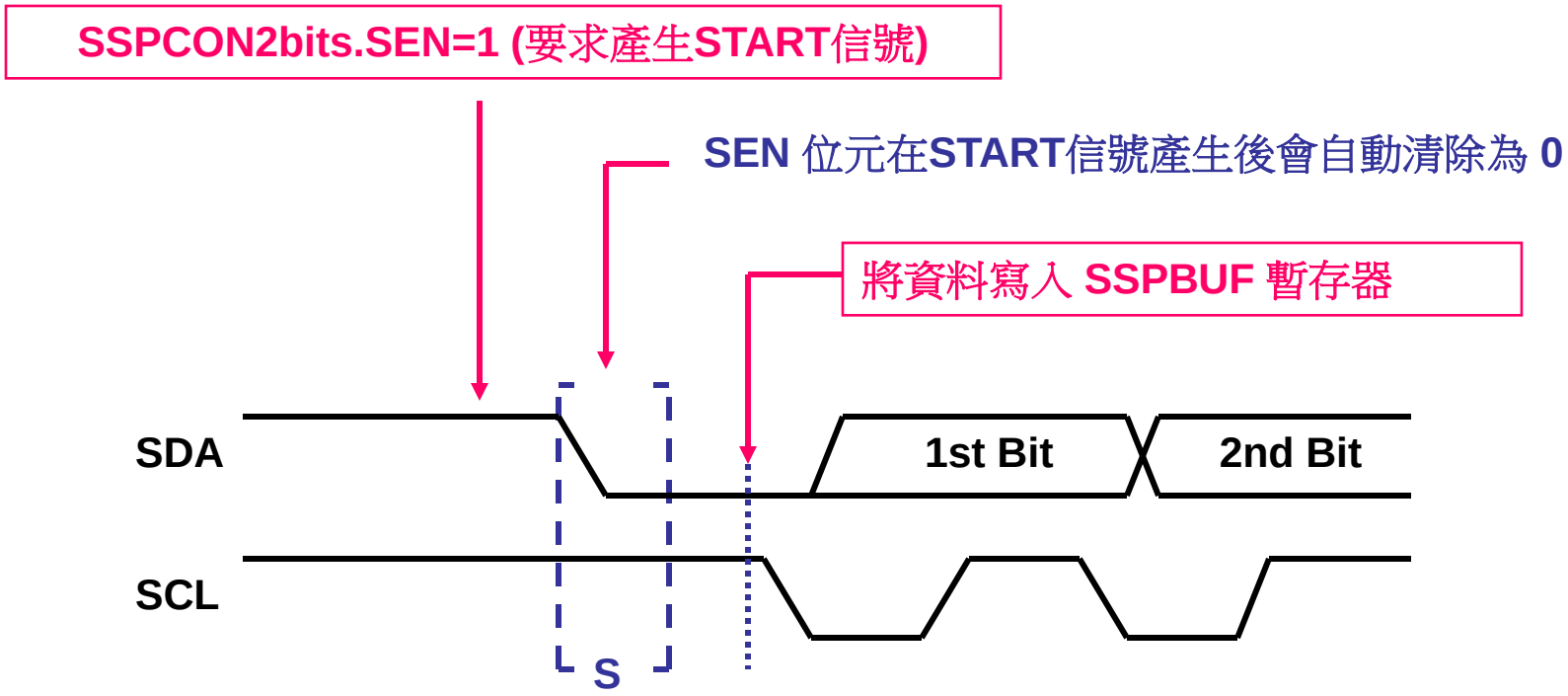


I²C Slave 模式方塊圖



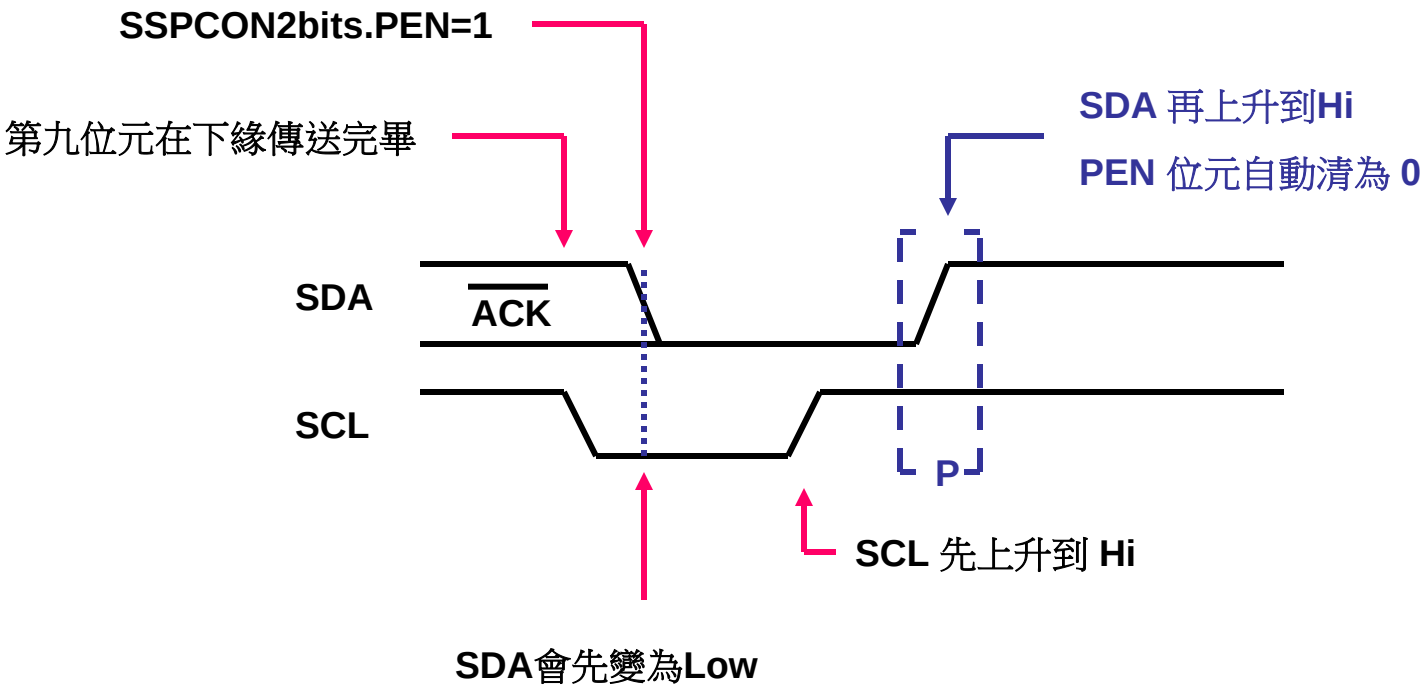
I²C Start Condition 時序

- **Bus Ready : SDA和SCL長時間都在Hi時**
- **一般而言，在SCL為Hi時SDA是不能有位準上的改變**
  - (除了Start & Stop 條件外)
- **Start : 在 SCL 為 Hi，SDA 由 Hi 變 Low 時**



