Mapping of COs to POs

Course Name: **Digital Signal Processing Laboratory** Course Code: **15ECL57**

CO's	DESCRIPTION OF THE OUTCOMES
15ECL57.1	**Understand** the basic concepts of Digital Signal processing techniques with their Properties both in time and frequency domain.
15ECL57.2	**Design and develop** the software modules for digital filters and digital signal processing techniques/operations using Modern Software Tools.
15ECL57.3	**Implement** Digital signal processing techniques/operations and Digital filters using DSP Processor.
15ECL57.4	**Documentation** of the complete experimental process and results.

CO No	PO No												PSO		
	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3
15ECL57.1	3	3								2			3		
15ECL57.2	3	3	3		3								3	3	
15ECL57.3			2		3								2	3	
15ECL57.4										3					
CO Average	**3**	**3**	**2.5**		**3**					**2.5**			**2.66**	**3**	

Prof. Devendran B **Prof. Dhanushree V.** **Prof. Niveditha H.R.**	**Prof. Anisha P.S.** **Prof. Nihal Mohammadi.** **Prof. Suma R.**	**Prof. Devendran B.**
Faculty		**Course Coordinator**

Criterion 3 Coordinator	**NBA coordinator**	**HOD**
Convener		**Principal**

MAHARAJA INSTITUE OF TECHNOLOGY MYSORE

Belawadi, S.R. Patna Taluk, Mandya – 571438

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

LESSON PLAN (Batch - 1)

Subject Name: **Digital Signal Processing Lab**

Subject Code	:	**15ECL57**	Academic Year:	**2018-19**
Hours/week (L.T.P)	:	**1.0.2**	Exam Hours:	**03**
IA Marks	:	**20**	Exam Marks:	**80**

	Topics Proposed to be Covered	Proposed Date of Coverage	Actual Date of Coverage
Introduction class			
	Cycle - 1		
1.	Verification of sampling theorem.		
2.	Linear and circular convolution of two given sequences, Commutative, distributive and associative property of convolution.		
3.	Auto and cross correlation of two sequences and verification of their properties.		
	Cycle - 2		
4.	Solving a given difference equation.		
5.	Computation of N point DFT of a given sequence and to plot magnitude and phase spectrum (using DFT equation and verify it by built-in routine).		
6.	(i) Verification of DFT properties (like Linearity and Parseval's theorem, etc.) (ii) DFT computation of square pulse and Sinc function etc.		
	Cycle - 3		
7.	Design and implementation of FIR filter to meet given specifications (using different window techniques).		
8.	Design and implementation of IIR filter to meet given specifications.		
	Cycle - 4		
11.	Linear convolution of two sequences		
12.	Circular convolution of two sequences		
13.	N-point DFT of a given sequence		
14.	Impulse response of first order and second order system		
15.	Implementation of FIR filter		

Signature of Staff Member(s) Signature of NBA Coordinator Signature of HoD

MAHARAJA INSTITUE OF TECHNOLOGY MYSORE

Belawadi, S.R. Patna Taluk, Mandya – 571438

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

LESSON PLAN (Batch - 2)

Subject Name: **Digital Signal Processing Lab**

Subject Code	:	**15ECL57**	Academic Year:	**2018-19**
Hours/week (L.T.P)	:	**1.0.2**	Exam Hours:	**03**
IA Marks	:	**20**	Exam Marks:	**80**

	Topics Proposed to be Covered	Proposed Date of Coverage	Actual Date of Coverage
Introduction class			
	Cycle - 1		
4.	Verification of sampling theorem.		
5.	Linear and circular convolution of two given sequences, Commutative, distributive and associative property of convolution.		
6.	Auto and cross correlation of two sequences and verification of their properties.		
	Cycle - 2		
4.	Solving a given difference equation.		
5.	Computation of N point DFT of a given sequence and to plot magnitude and phase spectrum (using DFT equation and verify it by built-in routine).		
6.	(i) Verification of DFT properties (like Linearity and Parseval's theorem, etc.) (ii) DFT computation of square pulse and Sinc function etc.		
	Cycle - 3		
7.	Design and implementation of FIR filter to meet given specifications (using different window techniques).		
8.	Design and implementation of IIR filter to meet given specifications.		
	Cycle - 4		
11.	Linear convolution of two sequences		
12.	Circular convolution of two sequences		
13.	N-point DFT of a given sequence		
14.	Impulse response of first order and second order system		
15.	Implementation of FIR filter		

Signature of Staff Member(s) Signature of NBA Coordinator Signature of HoD

SEMESTER

5

MAHARAJA INSTITUTE OF TECHNOLOGY MYSORE

Department of Electronics & Communication Engineering

Name:_____

USN :_____

Manual for

Digital Signal Processing Laboratory (15ECL57)

DEPT. OF ELECTRONICS AND COMMUNICATION ENGINEERING

Digital Signal Processing Laboratory

MAHARAJA INSTITUTE OF TECHNOLOGY MYSORE

Belwadi, SR Patna, Mandya Dist. 571438

Vision of the Institute

'To Gift the Nation with Eminent Engineers and Managers, Capable of Contributing Towards Technological Advancement and Betterment of the Society.'

Mission of the Institute

1. To advance the well-being of society through excellence in teaching, research and service that exploits the rapidly changing technical diversity via a collaborative environment that stimulates faculty, staff and students to reach their highest potential through continuous learning.
2. Instill the foundation of professionalism and provide the tools required to advance the technological world.
3. Inculcate knowledge and ethics, and nurture/foster innovation and team man ship among the graduates/alumnae.
4. Endow eminent education to serve the society with graduates/alumnae possessing ace skills and potential.
5. Sustain highest reception of the institute's alumnae among the stakeholders.

Vision of the Department

'To be recognized by the society at large as offering Value based Quality Education to groom the next generation entrepreneurs, leaders and researchers in the field of Electronics and Communication to meet the challenges at global level.'

Mission of the Department

1. To groom the students with strong foundations of Electronics and Communication Engineering and to facilitate them to pursue higher education and research.
2. To educate and prepare the students to be competent to face the challenges of the industry/society and /or to become successful entrepreneurs.
3. Provide ethical and value-based education by promoting activities addressing the societal needs.
4. Enable students to develop skills to solve complex technological problems of current times and also provide a framework for promoting collaborative and multidisciplinary activities.

Program Educational Objectives

1. To prepare the students to be able to have a successful career in dynamic industry that is global, multi-disciplinary, and evolving;
2. Develop their engineering skills in problem solving, design and innovation as they work individually and/or in multi-disciplinary teams with sense of professional ethics and social responsibility.
3. Communicate effectively and manage resources skillfully as members and leaders of the profession.

Program Specific Outcomes:

1. An ability to apply the basic concepts of engineering science into various areas of Electronics & Communication Engineering.
2. An ability to solve complex Electronics and Communication Engineering problems, using state of the art hardware and software tools, along with analytical skills to arrive at cost effective and efficient solutions.
3. Wisdom of social and environmental awareness along with ethical responsibility to have a successful career and to sustain passion and zeal for real-world applications using optimal resources.

Do's:

1. Students must bring observation/Manual book along with pen, pencil and eraser etc.
2. Students must handle the hardware kit and other components carefully as they are expensive
3. Before entering to lab, must prepare for viva for which they are going to conduct experiment.
4. Before switching on the hardware kit, the connections must be shown to one of the lab in-charge faculty members or instructors.
5. After the completion of the experiment should return the components to the lab instructors.
6. Strictly follow the procedures for conduction of experiments.
7. Any breakdown/failure of equipment must be reported to the technical staff immediately.
8. Uniform and ID card are must.
9. Maintain silence inside the laboratory
10. Keep your belongings in designated area.
11. Wear shock–proof footwear while entering into the circuit lab
12. Chairs and stools should be kept under the workbenches when not in use.
13. Sit upright on chairs or stools, keeping feet on the floor.
14. Every student should know the location and operating procedures of all Safety equipment including First Aid Kit and Fire extinguisher.

DONT'S:

1. Don't eat food, drink beverages or chew gum in the laboratory.
2. Don't touch any live terminals.
3. Don't spread unwanted connecting wires on the table.
4. Don't play with the instruments laid on work bench.
5. Don't transfer equipment to other workbench or other labs without permission.
6. Don't change connection when supply is on.
7. Don't leave the experiment table unattended when the experimental setup supply is on.

Syllabus

Following Experiments to be done using MATLAB / SCILAB / OCTAVE or equivalent:

1. Verification of sampling theorem.
2. Linear and circular convolution of two given sequences, Commutative, distributive and associative property of convolution.
3. Auto and cross correlation of two sequences and verification of their properties
4. Solving a given difference equation.
5. Computation of N point DFT of a given sequence and to plot magnitude and phase spectrum (using DFT equation and verify it by built-in routine).
6. (i) Verification of DFT properties (like Linearity and Parseval's theorem, etc.) (ii) DFT computation of square pulse and Sinc function etc.
7. Design and implementation of FIR filter to meet given specifications (using different window techniques).
8. Design and implementation of IIR filter to meet given specifications.

Following Experiments to be done using DSP kit

9. Linear convolution of two sequences
10. Circular convolution of two sequences
11. N-point DFT of a given sequence
12. Impulse response of first order and second order system
13. Implementation of FIR filter

Course Outcomes

Course Name: Digital Signal Processing Laboratory **Course Code: 15ECL57**

CO's	DESCRIPTION OF THE OUTCOMES
15ECL57.1	**Understand** the basic concepts of Digital Signal processing techniques with their Properties both in time and frequency domain.
15ECL57.2	**Design and develop** the software modules for digital filters and digital signal processing techniques/operations using Modern Software Tools.
15ECL57.3	**Implement** Digital signal processing techniques/operations and Digital filters using DSP Processor.
15ECL57.4	**Documentation** of the complete experimental process and results.

Scheme of Evaluation

1. Lab manuals & record will be evaluated for 05 marks (CO4).
2. Internal Assessment will be conducted for 15marks; Write-up & Conduction (CO2-05M, CO3-05M), Viva-Voce (CO1-05M).
3. Average of (Lab Manual Marks & Lab Record Marks (05)) and Internal Assessment conducted (15) will be considered for final IA marks.

Conduct of Practical Examination

1. All laboratory experiments are to be included for practical examination.
2. Students are allowed to pick one experiment from the lot.
3. Strictly follow the instructions as printed on the cover page of answer script for breakup of marks.
4. Change of experiment is allowed only once and Marks allotted to the procedure part to be made zero.

Contents

EXPERIMENT - 1

AIM: VERIFICATION OF SAMPLING THEOREM

Sampling: Is the process of converting a continuous time signal into a discrete time signal. It is the first step in conversion from analog signal to digital signal.

Sampling theorem: Sampling theorem states that "Exact reconstruction of a continuous time base-band signal from its samples is possible, if the signal is band-limited and the sampling frequency is greater than twice the signal bandwidth" i.e. $f_s > 2W$, where W is the signal bandwidth.

Nyquist Rate Sampling: The Nyquist rate is the minimum sampling rate required to avoid aliasing, equal to the highest modulating frequency contained within the signal. In other words, Nyquist rate is equal to two sided bandwidth of the signal (Upper and lower sidebands). To avoid aliasing, the sampling rate must exceed the Nyquist rate. i.e. $f_s > f_N$.

Program:

```
clear all;close all;clc;

% signal parameters

f1=1000;                          %Signal 1 frequency=1kHz
f2=1900;                          %Signal 2 frequency=1.9kHz
fmax=max(f1,f2);      %Maximum frequency component of the signal
T=1/min(f1,f2);       % Signal period should cover entire length
t=0:0.01*T:2*T;                   %Time index

%Generate the original signal and plot it;

x=cos(2*pi*t*f1)+cos(2*pi*t*f2);      %Composite signal
subplot(2,2,1);
plot(t,x);grid on;
title('continuous signal');
xlabel('t');
ylabel('x(t)');

%Over sampling condition;

fs1=10*fmax;                          %Over sampling(fs>2f)
n1=0:1/fs1:2*T;                       %Time scale
x1=cos(2*pi*f1*n1)+cos(2*pi*f2*n1);   %Generated sampled signal
subplot(2,2,2);
stem(n1,x1);
hold on;
plot(n1,x1,'r');grid on;
hold off;
title('over sampling condition:Fs=10F');
xlabel('n');
ylabel('x[n]');
```

```
%Exact sampling condition;

fs2=2*fmax;%Exact sampling(fs=2f)
n2=0:1/fs2:2*T;%Time scale
x2=cos(2*pi*f1*n2)+cos(2*pi*f2*n2);    %Generated sampled signal
subplot(2,2,3);
stem(n2,x2);
hold on;
plot(n2,x2,'r');grid on;
hold off;
title('exact sampling condition:Fs=2f');
xlabel('n');
ylabel('x[n]');

%Under sampling condition;

fs3=1.2*fmax;%Under sampling(fs=2f)
n2=0:1/fs3:2*T;%Time scale
x3=cos(2*pi*f1*n3)+cos(2*pi*f2*n3);    %Generated sampled signal
subplot(2,2,4);
stem(n3,x3);
hold on;
plot(n3,x3,'r');grid on;
hold off;
title('Under sampling condition:Fs=1.2f');
xlabel('n');
ylabel('x[n]');
```

PROBABLE VIVA QUESTIONS:

1) What is sampling theorem?
2) What is Nyquist criterion?
3) How the signals are get sampled?
4) What is aliasing?
5) How it is removed?
6) What is sampling frequency?
7) How the signals can be reconstructed?
8) What is sampling frequency?
9) What is modulating frequency?
10) What is the function of sample and hold circuit in DSP system?

OUTPUT

EXPERIMENT - 2

AIM: TO IMPLEMENT LINEAR AND CIRCULAR CONVOLUTION OF TWO GIVEN SEQUENCES

Theory: Convolution is an integral concatenation of two signals. It has many applications in numerous areas of signal processing. The most popular application is the determination of the output signal of a linear time-invariant system by convolving the input signal with the impulse response of the system. Note that convolving two signals is equivalent to multiplying the Fourier Transform of the two signals.

a. Linear Convolution:

Formula:

The linear convolution of two continuous time signals $x(t)$ and $h(t)$ is defined by

$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau)\, h(t-\tau)\, d\tau$$

For discrete time signals $x(n)$ and $h(n)$, is defined by

$$y(n) = x(n) * h(n) = \sum_{k=-\infty}^{\infty} x(k)\, h(n-k)$$

Where $x(n)$ is the input signal and $h(n)$ is the impulse response of the system.

