||Jai Sri Gurudev||

Sri Adichunchanagiri Shikshana Trust®

SJC INSTITUTE OF TECHNOLOGY

Chickballapur

Affiliated to VTU approved by AICTE, New Delhi accredited by NBA:ISO 9001:2008 certified

DEPARTMENT OF ELECTRONICS & COMMUNICATION

DIGITAL SIGNAL PROCESSING LABORATORY MANUAL

(21EC42)

Prepared By: *P.SUDIR*

Assoc.Professor,Dept of E&C,SJCIT

Verfied by:

Dr.SHOBHA.N

HOD,Dept of E&C,SJCIT

Course Objective

C307.1	Develop programs for digital signal processing algorithms using Scilab
C307.2	Build interfacing programs to DSP chip using C
C307.3	Make use of tool such as Scilab and code composer studio for providing solutions
C307.4	Experiment with Texas processors TMS320C6713
C307.5	Improve the programs to meet DSP applications like building digital filters
C307.6	Demonstrate the system to meet real time signal processing

Course Specific Outcome

	PSO1	PSO2
C307.1	3	3
C307.2	3	3
C307.3	3	3
C307.4	3	3
C307.5	3	3
C307.6	3	3
C307	3	3

At the end of the program students will have

PSO1: Ability to absorb and apply fundamental knowledge of core Electronics and Communication Engineering in the analysis, design and development of Electronics Systems as well as to interpret and synthesize experimental data leading to valid conclusions

PSO2: Ability to solve complex Electronics and Communication Engineering problems, using latest hardware and software tools, along with analytical and managerial skills to arrive at appropriate solutions, either independently or in team

RUBRICS FOR LAB

- **FOR 20 MARKS**

Sl.No.	**DESCRIPTION**	**MARKS**	Scaled marks
1.	**CONTINUOUS EVALUATION** • Observation write up & punctuality • Conduction of experiment and output • Viva voce • Record write up	**30** 5 10.0 5 10.0	15
2.	**INTERNAL TEST**	**50**	5

INDEX

Student Name: **Max Marks: ……..**

USN:

SL. NO	NAME OF THE EXPERIMENT	Revised Bloom's Taxonomy (RBT) Level
	PART-A:	
1	Computation of N point DFT of a given sequence and to plot magnitude and phase spectrum.	L2, L3, L4
2	Computation of circular convolution of two given sequences and verification of commutative, distributive and associative property of convolution.	L2, L3, L4
3	Computation of linear convolution of two sequences using DFT and IDFT.	L2, L3, L4
4	Computation of circular convolution of two given sequences using DFT and IDFT	L2, L3, L4
5	Verification of Linearity property, circular time shift property & circular frequency shift property of DFT.	L2, L3, L4
6	Verification of Parseval's theorem	L2, L3, L4
7	Design and implementation of IIR (Butterworth) low pass filter to meet given specifications.	L2, L3, L4
8	Design and implementation of IIR (Butterworth) high pass filter to meet given specifications.	L2, L3, L4
9	Design and implementation of low pass FIR filter to meet given specifications.	L2, L3, L4
10	Design and implementation of high pass FIR filter to meet given specifications.	L2, L3, L4
11	To compute N- Point DFT of a given sequence using DSK 6713 simulator.	L2, L3, L4
12	To compute linear convolution of two given sequences using DSK 6713 simulator	L2, L3, L4
13	To compute circular convolution of two given sequences using DSK 6713 simulator	L2, L3, L4

Experiment – 1

Aim: Computation of N point DFT of a given sequence and to plot magnitude and phase spectrum.

Theory: In mathematics, the **discrete Fourier transform** (**DFT**) converts a finite sequence of equally-spaced samples of a function into a same-length sequence of equally-spaced samples of the discrete-time Fourier transform (DTFT), which is a complex-valued function of frequency. The interval at which the DTFT is sampled is the reciprocal of the duration of the input sequence. An inverse DFT is a Fourier series, using the DTFT samples as coefficients of complex sinusoids at the corresponding DTFT frequencies. It has the same sample-values as the original input sequence. The DFT is therefore said to be a frequency domain representation of the original input sequence. If the original sequence spans all the non-zero values of a function, its DTFT is continuous (and periodic), and the DFT provides discrete samples of one cycle. If the original sequence is one cycle of a periodic function, the DFT provides all the non-zero values of one DTFT cycle.

