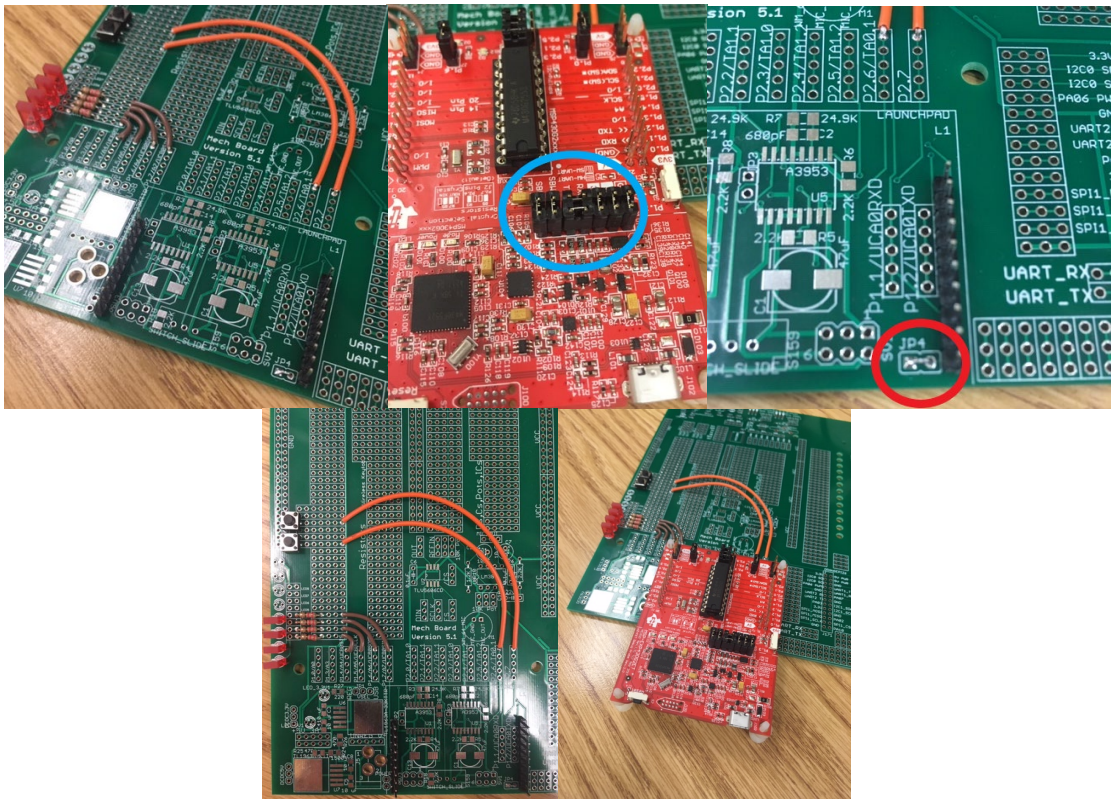
SE 423 Mechatronics Homework Assignment #1
Spring 2020, Due In Lecture Wednesday February 5th. The Microcontroller Demonstration Check-Offs for Questions 2, 3, 9 and 10 are due by 5PM Tuesday February 4th.
Most answers should be typed. C code included. Graphs, etc. can be hand drawn if you wish.

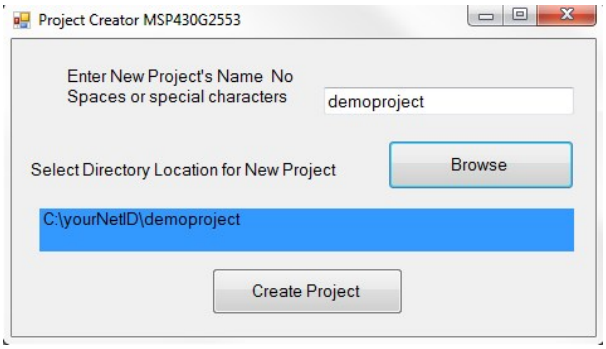
1. Read Chapters 1-5 in "Teach Yourself C". You will probably also have to look at Chapter 11 if you are not already familiar bitwise operators. Read about "RS-232" at Wikipedia. Also at the bottom of the "RS-232" Wikipedia page go to the external link: *Serial Port Basics* (http://www.acumeninstruments.com/Support/documentation/SerialPortBasics/index.shtml). Read the LaunchPad's Users Guide http://coecsl.ece.illinois.edu/ge423/datasheets/MSP430Ref_Guides/LaunchPadUsersGuide.pdf. Keep in mind that we will be using the MSP430G2553 microcontroller.