In linear convolution length of output sequence is,

$$\text{length}(y(n)) = \text{length}(x(n)) + \text{length}(h(n)) - 1$$

Example:

$x(n) = \{1, 2, 3, 1\}$

$h(n) = \{1, 1, 1\}$

$$\text{length}(y(n)) = \text{length}(x(n)) + \text{length}(y(n)) - 1$$

$$= 4 + 3 - 1 = 6$$

When,

$$n=0;\ h(-k);\quad y(0) = \sum_{k=0}^{3} x(k)\, h(-k) = 1$$

$$n=1;\ h(1-k);\quad y(1) = \sum_{k=0}^{3} x(k)\, h(1-k) = 1+2=3$$

$$n=2;\ h(2-k);\quad y(2)=\sum_{k=0}^{3} x(k)\,h(2-k)=1+2+3=6$$

$$n=3;\ h(3-k);\quad y(3)=\sum_{k=0}^{3} x(k)\,h(3-k)=2+3+1=6$$

$$n=4;\ h(4-k);\quad y(4)=\sum_{k=0}^{3} x(k)\,h(4-k)=3+1=4$$

$$n=5;\ h(5-k);\quad y(5)=\sum_{k=0}^{3} x(k)\,h(5-k)=1$$

$$y(n)=\{1, 3, 6, 6, 4, 1\}$$

Program:

```
clc;                                    % Clear screen
x1 = input('Enter the 1st seq:');       % Get the first sequence
x2 = input('Enter the 2nd seq:');       % Get the second sequence
y = conv(x1, x2);                       % Convolve two signals
disp('The linear convolution of two sequences:');% Displays the result
disp(y);                    % Displays the result on command window
n = 0:length(y)-1;          % Defines the length for x-axis

stem(n, y);                         % Plots the output
xlabel('Time');                     % Name the x-axis as Time
ylabel('Magnitude');                % Name the y-axis as Magnitude
title('Linear convolution');        % Title as "Linear convolution"
```

Using Equation:

```
clear all;close all;clc;
x=input('Enter the input signal x[n]:');
h=input('Enter the input signal h[n]:');
n1=length(x);
n2=length(h);
N=n1+n2-1;                          %length of convoluted sequence
T=1:N;                              %create time index
```

%zero padding to make sequence of length N

```
x=[x zeros(1,(N-n1))];
h=[h zeros(1,(N-n2))];
```

```
%Initializing output sequence of zeros

y=zeros(1,N);
for n=1:N
    for k=1:n
    y(n)=y(n)+x(k)*h(n-k+1);
    end
end

%Plot input and output sequence

subplot(3,1,1);
stem(T,x);
title('input signal x[n]');
xlabel('n');ylabel(x[n]);
subplot(3,1,2);
stem(t,h);
title('input signal h[n]');
xlabel('n');ylabel(h[n]);
subplot(3,1,3);
stem(t,y);
title('convolutes signal y(n)=x(n)*h(n)');
xlabel('n');ylabel(y[n]);

%Display the convolved sequence in command window

disp('convolved sequence y[n]:');
disp(y);
```

PROBABLE VIVA QUESTIONS:

01) What is Convolution?
02) What is Linear Convolution?
03) What are the applications of Linear Convolution?
04) How Linear Convolution is different from Circular Convolution?
05) What are the advantages of Linear Convolution compared to Circular convolution?
06) What is Time domain sequence?
07) What is Frequency domain sequence?
08) Which is more convenient?
09) Why the conversion of signal from Time domain to Frequency domain is necessary?
10) What are the applications of Linear Convolution?

OUTPUT:

b. Circular convolution:

Let $x_1(n)$ and $x_2(n)$ are finite duration sequences both of length N with DFT's $X_1(k)$ and $X_2(k)$. Convolution of two given sequences $x_1(n)$ and $x_2(n)$ is given by the equation,

$$x_3(n) = \text{IDFT}[X_3(k)]$$

$$X_3(k) = X_1(k)\, X_2(k)$$

$$x_3(n) = \sum_{m=0}^{N-1} x_1(m)\, x_2((n-m))_N$$

Example:

Let's take $x_1(n) = \{1, 1, 2, 1\}$ and

$$x_2(n) = \{1, 2, 3, 4\}$$

Arrange $x_1(n)$ and $x_2(n)$ in matrix form (circular shift) as shown below.

$$\begin{pmatrix} 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 \\ 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} = \begin{pmatrix} 13 \\ 14 \\ 11 \\ 12 \end{pmatrix}$$

The convoluted signal is,

$$x_3(n) = \{13, 14, 11, 12\}$$

Program:

```
clc;                                % Clear screen
x1 = input('Enter the first sequence:');% Get the first sequence
x2 = input('Enter the second sequence:');% Get the second sequence
N1 = length(x1);       % Returns the length of first sequence
N2 = length(x2);       % Returns the length of second sequence
N = max(N1,N2);        % Get the Maximum length out of two signals
N3 = N1-N2;            % Get equal length sequence
if(N3>0)
x1 = [x1,zeros(1, N3)];% Pad zeros to the first sequence if N2>N1
else
x2 = [x2,zeros(1, -N3)];% Pad zeros to the second sequence is N1>N2
end
```

```
n=1:N;
disp(n);
y=0;
for m=1:N
disp(m);
y=y+x(m)*h(mod(n-m,N)+1);
end
disp('The circular convolution of two given sequences is:');
disp(y);              % Displays the result on command window
n=0:N-1;              % Get the length for x-axis
stem(n, y);           % Plot the result
xlabel('Time');       % Name the x-axis as "Time"
ylabel('Magnitude');  % Name the y-axis as "Magnitude"
title('Circular convolution');   % Title as Circular convolution
```

PROBABLE VIVA QUESTIONS:

1) What is Convolution?
2) What is Circular Convolution?
3) What are the applications of Circular Convolution?
4) How Circular Convolution is different from Linear Convolution?
5) What are the advantages of Circular Convolution compared to Linear convolution?
6) What is Time domain sequence?
7) What is Frequency domain sequence?
8) Which is more convenient?
9) Why the conversion of signal from Time domain to Frequency domain is necessary?
10) What are the applications of circular convolution?

OUTPUT:

EXPERIMENT - 3

AIM: AUTOCORRELATION AND CROSS CORRELATION OF A GIVEN SEQUENCES AND VERIFICATION OF THEIR PROPERTIES

Correlation: Correlation determines the degree of similarity between two signals. If the signals are identical, then the correlation coefficient is 1; if they are totally different, the correlation coefficient is 0, and if they are identical except that the phase is shifted by exactly 180^0(i.e. mirrored), then the correlation coefficient is -1.

Autocorrelation: The Autocorrelation of a sequence is correlation of a sequence with itself. The autocorrelation of a sequence $x(n)$ is defined by,

$$R_{xx}(k) = \sum_{n=-\infty}^{\infty} x(n)\, x(n-k) \qquad k = 0, \pm1, \pm2, \pm3 \ldots$$ Where k is the shift parameter.

Example:

Let's take a sequence $x(n) = \{1, 2, 3, 4\}$

$k=-3;\ x(n+3); \qquad R_{xx}(-3) = \sum x(n)\, x(n) = 4$

$k=-2;\ x(n+2); \qquad R_{xx}(-2) = \sum x(n)\, x(n+2) = 8+3=11$

$k=-1;\ x(n+1); \qquad R_{xx}(-1) = \sum x(n)\, x(n+1) = 12+6+2=20$

$k=0;\ x(n); \qquad R_{xx}(0) = \sum x(n)\, x(n+0) = 16+9+4+1=30$

$k=1;\ x(n-1); \qquad R_{xx}(1) = \sum x(n)\, x(n-1) = 12+6+2=20$

$k=2;\ x(n-2); \qquad R_{xx}(2) = \sum x(n)\, x(n-2) = 8+3=11$

$k=3;\ x(n-3); \qquad R_{xx}(3) = \sum x(n)\, x(n-3) = 4$

$$R_{xx}(k) = \{4,11,20,\underset{\uparrow}{30},20,11,4\}$$

Program:

```
clc;                                          % Clear screen
x = input('Enter the input sequence:');       % Get the input sequence
Rxx = xcorr(x);                    % Returns auto-correlated sequence
disp('Auto correlated sequence, Rxx:');       % Display the result
disp(Rxx);                      % Display the result on command window
t = -(length(x)-1):1:(length(x)-1);    % Length to plot the graph
```

```
stem(t,Rxx);                % plots the result on figure window
xlabel('Time');             % xlabel
ylabel('Magnitude');        % ylabel
title('Auto-correlation of the given sequence:'); % Title
```

%To verify Rxx(0)=energy

```
energy=sum(x.^2);
disp('Energy of the signal is:');
disp(energy);

center=ceil(length(Rxx)/2);  % to find the center index
Rxx_0=Rxx(center);
if Rxx_0==energy
     disp('Rxx(0) gives energy');
else
     disp('Rxx(0) doesn't give energy');
end
```

% to verify symmetric property

```
Rxx_right=Rxx(center:1:length(Rxx));
Rxx_left=Rxx(center:-1:1);
if Rxx_right==Rxx_left
     disp('Rxx is symmetric');
else
     disp('Rxx is not symmetric');
```

Properties of Autocorrelation:

1. Autocorrelation function is **symmetric.** i.e. $R_{xx}(m) = R_{xx}(-m)$
2. **Mean square value:** autocorrelation function at $k=0$, is equal to mean square value of the process. $R_{xx}(0) = E\{|x(n)|^2\} \geq 0$

PROBABLE VIVA QUESTIONS:

1. What is Correlation?
2. What are the different types of Correlation?
3. What is auto correlation?
4. What is cross correlation?
5. What are the advantages of correlation?
6. What are applications of correlation?
7. What are the differences between Auto and Cross correlation?
8. What is cross correlation?
9. What is circular cross correlation?
10. How correlation is different from convolution?

OUTPUT:

Cross-correlation: When two independent signals are compared, the procedure is known as cross-correlation. It is given by,

$$R_{xy}(k) = \sum_{n=-\infty}^{\infty} x(n)\, y(n-k) \qquad ;k = 0, \pm1, \pm2, \pm3 \ldots$$ ...where k is the shift parameter

Or equivalently

$$R_{yx}(k) = \sum_{n=-\infty}^{\infty} y(n+k)\, x(n)$$

Comparing above two equations, we find that,

$$R_{xy}(k) = R_{yx}(-k)$$

Where $R_{yx}(-k)$ is the folded version of $R_{xy}(k)$ about $k = 0$.

So, we can write Cross correlation of the two sequences is given by,

$$R_{xy}(k) = \sum_{n=-\infty}^{\infty} x(n)\, y[-(k-n)]$$

$$R_{xy}(k) = x(k) * y(-k) \qquad \ldots (1)$$

Equation (1) shows that cross correlation is the essentially the convolution of two sequences in which one of the sequences has been reversed.

Example: Let's take $x(n) = \{1, 1, 1, 1\}$ and $y(n) = \{1, 2, 3, 4\}$

k=-3; y(n+3); $R_{xy}(-3) = \sum x(n)\, y(n+3) = 4$

k=-2; y(n+2); $R_{xy}(-2) = \sum x(n)\, y(n+2) = 4+3=7$

k=-1; y(n+1); $R_{xy}(-1) = \sum x(n)\, y(n+1) = 4+3+2=9$

k=0; y(n); $R_{xy}(0) = \sum x(n)\, y(n) = 4+3+2+1=10$

k=1; y(n-1); $R_{xy}(1) = \sum x(n)\, y(n-1) = 3+2+1=6$

k=2; y(n-2); $R_{xy}(2) = \sum x(n)\, y(n-2) = 2+1 = 3$

k=3; y(n-3); $R_{xy}(3) = \sum x(n)\, y(n-3) = 1$

$$R_{xy}(k) = \{4, 7, 9, \underset{\uparrow}{10}, 6, 3, 1\}$$

Program:

```
clc;                    % Clear screen
x = input('Enter the first sequence:');% Get the first sequence
Rxx=xcorr(x);
disp('Auto correlated sequence,Rxx:');
disp(Rxx);
y = input('Enter the second sequence:');% Get the second sequence
Ryy=xcorr(y);
disp('Auto correlated sequence,Ryy:');
disp(Ryy);
Rxy = xcorr(x,y);                % Returns cross correlated sequence
disp('Cross correlated sequence, Rxy:');% Display the result
disp(Rxy);                    % Display the result on command window
Ryx = xcorr(y,x);             % Returns cross correlated sequence
disp('Cross correlated sequence, Ryx:');% Display the result
disp(Ryx);                    % Display the result on command window
t = -(length(x)-1):1:(length(x)-1);     % Length to plot the graph
stem(t, Rxy);                 % plots the result on figure window
xlabel('Time');               % xlabel
ylabel('Magnitude');          % ylabel
title('Cross correlation of the given two sequences:');
```

Properties of cross correlation:

1. $R_{xy}(k)$ is always a real valued function which may be a positive or negative.
2. $R_{xy}(-k) = R_{yx}(k)$
3. $|R_{xy}(k)|^2 \leq R_{xx}(0)\, R_{yy}(0)$
4. When $R_{xy}(k) = 0, x(n)$ and $y(n)$ are said to be uncorrelated or they said to be statistically independent.
5. $R_{xy}(k)$ may not be necessarily have a maximum at $k=0$ nor $R_{xy}(k)$ an even function.

   Eg: Let x={1,1,2,2} and y={1,3,2,1}

Program:

```
E1=sum(x.^2);               % energy of signal x[n]
E2=sum(y.^2);               % energy of signal y[n]
mid=ceil(length (rxy)/2);   %find index of centre of sequence
E0=abs(max(rxy));           %Detect maximum amplitude of sequence
fprintf('Energy of input signal x:%d\n',E1);
fprintf('Energy of input signal h:%d\n',E2);
fprintf('max amplitude of cross correlation sequence:%d\n',E0);

%max amplitude of sequence should be less than sqrt(E1*E2)
if int8(E0)<=int8(sqrt(E1*E2))
```

```
disp('cross correlation energy property is verified');
else
disp('cross correlation energy property is not verified');
end
%verify signal property;rxy(1)=ryx(-1)
if rxy==fliplr(rxy)
disp('since rxy(1)=ryx(-1),cross correlation property is verified');
else
disp('cross correlation property is not verified');
end
```

OUTPUT:

EXPERIMENT - 4

AIM: TO SOLVE A GIVEN DIFFERENCE EQUATION

A General N^{th} order Difference equations looks like,

$$y[n] + a_1y[n-1] + a_2y[n-2] + \ldots + a_Ny[n-N) = b_0x[n] + b_1x[n-1] + \ldots. + b_Mx[n-M]$$

Here $y[n]$ is "Most advanced" output sample and $y[n-m]$ is "Least advanced" ouput sample.

