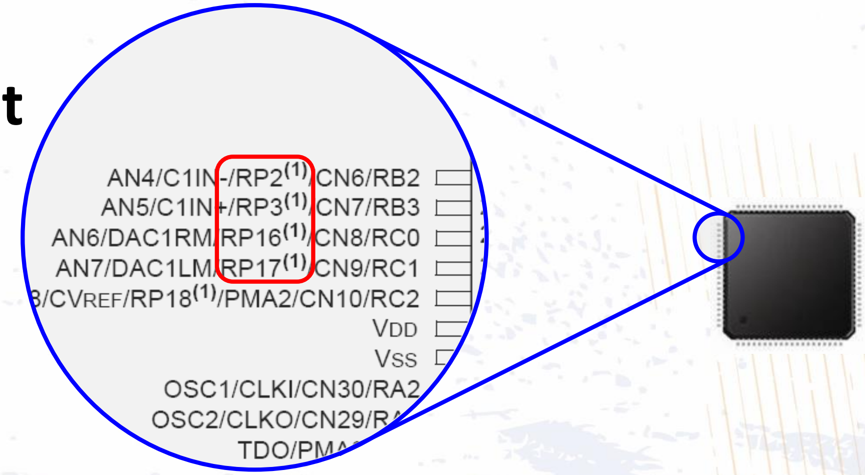
MICROCHIP

Regional Training Centers

Section 10
PPS and UART

What’s PPS

- **PPS, Peripheral Pin Select**
- **Provides a flexible peripheral pins remapping mechanism users can better tailor the microcontroller to their entire application.**
- **RPn : Input/Output**
  RPIn Input Only
- **The peripherals managed by the peripheral pin select are all digital only peripherals.**



Available Peripherals

- External Interrupts
- Timer External Clocks
- Input Captures
- Output Compares
- PWM Fault Input pins
- SPI,UART Functions
- QEI Inputs
- Data Converter Interface
- CAN

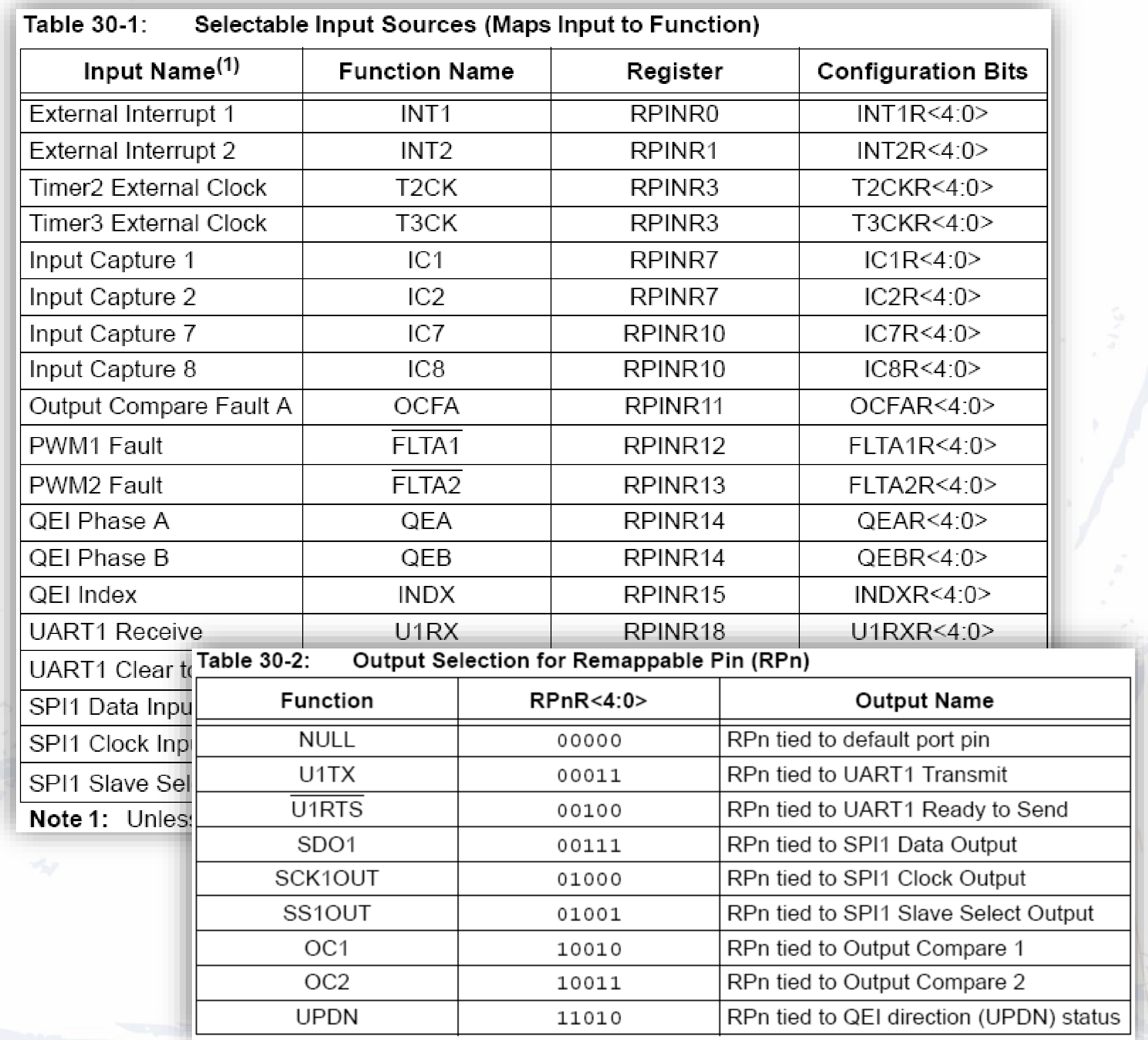


Table 30-1: Selectable Input Sources (Maps Input to Function)

Input Name[1]	Function Name	Register	Configuration Bits
External Interrupt 1	INT1	RPINR0	INT1R<4:0>
External Interrupt 2	INT2	RPINR1	INT2R<4:0>
Timer2 External Clock	T2CK	RPINR3	T2CKR<4:0>
Timer3 External Clock	T3CK	RPINR3	T3CKR<4:0>
Input Capture 1	IC1	RPINR7	IC1R<4:0>
Input Capture 2	IC2	RPINR7	IC2R<4:0>
Input Capture 7	IC7	RPINR10	IC7R<4:0>
Input Capture 8	IC8	RPINR10	IC8R<4:0>
Output Compare Fault A	OCFA	RPINR11	OCFAR<4:0>
PWM1 Fault	$\overline{\text{FLTA1}}$	RPINR12	FLTA1R<4:0>
PWM2 Fault	$\overline{\text{FLTA2}}$	RPINR13	FLTA2R<4:0>
QEI Phase A	QEA	RPINR14	QEAR<4:0>
QEI Phase B	QEB	RPINR14	QEBR<4:0>
QEI Index	INDX	RPINR15	INDXR<4:0>
UART1 Receive	U1RX	RPINR18	U1RXR<4:0>
UART1 Clear t			
SPI1 Data Inpu			
SPI1 Clock Inp			
SPI1 Slave Sel			

