SE 423 Mechatronics Homework Assignment #2

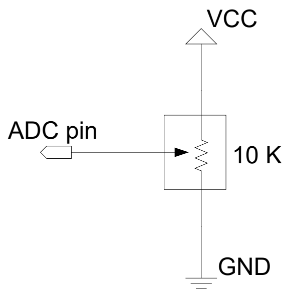
Spring 2020, Due In Lecture February 26th. The Microcontroller Demonstration Check-Offs for Questions 2, 3, 4 and 5 are due by 5PM Tuesday February 25th.

Most answers should be typed. Graphs, etc. can be hand drawn if you wish.

1. Read Chapters 6-8 in "Teach Yourself C". Read Section 10.5.3 in "Introduction to Mechatronics". Read the article "Serial Protocols Compared" (http://coecsl.ece.uiuc.edu/ge423/datasheets/SerialProtocolsCompared.pdf). Look at Wikipedia for information on both I^{2}C and Serial Peripheral Interface (SPI). The OMAPL138 SPI and I^{2}C Module Reference Guide (Technical Reference Guide, Chapter 23 and 30) http://coecsl.ece.uiuc.edu/ge423/datasheets/OMAP138Ref_Guides/TechnicalReferenceManual.pdf has good information and diagrams.

   Questions 2 through 5 below build on each other. So just use the project creator once to give you a starter shell for question 2. After that, add the needed code to the previous problem's code. If the question asks you to remove portions of the code from the previous question leave the code but comment it out. You will be graded on these commented sections. Demonstrate each assignment working to your instructor and print and hand in your **commented** code.

2. Write code to read ADC channel A0 of the MSP430G2553. IMPORTANT: since you will be using P1.0/A0 as A0 (an analog input) remove the jumper on the LaunchPad board that connects the P1.0/A0 pin to the surface mount red LED on the LaunchPad. First you will need to solder a 10Kohm potentiometer to the LaunchPad Break-Out board. Pin 1 of the pot should be wired to GND and pin 3 should be wired to VCC (+3.3V). Pin 2 of the pot should be wired directly to P1.0/A0 breakout pad.



   Modify your default project creator C-file in order that every 1 millisecond analog input A0 is sampled and then every 0.25 seconds its value is printed to the UART serial port (Tera Term). Make sure to read the sampled ADC value inside the ADC's interrupt function because the interrupt function is called when the ADC has finished converting the analog value. Move all the default code from Timer A's ISR function into the ADC's ISR function. The only code in Timer A's function should be the code that starts the ADC conversion, therefore making the ADC start every 1 millisecond. In the ADC's interrupt function use state machine code so that when the ADC reading is below one fourth Vcc all LEDs are off. When the ADC reading is above one fourth Vcc, two LEDs are on. When the ADC reading is above one half Vcc all four LEDs are turned on. Demonstrate this working to your instructor and hand in a print out of your code with your homework submission. Your code should be commented well enough that a beginner working with this microcontroller can understand what each added instruction is accomplishing. Also us the oscilloscope in the lab to watch the voltage change as you adjust the potentiometer. The example http://coecsl.ece.illinois.edu/ge423/datasheets/MSP430Ref_Guides/Cexamples/MSP430G2xx3%20Code%20Examples/C/msp430g2x33_adc10_01_SE423.

c gives you a start on using the ADC10 peripheral of the MSP430G2553. You will also need to read the ADC10 section of the MSP430G2553's users guide http://coecsl.ece.uiuc.edu/ge423/datasheets/MSP430Ref_Guides/MSP430x2xx_usersguide.pdf to become familiar with the ADC and its registers. Make sure your ADC is initialized as follows in your `main()` function. Note that a number of these are the default settings of the ADC10 and therefore no code is needed to set the ADC in that mode.