The difference between these two index values is the order of the difference equation. Here we have: $n-(n-N) = N$

We can rewrite the above equation as,

$$y[n] + \sum a_i\, y[n-i] = \sum b_i\, x[n-i]$$

$y[n]$ becomes, $y[n] = -\sum a_i\, y[n-i] + \sum b_i\, x[n-i]$

Example:

$$y[n+2] - 1.5y[n+1] + y[n] = 2x[n]$$

In general we start with the "Most advanced" output sample. Here it is $y[n+2]$. So, here we need to subtract 2 from each sample argument. We get

$$y[n] - 1.5y[n-1] + y[n-2] = 2x[n-2]$$

$$\Rightarrow\ y[n] = 1.5y[n-1] - y[n-2] + 2x[n-2]$$

Let us assume our input $x[n] = u[n] = \begin{cases} 0 & x<0 \\ 1 & x\geq 0 \end{cases}$

In our example we have taken $x[n] = u[n] = \begin{cases} 0 & x<0 \\ 1 & 0\leq x<10 \end{cases}$

We need N past values to solve N^{th} order difference equation.

$$y[-2] = 1$$

$$y[-1] = 2$$

Compute $y[n]$ for various values of n

$y[0] = 1.5y[-1] - y[-2] + 2x[-2]$

$= 1.5*2 - 1 + 2*0$

$y[0] = 2$

$y[1] = 1.5y[0] - y[-1] + 2x[-1]$

$= 1.5*2 - 2 + 2*0$

$y[1] = 1$

$y[2] = 1.5y[1] - y[0] + 2x[0]$

$= 1.5*1 - 2 + 2*1$

$y[2] = 1.5$ and so on...

$$y[n] = \{1, 2, 2, 1, 1.5, 3.25, 5.375, 6.8125.....\}$$
$$\uparrow$$

Program:

```
clc;                          % Clear screen
x = [0 0 ones(1, 10)];        % Input x[n] = u[n]
y_past = [1 2];               % Past values to solve difference equation

y(1) = y_past(1);             % y[1] <= y[-2]
y(2) = y_past(2);             % y[2] <= y[-1]

for k = 3:(length(x)) ;       % Compute y[n] for various values of n
    y(k) = 1.5*y(k-1) - y(k-2) + 2*x(k-2);
end

disp('Solution for the given difference equation:');
disp(y);                      % Display the result on command window
subplot(1,1,1);               % Divide the window to plot the result
stem (-2:(length(y)-3),y);    % Plot the result
xlabel('Input x[n]');         % Name x-axis as Input x[n]
ylabel('Output y[n]');        % Name y-axis as Output y[n]
title('Difference equation');    % Title as "Difference equation"
```

(OR)

```
clc;
a=[1,-1.5,1];                 %coefficients of y(n)
b=[2];                        %coefficients of x(n)
y=[2,1];                      %initial conditions of y(n)
x=[1,1];                      %initial conditions of x(n)
z=filtic(b,a,y,x);           %using initial condition
Y=filter(b,a,x,z);
disp(Y);
stem(Y);
```

PROBABLE VIVA QUESTIONS:

1) What is difference equation?
2) What is differential equation?
3) What are the differences between difference and differential equation?
4) What are applications of difference equation?
5) What is the function of 'stem' command?
6) What is the function of 'title' command?
7) What is the function of 'disp' command?
8) What are the advantages of difference equation?
9) How the limit is specified in the Mat lab?
10) What happens if the semicolon (;) symbol is included? What is the effect of the particular line?

OUTPUT:

EXPERIMENT - 5

AIM: TO COMPUTE N-POINT DFT OF A GIVEN SEQUENCE AND TO PLOT MAGNITUDE AND PHASE SPECTRUM.

Discrete Fourier Transform: The Discrete Fourier Transform is a powerful computation tool which allows us to evaluate the Fourier Transform $X(e^{j\omega})$ on a digital computer or specially designed digital hardware. Since $X(e^{j\omega})$ is continuous and periodic, the DFT is obtained by sampling one period of the Fourier Transform at a finite number of frequency points. Apart from determining the frequency content of a signal, DFT is used to perform linear filtering operations in the frequency domain.

The sequence of N complex numbers $x_0,\ldots, x_{N-1}$ is transformed into the sequence of N complex numbers $X_0, \ldots, X_{N-1}$ by the DFT according to the formula:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi nk/N} \qquad k = 0,1, \ldots N-1$$

Example:

Let us assume the input sequence $x[n] = [1\ 1\ 0\ 0]$

We have,

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi nk/N} \qquad k = 0,1, \ldots N-1$$

For $k = 0$,

$$X(0) = \sum_{n=0}^{3} x(n) = x(0) + x(1) + x(2) + x(3)$$

$$X(0) = 1+1+0+0 = 2$$

For $k = 1$,

$$X(1) = \sum_{n=0}^{3} x(n) e^{-j\pi n/2} = x(0) + x(1)\, e^{-j\pi/2} + x(2)\, e^{-j\pi} + x(3)\, e^{-j3\pi/2}$$

$$= 1 + \cos(\pi/2) - j\sin(\pi/2)$$

$$X(1) = 1 - j$$

For k = 2,

$$X(2) = \sum_{n=0}^{3} x(n)e^{-j\pi n} = x(0) + x(1)\, e^{-j\pi} + x(2)\, e^{-j2\pi} + x(3)\, e^{-j3\pi}$$

$$= 1 + \cos\pi - j\sin\pi$$

$$X(2) = 1-1 = 0$$

For k = 3,

$$X(3) = \sum_{n=0}^{3} x(n)e^{-j3n\pi/2} = x(0) + x(1)\, e^{-j3\pi/2} + x(2)\, e^{-j3\pi} + x(3)\, e^{-j9\pi/2}$$

$$= 1 + \cos(3\pi/2) - j\sin(3\pi/2)$$

$$X(3) = 1 + j$$

The DFT of the given sequence is,

$$X(k) = \{\, 2, 1-j, 0, 1+j \,\}$$

Program:

```
clc;                                    % Clear screen
x1 = input('Enter the sequence:');      % Get the input sequence
n = input('Enter the length:');         % Get the value of N
m = fft(x1,n);              % Computes the DFT using FFT algorithm
disp('N-point DFT of a given sequence:');% Display the results
disp(m);                    % Displays the result on command window
z=abs(fft(x1,n));
disp('magnitude of the DFT seq:');
disp(z);
N = 0:1:n-1;                            % Decides the length to plot
the results
subplot(2,2,1);                         %Divide the figure window
stem(N,z);                              % Plots the magnitude spectrum
xlabel('Length');                       % Name x-axis as "Length"
ylabel('Magnitude of X(k)');% Name y-axis as "Magnitude of X(k)"
title('Magnitude spectrum:');    % Title as "Magnitude spectrum"
an = angle(m);           % Get the angle of the output sequence X(k)
subplot(2,2,2);                  % Divide the figure window
```

```
stem(N, an);                    % Plots the phase spectrum
xlabel('Length');               % Name x-axis as "Length"
ylabel('Phase of X(k)');        % Name y-axis as "Phase of X(k)"
title('Phase spectrum:');       % Title as "Phase spectrum"
```

% DFT using defining equation

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi nk/N} \qquad k = 0,1, \ldots N-1$$

Program

```
Clc;
X=input("enter the sequence");
Disp(X)
N=length(x)
W=exp(-i*2*pi*n/N);
A=0:1:N-1
For k=0:N-1;
Z=W.^k;
t=X.*z;
X(k+1)=sum(t);
end
disp('N-pt DFT is : ');
disp(X)
 m=abs(X);
disp(m);
subplot(2,1,1);
stem(A,m);
xlabel('time');             % Name x-axis as "Length"
ylabel('Magnitude of X(k)');% Name y-axis as "Magnitude of X(k)"
title('Magnitude spectrum:');% Title as "Magnitude spectrum"
an = angle(m);
disp('phase spectrum');
disp(an);             % Get the angle of the output sequence X(k)
subplot(2,1,2);       % Divide the figure window
stem(A, an);          % Plots the phase spectrum
xlabel('time');       % Name x-axis as "Length"
ylabel('Phase of X(k)');        % Name y-axis as "Phase of X(k)"
title('Phase spectrum:');       % Title as "Phase spectrum"
```

OUTPUT:

EXPERIMENT - 6

AIM: TO VERIFY DFT PROPERTIES AND TO COMPUTE DFT OF SQUARE PULSE AND SINC FUNCTIONS

Program:

1. Linearity property

```
clc;
clear all;
x1=input('enter the sequence');
x2=input('enter the second sequence');
a=2;
b=3;
X1=fft(x1);
X2=fft(x2);
S1=a*x1+b*x2;
s_d=fft(S1);
disp('LHS=');
disp(s_d);
S2=a*X1+b*X2;
disp('RHS=');
disp(S2);
if(s_d == S2)
disp('linearity property is satisfied);
else
disp('linearity property is not satisfied);
end
```

2)Parseval's theorem[method 1]

```
clc;
clear all;
x=input('enter the sequence');
N=length(x);
S1=sum(x.^2);
X=fft(x);
x_sq=abs(x.2^);
S2= 1/N * sum(x_sq);
if(S1 = S2);
disp('parseval's theorem is verified');
else
disp('parseval's theorem is not verified');
end
```

3)Parseval's theorem[method 2]

```
clc;
clear all;
x=input('x(n)=');
```

```
y=input('y(n)=');
N=length(x);
X=fft(x);
Y=fft(y);
y_c=conj(y);
S1=sum(x.* y_c);
Y_C=conj(Y);
S2=(1/N) * sum(X.* Y_C);
disp('LHS');
disp(S1);
disp('RHS');
disp(S2);
if(S1 == S2)
disp('parseval's theorem is verified');
else
disp('parseval's theorem is not verified');
end
```

EXPERIMENT - 7

AIM: DESIGN AND IMPLEMENTATION OF FIR FILTER TO MEET GIVEN SPECIFICATIONS.

Finite Impulse Response (FIR) Filter: The FIR filters are of non-recursive type, whereby the present output sample is depending on the present input sample and previous input samples.

The transfer function of a FIR causal filter is given by,

$$H(z) = \sum_{n=0}^{N-1} h(n)z^{-n}$$ Where $h(n)$ is the impulse response of the filter.

The Fourier transform of $h(n)$ is

$$H(e^{jw}) = \sum_{n=0}^{N-1} h(n)e^{-jwn}$$

In the design of FIR filters most commonly used approach is using windows. The desired frequency response $H_d(e^{jw})$ of a filter is periodic in frequency and can be expanded in Fourier series. The resultant series is given by,

$$h_d(n) = (1/2\pi) \int_{-\pi}^{\pi} H(e^{jw})e^{jwn}\, dw$$

And known as Fourier coefficients having infinite length. One possible way of obtaining FIR filter is to truncate the infinite Fourier series at $n = \pm [(N-1)/2]$ Where N is the length of the desired sequence.

The Fourier coefficients of the filter are modified by multiplying the infinite impulse response with a finite weighing sequence $w(n)$ called a window.

Where $w(n) = w(-n) \neq 0$ for $|n| \leq [(N-1)/2]$

$= 0$ for $|n| > [(N-1)/2]$

After multiplying $w(n)$ with $h_d(n)$, we get a finite duration sequence $h(n)$ that satisfies the desired magnitude response,

$h(n) = h_d(n)\, w(n)$ for $|n| \leq [(N-1)/2]$

$= 0$ for $|n| > [(N-1)/2]$

The frequency response $H(e^{jw})$ of the filter can be obtained by convolution of $H_d(e^{jw})$ and $W(e^{jw})$ is given by,

$$H(e^{jw}) = (1/2\pi) \int_{-\pi}^{\pi} H_d(e^{j\theta})\, W(e^{j(w-\theta)})\, d\theta$$

$$H(e^{jw}) = H_d(e^{jw}) * W(e^{jw})$$

Example:

Here we design a LPF using hamming window. Hamming window function is given by,

$$w_H(n) = 0.54 + 0.46 \cos((2\pi n)/(N-1)) \quad \text{for } -(N-1)/2 \leq n \leq (N-1)/2$$

$$= 0 \quad \text{otherwise}$$

The frequency response of Hamming window is,

$$W_H(e^{jw}) = 0.54[(\sin(wN/2))/(\sin(w/2))]$$
$$+ 0.23[\sin(wN/2 - \pi N/N - 1)/\sin(w/2 - \pi/N - 1)]$$
$$+ 0.23[\sin(wN/2 + \pi N/N - 1)/\sin(w/2 + \pi/N - 1)]$$

Program:

```
clc;                          % Clear screen
wp = input('Pass band edge freq:');% Get Passband edge frequency
ws = input('Stop band edge freq:');% Get Stopband edge frequency
tw = ws-wp;                   % Subtract PB frequency from SB frequency

N = ceil(6.6*pi/tw)+1;        % Compute length
wn = (hamming(N)); % Returns N-point symmetric hamming window in
                              % column vector
B = fir1(N-1,wp,wn);  % Designs (N-1)th order lowpass FIR filter and
          % returns filter coefficients in length N in vector B
disp('Impulse response coeff='); %   Displays   Impulse   response
coefficients
disp(B);   % Displays the coefficients on command window
[H,w] = freqz(B,1,256);% Digital filter frequency response.
                              %This function returns the N-point complex
                              frequency response % vector H and the N-
                              point   frequency   vector   w   in   %
                              radians/sample of the filter.
Mag = 20*log10(abs(H));       % Get the magnitude
plot(w/pi*pi, Mag);           % Plot the Magnitude spectrum
xlabel('Frequency in radians in terms of pi');   % Name x-axis
ylabel('Gain in db');         % Name y-axis as "Gain in db"
```

PROBABLE VIVA QUESTIONS:

1) What is IIR Filter and What is FIR Filter ?
2) What are the differences between IIR and FIR Filter ?
3) What are applications of IIR and FIR Filter ?
4) What is quantization noise ?
5) What is normalized filter ?
6) What is Stopband ,Passband, transition band ?
7) What is ripple ?
8) What are the differences between Butterworth and Chebyshev Filter ?

OUTPUT:

EXPERIMENT - 8

AIM: DESIGN AND IMPLEMENTATION OF IIR FILTER TO MEET GIVEN SPECIFICATIONS

Infinite Impulse Response(IIR) filter: IIR filters are of recursive type, whereby the present output sample depends on the present input, past input samples and output samples.

The impulse response $h(n)$ for a realizable filter is,

$$h(n) = 0 \quad \text{for } n \leq 0$$

And for stability, it must satisfy the condition,

$$\sum_{n=0}^{\infty} |h(n)| < \infty$$

Example:

Let's design an analog Butterworth lowpass filter.

Steps to design an analog Butterworth lowpass filter.

