CSCE 236 Embedded Systems, Spring 2012
Lab 2

Thursday, February 2, 2012

Names of Group Members:

1 Instructions

This is a group assignment to work on during class. You only need to hand in one copy of this, but make sure that the names of all of your group members are on this sheet to receive credit. Complete all of the sections below *during class* and make sure to get the instructor or TA to sign off where required. You may want to keep your own notes on what you complete since parts of future homework will build on this lab.

2 Detecting Button Bounces

In class, you saw how to use `Timer1`, configured with pin `T1` as the button input to count the number of times the button was pressed. Download this code from the course website (in the assignments section next to lab2). Examine this code and determine which pins the LEDs and button should be connected to. Connect everything and verify the functionality of the code by pressing the button and observing the counter output on the serial console.

As you saw in class, occasionally the counter incremented more than once for a single press due to the button "bouncing." Bouncing occurs when the button is being pressed and it rapidly oscillates between logical zero and one. In this section, you will implement two different ways to detect this.

2.1 Manual Delay

First, modify the code so that it only prints the value of the register `TCNT1` when it changes. Be aware that TCNT1 may change at any point, so you should only read it once in each iteration of your main `loop()` to ensure that the value you are using is consistent.

One way to detect that the button bounced is if `TCNT1` has increased by more than one since the last iteration. However, since the main loop runs very quick, you should add a delay in the main loop so that you only check it periodically, perhaps every 100 milliseconds.
Checkoff: *Implement this approach and print a warning every time the button bounces. In addition to printing a warning, turn on the red LED for 500 milliseconds every time the button bounces (but still ensure you are checking for additional bounces while the LED is on).*

2.2 Checking Time

Using a manual delay is somewhat wasteful in that nothing else can occur during the delay. There is a preexisting Arduino sketch `Examples->Digital->Debounce` which implements debouncing by using the command `millis()` to record when the button was pressed and then to only check the state again after a fixed delay period. Look at this code to see how they implemented it.
Checkoff: *Use a similar approach as in the debounce example to remove the manual delay used in the previous section while still achieving the same results.*

3 Timers

In the previous sections you detected button bouncing. You detected this by looking at the counter which counted the number of falling edges on the button input pin and seeing if this value increased by more than one within a small window of time. The assumption was that any oscillations would calm down within that time period. In this section, you will reconfigure the `Timer1` to time how fast bouncing occurs. This will let you minimize the delay in the previous sections so that you can detect very fast button pressing, while still ignoring bounces.

Start by reading section 16.6 of the datasheet on the input capture unit. The way the input capture unit works (when it is configured) is that when a transition on pin `ICP1` (labeled on the Arduino schematic simply as `ICP`) occurs, the current value of the counter, `TCNT1` is copied into the register `ICR1`. An interrupt can be triggered whenever this occurs, but do not worry about using interrupts at this point. Instead, modify your code so that when the button is pressed you reset `TCNT1` and `ICR1` to zero. After 100 milliseconds (or there about) read the value in `ICR1`. If any bouncing occurred (or if you released the button within this time), the register will contain the number of ticks that occurred between the initial press and the last bounce.
Checkoff: *If (and only if) a bounce occurs, print out the time (in micro or milliseconds) since the initial press of the button occurred. If no bounce occurs, simply print out the number of times the button has been pressed. See notes below about doing this.*

For this checkoff, you will no longer be counting the number of falling edges using the timer. You will have to modify your code to properly configure the timer to input capture mode and you will need to go back to manually checking if the button is pressed. Pay attention to the clock source and divider. If you run the clock too fast you will overflow the 16-bit counter before a bounce occurs, but if you run it too slow, you will not have very good resolution. You should also think about if you should use the input noise canceler.

4 Counting Button Presses

Checkoff: *Build on the previous code and create a program that will turn on the red LED if the button is pressed once, green if pressed twice, blue if pressed three times. For each of these you should only turn on the LED after the button pressing has completed (after a short timeout) and you should turn it off after 2 seconds. Also make sure that it functions properly with both short and long button presses, but also avoids bounces.*