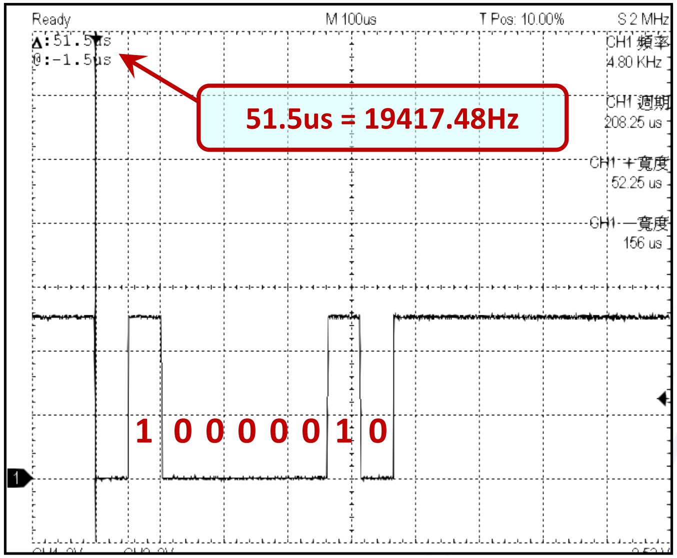
Note 1: Unles

Table 30-2: Output Selection for Remappable Pin (RPn)

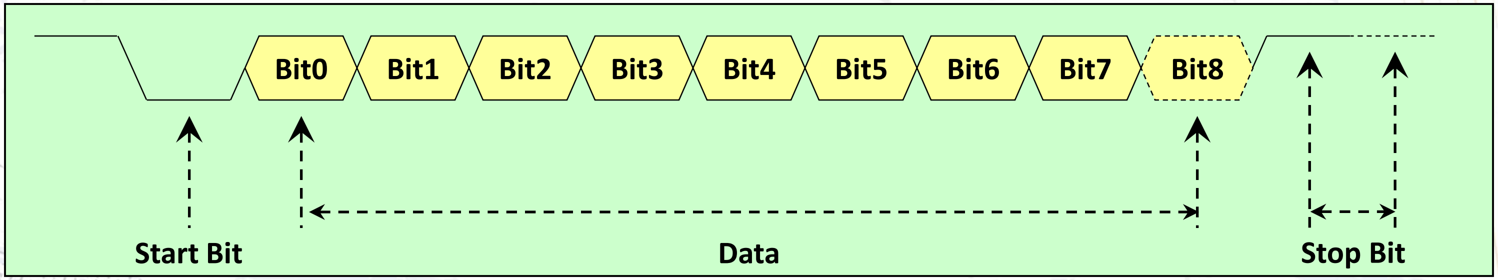
Function	RPnR<4:0>	Output Name
NULL	00000	RPn tied to default port pin
U1TX	00011	RPn tied to UART1 Transmit
$\overline{\text{U1RTS}}$	00100	RPn tied to UART1 Ready to Send
SDO1	00111	RPn tied to SPI1 Data Output
SCK1OUT	01000	RPn tied to SPI1 Clock Output
SS1OUT	01001	RPn tied to SPI1 Slave Select Output
OC1	10010	RPn tied to Output Compare 1
OC2	10011	RPn tied to Output Compare 2
UPDN	11010	RPn tied to QEI direction (UPDN) status

What’s UART

- **UART : Universal Asynchronous Receiver Transmitter.**
- **The module data exchange through serial communication.**
- **The data formatting include 1 start bit 5 to 9 data bits and 1 or 2 stop bits.**



UART Transmit Example : ’A’(0x41), 19200 bps



PIC24 UART

- UART module is full-duplex, asynchronous communication., Support RS-232, RS-485, IrDA and LIN 1.2.
- The module also support hardware flow control, Parity check 8 or 9 Data Bits.
- 4-Deep FIFO for Tx and Rx.
- Error check for
  Parity Error
  Frame Error
  Overflow, etc..).
- Provide Loopback,
  Address Detect(RS-485),
  Auto Baud Detect(LIN bus)

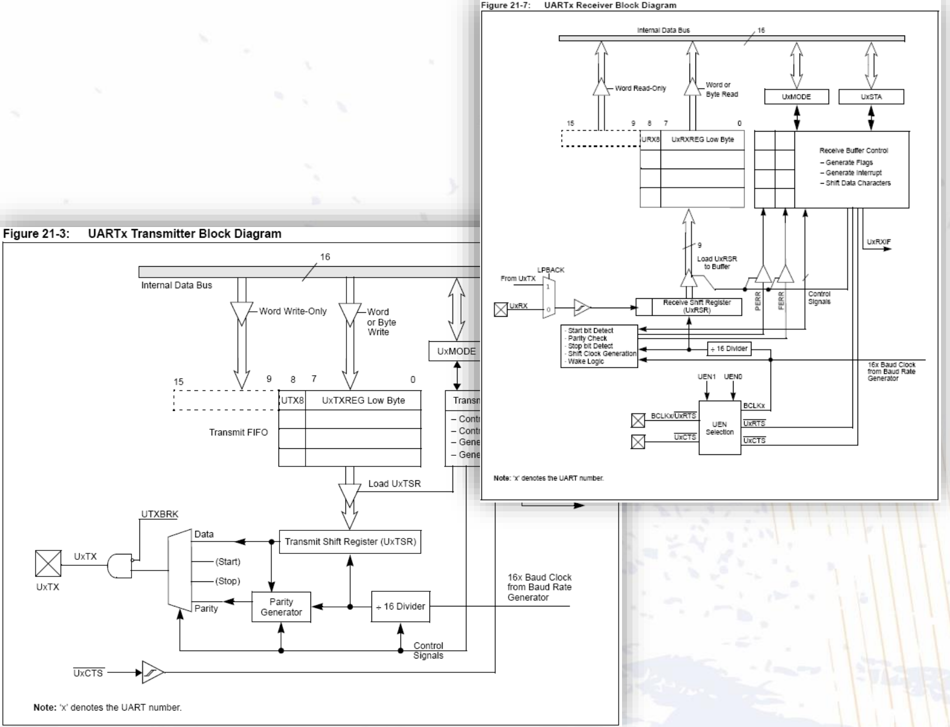


Figure 21-3: UARTx Transmitter Block Diagram

Note: 'x' denotes the UART number.