1. From the given specifications find the order of the filter N.
2. Round off it to the next higher integer.
3. Find the transfer function $H(s)$ for $\Omega c = 1 \text{rad/sec}$ for the value of N.
4. calculate the value of cutoff frequency Ωc
5. find the transfer function $H_a(s)$ for the above value of Ωc by substituting

   $s \rightarrow (s/\Omega c)$ in $H(s)$.

Program:

```
clc;                          % Clear screen
Pa = input('Passband attenuation in DB:');
                              % Get the passband attenuation
Sa = input('Stopband attenuation in DB:');
                              % Get the stopband attenuation
Fpb = input('Passband edge frequency in Hz:');
                              % Get Passband edge frequency
Fsb = input('Stopband edge frequency in Hz:');
                              % Get Stopband edge frequency
fs = input('Sampling frequency:');    % Get Sampling frequency
wp = 2*Fpb/fs;        % Convert PB edge frequency in Hz to radians
ws = 2*Fsb/fs;        % Convert SB edge frequency in Hz to radians
[N,wc] = buttord(wp,ws,Pa,Sa);
                % Find cutoff frequency and order of the filter
```

```
disp('Order:');              % Display the Order
disp(N);                     % Display order N on command window
disp('Cutoff frequency:'); % Display Cutoff frequency
disp(wc);             % Display Cutoff frequency on command window
[b,a] = butter(N,wc,'low');% for LPF
disp('b='); disp(b);  % Display the numerator coefficients
disp('a='); disp(a);  % Display the denominator coefficients
w = 0:0.01:pi;        % Defines length for x-axis to plot the result
[h,om] = freqz(b,a,w,'whole');    % Frequency response of the filter
m = 20*log10(abs(h)); %  Absolute  value  of  the  frequency  response
vector
an = angle(h);   % Get the phase of the frequency response vector
subplot(2,1,1);  % Divide the figure window
plot(om/pi,m);   % Plot the response
xlabel('Normalised frequency:');
                % Name x-axis as "Normalized frequency"
ylabel('Gain in db:');           % Name y-axis as " Gain in db"
title('Frequency Response:');    % Title as Frequency Response
subplot(2,1,2);                  % Divide the figure window
plot(om/pi,an);                  % Plot the Phase spectrum
xlabel('Normalised frequency:');
                           % Name x-axis as "Normalized frequency"
ylabel('Phase in radians:');      % Name y-axis as Phase in radians
title('Phase spectrum');          % Title as Phase spectrum
```

% FOR OTHER FILTER TYPES, REPLACE

HPF - [b,a] = butter(N,wc,'high');

BPF - wn=[wp,ws]; [b,a] = butter(N,wn,'bandpass');

BSF - wn=[wp,ws]; [b,a] = butter(N,wn,'bandstop');

% FOR CHEBYSHEV FILTER

[N,wc] = cheb1ord(wp,ws,Pa,Sa);

LPF - [b,a] = cheby1(N,wc,'low');

HPF - [b,a] = cheby1(N,wc,'high');

BPF - wn=[wp,ws]; [b,a] = cheby1(N,Pa,wn,'bandpass');

BSF - wn=[wp,ws]; [b,a] = cheby1(N,Pa,wn,'bandstop');

PROBABLE VIVA QUESTIONS:

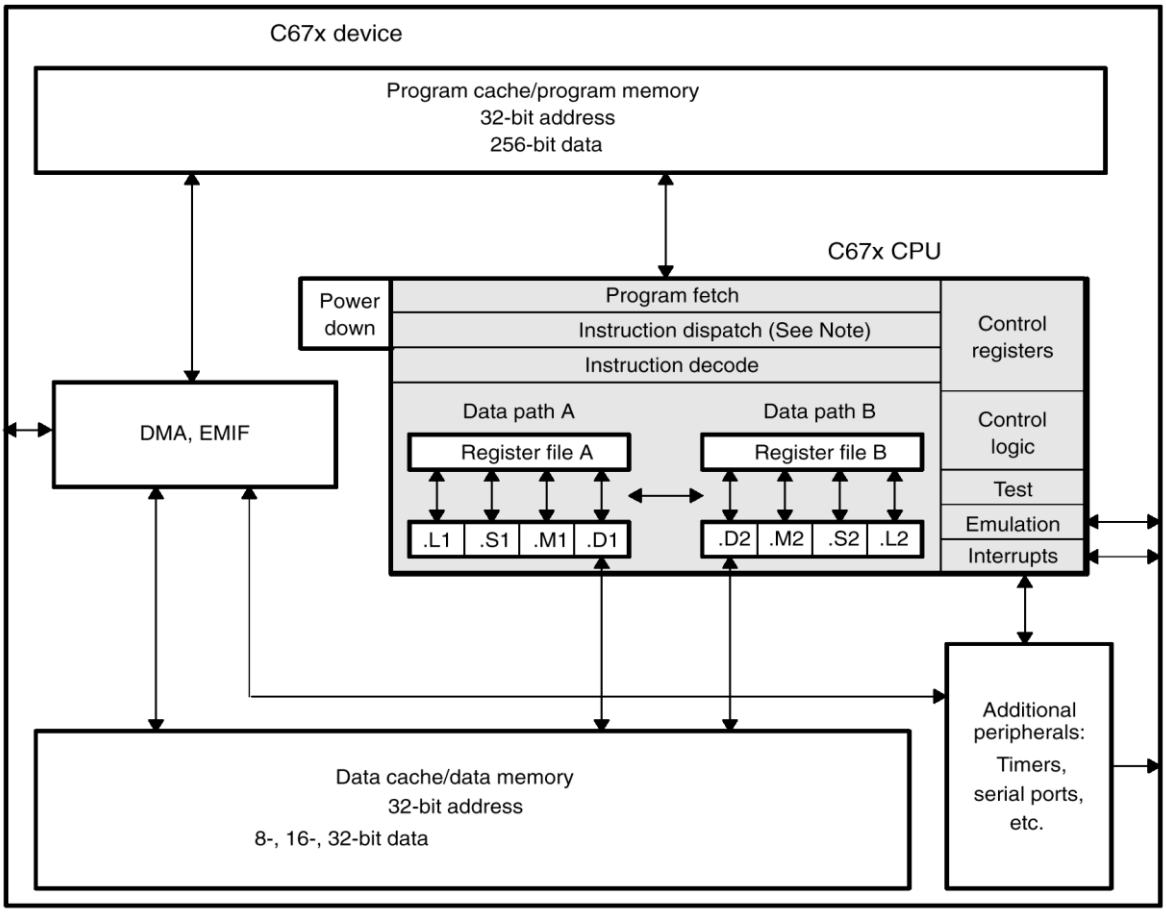
1) What is Filter? How the filters are mainly classified?
2) What are the applications of Filter?
3) What is the difference between Low Pass and High Pass Filter?
4) What is the difference between Band Pass and Band Reject Filter?
5) What is Cut-off frequency and gain?
6) What is order of the filter?

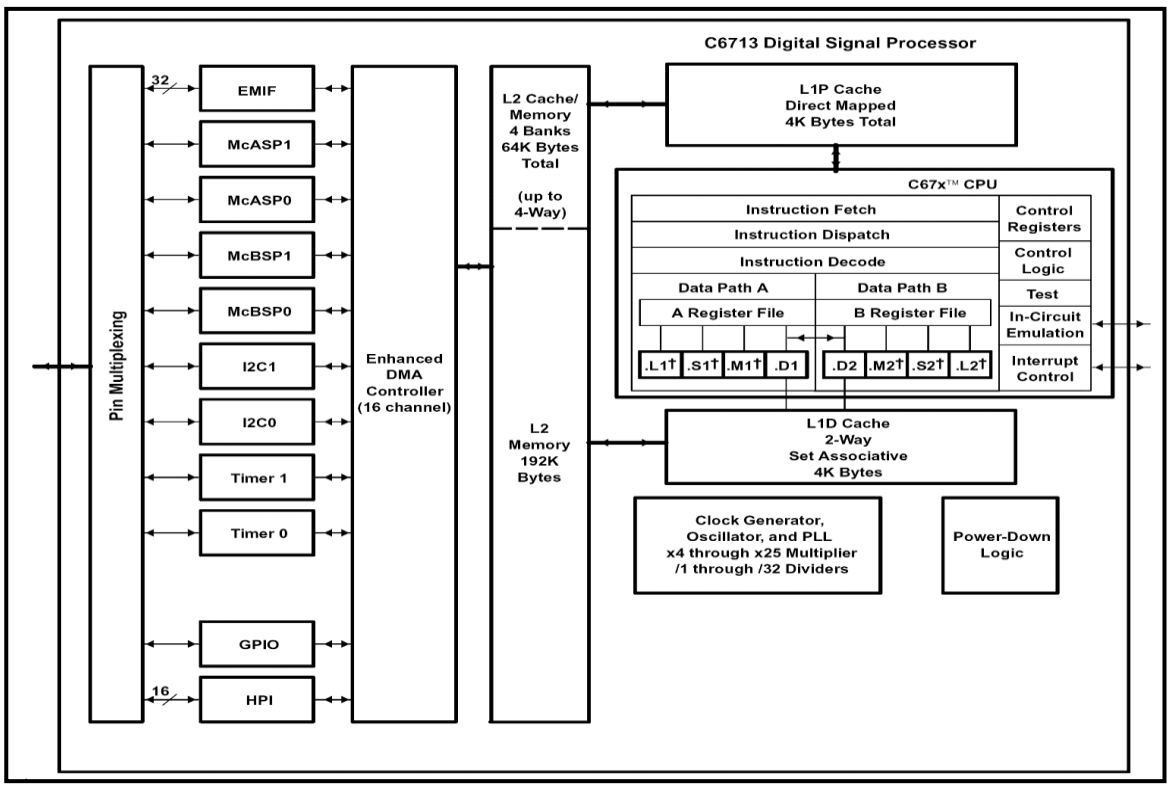
7) What is magnitude and frequency response?
8) What is Gibbs phenomenon?
9) What is Window technique and what are the different types of Window?
10) What is S-plane and what is Z-plane?
11) What is frequency transformation and why the transformation is necessary?
12) What is the advantage of mapping technique?
13) What is the function of 'buttord' command?
14) What do mean by higher and lower cut-off frequency?
15) What is pre-warping?

OUTPUT:

Part B

ARCHITECTURE OF TMS320C67XX





Highest-Performance Floating-Point Digital Signal Processor TMS320C6713

- Eight 32-Bit Instructions/Cycle
- 32/64-Bit Data Word
- 4.4-, 6.7-ns Instruction Cycle Time
- 1800 MIPS/1350 MFLOPS
- Rich Peripheral Set, Optimized for Audio
- Highly Optimized C/C++ Compiler

Functional Units and Operations Performed:

Functional Unit	Fixed-Point Operations	Floating-Point Operations
.L unit (.L1,.L2)	32/40-bit arithmetic and compare operations Leftmost 1 or 0 bit counting for 32 bits Normalization count for 32 and 40 bits 32-bit logical operations	Arithmetic operations Conversion operations: DP → SP, INT → DP, INT → SP
.S unit (.S1, .S2)	32-bit arithmetic operations 32/40-bit shifts and 32-bit bit-field operations 32-bit logical operations Branches Constant generation Register transfers to/from the control register file (.S2 only)	Compare reciprocal and reciprocal square-root operations Absolute value operations SP to DP conversion operations
.M unit (.M1, .M2)	16 × 16 bit multiply operations	32 × 32 bit multiply operations Floating-point multiply operations
.D unit (.D1, .D2)	32-bit add, subtract, linear and circular address calculation Loads and stores with a 5-bit constant offset Loads and stores with a 15-bit constant offset (.D2 only)	Load double word with a 5-bit constant offset

Procedure to Setup Emulator:

1.Open the Setup CCStudio v3.1 

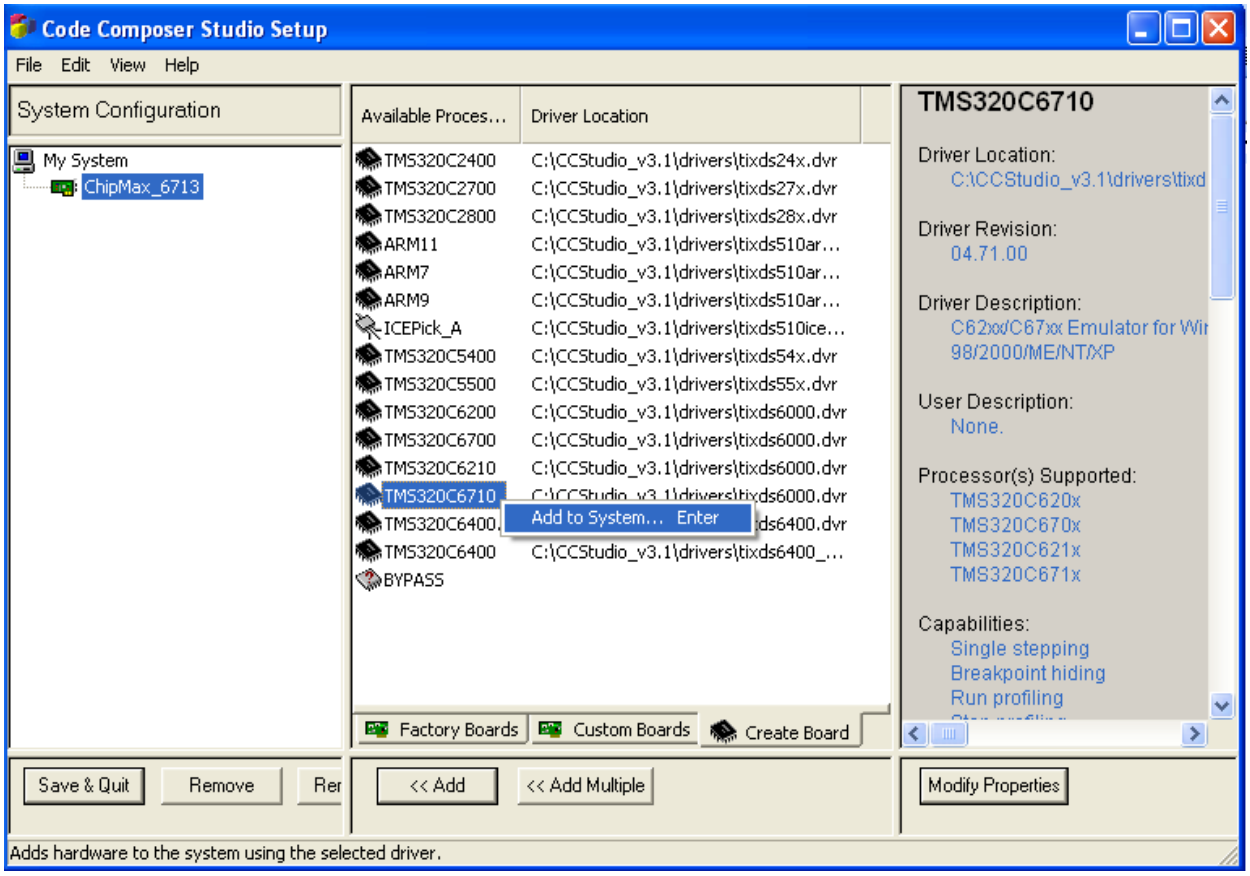
PROBABLE VIVA QUESTIONS:

1) What is Processor?
2) What is Fixed-point processor and what is floating point processor?
3) Which is more convenient and fast?
4) What is simulator and emulator?
5) What is synthesis?
6) What is Code Composer Studio?
7) What is DSK?
8) What is the operating voltage of DSP Processor?

