580.471 Principles of Design of Biomedical Instrumentation **Fall 2008**

RS232 and Wireless Communication

In this lab, you will be learning some of the basics of serial and wireless communication. Previously, you learned some fundamental ways to interface with the PIC – analog input and digital input/output. These are especially useful when communicating simple commands, such as turning on a motor or reading a value. However when interfacing the PIC with more complicated devices, such as LCD's, Wireless Devices, or EEPROM's, a standard digital communication protocol is used. One of them, RS232 (AKA serial communication), will be employed in this lab.

RS232 is a communication protocol that utilizes a single communication wire to communicate digital data (in the form of bytes) from one device to another. Though now outdated (due to the advent of USB protocols) RS232 is still commonly used in hobby circuits because of its ease of implementation. With each byte of data communicated, 10 bits are sent, a "start" bit that warns the device of incoming data, 8 bits that represents the byte, and a "stop" bit. The advantage of this method of communication is its simplicity of implementation - all you need is a single wire, provided both devices have the same grounding reference. Sometimes an additional 'parity' bit is used to check for errors in the transmission process. For this lab we will be using serial communication protocol without parity but feel free to explore this option if you wish to.

We will not go into the details of serial communication but you should keep in mind the following points:

1. **Baud Rate:** It is essentially the speed of communication and needs to be configured to be the same for both the devices. For example, when one device transmits at 9600 bps (bits per second), the other device needs to read at 9600 bps. 9600 bps usually means 960 'data-bytes' per second. (What's the difference between a normal byte and a serial communication data-type?) You must keep the baud rate in mind when setting your transmission rate requirements.
2. Each device has specific requirements on what is appropriate in terms of **voltage levels**
    i. Many devices, such as RFID readers, PIC chips, and wireless communication require TTL/CMOS signals - a high of +5V and a low of 0V.
    ii. A computer's serial port, however, requires a high of +10V and a low of -10V (this is part of the RS232 standard), yet your PIC can output only 0V to +5V. To accommodate these discrepancies, a MAX232 chip is used to convert input and output voltages.

In this lab, you will be using RS232 to interface with an XBEE chip in order to wirelessly communicate your sensor data to the serial port (COM port) on your computer. Your PIC will interface with your sensor circuits, and your transmitter XBEE. The receiver XBEE will interface with your serial port via a MAX232. Once your data is sent to the computer, it will be displayed on Hyper-terminal, a program that is specifically designed to view serial port information. A hyper-terminal can only display ASCII information. Therefore, you need to send your signal data in the form of ASCII characters (ex. instead of sending the value 2 (0x02), you need to send the ASCII value "2" (0x32).

Procedure

Note: This procedure is tentative and subject to change! If a change occurs a new procedure sheet will be handed to you during your laboratory session. Any changes to the procedure will not affect your pre-lab circuit design or questions.

Part 1 – Programming the PIC to incorporate Serial Communication

- Program your PIC to take in sensor information and dump that data onto your RS232 output pin (9600 baud rate, no parity, 1 start and stop bit). Make sure you are outputting ASCII data (instead of standard numerical data that you read from your sensors - you will need ASCII to display your output on Hyper-terminal in part 3). Be careful with how often you send your sensor data to the RS232 pin. If too fast, you won't be able to view your data on the computer (a good refresh rate is around 2-3 Hz). Please look through the appendix for sample code on sending RS232 information.

Part 2 - Display the sensor output to the computer

- In order to convert your PIC output into input that your computer can read, you need a MAX232 chip. Interface your PIC with a MAX-232 chip.
- If you are at this step, ask your TA's for the serial port cable. Make note of the output (red) and ground (black) wire when connecting it to your MAX232. Once connected to the computer, open hyper-terminal by going to
  Start > Programs > Accessories > Communication > Hyper-terminal
- Enter **"Lab 6"** in Name
- Select **"COM 1"** under "Connect using:"
- Select 9600 under "bits per second", 8 data bits, no parity bit, 1 stop bit, and no flow control.
- Start sending data and observe your output on hyper-terminal. Did you remember to output ASCII data in Part 3? If not, you will probably see junk on Hyper-terminal as it can only display ASCII values.
- **IMPORTANT FOR POSTLAB:** Send a biomedical pulse through your sensor (pressing on your force sensor, pressurizing your pressure sensor, waving your finger across the distance sensor) and record your output through hyper-terminal (be careful with the refresh rate of your sensor data - if too fast, you might miss the transition on hyper-terminal!). Save your hyper-terminal data (a screenshot is fine) for your post-lab.

