



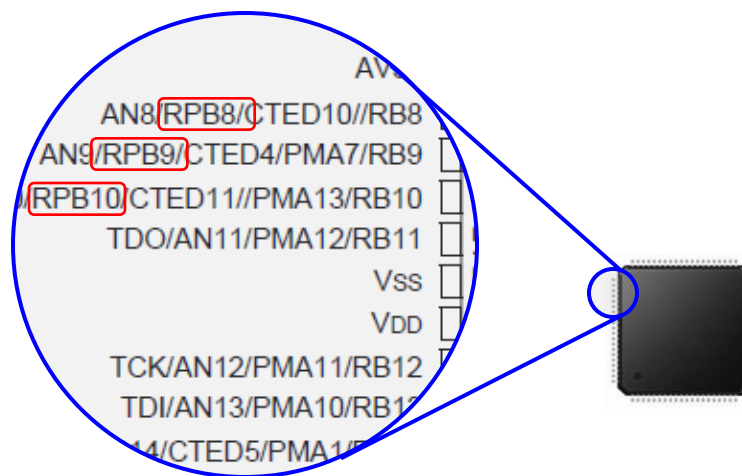
MICROCHIP

Regional Training Centers

Section 10
PPS and UART

What's PPS

- PPS : Peripheral Pin Select, 周邊接腳選擇
一種可以重新改變周邊訊號接腳位置的功能。讓使用者可以自行安排(Remapping)周邊訊號接腳位置,達到最彈性化的接腳使用。
- PIC32MX470具有PPS功能。在接腳上有標示RP_{*n*}x的, 都是可以Remapping的接腳。可已透過設定將UART,SPI,etc..的訊號接腳透過該Pin連接。
- PPS支援大部分的數位週邊,如:
UART,SPI,OC,IC等。
- 但需要較複雜狀態處理的數位週邊及類比功能則不支援, 其接腳為固定不可變更的,如:
I²C,USB,PMP及類比輸入等。



PPS Lock, Unlock

- PPS 設定前必須先進行解鎖,設定完後再上鎖。以避免非預期的動作更動PPS的設定。任何位元的非預期改變都會造成 MCU Reset。
- 解鎖動作必須將OSCCON的IOLOCK清除為0;
`OSCCONbits.IOLOCK = 0;`
鎖定動作必須必須將OSCCON的IOLOCK設為1。
`OSCCONbits.IOLOCK = 1;`
- PPS 的鎖定模式,在Configuration Bit中設定
IOL1WAY:IOLOCK One-Way Set Enable bit
IOL1WAY = ON, PPS的Mapping只能被設定一次。
IOL1WAY = OFF, PPS的Mapping可以重複的設定。

XC32 PPS Function & Macro

- PPSOutput(*Group* , *RPPin* , *Function*);
Group : PPS群組
RPPin : PPS的接腳編號(RPn)
Function : 周邊模組輸出訊號
NULL , CxOUT , UxTX , UxRTS , SDOx , SCKxOUT ,
SSxOUT , OCx , CTMU
- PPSInput(*Group* , *Function* , *RPPin*);
Group : PPS群組
Function : 周邊模組輸入訊號
TxCK , INTx , ICx , OCFx , UxRX , UxCTS , SDIx ,
SCKxIN , SSxIN
RPPin : PPS的接腳編號(RPn)
- PPSUnlock;
- PPSLock;

PPS Group

- PIC32MX470的PPS有群組分配, 周邊功能與PPS接腳不能隨意分配, 必須遵照Datasheet的規劃。

TABLE 12-2: OUTPUT PIN SELECTION

RPn Port Pin	RPnR SFR	RPnR bits	RPnR Value to Select
Group 1 Selections			
RPD2	RPD2R	RPD2R<3:0>	0000 = No Connection
RPG8	RPG8R	RPG8R<3:0>	0001 = U3TX
RPF4	RPF4R	RPF4R<3:0>	0010 = U4RTS
RPD10	RPD10R	RPD10R<3:0>	0011 = Reserved
RPF1	RPF1R	RPF1R<3:0>	0100 = Reserved
RPB9	RPB9R	RPB9R<3:0>	0101 = Reserved
RPB10	RPB10R	RPB10R<3:0>	0110 = SDO2
RPC14	RPC14R	RPC14R<3:0>	0111 = Reserved
RPB5	RPB5R	RPB5R<3:0>	1000 = Reserved
RPC1 ⁽⁴⁾	RPC1R	RPC1R<3:0>	1001 = Reserved
RPD14 ⁽⁶⁾	RPD14R	RPD14R<3:0>	1010 = Reserved
RPG1 ⁽⁴⁾	RPG1R	RPG1R<3:0>	1011 = OC3
RPA14 ⁽⁴⁾	RPA14R	RPA14R<3:0>	1100 = Reserved
			1101 = C2OUT
			1110 = Reserved
			1111 = Reserved

