SE 423 Mechatronics Homework Assignment #3
Spring 2020, Due In Lecture March 11th. The Microcontroller Demonstration Check-Offs for Questions 4 and 5 are due by 5PM Tuesday March 10th.
Most answers should be typed. Graphs, etc. can be hand drawn if you wish.

1. Review Chapters 5-6 in "Teach Yourself C".
2. Assume you have designed the switch glue logic shown in Figure 1. Assume that this crazy circuit is a part of a circuit used in an industrial machine, where the switches are located strategically to indicate the different states of the machine.

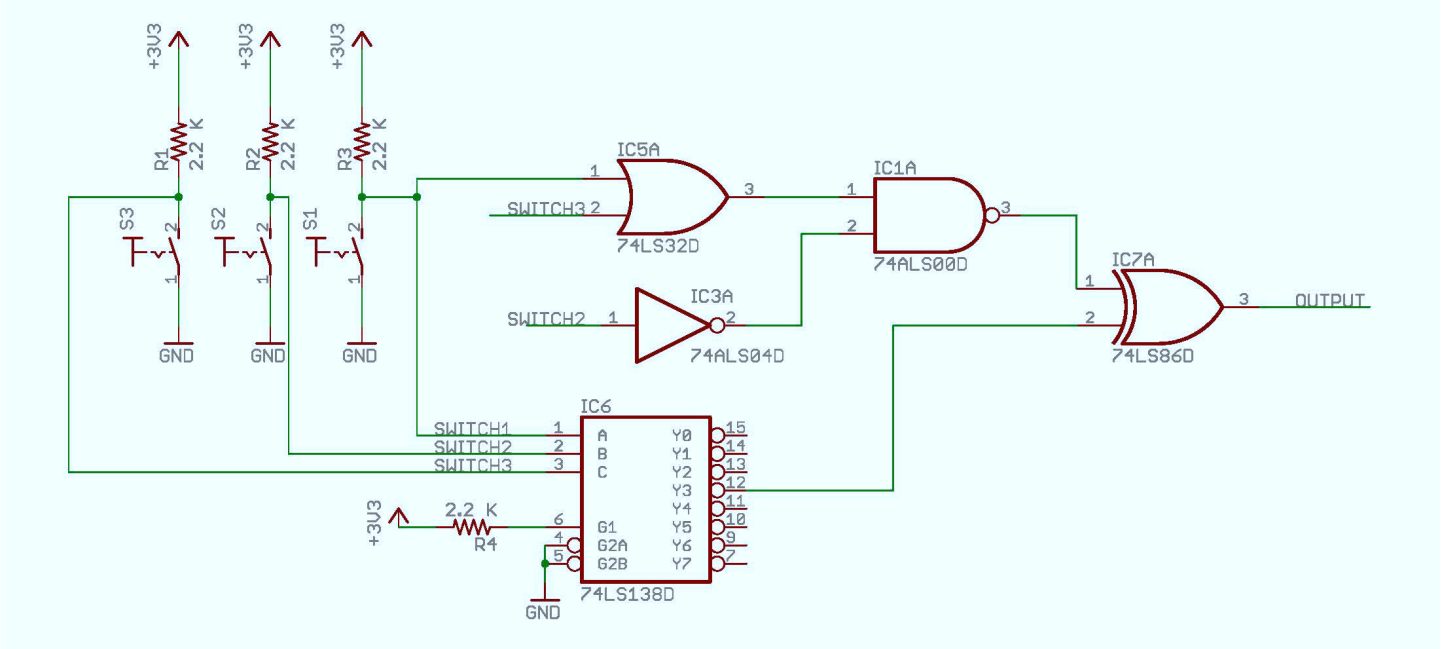

Figure 1.

Depending on the ON/OFF configuration of the switches, the circuit outputs a 0 or a 1 (GND or 3.3V) on the output line. Determine all the possible input/output combinations of your digital circuit. Note: Double check what each of these logic gates are by performing a web search on each part number, i.e. 74LS32.

