MICROCHIP

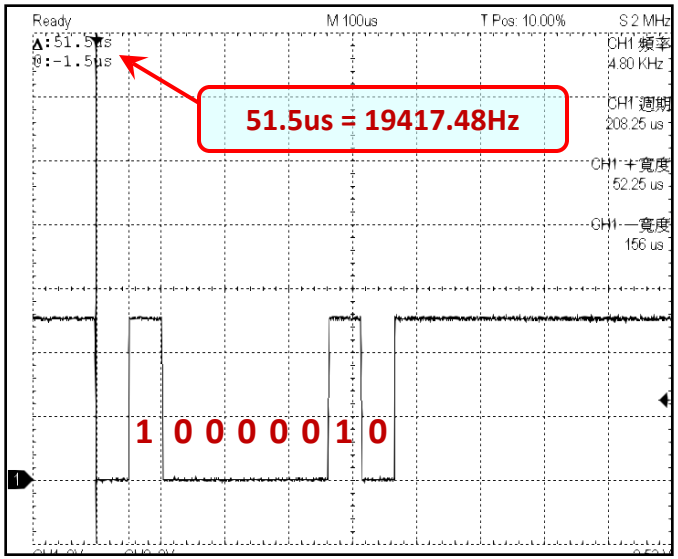
Regional Training Centers

Section 9
UART

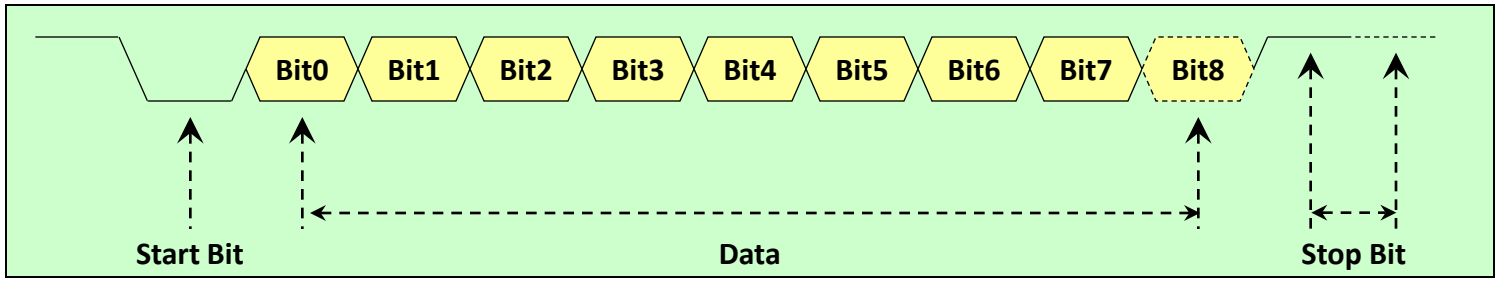
MICROCHIP
Regional Training Centers

What's 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。

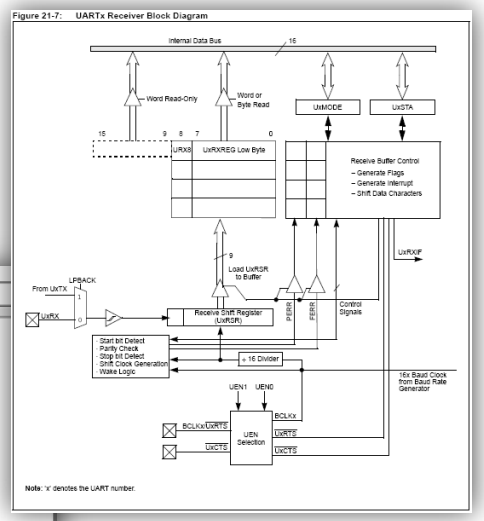


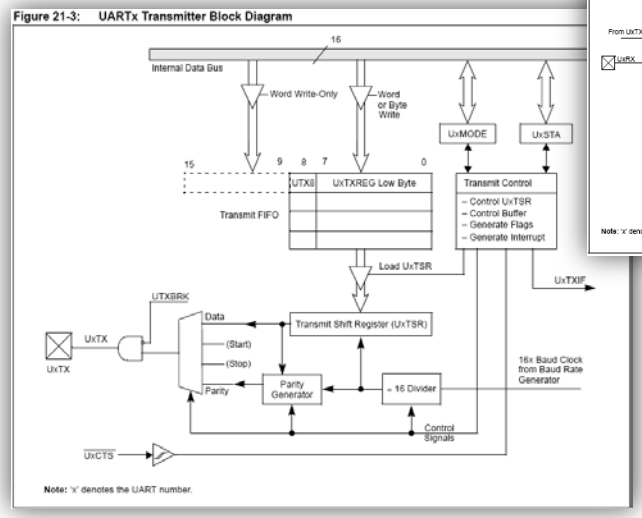
UART傳送範例'A'(0x41), 19200 bps



PIC24 UART Tran./Rece. Block

- PIC24的UART Module支援, 8 or 9-Bits Data, 同位檢查及硬體流量控制(DTS, RTS)。
- 採全雙工, 非同步通訊。支援RS-232, RS-485, IrDA及LIN Bus。
- 四層深度的輸出入緩衝器(Buffer) 。
- 具有錯誤檢查機制
  (Parity Error, Frame Error, Overflow, etc..),
  Loopback及
  Address Detect
  (RS-485), Auto Baud
  Detect(LIN bus)功能。





MICROCHIP
Regional Training Centers

UART's Baud Rate

- PIC24的鮑率(Baud Rate)的計算分為高低兩種速率, 透過UxMODE中BRGH來設定。BRGH=0為低速;反之則為高速。
- 低速率時(BRGH=0), 計算方式如下:

$$UxBRG = \frac{SystemClock}{16 \times IdeaBaulRate} - 1 \qquad ActualBaudRate = \frac{SystemClock}{16 \times (UxBRG + 1)}$$

- 高速率時(BRGH=1),計算方式如下:

$$UxBRG = \frac{SystemClock}{4 \times IdeaBaulRate} - 1 \qquad ActualBaudRate = \frac{SystemClock}{4 \times (UxBRG + 1)}$$

Baud Rate Error

- UART模組鮑率計算誤差?
  假設System Clock為16MHz。預設鮑率115,200 bps。計算鮑率與誤差。
  低速率時(BRGH=0),鮑率與誤差計算:

$$\frac{16MHz}{16\times 115200}-1=7.68\cong 8 \qquad \frac{16MHz}{16\times(8+1)}\cong 111111 \qquad \frac{111111-115200}{115200}\times 100\%\cong -3.55\%$$

  高速率時(BRGH=1),鮑率與誤差計算:

$$\frac{16MHz}{4\times 115200}-1=33.7\cong 34 \qquad \frac{16MHz}{4\times(34+1)}\cong 114286 \qquad \frac{114286-115200}{115200}\times 100\%\cong -0.79\%$$

- UART誤差容忍度約為3%,在範例中,以低速率設定時,產生的誤差過高。此時就算UART的硬體設定無誤,也會因為過高的誤差導致UART動作不正常。

MICROCHIP