The *discrete Fourier transform* transforms a sequence of *N* complex numbers $\{\mathbf{x_n}\} := x_0, x_1, \ldots, x_{N-1}$ into another sequence of complex numbers

$\{\mathbf{X_k}\} := X_0, X_1, \ldots, X_{N-1}$, which is defined by

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{2\pi i}{N} kn} = \sum_{n=0}^{N-1} x_n \cdot [\cos(2\pi kn/N) - i \cdot \sin(2\pi kn/N)],$$

where the last expression follows from the first one by Euler's formula.

The transform is sometimes denoted by the symbol $\mathcal{F}$, as in $\mathbf{X} = \mathcal{F}\{\mathbf{x}\}$ or $\mathcal{F}(\mathbf{x})$ or $\mathcal{F}\mathbf{x}$.

Example

Let $N = 4$ and $\mathbf{x} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2-i \\ -i \\ -1+2i \end{pmatrix}$. Here we demonstrate how to calculate the DFT of $\mathbf{x}$ using **Eq.1**:

$$X_0 = e^{-i2\pi 0 \cdot 0/4} \cdot 1 + e^{-i2\pi 0 \cdot 1/4} \cdot (2-i) + e^{-i2\pi 0 \cdot 2/4} \cdot (-i) + e^{-i2\pi 0 \cdot 3/4} \cdot (-1+2i) = 2$$

$$X_1 = e^{-i2\pi 1 \cdot 0/4} \cdot 1 + e^{-i2\pi 1 \cdot 1/4} \cdot (2-i) + e^{-i2\pi 1 \cdot 2/4} \cdot (-i) + e^{-i2\pi 1 \cdot 3/4} \cdot (-1+2i) = -2-2i$$

$$X_2 = e^{-i2\pi 2 \cdot 0/4} \cdot 1 + e^{-i2\pi 2 \cdot 1/4} \cdot (2-i) + e^{-i2\pi 2 \cdot 2/4} \cdot (-i) + e^{-i2\pi 2 \cdot 3/4} \cdot (-1+2i) = -2i$$

$$X_3 = e^{-i2\pi 3 \cdot 0/4} \cdot 1 + e^{-i2\pi 3 \cdot 1/4} \cdot (2-i) + e^{-i2\pi 3 \cdot 2/4} \cdot (-i) + e^{-i2\pi 3 \cdot 3/4} \cdot (-1+2i) = 4+4i$$

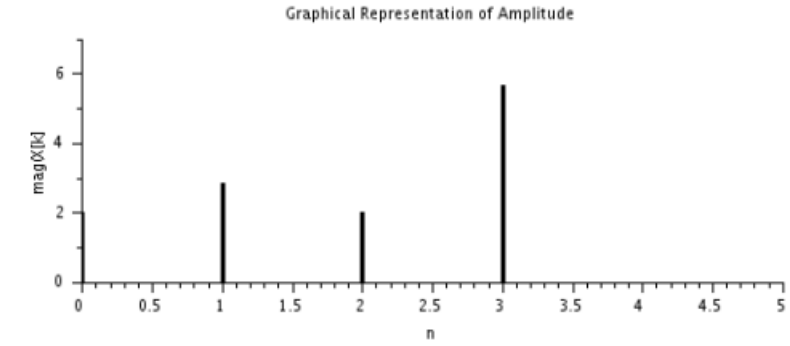
$$\mathbf{X} = \begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} 2 \\ -2-2i \\ -2i \\ 4+4i \end{pmatrix}$$

