EECS 473 Midterm Exam KEY

Fall 2014

Name: ________KEY____________________________ 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. (OBVIOUSLY EITHER ORDER IS FINE)

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]**

In this context "loadable" means that the module can be added or removed from the Kernel at runtime. The only other options are to include all possible modules when the kernel is built (which would make the kernel extremely bloated and is in any case impossible because the module might not even exist when the kernel is built) or have to rebuild and reload the kernel when you want to add a module (generally a driver) to the kernel. That last option would be very slow and extremely cumbersome.

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

Soft deadline: There is a deadline, but the result has some degraded value after the deadline.

Firm deadline: There is no value to the result after the deadline.

Hard deadline: A firm deadline where the results of missing the deadline could be catastrophic.

I will remind you that it is common to only break things into two categories: soft and hard deadlines. In that context, hard deadlines are defined the way firm deadlines are in this three-category scheme. Yes, semantic confusion reigns.

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

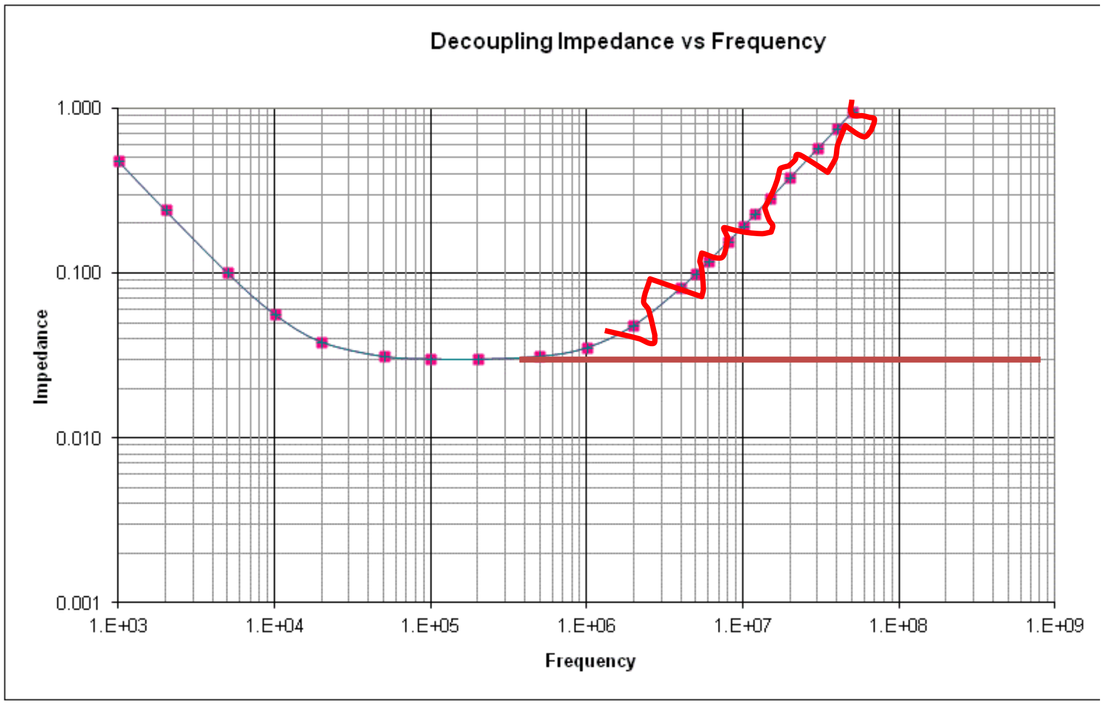
IT IS REGISTERING THE MODULE WITH THE KERNEL. IN PARTICULAR, THE FIRST ARGUMENT TELLS THE KERNEL WHAT MAJOR NUMBER THE MODULE WANTS TO USE, THE SECOND IS THE HUMAN-READABLE NAME THE MODULE IS ASSIGNED (DISPLAYED WHEN DOING AN LSMOD FOR EXAMPLE) AND THE LAST PASSES POINTERS TO ALL THE MODULE FUNCTIONS THE KERNEL NEEDS TO BE ABLE TO CALL.

