Palestine polytechnic university

Electrical and Computer Engineering Department

Communication Engineering Program

Graduation Project

DSP Experiments using DSK 6713 Kit

Project Team

Ghada Amareen Nagham Tamimi

Shadwana Saree Walaa Al-Nazer

Project Supervisor

Eng: Ahmad Qudaimat

Hebron – Palestine

June , 2012

بسم الله الرحمن الرحيم

شهادة تقييم مشروع التخرج

جامعة بوليتكنك فلسطين

الخليل - فلسطين

مشروع تخرج بعنوان

DSP Experiments using DSK 6713 Kit

عمل الطالبات:

شدوانه محمد صريع

غادة محمد عمارين

نغم سلطان تميمي

ولاء الناظر

بناء على توجيهات الأستاذ المشرف على المشروع ، وبموافقة جميع أعضاء اللجنة الممتحنة تم تقديم هذا المشروع إلى دائرة الهندسة الكهربائية والحاسوب في كليه الهندسة والتكنولوجيا ،للوفاء الجزئي بمتطلبات الدائرة لدرجة البكالوريوس .

توقيع مشرف المشروع	توقيع رئيس الدائرة
[illegible] 20/6/2012	....
م.أحمد قديمات	د.رمزي القواسمي

توقيع اللجنة الممتحنة

....

أيار_٢٠١٢

Abstract

In the 21st century, digital signal processing (DSP) is at the core of most technologies which either directly or indirectly rely on the representation and processing of digital data. All engineering disciplines in this information age need to have a basic understanding of how data from real-world experiments and systems are acquired and processed to extract useful information.

The aim of these series of DSP MATLAB laboratory experiments is to give students a experience with DSP algorithm implementation and applications. MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. After these lab sessions, the students are expected to have a in depth knowledge about how to use MATLB to execute DSP algorithms.

Table of Contents

Chapter Four:

List of appreviations

(ASK)	Amplitude Shift Keying
(A/D)	Analog to Digital
(BPSK)	Binary Phase Shift Keying
(CCS)	Code Composer Studio
(COFF)	Common Object File Format
(CPLD)	Complex Programmable Logic Device
(DSP)	Digital Signal Processing
(DSPs)	Digital Signal Processors
(D/A)	Digital to Analog
(DFT)	Discrete Fourier Transform
(DTFT)	Discrete Time Fourier Transform
(DRAM)	Dynamic Random-Access Memory
(EMIF)	External Memory Inter Face
(FFT)	Fast Fourier transforms
(FPGAs)	Field Programmable Gate Arrays
(FIR)	Finite Impulse Response
(FSK)	Frequency Shift Keying
(HDTV)	High Definition television

(HPI)	Host Port Interface
(IIR)	Infinite Impulse Response
(IDE)	Integrated Development Environment
(IDFT)	Inverse Discrete Fourier Transforms
(JTAG)	Joint Team Action Group
(LED)	light Emitting Diode
(LTI)	linear, Time-Invariant
(MISC)	Miscellaneous
(PLL)	Phase lock loop
(PSK)	Phase Shift Keying
(QPSK)	Quadrature Phase Shift Keying
(ROM)	Read Only Memory
(RTDX)	Real Time Data Exchange
(SFM)	Surface Mount
(SDRAM)	Synchronous DRAM
(TI)	Texas Instruments
(USB)	Universal Serial Bus
(VLIW)	Very long Instruction Word
(VCO)	Voltage Controlled Oscillator

List of figures

List of tables

CHAPTER1

Introduction

1.1 Overview.

DSP Lab is an integrated solution for establishing DSP based Embedded System Lab, based on TI 6000 platform. Basically the lab is designed to learn about the signal processing in Digital Domain. DSP Lab is equipped with complete set of Hardware and Software to perform DSP experiments. The approach enables the understanding of real-time DSP systems principles and real world applications using, C and various assembly programs based on TI's TMS320C6713 Processor. The processing power of the integrated 'C6713 (floating point) DSP allow real-time processing of high bandwidth data streams thereby reducing the processing load placed on the host CPU.

Our project uses DSK6713 kit, we want to mention that DSP lab is to enhance the students' knowledge and understanding of DSP concepts. The lab gives the students a means to relate the (somewhat) abstract concepts of DSP theory to the physical world.

Basic Experiments:

- Sine wave Generation.
- Square wave Generation using Fourier series.
- Convolution.
- Phase shift keying –BPSK encoding.
- Sampling of Audio at different sampling Rate.
- Generate ECHO using DSK6713.
- Demonstrate Different types of FIR Filters using DSK6713.
- Demonstrate Different types of IIR Filter using DSK6713.
- Generate a Amplitude Modulated Signal using DSK6713.

1.2 Aim of the project.

The Digital Signal Processing Lab comprises of test and measurement equipment for doing advanced research and development work in digital signal processing field. The Lab is equipped with complete set of Hardware and Software to perform DSP experiments.

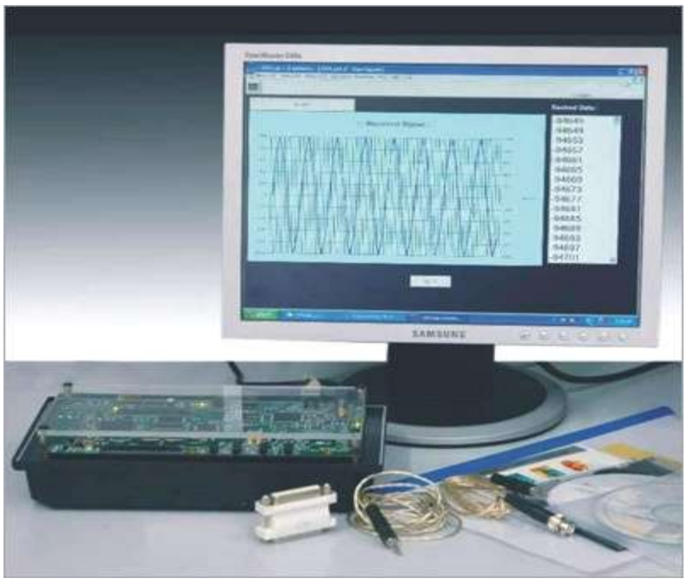


Figure 1.1 system overview

These concepts are dealt with in an ideal setting: infinite precision to represent filter coefficients, no noise, perfect analog-to-digital conversion, perfect sampling, etc. Tools such as Matlab are used to enhance the students understanding of the concepts that are learned in the classroom. However, the students get little or no knowledge of how to implement algorithms in a hardware environment, since most of the algorithms are already implemented in Matlab and the course is designed to develop an understanding of the theoretical concepts. In addition, students do not generally get an in-depth idea of the applications that DSP technology is useful for.

We wanted to develop a course not just to give students the opportunity to implement algorithms on a DSP board, but to give them practical real-world experience with DSP hardware and issues related to it. Consequently, the course was developed with certain goals in mind:

- Enhance/strengthen the knowledge learned in the DSP course.
- Provide the students with an understanding of the wide-range of applications that DSP technology supports.
- Give the students hands-on experience with DSP technology.
- Provide theory pertaining to real-world limitations (quantization, round off errors, overflow).

- Give the students experience with these limitations first-hand to facilitate understanding of the theory.

1.3 Project Risk Management.

During the implementation and executing of the project , many problems and risks may occur , we must solve this in early time of the project in order to operate the project in efficiently and effectively manner .

The project risks are:

- Group meeting difficulties.
- The device kite differently from what are expected.
- The result we obtained are not we expected .

We can come over these risks by :

- Starting working on our project at early time.
- Organizing our meeting.
- Taking many of values to approach to correct result
- Starting to learn about simulation programs at earlier time

1.4 Project Schedule.

1.4.1   Time Schedule

The following table explains the expected timing plan.

Table 1.1 Schedule Table.

Weeks Tasks	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Designing on matlab	▨	▨	▨	▨	▨	▨	▨	▨	▨							
Test the result						▨	▨	▨	▨	▨	▨					
Implementation on the KIT							▨	▨	▨	▨	▨	▨	▨	▨	▨	
Documentation												▨	▨	▨	▨	▨

1.5 Estimated coast:

Table 1.2 Schedule Table

Component	Cost ($)
TMS320C6713 DSP Starter Kit (DSK)	600
Other HW components (IC's, wires ,headphones, and…)	100
Total cost	700

1.6 Outline.

Our documentation of this project is divided into six chapters that describe the hardware and software implementation, the following explains the content of each chapter:

Chapter 1: Introduction

This chapter presents overview , objectives of the project ,project scheduling, risks, and estimated cost.

Chapter 2: Theoretical Background

Theories and materials that are related to our system operating and behavior will be discussed in this chapter.

Chapter 3: hardware and software

This chapter includes information about hardware and software of the project.

Chapter 4: Project design Details and implementation ,results and Conclusion **.**

This chapter discusses the procedure and the outlook design for the experiments theoretically.

1.7 Related work :

There are many research areas related to our work, including digital signal processing ,These topics are described in this section.

"AC 2008-844: matlab/simulink lab exercises designed for teaching digital signal processing applications" ,Kathleen Ossman, University of Cincinnati. Four lab exercises are presented in this lab to illustrate important digital signal processing concepts and applications. The lab exercises are based in MATLAB/Simulink and therefore do not require specific DSP hardware. MATLAB/Simulink is an excellent tool for allowing students to explore the critical concepts of sampling, aliasing, interpolation, convolution, correlation, filter design and realization, word length effects, and Fast Fourier transforms.

The similarity between us, that we take this strategy and improve it by adding the hardware requirements and to be able to downloading it on the c6713 kit .[1]

"Digital Signal Processing and Applications with the C6713 and C6416 DSK" ,Rulph Chasseing, This book is a tutorial on digital techniques for waveform generation, digital filters, and digital signal processing tools and techniques ,The typical chapter begins with some theoretical material followed by working examples and experiments using the TMS320C6713-based DSP Starter Kit (DSK) ,The C6713 DSK is TI's newest signal processor based on the C6x processor (replacing the C6711 DSK).

In this book their work based on translating each theory to Matlab code . instead of using Simulink blocks ,in opposite we use Simulink blocks .[2]

Chapter 2

Theoretical Background

2.1.Introduction

Signals can be divided in to three categories: continuous-time (analog) signals, discrete-time signals, and digital signals. Analog signals are signals that represent physical quantities with a function, which is continuous both in time and amplitude. Discrete-time signals model physical quantities with a function, which is continuous in amplitude but defined only at some specific time instances. On the other hand Digital signals are defined for a particular time instances as discrete-time signals but they take only discrete amplitude values, so they are discrete both in time and amplitude.

Signal processing systems may also be classified along the same line as signals. Digital systems are those for which the input and output of the system are digital signals. Digital Signal Processing (DSP) is therefore concerned with representation of physical quantities using digital signals and the processing of these signals to extract information from them.

Why DSP?

There are lots of reasons that DSP technique is preferred over analog signal processing technique, which uses active and/or passive devices to process a signal .Some of the advantages of a DSP system over the analog counterpart are:

Flexibility: It is easy to change and/or upgrade a DSP system by modifying the software that has been used to implement the specific algorithm on the same hardware. But in the analog counterpart we need to change the whole/part of the circuitry that implements the signal-processing algorithm if we want to modify or upgrade the system. DSP systems, due to these flexibility characteristics, can also be used to simulate signal processing that will be implemented on an analog system before constructing the analog hardware. This enables the analog system designer to have experimental flexibility without wasting engineering or economic resources.

Complexity: Sophisticated applications can be implemented by using power efficient (less complex) DSP algorithms.

Reliability: The DSP hardware components (e.g. memory, logic) are resistant to changing environmental conditions or aging of electronic components than the analog counterparts.

Cost: Due to the advent of technology, which results in reduction of cost of digital computers, it is possible to implement a cost effective DSP systems.

Due to these and other reasons Digital Signal Processors (DSPs) are used for a wide range of applications from communication and control to speech and image processing. Application embedded DSPs are found in cellular phones, fax/modem, disk drives, radio, printers, MP3 players, high-definition television (HDTV), digital cameras, and so on.

Having said some the advantages of DSP systems, it is worthwhile to mention some of their limitations. For example, the bandwidth of the DSP system is dependent on the sampling rate and hardware peripherals, initial costs of the DSP system may be expensive when large bandwidth signals are involved.

2.2. The square wave:

A square wave is a kind of non-sinusoidal waveform, most typically encountered in electronics and signal processing. An ideal square wave alternates regularly and instantaneous - ly between two levels. Its stochastic counterpart is a two-state trajectory. The square wave contains only odd integer harmonics of the sine signal. [3]

Using Fourier series we can write an ideal square wave as an infinite series of the form:

$$X(t)_{square} = \sum_{k=1}^{\infty} \frac{\sin([2k-1]2\pi f)}{(2k-1)}$$

$$= \left(\sin(2\pi ft) + \tfrac{1}{3}\sin(6\pi ft) + \tfrac{1}{5}\sin(10\pi ft) + \cdots\right)$$

2.3. Amplitude modulation (AM)

AM is a technique used in electronic communication, most commonly for transmitting information via a radio carrier wave. AM works by varying the strength of the transmitted signal in relation to the information being sent. For example, changes in signal strength may be used to specify the sounds to be reproduced by a loudspeaker, or the light intensity of television pixels.

A carrier wave is modeled as a sine wave:

$$c(t) = C.sin(\omega ct + \phi c)$$

The constants Cand ϕ_c represent the carrier amplitude and initial phase, and are introduced for generality. For simplicity, their respective values can be set to 1 and 0. Let m(t) represent an arbitrary waveform that is the message to be transmitted, and let the constant M represent its largest magnitude:

$$m(t) = M.cos(\omega_m t + \phi)$$

It is assumed that $\omega_m << \omega_c$ and that $min[m(t)] = -M$

Amplitude modulation is formed by the product:

$$y(t) = [1 + m(t)].c(t)$$
$$= A.[M.cos(\omega m\, t + \phi)].Sin(\omega c\, t)$$

A: represents the carrier amplitude, which is a constant that demonstrates the modulation index.

Therefore, the modulated signal has three components: a carrier wave and two sinusoidal waves (known as sidebands), whose frequencies are slightly above and below ω_c . [4]

The choice A=0 eliminates the carrier component, but leaves the sidebands. To generate a double-sideband full carrier signal, we must choose:

$$A \geq M$$

2.4. Convolution:

The convolution of two discrete-time functions h(n) and x(n)written h(n) , x(n) can be found using the convolution sum:

$$\mathrm{h}[\mathrm{n}] * \mathrm{x}[\mathrm{n}] = \sum_{\mathrm{k}=-\infty}^{\infty} \mathrm{h}[\mathrm{k}]\mathrm{x}[\mathrm{n}-\mathrm{k}] = \sum_{\mathrm{k}=-\infty}^{\infty} \mathrm{h}[\mathrm{n}-\mathrm{k}]\mathrm{x}[\mathrm{k}]$$

Note that convolution is both commutative and distributive. A system is an interconnection of components with terminals or ports where inputs can be applied and outputs extracted. Systems can be represented through relationships between the system variables, usually between the input and output variables of the system. Figure 2.5.1 illustrates a black box approach of a system representation. [5]

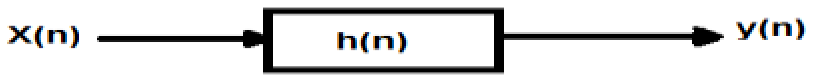


Figure 2.4.1 block system representation

2.5. FIR Filters

FIR filters are relatively easy to design using modern Matlab tools. The characteristics (listed below) of FIR filters as well as the most popular design techniques. The fundamental concept of FIR filter design is that the filter frequency response is determined by the impulse response, and the quantized impulse response and the filter coefficients are identical. This can be understood by examining Figure

2.6.1 The input to the FIR filter is an impulse, and as the impulse propagates through the delay elements, the filter output is identical to the filter coefficients. The FIR filter design process therefore consists of determining the impulse response from the desired frequency response, and then quantizing the impulse response to generate the filter coefficients.

It is useful to digress for a moment and examine the relationship between the time domain and the frequency domain to better understand the principles behind digital filters such as the FIR filter. In a sampled data system, a convolution operation can be carried out by performing a series of multiply-accumulates. The convolution operation in the time or frequency domain is equivalent to point-by-point multiplication in the opposite domain. For example, convolution in the time domain is equivalent to multiplication in the frequency domain. This is shown graphically in Figure 2.5.1 It can be seen that filtering in the frequency domain can be accomplished by multiplying all frequency components in the passband by a 1 and all frequencies in the stopband by 0. Conversely, convolution in the frequency domain is equivalent to point by point multiplication in the time domain.[6]

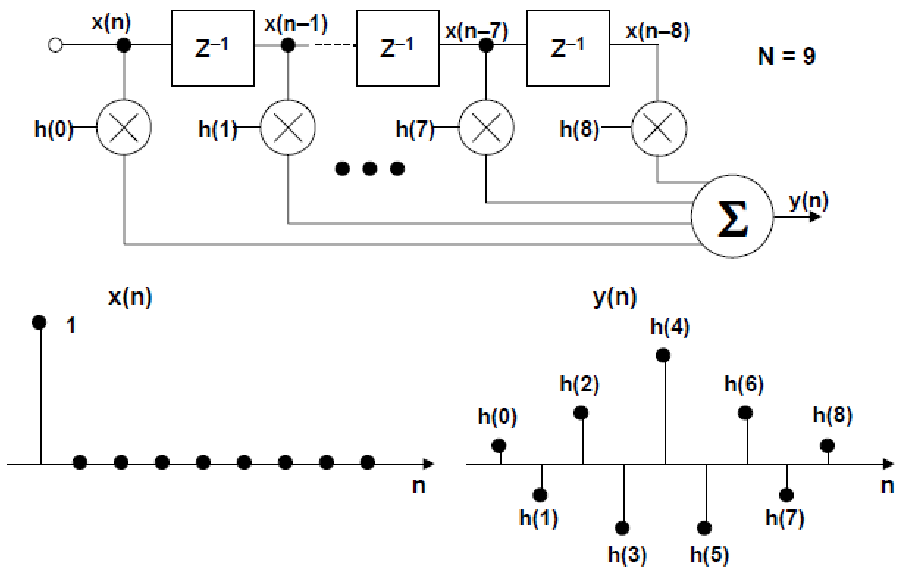


Figure 2.5.1 FIR filter impulse response Determines the filter coefficients

2.5.1. Characteristics of FIR filters:

An FIR filter has a number of useful properties which sometimes make it preferable to an infinite impulse response (IIR) filter. FIR filters:

- Require no feedback. This means that any rounding errors are not compounded by summed iterations. The same relative error occurs in each calculation. This also makes implementation simpler.

- Are inherently stable. This is due to the fact that, because there is no required feedback, all the poles are located at the origin and thus are located within the unit circle (the required condition for stability in a Z transformed system).
- They can easily be designed to be linear phase by making the coefficient sequence symmetric; linear phase, or phase change proportional to frequency, corresponds to equal delay at all frequencies. This property is sometimes desired for phase-sensitive applications, for example data communications, crossover filters, and mastering.

The transfer function in the frequency domain (either a 1 or a 0) can be translated to the time domain by the discrete Fourier transform (in practice, the fast Fourier transform is used). This transformation produces an impulse response in the time domain. Since the multiplication in the frequency domain (signal spectrum times the transfer function) is equivalent to convolution in the time domain (signal convolved with impulse response), the signal can be filtered by convolving it with the impulse response. The FIR filter is exactly this process. Since it is a sampled data system, the signal and the impulse response are quantized in time and amplitude yielding discrete samples. The discrete samples comprising the desired impulse response are the FIR filter coefficients. [7]

2.5.2Window function

The window method is most commonly used method for designing FIR filters. The simplicity of design process makes this method very popular. We essentially put a rectangular window function with an amplitude of 1 between -Q and +Q and ignored the coefficients outside that window. The wider this rectangular window, the larger Q is ,The rectangular window function can therefore be defined as

$$W_R(n) = \begin{cases} 1 & \text{for} |n| \leq Q \\ 0 & \text{otherwise} \end{cases}$$

The transform of the rectangular window function $W_R(n)$ yields a sinc function in

the frequency domain. It can be shown that:

$$W_R(n) = \sum_{N=-Q}^{Q} e^{-jn\pi N} = e^{-jQ\pi n} \sum_{N=0}^{2Q} e^{jn\pi N} = \frac{\sin\,[(\frac{2Q+1}{2})\pi n]}{\sin\,(\frac{\pi n}{2})}$$

which is a sinc function that exhibits high sidelobes or oscillations caused by the abrupt truncation, specifically, near discontinuities.

A number of window functions are currently available to reduce these high amplitude oscillations; they provide a more gradual truncation to the infinite series expansion. However, while these alternative window functions reduce the amplitude of the sidelobes, they also have a wider mainlobe, which results in a filter with lower selectivity. A measure of a filter's performance is a ripple factor that compares the peak of the first sidelobe to the peak of the mainlobe (their ratio).A compromise or trade-off is to select a window function that can reduce the sidelobes while approaching the selectivity that can be achieved with the rectangular window function. The width of the mainlobe can be reduced by increasing the width of the window (order of the filter) .In general, the Fourier series coefficients can be written as:

$$C_n^{'} = C_n \ w(n)$$

where w(n) is the window function. In the case of the rectangular window function, $C_n^{'} = C_n$. The transfer function written as

$$H^{'}(z) = \sum_{i=0}^{N-1} h_i^{'} \ z^{-i}$$

Where

$$h_i^{'} = C_{Q-i}^{'} \quad 0 \le i \le 2Q$$

The rectangular window has its highest sidelobe level, down by only -13dB from the peak of its mainlobe, resulting in oscillations with an amplitude of considerable size. On the other hand, it has the narrowest mainlobe that can provide high selectivity. [8]

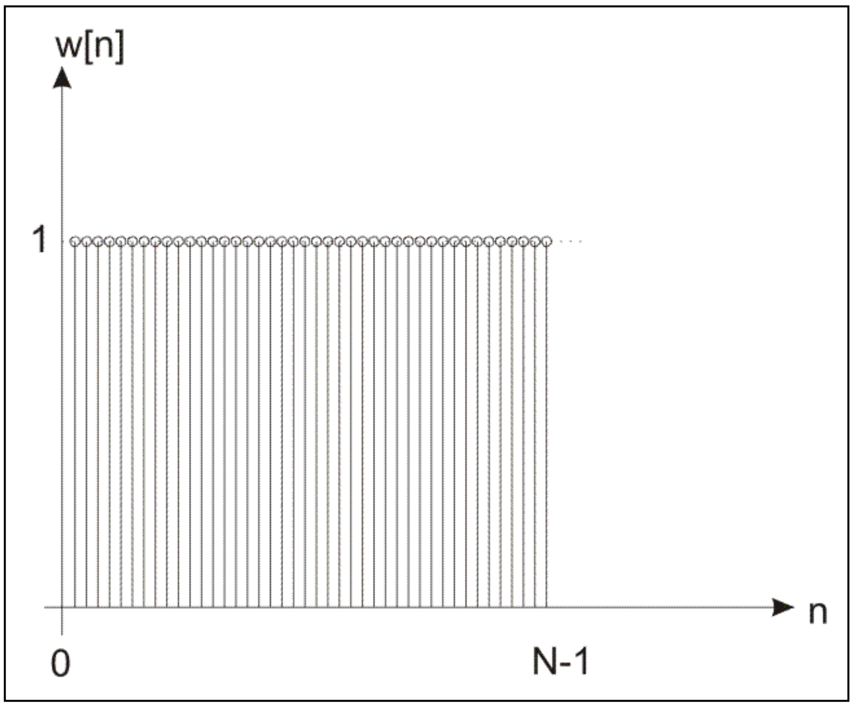


Figure2.5.2 Rectangular window in the time domain

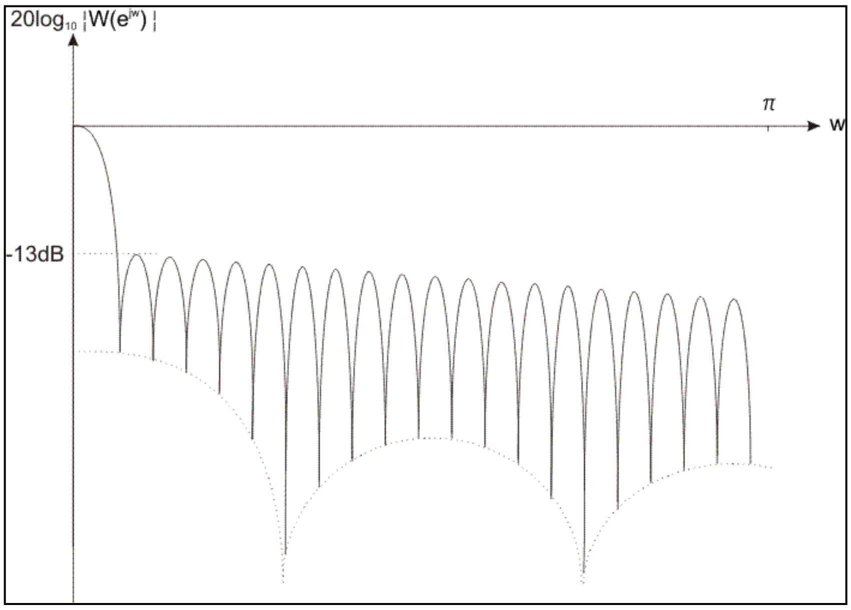


Figure.2.5.3 Rectangular window in the frequency domain (spectrum)

The following window functions are commonly used in the design of FIR

Filters:

2.5.2.2.Hamming Window

The Hamming window function is:

$$W_H(n) = \begin{cases} 0.54 + 0.46\cos\left(\frac{n\pi}{Q}\right) & \text{for} |n| \le Q \\ 0 & \text{otherwise} \end{cases}$$

which has the highest or first sidelobe level at approximately -43 dB from the peak of the main lobe.[9]

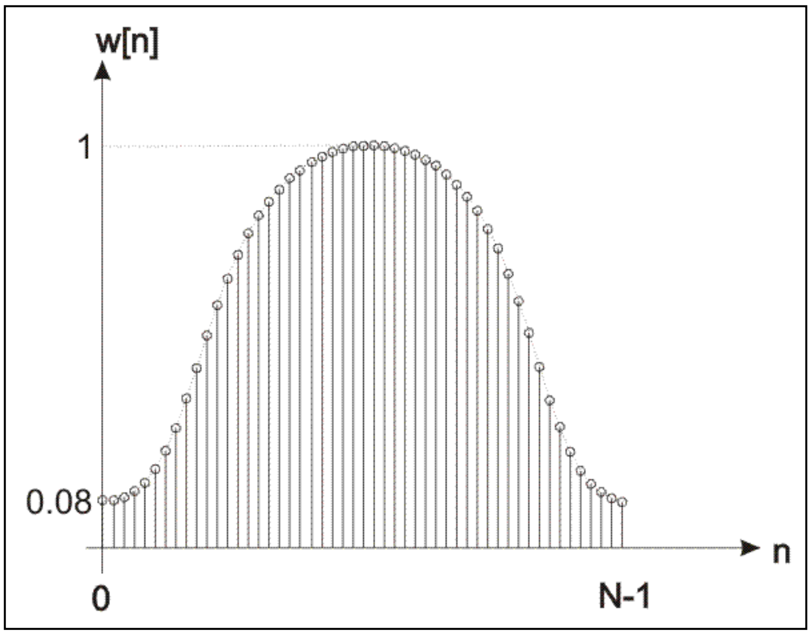


Figure.2.5.4 The Hamming window coefficients in the time domain

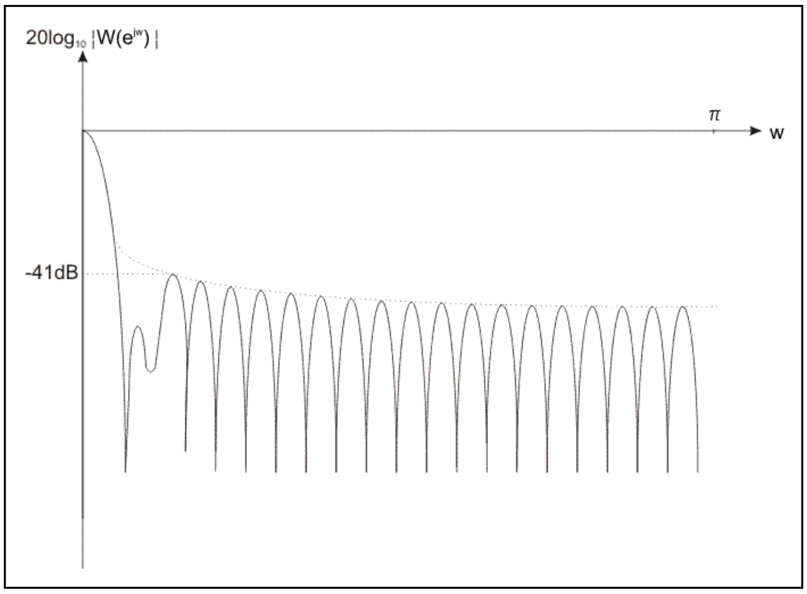


Figure 2.5.6. The Hamming window coefficients in the frequency domain

2.5.2.3 Hanning Window

the Hanning or raised cosine window function is

$$W_{HA}(n) = \begin{cases} 0.5 + 0.5\cos\left({}^{n\pi}/_{Q}\right) & \text{for}|n| \leq Q \\ 0 & \text{otherwise} \end{cases}$$

which has the highest or first sidelobe level at approximately -31 dB from the peak of the mainlobe.

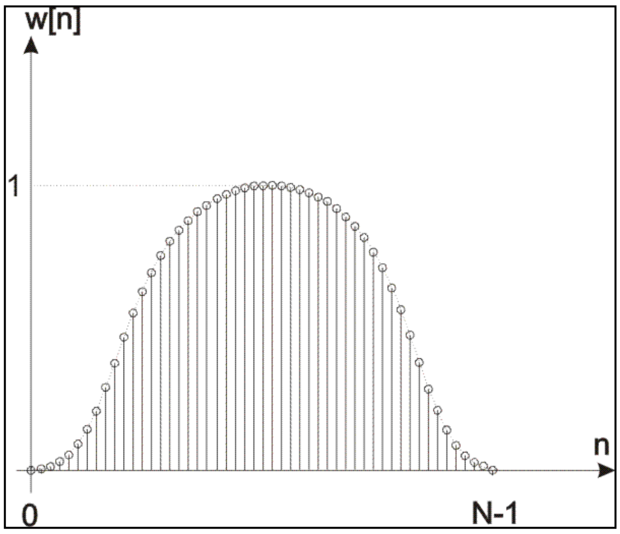


Figure2.5.7 . The Hanning window coefficients in the time domain

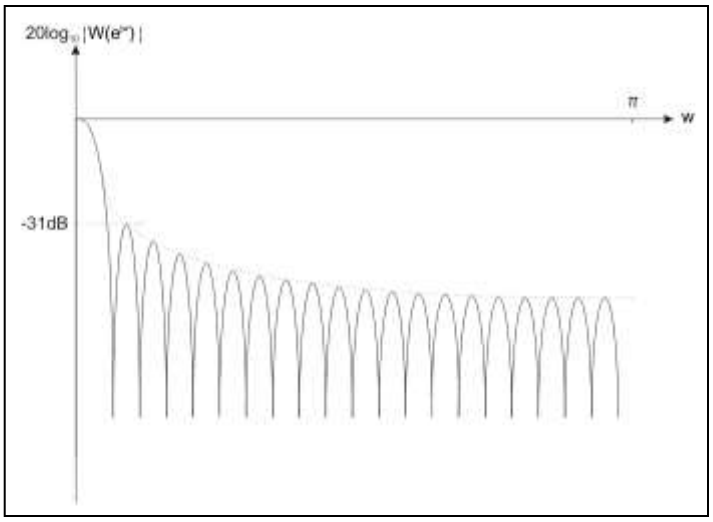


Figure2.5.8. The Hanning window coefficients in the frequency domain

2.6.2.4 Blackman Window

The Blackman window function is

$$W_B(n) = \begin{cases} 0.42 + 0.5\cos\left(n\pi/Q\right) + 0.8\cos\left(2n\pi/Q\right) & \text{for}|n| \le Q \\ 0 & \text{otherwise} \end{cases}$$

which has the highest sidelobe level down to approximately -58 dB from the peak of the mainlobe. While the Blackman window produces the largest reduction in the sidelobe compared with the previous window functions, it has the widest mainlobe.

As with the previous windows, the width of the mainlobe can be decreased by increasing the width of the window. .[8]

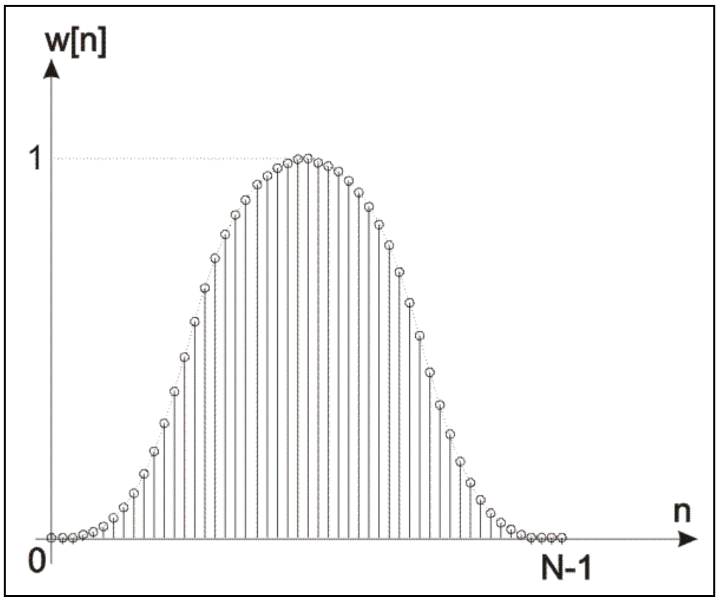


Figure 2.5.9 The Blackman window coefficients in the time domain

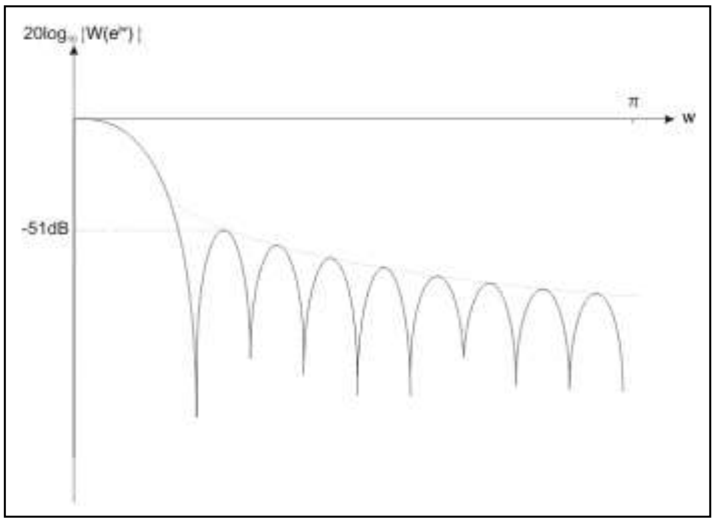


Figure2.5.10: The Blackman window coefficients in the frequency domain

2.5.2.5 Kaiser Window

The design of FIR filters with the Kaiser window has become very popular in recent years. It has a variable parameter to control the size of the sidelobe with respect to the mainlobe. The Kaiser window function is

$$W_K(n) = \begin{cases} I_0\ (b)/I_0\ (a) & \text{for}|n| \le Q \\ 0 & \text{otherwise} \end{cases}$$

Where a is an empirically determined variable, and b = a[1 - (n/Q)2]1/2. $I_0(x)$ is the modified Bessel function of the first kind defined by

$$I_0\ (x) = 1 + \frac{0.25x^2}{(2!)^2} + \frac{(0.25x^2)^2}{(2!)^2} + \cdots . = 1 + \sum_{n=1}^{\infty} \left[\frac{(x/2)^n}{n!}\right]^2$$

Which converges rapidly. A trade-off between the size of the sidelobe and the width of the mainlobe can be achieved by changing the length of the window & the parameter a. [9]

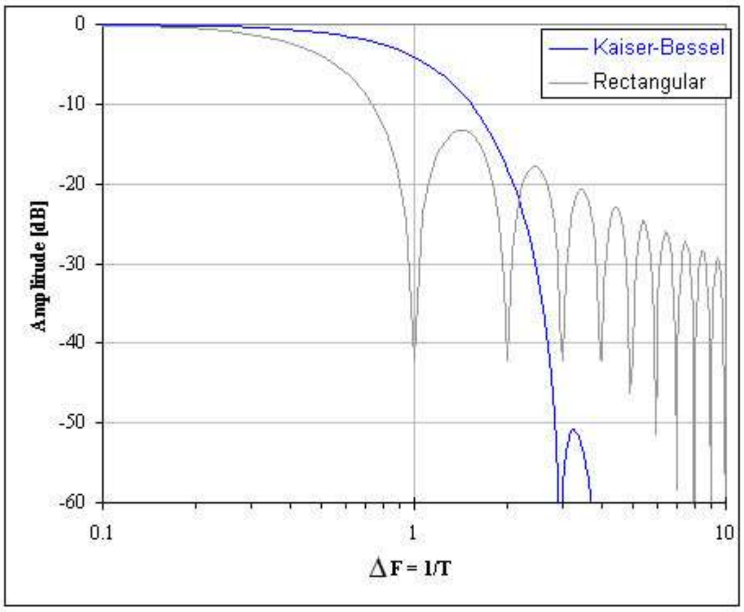


Figure 2.5.11 the Kaiser Bessel in frequency domain

2.6. (IIR) filters

As was mentioned previously, FIR filters have no real analog counterparts, the closest analogy being the weighted moving average. In addition, FIR filters have only zeros and no poles. On the other hand, IIR filters have traditional analog counterparts (Butterworth, Chebyshev, Elliptic, and Bessel) and can be analyzed and synthesized using more familiar traditional filter design techniques.[9]

Infinite impulse response filters get their name because their impulse response extends for an infinite period of time. This is because they are recursive, i.e., they utilize feedback. Although they can be implemented with fewer computations than FIR filters, IIR filters do not match the performance achievable with FIR filters, and do not have linear phase. Also, there is no computational advantage achieved when the output of an IIR filter is decimated because each output value must always be calculated.

2.6.1 Characteristics of (IIR) FILTERS

- Uses Feedback (Recursion).
- Impulse Response has an Infinite Duration.
- Potentially Unstable.
- Non-Linear Phase.
- More Efficient than FIR Filters.

2.6.2 IIR filter design techniques

A popular method for IIR filter design is to first design the analog-equivalent filter and then mathematically transform the transfer function H(s) into the z-domain, H(z). Multiple pole designs are implemented using cascaded biquad sections. The most popular analog filters are the Butterworth, Chebyshev, Elliptical, and Bessel.

The all-pole Butterworth (also called maximally flat) has no ripple in the passband or stopband and has monotonic response in both regions. The all-pole Type 1 Chebyshev filter has a faster rolloff than the Butterworth (for the same number of poles) and has ripple in the passband. The Type 2 Chebyschev filter is rarely used, but has ripple in the stopband rather than the passband. The Elliptical (Cauer) filter has poles and zeros and ripple in both the passband and stopband. This filter has even faster rolloff than the Chebyshev for the same number of poles. The Elliptical filter is often used where degraded phase response can be tolerated.

Finally, the Bessel (Thompson) filter is an all-pole filter which is optimized for pulse response and linear phase but has the poorest rolloff of any of the types discussed for the same number of poles.[6]

2.6.3 IIR Filter Types

2.6.3.1 Butterworth filter

Provides the best Taylor series approximation to the ideal lowpass filter response at analog frequencies Ω = 0 and Ω = 1; for any order n, the magnitude squared response has 2n - 1 zero derivatives (that is, it is maximally at) at these locations. Response is monotonic overall, decreasing smoothly from Ω= 0 to Ω = 1.

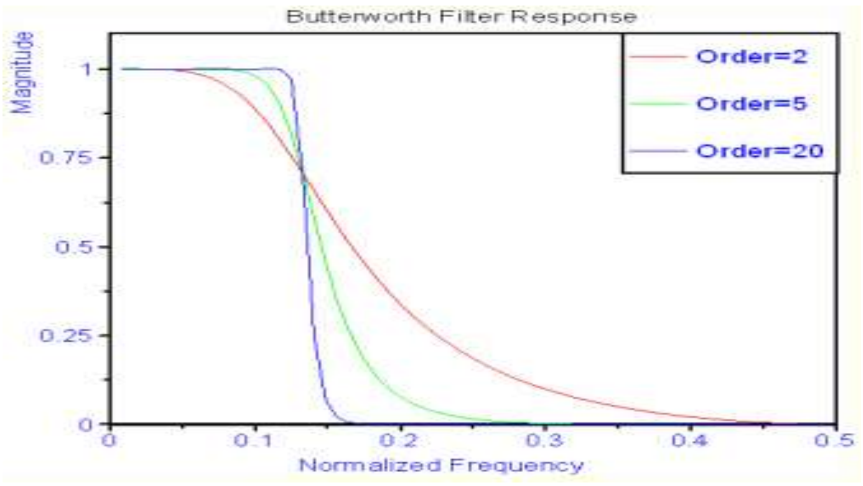


Figure 2.6.1 :Butterworth filter response**.**

2.6.3.2 Chebyshev Type I filter

Minimizes the absolute difference between the ideal and the actual frequency response over the entire passband by using an equal ripple in the passband . Stopband response is maximally at. The transition from passband to stopband is more rapid than for the Butterworth filter. [7]

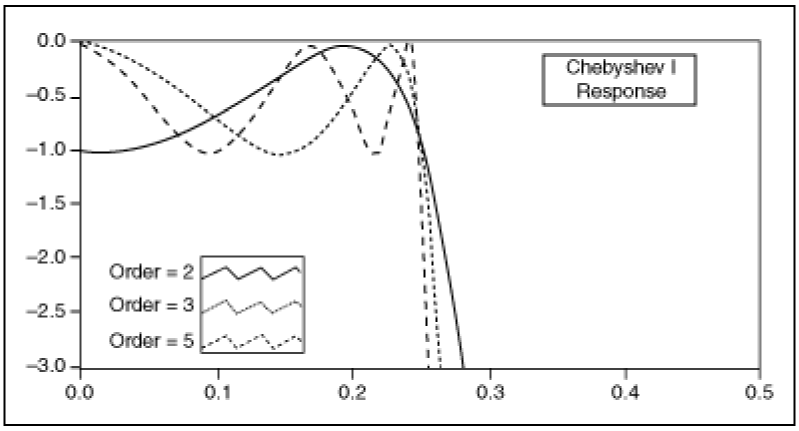


Figure 2.6.2 Chebyshev filter response

2.6.3.3 Chebyshev Type II filter

Minimizes the absolute difference between the ideal and the actual frequency response over the entire stopband by using an equal ripple in the stopband . Passband response is maximally at. The stopband does not approach zero as quickly as the type I filter (and does not approach zero at all for even-valued filter order n). The absence of ripple in the passband, however, is often an important advantage. [8]

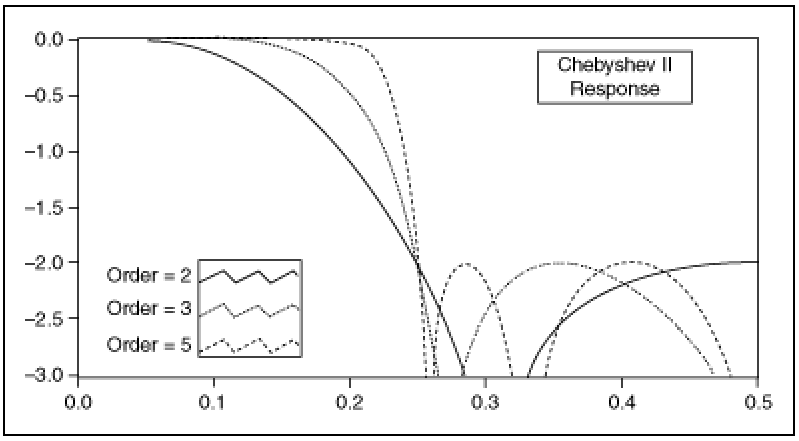


Figure 2.6.3 Chebyshev II filter response

2.6.3.3 Elliptic filter

Equiripple in both the passband and stopband. Generally meets filter requirements with the lowest order of any supported filter type. Given a filter order n, passband ripple, and stopband ripple, elliptic filters minimize transition width.

