Systems & Biomedical Engineering Department

Faculty of Engineering

Cairo University

Digital Filter Design

Ahmed Salah El-Din	|Sec: 1|B.N: 5
Salma Ayman Ahmed	|Sec: 1|B.N: 37
Abdullah Mohammed Sabry	|Sec: 2|B.N: 8
Nancy Salah El-Din	|Sec: 2|B.N: 37
Ehab Abdelrahman	External Student

Submitted to: **Dr. Ahmed Ehab**

2019 – 2020

Task Description

This project aims to design a digital filter and apply it on a noisy signal. Using FIR filter and 3 different types (Low Pass, High Pass and Notch) filters. We could toggle between them using switches to choose which type of filter we want to apply.

Used Components

- 8051 Microcontroller (AT89C51)
- ADC0808
- DAC0808
- Op-Amp 741
- Op-Amp LF351
- Capacitors
- Resistors
- Crystal

ADC

Here's the schematic design of the ADC chip.

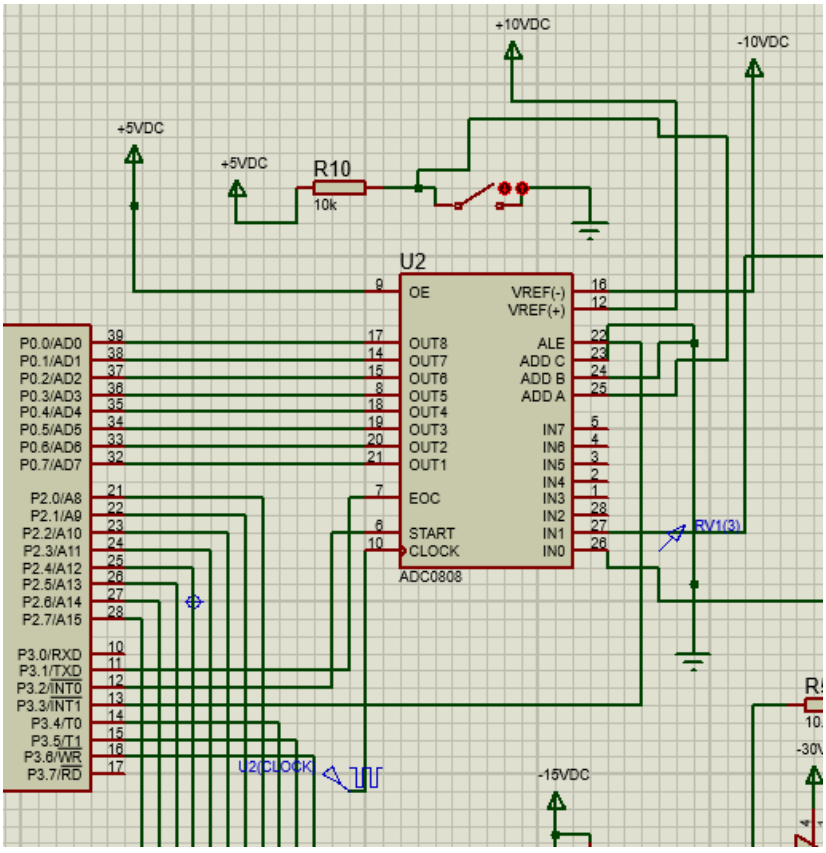


Figure 1: ADC0808

How Does it work?

The ADC Converts Our Analog Signal into a Digital One, It has a built-in Multiplexer Which we use to toggle between our two input signals using the switch at the top, Vref is set at ± 10 Volts, An external Clock is using set at 800KHz, The Start and EOC are controlled by the microcontroller, to read a signal set ALE & SC to one, hold them for 100ns, the input value is recorded then we set the to 0, EOC outputs 1 signaling the start of the conversion, when it turns to 0 it means that the conversion process is complete, we than read the values outputted by the ADC.

DAC

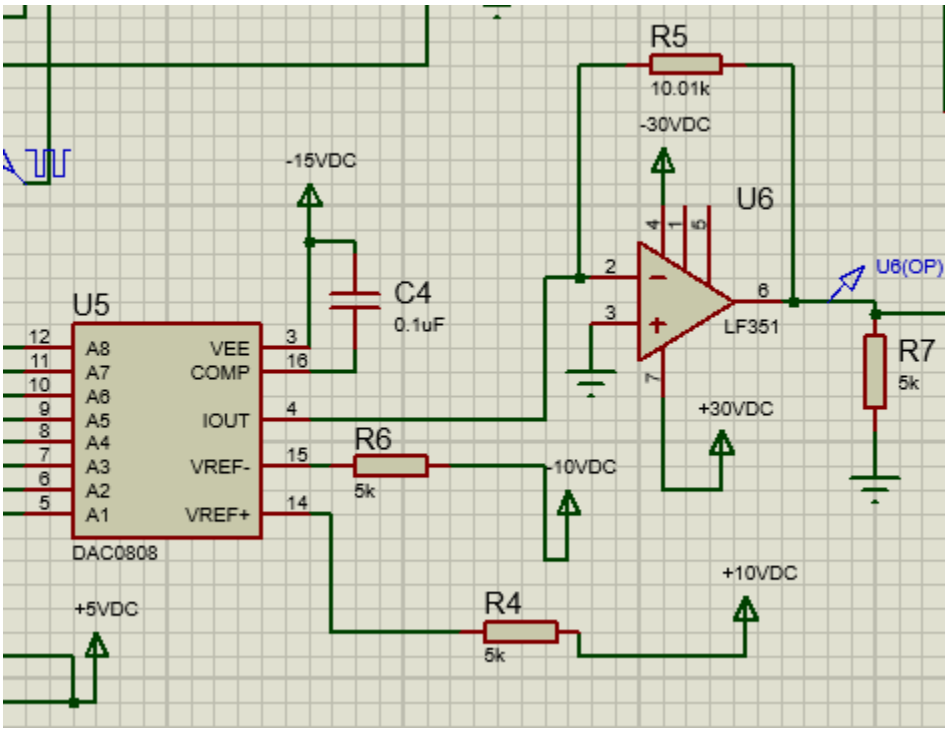


Figure 2: DAC0808

The DAC is connected to the same V_{ref} as the ADC, the output is connected to an Op Amp, The DAC takes the digital values outputted by the Microcontroller and outputs an analog value proportional to it, the Op Amp amplifies this output so that it reaches the same values as the original signal or to a suitable value.

