ECE 354 – Lab 1: Serial Interface to a Terminal

In this lab assignment you will learn how to interface a Microchip PIC 16F877 to a terminal which contains a keyboard and a CRT display. The amount of hardware required to complete the lab will be fairly small (a 16F877, SP233A transceiver, a clock "can", and a few resistors and capacitors), but there will be a substantial amount of software to write and debug.

A discussion of the assignment and a recommended procedure for completing the work is given below. If you have additional questions, please ask the TAs. Please get started on the assignment early. Review the lecture material and remember that there is a online quiz 2/9/04-2/11/04.

Tasks

For this lab you should complete the following tasks.

1. Three single-bit data switches from your lab kit should be connected as input to the lower three data bits of 16F877 external port A.
2. Bits RB1 through RB3 of 16F877 external port B should drive LEDs located on your lab kit. Initially, these bits should be set to all zero (e.g. 000). Later, they will reflect the value of the inputs on port A.
3. The serial data output from the VT220 on your lab bench should be connected to the RX pin on the 16F877 via a MAX232 transceiver. This data pin serves as input for the 16F877 UART.
4. Software running on the 16F877 should obtain characters from the RECEIVE port of the UART using UART asynchronous mode.
5. After reset, the 16F877 should send the message “Number?” to the VT220. The user sends a response by pressing a character on the VT220 keyboard. If a character other than ‘0’-‘7’ is pressed the prompt should be resent. Individual characters can be sent from the 16F877 to the VT220 CRT via the UART on the 16F877 in asynchronous mode. Output pin TX drives the VT220 CRT with serial output via a MAX232 transceiver. The value of the character pressed should appear on the VT220 screen next to the ‘Number?’ prompt and a single blank space.
6. Once a number is sent, the value should be compared to the three-bit input on port A. If the value inputted by the user matches the value of port A, a message “equal” should be sent to the terminal. If the values do not match, a message “not equal” should be sent. The value of port A should be echoed to port B (e.g. an input of ‘001’ should light up the lowest-order LED).
7. Following step 6, each change of a switch attached to a port A input should trigger a repeat of the actions of step 6. That is, with every change of the switches a new equal/not equal message should be sent.
8. To successfully complete the lab, your hardware should run in "standalone" mode. In other words, there should be a connection from a push button to the MCLR signal on the 16F877 and the 16F877 should be free of the MPLAB-ICD emulator. After reset, the 16F877 should jump to code that sends the start prompt to the terminal, and then waits from input from the terminal keyboard. Note that the 16F877 should sample port A often since changes in the switch setting should immediately be reflected in the LEDs attached to Port B after the ‘Y’ is pressed.

Procedure

The best way to complete the tasks listed above is through a divide-and-conquer approach. If you have completed the introductory lab exercise, you have learned that the MPLAB-ICD can be very helpful in identifying and fixing design bugs. I also highly recommend the use of MPSIM to test the functionality of your software.

Step 1: System start-up / Writing to Port B

After reviewing the lecture notes for the lab and reading the assigned sections in the data sheet and Peatman, I suggest the following simple task: write a three bit value to Port B to illuminate three LEDs. As discussed in lecture, consider the issues that must be addressed to accomplish this [setting the Reset vector at 0000h to jump to the start of code, configuring the data direction register (TRISB), and writing an three-bit value to the Port B data register (PORTB), address 0006h]. Consider using the MPLAB-ICD for single-stepping to test your hardware in addition to programming your 16F877.

Step 2: Reading from Port A

Once you have verified that your 16F877 starts up correctly from reset and is writing a value to Port B that you have hard-coded in the program, change it to echo the three data bits from Port A to the three LEDs attached to Port B. This requires the configuration of Port A using the TRISA and ADCON1 registers.

Step 3: Configuring the UART

As mentioned in lecture, several registers must be configured in order to get the UART working correctly. These registers include TXSTA, RCSTA, and SPBRG. Before attempting to read or write an eight-bit data value from/to the UART, make sure these registers have been set to the correct value. Pay particular attention to setting the baud rate generator correctly for 9600 baud communication. Check the lecture notes, Chapter 11 of Peatman, and the 16F877 data sheet for more detail on setting the baud rate generator.

