Lab 10
Deliverable Specifications

ECE341, University of Idaho

Spring 2015

Demonstration

- All input capture related code (input capture initialization, ISRs, ect.) should be partitioned into two files, `input_capture.h` and `input_capture.c`.
- Refer to the "Project Tasks" section of the lab handout for the demonstration requirements.
- Refer to the "Project Specifications" section for implementation suggestions. You're welcome (encouraged, even!) to choose different configuration options for the input capture peripheral. Be prepared to justify your configuration options when you demonstrate, even if you configure the peripheral exactly as suggested in the lab handout.
- The RPS value reported on the LCD should be an average of the n most recent RPS calculations where $n >= 16$. This is called a rolling average (also called a moving or running average), and behaves similar to a low-pass filter. The number of recent RPS calculations used in the average, n, should be large enough to smooth out the displayed RPS value, but not so big that the displayed RPS value "lags" when a button is pressed.

Report

In the implementation section of the report be sure to address the following:

- Justify your configuration settings for the input capture peripheral.
- Derive an expression for calculating motor RPS from the timer count difference t_{DIFF}. Where possible, use variables instead of numbers. For example, use f_{PB} for the frequency of peripheral bus clock instead of 10 MHz, t_{ps} for the timer prescale value instead of 256. State any assumptions and show each step in the derivation. Compare the derived equation with your C code that computes the RPS and comment on any differences.
- Discuss and defend your moving average implementation. Make sure you include the actual code along with the discussion.

In the testing and verification section of the report:

- Include oscilloscope screen capture(s) of the PWM cycle frequency, CN interrupt length, the Timer 3 interrupt frequency, and the input capture interrupt frequency.
- Copy the table below into your report and for each duty cycle, measure and record the motor supply voltage V_M (header J3 on the PmodHB5), the Hall effect sensor frequency (SB) f_{SB}, and RPS value displayed on the LCD . Measure the motor supply voltage with a multimeter, and use the oscilloscope's measurement menu to measure the frequency of the SB signal (for each new duty cycle, reset the statistics record the mean value into the table. Record the displayed RPS value after it has converged (settled). Hint: to get a 0% duty cycle, simply write 0 into the OC3RS register right before while loop. Then, when you run your design, you should have a 0% duty cycle until you press a button.

PWM Duty Cycle (%)	V_{M} **(V)**	f_{SB} **(Hz)**	**Motor Speed (RPS)**
0			
40			
65			
80			
95			

- What might it imply if your supply voltage drops as your duty cycle increases?
- Plot the motor speed in RPS as a function of PWM duty cycle. Is the relationship linear? Why or why not?

In the conclusion of the report, address the following:

- Implementing feedback:
  - Describe and define feedback in general.
  - Describe (using diagrams, pseudo-code, ect.) how you might add feedback your design to maintain a constant motor speed under a variable mechanical load?[1] More specifically, how might you use the RPS value derived from the Hall effect sensor to adjust the PWM duty cycle such that the motor runs at some specific speed.
- What are the limiting factors when measuring the a signal's frequency with the input capture peripheral as implemented in your project? Starting from the expression derived in the implementation section, derive an expression for the minimum measurable frequency and an expression for maximum measurable frequency. Use the expressions to determine your measurable frequency range.
- How might you use the input capture peripheral to build an auto-ranging tachometer? Describe your approach and include a pseudo-code outline of your algorithm.

[1] Hint: think about a vehicle's cruise control system.