Regional Training Centers

MCC's UART Function & Buffer

- MCC中的UART Function已經預先建構了傳送與接收的環狀佇列。主要的核心函數如下:

```
// 啟用UART1, 設定UART1工作模式。
        void UART1_Initialize(void);
// 傳送任務維護函數, 在主迴圈或傳送中斷內執行, 用以維護傳送工作。
        void UART1_TasksTransmit(void);
// 接收任務維護函數,在主迴圈或接收中斷內執行, 用以維護接收工作。
        void UART1_TasksReceive(void);
// 傳送檢查與函數。
        bool UART1_TransmitBufferIsFull(void);
        void UART1_Write(byte);
        unsigned int UART1_WriteBuffer(*buffer, bufLen);
// 接收檢查與函數
        void UART1_ReceiveBufferIsEmpty(void);
        void UART1_Read();
        unsigned int UART1_ReadBuffer(*buffer, bufLen);
```

MICROCHIP
Regional Training Centers

About XC16's String Function

- UART 所處理的資料大多是字串或文字資料, 使用XC16所提供的字串處理函式將能更方便的進行**"資料" <-> "字串"**的轉換。常用的轉換函式有sprintf( ), strlen( )。
- **int sprintf(char *s, const char *format, ...);**
  將資料依據指定格式轉換後的字串存入指定的buffer中。
  *** 必需 include <stdio.h>**
  size_t strlen(const char *s);
  計算字串(string)的長度(不包含字串結尾的的0x00('\0'))。
  *** 必需 include <string.h>**
- 詳細資訊, 可以參考說明文件:
  \XC16\vx.xx\16-Bit_Language_Tools_Libraries_Manual.pdf

MICROCHIP
Regional Training Centers

UART Transmit Example

- **UART Transmit Polling Example1**
  // 將字串陣列的字元, 逐一透過**UART1**傳送出去。
  printf("Now Counter : %4d\r\n", Counter++);

- **UART Transmit Polling Example2**
  // 將字串陣列的字元, 逐一透過UART*x*傳送出去,直到字串結尾。

```
unsigned char UARTString[ 100 ];
sprintf( ( char * ) UARTString , "Now Counter : %4d\r\n", Counter++ );

void UART_PutRAMString(unsigned char * String)
{
        while( *String != 0x00 )
        {
                while(UxSTAbits.UTXBF);
                UxTXREG = *String++;
        }
}
```