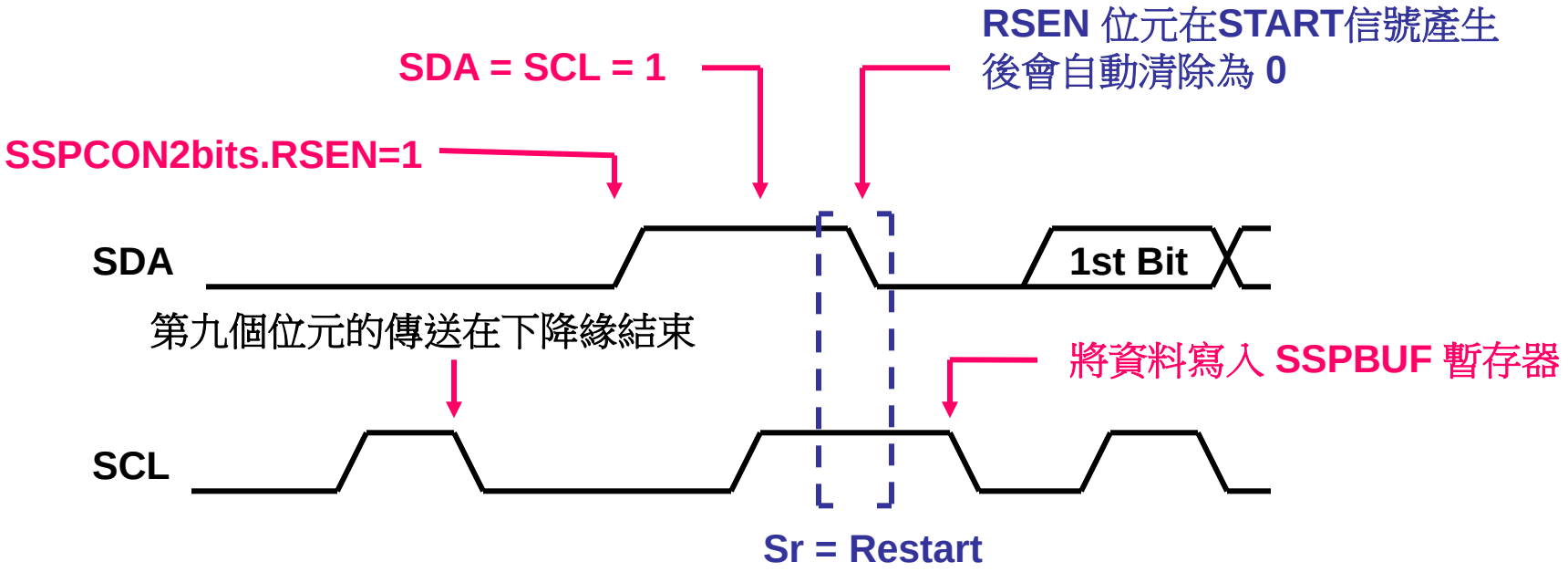
I²C Stop Condition 時序

- 產生方式 : SSPCON2bits.PEN=1
- Master用來宣告本次的I²C的傳送結束



I²C Restart Condition 時序

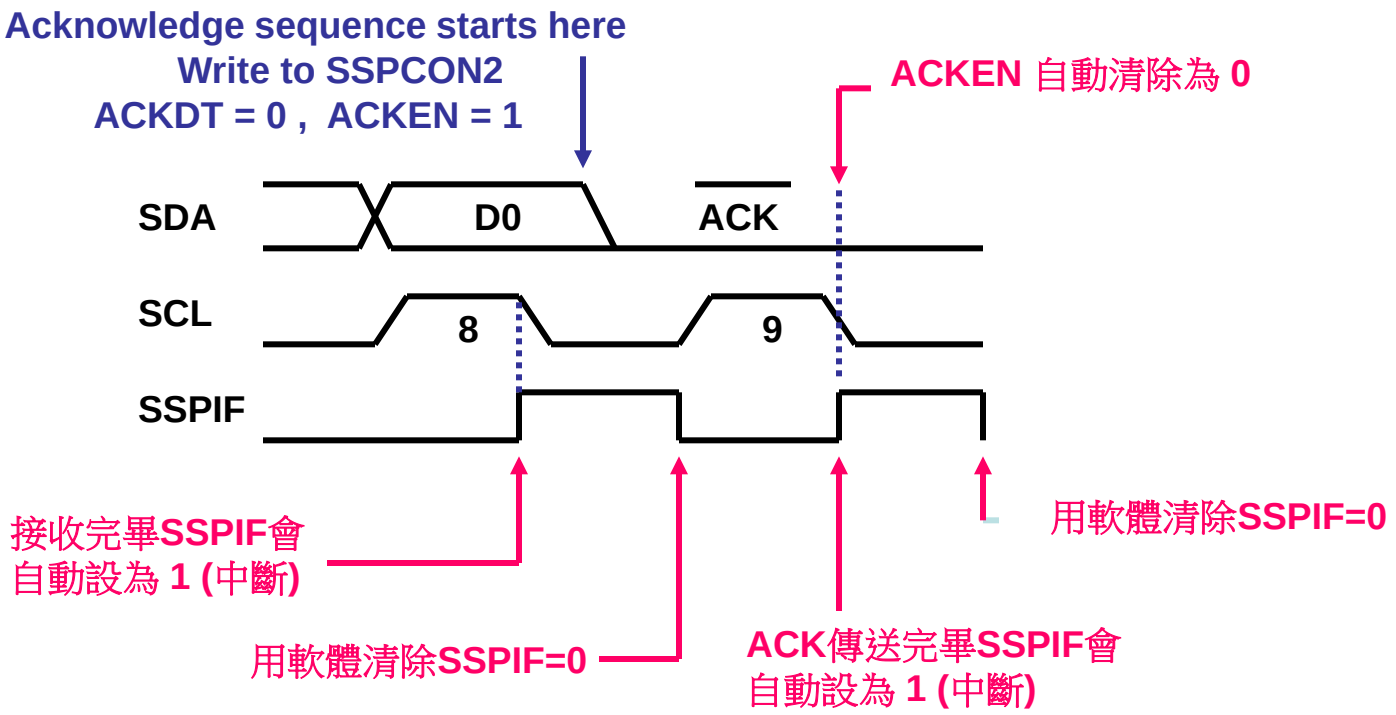
- **Restart：在第九個位元(ACK)傳送結束後，無須送出STOP訊號直接將SCL和SDA拉成Hi電位後再產生START訊號**



I²C ACK回應訊號時序

以Master回應Slave為例說明

- ✶ 設定 **SSPCON2bits.ACKDT**
  - ➢ = 0 Acknowledge , =1 Not Acknowledge
- ✸ 在設定 **SSPCON2bits. ACKEN =1** 以啟動ACK的傳送

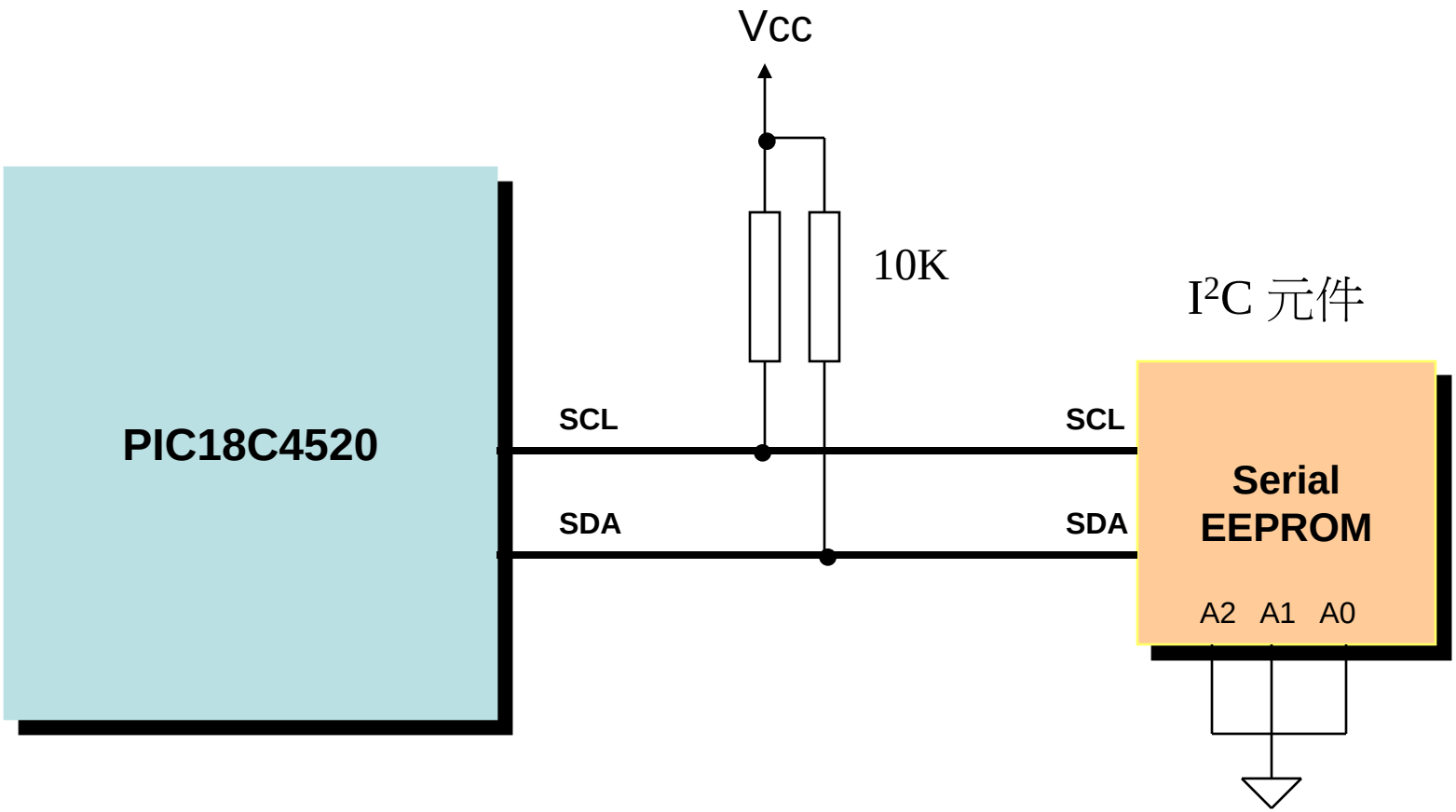


I²C ACK 回應訊號說明

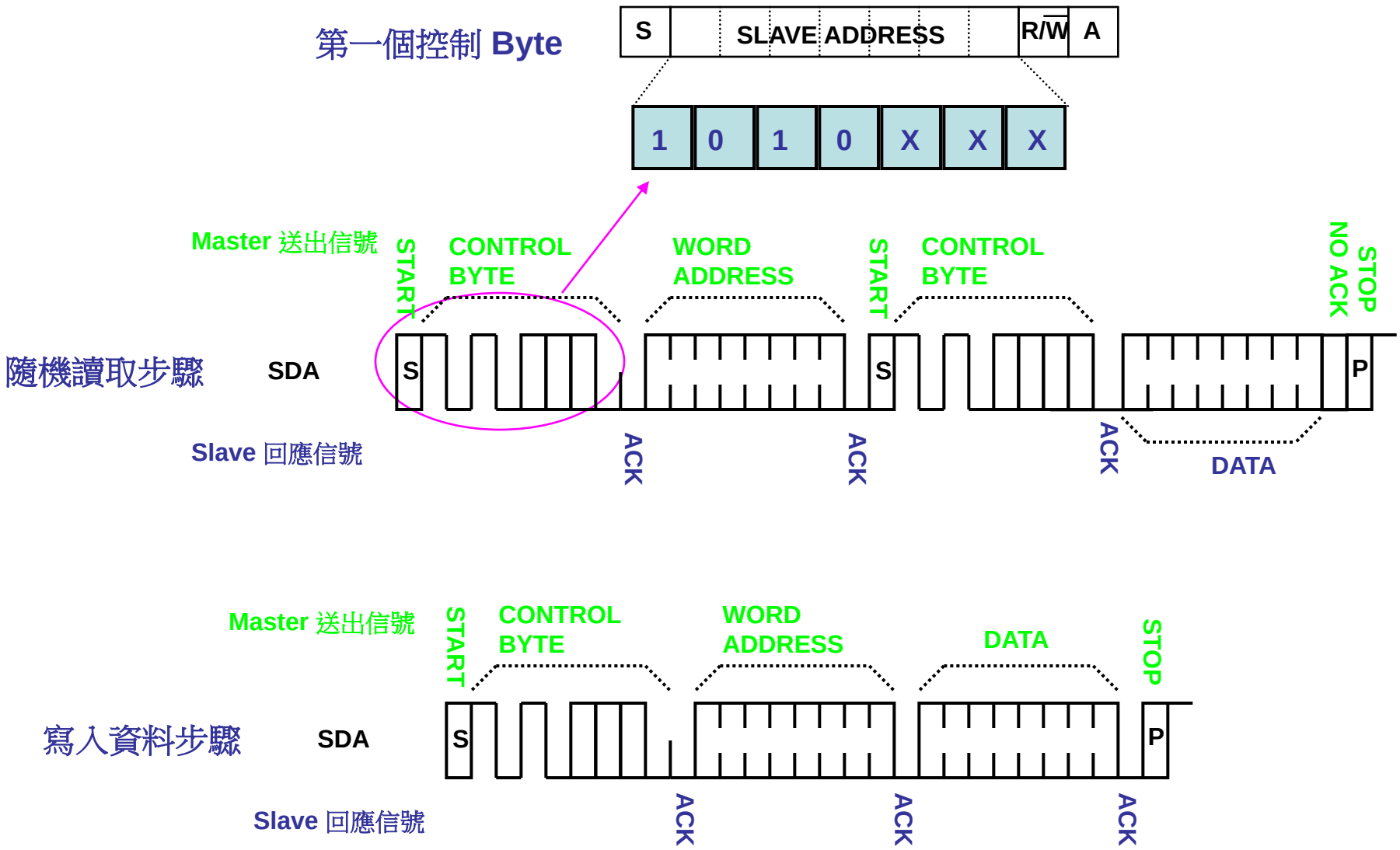
- 每一位元的傳送都在 SCL下緣時結束

- **Slave 回應 Master 以確定已接收到位址或資料**
  - 位址比對正確時
  - 收到正確資料
  - Slave回應ACK位元，0表示接收正確，1為錯誤
- **Master可以檢查 SSPCON2bits.ACKSTAT以了解Slave回覆的ACK狀態**
- **Master回應Slave以要求Slave繼續傳送資料**
  - Master回應ACK位元，0表示繼續傳送，1為結束傳送
- **Master可以設定SSPCON2bits.ACKDT再啟動 SSPCON2bits.ACKEN 位元來知會Slave**

I²C Slave 基本接線圖



I²C 存取 EEPROM 時序圖



I²C 函數庫的版本問題

- **C18** 在支援 **I²C** 的函數庫有分成數個版本，使用之前須知道該 **I²C** 被歸類在那個版本裡。
- 有關版本的歸類要參考:
  - C:\MCC18\doc\PIC18F Peripheral Library Help Document.chm 的檔案說明
  - PIC18F4520 是屬於 I2C_v1 的設定

- 例如：單是 **OpenI2C( )** 就有三種，用那一個才是正確的?
  - OpenI2C( ) , OpenI2C1( ) , OpenI2C2( )