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]**

MEMORY_INIT CAN'T BE REGISTERED VIA THE STRUCT BECAUSE IT IS MEMORY_INIT THAT DOES THE REGISTERING (PER PART A ABOVE). IN THE SAME WAY, MEMORY_EXIT NEEDS TO UNREGISTER THE MODULE (THAT SAID, I DON'T KNOW THAT IT COULDN'T BE DONE VIA THE REGISTRATION PROCESS AND JUST CLEAN UP ON ITS WAY OUT). BUT FOR THE OTHER MODULES, IT WOULD BE REALLY ANNOYING TO HAVE TO CALL A FUNCTION FOR EACH FUNCTION WE WANTED TO REGISTER—THE FOPS SCHEME IS EASIER AND LIKELY RUNS MORE QUICKLY THAN A BUNCH OF FUNCTION CALLS. THE STRUCT INITIALIZATION IS DONE AT COMPILE TIME, SO THAT HELPS TOO.

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]**

NOT LOOKING FOR A FORMAL DEFINITION OF ANTI-RESONSANCE. MERELY WANT TO HEAR THAT IT CAN CAUSE A SPIKE IN THE IMPEDANCE. AND NO, WE WOULDN'T EXPECT TO SEE AN IMPACT—IT HAPPENS WHEN THERE IS AN INDUCTOR IN PARALLEL WITH A CAPACITOR. IN THIS CASE, THEY ARE IN SERIES.

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: _____**660µF**____________

Inductance: _____**1.5nH**______________

Resistance: _______**0.015Ω**____________

Why this is useful to us:
WE *WANT* THE CAPACITANCE BUT THE PARASITICS (INDUCTANCE AND RESISTANCE) ARE BAD. HERE WE ARE INCREASING THE CAPACITANCE AND DECREASING THE PARASITICS. THIS ALLOWS US TO ACHIEVE A LOWER EFFECTIVE IMPEDANCE.

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	63.33
B	3	10	4	13	4	15	87.44
C	2	5	3	6	2	11	108.18
D	1	5	3	6	3	11	97.27

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

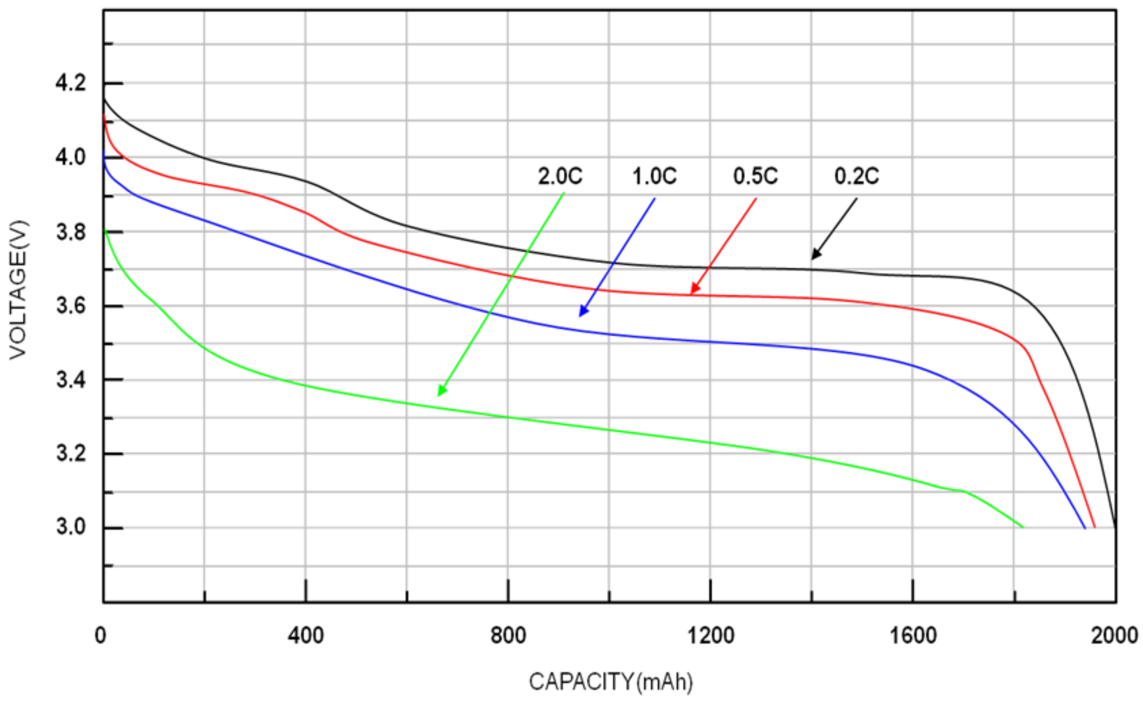
B	0-1	1-2	2-3	3-4	4-5	5-6	6-7	7-8	8-9	9-10	10-11	11-12	12-13	13-14	14-15
T1	X	X	X								X	X	X		
T2				X	X	X	X							X	X
T3								X	X	X					