Part 3 – Incorporating XBEE chips into this circuit

- If you read the data sheet of the XBEE you will realize that something else needs to be done in preparation for the incorporation of the XBEE wireless chips in your circuit. What is it that needs to be done?
- Ask your TA if you are not sure about what needs to be done here in lab and do not move on before checking in with a TA at this point.
- You should have now received 2 XBEE's – one for transmission and the other for reception. The XBEE chips that will be given to you have been pre-programming to communicate with each other alone and so you should see no 'cross-talk' in the lab.

- Disconnect the PIC from the MAX-233 and connect it to the transmitter XBEE. Once again, you cannot connect the PIC output pin to the XBEE directly. Why? What should you do to get around this problem?
- Once the transmission is ready, see if the powered receiver chip is getting the wirelessly transmitted data by probing the receiver XBEE's pins and observing the output on the oscilloscope
- Now connect the receiver to the MAX-232 and send the output to the computer hyper-terminal just like you did in the first phase. Did you notice anything different this time around? If so, try to explain the reason for the discrepancy.

If you have reached the end of this lab successfully without running into too many problems, well done! You are all set to begin your challenge projects!

Pre-lab [35 points]

Please Note: Since you will need the corrections in circuit schematics for your post lab write up so points are not taken off twice, we will grade these and leave them in the BME office Clark 318 for you to pick up beginning Monday Morning at 10am. You can pick these up when you drop off the challenge project conceptual design document the same day.

1. Look through the Datasheet for the XBEE and draw a schematic for interfacing the transmitter XBEE with a PIC microchip. Remember to indicate supply voltages and ground etc on your circuit diagram. Also, assume that your circuit is going to be powered by an unconventional newly designed 5V battery. Your design should be such that if replicated in its entirety and powered up it will work! **[5 + 5 = 10]**

2. Look through the Datasheet for the MAX232 and draw a schematic for interfacing the receiver XBEE to the computer. Make sure you indicate the input and output signal line. What kind of connector would you use to connect your circuit to the computer's port? Make sure this is drawn and fully labeled on your schematic. Again, draw your schematic in a way that if someone were to build it and hook it up to power it would work. **[5 + 3 + 2 =10]**
Note: For **Integrated Chips (IC's)** like the PIC or MAX232, it is essential you mark pin numbers on your circuit diagram so that the person reading the schematic understands what's going on. This becomes especially useful when certain pins have specific pre-determined roles (like Clock Input to the PIC), while designing Printed Circuit Boards (PCB's) for your projects and when the design becomes complicated due to a large number of components.

3. What is meant by Baud Rate of a communication device? If the Baud Rate for the device that sends a serial signal with no parity is 19200, then what is the pulse width? What is the pulse width if the signal is transmitted at a baud rate 38400 instead? **[1 + 2 + 2 = 5]**

4. Draw an oscilloscope trace as seen from the PIC pin when it is transmitting the ASCII byte "A" using an RS232 protocol. Indicate the different portions of this signal, voltage levels and pulse widths. Assume no parity and a 9600 baud rate. **[5]**

5. Why can't you transmit signals from the XBEE directly to the computer? In other words, what purpose does the MAX-232 chip solve? **Hint:** You should ideally mention the voltages that are being manipulated here. **[5]**

Post-Lab [50 points]

1. Draw your entire circuit schematic including the sensor circuit from last week that you interfaced with the Wireless Chips. Clearly separate out the transmission and receiving stations. Make sure all components are labeled and all power sources indicated. **[5 + 5 + 10 = 20]**

2. During the lab you got some meaningful output on the Hyper-terminal. Attach a screen shot of this output and indicate what was done to the sensor side of the circuit to obtain this? **[5]**