3. Give yourself a small introduction working in Linux, especially at the command prompt, by reading through the help at http://community.linuxmint.com/tutorial/view/244. Answer the following questions:
   a. What command along with an option would you type to list all the files in a directory and more information like file date and file size.
   b. What command is used to change to a new directory? What command displays the directory you are currently in?
   c. What command is used to copy a file in Linux? In the /home/root directory there is a file named "hw3data.dat". Also in the /home/root directory there is a directory named "hw3". What text would you type to copy "hw3data.dat" to the directory "hw3" with the new name "hw3trial1.dat?"
   d. What does the "less" and "cat" commands do? How are they different?
   e. Experiment with the "ifconfig" and "ping" commands. *These functions are not explained at the given website.* Why are these functions useful for us working with the robots?
   f. What command line text will set the date to 2:30PM March 5th, 2012 using the "date" command?

4. Interface the MSP430G2553 microcontroller to the TLV5606 digital to analog converter http://coecsl.ece.uiuc.edu/ge423/datasheets/MSP430Ref_Guides/tlv5606.pdf. Notice that the LaunchPad Break-Out board has the surface mount pattern/spot for the TLV5606. As a first step solder the TLV5606 chip to the board. The solder traces on the board connect pin 8 (VDD) to +3.3V and pin 5 (AGND) to GND so you do not have to worry about powering the chip. Solder a 0.1uF capacitor in the capacitor spot just above the DAC (TLV5606). This capacitor is just decoupling possible noise on the power traces connecting the DAC. Pin 6 (REFIN) is connected to the output of the potentiometer pins located

just to the right of the TLV5606. Solder a 10Kohm potentiometer in that POT section. Pin 1 of the pot is connected to +3.3V for you and Pin 3 is connected to GND. You will adjust this potentiometer later to produce the correct REFIN voltage.

To interface to the TLV5606 you will use the USCIB serial port in SPI mode. The TLV5606 has four pins to interface to a processor, \CS, FS, SCLK and DIN. There is no DOUT pin on the TLV5606 because a DAC is just an output device. \CS and FS are both types of \SS (slave select) pins. \CS is used if we desire to communicate to more than one chip with the SPI port. In this exercise you will only be communicating to the one device so you will not use the \CS line except for wiring it to GND making the TLV5606 always selected. Reading the TLV5606 datasheet, this is referred to as a three wire SPI interface. Wire the \CS pin to GND on the break-out board and note that potentiometer's GND pins are just to the right of the \CS pin. Again looking at the datasheet you will find that the FS (frame sync) pin is not a standard SPI pin. We will wire it to a general purchase output pin and produce its signal manually.

You will use pins P1.5, P1.6 and P1.7 to communicate with the TLV5606 so you will need to disconnect those lines from the LED it is currently wired to. Wire P1.6 to TLV5606's FS (as stated above, you will use P1.6 as a GPIO and manually set and clear it), P1.5/SCLK to TLV5606's SCLK and P1.7/SIMO to TLV5606's DIN. This will leave only P1.4 still connected to one of your LEDs.

Now you are ready to develop your C code to interface to the TLV5606. (Start with the code you developed for reading the photo-resistor's output because the last step asks you to wire the photo-resistor to A3's input). First power on the board and adjust the potentiometer to output approximately half Vcc (1.65V) volts to the REFIN pin. Do this by measuring the voltage at REFIN with a DMM on the benches.

http://coecsl.ece.illinois.edu/ge423/datasheets/MSP430Ref_Guides/Cexamples/MSP430G2xx3%20Code%20Examples/C/msp430g2xx3_uscib0_spi_09_SE423.c is a good start for this problem but it does not have all the appropriate settings for our interface. Study the TLV5606 datasheet to determine the polarity (update on rising or falling edge) and default state (Hi or Low) of the SCLK pin.

P1.6, wired to FS, should be setup as a Digital Output and initially low. In your code when you want to write a new DAC value through the SPI make sure to first pull P1.6 high and then right back low to create the FS pulse.