UART Transmit Example

-MCC Base-

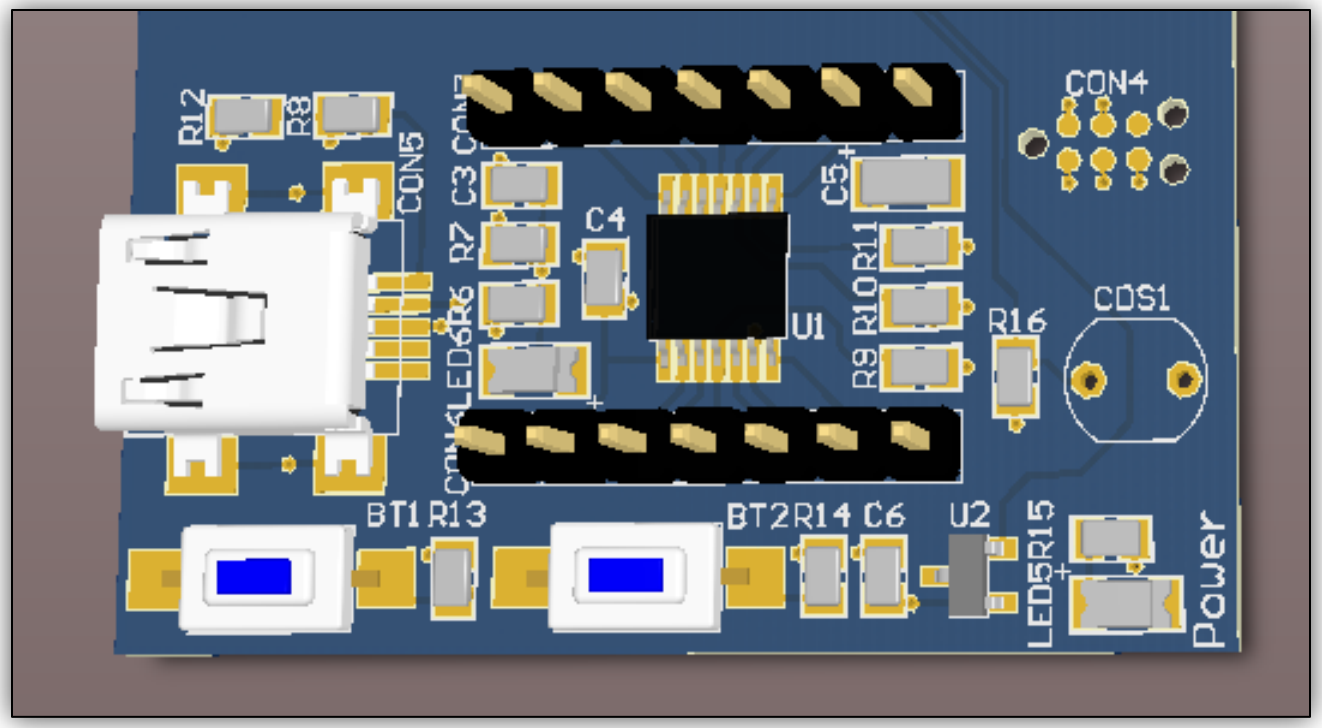
- **UART Transmit Polling Example3**

```
// 使用MCC所提供函數, 將字串陣列的字元, 逐一傳送出去。
unsigned char UARTString[ 100 ];

while(1)
{
   if( !UART1_TransmitBufferIsFull( ) )
   {
      sprintf( ( char * ) UARTString , “Now Counter : %4d\r\n”, Counter++ );
      UART1_WriteBuffer(UARTString, strlen(UARTString));
   }
   UART1_TasksTransmit(void);
}
```