2. Solder your microcontroller board, solder your breakout board.
   a. Solder the male header pins to the breakout board. Also make sure the TXD and the RXD jumpers are set in the correct configuration. See the pictures below and the already soldered "demo" board found in the lab as references.
   b. Solder the LaunchPad Breakout board as follows (refer to the already soldered demo board when in doubt):
      i. Solder two push button switches to your board. One side of each switch must be soldered to GND. Solder two wires to connect the 2 switches to the I/O lines P2.6 and P2.7. Note where the switches are located on the breakout board.
      ii. Solder 4 LEDS to your board in LED spots LED1, LED2, LED3 and LED4. The LED's short wire is its cathode and this should be connected to GND which is towards the outside of the breakout board. See http://en.wikipedia.org/wiki/LED. Limit the current to the LED with a 220 ohm resistor. Connect the input of these in-series resistor-LED circuits to the I/O Lines P1.4, P1.5, P1.6 and P1.7.
      iii. Using a lead you cut from your LEDS or resistors, solder a jumper at JP4. See below picture.



3. Set up the recommended file structure for your Code Composer Studio 8 workspace and homework projects.
   a. On the hard drive (C:\) of the computer, create a folder that is named your netID. i.e. C:\yourNetID\
   b. In the new folder you created, make a new folder called workspace. i.e. C:\yourNetID\workspace\
   c. As you get familiar with Git in lab, you will want to make a repository for your homework assignments. If there is time during Lab 1 we will get you started using Git for your homework repository. As the semester progresses you will become more and more familiar with Git. Until you understand Git well, make sure to make other backups of your homework code.
   d. Run the MSP430G2553 project creator found at N:\msp430\G2553ProjectCreator\G2553ProjectCreator.exe or http://coecsl.ece.illinois.edu/ge423/datasheets/MSP430Ref_Guides/ProjectCreator/G2553ProjCreator.zip.

e. Create a new project, e.g. demoproject, and browse to your netID folder you created (C:\yourNetID\). *We recommend that you do NOT place your projects in your workspace directory (C:\yourNetID\workspace\). If for some reason you need to create a new workspace, you can simply delete the contents of your old workspace directory without having to move your projects.* Click "Create Project" once your project creator looks similar to the following:



f. Open Code Composer Studio 8 (cube looking icon on the desktop). When you are prompted for the workspace, click browse and browse to the workspace directory you created. i.e. C:\yourNetID\workspace. ***DO NOT check the box to make this the default workspace***. Click "Ok". CCS will present you with a welcome screen when it loads. Click the x to close the "TI Resource Explorer" tab.
    i. *If you are not prompted for a workspace when you opened CCS, another workspace has been made the default. Change this in CCS by navigating to Window->Preferences->General->Startup and Shutdown->Workspace and checking "Prompt for workspace on startup." You can switch workspaces once CCS has already started by File->Switch Workspace.*
g. Import the project you created with the project creator by selecting Project->Import Existing CCS Eclipse Project. With "Select search-directory" selected, browse to the project you created in your netID folder and click Finish.
h. Make the project active by selecting it in the Project Explorer on the left hand side of the CCS interface. It should become bold and say active/debug. Build the project by clicking the "hammer" icon (or Project->Build Project).
i. You will need to plug one USB cable into your microcontroller board set. There should be a cable that came in your LaunchPad box which connects to the LaunchPad's micro USB plug. Now that your board is plugged into the computer, click the "bug" icon to program the MSP430G2553 with the starter code. When the project is loaded, CCS will change "perspectives" into the CCS Debug perspective. Remember you can always change perspectives by clicking them in the toolbar or through Window->Open Perspective. Run the code by clicking the "resume" green arrow. You can halt the code by clicking the yellow "suspend" button and stop the debug session by clicking the red "terminate" button. When the code is running, LED1 (located towards the bottom of the LaunchPad, not on the breakout board) should be blinking and text should be printing to the serial port. Launch TeraTerm (easiest to find by just typing in "Tera Term" in the windows search bar) and set it up for "COM??: MSP430 Application UART1" at 115200 bit/second (change the baud rate using Setup->Serial Port…). The specific COM port number may vary for different computers. You should then see state and count text printing to the PC's screen. TeraTerm can be downloaded at http://ttssh2.sourceforge.jp/index.html.en if you need it install on your own computer.
j. At this point you have successfully programmed your microcontroller.