CODE COMPOSER STUDIO

1. Start CCS set up by double clicking on the set up CCS desktop icon and follow the steps given below:

2. Right Click on TMS320C6710 and Select Add to System…Enter.



3. Click on Save and quit.

4. Click on Yes to start Code Composer Studio.

EXPERIMENT - 9

AIM: LINEAR CONVOLUTION OF THE TWO GIVEN SEQUENCES

Formula:

The linear convolution of two continuous time signals x(t) and h(t) is defined by

$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau)\, h(t-\tau)\, d\tau$$

For discrete time signals x(n) and h(n), is defined by

$$y(n) = x(n) * h(n) = \sum_{k=-\infty}^{\infty} x(k)\, h(n-k)$$

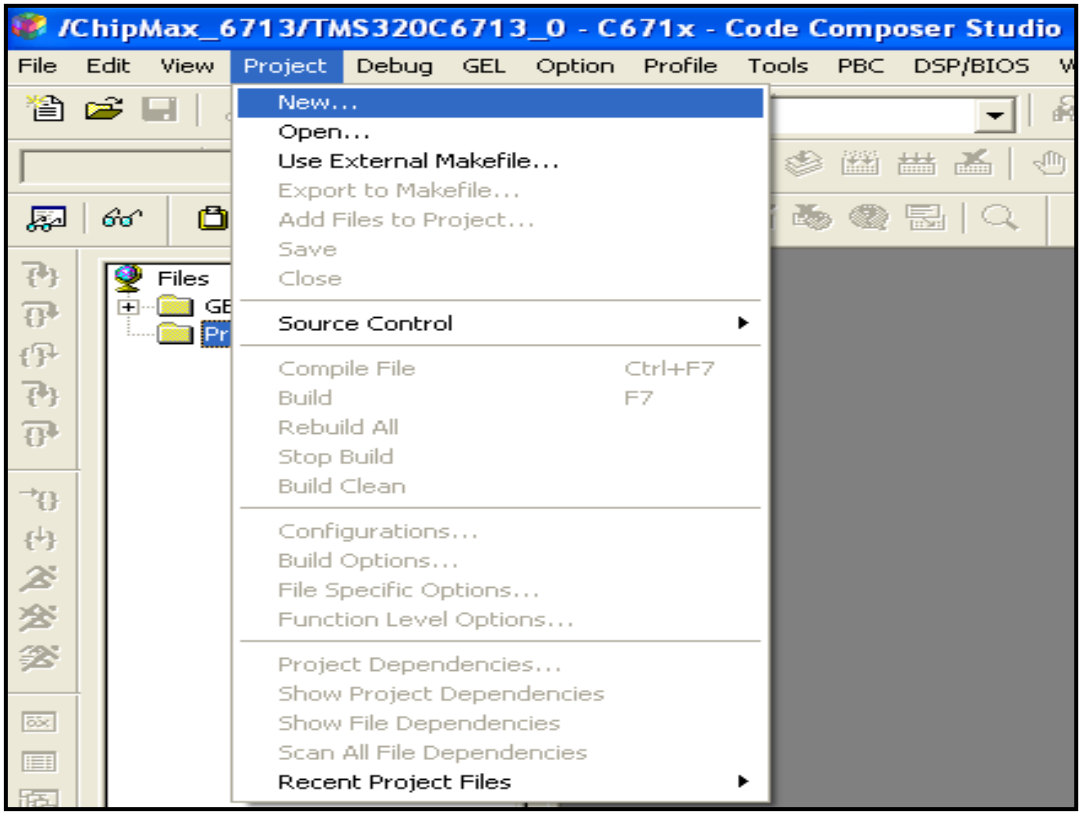
Where x(n) is the input signal and h(n) is the impulse response of the system.

In linear convolution length of output sequence is,

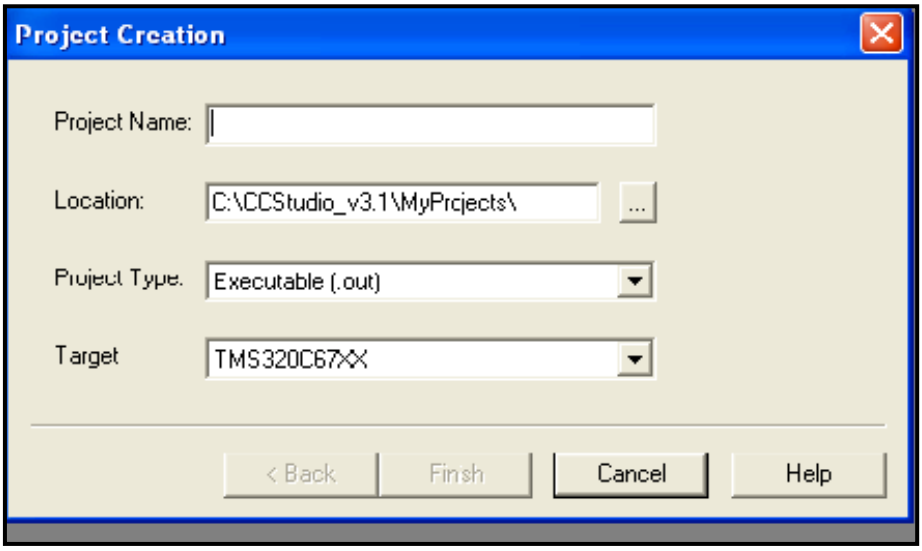
Length (y(n)) = length(x(n)) + length(h(n)) – 1

Procedure to create new Project:

1. To create project, Go to Project and Select New.

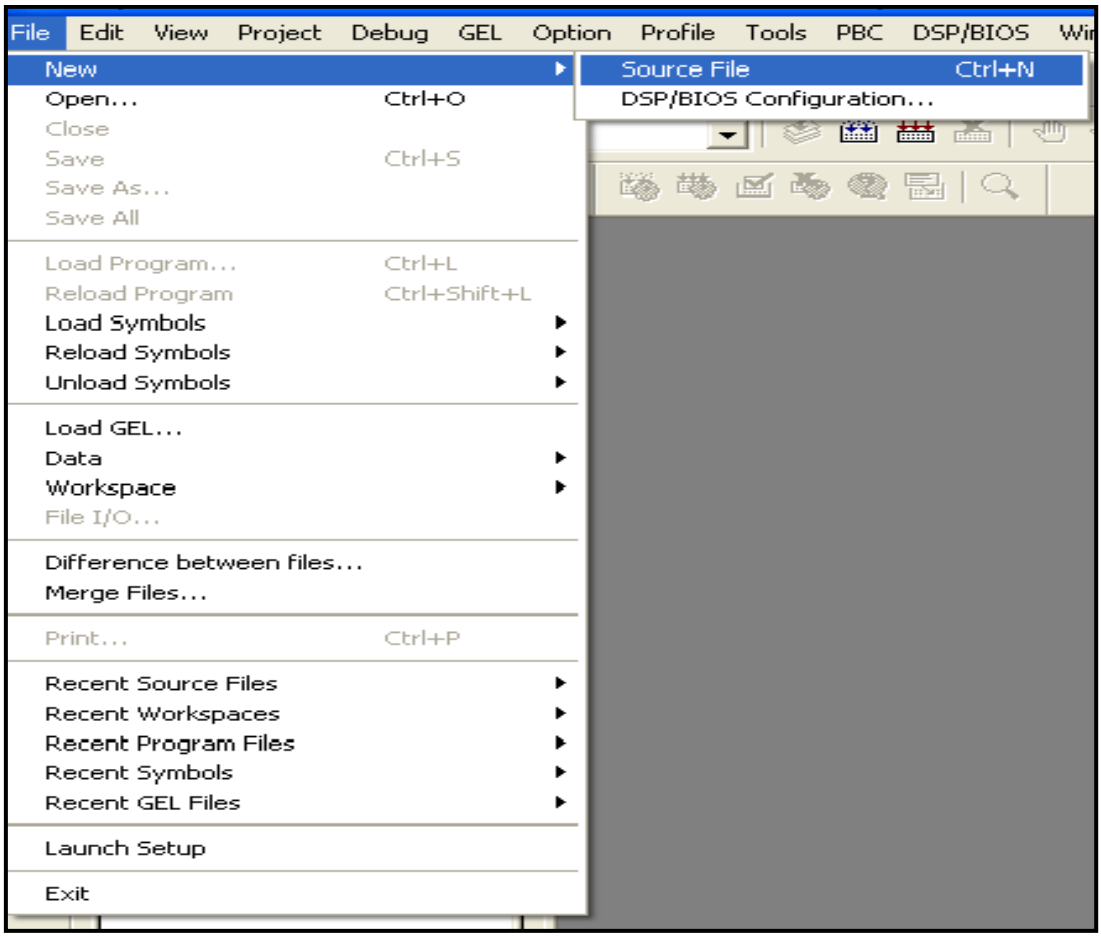


2. Give project name and click on finish.



(**Note:** Location must be c:\CCStudio_v3.1\MyProjects).

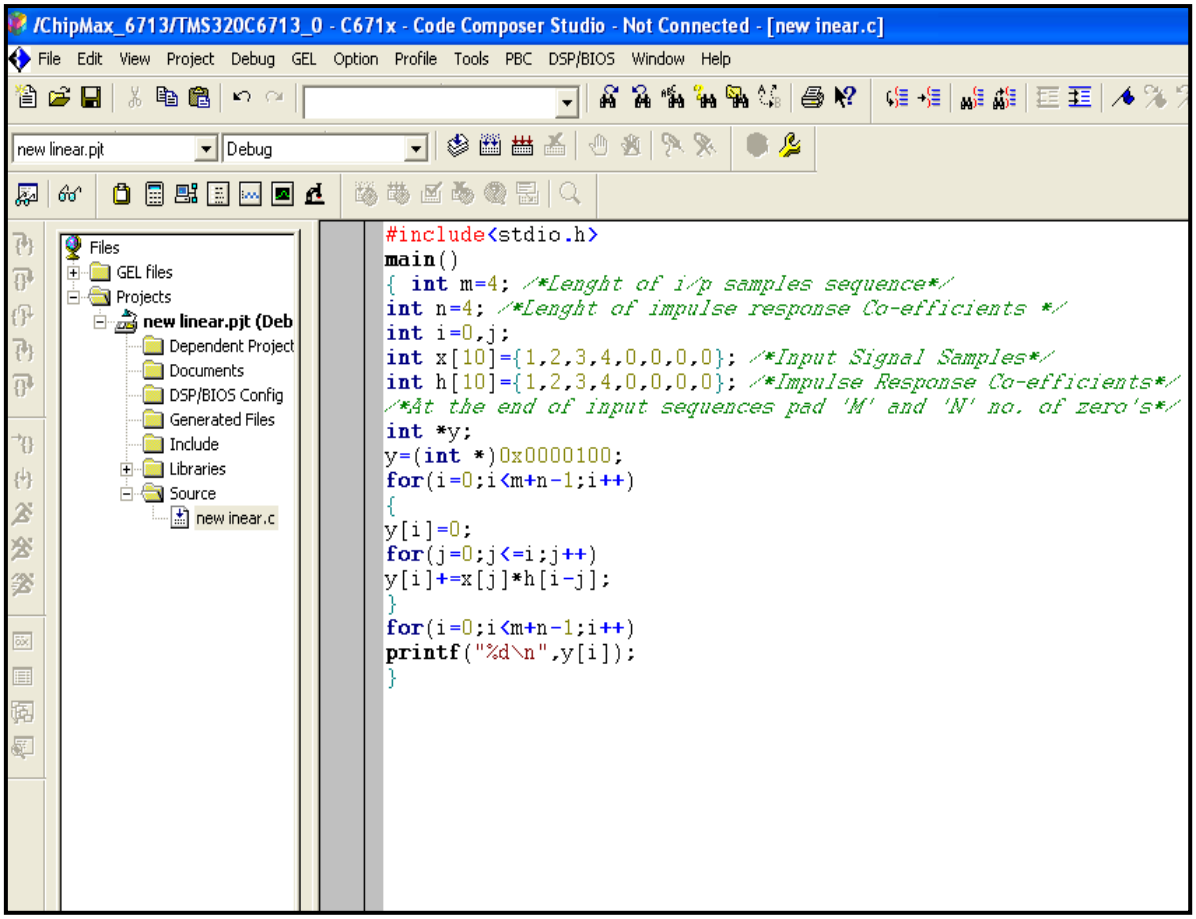
3. Click on File ⇨ New ⇨ Source File, to write the Source Code.