Used Filters

We used Digital **FIR (Finite Impulse Response)** filter and implemented it on the 8051-microcontroller.

A digital filter is a system that performs mathematical algorithm that operates on a digital input signal to improve output signal for the purpose of achieving a filter objective such as: separation of signals that have been combined, or restoration of signals that have been distorted. It refers to the specific hardware and software routine that performs the filtering algorithm. Digital filter mostly operates on digitized analog signals or just numbers, representing some variable, stored in a computer memory. A simplified block diagram of a real-time digital filter

with analog input and output signals, is given below. The band limited analog signal is sampled periodically to generate a discrete time signal, which further quantized to convert into series of digital samples x(n), n = 0,1,… and the digital processor implements the filtering operation, mapping the input sequence x(n) into the output sequence y(n) in accordance with a computational algorithm of the filter. The DAC converts the digitally filtered output into analog values.

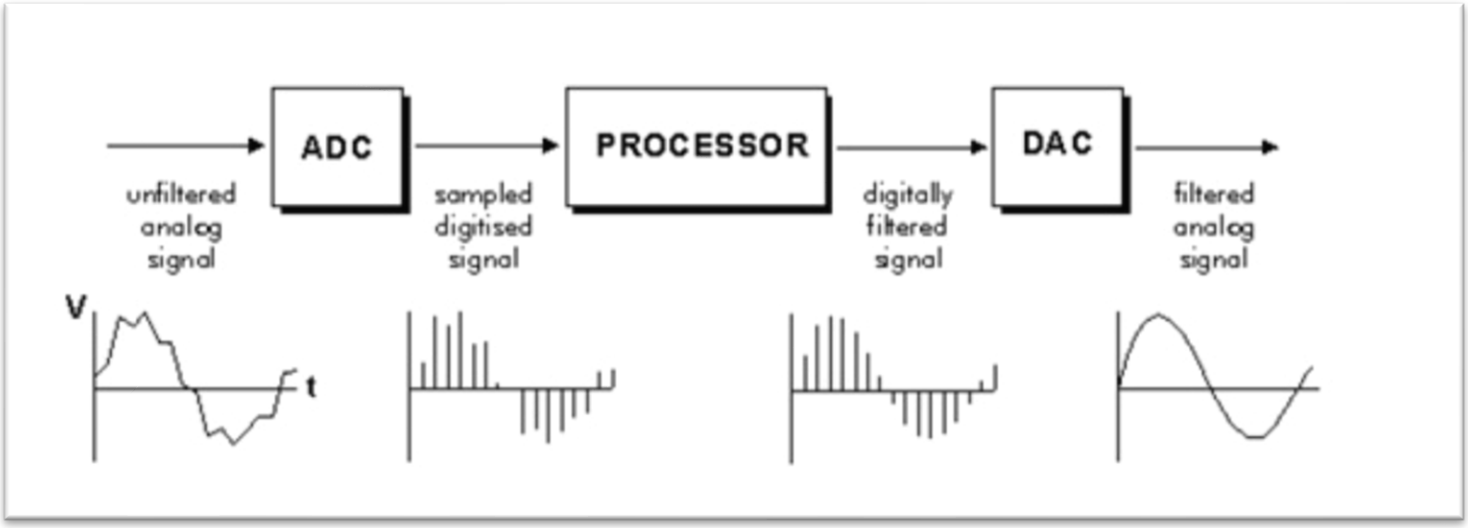


Figure 3: Digital Filter

A Finite Impulse Response (FIR) filter is a type of signal processing filter whose impulse response (or response to any finite length input) is of finite duration, because it settles to zero in finite time. The impulse response of an Nth-order discrete-time FIR filter lasts for N+1 samples, and then dies to zero.

For a discrete-time FIR filter, the output is a weighted sum of the current and a finite number of previous input values. The operation is described by the following equation, which defines the output sequence y[n] in terms of its input sequence x[n]:

$$y[n] = b_0 \cdot x[n] + b \cdot x[n-1] + \ldots\ldots\ldots\ldots + b \cdot x[n-N]$$

$$y[n] = \sum_{i=0}^{n} b_i \cdot x[n-i]$$

- x[n] ---------- input signal
- y[n] ---------- output signal
- bi ---------- filter co-efficients
- N ---------- filter order

Figure 4: FIR Equation

Filter Coefficients using Fdatool by MATLAB

To get the Filter Coefficients to apply it in the equation, we used Fdatool for filter design in MATLAB. This tool provides the ability to create and design the filter you want and you can specify and configure it as you wish. There are options for selecting the cut-off frequency, the sampling frequency, filter type and other settings. After designing the filter, we get its coefficients and save it to use it in the C code to implement the filter function.

Issues about the filters order and limitations of 8051 microcontroller

If we want to design a good filter with a good response, we have to increase its order. But as we know that the number of coefficients equals to N+1 where N is the order, that would make a big problem to us as we now need a huge space to store all the coefficients.

As you see in the figure below when we tried to design a Low-pass filter with the minimum order, the Fdatool specified to order to be **26765** which is impossible to store 26766 coefficients in an array because of the limited RAM and ROM in the 8051-microcontroller.

