CSCE 236 Embedded Systems, Spring 2013
Homework 3

Started: Tuesday, January 29, 2013
Due: Beginning of class Thursday, February 7, 2013

Instructions: This homework is an individual assignment, collaboration is not allowed. If you discuss any problems with others, please note this on the assignment as described in the syllabus. Also note any materials outside of lecture notes, course textbooks, and datasheets that you used. Show your work and describe your reasoning to get partial credit if your solution is incorrect. This homework is due on the date listed above before the start of class.

Name:

Problem 1 (5pts)**.** *(To be completed at end of assignment) Approximately how much time did the total assignment take? Which problem took longest and how much time did it take?*

Problem 2. *Timing operations. You may want to refer to Section 5 of Lab 1 for parts of this problem. In other words, you can simply repeat each operation until you can accurately time them using a stop watch and a blinking LED. There are other ways to do this, but this is sufficintly accurate for these problems (you are welcome to try other better ways).*

a) (10pts)**.** *How long does each iteration of a basic for loop, shown below take? What about when the loop counter is a* `uint16_t` *and* `uint32_t`*? Answer in micro seconds. Describe how you came up with your answer.*

```
for(uint8_t i = 0; i < 255; i++){
  asm volatile("nop");
}
```

b) (5pts)**.** *How many CPU clock cycles does the each iteration of the loop take for* `uint8_t`, `uint16_t`, *and* `uint32_t` *types?*

c) (5pts). *Now insert a multiplication command of a variable times an integer constant into the body of this loop for each of the three types. Note, you may need to declare your variable as* `volatile` *so the compiler does not optimize it out. How many clock cycles does a multiplication take based on your measurements? Include the relevant line of code here and indicate if this time is just for the multiplication or if other operations are included in this timing (e.g. memory operations).*

d) (5pts). *Now repeat the above, except multiply by a floating point number (e.g. 3.141) instead of an integer value. How many clock cycles does this take?*

e) (5pts). *Examine the assembly code generated for the above code using the* `avr-objdump` *command discussed in class. What assembly instructions or functions are actually called to perform the two different types of multiplication?*

Problem 3. *Input*

a) (10pts). *In Lab 1, you configured the button by using a pullup resistor. However, as you learned in class, the input pins also have the ability to activate an internal pullup resistor. Write the* `C` *code below to activate the internal pullup resistor of pin* `PC2`. **Instructor sign off required:** *You should also implement this and show the instructor the functionality of using your button in Problem 4 without the resistor.*

b) (5pts). *Using the internal pullup means you can remove resistor R4 in Figure 3, Lab 1. What happens if you do not have this resistor and do not activate the internal pullup?*

Problem 4. *For this problem you should complete the sections in* `morse.c` *where STUDENT CODE is indicated. To do this problem, you will need to include* `morse.c` *and* `morse.h` *in your sketch (use the menu* `Sketch->Add File`*. To call functions from* `morse.c`*, you will need to put* `#include "morse.h"` *in your main sketch file.*

Complete the `morse.c` *code so you will be able to send Morse code blinks. You should make sure that you are able to specify any of* `LED_RED`*,* `LED_GREEN`*, or* `LED_BLUE` *as the LED to output blinks or any combination of them (e.g.* `LED_BLUE | LED_RED`*). I would recommend creating helper functions that turn on or off LEDs so if you switch the pin that controls the LEDs, you only have to change code in one or two places. All of the code described here should be in a single program that runs at the same time.*

You must also turn in your code for all parts of this problem by visiting `http://cse.unl.edu/~cse236/handin/`. *Failing to electronically turn in your code will result in a 10 point penalty on this assignment. Points may also be deducted for coding errors, poor style, or poor commenting.* Note, you do not need to turn in a printout of your code for this assignment.

a) (5pts). *In* `morse.c`*, the Morse blink pattern (dots and dashes) for each character are stored in a single byte. Read the code and describe how this is done and what the meaning of each bit is.*

b) (10pts). *Now write the* `C` *code to turn on and off the LEDs by setting the registers (e.g. DDRx, etc.). What pins did you connect the LEDs to? How did you configure these pins as output? How do you turn the LED on and off? Make sure to include the relevant code here.*

c) (10pts). *Write code that will blink "Hi W" when the board starts with "Hi" blinked with the red LED and "W" blinked using the red and blue together.* **Instructor sign off required, no written answer needed.**

d) (15pts). *Now implement code that will output "d" (for dot), "D" (for dash), or "s" (for a long pause) over the serial port depending on how long the button is pressed. Use fixed times for your dot, dash, and long pauses (e.g. anything press under a second is a dot and anything over a second is a dash and any pause over a second is a long pause).* **Describe** *how you implemented this and how you watched out for bouncy buttons.* **Instructor sign off and written answer required.**

e) (10pts). *Finally, implement code that will turn on the Red, Green, or Blue led if the Morse code for 'r', 'g', or 'b' is entered, respectively. Keep the LED on for approximately 1 second and then turn it back off.* **Instructor sign off required, no written answer needed.**

Do not forget to fill in the amount of time you spent on this assignment in Question 1.