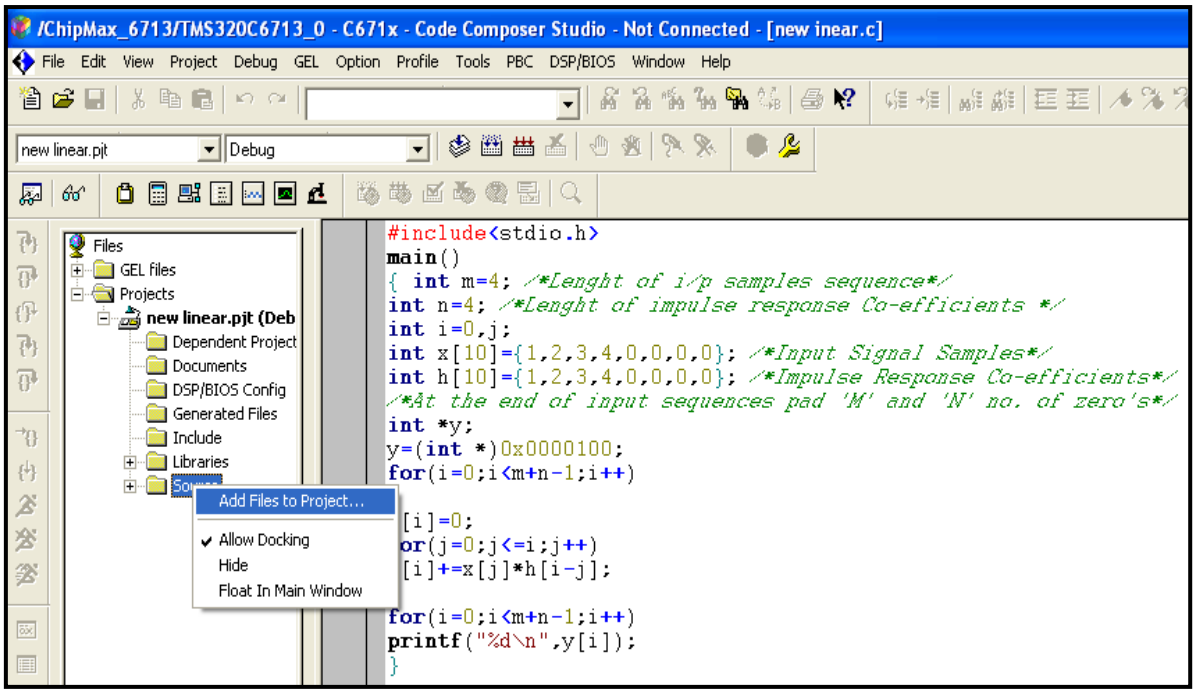
Program:

```
#include<stdio.h>
main()
{
int m=4;                          /*Lenght of i/p samples sequence*/
int n=4;                          /*Lenght of impulse response Co-efficients */
int i=0,j;
int x[10]={1,2,3,4,0,0,0,0};      /*Input Signal Samples*/
int h[10]={1,2,3,4,0,0,0,0};      /*Impulse Response Co-efficients*/
                                  /*At the end of input sequences pad 'M' and 'N' no. of zero's*/
int *y;
y=(int *)0x0000100;               /*start address*/
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]);
}
```

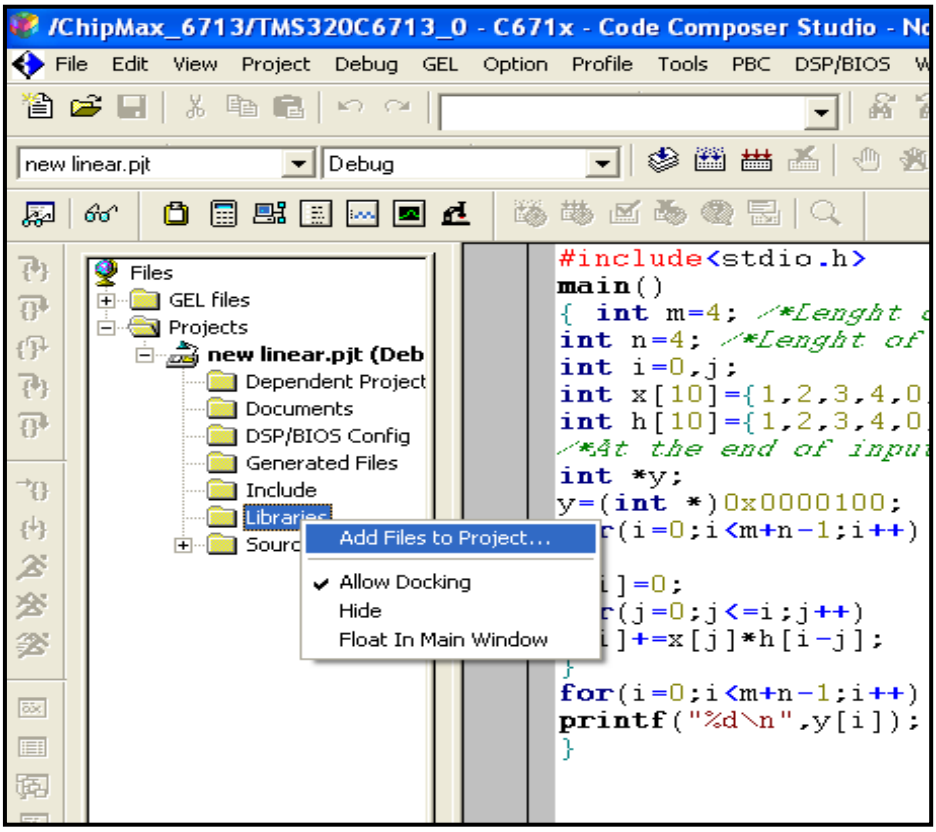
4. Enter the source code and save the file with ".C" extension in the created project itself.

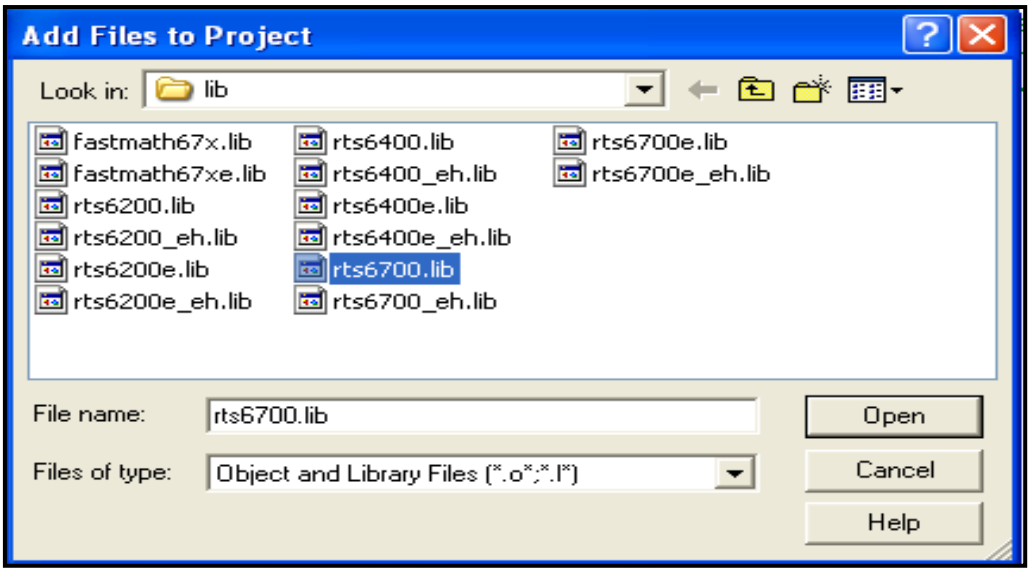


5. Right click on source, Select add files to project and Choose " .C " file Saved before in created project.

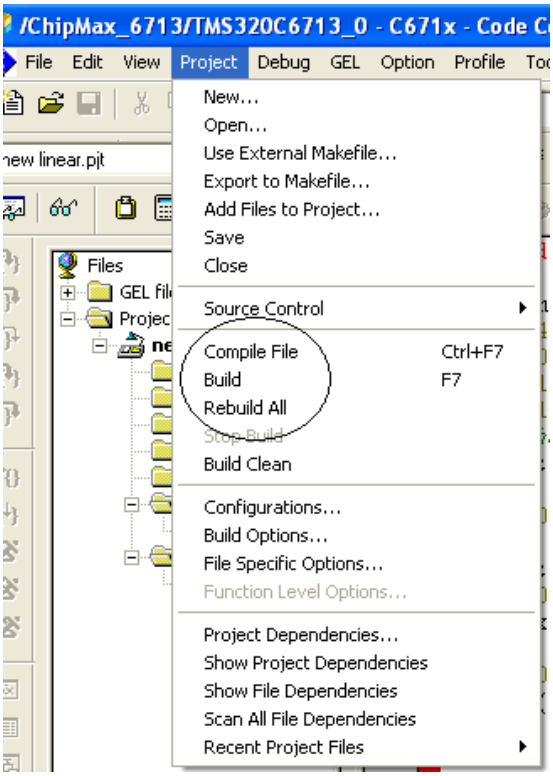


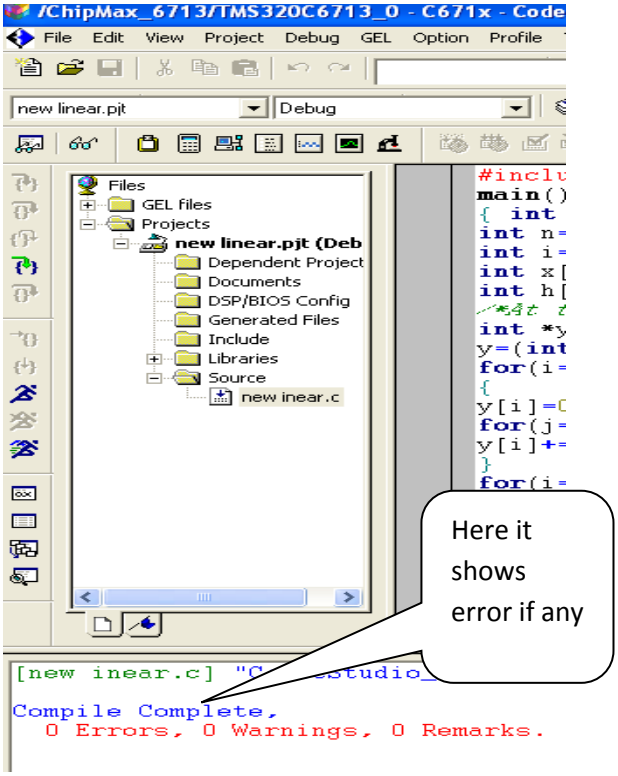
6. Right Click on libraries and select add files to Project and choose C:\CCStudio_v3.1\C6000\cgtools\lib\rts6700.lib and click open.



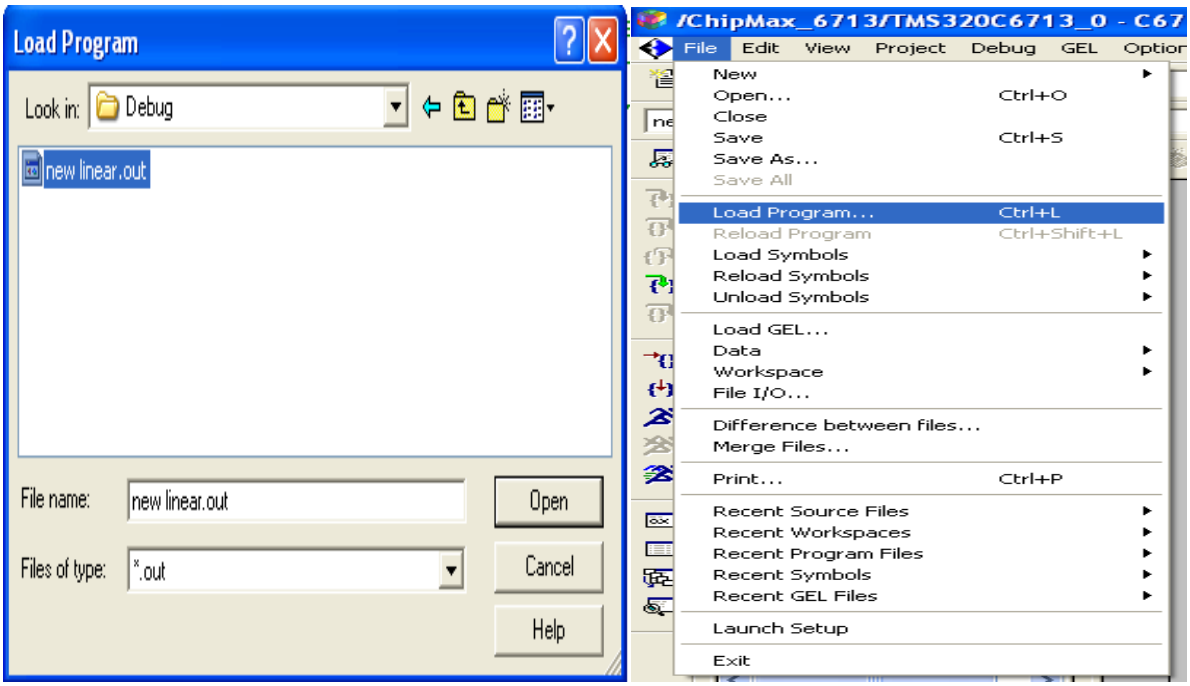


7. a) Go to Project to Compile .
   b) Go to Project to Build.
   c) Go to Project to Rebuild All.