Figure 21-7: UARTx Receiver Block Diagram

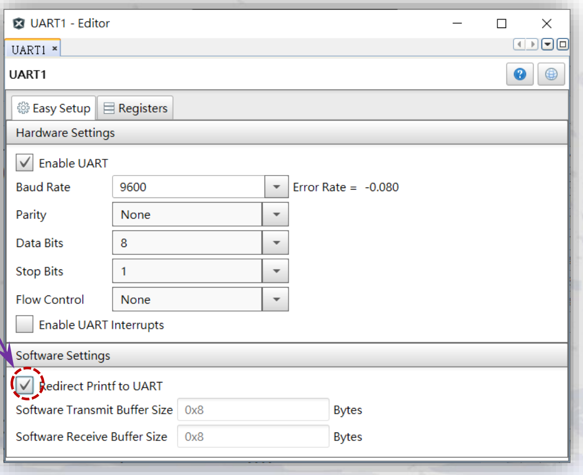
Note: 'x' denotes the UART number.

MCC’s UART Redirect

- MCC provide STDOUT redirect function, user can redirect STDOUT to UARTn.
- You can use printf() to export data by UARTn, if redirect successfully.
- For Example :

```
//UART Transmit Example
printf( “Now Counter : %4d\r\n”, Counter++ );
```

Redirect Printf to UART



Hardware Settings

☑ Enable UART

Baud Rate	9600	Error Rate = -0.080
Parity	None	
Data Bits	8	
Stop Bits	1	
Flow Control	None	

☐ Enable UART Interrupts

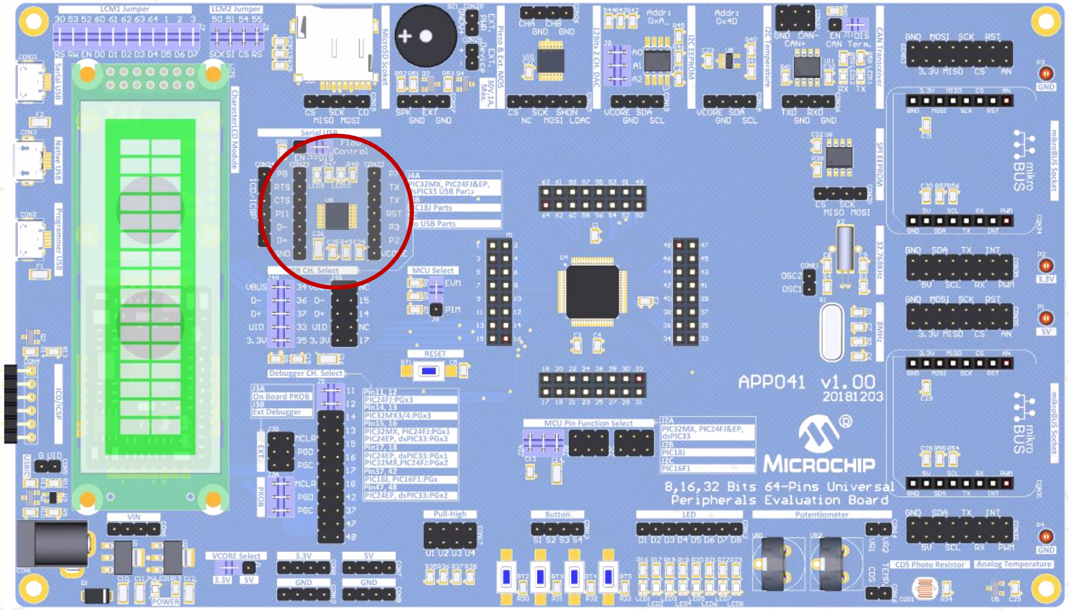
Software Settings

☑ Redirect Printf to UART

Software Transmit Buffer Size	0x8	Bytes
Software Receive Buffer Size	0x8	Bytes

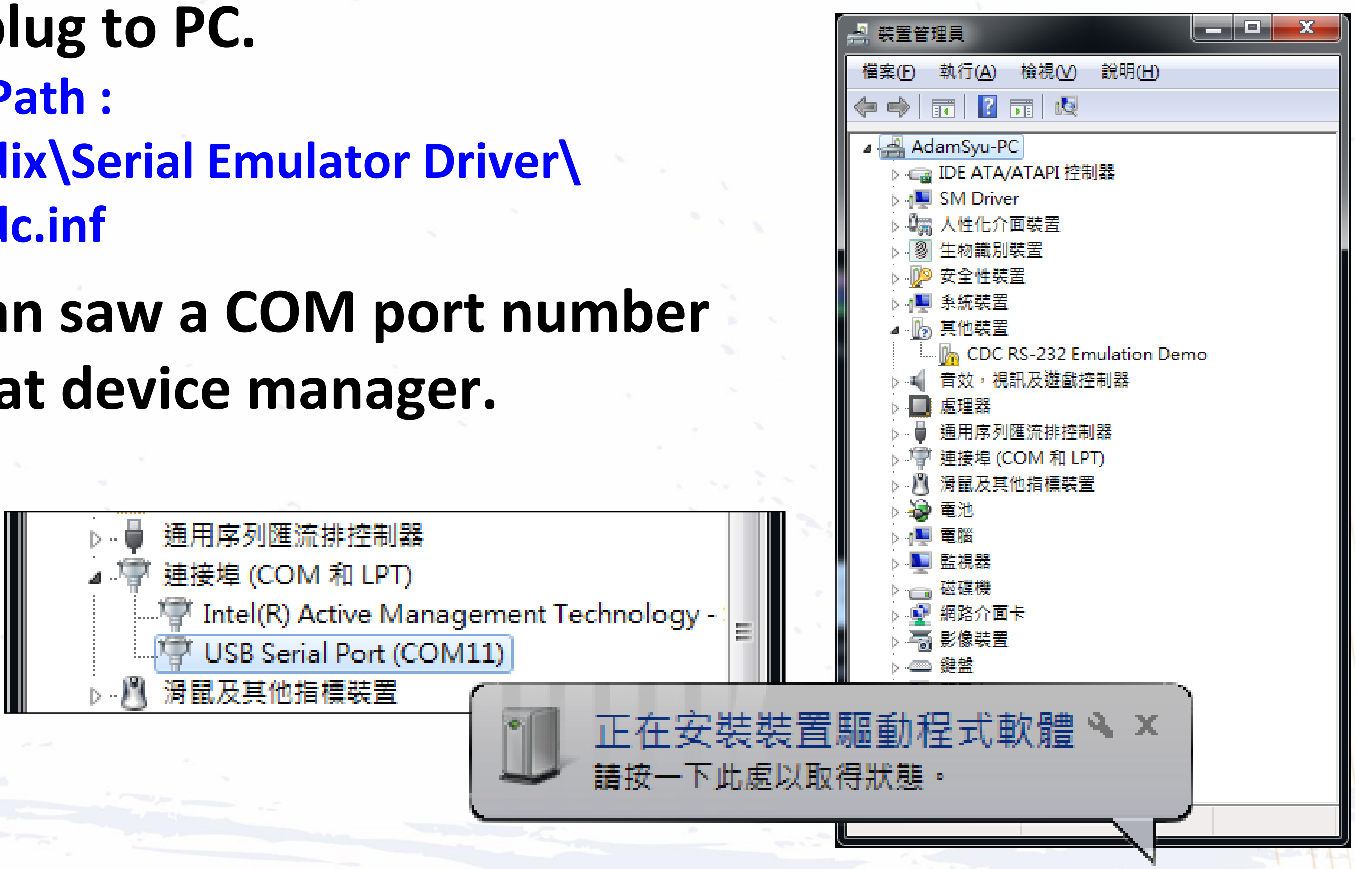
USB CDC Emulator

- **The PIC16F1455 already preload a call USB Serial Emulator firmware (COM Port).**



