Embedded Systems
3.5 - UART (Clicker 2)

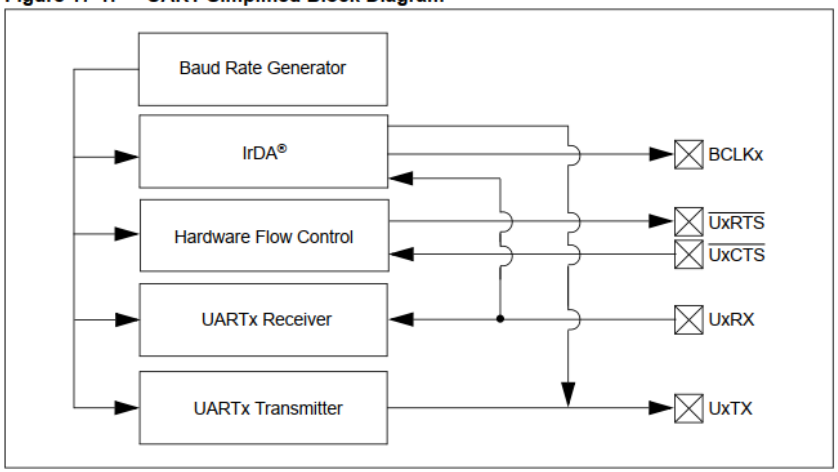
Enrico Simetti

ISME - Interuniversity Research Center on Integrated Systems for Marine Enviroment
DIBRIS - Department of Computer Science, Bioengineering, Robotics and System Engineering
University of Genova, Italy
enrico.simetti@unige.it

dsPIC33F UART I

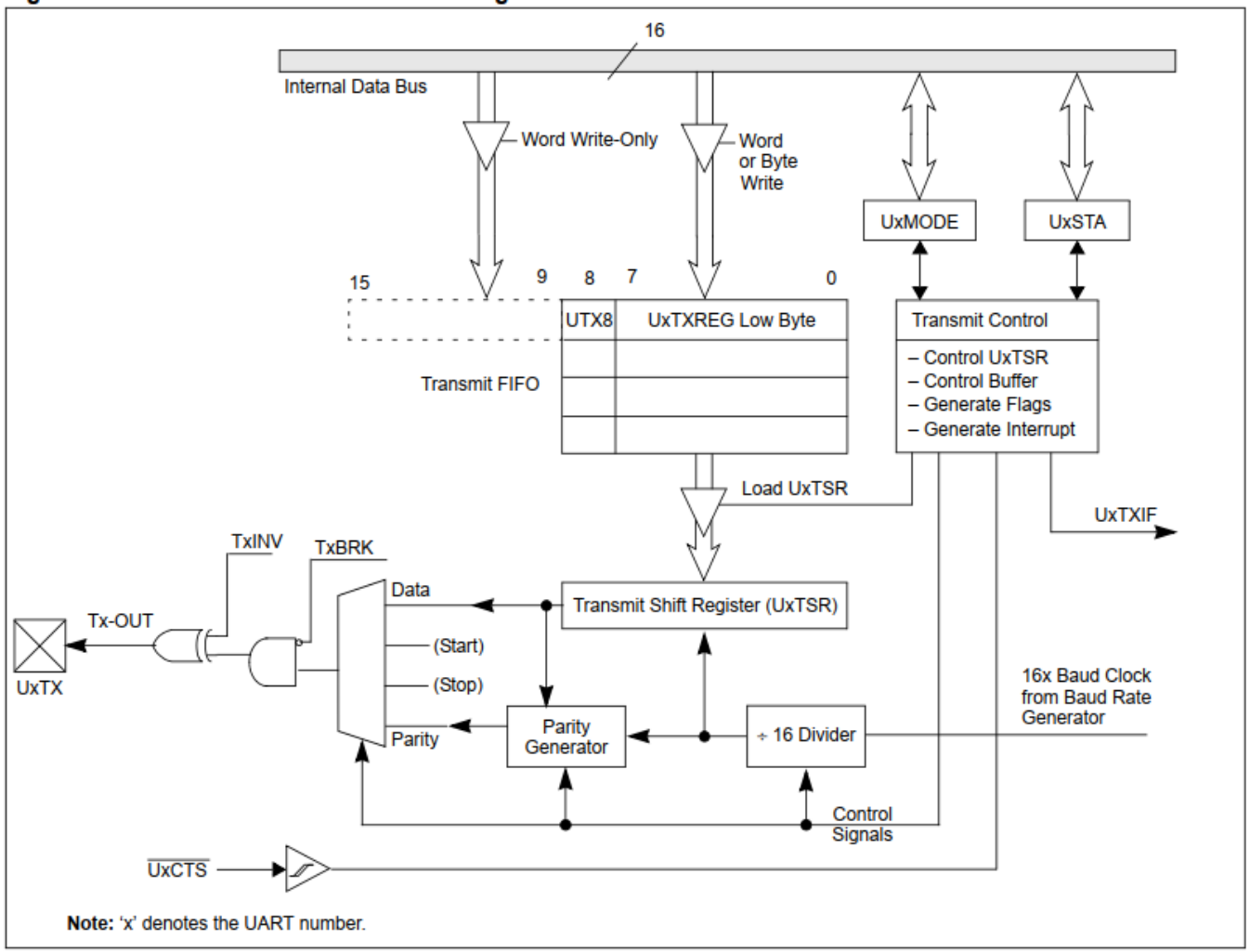
- Full-duplex 8- or 9-bit data transmission
- Even, Odd or No Parity options (for 8-bit data)
- One or two Stop bits
- Fully integrated Baud Rate Generator with 16-bit prescaler
- Baud rates ranging from 38 bps to 10 Mbps at FCY = 40 MHz
- 4-deep First-In-First-Out (FIFO) transmit data buffer
- 4-deep FIFO receive data buffer
- Parity, Framing and Buffer Overrun error detection
- Support for 9-bit mode with Address Detect (9th bit = 1)
- Transmit and Receive Interrupts
- Loopback mode for diagnostic support

