EE 311 **February 5, 2018**
Assignment 04 **Due: February 22, 2018**

For this assignment you will implement a 12th order FIR Hamming windowed filter. The coefficients for the filter can be obtained from the following m-file.

```
%FIRHamming.m
fs = 11025;
N = 12;
[num den] = fir1(N, 2000/(fs/2), hamming(N+1));
[H f] = freqz(num, den, 1024, fs);
figure(1);clf;
db = 20*log10(abs(H));
figure(1);clf;
plot(f, db);
figure(2);clf;
plot(f, abs(H));
figure(3);clf;
zplane(num, den);
for i=1:N+1
  fprintf(1,'%12.9f \n', num(i));
end
```

Figure 1

m-file which produces the coefficients for a 12th order FIR filter using a Hamming window.

Use the coefficients produced by this Matlab m-file with the c-program on the following page to implement your filter on the ARM M4 Nucleo board.

After implementing your filter, you should collect data to plot the frequency response of the filter. Do this by measuring the gain of the filter from 0Hz to fs/2Hz in increments of 200 Hz. Use MatLab to plot your measured data along with the calculated frequency response from the m-file in Figure 1 above.

Turn in the following:

1. Title page with your name, date turned in, and assignment number.
2. A tabulated list of your measurements and the calculated gain.
3. A Matlab plot that has the measured frequency response and the frequency response calculated from Figure 1 above on the same graph. Use different colors or line styles and clearly mark which plot is measured and which is calculated.
4. A signed verification sheet (see form attached to this assignment).

```
/*FIRHamming
*/
#include "stm32f446.h"
const float b0 = 0.002225486;
const float b1 = -0.004942903;
const float b2 = -0.024235444;
const float b3 = -0.015621047;
const float b4 =  0.092478636;
const float b5 =  0.269766703;
const float b6 =  0.360657138;
void InitializeClock(void);
int main()
  {int uInt, yInt, tmp;
   float u, y;
   float u1, u2, u3, u4, u5, u6, u7, u8, u9, u10, u11, u12;

   //Clock bits
   InitializeClock();
   RCC_AHB1ENR |= 1;         //Bit 0 is GPIOA clock enable bit
   RCC_APB1ENR |= (1 << 29); //Bit 29 is DAC clock enable bit
   RCC_APB2ENR |= 0x100;     //Bit 8 is ADC 1 clock enable bit
   RCC_APB1ENR |= (1 << 4);  //Enable peripheral timer for timer 6
   //I/O bits
   GPIOA_MODER |= 0x4000;    //Bits 15-14 = 01 for digital output on PA7
   //OTYPER register resets to 0 so it is push/pull by default
   GPIOA_OSPEEDER |= 0xC000;  //Bits 15-14 = 11 for high speed on PA7
   //PUPDR defaults to no pull up no pull down
   GPIOA_MODER |= 0xF00;     //PA4-PA5 are analog
   GPIOA_PUPDR &= 0xFFFFF0FF;//Pins PA4 PA5 are no pull up and no pull down
   //DAC bits
   DAC_CR |= 0x3E;           //Bits 3, 4, 5 = 111 for software trigger ch1
                             //Bit 2 = 1 for Ch 1 trigger enabled
                             //Bit 1 = 1 for Ch 1 output buffer enabled
   DAC_CR |= 1;              //Bit 0 = 1 for Ch 1 enabled
   //ADC bits
   ADC1_CR2 |= 1;             //Bit 0 turn ADC on
   ADC1_CR2 |= 0x400;         //Bit 10 allows EOC to be set after conversion
   ADC_CCR |= 0x30000;       //Bits 16 and 17 = 11 so clock divided by 8
   ADC1_SQR3 |= 0x5;          //Bits 4:0 are channel number for first conversion
                             // Channel is set to 5 which corresponds to PA5
   //Timer 6 bits
   TIM6_CR1 |= (1 << 7);   //Auto reload is buffered
   TIM6_CR1 |= (1 << 3);   //One pulse mode is on.
   TIM6_PSC = 0;           //Don't use prescaling
   TIM6_ARR = 8163;        //(180 MHz/2)/8163 = 11025.3 Hz
   TIM6_CR1 |= 1;          //Enable Timer 6
   tmp = 0;
   //Main program loop
   while(1)
     {GPIOA_ODR = tmp;      //Set bit 7 to 1
      tmp = ~tmp;
     ADC1_CR2 |= 0x40000000;      //Bit 30 does software start of A/D conversion
     while((ADC1_SR & 0x2) == 0); //Bit 1 is End of Conversion
          uInt = ADC1_DR;
     u = ((float)(uInt & 0xFFF))/(float)4095.0;
     y = b0*(u + u12) + b1*(u1 + u11) + b2*(u2 + u10) +
                 b3*(u3+u9) + b4*(u4 + u8) + b5*(u5 + u7) + b6*u6;

        yInt = (int)(1500*(y+1));   //Data to D/A
     DAC_DHR12R1 = yInt & 0xFFF; //Converted number to D/A
     DAC_SWTRIGR |= 0x1;         //Start the D/A conversion
```

```
    u12 = u11;
    u11 = u10;
    u10 = u9;
    u9 = u8;
    u8 = u7;
    u7 = u6;
    u6 = u5;
    u5 = u4;
    u4 = u3;
    u3 = u2;
    u2 = u1;
    u1 = u;
    while((TIM6_CR1 & 1) != 0); //Wait here until timer runs out
    TIM6_CR1 |= 1;              //Restart timer
   }
}

     //This function resets the system clock to 180 MHz.

void InitializeClock()
    {RCC_CFGR = 0x00000000;         //Reset Clock Configuration Register
     RCC_CR &= 0xFEF6FFFF;          //Reset HSEON, CSSON and PLLON Bits
     RCC_CR |= (1 << 16);           //Turn on HSE clock
     while((RCC_CR & (1 << 17)) == 0); //Wait until HSE is ready
     RCC_CR |= (1 << 19);
     RCC_PLLCFGR = 0x27405A08;      //Set PLLP = 0, PLLN = 360, PLLM = 8,
                                    //PLLQ = 7, PLL Src = HSE
     FLASH_ACR &= 0xFFFFFFF8;       //Set flash wait states to 5
     FLASH_ACR |= 0x5;
      RCC_CR |= (1 << 24);           //Enable PLL on
      while((RCC_CR & (1 << 25)) == 0); //Wait for PLL to lock on
     RCC_CFGR = 0x9402;             // APB2/2, APB1/4, AHB/1
    }
```

Verification Sheet

EE 311
Assignment 4

I verify that

_____

has successfully implemented a lowpass FIR filter that has a sample frequency of 11025Hz

_____
Signature (Blandford or Cron)