Driver Installation

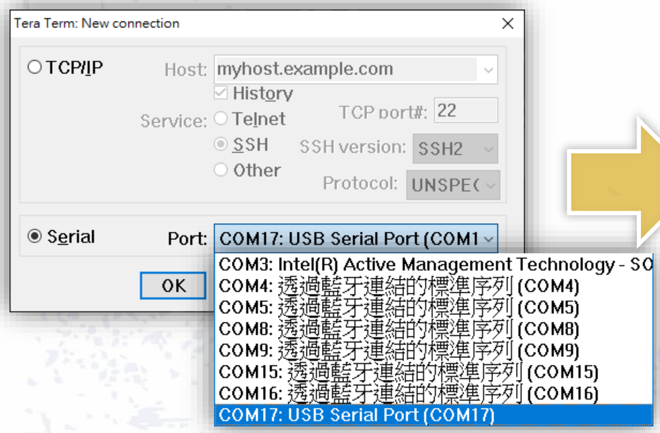
- **Please plug-in micro USB cable to COM21 and confirm device manager at your OS. You need install driver if first time plug to PC.**
  Driver Path : Appendix\Serial Emulator Driver\ mchpcdc.inf
- **You can saw a COM port number show at device manager.**



正在安裝裝置驅動程式軟體
請按一下此處以取得狀態。

Terminal Software

- **Tera Term is a terminal software. We receive and transmit through Tera Term. (http://ttssh2.sourceforge.jp/index.html.en)**
- **Select COM Port "USB Serial Port"**
- **Set 9600 , N , 8 , 1.**



Tera Term: New connection

○ TCP/IP Host: myhost.example.com

☑ History

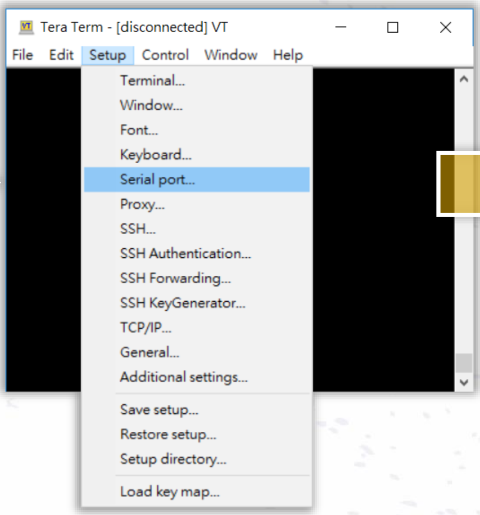
Service: ○ Telnet TCP port#: 22

● SSH SSH version: SSH2

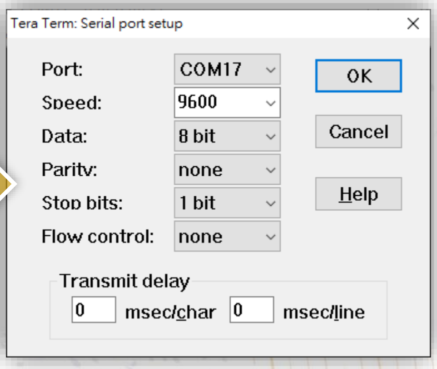
○ Other Protocol: UNSPEC

● Serial Port: COM17: USB Serial Port (COM1

OK
COM3: Intel(R) Active Management Technology - SO
COM4: 透過藍牙連結的標準序列 (COM4)
COM5: 透過藍牙連結的標準序列 (COM5)
COM8: 透過藍牙連結的標準序列 (COM8)
COM9: 透過藍牙連結的標準序列 (COM9)
COM15: 透過藍牙連結的標準序列 (COM15)
COM16: 透過藍牙連結的標準序列 (COM16)
COM17: USB Serial Port (COM17)