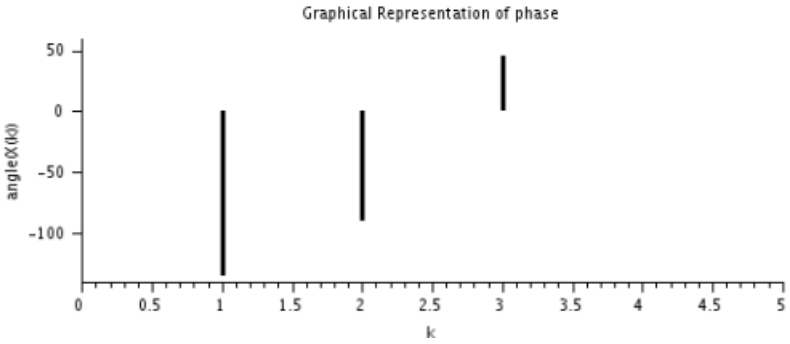
Code1:

```
//    Computation of N point DFT
clear ;
clc ;
close ;
i=sqrt(-1)
L=4; // Length of a sequence
N=4; // N-Point DFT
x=[1,2-i,-i,-1+2.*i];
// Computing DFT
X=fft(x,-1) ;
disp(X,'FFT of x[n] is X(k)=')
mag=abs(X)
// Plotting the spectrum
subplot(2,1,1)
a=gca();
a.data_bounds=[0,0;5,7];
plot2d3('gnn',0:length(mag)-1,mag)
b=gce() ;
b.children(1).thickness=3;
xtitle('Graphical Representation of
Amplitude','n','mag(X[k]');
phase=phasemag(X)
subplot(2,1,2)
a=gce();
a.data_bounds=[0,0;5,7];
plot2d3('gnn',0:length(XX)-1,XX)
b=gce();
b.children(1).thickness=3;
xtitle('Graphical Representation of phase','k','angle(X(k))') ;
```

Output:

```
 FFT of x[n] is X(k)=

  2.  - 2. - 2.i  - 2.i    4. + 4.i

 mag  =

  2.    2.8284271    2.    5.6568542

 phase  =

  0.  - 135.  - 90.    45.
```


Graphical Representation of Amplitude
magX[k]
n


Graphical Representation of phase
angleX(k)
k

Experiment – 2

Aim: Computation of circular convolution of two given sequences and verification of commutative, distributive and associative property of convolution.

Theory

Circular Convolution

The **circular convolution**, also known as **cyclic convolution**, of two aperiodic functions) occurs when one of them is convolved in the normal way with a periodic summation of the other function. That situation arises in the context of the Circular convolution theorem. The identical operation can also be expressed in terms of the periodic summations

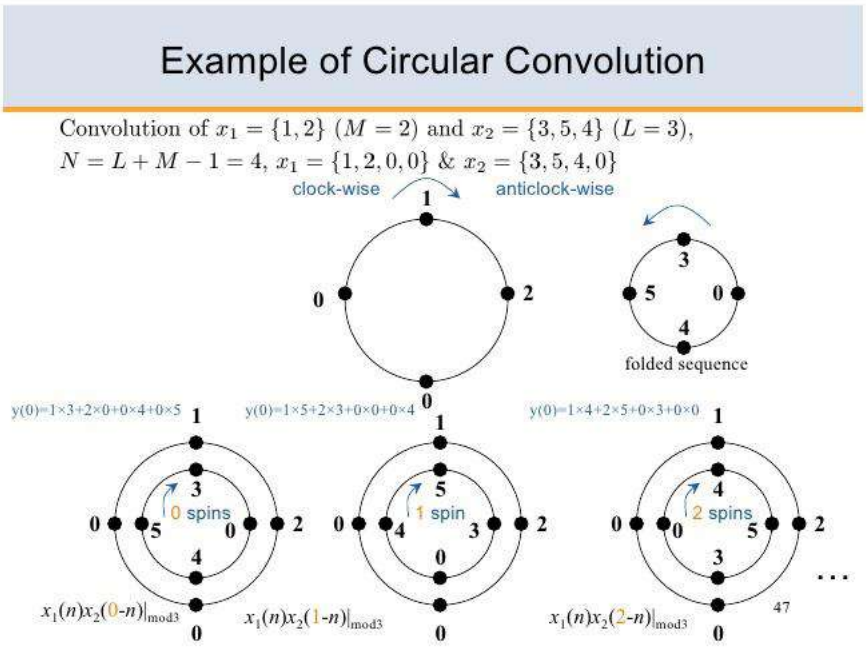
of both functions, if the infinite integration interval is reduced to just one period. That situation arises in the context of the discrete-time Fourier transform (DTFT) and is also called **periodic convolution**.

