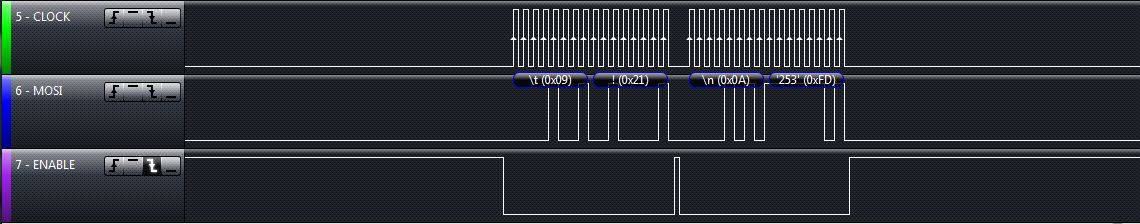
**Description of each Part**

1. In Part 1, analog signal data will be acquired from an ADC channel that we choose (ADC3) at a sampling rate. The general purpose timer that is used alongside the ADC channel will trigger the ADC sequence and acquire data from the input. The values will then be stored in a global circular buffer and will always contain the most recent 128 values. The moving average will also be determined (which consists of the last 4 ADC values). These values will be used to generate outputs from the DAC (one being the original reconstructed signal and the other will be the one’s complement). Also, a switch interrupt handler is introduced it will print the circular buffer. To prevent multiple prints of the data from pressing/holding the switch, the interrupt for the switch was disabled once it was detected. We mostly followed code from ADC2 example from our resources for this part.
2. In Part 2, the purpose was to output a waveform on the DAC using the micro-DMA that is supplied as part of the TIVA set. The sine wave data arbitrarily created using a function and all the data points were stored in an 8-bit array in memory. Instead of having the processor send the values to the DAC, the DMA commands were used instead to grab the data points from SRAM and then sent out to the SSI/SPI Data Registers, which would be transferred to the DAC.
3. In Part 3, we implemented ping-pong buffers to analyze whether data being sent in was determined to be from a sine, square, or triangular waveform. To analyze it, we chose to find two points (one high and one low). For frequency, we found the period between the two points and divided the sampling rate by the period, which would give us the frequency. The sampling rate is related to the period and frequency. To determine the wave form, a counter and slope was used to check. We first checked the slope of the graph, if it was too big then it was a square. If it wasn’t we checked where most of the values laid. A triangle waveform is more bottom heavy, which meant that the values laid below the threshold and a sine wave was more top heavy. We implemented these ideas in our algorithm to determine wave form and frequency.

**Salae Logic Analysis**



Using the Logic Analyzer, we see how we send data from the SPI interface. It matches the way we sent data because the processor is sending to both addresses 0x09 and 0x0A. The values following each is the data that is being sent; one is the actual value in hex and the other is the one’s complement.

**Conclusion**

In our last lab experiment, the purpose of the lab was to introduce us to the Digital-to-Analog Converter, Analog-to-Digital channels of the TIVA, and the DMA. This entire lab showed how TIVA can be used for signals processing and analysis. The first part showed how to read analog input and convert it to discrete signals using the ADC channels onboard the TIVA. We would use the discrete data points to reconstruct the same exact signal that we received. In the second part, we were to use the DMA (that acts as a slave to the processor for obtaining data) to grab data points from a software generated sine wave. We would use these data points to output it to the DAC and obtain a sine wave. The last part is determining what kind of wave is being inputted from the function generator. We have to create algorithm to determine the frequency and to see what kind of function is inputted.

The most difficult portion of this experiment was part 2. After talking to a few classmates, the DMA example that we find within the TIVA example folder did not work when we used it with the 1.1 version. This required us to download an patch to upgrade to 2.0+ version, which allowed us to use certain functions. After that, it became a relatively easy because we based our part 2 off the example code.

The next challenging part was creating the algorithm to determine both the frequency of the incoming waves and the type of function inputted from the function generator. There definitely was a variety of methods to choose from to determine what the frequency and function inputted was. For us, to determine the frequency, we found data points and find the time in between the points (which inverted will give us the frequency). To find the type of function that was generated, we set up threshold values and determined how many values were above/below the threshold. It was very interesting to see how to use digital signals processing on a microcontroller.

Overall, this lab wasuseful in showing how the TIVA board can be used to analyze signals.