



MICROCHIP

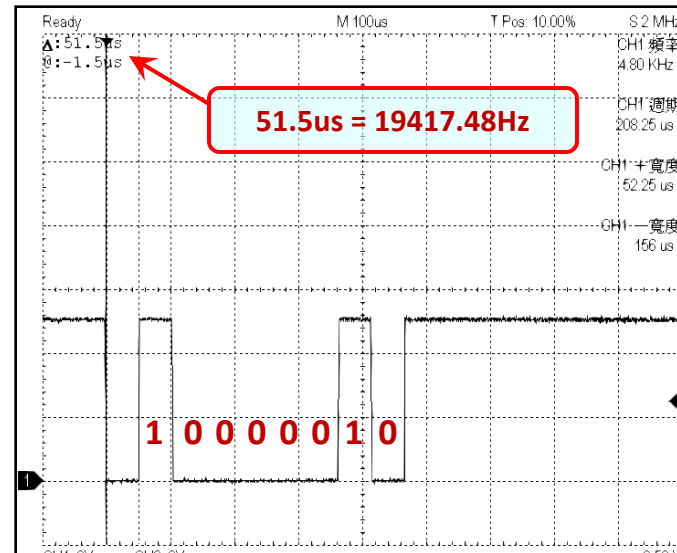
Regional Training Centers

Section 9

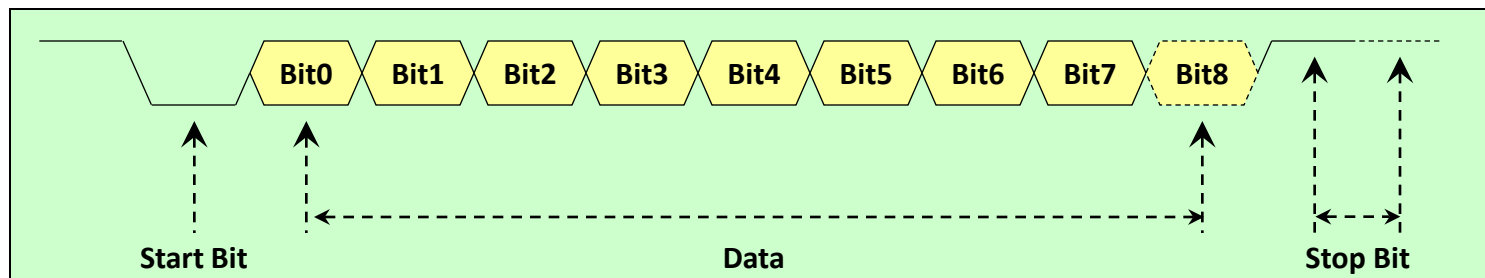
UART

What's UART

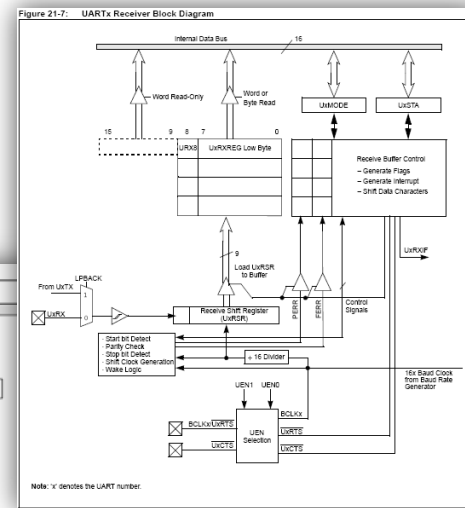
- **UART: Universal Asynchronous Receiver Transmitter,**
泛用非同步接收傳送模組。
一個以串列方式進行資料交換與溝通的通訊模組。
- UART 傳輸的資料由LSB先傳。
格式包含一個起始位元, 八或九位元的資料, 一至二個停止位元。



