Lab 5 Digital Filter

EE138

Colin Chen & Daniel Graham

4/30/2015

Introduction

In this lab we will be using the Timer/Counter (TC), ADC, and DAC peripherals of the SAMD20 microcontroller to filter an analog signal based on the following parameters:

- Create a first order low pass filter with a rejection frequency of 100Hz and a sampling frequency of 1000Hz.
- Create a second order notch filter with a rejection frequency of 60Hz, a bandwidth of 10Hz, and a sampling frequency of 1000Hz.

To successfully complete this task we will need to perform calculations in MATLAB in order to determine a discrete transfer function. Once we are able to create the discrete transfer functions for each required filter we will take an analog input signal, and pass it through the analog digital converter. Once the input is acquired we will adjust the signal with our discrete transfer function, and then we will output the filtered waveform via the DAC.

Procedure

MATLAB

Creating the digital filter began with determining the form of the first and second order required filters. Once we determined the proper form, we chose our cutoff frequency for the first order filter, implemented it, created a bode plot, and converted the form from continuous to discrete time using the following MATLAB code:

```
>> Hc = tf( [2*pi*100],[1,2*pi*100])

Hc =

628.3
---------
s + 628.3

Continuous-time transfer function.

>> bode(Hc)
>> c2d(Hc,0.001)

ans =

0.4665
----------
z - 0.5335

Sample time: 0.001 seconds
Discrete-time transfer function.
```

Once we finished with the first order lowpass filter we inputted the form for the second order notch filter, implemented it, created a bode plot, and converted the form from continuous to discrete time using the following MATLAB code:

```
>> H = tf([1 0 (2*pi*60)^2],[1 (2*pi*60)/6 (2*pi*60)^2])

H =

     s^2 + 1.421e05
 ------------------------
 s^2 + 62.83 s + 1.421e05

Continuous-time transfer function.

>> bode(H)
>> c2d(H,0.001)

ans =

  z^2 - 1.862 z + 0.9986
 ----------------------
  z^2 - 1.803 z + 0.9391

Sample time: 0.001 seconds
Discrete-time transfer function.
```

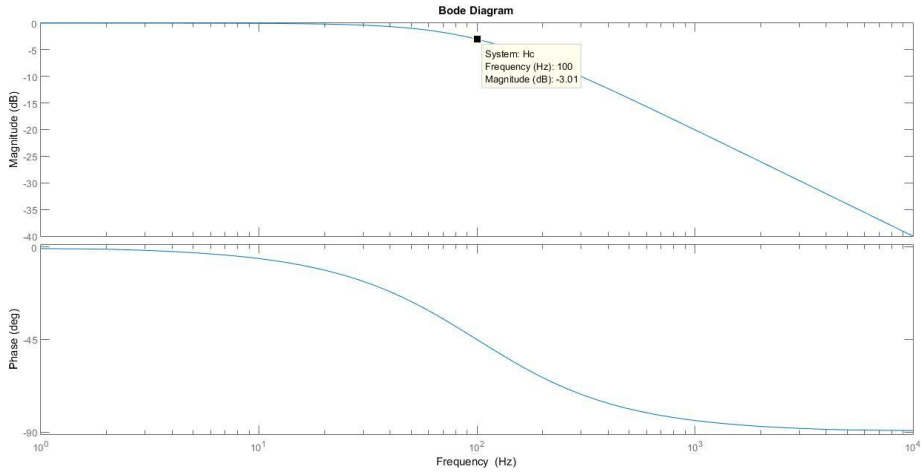
Once we finished with the second order filter we implemented the functions within Atmel Studio.