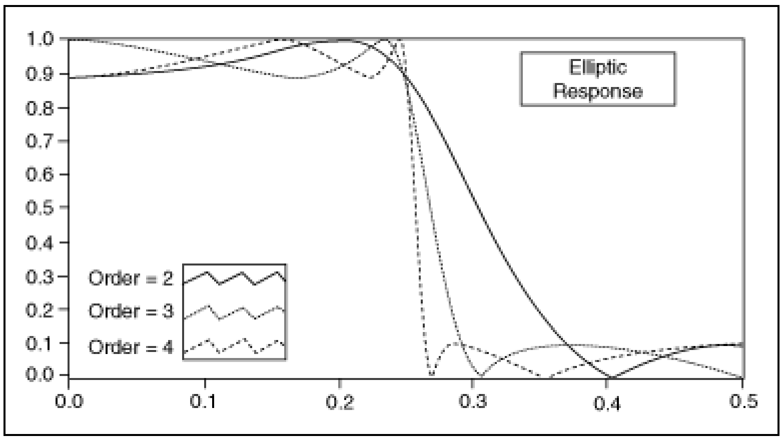


Figure 2.6.4 elliptic filter response

2.6.3.5 Bessel filter

Analog lowpass fillters have maximally at group delay at zero frequency and retain nearly constant group delay across the entire passband. Filtered signals therefore maintain their waveform in the passband. Digital Bessel filters, however, do not have this maximally at property, and are not supported by Matlab. Generally require a higher filter order than other filters for satisfactory stopband attenuation. [8]

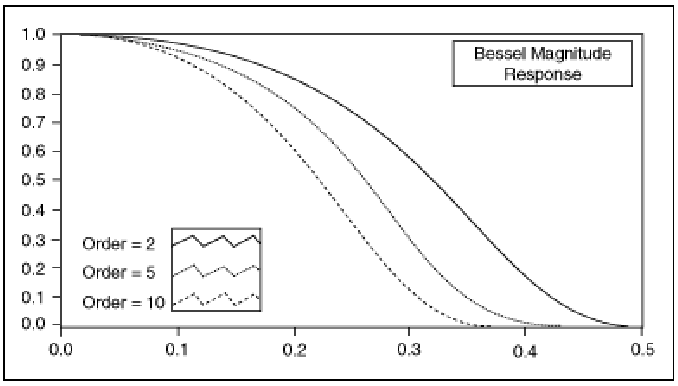


Figure 2.6.5 Bessel filter magnitude response

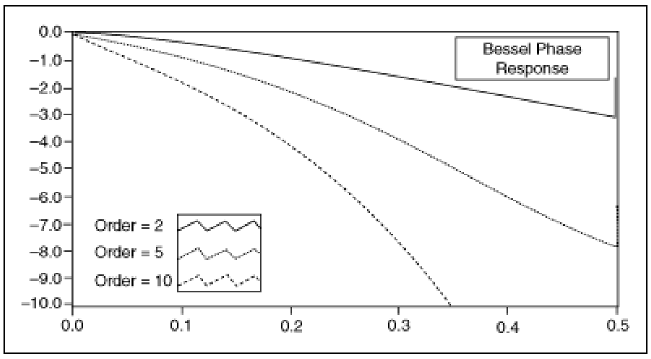


Figure 2.6.6 Bessel filter phase response

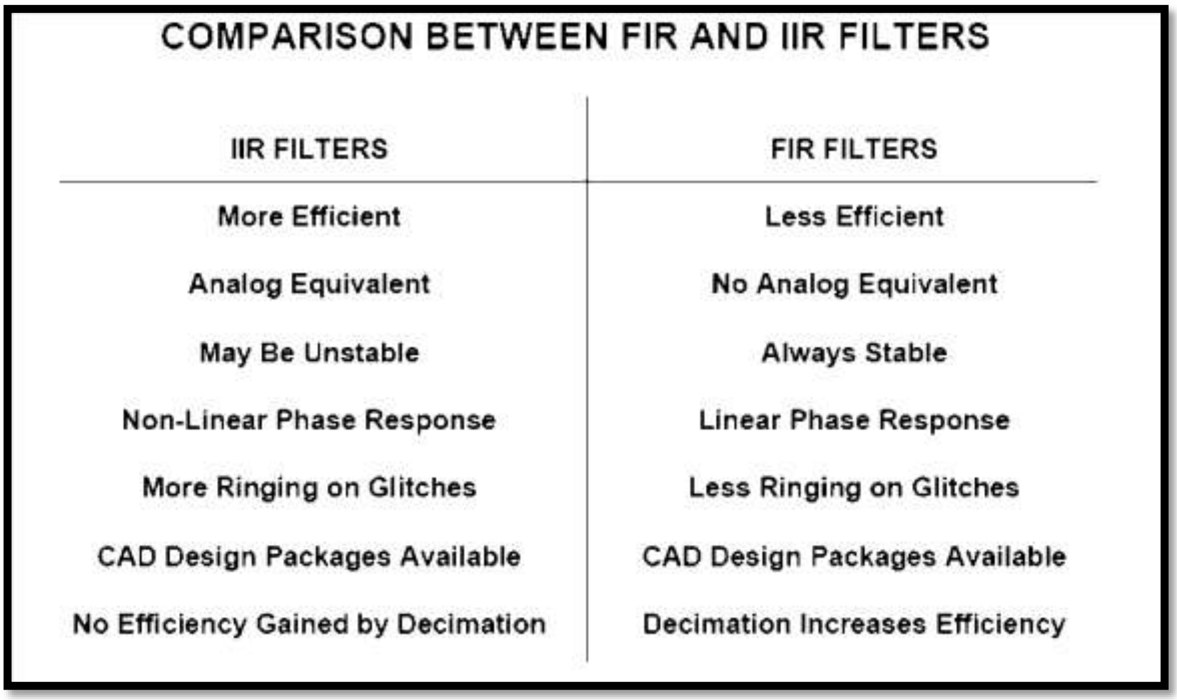
2.7 Summary, FIR versus IIR Filters

Choosing between FIR and IIR filter designs can be somewhat of a challenge, but a few basic guidelines can be given. Typically, IIR filters are more efficient than FIR filters because they require less memory and fewer multiply-accumulates are needed. IIR filters can be designed based upon previous experience with analog filter designs. IIR filters may exhibit instability problems, but this is much less likely to occur if higher order filters are designed by cascading second-order systems.

On the other hand, FIR filters require more taps and multiply-accumulates for a given cutoff frequency response, but have linear phase characteristics. Since FIR filters operate on a finite history of data, if some data is corrupted (ADC sparkle codes, for example) the FIR filter will ring for only N–1 samples. Because of the feedback, however, an IIR filter will ring for a considerably longer period of time.

If sharp cutoff filters are needed and processing time is at a premium, IIR elliptic filters are a good choice. If the number of multiply/accumulates is not prohibitive, and linear phase is a requirement, then the FIR should be chosen. [7]

Table 2.7 :comparison between FIR and IIR filter



COMPARISON BETWEEN FIR AND IIR FILTERS

IIR FILTERS	FIR FILTERS
More Efficient	Less Efficient
Analog Equivalent	No Analog Equivalent
May Be Unstable	Always Stable
Non-Linear Phase Response	Linear Phase Response
More Ringing on Glitches	Less Ringing on Glitches
CAD Design Packages Available	CAD Design Packages Available
No Efficiency Gained by Decimation	Decimation Increases Efficiency

2.8 Phase-shift keying

There are three major classes of digital modulation techniques used for transmission of digitally represented data:

- Amplitude-shift keying (ASK).
- Frequency-shift keying (FSK).
- Phase-shift keying (PSK).

All convey data by changing some aspect of a base signal, the carrier wave (usually a sinusoid), in response to a data signal. In the case of PSK, the phase is changed to represent the data signal. There are two fundamental ways of utilizing the phase of a signal in this way:

- By viewing the phase itself as conveying the information, in which case the demodulator must have a reference signal to compare the received signal's phase against; or
- By viewing the change in the phase as conveying information — differential schemes, some of which do not need a reference carrier (to a certain extent).

A convenient way to represent PSK schemes is on a constellation diagram. This shows the points in the complex plane where, in this context, the real and imaginary axes are termed the in-phase and quadrature axes respectively due to their 90° separation. Such a representation on perpendicular axes lends itself to straightforward implementation. The amplitude of each point along the in-phase axis is used to modulate a cosine (or sine) wave and the amplitude along the quadrature axis to modulate a sine (or cosine) wave.

In PSK, the constellation points chosen are usually positioned with uniform angular spacing around a circle. This gives maximum phase-separation between adjacent points and thus the best immunity to corruption. They are positioned on a circle so that they can all be transmitted with the same energy. In this way, the module of the complex numbers they represent will be the same and thus so will the amplitudes needed for the cosine and sine waves. Two common examples are "binary phase-shift keying" (BPSK) which uses two phases, and "quadrature phase-shift keying" (QPSK) which uses four phases, although any number of phases may be used. Since the data to be conveyed are usually binary, the PSK scheme is usually designed with the number of constellation points being a power of 2.[10]

2.8.1 Binary phase-shift keying (BPSK)

BPSK (also sometimes called PRK, Phase Reversal Keying, or 2PSK) is the simplest form of phase shift keying (PSK). It uses two phases which are separated by 180° and so can also be termed 2-PSK. It does not particularly matter exactly where the constellation points are positioned, and in this figure they are shown on the real axis, at 0° and 180°. This modulation is the most robust of all the PSKs since it takes the highest level of noise or distortion to make the demodulator reach an incorrect decision. It is, however, only able to modulate at 1 bit/symbol (as seen in the figure 2.8.1) and so is unsuitable for high data-rate applications when bandwidth is limited.

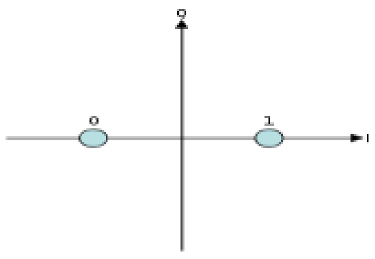


Figure 2.8.1 Constellation diagram example for BPSK.

In the presence of an arbitrary phase-shift introduced by the communications channel, the demodulator is unable to tell which constellation point is which. As a result, the data is often differentially encoded prior to modulation. [11]

Implementation:

The general form for BPSK follows the equation:

$$S_n(t) = \sqrt{\frac{2Eb}{T_b}}\cos\big(2\pi f_c + \pi(1-n)\big), n = 0,1,..$$

This yields two phases, 0 and π. In the specific form, binary data is often conveyed with the following signals:

for binary "0"

$$S_0(t) = \sqrt{\frac{2Eb}{T_b}}\cos(2\pi f_c + \pi) = -\sqrt{\frac{2Eb}{T_b}}\cos(2\pi f_c)$$

for binary "1"

$$S_1(t) = \sqrt{\frac{2Eb}{T_b}}\cos(2\pi f_c)$$

where f_c is the frequency of the carrier-wave.

Hence, the signal-space can be represented by the single basis function

$$\emptyset(t) = \sqrt{\frac{2}{T_b}}\cos(2\pi f_c)$$

where 1 is represented by $\sqrt{E_b}\emptyset(t)$ and 0 is represented by $-\sqrt{E_b}\emptyset(t)$. This assignment is, of course, arbitrary.[11]

2.9 Sampling Theory

Samples are successive snapshots of a signal. In the case of audio, the signal is a sound wave. A microphone converts the acoustic signal into a corresponding analog electric signal, and an analog-to-digital converter transforms that analog signal into a sampled digital form. The accuracy of the digital approximation of the analog signal depends on its resolution in time (the sampling rate) and its quantization, or resolution in amplitude(the number of bits used to represent each sample).

To convert an analog signal to a digital form it must first be band-limited and then sampled. Signals must be first filtered prior to sampling. Theoretically the maximum frequency that can be represented is half the sampling frequency. In practice a higher sample rate is used for non-ideal filters. The signal is now represented at multiples of the sampling period, T, as (nT) where n is an integer.

A typical signal processing system includes an A/D converter, D/A converter and a CPU that performs the signal processing algorithm. The input analog signal x(t) is first passed through an input filter (commonly called the anti-aliasing filter) whose function is to band limit the signal to below the Nyquist rate (one half the sampling frequency) to prevent aliasing. The signal is then digitized by the A/D converter at a rate determined by the sample clock to produce x(n), the discrete-time input sequence. The system transfer function, H (z) , is typically implemented in the time-domain using a linear difference equation. The sample output is y (n), then converted back into a continuous-time signal, y(t), by the D/A converter and output low-pass filter(see figure-2.9.1).

The calculation of the output signal using a difference equation requires a multiply and accumulate operation. This is typically a single-cycle instruction on DSP chips. [12]

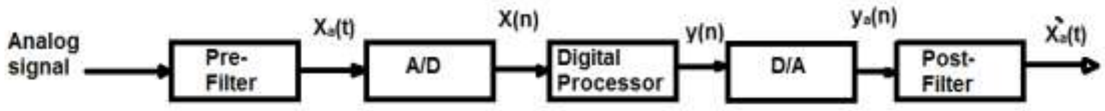


Figure.2.9.1 signal processing system

2.9.1 The sampling theorem:

If the highest frequency contained in an analog signal $x_a(t)$ is

$F_{max} = B$, and the signal is sampled at a frequency $Fs > 2B$, then the analog signal can be exactly recovered from its samples using the following reconstruction formula:

$$\mathrm{X_a}(t) = \sum_{n=-\infty}^{\infty} X_a(nT) \frac{\sin(\left(\frac{\pi}{T}\right)(t-nT))}{(\frac{\pi}{T})(t-nT)}$$

Note that at the original sample instances (t = nT), the reconstructed analog signal is equal to the value of the original analog signal because the sinc functions take on values of zero at multiples of the sample period. At times between the sample instances, the signal is the weighted sum of shifted sinc functions.[3]

For : (Fs < 2B) the analog signal image centered at (2π/T) overlaps into the base band image. The distortion caused by high frequencies overlapping low frequencies is called aliasing. To avoid aliasing distortion, either the input analog signal has to be band limited to a maximum of half the sampling frequency, or the sampling frequency has to be increased to at least twice the highest frequency in the analog signal.

2.10. Audio Effects

In natural environments, sounds we perceive depend on the listening conditions and in particular on the acoustic environment. The same sound signal played in a concert hall; bathroom or a small room will not be “perceived” the same. Since these effects are important for musicians, the digital technology can be used to simulate them. In this section, we will explain some of these effects to our audio signal.

2.10.1 Delay

Delay is the simplest audio effect which holds input signal and then plays it back after a period of time. Delay is used very often by musicians. It is also the main block for other audio effects such as reverb, chorus, and flanging.

The difference equation for the delay operation is:
$y[n] = x[n-n_0]$

where'n_0'is the delay amount. Since the difference equation between the output and the input is specified, it can be directly coded in language. To implement the delay operation, the best way is defining an array which stores input audio signals. In Figure 2.11.1, we demonstrate the delay operation using “n_0+1” length array which should be defined beforehand. To feed the delayed audio signal to output, first we should store the audio signal on the first entry of the array. At each operation cycle, each entry in the array should be shifted towards right, to open space to the new input. This way, an input which is taken at time “n” will reach to the end of the array “n_0” cycles later. [13]

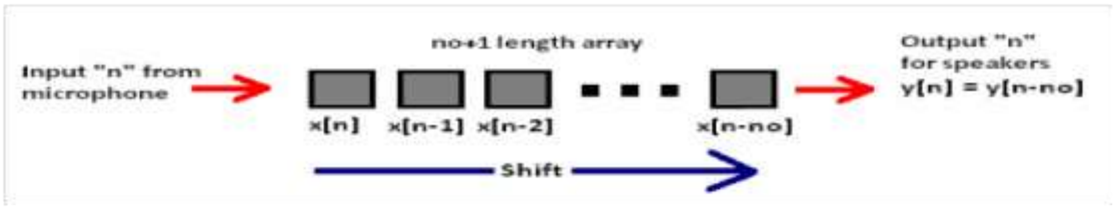


Figure 2.10.1 Demonstration of the delay operation.

2.10.2 Echo

Echo is the repetition of a waveform due to reflection from points where the characteristics of the medium through which the wave propagates changes. Echo is usefully employed in sonar and radar for detection and exploration purposes. Echo has a time delay that is relative long, which is more than 1 second. So that in echo effect, the true sound and the artificial sound are clearly separated, so that human hearings can tell the difference.[14]

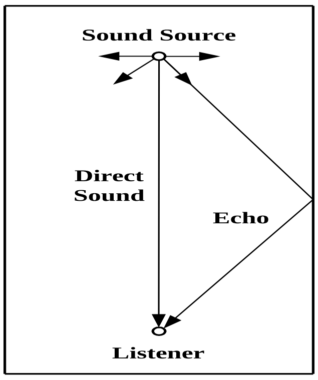


Fig. 2.10.2 Echo mechanism

As seen in Fig. 2.10.2, the signal goes from the source to the listener in two paths [2]. First, the signal from the source goes directly to the listener. Second, the signal goes to the wall and then reflected to the user. The second process will takes more time than the first process, so the listener will hear two sounds in a different period of time. The signal power from the second process will be attenuated due to the reflection process. [15]

The echo block simply takes an audio signal and plays it after summing with a delayed version of it. The delay time can range from several milliseconds to several seconds. The basic echo block is given in Figure 2.10.3.

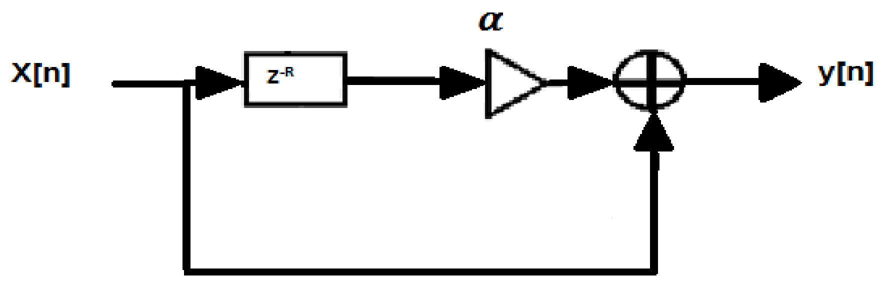


Figure 2.10.3 Echo block.

The difference equation for this system is :

$$y[n] = x[n] + \alpha x[n-n_0]$$

where 'α' is the delay mix parameter, 'n_0' is the delay amount. As can be seen in this equation, to generate the echo effect we need both the present and the delayed signal. In order to access to the delayed signal, we should store the input audio signal in an array as in previous section. Using this array, we can process the audio signal $x[n-n_0]$. Summing this $x[n- n_0]$ value (after multiplying with α) with the present input x[n], we can generate the echo effect. [16]

Chapter 3

Hardware and software

3.1. Introduction

DSP is one of the core technologies, in rapidly growing application areas, such as wireless communications, audio and video processing and industrial control.

DSPs such as the TMS320C6x (C6x) family of processors are like fast special-purpose microprocessors with a specialized type of architecture and an instruction set appropriate for signal processing .The TMS320C62x is a 16-bit fixed point processor and the ‘67x is a floating point processor, with 32-bit integer support. The discussion in this chapter is focused on the TMS320C67x processor. [17]

The C6713 DSK is a low-cost standalone development platform that enables users to evaluate and develop applications for the TI C67xx DSP family. The DSK also serves as a hardware reference design for the TMS320C6713 DSP. Schematics, logic equations and application notes are available to ease hardware development and reduce time to market. [18]

The C6713 DSK is a low-cost standalone development platform that enables users to evaluate and develop applications for the TI C67xx DSP family. The DSK also serves as a hardware reference design for the TMS320C6713 DSP. Schematics, logic equations and application notes are available to ease hardware development and reduce time to market. [19]

3.2. Diagrams of TMS320C6713 DSK

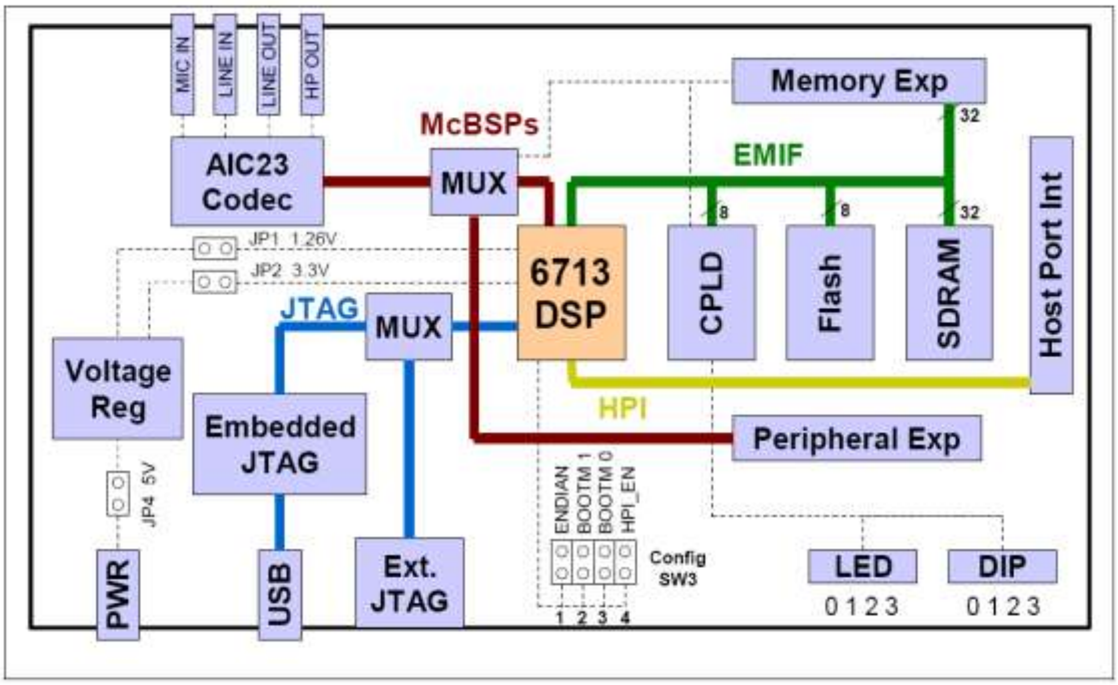


Figure 3.2.1 Block diagram of TMS320C6713 DSK

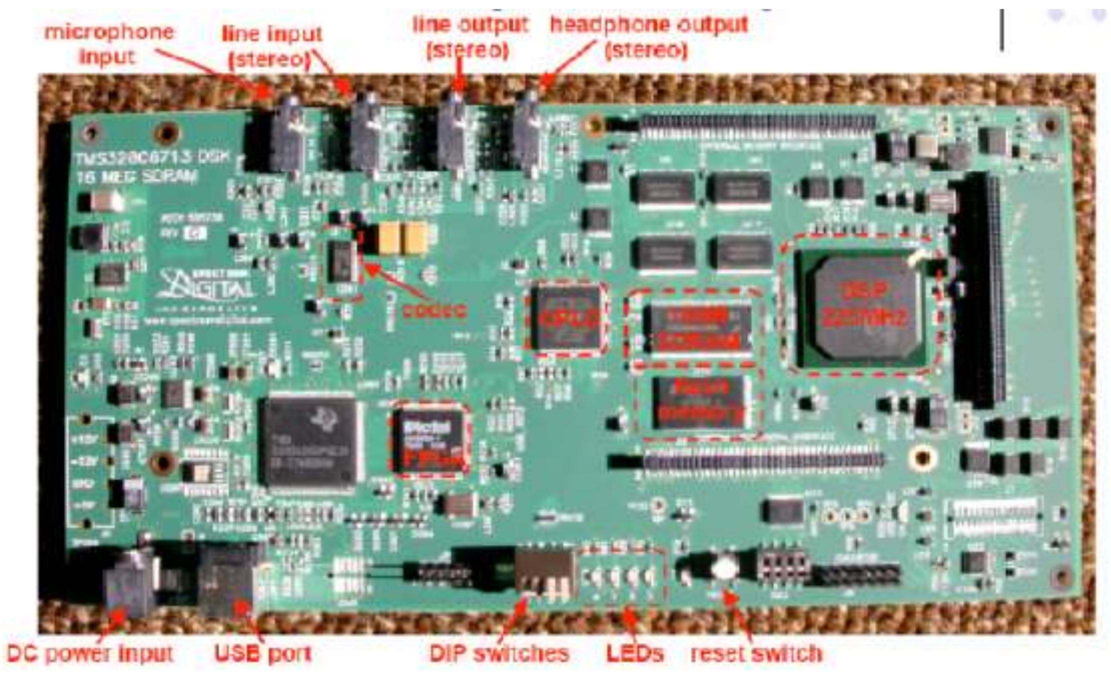


Figure 3.2.2 Board diagram of TMS320C6713 DSK

3.3. Features

The DSK comes with a full compliment of on-board devices that suit a wide variety of application environments. Key features include:

- A Texas Instruments TMS320C6713 DSP operating at 225 MHz.
- An AIC23 stereo codec.
- 16 Mbytes of synchronous DRAM.
- 512 Kbytes of non-volatile Flash memory (256 Kbytes usable in default configuration).
- 4 user accessible LEDs and DIP switches.
- Software board configuration through registers implemented in CPLD.
- Configurable boot options.
- Standard expansion connectors for daughter card use.
- JTAG emulation through on-board JTAG emulator with USB host.
- Interface or external emulator.
- Single voltage power supply (+5V).

3.4. Functional Overview

The DSP on the 6713 DSK interfaces to on-board peripherals through a 32-bit wide EMIF (External Memory Inter Face). The SDRAM, Flash and CPLD are all connected to the bus. EMIF signals are also connected daughter card expansion connectors which are used for third party add-in boards.

The DSP interfaces to analog audio signals through an on-board AIC23 codec and four 3.5 mm audio jacks (microphone input, line input, line output, and headphone output). The codec can select the microphone or the line input as the active input. The analog output is driven to both the line out (fixed gain) and headphone (adjustable gain) connectors. McBSP0 is used to send commands to the codec control interface while McBSP1 is used for digital audio data. McBSP0 and McBSP1 can be re-routed to the expansion connectors in software.

A programmable logic device called a CPLD is used to implement glue logic that ties the board components together. The CPLD has a register based user interface that lets the user configure the board by reading and writing to its registers.
The DSK includes 4 LEDs and a 4 position DIP switch as a simple way to provide the user with interactive feedback. Both are accessed by reading and writing to the CPLD registers. An included 5V external power supply is used to power the board. On-board switching voltage regulators provide the +1.26V DSP core voltage and +3.3V I/O supplies. The board is held in reset until these supplies are within operating specifications.
Code Composer communicates with the DSK through an embedded JTAG emulator with a USB host interface. The DSK can also be used with an external emulator through the external JTAG connector.

3.5. Architecture

The simplified architecture of TMS320C6713 is shown in the Figure 3.5.1 below. The processor consists of three main parts: CPU, peripherals and memory.

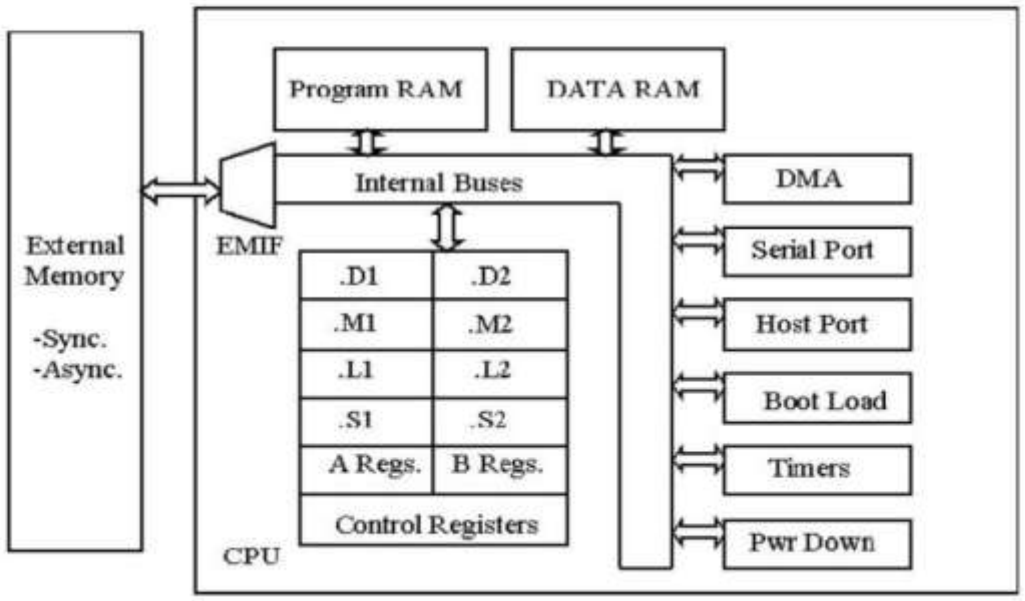


Figure 3.5.1: Simplified block diagram of TMS320C67xx family

3.5.1.CPLD (Programmable Logic)

The C6713 DSK uses an Altera EPM3128TC100-10 Complex Programmable Logic Device (CPLD) device to implement:

• 4 Memory-mapped control/status registers that allow software control of various board features.

• Control of the daughter card interface and signals.

• Assorted "glue" logic that ties the board components together.

3.5.1.1. CPLD Overview

The CPLD logic is used to implement functionality specific to the DSK. Your own hardware designs will likely implement a completely different set of functions or take advantage of the DSPs high level of integration for system design and avoid the use of external logic completely. The CPLD implements simple random logic functions that eliminate the need for additional discrete devices. In particular, the CPLD aggregates the various reset signals coming from the reset button and power supervisors and generates a global reset.

The EPM3128TC100-10 is a 3.3V (5V tolerant), 100-pin QFP device that provides 128 macro cells, 80 I/O pins, and a 10 ns pin-to-pin delay. The device is EEPOM-based and is in-system programmable via a dedicated JTAG interface (a 10-pin header on the DSK). The CPLD source files are written in the industry standard VHDL (Hardware Design Language) and included with the DSK.

3.5.1.2. CPLD Registers

The 4 CPLD memory-mapped registers allows users to control CPLD functions in software. On the 6713 DSK the registers are primarily used to access the LEDs and DIP switches and control the daughter card interface. The registers are mapped into EMIF CE1 data space at address 0x90080000. They appear as 8-bit registers with a simple asynchronous memory interface. The following table gives a high level overview of the CPLD registers and their bit fields:

The table below shows the bit definitions for the 4 registers in CPLD.

Table 3.5.1: CPLD Register Definitions

Offset	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	USER_REG	USR_SW3 R	USR_SW2 R	USR_SW1 R	USR_SW0 R	USR_LED3 R/W 0(Off)	USR_LED2 R/W 0(Off)	USR_LED1 R/W 0(Off)	USR_LED0 R/W 0(Off)
1	DC_REG	DC_DET R	0	DC_STAT1 R	DC_STAT0 R	DC_RST R 0(No reset)	0	DC_CNTL1 R/W 0(low)	DC_CNTL0 R/W 0(low)
4	VERSION	CPLD_VER[3.0] R				0	BOARD VERSION[2.0] R		
6	MISC	SCR_5 R/W 0	SCR_4 R/W 0	SCR_3 R/W 0	SCR_2 R/W 0	SCR_1 R/W 0	FLASH_PAGE R/W 0 (Flash A19=0)	McBSP1 ON/OFF Board R/W 0 (Onboard)	McBSP0 ON/OFF Board R/W 0 (Onboard)

3.5.1.3. USER_REG Register

USER_REG is used to read the state of the 4 DIP switches and turn the 4 LEDs on or off to allow the user to interact with the DSK. The DIP switches are read by reading the top 4 bits of the register and the LEDs are set by writing to the low 4 bits.

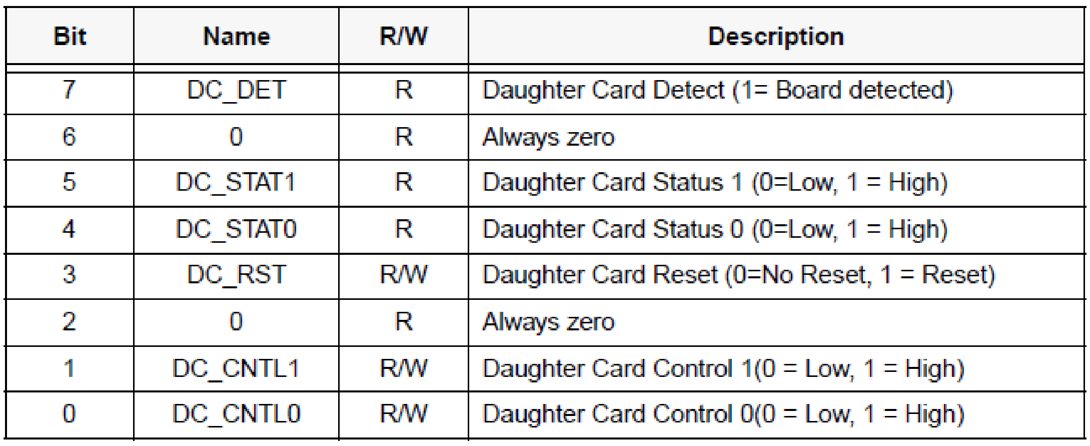
Table 3.5.2: CPLD USER_REG Register

Bit	Name	R/W	Description
7	USER_SW3	R	User DIP Switch 3(1 = Off, 0 = On)
6	USER_SW2	R	User DIP Switch 2(1 = Off, 0 = On)
5	USER_SW1	R	User DIP Switch 1(1 = Off, 0 = On)
4	USER_SW0	R	User DIP Switch 0(1 = Off, 0 = On)
3	USER_LED3	R/W	User-defined LED 3 Control (0 = Off, 1 = On)
2	USER_LED2	R/W	User-defined LED 2 Control (0 = Off, 1 = On)
1	USER_LED1	R/W	User-defined LED 1 Control (0 = Off, 1 = On)
0	USER_LED0	R/W	User-defined LED 0 Control (0 = Off, 1 = On)

3.5.1.4. DC_REG Register

DC_REG is used to monitor and control the daughter card interface. DC_DET detects the presence of a daughter card. DC_STAT and DC_CNTL provide simple communications with the daughter card through readable status lines and writable control lines. The daughter card is released from reset when the DSP is released from reset. DC_RST can be used to put the card back in reset.

Table 3.5.13.: DC_REG Register

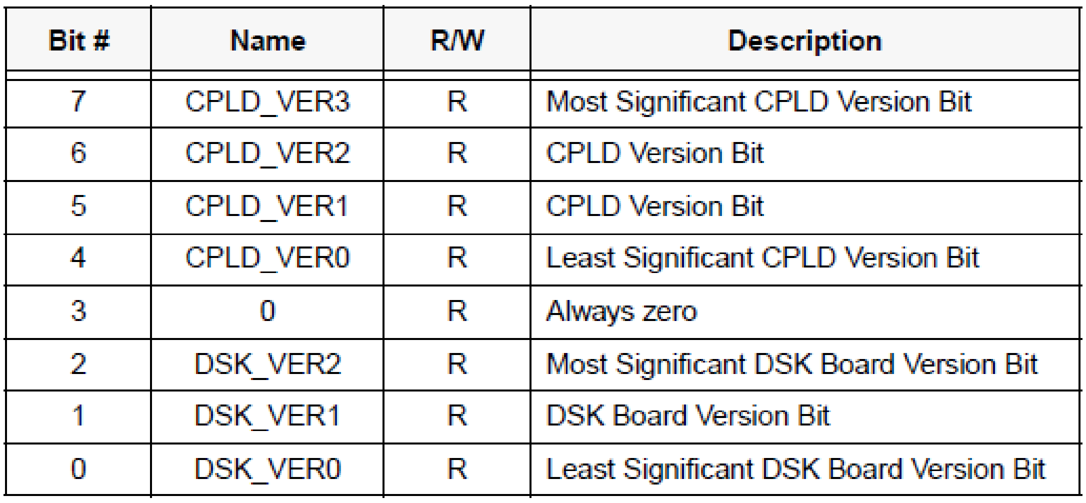


Bit	Name	R/W	Description
7	DC_DET	R	Daughter Card Detect (1= Board detected)
6	0	R	Always zero
5	DC_STAT1	R	Daughter Card Status 1 (0=Low, 1 = High)
4	DC_STAT0	R	Daughter Card Status 0 (0=Low, 1 = High)
3	DC_RST	R/W	Daughter Card Reset (0=No Reset, 1 = Reset)
2	0	R	Always zero
1	DC_CNTL1	R/W	Daughter Card Control 1(0 = Low, 1 = High)
0	DC_CNTL0	R/W	Daughter Card Control 0(0 = Low, 1 = High)

3.5.1.5 VERSION Register

The VERSION register contains two read only fields that indicate the BOARD and CPLD versions. This register will allow your software to differentiate between production releases of the DSK and account for any variances. This register is not expected to change often, if at all.

Table 3.5.4: Version Register Bit Definitions



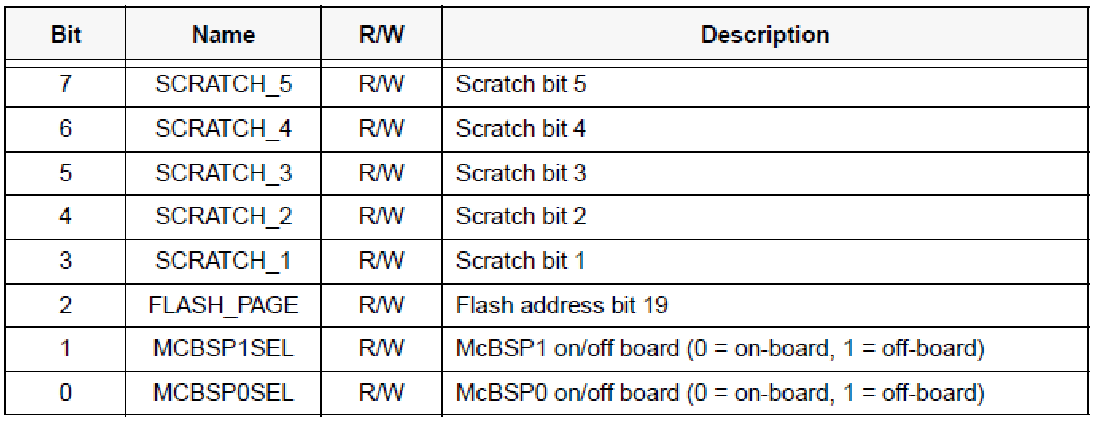
Bit #	Name	R/W	Description
7	CPLD_VER3	R	Most Significant CPLD Version Bit
6	CPLD_VER2	R	CPLD Version Bit
5	CPLD_VER1	R	CPLD Version Bit
4	CPLD_VER0	R	Least Significant CPLD Version Bit
3	0	R	Always zero
2	DSK_VER2	R	Most Significant DSK Board Version Bit
1	DSK_VER1	R	DSK Board Version Bit
0	DSK_VER0	R	Least Significant DSK Board Version Bit

3.5.1.6. MISC Register

The MISC register is used to provide software control for miscellaneous board functions. On the 6713 DSK, the MISC register controls how auxiliary signals are brought out to the daughter-card connectors.
McBSP0 and McBSP1 are usually used as the control and data ports of the on-board AIC23 codec. The power-on state of these bits (both 0s) represents that situation. Set

the corresponding McBSP select bit to use the McBSP with a daughter card instead. The Flash and CPLD share CE1 which means that the highest DSP address bit (A21) is used to differentiate between the two. The FLASH_PAGE bit is driven to the Flash as a replacement for that address line which is connected to A19 of the Flash. On a standard DSK, the on-board Flash is not large enough for this bit to be significant. FLASH_PAGE is only useful if the board is re-populated with a larger pin-compatible Flash chip. The scratch bits are unused. They can be set to any value.

Table 3.5.5: MISC Register



Bit	Name	R/W	Description
7	SCRATCH_5	R/W	Scratch bit 5
6	SCRATCH_4	R/W	Scratch bit 4
5	SCRATCH_3	R/W	Scratch bit 3
4	SCRATCH_2	R/W	Scratch bit 2
3	SCRATCH_1	R/W	Scratch bit 1
2	FLASH_PAGE	R/W	Flash address bit 19
1	MCBSP1SEL	R/W	McBSP1 on/off board (0 = on-board, 1 = off-board)
0	MCBSP0SEL	R/W	McBSP0 on/off board (0 = on-board, 1 = off-board)

3.5.2. AIC23 Codec

The DSK uses a Texas Instruments AIC23 (part #TLV320AIC23) stereo codec for input and output of audio signals. The codec samples analog signals on the microphone or line inputs and converts them into digital data so it can be processed by the DSP.