Step 4: Reading a character from the terminal keyboard and writing it to the terminal display

As an initial step to verify the correct operation of the UART, it is desirable to program the 16F877 to READ a character from the terminal keyboard and then immediately send the character back to the terminal display (CRT) unchanged. If you run into difficulty in performing this test, consider hard-coding a character into your code and transmitting it the display before attempting to read a character. Note that for this assignment you will be using polling instead of interrupts to determine the status of the receive and transmit buffers. This indicates that your code should repetitively check the full/empty bits in register PIR1 for transmission/receipt of UART data rather than using interrupts to indicate when the resources are free. To convert ASCII values received from the terminal to numbers, use a simple calculation to determine the value of the character (don’t write a if statement with 8 different cases).

Step 5: Putting it all together

I suggest an incremental approach to completing the lab. One solution is to have one partner focus on the hardware setup including Ports A and B while the other partner writes software to configure and use the USART. The MPLAB-ICD can be used to single-step the 16F877 if you run into difficulty.

Things to keep in mind

Often it may appear that hardware is not working the way that is expected. An initial reaction would be to assume that the hardware is broken and needs to be replaced. Before going to find the TA keep the following questions in mind:

- Are all the inputs on the digital components driven by some value? Often, floating signals will cause chips, such as the 16F877, to operate incorrectly.
- Is there a simple test case I can try to verify functionality? Simply coding the entire lab at once or wiring the entire kit at once will likely lead to frustration. Instead, plan for simple tests that verify the functionality of specific components and software segments.
- Is the chip plugged into the board correctly? Please be careful plugging the MPLAB-ICD, the 16F877, and other components into the breadboard. Aligning the parts incorrectly will lead to broken pins and hours of debugging. Carefully plug and unplug each component from the breadboard.

Grading

Quiz

For the pre-demo quiz I will ask each student individually a few general questions about the main concepts used in the lab (e.g., UART operation and basic PIC properties) as well as issues encountered during implementation. It is expected that each team member has an understanding of all components of the lab.

Demonstration

The demonstration should show communication between the 16F877 and the terminal via the MAX232 part. The specific tasks to be presented are listed above in the "Tasks" section. The 16F877 and MAX232 part should be populated on the breadboard along with a 4 MHz clock "can" which is used to drive the 16F877 clock. It is OK to use the MPLAB-ICD for debugging but the 16F877 should be populated separately on the board for final check-off.

Lab Report

In addition to the demo a complete final report for each two-person lab group has to be turned in. Specific sections in the report include:

- **Introduction**: In your own words tell me what this lab accomplishes and what was learned. This one-page intro should summarize what is to follow in the remainder of the report. Please DO NOT simply clip portions of the lab description from the course web pages to fill the Introduction. A little thought and words that summarize the lab often brings the lab objectives into better focus.

- **Lab Summary**: Briefly summarize the procedure that was followed in completing the lab. Which experiments did you try first? What would you have done differently? What was the result that you found?
- **Problems Encountered/Solutions**: Please indicate problems you encountered completing the lab assignment and the specific approaches that you developed to solve it.
- **Hardware Schematics**: A full schematics expected including details such as pin numbers, resistor values, and capacitor values included on the diagrams for all components used. Good documentation is extremely important for hardware projects.
- **Discussion of Code**: Please lead the reader through the various code segments you wrote. Give a brief summary in words of each module and subroutine.
- **Reference List**: Provide a full listing of any manuals or books that you used. Please include the year of publication, author (if listed), publisher, and full name of the publication.
- **Documented Code**: Please include all assembly code you wrote for the lab. Each *significant* line of code should be commented.

The overall lab report should be 5-8 pages long (not including the schematics and code). Please type the report – you may draw the schematics by hand (using a ruler ☺) if you prefer.