UART傳送範例'A'(0x41), 19200 bps



- Figure 21-3: UARTx Transmitter Block Diagram**
-
- The diagram illustrates the UARTx Transmitter architecture. Key components and their interconnections include:
- Internal Data Bus (16-bit):** Provides data to the **Transmit FIFO** (bits 9-15) and the **UxTXREG Low Byte** (bits 7-8).
 - Transmit FIFO:** A 7-bit buffer (bits 9-15) that feeds into the **UxTXREG Low Byte**.
 - UxTXREG Low Byte:** An 8-bit register (bits 7-8) that feeds into the **Transmit Shift Register (UxTSR)**.
 - Transmit Shift Register (UxTSR):** An 8-bit register that feeds into the **Parity Generator** and the **UxTX pin**.
 - Parity Generator:** Generates parity for the data in the UxTSR, controlled by the **Parity** bit in the **Transmit Control** register.
 - Transmit Control:** A control register that manages the transmission process, including **Control UxTSR**, **Control Buffer**, **Generate Flags**, and **Generate Interrupt**. It is connected to **UxMODE** and **UxSTA**.
 - UxMODE and UxSTA:** Mode and status registers that interface with the **Transmit Control** register.
 - 16x Baud Clock from Baud Rate Generator:** Provides a clock signal to the **Parity Generator** and the **UxTX pin**.
 - 16-bit Divider:** Divides the baud clock by 16, controlled by **Control Signals**.
 - UxTX pin:** The output pin for the transmitter, which is also connected to the **UxTX pin** through a buffer.
- Note:** 'x' denotes the UART number.



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:

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

- 高速率時(BRGH=1), 計算方式如下:

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

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:

$$\frac{16\text{MHz}}{16 \times 115200} - 1 = 7.\underline{68} \cong 8 \quad \left| \quad \frac{16\text{MHz}}{16 \times (8+1)} \cong 111111 \quad \left| \quad \frac{111111 - 115200}{115200} \times 100\% \cong \boxed{-3.55\%}$$

高速率時(BRGH=1),鮑率與誤差計算:

$$\frac{16\text{MHz}}{4 \times 115200} - 1 = 33.\underline{7} \cong 34 \quad \left| \quad \frac{16\text{MHz}}{4 \times (34+1)} \cong 114286 \quad \left| \quad \frac{114286 - 115200}{115200} \times 100\% \cong -0.79\%$$

- 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。

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);

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printf(), 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

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**UARTx**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**

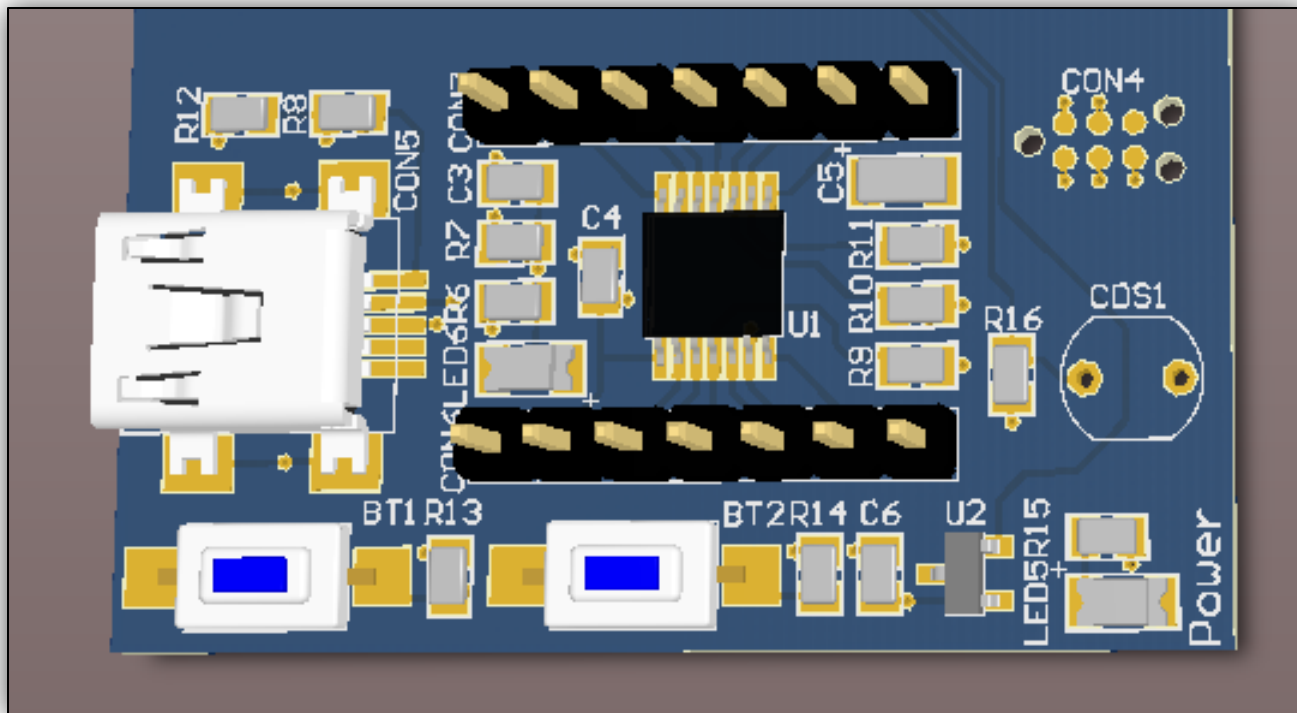
// 使用MCC所提供函數, 將字串陣列的字元, 逐一傳送出去。

```
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);
}
```