Circular convolution of two sequences x_1 and x_2 is defined as

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

The two sequences are $x_1(n) = \{1,2\}$ and $x_2(n) = \{3,5,4\}$

Each sequence consists of four nonzero points. For purpose of illustrating the operations involved in circular convolution it is desirable to graph each sequence as points on a circle. Thus the sequences x1 (n) and x2 (n) are graphed as illustrated in the **fig**. We note that the sequences are graphed in a counterclockwise direction on a circle. This establishes the reference direction in rotating one of sequences relative to the other. Now, y (m) is obtained by circularly convolving x (n) with h (n).



Code for Circular Convolution

```
//  Code for Circular Convolution
clc ;
function [c]=circpro(a,b)
 x=[[a(1),a(4),a(3),a(2)] ;
    [a(2),a(1),a(4),a(3)] ;
    [a(3),a(2),a(1),a(4)] ;
    [a(4),a(3),a(2),a(1)]] ;
 c=x*b';
endfunction
a = [1,2,0,0]; // First sequence
b = [3,5,4,0]; // Second sequence
m = length (a) ; // length of first sequence
n = length (b) ; // length of second sequence

C1=circpro(a,b);
disp(C1','Circular Convolution using Matrix Method result ')
```

```
Result:

Circular Convolution using Matrix Method result
    3.    11.    14.    8.
```

Commutative Property: $a \circledast b = b \circledast a$

```
// Commutative Property
clc
lhs=circpro(a,b);
rhs=circpro(b,a);
disp(lhs','LHS')
disp(rhs','RHS')
disp('Commutative Property is verified')
Result:
LHS
  3.   11.   14.   8.
 RHS
  3.   11.   14.   8.
 Commutative Property is verified
```

Distributive Property

a $\circledast$ (b+c)=a $\circledast$ b + a $\circledast$ c

Code:

// Distributive Property

```
clc
a = [1,2,0,0]; // First sequence
b = [3,5,4,0]; // Second sequence
c= [1,2,3,0]; // Third Sequence
lhs=circpro(a,(b+c));
rhs=(circpro(a,b)+ circpro(a,c));
disp(lhs','LHS')
disp(rhs','RHS')
disp('Distributive Property is verified')
Result:
LHS
   4.   15.   21.   14.
 RHS
   4.   15.   21.   14.
 Distributive Property is verified
```

Associativity Property : (a $\circledast$ b) $\circledast$ c = a $\circledast$ (b $\circledast$ c)

```
// Associativity Property
clc
x1=circpro(a,b);
lhs=circpro(x1',c);
x2=circpro(b,c);
rhs=circpro(a,x2');
disp(lhs,'LHS')
disp(rhs,'RHS')
disp('Associativity Property is verified')
```

Experiment –3

Aim: **Computation of linear convolution of two sequences using DFT and IDFT.**

Theory:

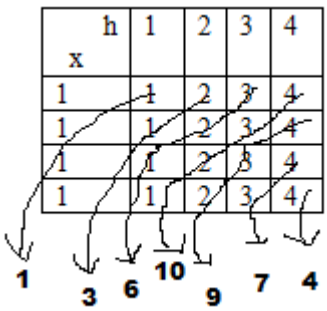
The mathematical definition of convolution in discrete time domain

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

where *x*[n] is input signal, h(n) is called Impulse Response and y[n] is output convolution. Here we multiply the terms of x[k] by the terms of time-shifted h[n] and add them up.
Here one of the input is shifted in time by a value every time it is multiplied with the other input signal. Linear Convolution is quite often used as a method of implementing filters of various types.

Eg: Perform Linear convolution of x[n] $= \{1, \overset{\leftrightarrow}{1}, 1,1\}$ and h[n] $= \{\overset{\leftrightarrow}{1}, 2,3,4\}$

