DIGITAL SIGNAL PROCESSING LABORATORY

LAB MANUAL

Academic Year	**:**	**2019 - 2020**
Course Code	**:**	**AEC107**
Regulations	**:**	**IARE - R16**
Class	**:**	**VI Semester (ECE)**

Prepared by
Mr. K Chaitanya
Assistant Professor

Department of Electronics & Communication Engineering

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)
Dundigal – 500 043, Hyderabad

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal – 500 043, Hyderabad

Electronics & Communication Engineering

Vision
To produce professionally competent Electronics and Communication Engineers capable of effectively and efficiently addressing the technical challenges with social responsibility.
Mission
The mission of the Department is to provide an academic environment that will ensure high quality education, training and research by keeping the students abreast of latest developments in the field of Electronics and Communication Engineering aimed at promoting employability, leadership qualities with humanity, ethics, research aptitude and team spirit.
Quality Policy
Our policy is to nurture and build diligent and dedicated community of engineers providing a professional and unprejudiced environment, thus justifying the purpose of teaching and satisfying the stake holders. A team of well qualified and experienced professionals ensure quality education with its practical application in all areas of the Institute.
Philosophy
The essence of learning lies in pursuing the truth that liberates one from the darkness of ignorance and Institute of Aeronautical Engineering firmly believes that education is for liberation. Contained therein is the notion that engineering education includes all fields of science that plays a pivotal role in the development of world-wide community contributing to the progress of civilization. This institute, adhering to the above understanding, is committed to the development of science and technology in congruence with the natural environs. It lays great emphasis on intensive research and education that blends professional skills and high moral standards with a sense of individuality and humanity. We thus promote ties with local communities and encourage transnational interactions in order to be socially accountable. This accelerates the process of transfiguring the students into complete human beings making the learning process relevant to life, instilling in them a sense of courtesy and responsibility.

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal, Hyderabad - 500 043

Electronics & Communication Engineering

Program Outcomes	
PO1	**Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	**Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3	**Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4	**Conduct investigations of complex problems**: Use research-based knowledge and research methodsincludingdesignofexperiments,analysisandinterpretationofdata,andsynthesisofthe information to provide valid conclusions.
PO5	**Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6	**The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	**Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8	**Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	**Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary Settings.
PO10	**Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	**Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12	**Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.
Program Specific Outcomes	
PSO 1	**Professional Skills:** An ability to understand the basic concepts in Electronics & Communication Engineering and to apply them to various areas, like Electronics, Communications, Signal processing, VLSI, Embedded systems etc., in the design and implementation of complexsystems.
PSO 2	**Problem-Solving Skills:** An ability to solve complex Electronics and communication Engineering problems, using latest hardware and software tools, along with analytical skills to arrive cost effective and appropriate solutions.
PSO 3	**Successful Career and Entrepreneurship**: An understanding of social-awareness & environmental-wisdom along with ethical responsibility to have a successful career and to sustain passion and zeal for real-world applications using optimal resources as an Entrepreneur.

ATTAINMENT OF PROGRAM OUTCOMES & PROGRAM SPECIFIC OUTCOMES			
S. No.	**Experiment**	**Program Outcomes Attained**	**Program Specific Outcomes Attained**
1	a) Generation of linear convolution without using built infunction and the function conv in MATLAB b) Generation of circular convolution without using built in function inMATLAB	PO1, PO2	PSO1
2	Compute the Discrete Fourier Transform and IDFT with and without fft and ifft in MATLAB	PO1, PO2	PSO1
3	Implementation of Linear convolution using DFT (Overlap-add and Overlap-Save methods)	PO1, PO2	PSO1
4	Implementation of Decimation-in-time radix-2 FFT algorithm	PO1, PO2	PSO1, PSO2
5	Implementation of Decimation-in-frequency radix-2 FFT algorithm	PO1, PO2	PSO1
6	Implementation of IIR digital filter using Butterworth method and bilinear transformation	PO1, PO2, PO3	PSO1, PSO2
7	Implementation of IIR digital filter using Chebyshev (Type I and II) method	PO1, PO2, PO3	PSO1
8	Implementation of FIR digital filter using window (Rectangular, Hamming, Hanning, Bartlett) methods	PO1, PO2	PSO1
9	Implementation of FIR digital filter using frequency sampling method	PO1, PO2	PSO1
10	Implementation of optimum equiripple FIR digital filter using window methods	PO1, PO2, PO3	PSO1
11	DTMF Tone Generation and Detection Using Goertzel Algorithm	PO1, PO2	PSO1
12	Implementation of sampling rate conversion by decimation, interpolation and a rational factor using MATLAB	PO1, PO2	PSO1
13	a) Implementation ofDFT b) Sine wave generation using lookup table with valuesgenerated fromMATLAB	PO1, PO2	PSO1, PSO2
14	IIR and FIR Filter Implementation using DSP Kits.	PO1, PO2, PO3	PSO1

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal, Hyderabad - 500 043

Certificate

This is to Certify that it is a bonafied record of Practical work donebySri/Kum.__________________________________bearingthe Roll No. __________________________ of _______________ Class ____Branch in the ________________________________ laboratory during the Academicyear________________________under oursupervision.

Head oftheDepartment **LectureIn-Charge**

ExternalExaminer **InternalExaminer**

Electronics and Communication Engineering

Course Overview:

This course provides practical handson exposure to communication system building blocks. The objective of this lab is to teach students Amplitude and Frequency modulation. Generation and detection of AM, DSB-SC, SSB and FM signals. Time-division multiplexing systems, Frequency division multiplexing systems. Sampling THEORY, Pulsemodulation.

Course Out-Come:

I. Implementation of convolution inMATLAB.

II. Implementation of digital signal processing algorithms in MATLAB andC.

III. Understand the real-time operation of digitalfilters.

IV. Analyze the Multirate signal processingalgorithms.

Electronics & Communication Engineering

INSTRUCTIONS TO THE STUDENTS

1. Students are required to attend alllabs.
2. Students should work individually in the hardware and softwarelaboratories.
3. Students have to bring the lab manual cum observation book, record etc along with them whenever they come for labwork.
4. Should take only the lab manual, calculator (if needed) and a pen or pencil to the work area.
5. Should learn the prelab questions. Read through the lab experiment to familiarize themselves with the components and assemblysequence.
6. Should utilize 3 hour's time properly to perform the experiment and to record the readings. Do the calculations, draw the graphs and take signature from theinstructor.
7. If the experiment is not completed in the stipulated time, the pending work has to be carried out in the leisure hours or extendedhours.
8. Should submit the completed record book according to the deadlines set up by the instructor.
9. For practical subjects there shall be a continuous evaluation during the semester for 25 sessional marks and 50 end examinationmarks.
10. Out of 25 internal marks, 15 marks shall be awarded for day-to-day work and 10 marks to be awarded by conducting an internal laboratorytest.

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)
Dundigal – 500 043, Hyderabad

DIGITAL SIGNAL PROCESSING LAB SYLLABUS

Recommended Systems/Software Requirements:

Intel based desktop PC with minimum of 166 MHZ or faster processor with at least 4GB RAM and 500GB free disk space. MATLAB and hardware related to experiments. C6713 DSK Code Composer Studio

S.No.	List of Experiments	Page No.	Date	Remarks
1.	a) Generation of linear convolution without using built in function and the function conv inMATLAB b) Generation of circular convolution without using built in function inMATLAB			
2.	Compute the Discrete Fourier Transform and IDFT with and without fft and ifft in MATLAB			
3.	Implementation of Linear convolution using DFT (Overlap-add and Overlap-Save methods)			
4.	Implementation of Decimation-in-time radix-2 FFT algorithm			
5.	Implementation of Decimation-in-frequency radix-2 FFT algorithm			
6.	Implementation of IIR digital filter using Butterworth method and bilinear transformation			
7.	Implementation of IIR digital filter using Chebyshev (Type I and II) method			
8.	Implementation of FIR digital filter using window (Rectangular, Hamming, Hanning, Bartlett) methods			

S.No.	List of Experiments	Page No.	Date	Remarks
9.	Implementation of FIR digital filter using frequency sampling method			
10.	Implementation of optimum equiripple FIR digital filter using window methods			
11.	DTMF Tone Generation and Detection Using Goertzel Algorithm			
12.	Implementation of sampling rate conversion by decimation, interpolation and a rational factor using MATLAB			
13.	a) Implementation ofDFT b) Sine wave generation using lookup table withvalues generated fromMATLAB			
14.	IIR and FIR Filter Implementation using DSP Kits			

INSTITUTE OF AERONAUTICAL ENGINERING

(Autonomous)

DUNDIGAL, HYDERABAD– 500 043

DIGITAL SIGNAL PROCESSING LAB

Work Book

Name of the Student			
Roll No.			
Branch			
Class		**Section**	

ELECTRONICS AND COMMUNICATION ENGINEERING

Certificate

This is to certify that it is a bonafide record of practical work done by Mr./Ms.____, Reg. No.______________________in the Digital Signal Processing Laboratory in_____semesterof_____year during20____-20_____.

LAB INCHARGE

CONTENTS

5. FIR filter using Rectangular window (lowpass)
6. FIR filter using Kaiser Window (highpass)
7. IIR filter using Butterworth Approximation (lowpass)

Annexure –I Viva Questions

Experiment No. 1

1. a)Generation of linear convolution without using built in function andthe function convolution inMATLAB

```
clc;

close all

clearall

x=input('Enterx:    ')

h=input('Enterh:    ')

m=length(x);

n=length(h);

X=[x,zeros(1,n)];

H=[h,zeros(1,m)];

for i=1:n+m-1

Y(i)=0;

for j=1:m

if(i-j+1>0)

Y(i)=Y(i)+X(j)*H(i-j+1);

else

end

ende

nd Y
```

```
stem(Y);

ylabel('Y[n]');

xlabel('---- >n');

title('Convolution of Two Signals without conv function');
```

(b) Generation of circular convolution without using built in function inMATLAB

```
Clc;

close all
clear all
x=input('Enterx:    ')
h=input('Enterh:    ')
m=length(x);
n=length(h);
X=[x,zeros(1,n)];
H=[h,zeros(1,m)];
for i=1:n+m-1
Y(i)=0;
for j=1:m
if(i-j+1>0)
Y(i)=Y(i)+X(j)*H(i-j+1);
else
end
ende
nd Y
stem(Y);
ylabel('Y[n]');
xlabel('----->n');
title('Convolution of Two Signals without conv function');
```

Experiment No. 2

2.Compute the Discrete Fourier Transform and IDFT with and without fft and ifft in MATLAB

AIM:

To find the DFT and IDFT of a given sequence

TOOLS REQUIRED:

1. Mat labsoftware
2. Personalcomputer

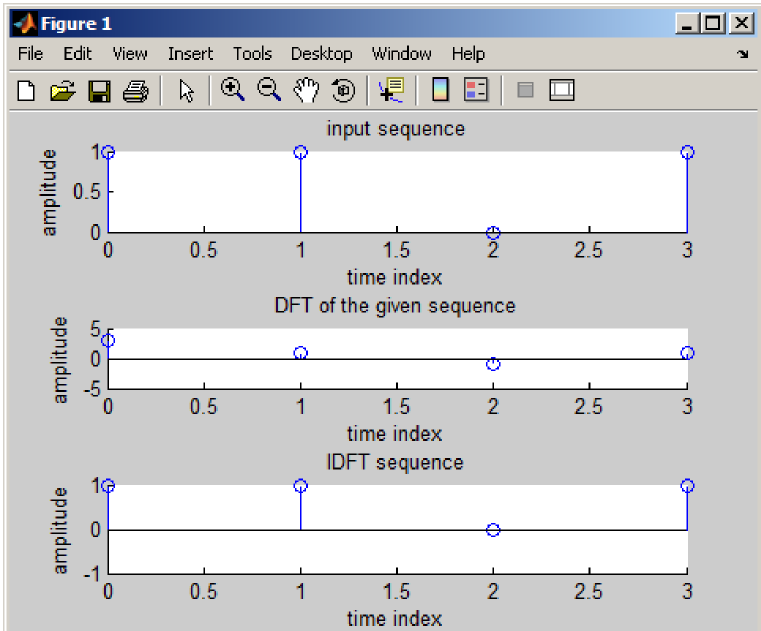
PROGRAM:

```
clc;
close all;
clearall;
xn=input('Enter the sequence x(n)'); %Get the sequence from user
ln=length(xn);                    %find the length of thesequence
xk=zeros(1,ln);                          %initilise an array of same size as that of inputsequence
ixk=zeros(1,ln);                          %initilise an array of same size as that of input
sequence

%code block to find the DFT of the sequence
%-----------------------------------------------------------
for k=0:ln-1
  for n=0:ln-1
     xk(k+1)=xk(k+1)+(xn(n+1)*exp((-i)*2*pi*k*n/ln));
  end
end
%-----------------------------------------------------------
%code block to plot the input sequence
%-----------------------------------------------------------
t=0:ln-1;
subplot(311);
stem(t,xn);
ylabel ('Amplitude');
xlabel ('Time Index');
```

```
title('Input Sequence');
%---------------------------------------------------------------
magnitude=abs(xk);                    % Find the magnitudes of individual DFTpoints

%code block to plot the magnitude response
%------------------------------------------------------------
t=0:ln-1;
subplot(312);
stem(t,magnitude);
ylabel ('Amplitude');
xlabel ('K');
title('Magnitude Response');
%------------------------------------------------------------
% Code block to find the IDFT of the sequence
%------------------------------------------------------------
for n=0:ln-1
   for k=0:ln-1
      ixk(n+1)=ixk(n+1)+(xk(k+1)*exp(i*2*pi*k*n/ln));
   end
end
ixk=ixk/ln;
%------------------------------------------------------------
%code block to plot the input sequence
t=0:ln-1;
subplot(313);
stem(t,ixk);ylabel ('Amplitude');xlabel ('Time Index');title('IDFT sequence');
```

OUTPUT AND WAVEFORMS:

```
enter the input sequence[1 1 0 1]
   3.0000              1.0000            -1.0000 - 0.0000i    1.0000


ixk =

   1.0000 - 0.0000i    1.0000            -0.0000 + 0.0000i    1.0000 + 0.0000i

>>
```



Experiment No. 3

3. Implementation of Linear convolution using DFT (Overlap-add and Overlap-Savemethods)

```
a)Over lap add method
clc
clear all
close all
x=[1 2 3 4 5 6 7 8 9 3 5 6 7];
h=[2 2 1];
% x=input('enterx');
% h=input('enterh');
% L=input('enter L')
M=length(h)
lx=length(x)
L=5;
r=rem(lx,L);
x1=[x zeros(1,L-r)];
lx1=length(x1);
nr=length(x1)/L;
h1=[h zeros(1,L-1)];
for k=1:nr
    M1(k,:)=x1(((k-1)*L+1):k*L);
    M2(k,:)=[M1(k,:) zeros(1,M-1)];
    M3(k,:)=ifft(fft(M2(k,:)).*fft(h1));
    M4(k,:)=[zeros(1,(k-1)*L) M3(k,:) zeros(1,(nr-k)*L)];
end
y=sum(M4)


(b) Over lap save method
clc;
clear all;
x=input('Enter 1st sequence X(n)= ');
h=input('Enter 2nd sequence H(n)= ');
L=input('Enter length of each block L = ');

% Code to plot X(n)
subplot (2,2,1);
stem(x);
stem(x,'blue');
xlabel('n----->');
ylabel('Amplitude------>');
title('X(n)');

%Code to plot H(n)
subplot (2,2,2);
stem(h);
stem(h,'black');
xlabel('n----->');
```

```
ylabel('Amplitude------>');
title(' H(n)');

% Code to perform Convolution using Overlap Save Method
M=length(h);
lx=length(x);
r=rem(lx,L);
x1=[x zeros(1,L-r)];
nr=(length(x1))/L;
h1=[h zeros(1,L-1)];
for k=1:nr
    Ma(k,:)=x1(((k-1)*L+1):k*L)
    if k==1
        Ma1(k,:)=[zeros(1,M-1) Ma(k,:)];
    else
        Ma1(k,:)=[Ma(k-1,(L-M+2):L) Ma(k,:)];
    end
     Ma2(k,:)=ifft(fft(Ma1(k,:)).*fft(h1));
end
Ma3=Ma2(:,M:(L+M-1));
y1=Ma3';
y=y1(:)'

% Representation of the ConvoledSignal
subplot(2,2,3:4);
stem(y,'red');
xlabel('n----->');
ylabel('Amplitude------>');
title ('ConvolvedSignal');
```

Experiment No. 4

4. Implementation of Decimation-in-time radix-2 FFTalgorithm

AIM:

FFT of a sequence using DIT-FFT method

TOOLS REQUIRED:

Mat lab software

Personal computer

% Direct computation of FFT

```
x=[1 1 0 0];
N=4;
y=fft(x,N);
stem(abs(y));
ylabel ('Amplitude');
xlabel ('N');
title('Magnitude Response');
```

%Matlab Program for FFT using DIT algorithm

```
clc; clear all; close all;
x=input('enter x[n]:');
N=length(x);
levels=nextpow2(N);
xn=[x,zeros(1,(2^levels)-N)];
x=bitrevorder(xn)
N=length(xn);
tw=cos(2*pi*(1/N)*(0:N/2-1))-j*sin(2*pi*(1/N)*(0:N/2-1));
for level=1:levels;
  L=2^level;
  twlvl=tw(1:N/L:N/2);
  for k=0:L:N-L;
```

```
        for n=0:L/2-1;
            A=x(n+k+1);
            B=x(n+k+(L/2)+1)*twlvl(n+1);
            x(n+k+1)=A+B;
            x(n+k+(L/2)+1)=A-B;
         end
    end
    x
end
XK=x
n=0:N-1;
subplot(2,2,1);stem(n,xn);title('x(n)');xlabel('n');ylabel('Amplitude');
subplot(2,2,2);stem(n,real(XK));title('Real part of X(K)');xlabel('n');ylabel('Amplitude');
subplot(2,2,3);stem(n,imag(XK));title('Imag part of X(K)');xlabel('n');ylabel('Amplitude');
```

OUTPUT AND WAVEFORM:

```
enter x[n]:[1 2 3 4 4 3 2 1]

XK =

20.0000     -5.8284-2.4142i     0     -0.1716-0.4142i     0     -0.1716 +0.4142i

0     -5.8284 +2.4142i
```

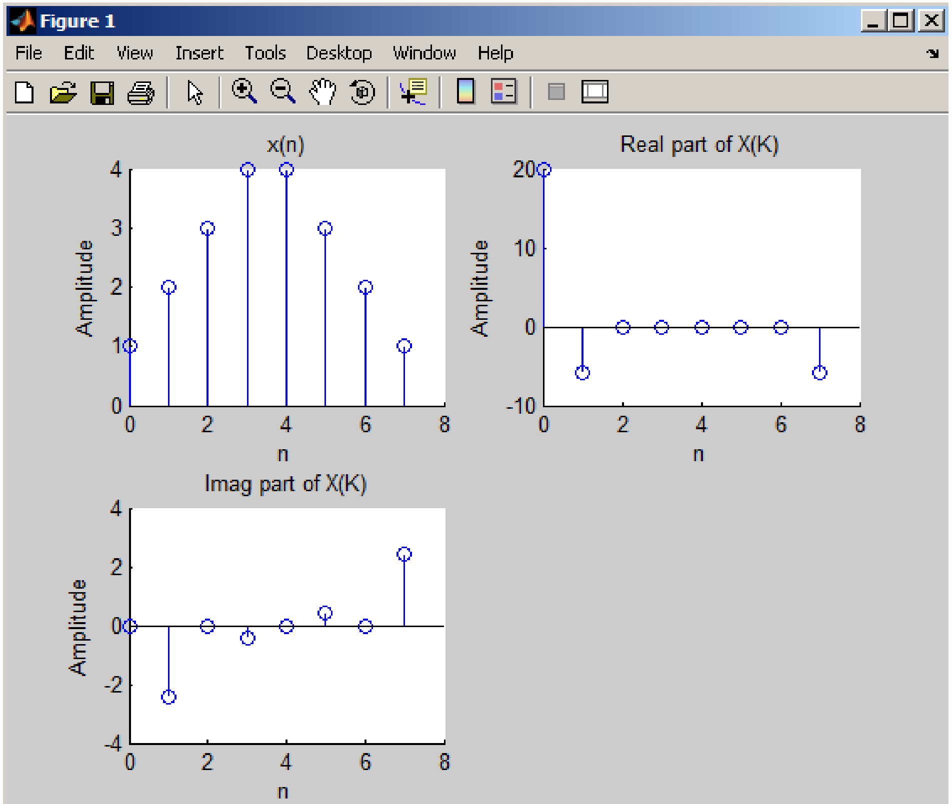


Figure 1
File Edit View Insert Tools Desktop Window Help
x(n)
Amplitude
n
Real part of X(K)
Amplitude
n
Imag part of X(K)
Amplitude
n

Experiment No. 5

5 Implementation of Decimation-in-frequency radix-2 FFT algorithm

AIM:

Implementation of Decimation-in-frequency radix-2 FFT algorithm

Tools Required:

Mat lab software

Personal computer

PROGRAM:

Clc;

Clear all;

Close all;

```
DECIMATION IN FREQUENCY [DIF] ALGORITHM

function q=dif(x)

t=nextpow2(length(x));           %Calculate the ndearest exponent of 2

j=[x zeros(1,(2^t)-length(x))] ;% zeropadding

N=length(j);                     % Length of paddedstructure

S=log2(N);                       %stages

for stage=S:-1:1

a=1;

b=1+2^(stage-1);                 %Initialise a and b for each stage

n=0;

while(n<=2^(stage-1)-1   && a<=N &&b<=N)

 l=(n).*(2^(S+1-stage))./2;

 e=exp((-1i)*2*pi*l/(16));            %Twiddle factor

 y=j(a)+j(b);

 z=(j(a)-j(b)).*e;                    % Butterfly structure

 j(a)=y;
```

```
        j(b)=z;
        a=a+1;                        % Increment a,b and n

        b=b+1;

        n=n+1;

        if (stage==1)                 % Discontinuity in the butterfly
structure

          if(rem(1,a)==1)             % in a particular stage

          a=a+2^(stage-1);

          b=b+2^(stage-1);

          n=0;

          end

        end


        if(stage~=1)

          if(rem(a,2^(stage-1))==1)

          a=a+2^(stage-1);

          b=b+2^(stage-1);

          n=0;

          end

        end

       end

end

j=bitrevorder(j);                   % Bit reverse the output sequence

disp(j);

q=j;

end
```

WAVEFORMS:

EMPTY SPACE FOR CALCULATIONS

Experiment No. 6

6) Implementation of IIR digital filter using Butterworth method and bilinear transformation

a) To Design a Butterworth High pass filter for the given specifications usingMatlab.

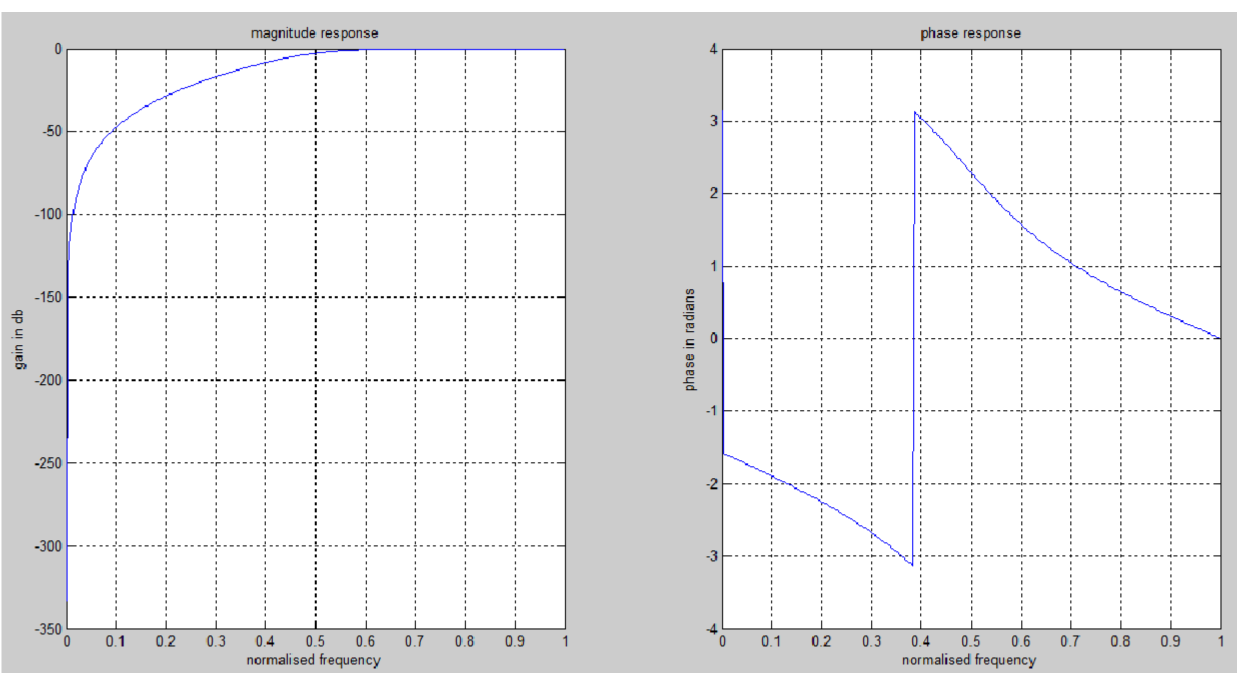
```
%To design a Butterworth Highpass filter for the given specifications

clf;

alphap=input('enter pass attenuation in db=');%passband attenuation in db
alphas=input('enter stopband attenuation in db=');% stopband attenuation in

dbfp=input('enter passband frequency in hz='); % passband frequency in hz

fs=input('enter stopband frequency in hz='); % stopband frequency in hz

F=input('enter sampling frequency in hz='); % sampling frequency in

hzomp=2*fp/F; %frequency in radians

oms=2*fs/F;

%to find cutoff frequency and order of the filter

[n,wn]=buttord(omp,oms,alphap,alphas);

%system function of the filter

[b,a]=butter(n,wn,'high');

w=0:0.01:pi;

[h,om]=freqz(b,a,w,'whole');

m=20*log10(abs(h));

an=angle(h);

subplot(1,2,1);

plot(om/pi,m);

grid;

xlabel('normalised frequency');

ylabel('gain in db');
```

```
title('magnitude response');

subplot(1,2,2);

plot(om/pi,an);

grid;

xlabel('normalised frequency');

ylabel('phase in radians');

title('phase response');

disp(b);

disp(a);
```

INPUTS:

enter pass attenuation in db=.4
enter stopband attenuation in db=30
enter passband frequency in hz=800
enter stopband frequency inhz=400
enter sampling frequency in hz=2000

OUTPUT WAVEFORMS:



B=0.0265 -0.1058   0.1587 -0.1058   0.0265
A= 1.0000   1.2948   1.0206   0.3575   0.0550

Experiment No. 7

7. Implementation of IIR digital filter using Chebyshev (Type I andII) method

To Design a Chebyshev-I High pass filter for the given specifications using Matlab.

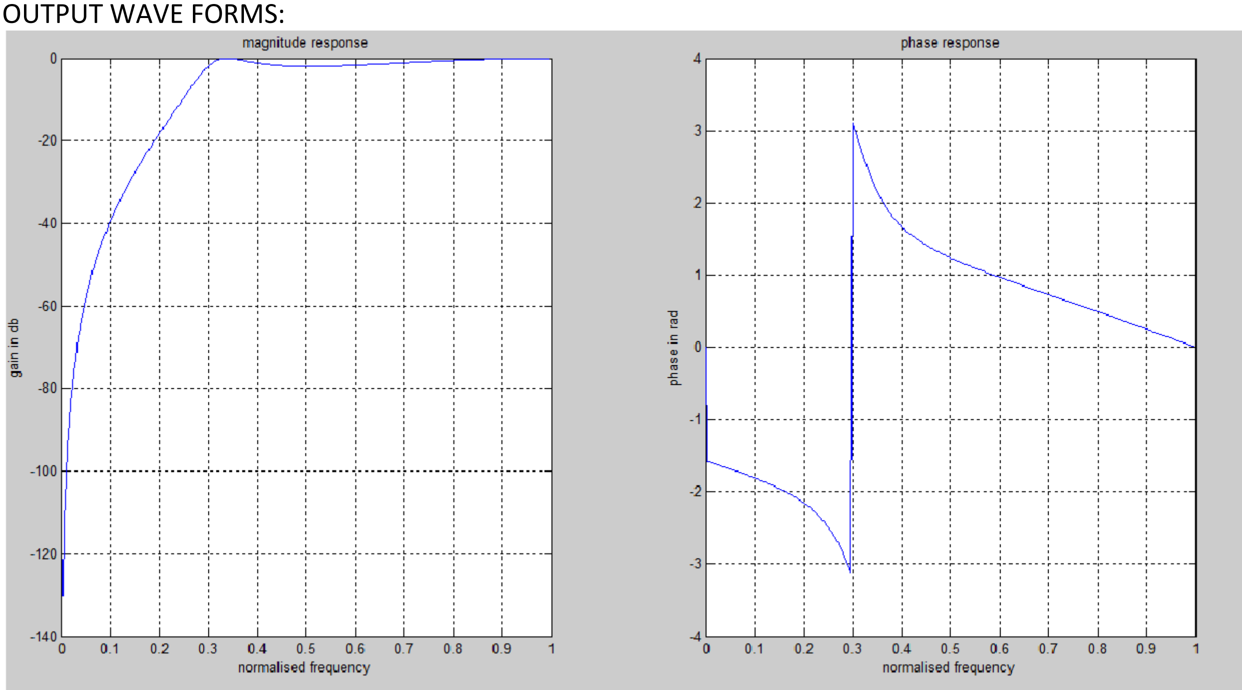
```
% To design a chebyshev-1 Hihpass filter for the given specifications

clf;

aphap=input('passband attenuation in db='); %passband attenuation in

dbalphas=input('stopband attenuation in db=')% stopband attenuation in

dbwp=.3*pi;% passband frequency in rad

ws=.2*pi;% stopband frequency in rad

%order and cutoff frequency of the filter

[n,wn]=cheb1ord(wp/pi,ws/pi,alphap,alphas);

%system function of the filter

[b,a]=cheby1(n,alphap,wn,'high');

w=0:0.01:pi;

[h,ph]=freqz(b,a,w);

m=20*log10(abs(h));

an=angle(h);

subplot(1,2,1);

plot(ph/pi,m);

grid;

xlabel('normalised frequency');

ylabel('gain in db');

title('magnitude response');

subplot(1,2,2);
```

```
plot(ph/pi,an);

grid;

xlabel('normalised frequency');

ylabel('phase in rad');

title('phase  response');

disp(b);

disp(a);
```

INPUTS:

passband attenuation in db=1
stopband attenuation in db=15

OUTPUT WAVE FORMS:



B=0.2790 -0.8371 0.8371 -0.2790
A=1.0000 -0.7794 0.5677 0.1150

Experiment No. 8

8)Implementation of FIR digital filter using window (Rectangular, Hamming, Hanning, Bartlett) methods

AIM:

To Write a Matlab program of FIR Low pass and high pass filter using rectangular, Hanning Hamming, Blackman and Kaiser window.

.