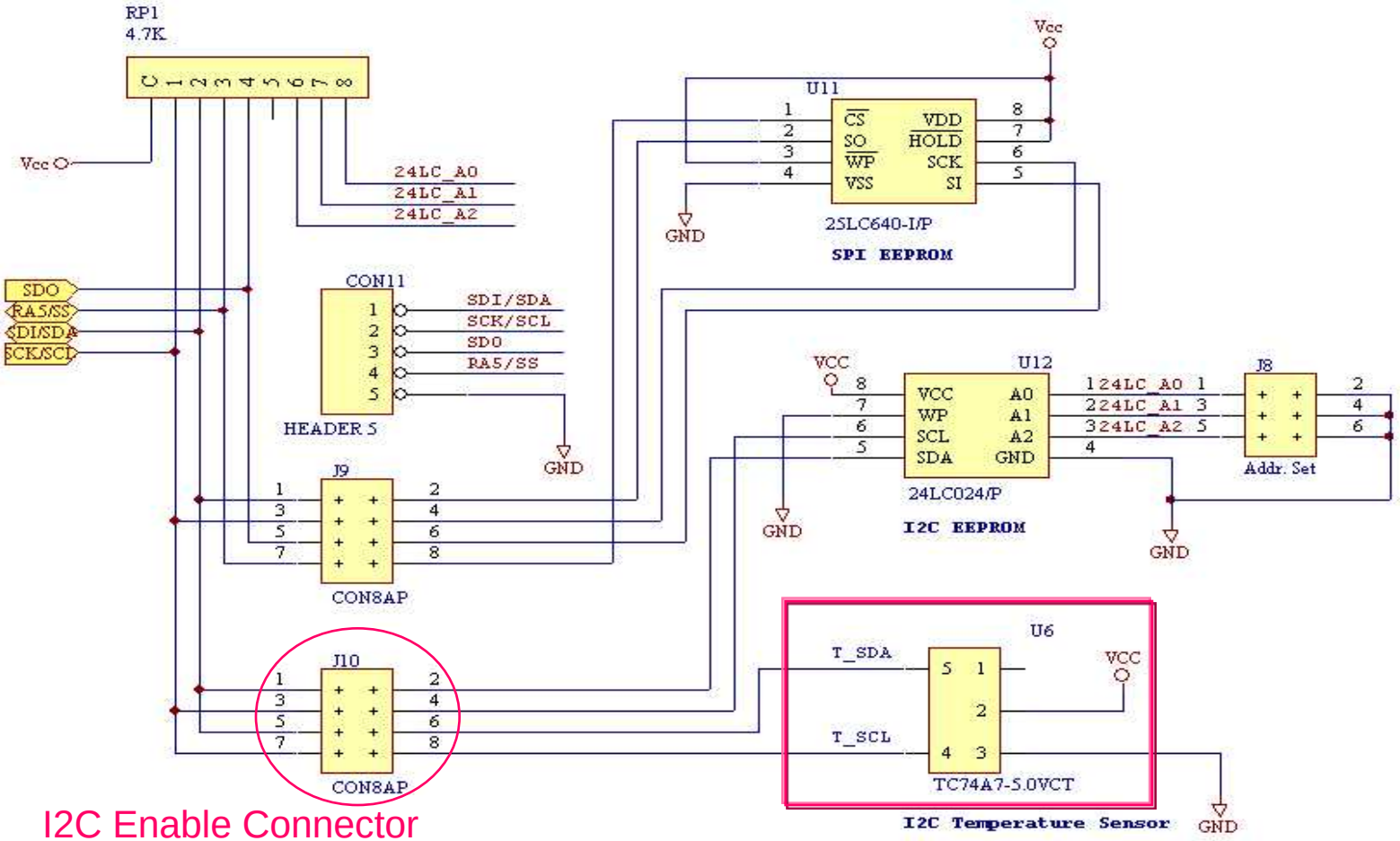
I²C Serial EEPROM 函數庫

底下 I2C EEPROM 函數適用於 24LC00 ~ 24LC16 的元件使用
也適用於大部分的 I2C Slave 元件

- **EEByteWrite (控制碼，位址，寫入的資料)**
  - 將一Byte的資料寫到指定的位址上
- **EECurrentAddRead (控制碼)**
  - 讀取目前位址上的資料
- **EEPageWrite (控制碼，位址，寫入資料的指標)**
  - 將一字串寫到指定的位址上，直到字串結束(null)
- **EEAckPolling (控制碼)**
  - 檢查 EEPROM 工作狀態
- **EERandomRead (控制碼，位址)**
  - 讀取指定的位址上的資料
- **EESequentialRead (控制碼，位址，資料存放的指標，長度)**
  - 將讀取一字串並存入到指定的RAM位址

APP001 I²C 線路

- **APP001 板子上有兩個並聯的 I²C 元件**
  - 24LC02B 及 TC74-A7 溫度感測器
  - 因為 I2C Slave Address 不一樣，不會打架



量測數位式溫度輸出

TC74 SMBus / I^2C 介面

TC74 的基本規格

- **SMBus / I²C 介面，傳送速率100KHZ**(Max.)
- 量測溫度範圍
  - +25°C ~ +85°C (精確度 +-2°C)
  - 0°C ~ +125°C (精確度 +-3°C)
- 內建溫度二極體，**Delta-Sigma A/D 轉換器**
- 轉換速率：每秒 **8** 次
- 工作電壓：**2.7V ~ 5.5V**
- 消耗電流：**200uA**，靜態電流 **5uA**
- **SOT-23 5-pin** 包裝

TC74 的應用

- CPU、硬碟溫度保護
- 電源供應器溫度保護
- PC週邊介面卡、筆記型電腦溫度保護
- 自動溫控系統
- 基板溫度量測、系統保護
- 一般室溫量測

讀取 TC74 Configuration Data

Command	Code	Function
RTR	00h	Read Temperature (TEMP)
RWCR	01h	Read/Write Configuration (CONFIG)

Bit	POR	Function	Type	Operation
D[7]	0	STANDBY Switch	Read/ Write	1 = standby, 0 = normal
D[6]	0	Data Ready *	Read Only	1 = ready 0 = not ready
D[5]- D[0]	0	Reserved - Always returns zero when read	N/A	N/A

動作順序

1. 先送出 0x01 讀取狀態
2. 讀取狀態資料後檢查 Config. 的 **Bit 6** 以了解溫度轉換是否完成
3. 再送出 0x00 讀取溫度

讀取溫度資料

- 溫度採 **8-bit** 輸出，負溫採 **2 ‘S** 格式
- **1 LSB** 代表 **1ºC** 的溫度變化

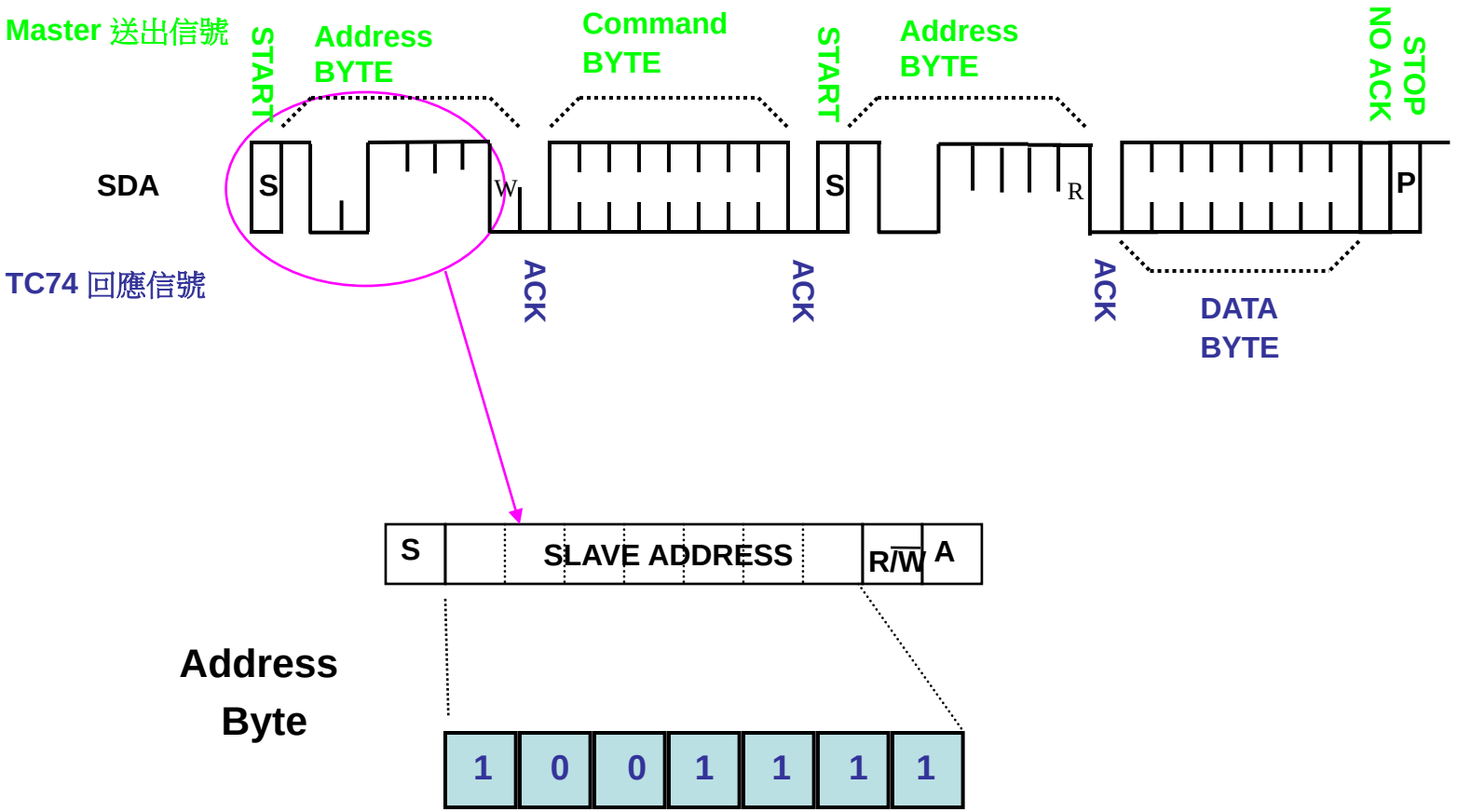
Actual Temperature	Registered Temperature	Binary Hex
+130.00°C	+127°C	0111 1111
+127.00°C	+127°C	0111 1111
+126.50°C	+126°C	0111 1110
+25.25°C	+25°C	0001 1001
+0.50°C	0°C	0000 0000
+0.25°C	0°C	0000 0000
0.00°C	0°C	0000 0000
-0.25°C	-1°C	1111 1111
-0.50°C	-1°C	1111 1111
-0.75°C	-1°C	1111 1111
-1.00°C	-1°C	1111 1111
-25.00°C	-25°C	1110 0111

TC74-Ax

- **TC74-Ax 的讀取方式**
  - TC74 為使用 SMBus / I^2C 通信協定的溫度偵測器
  - TC74 的基本位址為 1001xxx0
    - xxx 為位址位元，範圍由 0 .. 7，料號中的 Ax 表示其使用到的位址 (TC74-A0 .. TC74-A7)
    - TC74-A7 表示使用的是 10011110 為 I^2C 位址
  - TC74 的讀 / 寫 格式與一般標準的 EEPROM 相同，只是位址不同而已

TC74-A7 的 I²C時序

讀取資料方式與 24LCxx 的 EEPROM 類似



I²C 的相關函數

- **MPLAB C18 Library 提供的基本 I²C Functions**

Function	Description
AckI2C	Generate I²C bus *Acknowledge* condition.
CloseI2C	Disable the SSP module.
DataRdyI2C	Is the data available in the I²C buffer?
getcI2C	Read a single byte from the I²C bus.
getsI2C	Read a string from the I²C bus operating in master I²C mode.
IdleI2C	Loop until I²C bus is idle.
NotAckI2C	Generate I²C bus *Not Acknowledge* condition.
OpenI2C	Configure the SSP module.
putcI2C	Write a single byte to the I²C bus.
putsI2C	Write a string to the I²C bus operating in either Master or Slave mode.
ReadI2C	Read a single byte from the I²C bus.
RestartI2C	Generate an I²C bus *Restart* condition.
StartI2C	Generate an I²C bus *START* condition.
StopI2C	Generate an I²C bus *STOP* condition.
WriteI2C	Write a single byte to the I²C bus.