Tabular Column method

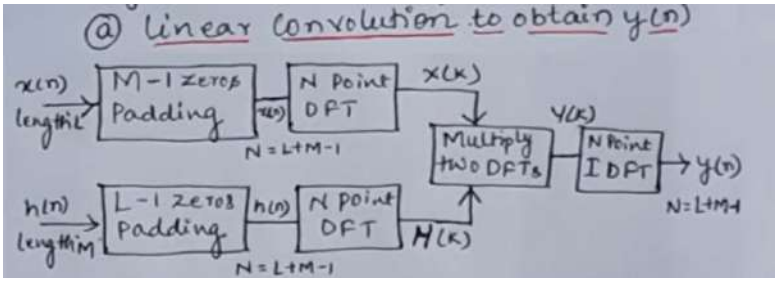


Answer : y[n] $= \{1, \overset{\leftrightarrow}{3}, 6,10,9,7,4\}$

The linear convolution of two sequences can be computed using the DFT (Discrete Fourier Transform) and IDFT (Inverse Discrete Fourier Transform) as follows:

1. Compute the DFT of both sequences.
2. Multiply the DFTs element-wise.
3. Compute the inverse DFT of the result from step 2.
4. The result of the inverse DFT is the convolution of the two sequences.

Note: The length of the DFTs must be zero-padded to ensure that the result of the inverse DFT is the correct length.



Code:

```
//Computation of linear convolution
clc
// Input sequences
x = [1,1,1,1];
y = [1,2,3,4];
// Find the length of the result
N = length(x) + length(y) - 1;
// Zero-pad the sequences to length N
x = [x, zeros(1, N-length(x))];
y = [y, zeros(1, N-length(y))];
// Compute the DFT of both sequences
X = fft(x);
Y = fft(y);
// Multiply the DFTs element-wise
Z = X .* Y;
// Compute the inverse DFT
z = ifft(Z);
// Round the result to remove any imaginary parts
z = round(real(z));
// Display the result
disp(z,' Linear Convolution Output');
```

Output:

Linear Convolution Output

1. 3. 6. 10. 9. 7. 4.

Experiment –4

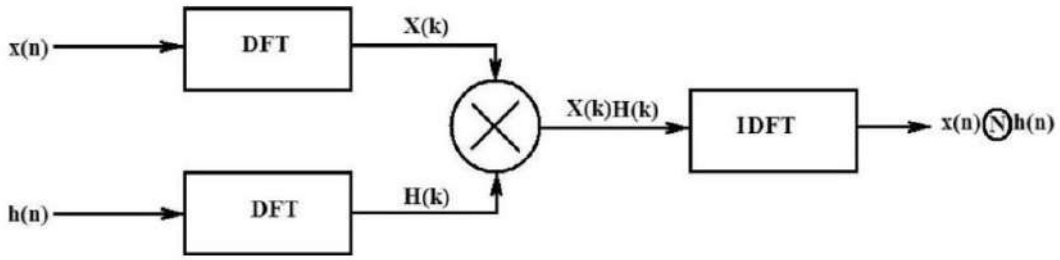
Aim: Computation of circular convolution of two given sequences using DFT and IDFT

Theory:

Circular convolution of two sequences x and y is computed as follows using DFT and IDFT:

1. Compute the DFT of both sequences x and y, resulting in X and Y respectively.
2. Multiply element-wise X and Y to get Z.
3. Compute the IDFT of Z to obtain the circular convolution of x and y.

Note: The length of the sequences must be equal and a power of 2 for efficient computation using the fast Fourier transform (FFT) algorithm.



Code:

```
// Circular convolution
clc ;
L=4; // Length of the Sequence
N=4; // N –p o i n t DFT
x=[1,2,0,0];
h=[3,5,4,0];
// Computing DFT
X1=fft(x ,-1) ;
X2=fft(h ,-1) ;
// M u l t i p l i c a t i o n of 2 DFTs
X3=X1.*X2 ;
// 'Circular Convolution Result
x3=abs(fft(X3,1) );
disp ( x3 ,'Circular Convolution Result')
```

Output: **Circular Convolution Result**

3. 11. 14. 8.

Experiment –5

Aim: Verification of Linearity property, circular time shift property & circular frequency shift property of DFT.

Proof :

a) Linearity property

The linearity property of the discrete Fourier transform (DFT) states that:
For any two sequences x and y, and any two scalars a and b, the DFT of the sequence (ax + by) is equal to (aX) + (bY), where X and Y are the DFTs of x and y, respectively.
Mathematically, it can be expressed as:
DFT(ax + by) = aDFT(x) + bDFT(y)
This property makes the DFT a useful tool for analyzing and processing linear systems.

