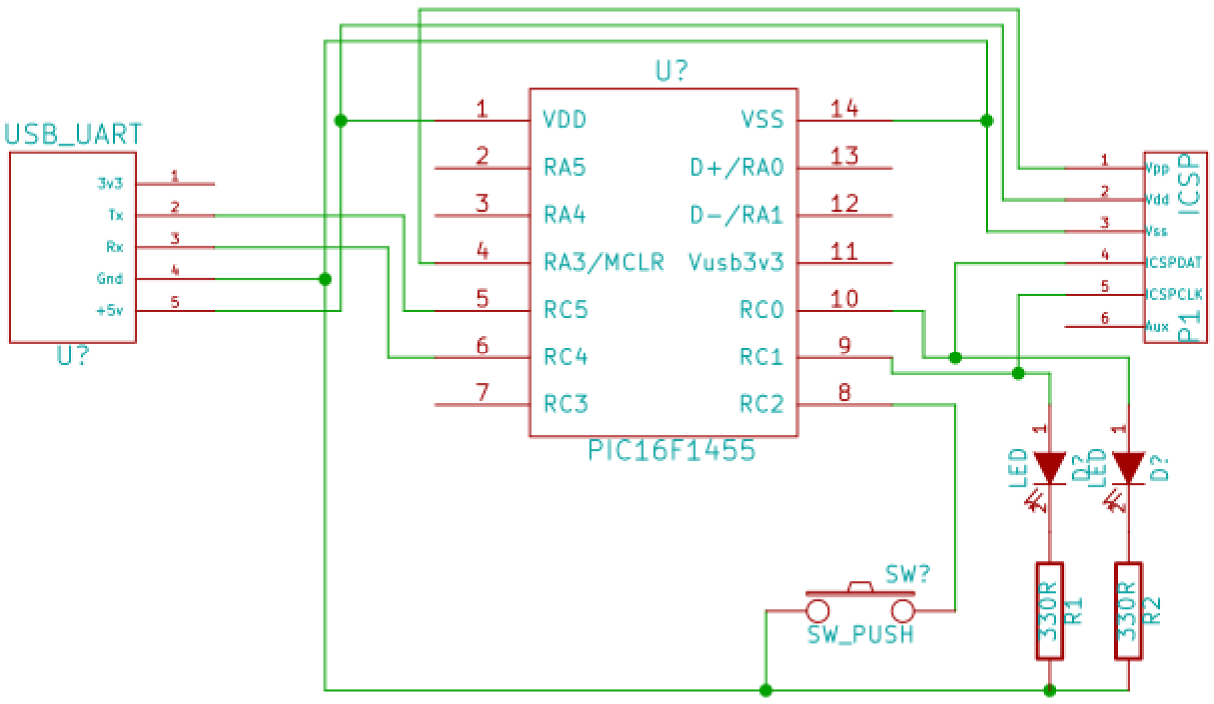
PIC Lab 2 – Serial Ports

A simple introduction to interfacing a PIC to a PC.



Component List

- PIC16F1455
- 0.1” square pin header, 6 pins
- 3x standard 5mm LEDs (one red, one green, one blue)
- 3x 220ohm resistors
- USB TTL UART Board
- momentary contact push switch
- PIC Programming Interface (eg PicKit 2)

Create the Project

1. Open MPLAB-X
2. File -> New Project -> Microchip Embedded -> Stand Alone
3. Mid-range 8bit -> PIC16F1455
4. Debug Header: None
5. Hardware Tools: Simulator
6. Compiler Toolchain: XC8
7. Project Name: lab2
8. Finish

Include the Serial Helper Library

9. Copy the serial.c and serial.h files into your project directory.
10. Source Files -> Add Existing Item... -> serial.c
11. Header Files -> Add Existing Item... -> serial.h

Set some global values

12. Header Files -> New -> C Header File -> system.h
13. Tell the rest of the code what the system clock speed is

```
#define _XTAL_FREQ 4000000L
```

Create the file

14. Projects -> Source Files -> New -> C Main File. Name: main.c
15. Add the device specific definitions

```
#include <pic16f1455.h>
#include <stdint.h>
#include “system.h”
#include “serial.h”
```

16. Window -> PIC Memory Views -> Configuration Bits

```
FOSC -> INTOSC
WDTE -> Off
CPUDIV -> NOCLKDIV
```

17. Generate Source Code -> Cut and Paste into code.
18. Configure the CPU clock. In function main()

```
    OSCCONbits.IRCF = 0b1101;
```

19. Configuration of the PINS

```
      /* 12.7  setup the ports for the LEDs */
       TRISCbits.TRISC0 = 0;
       TRISCbits.TRISC1 = 0;
       TRISCbits.TRISC2 = 0;

       /* disable analog features */
       ANSELCbits.ANSC0 = 0;
       ANSELCbits.ANSC1 = 0;
       ANSELCbits.ANSC2 = 0;

      /* now setup the buttons IO port */
      TRISAbits.TRISA4 = 1;
      ANSELAbits.ANSA4 = 0;
      WPUAbits.WPUA4 = 1;
      OPTION_REGbits.nWPUEN = 0;
```

20. Attach Serial handler routines to interrupts, this takes care of sending and receiving data in the background.

```
void interrupt isr(void)
{
     if (PIE1bits.TXIE && PIR1bits.TXIF)
          msg_sendnext();

     if (PIE1bits.RCIE && PIR1bits.RCIF)
          msg_recvnext();
}
```

21. Now the main function. First call the function to initialise the serial port

```
serial_init();
```

22. Now a loop, waiting for a command from the serial port

```
while (1)
{
     /* make sure we have finished sending everything */
     if (!msg_empty()) continue;

     /* if there is nothing waiting go around again */
     if (!msg_recvready()) continue;

     /* there is a character, read it */
     char cmd = msg_recv();

     /* now decide what to do */
     if (cmd == 'r') {
          PORTCbits.RC0 = !PORTCbits.RC0;
     } else
     if (cmd == 'g') {
          PORTCbits.RC1 = !PORTCbits.RC1;
     } else
     if (cmd == 'b') {
          PORTCbits.RC2 = !PORTCbits.RC2;
     }
}
```

23. Extra exercise #1 –

    Monitor the button for presses and report to the serial port.
    Button is on port RA4 and reads 0 when pressed.

24. Extra exercise #2 –

    Accept a number after the letter and use PWM to set brightness of each LED separately.