At this point you need to send your data to the TLV5606 over the SPI serial port. Initiate this send to the DAC from inside the ADC interrupt function every 1 millisecond. The USCI can only send 8 bits at a time. So to send the 16bit data value the TLV5606 requires, you will need to perform two write commands. One right after you pull FS high and then right back low. The second byte should be written inside the SPI's RX interrupt, which occurs when the first byte has been transmitted. *Remember here we use the RX interrupt instead of the TX interrupt because the RX interrupt indicates that 8 bits of data has be both received and transmitted through the SPI. Actually since this is a TX only device it would be OK to use the TX interrupt in this case, but in general I find it better to use the RX interrupt in most SPI implementations.* After the second byte is transferred, another interrupt will be trigger. Make sure to handle this second interrupt call differently than the first call. Connect the SPI pins SIMO, SCLK and FS to the oscilloscope's logic analyzer channels to make sure your SPI communication is working correctly.

With your DAC working, write a program that ramps the DAC voltage from 0V to Vcc. To do this increment an integer every 1 millisecond and output this value to the DAC. Once the DAC gets to its maximum value start back at 0V and increment up again. Watch the DAC voltage increase on the Oscilloscope. Also make sure that your program can still write data to the serial port to be displayed in Tera Term.

As a final step, wire the photo-resistor's output to pin P1.3, ADC channel A3 (should already be done). Sample the photo-resistor's value every 1 millisecond and echo that value to the DAC. Scope the output on the oscilloscope. Again

make sure that your program can still write data to the serial port. Submit your code for this assignment and demonstrate each step working to your TA.

5. In this question you will be commanding/controlling two RC servos. Your first step will be to solder the 5V regulator to the board in order to power the RC servos. Using Figure 2 and the demonstration board found in lab as a guide, solder the 5V regulator (TL1963A-KTTR), the surface mount parts: two 10uF Capasitors, two 470ohm resistors and one 1.5Kohm resistor, a LED, and power connector to the board. All the soldering positions for these components are found on the breakout board close to the LaunchPad..

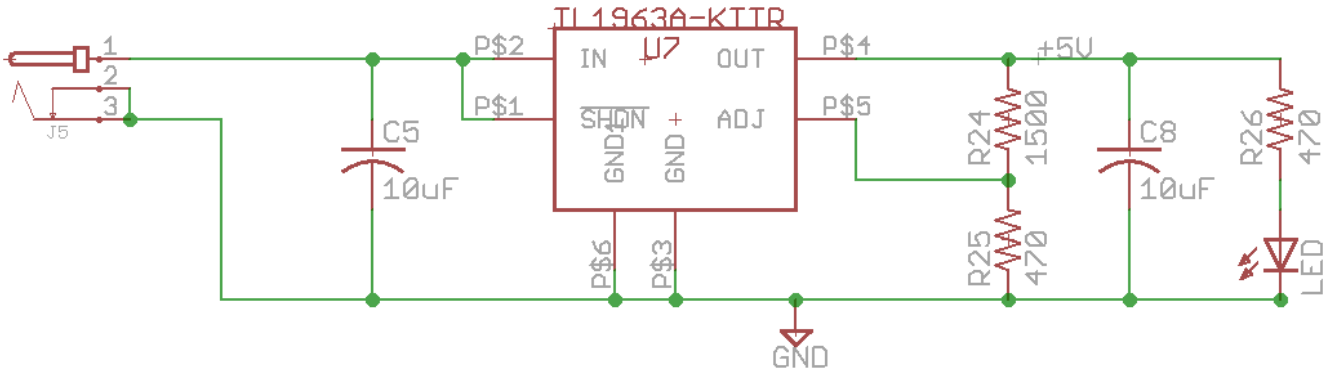

Figure 2:

Write code to command both of your RC servo motors. **Note: you will not use the DAC to drive the RC servos. The code you developed for commanding the RC servo motors does not use the code you just finished in question 5.** To command an RC servo you need use a PWM signal similar to the dimming LED problem in homework #2. The carrier frequency for this PWM signal is much slower, only 50Hz and the duty cycle is only varied from approximately 3% to 14%. Another way to think of this PWM signal is as a high going pulse that occurs every 20ms and the width of the pulse can vary from 0.6ms to 3ms. For an RC Servo, the pulse's length of time determines what angle the RC servo motor will turn to. 0.6ms (0°) to 3.0ms (180°). For this assignment use TA1 Compare #1 to command one RC servo and TA1 Compare #2 to command a second RC servo.