Tera Term - [disconnected] VT



Tera Term: Serial port setup

Port: COM17

Speed: 9600

Data: 8 bit

Parity: none

Stop bits: 1 bit

Flow control: none

Transmit delay

0 msec/char 0 msec/line

Lab12 UART Tx Polling

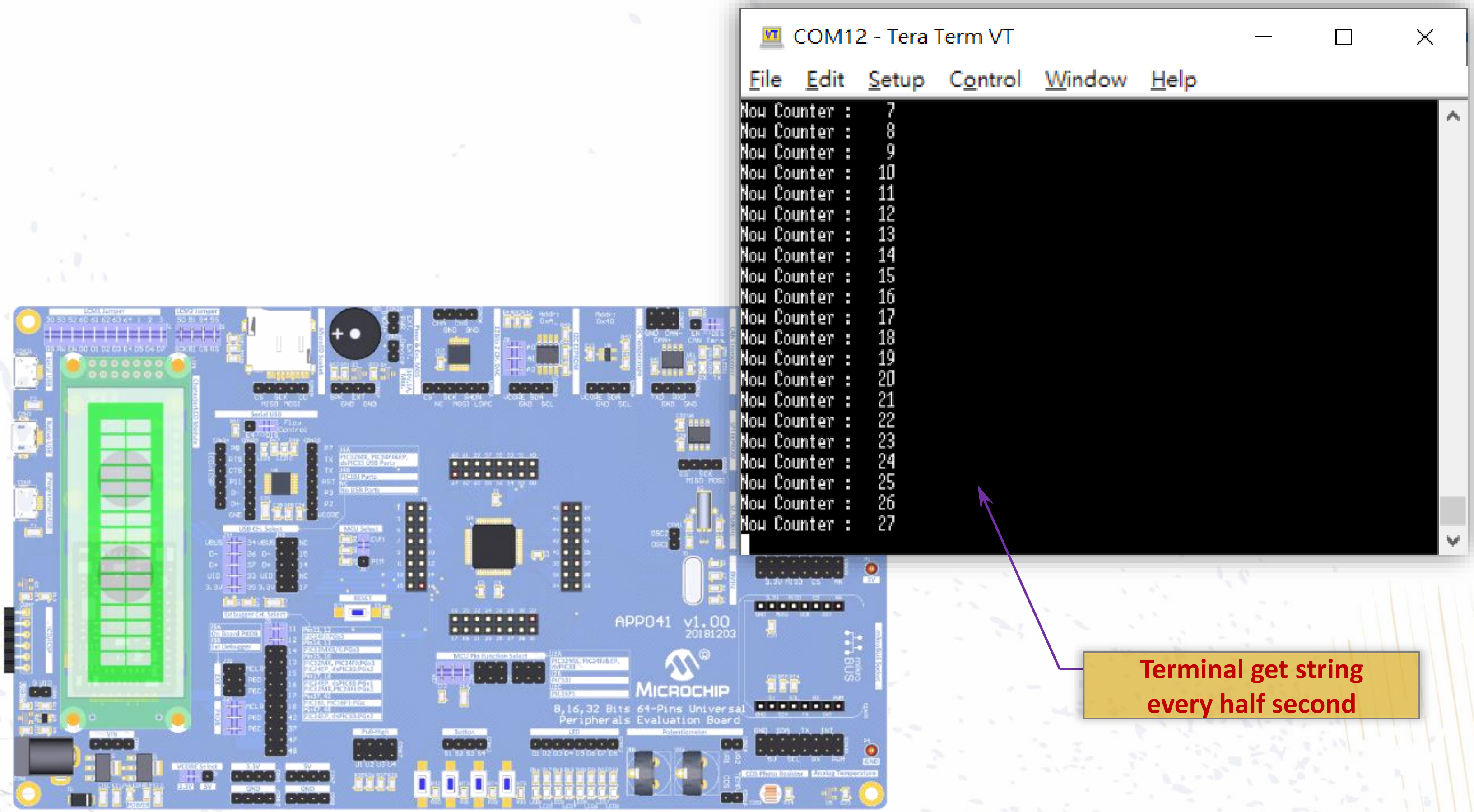
Lab12 UART Tx Polling

- Try to initial UART1 module, then send string from PIC to PC via UART module every half second.
- UART1 set to 9600, N, 8 , 1, No flow control.
- Please connect RG7(TX) and RG8(RX) to PIC16F1455 Tx, Rx pins.
  - RG7(TX) <-> Rx(D3)
  - RG8(RX) <-> Tx(D2)
- Send “Now Counter : 0000” string to PC one time per half second continuously.

- **Let’s go!**

Lab12 UART Tx Polling

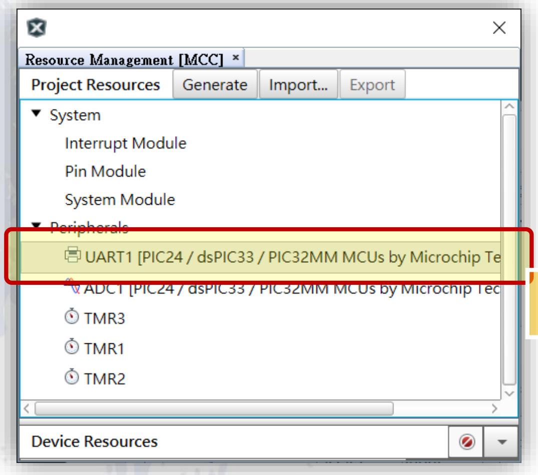
Result


COM12 - Tera Term VT
File Edit Setup Control Window Help
APP041 v1.00 20181203
MICROCHIP
8,16,32 Bits 64-Pins Universal Peripherals Evaluation Board

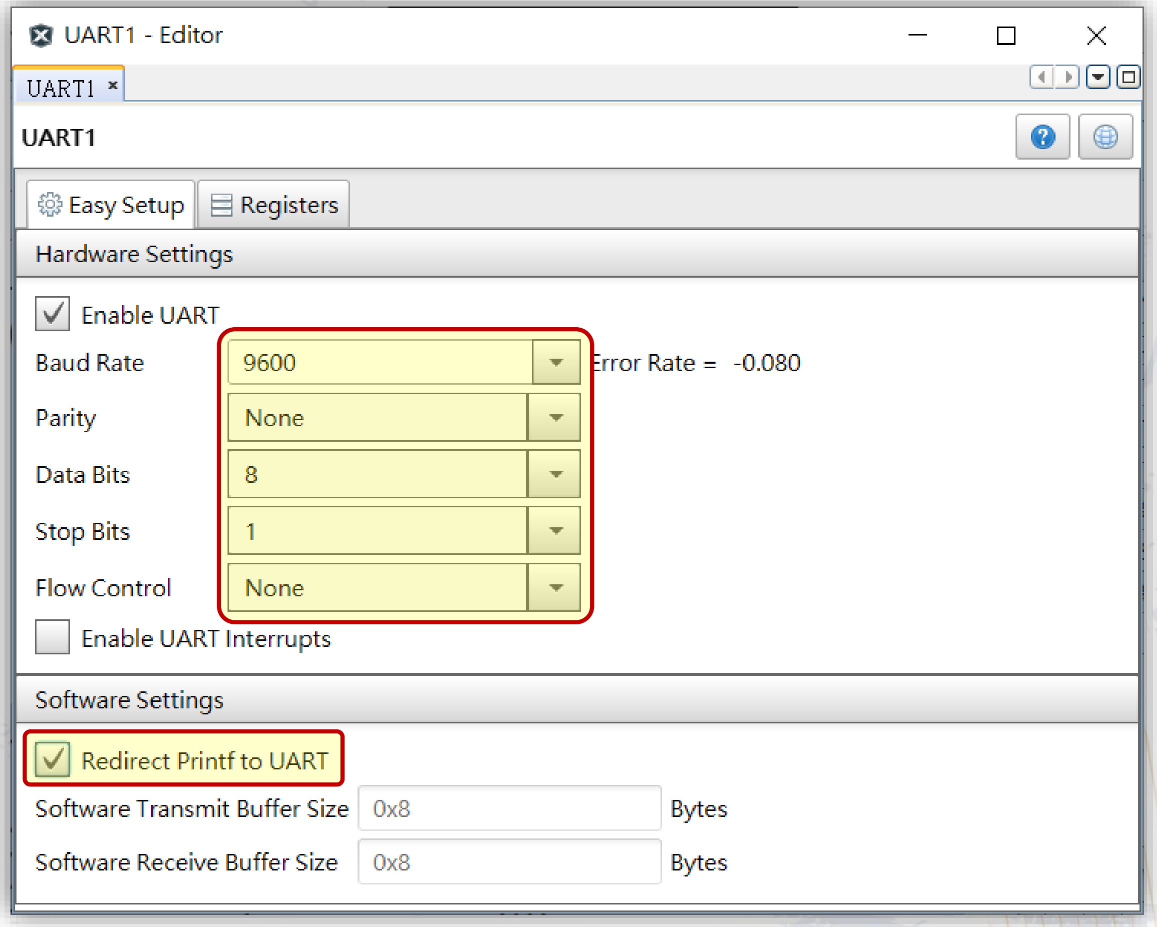
```
Now Counter :    7
Now Counter :    8
Now Counter :    9
Now Counter :   10
Now Counter :   11
Now Counter :   12
Now Counter :   13
Now Counter :   14
Now Counter :   15
Now Counter :   16
Now Counter :   17
Now Counter :   18
Now Counter :   19
Now Counter :   20
Now Counter :   21
Now Counter :   22
Now Counter :   23
Now Counter :   24
Now Counter :   25
Now Counter :   26
Now Counter :   27
```

