Interfacing an XBee series 1 with MSP430F5529

By Kyle Hale

This paper is going to cover the basic wiring and initializing of the XBEE for serial communication with one of the UART ports of the MSP430F5529. There are various other uses for the XBee (ADC, Mesh network, etc.), but these are uses that are out of the scope of this paper and various other resources can be found on the internet to help with these XBee configurations.[1] There are several tools that I would recommend in order to follow this tutorial.[2][3][4][5][6] Note that this example will provide MSP430 UART code to send a few basic ASCII characters with an XBee, but once configured any UART device can be used with the XBee.

Configuring XBee

The first thing that you need to do in order to configure the XBee is to open X-CTU and test your XBee to see if it can be correctly communicated with. If not try changing the Baud rate X-TCU, but this is the default values of the XBee that I am using shown in Figure 1.

![X-CTU interface](image)

**Figure 1** – This shows the various COM devices that I have open and one of them is the XBee. You can use the device manager to see what COM port is your XBee.
Figure 2 – After testing the XBee it should come back and show this message saying that it is successful.

After doing this click on the Modem Configuration tab and click the read button. This will show you the various settings and the current settings for the XBee. There are only a few settings that you must change in order for the two XBee to communicate. The figure below highlights the specific setting that need to be changed. In figure 3 there are four fields that you need to change. On both XBees you need to make the channel and PAN ID exactly the same. On one XBee you need to make Destination Address an arbitrary number within the given range and make the 16-bit Source Address another different arbitrary number within the given range. For instance, in XBee1 the DL field could be 11 the MY field could be 10 and in XBee2 the DL field should be 10 the MY field should be 11.

Figure 3 – In this figure the fields that need to be changed have squares around them.
In Figure 4 below the fields that are pointed out must be the same as you configuration for UART in your RX/TX module of your controller. In this case my Interface Data Rate (Baud Rate) is 9600 baud and the Parity is set to no parity bit. Once you complete these few steps for both XBee modules you should wirelessly be able to transmit using serial communication from one serial device to another. You can use a terminal program such as Putty or X-TCU has a built in terminal functionality that you can test your XBee’s with to see if they are properly configured before you try to use them with a controller. There is a basic code example provided for a MSP430F5529 sending the ASCII characters ‘A’ ‘B’ ‘C’ to the UART port and when wired to the XBee it will be seen at the other XBee module’s RX pin.

**Figure 4** – This shows fields that you have to make sure are consistent with the UART configuration in the MSP430.

**Pinout**

<table>
<thead>
<tr>
<th>MSP430F5529</th>
<th>XBee</th>
</tr>
</thead>
<tbody>
<tr>
<td>RX (Pin 41)</td>
<td>TX (Pin 2)</td>
</tr>
<tr>
<td>TX (Pin 40)</td>
<td>RX (Pin 3)</td>
</tr>
<tr>
<td></td>
<td>VCC (Pin 1)</td>
</tr>
<tr>
<td></td>
<td>GND (Pin 10)</td>
</tr>
</tbody>
</table>

**Table 1** – This table shows the connections that need to be made from the Xbee to the MSP430F5529.
#include <msp430f5529.h>

void main(void)
{
    WDTCTL = WDTPW + WDTHOLD;  // Stop WDT
    P3SEL |= BIT3+BIT4;         // P3.3,4 = USCI_A0 TXD/RXD
    UCA0CTL1 |= UCSWRST;       // Reset the state machine
    UCA0CTL1 |= UCSSEL_2;      // SMCLK
    UCA0BR0 = 0x68;            // 1MHz/9600=0x68 (see Data Sheet for calculations)
    UCA0BR1 = 0x00;
    UCA0MCTL |= UCBRS_1 + UCBRF_0;  // Modulation UCBRSx=1, UCBRFx=0
    UCA0CTL1 &= ~UCSWRST;      // Initialize the state machine
    UCA0IE |= UCRXIE;          // Enable USCI_A0 RX interrupt

    while (!(UCA0IFG&UCTXIFG)); // Is the USCI_A0 TX buffer ready?
        UCA0TXBUF = 0x41; // TX -> A
        UCA0TXBUF = 0x42; // TX -> B
        UCA0TXBUF = 0x43; // TX -> C

    __bis_SR_register(GIE);     // interrupts enabled
    __no_operation();
}

// This is the RX UART ISR and is entered when the RX buffer is full
#pragma vector=USCI_A0_VECTOR
__interrupt void USCI_A0_ISR(void)
{
    //Insert Code for the RX interrupt
}

**Parts Used**

[2] - 2 XBee Modules


[3] –1 XBee Explorer (To save money you can make your own PCB with a FTDI IC and some resistors)

http://www.sparkfun.com/products/8687
[4] – 1 MSP430F5529 (I used a target board found at the link below, but you can use the controller of your choice.)


**Software**

[5] X-CTU

http://www.digi.com/support/productdetail?pid=3352


**Resources**

XBee Data Sheet


MSP430F5529 Family Document


**Recommended Material**