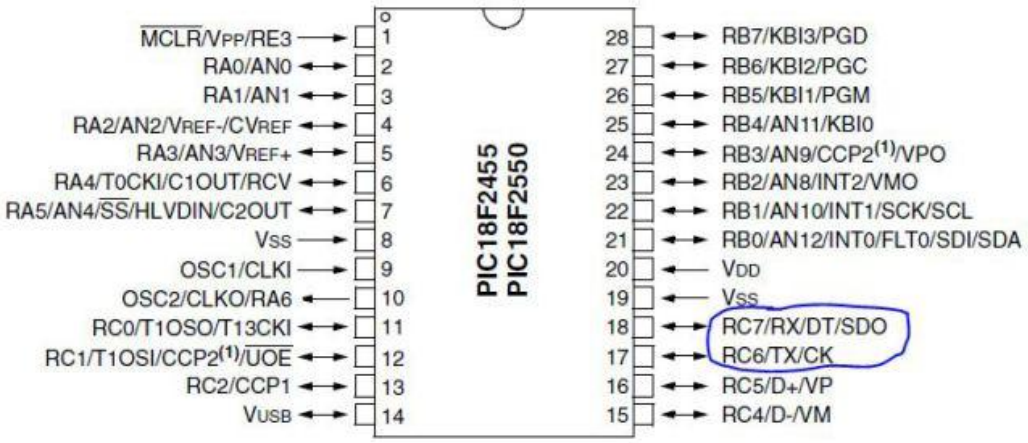
USART Communication For PIC18 Microcontroller

In my last post, I mentioned two awesome devices I've been messing around with lately. The cool thing about those devices is that it only needed one actual pin from the PIC microcontroller to properly work. This pin is one of the pins needed to achieve proper USART communication. Today's tutorial will serve as a guide in order to achieve proper USART communication. Although I mentioned that USART is used to communicate with the devices I mentioned in the last post, today's focus will be on achieving communication between PIC18F2550 and the computer using USART.

