EECS 473 Midterm Exam

Fall 2014

Name: _____ unique name: _____

Sign the honor code:

I have neither given nor received aid on this exam nor observed anyone else doing so.

_____

Scores:

Problem #	Points
1.	/10
2.	/10
3.	/12
4.	/15
5.	/15
6.	/10
7.	/28
Total	**/100**

NOTES:

1. Closed book and Closed notes
2. There are **12** pages total for the exam as well as a handout. The last 2 pages of the exam can be removed and used as reference for the last problem.
3. Calculators are allowed, but no PDAs, Portables, Cell phones, etc. Using a calculator to store notes is not allowed.
4. You have about 120 minutes for the exam.
    - Some of the low point questions might take you a while to answer. Consider skipping them.
5. **Be sure to show work and explain what you've done when asked to do so. That may be very significant in the grading of this exam.**

1) Circle the best answer. **[10 points, -2 per wrong or blank answer, minimum 0]**
   a. The "L" in an LDO converter indicates that
      - **the output voltage will be considerably lower than the input voltage.**
      - **the output voltage can be nearly as high as the input voltage.**
      - **you only need a low value output capacitor, often less than a µF.**
      - **the output has very little noise.**

   b. With respect to a PCB design, a ***schematic / artwork / pattern / via / trace*** describes how devices are connected together without describing layout.

   c. ***Solder Mask / Limpets / Patterns / Vias / Traces*** and ***solder mask / limpets / patterns / vias / traces*** connect devices to each other on a PCB.

   d. Busybox is a utility commonly found on embedded Linux platforms. It is generally used because:
      - **It provides a relatively small executable that implements a wide variety of standard programs.**
      - **It enables an easier method of writing LKMs**
      - **It provides a guarantee of minimum response time on interrupts**
      - **It is thought to reduce the exposure of kernel modifications to legal issues with source-free distribution.**

   e. The Lesser GPL (LGPL) is generally used to insure that
      - **Merely using a library (and thus potentially having the code in the final executable) doesn't require the source code be made freely available to anyone with the binary.**
      - **Using a GPL'ed complier doesn't require that the generated binary be licensed under the GPL.**
      - **That non-comercial uses of the software doesn't require the generated binary be licensed under the GPL.**
      - **The code can be dual licensed under the GPLs and the Creative Commons**

   f. What does it mean when we say the GPL license is "viral"?
      - **It is very popular on the website Imgur.**
      - **Any use of the binary (complier, editor, OS, etc.) when building software requires that that software also be licensed under the GPL.**
      - **Any use of GPL'ed source code in your program means that your code must also be licensed under the GPL.**
      - **Any use of the GPL'ed source code by a company requires that they license all of their software under the GPL.**

2) Short answer questions **[10 points]**

a. What makes a Loadable Kernel Module (LKM) "loadable"? What are the options we have if the module isn't loadable and what are the downsides to those options? **[7]**

b. With respect to real-time systems, define the terms soft deadline, firm deadline, and hard deadline. **[3]**

Soft deadline:

Firm deadline:

Hard deadline:

3) Consider the following code for a Linux module. **[12 points]**

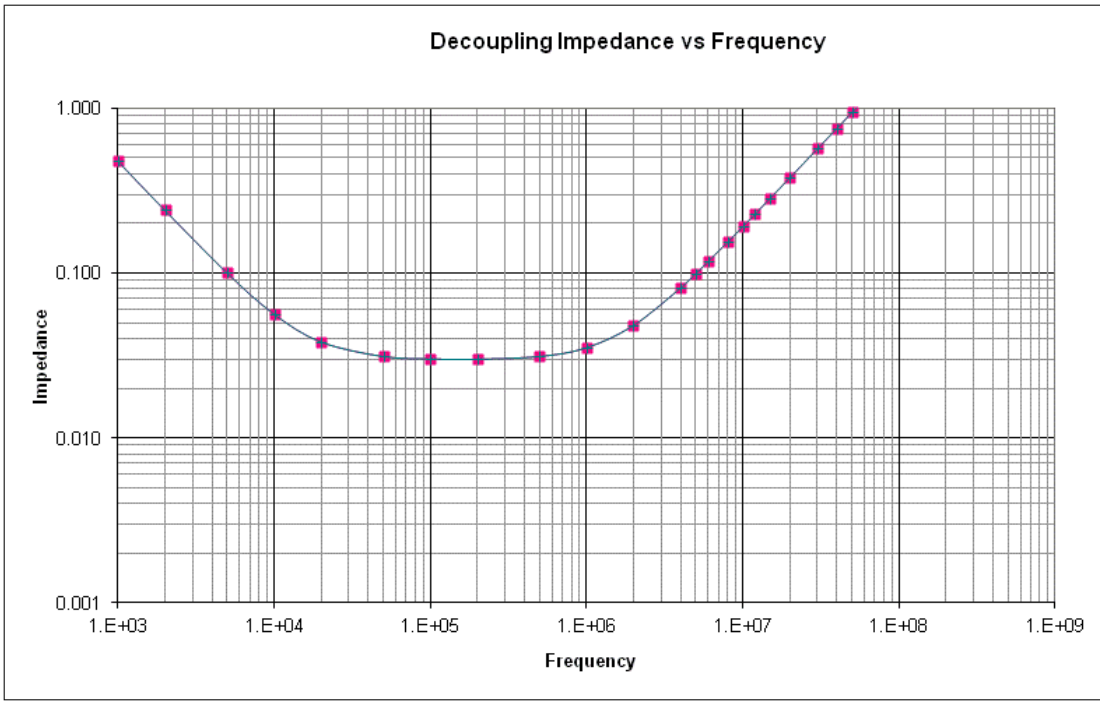
```
struct file_operations memory_fops =
{
  .read = memory_read,
  .write = memory_write,
  .open = memory_open,
  .release = memory_release
};

module_init (memory_init);
module_exit (memory_exit);

int memory_major = 60;
char *memory_buffer;

int
memory_init (void)
{
  int result;
  result = register_chrdev (memory_major, "memory", &memory_fops);
  if (result < 0)
    {
      printk ("<1>memory: cannot obtain major number %d\n",
               memory_major);}
```

a) What is "register_chrdev" doing? Describe how each of the 3 arguments is used. **[6]**

b) The module initialization functions are registered with memory_init() and memory_exit() while the file operations functions are registered using the file_operations struct. Why are they being done in a different way? Address why we don't use just one scheme or the other. **[6]**

4) Using capacitors **[15 points]**

a. The following is a graph of the effective impedance of a 330µF capacitor with an ESR of 0.03Ω and an ESL of 3nH at a wide range of frequencies. Modify the curve to show what it would look like if the ESL were instead 0. **[5]**



b. Explain what anti-resonance is. If we were to graph the above in Spice (or some other circuit-analysis tool) would we see any significant anti-resonance effects? **[4]**

c. Say you have the same capacitor as above (affective impedance of a 330µF capacitor with an ESR of 0.03Ω and an ESL of 3nH). If you had 2 of these in parallel, what would be the total effective capacitance, resistance and inductance of the two together? Explain why this is useful in the context of power integrity. **[6]**

Capacitance: _____

Inductance: _____

Resistance: _____

Why this is useful to us:

5) Say you have the following groups of tasks. For each group find the CPU utilization and identify which groups are RM and which are EDF schedulable. Indicate if you needed to do the critical instant analysis. *If needed*, ***clearly*** show that analysis. The following equation may prove useful. **[15 points]**