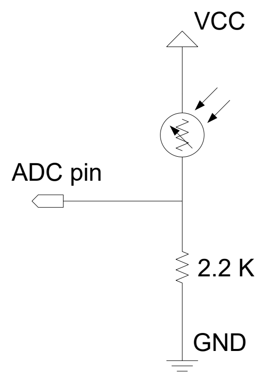
a. Vr+=VCC and Vr- = Vss(GND)
b. Sample-and-hold time 8 x ADC10CLKS
c. Fast mode (up to 200K sample per second)
d. Reference output off
e. Reference buffer on continuously
f. Multiple sample and conversion disabled (Single conversion mode)
g. Reference generator off.
h. Interrupt enabled
i. Select input channel A0
j. Sample-and-hold source = ADC10SC (software starts conversion)
k. Straight binary Data format
l. Clock divider--- try 1, 4 and 8 and see if you can notice any difference in signal noise just by watching the values printing in HyperTerminal / Tera Term.
m. Clock source = ADC10OSC
n. Conversion sequence = Single-channel-single-conversion
o. Enable A0 as the ADC channel.

(Answer) What is the minimum and maximum integer ADC value you would expect to see printed to the UART Terminal?
(Answer) What bit in an ADC control register could you poll on (check periodically) to determine that an ADC conversion was complete if you did not use the ADC interrupt?

3. Make your microcontroller say "HO HO HO" or "GOBBLE GOBBLE GOBBLE" when the lights are turned off.

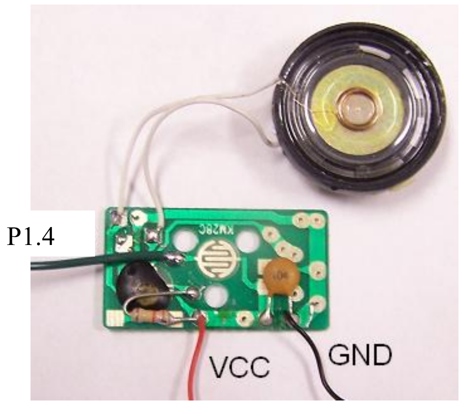
Sorry class, I have run out of enough noise makers for each of you to make your board say "HO HO HO" or "GOBBLE". I do have a few so if some of you would like to use them please ask me. Actually many people in the past got a bit annoyed with the noise makers so maybe it is a blessing in disguise. Instead of turning on your noise maker, after the lights have gone off for 2 seconds simply turn on all of your LEDs. When the lights come back on turn off the LEDs.

Here you will wire/solder a photo-resistor to your LaunchPad Break-Out board. A photo-resistor is a device that changes resistance with respect to light intensity. If you wire the photo-resistor in series with a resistor that has a similar value as the minimum photo-resistor resistance (2.2K ohms should probably work), you can measure a voltage that varies with light intensity. The node in the circuit you should measure is the point where the resistor and photo-resistor are connected. See the figure below:



This circuit is called a voltage divider. One end of the voltage divider should be connected to VCC (+3.3V) and the other end to GND. The middle point is then wired to an ADC input and measured. For this assignment wire this point to ADC channel

A3. Modify problem #3's code to sample A3 instead of A0 every 1 millisecond and print the value to the UART terminal every 0.25 seconds. As a final step, wire a voice box to digital output P1.4. This pin will emulate the pressing of the push button of the "HO HO HO" or "GOBBLE GOBBLE GOBBLE" device. When P1.4 transitions from low to high the voice box is activated. Use the following picture as a guide on how to wire the sound circuit to your LaunchPad breakout board.



You will only be soldering the three wires, GND, VCC and P1.4 to the sound circuit. The picture is of the "HO HO HO" device. The "GOBBLE" device looks slightly different but the wire connections are at the same points on its circuit board. As always, use the DEMO board kept in lab as a guide.