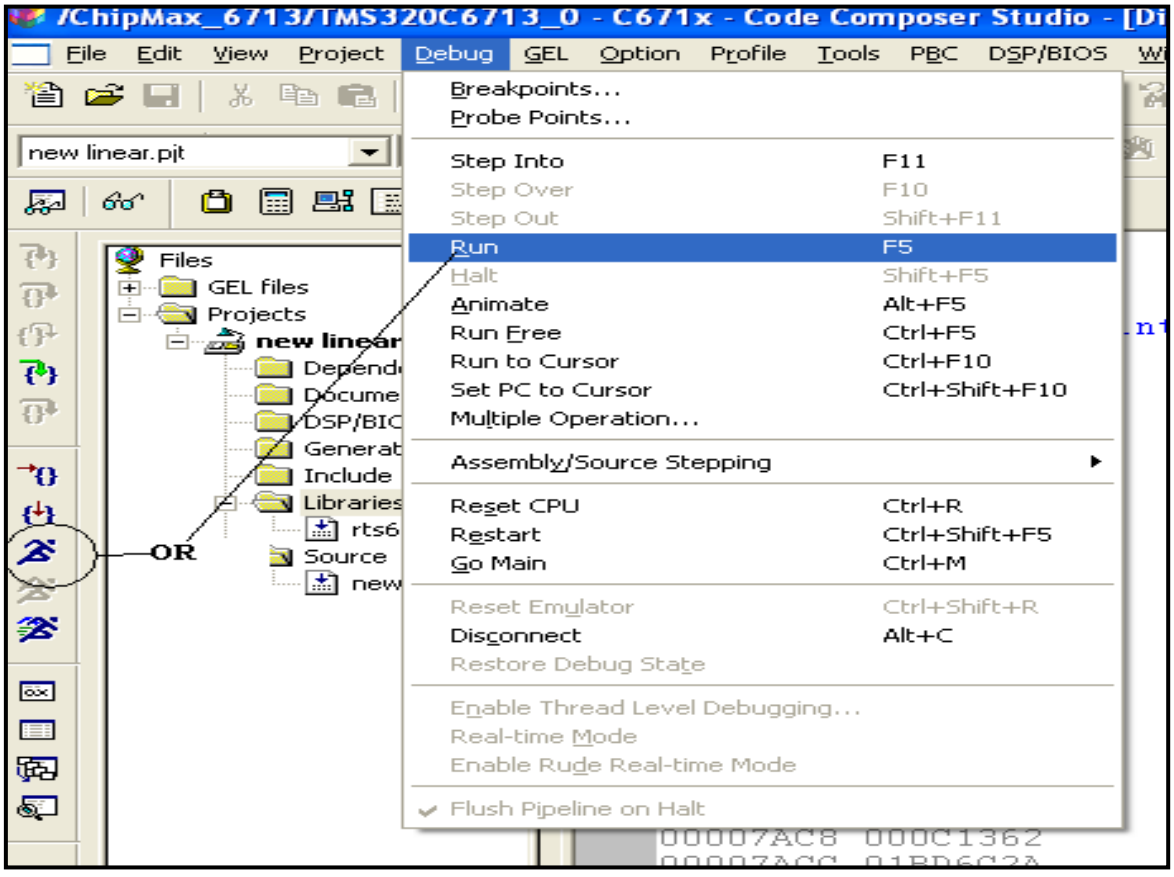




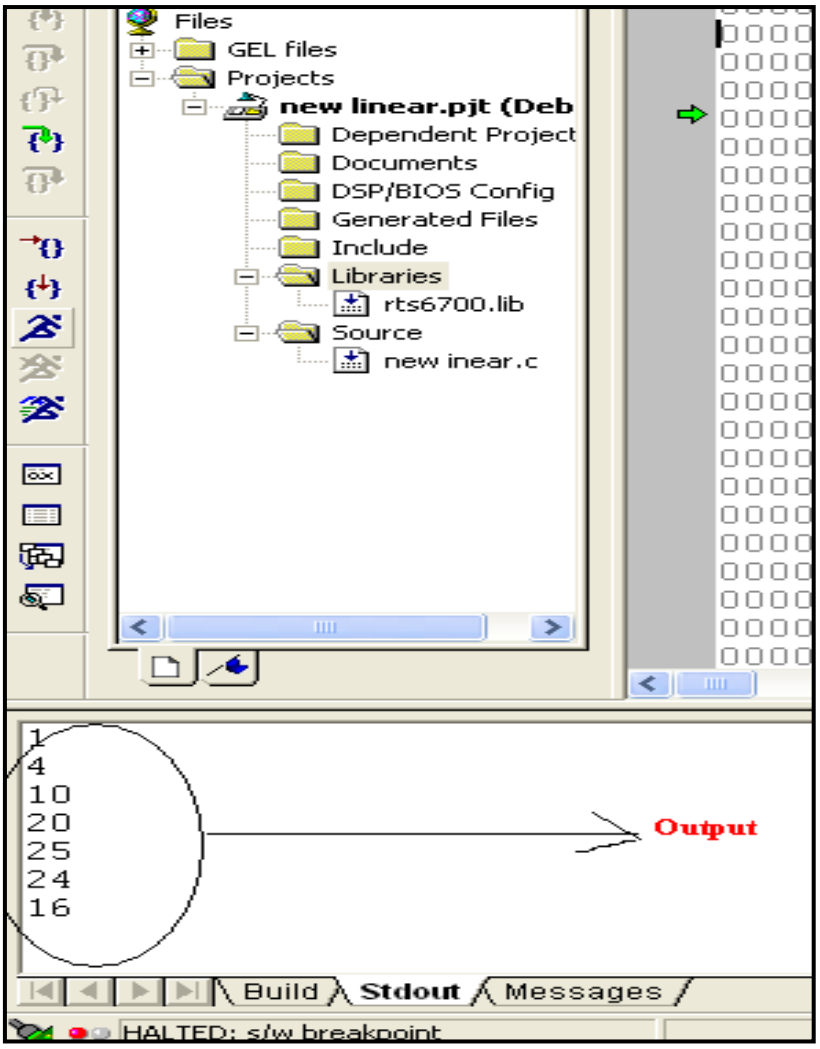
8. Go to file and load program and load **".out"** file into the board..



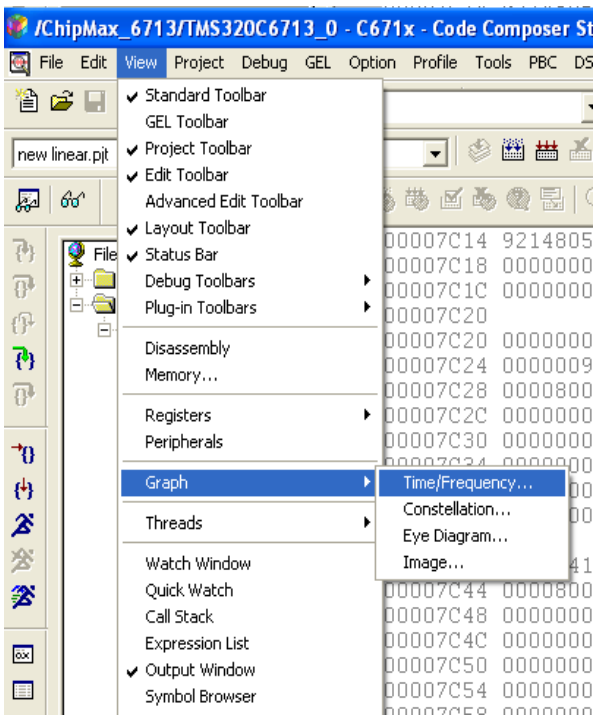
9. Go to Debug and click on run to run the program.

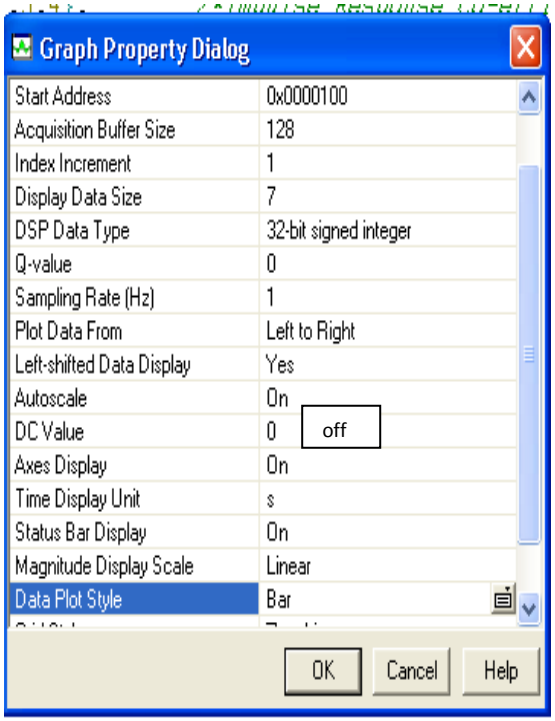


10. Observe the output in output window.

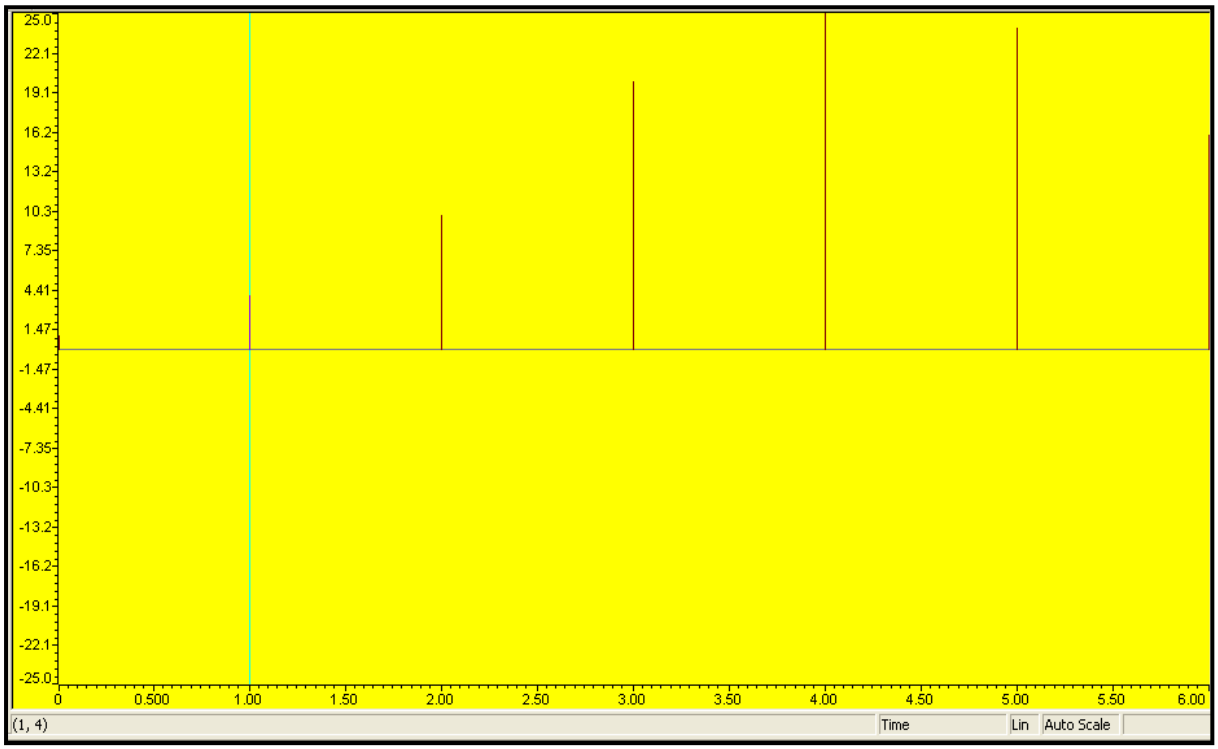


11. To see the Graph go to View and select time/frequency in the Graph, and give the correct Start address provided in the program, Display data can be taken as per user.





12. Green line is to choose the point, Value at the point can be seen (Highlighted by bracket at the left corner).



OUTPUT:

EXPERIMENT -10

AIM: TO IMPLEMENT CIRCULAR CONVOLUTION OF TWO SEQUENCES.

Circular Convolution:

Let $x_1(n)$ and $x_2(n)$ are finite duration sequences both of length N with DFT's $X_1(k)$ and $X_2(k)$. Convolution of two given sequences $x_1(n)$ and $x_2(n)$ is given by the equation,

$x_3(n) = IDFT[X_3(k)]$

$X_3(k) = X_1(k)\, X_2(k)$

$$x_3(n) = \sum_{m=0}^{N-1} x_1(m)\, x_2((n-m))_N$$

Program:

```
#include<stdio.h>
int m,n,x[30],h[30],y[30],i,j,temp[30],k,x2[30],a[30];
void main()
{
int *y;
y=(int *)0x0000100;
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 length of both sequences are not equal*/
{
if(m>n)                       /* Pad the smaller sequence with zero*/
{
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++)                    /*folding h(n) to h(-n)*/
a[j]=h[n-j];

/*Circular convolution*/
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];
}
}

/*displaying the result*/
printf(" the circular convolution is\n");
for(i=0;i<n;i++)
printf("%d\n ",y[i]);
}
```

Procedure to create new Project:

1. To create project, go to Project and Select New.
2. Give project name and click on finish.
3. Click on File ⇨ New ⇨ Source File, to write the Source Code.
4. Enter the source code and save the file with "**.C**" extension.
5. Right click on source, Select add files to project .. and Choose "**.C** " file Saved before.
6. Right Click on libraries and select add files to Project.. and choose C:\CCStudio_v3.1\C6000\cgtools\lib\rts6700.lib and click open.
7. a) Go to Project to Compile .
   b) Go to Project to Build.
   c) Go to Project to Rebuild All.
8. Go to file and load program and load "**.out**" file into the board.
9. Go to Debug and click on run to run the program.
10. Enter the input data to calculate the circular convolution.
11. To see the Graph go to View and select time/frequency in the Graph, and give the correct Start address (y) provided in the program, Display datatype (32-bit signed integer),autoscale (off), axies display-y axis (20).
12. Green line is to choose the point, Value at the point can be seen (Highlighted by circle at the left corner).

OUTPUT:

EXPERIMENT - 11

AIM: TO COMPUTE N-POINT DFT OF A GIVEN SEQUENCE AND TO PLOT MAGNITUDE AND PHASE SPECTRUM.

Discrete Fourier Transform: The Discrete Fourier Transform is a powerful computation tool which allows us to evaluate the Fourier Transform $X(e^{j\omega})$ on a digital computer or specially designed digital hardware. Since $X(e^{j\omega})$ is continuous and periodic, the DFT is obtained by sampling one period of the Fourier Transform at a finite number of frequency points. Apart from determining the frequency content of a signal, DFT is used to perform linear filtering operations in the frequency domain.

The sequence of N complex numbers $x_0,\ldots, x_{N-1}$ is transformed into the sequence of N complex numbers $X_0, \ldots, X_{N-1}$ by the DFT according to the formula:

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi nk/N} \qquad k = 0,1, \ldots. N-1$$

Program:

```
#include <stdio.h>
#include <math.h>
#define N 4                                   //number of data values

float pi = 3.1416;
short x[N] = {1,1,0,0};                       //1-cycle cosine
float out[2] = {0,0};                         //intermediate Re and Im results
float real[10],imaginary[10];                 //Output results
int index=0;
void dft(short *x, short k, float *out)       //DFT function
{
 float sumRe = 0;                             //initialize real component
 float sumIm = 0;                             //initialize imaginary component
 int i = 0;
 float cs = 0;                                //initialize cosine component
 float sn = 0;                                //initialize sine component
 for (i = 0; i < N; i++)                      //for N-point DFT
   {

    cs = cos(2*pi*(k)*i/N);                   //real component
    sn = sin(2*pi*(k)*i/N);                   //imaginary component
    sumRe = sumRe + x[i]*cs;                  //sum of real components
    sumIm = sumIm - x[i]*sn;                  //sum of imaginary components
   }
```

```
 out[0] = sumRe;                          //sum of real components
 real[index]=out[0];
 out[1] = sumIm;                                  //sum of imaginary components
 imaginary[index]=out[1];
 index++;
 printf("%f  j%f\n",out[0],out[1]);
}

void main()
{
 int j;
 for (j = 0; j < N; j++)
   {
    dft(x,j,out);                          //call DFT function
   }
}
```

Procedure to create new Project:

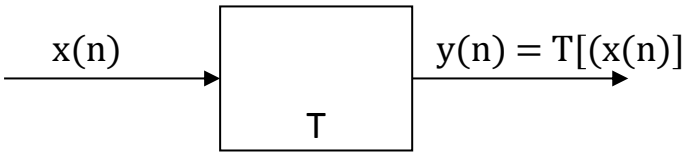
1. To create project, go to Project and Select New.
2. Give project name and click on finish.
3. Click on File ⇨ New ⇨ Source File, to write the Source Code.
4. Enter the source code and save the file with **".C"** extension.
5. Right click on source, Select add files to project and Choose "**.C** " file Saved before.
6. Right Click on libraries and select add files to Project.. and choose C:\CCStudio_v3.1\C6000\cgtools\lib\rts6700.lib and click open.
7. a) Go to Project to Compile .
   b) Go to Project to Build.
   c) Go to Project to Rebuild All.
8. Go to file and load program and load **".out"** file into the board.
9. Go to Debug and click on run to run the program.
10. Observe the output in output window.
11. To see the Graph go to View and select time/frequency in the Graph, select dual time with start address real and imaginary, select DSP data type as 32-bit floating point ,& autoscale as off , Display data can be taken as per user.
12. Green line is to choose the point, Value at the point can be seen (Highlighted by circle at the left corner).