TABLE 12-1: INPUT PIN SELECTION

Peripheral Pin	[pin name]R SFR	[pin name]R bits	[pin name]R Value to RPN Pin Selection
INT3	INT3R	INT3R<3:0>	0000 = RPD2
T2CK	T2CKR	T2CKR<3:0>	0001 = RPG8
IC3	IC3R	IC3R<3:0>	0010 = RPF4
U1RX	U1RXR	U1RXR<3:0>	0011 = RPD10
U2RX	U2RXR	U2RXR<3:0>	0100 = RPF1
U5CTS	U5CTSR	U5CTSR<3:0>	0101 = RPB9
REFCLKI	REFCLKIR	REFCLKIR<3:0>	0110 = RPB10
			0111 = RPC14
			1000 = RPB5
			1001 = Reserved
			1010 = RPC1 ⁽⁸⁾
			1011 = RPD14 ⁽³⁾
			1100 = RPG1 ⁽³⁾
			1101 = RPA14 ⁽³⁾
			1110 = Reserved
			1111 = RPF2 ⁽¹⁾
INT4	INT4R	INT4R<3:0>	0000 = RPD2
T5CK	T5CKR	T5CKR<3:0>	0001 = RPG7
IC4	IC4R	IC4R<3:0>	0010 = RPF5
U3RX	U3RXR	U3RXR<3:0>	0011 = RPD11
U4CTS	U4CTSR	U4CTSR<3:0>	0100 = RPF0
SDI1	SDI1R	SDI1R<3:0>	0101 = RPB1
SDI2	SDI2R	SDI2R<3:0>	0110 = RPE5
			0111 = RPC13
			1000 = RPB3
			1001 = Reserved
			1010 = RPC4 ⁽⁸⁾
			1011 = RPD15 ⁽³⁾
			1100 = RPG0 ⁽³⁾
			1101 = RPA15 ⁽³⁾
			1110 = RPF2 ⁽¹⁾
			1111 = RPF7 ⁽²⁾
INT2	INT2R	INT2R<3:0>	0000 = RPD9
			0001 = RPG6
			0010 = RPF8

PPS UART Mapping Example

- PPS設定UART2, Tx, Rx的Mapping:

Ex:

```
void main( void )
```

```
{
```

```
...
```

```
...
```

```
...
```

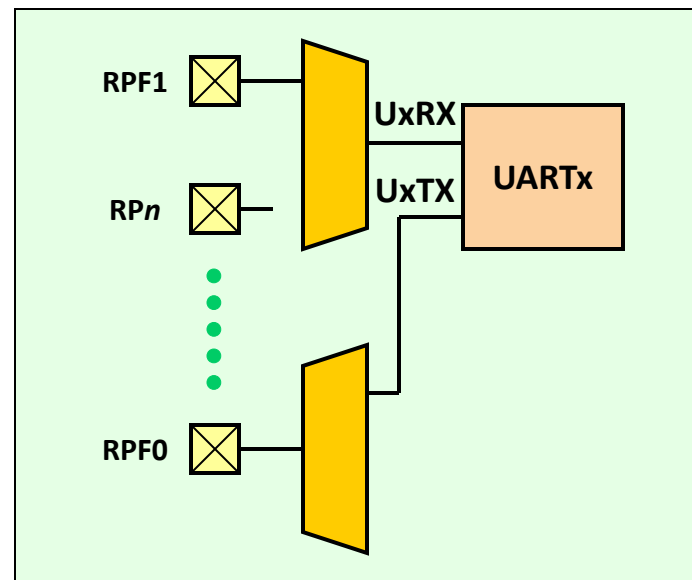
```
...
```

```
...
```

```
PPSUnlock;  
PPSOutput(2, RPF0, U2TX);  
PPSInput(1, U2RX, RPF1);  
PPSLock;
```