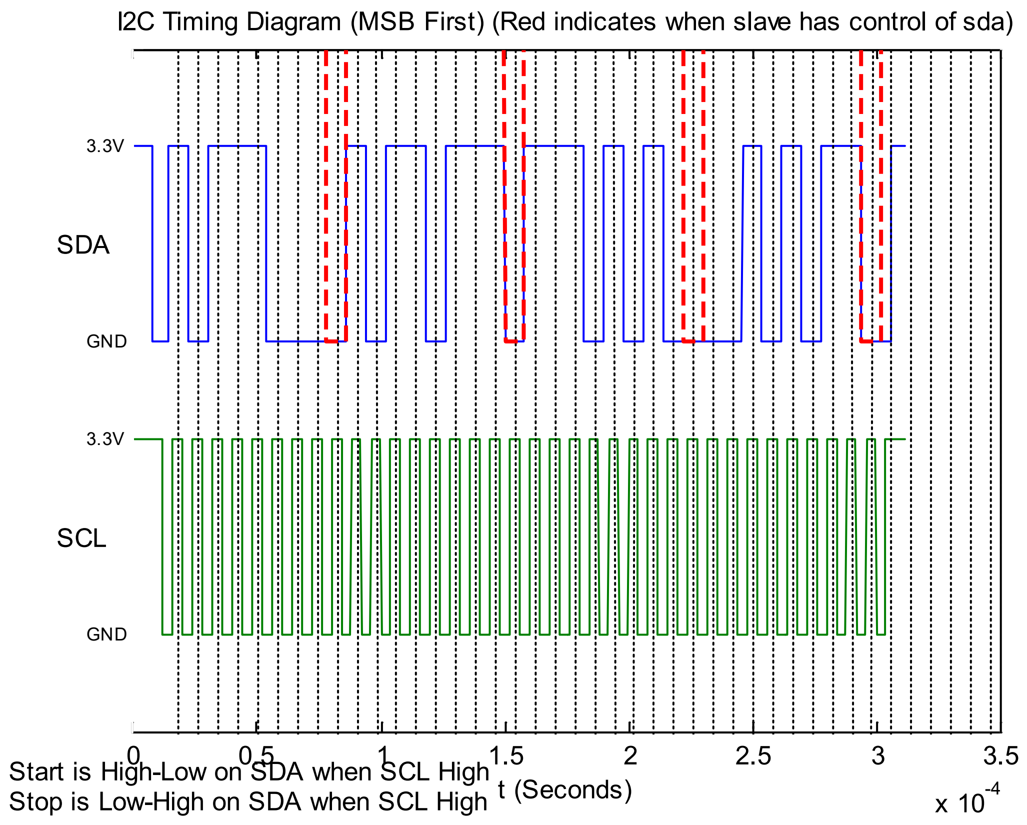
Modify your code to monitor A3's voltage value to determine when the lights have been turned off. When the lights have been turned off for 2 seconds activate the sound device or turn on the LEDs. Your program should be able to continue working with repeated on/offs of the lights and if the lights are not off for 2 seconds. Think about what the different states are for this exercise. Again your state machine code should remain inside the ADC ISR function. Use the oscilloscope to monitor the voltage of the photo-resistor as you develop your code. Demonstrate your code working to your instructor and hand in a printout of your commented code with your HW submission.

4. To introduce PWM generation, add another feature to your microcontroller program. Continuously vary the light intensity of a fifth LED as a function of the photo-resistor's light intensity read by ADC channel 3. The LED's light intensity can be changed by changing the duty cycle (PWM signal) of the logic pin controlling the LED. Wire a fifth LED along with its current limiting 220ohm resistor to pin P2.2/TA1.1. In your code you will need to change the functionality of the P2.2 pin to the TA1.1 mode. Use the P2DIR, P2SEL and P2SEL2 registers to change this mode. See pages 50-51 in the MSP430G2553's datasheet http://coecsl.ece.illinois.edu/ge423/datasheets/MSP430Ref_Guides/msp430g2553datasheet.pdf for help in setting up P2.2 to its alternate "compare" function "TA1.1 (Timer1_A3.TA1)". Read chapter 12, especially section 12.2.5, of the MSP430 users guide for help on setting up the output mode http://coecsl.ece.uiuc.edu/ge423/datasheets/MSP430Ref_Guides/MSP430x2xx_usersguide.pdf. The C example http://coecsl.ece.illinois.edu/ge423/datasheets/MSP430Ref_Guides/Cexamples/MSP430G2xx3%20Code%20Examples/C/msp430g2xx3_ta_16_SE423.c gives help on how to initialize the timer registers for PWM generation. Your PWM's carrier frequency is controlled by the TA1CCR0 register. Set this to a carrier frequency of 10KHz. TA1CCR1's value determines the percentage of duty cycle. Once the timer registers are setup correctly, simply change the value of TA1CCR1 as a function of the value received from ADC channel 3. Using the oscilloscope monitor both the voltage from the photo-resistor along with the duty cycle of the PWM signal. Here you will not need to add any additional state machine code because every millisecond we want to be updating the PWM output.