Terminal get string every half second

Lab12 UART Tx Polling
MCC’s Setting & Code Example



- System
  - Interrupt Module
  - Pin Module
  - System Module
- Peripherals
  - UART1 [PIC24 / dsPIC33 / PIC32MM MCUs by Microchip Te
  - ADC1 [PIC24 / dsPIC33 / PIC32MM MCUs by Microchip Tec
  - TMR3
  - TMR1
  - TMR2



Hardware Settings

☑ Enable UART

Baud Rate: 9600 Error Rate = -0.080

Parity: None

Data Bits: 8

Stop Bits: 1

Flow Control: None

☐ Enable UART Interrupts

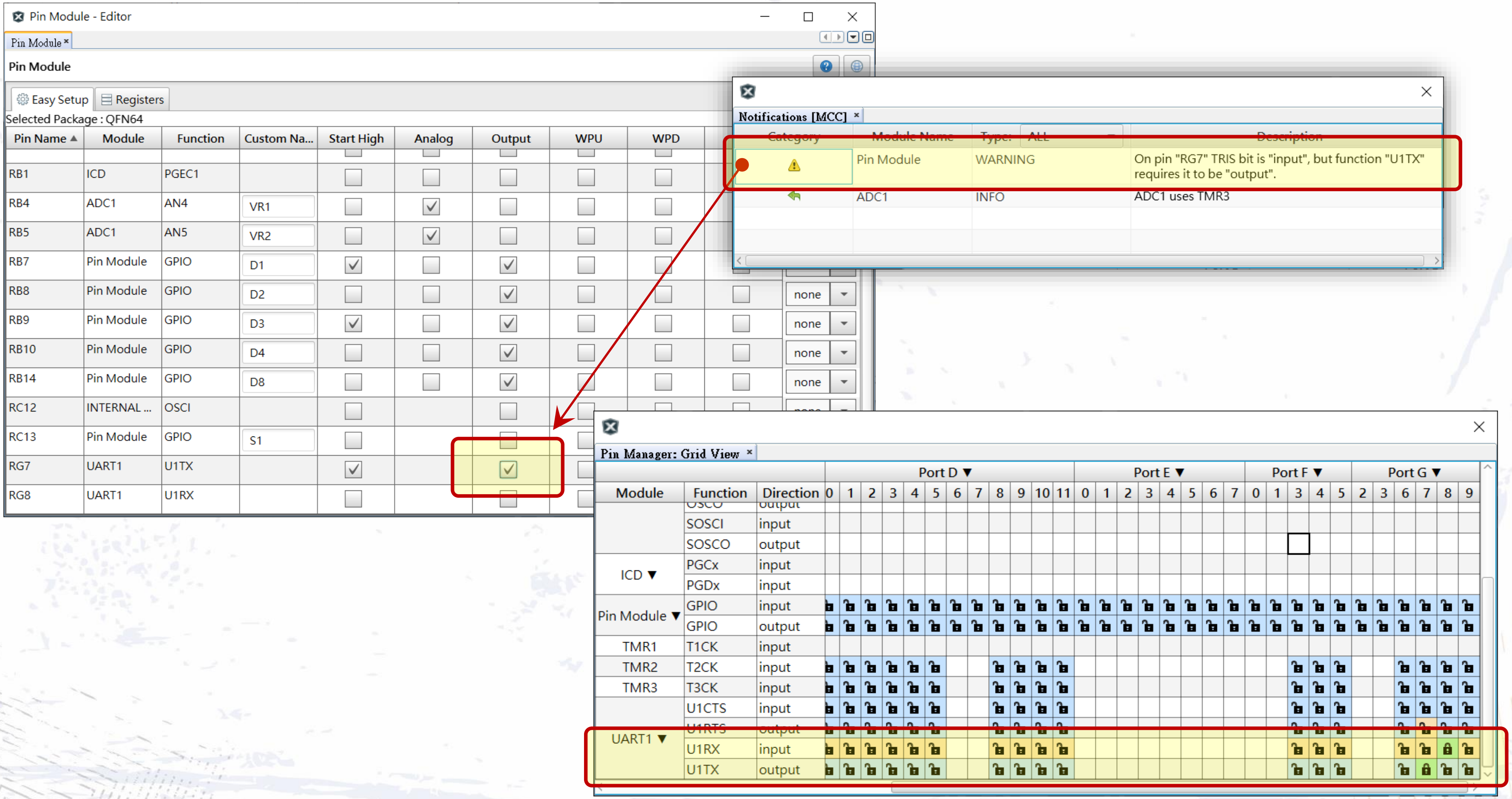
Software Settings

☑ Redirect Printf to UART

Software Transmit Buffer Size: 0x8 Bytes

Software Receive Buffer Size: 0x8 Bytes

Lab12 UART Tx Polling
MCC’s Setting & Code Example



Selected Package : QFN64

Pin Name ▲	Module	Function	Custom Na...	Start High	Analog	Output	WPU	WPD		
RB1	ICD	PGEC1		☐	☐	☐	☐	☐		
RB4	ADC1	AN4	VR1	☐	☑	☐	☐	☐		
RB5	ADC1	AN5	VR2	☐	☑	☐	☐	☐		
RB7	Pin Module	GPIO	D1	☑	☐	☑	☐	☐		
RB8	Pin Module	GPIO	D2	☐	☐	☑	☐	☐	☐	none
RB9	Pin Module	GPIO	D3	☑	☐	☑	☐	☐	☐	none
RB10	Pin Module	GPIO	D4	☐	☐	☑	☐	☐	☐	none
RB14	Pin Module	GPIO	D8	☐	☐	☑	☐	☐	☐	none
RC12	INTERNAL ...	OSCI		☐		☐	☐	☐	☐	
RC13	Pin Module	GPIO	S1	☐		☐	☐			
RG7	UART1	U1TX		☑		☑	☐			
RG8	UART1	U1RX		☐		☐	☐			

Category	Module Name	Type: ALL	Description
⚠	Pin Module	WARNING	On pin "RG7" TRIS bit is "input", but function "U1TX" requires it to be "output".
	ADC1	INFO	ADC1 uses TMR3