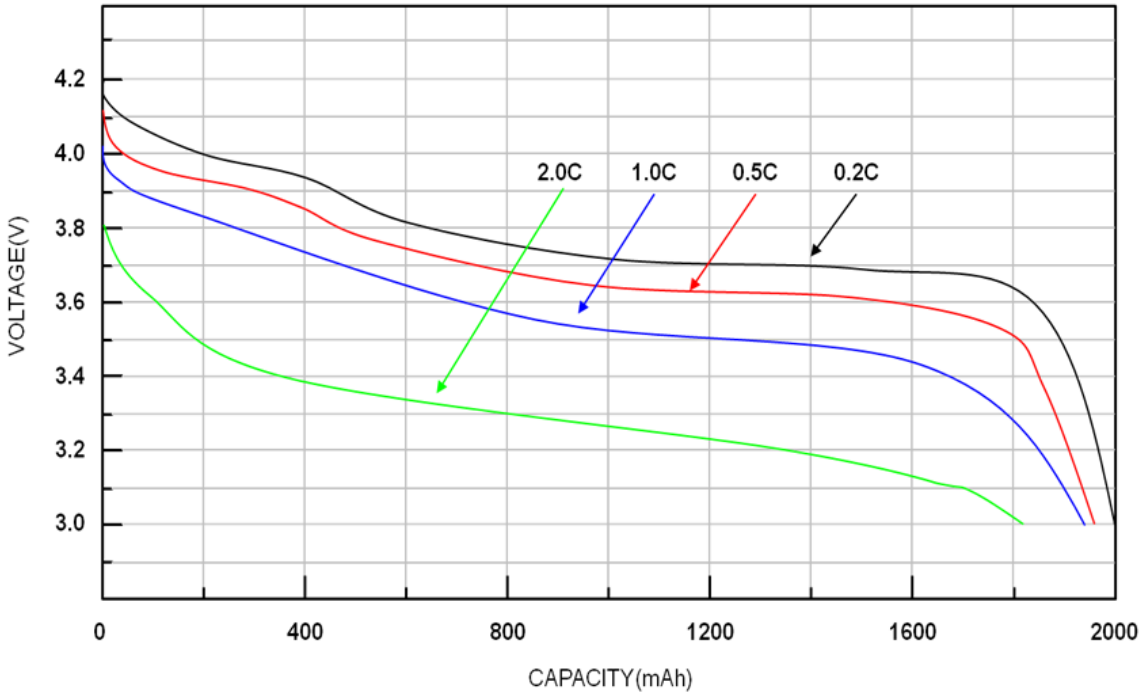
$$\sum_{i=1}^{n} U \leq n(2^{1/n}-1)$$

Group	T1 Execution Time	T1 Period	T2 Execution Time	T2 Period	T3 Execution Time	T3 Period	% Utilization (Total)
A	2	10	2	15	6	20	
B	3	10	4	13	4	15	
C	2	5	3	6	2	11	
D	1	5	3	6	3	11	

Group	EDF Schedulable?	RM Schedulable?	Did you need to examine the critical instance?
A			
B			
C			
D			

6) Batteries **[10 points]**

Say you have a 2000mAh battery with the following characteristics:



a. If your embedded system (e.g. a quadcopter) needs 4.5-3.5V to function and draws 4A, how long will it be able to run on this battery? Show your work. **[2]**

b. How long would you expect your 4A system could run off two of these batteries in parallel? Show your work. **[4]**

c. If you used two of these batteries in series and used an ideal (i.e. current in=current out and no minimum voltage drop) linear regulator, how long could your 4A system run? Show your work. **[4]**

7) You have been asked to build a device for a trucking company. They carry some materials that are sensitive to pressure and temperature. You've been asked to throw together a quick prototype of a device that can detect when the temperature is at 25 degrees centigrade and there is a pressure of 1000 millibars or more. When that happens, you are to turn on an LED.

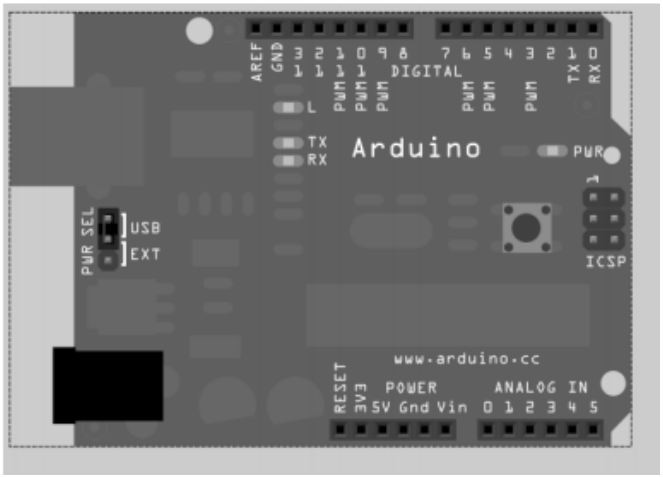
You have chosen to use the BMP180 digital pressure and temperature sensor (much of the datasheet is supplied). For this question, you will A) answer a few questions about the device and then B) show how to have an Arduino light the LED when the conditions above are met. **Use 5V from the Arduino for the device's power.** A simple Arduino sketch, definitions of some Arduino functions and a struct are also attached as the last two pages of this exam. Please feel free to rip them out of the exam if you desire. We'd suggest you read the entire question before proceeding.