TOOL:

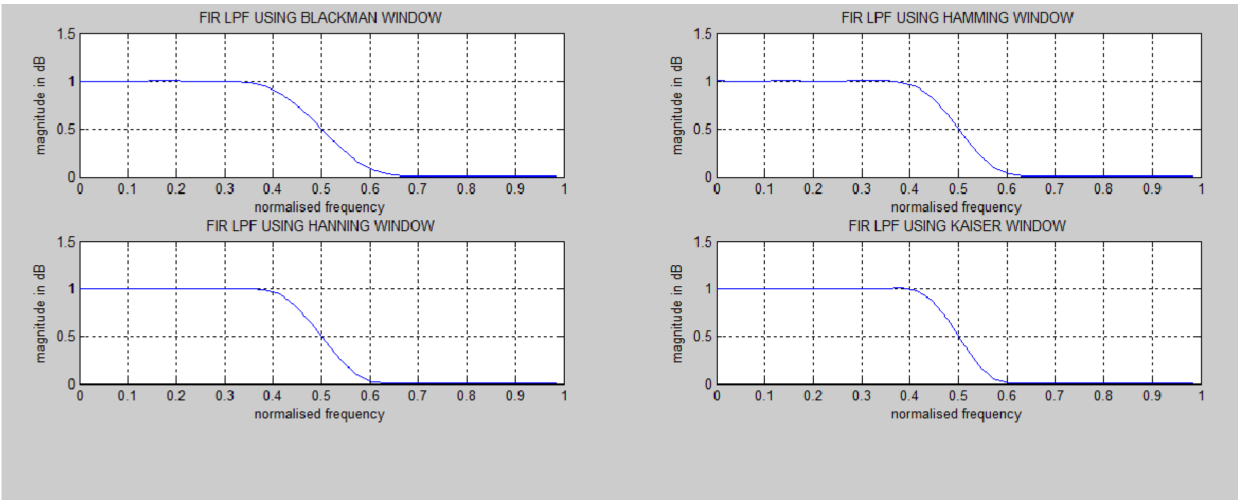
MATLAB Software 9.0

PROGRAM:

```
%MATLAB program of FIR Low pass filter using Hanning %Hamming, Blackman and Kaiser window

clf;

wc=.5*pi;

N=25;

w=0:0.1:pi;

b=fir1(N,wc/pi,blackman(N+1));

h=freqz(b,1,w);

 subplot(3,2,1)

plot(w/pi,abs(h))

grid;xlabel('normalised frequency');

ylabel('magnitude in dB')

title('FIR LPF USING BLACKMAN WINDOW')

b=fir1(N,wc/pi,hamming(N+1));

h=freqz(b,1,w);
```

```
subplot(3,2,2)

plot(w/pi,abs(h));

grid;

xlabel('normalised frequency');

ylabel('magnitude in dB')

title('FIR LPF USING HAMMING WINDOW')

b=fir1(N,wc/pi,hanning(N+1));

h=freqz(b,1,w);

subplot(3,2,3)

plot(w/pi,abs(h));

grid;

xlabel('normalised frequency');

ylabel('magnitude in dB')

title('FIR LPF USING HANNING WINDOW')

b=fir1(N,wc/pi,kaiser(N+1,3.5));

h=freqz(b,1,w);

subplot(3,2,4)

plot(w/pi,abs(h));

grid;

xlabel('normalised frequency');

ylabel('magnitude in dB')

title('FIR LPF USING KAISER WINDOW')
```

OUTPUT WAVEFORMS:



PROGRAM:

```
%FIR Filter design window techniques
clc;
clear all;
close all;
rp=input('enter passband ripple');
rs=input('enter the stopband ripple');
fp=input('enter passband freq');
fs=input('enter stopband freq');
f=input('enter sampling freq ');
beta=input('enter beta value');
wp=2*fp/f; ws=2*fs/f;
num=-20*log10(sqrt(rp*rs))-13;
dem=14.6*(fs-fp)/f;
n=ceil(num/dem);
n1=n+1; if(rem(n,2)~=0) n1=n; n=n-1;
end
c=input('enter your choice of window function 1. rectangular 2. triangular 3.kaiser: \n ');
if(c==1) y=rectwin(n1);
   disp('Rectangular window filter response');
end
```

```
if (c==2) y=triang(n1);
   disp('Triangular window filter response');
end
if(c==3) y=kaiser(n1,beta);
   disp('kaiser window filter response');
end
%HPF
b=fir1(n,wp,'high',y);
[h,o]=freqz(b,1,256);
m=20*log10(abs(h));
plot(o/pi,m);
title('HPF');
ylabel('Gain in dB-->');
xlabel('(b) Normalized frequency-->');
```

INPUT:

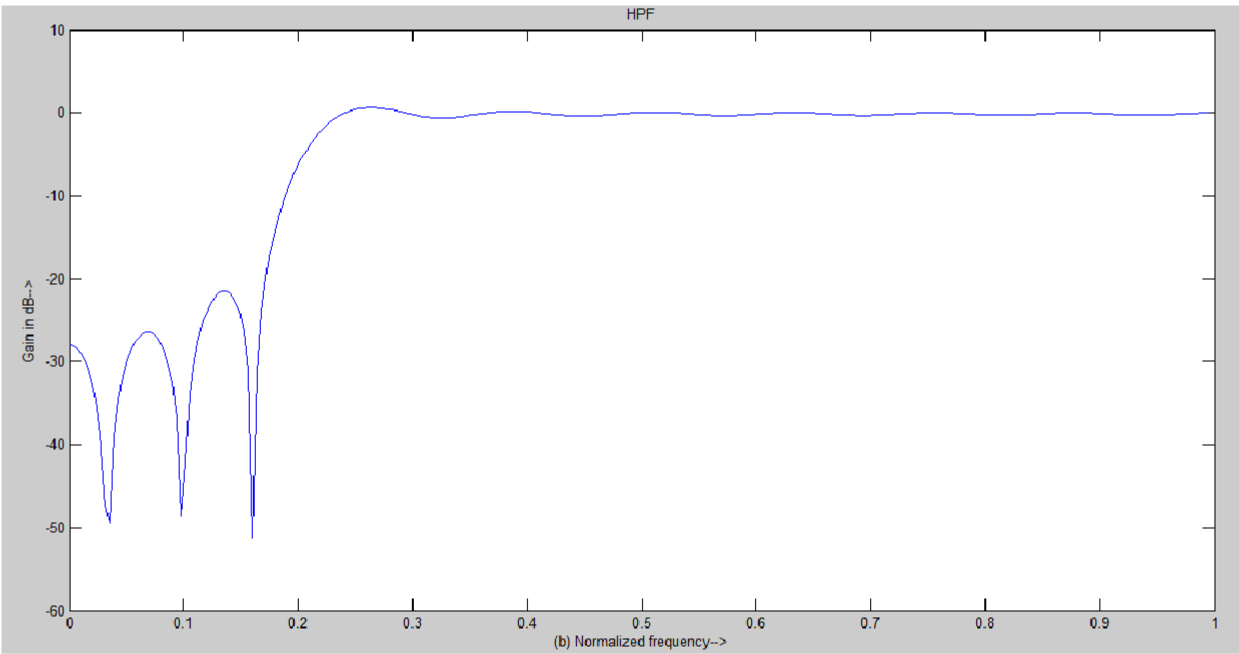
```
enter passband ripple:0.02
enter the stopband ripple:0.01
enter passband freq:1000
enter stopband freq:1500
enter sampling freq: 10000
enter beta value:5
```

OUTPUT WAVEFORM:

enter your choice of window function 1. rectangular 2. triangular 3.kaiser:

2

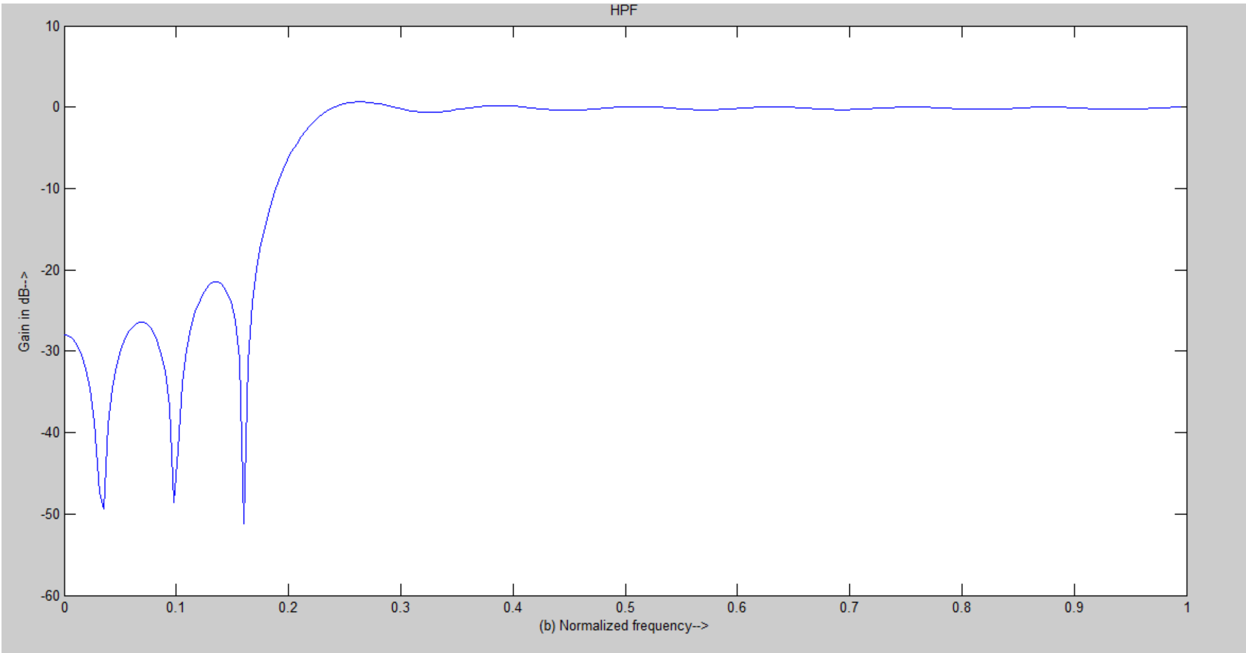
Triangular window filter response



enter your choice of window function 1. rectangular 2. triangular 3.kaiser:

1

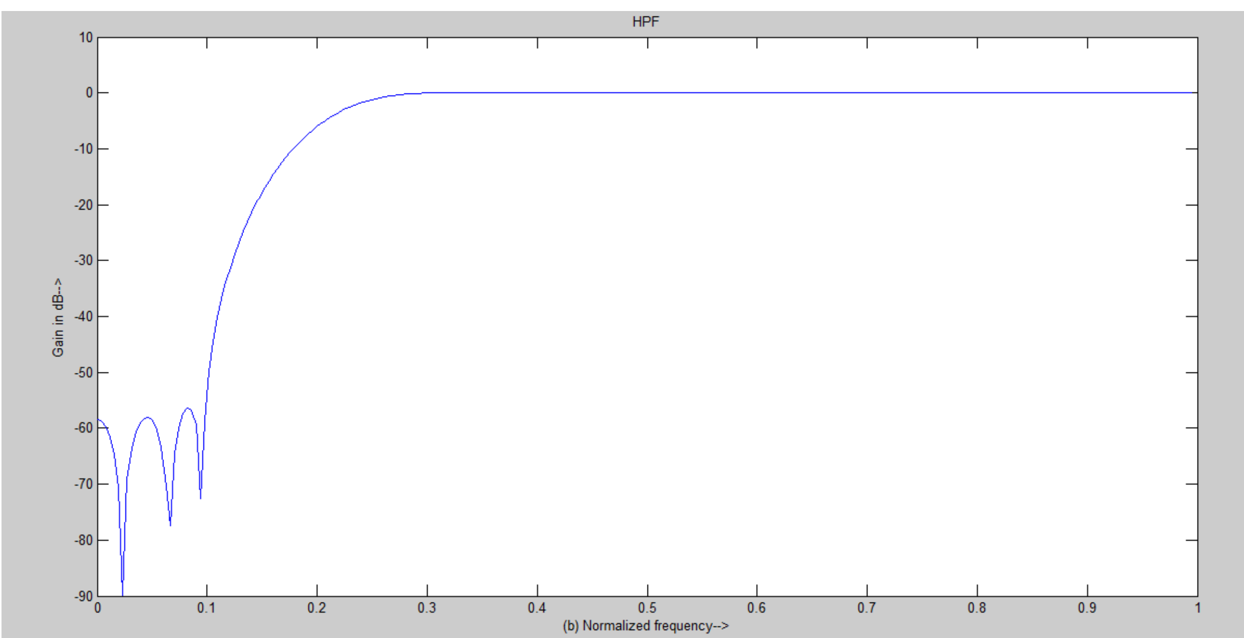
Rectangular window filter response



enter your choice of window function 1. rectangular 2. triangular 3.kaiser:

3

kaiser window filter response



Experiment No. 9

9. Implementation of FIR digital filter using frequencysampling method

AIM:

Decimation by factor D

TOOLS REQUIRED:

Mat lab software

Personal computer

PROGRAM:

```
N=64;                          % FFT length = filterlength
np = floor(N/2)+1;              % number of independent frequency points
n =0:np-1;
w=n*2*pi/N;                     % frequencyvector
M=sin(n*pi/(np-1));             % some desired magnituderesponse
D=M.*exp(-1i*(N-1)/2*w);        % desired complex frequency response (linear
phase)
D=[D,conj(D(N-np+1:-1:2))];     % append redundant points for IFFT
h=ifft(D);                      % compute impulseresponse
max(abs(imag(h)))               % should be very close to0
h=real(h);                      % remove numericalinaccuracies

% check result
[H,w2] = freqz(h,1,4*N);
plot(w/2/pi,abs(D(1:np)),'.',w2/2/pi,abs(H))
```

OUTPUT AND WAVEFORM:

Experiment No. 10

10. Implementation of optimum equiripple FIR digital filterusing windowmethods

AIM:

To Write a Matlab program of FIR Low pass and high pass filter using rectangular, Hanning Hamming, Blackman and Kaiser window.

.

TOOL:

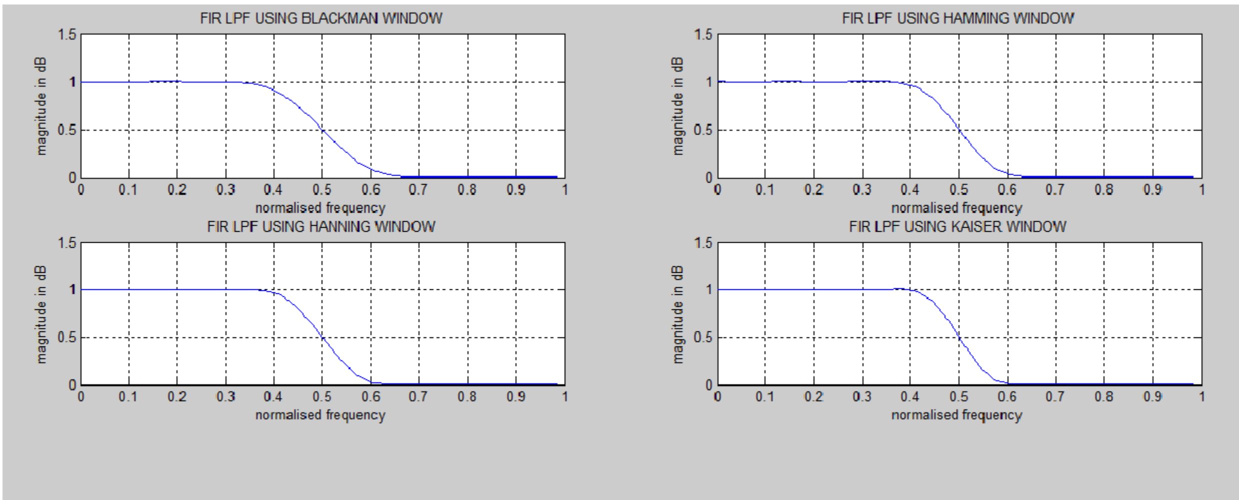
MATLAB Software 9.0

PROGRAM:

```
%MATLAB program of FIR Low pass filter using Hanning
%Hamming, Blackman and Kaiser window

clf;

wc=.5*pi;

N=25;

w=0:0.1:pi;

b=fir1(N,wc/pi,blackman(N+1));

h=freqz(b,1,w);

 subplot(3,2,1)

plot(w/pi,abs(h))

grid;xlabel('normalised frequency');

ylabel('magnitude in dB')
```

```
title('FIR LPF USING BLACKMAN WINDOW')
b=fir1(N,wc/pi,hamming(N+1));
h=freqz(b,1,w);
subplot(3,2,2)
plot(w/pi,abs(h));
grid;
xlabel('normalised frequency');
ylabel('magnitude in dB')
title('FIR LPF USING HAMMING WINDOW')
b=fir1(N,wc/pi,hanning(N+1));
h=freqz(b,1,w);
subplot(3,2,3)
plot(w/pi,abs(h));
grid;
xlabel('normalised frequency');
ylabel('magnitude in dB')
title('FIR LPF USING HANNING WINDOW')
b=fir1(N,wc/pi,kaiser(N+1,3.5));
h=freqz(b,1,w);
subplot(3,2,4)
plot(w/pi,abs(h));
grid;
xlabel('normalised frequency');
ylabel('magnitude in dB')
title('FIR LPF USING KAISER WINDOW')
```

OUTPUT WAVEFORMS:



PROGRAM:

```
%FIR Filter design window techniques

clc;

clear all;

close all;

rp=input('enter passband ripple');

rs=input('enter the stopband ripple');

fp=input('enter passband freq');

fs=input('enter stopband freq');

f=input('enter sampling freq ');

beta=input('enter beta value');

wp=2*fp/f; ws=2*fs/f;

num=-20*log10(sqrt(rp*rs))-13;

dem=14.6*(fs-fp)/f;

n=ceil(num/dem);

n1=n+1; if(rem(n,2)~=0) n1=n; n=n-1;

end

c=input('enter your choice of window function 1. rectangular 2. triangular 3.kaiser: \n ');

if(c==1) y=rectwin(n1);

   disp('Rectangular window filter response');

end

if (c==2) y=triang(n1);

   disp('Triangular window filter response');
```

```
end
if(c==3) y=kaiser(n1,beta);
  disp('kaiser window filter response');
end
%HPF
b=fir1(n,wp,'high',y);
[h,o]=freqz(b,1,256);
m=20*log10(abs(h));
plot(o/pi,m);
title('HPF');
ylabel('Gain in dB-->');
xlabel('(b) Normalized frequency-->');
```