Linearity. If

$$x_1(n) \overset{\text{DFT}}{\underset{N}{\longleftrightarrow}} X_1(k)$$

and

$$x_2(n) \overset{\text{DFT}}{\underset{N}{\longleftrightarrow}} X_2(k)$$

then for any real-valued or complex-valued constants a_1 and a_2,

$$a_1x_1(n) + a_2x_2(n) \overset{\text{DFT}}{\underset{N}{\longleftrightarrow}} a_1X_1(k) + a_2X_2(k)$$

b) Circular Time Shift Property

The circular time shift property of the discrete Fourier transform (DFT) states that:
For any sequence x of length N, and an integer k such that 0 <= k < N, the DFT of the circularly shifted sequence x_shift, where x_shift[n] = x[(n-k) mod N], is equal to the element-wise multiplication of the DFT of x by a complex exponential sequence.
Mathematically, it can be expressed as:
DFT(x_shift) = DFT(x) * exp(-j * 2 * π * k * n / N)
where j is the imaginary unit, π is the mathematical constant pi, and * represents element-wise multiplication.
This property makes the DFT a useful tool for analyzing and processing signals that have undergone circular time shifts.

CIRCULAR TIME SHIFT

Statement: The Circular time shift property of DFT says that if a discrete time signal is circularly shifted in the time by 'm' units then its DFT is multiplied by $e^{-j2\pi km/N}$

Let DFT[x(n)]=X(K) then,

$X(K).\ e^{-j2\pi km/N}=DFT[x(n-m)_N]$

c) circular frequency shift property

The circular frequency shift property of the discrete Fourier transform (DFT) states that:

For any sequence x of length N, and an integer k such that 0 <= k < N, the inverse DFT (IDFT) of the element-wise multiplication of the DFT of x by a complex exponential sequence, is equal to the circularly shifted sequence x_shift, where x_shift[n] = x[(n+k) mod N].

Mathematically, it can be expressed as:

IDFT(DFT(x) * exp(j * 2 * π * k * n / N)) = x_shift

where j is the imaginary unit, π is the mathematical constant pi, and * represents element-wise multiplication.

This property makes the DFT a useful tool for analyzing and processing signals that have undergone circular frequency shifts.

CIRCULAR FREQUENCY SHIFT

Statement: **The Circular frequency shift property of DFT says that if a discrete time signal is multiplied by $e^{j2\pi mn/N}$ its DFT is circularly shifted by m units**

Let DFT{x(n)}=X(K) then,

$\text{DFT}\{x(n).e^{j2\pi mn/N}\}=x((K-m))_N$

Linearity:

```
// Linearity Property
clc
// Define the two sequences x and y
x = [1, 2, 3, 4];
y = [5, 6, 7, 8];
// Define the scalars a and b
a = 2;
b = 3;
// Compute the DFT of x and y using the fft function
X = fft(x);
Y = fft(y);
// Verify the linearity property
z = a * x + b * y;
Z = fft(z);
disp("DFT of (ax + by):");
disp(Z);
Y=a*fft(x)+b*fft(y);
disp("aDFT(x) + bDFT(y):");
disp(Y);
```

Output:

```
DFT of (ax + by):
   98.  - 10. + 10.i  - 10.  - 10. - 10.i

 aDFT(x) + bDFT(y):
   98.  - 10. + 10.i  - 10.  - 10. - 10.i
```

Circular Time Shift Property

```
// Circular Time Shift Property
clc;
x = [1, 2, 3, 4, 5];
// Apply a circular time shift to the signal
y = circshift(x, 2);
// Verify the circular time shift property
N = length(x);
X = fft(x);
Y = fft(y);
for k=0:N-1
  phase = exp(-%i*2*%pi*k*2/N);
  disp( X(k+1)*phase ,Y(k+1), );
end
```

Output:

```
15. + 0.i
 15.
 4.045085 - 1.3143278i
 4.045085 - 1.3143278i
-1.545085 - 2.126627i
-1.545085 - 2.126627i
-1.545085 + 2.126627i
 -1.545085 + 2.126627i
```

Circular Frequency Shift Property

```
// Circular Frequency Shift Property
clc
x = [1, 2, 3, 4, 5];
y = (abs(fft(x)));
shift=3;
d1=[0 0 0 0 0]
N = length(x);
for k=0:N-1
phase = exp(-%i*2*%pi*k*shift/N);
d1(k+1)= x(k+1).*phase
end
disp('lhs',abs(fft(d1)))
disp('rhs',y)
```

Output:

```
 "lhs"
  2.6286556   4.253254   15.   4.253254   2.6286556
 "rhs"
  15.   4.253254   2.6286556   2.6286556   4.253254
```

Experiment –6

Aim: Verification of Parseval's Theorem.

