The c-program on the following pages implements a 30th order FIR low pass filter for the ARM Cortex STM32F4 board. The sample frequency is about 11025 Hz and a cutoff frequency of 2 KHz. The Keil project file for the program is on the website as FIR30.zip.

A) Download the zipped file, compile it, and load it into the ARM Cortex board. Verify that it runs as a low pass filter.

B) The filter was designed from MATLAB® using the following code:

```matlab
N = 30;
familiar = 2000; fs = 11025; % cutoff and sample frequency
[num den] = fir1(N,familiar/(fs/2),hamming(N+1));
```

Alter the MATLAB® code so that the cutoff frequency is 1 KHz and the window is a Blackman window. Implement your redesigned filter on the ARM Cortex board and have it verified by your instructor.

Turn in the following:
1. Cover page with your name, the date turned in, and the assignment number.
2. The MATLAB® code which produces a plot of the original filter frequency response and your modified filters frequency response on the same axis. Use two different colors. You should plot the magnitude for both filters on one plot and the phase for the two on a second plot.
3. The plots from your MATLAB® code with appropriate labels.
4. A listing of your modified c-code for the new filter.
5. A table which lists the coefficients for the original filter and your modified filter side-by-side.
6. A signed verification sheet that your modified filter worked. For the verification sheet you will need to use a signal generator input sinusoids. Note that the sinusoidal input must be in the range of $0 < v < 3.3$ to avoid damage to your chip.
Verification sheet
EE 311 STM Assignment 2

Student _____________________ has demonstrated a working digital filter running on the ARM Cortex board that produces an output signal indicating a low pass filter with a cutoff frequency of 1 KHz

Instructor____________________________   Date ______________________

Blandford, Cron, or Randall
This program implements a filter using integer arithmetic. This filter was designed in MatLab as a 30th order FIR filter with

```matlab
N = 30;
fc = 2000; fs = 11025; % cutoff and sample frequency
[num den] = fir1(N,fc/(fs/2),hamming(N+1));
```

```c
#include "stm32f4vg07.h"
const float b0 = -0.001671740;
const float b1 = -0.000505838;
const float b2 = 0.002282468;
const float b3 = 0.003995449;
const float b4 = -0.000191671;
const float b5 = -0.009090046;
const float b6 = -0.010429167;
const float b7 = 0.005909405;
const float b8 = 0.026566393;
const float b9 = 0.019112863;
const float b10 = -0.027066681;
const float b11 = -0.066756728;
const float b12 = -0.026576262;
const float b13 = 0.116155203;
const float b14 = 0.286640849;
const float b15 = 0.363251007;
int main()
{
    int uInt, yInt;
    float u, y;
    float u1, u2, u3, u4, u5, u6, u7, u8, u9, u10;
    float u11, u12, u13, u14, u15, u16, u17, u18, u19, u20, u21;
    float u22, u23, u24, u25, u26, u27, u28, u29, u30;

    //Clock bits
    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
    GPIOA_MODER |= 0x4000;    //I/OPER register resets to 0 so it is push/pull by default
    GPIOA_MODER |= 0x300;     //Bits 15-14 = 11 for high speed on PA7
    ADC_CR2 |= 1;             //Bit 0 turn ADC on
    ADC_CCR |= 0x300;         //Bits 16 and 17 = 11 so clock divided by 8
    ADC_SQR3 |= 0x5;          //Bits 4:0 are channel number for first conversion
    //DAC bits
    DAC_CR  |= 0x3;           //Bits 3, 4, 5 = 111 for software trigger ch1
    DAC_CR  |= 1;             //Bit 0 = 1 for Ch 1 enabled
    //ADC bits
    ADC_CR2 |= 1;             //Bit 0 turn ADC on
    ADC_CCR |= 0x400;         //Bit 10 allows EOC to be set after conversion
    ADC_SQR3 |= 0x5;          //Bits 4:0 are channel number for first conversion
    //Timer 6 bits
    TIM6_CCR1 |= (1 << 7);    //Auto reload is buffered
    TIM6_CCR1 |= (1 << 3);    //One pulse mode is on.
    TIM6_PSC = 0;            //Don't use prescaling
TIM6_ARR = 7619;  // (168 MHz/2)/7619 = 11025 Hz
TIM6_CR1 |= 1;    // Enable Timer 6

// Main program loop
while(1)
{
    GPIOA_ODR |= (1 << 7);  // Set bit 7 to 1
    ADC_CR2 |= 0x40000000; // Bit 30 does software start of A/D conversion
    while((ADC_SR & 0x2) == 0); // Bit 1 is End of Conversion
    uInt = ADC_DR;
    u = ((float)(uInt & 0xFFF))/(float)4095.0;
    y = b0*(u + u30) + b1*(u1 + u29) + b2*(u2 + u28) + b3*(u3+u27) + b4*(u4 + u26) + b5*(u5 + u25) + b6*(u6 + u24) + b7*(u7 + u23) + b8*(u8 + u22) + b9*(u9 + u21) + b10*(u10 + u20) + b11*(u11 + u19) + b12*(u12 + u18) + b13*(u13 + u17) + b14*(u14 + u16) + b15*u15;
    yInt = (int)(2048*y); // Data to D/A
    DAC_DHR12R1 = yInt & 0xFFF; // Converted number to D/A
    DAC_SWTRIGR |= 0x1;         // Start the D/A conversion

    u30 = u29;
    u29 = u28;
    u28 = u27;
    u27 = u26;
    u26 = u25;
    u25 = u24;
    u24 = u23;
    u23 = u22;
    u22 = u21;
    u21 = u20;
    u20 = u19;
    u19 = u18;
    u18 = u17;
    u17 = u16;
    u16 = u15;
    u15 = u14;
    u14 = u13;
    u13 = u12;
    u12 = u11;
    u11 = u10;
    u10 = u9;
    u9 = u8;
    u8 = u7;
    u7 = u6;
    u6 = u5;
    u5 = u4;
    u4 = u3;
    u3 = u2;
    u2 = u1;
    u1 = u;
    GPIOA_ODR &= ~(1 << 7);  // Set bit 7 to 0
    while((TIM6_CR1 & 1) != 0); // Wait here until timer runs out
    TIM6_CR1 |= 1;              // Restart timer
}
}