Figure 17-1: UART Simplified Block Diagram



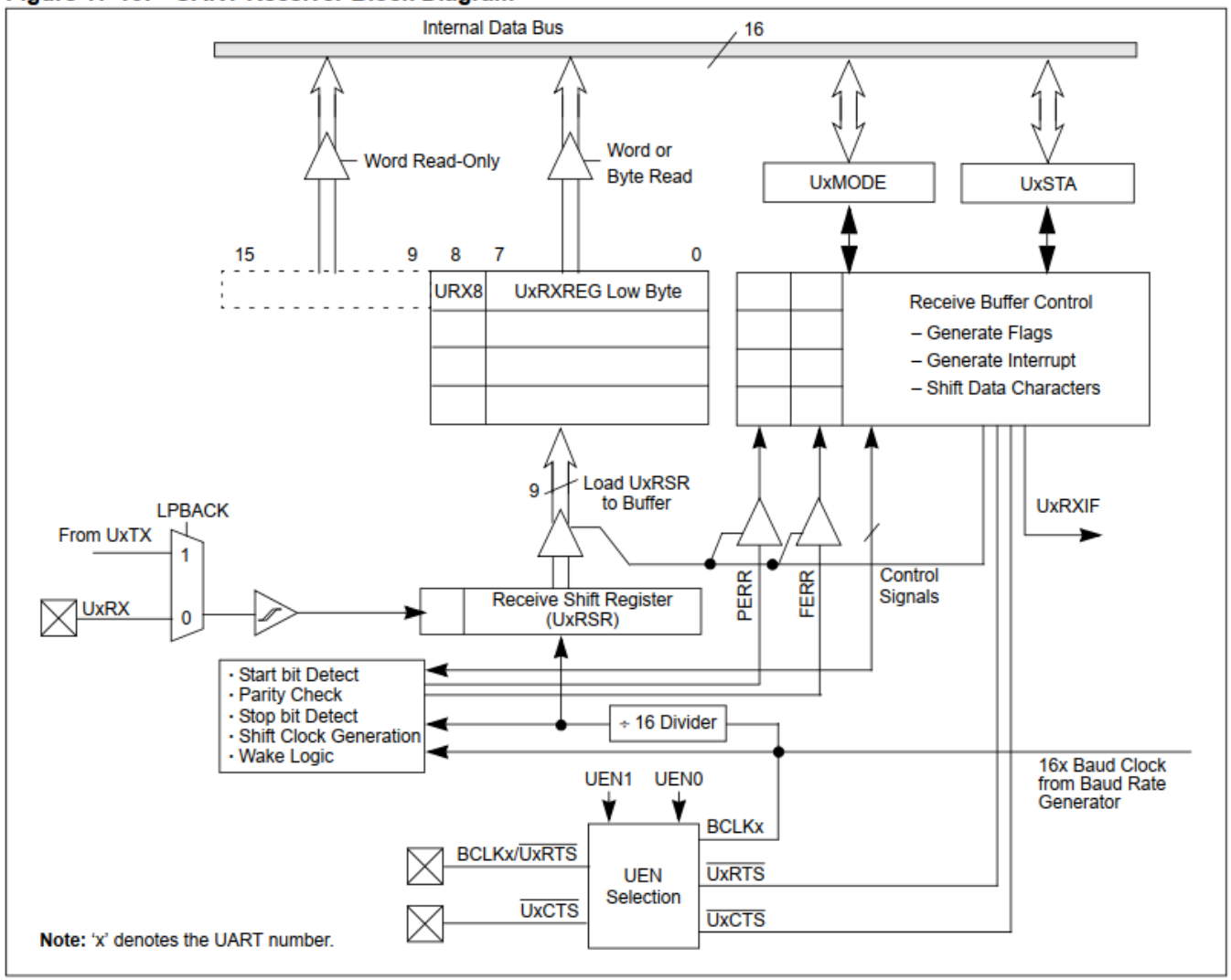
UART Transmitter

Figure 17-3: UART Transmitter Block Diagram



UART Receiver

Figure 17-10: UART Receiver Block Diagram



UART related registers I

REGISTER 20-2: UxSTA: UARTx STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	U-0	R/W-0, HC	R/W-0	R-0	R-1
UTXISEL1	UTXINV	UTXISEL0	—	UTXBRK	UTXEN(1)	UTXBF	TRMT
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R-1	R-0	R-0	R/C-0	R-0
URXISEL<1:0>		ADDEN	RIDLE	PERR	FERR	OERR	URXDA
bit 7							bit 0

UxSTA: UART Status registers, contains the flags that notify the program of certain events.

- *UTXISEL*: selects when the IF is set to 1 during transmission
- *URXISEL*: selects when the IF is set to 1 during reception
- *UTXEN*: enables transmission
- *TRMT*: notifies if there is an on-going transmission
- *URXDA*: notifies if there are characters to be read
- *OERR*: notifies if an overflow error occurred

UART related registers II

After an overflow occurred, and UxSTAbits.OERR = 1, the peripheral will discard any incoming bytes, until the user resets UxSTAbits.OERR = 0;
We have the following choices when an overflow occurred:

- read all the bytes in the UART buffer, then clear OERR;
- clear OERR immediately (the bytes are lost);

UART related registers III

bit 15 **UTXISEL:** Transmission Interrupt Mode Selection bit
1 = Interrupt when a character is transferred to the Transmit Shift register and as result, the transmit buffer becomes empty
0 = Interrupt when a character is transferred to the Transmit Shift register (this implies that there is at least one character open in the transmit buffer)

bit 7-6 **URXISEL<1:0>:** Receive Interrupt Mode Selection bit
11 =Interrupt flag bit is set when Receive Buffer is full (i.e., has 4 data characters)
10 =Interrupt flag bit is set when Receive Buffer is 3/4 full (i.e., has 3 data characters)
0x =Interrupt flag bit is set when a character is received

Changing these bits influences when UxTXIF and UxRXIF go to 1, i.e. about which kind of event the peripheral will alert us

UTXISEL and URXISEL do not enable the interrupts!

Check IEC0 register to enable the interrupts.

