Lab3

In this lab, you will perform an analog-to-digital conversion (ADC), and use that to make decisions in software.

1) Obtain the following from the electrical technician for use during this lab:
   a. A function generator
   b. Four stripped wires that can be used to access the connectors on the DSP board
   c. A BNC cable with alligator clips (or equivalent) to connect between the function generator and the DSP board

2) Generate a 1V peak-to-peak sinusoidal waveform with your function generator, with a dc bias of 0.5V and a frequency of 0.5Hz. Your sine wave should oscillate between zero and 1 volt. Note that the analog input channels are designed for signals ranging between 0 and 3 volts. Values less than that will be read as zero, while values larger than that will produce a full-scale conversion result. Of course, if you put in values that are too far out of range, you may damage the DSP board. Use your Bitscope (or other oscilloscope) to ensure that the proper signal is being generated by the function generator.

3) Power up the DSP board, and connect to the computer via the USB port. Start up CCS, and open up (from the example programs downloaded last week) the project: Example_2833xAdcSeqModeTest. This program acquires data from channel A0 of the ADC. So, you will need to connect the function generator as follows (verify the pin connections in the eZdsp F28335 Technical Reference):
   a. Signal lead to connector P9, pin 2
   b. Ground lead to connector P9, pin 5 (or other odd numbered pin)

4) When you run the program, you can look at the Watch Window like you did in Lab2 to see how your variables are changing. In this lab, the variable of interest is actually an array of values acquired by the ADC, called ‘SampleTable’. For this lab, we’re going to use the graphing capabilities of CCS. When the program has been built, loaded, and is ready to be run, select Debug/Real-time Mode. Then select View/Graph/‘Time/Frequency…’. You will then be presented with a table of properties to be filled in, such as follows:
Change the property values as follows:

- Graph Title = A/D Conversion Channel A0
- Start Address = SampleTable (can you verify that the first address of this array is in L4 SARAM?)
- Page = Data
- Acquisition Buffer Size = 1
- Index Increment = 1
- Display Data Size = 30
- DSP Data Type = 16-bit unsigned integer
- Q-value = 0
- Sampling Rate (Hz) = 50000
- Plot Data From = Left to Right
- Left-shifted Data Display = Yes
- Autoscale = On
- DC Value = 0
- Axes Display = On
- Time Display Unit = sample
- Status Bar Display = On
- Magnitude Display Scale = Linear
- Data Plot Style = Line
- Grid Style = Zero Line
- Cursor Mode = Data Cursor

5) When the graphical display appears in the Source Code Window, right click inside the graphical display area and select ‘Continuous Refresh’. Then run the program and watch the display window. You should be able to see your dc biased sine wave updating in the graphical display. You can also update the properties in the table ‘on the fly’, by again right clicking inside the graphical display area and selecting ‘Properties…’.

6) One of the details from the above procedure that you may have noticed is the relatively slow frequency of the sine wave that you are graphing in CCS. If you try to increase the frequency above 1Hz, reconstruction of the signal in the graphical display window may yield unacceptable quantization errors (or, if the frequency is too high, aliasing may occur). This is most likely a limitation of the continuous refresh capability in CCS. To get a better sense of how fast the ADC works, halt the program and deselect ‘Real-time Mode’. Keep your frequency at 0.5Hz for your sine wave. Select View/Memory. You will see a new window pop up. In the new window, there should be a text bar with the words ‘Enter An Address’. Type in ‘&SampleTable’. You should then see the contents of memory starting with the initial address of the buffer storing your ADC values.

7) Run the program without graphing, and without continuous refresh, for a few seconds. Halt the program. You should see the contents of memory where your buffer is located change. Scroll down in memory. How many values are stored in your buffer? Do the values change significantly between the beginning of your buffer in memory and the end of the buffer (where you no longer are storing ADC values)?
8) Change the sine wave frequency to 500Hz, and rerun the step above. Look at the buffer values again. Change the sine wave frequency to 5kHz, and rerun the step above. Can you estimate what sine wave frequency limit your ADC can handle (considering the Nyquist sampling frequency)?

9) Your task in this lab is to perform the following:
   a. Acquire a sinusoidal signal through the ADC
   b. Determine if the signal at any one time is below a threshold (i.e., nearing zero)
   c. If the signal is below that threshold, output a signal to one of the GPIO pins.

   In essence, you are building a modified zero crossing detector for the sine wave you have inputted into the DSP board:

10) You are now able to acquire a signal through the ADC, perform an operation, and output a control signal based on that input signal. This ends Lab 3. Be sure the DSP board is disconnected from the computer and from the function generator prior to removing power.