Lab 2 - ADC/DAC

EE138

Colin Chen & Daniel Graham

3/5/2015

Introduction

The objective of this lab is to interface with the ADC/DAC modules of the SAMD20 microcontroller board.

- Using the ADC, read the voltage drop across a potentiometer and display the reading on the 7 segment display
- Using the DAC, produce a sine wave of 1kHz

By completing the following tasks, our group will demonstrate that we have knowledge to properly set-up, and use the ADC/DAC modules of the SAMD20.

Procedure/Methodology

Setup

For the setup of both the DAC/ADC, certain registers need to be properly set up for the implementations that we desire. For the ADC, CTRLA, CTRLB, REFCTRL, AVGCTRL, SAMPCTRL, INPUTCTRL, are the primary registers that we need to worry about. For the DAC, mainly CTRLA, CTRLB, and REFCTRL are used. Both DAC and ADC needed PA02 to be properly muxed/configured to be an input or an output of the ADC/DAC. CTRLA's primary function is the enabling and disabling of both modules, the modules will not be allowed to be reset/setup while it is enabled so to properly configure all the registers, the CTRLA will be needed to disable the ADC/DAC. CTRLB's is used to configure peripheral clock divisions and bit size of the results. REFCTRL allows us to set the reference voltage of the ADC/DAC and we want this to be set to 3.3V. The REFCTRL does not have this as a function so we need to set it to half of Vcc and half the gain using the INPUTCTRL register. INPUTCTRL is in charge of selecting the inputs and the gain of the ADC.

ADC

For the ADC, we configured the registers so that the reference voltage would be half of Vcc (the highest available setting outside of using an external reference) and set the gain to half of the incoming voltage. The way we set up our registers, the input of our ADC would divide the input between 0 and Vcc by 2048 different divisions. We ratioed the 7 segment reading to scale properly so that the largest ADC reading would be equivalent to the measured reading of PA02. The result register is read by the microcontroller

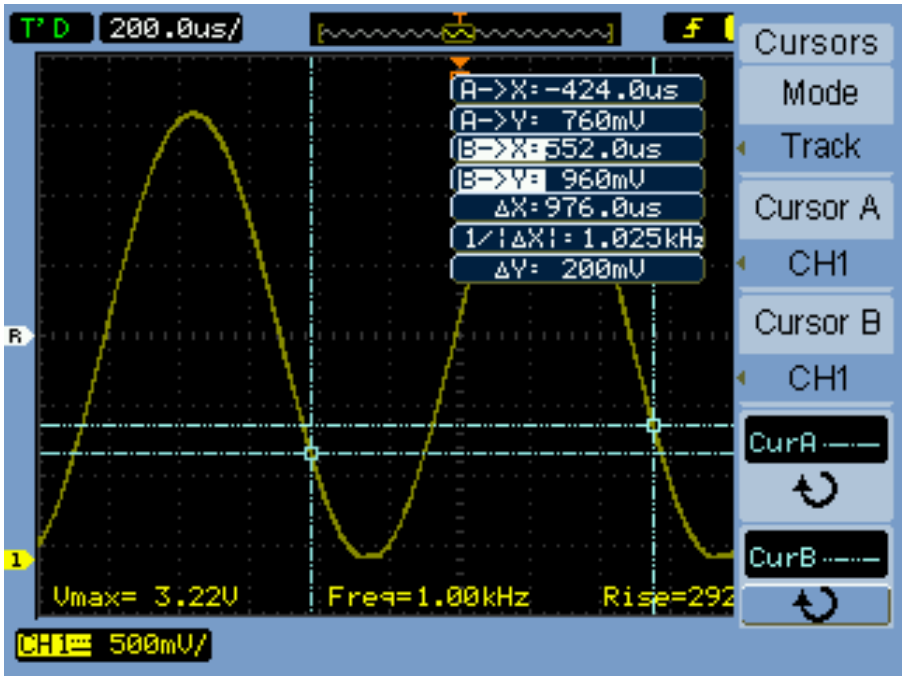
DAC

The DAC begins with configuring the DAC peripheral. We started this by creating a pointer to the DAC and configuring PORT_PA02 to properly use that peripheral via PINCFG and PMUX. The code waits for synchronization and then sets the reference voltage via CTRLB. The code waits for synchronization again and then enables the DAC output via CTRLA and output to

external pin via CTRLB. We set both pins PORT_PA02 and PORT_PA13 to output, and PORT_PA13 to output a constant 3.3V.

Next we created the sine wave. By using the maximum input of the DAC (0x3FF) as the peak and 0 as a minimum, we used a function to map one cycle of a sine wave with a distinct # of horizontal samples. This included a conversion from degrees to radians. Adjusting the clock rate of the main clock wasn't precise enough to create a specific frequency, so by adjusting the amount of horizontal samples, we were able to dial in a 1kHz sine wave. By increasing the number of samples, we were able to decrease the frequency of the waveform, and by decreasing the number of samples, we were able to increase the frequency (ultimately sacrificing resolution of the sine wave).

After synchronization has occurred again, this constructed sine wave was then looped to the DATA register within the DAC peripheral, thus outputting a sine wave. The output was verified via oscilloscope.



Conclusion/Results

With the ADC, we were able to read the voltage drop across the potentiometer properly (verified with an external multimeter) up to two digits. We exhibited good accuracy near the outer bounds of the readings (0V/3.3V) but exhibited up to 20mV of inaccuracy near the 1-2V range. We believe this inaccuracy is due the hardware within the ADC itself rather than a conversion issue as this problem was exhibited in other groups as well. For the 7-segment display, our group experienced a few issues with brightness due to how our display code was set up. We imported the code from our previous lab, but while our code worked great with our

previous lab, it was not nearly as bright for this lab due to the time it takes for computation and reading of our ADC. In order to solve this issue, the code was changed to take the time for computation into account. The 7 segment display was significantly brighter after this change.

The DAC was able to produce the sine wave of 1 khz, verified by an oscilloscope. To implement this, we first programmed the DAC to output a triangle wave (due to the simplicity of the equation needed to produce it) before creating a function to model a sine wave. Volume of the sound being played is controlled via the potentiometer.