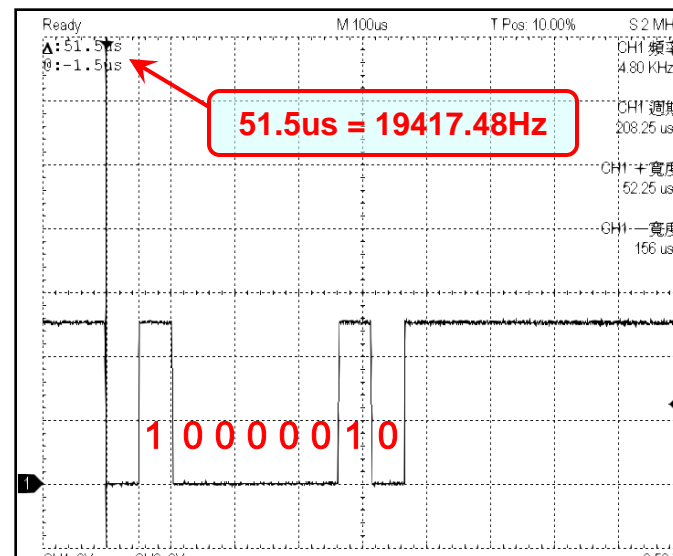
```
...
```

```
}
```

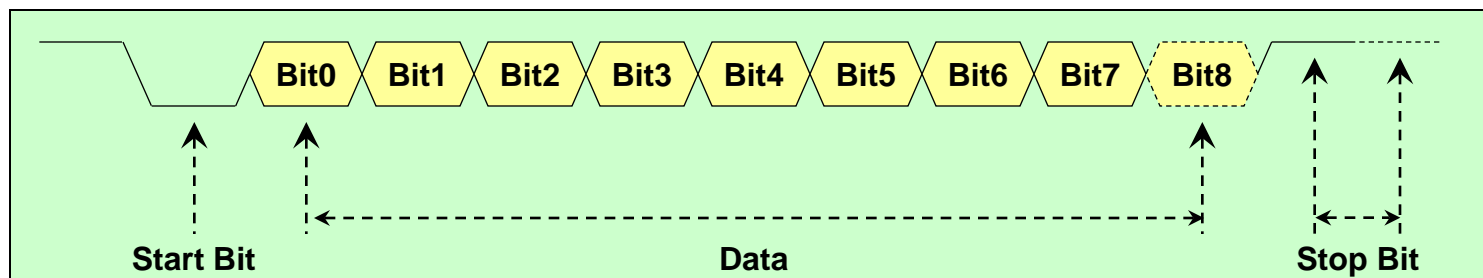


What's UART

- 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。

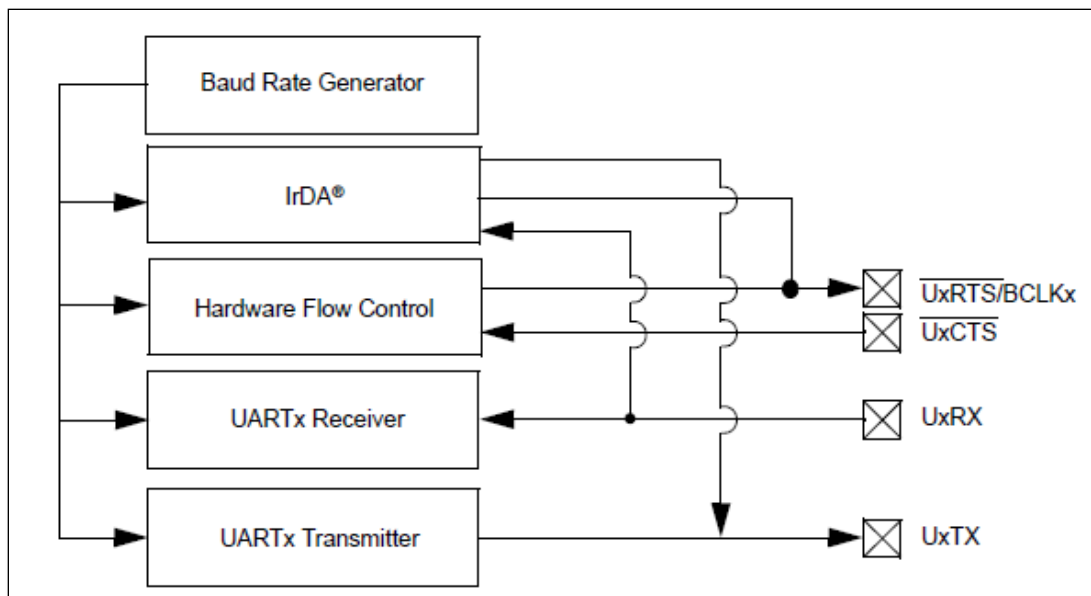


UART傳送範例'A'(0x41), 19200 bps



PIC32 UART Tran./Rece. Block

- PIC32的UART Module支援,8 or 9-Bits Data, 同位檢查及硬體流量控制(CTS,RTS)。
- 採全雙工, 非同步通訊。支援RS-232, RS-485, IrDA及LIN Bus。
- 八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。



XC32 UART Function & Example

- **UART Initial Example**

```
UARTConfigure( UART2, UART_ENABLE_PINS_TX_RX_ONLY );  
UARTSetFifoMode( UART2, UART_INTERRUPT_ON_TX_NOT_FULL |  
                  UART_INTERRUPT_ON_RX_NOT_EMPTY );  
UARTSetLineControl( UART2, UART_DATA_SIZE_8_BITS |  
                      UART_PARITY_NONE | UART_STOP_BITS_1 );  
UARTSetDataRate( UART2, PBFrequency, 19200 );  
UARTEnable( UART2, UART_ENABLE | UART_PERIPHERAL |  
             UART_TX | UART_RX );
```

- **UART Interrupt Example**

```
INTSetVectorPriority( INT_UART_2_VECTOR, INT_PRIORITY_LEVEL_5 );  
INTSetVectorSubPriority( INT_UART_2_VECTOR,  
                         INT_SUB_PRIORITY_LEVEL_0 );  
INTEnable( INT_U2RX, INT_ENABLED );
```