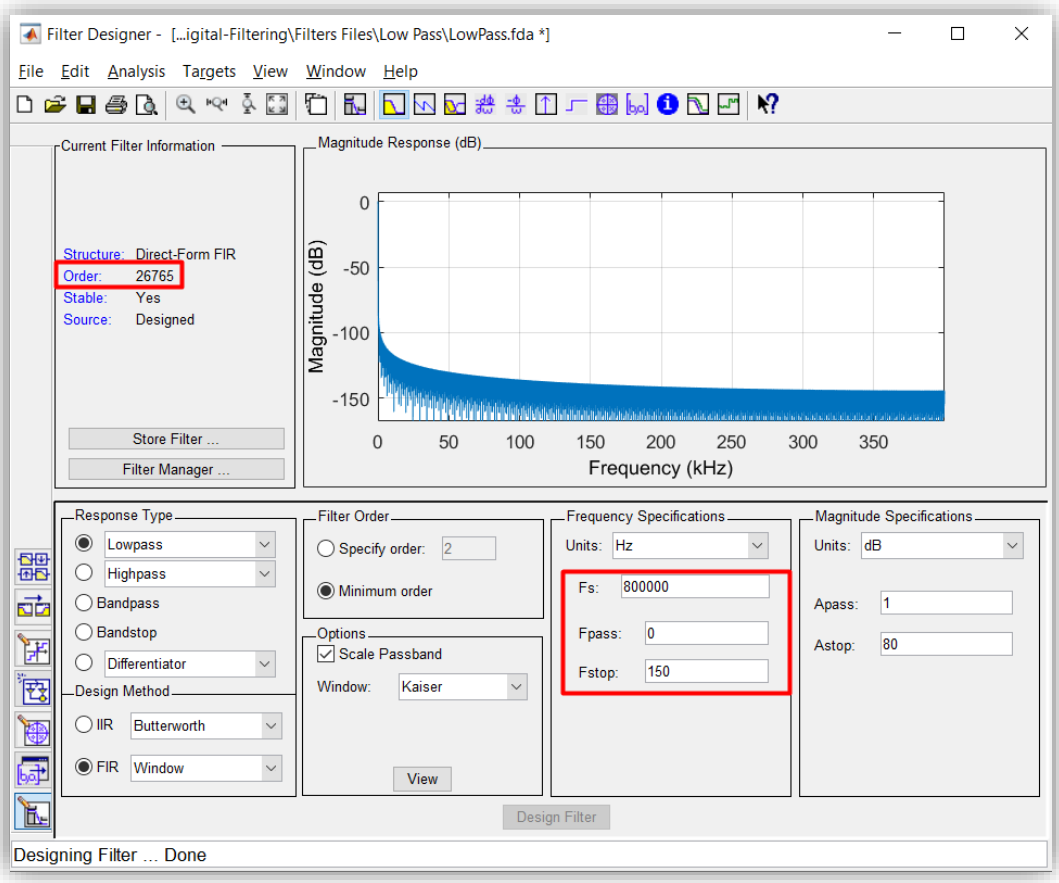


Figure 5: Low-pass filter with minimum order

And this was the coefficients.

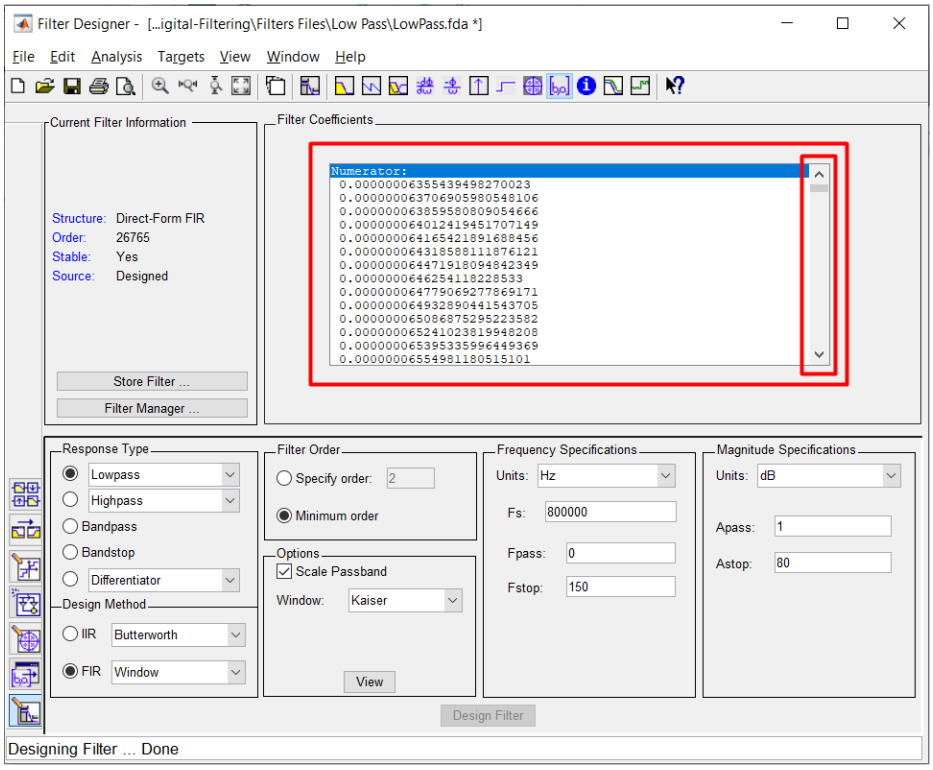


Figure 6: Low-pass filter coefficients - Minimum order

The same when we tried with the High-pass filter. The order was 26946 so it's the same problem.