Parseval's Theorem

$$\sum_{n=0}^{N-1} x[n] \cdot g^*[n] = \frac{1}{N} \cdot \sum_{k=0}^{N-1} X[k] \cdot G^*[k]$$

```
// Verification of parseval's Theorem
clc
N = 4;
x =[1  %i 1-%i -1+%i]
g = [2  2-%i 3 4*%i]
X = fft(x);
G = fft(g);
lhs=sum(x.*conj(g))
rhs=sum(X.*conj(G))/N
disp('LHS',lhs)
disp('RHS',rhs)
```

Output:

"LHS" : 8. + 3.i

"RHS" : 8. + 3.i

Experiment –7

Aim: Design and implementation of IIR filter (Low pass and High Pass) to meet given specifications.

Theory:

IIR vs. FIR Filters

The main difference between IIR filters and FIR filters is that

- An IIR filter is more compact in that it can usually achieve a prescribed frequency response with a smaller number of coefficients than an FIR filter. A smaller number of filter coefficients imply less storage requirements and faster calculation and a higher throughput. Therefore, generally IIR filters are more efficient in memory and computational requirements than FIR filters.
- However, it must be noted that an FIR filter is always stable, whereas an IIR filter can become unstable (for example if the poles of the IIR filter are outside the unit circle) and care must be taken in design of IIR filters to ensure stability.
- Also, All IIR filter have Non-Linear Phase, while the FIR can be designed to be Linear Phase filter or Non-Linear Phase.

	IIR Filters	FIR Filters
Phase (grp delay)	difficult to control, no particular techniques available	linear phase always possible
Stability	can be unstable, can have limit cycles	always stable, no limit cycles
Order	less	more
History	derived from analog filters	no analog history
Others		polyphase implementation possible can always be made causal

Classical IIR Filters

The classical IIR filters, Butterworth, Chebyshev Types I and II, elliptic, and Bessel, all approximate the ideal "brick wall" filter in different ways.

This toolbox provides functions to create all these types of classical IIR filters in both the analog and digital domains (except Bessel, for which only the analog case is supported), and in lowpass, highpass, bandpass, and bandstop configurations. For most filter types, you can also find the lowest filter order that fits a given filter specification in terms of passband and stopband attenuation, and transition width(s).

Filter Method	**Description**
Analog Prototyping	Using the poles and zeros of a classical lowpass prototype filter
Direct Design	Design digital filter directly in the discrete time-domain by approximating a piecewise linear magnitude response
Generalized Butterworth Design	Design lowpass Butterworth filters with more zeros than poles.
Parametric Modeling	Find a digital filter that approximates a prescribed time or frequency domain response.

Filter Design Functions

Filter Type	Design Function

Bessel (analog only)	`[b,a] = besself(n,Wn,options)`
	`[z,p,k] = besself(n,Wn,options)`
	`[A,B,C,D] = besself(n,Wn,options)`

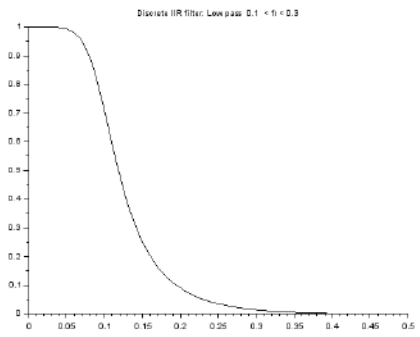
Butterworth	`[b,a] = butter(n,Wn,options)`
	`[z,p,k] = butter(n,Wn,options)`
	`[A,B,C,D] = butter(n,Wn,options)`

Chebyshev Type I	`[b,a] = cheby1(n,Rp,Wn,options)`
	`[z,p,k] = cheby1(n,Rp,Wn,options)`
	`[A,B,C,D] = cheby1(n,Rp,Wn,options)`