XC32 UART Function & Example

- **UART Receive (Polling) Example**

```
if( UARTReceivedDataIsAvailable(UART2 ) )  
{  
    Data = UARTGetDataByte( UART2 );  
}
```

- **UART Transmit(Polling) Example**

```
if( UARTTransmitterIsReady( UART2 ) )  
{  
    UARTSendDataByte( UART2 , Data );  
}
```

XC32 UART Function & Example

- **UART Receive (Interrupt) Example**

```
void __ISR( _UART_2_VECTOR, IPL5 ) ISR_UART2(void)
{
    if (INTGetFlag(INT_U2RX))
    {
        INTClearFlag( INT_U2RX );
        Data = UARTGetDataByte( UART2 );
    }
}
```

```
INTSetVectorPriority( INT_UART_2_VECTOR, INT_PRIORITY_LEVEL_5 );
INTSetVectorSubPriority( INT_UART_2_VECTOR ,
                          INT_SUB_PRIORITY_LEVEL_0 );
INTEnable( INT_U2RX , INT_ENABLED );
```

Lab10 UART Tx Polling Rx Interrupt

- UART的應用其實在前面的練習我們已經使用過了, 還記得printf()嗎?但是我們都只負責送資料, 那如何接收呢?
- 利用Lab6的程式,將UART的接收功能加入。透過UART的中斷,接收超級終端機輸入的字元,並判斷對應的字元來控制LED。
(‘+’ : Toggle RD8, ‘-’ : Toggle RD9)
- 先修改UART2的初始化,設定為非同步傳輸,規格為**19200, N, 8, 1**。建立UART2 接收的中斷服務常式,並開啟UART2的接收中斷,設定優先權。優先權設定為”5”。
- 設定PPS, 指定UART2的傳送接腳(U2TX)為**RPFO**, 接收(U2RX)為**RPF1**。閱讀說明文件,了解UART Function與PPS Function的使用方法。
- 使用Bootloader將程式燒錄進APP028-1。觀察程式執行的情況。驗證看看程式是否正常動作。