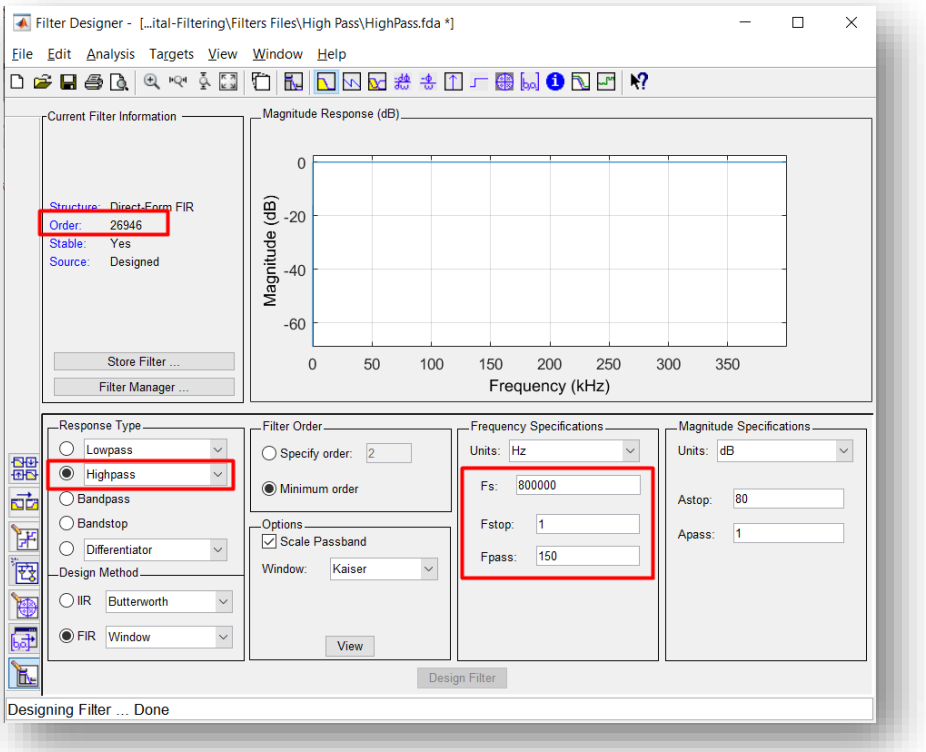


Figure 7: High-pass filter with minimum order

We cannot get a good performance unless the order was so high, but this is impossible to apply because of the limits we have so we needed to low the order to 2 in which the coefficients would be 3 and we could store it and apply the filtering equation.

Results With Filter Order Equals To 2

For the Low-pass filer this was the response when selecting the cut-off frequency to be 150 Hz with 800khz sampling frequency.

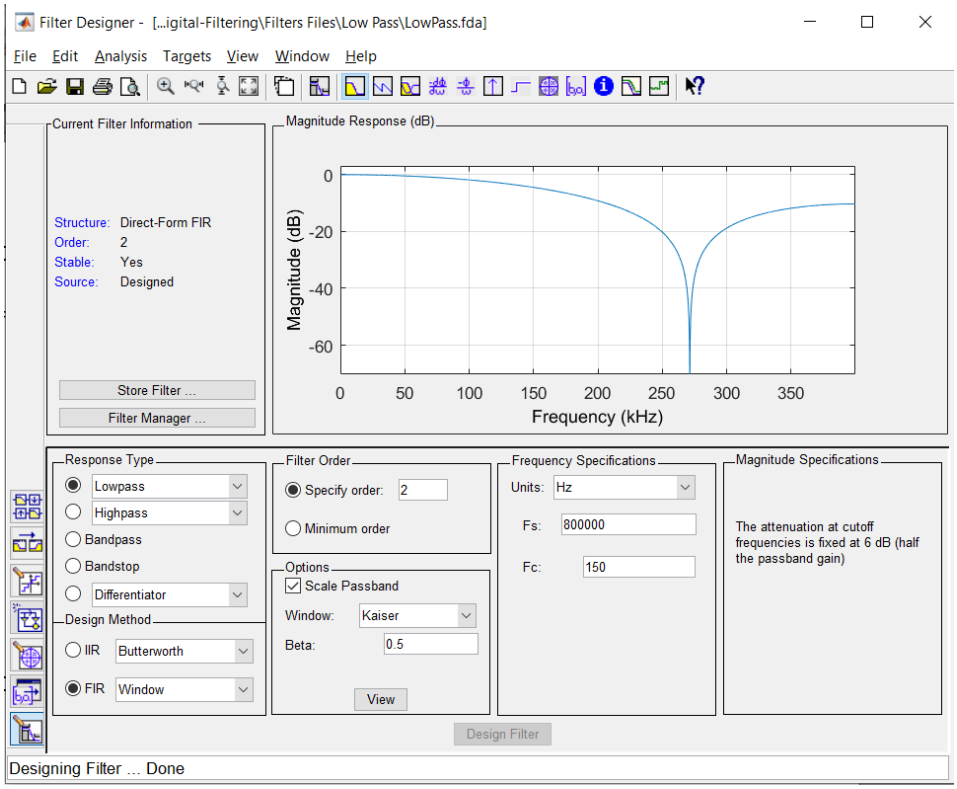


Figure 8: Low-pass Filter Response

And we got these coefficients which is good to apply in the code and use it.

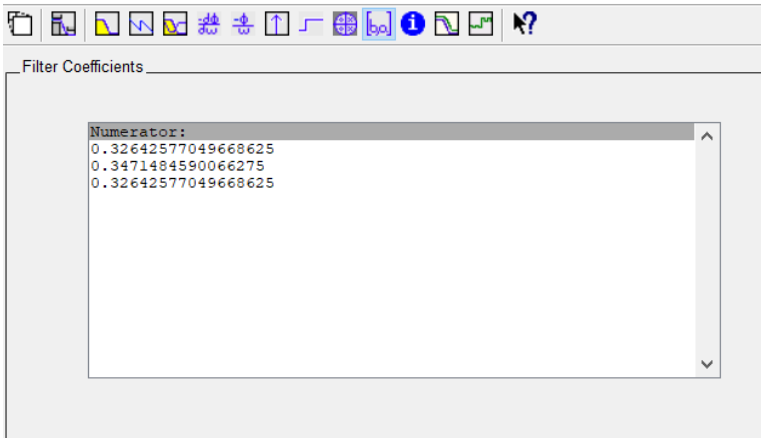


Figure 9: Low-pass Filter Coefficients

We specified the order of the High-pass filter also to be 2 and this was its response.

