METHODIST
COLLEGE OF ENGINEERING AND TECHNOLOGY
Approved by AICTE New Delhi | Affiliated to Osmania University, Hyderabad
Abids, Hyderabad, Telangana, 500001
Estd : 2008

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

SYSTEMS AND SIGNAL PROCESSING LABORATORY

STUDENT LABORATORY MANUAL

(As per 2019-2020Academic Regulations)

B.E V SEMESTER E&CE

SUBJECT CODE: PC 552 EC

Name:_____

Roll No.:_____

METHODIST
COLLEGE OF ENGINEERING AND TECHNOLOGY
Approved by AICTE New Delhi | Affiliated to Osmania University, Hyderabad
Abids, Hyderabad, Telangana, 500001
Estd : 2008

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

SSP LABORATORY

STUDENT LABORATORY MANUAL

(As per 2019-2020 Academic Regulations)

B.E V SEMESTER E&CE

SUBJECT CODE: PC 552 EC

Prepared by

Mr. C.Balarangaswamy

Methodist College of Engineering & Technology

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

Experiment No.	Title of the Experiment	Date	Page No.	Marks				Remarks/ Signature
				E	O	R	T	
1								
2								
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								
13								
14								

E: Experiment (10 Marks) O: Observation (10 Marks) R: Record (5Marks) T: Total (25 Marks)

Experiment No.	Title of the Experiment	Date	Page No.	Marks				Remarks/ Signature
				E	O	R	T	
15								
16								
17								
18								
19								
20								
21								
22								
23								
24								
25								
26								
27								
28								

E: Experiment (10 Marks) O: Observation (10 Marks) R: Record (5Marks) T: Total (25 Marks)

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

Vision of the Institute:

To produce ethical, socially conscious and innovative professionals who would contribute to sustainable technological development of the society.

Mission of the Institute:

To impart quality engineering education with latest technological developments and interdisciplinary skills to make students succeed in professional practice

To encourage research culture among faculty and students by establishing state of art laboratories and exposing them to modern industrial and organizational practices

To inculcate humane qualities like environmental consciousness, leadership, social values, professional ethics and engage in independent and lifelong learning for sustainable contribution to the society

Vision of the Department:

To strive to become centre of excellence in Education, Research with moral, ethical values and serve society

Mission of the Department:

M1: To provide Electronics & Communication Engineering knowledge for successful career either in industry or research

M2: To develop Industry-Interaction for innovation, product oriented research and development.

M3: To facilitate value added education combined with hands-on trainings

Program Educational Objectives:

PEO 1: Apply the knowledge of Basic sciences and Engineering in designing and implementing the solutions in emerging areas of Electronics and Communication Engineering.

PEO 2: Pursue the research or higher education and practise profession.

PEO 3: Adapt to the technological advancements for providing the sustainable Engineering solutions to meet organisation/society needs

PEO 4: Work as an individual or in a team with professional ethics and values.

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

Program Outcomes:

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. 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.

3. 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.

4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. 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.

6. 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.

7. 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.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. 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.

11. 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.

12. 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 (PSOs):

PSO1: Professional Competence: Apply the knowledge of Electronics & Communication Engineering principles in different domains like VLSI, Signal processing, Communication, Embedded system & Control Engineering.

PSO2: **Technical Skills**: Able to design and implement products using the cutting- edge software and hardware tools and hence provide simple solutions to complex problems.

PSO3: Social consciousness: Graduates will be able to demonstrate the leadership qualities and strive for the betterment of organization, environment and society

METHODIST
COLLEGE OF ENGINEERING AND TECHNOLOGY
Approved by AICTE New Delhi | Affiliated to Osmania University, Hyderabad
Abids, Hyderabad, Telangana, 500001
Estd : 2008

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

LAB INSTRUCTIONS

Lab Rule of conduct, DO's & DON'Ts

Conduct yourself in a responsible manner at all times in the laboratory. Don't talk aloud or crack jokes in lab.

A lab coat should be worn during laboratory experiments.

Dress properly during a laboratory activity. Long hair, dangling jewellery and loose or baggy clothing are a hazard in the laboratory.

Observe good housekeeping practices. Replace the materials in proper place after work to keep the lab area tidy.

Do not wander around the room, distract other students, startle other students or interfere with the laboratory experiments of others.

Do not eat food, drink beverages or chew gum in the laboratory and do not use laboratory glassware as containers for food or beverages.

Rules & Guidelines for conducting Lab-Work

Students are not allowed to touch any equipment, in the laboratory area until you are instructed by Teacher or Technician.

Before starting Laboratory work follow all written and verbal instructions carefully. If you do not understand a direction or part of a procedure, ASK YOUR CONCERNED TEACHER BEFORE PROCEEDING WITH THE ACTIVITY.

Before use equipment must be read carefully Labels and instructions. Set up and use the equipment as directed by your teacher. If you do not understand how to use a piece of equipment, ASK THE TEACHER FOR HELP!

Perform only those experiments authorized by your teacher. Carefully follow all instructions, both written and oral. Unauthorized experiments are not allowed in the Laboratory. Students are not allowed to work in Laboratory alone or without presence of the teacher. Any failure / break-down of equipment must be reported to the teacher. Protect yourself from getting electric shock.

METHODIST
COLLEGE OF ENGINEERING AND TECHNOLOGY
Approved by AICTE New Delhi | Affiliated to Osmania University, Hyderabad
Abids, Hyderabad, Telangana, 500001
Estd : 2008

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

Faculty of Engineering O.U. *With effect from Academic Year 2018 – 19*

Course Code	Course Title						Core / Elective
PC552EC	**SYSTEMS AND SIGNAL PROCESSING LAB**						Core
Prerequisite	Contact Hours per Week				CIE	SEE	Credits
SATT PC 304 EC DSP PC 503 EC	L	T	D	P			
	-	-	-	2	25	50	1

Course Objectives:

1. Implement the basic algorithms of DFT,IDFT,FFT and IFFT.
2. Design FIR Filter with specific magnitude and phase requirements.
3. Design IIR Filter with specific magnitude and phase requirements.
4. Describe the basics of Multirate signal processing.
5. Design and implement digital filters on DSP processors.

