Microprocessor-Based Systems (E155)

J. Spjut and M. Spencer **Fall 2014**

Lab 6: Wireless Communication

Requirements

Build a wireless calculator. Use a PIC linked by Bluetooth to a PC. The PIC should perform the computations, while the PC provides the keyboard and display. The calculator should at least be capable of addition of 3-digit positive decimal numbers. You are free to include support for other operations and number formats if desired. Write the PIC code in C. Use a BlueSMiRF wireless module connected to the PIC's UART and a Bluetooth dongle connected to the PC's USB port. Use HyperTerm as the PC terminal client.

For example, the user should be able to type

```
173 + 349
```

into HyperTerm, then press Enter and observe

```
522
```

on the HyperTerm console.
Alternatively, you may build an RPN calculator that accepts inputs such as

```
173 349 +
```

The calculator should provide reasonable behavior for any set of inputs. This includes remaining functional under any set of inputs and displaying error and/or warning messages when unsupported/unrecongnized inputs are entered. CLARIFY FURTHER? 10 20 + 30, A+1, +1, 1+, HANDLING NO AND MULTIPLE SPACES BETWEEN NUM & OPERATOR

C Compiler

To learn to use the C compiler, examine the Hitchiker's Guide document handed out in class. Also, browse the Getting Started and Libraries manuals for the C32 compiler (on the web site).

In the Memory window, you can see the actual assembly instructions that run on the PIC, produced by the C compiler. Open the Program Memory section and look at it to see

how the code generation works. You should understand the assembly language generated from your C code. If any of your C code produces unusually lengthy or ugly assembly language, consider simplifying.

Do not use the custom boot code and linker script from the assembly labs, as the C compiler output needs the full functionality of the default init routines to run properly.

Bluetooth Link

The Bluetooth link consists of an Abe USB Bluetooth dongle on the PC and the BlueSMiRF module on your breadboard. The dongle emulates a serial port on the PC and the BlueSMiRF uses a serial link to the PIC. As with the speakers in Lab 5, there are a limited number of USB and BlueSMiRF modules, so *please leave them both in the lab when you leave.*

Insert the USB Bluetooth module to a USB port on your PC. Windows should automatically recognize it.

Mount your BlueSMiRF module on your breadboard. The pinout is labeled on the underside of the module. Connect 3.3 V power and ground, and connect CTS to RTS so that the module performs its own handshaking. Once the module is powered, you should see a blinking red STAT LED indicating that the module is looking for a connection. If you turn on power and the red LED does not turn on, leave the power on, remove the BlueSMiRF module from the breadboard and reinsert into the same place. The LED should now be blinking. If not, try another module.

Now, double click on the Bluetooth icon in your system tray. In the Devices tab, click Add to make a connection to the BlueSMiRF. The Add Bluetooth Device Wizard will open. Check the box to indicate that the device (the BlueSMiRF) is set up and ready to be found, then click Next. The Wizard will search for several seconds and find one or more devices in the room. Your BlueSMiRF is named FireFly-XXXX, where XXXX are the last four hexadecimal digits of the MAC number printed on the BlueSMiRF; be sure to talk to the correct device if several are in use in the lab.) When asked for a passkey, select “Use Pass Key Found in Documentation” and enter the default passkey of ‘1234’. Do not change this passkey! The Wizard may take a while to add the device. Click on the COM Ports tab and make note of the outgoing COM port to which the device is assigned (e.g. COM11).

HyperTerm is a terminal client that is often used to run a remote session on a distant computer. For our purposes, we will be using HyperTerm to communicate with our BlueSMiRF Bluetooth module over the COM port mentioned before. The COM port uses the following settings:

Speed:	9600 baud
Data bits:	8
Stop bits:	1
Parity:	none

Flow control: none

HyperTerm automatically recognizes these values, so you do not need to change them.

HyperTerm can be found under Accessories • Communications from the Windows Start Menu. Under File • Properties • Settings, choose ASCII Setup and check "Echo typed characters locally" so that the keys you press appear in the terminal as you type them.

When you begin typing in HyperTerm, the red status LED on the BlueSMiRF should stop blinking and the green connection LED should turn ON.

The BlueSMiRF connects to the PIC over another serial link running at 3.3 V (not the RS-232 standard, but more convenient). The BlueSMiRF expects the following settings:

Speed:	115.2k baud
Data bits:	8
Stop bits:	1
Parity:	none
Flow control:	none

Each PIC32MX675 has six UARTs. UART2 uses pins RF4 and RF5 for RX and TX respectively. They should connect to the opposite pin on the BlueSMiRF module (TX to RX, and vice versa.) Configure the UART of your choice to have the correct settings.

Hints

- You may want to configure MPLAB to automatically program the PIC after a successful build. To do this, choose Debugger • Settings • Program and check "Program after successful build."

- You may want to configure MPLAB to begin debugging at the start of your C `main()` function rather than at the start of the initialization code. To do this, choose Configure • Settings • Debugger and check "Reset device to the beginning of main function."

- Once the serial port is initialized, you can print to the HyperTerm console over the serial port using the `printf` statement in the `stdio` library.

- The Enter key corresponds to a carriage return character called `'\r'` in C (ASCII code 0x0D).

- If you want to advance to the next line when printing, send both the `'\n '` and `'\r '` (new line and carriage return) characters.

- You may wish to build a set of functions to receive data from the serial port. Two such functions might have the following characteristics:

```
char getcharserial(void)
```

```
// waits to receive a character from the serial port, then returns it

void getstrserial(char *buf)
// receives a string over the serial port by calling getcharserial
// until a carriage return is received
```

- If you are having trouble determining if you are receiving data from the serial port, consider writing each character to the LEDs as it comes in.

Credits

This lab was originally developed in 2008 by Sam Gordon `09 and Trevor Ashley `09.