T3 DIDN'T FINISH.

D	0-1	1-2	2-3	3-4	4-5	5-6	6-7	7-8	8-9	9-10	10-11
T1	X					X					X
T2		X	X	X			X	X	X		
T3					X					X	🟩

T3 DID NOT FINISH.

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]**

4A PUTS US AT 2.0C. WE CAN ONLY GET 200MAH AT 3.5 V. SO 200MAH/4A= 3 MINUTES.

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

NOW WE ARE ONLY DRAWING AT 1C. WE CAN GET ABOUT 1200MAH OUT. SO PER BATTERY WE HAVE 1200MAH/2A=36 MINUTES.

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]**

WE ARE BACK TO RUNNING AT 2C, BUT GIVEN OUR IDEAL LINEAR REGULATOR, WE CAN RUN DOWN TO 1.75V. AT 2C, THAT GIVES US ~1700MHA. SO 1700MAH/4A IS 25.5 MINUTES. ANSWERS AROUND 25-30 MINUTES ARE ALL REASONABLE.

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[1].** 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^{2}C library will provide an active pull-up to 5V with an internal 40KΩ pull-up resistor. **[28 points]**

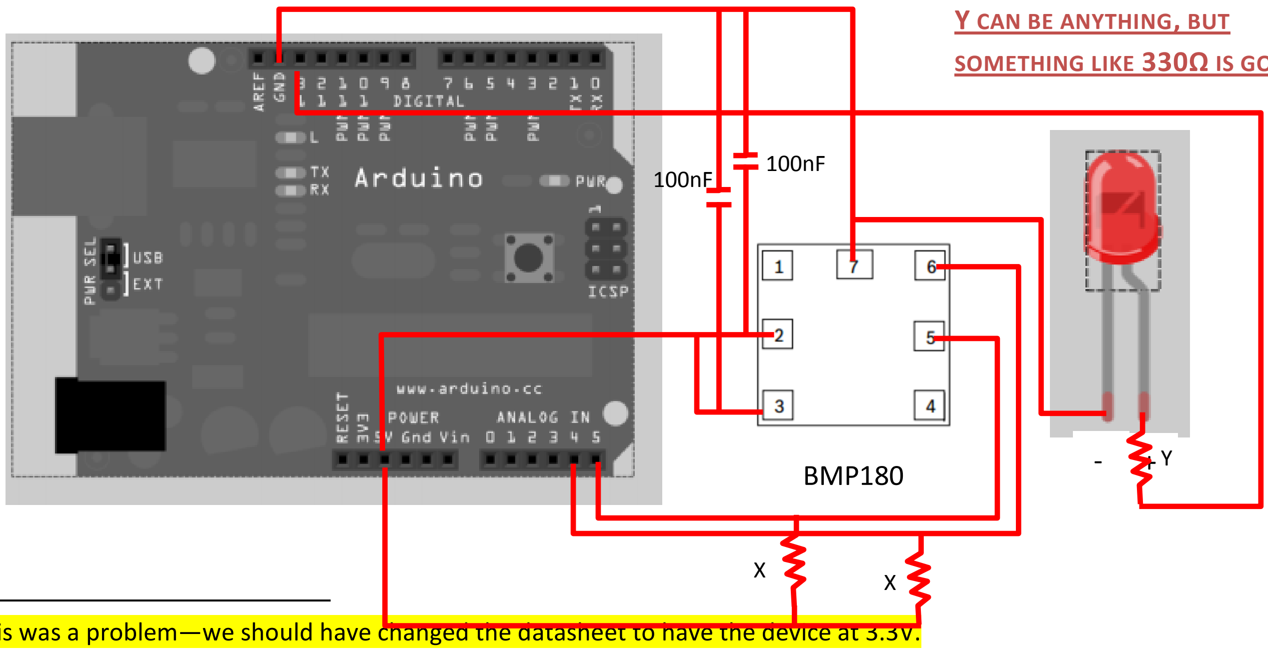
a. What is the I^{2}C module (7-bit) address of the BMP180? **[2]**
(0xEE) >> 1 == 0x77