INPUT:

enter passband ripple:0.02

enter the stopband ripple:0.01

enter passband freq:1000

enter stopband freq:1500

enter sampling freq: 10000

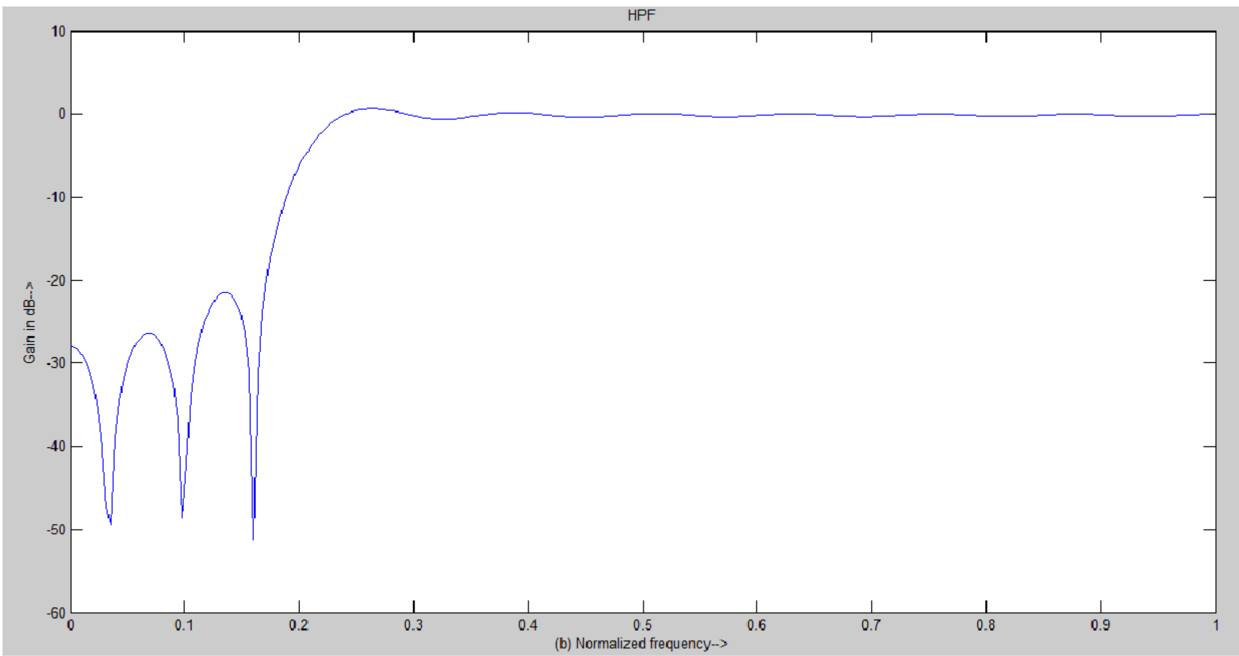
enter beta value:5

OUTPUT WAVEFORM:

enter your choice of window function 1. rectangular 2. triangular 3.kaiser:

2

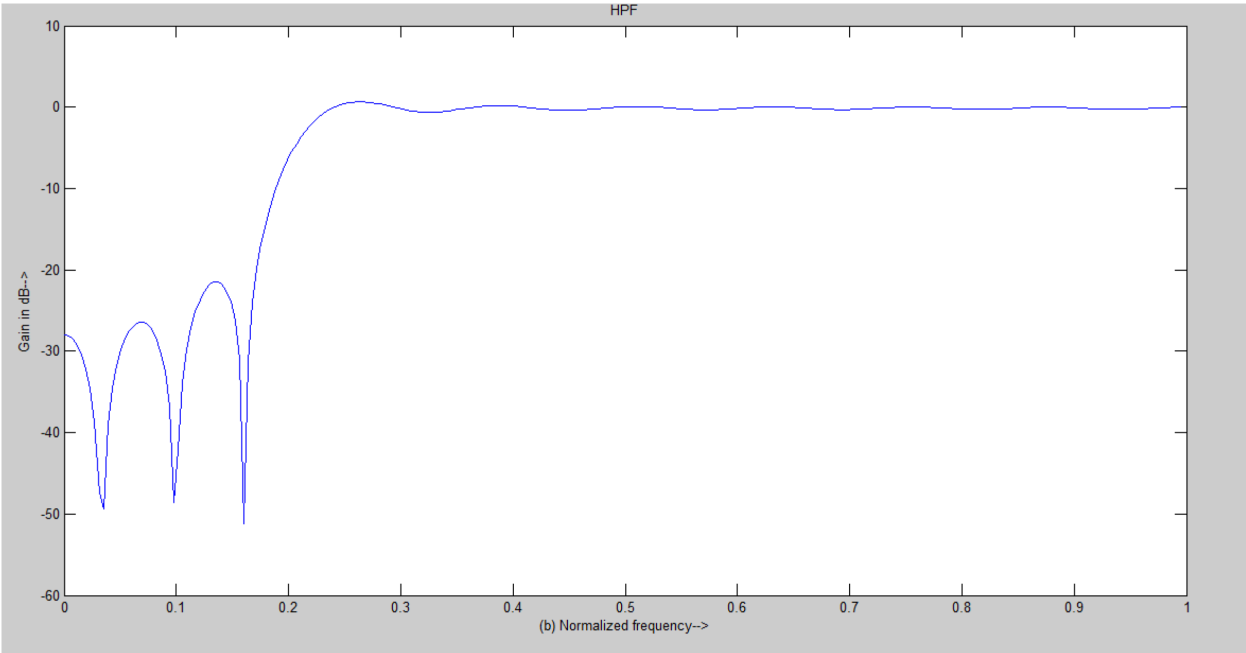
Triangular window filter response



enter your choice of window function 1. rectangular 2. triangular 3.kaiser:

1

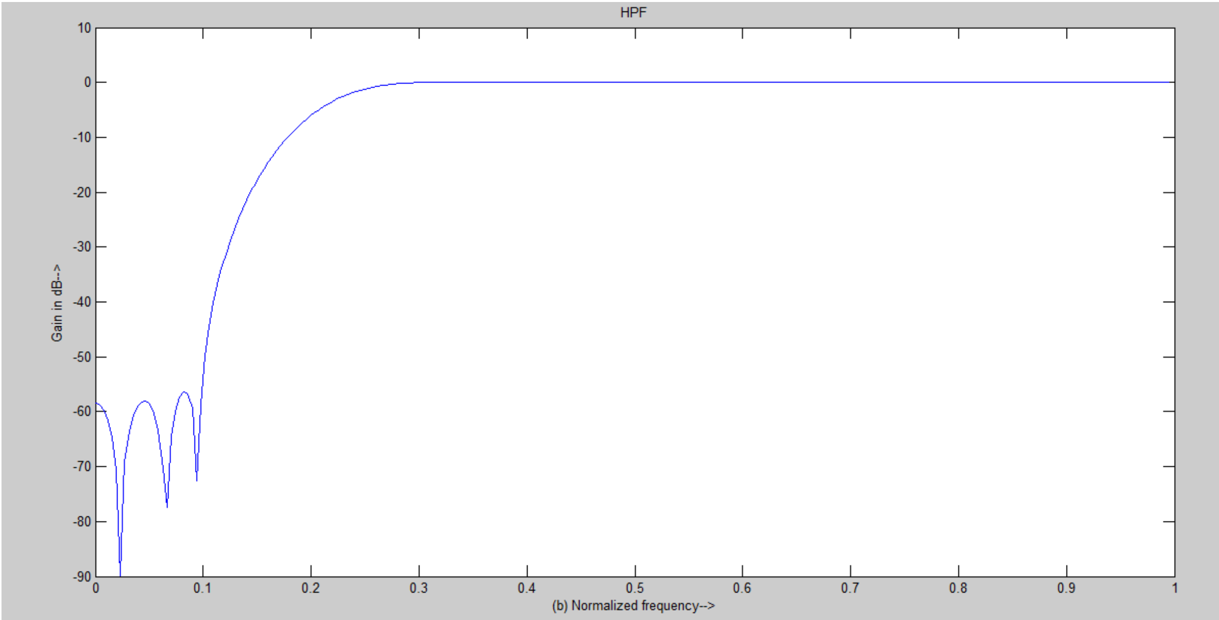
Rectangular window filter response



enter your choice of window function 1. rectangular 2. triangular 3.kaiser:

3

kaiser window filter response


HPF
Gain in dB-->
(b) Normalized frequency-->
10
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Experiment No. 11

11. DTMF Tone Generation and Detection Using GoertzelAlgorithm

AIM:

DTMF Tone Generation and Detection Using Goertzel Algorithm

TOOLS REQUIRED:

Mat lab software

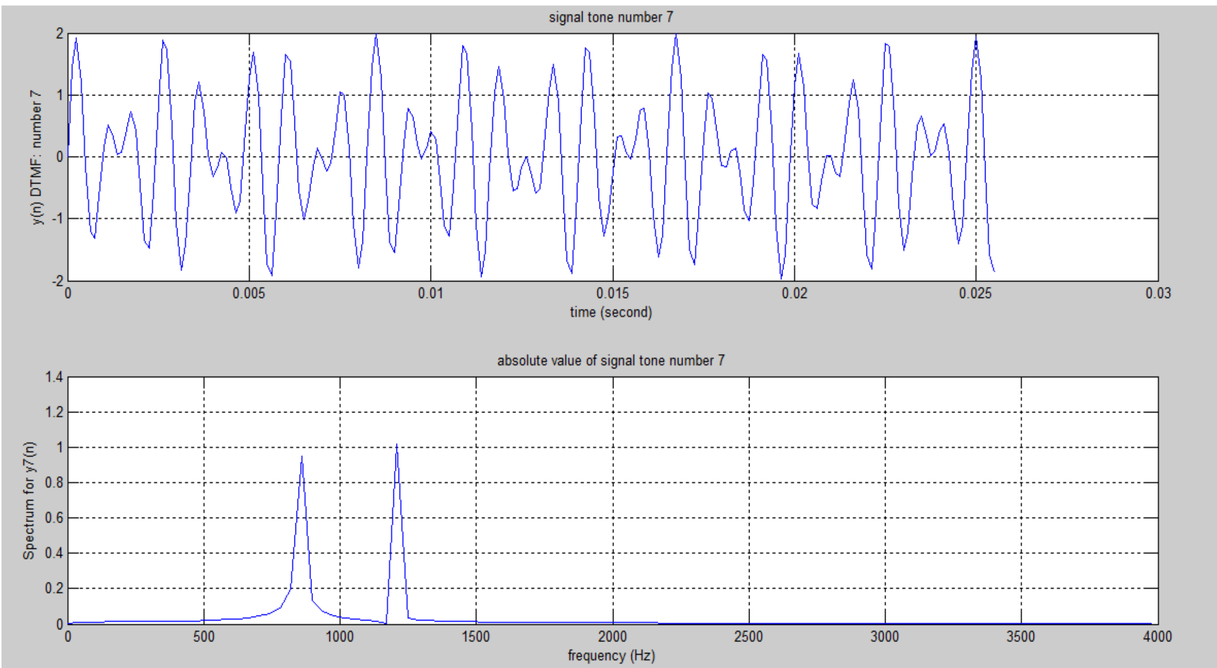
Personal computer

PROGRAM:

```
close all;clear all

% DTMF tone generator

fs=8000;

t=[0:1:204]/fs;

x=zeros(1,length(t));

x(1)=1;

y852=filter([0 sin(2*pi*852/fs) ],[1 -2*cos(2*pi*852/fs) 1],x);

y1209=filter([0 sin(2*pi*1209/fs) ],[1 -2*cos(2*pi*1209/fs) 1],x);

y7=y852+y1209;

subplot(2,1,1);plot(t,y7);grid

ylabel('y(n) DTMF: number 7');

xlabel('time (second)');

title('signal tone number 7')

 Ak=2*abs(fft(y7))/length(y7);Ak(1)=Ak(1)/2;

f=[0:1:(length(y7)-1)/2]*fs/length(y7);

subplot(2,1,2);plot(f,Ak(1:(length(y7)+1)/2));grid
```

```
ylabel('Spectrum for y7(n)');

xlabel('frequency (Hz)');

title('absolute value of signal tone number 7')
```

OUTPUT WAVEFORM:



Experiment No. 12

12. Implementation of sampling rate conversion by decimation, interpolation and a rational factor usingMATLAB

AIM:

Sampling rate conversion by a factor I/D

TOOLS REQUIRED:

Mat lab software

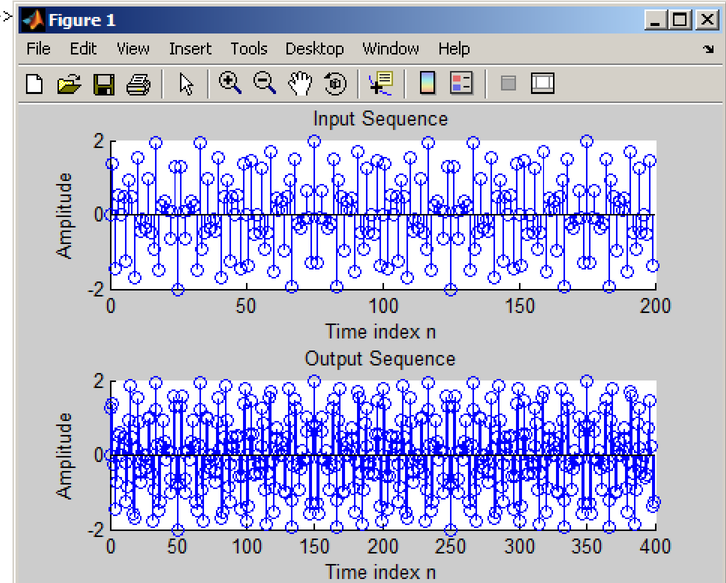
Personal computer

PROGRAM:

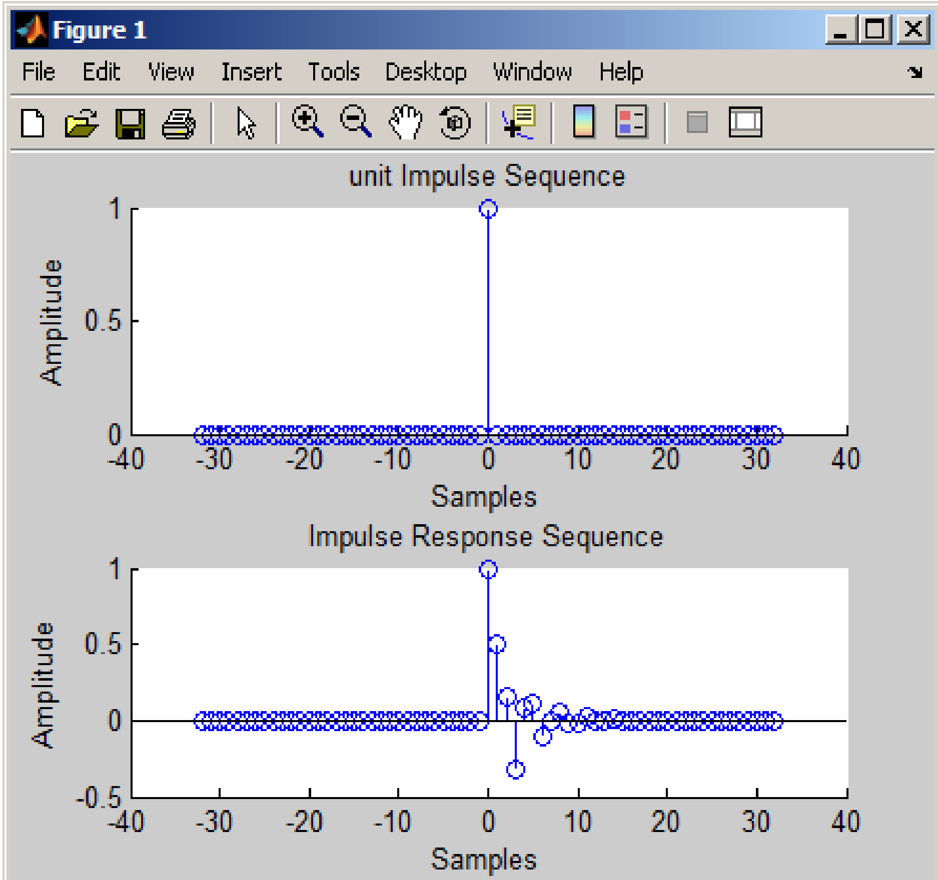
```
clc; close all; clear all;
L = input('Enter Up-sampling factor :');
M = input('Enter Down-sampling factor :');
N = input('Enter number of samples :');
n = 0:N-1;
x = sin(2*pi*0.43*n) + sin(2*pi*0.31*n);
y = resample(x,L,M);
subplot(2,1,1); stem(n,x(1:N));
axis([0 29 -2.2 2.2]);
title('Input Sequence');
xlabel('Time index n'); ylabel('Amplitude');
subplot(2,1,2);
m = 0:(N*L/M)-1;
stem(m,y(1:N*L/M));
axis([0 (N*L/M)-1 -2.22.2]);
title('Output Sequence');
xlabel('Time index n'); ylabel('Amplitude');
```

OUTPUT AND WAVEFORM:

```
Enter Up-sampling factor :6
Enter Down-sampling factor :3
Enter number of samples :200
>>
```



```
enter the coefficients of numerator polynomial= [1 1 0.9]
enter the coefcients of denominator polynomial= [1 0.5 0.5]
enter the length of sequence= 32
>>
```



EXPERIEMENTS
(Using DSP Kit)

INTRODUCTION TO DSP PROCESSORS

A signal can be defined as a function that conveys information, generally about the state or behavior of a physical system. There are two basic types of signals viz Analog (continuous time signals which are defined along a continuum of times) and Digital (discrete-time).