I²C EEPROM 的相關函數

- **MPLAB C18 Library 提供的整合式 I²C Functions**
  - 這些 Functions 提供批次處理的功能, 只要提供正確的位址資訊及欲寫入的資料即可
  - EERandonRead 及 EECurrentAddRead 會將讀入的資料及執行結果放在一個整數 (int) 的資料型別中傳回

Function	Description
EEAckPolling	Generate the Acknowledge polling sequence.
EEByteWrite	Write a single byte.
EECurrentAddRead	Read a single byte from the next location.
EEPageWrite	Write a string of data.
EERandomRead	Read a single byte from an arbitrary address.
EESequentialRead	Read a string of data.

EERandonRead 函數

- **EERandonRead** 的使用範例與傳回結果

```
int temp ;                                  // 宣告一個整數值來接收傳回值
temp = EERandonRead(0xA0,0x30) // 讀取位址 0x30 的資料
```

- 如果讀取的過程無誤, 則傳回值的 **High Byte** 會是 **0** 而 **Low Byte** 則包含讀入的資料 , 此時的傳回值為正數
- 若讀入的過程中有錯誤, 則傳回值的狀態值會放在 **High Byte** ， 此時可檢查整個傳回值的內容來得知錯誤訊息 , 此時的傳回值為負值
  - -1 Bus Collision Error Happened
  - -2 No ACK Error Happened
  - -3 Write Collision Error Happened

MICROCHIP

Regional Training Centers

MICROCHIP

Regional Training Centers

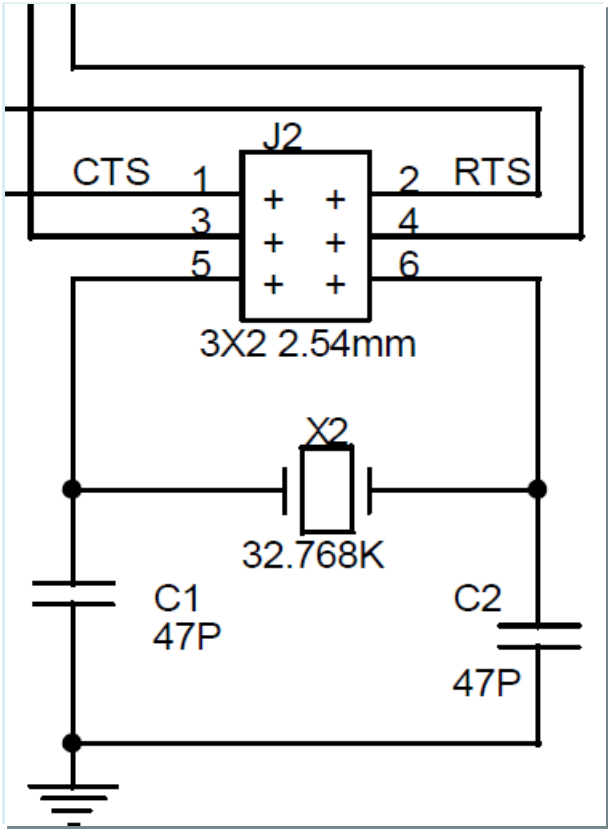
練　習　五

讀取 I2C 溫度感應器並顯示溫度

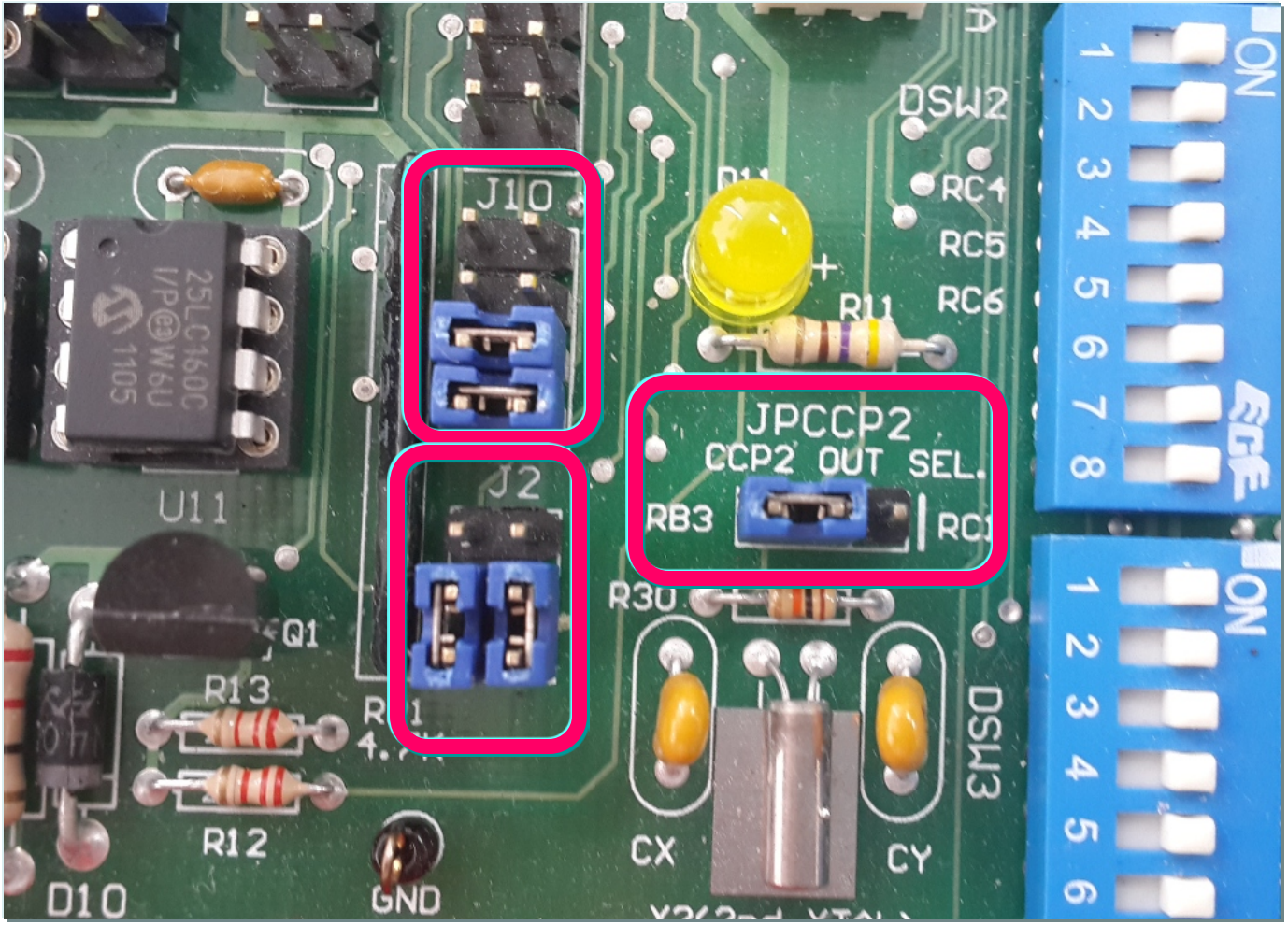
Lab5 Jumper 設定

- 因為使用 **Timer1** 的 **32768Hz** 石英振盪，注意 **JP2** 需短路 **3&5** 及 **4&6**

- **JPCCP2 PWM2 輸出選擇**
  - PWM2 輸出接至 RB3
  - RC1 空接 (給 T1OSI 使用)
- **JP10 需短路最底下的兩個 Jumper 接通 TC74 的I²C**



Lab5 Jumper 設定



J10 :選 TC74A

J2 :啟用 32768Hz Crystal

JPCCP2 : RC1
此處為空腳，PWM 輸出接至 RB3

JP13 : USBCAP
腳位在 F4520 為 RC3/SCL 所以不要接。使用18F4550 才需要短路。

Lab5

I2C 溫度感測器的讀取與顯示

開起專案及相關檔案內容

開啟 **..\TLS2118\Lab5\Read TC74.mcp** 的專案，完成程式中的練習後可正確將溫度顯示在 **APP001 Demo Board** 的 **LCD.**

Project 中有四的程式模組:

1. **Init_MCU.c**
2. **Main.c**
3. **Read_Temp.c**
4. **WAP_LCD.c**

以及共用的含入檔 **main.h**

Lab5

函數功能

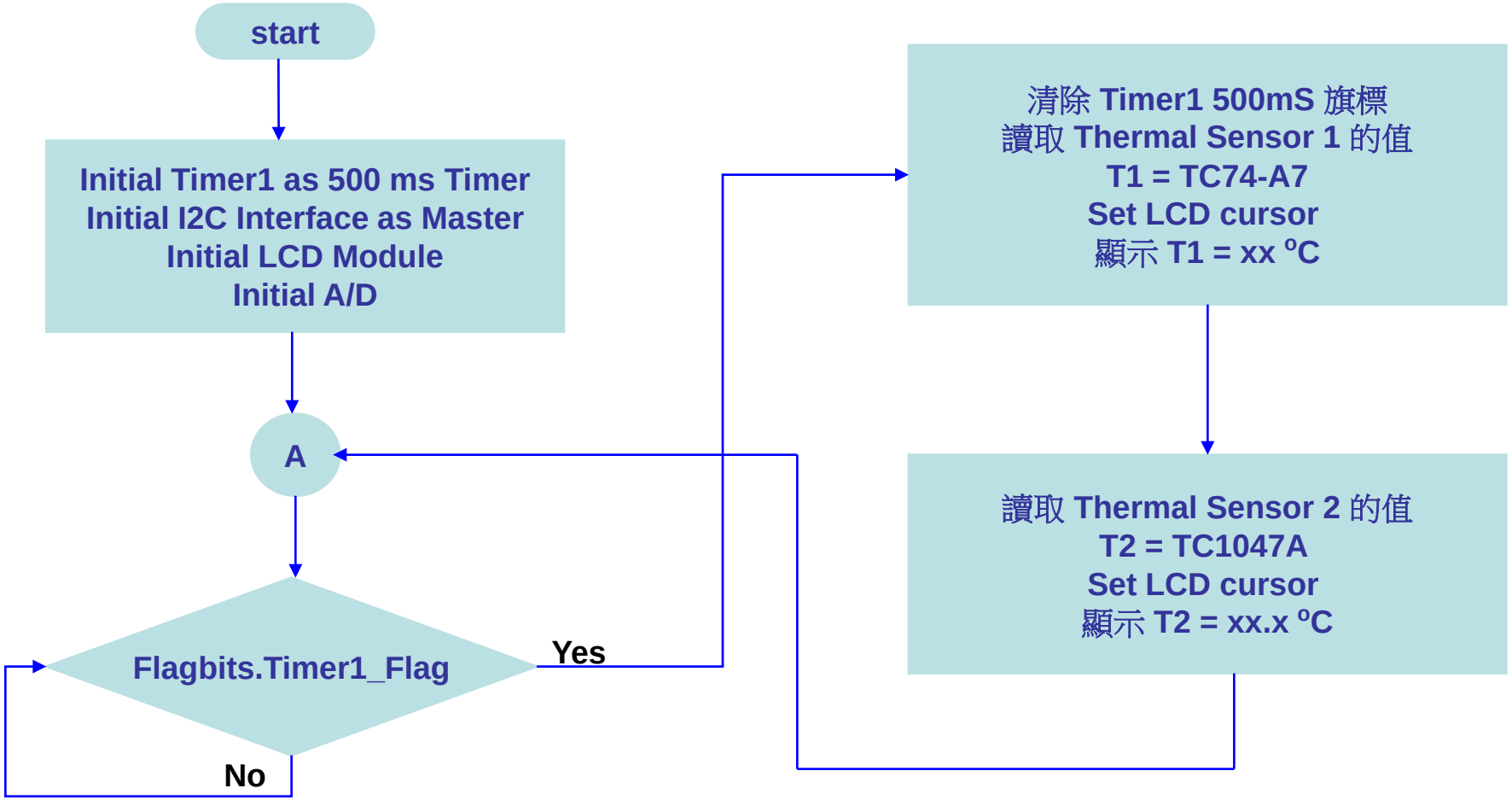
了解各函數的主要功能

1. **Init_MCU.c**
   - 初始設定 **PIC** 的各種周邊及工作條件
2. **Main.c**
   - 主程式，在主迴圈內固定讀取溫度值並顯示
3. **Read_Temp.c**
   - 讀取 **TC74-A7** 及類比式溫度電壓值 **(TC1047A)**
4. **WAP_LCD.c**
   - **LCD** 顯示用函數