Course Outcomes:

1. Illustrate various signal processing algorithms.
2. Analyze FIR Filter with specific magnitude and phase requirements.
3. Analyze IIR Filter with specific magnitude and phase requirements.
4. Illustrate the basics of Multirate signal processing.
5. Analyze digital filters on DSP processors.

PART-A

Signal Processing Experiments

1. Introduction to Software used with details of some basics.
2. DFT andFFT algorithm.
3. Linear convolutions.
4. Circular Convolutions.
5. FIR filters design using different window functions.
6. IIR filters design: Butterworth and Chebyshev.
7. Interpolation and Decimation.
8. Implementation of multi-rate systems.
9. Time response of non –linear systems.
10. Design of P, PI, PD and PID controllers (any two)

PART-B
DSP Processor Experiments

1. Introduction to DSP processor kits and Software used with details of some basics.
2. Solution of difference equations.
3. Impulse Response.
4. Linear Convolution.
5. Circular Convolution.
6. Study of procedure to work in real-time.
7. Fast Fourier Transform Algorithms.
8. Design of FIR (LP/HP) USING windows: (a) Rectangular (b) Triangular (c) Hamming windows.
9. Design of IIR (HP/LP) filters.

NOTE:

1. Atleast ten experiments to be conducted in the semester.
2. Minimum of 5 from Part A and 5 from Part B is Compulsory.
3. For Section 'A' MATLAB with different toolboxes like signal processing,signal processing
4. Block set and SIMULINK / MATHEMATICA / any popular software can be used.

Suggested Reading:

1. Jaydeep Chakravorthy, ''Introduction to MATLAB Programming: Toolbox and Simulink",1/e, University Press,2014.

METHODIST
COLLEGE OF ENGINEERING AND TECHNOLOGY
Approved by AICTE New Delhi | Affiliated to Osmania University, Hyderabad
Abids, Hyderabad, Telangana, 500001
Estd : 2008

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

EXPERIMENT 1

GENERATION OF DISCRETE TIME SIGNALS

AIM: - To write a "MATLAB" Program to generate of discrete time signals like unit impulse, unit step, unit ramp, exponential signal and sinusoidal signals.

SOFTWARE REQURIED: - MATLAB

PROCEDURE:-

- Open MATLAB
- Open new M-file
- Type the program
- Save in current directory
- Compile and Run the program
- For the output see command window\ Figure window

PROGRAM:-

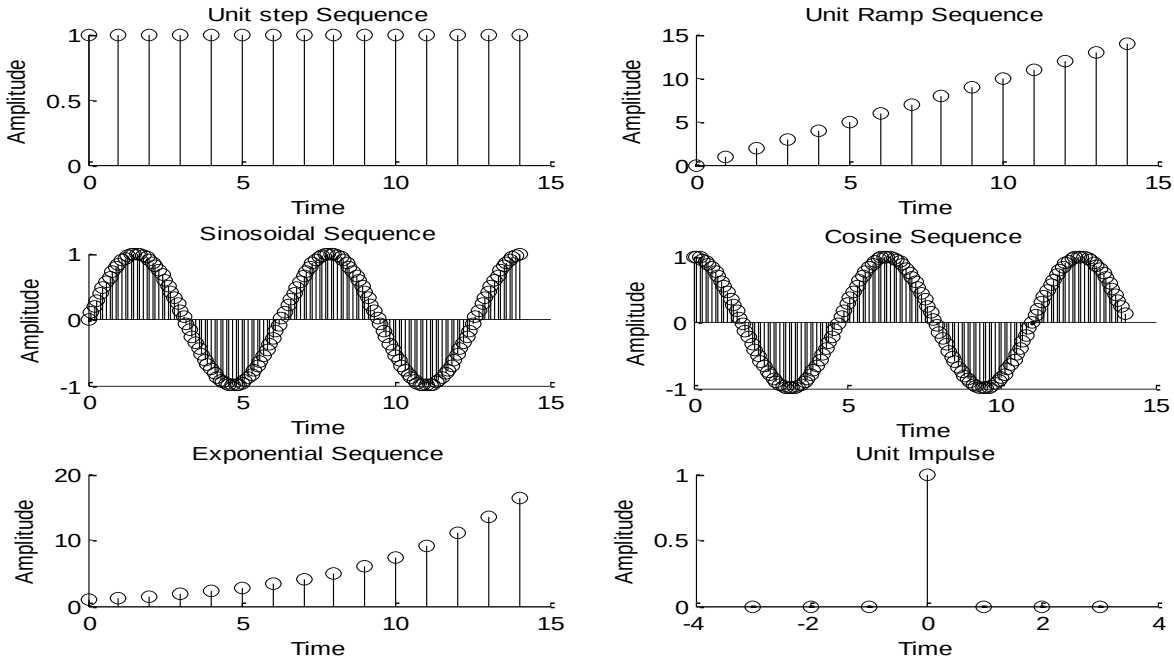
```
% GENERATION OF BASIC DISCRETE TIME SIGNALS

clc;    clear all;  close all;
N=input('Enter the length of unit step sequence(N)= ');
n=0:1:N-1;
y=ones(1,N);
figure;
subplot(3,2,1);
stem(n,y,'k');
xlabel('Time');
ylabel('Amplitude');
title('Unit step Sequence');

% program for Ramp Sequence
N1=input('Enter the length of the unit ramp sequence(N)= ');
n1=0:1:N1-1;
y1=n1;
subplot(3,2,2);
stem(n1,y1,'k');
xlabel('Time');
ylabel('Amplitude');
title('Unit Ramp Sequence');

% program for Sine wave
N2=input('Enter the length of sinusoidal sequence(N)= ');
n2=0:0.1:N2-1;
y2=sin(n2);
subplot(3,2,3);
stem(n2,y2,'k');
xlabel('Time');
```

```
ylabel('Amplitude');
title('Sinosoidal Sequence');

% program for Cosine Wave
N3=input('Enter the length of cosine sequence(N)= ');
n3=0:0.1:N3-1;
y3=cos(n3);
subplot(3,2,4);
stem(n3,y3,'k');
xlabel('Time');
ylabel('Amplitude');
title('Cosine Sequence');

% program for exp sequence

N4=input('Enter the length of Exponential sequence(N)= ');
a=input('Enter the value of Exponential sequence(a)= ');
n4=0:1:N4-1;
y4=exp(a*n4);
subplot(3,2,5);
stem(n4,y4,'k');
xlabel('Time');
ylabel('Amplitude');
title('Exponential Sequence');


% program for Unit Impulse

n5=-3:1:3;
y5=[zeros(1,3),ones(1,1),zeros(1,3)];
subplot(3,2,6);
stem(n5,y5,'k');
xlabel('Time');
ylabel('Amplitude');
title('Unit Impulse');
```