When the DSP is finished with the data it uses the codec to convert the samples back into analog signals on the line and headphone outputs so the user can hear the output. The codec communicates using two serial channels, one to control the codec's internal configuration registers and one to send and receive digital audio samples. McBSP0 is used as the unidirectional control channel. It should be programmed to send a 16-bit control word to the AIC23 in SPI format. The top 7 bits of the control word should specify the register to be modified and the lower 9 should contain the register value.

The control channel is only used when configuring the codec, it is generally idle when audio data is being transmitted, McBSP1 is used as the bi directional data channel. All audio data flows through the data channel. Many data formats are supported based on the three variables of sample width, clock signal source and serial data format. The DSK examples generally use a 16-bit sample width with the codec in master mode so it generates the frame sync and bit clocks at the correct sample rate without effort on the DSP side. The preferred serial format is DSP mode which is designed specifically to operate with the McBSP ports on TI DSPs.

The codec has a 12MHz system clock. The 12MHz system clock corresponds to USB sample rate mode, named because many USB systems use a 12MHz clock and can use the same clock for both the codec and USB controller. The internal sample rate generate subdivides the 12MHz clock to generate common frequencies such as 48KHz, 44.1KHz and 8KHz. The sample rate is set by the codec's SAMPLERAT register. The figure below shows the codec interface on the C6713 DSK.

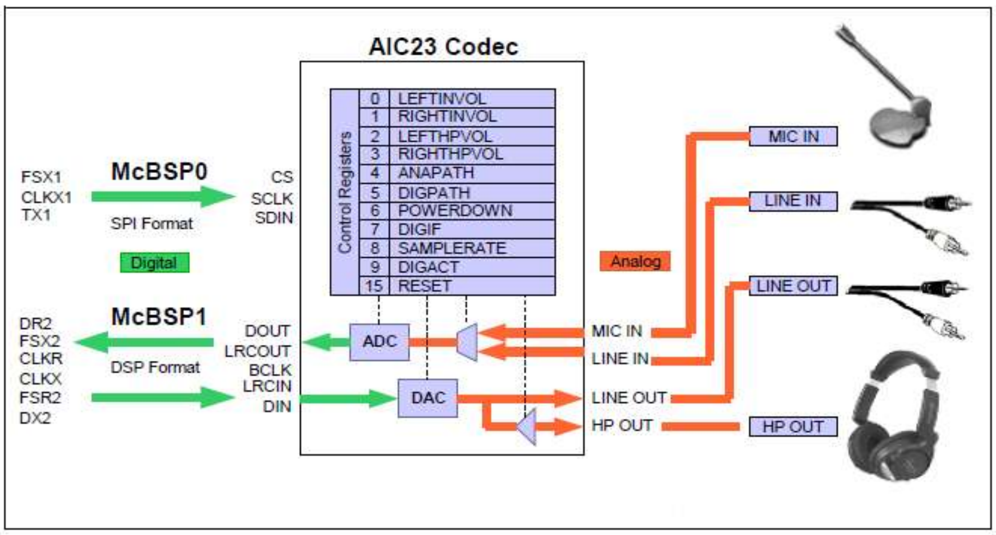


Figure 3.5.2: TMS320C6713 DSK CODEC INTERFACE

3.5.3. Synchronous DRAM

The DSK uses a 64 megabit synchronous DRAM (SDRAM) on the 32-bit EMIF. The SDRAM is mapped at the beginning of CE0 (address 0x80000000). Total available memory is 8 megabytes. The integrated SDRAM controller is part of the EMIF and must be configured in software for proper operation. The EMIF clock is derived from the PLL settings and should be configured in software at 90MHz. This number is based on an internal PLL clock of 450MHz required to achieve 225 MHz operation with a divisor of 2 and a 90MHz EMIF clock with a divisor of 5. When using SDRAM, the controller must be set up to refresh one row of the memory array every 15.6 microseconds to maintain data integrity. With a 90MHz EMIF clock, this period is 1400 bus cycles.

3.5.4. Flash Memory

Flash is a type of memory which does not lose its contents when the power is turned off. When read it looks like a simple asynchronous read-only memory (ROM). Flash can be erased in large blocks commonly referred to as sectors or pages. Once a block has been erased each word can be programmed once through a special command sequence. After that the entire block must be erased again to change the contents. The DSK uses a 512Kbyte external Flash as a boot option. It is visible at the beginning of CE1 (address 0x90000000). The Flash is wired as a 256K by 16 bit device to support the DSK's 16-bit boot option. However, the software that ships with the DSK treats the Flash as an 8-bit device (ignoring the top 8 bits) to match the 6713's default 8-bit boot mode. In this configuration, only 256Kbytes are readily usable without software changes.

3.5.5. LEDs and DIP Switches

The DSK includes 4 software accessible LEDs (D7-D10) and DIP switches (SW1) that provide the user a simple form of input/output. Both are accessed through the CPLD USER_REG register.

3.5.6. Daughter Card Interface

The DSK provides three expansion connectors that can be used to accept plug-in daughter cards. The daughter card allows users to build on their DSK platform to extend its capabilities and provide customer and application specific I/O. The expansion connectors are for memory, peripherals, and the Host Port Interface (HPI) The memory connector provides access to the DSP’s asynchronous EMIF signals to interface with memories and memory mapped devices. It supports byte addressing on 32 bit boundaries. The peripheral connector brings out the DSP’s peripheral signals like McBSPs, timers, and clocks. Both connectors provide power and ground to the daughter card The HPI is a high speed interface that can be used to allow multiple DSPs to communicate and cooperate on a given task. The HPI connector brings out the HPI specific control signals. Most of the expansion connector signals are buffered so that the daughter card cannot directly influence the operation of the DSK board. The use of TI low voltage, 5V tolerant buffers, and CBT interface devices allows the use of either +5V or +3.3V devices to be used on the daughter card.

Other than the buffering, most daughter card signals are not modified on the board. However, a few daughter card specific control signals like DC_RESET and DC_DET exist and are accessible through the CPLD DC_REG register. The DSK also multiplexes the McBSP0 and McBSP1 of on-board or external use. This function is controlled through the CPLD MISC register.

3.6. Physical Description

3.6.1. Board Layout

The C6713 DSK is a 8.75 x 4.5 inch (210 x 115 mm.) multi-layer board which is powered by an external +5 volt only power supply. Figure 3.6.1 shows the layout of the C6713 DSK.

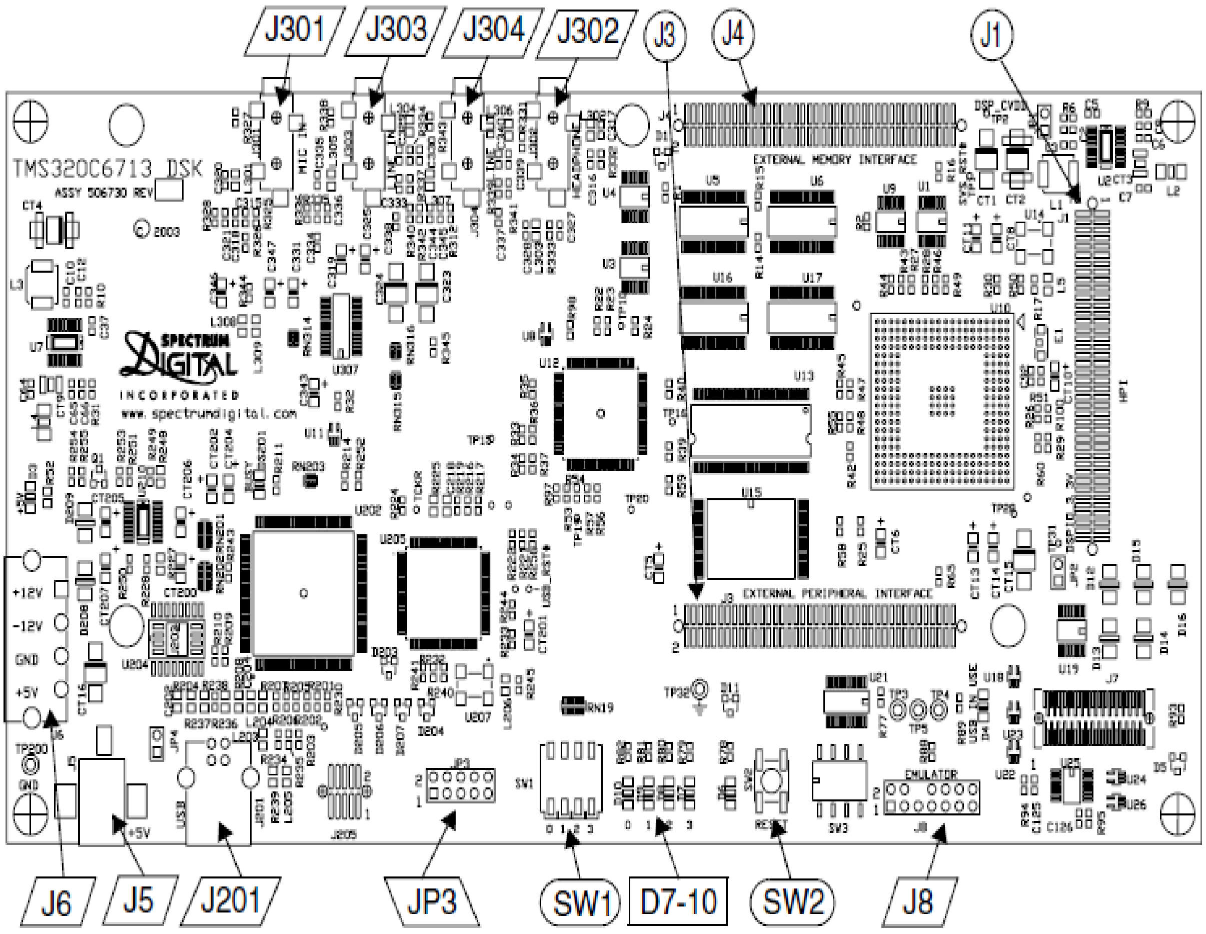
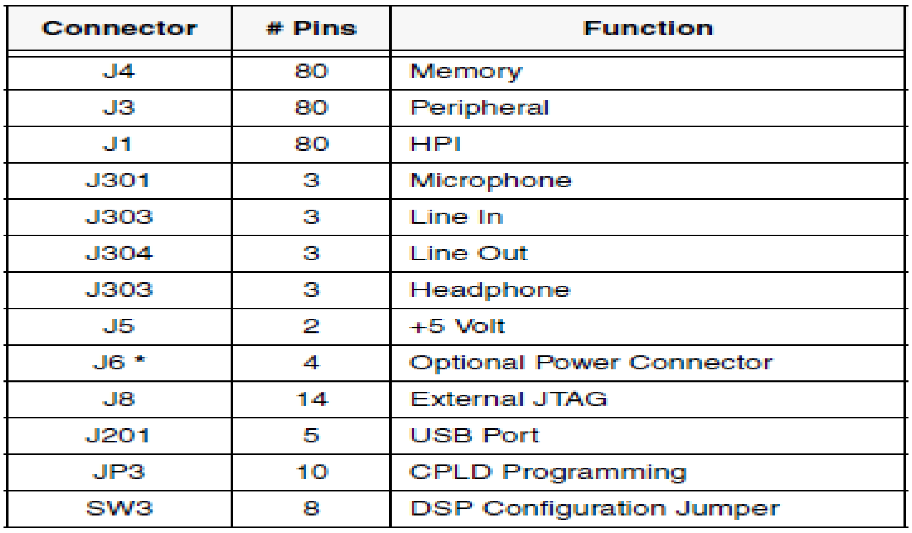


Figure 3.6.1, TMS320C6713 DSK

3.6.2. Connector Index

The TMS320C6713 DSK has many connectors which provide the user access to the various signals on the DSK.

Table 3.6.1: TMS320C6713 DSK Connectors



Connector	# Pins	Function
J4	80	Memory
J3	80	Peripheral
J1	80	HPI
J301	3	Microphone
J303	3	Line In
J304	3	Line Out
J303	3	Headphone
J5	2	+5 Volt
J6 *	4	Optional Power Connector
J8	14	External JTAG
J201	5	USB Port
JP3	10	CPLD Programming
SW3	8	DSP Configuration Jumper

3.6.3. Expansion Connectors

The TMS320C6713 DSK supports three expansion connectors that follow the Texas Instruments interconnection guidelines. The expansion connector pin outs are described in the following three sections.

The three expansion connectors are all 80 pin 0.050 x 0.050 inches low profile connectors from Samtec or AMP. The Samtec SFM Series (surface mount) connectors are designed for high speed interconnections because they have low propagation delay, capacitance, and cross talk. The connectors present a small foot print on the DSK. Each connector includes multiple ground, +5V, and +3.3V power signals so that the daughter card can obtain power directly from the DSK. The peripheral expansion connector additionally provides both +12V and -12V to the daughter card. The recommended mating connector, whose part number is TFM-140-32-S-D-LC, is a surface mount connector that provides a 0.465” mated height.

3.6.4. Audio Connectors

The C6713 DSK has 4 audio connectors. They are described in the following Sections:

1. J301, Microphone Connector

The input is a 3.5 mm. stereo jack. Both inputs are connected to the microphone so it is monaural. The signals on the plug are shown in the figure below:

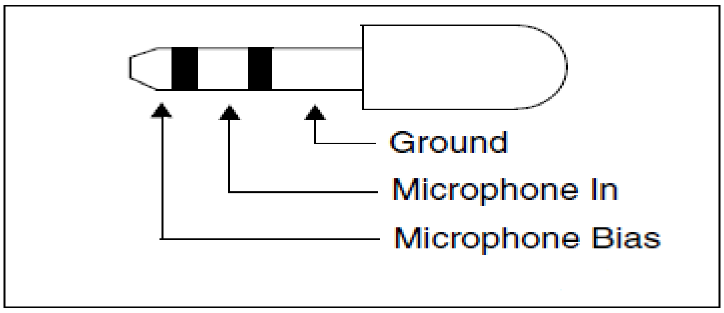


Figure 3.6.2, Microphone Stereo Jack

2. J303, Audio Line In Connector

The audio line in is a stereo input. The input connector is a 3.5 mm stereo jack. The signals on the mating plug are shown in the figure below:

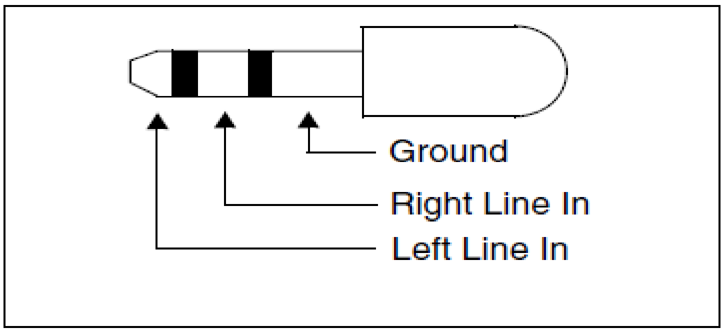


Figure 3.6.3, Audio Line In Stereo Jack

3. J304, Audio Line Out Connector

The audio line out is a stereo output. The output connector is a 3.5 mm stereo jack. The signals on the mating plug are shown in the figure below:

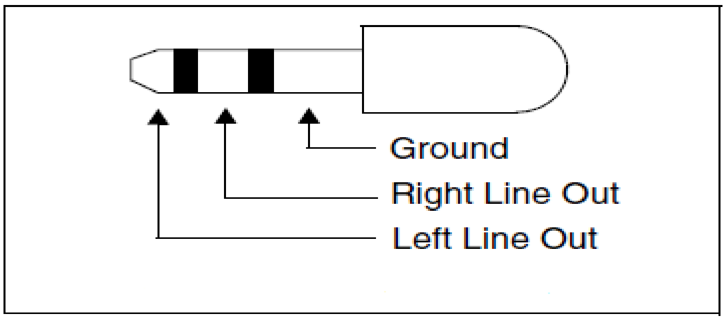


Figure 3.6.4, Audio Line Out Stereo Jack

4. J303, Headphone Connector

Connector J4 is a headphone/speaker jack. It can drive standard headphones or a high impedance speaker directly. The standard 3.5 mm jack is shown in the figure below:

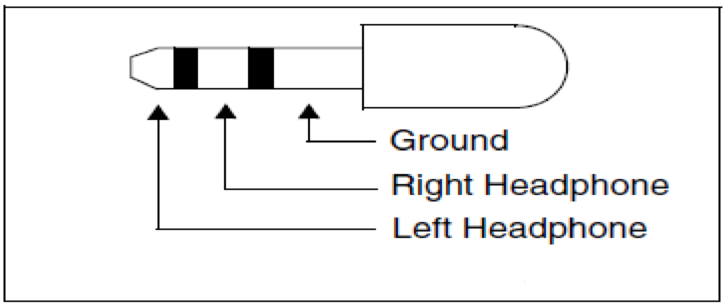


Figure 3.6.5, Headphone Jack

3.7. Power Connectors

The C6713 DSK has 2 power connectors. They are described in the following Sections:

1. J5, +5 Volt Connector

Power (+5 volts) is brought onto the TMS320C6713 DSK via the J5 connector. The connector has an outside diameter of 5.5 mm. and an inside diameter of 2.5 mm. The A diagram of J5 is shown below:

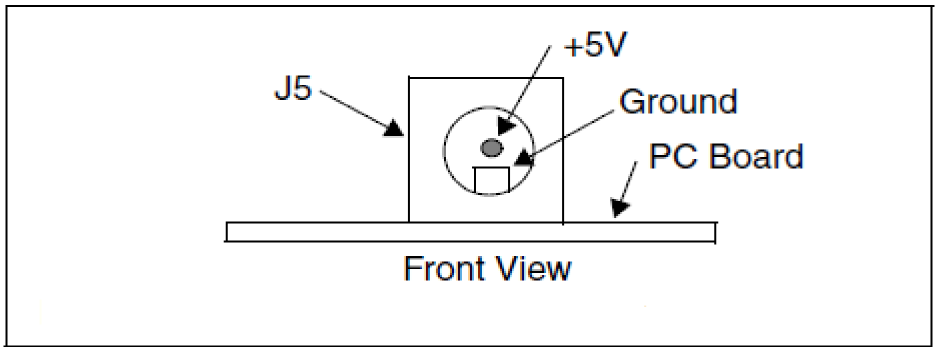
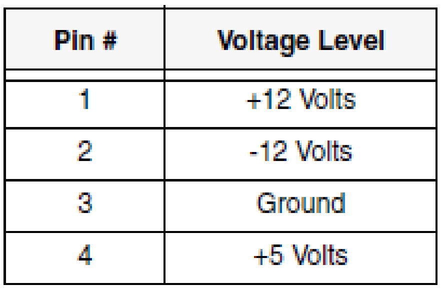


Figure 3.7.1, TMS320C6713 DSK Power Connector

2. J6, Optional Power Connector

Connector J6 is an optional power connector. It will operate with the standard personal computer power supply. To populate this connector use a Molex #15-24-4041. The table 3.7.1below shows the voltages on the respective pins.

Table 3.7.1: J6, Optional Power Connector



Pin #	Voltage Level
1	+12 Volts
2	-12 Volts
3	Ground
4	+5 Volts

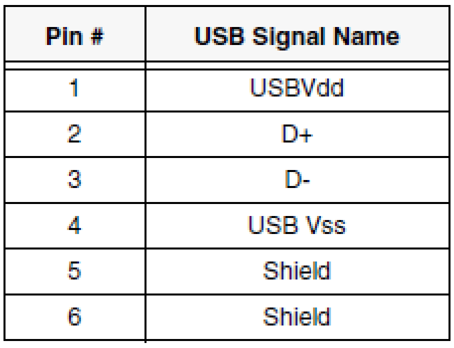
3.8. Miscellaneous Connectors

The C6713 DSK has 3 additional connectors to aid the user in developing with this product. They are described in the following sections:

1. J201, USB Connector

Connector J201 provides a Universal Serial Bus (USB) Interface to the embedded JTAG emulation logic on the DSK . This allows for code development and debug without the use of an external emulator. The signals on this connector are shown in the table below:

Table 3.8.1: J201, USB Connector



Pin #	USB Signal Name
1	USBVdd
2	D+
3	D-
4	USB Vss
5	Shield
6	Shield

2. J8, External JTAG Connector

The TMS320C6713 DSK is supplied with a 14 pin header interface, J8. This is the standard interface used by JTAG emulators to interface to Texas Instruments DSPs. The pin out for the connector is shown in the figure below:

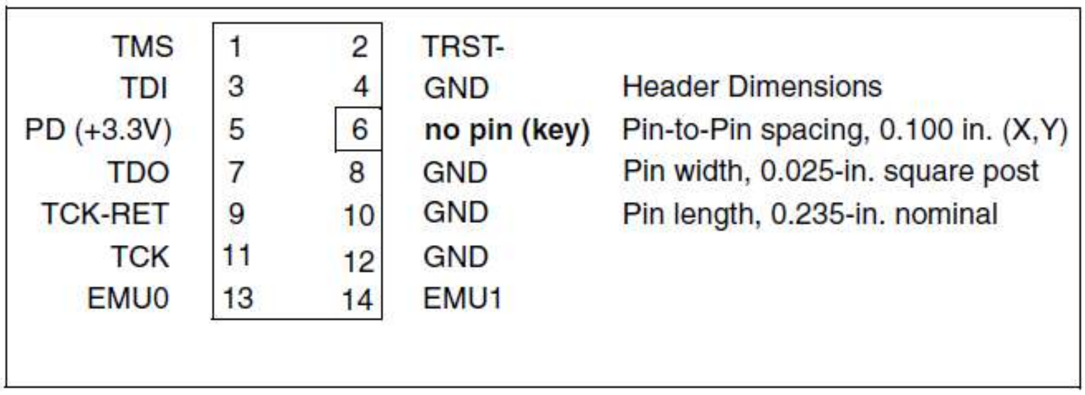


Figure 3.8.1, J8, JTAG INTERFACE

3. JP3, PLD Programming Connector

This connector interfaces to the Altera CPLD, U12. It is used in the in the factory for the programming of the CPLD. This connector is not intended to be used outside the factory.

3.9 System LEDs

TheTMS320C6713 DSK has four system light emitting diodes (LEDs). These LEDs indicate various conditions on the DSK. This function of each LED is shown in the table below:

Table 3.9.1: System LEDs

Reference Designator	Color	Function	On Signal State
D4	Green	USB Emulation in use. When External JTAG Emulator is used this LED is off.	1
D3	Green	+5 Volt present	1
D6	Orange	RESET Active	1
DS201	Green	USB Active, Blinks during USB data transfer	1

3.10. Reset Switch

There are three resets on the TMS320C6713 DSK. The first reset is the power on reset. This circuit waits until power is within the specified range before releasing the power on reset pin to the TMS320C6713. External sources which control the reset are push button SW2, and the on board embedded USB JTAG emulator.

3.11. Software

3.11.1 Code Composer Studio (CCS)

Figure 3.11.1: CCS face.

CCS provides an integrated development environment (IDE) to incorporate the software tools. CCS includes tools for code generation, such as a C compiler, an assembler, and a linker. It has graphical capabilities and supports real time debugging. It provides an easy-to-use software tool to build and debug programs. The C compiler compiles a C source program with extension .c to produce an assembly source file with extension.asm. The assembler assembles an.asm source file to produce a machine language object file with extension.obj. The linker combines object files and object libraries as input to produce an executable file with extension. out. This executable file represents a linked common object file format (COFF), popular in Unix-based systems and adopted by several makers of digital signal processors .This executable file can be loaded and run directly on the C6713 processor. A linear optimizer optimizes this source file to create an assembly file with extension .asm (similar to the task of the C compiler).

To create an application project, one can “add” the appropriate files to the project. Compiler/linker options can readily be specified. A number of debugging features are available, including setting breakpoints and watching variables; viewing memory, registers, and mixed C and assembly code; graphing results; and monitoring execution time. One can step through a program in different ways (step into, over, or out).

Real-time analysis can be performed using real-time data exchange (RTDX). RTDX allows for data exchange between the host PC and the target DSK, as well as analysis in real time without stopping the target. Key statistics and performance can be monitored in real time. Through the joint team action group (JTAG), communication with on-chip emulation support occurs to control and monitor program execution. The C6713 DSK board includes a JTAG interface through the USB port.

3.11.2. CCS installation and Support

Use the USB cable to connect the DSK board to the USB port on the PC. Use the 5-V power supply included with the DSK package to connect to the +5-V power connector on the DSK to turn it on. Install CCS with the CD-ROM included with the DSK.

3.11.3. Simulink in Matlab

Figure 3.11.2: Matlab Simulink

Simulink is an environment for multi domain simulation and Model-Based Design for dynamic and embedded systems. It provides an interactive graphical environment and a customizable set of block libraries that let you design, simulate, implement, and test a variety of time-varying systems, including communications, controls, signal processing, video processing, and image processing. In our project we use the Simulink tool to design the experiments in order to produce the C++ code to down load it on the 6713 Kit using CCS program. We will use (MATLAB R2006a) in our project.

Chapter 4

Design and conclusion

4.1 Sine wave generation :

In sine wave generation experiment , we need to generate a sine wave from the DSK6713 internally ,by add DSP sine wave block to the circuit as fig.4.1.1 , in order to down load it on the Kit in associate with code composer ; we put a DAC block and the C6713 DSK symbol.

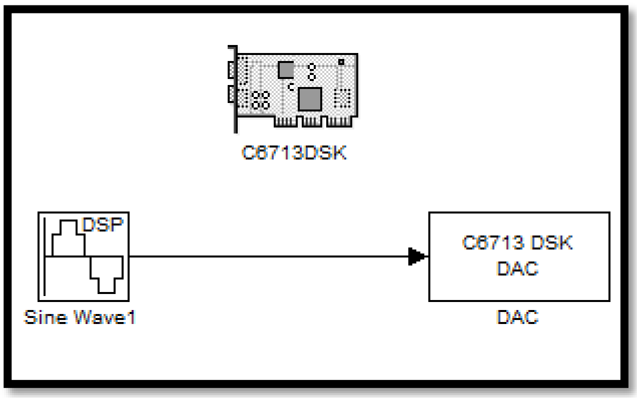
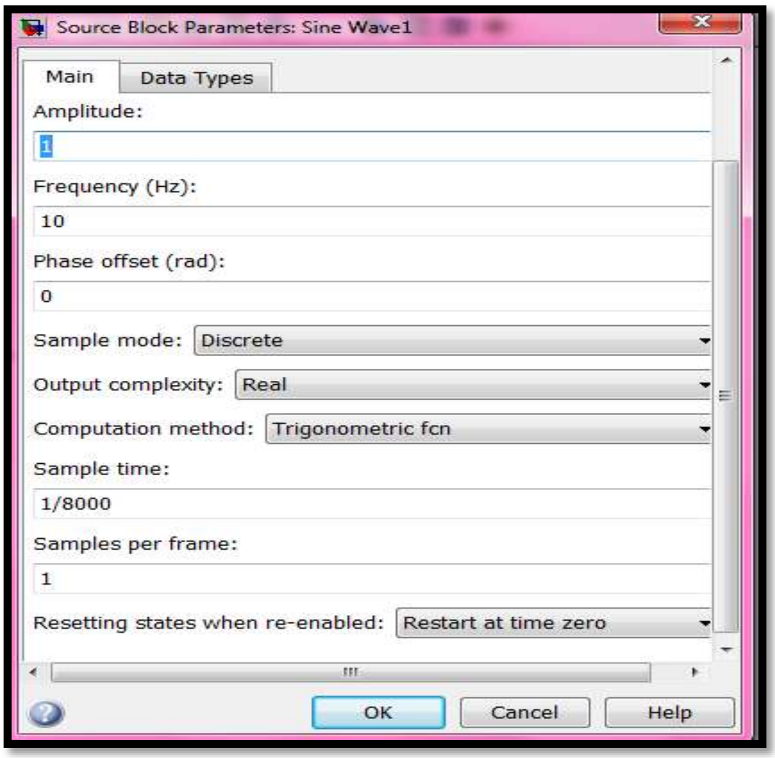
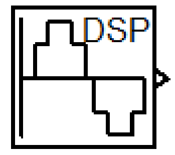


Fig 4.1.1sin wave circuit diagram



Amplitude: 1
Frequency (Hz): 10
Phase offset (rad): 0
Sample mode: Discrete
Output complexity: Real
Computation method: Trigonometric fcn
Sample time: 1/8000
Samples per frame: 1
Resetting states when re-enabled: Restart at time zero

Fig 4.1.2 sine wave parameters



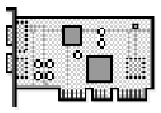
Sine Wave

The Sine Wave block generates a multichannel real or complex sinusoidal signal, with independent amplitude, frequency, and phase in each output channel. A real sinusoidal signal is generated when the Output complexity parameter is set to Real, and is defined by an expression of the type

$$y = Asin(2\pi f t + \emptyset)$$

where you specify A in the Amplitude parameter, f in hertz in the Frequency parameter, and ϕ in radians in the Phase offset parameter. A complex exponential signal is generated when the Output complexity parameter is set to Complex, and is defined by an expression of the type

$$y = Ae^{j(2\pi f t + \emptyset)} = A\{\cos(2\pi f t + \emptyset) + jsin(2\pi f t + \emptyset)\}$$



C6713DSK

Use this block to configure hardware settings and code generation features for your custom board. Include this block in models you use to generate Real-Time Workshop code to run on processors and boards. It does not connect to any other blocks, but stands alone to set the processor preferences for the model.

DAC

Adding the C6713 DSK DAC (digital-to-analog converter) block to your Simulink model lets you connect an analog signal to the analog output jack on the C6713 DSK. When you add the C6713 DSK DAC block, the digital signal received by the codec is converted to an analog signal and sent to the output jack.

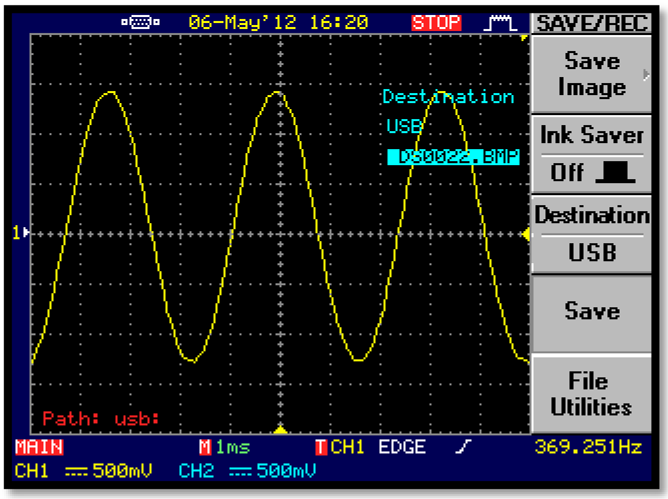


Fig 4.1.3 sine wave at oscilloscope

4.2 square wave generation :

The aim of this experiment to obtain a square wave according to Fourier series by talking the sum of odd harmonic . we sum 40 DSP sine wave together to have a square wave . every time we increase the number of odd harmonic the square wave going to be perfect , after applying on matlab we get this wave at scope Fig.4.2.1

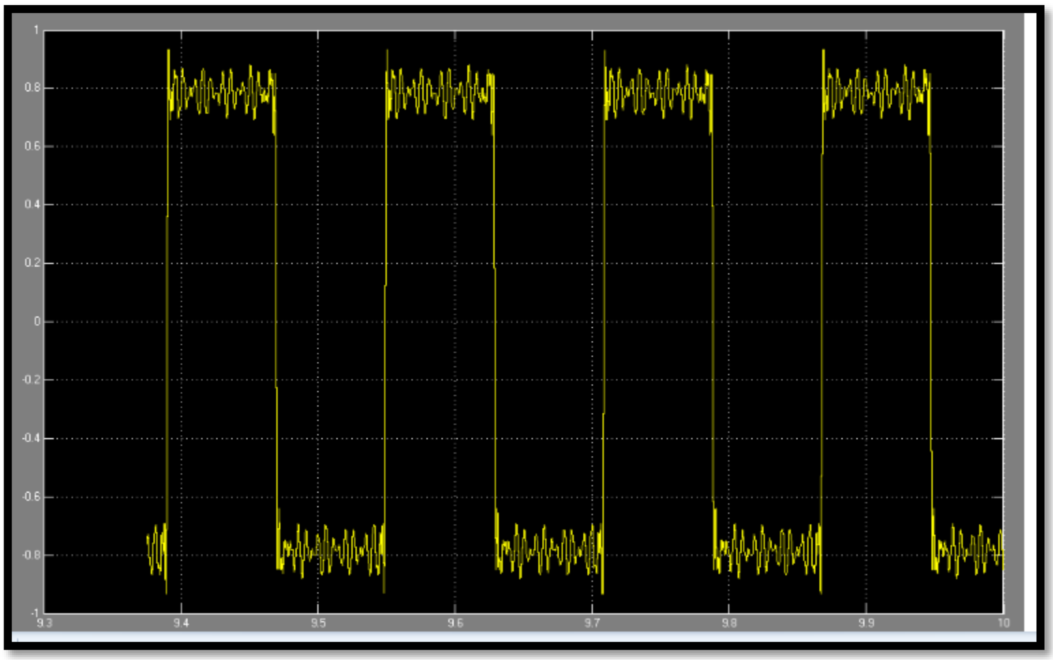


Fig.4.2.1 the generated square wave on matlab simulink

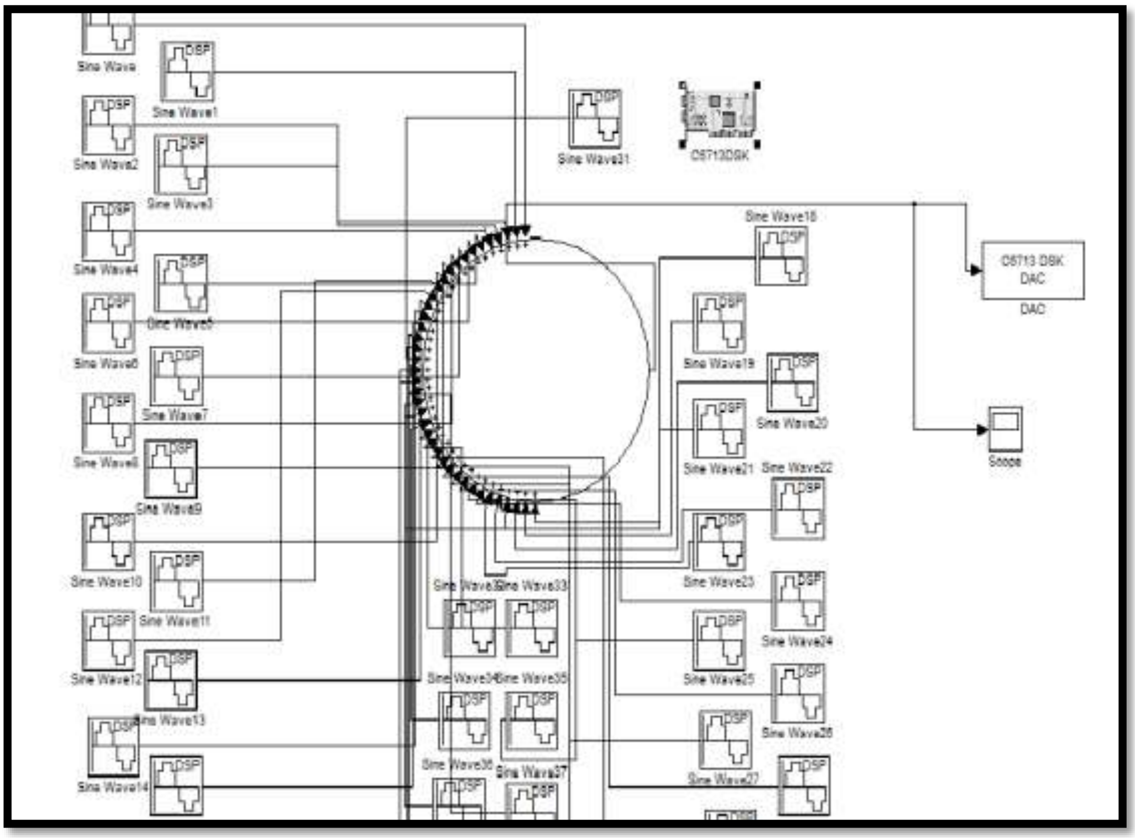
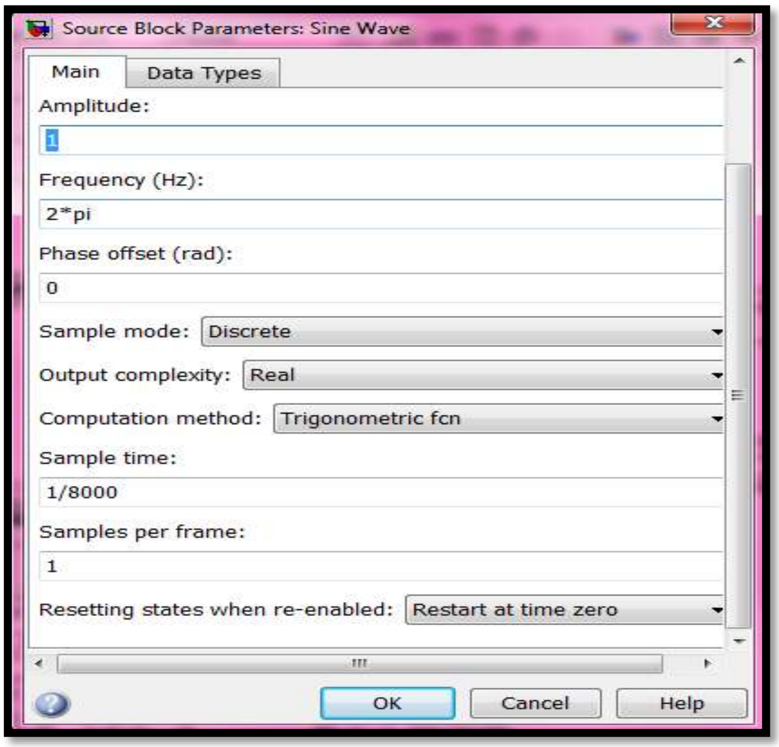


Fig.4.2.2 square wave circuit diagram


Source Block Parameters: Sine Wave

Main | Data Types

Amplitude:
1

Frequency (Hz):
2*pi

Phase offset (rad):
0

Sample mode: Discrete

Output complexity: Real

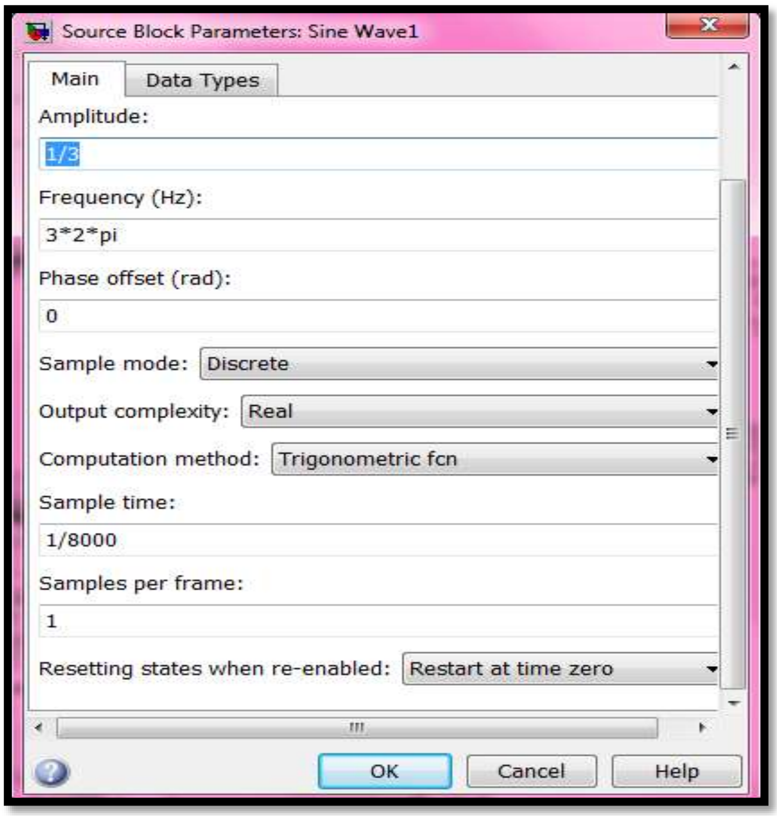
Computation method: Trigonometric fcn

Sample time:
1/8000

Samples per frame:
1

Resetting states when re-enabled: Restart at time zero

Fig.4.2.3 the first odd harmonic


Source Block Parameters: Sine Wave1

Main | Data Types

Amplitude:
1/3

Frequency (Hz):
3*2*pi

Phase offset (rad):
0

Sample mode: Discrete

Output complexity: Real

Computation method: Trigonometric fcn

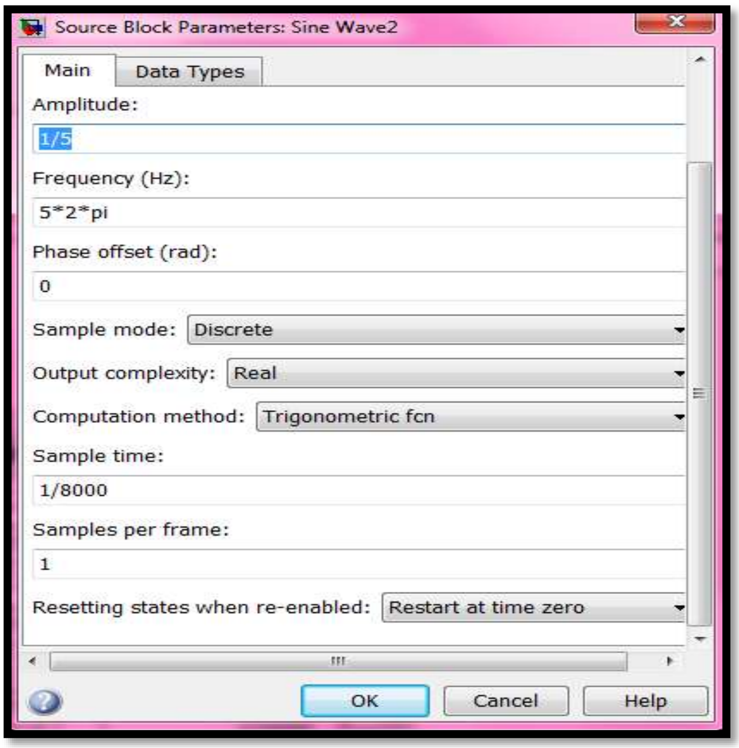
Sample time:
1/8000

Samples per frame:
1

Resetting states when re-enabled: Restart at time zero

OK Cancel Help

Fig.4.2.4 the second odd harmonic


Source Block Parameters: Sine Wave2

Main | Data Types

Amplitude:
1/5

Frequency (Hz):
5*2*pi

Phase offset (rad):
0

Sample mode: Discrete

Output complexity: Real

Computation method: Trigonometric fcn

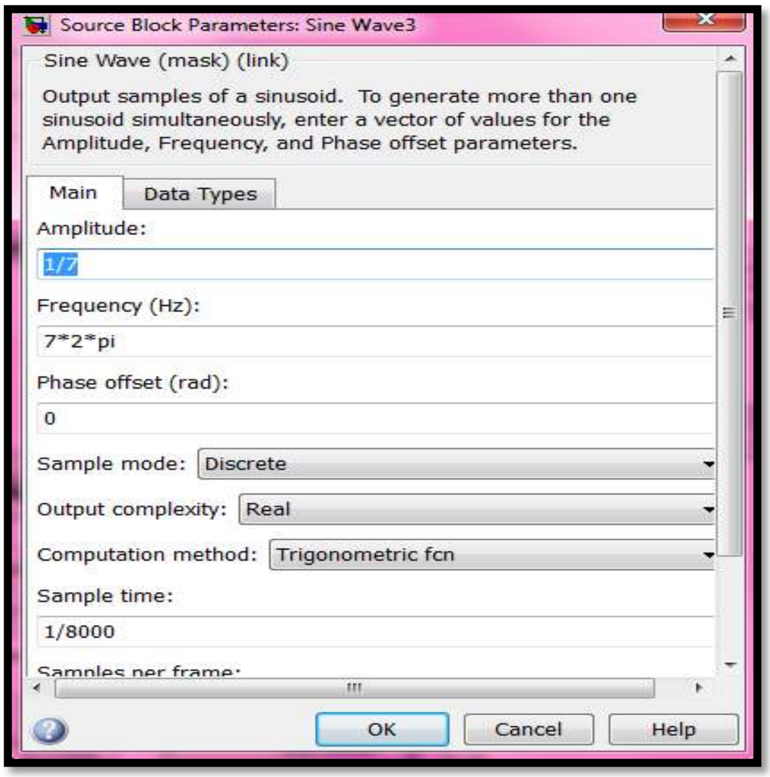
Sample time:
1/8000

Samples per frame:
1

Resetting states when re-enabled: Restart at time zero

OK Cancel Help

Fig.4.2.5 the third odd harmonic


Source Block Parameters: Sine Wave3

Sine Wave (mask) (link)

Output samples of a sinusoid. To generate more than one sinusoid simultaneously, enter a vector of values for the Amplitude, Frequency, and Phase offset parameters.