main.h 以上所以程式的函數原型宣告

Lab5
流程圖

練習五的流程圖



Lab5-1

規劃 Timer1 使用外部 32768Hz 石英震盪

Timer1 每 0.5 秒中斷一次

- **請使用 OpenTimer1 ( ) , WriteTimer1( ) 來將 Timer1 規劃為一個 500 ms 中斷一次的計時器**
  - 使用 Timer1外部的石英振盪器 (32768Hz 的 XTAL)
  - 將中斷打開 – RCON 暫存器的 IPEN 位元要設為 1 , 如此才有高/低優先權的區分
  - IPR1 中的 TMR1IP 設為 “0” , Timer1 的中斷為低優先權
  - 記得 ! 將 Timer1 設為 16 bit RW 模式
  - 因為 Timer1 使用的是 32768Hz 的振盪器, 故寫入值若為 (65536 - 16384) 則可讓 Timer1 每 500ms 後產生中斷

Lab5-2

Timer1 中斷部分

Timer1 中斷處理

- 低優先權的 ISR 名稱為 isr_low，請在 Timer1 中斷後進行下列的工作
  - 使用 WriteTimer1( ) 將 Timer1 的 500 ms 值重新載入設定
  - 清除 PIR1bits. TMR1IF ，讓 Timer1 可再次中斷
  - 設定 Flagbits. Timer1_Flag 已通知主程式
    - Flagbits 已被宣告為 “位元結構”

Lab5-1 & Lab5-2

Timer1 的修改

- 依據上兩張投影片的說明修改或加入程式
  - Init_MCU.c 的程式下:
    - OpenTimer1( ) 函數 – 設定使用外部的 32768Hz 的石英振盪及啟動震盪電路。
    - WriteTimer1( ) 函數 – Timer1 的直要填入多少才會有0.5秒的中斷時間
    - 設定 Timer1 為低優先權中斷
  - main.c 的程式下:
    - 程式最下方的 isr_low_direcr( ) 中斷向量轉移函數
    - 利用 #pragme 宣告低優先權位址，及使用嵌入式組語跳到 isr_low( ) 的中斷函數去執行

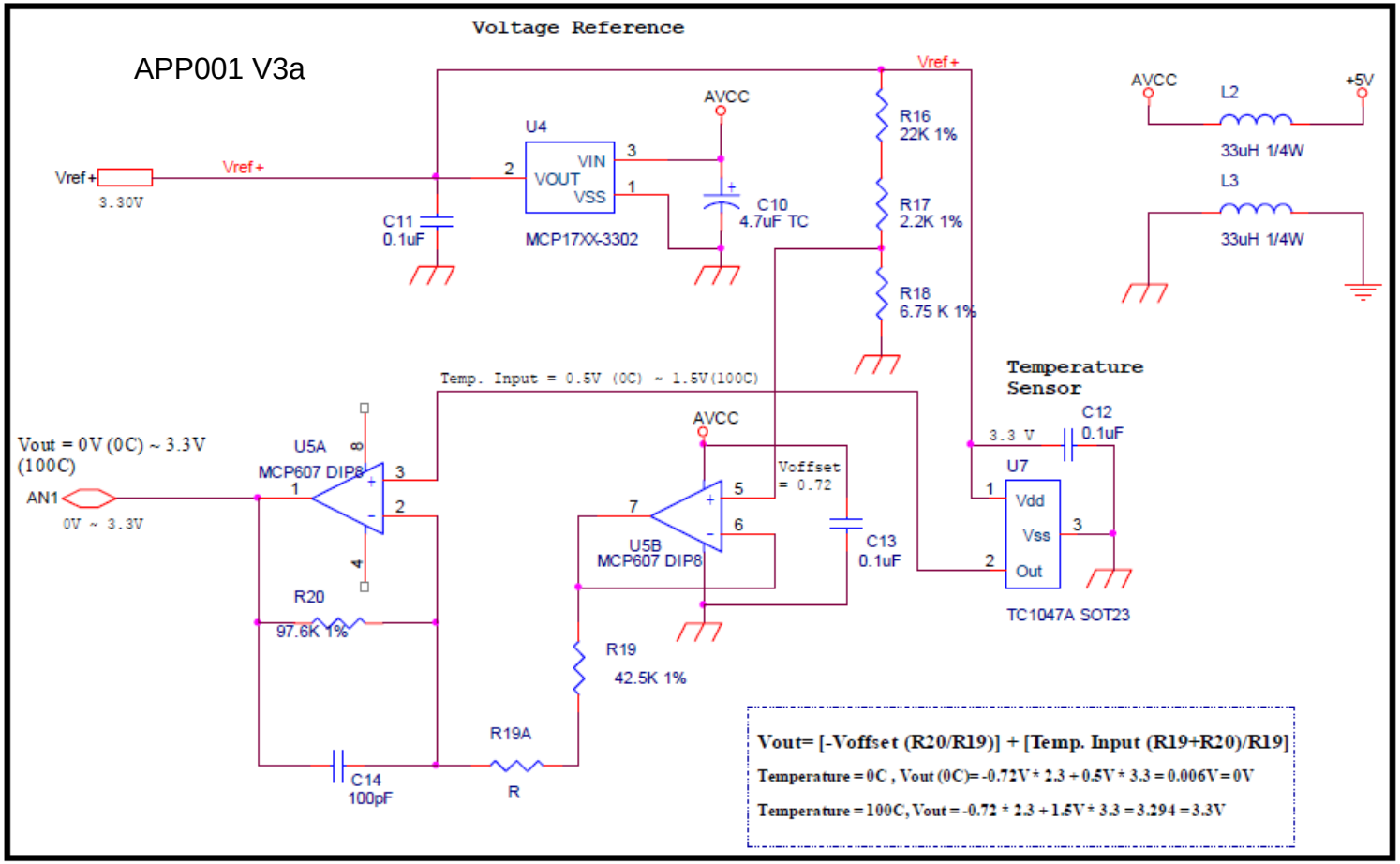
練習五
Read-Temp.c

讀取 I2C 及類比溫度值

- **讀取 TC74-A7 溫度值**
  - 用內建 EEPROM 的函數讀取溫度值
  - 使用函數 EERandomRead( )
- **讀取 TC1047A 溫度值**
  - 用18F4520 內建的 10-bit A/D 轉換器讀取
  - 參考電壓使用外部 4.096V

APP001 V3a 類比溫度電路

- APP001 V3a 實驗板使用 MCP1700-3.3V 的參考電壓源
- APP001 V1 & V2 使用 MCP1541 4.096V 的參考電壓源



練習五
main.c

主程式的處理程序

- 每次 **Timer1** 中斷發生一次後就呼叫一次函數 **LCD_Temp_Update( )**
  - 請在 LCD_Temp_Update( ) 中 ….
    - 讀取 TC74-A7 的溫度值，並顯示在 LCD
    - 讀取 TC1047A 的溫度值，經轉換成字串型態並加入小數點後，顯示在 LCD
  - 讀取 TC74 的函數使用 EERandomRead ( )， 可以偵測如 IC 不存在或異常的多種錯誤狀況

Lab5-3 & Lab5-4

讀取T1 & T2 的修改

- **T1 (TC74-A7) I²C 的溫度讀取**
  - Read_Temp.c 下的 Read_TC74_temperature( )
    - 使用 EERandomRead( ) 函數先讀取 TC74 的狀態 (bit 6) 旗號
    - 使用 EERandomRead( ) 函數讀取溫度值
- **T2 (TC1047) 類比式的溫度讀取**
  - Read_Temp.c 下的　Read_TC1047_Temperature( )
    - 將 ADC 輸入由原先的 VR1 (測試用）改成類比溫度的輸出端
- **討論一下**
  - main.c 行號136~150
  - for 迴圈計算小數點的位置的方法

練習五

程式完成後的顯示

完成溫度的顯示

- 讀取兩個不同型態的溫度感應器並將其轉換後的溫度顯示在 LCD 顯示幕上
  - T1 為數位式溫度，範圍為兩位數
  - T2 為類比式溫度，範圍為 xx.x
  - 每隔 500mS 更新顯示的溫度

```
Lab 5 Read Temp.
T1=24⁰  T2=24.5⁰
```

LCD 顯示幕

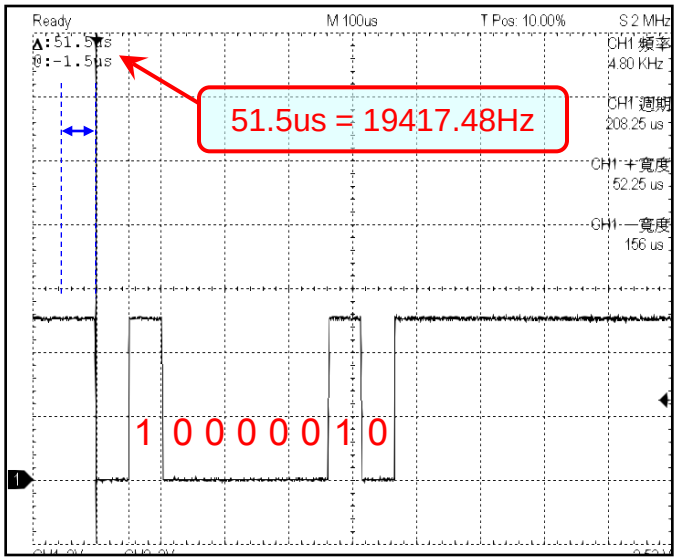
非同步串列通訊

UART

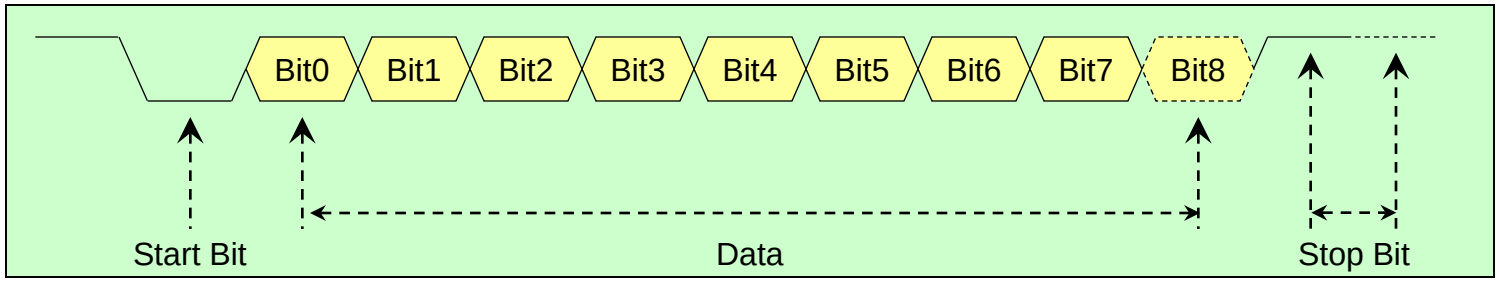
本練習摘自 W401 V3 的教育訓練

何謂 UART

- UART:Universal Asynchronous Receiver Transmitter,泛用非同步接收傳送模組。
  一個以串列方式進行資料交換與溝通的通訊模組。
- UART 傳輸的資料由LSB先傳。格式如下,包含一個起始位元,八或九位元的資料,一至二個停止位元。