Implementing the Filters in Atmel Studio

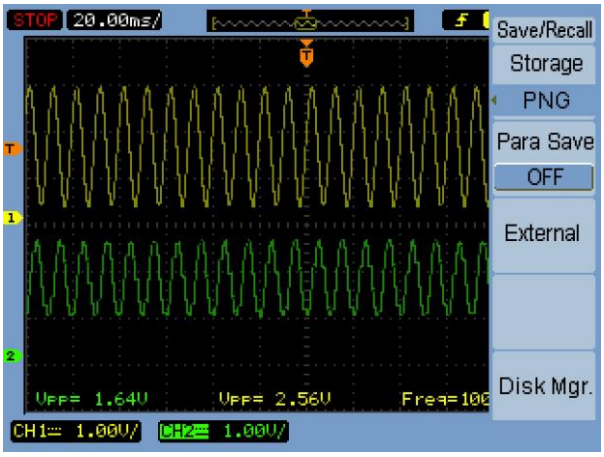
Implementing the filters within Atmel Studio began with copying the ADC, DAC, and TC code from previous labs in order to implement in this lab. Next we created a minimal state machine in order to toggle between either filter. We implemented the discrete time filters within Atmel Studio. After creating the filters, we measured and calculated the amount of time it would take for each function (ADC, DAC, Filter) to complete for each filter. Lastly, we took screenshots on the oscilloscope in order to determine the bandwidth and aliasing for each of the filters.

Results

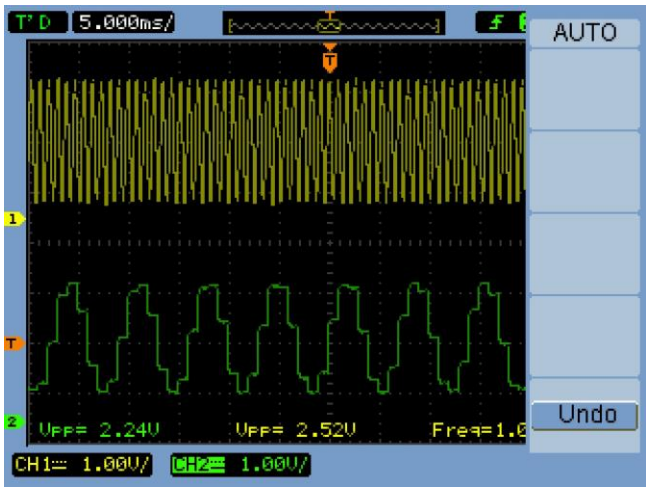
The first order lowpass filter worked as expected. The bode plot of the first order lowpass can be seen below.



As the picture above shows, the -3dB dropoff occurs at exactly 100Hz, however the actual output of the filter was different than the calculated value. The -3dB dropoff can be seen in the screenshot below. The calculated value deemed the output should drop to 1.55, thus our output is off by 0.1V.



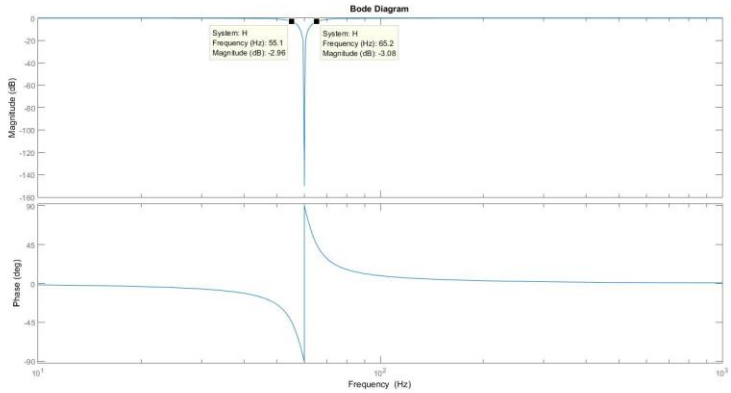
Once we determined the cutoff occurred at approximately 100Hz, we increased the frequency to 1000Hz (twice the sampling rate) in order to observe aliasing. This can be seen in the screenshot below.



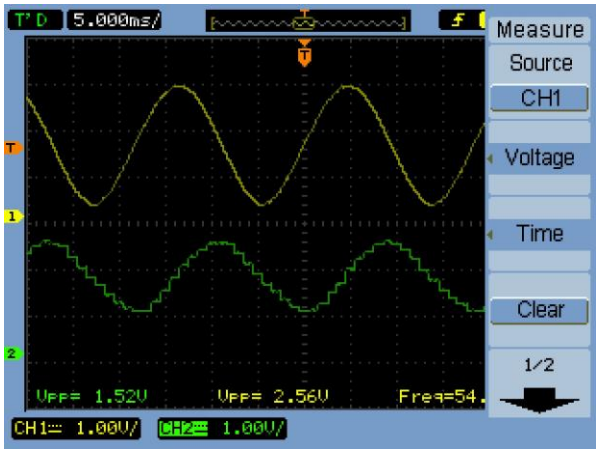
As expected, the frequency and the aliasing of the filtered signal is incorrect as the frequency increases above twice the sampling rate. After we observed the aliasing we calculated the amount of time each process within the first order filter took. This can be seen in the table below.