4. Find the result of the following C statements. First do the calculations by hand and show your work. Then try these calculations one at a time on your microcontroller to verify the answers. An easy way to display the result is to send the value over the serial port using the "UART_printf" function. The microcontroller can only handle one UART_printf at a time. This is due to the limited memory of the microcontroller and the fact that each character sent over the serial port takes 10/9600 seconds so if you send 15 characters each UART_printf you would need to wait at least a 50th of a second in between each UART_printf. To be safe only print about every 10th of a second or slower. The default code prints every 500 milliseconds.
   a. `First explain why each character sent over the serial port takes 10/9600 seconds.`
   b. `x = 532 & 205;`
   c. `x = 0x4f & 0x1ad;`
   d. `x = 0x02ad | 0x1a1;`
   e. `x = 0x3ba >> 4;`
   f. `x =  104 << 3;`
   g. `x = 495 & ( 0x5 << 4);`

5. What are the five basic types in C? See the "Teach Yourself C" book.

6. Read section 6.4 of the TMS320C6748 (DSP used for our Robots) C reference data sheet (http://coecsl.ece.uiuc.edu/ge423/datasheets/OMAP138Ref_Guides/C6000CCompiler7.3.pdf) How many bytes does this processor use for each of the following variable types: int, long int, unsigned int, short int, char, double, float?

7. Same as in question 5 but now for the microcontroller, read section 5.3 of the MSP430 C reference guide (http://coecsl.ece.illinois.edu/ge423/datasheets/MSP430Ref_Guides/MSP430CCompilerUsersGuide.pdf). How many bytes does this processor use for each of the following variable types: int, long int, unsigned int, short int, char, double, float? *We will need to be careful when using floats on the MSP430G2553 microcontroller. It is not a floating point processor so all the float manipulation needs to be done in software. This means the operations will be much slower than ints and take up more program space.*

8. *Please note that this question is not asking you to buy anything. It is asking you to request FREE samples from Texas Instruments.* You are going to be soldering a number of different ICs to your microcontroller circuit board throughout this semester. Texas Instruments has an outstanding service of providing FREE samples of a large number of their chips. Obviously their goal is to give you a few chips to develop your product and then when you go to production you will purchase their ICs. TI encourages students to do the same in the hope that in the future you will choose TI parts for your future designs. So for this home work problem I would like you to go to TI's web site (www.ti.com) and find a link to samples. There I would like you to get the following ICs. To show me that you ordered your samples either turn in with your home work the packing slip from your sample order or the web confirmation that your order has been placed. The part numbers I give below are the exact number you need to order. Sometimes the website will list other similar part numbers. Make sure to order the exact part number I list here. If any of the chips are on backorder go ahead and order them and they will come in sometime during the semester. Don't worry, I have plenty of extras if all the parts do not ship right away.

```
TLV5606ID              Quantity 3
TL1963AKTTR            Quantity 3
TL1963A-33KTTR         Quantity 3
```