OUTPUT:

OUTPUT FOR GENERATION OF CONTINUOUS TIME SIGNAL:

Enter the length of unit step sequence(N) = 15
Enter the length of the unit ramp sequence(N) = 15
Enter the length of sinusoidal sequence(N) = 15
Enter the length of cosine sequence(N) = 15
Enter the length of Exponential sequence(N) = 15
Enter the value of Exponential sequence(a) = 0.2



EXPERIMENT 2

(A) COMPUTATION OF N POINT DFT

Aim: To compute DFT of the given sequence.

Program:

```
% Program for DFT & IDFT

clc;
close all;
clear all;
x1=input('enter the sequence');
N=length(x1);
n=0:1:N-1;
k=0:1:N-1;
WN=exp(-1i*2*pi/N);
nk=n'*k;
xk=x1*WN.^nk
xkmag=abs(xk)
xn=xk*WN.^(-nk)/N
xnmag=abs(xn)
subplot(3,1,1);
stem(n,x1);
xlabel('n.....>')
ylabel('Amplitude');
title('input sequence');
subplot(3,1,2);
stem(n,xkmag);
xlabel('n.....>')
ylabel('Amplitude');
title('DFT');
subplot(3,1,3);
stem(n,xn);
xlabel('n.....>')
ylabel('Amplitude');
title('IDFT');
```

Output for DFT:

Enter the input sequence(x) = [1 2 3 4 5 6 7 8]

DFT of X = [36.0000, -4.0000 + 9.6569i, -4.0000 + 4.0000i, -4.0000 + 1.6569i, -4.0000, -4.0000 -

1.6569i, -4.0000- 4.0000i, -4.0000 - 9.6569i]

(B) COMPUTATION OF N POINT FFT

Aim: To compute FFT of the given sequence.

Theory:

- Fast Fourier Transform (FFT) is used for performing frequency analysis of discrete time signals. FFT gives a discrete frequency domain representation whereas the other transforms are continuous in frequency domain.
- The N point FFT of discrete time signal x[n] is given by the equation

$$X(k) = \sum_{n=0}^{N-1} x[n] e^{\frac{-j2\pi kn}{N}} \quad ; \quad k = 0,1,2,....N-1$$

Where N is chosen such that $N \geq L$, where L=length of x[n].
FFT is calculated using two algorithms: 1. DIT-FFT, 2. DIF-FFT

Algorithm:

1. Input the sequence for which DFT is to be computed.
2. Input the length of the DFT required (say 4, 8, >length of the sequence).
3. Compute the DFT using the 'fft' command.
4. Plot the magnitude & phase spectra.

MATLAB Program:

```
% DFT using FFT algorithm
clc;    clear all;  close all;
N=input('Enter the fft points (N)= ');
x=input('Enter the input sequence(x)= ');
xlen=length(x)
n=0:1:(xlen-1);
subplot(2,2,1);
stem(n,x,'k');
xlabel('N--->');
ylabel('Amplitude');
title('Input sequence');
k=0:1:(N-1);
X=fft(x,N)
subplot(2,2,2);
stem(k,abs(X),'k');
xlabel('k--->');
ylabel('Amplitude');
title('Magnitude Plot(FFT)');
subplot(2,2,3);
stem(k,angle(X),'k');
xlabel('k--->');
ylabel('Angle');
```

```
title('Phase Plot(FFT)');
n=0:1:(N-1);
x=ifft(X,N)
subplot(2,2,4);
stem(n,abs(x),'k');
xlabel('n--->');
ylabel('Amplitude');
title('ifft signal(input signal)');
```

Output for FFT:

Enter no of fft points N= 8;

Enter the input sequence(x) = [1 2 3 4 5 6 7 8]

Fft of X =[36.0000 , -4.0000 + 9.6569i , -4.0000 + 4.0000i , -4.0000 + 1.6569i , -4.0000 , -4.0000 –

1.6569i, -4.0000- 4.0000i, -4.0000 - 9.6569i]

EXPERIMENT 3

LINEAR CONVOLUTION

Aim: To obtain convolution of two finite duration sequences.

Theory:

- The output y[n] of a LTI (linear time invariant) system can be obtained by convolving the input x[n] with the system's impulse response h[n].
- The convolution sum is $y[n] = x[n] * h[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k] = \sum_{k=-\infty}^{+\infty} x[n-k]h[k]$
- x[n] and h[n] can be both finite or infinite duration sequences.
- If both the sequences are of finite duration, then we can use the MATLAB function '***conv***' to evaluate the convolution sum to obtain the output y[n].
- The length of y[n] = x(n)length + h(n)length -1.