Remarkably, under reasonable constraints, a continuous time signal can be adequately represented by samples, obtaining discrete time signals. Thus digital signal processing is an ideal choice for anyone who needs the performance advantage of digital manipulation along with today's analogreality.

Hence a processor which is designed to perform the special operations (digital manipulations) on the digital signal within very less time can be called as a Digital signal processor. The difference between a DSP processor, conventional microprocessor and a microcontroller are listed below.

Microprocessor: or General Purpose Processor such as Intel xx86 or Motorola 680xx

Family

Contains - only CPU

-No RAM

-NoROM

-No I/O ports

-No Timer

MICROCONTROLLER

Such as 8051 family Contains - CPU

- RAM
- ROM
- I/Oports
- Timer&
- Interruptcircuitry

Some Micro Controllers also contain A/D, D/A and Flash Memory

DSP PROCESSORS

Such as Texas instruments and Analog Devices Contains

- CPU
- RAM
- ROM
- I/Oports
- Timer

Optimizedfor

- fastarithmetic
- Extendedprecision
- Dual operandfetch
- Zero overheadloop
- Circularbuffering

The basic features of a DSP Processor are

Feature	Use
Fast-Multiply accumulate	Most DSP algorithms, including filtering, transforms, etc. are multiplication-intensive
Multiple – access memory architecture	Many data-intensive DSP operations require reading a program instruction and multiple data items during each instruction cycle for best performance
Specialized addressing modes	Efficient handling of data arrays and first-in, first-out buffers inmemory
Specialized program control	Efficient control of loops for many iterative DSP algorithms. Fast interrupt handling for frequent I/Ooperations.
On-chip peripherals and I/O interfaces	On-chip peripherals like A/D converters allow for small low cost system designs. Similarly I/O interfaces tailored for common peripherals allow clean interfaces to off-chip I/O devices.

A digital signal processor (DSP) is an integrated circuit designed for high-speed data manipulations, and is used in audio, communications, image manipulation, and other data-acquisition and data-control applications. The microprocessors used in personal computers are optimized for tasks involving data movement and inequality testing. The typical applications requiring such capabilities are word processing, database management, spread sheets, etc. When it comes to mathematical computations the traditional microprocessor are deficient particularly where real-time performance is required. Digital signal processors are microprocessors optimized for basic mathematical calculations such as additions and multiplications.

FIXED VERSUS FLOATING POINT:

Digital Signal Processing can be divided into two categories, fixed point and floating point which refer to the format used to store and manipulate numbers within the devices. Fixed point DSPs usually represent each number with a minimum of 16 bits, although a different length can be used. There are four common ways that these 216 i,e., 65,536 possible bit patterns can represent a number. In unsigned integer, the stored number can take on any integer value from 0 to 65,535, signed integer uses two's complement to include negative numbers from -32,768 to 32,767. With unsigned fraction notation, the 65,536 levels are spread uniformly between 0 and 1 and the signed fraction format allows negative numbers, equally spaced between -1 and 1. The floating point DSPs typically use a minimum of 32 bits to store each value. This results in many more bit patterns than for fixed point, 232 i,e., 4,294,967,296 to be exact. All floating point DSPs can also handle fixed point numbers, a necessity to implement counters, loops, and signals coming from the ADC and going to the DAC. However, this doesn't mean that fixed point math will be carried out as quickly as the floating point operations; it depends on the internalarchitecture.

C VERSUS ASSEMBLY:

DSPs are programmed in the same languages as other scientific and engineering applications, usually assembly or C. Programs written in assembly can execute faster, while programs written in C are easier to develop and maintain. In traditional applications, such as programs run on PCs and mainframes, C is almost always the first choice. If assembly is used at all, it is restricted to short subroutines that must run with the utmostspeed.

HOW FAST ARE DSPS?

The primary reason for using a DSP instead of a traditional microprocessor is speed: the ability to move samples into the device and carry out the needed mathematical operations, and output the processed data. The usual way of specifying the fastness of a DSP is: fixed point systems are often quoted in MIPS (million integer operations per second). Likewise, floating point devices can be specified in MFLOPS (million floating point operations per second).

TMS320 FAMILY:

The Texas Instruments TMS320 family of DSP devices covers a wide range, from a 16-bit fixed-point device to a single-chip parallel-processor device. In the past, DSPs were used only in specialized applications. Now they are in many mass-market consumer products that are continuously entering new market segments. The Texas Instruments TMS320 family of DSP devices and their typical applications are mentioned below.

C1x, C2x, C2xx, C5x, and C54x:

The width of the data bus on these devices is 16 bits. All have modified Harvard architectures. They have been used in toys, hard disk drives, modems, cellular phones, and active car suspensions.

C3x:

The width of the data bus in the C3x series is 32 bits. Because of the reasonable cost and floating-point performance, these are suitable for many applications. These include almost any filters, analyzers, hi-fi systems, voice-mail, imaging, bar-code readers, motor control, 3D graphics, or scientific processing.

C4x:

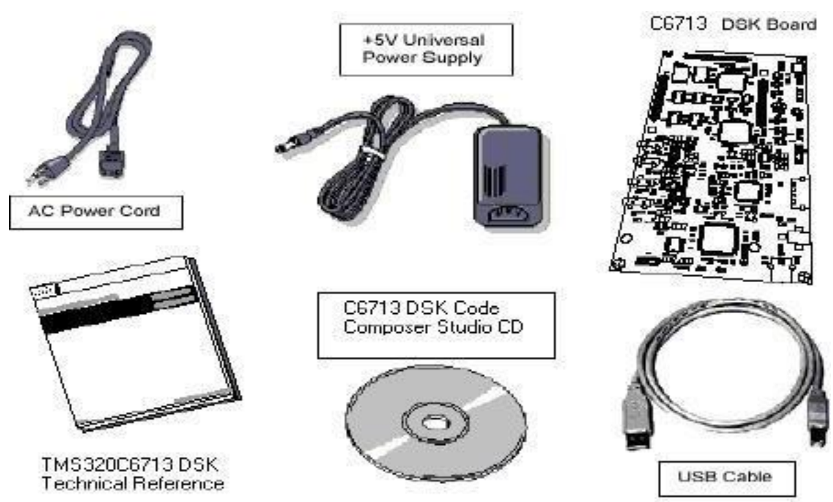
This range is designed for parallel processing. The C4x devices have a 32-bit data bus and are floating-point. They have an optimized on-chip communication channel, which enables a number of them to be put together to form a parallel-processing cluster. The C4x range devices have been used in virtual reality, image recognition, telecom routing, and parallel-processing systems.

C6x:

The C6x devices feature Velocity, an advanced very long instruction word (VLIW) architecture developed by Texas Instruments. Eight functional units, including two multipliers and six arithmetic logic units (ALUs), provide 1600 MIPS of cost-effective performance. The C6x DSPs are optimized for multi-channel, multifunction applications, including wireless base stations, pooled modems, remote-access servers, digital subscriber loop systems, cable modems, and multi-channel telephone systems.

INTRODUCTION TO TMS 320 C6713 DSK

The high–performance board features the TMS320C6713 floating-point DSP. Capable of performing 1350 million floating point operations per second, the C6713 DSK the most powerful DSK developmentboard.



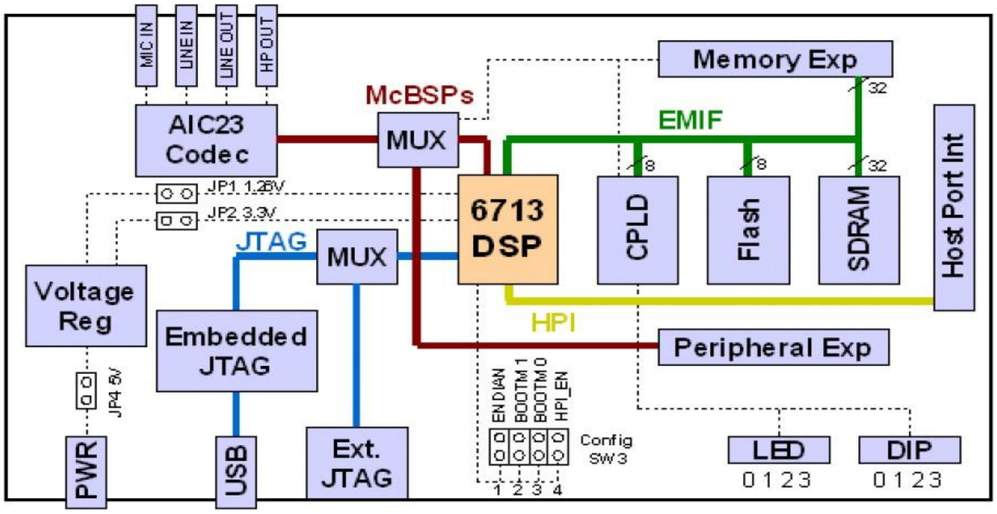
The DSK is USB port interfaced platform that allows to efficiently develop and test applications for the C6713. With extensive host PC and target DSP software support, the DSK provides ease-of-use and capabilities that are attractive to DSP engineers. The 6713 DSP Starter Kit (DSK) is a low-cost platform which lets customers evaluate and develop applications for the Texas Instruments C67X DSP family. The primary features of the DSK are:

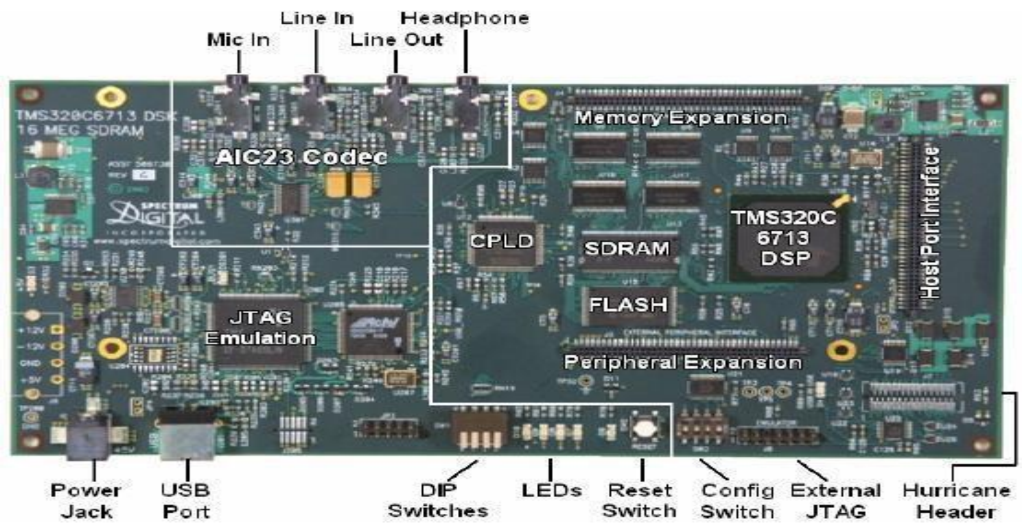
1. 225 MHz TMS320C6713 Floating PointDSP
2. AIC23 StereoCodec
3. Four Position User DIP Switch and Four UserLEDs
4. On-board Flash andSDRAM

TI‟sCodeComposerStudiodevelopmenttoolsarebundledwiththe6713DSKprovidingtheuser with an industrial-strength integrated development environment for C and assembly programming. Code Composer Studio communicates with the DSP using an on-board JTAG emulator through a USB interface. The TMS320C6713 DSP is the heart of the system. It is a core member of Texas Instruments‟ C64X line of fixed point DSPs whose distinguishing features are an extremely high performance 225MHz VLIW DSP core and 256Kbytes of internal memory. On- chip peripherals include a 32-bit external memory interface (EMIF) with integrated SDRAM controller, 2 multi-channel buffered serial ports (McBSPs), two on-board timers and an enhanced DMA controller (EDMA). The 6713 represents the high end of TI‟ s C6700 floating point DSP line both in terms of computational performance and on-chipresources.

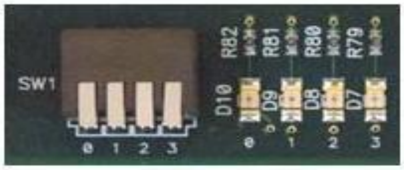
The 6713 has a significant amount of internal memory so many applications will have all code and data on-chip. External accesses are done through the EMIF which can connect to both synchronous and asynchronous memories. The EMIF signals are also brought out to standard TI expansion bus connectors so additional functionality can be added on daughter card modules. DSPs are frequently used in audio processing applications so the DSK includes an on-board codec called the AIC23. Codec stands for coder/decoder, the job of the AIC23 is to code analog input samples into a digital format for the DSP to process, then decode data coming out of the DSP to generate the processed analog output. Digital data is sent to and from the codec on McBSP1.

TMS320C6713 DSK OVERVIEW BLOCK DIAGRAM





The DSK has 4 light emitting diodes (LEDs) and 4 DIP switches that allow users to interact with programs through simple LED displays and user input on the switches. Many of the included examples make use of these user interfaces Options.



The DSK implements the logic necessary to tie board components together in a programmable logic device called a CPLD. In addition to random glue logic, the CPLD implements a set of 4 software programmable registers that can be used to access the on-board LEDs and DIP switches as well as control the daughter card interface.

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 theoutput.

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.

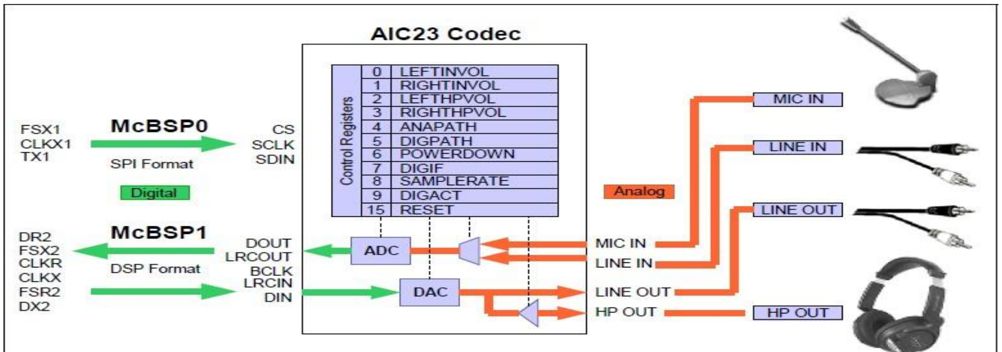


Figure 2-1, TMS320C6713 DSK CODEC INTERFACE

DSK hardware installation

- Shut down and power off thePC
- Connect the supplied USB port cable to theboard
- Connect the other end of the cable to the USB port ofPC
- Plug the other end of the power cable into a poweroutlet
- Plug the power cable into theboard
- The user LEDs should flash several times to indicate board isoperational
- When you connect your DSK through USB for the first time on aWindows

Loaded PC the new hardware found wizard will come up. **So, Install the drivers** (The CCS CD contains the require drivers for C6713 DSK).