9. Make the 4 LEDs soldered to your microcontroller board chase each other. Are you too young to remember the "old" Knight Rider car http://www.knightrideronline.com or the "old" Cylons from the "old" BattleStar Galactica http://www.youtube.com/watch?v=YPL7QPCfVZc? Make sure to put most of your code in the Timer_A0 interrupt function. The timer interrupt function is called every one millisecond. You must implement the on and off of the 4 LEDs using a state machine. Also make sure that your code is still able to send text messages to the PC through the UART_printf statement. **Turn in your C code and demo your working code to your TA to get credit for this problem.**

10. Write a function, `get_switchstate()`, that returns a `char` that represents the state of the two switches that you soldered to the breakout board. The function should be passed `void` parameters and return a `char`. (i.e. `char get_switchstate(void)`) Return a `char` instead of an `int` because we always have to keep in mind that there is only 512 bytes of RAM on the MSP430G2553 so we need to conserve memory where ever possible. In this case we will be using the `char` as an 8-bit integer not as an ASCII character. This function will return the state of the two switches wired to P2.6 and P2.7. The possible return values then will be 0, 1, 2, or 3. You will have to read the "Digital I/O" section of the MSP430 users guide http://coecsl.ece.uiuc.edu/ge423/datasheets/MSP430Ref_Guides/MSP430x2xx_usersguide.pdf to find how to set P2.6 and P2.7 as general purpose inputs with internal pull-up resistors enabled. (Also setup P1,4,P1.5, P1.6, and P1.7 as outputs so you can turn on and off the 4 LEDS. Perform all these initializations in `main()`) Change the default project so that it prints the value returned from this new function to the serial port (viewed with Tera Term). Also blink a different LED depending on the combination of

pressed and un-pressed switches. Blinking an LED means that if you hold the switches constant at a certain state, an LED should be turning on and off at whatever rate you program. So even if neither of the switches are pressed one of the LEDs should be turning on and off because the switches are being held in the not pressed state. The example C file msp430g2xx3_P1_03.c found at http://coecsl.ece.illinois.edu/ge423/datasheets/MSP430Ref_Guides/Cexamples/MSP430G2xx3%20Code%20Examples/C/msp430g2xx3_P1_03_SE423.c is also helpful in figuring out how to setup of the port pin registers, but you do not want to copy the entire file into your C file. **Turn in your C code and demo your working code to your TA to get credit for this problem.**

11. If an oscilloscope was connected to the transmit line (TXD) of your MSP430G2553 microcontroller, what would be displayed if the baud rate was set to 115200 bits per second and only two ascii characters, "4" and "B", were sent in sequence? *Scoping the TXD line is not required but you are welcomed to figure out the waveform that way.*

12. Given the following SYS/BIOS application specifications, draw a *time loading diagram* for 20 milliseconds for the following application's processes. One Timer process runs every 1ms for 0.15 ms. One Clock process runs every 2 ms for 0.2 ms. One SWI process with priority 2 is posted by the Timer process every 4 ms and runs for 0.4 ms. A second SWI process with priority 1 is posted by the Timer process every 7 ms and runs for 0.55 ms. The Clock object has been assigned the SWI priority of 3. Notes: open the SYS/BIOS configuration tool and select SWI to see how SYS/BIOS prioritizes individual SWIs and Clocks. Also note that none of these processes start at time 0 seconds. They wait until their first sample period to attempt to launch their process.

13. Following the example given in lecture for a similar soda machine, draw the state transition diagram for a soda machine that:
    a. only dispenses one type of soda.
    b. the soda costs 40 cents.
    c. exact change is required.
    d. if a wrong coin is deposited, like a quarter when 30 cents has already been deposited, the wrong coin will be returned.
    e. the soda machine only accepts nickels, dimes and quarters.

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

1. Start reading about the ADC10 (analog to digital converter) unit of the G2553.
    a. Read the internal temperature sensor and print the result to TeraTerm.
    b. Read an external voltage source in the range 0 to 3.3V and print to TeraTerm. Be careful here, the G2553's inputs cannot handle voltages greater than 3.3V or less than 0V. Make sure to scope the input voltage before you connect it to the G2553's ADC.
2. Flash LEDs in different patterns depending on how long a switch is held in the low state.