Algorithm:

1. Input the two sequences as xn, hn
2. Convolve both to get output yn.
3. Plot the sequences.

MATLAB Program:

```
% Program for linear convolution
Clc;
close all;
clear all;
x=input('enter the first sequence');
h=input('enter the second sequence ');
y=conv(x,h);
subplot(3,1,1);
stem(x);
xlabel('time period');
ylabel('amplitude');
title('First  sequence');
subplot(3,1,2);
stem(h);
xlabel('time period');
ylabel('amplitude');
title('second  sequence');
subplot(3,1,3);
stem(y);
xlabel('time period');
ylabel('amplitude');
title('Linear convolution of two  sequences');
```

OUTPUT FOR LINEAR CONVOLUTION:

Enter the input sequence x(n) = [1 2 3 4]

Enter the impulse sequence h(n) = [1 2 3 1]

y = 1 4 10 17 19 15 4

EXPERIMENT 4

CIRCULAR CONVOLUTION

Aim: To obtain circular convolution of two finite duration sequences.

Theory: The circular convolution sum is $y[n] = x(n) \otimes h[n] = \sum_{k=-\infty}^{+\infty} x[k]h[\langle n-k \rangle_N]$ where the index $\langle n-k \rangle_N$ implies circular shifting operation and $\langle -k \rangle_N$ implies folding the sequence circularly.

- Steps for circular convolution are the same as the usual convolution, except all index calculations are done "mod N" = "on the wheel".
  - Plot f [m] and h [−m] as shown in Fig. 4.1. (use f(m) instead of x(k))
  - Multiply the two sequences
  - Add to get y[m]
  - "Spin" h[−m] n times Anti Clock Wise (counter-clockwise) to get h[n-m].
- x[n] and h[n] can be both finite or infinite duration sequences. If infinite sequences, they should be periodic, and the N is chosen to be at least equal to the period. If they are finite sequences N is chosen as >= to max (xlength, hlength). Whereas in linear convolution N>= xlength+hlength-1.
- Say $x[n] = \{-2,3,\underset{\uparrow}{1},1\}$ and N = 5, then the x[-1] and x[-2] samples are plotted at x[N-1] = x[-4] and x[N-2] =x[-3] places. Now x[n] is entered as $x[n] = \{\underset{\uparrow}{1},1,0,-2,3\}$.

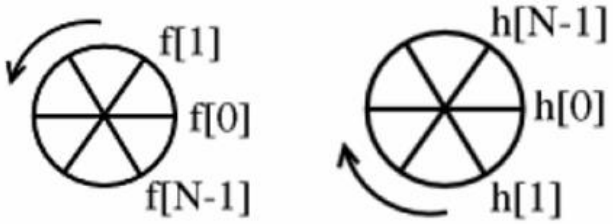


Fig. 4.1 Plotting of f(m) and h(-m) for circular convolution

Algorithm:

1. Input the two sequences as x and h.
2. Circularly convolve both to get output y.
3. Plot the sequences.

MATLAB Program:

```
%Program for circular convolution

clc;
close all;
clear all;
x=input('enter first sequence')
N1=length(x);
h=input('enter second sequence')
N2=length(h);
M=max(N1,N2);
y=cconv(x,h,M);
subplot(3,1,1);
stem(x);
xlabel('x(a)');
ylabel('amplitude');
title('first sequence');
subplot(3,1,2);
stem(h);
xlabel('h(a)');
ylabel('amplitude');
title('second sequence');
subplot(3,1,3);
stem(y);
xlabel('x(a)');
ylabel('amplitude');
title('output sequence');
disp('convolved output');
disp(y);
```

OUTPUT FOR CIRCULAR CONVOLUTION:

Enter the impulse sequence x1(n) = [1 2 3 4]
Enter the impulse sequence x2(n) = [1 2 3 1]
Circular Convolution of x1(n) & x2(n) is = 20 19 14 17

EXPERIMENT 5

DESIGN OF LOW PASS FIR FILTER

Aim: To design Low pass FIR filter using Rectangular, Hamming and Kaiser Windows for given specifications.

Theory:

There are two types of systems – Digital filters (perform signal filtering in time domain) and spectrum analyzers (provide signal representation in the frequency domain). The design of a digital filter is carried out in 3 steps- specifications, approximations and implementation.

DESIGNING AN FIR FILTER (using window method):

Method I: Given the order N, cutoff frequency fc, sampling frequency fs and the window.

- Step 1: Compute the digital cut-off frequency Wc (in the range $-\pi < Wc < \pi$, with π corresponding to fs/2) for fc and fs in Hz. For example let fc=400Hz, fs=8000Hz

  $Wc = 2*\pi* fc / fs = 2* \pi * 400/8000 = 0.1* \pi$ radians

  For MATLAB the Normalized cut-off frequency is in the range 0 and 1, where 1 corresponds to fs/2 (i.e., fmax)). Hence to use the MATLAB commands

  wc = fc / (fs/2) = 400/(8000/2) = 0.1

  Note: if the cut off frequency is in radians then the normalized frequency is computed as wc = Wc / π
- Step 2: Compute the Impulse Response h(n) of the required FIR filter using the given Window type and the response type (lowpass, bandpass, etc). For example given a rectangular window, order N=20, and a high pass response, the coefficients (i.e., h[n] samples) of the filter are computed using the MATLAB inbuilt command 'fir1' as

  h =fir1(N, wc , 'high', boxcar(N+1));

  Note: In theory we would have calculated h[n]=hd[n]×w[n], where hd[n] is the desired impulse response (low pass/ high pass,etc given by the sinc function) and w[n] is the window coefficients. We can also plot the window shape as stem(boxcar(N)).

  Plot the frequency response of the designed filter h(n) using the freqz function and observe the type of response (lowpass / highpass /bandpass).

