ECE2049 -- Homework 7 / Exam 2 Review
Clocks, Timers, ADCs, and Operating Modes

Submission notes: For homework credit you must complete problem 1 and any one other problem from this document—we will review all of the problems in class!

Problem 1: Alkaline AAA batteries have a capacity of 1100 mAh. A certain embedded system uses the MSP430F5529 powered by 2 AAA batteries in series (i.e. Vcc = 3V). How long can the system run if the MSP430F5529 is always in active mode? How long can it run if the system is in LPM0 84% of the time?

You may assume the system is running at the default clock frequencies. If you need to make any further assumptions, state them clearly.

Problem 2: Answer the questions below completely.

a. How does the CPU know where to go on an interrupt? How does it find the correct ISR?

b. Explain the operation of a Timer in "up mode" (ie. What happens to the timer count? When is an interrupt triggered?).

c. True or False: The operation of peripherals like the Timers or ADC12 causes a big drain on CPU speed. Why or why not?

d. True or False: Interrupt Service Routines (ISRs) should be kept short because the CPU will stop executing them after 255 instructions.

Problem 3: Answer the following questions regarding timers. For these problems, you can assume that the MSP430F5529's *system clocks are running in their default configuration*.

a. Based on the timer configuration below, what is the time between interrupts?

```
  void runtimerA2(void)
{
  TA2CTL = TASSEL_1 + MC_1 + ID_0;
  TA2CCR0 = 16383;
  TA2CCTL0 = CCIE;
}
```

b. Now *disregard any of the settings from part (a)*. Instead, assume TimerA2's interrupts are enabled and the clock source is set to ACLK and `TA2CCR0 = 24000`. What would be the time between interrupts if "***Continuous Mode***" were selected? What if ***Up-down Mode*** were selected?

c. Set TimerA2's control registers to use ***SMCLK*** to count intervals of 25 milliseconds.

d. How long until a clock based on the timer from part (c) would be off by 0.025 sec? Would the clock be fast or slow?

Problem 4: Accelerometers are used for a variety of applications including determining orientation, detecting vibrations, or as components of navigation systems. Accelerometers provide output in g's, which is acceleration relative to the force of gravity. Let's say we want to use accelerometer as single-axis tilt sensor to measure the orientation of a device. We can use the accelerometer readings to measure tilt as shown in the examples below:

Note: This problem is a bit longer than a standard exam problem, but feel free to give it a try!

Tilt = 0°
Reading: 1g

Tilt = 45°
Reading: 0.707g

Tilt = 90°
Reading: 0g

Tilt = 180°
Reading: -1g

An older version of the ECE2049 lab boards had a AXDL335 3-axis accelerometer, which provides three output signals X_{Out}, Y_{Out}, and Z_{Out} as analog outputs with output voltages that correspond to the acceleration along each axis. For this problem, we only need to read the output on the z-axis, which provides enough information to measure tilt (as shown in the examples).

For this problem, assume that V_{CC} for the sensor is 3V. V_{CC} on the MSP30 is still 3.3V.

a. The datasheet for the ADXL335 is provided on the course website. The datasheet assumes that the chip has a source voltage of 3V. Using this assumption, what is the measurement range (in g) and sensitivity (ie. resolution, in mV/g) of the ADXL335?

b. Write an equation to express the voltage output voltage of the sensor (on the z axis) in terms of acceleration (in g). (Hint: You will also need to the *bias voltage*, or the voltage at a reading of 0g based on the datasheet.)

c. What ADC12 reference voltage would you select in order to measure acceleration over the full measurement range of the sensor?

d. What voltage would the sensor output for a tilt of 45 degrees? Using the reference voltage you selected in part (c), what is the ADC12 output for this voltage?

e. Assume that the ADC12 has already been configured using the reference voltage you selected. Write a C function `calc_tilt` that converts the ADC12 output code to a tilt angle from 0 to 180 degrees—you can assume that the standard library functions in `math.h` are available.
(For this part, you can also assume that the ADC12 output is always within ±1g.)