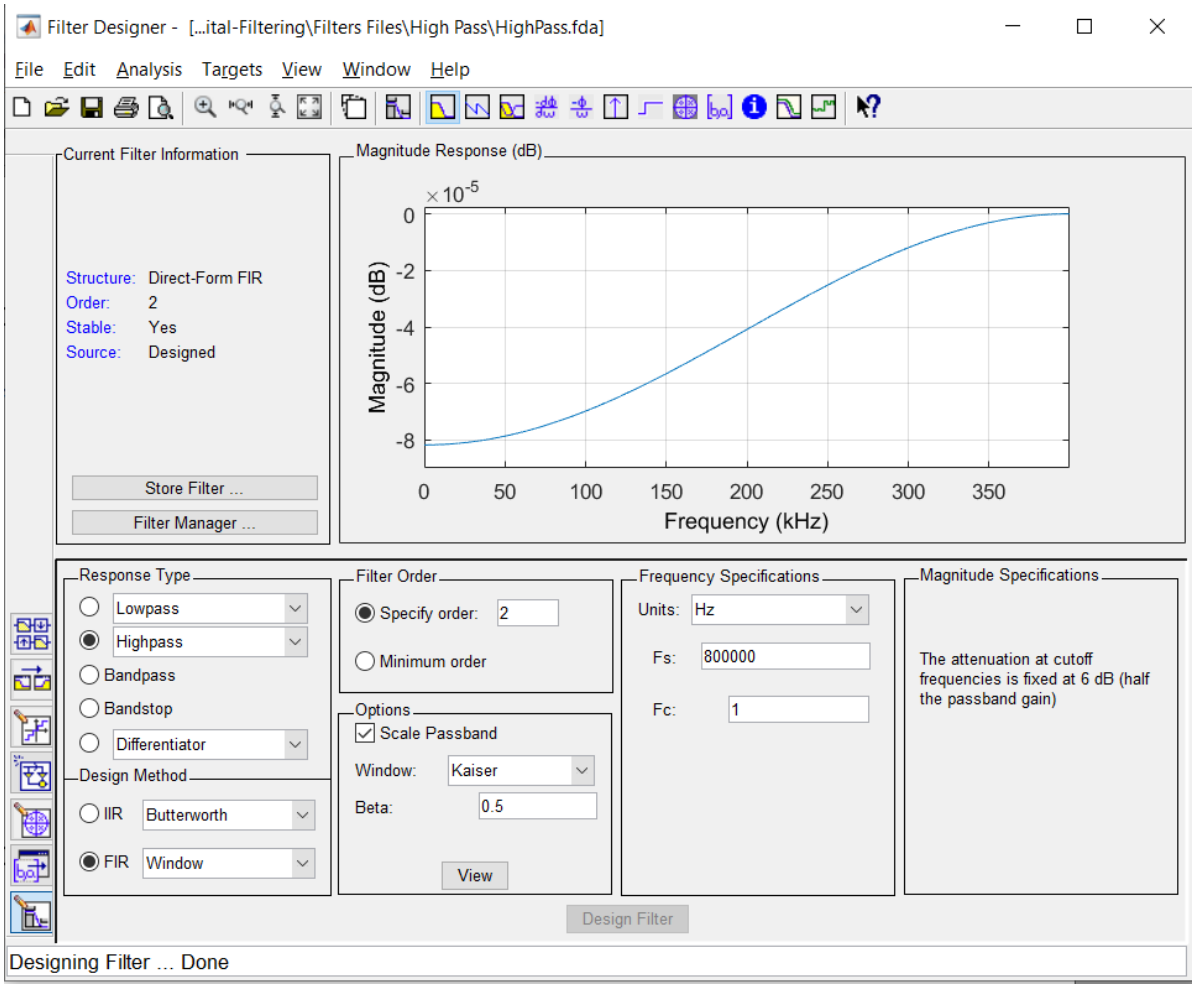


Figure 10: High-pass Filter Response

And we got these coefficients.