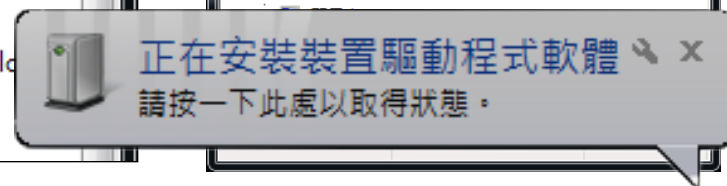
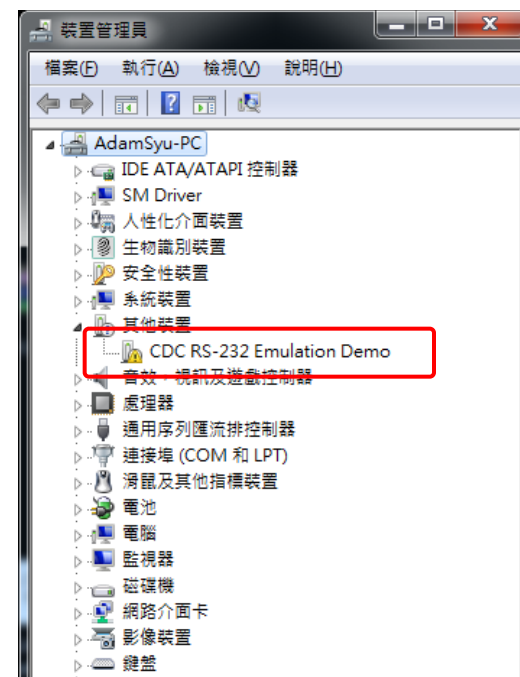
USB CDC Emulator

- 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。



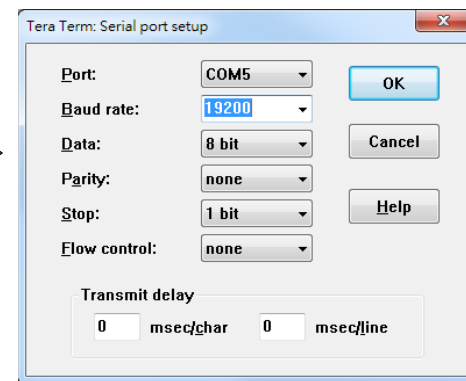
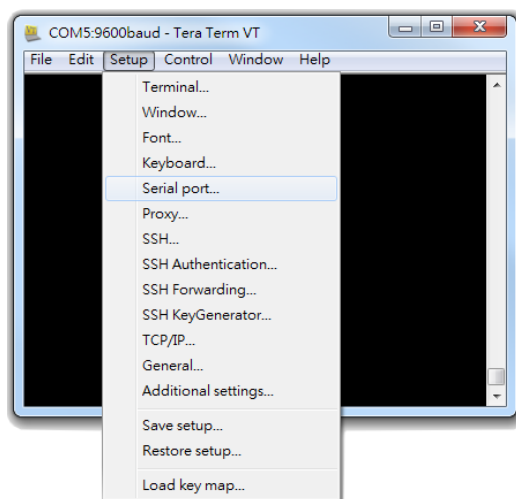
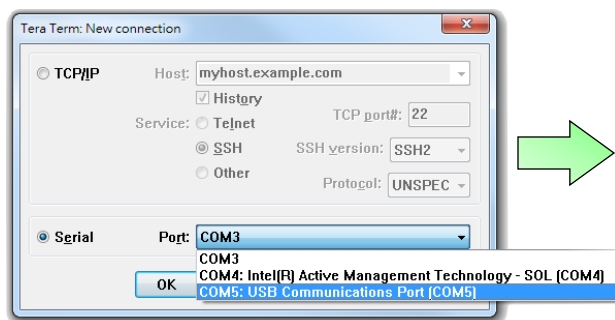
Driver Installation