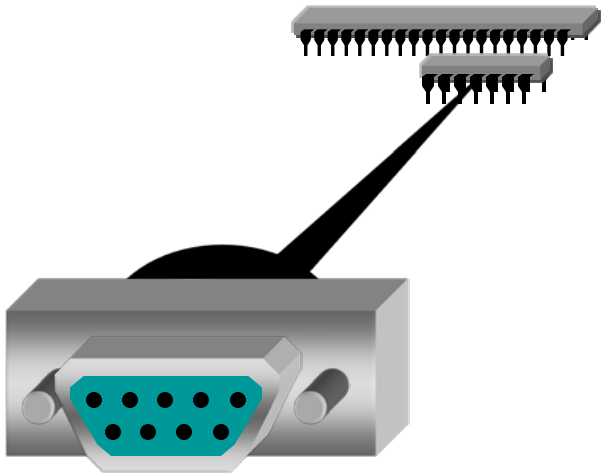


UART傳送範例'A'(0x41),19200 bps



EUSART 概述

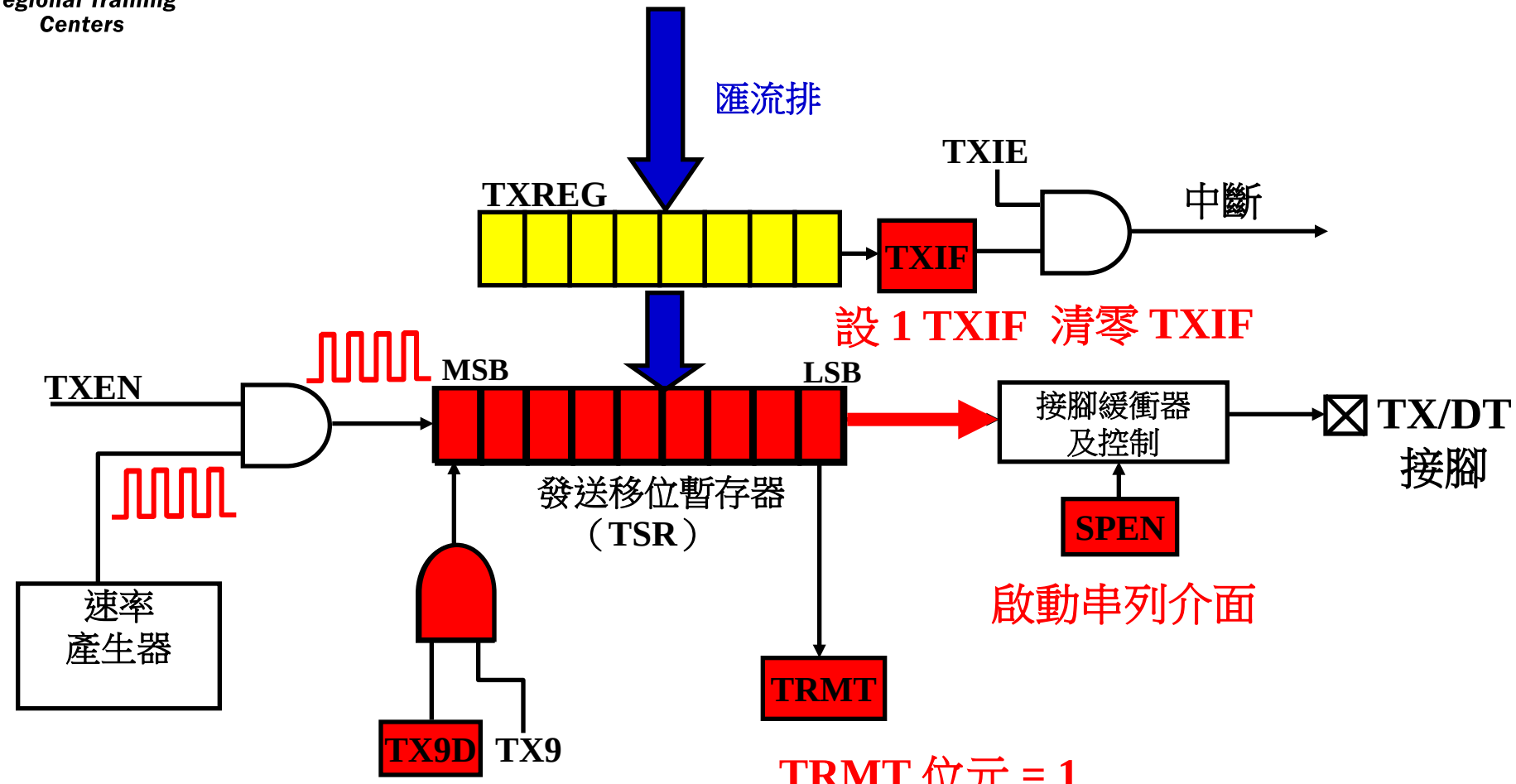
- **串列式 I/O 通信周邊**
  - 有時也稱為 “串列通信介面” 或 SCI
- **主要功能：**
  - 同步或非同步模式
  - 可以接收或發送
    - 全雙工非同步發送和接收 (UART)
    - 半雙工同步主模式和從模式
- **最常使用**
  - RS-232 與 PC 串列端通信
    - 需要用 RS-232 電位轉換的驅動器 (Max232)
- **增加 LIN Bus 介面**
- **非同步接收自動速率偵測功能**



EUSART 暫存器

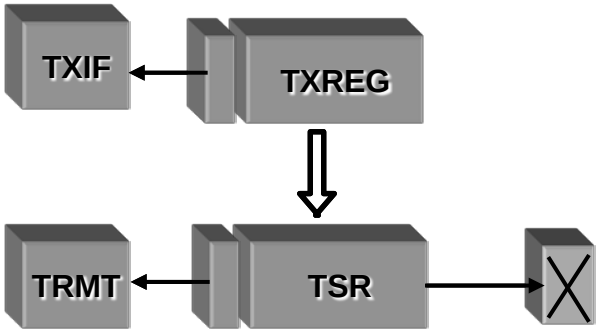
- **EUSART 所使用的暫存器：**
  - 速率產生暫存器：
    - SPBRG 和 SPBRGH
  - 發送狀態和控制暫存器（TXSTA）
  - 接收狀態和控制暫存器（RCSTA）
  - 接收和發送資料暫存器
    - 發送資料暫存器（TXREG）
    - 接收資料暫存器（RCREG）

發送端方塊圖


匯流排
TXIE
TXREG
中斷
TXIF
設 1 TXIF 清零 TXIF
TXEN
MSB
LSB
接腳緩衝器
及控制
TX/DT
接腳
發送移位暫存器
（TSR）
SPEN
速率
產生器
啟動串列介面
TRMT
TX9D
TX9
TRMT 位元 = 1
表示移位暫存器為空的
第9-bit 資料位元
TSR中有資料時
TMRT 位元 = 0

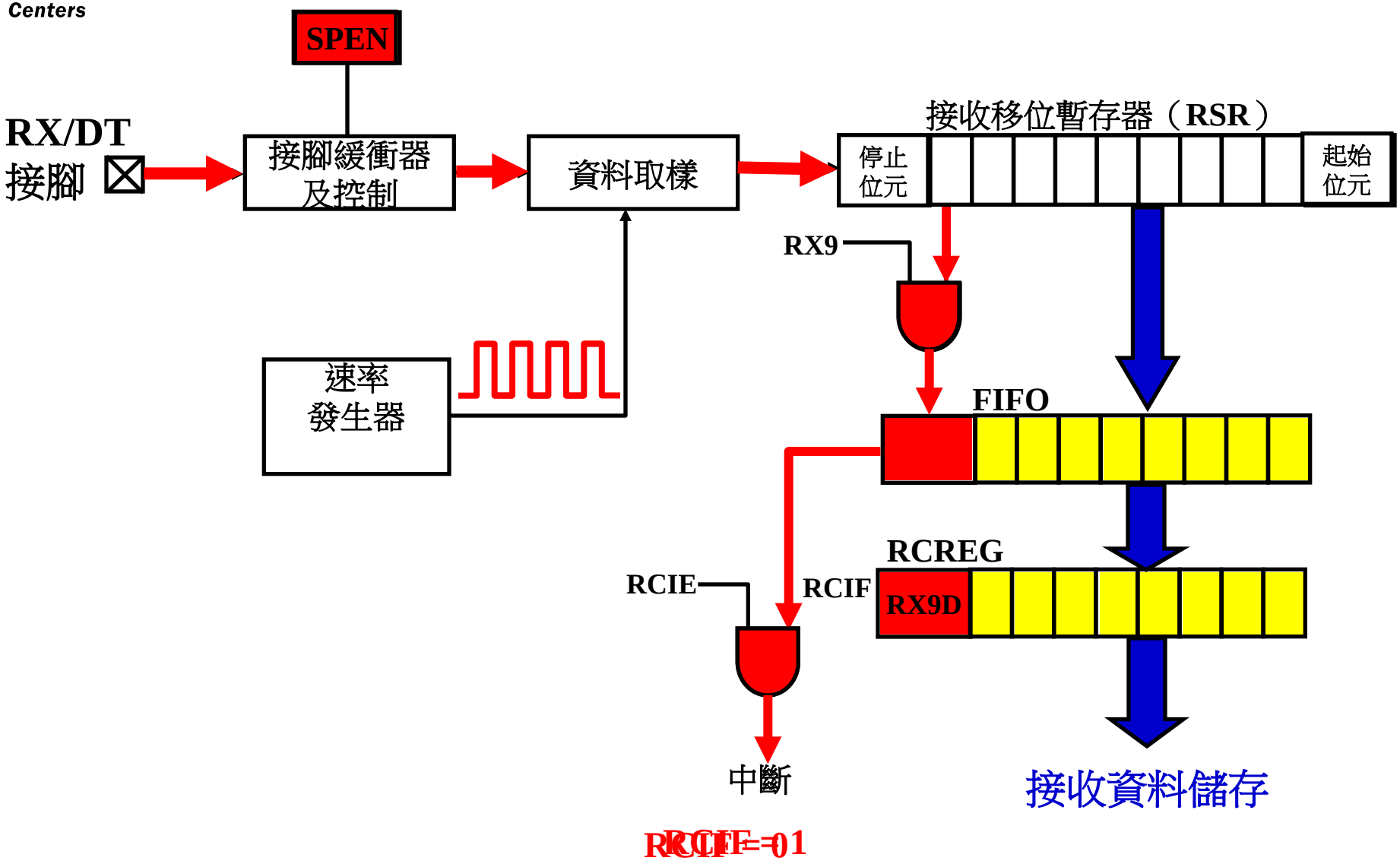
TXIF & TRMT 的動作

- 假如TXREG的資料被載到TSR，TXREG會空出;則TXIF = 1
- 假如TSR的資料串列傳送完畢;則TMRT = 1
- 假設TXREG剛載入資料時TMRT為空的(TMRT=1)， 則這筆資料會立即被送到TSR，串列傳送會動作，同時TXIF = 1
- TXIF是可單獨使用，即使USART的TX中斷是關閉的(TXIE=0)
- 由以上動作可知偵測發送狀態TXIF會比TMRT來的快