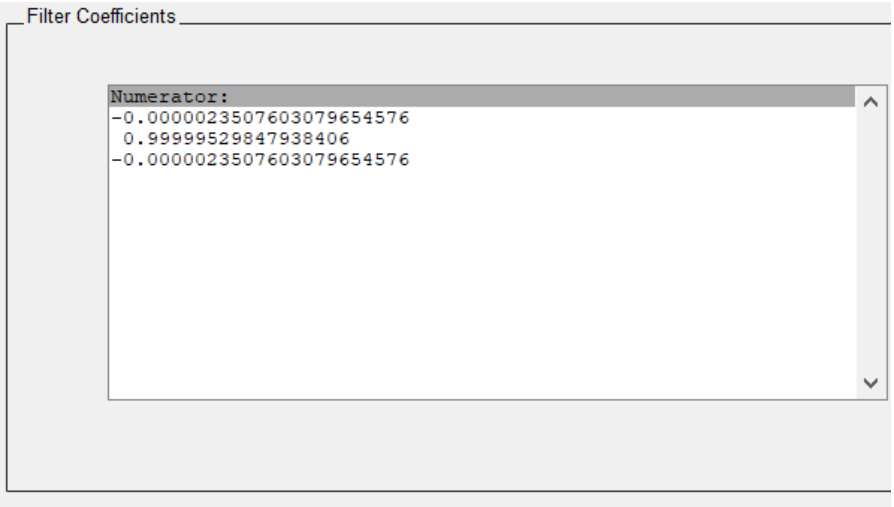


Figure 11: High-pass filter coefficients

The same with the Notch filter, this is its response with order 2

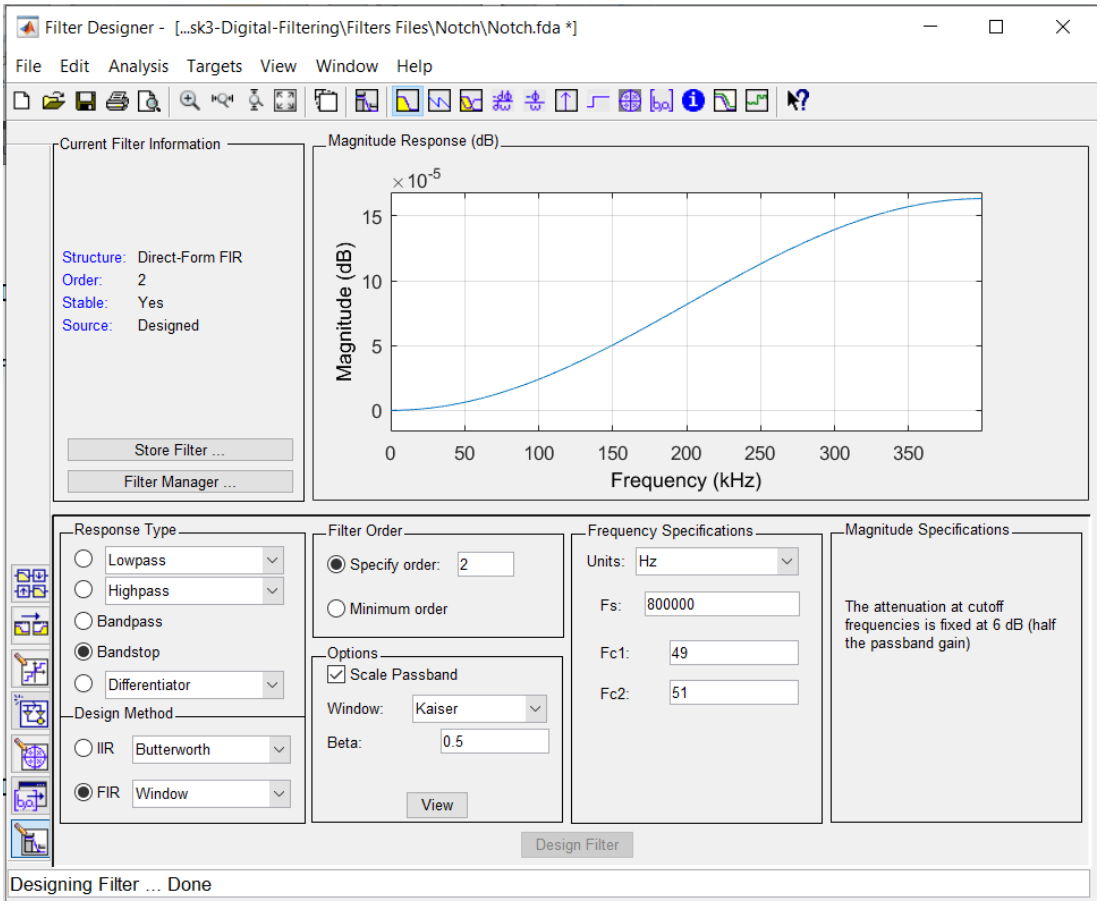


Figure 12: Notch Filter Response

And these was the coefficients

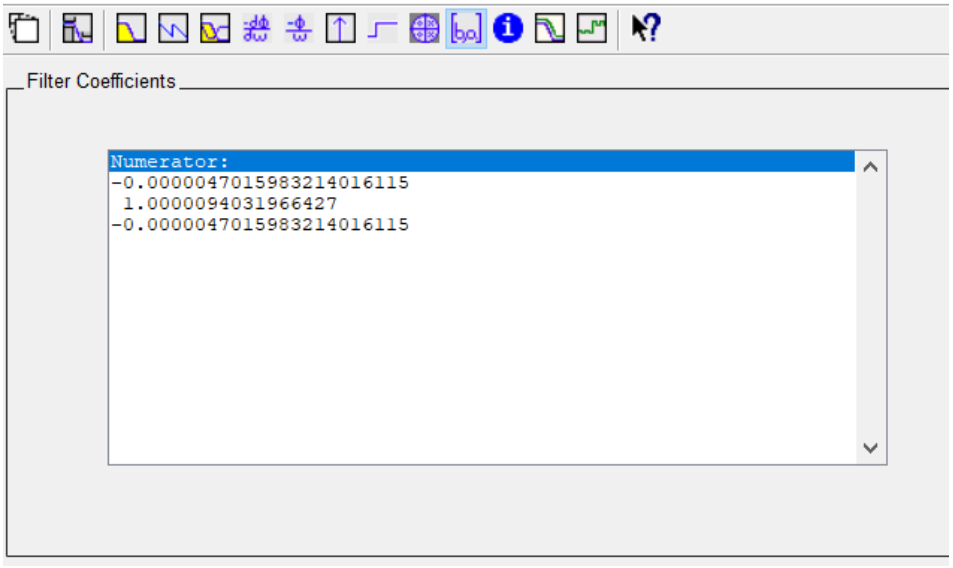


Figure 13: Notch Filter Coefficients

Issues About ADC and DAC

At first while using the converting the signal to analog, giving it to the Microcontroller then outputting the same signal again, we didn't receive the same signal, it was clipped.

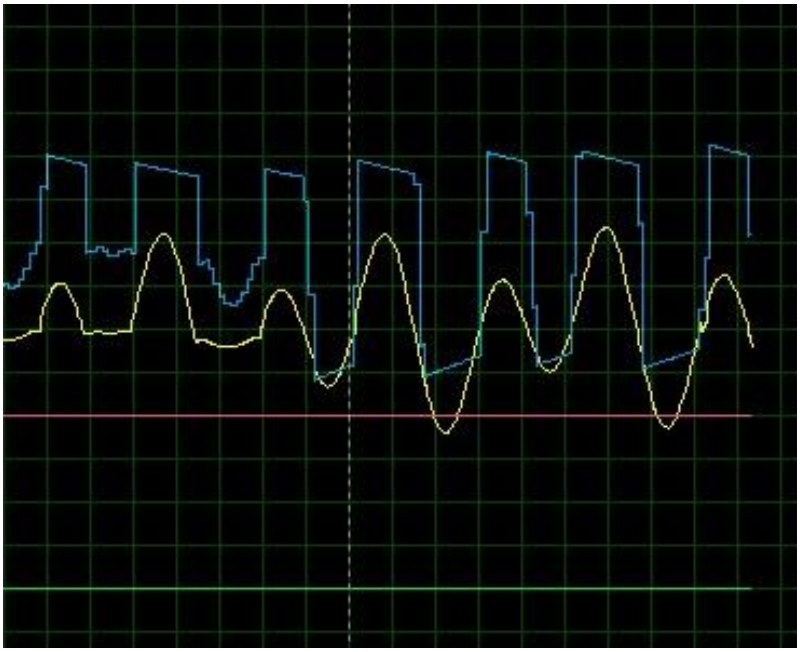


Figure 14 Clipped Signal, Blue is the DAC Output, Yellow is the input

Figure 15 Another Clipped Signal, Blue is the DAC Output, Yellow is the input

The Solution we did:

After numerous attempts and researching, we discovered that one or more of the circuit's components was reaching its saturation levels, after tweaking the gains of the Op Amps and increasing the Reference Voltage of the ADC & DAC, the signal was outputted as it should.