			Port D ▼												Port E ▼								Port F ▼					Port G ▼					
Module	Function	Direction	0	1	2	3	4	5	6	7	8	9	10	11	0	1	2	3	4	5	6	7	0	1	3	4	5	2	3	6	7	8	9
	OSCO	output																															
	SOSCI	input																															
	SOSCO	output																							☐								
ICD ▼	PGCx	input																															
	PGDx	input																															
Pin Module ▼	GPIO	input	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓
	GPIO	output	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓
TMR1	T1CK	input																															
TMR2	T2CK	input	🔓	🔓	🔓	🔓	🔓	🔓			🔓	🔓	🔓	🔓											🔓	🔓	🔓			🔓	🔓	🔓	🔓
TMR3	T3CK	input	🔓	🔓	🔓	🔓	🔓	🔓			🔓	🔓	🔓	🔓											🔓	🔓	🔓			🔓	🔓	🔓	🔓
	U1CTS	input	🔓	🔓	🔓	🔓	🔓	🔓			🔓	🔓	🔓	🔓											🔓	🔓	🔓			🔓	🔓	🔓	🔓
UART1 ▼	U1RTS	output	🔓	🔓	🔓	🔓	🔓	🔓			🔓	🔓	🔓	🔓											🔓	🔓	🔓			🔓	🔓	🔓	🔓
	U1RX	input	🔓	🔓	🔓	🔓	🔓	🔓			🔓	🔓	🔓	🔓											🔓	🔓	🔓			🔓	🔓	🔒	🔓
	U1TX	output	🔓	🔓	🔓	🔓	🔓	🔓			🔓	🔓	🔓	🔓											🔓	🔓	🔓			🔓	🔒	🔓	🔓

Lab12 UART Tx Polling
MCC’s Setting & Code Example

```
#include <stdio.h>
volatile unsigned char T1Flag = 0;
unsigned char Counter = 0;
int main(void)
{
    // initialize the device
    SYSTEM_Initialize();

    LCM_Init();

    while (1)
    {
        // Add your application code

        if(S1_GetValue())
            …

        if(T1Flag)
        {
            T1Flag = 0;
            printf("Now Counter : %4d\r\n" , Counter++);
        }
    }
    return -1;
}

void TMR1_CallBack(void)
{
    …
    T1Flag = 1;
}
```

MCC’s UART Interrupt

- **The code style is more easy if use enable UART interrupt mode.**
- **Receive and transmit buffer(queue) ready, So it’s easy to use don’t worrying data lost.**

MCC’s UART Function

- MCC Provide below common functions:

  void UART1_Initialize(void);

  // Initial UART.

- bool UART1_TransmitBufferIsFull(void);

  void UART1_Write(byte);

  unsigned int UART1_WriteBuffer(*buffer, bufLen);

  // Transmit & Tx Buffer Status Check.

- bool UART1_ReceiveBufferIsEmpty(void);

  void UART1_Read();

  unsigned int UART1_ReadBuffer(*buffer, bufLen);

  // Receive & Rx Buffer Status Check.

About String Function

- **UART 所處理的資料大多是字串或文字資料, 使用C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>**
- **For Example**

```
unsigned char USARTRTxBuffer[100];
unsigned int USART5_ReceiveData;
sprintf((char *) USARTRTxBuffer, "Received Data : %1c\r\n", USART5_ReceiveData);
UART1_ReadBuffer(USARTRTxBuffer, strlen((char *)USARTRTxBuffer));
```

UART Transmit Interrupt Example

-MCC Base-

- UART Transmit Interrupt Example

```
unsigned char UARTBuffer[ 100 ];

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

UART Receive Interrupt Example

-MCC Base-

- UART Receive Interrupt Example

```
unsigned char UARTBuffer[ 100 ];

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

Lab13 UART Tx & Rx Interrupt

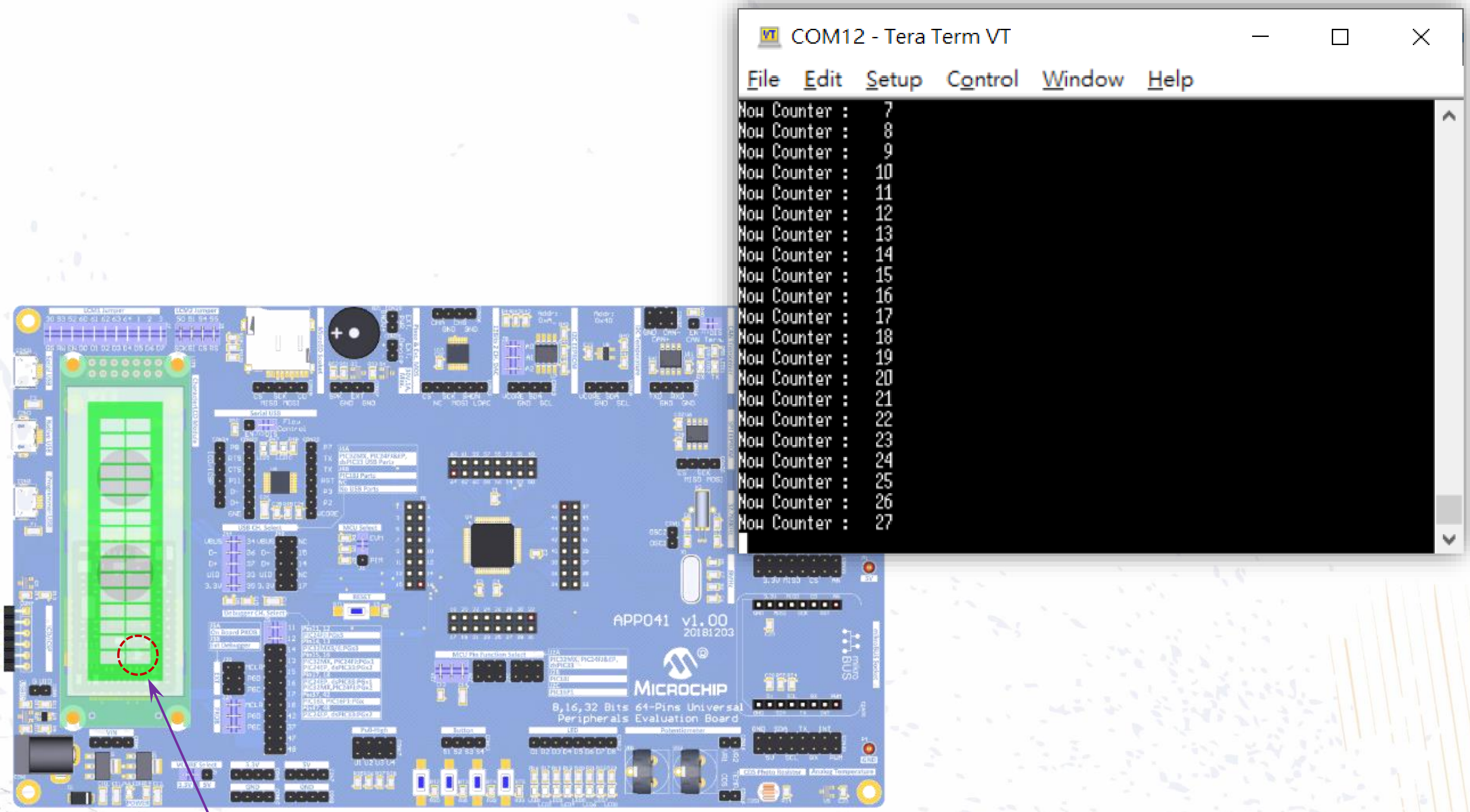
Lab13 UART Tx & Rx Interrupt

- Try to enable UART1 interrupt, interrupt priority set to 6 and set a suitable buffer size.
- Change UART1 code style from polling to interrupt.
- Send "Now Counter : 0000" string to PC one time per half second continuously, same as lab12.
- Put data to LCD Module, if received data from PC via UART1.