5. Write code to have your microcontroller receive simple commands, from you the user, typed at the UART terminal. I recommend you use Tera Term as your terminal for this exercise as it by default sends the characters typed at its terminal over the UART serial port. All the code for this exercise should be written in the UART's interrupt function `__interrupt void USCI0RX_ISR(void)` and inside the if statement `if(IFG2&UCA0RXIFG) {`. The MSP430G2553 has two USCI (Universal Serial Communication Interface) serial ports, USCIA and USCIB. Both of these serial port's transmit interrupts share the same interrupt function and so do both the serial port's receive interrupts. That is the reason for the two if statements in these interrupt service routine functions checking bits of the IFG2 register. When the USCIA receives data the UCA0RXIFG bit of the IFG2 register is set. The USCIA is the only serial port on the MSP430G2553 that can be configured as a RS-232 UART. So when you type a character in Tera Term's window the ASCII code of that character is sent over the UART to your microcontroller. On receiving a character over the USCIA serial port, its value is placed in the register UCA0RXBUF and the interrupt function USCI0RX_ISR is called. So inside this function and inside the correct if statement perform the following:
    a. Read the char in UCA0RXBUF into your char variable.
    b. So you get some feedback that the characters you type are actually received by the microcontroller, echo the character back over the serial port. This will then display the character in the terminal window. To send this character back call the function `sendchar(recchar);`
    c. Then check the character that was sent to the microcontroller. If the character is the ASCII character '1' turn on LED1. If it is '2' turn on LED2, '3' turn on LED3, and '4' turn on LED4. If it is '5' turn off LED1, '6' turn off LED2, '7' turn off LED3 and '8' turn off LED4. If it is any other ASCII character turn all LEDs off.

   Note that you will have to comment out the lines of code that turn on and off the LEDs in your previous code. Demonstrate your code working to your instructor along with showing the TX and RX lines of the UART on the oscilloscope. Hand in a printout of your commented code with your HW submission.

6. The plot below is the timing diagram for the transmission of data over an I2C serial port. What I2C address is the data being sent to? What data is being sent? Approximate the baud rate (bits/sec) from the plot.



7. Draw the timing diagram for the SPI serial port of the OMAPL138 processor set up to transmit and receive 8-bits at a time. The SPI registers have also been set so that POLARITY=1 and PHASE=1. See the SPI chapter (especially pages 1409-1411) of the OMAPL138 technical reference http://coecsl.ece.illinois.edu/ge423/datasheets/OMAP138Ref_Guides/TechnicalReferenceManual.pdf for details. Two values are sent, 210 then 56 while two values are received, 25 then 148.

8. Using problem 13 from HW1,write pseudocode for the 40 cent soda machine. The pseudocode is started below:

```
switch(state) {
    case 1: // No money
        Display 0 cents on Soda Machine
        if (coin == 5) { // if statements to decide want to do next time into switch statements
            state = 2;
        }
        if (coin == 10) {
            state = 3;
        }
        if (coin == 25) {
            state = 6;
        }
        break;
    case 2: // 5 cents deposited
        Display 5 cents on Soda Machine
        .
        .
        .


    case 9: // 40 cents deposited
        Dispense Soda
        state = 0;
        break;
}
```

G2553 Play-Time: **(These items are not graded nor required)**

1. Display the most significant bits (MSB) of ADC10's reading to the 4 LEDS. This would give you a quick macro display of the input voltage.
2. Start reading the data sheet for the Texas Instruments DAC TLV5606. A DAC changes a digital value to an analog voltage. This is useful for driving or controlling devices such as audio amplifiers and motor amplifiers. In HW 3 you will interface this DAC with the G2553. Ask your instructor for a DAC chip (or use the one you get sampled from TI) if you would like to get started early on this assignment.