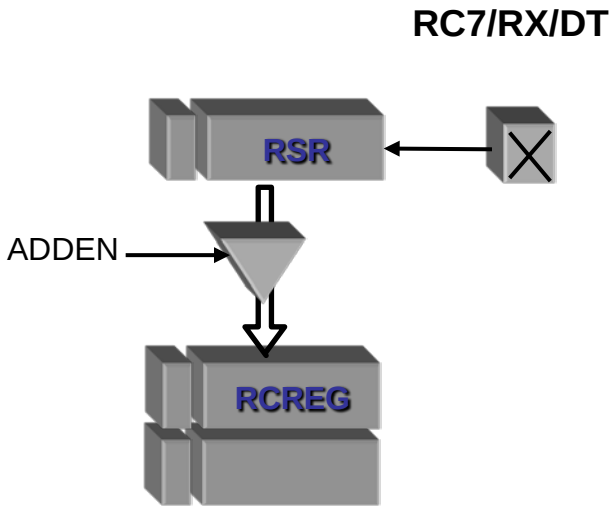
接收端方塊圖

啟動串列介面



RCIF 的動作

- **RSR** 為一個移位器，接收**8-bits** 的串列資料及 **start & stop bit.**
- 接收到的串列資料會被載入到 **RCREG FIFO** 並將設定 **RCIF**
- 假設上一筆資料在**FIFO**中未被拿走此時又有一筆串列資料接收近來，則這筆新的資料會被存在第二級的**FIFO**中
- 若兩級**FIFO**均有資料，接收中斷產生將第一級資料提走，中斷處理完畢後，第二級**FIFO**的資料會立即產生中斷



2 deep FIFO

常用的 USART 函數庫

- **USART的原始程式放在:**
  - ..\mplabc18\v3.43\src\pmc_common\USART
- **OpenUSART : 開啟並設定USART的工作模式**
- **BusyUSART : 測試發送是否忙線中?**
- **putsUSART : 從RAM中提取字串並發送出去**
- **putrsUSART : 從ROM中提取字串並發送出去**
- **ReadUSART : 讀取接收的資料(Byte)**
- **WriteUSART : 傳送一個資料 (Byte)**

使用 USART 的函數庫(一)

- **OpenUSART：開啟並設定USART的工作模式**

  void OpenUSART (unsigned char config , char SPBRG)

  - config中的定義字在 usart.h 檔案中定義
    - USART_TX_INT_OFF ==> PIE1bits.TXIE=0;
    - USART_RX_INT_ON ==> PIE1bits.RCIE=1;
    - USART_ASYNCH_MODE ==> TXASTAbits.SYNC=0;
    - USART_EIGHT_BIT ==> TXSTAbits.TX9=0; RCSTAbits.RX9=0;
    - USART_CONT_RX ==> RCSTAbits.CREN=1;
    - USART_BRGH_HIGH ==> TXSTAbits.BRGH=1;
  - 以下兩個位元也會被設定以啟動USART
    - RCSTAbits.SPEN=1; TXSTAbits.TXEN=1;
  - 中斷優先權控制:
    - RCONbits.IPEN=1; IPR1bits.RCIP=1;
  - SPBRG = 103 : 速率產生器設為 9600 bps
    - **fosc / [(SPBRG+1)*16]= 16MHz / [(103+1)*16] = 9615bps (+0.16%)**

使用 USART 的函數庫(二)

- **char ReadUSART (void): 讀取接收的資料(Byte)**
  - ReadUSART 會自RCREG暫存器中讀取接收資料

    例: Rec_Buff = ReadUSART( );
- **void WriteUSART (char data): 傳送一個資料 (Byte)**
  - 將 data 寫入到TXREG暫存器，並開始傳送資料

    例: WriteUSART('A') ==> 發送 A 字元

    WriteUSART('\n') ==> 發送 0x0A (換行字元)
- **char BusyUSART(void) : 測試發送是否忙線中?**

  例: while (BusyUSART( )); // TXREG busy?

  writeUSART('A') // 發送 A 字元

使用 USART 的函數庫(三)

- **putsUSART : 從RAM中提取字串並發送出去**

  void putsUSART (char *dptr)

  - 傳入該函數的參數必須是一個指向RAM的指標 (Pointer)，即字串或陣列的起始位址
  - 傳送字串直到 0x00 (null character)

- **putrsUSART : 從ROM中提取字串並發送出去**

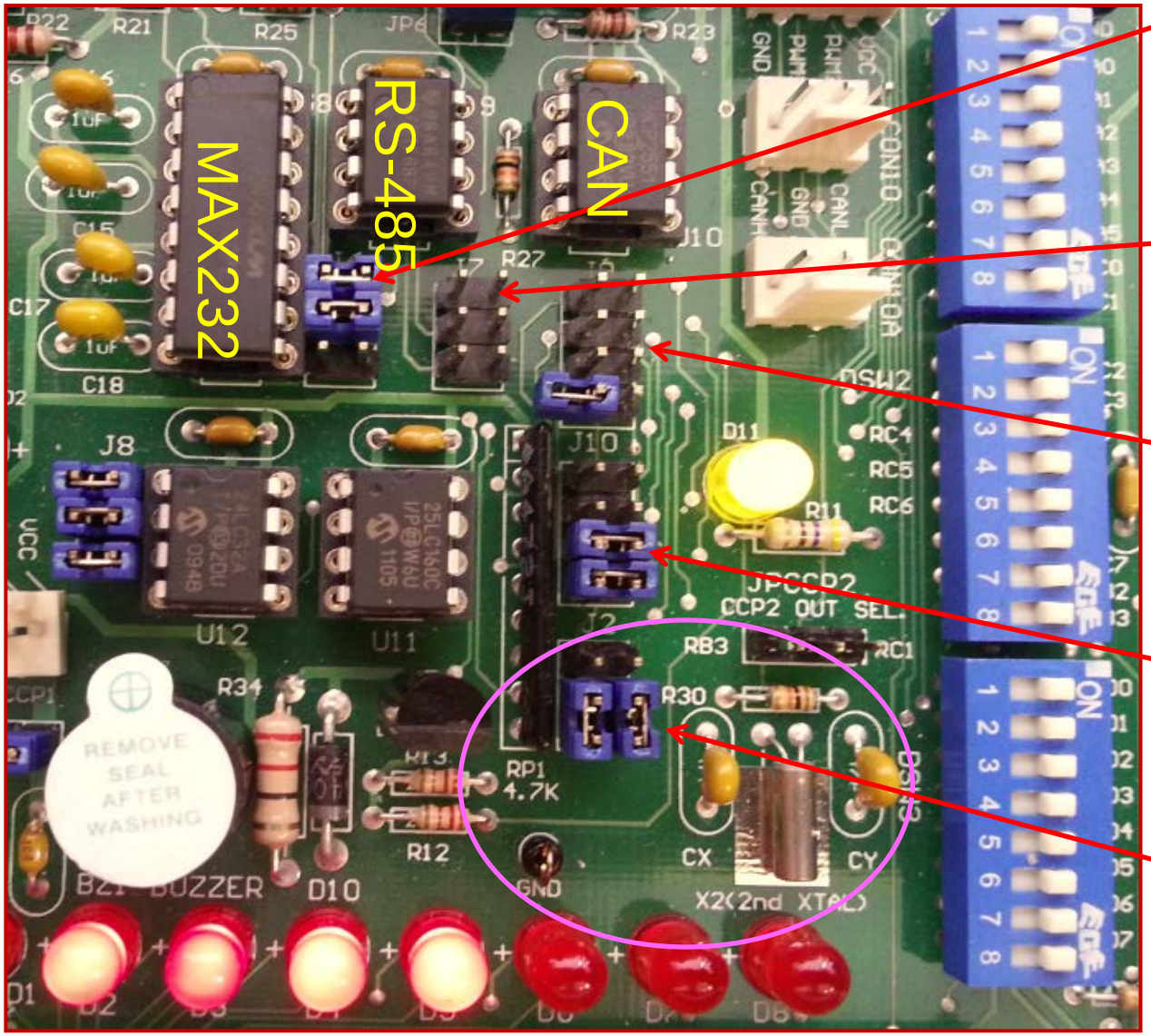
  void putrsUSART (const rom char *dptr)

  - 傳入該函數的參數必須是一個ROM的指標 (Pointer)

例: const rom char Msg0[ ]=“Hello Word!”;

putrsUSART(Msg0);

RS-232 Jump 的設定



JP6 用來連接 MAX232
Pin1 是 TxD (On)
Pin2 是 RxD (On)
Pin3 是 CTS (Off)