PROGRAM:

PROGRAM (1)

```
% LPF using Rectangular Window

clc;    clear all;  close all;
N=input('enter the order of the filter');
fp=input('enter the pass band frequency');
fs=input('enter the sampling frequency');
fn=2*fp/fs;
window=rectwin(N+1);
b=fir1(N,fn,window);
w=0:0.001:pi;
[h,om]=freqz(b,1,w);
a=20*log10(abs(h));
subplot(2,1,1);
plot(w/pi,a);
xlabel('Normalized frequency')
ylabel('gain in db')
title('magnitude plot of LPF');
b=angle(h);
subplot(2,1,2);
plot(w/pi,b);
xlabel('Normalized frequency')
ylabel('phase in radian')
title('phase response of LPF');
```

OUTPUT:

PROGRAM (2)

```
% LPF using Hamming Window

clc;    clear all;  close all;
N=input('enter the order of the filter');
fp=input('enter the pass band frequency');
fs=input('enter the sampling frequency');
fn=2*fp/fs;
window=hamming(N+1);
b=fir1(N,fn,window);
w=0:0.001:pi;
[h,om]=freqz(b,1,w);
a=20*log10(abs(h));
subplot(2,1,1);
plot(w/pi,a);
xlabel('Normalized frequency')
ylabel('gain in db')
title('magnitude plot of LPF');
b=angle(h);
subplot(2,1,2);
plot(w/pi,b);
xlabel('Normalized frequency')
ylabel('phase in radian')
title('phase response of LPF');
```

OUTPUT:

PROGRAM (3)

```
% LPF using Kaiser Window
clc;clear all;close all;
M = input ('enter the length of the filter = ');
beta= input ('enter the value of  beta = ');
Wc = input ('enter the digital cutoff frequency');
Wn= kaiser(M,beta);
disp('FIR Kaiser window coefficients');
disp(Wn);
hn = fir1(M-1,Wc,Wn);
disp('the unit sample response of FIR filter is hn=');
disp(hn);
freqz(hn,1,512);
grid on;
xlabel('normalized frequency');
ylabel('gain in db');
title('freq response of FIR filter');
```

Result:

```
enter the length of the filter        = 30
enter the value of  beta              = 1.2
enter the digital cutoff frequency = 0.3
```

EXPERIMENT 6

DESIGN AND IMPLEMENTATION OF IIR FILTERS

Aim: To design and implement an IIR filter for given specifications.

Theory:

There are two methods of stating the specifications as illustrated in previous program. In the first program, the given specifications are directly converted to digital form and the designed filter is also implemented. In the last two programs the butter worth and chebyshev filters are designed using bilinear transformation (for theory verification).

Given the pass band (Wp in radians) and Stop band edge (Ws in radians) frequencies, Pass band ripple Rp and stopband attenuation As.

- Step 1: Since the frequencies are in radians divide by π to obtain normalized frequencies to get wp=Wp/pi and ws=Ws/pi
  If the frequencies are in Hz (note: in this case the sampling frequency should be given), then obtain normalized frequencies as wp=fp/(fs/2), ws=fstop/(fs/2), where fp, fstop and fs are the passband, stop band and sampling frequencies in Hz
- Step 2: Compute the order and cut off frequency as
  [N, wc] = BUTTORD(wp, ws, Rp, Rs)
- Step 3: Compute the Impulse Response [b,a] coefficients of the required IIR filter and the response type as [b,a] =butter(N, wc , 'high');

IMPLEMENTATION OF THE IIR FILTER

1. Once the coefficients of the IIR filter [b,a] are obtained, the next step is to simulate an input sequence x[n], say input of 100, 200 & 400 Hz (with sampling frequency of fs), each of 20/30 points. Choose the frequencies such that they are >, < and = to fc.
2. Filter the input sequence x[n] with Impulse Response, to obtain the output of the filter y[n] using the ‘filter’ command.
3. Infer the working of the filter (low pass/ high pass, etc).

PROGRAM (1)

% BUTTERWORTH LOW PASS FILTER

```
clc;
 clear all;
 close all;
rp = input('pass ripple freqency=');
rs = input('stop ripple frequency=');
fp = input('pass band frequency=');
fs = input('stop band frequency=');
f = input('sampling frequency=');
w1=2*fp/f;
w2=2*fs/f;
[n,wn]= buttord(w1,w2,rp,rs);
[b,a] = butter(n,wn);
w=0:.1:pi;
[h,p] = freqz(b,a,w);
g=20*log10(abs(h));
A=angle(h);
subplot (2,2,1);
plot(p/pi,g);
ylabel('amplitude');
xlabel('freqency');
title('magnitude response');
subplot (2,2,2);
plot(p/pi,A);
xlabel('frequency');
ylabel('phase');
title('phase response');
```

OUTPUT:

pass ripple freqency=5

stop ripple frequency=40

pass band frequency=2000

stop band frequency=3000

sampling frequency=10000

PROGRAM (2)

% BUTTERWORTH HIGH PASS FILTER

```
clc;
clear all;
close all;
rp=input('pass ripple frequency=');
rs=input('stop ripple frequency=');
fp=input('pass band frequency=');
fs=input('stop band frequency=');
f=input('sampling frequency=');
w1= 2*fp/f;
w2= 2*fs/f;
[n,wn]= buttord(w1,w2,rp,rs);
[b,a] = butter(n,wn,'high');
w=0:.1:pi;
[h,p]= freqz(b,a,w);
g=20*log10(abs(h));
A=angle(h);
subplot (2,2,1);
plot(p/pi,g);
xlabel('frequency---->');
ylabel('amplitude--->');
title('magnitude plot');
subplot (2,2,2);
plot(p/pi,A);
xlabel('frequency--->');
ylabel('phase---->');
title('phase plot');
```

OUTPUT:

pass ripple freqency=5

stop ripple frequency=40

pass band frequency=2000

stop band frequency=3000

sampling frequency=10000

PROGRAM (3)