Circuity

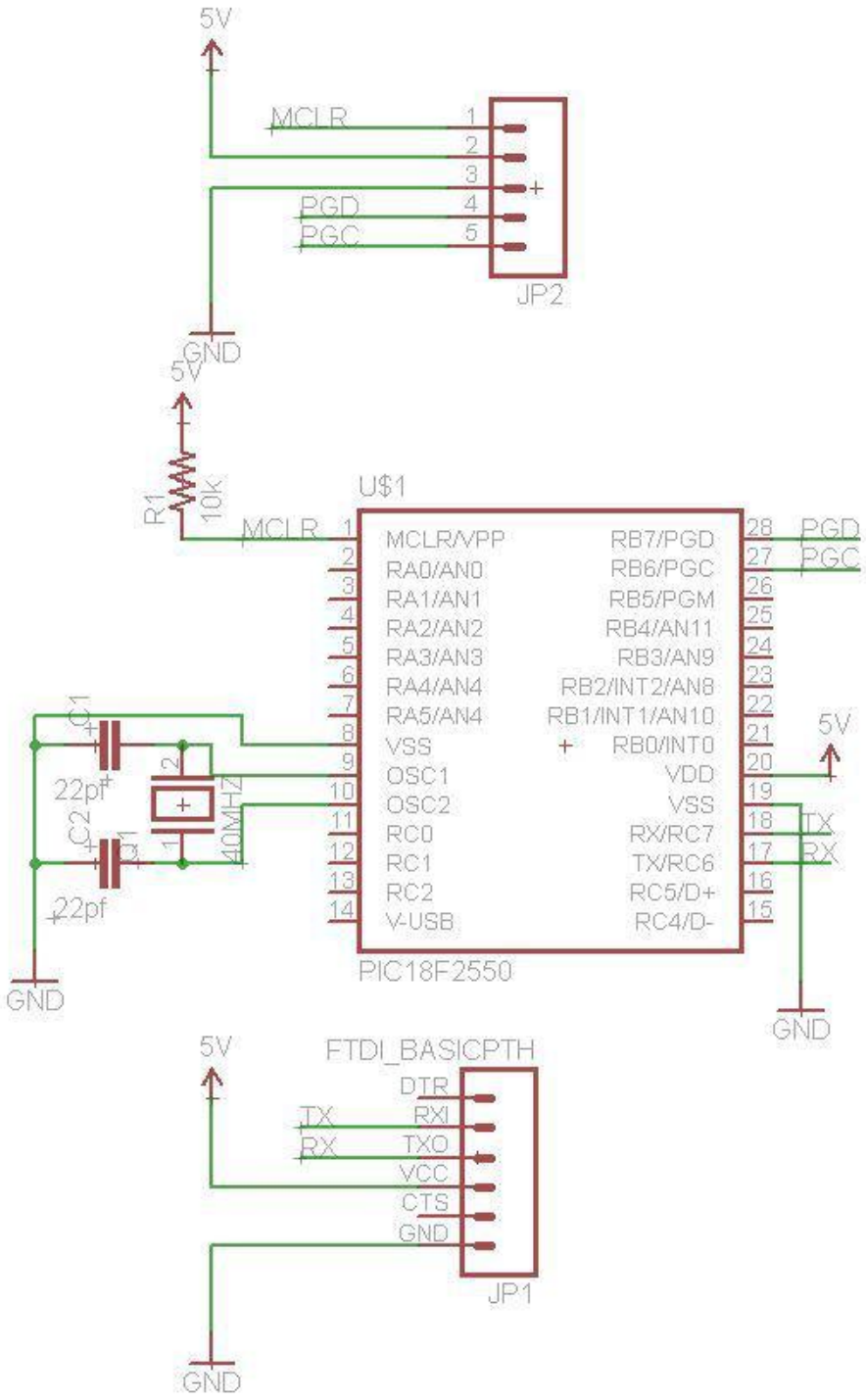
USART (Universal Synchronous Asynchronous Receiver Transmitter)/ UART (Universal Asynchronous Receiver Transmitter) requires only two pins from any microcontroller to send and receive serial data: TX and RX. You can properly guess that TX is the pin which the microcontroller data is transmitted through, and RX is responsible for data received by the microcontroller. The figure below shows the TX and RX pins on the PIC18F2550.

28-Pin PDIP, SOIC



Keep in mind; we need one more device to have the PIC18F2550 communicate with the computer. If you have an old computer, then you could use a MAX232 and connect it directly to the serial port of that computer. For this tutorial, we're going to use a

laptop to talk to the PIC. In order to do this, we're going to use a FTDI to USB board from Sparkfun. We're just going to use 4 pins of the FTDI to USB board: VCC, GND, TX and RX. Look at the figure below to learn how to set up the circuit. Please note that JP2 is connected to the PICKIT3.



Code

Now, create a new project in MPLab X, create a new C file, and code and paste the following code. If you forgot how to create a project in MPlab X, visit my tutorial on starting with PIC18 microcontrollers.

```
1   #include <p18f4553.h>
2   #include <delays.h>
3   #include <portb.h>
4   #include <usart.h>
5   #include <stdio.h>
6   #include <stdlib.h>
7
8   /*  PIC Configuratings */
9   #pragma config OSC = HS
10  #pragma config WDT = OFF
11  #pragma config PBADEN = OFF
12  #pragma config LVP = OFF
13  #pragma config PWRT = ON
14  #pragma config DEBUG= OFF
15
```

```
16  /* UART Global variables */
17  char message;                   //  Used for receiving commands from the computer
18
19
20
21  void main(void)
22  {
23      //  Set all of PORTC as outputs. TX and RX must be set as outputs first
24      TRISC=0x00;
25
26      //  Configure UART
27      OpenUSART(USART_TX_INT_OFF & USART_RX_INT_OFF & USART_ASYNCH_MODE &U
    USART_CONT_RX & USART_BRGH_HIGH, 10);
28          Delay1KTCYx(4);
29          while(1)
30          {
31              while(!DataRdyUSART());
32              message=ReadUSART();
```

```
33              putcUSART(message);
34              Delay1KTCYx(4);
35          }
36      }
```