TROUBLESHOOTING DSK CONNECTIVITY

If Code Composer Studio IDE fails to configure your port correctly, perform the following steps:

Test the USB port by running DSK Port test from the start menu

Use Start->Programs->Texas Instruments->Code Composer Studio-> Code Composer Studio

C6713 DSK Tools -> C6713 DSK Diagnostic Utilities

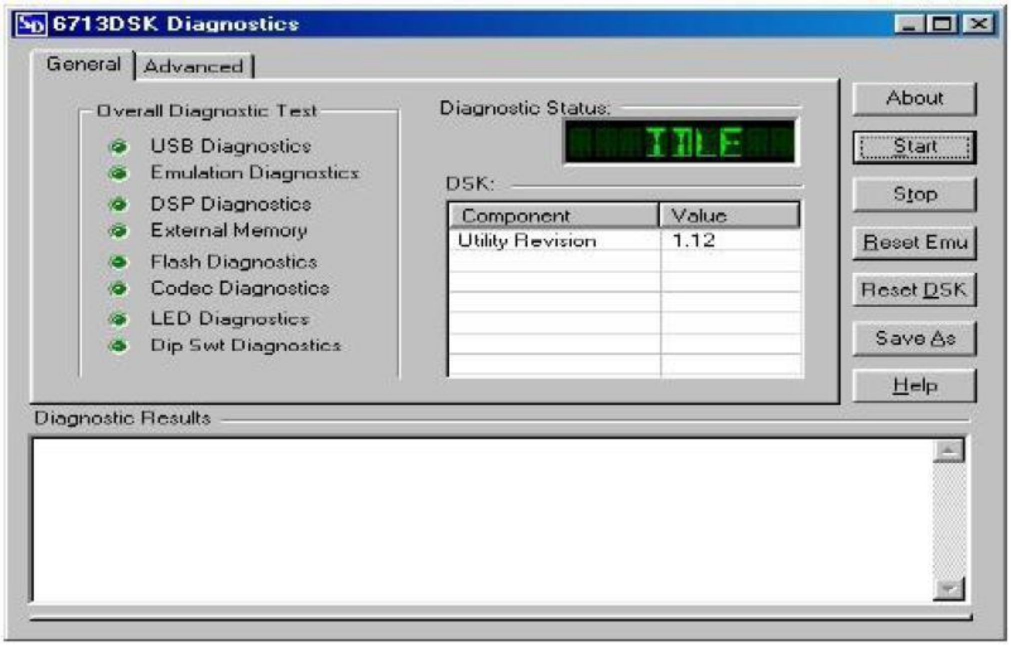
The below Screen will appear

Select 6713 DSK Diagnostic Utility Icon from Desktop, The Screen Look like as below

Select **Start** Option

Utility Program will test the board

After testing Diagnostic Status you will get **PASS**



INTRODUCTION TO CODE COMPOSER STUDIO

Code Composer is the DSP industry's first fully integrated development environment (IDE) with DSP-specific functionality. With a familiar environment liked MS-based C+, Code Composer lets you edit, build, debug, profile and manage projects from a single unified environment. Other unique features include graphical signal analysis, injection/extraction of data signals via file I/O, multi-processor debugging, automated testing and customization via a C-interpretive scripting language and much more.

CODE COMPOSER FEATURES INCLUDE:

- IDE
- DebugIDE
- Advanced watchwindows
- Integratededitor
- File I/O, Probe Points, and graphical algorithm scopeprobes
- Advanced graphical signalanalysis

- Interactiveprofiling
- Automated testing and customization viascripting
- Visual project managementsystem
- Compile in the background while editing anddebugging
- Multi-processordebugging
- Help on the targetDSP

TO CREATE A SYSTEM CONFIGURATION USING A STANDARD CONFIGURATION FILES:

STEP1: Start CCS Setup by double clicking on the Setup CCS desktopicon.

STEP2: Select Family ->c67xx

Platform->simulator

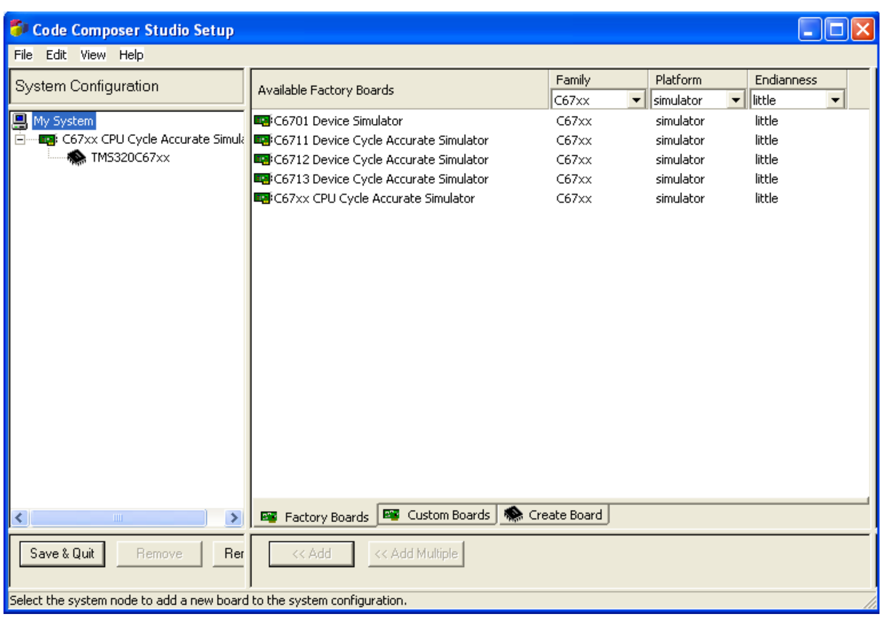
Endean's->little

STEP 3: Click the Import button (File-> import) to import our selection (c67xx_sim.ccs) to the

System configuration currently being created in the CCS Setup window.

STEP4: Click the Save and Quit button to save the configuration in the SystemRegistry.

STEP 5: Click the Yes button to start the CCS IDE when we exit CCS Setup. The CCS Setup



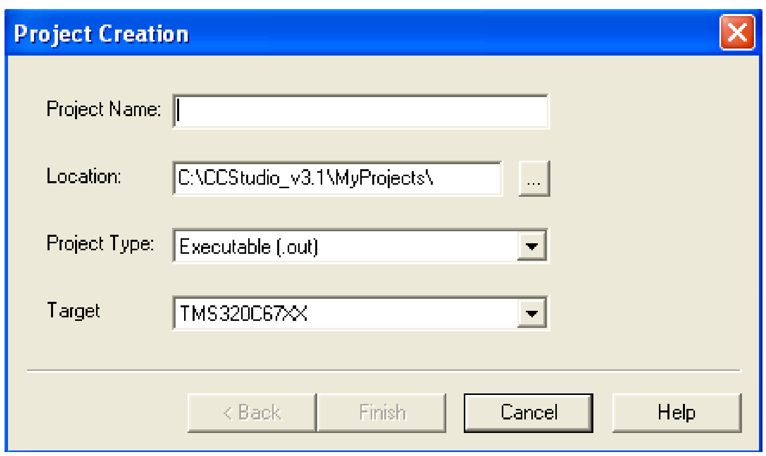
Closes and the CCS IDE automatically opens using the configuration we just created.

PROCEDURE TO WORK ON CODE COMPOSER STUDIO

STEP 1: Creating a New Project

From the Project menu, choose New. In the Project Name field, type the name we want for our project. Each project we create must have a unique name, and Click Finish. The CCS IDE creates a project file called projectname.pjt. This file stores our project settings and references the various files used by our project.

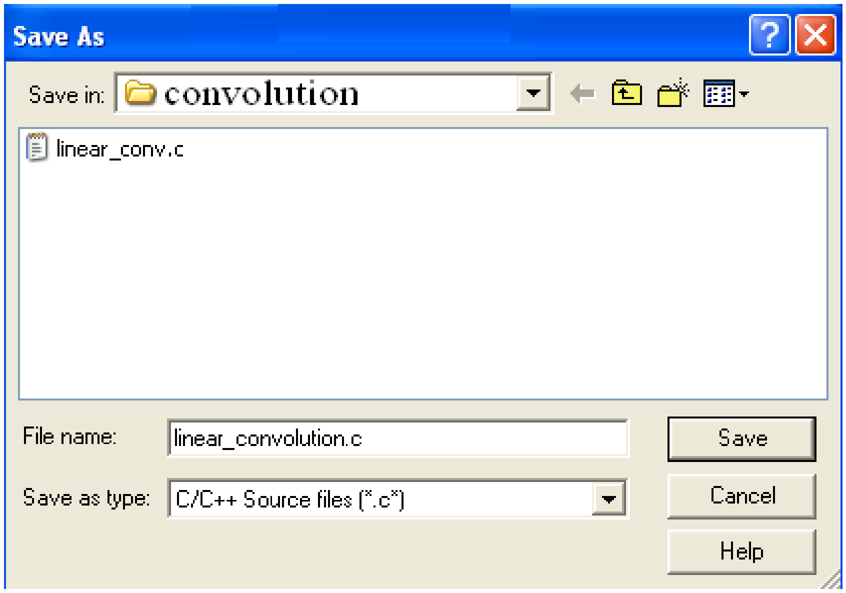
The Project Creation wizard window displays.



STEP 2: Creating a source file

Create a new source file using 'File ->new ->source file ' pull down menu and save the source file with **.c** extension in the current project name directory**.** Save as type: c/c++ source file (*.c*)

Path: C:\CCStudio_v3.1\ MyProjects\Project Name\



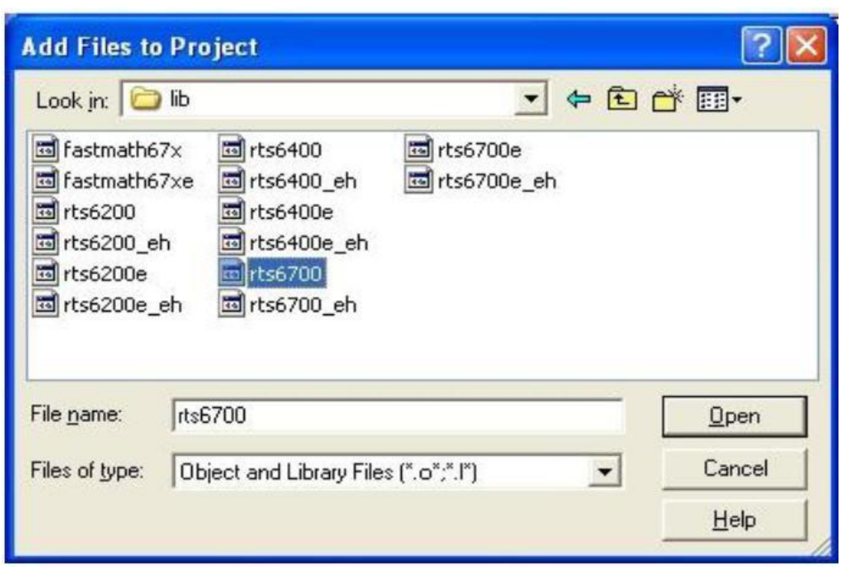
STEP 3: Add files to our project (source file\ library file\ linker file)

SOURCEFILE: Add the source file in the project using 'Project->add files to project' pull down menu. Files of type: c/c++ source file(*.c*)

Path: C:\CCStudio_v3.1\ My Projects\Project Name\filename's

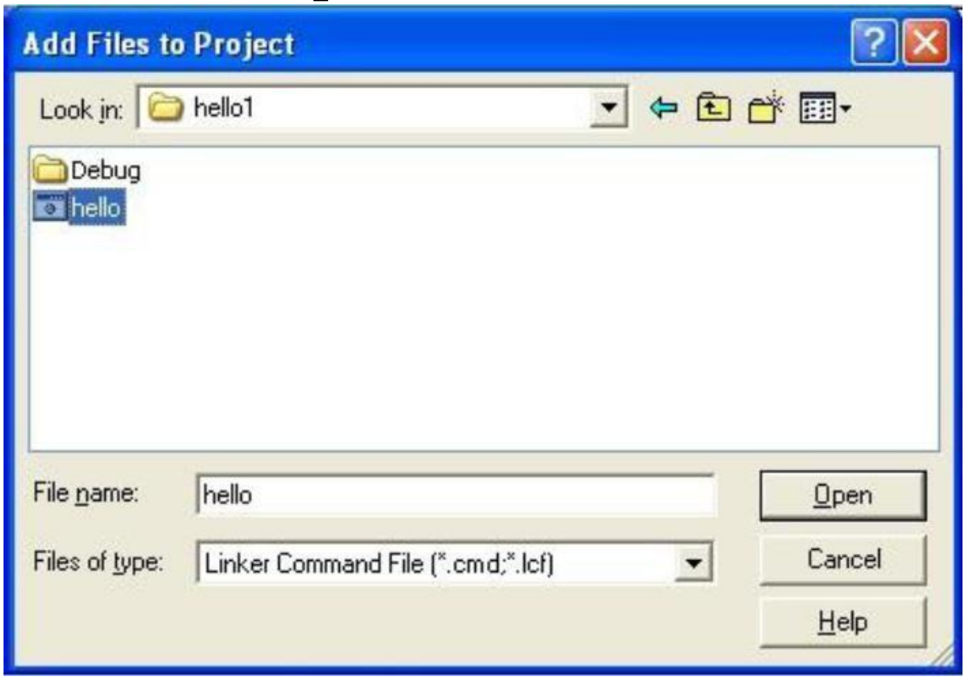
LIBRARYFILE: Add the library file in the project using 'Project-> add files to project' pull down menu. Files of type: Object and Library Files(*.o*,*.l*)

Path: C:\CCStudio_v3.1\ C6000\ cgtools\ lib \ rts6700.lib



LINKERFILE: Add the linker file in the project using 'Project-> add files to project' pull down menu. Files of type: Linker command Files(*.cmd*,*.lcf*)

Path: C:\CCStudio_v3.1\ tutorial\ dsk6713\ hello1 \ hello.cmd



STEP 4: Building and Running the Program (compile\ Build\ Load Program\ Run)

COMPILE: Compile the program using the 'Project-compile' pull down menu or by clicking the shortcut icon on the left side of programwindow.

BUILD: Build the program using the 'Project-Build' pull down menu or by clicking the shortcut icon on the left side of programwindow.

LOAD PROGRAM: Load the program in program memory of DSP chip using the 'File-load program' pull down menu. Files of type:(*.out*)

Path: C:\CCStudio_v3.1\ MyProjects\Project Name\ Debug\ Project ame.out

RUN: Run the program using the 'Debug->Run' pull down menu or by clicking the shortcut icon on the left side of programwindow.

STEP 5: observe output using graph

Choose View-> Graph-> Time/Frequency. In The Graph Property Dialog, Change The Graph Title, Start Address, And Acquisition Buffer Size, Display Data Size, Dsp Data Type, Auto Scale, And Maximum Y- Value Properties To TheValues.

Experiment No. 13

13.a) Implementation of DFT USING TMS 320C6713 Kit

b) Sine wave generation using lookup table with values generated fromMATLAB

AIM:

Computation of N-point DFT of a Sequence Using DSK Code composer studio

EQUIPMENTS:

TMS 320C6713 Kit.

RS232 Serial Cable

Power Cord

Operating System – Windows XP

Software – CCStudio_v3.1

THEORY:

In this program the Discrete Fourier Transform (DFT) of a sequence x[n] is generated by using the formula,

$$X(k) = \sum_{}^{N-1} x(n)\, e^{-2\pi jk/N}$$

Where, X(k) →DFT of sequencex[n]

n=0

N represents the sequence length and it is calculated by using the command ‘length’. The DFT of any sequence is the powerful computational tool for performing frequency analysis of discrete-timesignals.

PROGRAM:

```
#include  <stdio.h>

#include <math.h>

int N,k,n,i;

float  pi=3.1416,sumre=0,sumim=0,out_real[8]={0.0},out_imag[8]={0.0};

int x[32];

void main(void)

{

printf("enter the length of the sequence\n");

scanf("%d",&N);

printf("\nenter the sequence\n");
for(i=0;i<N;i++)

scanf("%d",&x[i]);

for(k=0;k<N;k++)

{

sumre=0;

sumim=0;

for(n=0;n<N;n++)
```

```
{
sumre=sumre+x[n]*cos(2*pi*k*n/N);
sumim=sumim-x[n]*sin(2*pi*k*n/N);
}
out_real[k]=sumre;
out_imag[k]=sumim;
printf("DFT of the sequence:\n");
printf("x[%d]=\t%f\t+\t%fi\n",k,out_real[k],out_imag[k]);
}
}
```

PROCEDURE:

- Open code composer studio, make sure the dsp kit is turnedon.
- Startanewprojectusing‘project-new‘pulldownmenu,saveitinaseparate directory(d:11951a0xxx) with name**dft**.
- Write the program and save it asdft.c
- Add the source files dft.c to the project using ‘project->add files to project’ pull downmenu.
- Add the linker command filehello.cmd.