3. What modifications did you need to make to your PIC code in order to transmit the signal? For your answer include the main commands that you included in your code for serial communication. **[5]**

4. You are transmitting wireless information from multiple sensors (say from an EEG array) through a single XBEE to a central receiving hub. Assuming all sensors get equal priority describes an algorithm in the form of a flowchart or bullet points as to how a PIC on the receiving end will interpret this information. As a second scenario assume some sensors need to read at a higher rate than others (a priority or preference order if you would like to call it that). Come up with a new algorithm for this kind of wireless signal reception for the PIC. Explain how this is more/less efficient than your first algorithm Note, you do NOT have to write PIC code for this part, just simple algorithms in bullet points would suffice. This is in preparation for your challenge and honor's projects. **[5 + 5 = 10]**

5. **Wireless Patient Protection** (Open Ended): You were introduced to patient protection circuits in one of the earlier labs. Let's say you are the designer of a new medical device and the FDA has asked you to revise your design to make the patient protection module better. You came up the idea of using some sort of a wireless patient protection circuit that controls the voltage and prevents it from going over a certain maximum using a photodiode and photo-transistor. Build a circuit with these components to achieve your goal. Explain very briefly what this circuit does? [**Note:** This has nothing to do with serial communication and the XBEE; it's just another kind of wireless capability you might want to add to the devices you design as a future biomedical engineer.] **[5]**

6. **Wireless Power Transmission** (Open Ended)**:** Your lab deals with wireless data transmission which is a well established science at this point. More recent efforts are being directed towards wireless power transmission. Once fully developed and commercialized this could potentially find interesting applications in the field of biomedical engineering. With the help of simple diagrams and equations explain the principles employed in wireless power transmission. You can look this up easily on the internet. We are not expecting a review paper on the subject but just an understanding of the basic principles involved. **[5]**

Bonus Question: PIC-XBEE Programming Exercise [10 points]

The XBEE is a very powerful wireless chip. You have just learned to operate it in the simplest way possible. If you read the data sheet of the XBEE module you will see that it can be programmed to achieve different functions like talking to specific chips. We actually pre-programmed your chips before the lab in order to avoid too much ’cross-talk' during the laboratory session. Let us now suppose you are taking in-vitro recordings of electrical signals from a cardiac myocyte and need a higher data transmission rate. Write a PIC C program that programs the XBEE chip to change its baud rate to 115200.

Appendix:

```
#include <16F877.h>
#include <stdio.h>
#include <stdlib.h>
#define TXBEE PIN_B6
#define RXBEE PIN_B7
#define TXLCD PIN_B5
#use delay(clock=20000000)

//#use rs232(baud=9600, XMIT=TXBEE, RCV = RXBEE, parity=N, bits=8)
//char timed_getc() //code for accepting a single byte of data every 0.5 seconds (not used in
this lab)
//{
// //this code allows you to poll the RS232 pin on your PIC without going into an infinite
// //loop if no data is sent.
// long timeout;
// timeout=0;
// while(!kbhit()&&(++timeout<50000)) // 0.5 second
//  delay_us(10);
// if(kbhit())
//  return(getc());
// else {
//  return(0);
// }
//}
    output_toggle()
   {
      while(true)
      {
         output_toggle( PIN_B1 );
         delay_ms(1000);
         output_toggle( PIN_B1 );
         delay_ms(1000);
      }
   }
//code to talk to both an XBEE and an LCD at the same time
    void main()
   {
      int8 bottles;
      bottles = 99;

      while(TRUE) //We always want the program to run continuous until we turn off power
      {
      #use rs232(baud=9600, XMIT=TXBEE, RCV = RXBEE, parity=N, bits=8)
      //sets the current RS232 to the XBEE

         printf("%u bottles of rootbeer on the wall",bottles);
         putc(13); //13 refers to a carriage return, which allows you
      //to skip to the next line

      #use rs232(baud=9600, XMIT=TXLCD, parity=N, bits=8)
      //sets the current RS232 to the LCD

         printf("hello world");
         delay_ms(5000);
      }
   }
```