% CHEBYSHEV LOW PASS FILTER

```
clc; clear all; close all;
rp = input('pass ripple freqency=');
rs = input('stop ripple frequency=');
fp = input('pass band frequency=');
fs = input('stop band frequency=');
f = input('sampling frequency=');
w1=2*fp/f;
w2=2*fs/f;
[n,wn]= cheb1ord(w1,w2,rp,rs);
[b,a] = cheby1(n,wn);
w=0:.1:pi;
[h,p] = freqz(b,a,w);
g=20*log10(abs(h));
A=angle(h);
subplot (2,2,1);
plot(p/pi,g);
ylabel('amplitude');
xlabel('freqency');
title('magnitude response');
subplot (2,2,2);
plot(p/pi,A);
xlabel('frequency');
ylabel('phase');
title('phase response');
disp(‘cutoff frequency’);
disp(wc);
```

OUTPUT:

pass ripple freqency=5

stop ripple frequency=40

pass band frequency=2000

stop band frequency=3000

sampling frequency=10000

PROGRAM (4)

% CHEBYSHEV HIGH PASS FILTER

```
clc;
clear all;
close all;
rp=input('pass ripple frequency=');
rs=input('stop ripple frequency=');
fp=input('pass band frequency=');
fs=input('stop band frequency=');
f=input('sampling frequency=');
w1= 2*fp/f;
w2= 2*fs/f;
[n,wn]= cheb1ord (w1,w2,rp,rs);
[b,a] = cheby1 (n,wn,'high');
w=0:.1:pi;
[h,p]= freqz(b,a,w);
g=20*log10(abs(h));
A=angle(h);
subplot (2,2,1);
plot(p/pi,g);
xlabel('frequency---->');
ylabel('amplitude--->');
title('magnitude plot');
subplot (2,2,2);
plot(p/pi,A);
xlabel('frequency--->');
ylabel('phase---->');
title('phase plot');
disp('cutoff frequency');
disp(wc);
```

OUTPUT:

pass ripple freqency=5
stop ripple frequency=40
pass band frequency=2000
stop band frequency=3000
sampling frequency=10000

EXPERIMENT 7

INTERPOLATION AND DECIMATION

a) Write a Matlab program to DOWN SAMPLE the signal x(n) by sampling rate reduction by a factor of a)2 b)3

Aim: -

To down sample the signal x(n) by sampling rate reduction by a factor of a)2 b)3

Procedure:-

Open MATLAB Open new M-file

Type the program Save in the current directory Compile and run the program For the output see the command window/figure window.

MATLAB CODE:

```
clc;
clear all;
close all;
display('The Given Signal is,');
xn=[1,-1,1,-1,2,-2,2,-2,3,-3,3,-3]
N=length(xn);
n=0:1:N-1;
subplot(3,1,1);
stem(n,xn,'k');
xlim([0 12]);
ylim([-3 3]);
xlabel('n');
ylabel('x(n)');
title('signal x(n)');
d=2;
display('The Signal downsampled by reduction factor 2 is');
xd2n=xn(1:d:N)
n1=1:1:N/d;
subplot(3,1,2);
stem(n1-1,xd2n,'k');
xlim([0 12]);
ylim([-3 3]);
xlabel('n');
ylabel('x_d_2(n)');
title('Down sampled signal,D=2',);
d=3;
display('The Signal downsampled by reduction factor 3 is');
```

```
 xd3n=xn(1:d:N)
 n1=1:1:N/d;
subplot(3,1,3);
stem(n1-1,xd3n,'k');
 xlim([0 12]);
ylim([-3 3]);
xlabel('n');
ylabel('x_d_3(n)');
title('Down sampled signal,D=3');
```

OUTPUT:

b) Write a Matlab program to UP SAMPLE the signal x(n) by sampling rate by a factor of a)2 b)3

```
clc;
clear all;
close all;
display('The Given Signal is,');
xn=[1,-1,2,-2] N=length(xn);
 n=0:1:N-1;
subplot(3,1,1);
stem(n,xn,'k');
xlim([0 12]);
ylim([-3 3]);
 xlabel('n');
 ylabel('x(n)');
 title('signal x(n)');
I=2;
 xi2n=[zeros(1,I*N)];
 n1=1:1:I*N;
 j=1:I:I*N;
 display('The Signal Upsampled BY Multiplication Factor 2 is');
xi2n(j)=xn
 subplot(3,1,2);
stem(n1,xi2n,'k');
xlim([0 12]);
 ylim([-3 3]);
xlabel('n');
ylabel('x_I_2(n)');
title('Upsampled signal ,I=2');
 I=3;
 xi3n=[zeros(1,I*N)];
 n1=1:1:I*N;
 j=1:I:I*N;
display('The Signal Upsampled BY Multiplication Factor 2 is');
 xi3n(j)=xn
subplot(3,1,3);
stem(n1,xi3n,'k');
xlim([0 12]);
ylim([-3 3]);
xlabel('n');
ylabel('x_I_2(n)');
title('Upsampled signal ,I=3');
```

OUTPUT

EXPERIMENT 8

CODE COMPOSER STUDIO EXPERIMENTS

INTRODUCTION TO TMS320C6713 PROCESSOR

The TMS C6713 DSK is low cost standalone development platforms that enable users to evaluate and develop applications for the TI 67XX DSP family. The DSK also serves as a hardware reference design for the TMS 320C6713 DSP. Schematics, logic equations and application notes are available ease to hardware development and reduce time to market. The DSK comes with a full complement of onboard devices that suite a wide variety of application environment. Key features include:

1) A TEXAS Instrument TMS 320 C6713 DSP operates at 225 MHz
2) An AIC23 stereo codec
3) 16 Mega Bytes of synchronous DRAM 4) 512 KB of non Volatile flash memory
5) 4 users accessible LED's and DIP switches.
6) Software board configuration through registered implemented in CPLD.
7) Configurable boot options
8) Standard expansion connector for daughter card use.
9) JTAG emulation through on-board JTAG emulator with USB host interface or external emulator
10) Single voltage power supply +5V.