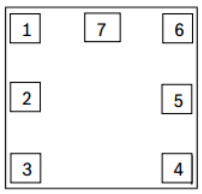
Be aware that the I^2C library will provide an active pull-up to 5V with an internal 40KΩ pull-up resistor. **[28 points]**

a. What is the I^2C module (7-bit) address of the BMP180? **[2]**

b. Which sampling mode seems reasonable for this application? What is the max sample rate (samples/s) of both temperature and pressure with your chosen mode? **[2]**

c. Indicate, by drawing wires, how you will connect the three components and draw any other components needed. **[7]**





BMP180

- +

d. Write an Arduino sketch that lights the LED if the sensor reads a temperature value above 25 degrees and a pressure above 1000 millibars. You should sample the sensor about once a second. **[17]**

Sample sketch:

```
int ledPin = 13;                  // LED connected to digital pin 13
Cal_coef my_coef;                 // Instantiate the configuration type
void setup()                      // run once, when the sketch starts
{
      pinMode(ledPin, OUTPUT);    // sets the digital pin as output
}
void loop()                       // run over and over again
{
      digitalWrite(ledPin, HIGH); // sets the LED on
      delay(1000);                // waits for a second
      digitalWrite(ledPin, LOW);  // sets the LED off
      delay(1000);                // waits for a second
}
```

Note: The functions listed below are not actually part of Arduino but for the sake of this exam please treat them as built-in functions. You may not use the actual I2C library.

int readBytes(unsigned char addr, unsigned char reg, unsigned char *data, char length)

Description

Uses the Arduino Wire library to implement an I^2C read transaction (compliant with the BMP180). The Arduino Wire (I^2C) library uses pins A4 (SDA) and A5 (SCL).

Parameters

- ADDR: The I^2C module (7-bit) address.
- REG: The address of the first register to be read (positive increment).
- DATA: A pointer to where the data will be written.
- LENGTH: The length of the data to be read.

Returns

One on success, zero on fail.

int writeBytes(unsigned char addr, unsigned char reg, unsigned char *data, char length)

Description

Uses the Arduino Wire library to implement an I^2C write transaction (compliant with the BMP180). The Arduino Wire (I^2C) library uses pins A4 (SDA) and A5 (SCL).

Parameters

- ADDR: The I^2C module (7-bit) address.
- REG: The address of the first register to be written (positive increment).
- DATA: A pointer to which data will be read.
- LENGTH: The length of the data to be written.

Returns

One on success, zero on fail.

int calcTemp(Cal_coef my_cal, unsigned char *raw_temp, double &T)

Description

Convert the raw temperature data returned by the BMP180 into degrees centigrade.

Parameters

- MY_CAL: An initialized Cal_coef.
- RAW_TEMP: Temperature data read from the BMP180.
- T: The temperature in centigrade (returned).

Returns

One on success, zero on fail and the T by reference.

int calcPress(Cal_coef my_cal, unsigned char *raw_press, double &P)

Description

Convert the raw Pressure data returned by the BMP180 into millibar.

Parameters

- MY_CAL: An initialized Cal_coef.
- RAW_PRESS: Pressure data read from the BMP180.
- P: The Pressure in millibar (returned).

Returns

One on success, zero on fail and the P by reference.

In addition, you are to use the following struct:

```
typedef struct Cal_coef {
     short AC1;
     short AC2;
     short AC3;
     unsigned short AC4;
     unsigned short AC5;
     unsigned short AC6;
     short B1;
     short B2;
     short MB;
     short MC;
     short MD;
};
```

HINT: Will Cal_coef's elements lay in continuous memory?