Lab 1.5 Sampling Systems

# Hardware Setup:

The Hardware used in this lab mainly consisted of an STM32F4 Discovery Board, a V4.1 Serial Analog Interface (SAI) board, and an analog discovery. The STM32F4 board was used to control the SAI board and analyze the interrupt timing with a pin of the analog discovery on the LED pin. The SAI board was used to take the input signal generated by the analog discovery, sample it at a rate of 5Khz, and then reproduce the signal back out with an IIR filter implemented. Both the input and output signals were monitored by the analog channels of the Analog Discovery. A picture of the setup is below.

A circuit board

Description automatically generated

# Software:

The code used consisted of the same code used in Lab 1, but with modifications to the AdcDriver. The code for the AdcDriver.h is located in Appendix A, and the rest of the code can be found online at <https://github.com/bstonebraker96/ece647_lab1_5/>.

# Questions:

## How well do the frequency response match the response posted to the web page?

Looking at figures 2 and 3 below, the expected result and my result respectively, it’s clear that the beginning frequencies (100-900Hz) struggle a bit with matching the wave form, but that could be attributed with the difficulties I was experiencing with using the lab equipment, as for some reason the analog discovery was putting out data was off and the data I collected had to be remeasured at least once. The table of data is located in Appendix B.

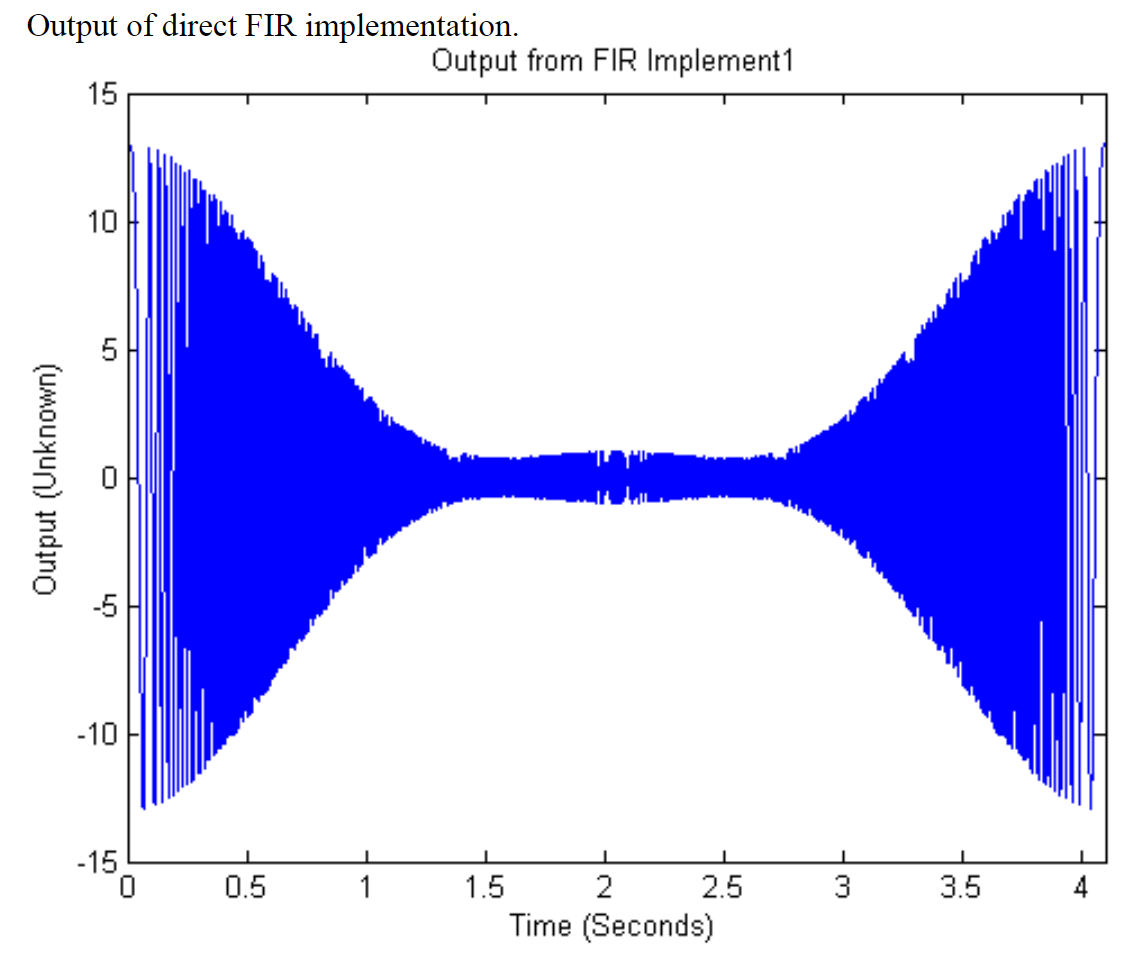


Fig. 2 – Expected response shape of results

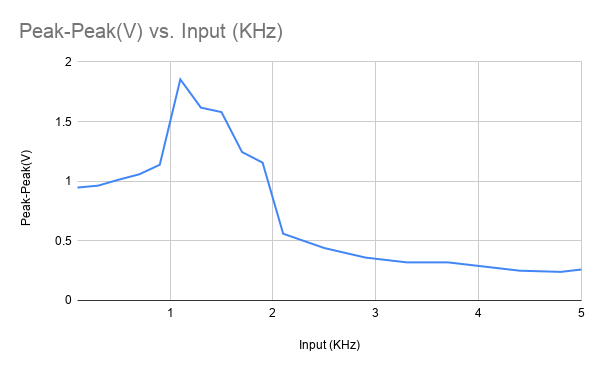


Fig. 3 – Plot of Peak Voltage vs frequency in KHz

## Based on the time required to run the filtering code, how large of a filter can be implemented assuming we would want to use 80 of the 100µs for filtering?