- 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**。



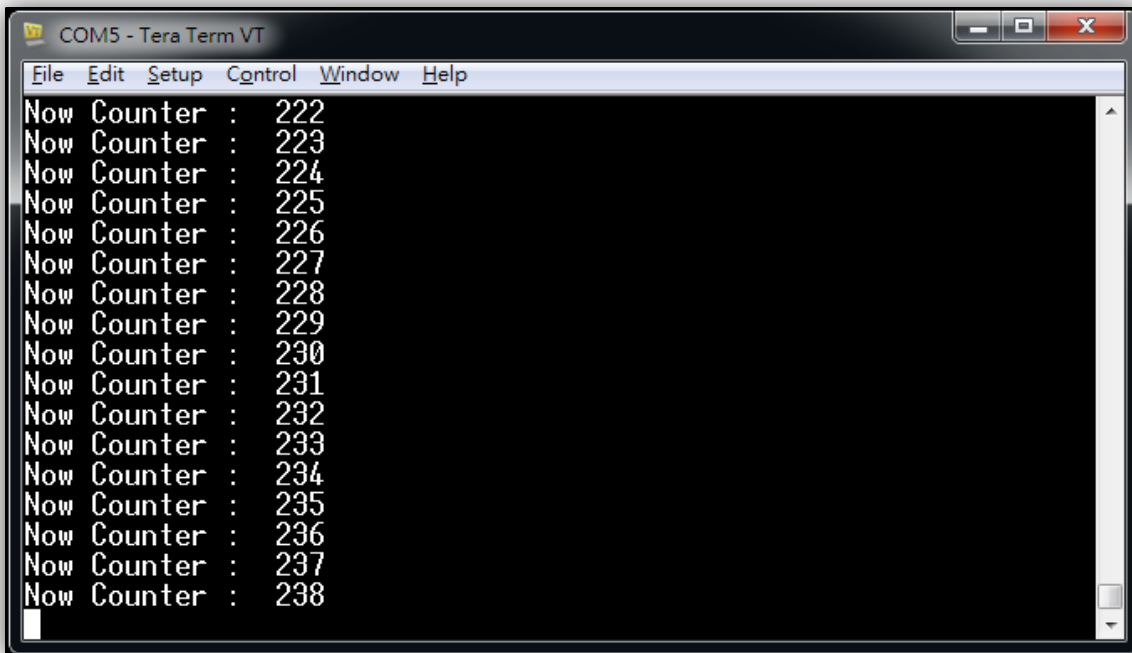
Lab7 UART Tx Polling

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。



COM5 - Tera Term VT

```
File Edit Setup Control Window Help
Now Counter : 222
Now Counter : 223
Now Counter : 224
Now Counter : 225
Now Counter : 226
Now Counter : 227
Now Counter : 228
Now Counter : 229
Now Counter : 230
Now Counter : 231
Now Counter : 232
Now Counter : 233
Now Counter : 234
Now Counter : 235
Now Counter : 236
Now Counter : 237
Now Counter : 238
```



Lab7 UART Tx Polling

MCC's Setting

The screenshot shows the Proteus IDE interface. On the left, the 'UART1::UART' component is selected. The 'Initialize' window is open, displaying the following settings:

- ☒ Enable UART
- Baud Rate: 9600 (Error: -0.080 %)
- Data Bits: 8
- Parity: None
- Stop Bits: 1
- Flow Control: None
- ☐ Enable All UART Interrupts
- ☐ Enable Transmit Interrupt
- ☐ Enable Receive Interrupt
- ☐ Enable Error Interrupt
- Software Transmit Buffer Size: 64 Bytes
- Software Receive Buffer Size: 64 Bytes

On the right, the I/O map is visible, showing the connection between the microcontroller and the UART1 module. The UART1 module is connected to the microcontroller's UART1 pins (U1RX, U1RTS, U1CTS, U1BCLK).