JP7 用來連接 RS-485
此練習不連接

JP9 使用在 SPI 介面
在本實驗中需開路

JP10 使用在 I2C 介面
1 & 2 短路接 24LC04
3 & 4 堵路接 TC74A7

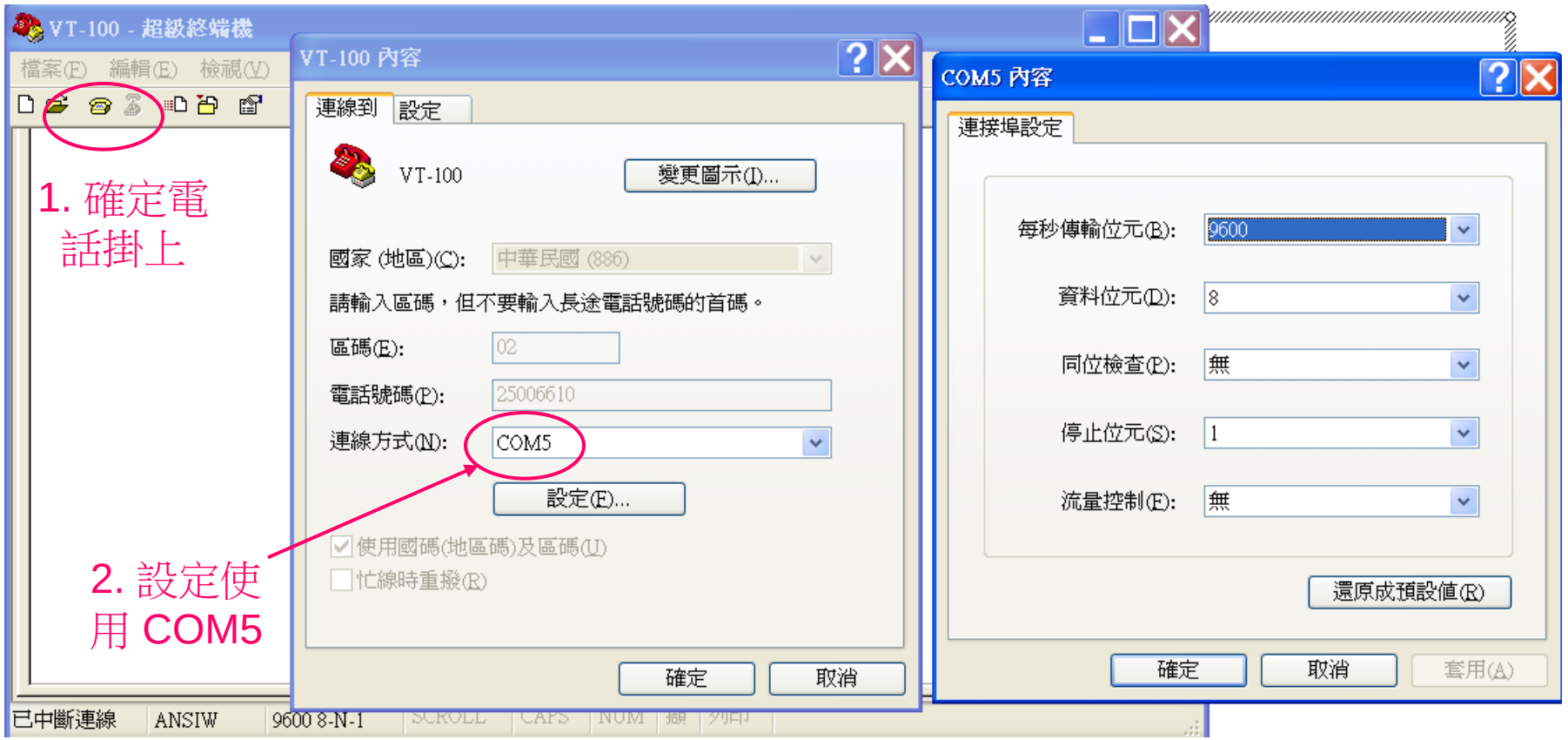
J2 設定
32768Hz
振盪電路

使用 RS-232 介面

- 因現今筆記型電腦已經沒有 **RS-232** 介面
- **USB to RS-232 轉接線/器** 可以使用
- 在裝置管理員下，找出 **USB** 所模擬的 **COM Port** （需安裝驅動程式）
  - 在下圖為模擬 COM5

設定 XP 的超級終端機

- 開啟超級終端機
  - 開始→ 所有程式 → 附屬應用程式 → 通訊
  - 開啟 “超級終端機”

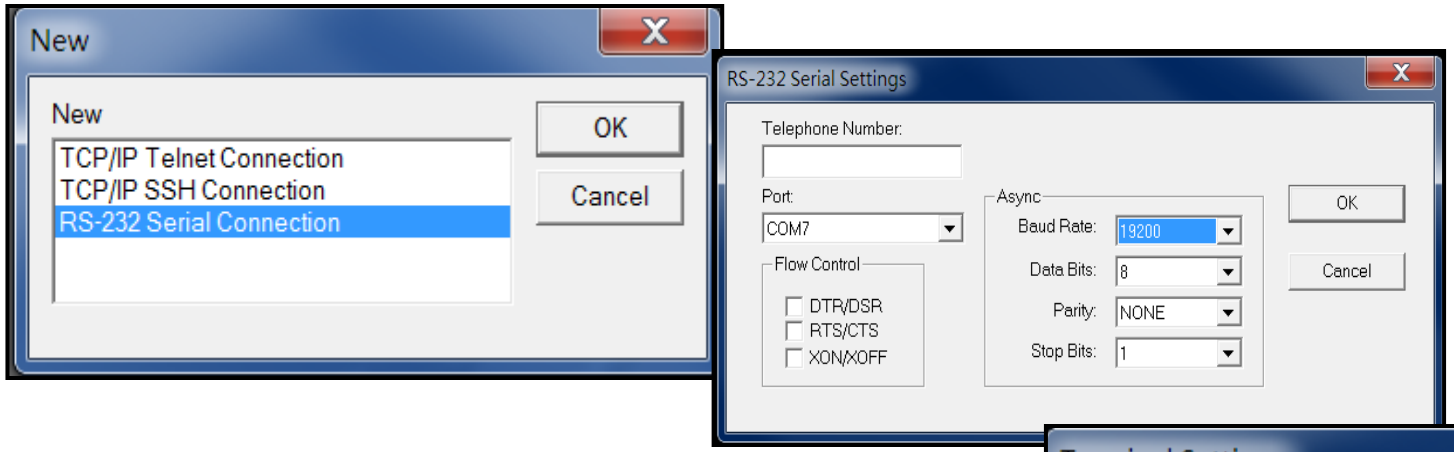


使用終端機模擬程式

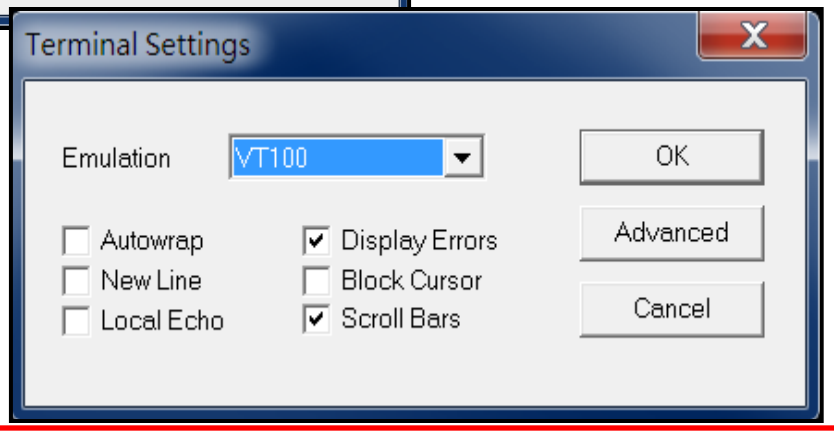
- **Win XP 有　Hyper-Terminal 可以使用**
- **Win 7 & Win 8 沒有 Hyper-Terminal**
  - 使用 Terterm 終端機模擬程式
  - 使用 AlphaCom 終端機模擬程式
  - 拜一下 GooGle 大神

使用 AlphaCom - Serial

- 執行 **Plpha.exe**
  - File → New Session 建立新的通訊協定



選用 USB 模擬的通訊
設定通訊協定:
19200, 8, N, 1
Flow Control : 無



最後一需選用
VT100 的終端
機型式來做通訊
的連線

練習六
非同步串列通訊 (UART)

接收採中斷方式以提高系統的效率

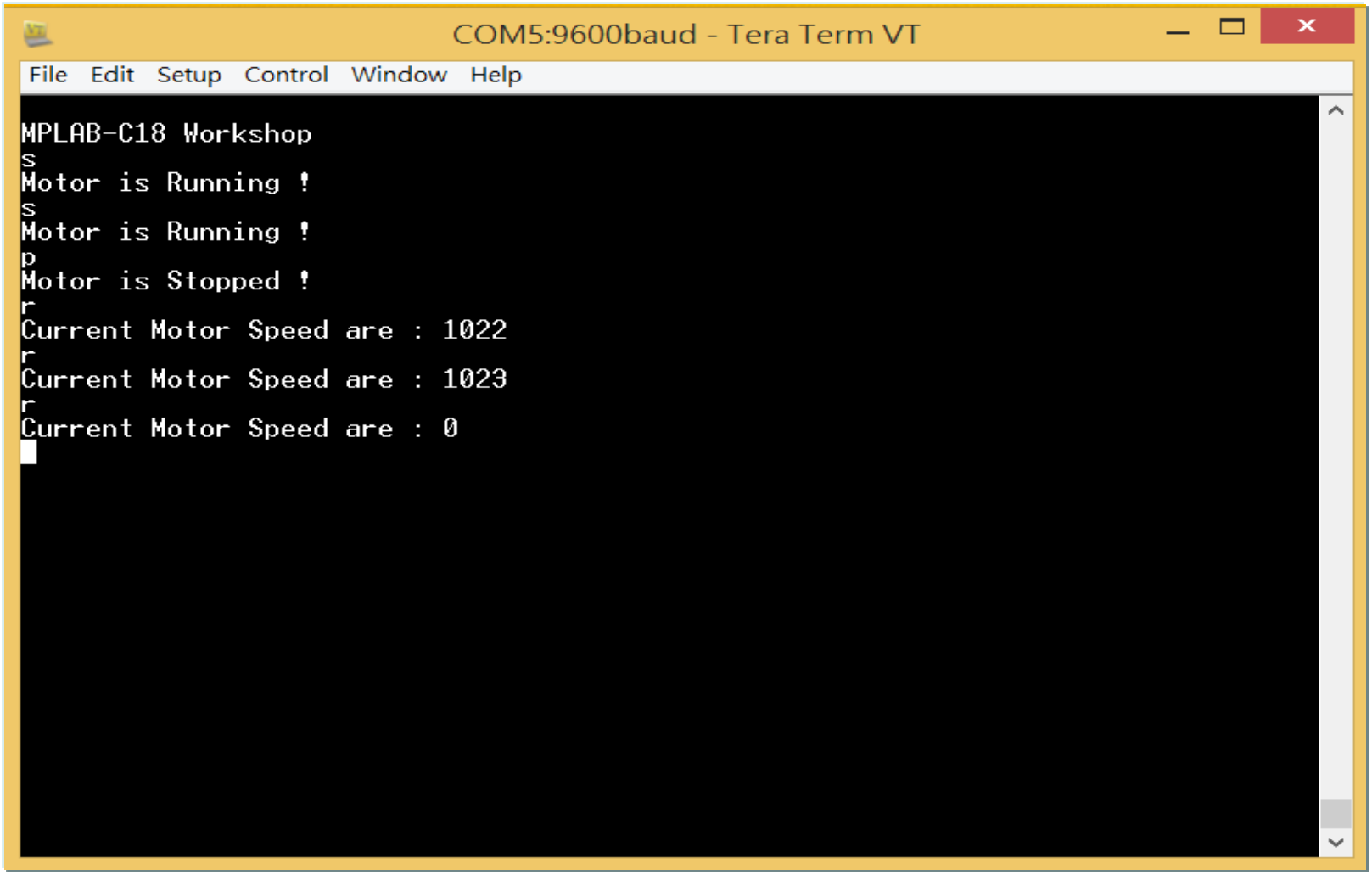
- 開啟” **..\TLS2110\Lab6\UART Comm.mcp**”
- 注意 **CCP2 (PWM)** 的輸出是規劃在 **RC1 (JPCCP2** 上選擇 **)**，**CCP2** 輸出接到一 **LED (D9-CCP2)** 做為 **PWM2** 輸出的 **Duty** 觀察
- 設定**USART**
  - 9600 , N , 8 , 1
  - 中斷設定: Disable TxD，Enable High Priority for RxD
- 送一字串 “ **MPLAB-C18 Workshop”**到終端機顯示
- **A/D** 轉換結果用 **PWM2** 控制 **LED** 亮度
- 終端機按 **“p” (**小寫**) LED** 熄滅，並在終端機顯示字串：

  “ **Motor is Stopped !”**
- 終端機按 **“s” (**小寫**) LED** 亮，並在終端機顯示字串：

  “ **Motor is Running !”**
- 終端機按 **“r” (**小寫**)** 讀取 **PWM** 控制值，並轉成十進制的**ASCII**在終端機顯示：

  “ **Current Motor Speed are : 1023”**

執行 Lab6 終端機畫面



MICROCHIP

Regional Training Centers

MICROCHIP

Regional Training Centers

第一階段
課程結束！