  (path: c:ccstudio_v3.1\tutorial\dsk6713\hello1\hello.cmd)

- Add the run time support library filerts6700.lib.

  (path: c:ccstudio_v3.1\c6000\cgtools\lib\rts6700.lib)

- Compile the program using the ‘project-compile’ pull downmenu
- Build the program using the ‘project-build’ pull downmenu
- Load the program (dft.out) in program memory of dsp chip using the‘file-load program’ pull downmenu.
- Debug->run

- To view output graphically select view ->graph ->time andfrequency.

OUTPUT AND WAVEFORM:

enter the length of the sequence

4

enter the sequence

1 2 3 4

DFT of the sequence:

x[0]= 10.000000 + 0.000000i

DFT of the sequence:

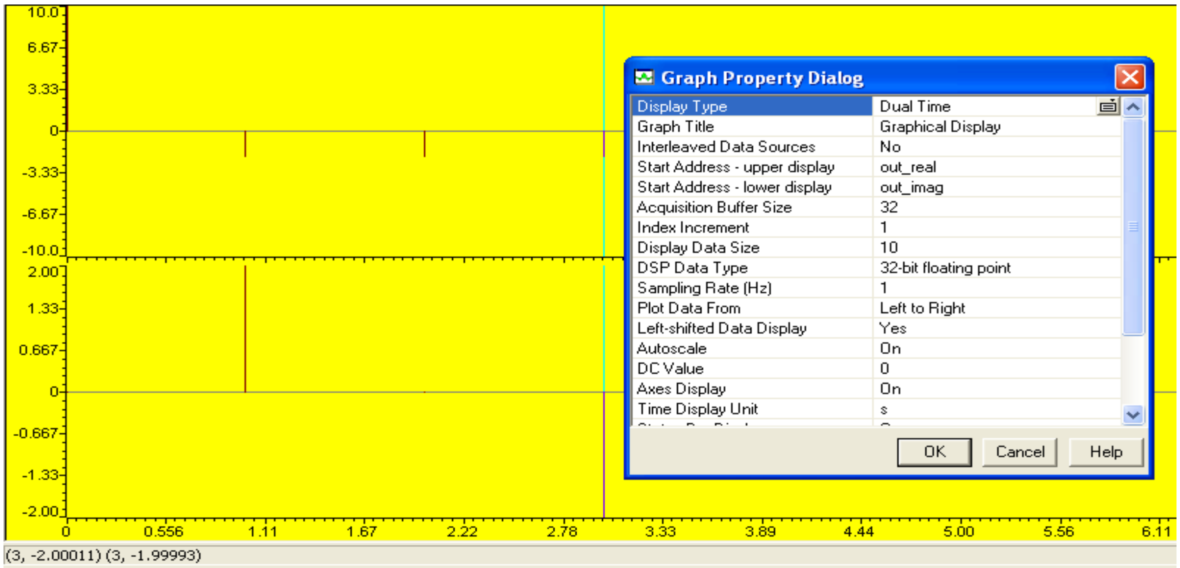
x[1]= -1.999963 + 2.000022i

DFT of the sequence:

x[2]= -2.000000 + 0.000059i

DFT of the sequence:

x[3]= -2.000108   +   -1.999934i



Empty Space for Calculations

Insert Graph Sheet (Normal)

GENERATION OF SINE WAVE USING C6713 DSK

AIM:

To generate a real time sinewave using TMS320C6713 DSK

EQUIPMENTS:

TMS 320C6713 Kit.

RS232 Serial Cable

Power Cord

Operating System – Windows XP

Software – CCStudio_v3.1

PROCEDURE:

1. Connect CRO to the LINE OUTsocket.
2. Now switch ON the DSK and bring up Code Composer Studio onPC
3. Create a new project with namesinewave.pjt
4. From File menu->New->DSP/BIOS Configuration->Select dsk6713.cdb and save it as " sinewave.cdb"
5. Add sinewave.cdb to the currentproject
6. Create a new source file and save it assinewave.c
7. Add the source file sinewave.c to theproject
8. Add the library file "dsk6713bsl.lib" to theproject
   (Path: C:\CCStudio\C6000\dsk6713\lib\dsk6713bsl.lib)
9. Copy files "dsk6713.h" and "dsk6713_aic23.h" to the Projectfolder
   (Path:C:\CCStudio_v3.1\C6000\dsk6713\include)
10. Build (F7) and load the program to the DSP Chip (File->Load Program(.outfile))
11. Run the program(F5)
12. Observe the waveform that appears on the CRO screen and ccstudiosimulator.

%% matlab code to generate the sine values of the look table

```
n=1:48;
x=sin(2*pi*n*1000/48000);
x1=round(x*2^15);
```

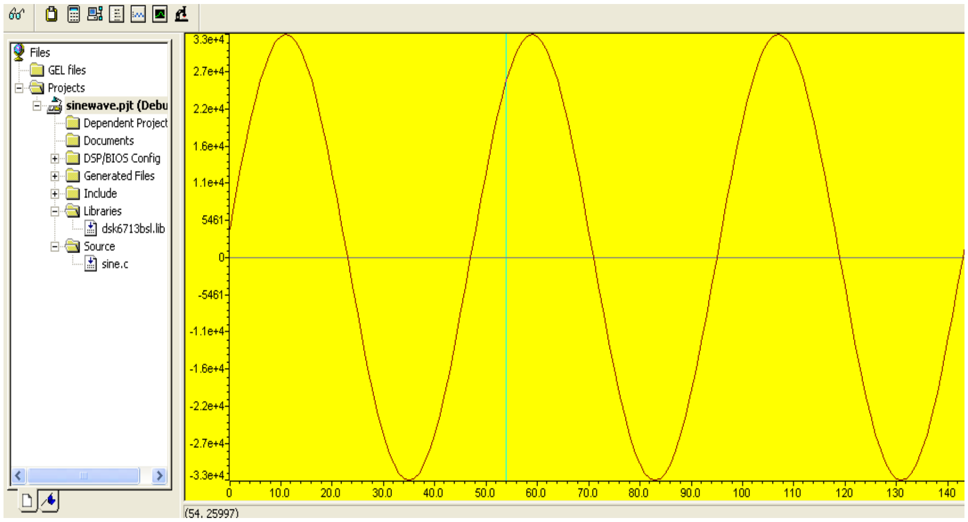
// c program for generation of sine wave using c6713 DSK

```
#include "sinewavecfg.h"
#include "dsk6713.h"
#include"dsk6713_aic23.h"
shortloop=0;
shortgain=1;
Int16 outbuffer[256];
```

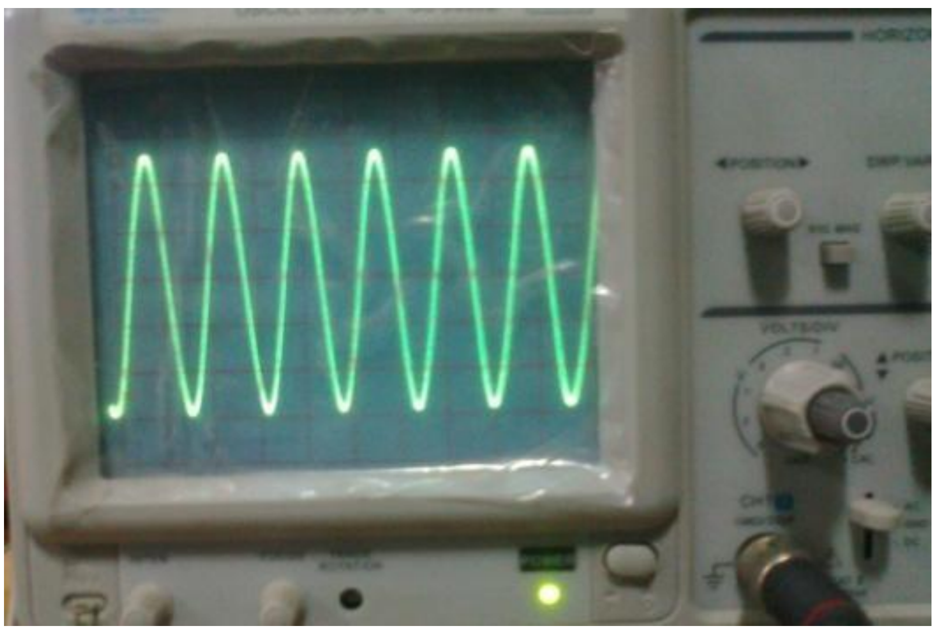
```
const short BUFFERLENGTH=256;
int i=0;
DSK6713_AIC23_Config
config={0x0017,0x0017,0x00d8,0x00d8,0x0011,0x0000,0x0000,0x0043,0x0081,0x0001};
Int16 sine_table[48]={4277,8481,12540,16384,19948,23170,25997,28378,30274,31651,32488,
32766,32488,31651,30274,28378,25997,23170,19948,16384,12540,8481,4277,0,-4277,-8481,
-12540,-16384,-19948,-23170,-25997,-28378,-30274,-31651,-32488,-32766,-32488,-31651,
-30274,-28378,-25997,-23170,-19948,-16384,-12540,-8481,-4277,0};
Uint32 fs=DSK6713_AIC23_FREQ_48KHZ;
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
DSK6713_init();
hCodec=DSK6713_AIC23_openCodec(0, &config);
DSK6713_AIC23_setFreq(hCodec, fs);
while(1)
{
outbuffer[i]=sine_table[loop];
while(!DSK6713_AIC23_write(hCodec, sine_table[loop])*gain);
i++;
if(i==BUFFERLENGTH) i=0;
if(++loop>47) loop=0;
}
}
```

WAVEFORM:

a) output from code composerstudio



b) output from theCRO



INSERT GRAPH SHEET (NORMAL)

Experiment No. 14

14. IR and FIR Filter Implementation using DSPKits

FIR FILTER USING RECTANGULAR WINDOW

AIM:

To generate a real time fir filter through Rectangular window using TMS320C6713 DSK

EQUIPMENTS:

TMS 320C6713 Kit.

RS232 Serial Cable

Power Cord

Operating System – Windows XP

Software – CCStudio_v3.1

PROCEDURE:

1. Connect CRO to the LINE OUTsockets.
2. Now switch ON the DSK and bring up Code Composer Studio onPC
3. Create a new project with namesinewave.pjt
4. From File menu->New->DSP/BIOS Configuration->Select dsk6713.cdb and save it as “ firfliter.cdb”
5. Add firfilter.cdb to the currentproject
6. Create a new source file and save it asfirfilter.c
7. Add the source file firfilter.c to theproject
8. Add the library file “dsk6713bsl.lib” to theproject
   (Path: C:\CCStudio\C6000\dsk6713\lib\dsk6713bsl.lib)
9. Copyfiles“dsk6713.h”and“dsk6713_aic23.h”tothe Projectfolder
   (Path:C:\CCStudio_v3.1\C6000\dsk6713\include)
10. Build (F7) and load the program to the DSP Chip (File->Load Program(.outfile))
11. Run the program(F5)

12. Observe the waveform that appears on the CRO screen and ccstudiosimulator.

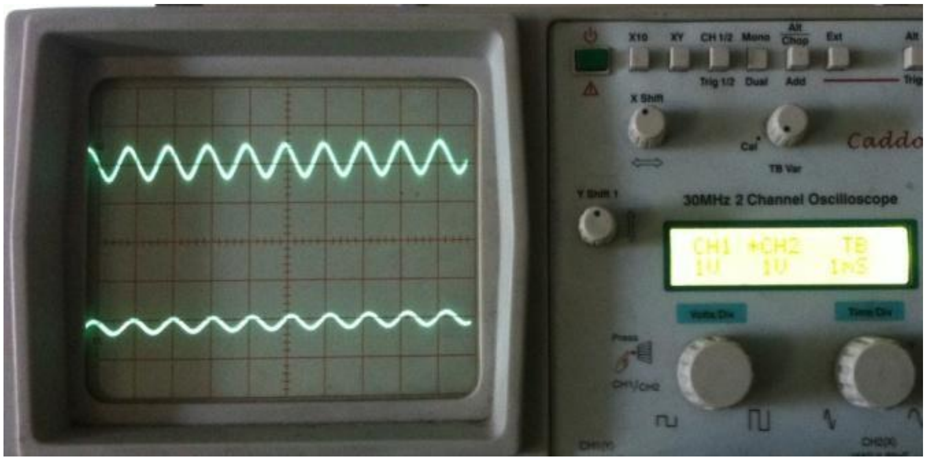
// c program for generation of fir filter using c6713 DSK

```
#include "firfiltercfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#include "stdio.h"
Float filter_coeff[ ]={-0.020203,-0.016567,0.009656,0.027335,0.011411,-0.023194,-
0.033672,0.000000,0.043293,0.038657,-0.025105,-0.082004,-0.041842,0.115971,0.303048,
0.386435,0.303048,0.115971,-0.041842,-0.082004,-0.025105,0.038657,0.043293,0.000000,-
0.033672,-0.023194,0.011411,0.027335,0.009656,-0.016567,-0.020203};//FIR Low pass
Rectangular Filter pass band range 0-1500Hz
DSK6713_AIC23_Config
config={0x0017,0x0017,0x00d8,0x00d8,0x0011,0x0000,0x0000,0x0043,0x0081,0x0001};
void main()
{
 DSK6713_AIC23_CodecHandle hCodec;
 Uint32 l_input, r_input,l_output, r_output;
 DSK6713_init();
 hCodec = DSK6713_AIC23_openCodec(0, &config);
 DSK6713_AIC23_setFreq(hCodec, 1);
 while(1)
 {
 while(!DSK6713_AIC23_read(hCodec, &l_input));
 while(!DSK6713_AIC23_read(hCodec, &r_input));
 l_output=(Int16)FIR_FILTER(&filter_coeff ,l_input);
 r_output=l_output;
 while(!DSK6713_AIC23_write(hCodec, l_output));
 while(!DSK6713_AIC23_write(hCodec, r_output));
 }
 DSK6713_AIC23_closeCodec(hCodec);
 }
```

```
signed int FIR_FILTER(float * h, signed int x)
{
int i=0;
signed long output=0;
static short int in_buffer[100];
in_buffer[0] = x;
for(i=30;i>0;i--)
in_buffer[i] = in_buffer[i-1];
for(i=0;i<32;i++)
output = output + h[i] * in_buffer[i];
return(output);
}
```

WAVEFORM:

a) Waveforms of input and output fromcro

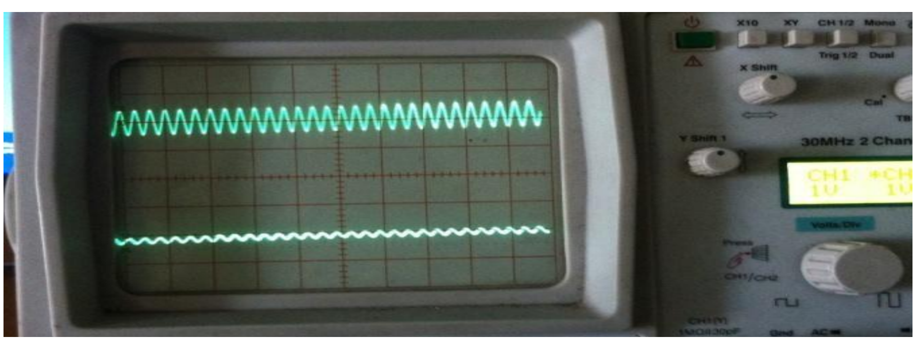


b) Function generator (input signal frequency at 1KHz)

c) C6713 DSK

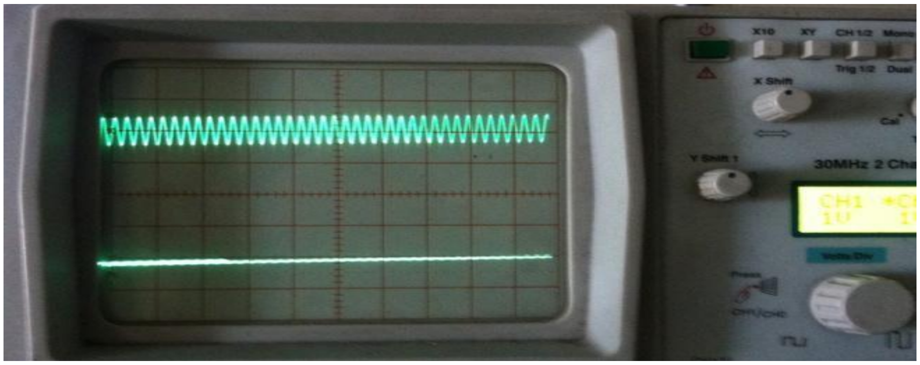
d) Function generator (input signal frequency at1500Hz)

e) Waveforms of input and output from CRO (output signalattenuates)



d) Function generator (input signal frequency at 2000Hz)

a) Waveforms of input and output from CRO (output signal fully attenuated)



INSERT GRAPH SHEET (NORMAL)

FIR FILTER USING KAISER WINDOW (HIGH PASS)

AIM:

To generate a real time fir filter through Kaiser Window using TMS320C6713 DSK

EQUIPMENTS:

TMS 320C6713 Kit.