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++);
        }
    }
}
```


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++;
    }
}
```

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( ) ;
    }
}
```

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( );  
}
```

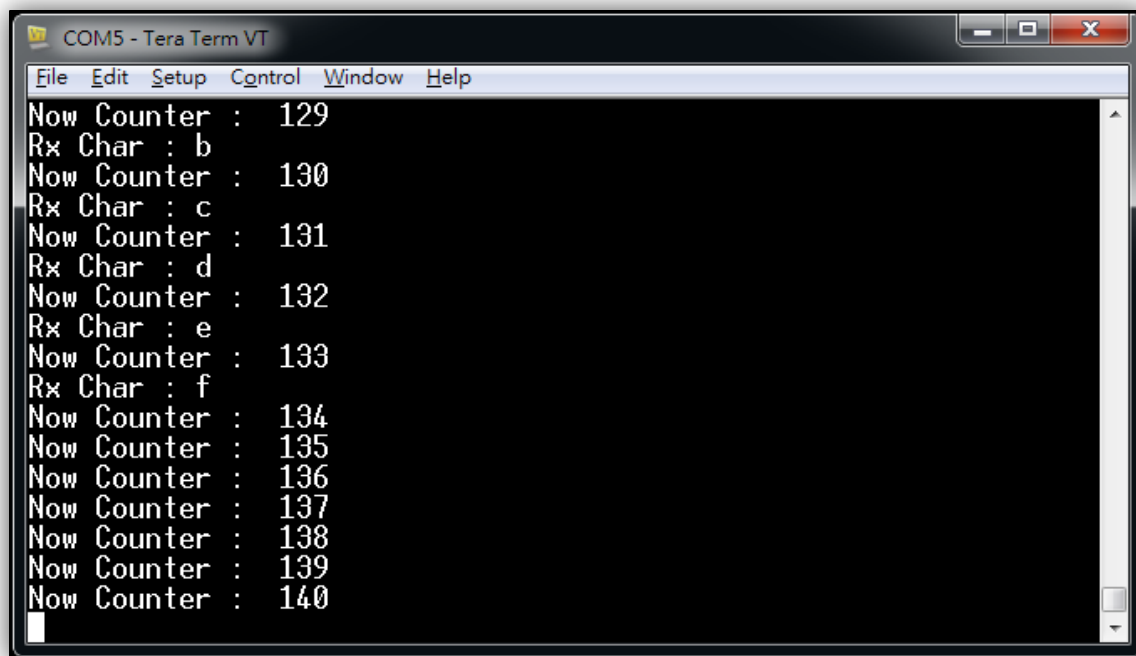
Lab8 UART Tx & Rx Interrupt

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。



COM5 - Tera Term VT

```
File Edit Setup Control Window Help
Now Counter : 129
Rx Char : b
Now Counter : 130
Rx Char : c
Now Counter : 131
Rx Char : d
Now Counter : 132
Rx Char : e
Now Counter : 133
Rx Char : f
Now Counter : 134
Now Counter : 135
Now Counter : 136
Now Counter : 137
Now Counter : 138
Now Counter : 139
Now Counter : 140
```

Lab8 UART Tx & Rx Interrupt

MCC's Setting & Code Example

Generate Code (3) << Resources Pin M

To set the interrupt vector's priority, use the Priority column.

Vector Number	Source	Description	Priority
3	TMR1	T1 - Timer1	4
7	TMR2	T2 - Timer2	5
11	UART1	UIRX - UART1 Receiver	3
12	UART1	UITX - UART1 Transmitter	3

Generate Code

UART1:UART

Initial

☒ Enable

Baud Rate

Data Bits: 8

Parity: None

Stop Bits: 1

Flow Control: None

☒ Enable All UART Interrupts

☒ Enable Transmit Interrupt

☒ Enable Receive Interrupt

☐ Enable Error Interrupt

Software Transmit Buffer Size: 64 Bytes

Software Receive Buffer Size: 64 Bytes

main.c

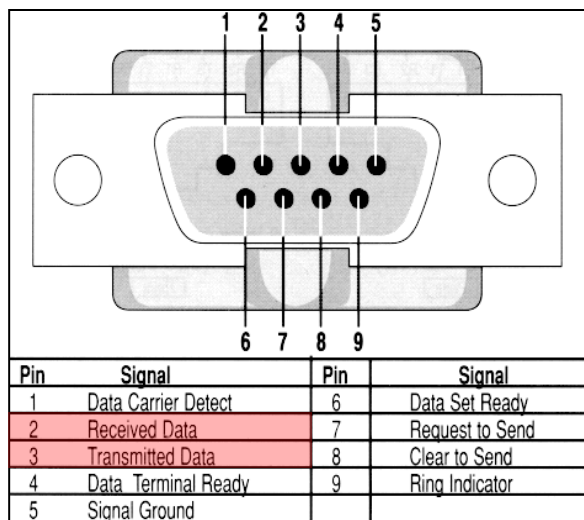
```

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 $0.8 V_{DD} \sim V_{DD}$;邏輯"0"則為 $V_{SS} \sim 0.2V_{DD}$ 。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。



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