UART related registers IV

Register 17-1: UxMODE: UARTx Mode Register

R/W-0	U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
UARTEN	—	USIDL	IREN(1)	RTSMD	—	UEN<1:0>	
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WAKE	LPBACK	ABAUD	URXINV	BRGH	PDSEL<1:0>		STSEL
bit 7							bit 0

UxMODE: UARTx Mode register.

- *UARTEN*: enables the UART module
- *PDSEL*: selects the parity and data bits

UART related registers V

UxRXREG: contains the character received
UxTXREG: write here the character to be sent to the transmit buffer
UxBRG: the baud rate register

$$Baud\ Rate = \frac{FCY}{16 \bullet (UxBRG + 1)}$$

$$UxBRG = \frac{FCY}{16 \bullet Baud\ Rate} - 1$$

Note: FCY denotes the instruction cycle clock frequency.

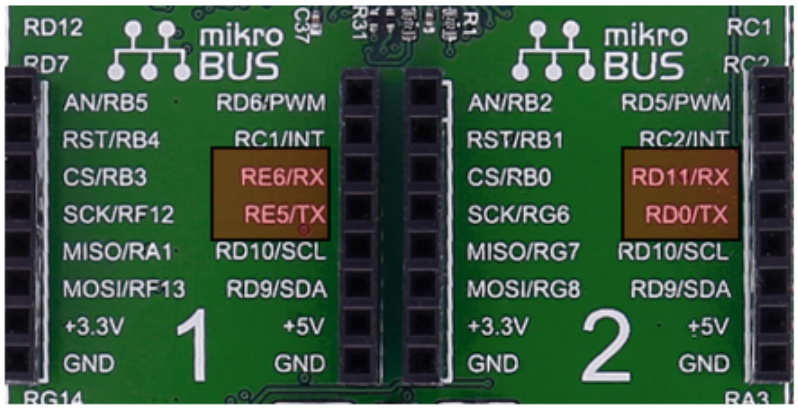
Example UART programming

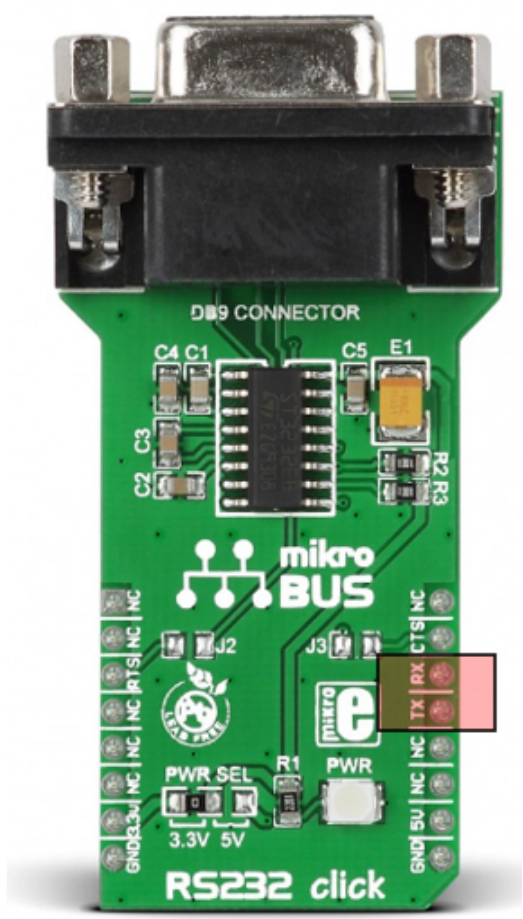
Example: sending one character at 9600 with UART1, assuming 7.3728 MHz oscillator (Fcy = 1.8432 MHz)

```
U1BRG = 11; // (7372800 / 4) / (16 * 9600) - 1
U1MODEbits.UARTEN = 1;  // enable UART
U1STAbits.UTXEN = 1;  // enable U1TX (must be after UARTEN)
U1TXREG = 'C'; // send 'C'
```