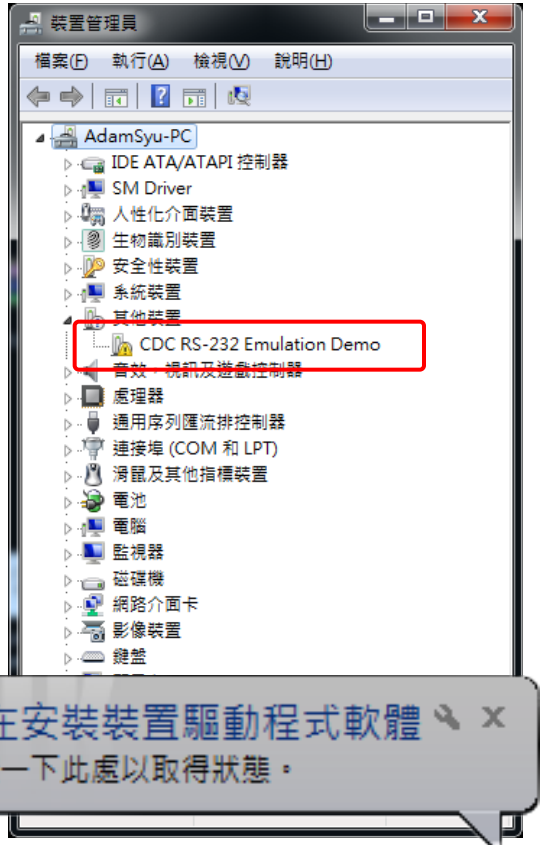
USB CDC Emulator

- PIC16F1455-I/SS內建有USB的Module, 我們利用此MCU實作了一個USB CDC類別的USB to UART Bridge(USB to TTL)。
- LED6是傳送/接收指示燈。指示目前資料傳輸的情形。



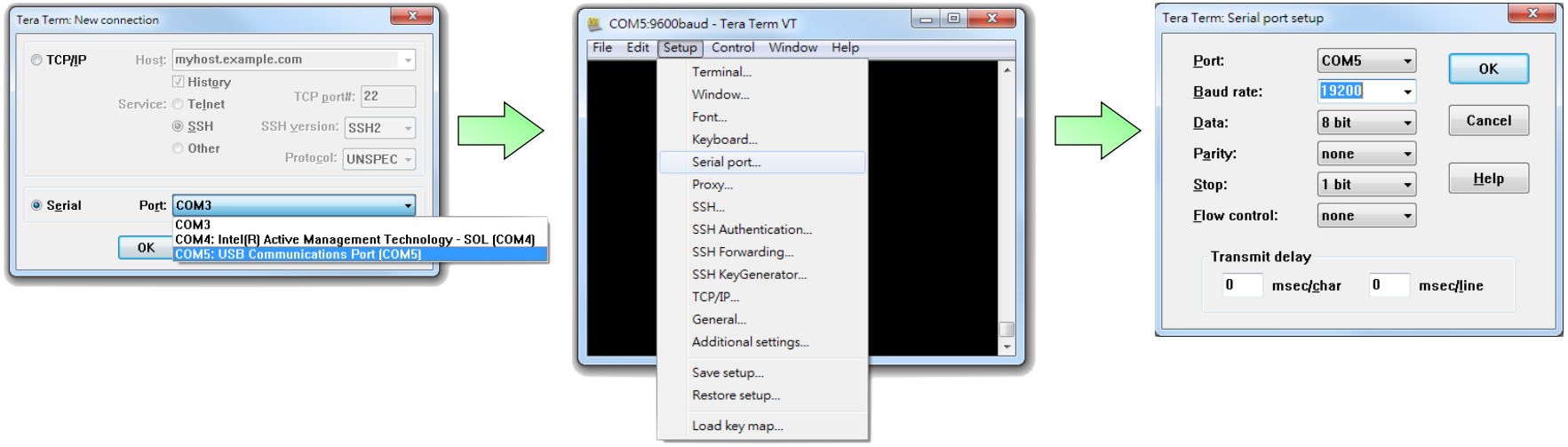
Driver Installation

- PIC16F1455已經預先燒錄USB Serial Emulator的firmware。
- 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, **(\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PIC24的UART1溝通。
- PIC24F16KA1022上UART1的Tx, Rx已預先與PIC16F1455連接完成。



Tera Term

- 由於Windows XP之後的作業系統, 不再內建終端機軟體, 因此必須另行安裝替代軟體, 才能監控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**。