Filter	ADC	DAC	Filter Time
100Hz	10us	2.4us	137us

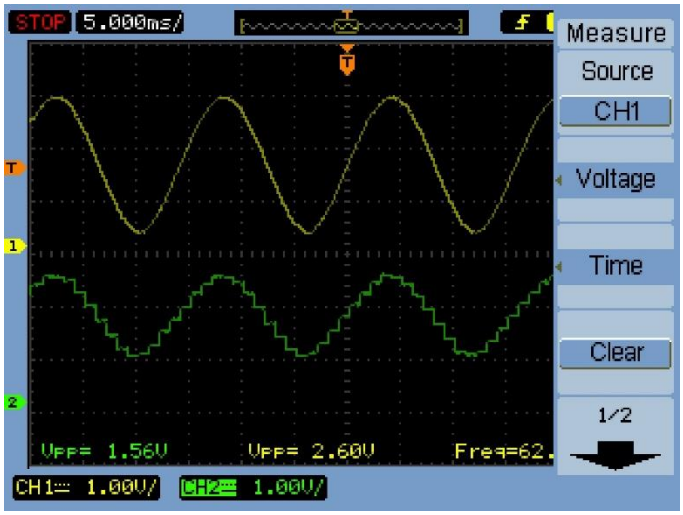
Once we finished working with the first order 100Hz lowpass filter we proceeded to work on the 60Hz second order notch filter. We began by analyzing the bode plot which can be seen on the following page.



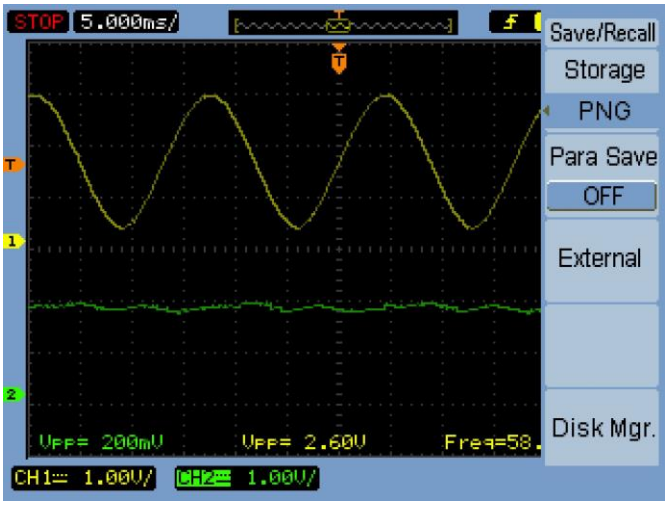
As expected, the notch filter filters out 60Hz with a bandwidth of ~10Hz. We then proceeded to test the bandwidth of the actual filter performance. The left side 3dB drop of the input signal can be seen in the picture below.



Once we determined the left side 3dB drop we measured the right side 3dB drop in order to calculate the actual bandwidth of the notch filter. The right side 3dB drop of the input signal can be seen in the picture below.



The calculated bandwidth of the second order was approximately 8Hz, which is less than the theoretical value. This could be due to the input or output scaling. Once we determined the bandwidth we measured the output at the desired notch frequency. The screenshot can be seen below.



As expected, the output signal decreases considerably at ~60Hz. Based on discussions with my peers, we determined that most people actually saw the cutoff at ~58-59Hz instead of the desired 60Hz. Once we determined the cutoff frequency was correct we increased the frequency in order to show the aliasing effect on the signal. We increased the signal to 1000Hz and the output can be seen below.

As expected, the frequency and the aliasing of the filtered signal is incorrect as the frequency increases above twice the sampling rate. After we observed the aliasing we calculated the amount of time each process within the first order filter took. This can be seen in the table below.

Filter	ADC	DAC	Filter Time
60Hz	10us	2.4us	340us

The results of the timing analysis shows that the 60Hz filter requires more processing time. This is to be expected as it is a second order filter. All tasks were completed successfully.