Clicker 2 Board UART I

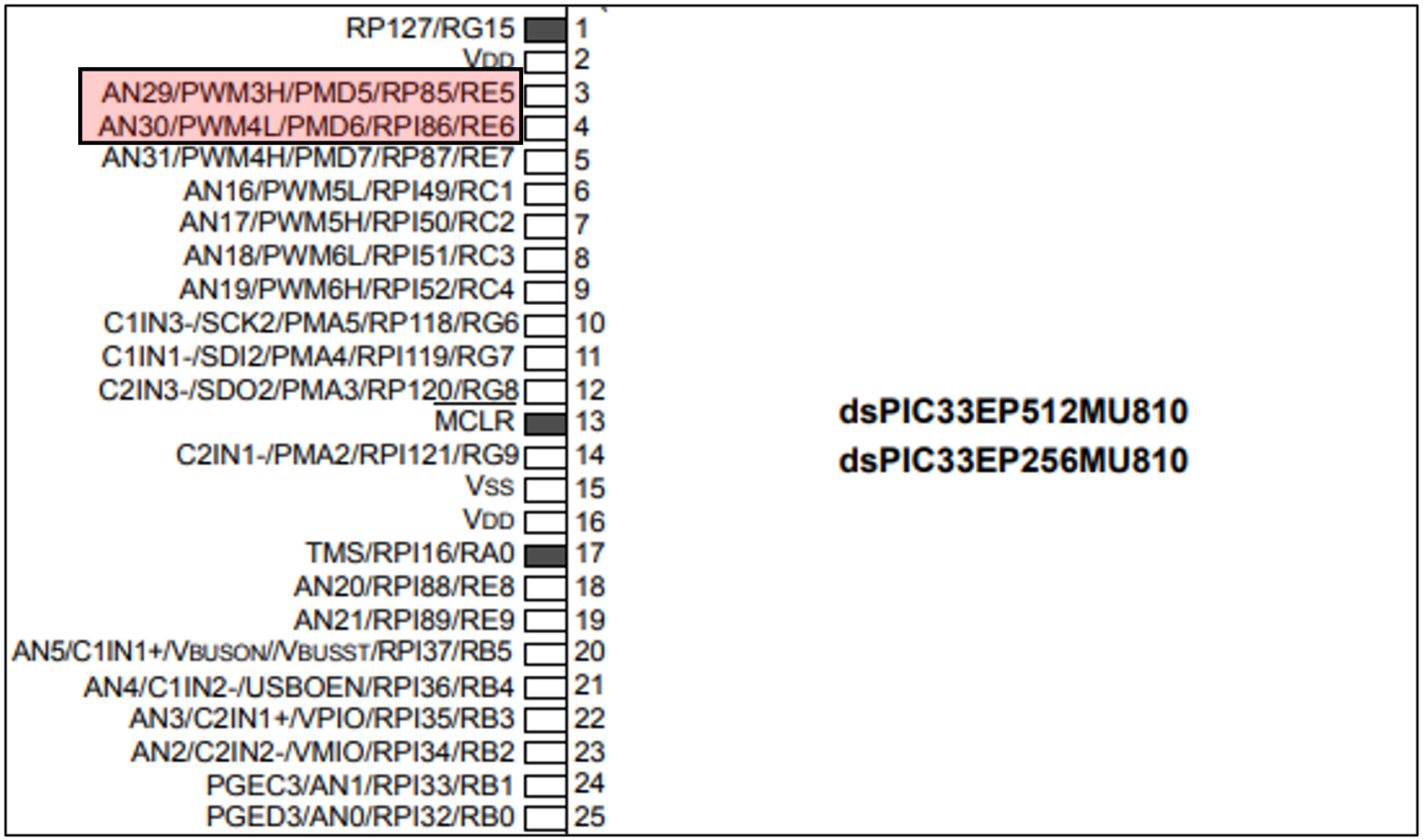
You must use the MikroBUS UART add-on board, inserted on one of the two on-board sockets:





Here we need to remap both input (RX) and output (TX)

Example: UART pins remap on MikroBUS 1



Input pin can be remapped as shown in the Interrupts section

Output pin remap with PPS:

input: reg that corresonds to functionality and select pin
outpu: select pin and then what output goes in that pin

TABLE 11-3: OUTPUT SELECTION FOR REMAPPABLE PINS (RPn)