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]**
Ultra Low Power (4.5 ms)
Sample rate: (1 / (4.5 + 4.5)) * 1000 = 111 samples/sec
Per the spec, one could sample temperature only 1/sec, which gives you around 221 pressure samples/sec.

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

X can be anything between 2.33KΩ and 13.3KΩ.

Y can be anything, but something like 330Ω is good.



[1] This was a problem—we should have changed the datasheet to have the device at 3.3V.

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]**

```
#define BMP180_ADDR 0x77
#define BMP180_TEMP 0x2E
#define BMP180_P0   0
#define BMP180_P1   1
#define BMP180_P2   2
#define BMP180_P3   3
#define BMP180_CONT 0xF4
#define BMP180_LSB  0xF6
#define BMP180_CAL_BASE 0xAA
#define BMP180_CAL_SIZE 0x16 // d22

int ledPin = 13;                          // LED connected to digital pin 13
Cal_coef my_coef;                         // Instantiate the configuration type
static char times[] = {5, 8, 14, 26};     // Delay time for sampling mode
static char speeds[] = {0x34, 0x74, 0xB4, 0xF4}; // Control Reg values for p0 - p3
static unsigned char raw[2];              // Communication buffer

void init_BMP180(){
        readBytes(BMP180_ADDR,BMP180_CAL_BASE,(unsigned char*)&my_coef,\
                  BMP180_CAL_SIZE);
}

double read_temp() {
        double ret;
        raw[0] = BMP180_TEMP;
        writeBytes(BMP180_ADDR, BMP180_CONT, raw, 1);
        delay(5);
        readBytes(BMP180_ADDR, BMP180_LSB, raw, 2);
        calcTemp(my_coef, raw, ret);
        return ret;
}

double read_pressure(char speed){
        double ret;
        raw[0] = speeds[speed];
        writeBytes(BMP180_ADDR, BMP180_CONT, raw, 1);
        delay((unsigned long)times[speed]);
        readBytes(BMP180_ADDR, BMP180_LSB, raw, 2);
        calcPress(my_coef, raw, ret);
        return ret;
}

void setup(){                           // Run once, when the sketch starts
        pinMode(ledPin, OUTPUT);        // Sets the digital pin as output
        init_BMP180();                  // Read in the configuration struct
}

void loop(){                            // Run over and over again
        if (read_temp() > 25.0 && read_pressure(BMP180_P0) > 1000.0)
                digitalWrite(ledPin, HIGH);
        else
                digitalWrite(ledPin, LOW);
        delay(990);
}
```

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

Parameters

- ADDR: The I^{2}C 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 readBytes(unsigned char addr, unsigned char reg, unsigned char *data, char length)

Description

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

Parameters

- ADDR: The I^{2}C 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 {

     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;

} Cal_coef;
```

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