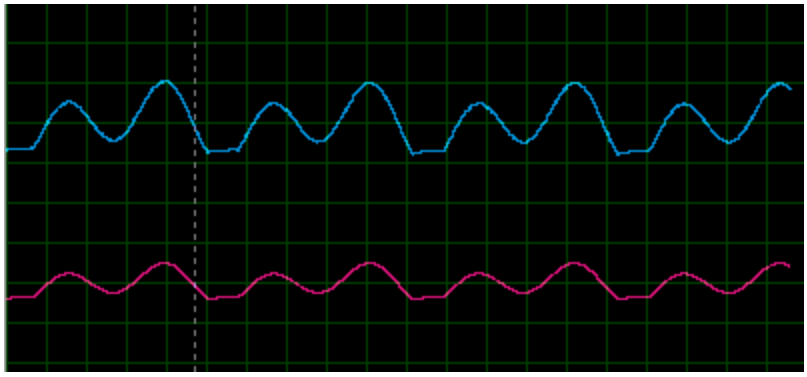


Figure 16: New DAC Output Signal (Blue), No Clipping is observed and the signal is as it should be

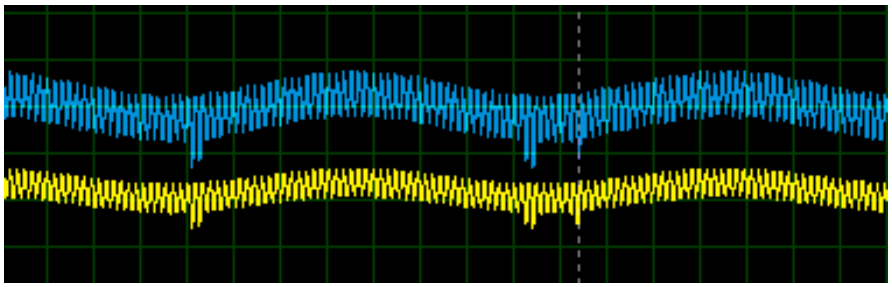


Figure 17: Another Signal (Yellow) and its DAC Output (Blue), No Problems Too

Final Results

After applying the filtering function for the 3 different types, here is the output we get. It's not the good output we want of course because of the limitations and the low sampling frequency we applied, and also the order of the filter is too low, so it's impossible to get any good result.