Lab7 UART Tx Polling

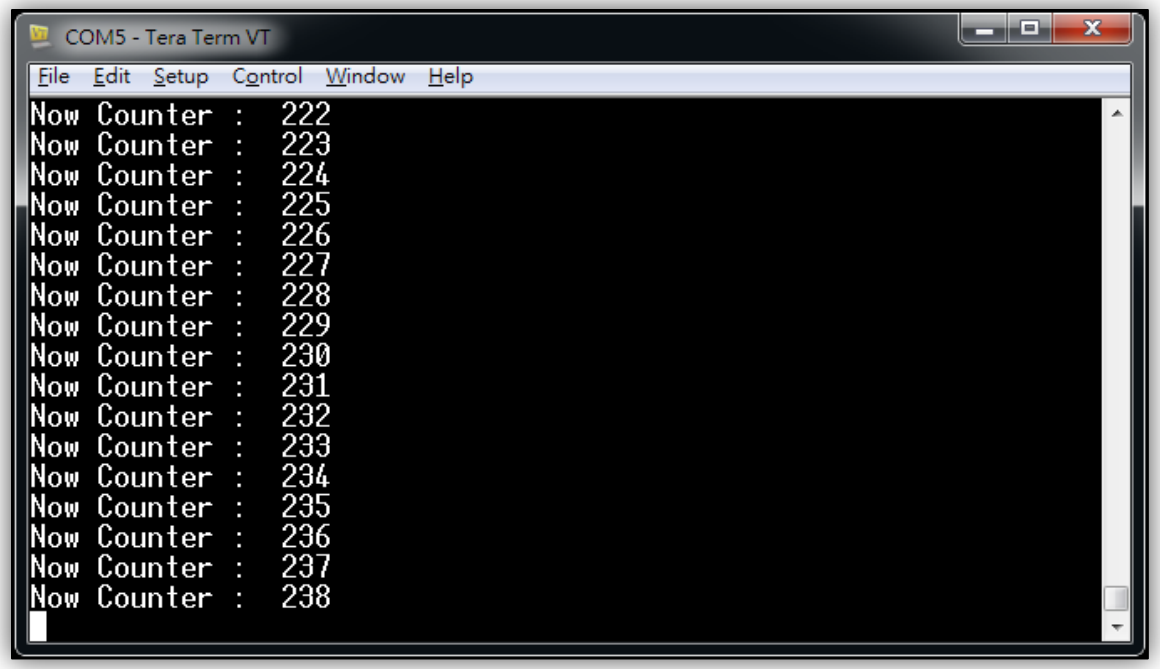
目 標

- 嘗試透過MCC的設定, 在Lab7架構上加入UART1的設定。並透過Timer1定期的將資訊, 透過UART1傳送至PC顯示。
- UART的規格請設定為**9600 , N , 8 , 1, no flow Control**。

- 該如何開始?