The maximum size of the filter can be 467 Orders large.

Appendix A

*// ADC Support.*

int Channel2, Channel1;

**const** float b0 = 0.1, b1 = 0.2, b2 = 0.1,a1 = -1.0, a2 = 0.5;

float IIR2\_y[2] = { 0.0 };

float IIR\_Implement2(float x){

float y; *// output*

*// Implement difference equation*

y = b0 \* x + IIR2\_y[0];

IIR2\_y[0] = b1 \* x - a1 \* y + IIR2\_y[1];

IIR2\_y[1] = b2 \* x - a2 \* y;

**return** y;

}

*// IIR\_Implement2*

*// Compute output using Transposed*

**const** int LED\_Indicator = 2;

*// DAC service.*

void DAC\_Data\_Send( int Ch1, int Ch2 )

{

DAC->DHR12RD = (Ch2<<16) | Ch1; *// Send data over to DAC*

DAC->SWTRIGR = 3; *// Software trigger DAC*

}

*// ADC\_IRQHandler Function â€“ Only called when conversion is complete.*

void ADC\_IRQHandler(void){

LED\_On( LED\_Indicator ); *// Set output pin showing Interrupt.*

Channel1 = ADC1->DR - 2048; *// Pull Data from ADC, channel 1,*

Channel2 = ADC2->DR - 2048; *// channel 2.*

*// Filtering could be added here to change Channel1 and Channel2*

*//Channel1 = (int) IIR\_Implement2((float)Channel1);*

*// Compute output using Transposed*

DAC\_Data\_Send( Channel1 + 2048, Channel2 + 2048 );

LED\_Off( LED\_Indicator ); *// Turn off Interrupt Flag.*

} *// End of ADC\_IRQHandler*

*// init\_adc Function â€“ Sets up ADC, Dac and Timer.*

void Adc\_Dac\_Init( )

{

RCC->APB2ENR |= RCC\_APB2ENR\_ADC1EN | RCC\_APB2ENR\_ADC2EN; *// Clock enabled for ADC1.*

RCC->AHB1ENR |= RCC\_AHB1ENR\_GPIOAEN; *// Enable GPIOA clock*

*// Enable DAC and Timer 2, 5 clock*

GPIOA->MODER |= (0x00000ff0); *// PA.2-5 as Analog*

*// Set CCR for ADC*

ADC->CCR |= 0x030f02; *// Maximum Prescaler and Max delay between samples*

*// Dual simultaneous mode.*

*// Set CR1 for ADC 1 and 2*

ADC1->CR1 |= 0x000820; *// Enable Discontinuous and EOC interrupt.*

ADC2->CR1 |= 0x000800; *// Discontinuous*

*// Set CR2 for ADC 1 and 2*

ADC1->CR2 |= 0x16000401; *// Set Trigger to Tim2, enable EOCS and turn on ADC1*

ADC2->CR2 |= 0x00000001; *// Turn on ADC2*

*// Set SMPR2 for ADC 1 and 2*

ADC1->SMPR2 |= 0x000000f8; *// Set sampling to maximum*

ADC2->SMPR2 |= 0x000000f8; *// Set sampling to maximum*

*// Set SQR1 for ADC1*

ADC1->SQR1 = 0x00000000; *// Set for 1 Conversions*

ADC2->SQR1 = 0x00000000; *// Set for 1 Conversions*

*// Set SQR3 for ADC1*

ADC1->SQR3 = 0x00000002; *// Set Initial sequence to channels 1 then 2*

ADC2->SQR3 = 0x00000003; *// Set Initial sequence to channels 1 then 2*

NVIC\_EnableIRQ( ADC\_IRQn ); *// Enable ADC interrupts.*

*// Enable both DAC's and set them to trigger on Tmr4.*

RCC->APB1ENR |= RCC\_APB1ENR\_DACEN;

DAC->CR |= DAC\_CR\_EN1

| DAC\_CR\_EN2;

RCC->APB1RSTR |= RCC\_APB1RSTR\_TIM2RST;

RCC->APB1ENR |= RCC\_APB1ENR\_TIM2EN;

}

*// This macro will start the adc, using the software trigger.*

#define Fire\_Adc() (ADC1->CR2 |= ADC\_CR2\_SWSTART)

Appendix B

|  |  |
| --- | --- |
| Input (KHz) | Peak-Peak(V) |
|  |  |
| 0.1 | 0.947 |
| 0.3 | 0.963 |
| 0.5 | 1.013 |
| 0.7 | 1.058 |
| 0.9 | 1.138 |
| 1.1 | 1.854 |
| 1.3 | 1.617 |
| 1.5 | 1.58 |
| 1.7 | 1.245 |
| 1.9 | 1.156 |
| 2.1 | 0.56 |
| 2.5 | 0.44 |
| 2.9 | 0.36 |
| 3.3 | 0.32 |
| 3.7 | 0.32 |
| 4.1 | 0.28 |
| 4.4 | 0.25 |
| 4.8 | 0.24 |
| 5 | 0.26 |