Amplitude:
1/7

Frequency (Hz):
7*2*pi

Phase offset (rad):
0

Sample mode: Discrete

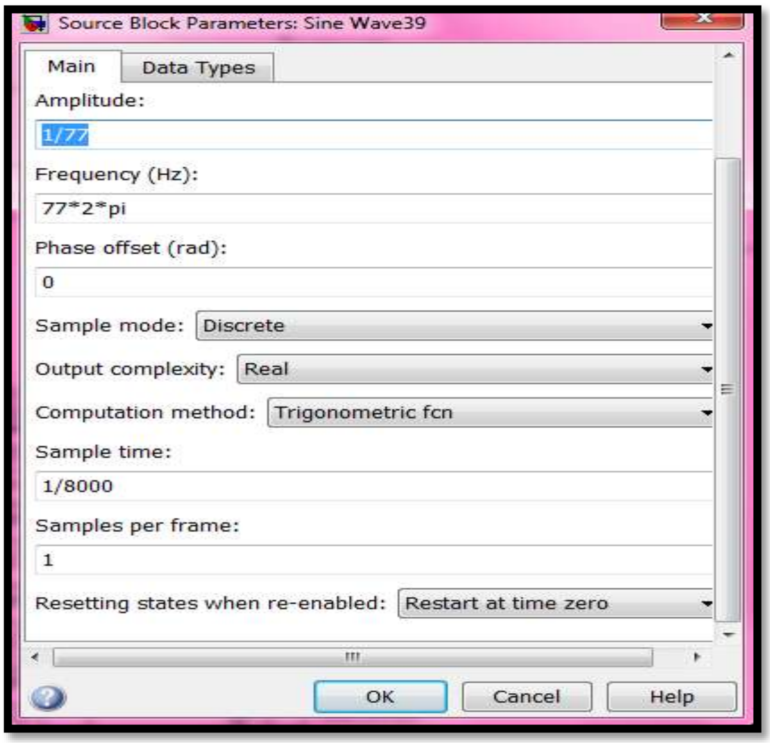
Output complexity: Real

Computation method: Trigonometric fcn

Sample time:
1/8000

Samples per frame:

Fig.4.2.6 the fourth odd harmonic


Source Block Parameters: Sine Wave39

Amplitude:
1/77

Frequency (Hz):
77*2*pi

Phase offset (rad):
0

Sample mode: Discrete

Output complexity: Real

Computation method: Trigonometric fcn

Sample time:
1/8000

Samples per frame:
1

Resetting states when re-enabled: Restart at time zero

Fig.4.2.7 the 40th odd harmonic

4.3Sampling :

Samples are successive snapshots of a signal. In the case of audio, the signal is a sound wave. A microphone converts the acoustic signal into a corresponding analog electric signal, and an analog-to-digital converter transforms that analog signal into a sampled digital form. The accuracy of the digital approximation of the analog signal depends on its resolution in time (the sampling rate) and its quantization, or resolution in amplitude(the number of bits used to represent each sample).

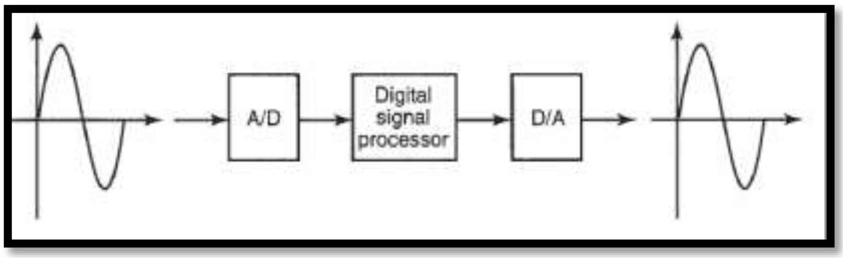


Fig 4.3.1 The signal sampling and reconstruction .

ADC is doing the sampling process with different fixed sampling rates which are:

1. 8KHz .
2. 32 KHz .
3. 44.1 KHz .
4. 48 KHz .
5. 96 KHz .

DAC is doing the reconstruction process with different fixed sampling rates the same as ADC rates .

Results :

We use an audio signal as input signal ,as shown in figure 4.3.2

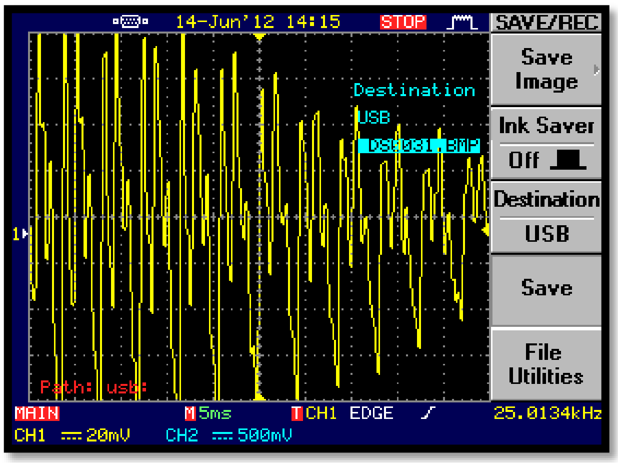


Fig 4.3.2 the input audio signal

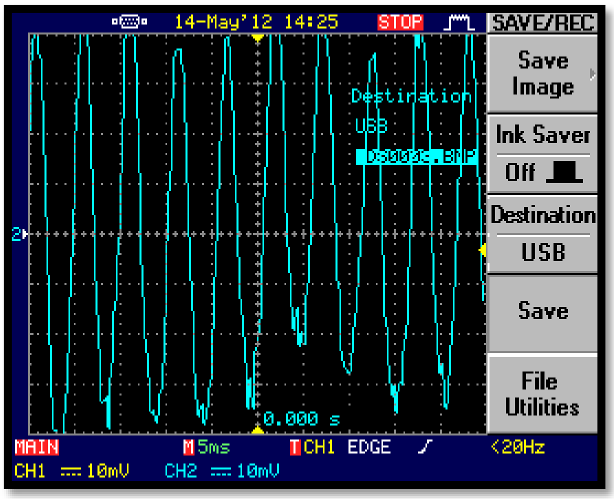


Fig 4.3.3 the output audio signal with sampling rate 8KHz .

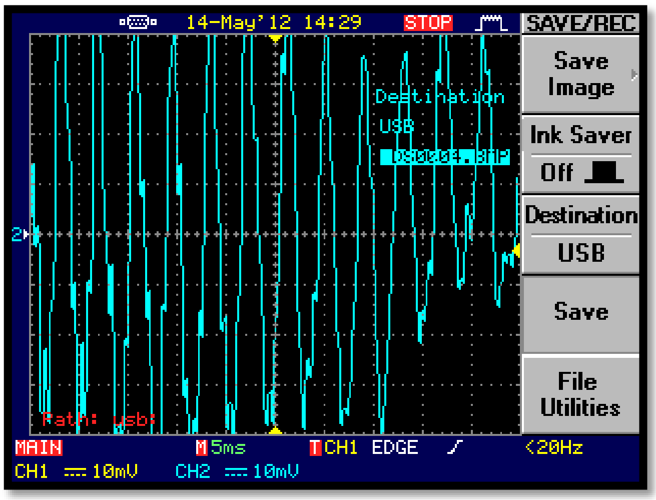
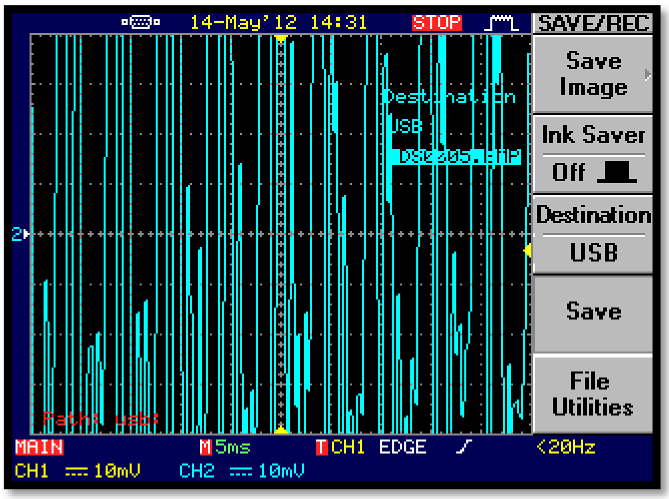
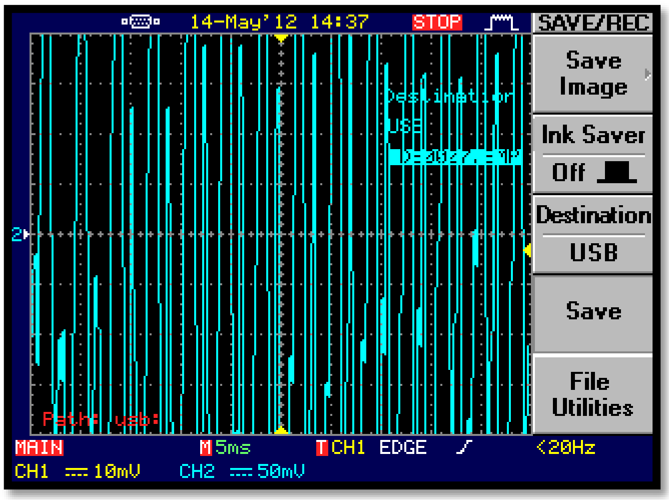


Fig 4.3.4 the output audio signal with sampling rate 32KHz .



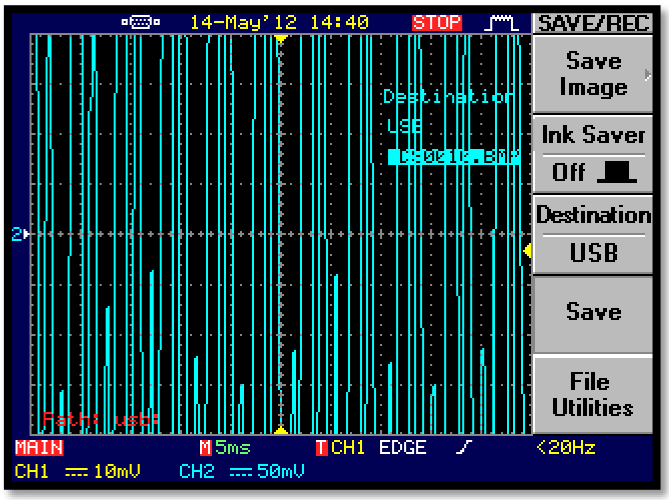
MAIN M 5ms T CH1 EDGE <20Hz
CH1 10mV CH2 10mV

Fig 4.3.5 the output audio signal with sampling rate 44.1KHz .



MAIN M 5ms T CH1 EDGE <20Hz
CH1 10mV CH2 50mV

Fig 4.3.6 the output audio signal with sampling rate 48 KHz .



MAIN M 5ms T CH1 EDGE <20Hz
CH1 10mV CH2 50mV

Fig4.3.7 the output audio signal with sampling rate 96 KHz .

In this experiment we use audio signal as input ,we apply sampling using different sampling rates. From previous figures fig.4.3.2-6 as we see every time we increase the sampling rate for the same audio signal ,the number of cycle per second is increase .Not that the signal shape is not regular because we use audio input signal.

4.4 IIR filter :

4.4.1 Butterworth:

Butterworth is a type of IIR ,we build here a circuit as Fig4.4.1 to study the characteristics of Butterworth filter, so we need an input signal with a different frequency component like the "chirp signal: sine wave whose frequency varies linearly with time", choosing F= 70 Hz is acceptable. In order to be able to implement the filter characteristics on chirp signal we must treat it ,so we change the signal sampling rate to make the filter accommodate it , by using the "rate transition: Handle transfer of data between ports operating at different rates", with time rate 1/8000 second, after that a digital filter design tool used, we build an IIR Butterworth high pass filter works at $f_s = 8000Hz$ and $f_{stop} = 20Hz, f_{pass} = 30Hz$ removes the undesired low frequency components, that appears on the scope! Fig4.4.4

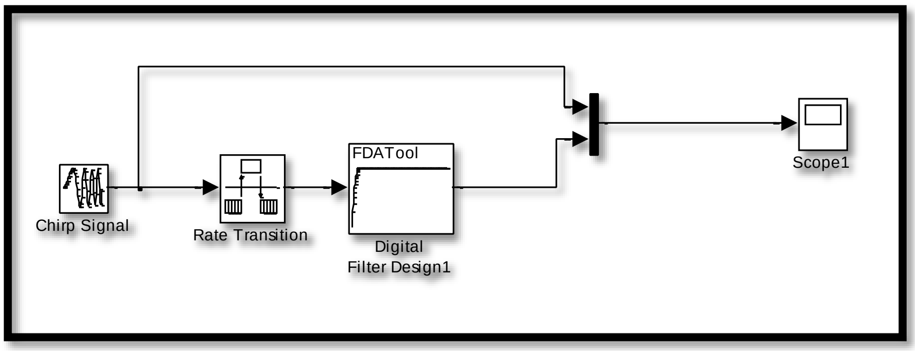


Fig4.4.1 Circuit diagram

We see in Fig4.4.2 the magnitude and phase response of Butterworth filter, it is pretty clear that the frequency response is maximally flat "with no ripples" .

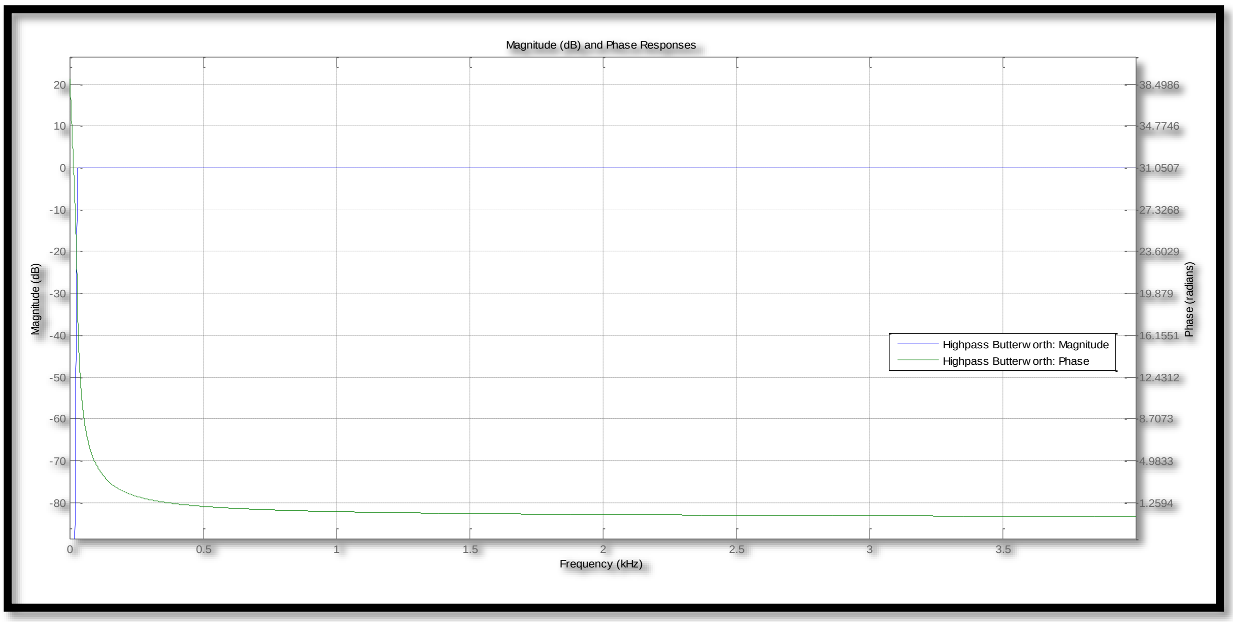


Fig4.4.2 Magnitude phase response

Note from the pole-zero location on the z plane that all poles are in the unit circle (the filter is stable here) ,the number of the poles determines the order of the filter; the more zeros the higher filter order

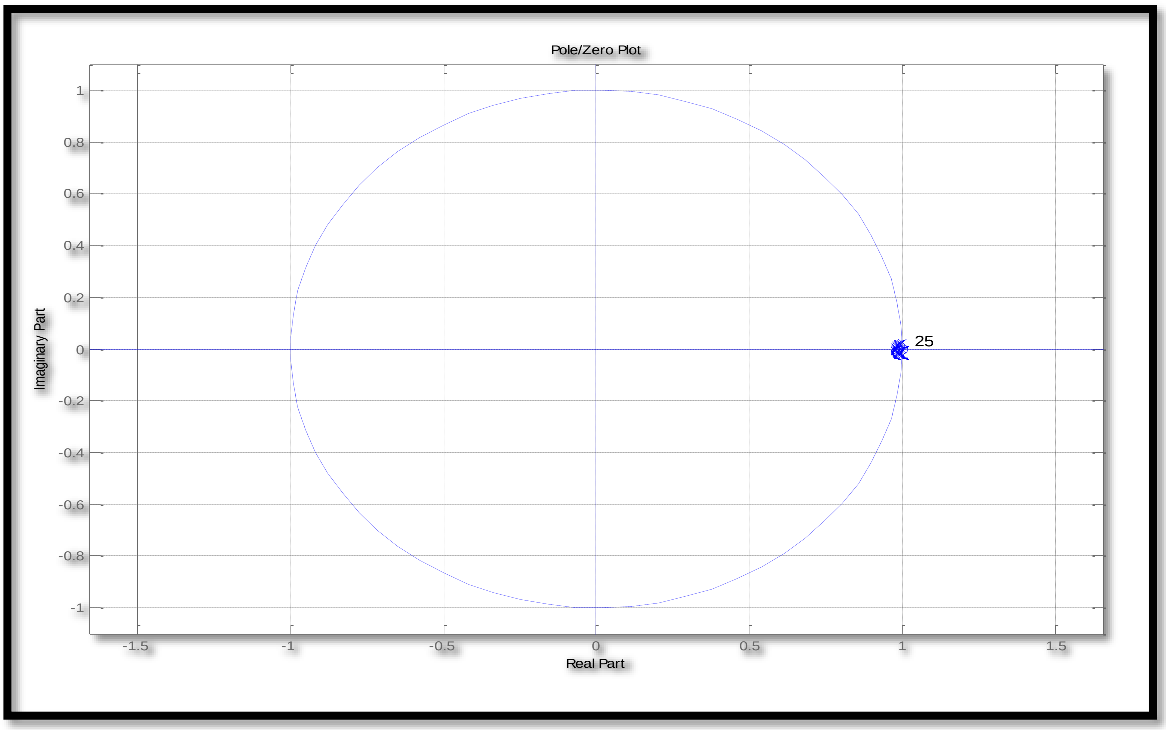


Fig4..4.3 Pole zero plot

Its clear here in the Fig4.4.3 the effect of the filter ,the purple signal is the signal after filtering , and the yellow is the original chirp signal, the filter removes the un desired low frequency components. It is very obvious the flat response .

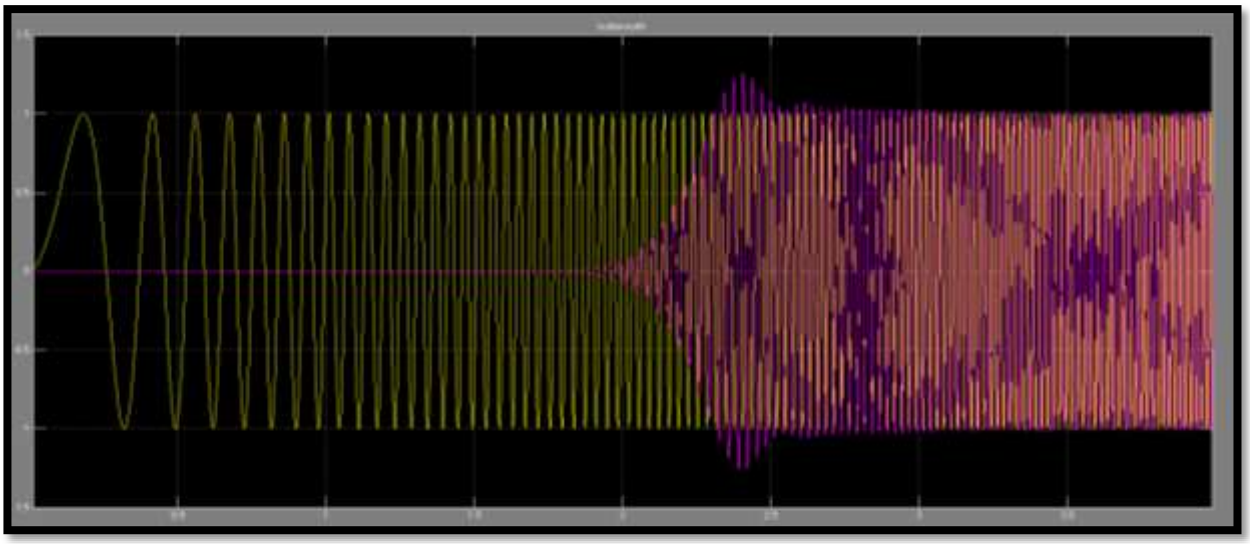


Fig4.4.4 The chirp signal before (yellow) and after(purple) the Butterworth HPF

Appling the experiment on the DSK6713:

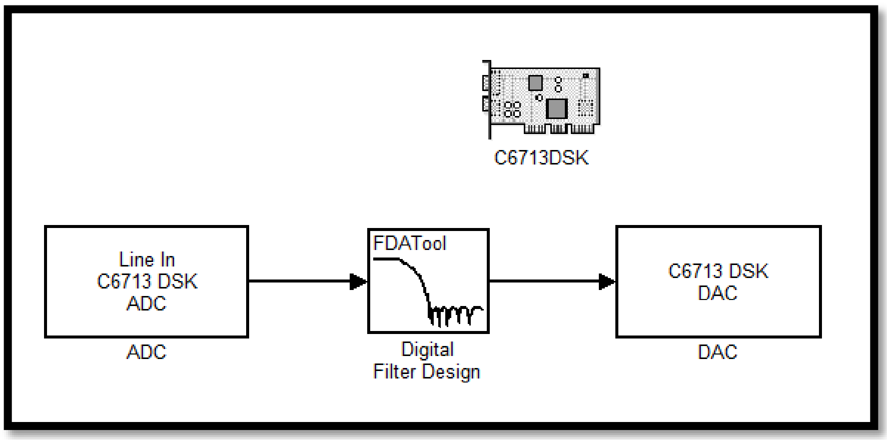


Fig4.4.5 HPF circuit on dsk

This time we add the DSK7613 symbol , that's mean this block diagram will download on the KIT in association with code composer studio program ,also we add the main parameters :ADC and DAC . Here we apply the HPF on a human voice signal so we will use a mike to inter the signal , and we must change the input signal type at ADC to "Mike In" instead of "Line in", then put a tic at the +20dB to amplify the voice signal!

At first we equate the time rate at all parts to 1/8000 sec "we can change the rate as we see in Sampling", but here 1/8000 is satisfied for a human voice. The main objective here is to eliminate the low frequency component and just take the high ,the main parameter here according to human voice (30-3000Hz) : $f_s = 8000Hz$,$f_{stop} = 1500Hz$,$f_{pass} = 1800Hz$, after pushing the design button the signal is filtered then go to ADC ,the signal processed at the same rate in order to converted it back to analog again , finally we can hear the filtered voice through a headphone or see its wave form on oscilloscope fig4.4.6

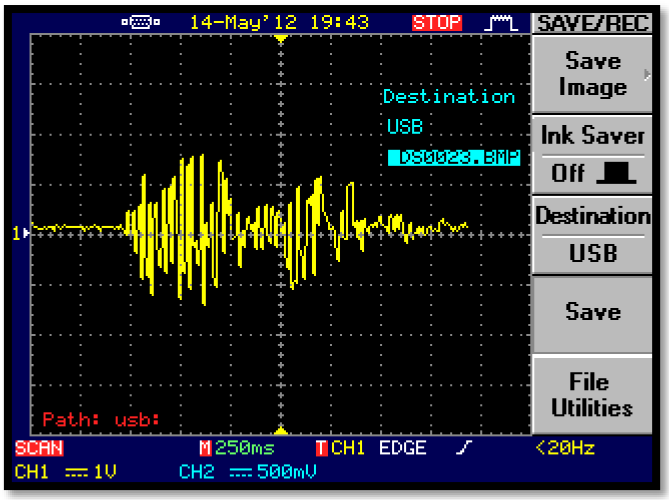
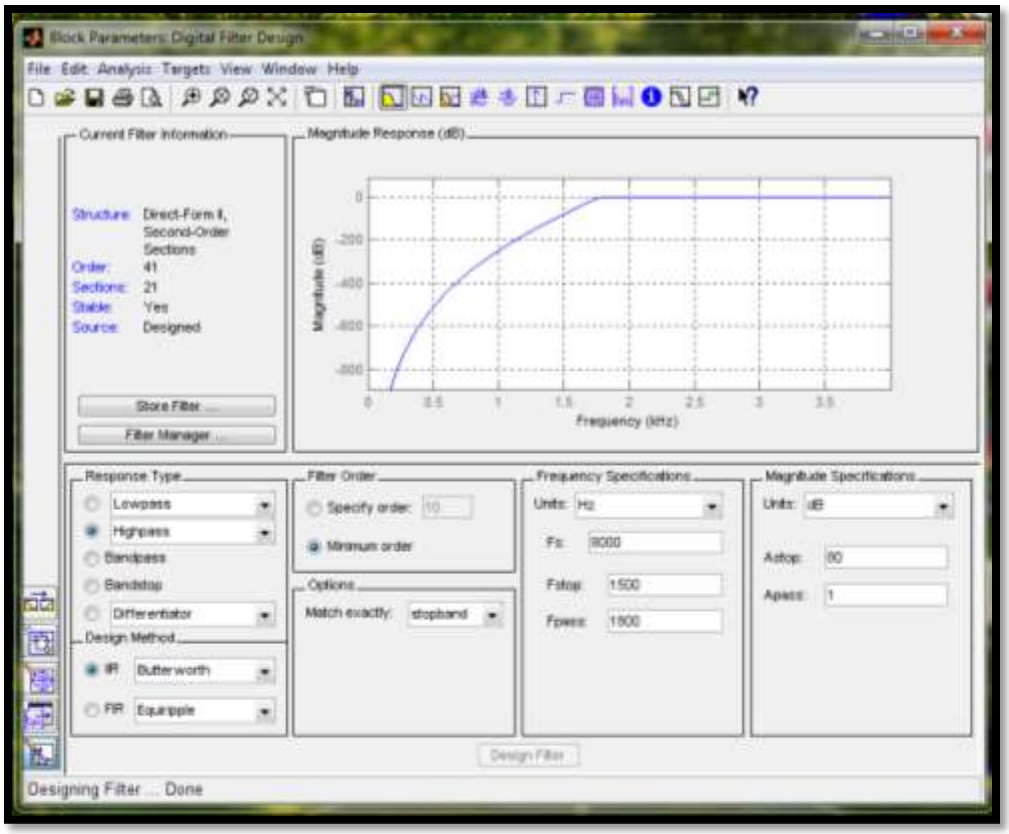


Fig4.4.6 the whistling signal on oscilloscope pass after Butterworth HPF



Current Filter Information

Structure: Direct-Form II, Second-Order Sections
Order: 41
Sections: 21
Stable: Yes
Source: Designed

Response Type: Lowpass [✓ Highpass] Bandpass Bandstop Differentiator

Design Method: [✓ IIR Butterworth] FIR Equiripple

Filter Order: Specify order: 10 [✓ Minimum order]

Options: Match exactly: stopband

Frequency Specifications: Units: Hz; Fs: 8000; Fstop: 1500; Fpass: 1800

Magnitude Specifications: Units: dB; Astop: 80; Apass: 1

Designing Filter ... Done

Fig4.4.7 the parameter of Butterworth

In this experiment the HPF pass' just the high frequency components so, after applying it by the incremental build ,we enter a human voice with different frequency component through the mike, it is noticeable that the low frequency (ordinary talk) blocked and won't appear at the oscilloscope , but if we whistling the signal will pass through the filter and appear, Simply because the high frequency ! Fig4.4.7.

Conclusion:

- The response of Butterworth filter is flat.
- The time response is large from the stop to pass frequency.
- The phase change at the first then it's back again exponentially

4.4.2 Chebychev I

Another type of IIR filters is Chebychev I ,this type of filters have the property that they minimize the error between the idealized and the actual filter characteristic over the range of the filter, but with ripples in the passband. Also it has a sharper edge than the Butterworth filter . we build a circuit as Fig4.4.8 chrip signal at the input treated by a rate transition with sample rate 8000 to be able to accommodate it by the filter ,and we use the filter design tool using sampling frequency $f_s = 8000Hz,$ f $f_{stop} = 20Hz, f_{pass} = 30Hz,$ after the filter we take the output and the input into a mux in order to compare the two signals .

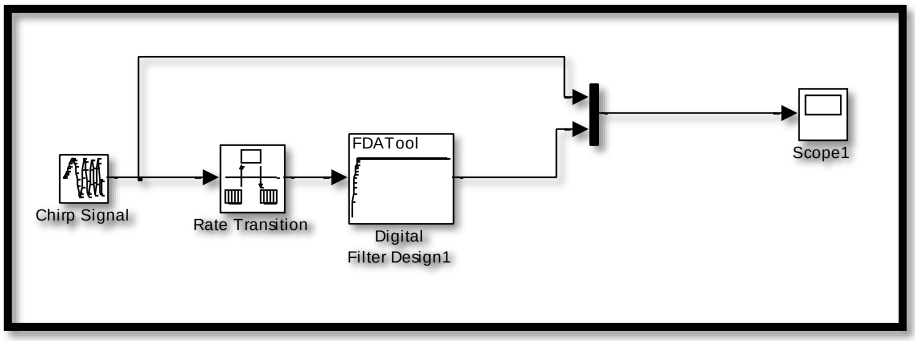


Fig4.4.8 Chebychev I circuit diagram

This is the magnitude and phase response of chebychev I filter, there is some ripples at the passband, and we note the small transient time between the stop and pass frequency. The phase at the first is high then it is decrease exponentially!

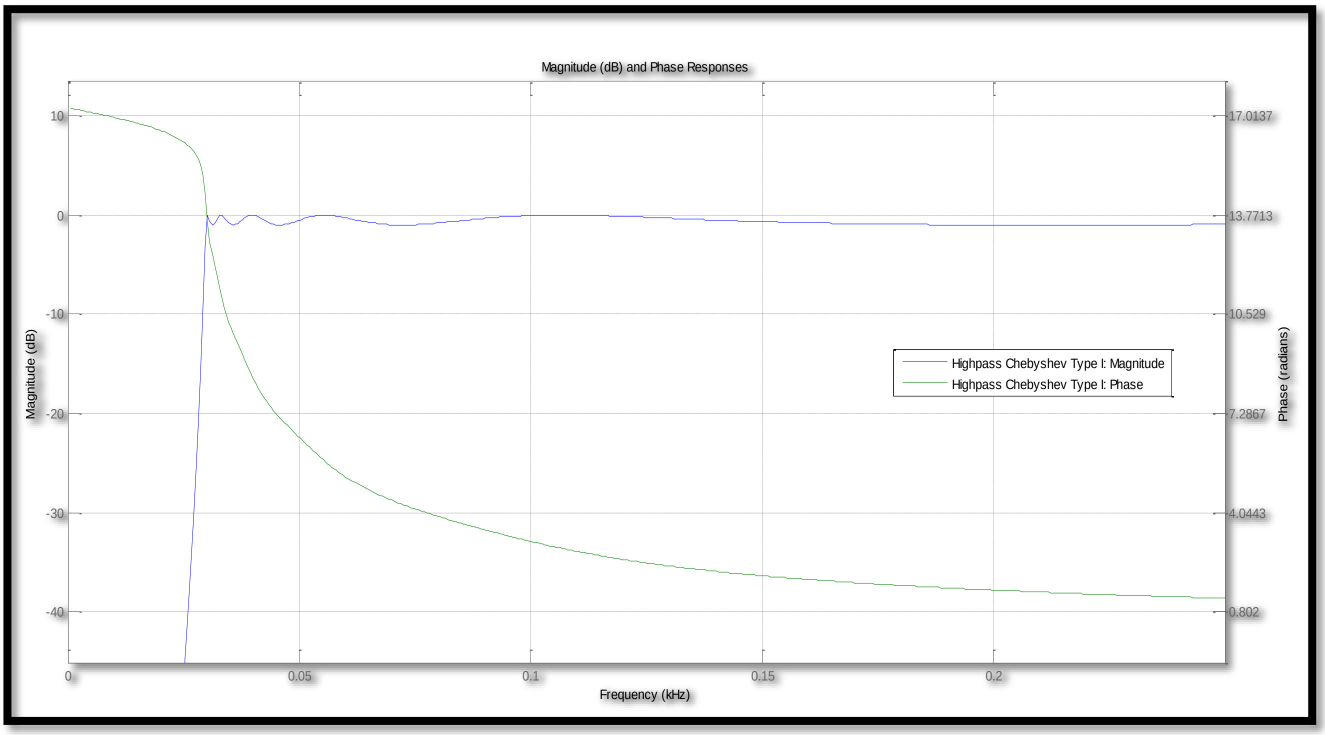


Fig 4.4.9Magnitude and respose of the chebychev I

The filter order is known from the number of poles of the system, so we have 11 pole at 1 so a filter with 11th order, the system is stable .

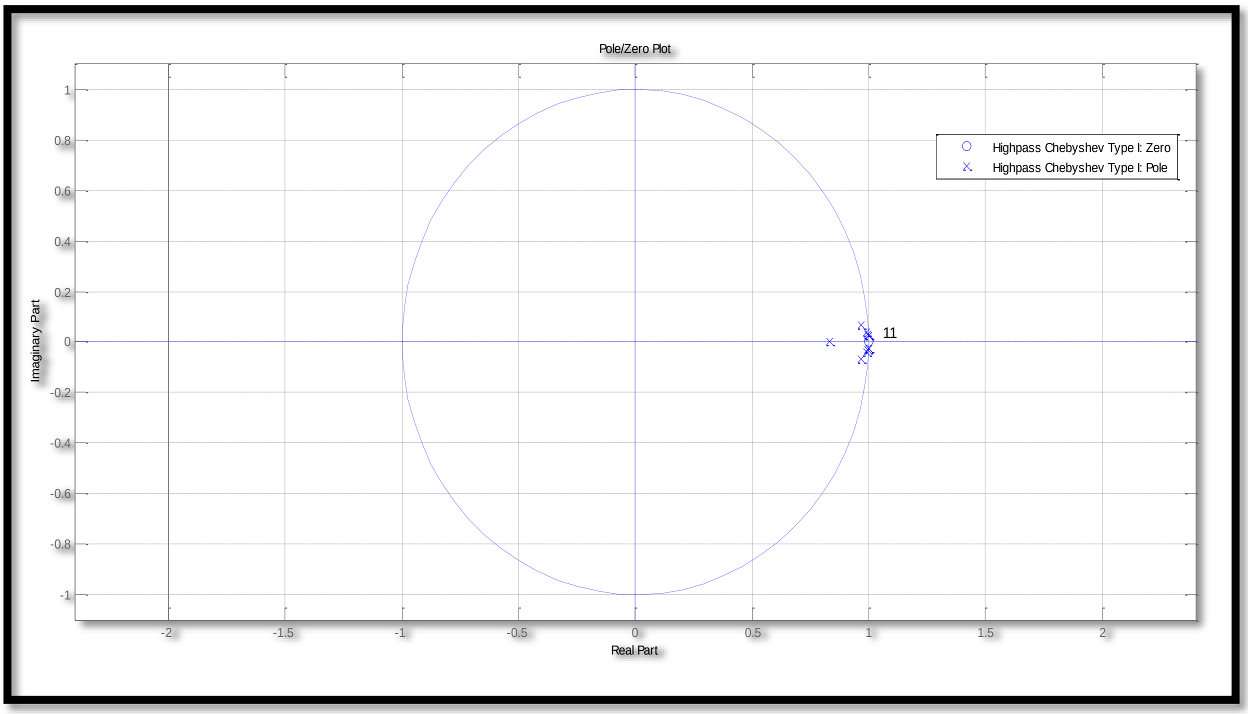


Fig4.4.10 Pole/zero plote

Chebychev filters always have ripples effect on the passband in fig4.4.11 , as we see before the low frequency attenuated.