Low Pass Filter Result

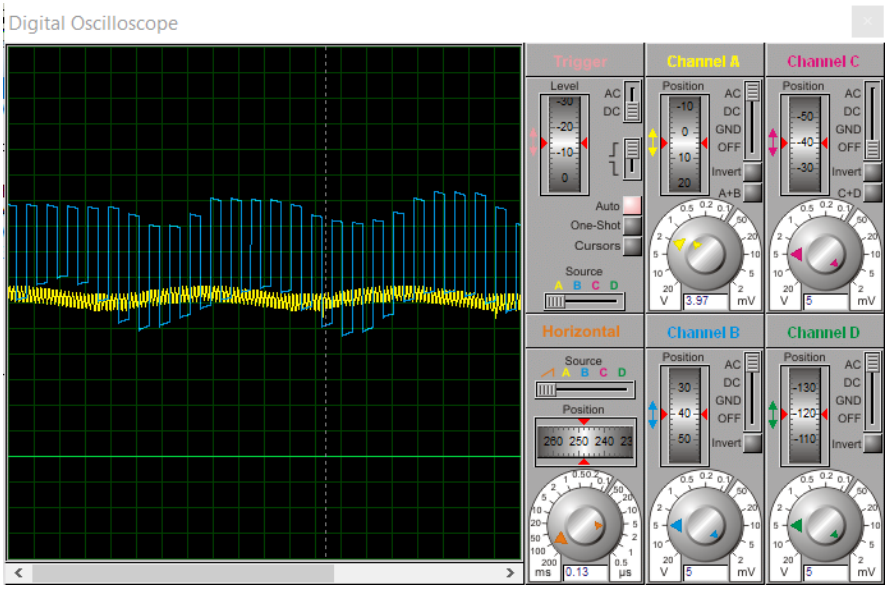


Figure 18: Low-pass filter result

High Pass Filter Result

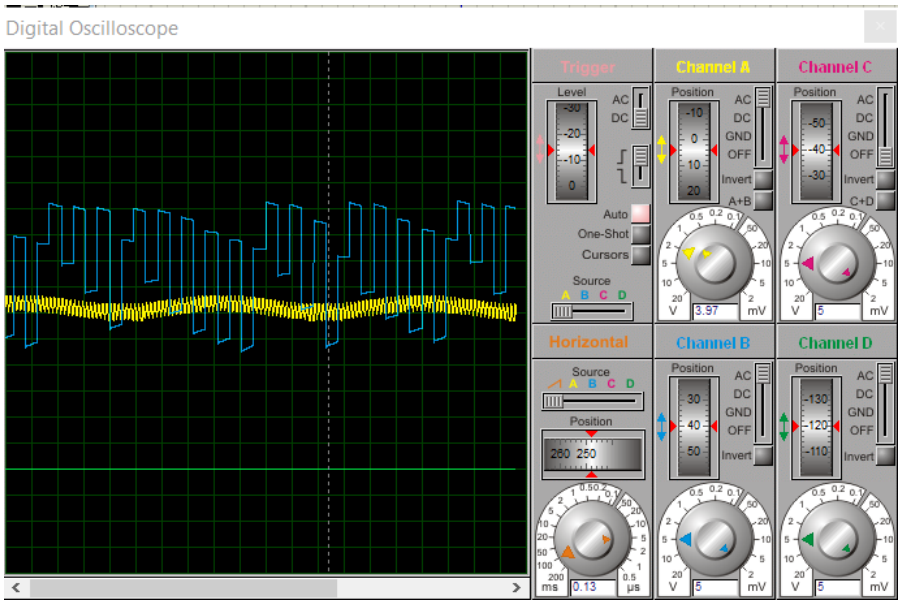


Figure 19: High-pass filter result

Notch Filter Result

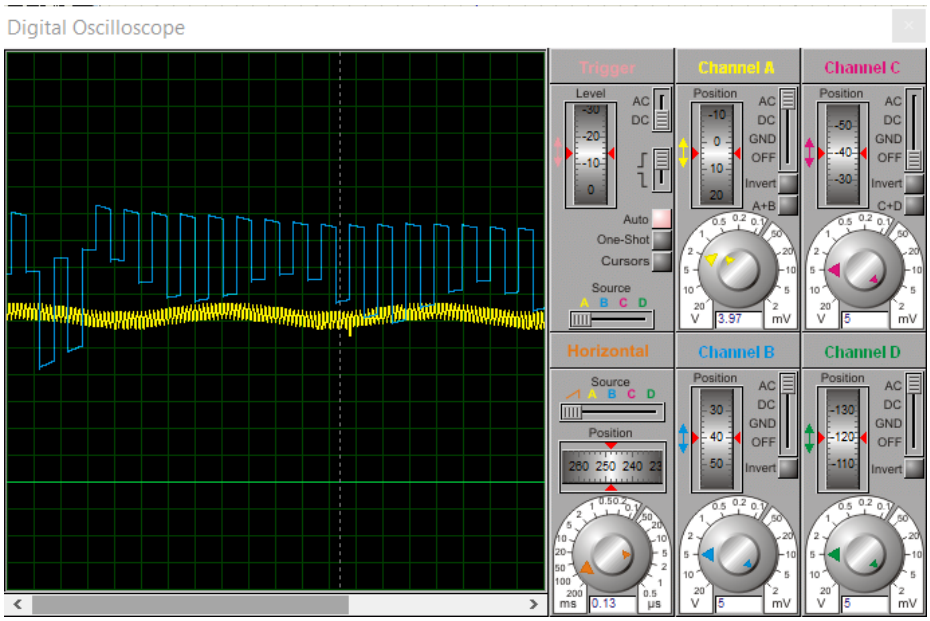
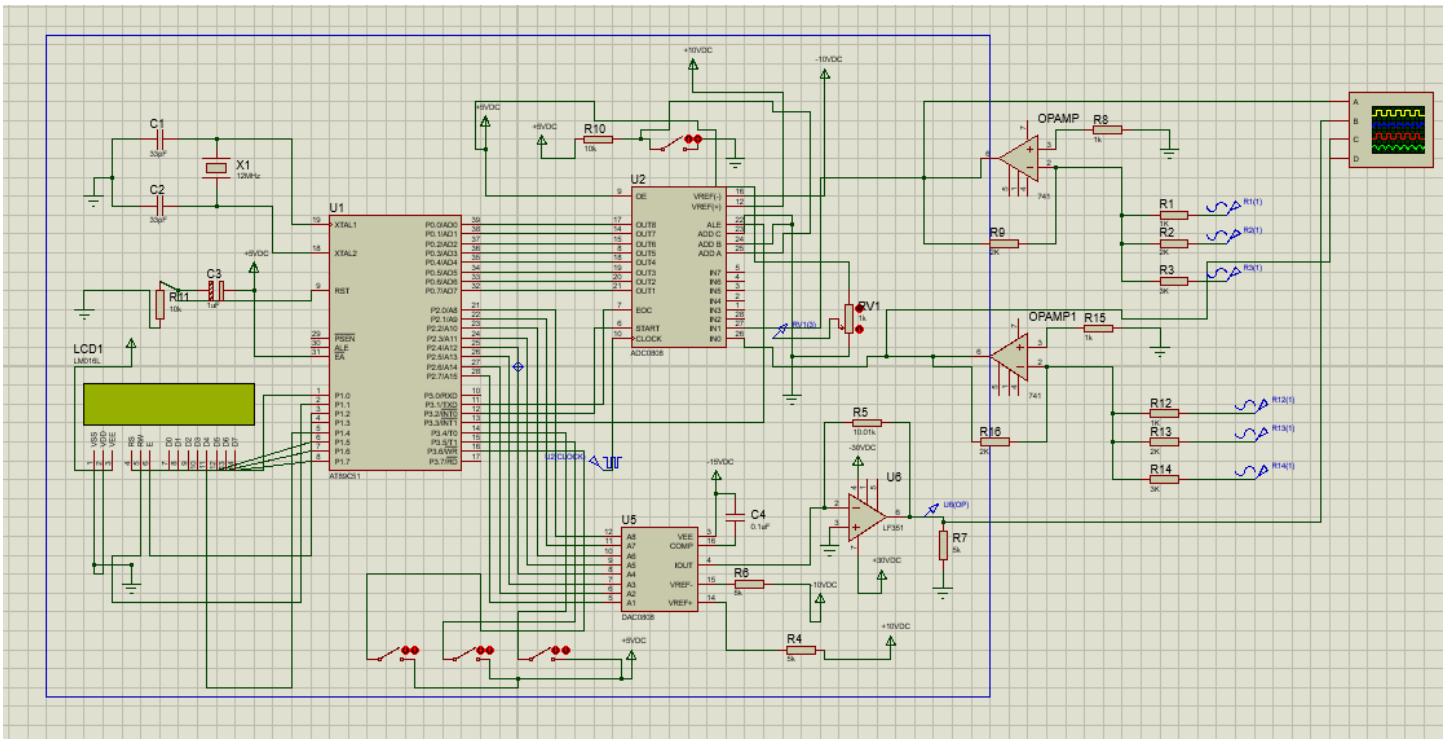


Figure 20: Notch filter result

Full Schematic Diagram



GitHub Repository

https://github.com/r-MC8051-Tasks/Task3-Digital-Filtering