Function	RPnR<5:0>	Output Name
DEFAULT PORT	000000	RPn tied to Default Pin
U1TX	000001	RPn tied to UART1 Transmit
$\overline{\text{U1RTS}}$	000010	RPn tied to UART1 Ready-to-Send
U2TX	000011	RPn tied to UART2 Transmit
$\overline{\text{U2RTS}}$	000100	RPn tied to UART2 Ready-to-Send
SDO1	000101	RPn tied to SPI1 Data Output
SCK1	000110	RPn tied to SPI1 Clock Output
$\overline{\text{SS1}}$	000111	RPn tied to SPI1 Slave Select
$\overline{\text{SS2}}$	001010	RPn tied to SPI2 Slave Select
CSDO	001011	RPn tied to DCI Data Output
CSCK	001100	RPn tied to DCI Clock Output
COFS	001101	RPn tied to DCI FSYNC Output
C1TX	001110	RPn tied to CAN1 Transmit
C2TX	001111	RPn tied to CAN2 Transmit
OC1	010000	RPn tied to Output Compare 1 Output
OC2	010001	RPn tied to Output Compare 2 Output
OC3	010010	RPn tied to Output Compare 3 Output
OC4	010011	RPn tied to Output Compare 4 Output
OC5	010100	RPn tied to Output Compare 5 Output
OC6	010101	RPn tied to Output Compare 6 Output
OC7	010110	RPn tied to Output Compare 7 Output
OC8	010111	RPn tied to Output Compare 8 Output
C1OUT	011000	RPn tied to Comparator Output 1
C2OUT	011001	RPn tied to Comparator Output 2
C3OUT	011010	RPn tied to Comparator Output 3
U3TX	011011	RPn tied to UART3 Transmit
$\overline{\text{U3RTS}}$	011100	RPn tied to UART3 Ready-to-Send

Note 1: This function is available in dsPIC33EPXXX(MC/MU)806/810/814 devices only.

Example UART remapping on MikroBUS 2

!!!!!!!!!!!!remap input and output of peripherals!!!!!!!!!!!

TABLE 4-37: PERIPHERAL PIN SELECT OUTPUT REGISTER MAP FOR dsPIC33EPXXXMU810/814 AND PIC24EPXXXGU810/814 DEVICES ONLY

File Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
RPOR0	0680	—	—	RP65R<5:0>						—	—	RP64R<5:0>						0000
RPOR1	0682	—	—	RP67R<5:0>						—	—	RP66R<5:0>						0000
RPOR2	0684	—	—	RP69R<5:0>						—	—	RP68R<5:0>						0000
RPOR3	0686	—	—	RP71R<5:0>						—	—	RP70R<5:0>						0000
RPOR4	0688	—	—	RP80R<5:0>						—	—	RP79R<5:0>						0000
RPOR5	068A	—	—	RP84R<5:0>						—	—	RP82R<5:0>						0000
RPOR6	068C	—	—	RP87R<5:0>						—	—	RP85R<5:0>						0000
RPOR7	068E	—	—	RP97R<5:0>						—	—	RP96R<5:0>						0000
RPOR8	0690	—	—	RP99R<5:0>						—	—	RP98R<5:0>						0000
RPOR9	0692	—	—	RP101R<5:0>						—	—	RP100R<5:0>						0000
RPOR11	0696	—	—	RP108R<5:0>						—	—	RP104R<5:0>						0000
RPOR12	0698	—	—	RP112R<5:0>						—	—	RP109R<5:0>						0000
RPOR13	069A	—	—	RP118R<5:0>						—	—	RP113R<5:0>						0000
RPOR14	069C	—	—	RP125R<5:0>						—	—	RP120R<5:0>						0000
RPOR15	069E	—	—	RP127R<5:0>						—	—	RP126R<5:0>						0000

Legend: x = unknown value on Reset, — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

Pins here are RD0/RP64 and RD11/RPI75

```
RPOR0bits.RP64R = 0x01; // Map UART 1 TX to pin RD0 which is RP64
RPINR18bits.U1RXR = 0x4B; // Map UART 1 RX to pin RD11 which is RPI75 (0x4B = 75)
```

uart1 TX

i put the pin that i need to use