OUTPUT:

EXPERIMENT - 12

AIM: TO FIND IMPULSE RESPONSE OF A FIRST ORDER AND SECOND ORDER SYSTEM.

A discrete time system performs an operation on an input signal based on predefined criteria to produce a modified output signal. The input signal $x(n)$ is the system excitation, and $y(n)$ is the system response. The transform operation is shown as,



The convolution sum can be represented by, $y(n) = x(n) * h(n)$

For Example let's find out an impulse response of a difference equation.

The general form of difference equation is,

$$y(n) = \sum_{k=1}^{M} a_k\, y(n-k) + \sum_{k=0}^{N} b_k\, x(n-k)$$

To find out the impulse response of second order difference equation.

Program:

```
#include<stdio.h>
#define Order  2
#define Len    5
float h[Len] = {0.0,0.0,0.0,0.0,0.0},sum;
void main()
{
int j, k;
float a[Order+1] = {0.1311, 0.2622, 0.1311};          // x(n) coefficients
float b[Order+1] = {1, -0.7478, 0.2722};                    // y(n) coefficients

for(j=0; j<Len; j++)
{
sum = 0.0;
for(k=1; k<=Order; k++)
{
if ((j-k) >= 0)
sum = sum+(b[k]*h[j-k]);
}
```

```
if (j <= Order)
h[j] = a[j]-sum;
else
h[j] = -sum;
printf (" %f \n  ",h[j]);
}
}
```

For first order difference equation.

Program:

```
#include<stdio.h>
#define Order  1
#define Len      5
float h[Len] = {0.0,0.0,0.0,0.0,0.0},sum;

void main()
{
int j, k;
float a[Order+1] = {0.1311, 0.2622};                    // x(n) coefficients
float b[Order+1] = {1, -0.7478};                        // y(n) coefficients

for(j=0; j<Len; j++)
{
sum = 0.0;
for(k=1; k<=Order; k++)
{
if((j-k)>=0)
sum = sum+(b[k]*h[j-k]);
}
if(j<=Order)
h[j] = a[j]-sum;
else
h[j] = -sum;
printf("%f \n", h[j]);
}
}
```

Procedure to create new Project:

1. To create project, Go to Project and Select New.
2. Give project name and click on finish.
3. Click on File ⇨ New ⇨ Source File, To write the Source Code.
4. Enter the source code and save the file with **".C"** extension.
5. Right click on source, Select add files to project .. and Choose "**.C** " file Saved before.

6. Right Click on libraries and select add files to Project.. and choose C:\CCStudio_v3.1\C6000\cgtools\lib\rts6700.lib and click open.

7. a) Go to Project to Compile.

   b) Go to Project to Build.

   c) Go to Project to Rebuild All.

8. Go to file and load program and load **".out"** file into the board..

9. Go to Debug and click on run to run the program.

10. Observe the output in output window.

11. To see the Graph go to View and select time/frequency in the Graph, and give the correct Start address or the output variable name provided in the program, Display data can be taken as per user. Select DSP data type as 32-bit floating point.

12. Green line is to choose the point, Value at the point can be seen (Highlighted by circle at the left corner).

OUTPUT:

EXPERIMENT - 13

AIM: REALIZATION OF FIR FILTER (ANY TYPE) TO MEET GIVEN SPECIFICATIONS. THE INPUT CAN BE A SIGNAL FROM FUNCTION GENERATOR/SPEECH SIGNAL

Finite Impulse Response (FIR) Filter: The FIR filters are of non-recursive type, whereby the present output sample is depending on the present input sample and previous input samples.

The transfer function of a FIR causal filter is given by,

$$H(z) = \sum_{n=0}^{N-1} h(n)z^{-n}$$

Where $h(n)$ is the impulse response of the filter.

The Fourier transform of $h(n)$ is

$$H(e^{jw}) = \sum_{n=0}^{N-1} h(n)e^{-jwn}$$

In the design of FIR filters most commonly used approach is using windows.

The desired frequency response $H_d(e^{jw})$ of a filter is periodic in frequency and can be expanded in Fourier series. The resultant series is given by,

$$h_d(n) = (1/2\pi) \int_{-\pi}^{\pi} H(e^{jw})e^{jwn}\, dw$$

And known as Fourier coefficients having infinite length. One possible way of obtaining FIR filter is to truncate the infinite Fourier series at $n = \pm [(N-1)/2]$

Where N is the length of the desired sequence.

The Fourier coefficients of the filter are modified by multiplying the infinite impulse response with a finite weighing sequence $w(n)$ called a window.

Where $w(n) = w(-n) \neq 0$ for $|n| \leq [(N-1)/2]$

$= 0$ for $|n| > [(N-1)/2]$

After multiplying $w(n)$ with $h_d(n)$, we get a finite duration sequence $h(n)$ that satisfies the desired magnitude response,

$h(n) = h_d(n)\, w(n)$ for $|n| \leq [(N-1)/2]$

$= 0$ for $|n| > [(N-1)/2]$

The frequency response $H(e^{jw})$ of the filter can be obtained by convolution of $H_d(e^{jw})$ and $W(e^{jw})$ is given by,

$$H(e^{jw}) = (1/2\pi) \int_{-\pi}^{\pi} H_d(e^{j\theta})\, W(e^{j(w-\theta)})\, d\theta$$

$$H(e^{jw}) = H_d(e^{jw}) * W(e^{jw})$$

Program:

```
#include <stdio.h>
#include "c6713dsk.h"
#include "master.h"
#include "aic23cfg.h"
#include "dsk6713_aic23.h"
#define FilLen 32

// Low Pass Filter
/*float FilCo[32]= {  -0.0006,-0.0056,-0.0129,-0.0193,-0.0176,-0.0043,  0.0156,  0.0282,
0.0190,-0.0129,0.0483,-0.0545,-0.0065, 0.0927, 0.2064, 0.2822, 0.2822, 0.2064, 0.0927,-0.0065,-
0.0545,-0.0483,-0.0129,  0.0190,  0.0282,  0.0156,-0.0043,-0.0176,-0.0193,-0.0129,-0.0056,-
0.0006};*/
// High Pass Filter
float FilCo[32] = { -0.0128,-0.0007,0.0068,0.0150, 0.0173, 0.0091,-0.0082,-0.0260,-0.0317,-
0.0160, 0.0200, 0.0611,0.0808,0.0481,-0.0803,-0.5875, 0.5875, 0.0803,-0.0481,-0.0808,-0.0611,-
0.0200,0.0160,0.0317, 0.0260, 0.0082,-0.0091,-0.0173,-0.0150,-0.0068,
0.0007, 0.0128};

DSK6713_AIC23_Config config = { \
  0x0017,  /* 0 DSK6713_AIC23_LEFTINVOL  Left line input channel volume */ \
  0x0017,  /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume */\
  0x00ff,  /* 2 DSK6713_AIC23_LEFTHPVOL  Left channel headphone volume */  \
  0x00ff,  /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */ \
  0x0010,  /* 4 DSK6713_AIC23_ANAPATH    Analog audio path control */     \
  0x0000,  /* 5 DSK6713_AIC23_DIGPATH    Digital audio path control */    \
  0x0000,  /* 6 DSK6713_AIC23_POWERDOWN  Power down control */           \

  0x0043,  /* 7 DSK6713_AIC23_DIGIF     Digital audio interface format */ \
  0x008c,  /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control */          \
  0x0001   /* 9 DSK6713_AIC23_DIGACT     Digital interface activation */   \
};

DSK6713_AIC23_CodecHandle hCodec;
Int32 LInc = 4;
Int32 RInc = 3;
Int32 LData;
```

```
Int32 RData;
Int32 InChannel =0;
Int32 OtChannel =0;
Int32 LIndex = 0;
Int32 RIndex = 0;
Int32 LDataArr[128];
Int32 RDataArr[128];
Int32 LWP=0;
Int32 LRP=100;
Int32 RWP=0;
Int32 RRP=100;
float LPrBuf[150];
float RPrBuf[150];
float FilOut= 0;

main(){
        int i;
        LED=0x0;
        // Filter Initialization
        for( i = 0 ; i < 100; i++ ) LPrBuf[i]=0.0;
        for( i = 0 ; i < 100; i++ ) RPrBuf[i]=0.0;

        // Initialize codec
        hCodec = DSK6713_AIC23_openCodec(0, &config);
        IRQ_globalEnable();
        IRQ_enable(IRQ_EVT_RINT1);
        IRQ_enable(IRQ_EVT_XINT1);
}
void led(){
        static int cc = 1;
        LED     = cc;
        // To Shift Pattern
        if (cc == 0x03) cc = 0x05;
        else if (cc == 0x05) cc = 0x06;
        else if (cc == 0x06) cc = 0x03;
        else cc = 0x03;
}

setpll200M(){
}
void read(){
        int i = 0;

        if (InChannel ==0){
                InChannel =1;
                // Process Left Channel
                LData=MCBSP_read(DSK6713_AIC23_DATAHANDLE);
                for( i = 0; i <= FilLen-2; i++) LPrBuf[i] = LPrBuf[i+1];
                LPrBuf[FilLen-1] = (float) LData;
```

```
                FilOut = 0.0;
                for( i = 0 ; i <= FilLen-1; i++ )          FilOut += (FilCo[i] * LPrBuf[i]);
                LData = (int) (FilOut/16);
                LDataArr[LWP++] =  LData;
                if (LWP >= 128) LWP =0;
        }
        else{
                InChannel =0;
                // Process Right Channel
                RData=MCBSP_read(DSK6713_AIC23_DATAHANDLE);
                RDataArr[RWP++] = RData;
                if (RWP >= 128) RWP =0;
        }
}

void write(){
        if (OtChannel == 1){
                MCBSP_write(DSK6713_AIC23_DATAHANDLE,LDataArr[LRP++]);
                if (LRP >= 128)         LRP = 0;
                OtChannel = 0;
        }
        else{
                MCBSP_write(DSK6713_AIC23_DATAHANDLE,RDataArr[RRP++]);
                if (RRP >= 128)         RRP = 0;
                OtChannel = 1;
        }
}
```

Note: Output can be seen on CRO.

Procedure to create new Project:

1. To create project, go to Project and Select New.
2. Give project name and click on finish.
3. Click on File ⇨ New ⇨ Source File, to write the Source Code.
4. Enter the source code and save the file with main.c extension.
5. Right click on source, Select add files to project .. and Choose main.c file Saved before.
6. Add the other supporting .c files which configure the audio codec.
7. a) Go to Project to Compile .
   b) Go to Project to Build.
   c) Go to Project to Rebuild All.
8. Go to file and load program and load **".out"** file into the board.
9. Go to Debug and click on run to run the program.
10. Output can be seen on CRO with the filtering effect.

OUTPUT:

MAHARAJA INSTITUTE OF TECHNOLOGY MYSORE
Belawadi, S.R. Patna Taluk, Mandya – 571438
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

Course Name: **Digital Signal Processing Laboratory** Course Code: **15ECL57**

CO's	DESCRIPTION OF THE OUTCOMES
15ECL57.1	**Understand** the basic concepts of Digital Signal processing techniques with their Properties both in time and frequency domain.
15ECL57.2	**Design and develop** the software modules for digital filters and digital signal processing techniques/operations using Modern Software Tools.
15ECL57.3	**Implement** Digital signal processing techniques/operations and Digital filters using DSP Processor.
15ECL57.4	**Documentation** of the complete experimental process and results.

INDEX

Sl No	TOPIC AND MARKS PER CO	**CO1 (05 Marks)** Comprehension	**CO2, (05 Marks)** Design & Develop	**CO3 (05 Marks)** Implementation	**CO4 (05 Marks)** Documentation	**Total** (20Marks)
1	Experiment 1					
2	Experiment 2					
3	Experiment 3					
4	Experiment 4					
5	Experiment 5					
6	Experiment 6					
7	Experiment 7					
8	Experiment 8					
9	Experiment 9					
10	Experiment 10					
11	Experiment 11					
12	Experiment 12					
13	Experiment 13					
Rounded Average						

Lab Internal Assessment:

CO1 (05 Marks) Comprehension	**CO2, CO3 (10 Marks)** Design, Develop & Implementation		**CO4(05)** Documentation	**Total** (20Marks)

INTERNAL SESSIONAL MARKS AWARDED	
In figures	**In words**
/20	

Signature of Staff Member(s) Signature of HoD

The evaluation committee has approved the following of the course **DIGITAL SIGNAL PROCESSING LAB** (15ECL57) for the academic year **2018-19**

1. Course Outcomes and mapping to Program outcomes
2. Lesson Plan
3. Lab Manual
4. Lab Evaluation Sheet
5. Evaluation scheme for IA
6. Attainment Targets

Approved on:
Evaluation Committee:

MAHARAJA INSTITUTE OF TECHNOLOGY MYSORE
Belawadi, S.R. Patna Taluk, Mandya – 571438
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

Attainment Targets

Course Coordinator : **Prof. Devendran B.**

Course : **Digital Signal Processing Lab**

Course Code : **15ECL57**

Semester : **V**

Total Number of Students :

Attainment Level for Internal Assessments for the academic year 2018-19

Level 1 : 70% of students score more than 70%marks

Level 2 : 70% of students score more than 75%marks

Level 3 : 70% of students score more than 80% marks

Attainment Level for Semester End Examination for the academic year 2018-19

Level 1 : 60% of students score more than 65%marks

Level 2 : 60% of students score more than 70%marks

Level 3 : 60% of students score more than 75%marks

Justification for the set level of attainment based on previous records

Proposed by:

Prof. Devendran B **Prof. Dhanushree V.** **Prof. Niveditha H.R.**	**Prof. Anisha P.S.** **Prof. Nihal Mohammadi.** **Prof. Suma R.**	**Prof. Devendran B**
Faculty		**Course Coordinator**

Approved by:

EVALUATION COMMITTEE		
Facilitator	**NBA coordinator**	**HOD**