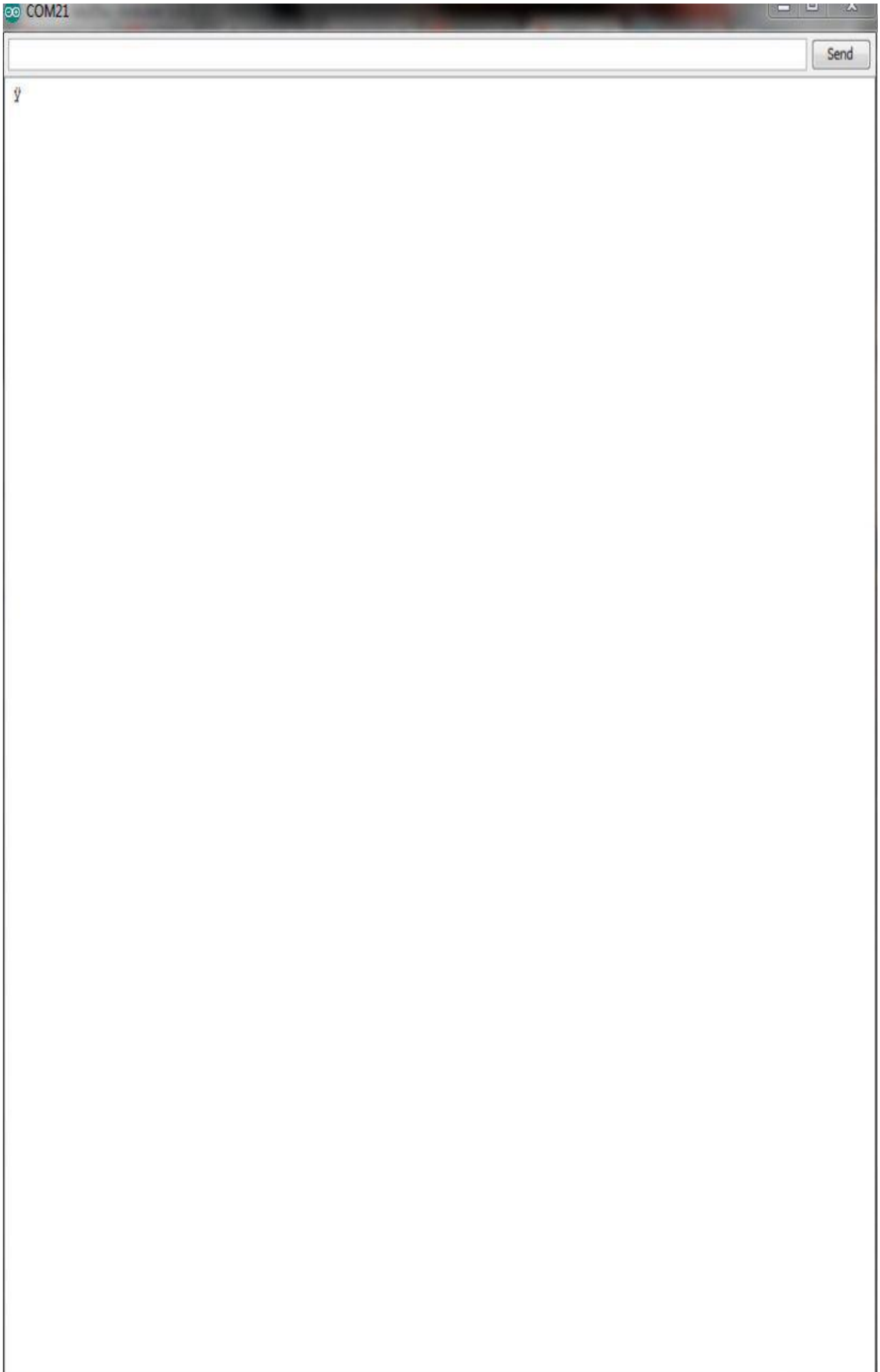
So open a Serial Terminal program. For this tutorial, I will use the Arduino IDE. Open the serial monitor in the Arduino IDE and set the baudrate to 115200. Finally, type a single character and hit send. You should see the same character you entered appear in the window.