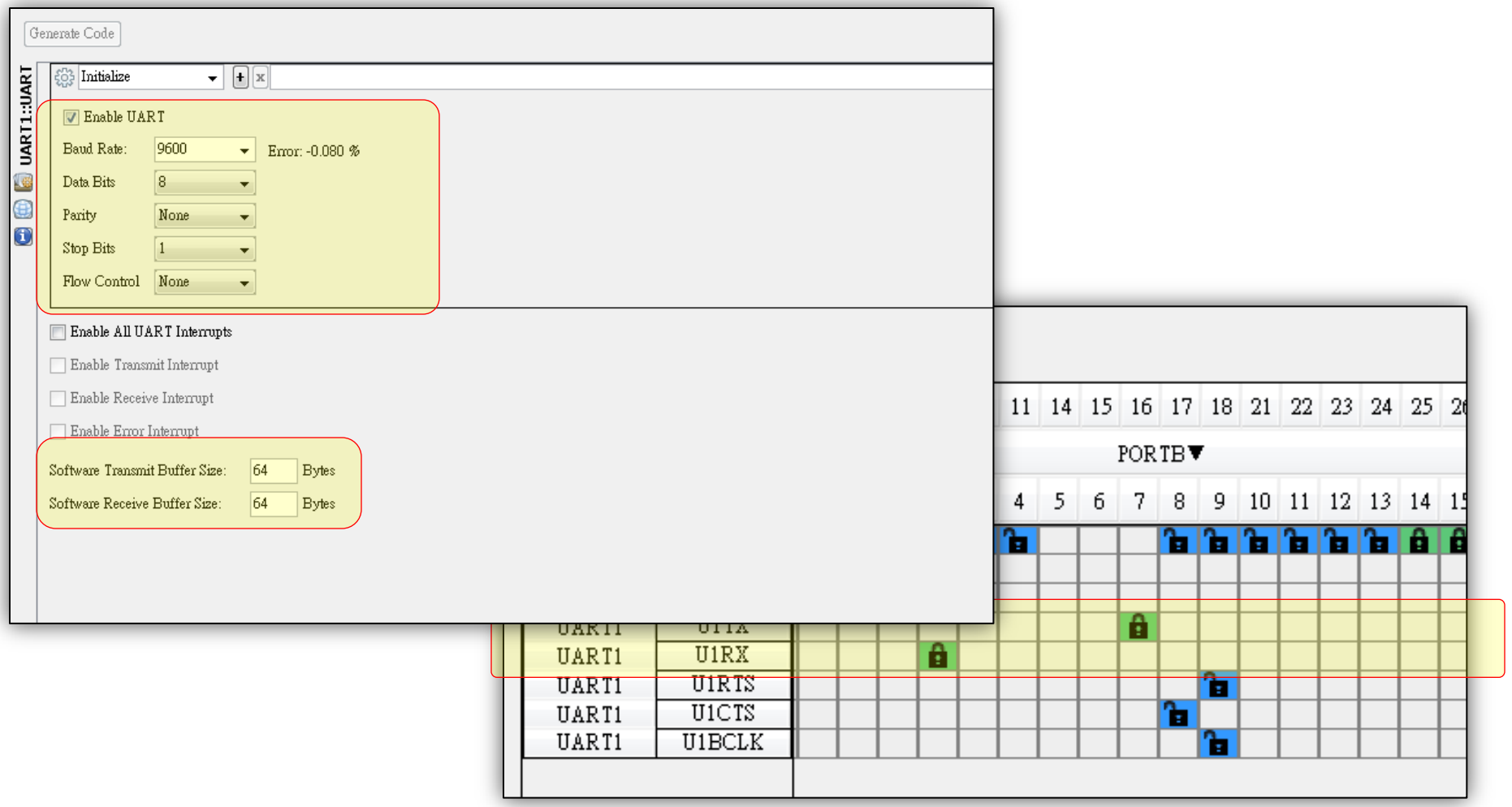
Lab7 UART Tx Polling Result

- 開啟既有專案(C:\Exercises\Exams\Lab7 UART Polling.X)
- 開啟MCC, 新增UART資源,根據題目要求完成設定。
- 期望能在終端機畫面看到資訊的輸出。



Lab7 UART Tx Polling

MCC's Setting



MICROCHIP
Regional Training Centers

Lab7 UART Tx Polling Code Example1

- use ***printf( );*** :

main.c

```
#include <stdio.h>
unsigned int Counter = 0;

int main(void)
{
   while(1)
   {
      ...
      if(T1Flag)
      {
         T1Flag = 0;
         ...
         printf( "Now Counter : %4d\r\n", Counter++);
      }
   }
}
```

MICROCHIP
Regional Training Centers

Lab7 UART Tx Polling Code Example2

- use ***sprintf( );*** & user define string function:

main.c

```
#include <stdio.h>
void UART_PutRAMString(unsigned char * String);
unsigned char UARTString[ 100 ];
unsigned int Counter = 0;

int main(void)
{
   while(1)
   {
      ...
      if(T1Flag)
      {
         T1Flag = 0;
         sprintf( ( char * ) UARTString , "Now Counter : %4d\r\n", Counter++);
         UART_PutRAMString(UARTString);
      }
   }
}
void UART_PutRAMString(unsigned char * String)
{
   while( *String != 0x00 )
   {
      while(U1STAbits.UTXBF);
      U1TXREG = *String++;
   }
}
```

MICROCHIP
Regional Training Centers

Lab7 UART Tx Polling Code Example3

- use ***sprintf( );*** , ***strlen( );*** & MCC function:

main.c

```
#include <stdio.h>
#include <string.h>
unsigned char UARTString[ 100 ];
unsigned int Counter = 0;

int main(void)
{
   while(1)
   {
      if(T1Flag)
      {
         T1Flag = 0;
         if( !UART1_TransmitBufferIsFull( ) )
         {
            sprintf( ( char * ) UARTString , "Now Counter : %4d\r\n", Counter++);
            UART1_WriteBuffer(UARTString, strlen(UARTString));
         }
      }

      UART1_TasksTransmit( ) ;
   }
}
```

MICROCHIP
Regional Training Centers

MCC's UART Interrupt

- MCC中的UART Function開啟中斷後可以更靈活的運用, **UART*x*_TasksTransmit( )** 與 **UART*x*_TasksReceive( )**也會改由中斷服務函式來處理, 不再需要由主程式維護, 使用起來更加便利。
- 一般來說, 通常傳送可以簡單的使用輪詢(Polling)的架構處理, 但接收必須使用中斷(Interrupt)來實現, 才能及時處理, 避免資料遺漏。
- 由於MCC已經幫我們建立了, 傳送與接收的佇列維護功能, 所以開啟UART的傳送與接收終端, 會讓資料收送共容易。

UART Transmit Interrupt Example

-MCC Base-

- **UART Transmit Interrupt Example**

```
// UART1_TasksTransmit( )  maintain form UART Tx ISR

unsigned char UARTString[ 100 ];

while(1)
{
   if(T1Flag)
   {
      T1Flag = 0;
      if( !UART1_TransmitBufferIsFull( ) )
      {
         sprintf( ( char * ) UARTString , “Now Counter : %4d\r\n”, Counter++ );
         UART1_WriteBuffer(UARTString, strlen(UARTString));
      }
   }
   UART1_TasksTransmit(void);
}
```