RS232 Serial Cable

Power Cord

Operating System – Windows XP

Software – CCStudio_v3.1

PROCEDURE:

1. Connect CRO to the LINE OUTsockets.
2. Now switch ON the DSK and bring up Code Composer Studio onPC
3. Create a new project with namesinewave.pjt
4. From File menu->New->DSP/BIOS Configuration->Select dsk6713.cdb and save it as "firfliter_kaiser.cdb"
5. Add firfilter.cdb to the currentproject
6. Create a new source file and save it asfirfilter_kaiser.c
7. Add the source file firfilter_kaiser.c to theproject
8. Add the library file "dsk6713bsl.lib" to theproject

   (Path: C:\CCStudio\C6000\dsk6713\lib\dsk6713bsl.lib)
9. Copy files "dsk6713.h" and "dsk6713_aic23.h" to the Projectfolder

   (Path:C:\CCStudio_v3.1\C6000\dsk6713\include)
10. Build (F7) and load the program to the DSP Chip (File->Load Program (.outfile))
11. Run the program(F5)
12. Observe the waveform that appears on the CRO screen and ccstudiosimulator.

// c program for generation of fir filter using c6713 DSK

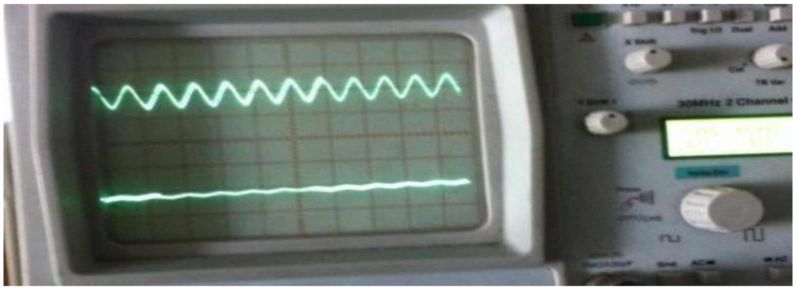
```
#include "firfilter_kaisercfg.h"
```

```
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#include"stdio.h"
float filter_coeff[ ]={0.000000,-0.000138,-0.000611,-0.001345,-0.001607, 0.000000, 0.004714,
0.012033,0.018287,0.016731,0.000000,-0.035687,-0.086763,-0.141588,-0.184011,0.800005,     -
0.184011,-0.141588,-0.086763,-0.035687,0.000000,0.016731,0.018287,0.012033,0.004714,     -
0.000000,-0.001607,-0.001345,-0.000611,-0.000138,0.000000};//FIR High pass Kaiser filter pass band
range 800Hz-3.5KHz
void main()
 {
  DSK6713_AIC23_CodecHandle hCodec;
  Uint32 l_input, r_input,l_output, r_output;
  DSK6713_init();
  hCodec = DSK6713_AIC23_openCodec(0, &config);
  DSK6713_AIC23_setFreq(hCodec, 1);
  while(1)
  {
   while(!DSK6713_AIC23_read(hCodec, &l_input));
  while(!DSK6713_AIC23_read(hCodec, &r_input));
  l_output=(Int16)FIR_FILTER(&filter_coeff ,l_input);
  r_output=l_output;
  while(!DSK6713_AIC23_write(hCodec, l_output));
  while(!DSK6713_AIC23_write(hCodec, r_output));
  }
  DSK6713_AIC23_closeCodec(hCodec);
  }
  signed int FIR_FILTER(float * h, signed int x)
  {
  int i=0;
  signed long output=0;
 static short int in_buffer[100];
 in_buffer[0] = x;
```

```
for(i=30;i>0;i--)
in_buffer[i] = in_buffer[i-1];
for(i=0;i<32;i++)
output = output + h[i] * in_buffer[i];
//output = x;
return(output);
}
```

WAVEFORM:

a) Waveforms of input and output fromCRO

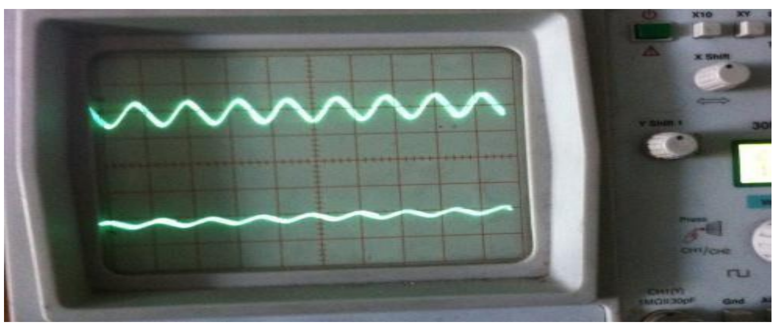


b) function generator (input signal frequency at500Hz)

c) C6713 DSK

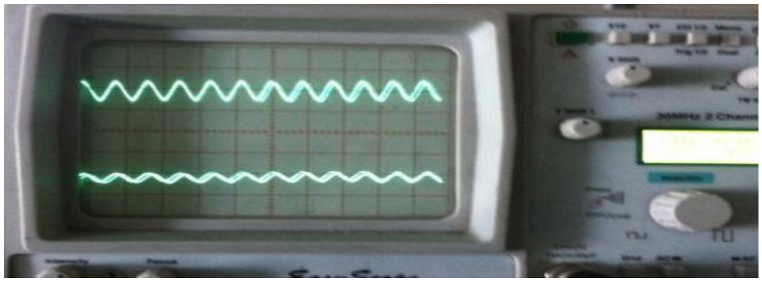
d) function generator (input signal frequency at800Hz)

e) Waveforms of input and output fromCRO



d) function generator (input signal frequency at1.1kHz)

e) Waveforms of input and output fromCRO



Insert Graph Sheet (Normal)

IIR filter using Butterworth Approximation (low pass)

AIM:

To generate a real time iir filter through Butterworth approximation using TMS320C6713 DSK

EQUIPMENTS:

TMS 320C6713 Kit.

RS232 Serial Cable

Power Cord

Operating System – Windows XP

Software – CCStudio_v3.1

PROCEDURE:

1. Connect CRO to the LINE OUTsocket.
2. Now switch ON the DSK and bring up Code Composer Studio onPC
3. Create a new project with namesinewave.pjt
4. From File menu->New->DSP/BIOS Configuration->Select dsk6713.cdb and save it as "iirfliter.cdb"
5. Add firfilter.cdb to the currentproject
6. Create a new source file and save it asiirfilter.c
7. Add the source file iirfilter.c to theproject
8. Add the library file "dsk6713bsl.lib" to theproject

(Path: C:\CCStudio\C6000\dsk6713\lib\dsk6713bsl.lib)

9. Copy files “dsk6713.h” and “dsk6713_aic23.h” to the Projectfolder (Path:C:\CCStudio_v3.1\C6000\dsk6713\include)
10. Build (F7) and load the program to the DSP Chip (File->Load Program(.outfile))
11. Run the program(F5)
12. Observe the waveform that appears on the CRO screen and ccstudiosimulator.

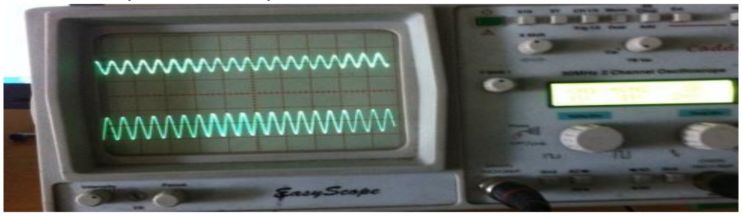
// c program for generation of iir filter using c6713 DSK

```
#include "iirfiltercfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#include "stdio.h"
const signed int filter_coeff[ ] = {15241,15241,15241,32761,10161,7877};
//IIR_BUTERWORTH_LP FILTER pass band range 0-8kHz
DSK6713_AIC23_Config
config={0x0017,0x0017,0x00d8,0x00d8,0x0011,0x0000,0x0000,0x0043,0x0081,0x0001};
void main()
{
 DSK6713_AIC23_CodecHandle hCodec;
 Uint32 l_input, r_input,l_output, r_output;
 DSK6713_init();
 hCodec = DSK6713_AIC23_openCodec(0, &config);
 DSK6713_AIC23_setFreq(hCodec, 3);
 while(1)
 {
 while(!DSK6713_AIC23_read(hCodec, &l_input));
 while(!DSK6713_AIC23_read(hCodec, &r_input));
 l_output=IIR_FILTER(&filter_coeff ,l_input);
 r_output=l_output;
 while(!DSK6713_AIC23_write(hCodec, l_output));
 while(!DSK6713_AIC23_write(hCodec, r_output));
}
```

```
DSK6713_AIC23_closeCodec(hCodec);
}
signed int IIR_FILTER(const signed int * h, signed int x1)
{
 static signed int x[6] ={0,0,0,0,0,0};
static signed int y[6] = {0,0,0,0,0,0};
inttemp=0;
temp = (short int)x1;
x[0] = (signed int) temp;
temp = ( (int)h[0] *x[0]);
temp += ( (int)h[1] *x[1]);
temp += ( (int)h[1] *x[1]);
temp += ( (int)h[2] *x[2]);
temp -= ( (int)h[4] *y[1]);
temp -= ( (int)h[4] *y[1]);
temp -= ( (int)h[5] * y[2]);
temp >>=15;
 if ( temp> 32767 )
 {
 temp = 32767;
 }
 else if ( temp< -32767)
 {
 temp = -32767;
 }
 y[0] = temp;
 y[2] =y[1];
 y[1] =y[0];
 x[2] =x[1];
 x[1] =x[0];
 return (temp<<2);
 }
```

RESULT:

a) Waveforms of input and output fromCRO



b) function generator (input signal frequency at1KHz)

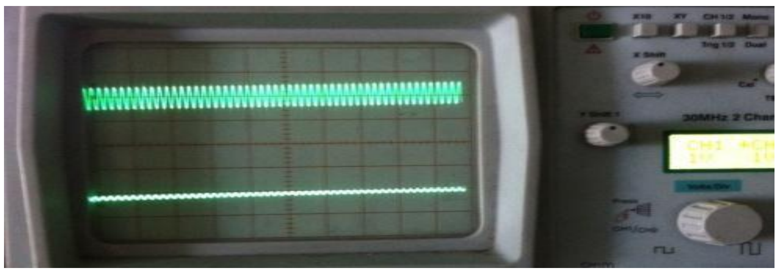
b) C6713DSK

c) function generator (input signal frequency at8kHz)

d) Waveforms of input and output fromCRO

e) function generator (input signal frequency at10kHz)

g) Waveforms of input and output from CRO (output signal attenuated)



Insert Graph Sheet (Normal)

Annexure –I
VIVAQUESTIONS

GENERATION OF SINUSOIDAL SIGNAL QUESTIONS

1. What is the difference between sin & cossignals?
2. What is meant bysignal?
3. What is the difference between time domain & frequency domainsignal?
4. What is the difference between periodic & a periodicsignal.
5. What is the difference between orthogonal and orthonormalsignals?
6. What is the need for Fourier series & Fouriertransform?
7. What is the difference between discrete & digitalsignals?
8. What is the difference between even signal & oddsignal?
9. What is the difference between power signal & energysignal?
10. What is the difference between amplitude scaling & time scaling of asignal?
11. What is the difference between deterministic & randomsignal?

LINEAR CONVOLUTION QUESTIONS

1. What is the requirement forconvolution?.
2. What is the difference between convolution &correlation?
3. What is meant by impulseresponse?
4. Is it possible to represent any discrete time signal in terms of impulses? If yes, represent by usingexample.
5. Drawtheh(2n-k)&h(n-2k)forthefollowingsequenceh(n)={ 4321}assume(i)k=3 (ii) k =5.
6. WritetheexpressionsforLTIsystemconvolutionformula&causalLTIsystem convolutionformula.
7. What us the length of linear convolution if length of input & impulse responses areN1 & N2respectively?
8. What is the difference between continuous and discreteconvolution?

CIRCULAR CONVOLUTION QUESTIONS

1. Why we need circularconvolution?
2. What is the difference between circular & linearconvolution?
3. Whatis the length of output sequence after circular convolution if the lengths of input & impulse responses are M1 & M2respectively?
4. State the circular convolution property ofDFT?
5. Where we required convolutionproperty?
6. What does zero padding mean? Where we required thisconcept?
7. Whatisdifferencebetweenlinearshifting&circularshiftingofsignal?Showwith example.
8. What is difference between linear & circular folding of signal? Show withexample.
9. What is the advantage with sectionedconvolution?

FAST FOURIER TRANSFORM QUESTION

1. What is the difference between continuous time & discrete time Fouriertransform?
2. What is the condition for convergence of Fouriertransform?
3. What is the difference between discrete Time Fourier Transform (DTFT)&DFT?
4. What is the difference between Z transform &DFT?
5. State convolution property of the DFT? Where we could use the convolutionproperty?
6. State Parseval'stheorem.
7. State correlation property of theDFT.?
8. What is the difference between radix 2 & radix4 FFTalgorithms?
9. Why we needFFT?.
10. What is the difference between decimation in time (DIT FFT) & Decimation in frequency(DIFFFT)algorithms?
11. What is meant by 'in-place' computation in DIF & DIFalgorithms?
12. Which properties are used in FFT to reduce no ofcomputations?

FIR FILTER QUESTIONS

1. What are the advantages of FIR as compared toIIR?
2. How many types of FIR design methods are used in realtime?.
3. What is meant by Gibbs Phenomenon? Where we found such type of effect inFIR Filters?

4. What are the advantages& disadvantages of Rectangular window FIR filterasCompared to remaining window techniques?
5. Which window technique having less peak amplitude of side lobe as compared toall?
6. What do you understand by linear phaseresponce?
7. To design all types of filters what would be the expected impulseresponse?
8. What are the properties of FIRfilter?.
9. How the zeros in FIR filter islocated?
10. What are the desirable characteristics of thewindow?
11. What are the specifications required to designfilter

IIR FILTER QUESTIONS

1. What is meant by IIRfilter?
2. What is the difference between recursive & non-recursivesystems?
3. Write the difference equation for IIRsystem.
4. What are the mapping techniques in IIR filter design? Discuss the advantage & disadvantages ofthem.
5. What are IIR analog filters? What are the advantages & disadvantages ofthem?
6. What is the disadvantage in impulse invariancemethod?
7. What does warping effect mean? Where we found this effect? How can we eliminate warpingeffect?
8. Explain the pole mapping procedure of Impulse invariant & bilinear transformation method.
9. For given same specification which difference we found in Butter worth &chebyshev filter.
10. What is the difference between type I & type II chebyshevfilters?.
11. Where the poles are located for Butter worth &chedbyshevfilters?
12. What is meant by spectraltransformation?
13. Why we need spectral transformation in IIRfilter?

POWER SPECTRUM DENSITY QUESTION

1. What is the difference between correlation & auto correlationfunction?
2. What is the difference between PSD &ESD?

3. What is the unit for energy densityspectrum?
4. What is the formula for PSD of afunction?
5. “Same power density spectrum signals always have same magnitude & phase spectrums” Is above statement true (or) False: Justify youranswer.
6. If we know the impulse response of the system, the How can you find output signal power density from the inputsignal?
7. What is the unit for power densityspectrum?
8. What is the relation between auto correlation & PSD of afunction?

DSP PROCESSORS QUESTIONS

1. How many types of DSP processors are available in themarket?
2. TMS 320C6X, ‘C’ stands forwhat?
3. What are the features of TMS 320C6Xprocessor?
4. What is meant by VLIW architecture? Why we required in DSPprocessor?
5. How many functional units are in TMS 320C6X DSPprocessor?
6. What is meant by Circular addressing mode how is it useful forDSP?
7. Which instruction is used to move 16 bit constant in to the upper bits of aregister?
8. What is the difference between Von Neumann architecture & Harvardarchitecture?
9. Which architecture is used in DSPprocessor?
10. How many instructions can we execute per cycle in TMS320C6X DSPprocessor?
11. What are the applications for the TMS320DSP’s?
12. Which soft ware tool is required to compile and run the DSP assemblyprogram?
13. What is the difference between full version Code composer studio &DSKCCS?