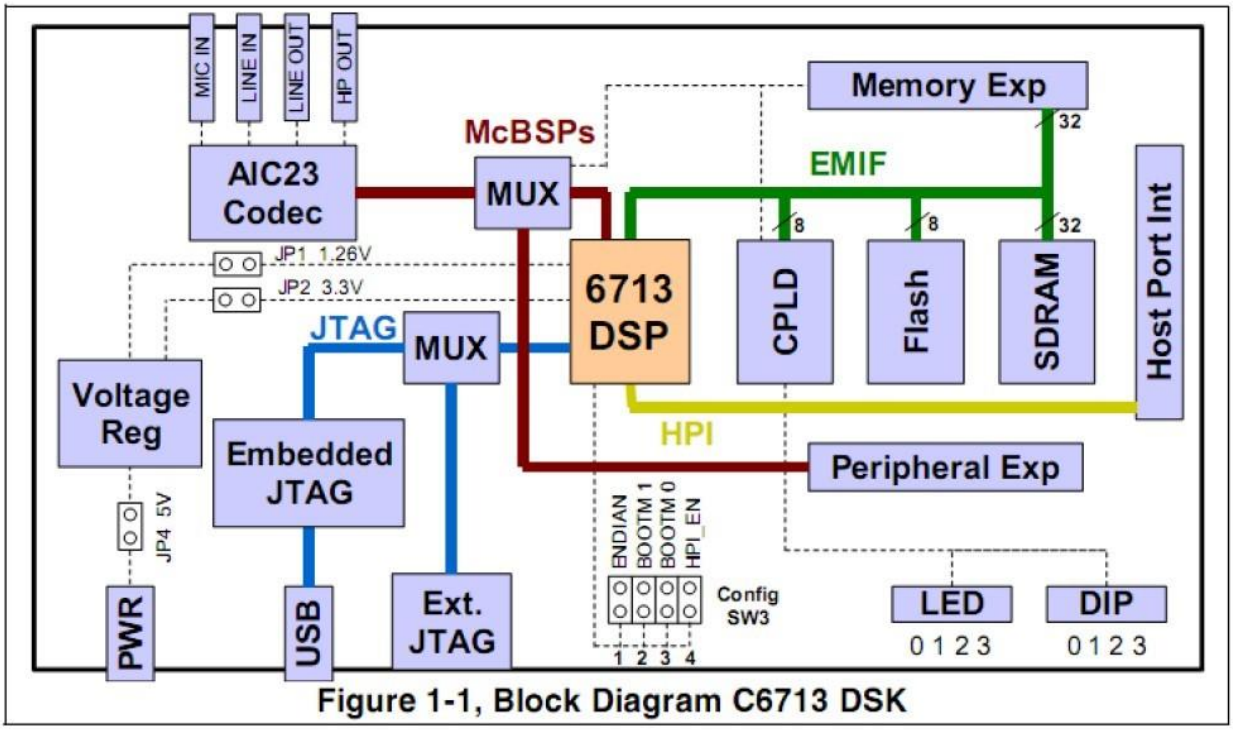


Figure 1-1, Block Diagram C6713 DSK

FUNCTIONAL OVERVIEW OF TMS 320C6713 PROCESSOR

The DSP on the 6713 DSK interfaces to on board peripherals through a 32-bit wide EMIF (External Memory Interface). The SDRAM, Flash and CPLD are 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 onboard AIC23CODEC and 4 three point audio jack (microphone input, line input, line output and head phone output).

The CODEC can select the microphone are the line input as the active input. The analog output is driven to both the line output (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. These ports 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 registered based user interface that lets the user configure the board by reading and writing to its register. The 4 LED's and 4 position DIP switches are accessible via CPLD registers.

An included 5V external power supply is used to power the board. Onboard switching voltage regulator provides +1.26V DSP core voltage and +3.3V I/O supplies. The board is held in reset until the supplies are within the operating specifications.
Code composer communicates with the DSK through an embedded JTAG emulator with a USB host interface. The DSP can also be used with an external emulator though external JTAG connector

CODE COMPOSER STUDIO :

STEPS TO EXECUTE A PROGRAM USING CCS:

1. In the main screen go to project - - > new.. The following screen will appear . Give a appropriate project name, project type as executable (.out) file. Select target as TMS320C67xx (depends on the target processor we are using, in our case it"s 67xx), and click finish in dialog box finally.
2. Now a new project will be created (e.g. linear.pjt) and a tree will be displayed on left hand side.
3. Now we need to write C source code. Go to File - - > New - - > Source file.
4. Now a blank screen will appear in which we have to write our C source code. Type the source code and click save.
5. We have to save the source code file as .c file .When we press save iconin main screen, following dialog box will open. Save the file as C\C++ source file (e.g. „lin.c").
6. Now we have to add 3 files to our project. C source code (e.g. lin.c).

   A library file (e.g. rts6700.lib).

   A linker command file (e.g. hello.cmd).

Firstly add the source code file.

Go to Project - - > Add files to project...

Now dialog box will open, here browse for the source code file in projects directory and click open. (Usually the common paths are ...\ti\MyProjects\"projectname"\...)

Similarly add two more files following same procedure.

First, add a library file from path...\ti\c6000\cgtools\lib\rts6700.lib (here file type is object and library file).

Secondly, add a linker command file from path …\ti\tutorial\dsk6713\hello1\hello.cmd (here the file type is linker command file *.lcf,*.cmd).

7. After the addition of all three files, the expanded project tree on left hand side will resemble this format. (Observe the LHS project tree which contains a library file, a source code and linker command file.)
8. Now we need to compile and build our program.

To compile go to project -- > compile file. Now the project will be compiled and any errors will be displayed below if there.

To build go to project - - >build. Now project will be built and any errors or warnings will be displayed below.