UART Receive Interrupt Example

-MCC Base-

- **UART Receive Interrupt Example**

```
// UART1_TasksReceive( )  maintain form UART Rx ISR

unsigned char UARTString[ 100 ];

while(1)
{
   if( !UART1_ReceiveBufferIsEmpty( ) )
   {
      sprintf( ( char * ) UARTString , “Rx Char : %c\r\n", UART1_Read( ));
      UART1_WriteBuffer(UARTString, strlen(UARTString));
   }

   UART1_TasksReceive( );
}
```

MICROCHIP
Regional Training Centers

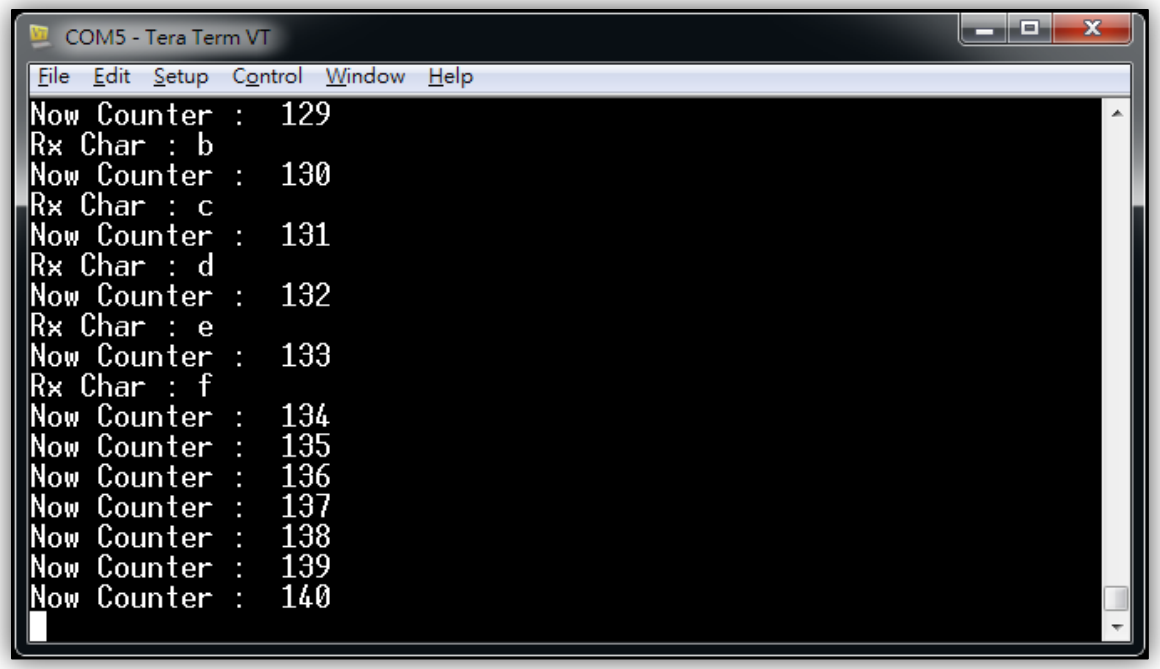
Lab8 UART Tx & Rx Interrupt

目 標

- 嘗試透過MCC的設定, 在Lab8架構上修改UART1的設定。開啟UART1的傳送與接收中斷。中斷優先權修改為”3”,”3”。
- 除了透過UART1將文字傳送至PC顯示外。也可接收PC端所輸入的字元, 再將該字元的對應訊息回傳給UART, 顯示在終端機畫面上。

