ECE 447 Lab Assignment #1: Counter Display (15 points)
Assigned week of 9/21/2009.

Due Dates:

1. A simulation demonstration of the assignment and your code is due before 10:00 PM of your lab section **1 week** after it is assigned. Zip all your code and submit it in Blackboard.

2. A hardware demonstration of the assignment and your code is due before 10:00 PM of your lab section **2 weeks** after it is assigned. Zip all your code and submit it in Blackboard.

3. The lab report is due in Blackboard by 9:00 PM the day following the hardware demonstration.

Objectives:

The student learns to use the IAR IDE to code, simulate, download, and debug a simple timer application. The student also learns to prototype external display devices.

Assignment:

Implement a button-controlled one-second timer and display the timer value on two seven segment displays. When your program first starts, the timer value is 0 seconds. Two buttons controls the timer. Pushing S1 starts the timer; pushing it again stops the timer. Thus button S1 toggles the enable-state of the timer. The timer value must always be displayed regardless of the timer enable-state. Pushing S2 changes the count direction (if previously counting up, then pressing S2 causes the timer to count down.) Thus button S2 toggles the direction-state of the timer. When the timer value reaches 99 or 0 seconds, the timer direction-state automatically toggles. For the purpose of this lab, we'll term this counter behavior at the counter boundaries as "bouncing" as opposed to "rolling over."

Software Requirements:

1. Do NOT use a header file and #define all constants where appropriate (i.e. minimize magic numbers in the code body.)
2. Direction and Enable-states transition on rising edge of a button event.

Hardware Requirements:

1. Both displays must be buffered with (2) 74HC244, HC245, HC373, or HC374s.
2. Use the following port mapping:
   Control the ones-digit with port P2: Seg A to P2.0, Seg B to P2.1, etc.
   Control the tens-digit with port P6: Seg A to P6.0, Seg B to P6.1, etc.

Bonus:

1. (1.0 points) Add an extra button, S3, to port P2.7, P6.7, or another available port. S3 controls the behavior of the counter when it reaches a boundary. When the program first starts, the counter bounces as described above. On the rising edge of S3, the behavior toggles to rolling-over. Thus, button S3 controls the counter-boundary-behavior-state.
2. (0.5 points) Add a #define switch that determines the display radix – either 10 (required from above) or 16. The radix-16 counter also changes the maximum counter and display value of 0xFF.

Additional Report Requirements:

1. Show the equation you used to calculate the resistor value(s) for your circuit.
2. Show the maximum current draw for the segments of your 7-segment displays.
3. Describe the differences in your simulation-code and your hardware-code.
4. Answer the following questions:

a. Which of the four buffers did you use, and why?
b. Is the buffer you are using sinking current or sourcing current? Do you have a choice? Why?
c. What is the maximum current capability of the buffer(s) you are using when it is sinking current and sourcing current?
d. What are the pros and cons of using a resistor per LED segment?
e. What are the pros and cons of using one resistor for each display rather than per segment?
f. Does pushing the button(s) produce erratic results? If so, describe the behavior and make a hypothesis on the cause.