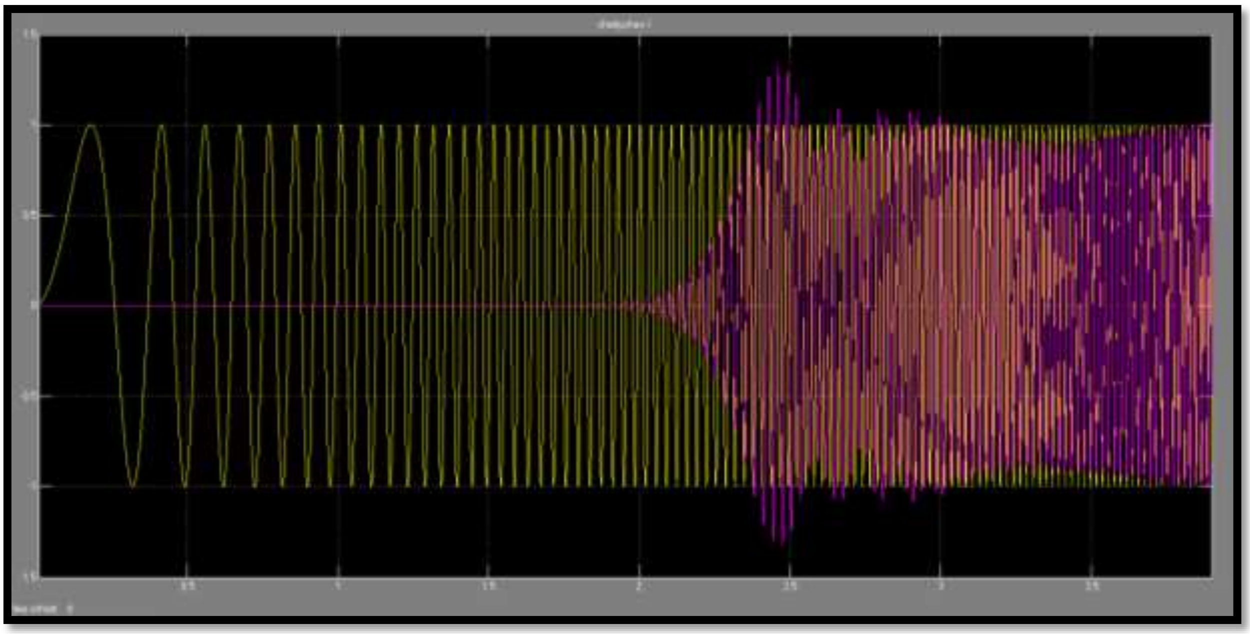


Fig4.4.11 The signal before (yellow) and after(purple) the Chebychev I HPF

Applying the experiment on DSK7613 :

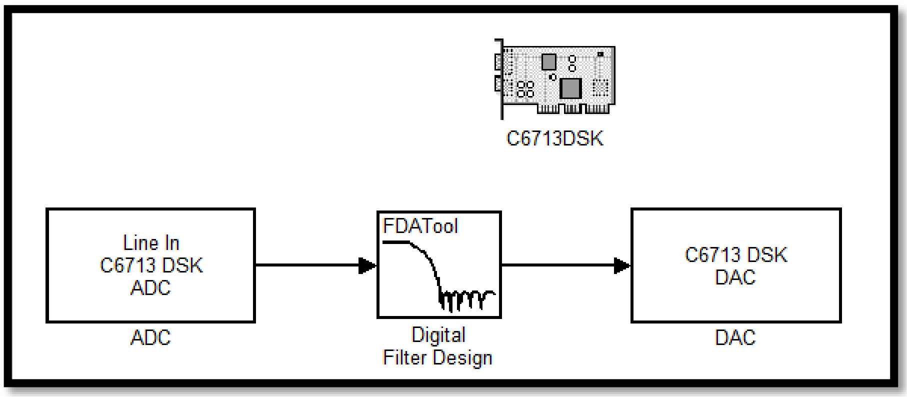
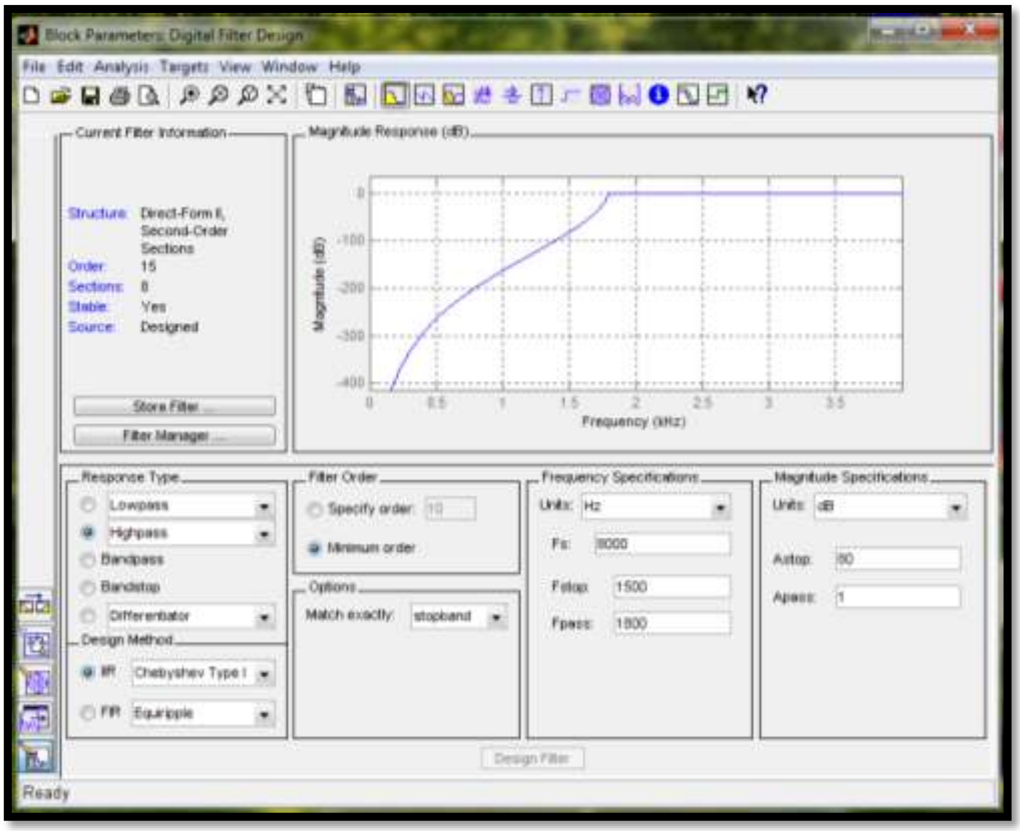


Fig4.4.12 cheybchve circuit on DSK

After making this circuit in fig4.4.12 as we did before, and using the same HPF parameter $f_s = 8000Hz, f_{stop} = 1500Hz, f_{pass} = 1800Hz$, but this time we use chebychev I , that mean we need to hear the voice with high frequency so, when you talk nothing will happen , just whistle or scream!

After downloading this system on the Kit we get an IIR chebychev I HPF portable ,due to inner memory, now make sure the time rate at all parts is the same , finally just chose the type of input and output to "mike in" and "headphone" respectively.


Block Parameters: Digital Filter Design
File Edit Analysis Targets View Window Help
Current Filter Information
Magnitude Response (dB)
Magnitude (dB)
Frequency (kHz)
Store Filter ...
Filter Manager ...
Design Filter
Ready

Structure: Direct-Form II, Second-Order Sections
Order: 15
Sections: 8
Stable: Yes
Source: Designed

Response Type: Lowpass [✓ Highpass] Bandpass Bandstop Differentiator
Design Method: [✓ IIR Chebyshev Type I] FIR Equiripple
Filter Order: Specify order: 10 [✓ Minimum order]
Options: Match exactly: stopband
Frequency Specifications: Units: Hz; Fs: 8000; Fstop: 1500; Fpass: 1800
Magnitude Specifications: Units: dB; Astop: 80; Apass: 1

Fig4.4.13 the parameters of chebychev I

The rapped and not flat response is clear at the oscilloscope shot.

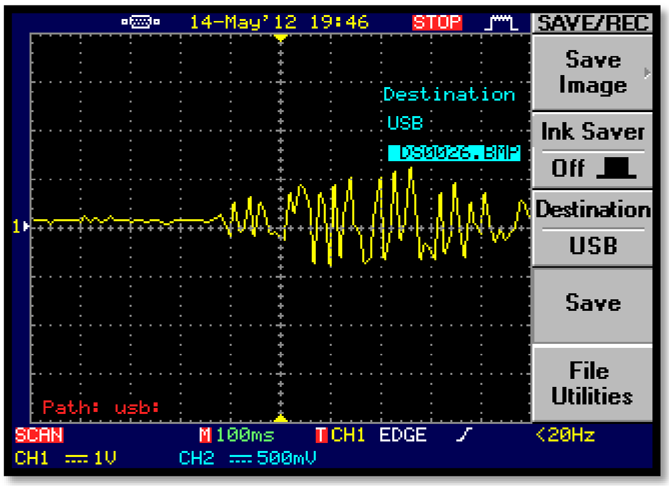


Fig4.4.14 the whistling signal on oscilloscope pass after chebychev I HPF

Conclusion:

- The Chebyshev Type I filter minimizes the absolute difference between the ideal and actual frequency response over the entire passband by incorporating an equal ripple in the passband.
- Stopband response is maximally flat.
- The transition from passband to stopband is more rapid than for the Butterworth filter.

4.4.3 Chebychev II

Type II is better than type I in pass band part , It is more flat, also the transient time between two bands is smaller, in other hand there is ripples in the stopband. The same circuit built again , with changing to chebychev type II, and parameter will be, $f_s = 1000Hz$, $f_{stop} = 50Hz, f_{pass} = 60Hz$ and $f_{chirp} = 100Hz$, the input chirp signal go into rate transition with time rate =1/1000 sec , then to the filter then to the scope with the original signal through mux.

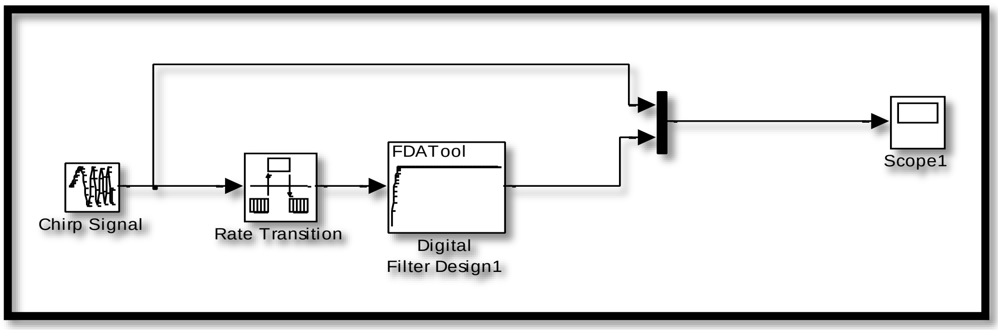
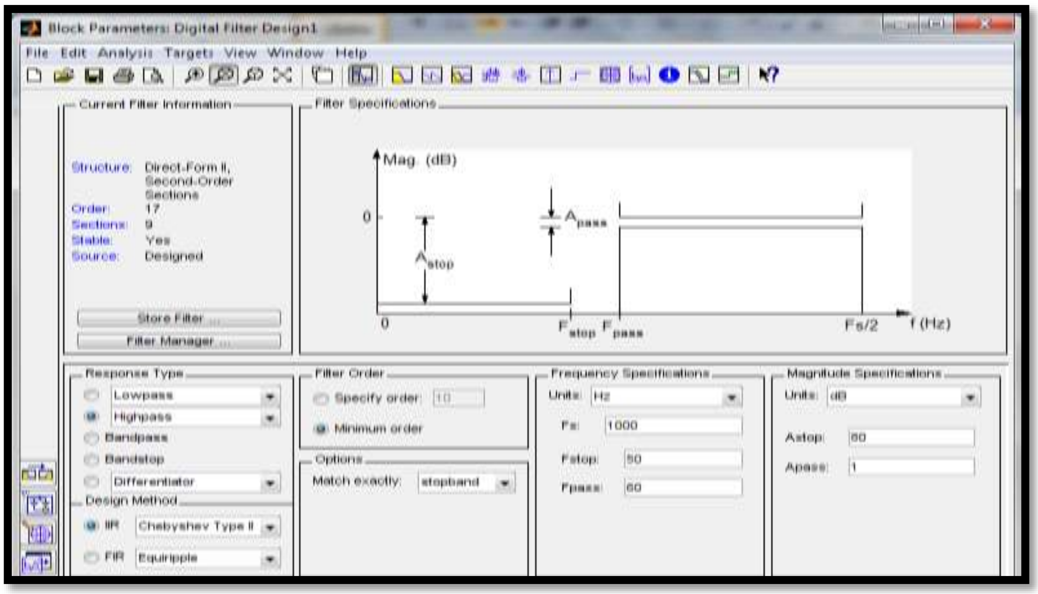


Fig4.4.15 circuit diagram for chebychev II


Block Parameters: Digital Filter Design1

Current Filter Information

Structure: Direct-Form II, Second-Order Sections
Order: 17
Sections: 9
Stable: Yes
Source: Designed

Response Type: ( ) Lowpass [✓ Highpass] ( ) Bandpass ( ) Bandstop ( ) Differentiator

Design Method: [✓ IIR Chebyshev Type II] ( ) FIR Equiripple

Filter Order: ( ) Specify order: 10 [✓ Minimum order]

Options: Match exactly: stopband

Frequency Specifications: Units: Hz; Fs: 1000; Fstop: 50; Fpass: 60

Magnitude Specifications: Units: dB; Astop: 80; Apass: 1

Fig4.4.16 parameter of chebychev II

The Chebyshev response has the fastest rate of phase change ,and the pass band area is flat, but the ripples located at the stopband region, as we see in fig4.4.16

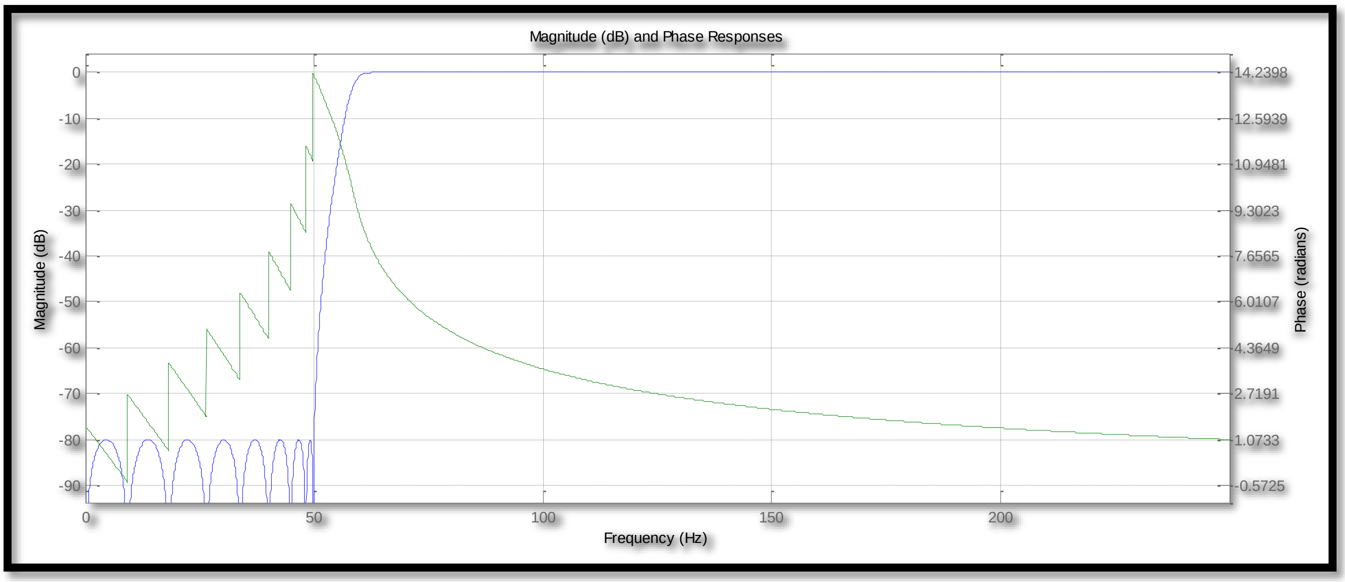


Fig4.4.17 the magnitude and phase response of chebychev II

The filter order is 17th, certainly the number of poles is 17.

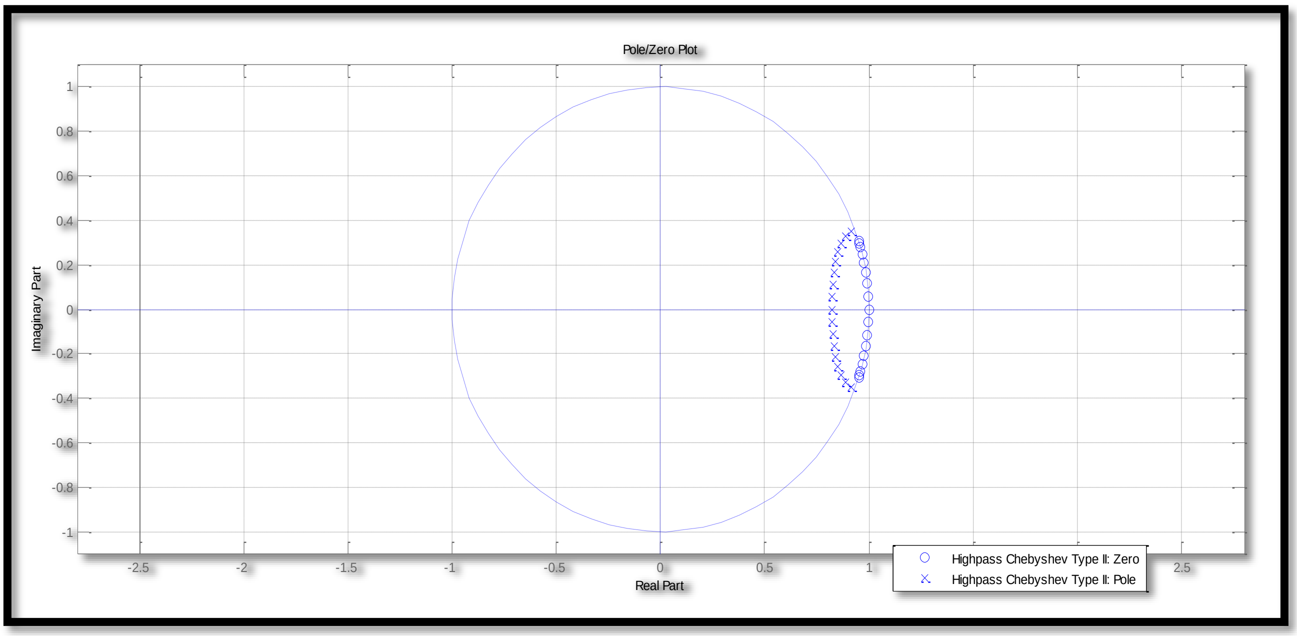


Fig4.4.18 the pole zero plot of chebychev II

As we said before The magnitude response is so flat (it is clear) from the scope fig4.4.18

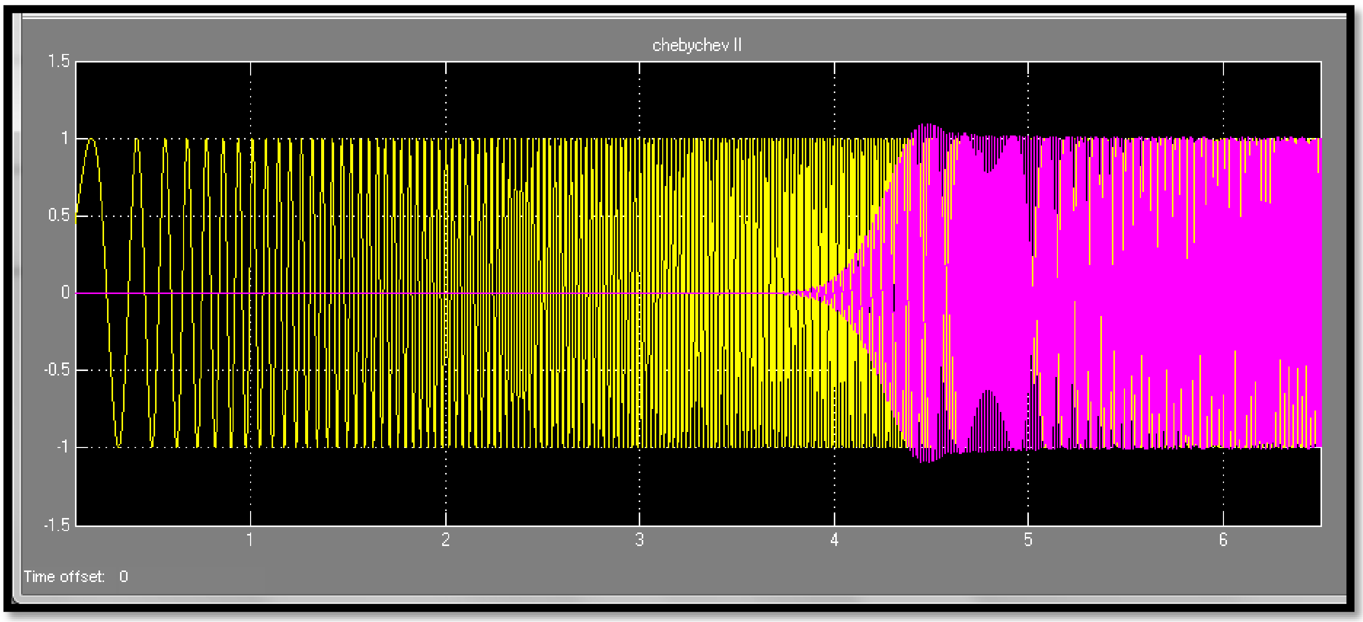


Fig4.4.19 the magnitude response of chebychev II from scope.

Applying the experiment on the DSK7613:

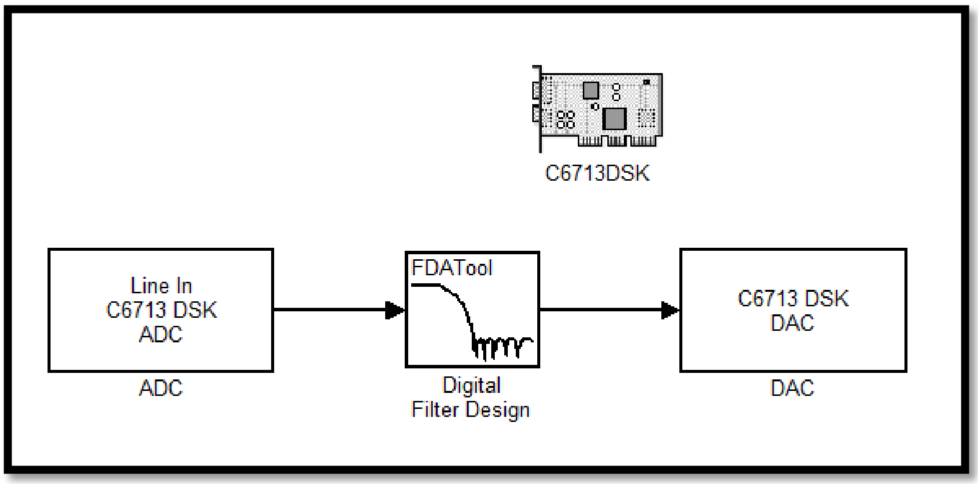
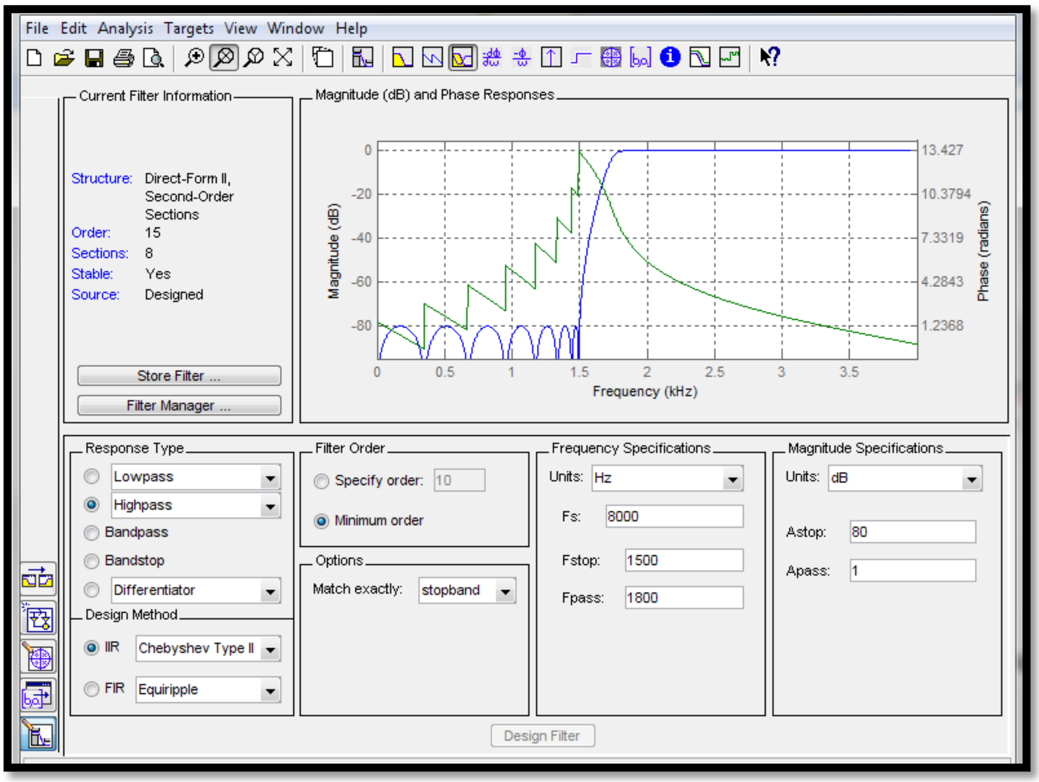


Fig4.4.20 chebychev II circuit on Kit

At new simulink page we build this circuit to study the effects of chebychev II on a voice signal , so the sampling rate is 1/8000 for ADC, DAC and digital filter ,thus $f_s = 8000Hz$, $f_{stop} = 1500Hz$, $f_{pass} = 1800Hz$,put IIR filter on chebychev type two ,then start and down load…

As expected , any low frequency is cut off , just high will pass!



Current Filter Information

Structure:	Direct-Form II, Second-Order Sections
Order:	15
Sections:	8
Stable:	Yes
Source:	Designed

Response Type: Lowpass [✓ Highpass] Bandpass Bandstop Differentiator

Design Method: [✓ IIR Chebyshev Type II] FIR Equiripple

Filter Order: Specify order: 10 [✓ Minimum order]

Options: Match exactly: stopband

Frequency Specifications: Units: Hz; Fs: 8000; Fstop: 1500; Fpass: 1800

Magnitude Specifications: Units: dB; Astop: 80; Apass: 1

Fig4.4.21 the parameter of chebychev II

Because of stopband ripples the first component is amplified more than the rest of component, and the passband approximately flat.

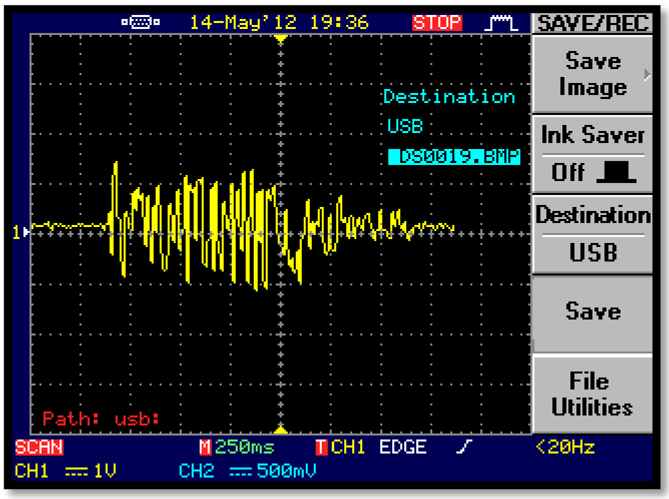


Fig4.4.22 the whistling signal on oscilloscope pass after chebychev II HPF

Conclusion:

- Passband response is maximally flat.
- he Chebyshev Type II filter minimizes the absolute difference between the ideal and actual frequency response over the entire stopband by incorporating an equal ripple in the stopband.
- The stopband does not approach zero as quickly as the type I filter.

4.4.4 Elliptic

The last kind of IIR filters is elliptic filter, it is a combination between the previous types , with equiripple in both the passband and stopband, we connect the circuit as shown in the fig. we set the parameters : $f_s = 1000Hz$, $f_{stop} = 50Hz$, $f_{pass} = 60Hz$, $f_{chirp} = 100Hz$ and the sampling time is 1/1000 sec.

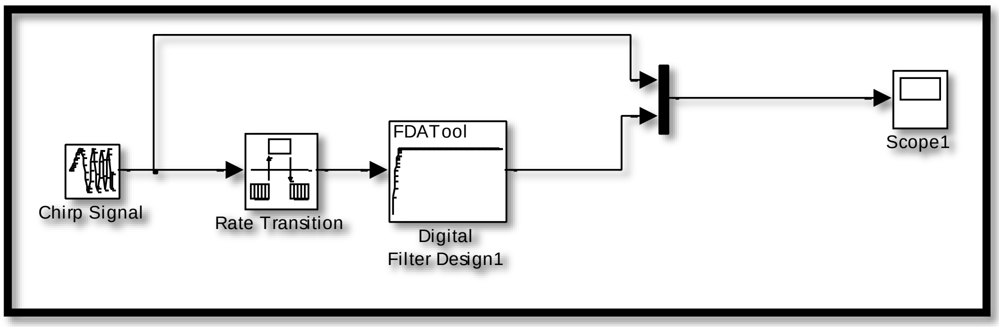
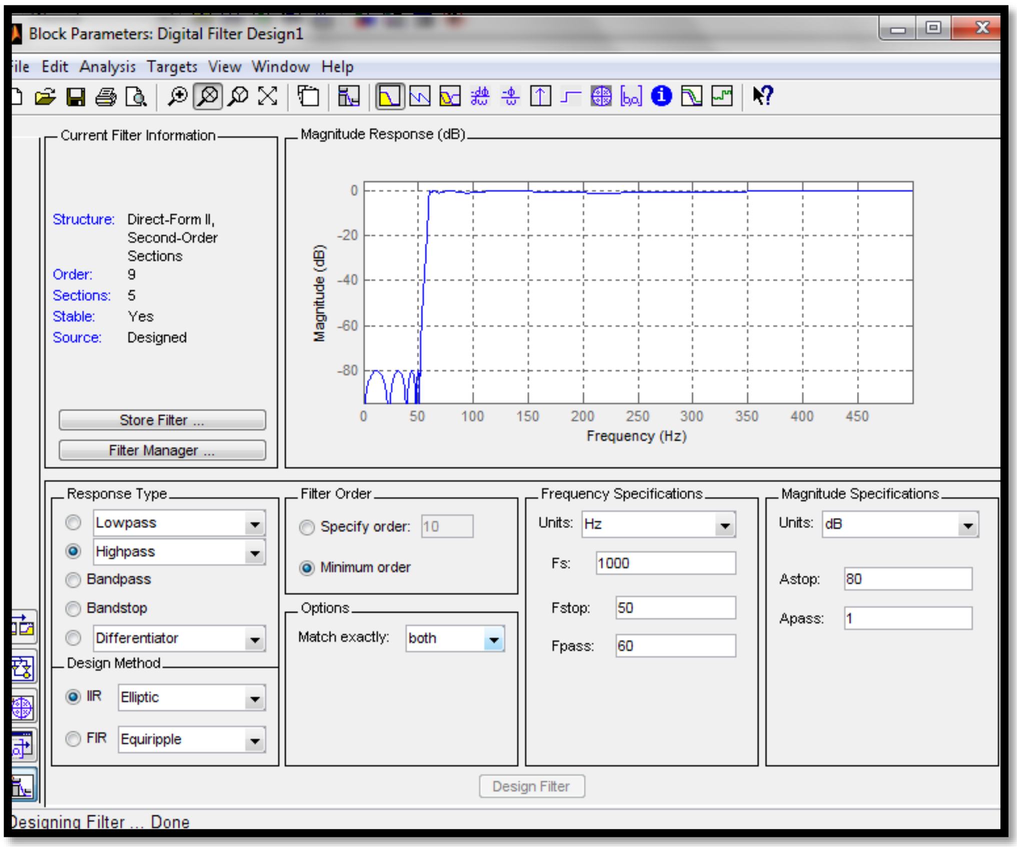


Fig4.4.23 elliptic filter circuit



Current Filter Information

Structure: Direct-Form II, Second-Order Sections
Order: 9
Sections: 5
Stable: Yes
Source: Designed

Response Type
- Lowpass
- [✓ Highpass]
- Bandpass
- Bandstop
- Differentiator

Design Method
- [✓ IIR] Elliptic
- FIR Equiripple

Filter Order
- Specify order: 10
- [✓ Minimum order]

Options
Match exactly: both

Frequency Specifications
Units: Hz
Fs: 1000
Fstop: 50
Fpass: 60

Magnitude Specifications
Units: dB
Astop: 80
Apass: 1

Designing Filter ... Done

Fig4.4.24 the parameter of elliptic filter

Elliptic IIR filter has ripples at the passband and stopeband , they could be equal or not "as we have here" , the phase change gradually first stage , then fall exponentially .see fig.4.4.24

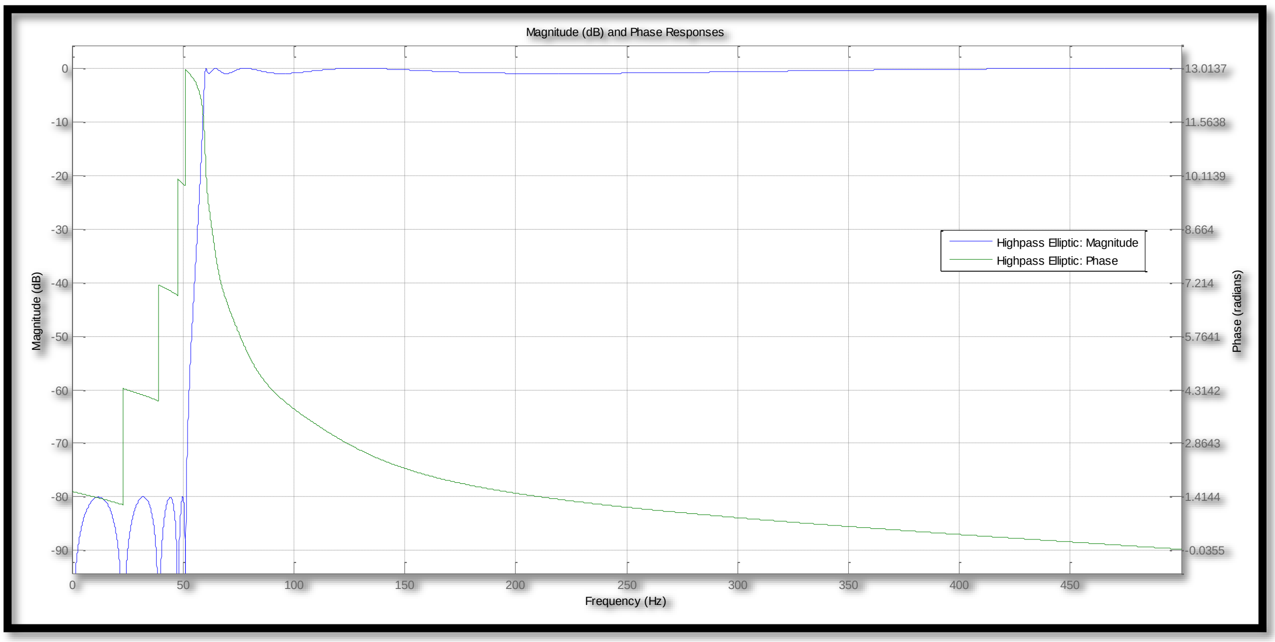


Fig4.4.25 the magnitude and phase response of elliptic filter

From the plot we see the stability.

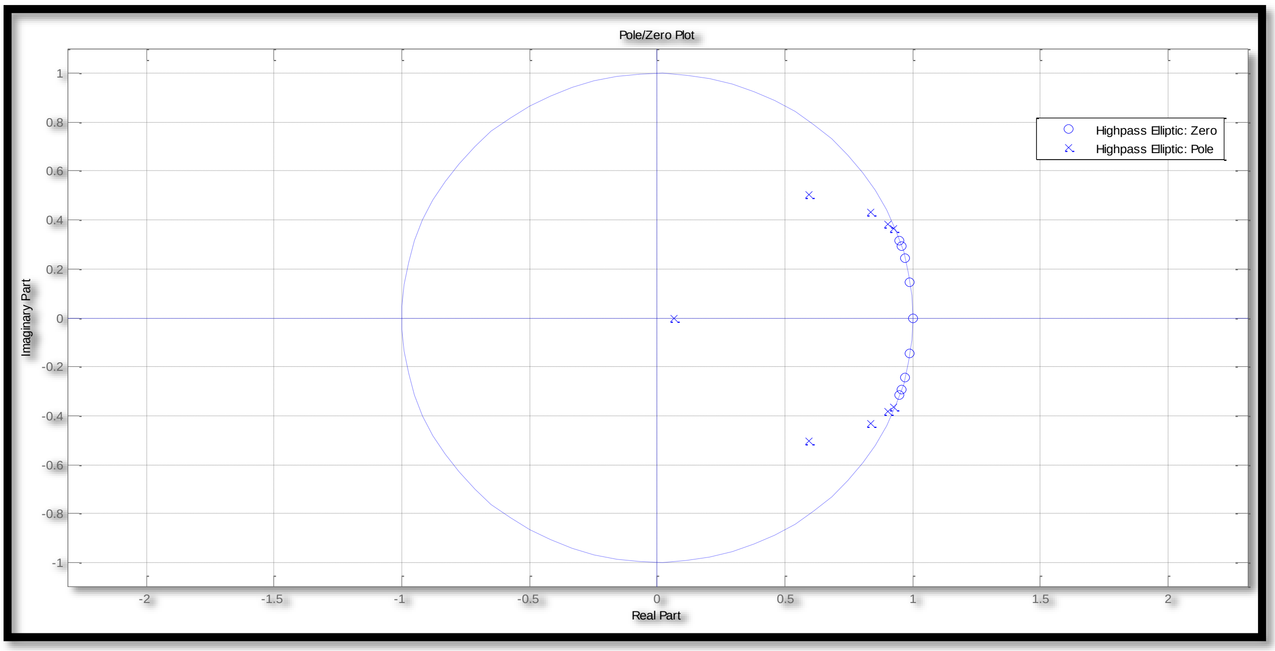


Fig4.4.26 pole zero plot for elliptic

Looking at the magnitude response we see the impact of the ripples on the input "chirp" signal (yellow), ripples more obvious at the passband than the stopband.

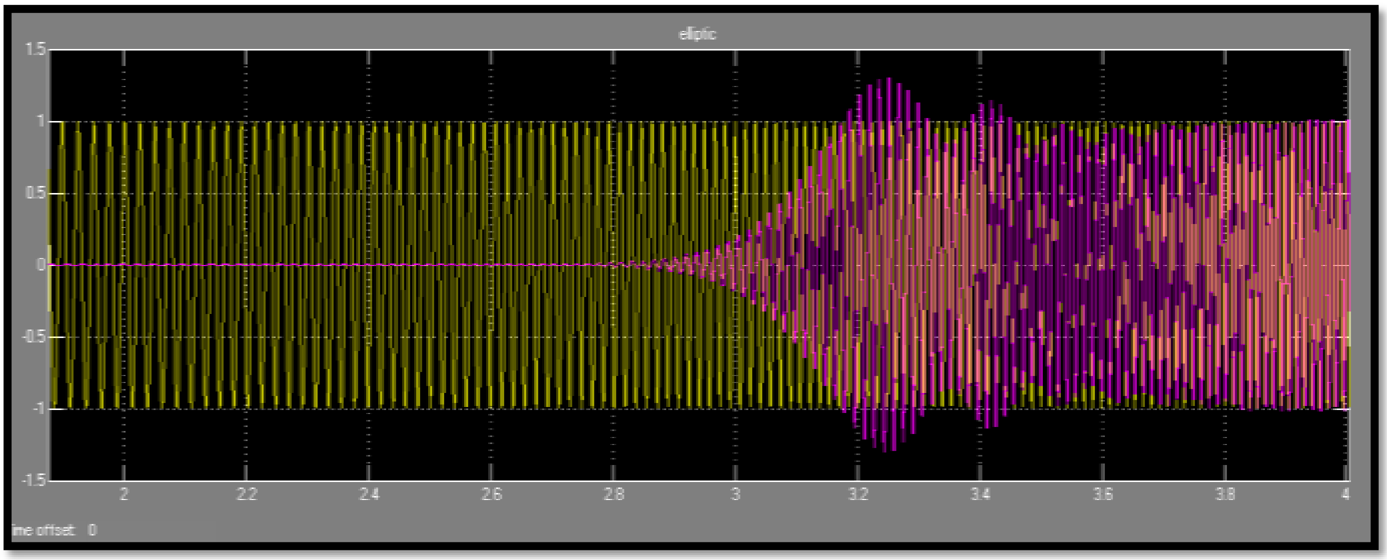


Fig4.4.27 The chirp signal before (yellow) and after(purple) the elliptic HPF

Applying the experiment on the DSK7613:

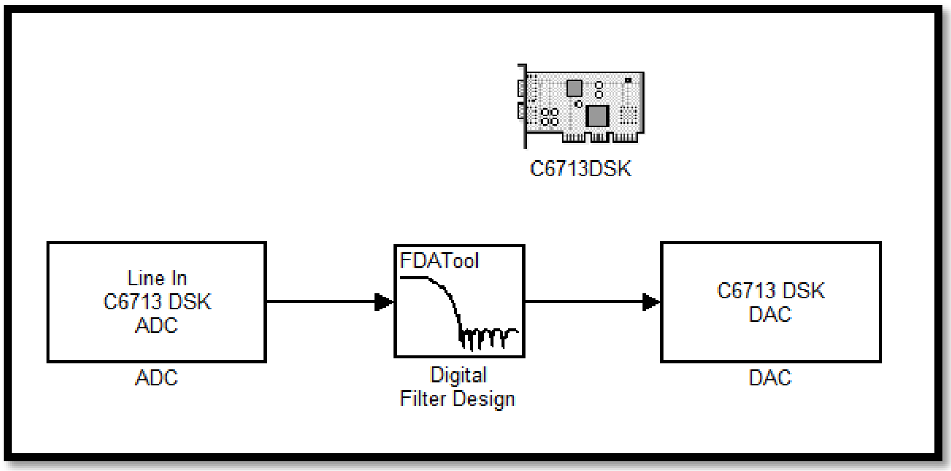
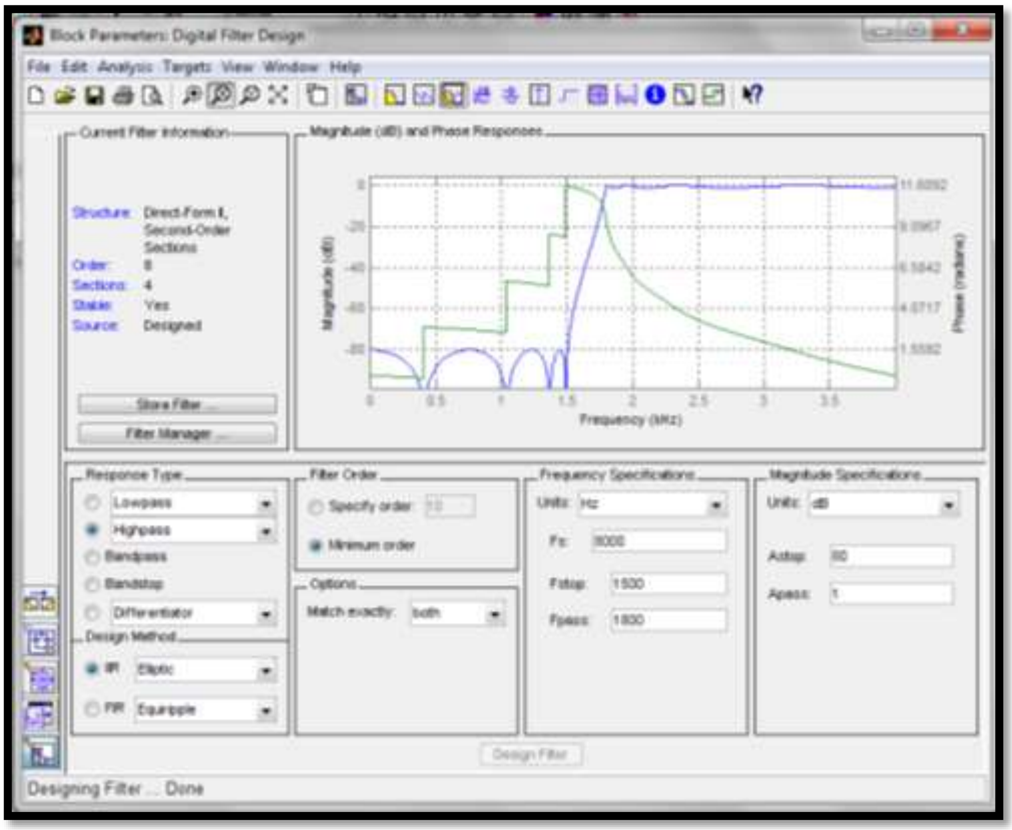


Fig4.4.28elliptic circuit on Kit

After making the circuit on simulink as previous ,with the same conditions and parameters fig4.4.27 the sampling rate is 1/8000 for ADC, DAC and digital filter ,thus $f_s = 8000Hz$, $f_{stop} = 1500Hz$, $f_{pass} = 1800Hz$,put IIR filter on elliptic ,then start and down load…


Block Parameters: Digital Filter Design
File Edit Analysis Targets View Window Help
Current Filter Information
Magnitude (dB) and Phase Responses
Magnitude (dB)
Phase (radians)
Frequency (kHz)
Store Filter ...
Filter Manager ...
Design Filter

Structure: Direct-Form II, Second-Order Sections
Order: 8
Sections: 4
Stable: Yes
Source: Designed

Response Type: Lowpass [✓ Highpass] Bandpass Bandstop Differentiator
Design Method: [✓ IIR Elliptic] FIR Equiripple
Filter Order: Specify order: 10 [✓ Minimum order]
Options: Match exactly: both
Frequency Specifications: Units: Hz; Fs: 8000; Fstop: 1500; Fpass: 1800
Magnitude Specifications: Units: dB; Astop: 80; Apass: 1

Designing Filter ... Done

Fig.4.4.29 the parameter of elliptic filter

After applying it we see the result on oscilloscope fig4.4.28 , the ripples appear at stope band ant little at pass band.

