*Department of Electrical and Computing Engineering*

UNIVERSITY OF CONNECTICUT

## ECE 3411 Microprocessor Application Lab: Fall 2016

# Lab Test VI

There are 2 long programming problems in this test. There are 8 pages in this booklet. Answer each question according to the instructions given.

You have **100 minutes** to answer the questions. Once you are done, you need to show the output to the Instructor or TA and e-mail the code to the TA.

Some questions are harder than others and some questions earn more points than others—you may want to skim all questions before starting.

If you find a question ambiguous, be sure to write down any assumptions you make.
**Be neat and legible.** If we can't understand your answer, we can't give you credit!

**Write your name in the space below.** Write your initials at the bottom of each page.

**THIS IS AN OPEN BOOK, OPEN NOTES TEST.**
**YOU ARE ALLOWED TO GOOGLE C SYNTAX.**
**YOU CAN USE YOUR LAPTOP BUT YOU MAY NOT USE INTERNET FOR ANY OTHER PURPOSE BESIDES THE ONE STATED ABOVE.**

Any form of communication with other students is considered cheating and will merit an F as final grade in the course.

*Do not write in the boxes below*

| **1. (x/35)** | **2. (x/65)** | **Total (xx/100)** |
|---|---|---|
| | | |

**Name:**

**Student ID:**

**1. [35 points]:** In this task you are required to implement a watchdog timer interrupt to keep the system safe. Use an ADC to measure the voltage drop over a potentiometer and display it on the first line of the LCD. If the measured voltage is higher than 3V for 2s, then the system should reset. Also the measured voltage should be stored in EEPROM before reset. After reset, the last measured voltage should be displayed on the second line of the LCD.

**Hint:** For writing and reading a floating point value to/ from EEPROM, you can use `eeprom_write_float` function and `eeprom_read_float` function provided in eeprom.h (look at the header file to find out how to use these functions).

**Initials:**

**2. [65 points]:** In this task, you need to implement **Non-blocking SPI**. Write a simple program to test SPI in loopback mode. Also, you need to use input capture interrupt to precisely capture the value of `Timer 1`, in order to measure the delay of one SPI transmission of a single byte.

In particular:

- Configure SPI in Master mode with SPI interrupt enabled.
- Configure Timer1, with input capture interrupt and prescaler = 1.
- In our main function, start one SPI transaction to transmit one data byte of choice over SPI. Before the start of the SPI transaction, you need to store the current value of Timer 1 somewhere else.
- Loopback the transmitted byte by connecting MOSI and MISO pins together.
- Once SPI interrupt triggers, you need to trigger Timer 1 input capture interrupt as soon as possible.
- Calculate the number of clock cycles between these two timer values and print it on LCD.

**a.** (40 pts) Implement this system and measure the number of clock cycles when the clock frequency of SPI is set to $\frac{1}{128}$ and $\frac{1}{2}$ respectively.

Notice that busy waiting on SPIF flag after initiating a SPI transmission is **not allowed**.

**Hint:** Do you need to trigger input capture interrupt before the start of the SPI transaction and at the end of the SPI transaction? Think about it. The fewer statements you add between the start and the end of an SPI transaction, the more accurate you can get.

**b.** (10 pts) What are the time component in the time difference you are measuring. Explain your answer.

**c.** (15 pts) Give an estimation about how many clock cycles are needed for entering an ISR. If your implementation and analysis are correct, you should get a consistent value even when you are using different clock frequencies for SPI. Show your calculations.

# End of Quiz

Please double check that you wrote your name on the front of the quiz.