COM21
c
c

Important! Please Read!

Make sure you enter the right baudrate. If you do not enter the correct baudrate, you will lose data. The figure below shows what happens if we set the serial program to

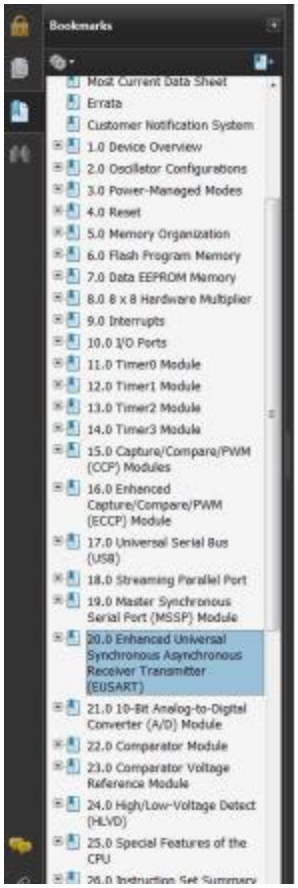
9600 baud rate even though the PIC is set to a baud rate of 115200.


COM21
Send
y

But how do you know if you set the USART to the right baud rate? Let's look at the following piece of code.

```
1    OpenUSART(USART_TX_INT_OFF & USART_RX_INT_OFF & USART_ASYNCH_MODE &USAR
     USART_CONT_RX & USART_BRGH_HIGH, 10);
```

As you can tell by the name, this is responsible for setting up the USART on the PIC. We're going to look at the last two parameters of this command. The last two parameters correspond to the baudrate of the USART. USART_BRGH_HIGH sets the USART to a high baudrate, but we can set it to a low baudrate by replacing USART_BRGH_HIGH with USART_BRGH_LOW. The final parameter is a spbrg value. Although there are a set of equations you can use to find this value, as long as you're using a 1Mhz, 2Mhz, 4Mhz, 10Mhz, 20Mhz, or 40Mhz crystal, you can find the SPBRG value in the datasheet. The figure below shows where we can find the spbrg value.



2.4	2.404	0.16	25	2403	-0.16	12	—	—	—		
9.6	8.929	-6.99	6	—	—	—	—	—	—		
19.2	20.833	8.51	2	—	—	—	—	—	—		
57.6	62.500	8.51	0	—	—	—	—	—	—		
115.2	62.500	-45.75	0	—	—	—	—	—	—		

BAUD RATE (K)	SYNC = 0, BRGH = 1, BRG16 = 0										
	Fosc = 40.000 MHz			Fosc = 20.000 MHz			Fosc = 10.000 MHz			Fosc = 8.000 M	
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error
0.3	—	—	—	—	—	—	—	—	—	—	—
1.2	—	—	—	—	—	—	—	—	—	—	—
2.4	—	—	—	—	—	—	2.441	1.73	255	2403	-0.16
9.6	9.766	1.73	255	9.615	0.16	129	9.615	0.16	64	9615	-0.16
19.2	19.231	0.16	129	19.231	0.16	64	19.531	1.73	31	19230	-0.16
57.6	58.140	0.94	42	56.818	-1.36	21	56.818	-1.36	10	55555	3.55
115.2	113.636	-1.36	21	113.636	-1.36	10	125.000	8.51	4	—	—

BAUD RATE (K)	SYNC = 0, BRGH = 1, BRG16 = 0								
	Fosc = 4.000 MHz			Fosc = 2.000 MHz			Fosc = 1.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	—	—	—	—	—	—	300	-0.16	207
1.2	1.202	0.16	207	1201	-0.16	103	1201	-0.16	51
2.4	2.404	0.16	103	2403	-0.16	51	2403	-0.16	25
9.6	9.615	0.16	25	9615	-0.16	12	—	—	—
19.2	19.231	0.16	12	—	—	—	—	—	—
57.6	62.500	8.51	3	—	—	—	—	—	—
115.2	125.000	8.51	1	—	—	—	—	—	—

Source: http://coolcapengineer.wordpress.com/2012/12/26/tutorials-usart-communication-for-pic18-microcontroller/