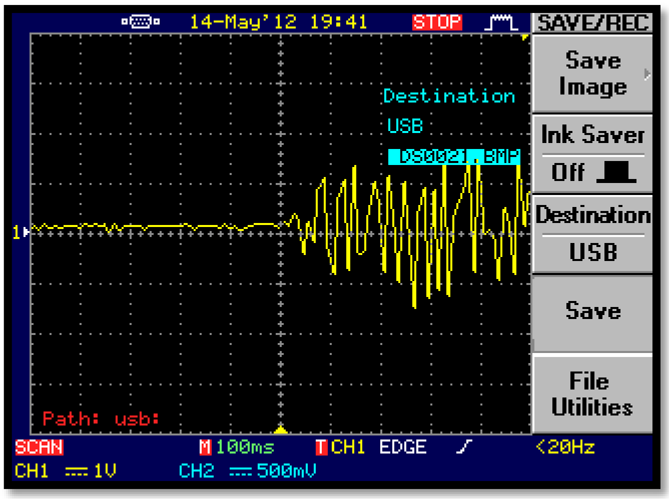


Fig4.4.30 the whistling signal on oscilloscope pass after cheelliptic HPF

Conclusion:

- Elliptic filters are equiripple in both the passband and stopband.
- As the ripple in the stopband approaches zero, the filter becomes a type I Chebyshev filter. As the ripple in the passband approaches zero, the filter becomes a type II Chebyshev filter and finally, as both ripple values approach zero, the filter becomes a Butterworth filter.

4.5 FIR filters:

As we study before FIR filters are very important for many applications , and it is finite ; because no feedback used ,also FIR filters have a linear phase , we will apply tow groups together , the first one apply the Hamming , Hann and Blackman filters, and the second group contains : rectangular and Kaiser filters.

4.5.1 Hamming / Hann / Blackman filter:

To study the characteristics of these filter we build this circuit on matlab as fig.4.5.1, using chirp signal as input signal with 40 Hz , go through rate transition with sampling time 1/100 , and LPF designed at $f_s = 100Hz$, f $f_c = 12Hz$, finally to the mux then the scope.

All three window types give approximately the same response , which is the effect of sidelobe not very large compared with Kaiser and rectangular.

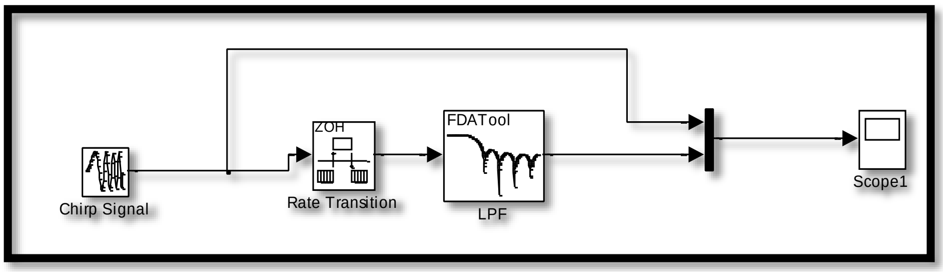
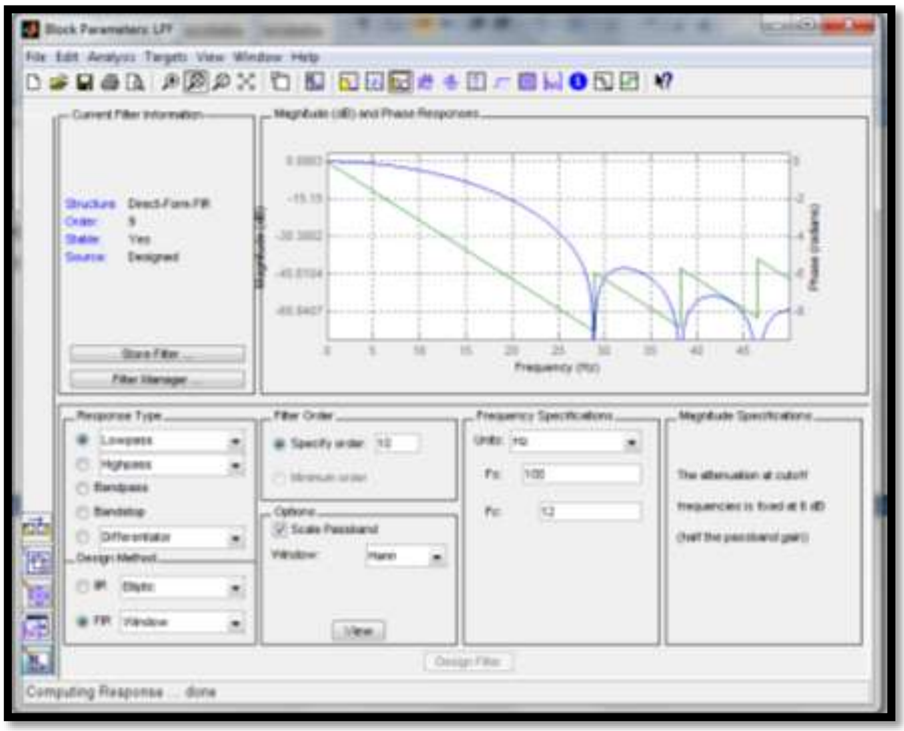


Fig.4.5.1 circuit diagram of LPF

- **Hann Filter:**



Structure Direct-Form FIR
Order 9
Stable Yes
Source Designed

Specify order 10

Scale Passband ✓
Window: Hann

Units: Hz
Fs: 100
Fc: 12

The attenuation at cutoff frequencies is fixed at 6 dB (half the passband gain)

Computing Response ... done

Fig.4.5.2 hann parameter

The effect of sidelobes very small.

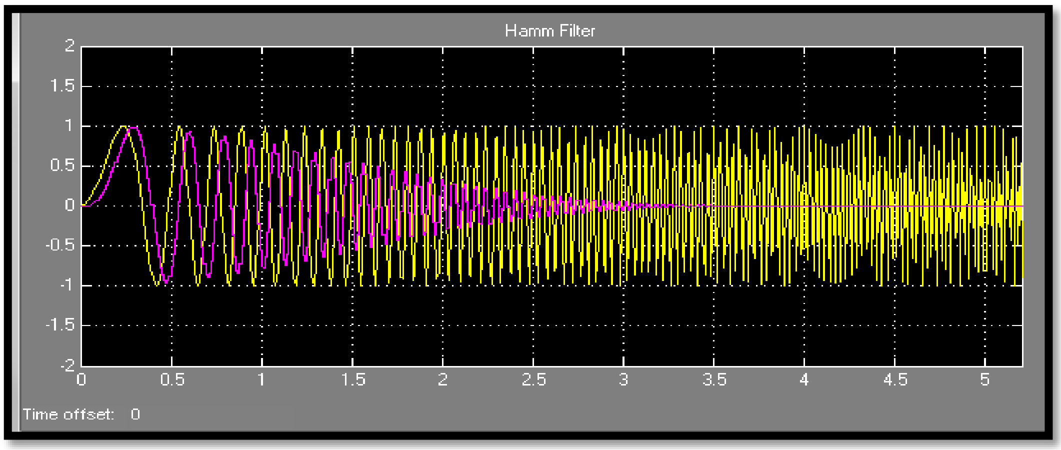


Fig.4.5.3 the magnitude response of hann filter from scope.

We see the liner phase, and the side lobes at the stope band.

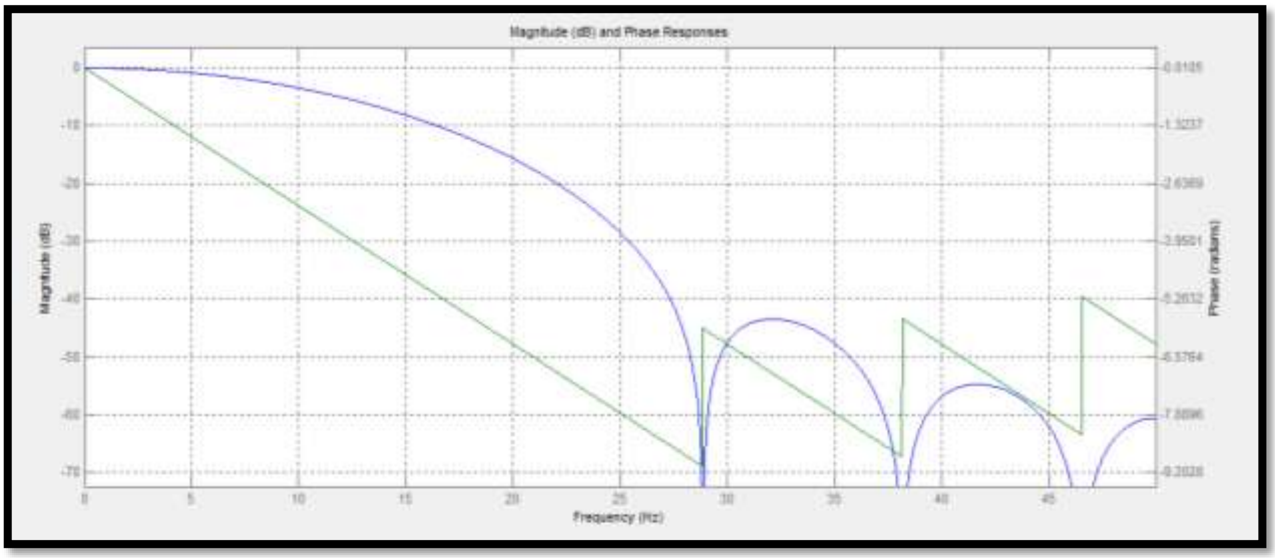


Fig.4.5.4 the magnitude and phase response of Hann filter

Certainly ,all poles located at the origin ,because it is FIR filter , also its stable always!   Fig.4.5.4.

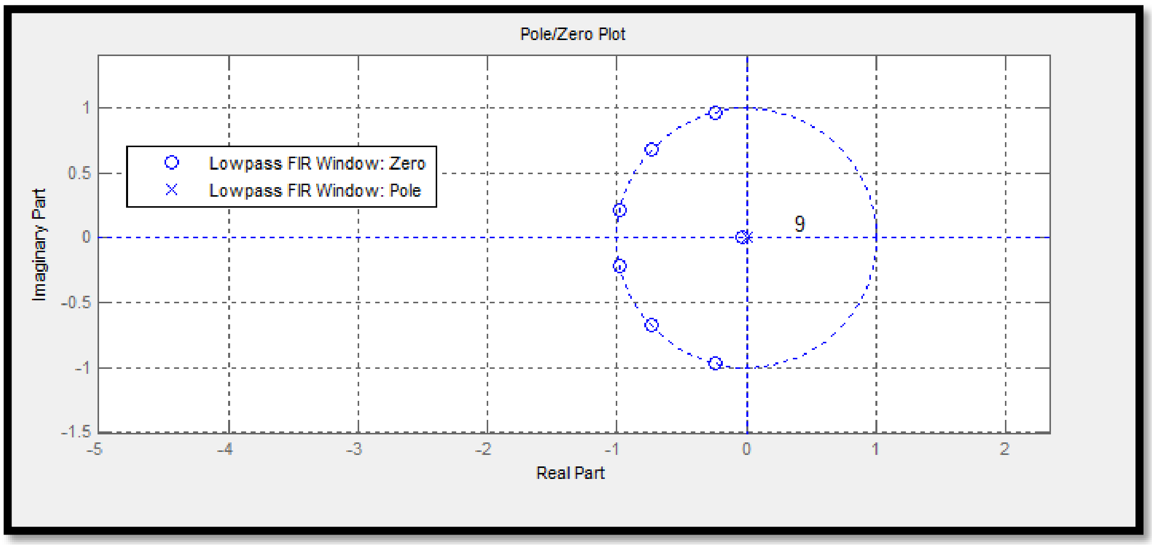
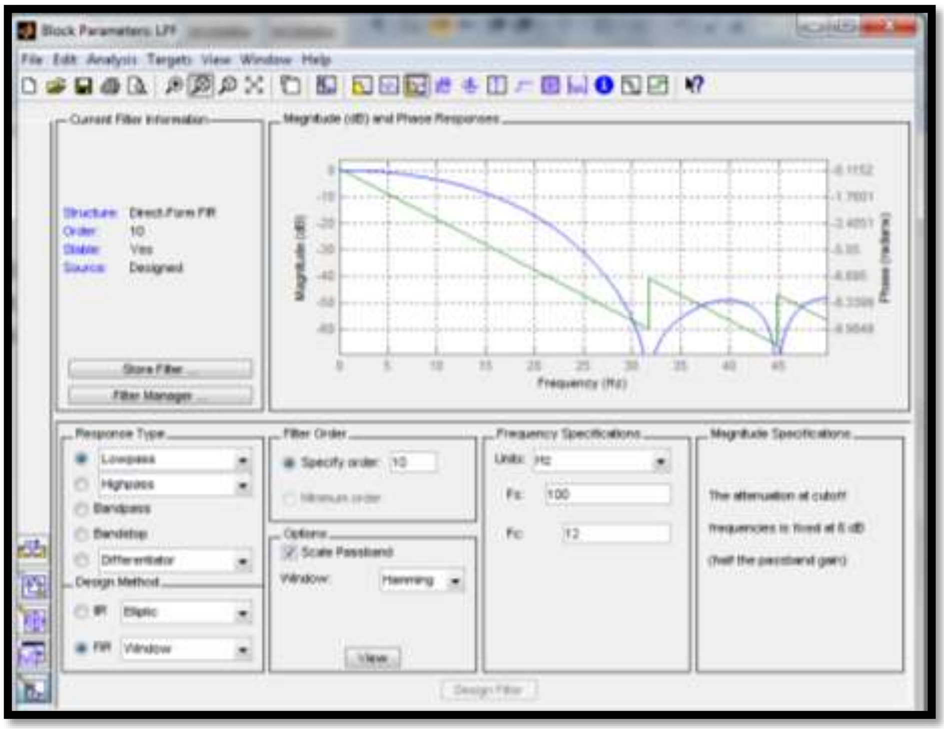


Fig.4.5.5 the pole zero plot of Hann filter

- **Hamming Filter:**
  Almost the same characteristics , the hamming is extended of Hann filter , we just change the type of window , and keep all parameters the same.



Structure: Direct-Form FIR
Order: 10
Stable: Yes
Source: Designed

Response Type: [✓ Lowpass] Highpass Bandpass Bandstop Differentiator
Design Method: IIR Elliptic [✓ FIR Window]
Filter Order: [✓ Specify order: 10] Minimum order
Options: ✓ Scale Passband
Window: Hamming
Units: Hz
Fs: 100
Fc: 12

The attenuation at cutoff frequencies is fixed at 6 dB (half the passband gain)

Fig4.5.6 Hamming parameter

Magnitude response , the side lobs appease , and the phase is liner.

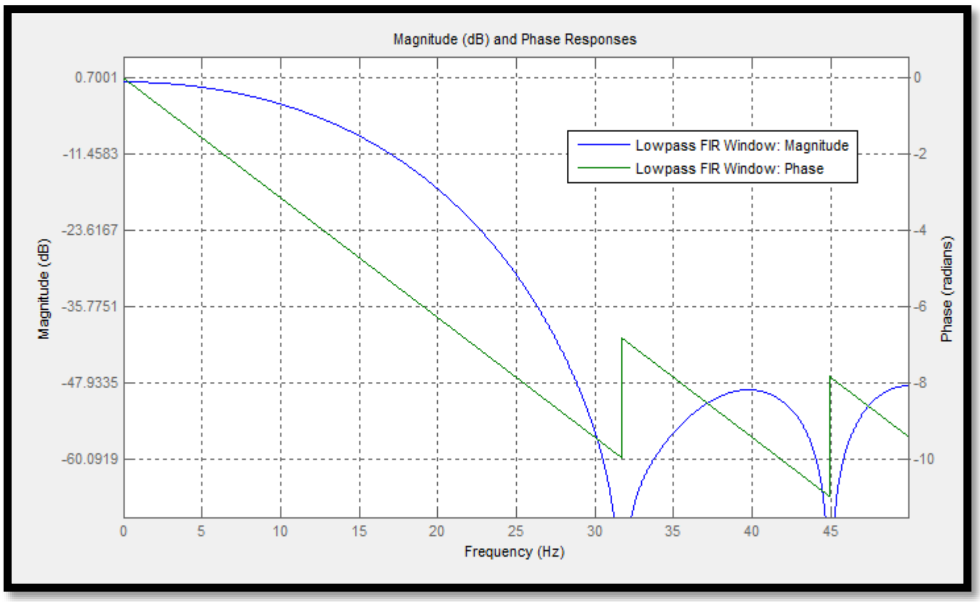


Fig.4.5.7 the magnitude and phase response Hamming filter

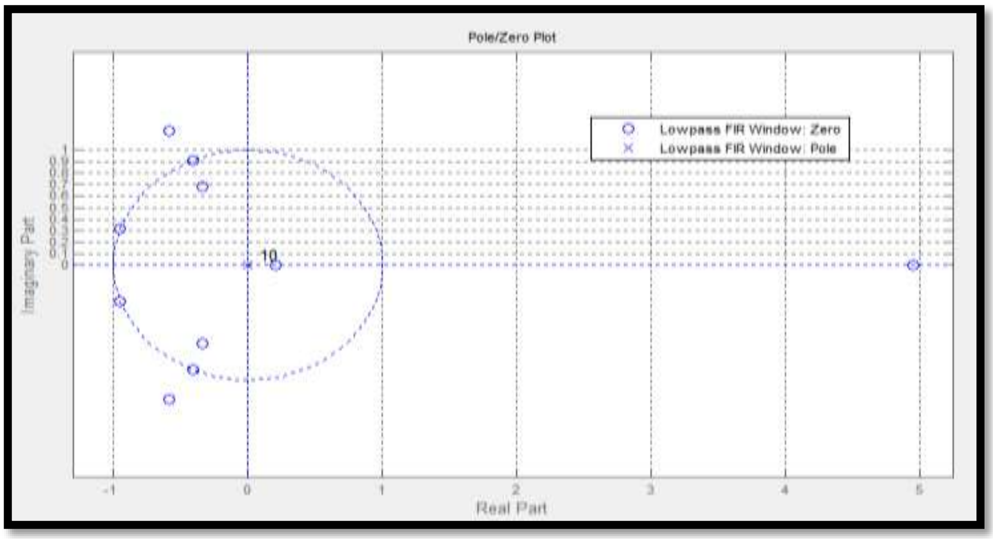


Fig.4.5.8 the magnitude and phase response Hamming filter

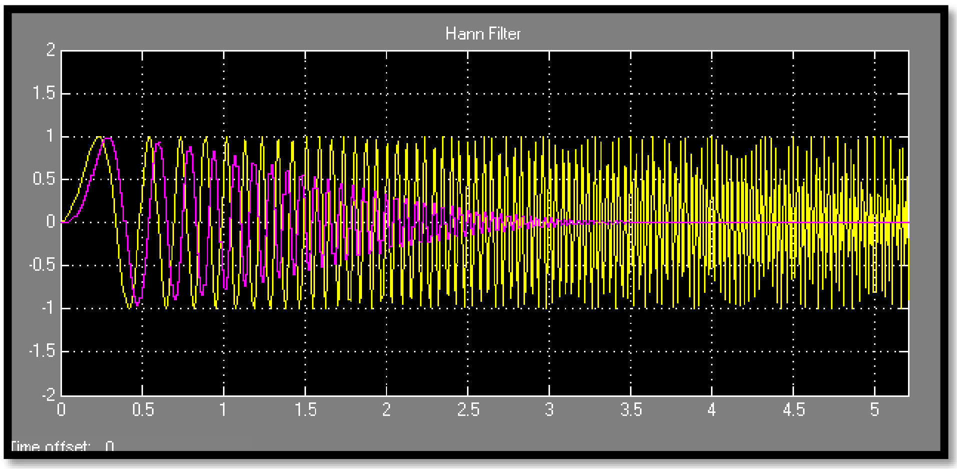
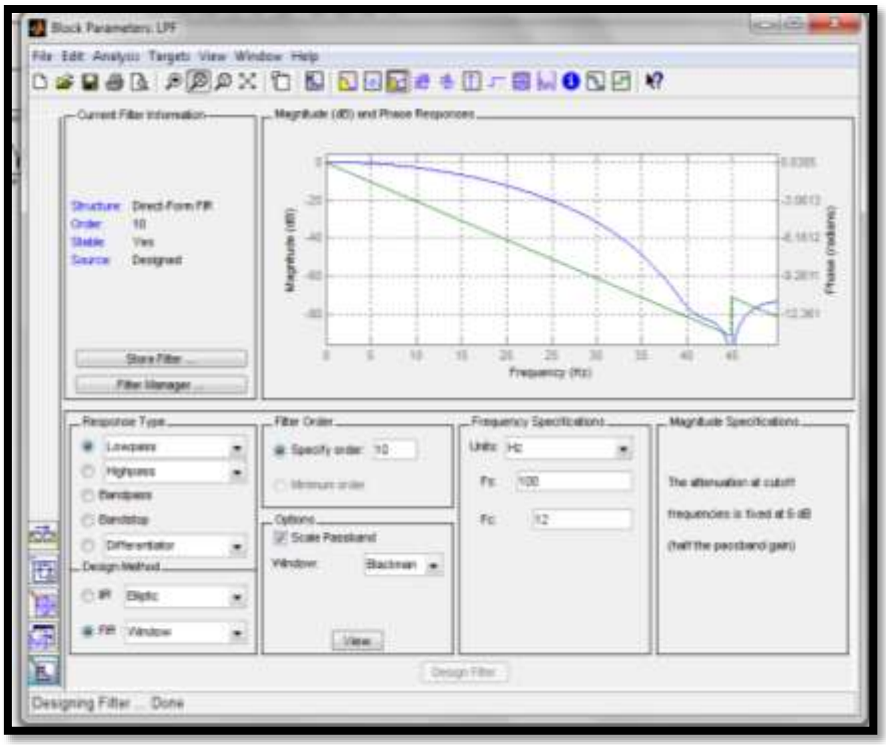


Fig.4.5.9 the magnitude response at the scope

- **Blackman Filter**

  used for single tone measurement, because it has a low max. side lobe and high side lobe roll-off rate, keep the same conditions and change just widow type.



Structure: Direct-Form FIR
Order: 10
Stable: Yes
Source: Designed

Response Type: Lowpass
Design Method: FIR Window
Filter Order: Specify order: 10
Options: Scale Passband ✓
Window: Blackman
Units: Hz
Fs: 100
Fc: 12

The attenuation at cutoff frequencies is fixed at 6 dB (half the passband gain)

Designing Filter ... Done

Fig4.5.10 Blackman filter parameter

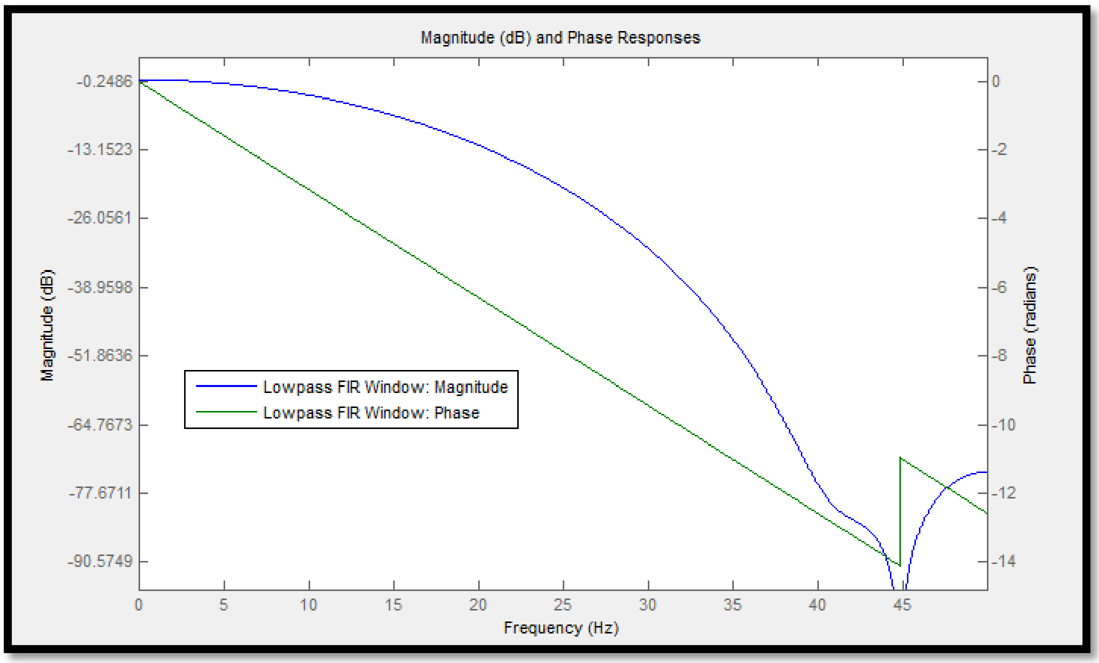


Fig.1.5.11 the magnitude and phase response Blackman filter

The filter with order 10 and 10 poles at origine ,surly Is stable some pole at origen and the rest are to the left .

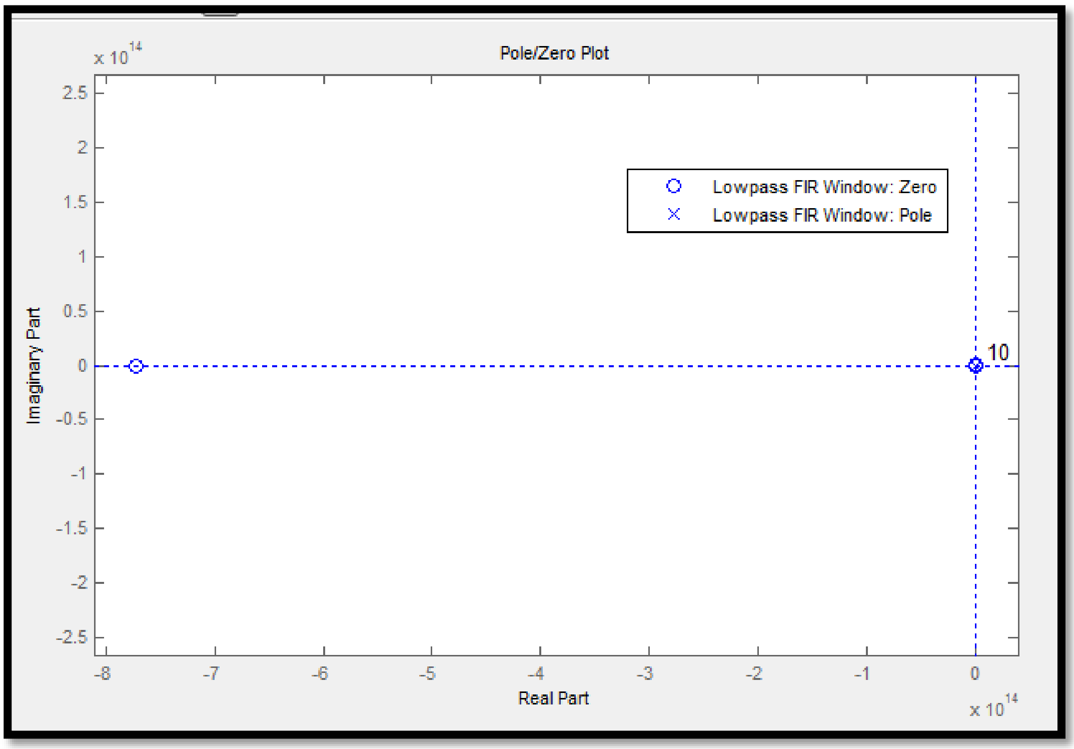


Fig.1.5.12 the magnitude and phase response Blackman filter

The transient time so flat .. no sidelobes!.

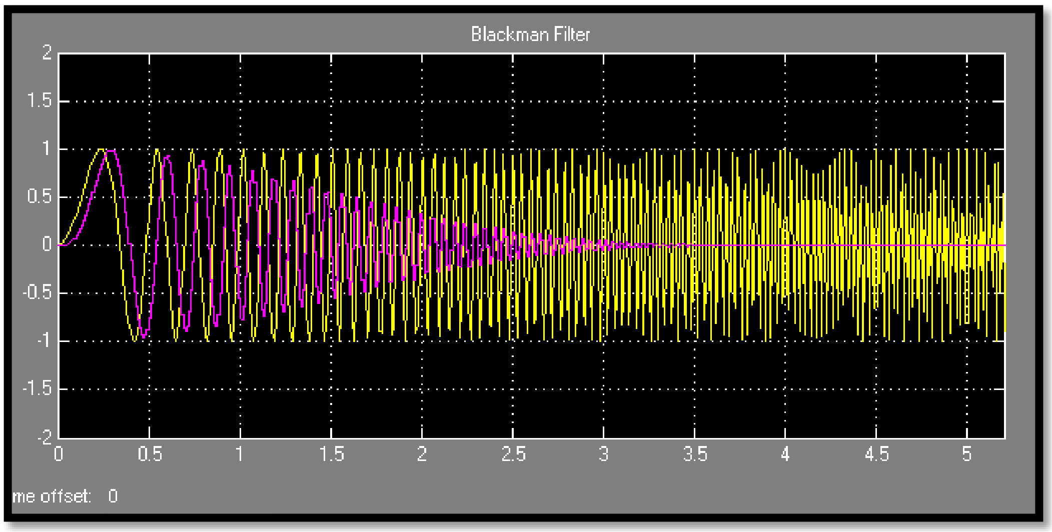


Fig.4.5.13 the magnitude response at the scope

Applying the experiment on 6713DSK :

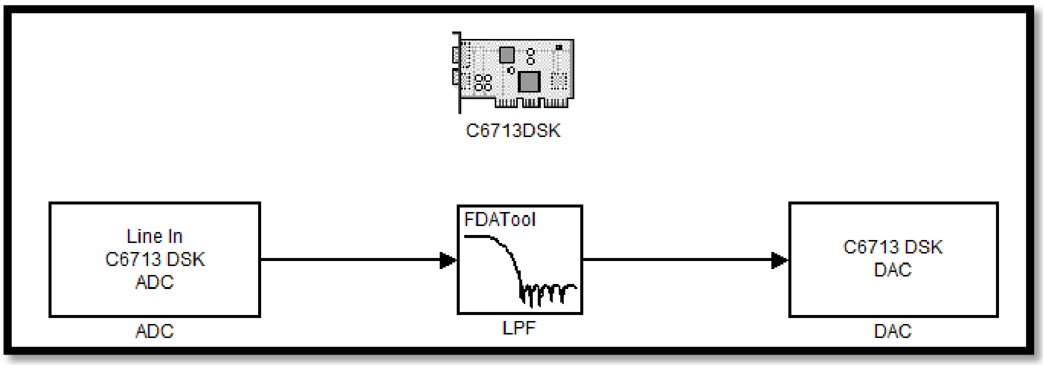
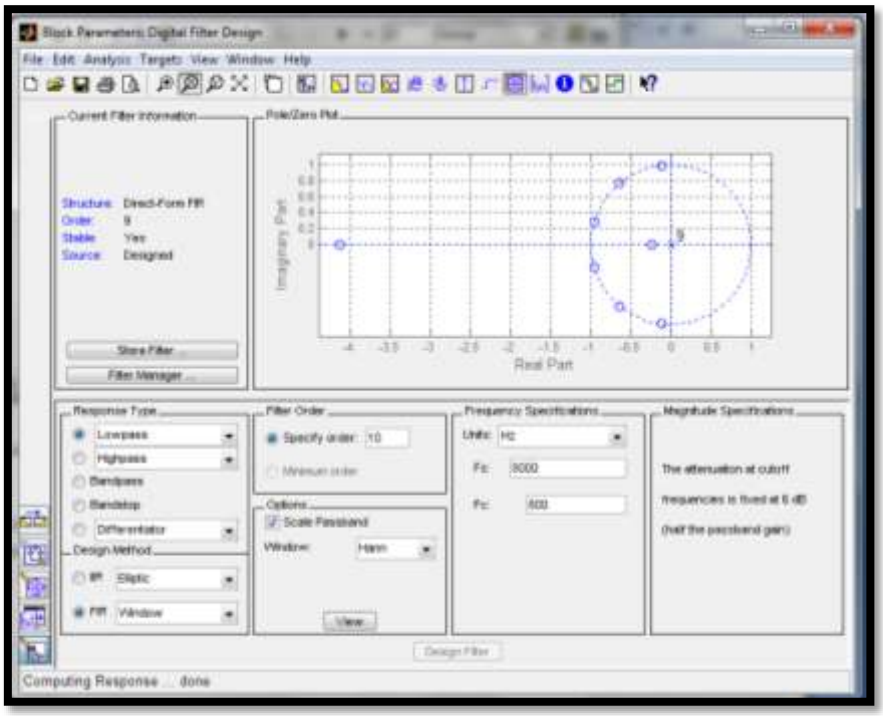


Fig.4.5.14 circuit diagram of Kit

Now we aim to download this experiment on the Kit , so at first we build the circuit in matlab with some extensions for the 6713 DSK .

The input signal is human voice wave form , so its necessary to change the input of the ADC to mike in , and make the sample time for each block 1/8000 sec , also : $f_s = 8000Hz$, $f_c = 800Hz$, with any window type of the first group.


Block Parameters: Digital Filter Design
File Edit Analysis Targets View Window Help
Current Filter Information
Pole/Zero Plot
Imaginary Part
Real Part
Store Filter ...
Filter Manager ...
Design Filter
View

Structure: Direct-Form FIR
Order: 9
Stable: Yes
Source: Designed

Response Type: [✓ Lowpass] Highpass Bandpass Bandstop Differentiator
Design Method: IIR Elliptic [✓ FIR Window]
Filter Order: [✓ Specify order: 10] Minimum order
Options: [✓ Scale Passband]
Window: Hann
Frequency Specifications
Units: Hz
Fs: 8000
Fc: 600
Magnitude Specifications
The attenuation at cutoff frequencies is fixed at 6 dB (half the passband gain)

Computing Response ... done

Fig.4.5.15 the Hann filter parameter of Kit

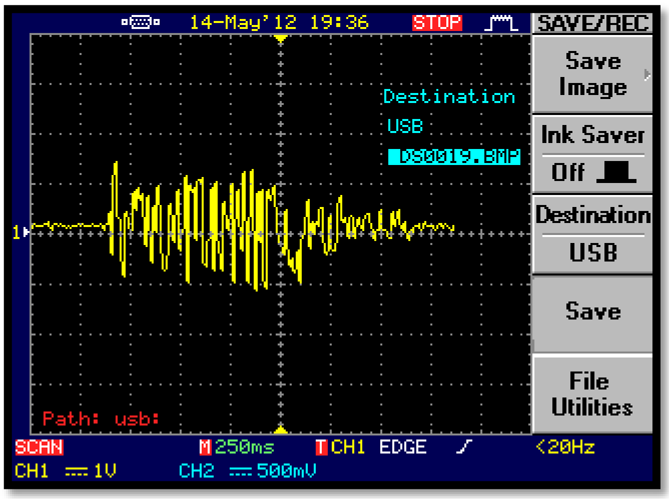


Fig.4.5.16 the ordinary talk pass from the LPF

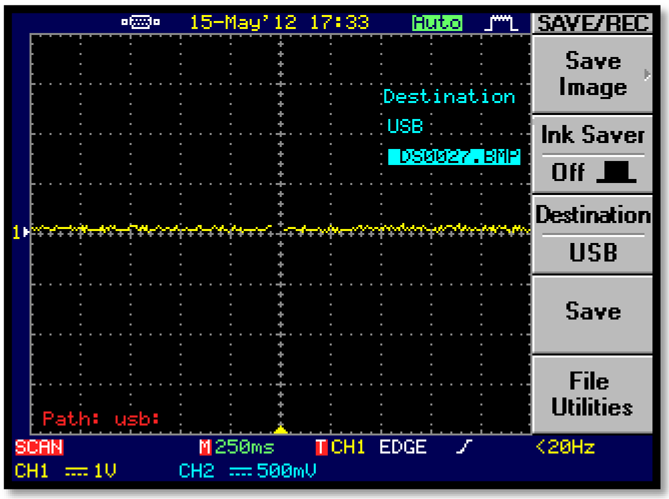


Fig.4.5.17 the whistling will not pass

4.5.2 Rectangular / Kaiser filters:

This group differs from the previous one in response , In Rectangular and Kaiser the effect of sidelobe will appear clearly.

Using the same situation of the first group , the same circuit and parameters , with change the window type .using chirp signal as input signal with 40 Hz , go through rate transition with sampling time 1/100 , and LPF designed at $f_s = 100Hz$, f $f_c = 12Hz$.

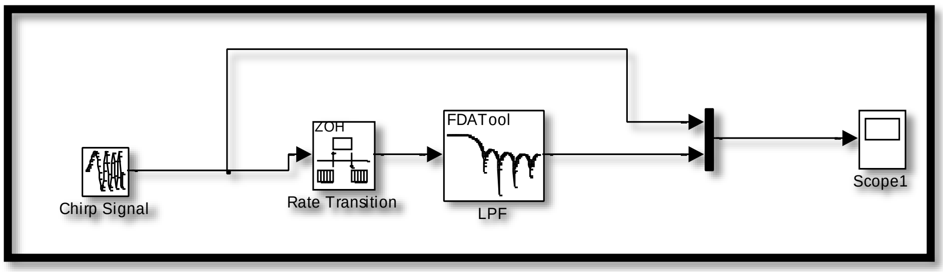
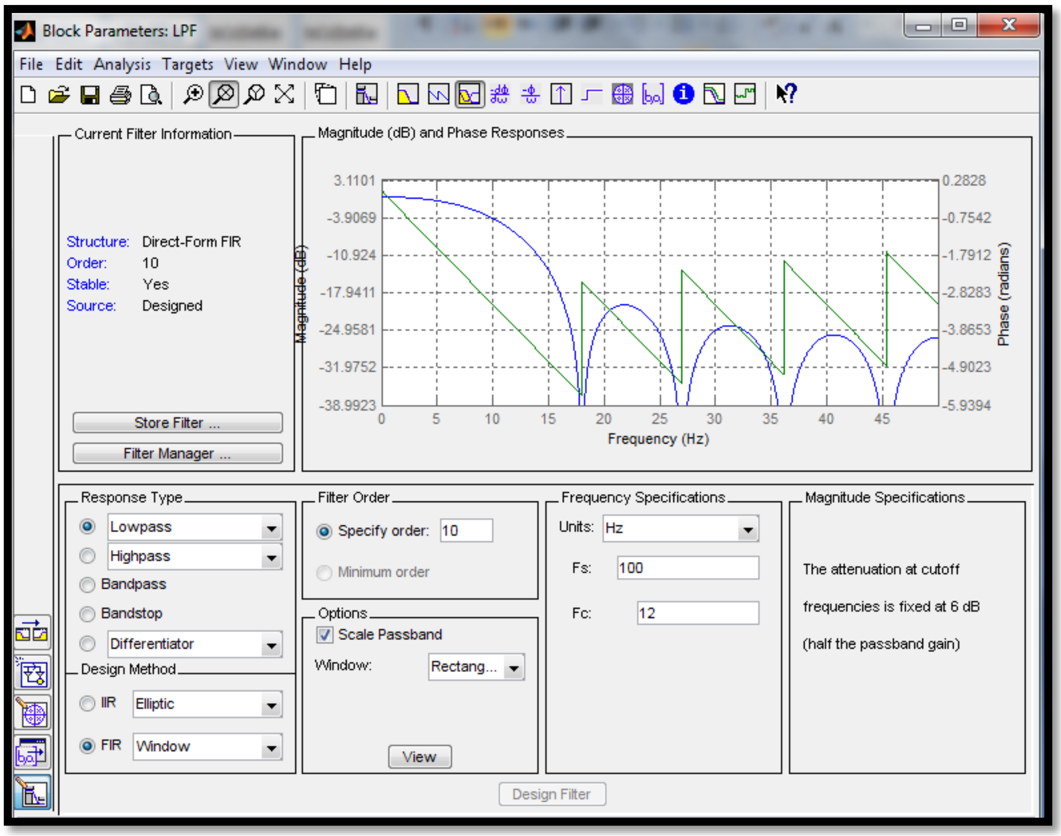


Fig.4.5.18 circuit diagram of LPF

- **Rectangular filter:**


Block Parameters: LPF
File Edit Analysis Targets View Window Help
Magnitude (dB) and Phase Responses
Store Filter ...
Filter Manager ...
View
Design Filter

Current Filter Information

Structure: Direct-Form FIR
Order: 10
Stable: Yes
Source: Designed

Response Type: [✓ Lowpass] Highpass Bandpass Bandstop Differentiator

Design Method: IIR Elliptic [✓ FIR Window]

Filter Order: [✓ Specify order: 10] Minimum order

Options: [✓ Scale Passband]
Window: Rectang...

Frequency Specifications
Units: Hz
Fs: 100
Fc: 12

Magnitude Specifications

The attenuation at cutoff frequencies is fixed at 6 dB (half the passband gain)

Fig.4.5.19 rectangular filter parameter

We note the sidelobs , and change in phase according to them, and the liner phase at mainlobe.