Notes: Setup TA1 Compare #1 and TA1 Compare #2 as PWM outputs.

To achieve a PWM carrier frequency of 50Hz you will need to use the input divisor of TA1CTL.

This PWM signal drives Pin3 of the RC servo.

I recommend that you scope this PWM signal to see if you have created a high going pulse every 20ms.

Submit your code for this assignment and demonstrate it commanding the RC servo motors to step back and forth between two or more positions. Also make sure that your program can still write data to the serial port. **RC Servo Motor Wiring:** Pin1 (Black) → GND, Pin2 (Red) → +5V, and Pin3 (White or Yellow) → PWM Signal. Your TA will also help you with this wiring and show you how to connect the +5V regulator's output as the RC servos power source. You will also solder two 1X3 headers to the breakout board for connecting the RC servos.

6. This homework problem is the start of a homework problem that will continue to build in the remaining homework sets. The goal of this homework problem and the continuing homework problems is to start you early thinking about the states the robot will have when autonomously navigating the final project course at the end of the semester. There is definitely not one answer for these state transition diagrams and, in addition, the answer you come up with in these homework assignments may not work when you start implementing the code on the actual robot. Items / issues with the real robot may cause you to

change your ideas on how to define the states of the system. I may even say that your state transitions look very good and then you find out that it actually does not work too well on the actual robot. That is ok, because you will be ahead of the game in your thoughts on how to program the robot for the desired tasks.

Grading of these homework problems will be more lenient, but I am looking for some good thought. I would like your answers written out very neatly so I can read all parts of the diagram and code. Do not give me garbage!

So to first start out I want you to create a state transition diagram and pseudocode (with switch case structure) with only three (you can use more if you wish) states: 1. Robot moving from one X,Y coordinate to another X,Y coordinate waypoint as if there are no obstacles in its way. If the robot gets to the waypoint it is finished and stops. 2. Using only the LIDAR's front distance and one of the right distance measurements, if an obstacle is detected, right wall follow around the obstacle until it is safe to break away from the right wall following and continue on to the X,Y waypoint. 3. While either moving to an X,Y position or avoiding an obstacle by right wall fallowing, if a bright pink color is seen, stop the robot and turn so that the color is directly in front of the car. Then drive forward until the color disappears below the camera's view. Once the color disappears, go back to moving to the desired X,Y waypoint.

The pseudocode does not have all the programming details of making the robot go to a X,Y point or right wall follow or follow bright pink color. When you are coming up with the pseudocode think as if there is a function that needs to be called every one millisecond that drives the robot to an X,Y point or right wall follows or follows pink. Every millisecond your pseudocode should call this function and every millisecond there should be if statements that make the decision what state should be run the next millisecond. Whether it should be the same state currently being performed or if a new state should be performed the next millisecond. In other words, the pseudocode is at a high level focusing on the decisions that need to be made more than the details of making the robot perform tasks.

As you are thinking about this state transition diagram and pseudocode, if there is some sensing missing or issues that you think will arise note them in a notes section and in the next homework assignment I may have added some sensing, etc., to solve that issue. Also, it will be very important for you to keep the solution you hand in, because you will be adding to this diagram and pseudocode in upcoming homework sets.

G2553 Play-Time: (These Items are not graded or required)

1. Experiment with how fast you can sample the ADC and echo its value to the DAC.
2. Wire the speaker to the DACs output. Write some code to produce a 0 to 50mV square wave. Vary the frequency of that square wave and you have just made a tone generator. Switch the tones at specific times and you can play some songs. Your TA can show you how to solder the speaker to the demo board.
3. Sample the microphone into an ADC10 channel and output (you possibly will need to scale the value) to the speaker. Turn on an LED for 1 second when a large noise is heard. See the datasheets for the microphone at http://coecsl.ece.uiuc.edu/ge423/datasheets/MSP430Ref_Guides/microphoneWM64_1.pdf and http://coecsl.ece.uiuc.edu/ge423/datasheets/MSP430Ref_Guides/microphoneWM64_2.pdf.