- **Let's go!**

Lab13 UART Tx & Rx Interrupt

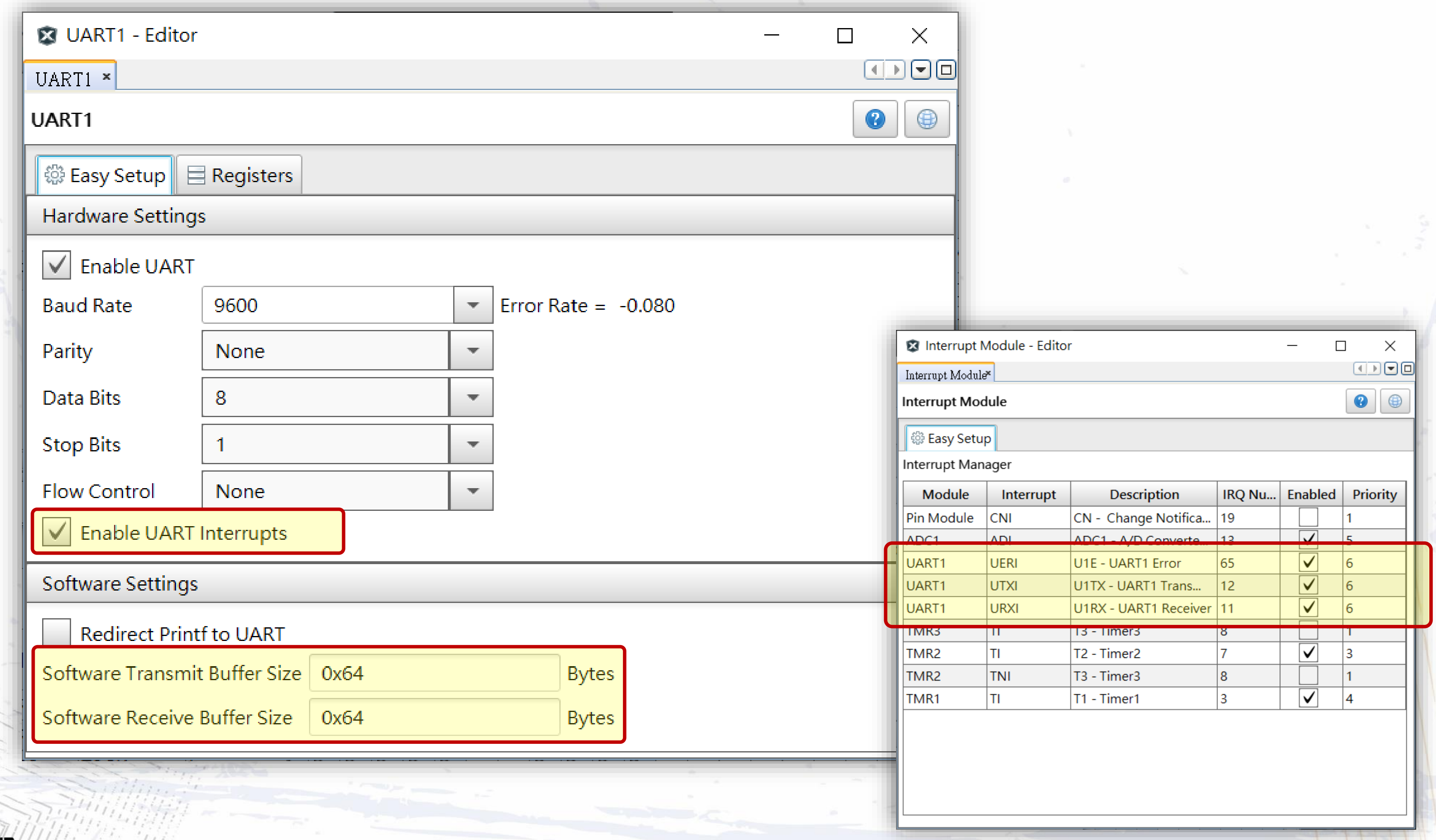
Result



```
Now Counter :   7
Now Counter :   8
Now Counter :   9
Now Counter :  10
Now Counter :  11
Now Counter :  12
Now Counter :  13
Now Counter :  14
Now Counter :  15
Now Counter :  16
Now Counter :  17
Now Counter :  18
Now Counter :  19
Now Counter :  20
Now Counter :  21
Now Counter :  22
Now Counter :  23
Now Counter :  24
Now Counter :  25
Now Counter :  26
Now Counter :  27
```

Show received Character at LCD Module

Lab13 UART Tx & Rx Interrupt
MCC’s Setting & Code Example



UART1

Hardware Settings

☑ Enable UART

Baud Rate: 9600 — Error Rate = -0.080

Parity: None

Data Bits: 8

Stop Bits: 1

Flow Control: None

☑ Enable UART Interrupts

Software Settings

☐ Redirect Printf to UART

Software Transmit Buffer Size: 0x64 Bytes

Software Receive Buffer Size: 0x64 Bytes

Interrupt Module

Interrupt Manager

Module	Interrupt	Description	IRQ Nu...	Enabled	Priority
Pin Module	CNI	CN - Change Notifica...	19	☐	1
ADC1	ADI	ADC1 - A/D Converte...	13	☑	5
UART1	UERI	U1E - UART1 Error	65	☑	6
UART1	UTXI	U1TX - UART1 Trans...	12	☑	6
UART1	URXI	U1RX - UART1 Receiver	11	☑	6
TMR3	TI	T3 - Timer3	8	☐	1
TMR2	TI	T2 - Timer2	7	☑	3
TMR2	TNI	T3 - Timer3	8	☐	1
TMR1	TI	T1 - Timer1	3	☑	4

Lab13 UART Tx & Rx Interrupt
MCC's Setting & Code Example

```
#include <stdio.h>
#include <string.h>
unsigned char UARTBuffer[ 100 ];
int main(void)
{
    …
    while (1)
    {
        if (T1Flag)
        {
            T1Flag = 0;
            if (!UART1_TransmitBufferIsFull())
            {
                sprintf((char *) UARTBuffer,
                                "Now Counter : %4d\r\n", Counter++);
                UART1_WriteBuffer((const uint8_t *) UARTBuffer,
                                strlen((const char *) UARTBuffer));
            }
        }

        if (!UART1_ReceiveBufferIsEmpty())
        {
            LCM_SetCursor(15, 0);
            LCM_PutASCII(UART1_Read());
        }
    …
    }
}
```