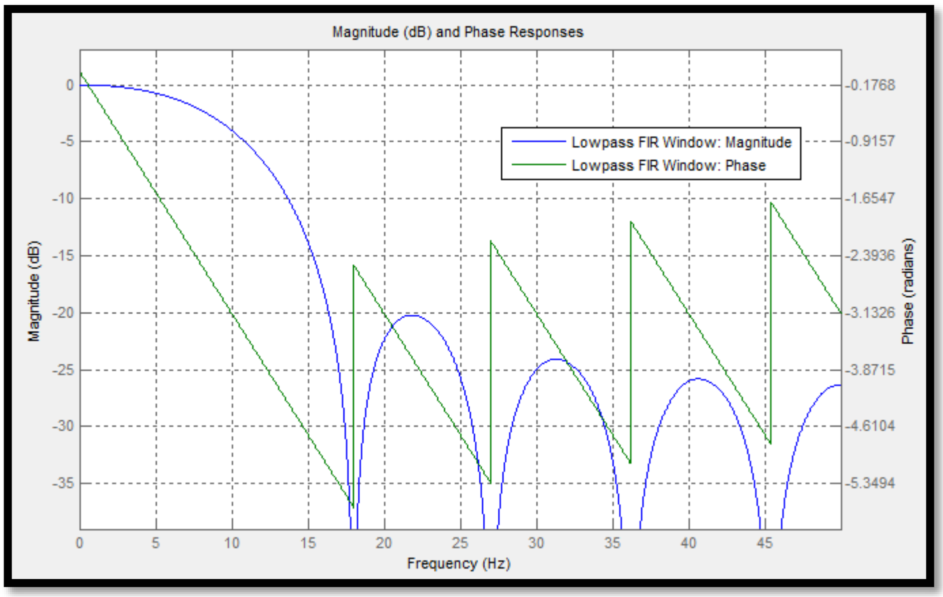


Fig.4.5. 20 the magnitude and phase response rectangular filter

The origin poles clear for stability , the zeros are distributed around the circle and one of them is out.

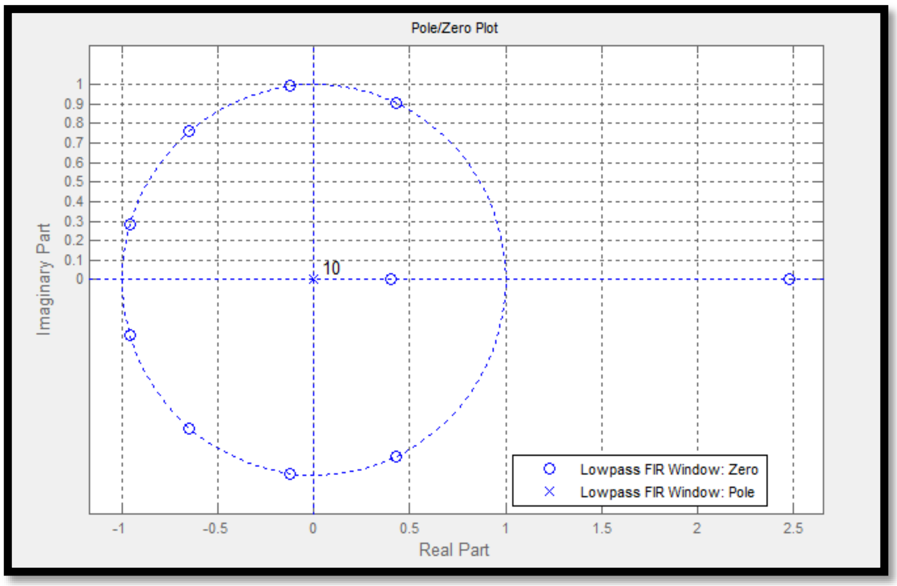


Fig.4.5.21 the magnitude and phase response rectangular filter

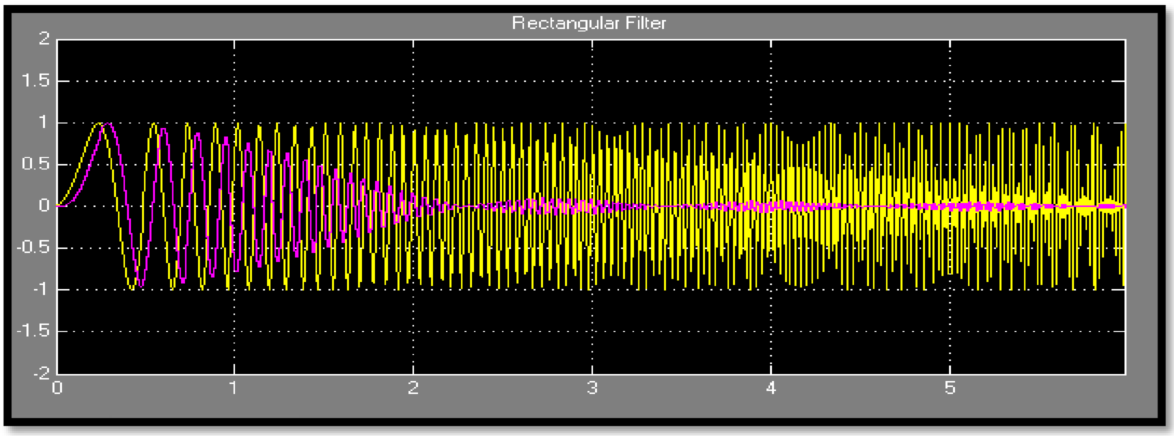
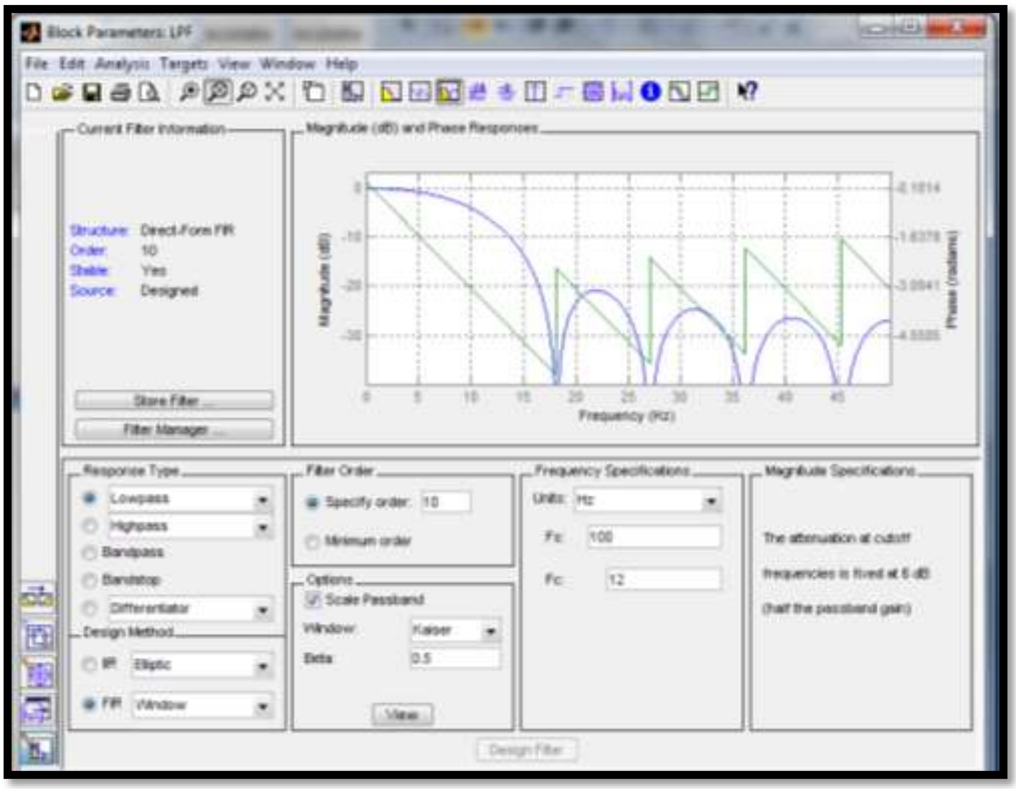


Fig.4.5.22 the magnitude response at the scope

- **Kaiser filter:**



Structure: Direct-Form FIR
Order: 10
Stable: Yes
Source: Designed

Response Type: [✓ Lowpass] Highpass Bandpass Bandstop Differentiator

Design Method: IIR Elliptic [✓ FIR Window]

Filter Order: [✓ Specify order: 10] Minimum order

Options: [✓ Scale Passband]
Window: Kaiser
Beta: 0.5

Frequency Specifications
Units: Hz
Fs: 100
Fc: 12

Magnitude Specifications
The attenuation at cutoff frequencies is fixed at 6 dB (half the passband gain)

Fig.4.5.23 Kaiser filter parameters

There is a similarity in the magnitude and phase response between rectangular an Kaiser, in sidelobs and liner phase!

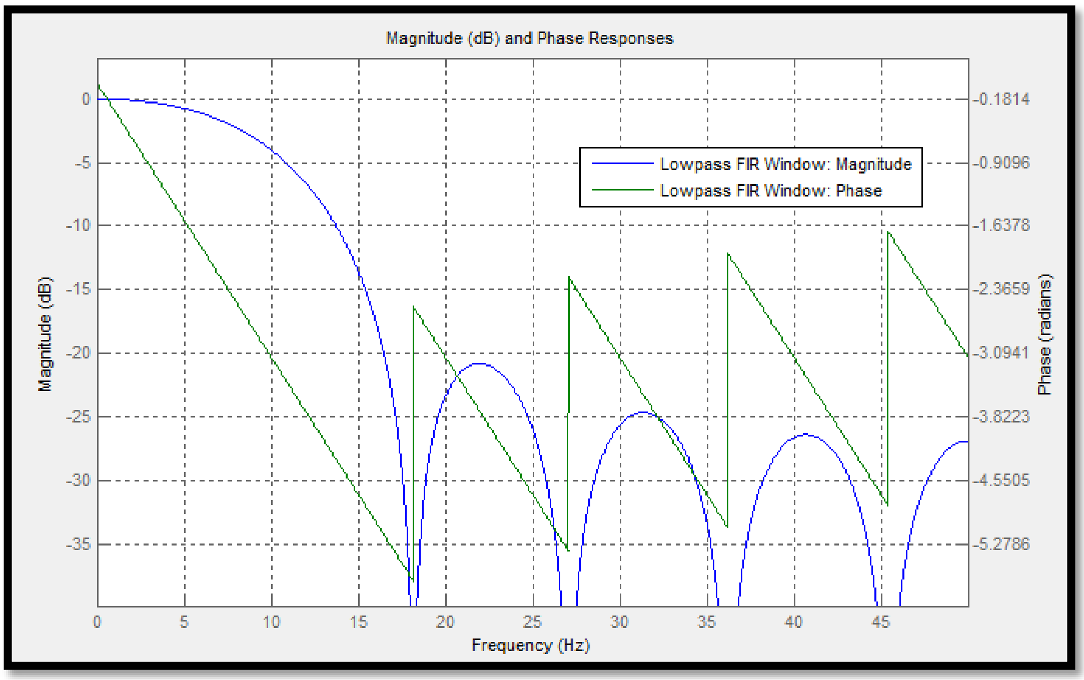


Fig.4.5.24 the magnitude and phase response kaiser filter

The side lobes noticeable at stopband.

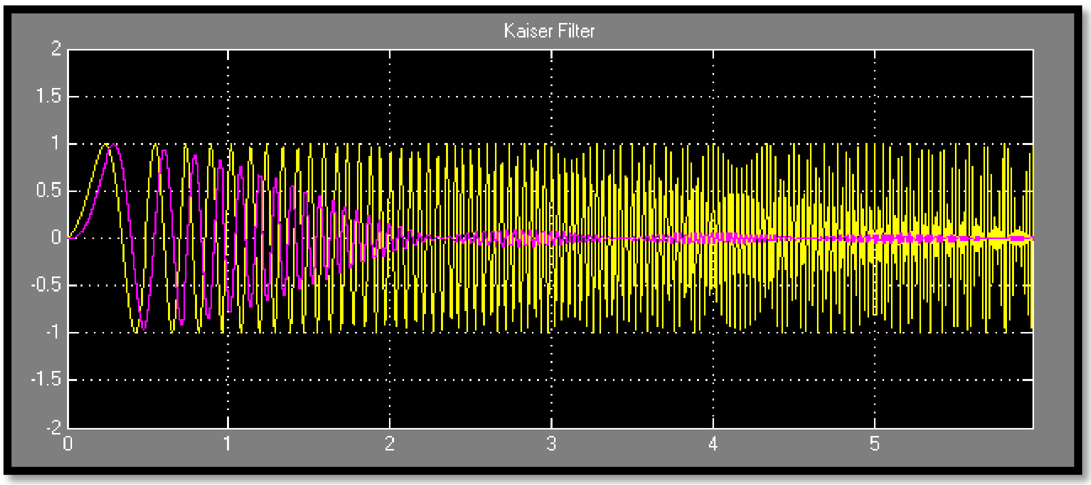


Fig.4.5.25 the magnitude response at the scope

Applying the experiment on 6713DSK :

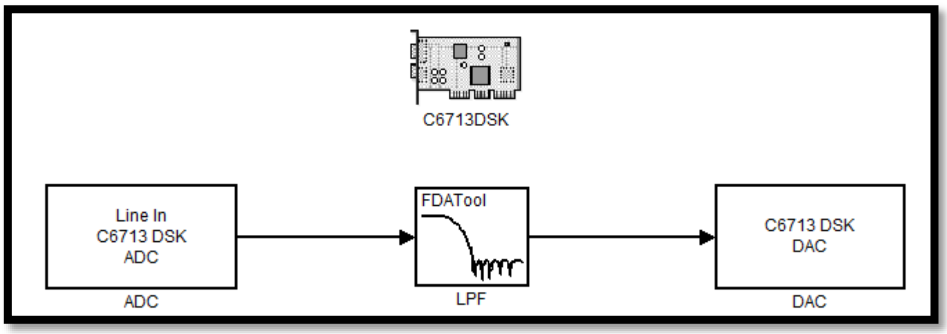


Fig.4.5.26 the circuit of LPF for Kit

Keeping everything as it is , just change the window to Kiser or rectangular, the sample time for each block 1/8000 sec , also : $f_s = 8000Hz$, $f_c = 800Hz$.

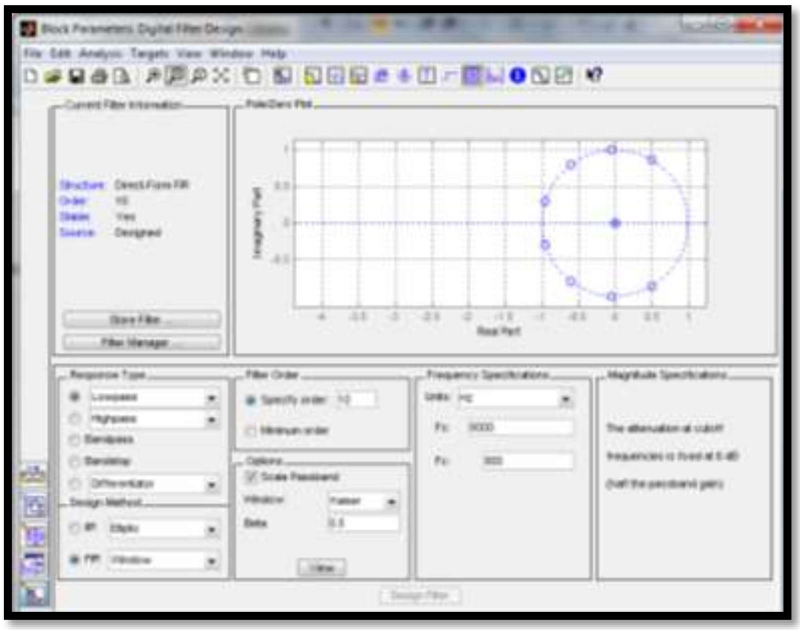


Fig.4.5.27 the Kaiser parameters for Kit

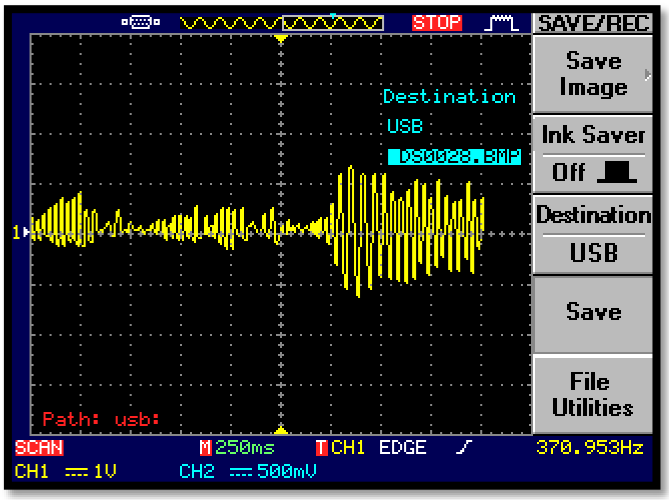


Fig 4.5.28 the output at oscilloscope for ordinary talk (note the sidelob)

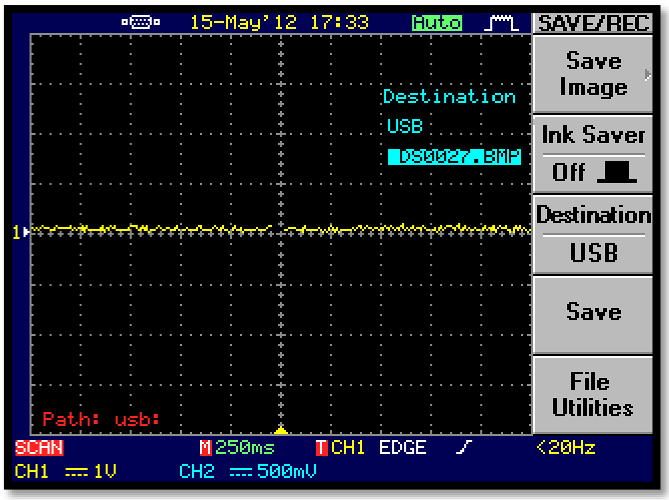


Fig.4.5 .29 the high frequency will not appear like whistling

4.6 Delay and Echo :

4.6.1 Delay:

Delay is the simplest audio effect which holds input signal and then plays it back after a period of time. Delay is used very often by musicians. It is also the main block for other audio effects such as reverb, chorus, and flanging.
The difference equation for the delay operation is:

$$y[n] = x[n-n_0]$$

where : n_0 is the delay amount .

We build the delay effect as shown in Simulink figure4.6.1 below :

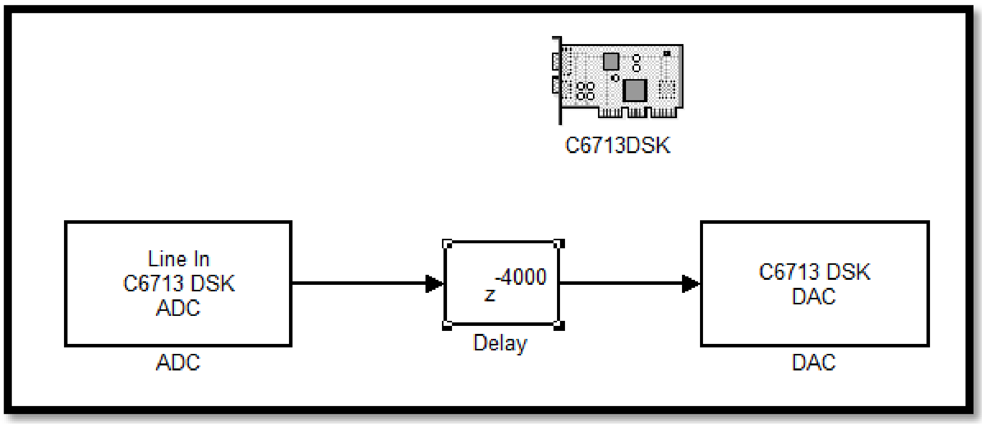
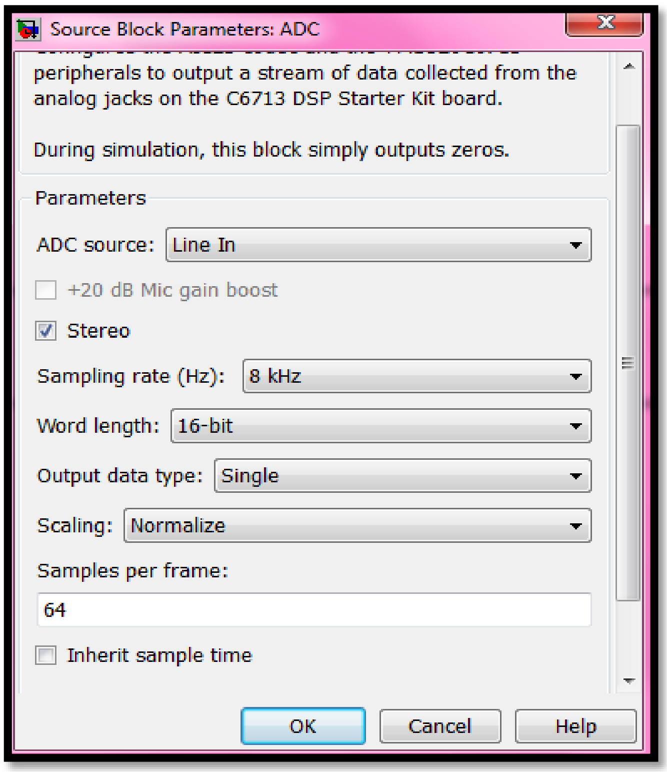


Figure 4.6.1.1 Delay effect on Simulink

Simulink components :

1. ADC: convert analog audio signal to digital signal with: sampling rate 8KHZ with 64 sample /frame. ADC source: Mic In with stereo +20 dB gain.
2. Delay block: this block create the delay effect .The delay time will be 0.5 seconds. Using a sampling rate of 8000 Hz, this requires
   8000 x0.5 = 4000 samples. This would make N = 4000.
1. 3. DAC : to convert back the digital signal to analog signal with : sampling rate 8KHZ .


Source Block Parameters: ADC

peripherals to output a stream of data collected from the analog jacks on the C6713 DSP Starter Kit board.

During simulation, this block simply outputs zeros.

Parameters

ADC source: Line In

[ ] +20 dB Mic gain boost

[✓] Stereo

Sampling rate (Hz): 8 kHz

Word length: 16-bit

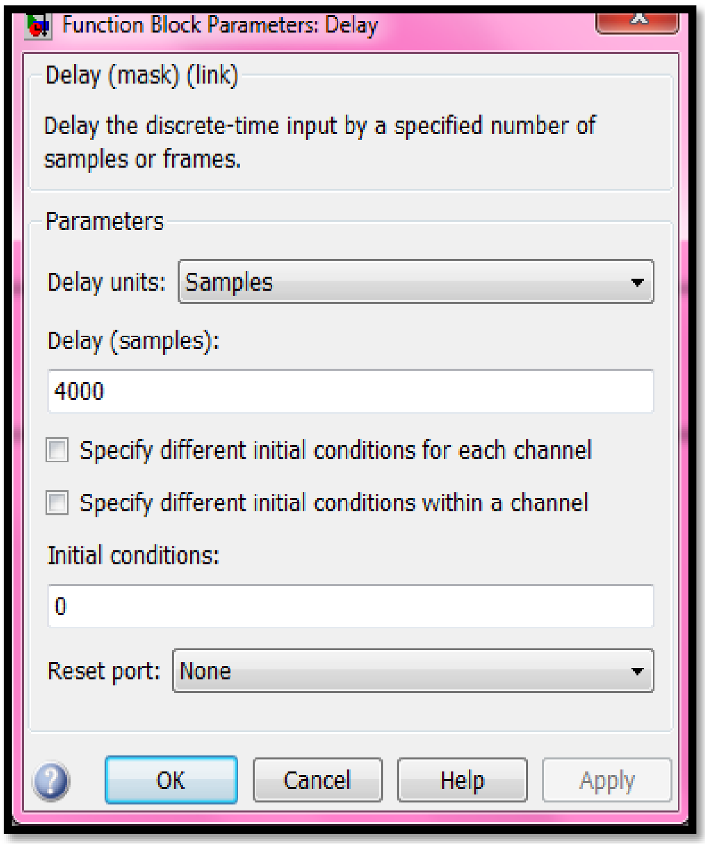
Output data type: Single

Scaling: Normalize

Samples per frame: 64

[ ] Inherit sample time

Figure 4.6.1.2 :ADC block parameters


Function Block Parameters: Delay

Delay (mask) (link)

Delay the discrete-time input by a specified number of samples or frames.

Parameters

Delay units: Samples

Delay (samples): 4000

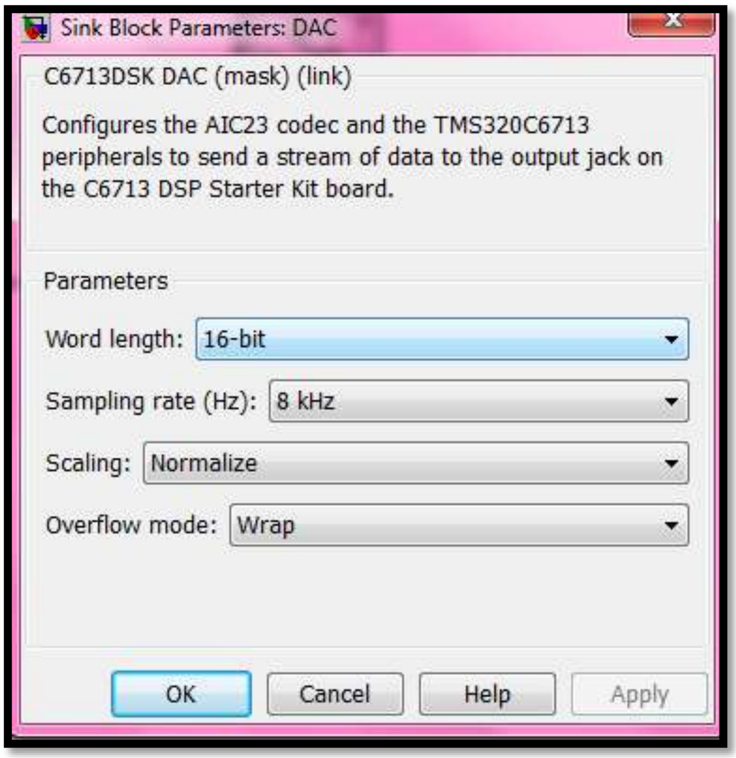
[ ] Specify different initial conditions for each channel

[ ] Specify different initial conditions within a channel

Initial conditions: 0

Reset port: None

Figure 4.6.1.3 :Delay block parameters


Sink Block Parameters: DAC

C6713DSK DAC (mask) (link)

Configures the AIC23 codec and the TMS320C6713 peripherals to send a stream of data to the output jack on the C6713 DSP Starter Kit board.

Parameters

Word length: 16-bit

Sampling rate (Hz): 8 kHz

Scaling: Normalize

Overflow mode: Wrap

Figure 4.6.1.4 : DAC block parameters

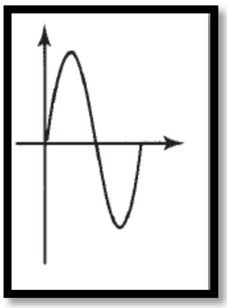
4.6.2 Echo :

An echo is simply an identical copy of the original audio signal, but delayed by a fixed amount in time. It is extremely easy to digitally create this fixed delay. As we read samples from the analog to digital converter, we store them in a circular buffer. When the buffer is filled, the store pointer wraps back around to the beginning of the buffer.

The delay comes from a single read pointer which is placed N slots "behind" the store pointer and marched along in step with the store pointer. As each new sample is stored, a sample is read from N samples behind it, thus creating a static delay of (N * 1/Fs) where Fs is the sampling rate.

This delayed signal is then mixed in with the original signal, usually at a somewhat reduced volume level

Output = Input + Delayed Input.



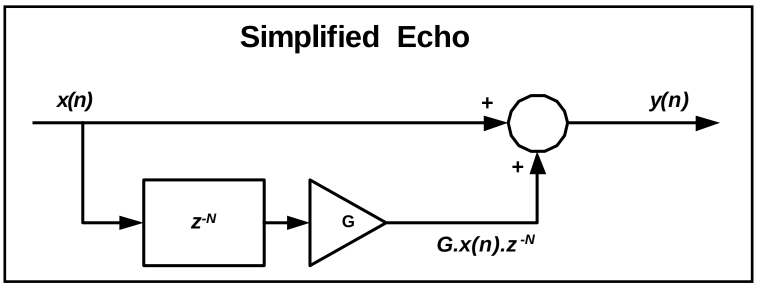
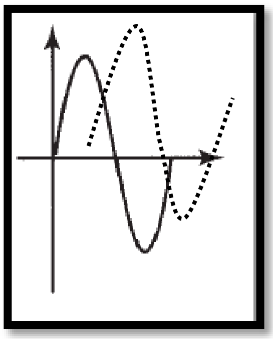


Figure 4.6.2.1 the echo signal



The delay time will be 0.5 seconds. Using a sampling rate of 8000 Hz, this requires 8000 x0.5 = 4000 samples. This would make N = 4000
.We build the echo system as shown in Simulink figure4.6.6 below:

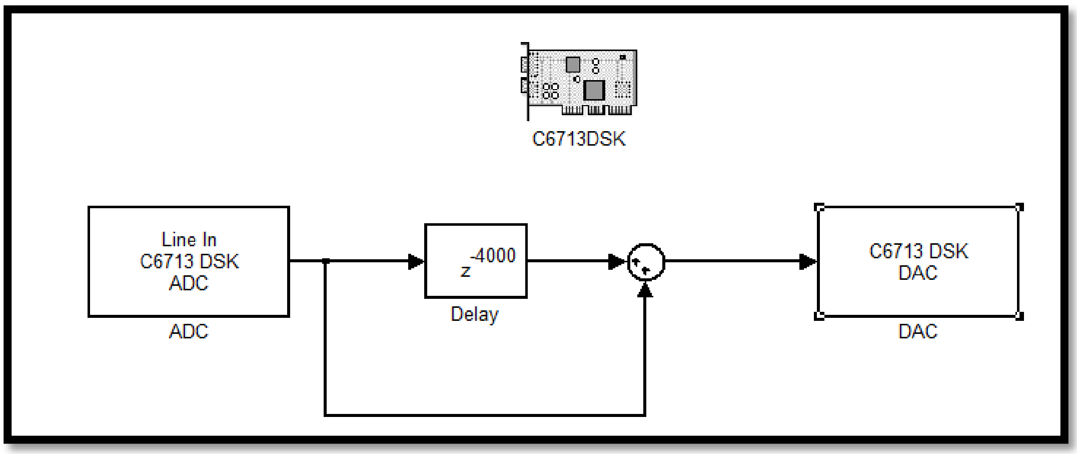


Figure 4.6.2.2 Echo system on Simulink

1. ADC: convert analog audio signal to digital signal with: sampling rate 8KHZ with 64 sample /frame. ADC source: Mic In with stereo +20 dB gain.

2. Delay block: this block create the delay effect .The delay time will be 0.5 seconds. Using a sampling rate of 8000 Hz, this requires
8000 x0.5 = 4000 samples. This would make N = 4000.

2. Addition operation: to adds up the delayed audio signal with the original audio signal.

3. DAC : to convert back the digital signal to analog signal with sampling rate 8KHZ

The figures below show the parameters of the Simulink blocks :

The result :

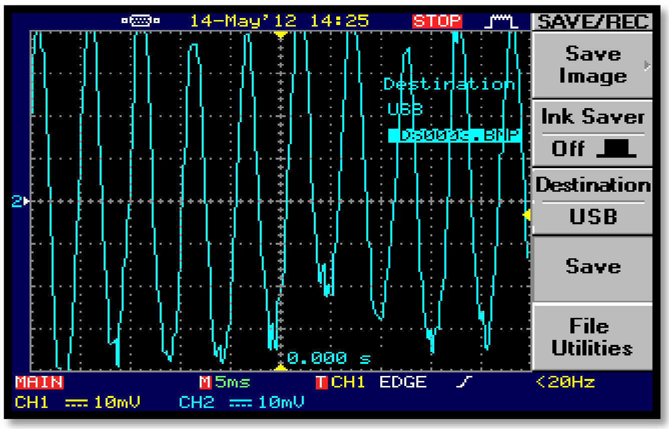


Fig 4.6.2.3 :The Audio signal without any effects

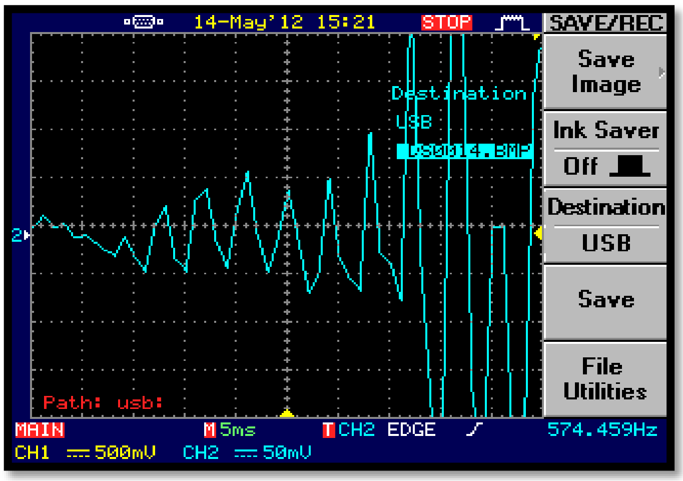


Fig 4.6.2.4 :The Audio signal with delay effects.

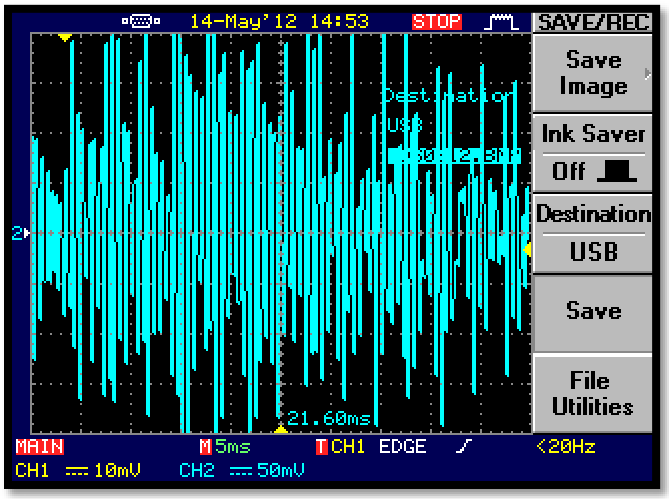


Fig 4.6.2.5 :The Audio signal with delay and echo effects .

Conclusion :

- The Echo output is derived solely from the input.
- Certain frequencies are attenuated.

4.7 AM modulation :

The function of the carrier in AM is to provide a signal to heterodyne mix with the modulated audio signal , to convert all the audio frequency component to a higher frequency in order to be able to transmit it . the carrier contain no information , it can be removed before transmission.

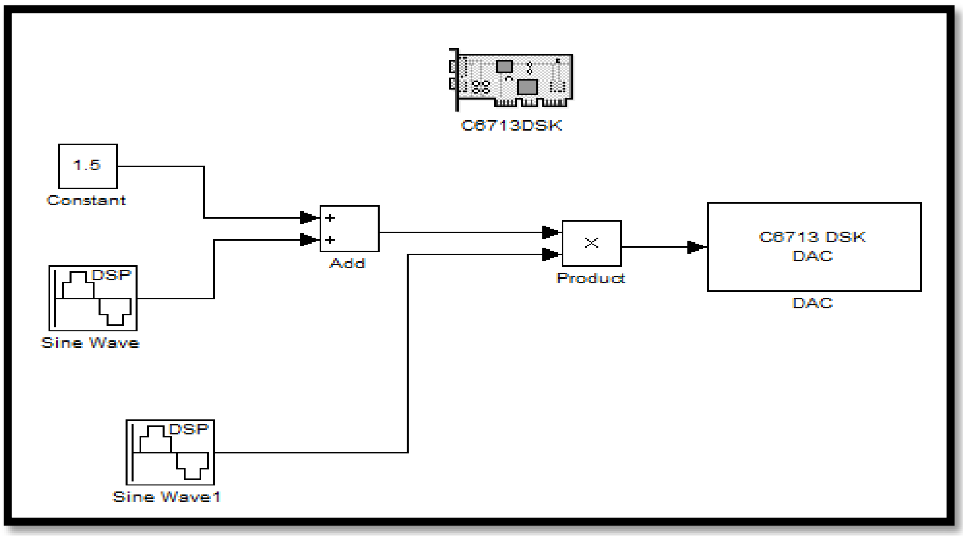
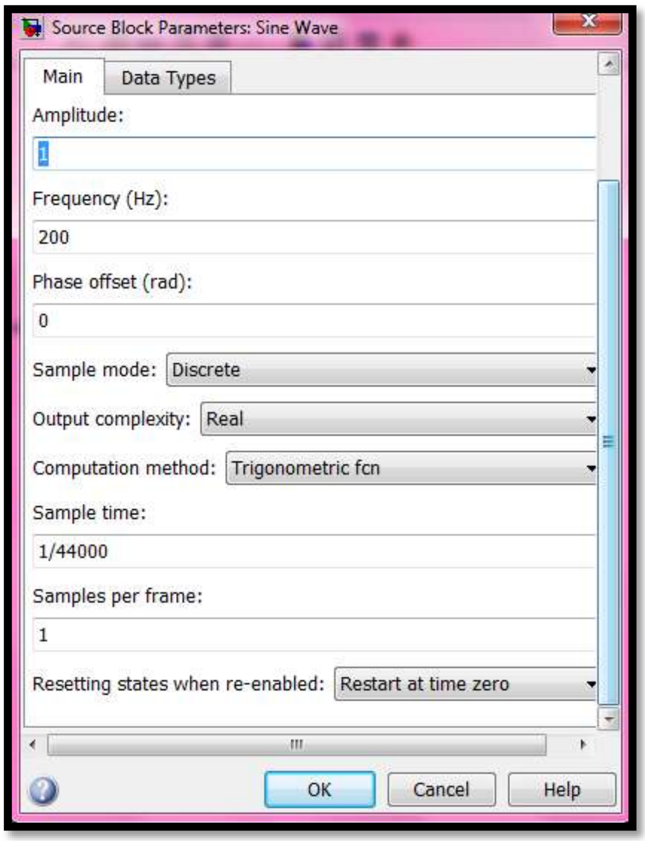


Figure 4.7.1The AM circuit diagram


Source Block Parameters: Sine Wave

Amplitude: 1

Frequency (Hz): 200

Phase offset (rad): 0

Sample mode: Discrete

Output complexity: Real

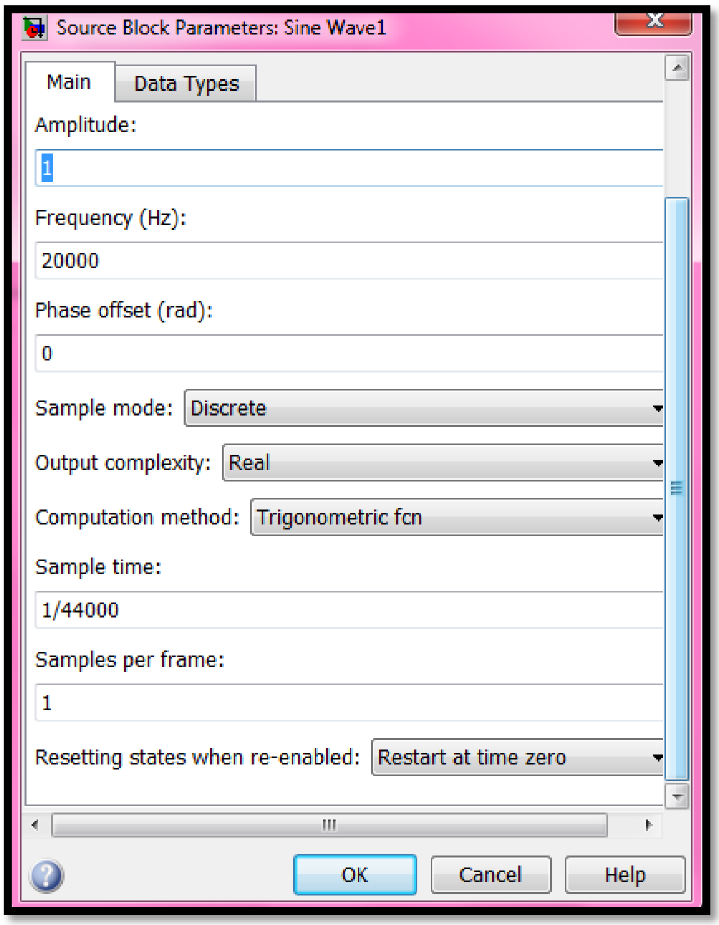
Computation method: Trigonometric fcn

Sample time: 1/44000

Samples per frame: 1

Resetting states when re-enabled: Restart at time zero

Figure 4.7.2 The parameters of the massage signal



Source Block Parameters: Sine Wave1

Amplitude: 1

Frequency (Hz): 20000

Phase offset (rad): 0

Sample mode: Discrete

Output complexity: Real

Computation method: Trigonometric fcn

Sample time: 1/44000

Samples per frame: 1

Resetting states when re-enabled: Restart at time zero

Figure 4.7.3 The parameters of the carrier signal

Results:

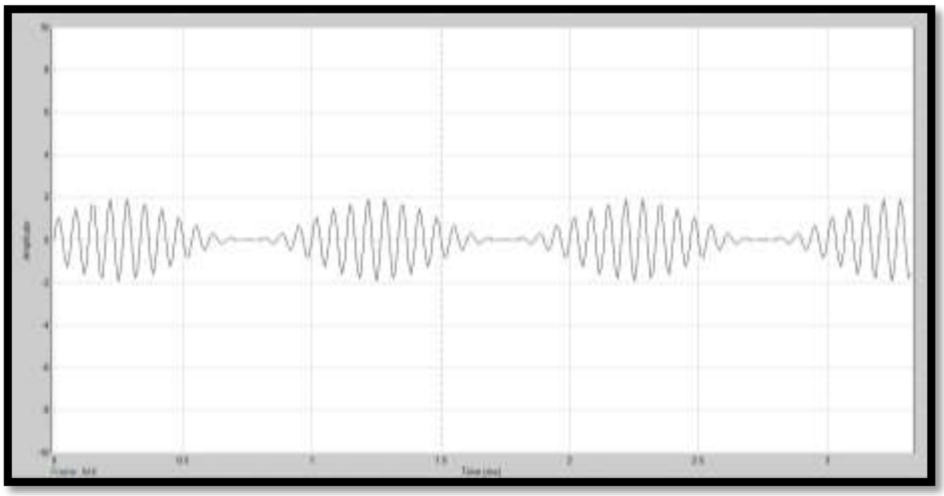


Figure 4.7.4 The AM signal with DSP constant = 1 using MATLAB.