9. Now we need to load program in CPU or simulator.

   Go to file - - > load program.

   A dialog box will open in which we have select executable .out file.

   The usual path is …\ti\MyProjects\linear\debug\linear.out
10. Now we can assembly equivalent of C source code which the simulatorhas generated.
11. To see the output of the program we have to run the program.

    Go to Debug - - > Run.
12. Now we can see output of the program below.

    If the program needs any user input, a pop-up window will open.
13. Now we need to see the plot of output.

    Go to View - - > Graph - - > Time\frequency..

    Now a graph property dialog box will open up.

    Select Display type = single time.

    Graph title = „Any name".

    Start address = " the variable which we want to display".

    Index increment= 1(usually).

    DSP data type = 32-bit signed integer.

    Data plot style = bar(for displaying discrete values).

    Grid style = Full grid.

    Now the graph will be displayed.

*We can select different elements by using left and right keys of keyboard.

EXPERIMENT 9

IMPULSE RESPONSE OF THE SYSTEM

PROBLEM: Compute the response of the system whose coefficients are a0=0.1311, a1=0.2622, a2=0.1311 and b0=1, b1= -0.7478, b2=0.2722, when the input is a unit impulse signal.

```
#include <stdio.h>
 float y[3]={0,0,0};
 float x[3]={0,0,0};
 float z[10]; float
impulse[10]={1,0,0,0,0,0,0,0,0,0};
main()
{
int j;
float a[3]={0.1311, 0.2622, 0.1311};
float b[3]={1, -0.7478, 0.2722};
 for(j=0;j<10;j++)
{ x[0]=impulse[j]; y[0] = (a[0] *x[0]) +(a[1]* x[1] ) +(x[2]*a[2]) - (y[1]*b[1])-
(y[2]*b[2]); printf("%f\n",y[0]);
z[j]=y[0];
y[2]=y[1]; y[1]=y[0];
 x[2]=x[1]; x[1] = x[0];
}
}
```

OUPUT :

EXPERIMENT 10

DESCRETE TIME CONVOLUTION OF TWO SEQUENCES

PROBLEM: Find the Discrete Time Convolution of the sequences x(n)={1 2 3 4 5} and h(n)={1 2 3 4}, using Code Composer Studio. Verify the results graphically.

```
#include<stdio.h>
int y[20];
 main()
 {
 int m=6;
int n=6;
 int i=0,j;
 int x[15]={1,2,3,4,5,6,0,0,0,0,0,0};
 int h[15]={1,2,3,4,5,6,0,0,0,0,0,0};
for(i=0;i<m+n-1;i++)
{
 y[i]=0;
for(j=0;j<=i;j++)
y[i]+=x[j]*h[i-j];
} for(i=0;i<m+n-1;i++)
printf("%d\n",y[i]);



}
```

Input:

x(n)=[1,2,3,4,5,6]

h(n)=[1,2,3,4,5,6]

Output:

y(n)=[1,4,10,20,35,56,70,76,73,60,36]

EXPERIMENT 11
CIRCULAR CONVOLUTION

PROBLEM: Perform the Circular Convolution of the given sequence using Code Composer Studio.

```
#include<stdio.h>
int m,n,x[30],h[30],y[30],i,j,temp[30],k,x2[30],a[30];
void main()
{
printf(" enter the length of the first sequence\n");
scanf("%d",&m);
printf(" enter the length of the second sequence\n");
scanf("%d",&n);
 printf(" enter the first sequence\n");
 for(i=0;i<m;i++)
 scanf("%d",&x[i]);
printf(" enter the second sequence\n");
for(j=0;j<n;j++)
 scanf("%d",&h[j]);
 if(m-n!=0)
{
 if(m>n)
 {
for(i=n;i<m;i++)
h[i]=0;
n=m;
}
for(i=m;i<n;i++)
x[i]=0;
 m=n;
}
y[0]=0;
 a[0]=h[0];
for(j=1;j<n;j++)
 a[j]=h[n-j];
 for(i=0;i<n;i++)
y[0]+=x[i]*a[i];
for(k=1;k<n;k++)
```

```
{
 y[k]=0;
for(j=1;j<n;j++)
x2[j]=a[j-1];
x2[0]=a[n-1];
for(i=0;i<n;i++)
{
 a[i]=x2[i];
y[k]+=x[i]*x2[i];
}

 }

printf(" the circular convolution is\n");
for(i=0;i<n;i++)

 printf("%d \t",y[i]);

 }
```

INPUT:

Length of first sequence = 4
Length of second sequence = 4
First sequence =[1 2 2 1]
Second sequence =[1 2 2 1]

OUTPUT:

The circular convolution is [9 8 9 10]

EXPERIMENT 12

N-POINT DFT COMPUTATION

PROBLEM: compute the DFT of the given sequence x(n)={1,3,2,4,1,6,4,7} using code composer.

```
#include<stdio.h> #include<math.h> void
main(void) // DFT main function
{
short N=8; short x[8]={1,3,2,4,1,6,4,7}; //Test
data float pi=3.1416;
float sumre=0,sumim=0;
 float cosine=0,sine=0;
 int i=0,k=0,n=0;
 for(k=0;k<N,k++)
{
sumre=0;
sumim=0;
for(n=0;n<N;n++)
{
cosine=cos(2*pi*k*n/N); //Real
sine=sin(2*pi*k*n/N); //imaginary
sumre=sumre+x[n]*cosine;
sumim=sumim-x[n]*sine;
}
out_real[k]=sumre;
out_img[k]=sumim;
Printf("[%d] %7.3f %7.3f\n",k,out_real[k],out_img[k]);
}
}
```

INPUT:

Coefficient of x(n)={0.1311,0.2622,0.1311}

Coefficient of y(n)={1,-0.7478,0.2722}

OUTPUT:

0.131100 0.360237
0.364799 0.174741
0.031373
-0.024104
-0.026565
-0.013304
-0.002718
0.001589