Lab10 UART Tx Polling Rx Interrupt *Step1*

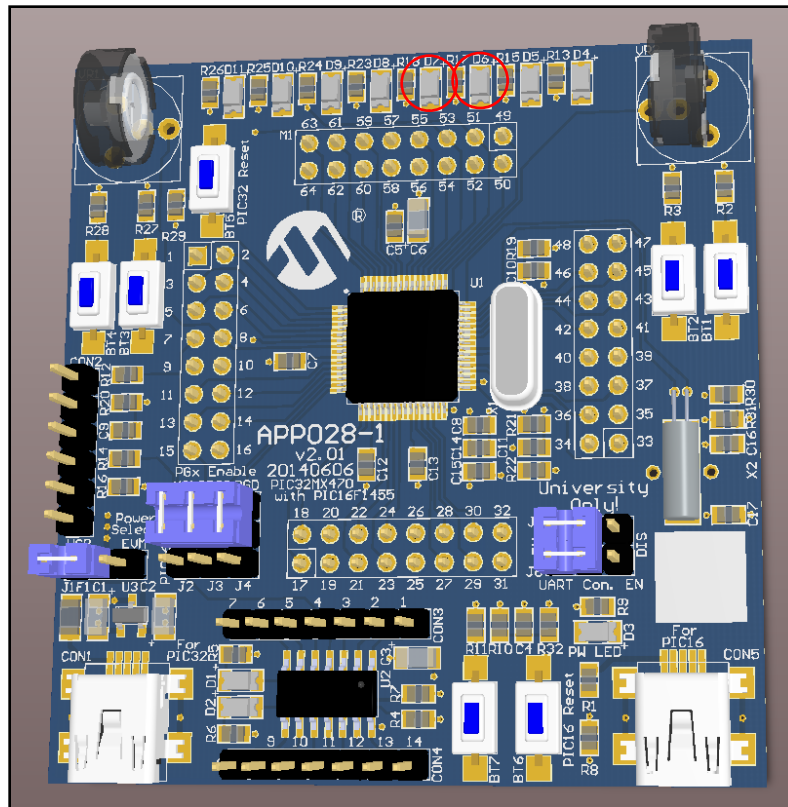
- 如何以中斷的方式, 來獲取UART收到的資料?

開啟UART_{*n*}的中斷, 設定優先權, 並加UART_{*n*}的中斷服務常式。
在中斷發生後使用 `RxValue=UARTGetDataByte(UARTn);` 把資料取出。

需注意的是, UART_{*n*}的中斷向量, 由U_{*n*}RX, U_{*n*}TX, U_{*n*}E。讀取資料前, 必須再次確認是否是U_{*n*}RX的事件。

Lab10 UART Tx Polling Rx Interrupt *Result!*

- 觀察LED的動作狀態, 當透過PC傳遞控制字元時, 對應的LED是否會正確的作動? 終端機軟體上是否有回應正確的訊息。
('+' : Toggle RD8, '-' : Toggle RD9)
- 記住Baud Rate改成19200了, 軟體的設定要跟著調整。



UART鮑率計算

- PIC32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:

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

鮑率計算的誤差

- UART模組鮑率計算誤差？

假設PB 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:

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

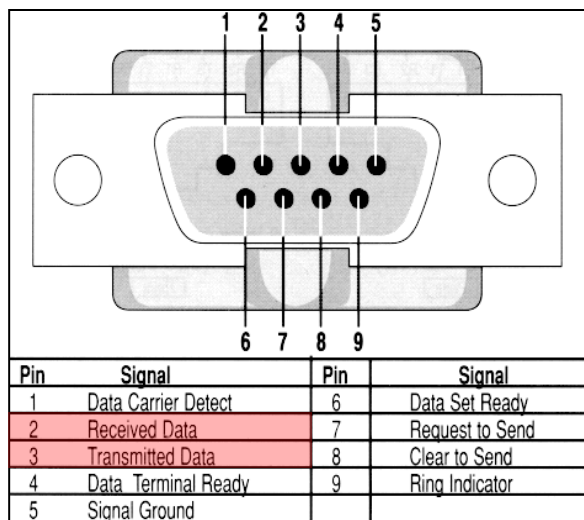
高速率時(BRGH=1),鮑率與誤差計算:

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

- UART誤差容忍度約為2%,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。

UART 與 RS232連接

- RS232是一般PC常見的串列通訊埠。
- UART設成非同步傳輸, 透過位準與邏輯轉換後, 可與RS232裝置連接。
- RS232採用負邏輯, 與MCU正邏輯系統不同。正邏輯"1"的電壓準位為 $0.8 V_{DD} \sim V_{DD}$; 邏輯"0"則為 $V_{SS} \sim 0.2V_{DD}$ 。負邏輯, 邏輯"1"的電壓準位為 $-3 \sim -12V$; 邏輯"0"則為 $+3 \sim +12V$ 。因次必須使用UART與RS232連接時, 必須進行位準與邏輯轉換。



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