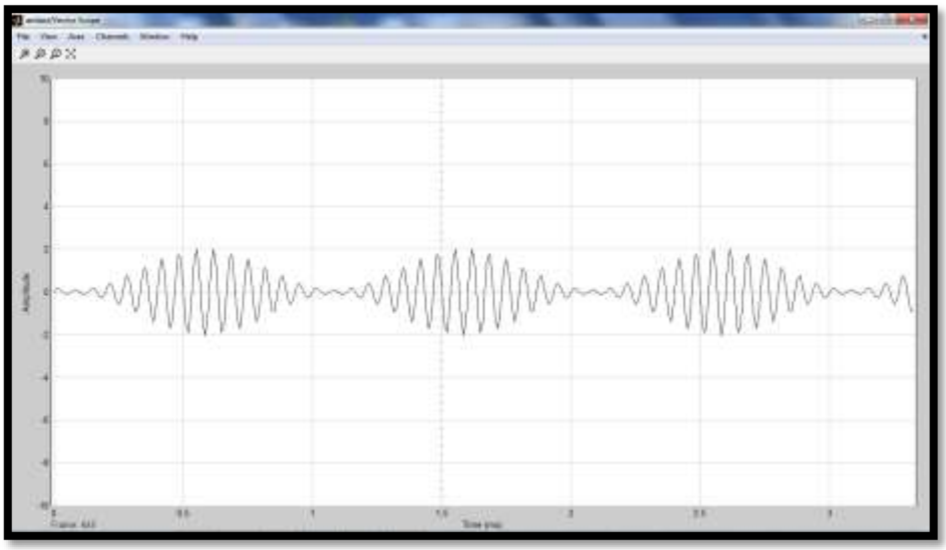


Figure 4.7.5 The AM signal with DSP constant = 1.1 using MATLAB.

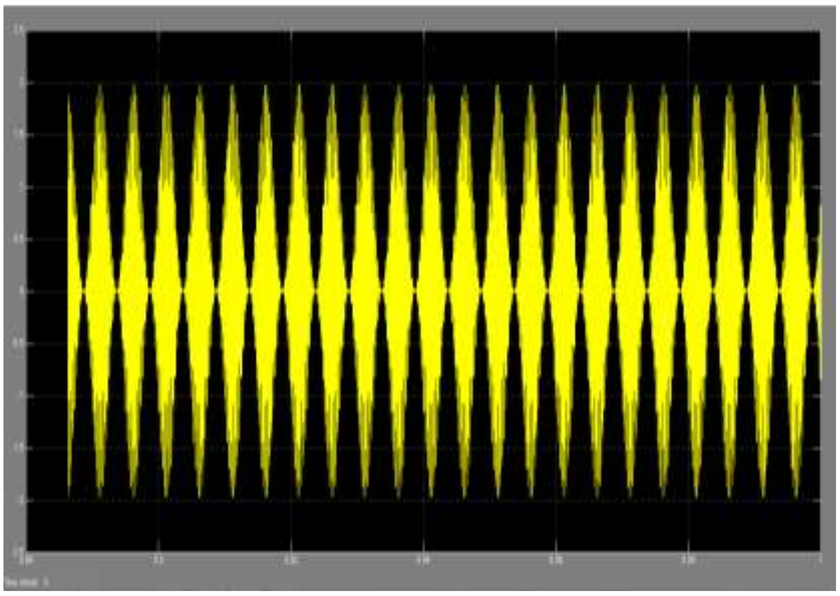


Fig 4.7.6 AM signal obtai using DSK 6713 with MATLAB

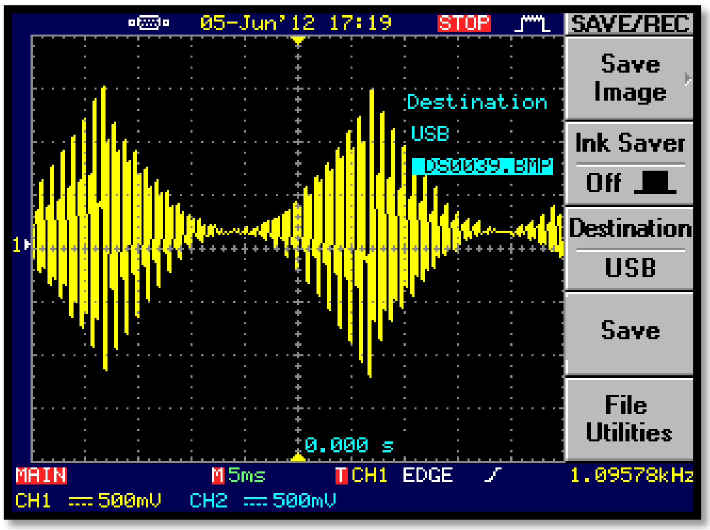


Figure 4.7.9 The AM signal with DSP constant = 1,signal amplitude=1 using DSK6713.

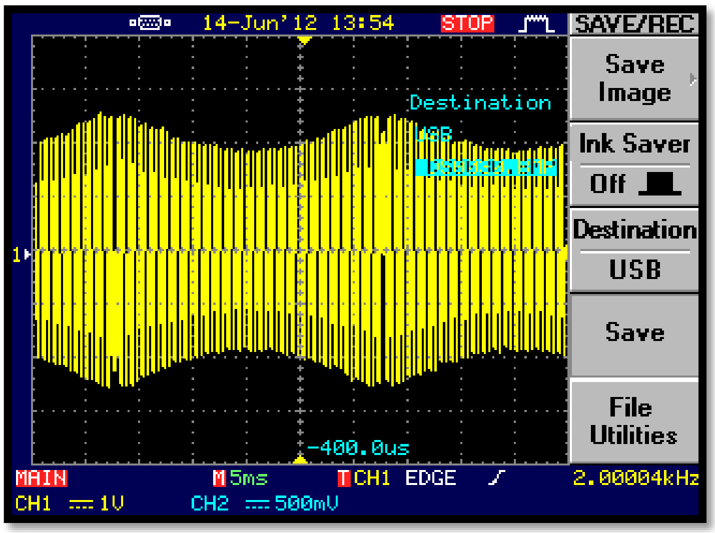


Figure 4.7.10 The AM signal with DSP constant = 1 , signal amplitude=0.3 using DSK6713.

Conclusion :

1- AM modulated signal has been obtained . It is simple to implement.

2- AM signal is very sensitive for error. The only one way to avoid the noise effect to happen is increasing power transmitted.

4.8 BPSK generation :

In PSK (Phase Shift Keying), phase of a carrier is changed between two values according to the binary signal level. The information about the bit stream is contained in the phase changes of the transmitted signal. For instance to transmit the signals “0” and ”1”,

PSK signals can be chosen as follows:

$\varphi_0(t) = A\sin(\omega_0 t + \theta_0)$

$\varphi_1(t) = A\sin(\omega_1 t + \theta_1)$, $0<t\leq Ts$

Here, θ1 and θ2 are constant phase shifts. For Binary PSK (BPSK) the state of θ1-θ0 = 180simplifies the modulator design. Moreover, π radian between phases of PSK signals will be most appropriate from error-performance point of view. For example, (θ0, θ1) phase values can be chosen as (0, π) or (π/2, 3π/2). Therefore, φ0(t) and φ1(t) waves can be written as follows:

$\varphi_0(t) = A\cos(\omega_0 t)$ $0<t\leq Ts$

$\varphi_0(t) = 0$ otherwise

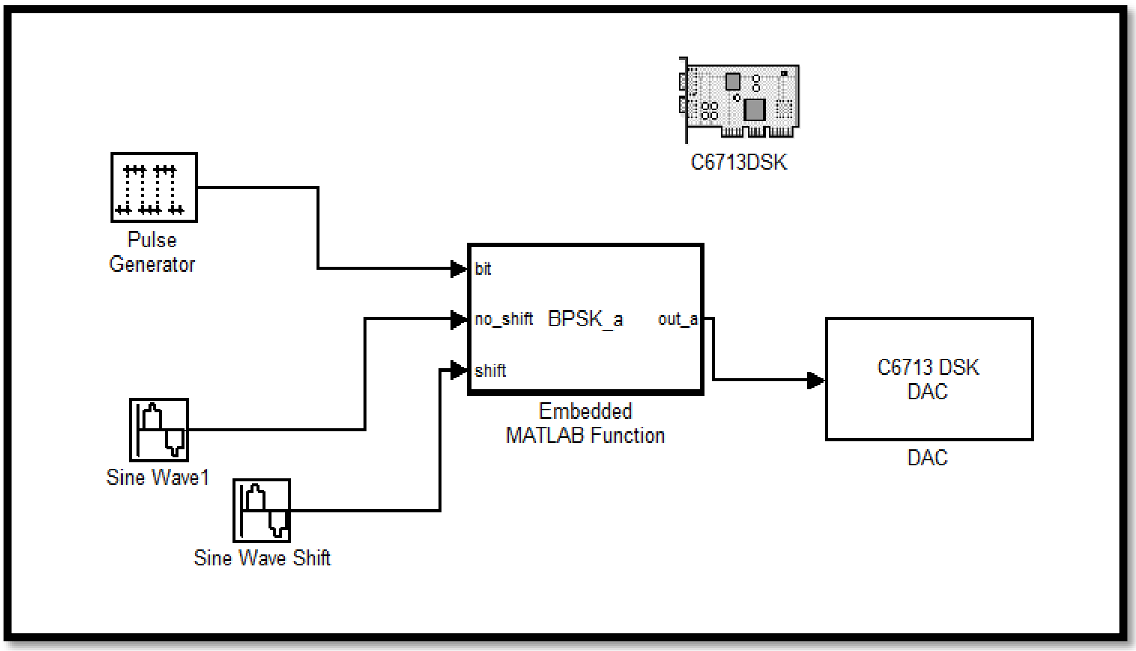
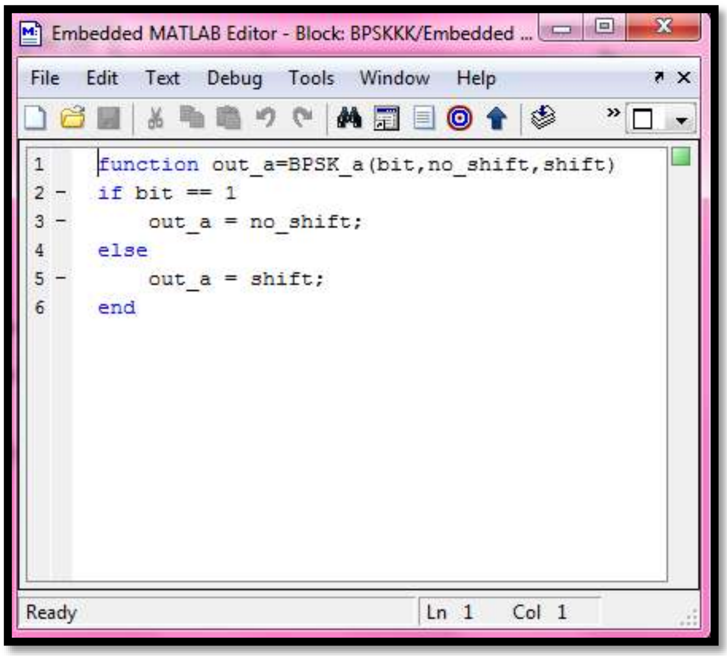


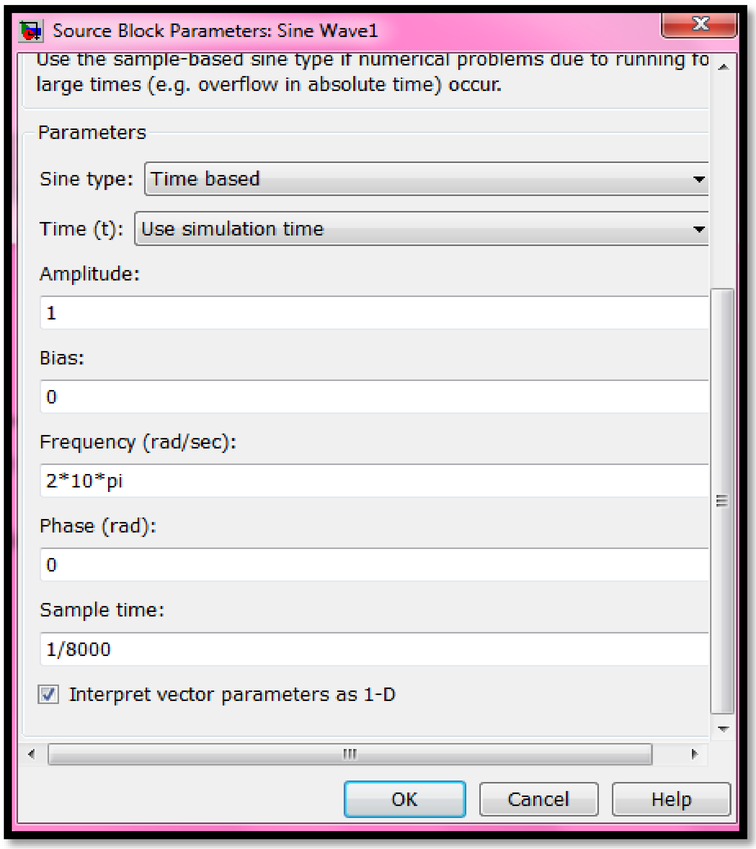
Fig 4.8.1 circuit diagram

In our experiment we use embedded matlab function. The Embedded MATLAB Function block can generate efficient embeddable code .here it generate the BPSk using the following command in fig4.8.2


Embedded MATLAB Editor - Block: BPSKKK/Embedded ...
File Edit Text Debug Tools Window Help
Ready
Ln 1 Col 1

```
function out_a=BPSK_a(bit,no_shift,shift)
if bit == 1
    out_a = no_shift;
else
    out_a = shift;
end
```

Fig 4.8.2 Embedded MATLAB Function



Use the sample-based sine type if numerical problems due to running fo
large times (e.g. overflow in absolute time) occur.

Parameters

Sine type: Time based

Time (t): Use simulation time

Amplitude: 1

Bias: 0

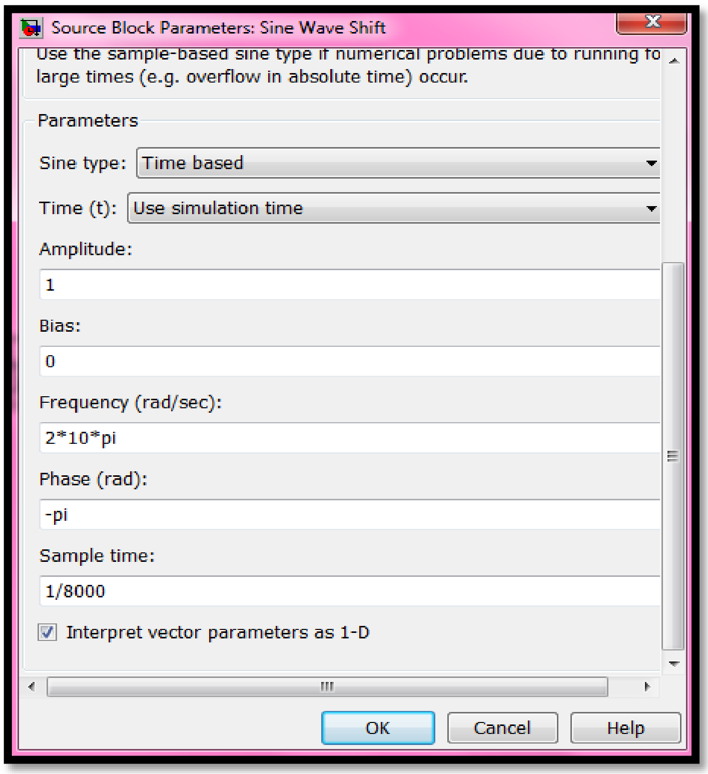
Frequency (rad/sec): 2*10*pi

Phase (rad): 0

Sample time: 1/8000

☑ Interpret vector parameters as 1-D

Fig 4.8.3 sine wave without shifting


Source Block Parameters: Sine Wave Shift

Use the sample-based sine type if numerical problems due to running fo
large times (e.g. overflow in absolute time) occur.

Parameters

Sine type: Time based

Time (t): Use simulation time

Amplitude:
1

Bias:
0

Frequency (rad/sec):
2*10*pi

Phase (rad):
-pi

Sample time:
1/8000

☑ Interpret vector parameters as 1-D

Fig 4.8.3 shifted sine wave

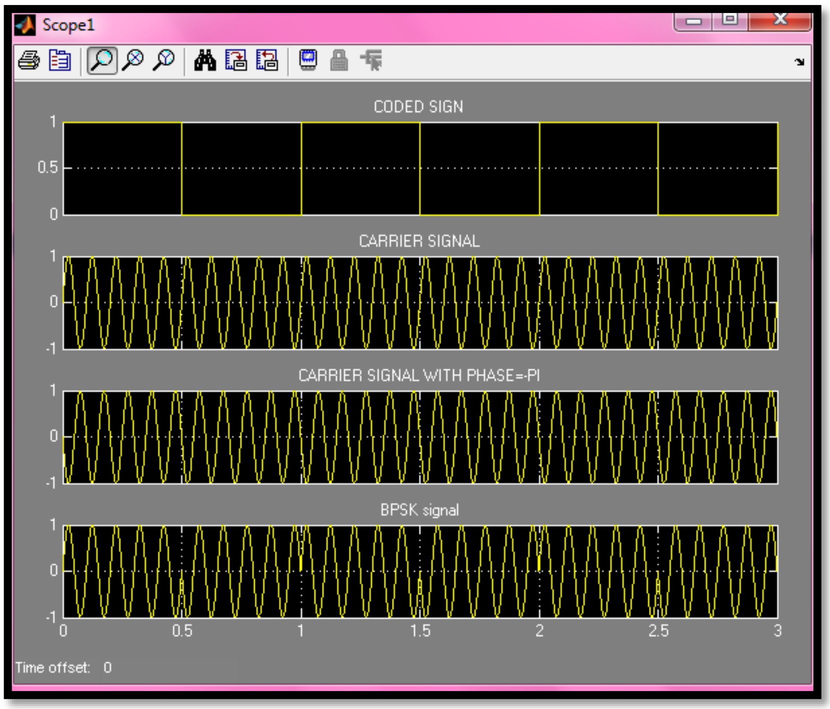


Fig 4.8.4 BPSK signal on oscilloscope

Fig 4.8.4 show the coded signal ,carrier signal without shift and one with shift ,also show the BPSK signal that we must obtain

The result :

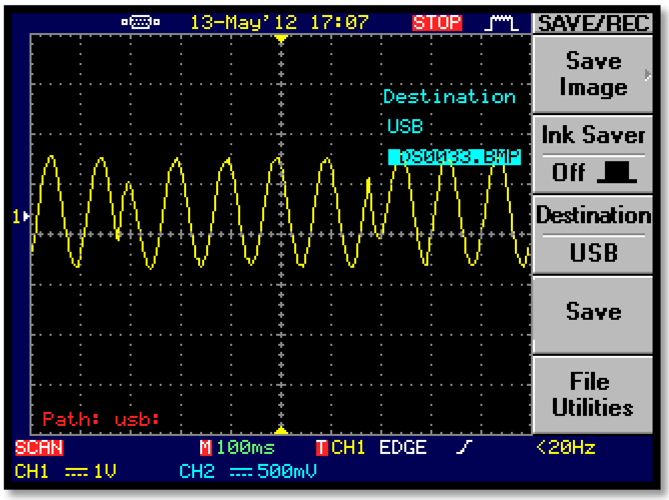


Fig 4.8.5 The BPSK signal

4.9 CONVOLUTION :

In this experiment we find the convolution of two square wave and get the result. By using a convolution block in simulink .

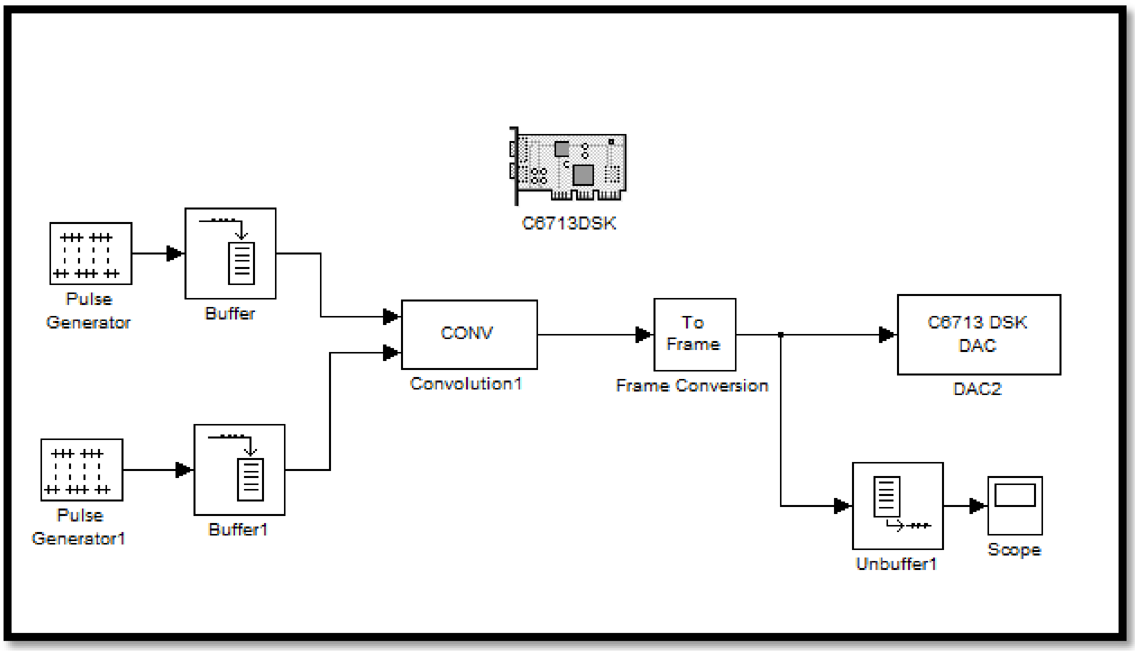
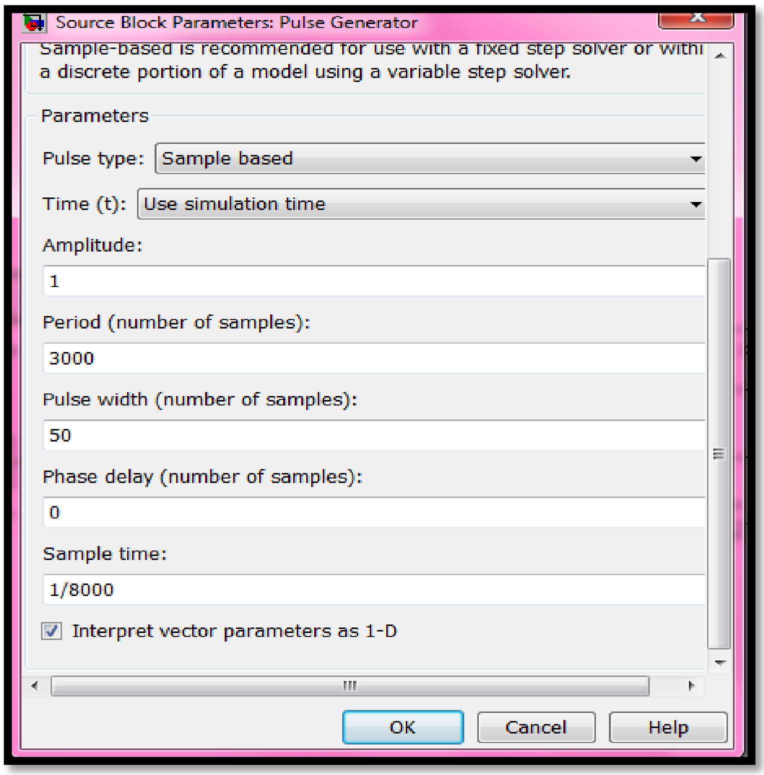


Fig 4.9.1 circuit diagram


Source Block Parameters: Pulse Generator

Sample-based is recommended for use with a fixed step solver or withi
a discrete portion of a model using a variable step solver.

Parameters

Pulse type: Sample based

Time (t): Use simulation time

Amplitude:

1

Period (number of samples):

3000

Pulse width (number of samples):

50

Phase delay (number of samples):

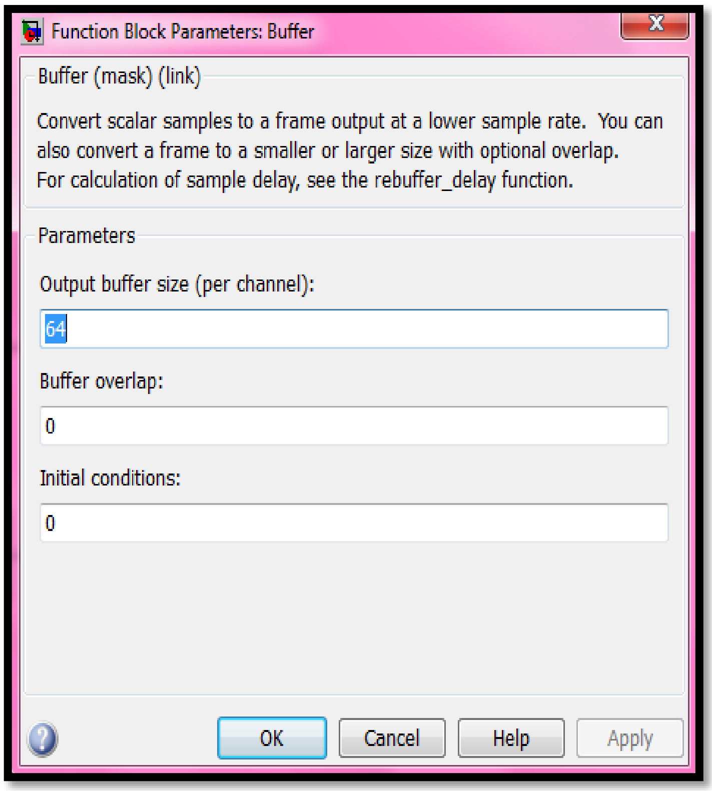
0

Sample time:

1/8000

☑ Interpret vector parameters as 1-D

Fig 4.9.2 pulse generator


Function Block Parameters: Buffer

Buffer (mask) (link)

Convert scalar samples to a frame output at a lower sample rate. You can also convert a frame to a smaller or larger size with optional overlap. For calculation of sample delay, see the rebuffer_delay function.

Parameters

Output buffer size (per channel):

64

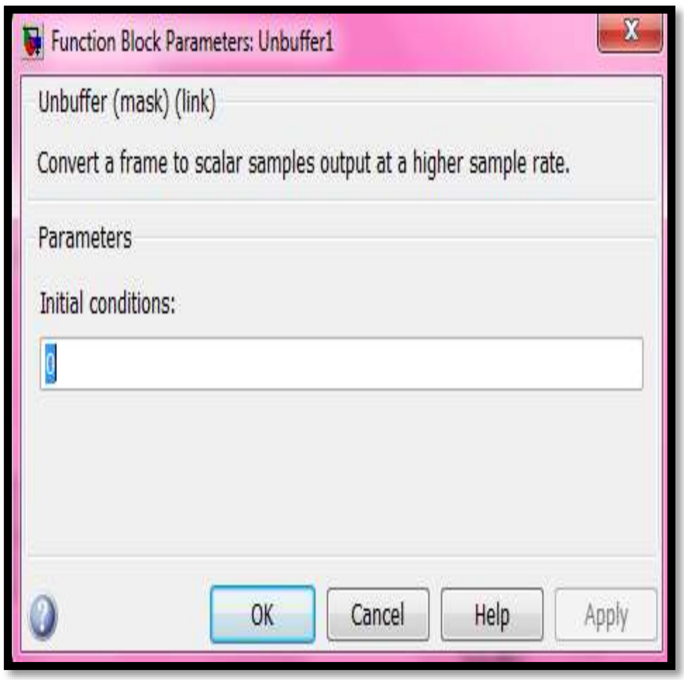
Buffer overlap:

0

Initial conditions:

0

Fig 4.9.3 Buffer block


Function Block Parameters: Unbuffer1

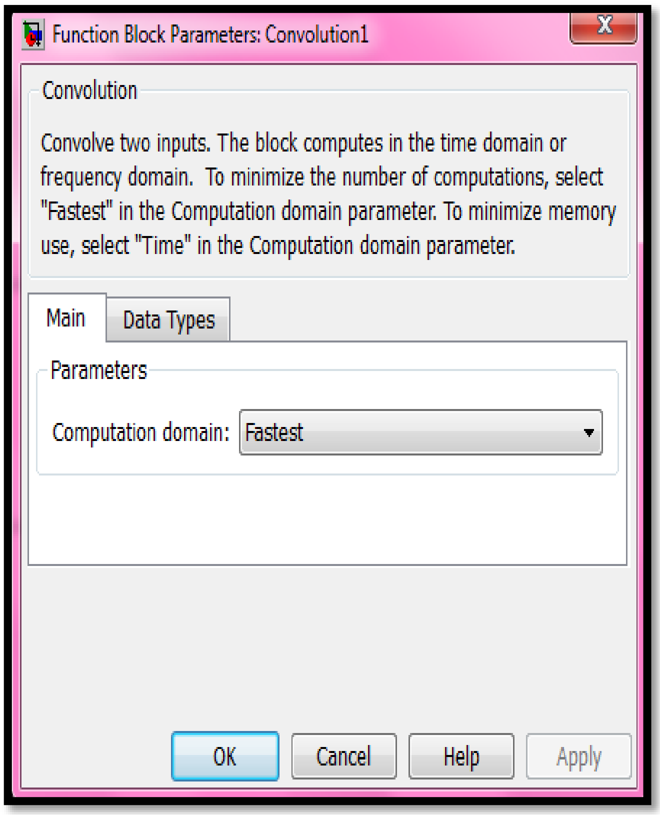
Unbuffer (mask) (link)

Convert a frame to scalar samples output at a higher sample rate.

Parameters

Initial conditions: 0

Fig 4.9.4 Unbuffer block


Function Block Parameters: Convolution1

Convolution

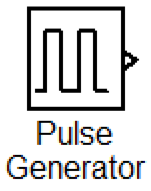
Convolve two inputs. The block computes in the time domain or frequency domain. To minimize the number of computations, select "Fastest" in the Computation domain parameter. To minimize memory use, select "Time" in the Computation domain parameter.

Main | Data Types

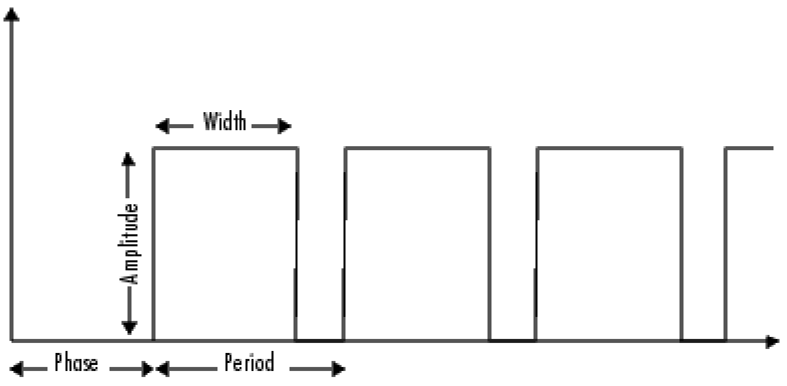
Parameters

Computation domain: Fastest

Fig 9.4.5 convolution block

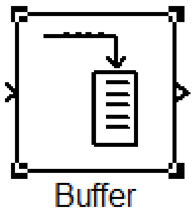


The Pulse Generator block generates square wave pulses at regular intervals. The block's waveform parameters, Amplitude, Pulse Width, Period, and Phase Delay, determine the shape of the output waveform. The following diagram shows how each parameter affects the waveform.

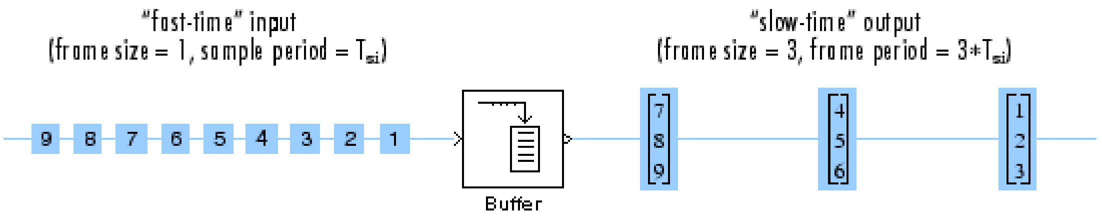


The Pulse Generator can emit scalar, vector, or matrix signals of any real data type. To cause the block to emit a scalar signal, use scalars to specify the waveform parameters. To cause the block to emit a vector or matrix signal, use vectors or matrices, respectively, to specify the waveform parameters. Each element of the waveform parameters affects the corresponding element of the output signal. For example, the first element of a vector amplitude parameter determines the amplitude of the first element of a vector output pulse. All the waveform parameters must have the same dimensions after scalar expansion. The data type of the output is the same as the data type of the Amplitude parameter.

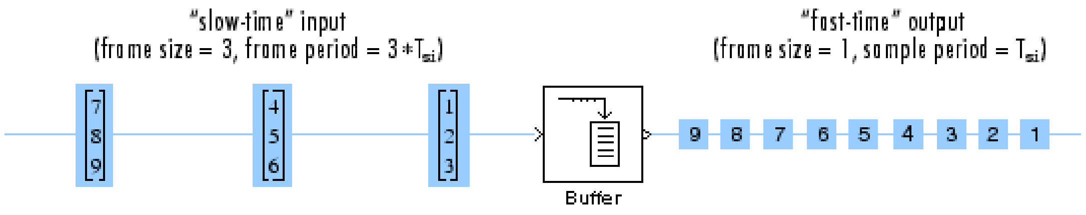
Buffer :



The Buffer block redistributes the input samples to a new frame size. Buffering to a larger frame size yields an output with a *slower* frame rate than the input, as illustrated below for scalar input.


"fast-time" input
(frame size = 1, sample period = T_{si})
"slow-time" output
(frame size = 3, frame period = 3*T_{si})
9 8 7 6 5 4 3 2 1
Buffer
7 8 9
4 5 6
1 2 3

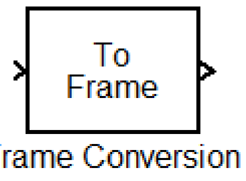
Buffering to a smaller frame size yields an output with a *faster* frame rate than the input, as illustrated below for scalar output.


"slow-time" input
(frame size = 3, frame period = 3*T_{si})
"fast-time" output
(frame size = 1, sample period = T_{si})
7 8 9
4 5 6
1 2 3
Buffer
9 8 7 6 5 4 3 2 1

The block coordinates the output *frame size* and *frame rate* of nonoverlapping buffers such that the sample period of the signal is the same at both the input and output: $T_{so} = T_{si}$.

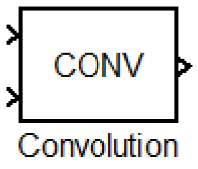
This block supports triggered subsystems when the block's input and output rates are the same.

Sample-based full-dimension matrix inputs are not accepted.



The Frame Conversion block passes the input through to the output and sets the output sampling mode to the value of the Sampling mode of output signal parameter, which can be either Frame-based or Sample-based. The output sampling mode can also be inherited from the signal at the Ref (reference) input port, which you make visible by selecting the Inherit output sampling mode from <Ref> input port check box.

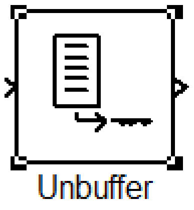
The Frame Conversion block does not make any changes to the input signal other than the sampling mode. In particular, the block does not rebuffer or resize 2-D inputs. Because 1-D vectors cannot be frame based, when the input is a length-*M* 1-D vector and the block is in Frame-based mode, the output is a frame-based *M*-by-1 matrix — that is, a single channel.



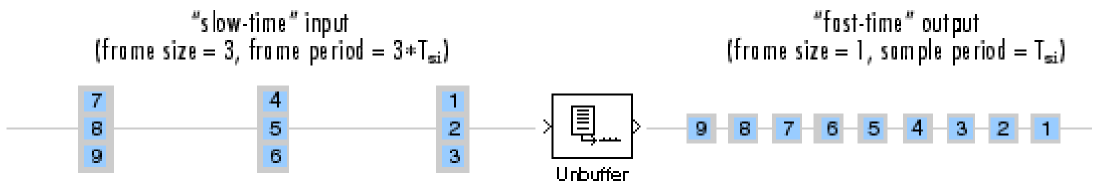
The Convolution block convolves the first dimension of a sample-based N-D input array *u*, with the first dimension of a sample-based N-D input array *v*. The block can also independently convolve a sample-based vector with the first-dimension of an N-D input array. For frame-based inputs, the Convolution block convolves analogous columns of an M_u-by-N input matrix *u* and an M_v-by-N input matrix *v*. The Convolution block can also independently convolve a single-channel frame-based column vector with each column of a multiple-channel frame-based matrix.

The frame status of both inputs to the Convolution block must be the same. The output of the block is always sample-based.

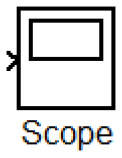
The Convolution block accepts real and complex floating-point and fixed-point inputs except for complex unsigned fixed-point inputs. Fixed-point signals are not supported for the frequency domain.



The Unbuffer block unbuffers an M_i-by-N frame-based input into a 1-by-N sample-based output. That is, inputs are unbuffered *row-wise* so that each matrix row becomes an independent time-sample in the output. The rate at which the block receives inputs is generally less than the rate at which the block produces outputs


"slow-time" input
(frame size = 3, frame period = 3*T_si)
7
8
9
4
5
6
1
2
3
Unbuffer
"fast-time" output
(frame size = 1, sample period = T_si)
9 8 7 6 5 4 3 2 1

The block adjusts the output rate so that the *sample period* is the same at both the input and output, T_{so}=T_{si}. Therefore, the output sample period for an input of frame size M_i and frame period T_{fi} is T_{fi}/M_i, which represents a *rate* M_i times higher than the input frame rate. In the example above, the block receives inputs only once every three sample periods, but produces an output once every sample period. To rebuffer frame-based inputs to a larger or smaller frame size, use the Buffer block.



The Scope block displays its input with respect to simulation time.

The Scope block can have multiple axes (one per port) and all axes have a common time range with independent *y*-axes. The Scope block allows you to adjust the amount of time and the range of input values displayed. You can move and resize the Scope window and you can modify the Scope's parameter values during the simulation.

the result of convolution tow square wave is triangular wave

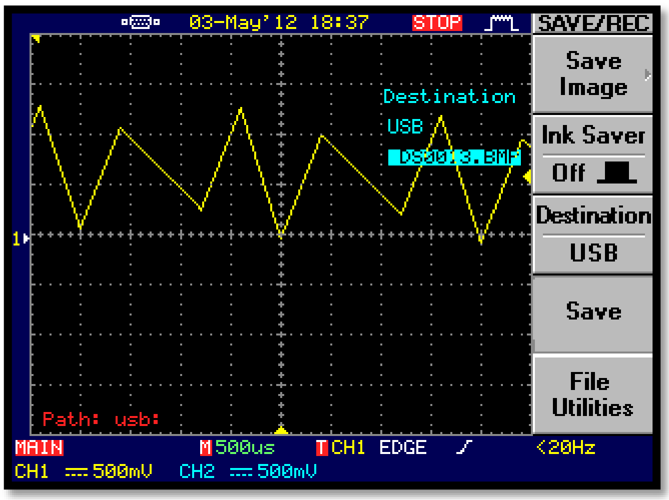


Fig 9.4.6 triangular wave (result of convolution)

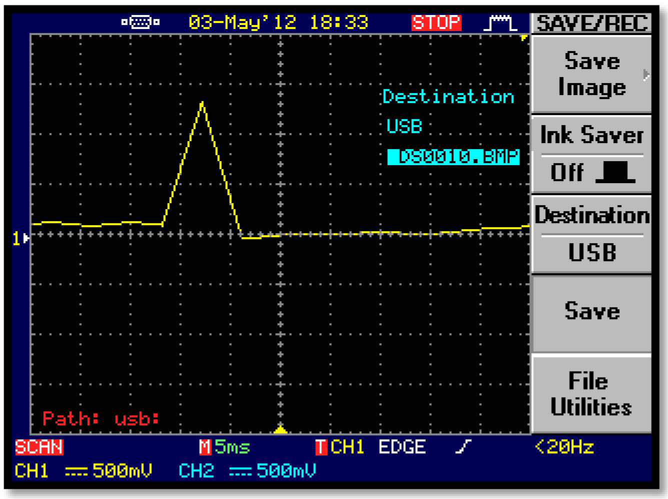


Fig 9.4.7 triangular wave (result for one square pulse)

General Conclusion :

In our project we study the application (experiment) for C6713 DSK , which dealing with DSP domain ,in order to ease the study of DSP theory , especially for students. This Kit work by Matlab program a2006 , simulink, in associations with the code composer studio .

Each experiment study a different subject in DSP , and we try to cover an critical subjects.

The first experiment is how to generate a sine wave from the C6713 DSK , so the KIT can generate all discreet signal we want by the matlab .

Another experiment ,to generate the square wave but at this time we use the basic of fourer series , so we find that every time we increase the number of odd terms we get a good shape of square wave.

In sampling experiment we test the range of sampling rate that the Kit can accept, and it is suitable for audio waves .

The experiments of digital filters(FIR) and (IIR) , explain the behavior of each type of filters , and the different types of both , and we apply it on human voices.

We generate echo signal depending on the basic of delay , that is the echo signal is the summation of the original signal and the delayed signal.

We also generate two types of modulation ,AM signal and binary phase shift keying (BPSK), depending on the definition of AM modulation and BPSK modulation .

Finally ,we take the convolution of two square waves and get a triangular wave.