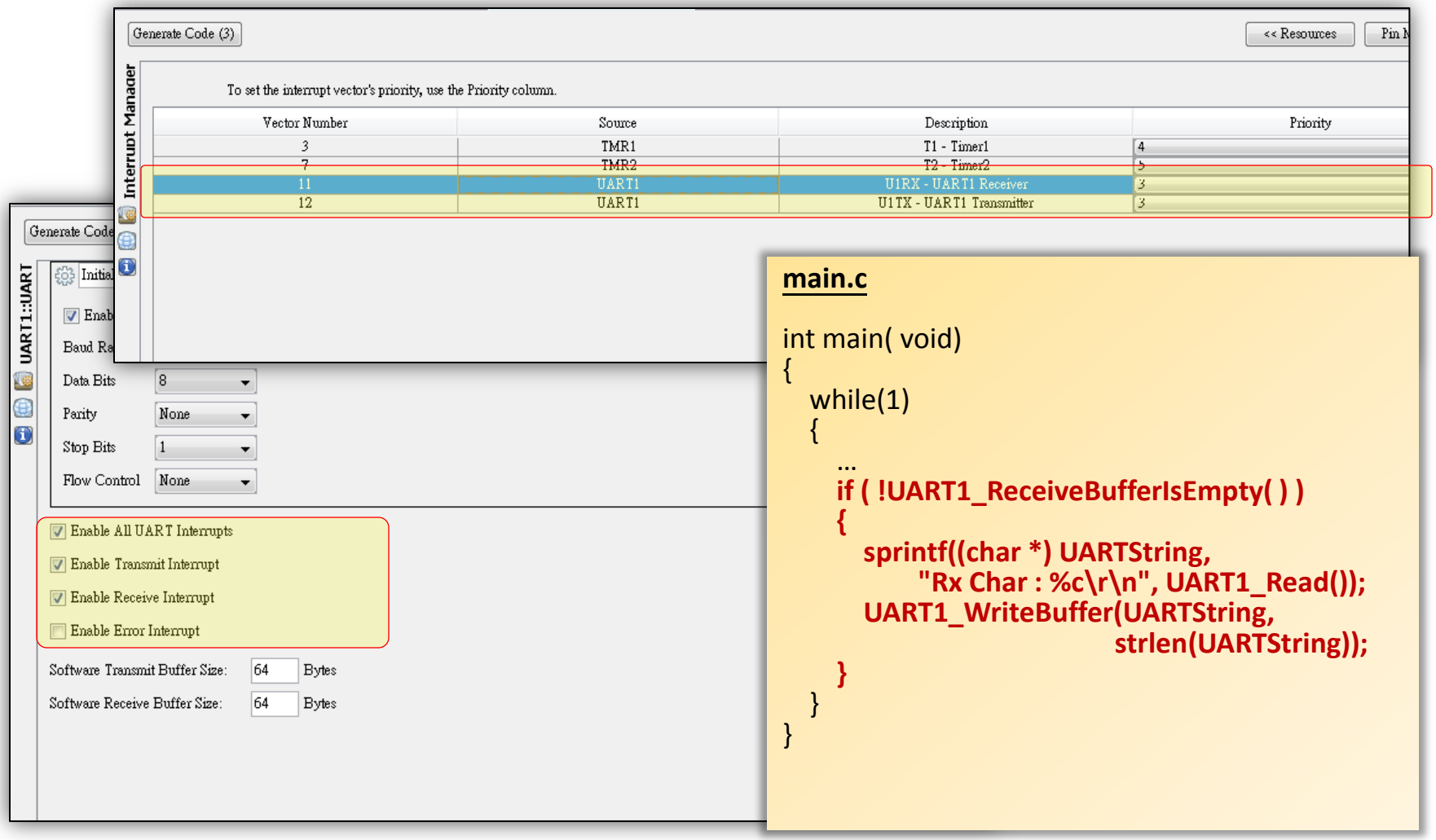
Lab8 UART Tx & Rx Interrupt Result

- 開啟既有專案(C:\Exercises\Exams\Lab8 UART Tx and Rx Interrupt.X)
- 開啟MCC, 修改UART資源,根據題目要求完成設定。
- 期望能在終端機畫面看到輸入後回送的資訊。



Lab8 UART Tx & Rx Interrupt

MCC's Setting & Code Example

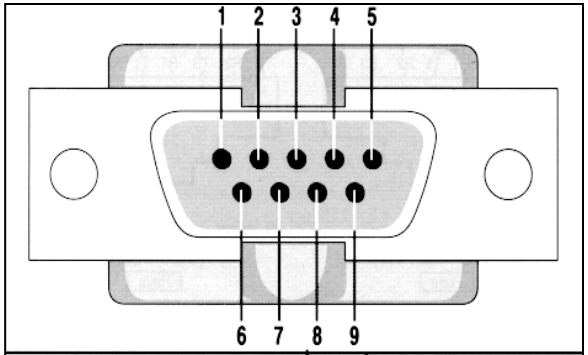


main.c

```
int main( void)
{
   while(1)
   {
      ...
      if ( !UART1_ReceiveBufferIsEmpty( ) )
      {
         sprintf((char *) UARTString,
               "Rx Char : %c\r\n", UART1_Read());
         UART1_WriteBuffer(UARTString,
                           strlen(UARTString));
      }
   }
}
```

UART & MAX232 Connection

- 也可以使用MAX232來與PC的COM Port連接。
- **PC上的RS232所使用邏輯系統與MCU不同。MCU使用正邏輯系統。其邏輯"1"的電壓準位為0.8 V_{DD} ~ V_{DD};邏輯" 0"則為V_{SS} ~ 0.2V_{DD}。RS232,使用負邏輯,邏輯"1"的電壓準位為-3~-12V;邏輯"0"則為+3 ~ +12V。**
- MCU與RS232因為邏輯系統與電壓位準均不同,因此無法直接連接,必須透過MAX232作電壓位準的調整。



Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		

http://www.aggsoft.com/rs232-pinout-cable/serial-cable-connections.htm