ECE-3120
Spring 2008

LAB 8 – Parallel I/O, part 1

The purpose of this lab is to learn the use of the 68HCS12 parallel ports to interface to simple devices, in a program that counts on the LEDs and 7-segments.

PRE-LAB:

Prepare pseudocode and the first draft of the program. This must be completed before coming to the lab and shown to the lab instructor at the start of the lab session. Note: The Pre-Lab must be typed into a proper *.ASM source file, following our standard Program Format requirements.

Approved: Lab TA _____ *Date* _____

PROGRAMMING ASSIGNMENT:

Write a fully-commented program for the Dragon12 board, following our standard Program Format requirements, including appropriate directives and labels for memory operands and constants, called **LEDCount.asm**. The program should do the following:

- This program will run forever in a loop, as most real embedded systems do. So run the program continuously and never exit, except when the reset button is pressed or power is cycled off/on.
- Main operation: continually count from 0 to 255, displaying the count in binary on the LEDs and in Hex on the two right-most 7-segment Digits. Increment the count every 500ms.
- The LEDs and 7-segment Digits must be time multiplexed, with refresh rates such that no flicker is visible. (Two devices must be multiplexed.)
- Initially reset the counter to zero.
- Include and use the standard files, HCS12.INC and DELAY.ASM, as appropriate.
- Follow the overall program organization that was recommended in class.
- As always, the code must start at $1000.

The program may use directives to create data in memory locations beginning at $1200 as needed. Clearly define all data in your code!

Procedure: First, use D-Bug12 to fill memory locations $1000 through $3FFF with zeros. Then assemble, download, and debug/execute the program as follows.

a. Set breakpoints at the end of all major steps, as necessary to debug your program, pausing at each breakpoint to display the important values. Verify that each value is correct at each breakpoint and the final results are correct at the end of the program. Use the listing file to determine the breakpoint address. TIME is a very important factor in the operation of this program, so using breakpoints will definitely mess up all your program timing plans. You will have to keep this in mind and work around this problem to verify the logic of your program.
b. Then reset the processor (which also removes the breakpoint), download the program again, run it at full speed, and verify that the program runs properly under all conditions.
c. When finished debugging and executing, copy the entire terminal window output and paste it into a Notepad or Word document for inclusion in the report. You should edit out mistakes and unnecessary repetitions before submission. Good hand-written comments and explanations are ESSENTIAL to make this meaningful to the grader.

Approved: Lab TA _____ *Date* _____

Things to turn in as your Lab Report, attached in this order:

1. This assignment sheet, with your name at the top, signed by the TA where shown.

2. Your uncorrected pre-lab document (commented source code).

3. A printout of the final **LEDCount.asm** and **LEDCount.lst** files. You'll need to print the listing file in landscape mode to make it fit. Use Notepad to print them.

4. A printout from the terminal screen (method: highlight text, type ctrl+c, and paste into a Notepad or Word document, using Courier New as the font) that includes everything done. Add brief hand-written comments and highlight important values at each step in the procedure, showing the relevant memory contents and register contents. Good hand-written comments and explanations are ESSENTIAL to make this meaningful to the grader.

5. Answer the following questions, in your document:

    1) On our Dragon12 board, the LEDs and all four 7-segment digits are all driven by the same data port (B). (a) Why is the circuit designed this way? (b) How else could the circuit be designed, to simplify the programming required? If we only show data on the LEDs it is not a problem. (c) But explain how a program can show different values on the LEDs and each of the four digits simultaneously (in text-only style, no code).
    2) What would need to change if we had one more 7-segment? Give details.