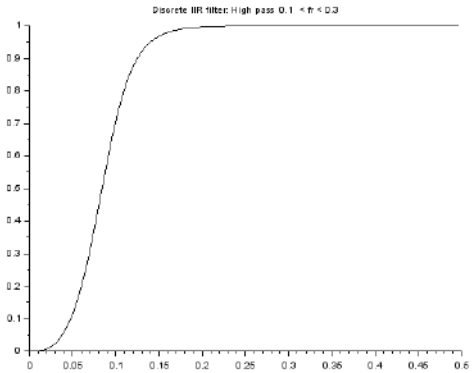
Chebyshev Type II	`[b,a] = cheby2(n,Rs,Wn,options)`
	`[z,p,k] = cheby2(n,Rs,Wn,options)`
	`[A,B,C,D] = cheby2(n,Rs,Wn,options)`

Elliptic	`[b,a] = ellip(n,Rp,Rs,Wn,options)`
	`[z,p,k] = ellip(n,Rp,Rs,Wn,options)`
	`[A,B,C,D] = ellip(n,Rp,Rs,Wn,option`

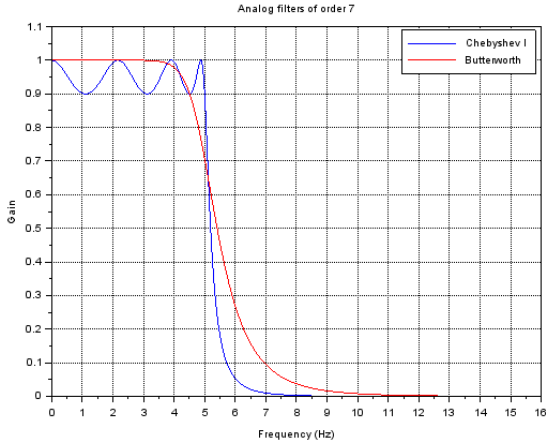
```
// Lowpass Butterworth Filter  (IIR)
clc;
close;
clf;
hz=iir(3,'lp','butt',[.1  .3],[.08 .03]);
[hzm,fr]=frmag(hz,256);
plot2d(fr',hzm')
xtitle('Discrete IIR filter: Low pass  0.1  < fr < 0.3 ',' ',' ');
q=poly(0,'q');     //to express the result in terms of the
delay operator q=z^-1
hzd=horner(hz,1/q)
```



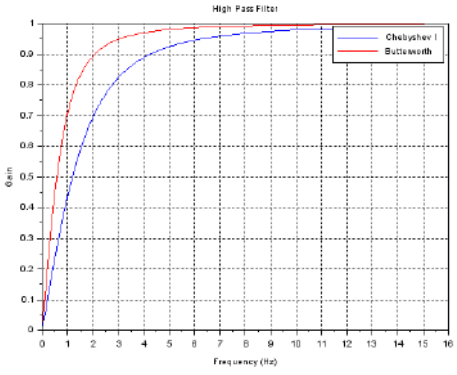
```
// Highpass Butterworth Filter  (IIR)
clc;
close;
clf;
hz=iir(3,'hp','butt',[.1  .3],[.08 .03]);
[hzm,fr]=frmag(hz,256);
plot2d(fr',hzm')
xtitle('Discrete IIR filter: High pass  0.1  < fr < 0.3 ',' ',' ');
q=poly(0,'q');     //to express the result in terms of the delay operator q=z^-1
hzd=horner(hz,1/q)
```




```
// Evaluate magnitude response of the Low Pass FIR filter
fcut = 5; //hz
n = 7; // Filter order
hc1 = analpf(n, 'cheb1', [0.1 0], fcut*2*%pi);
hb = analpf(n, 'butt', [0 0], fcut*2*%pi);
hc1.dt = 'c';
hb.dt = 'c';
clf();
[fr, hf] = repfreq(hc1, 0, 15);
plot(fr, abs(hf), 'b')
[fr, hf] = repfreq(hb, 0, 15);
plot(fr, abs(hf), 'c')

legend(["Chebyshev I","Butterworth"]);
xgrid()
xlabel("Frequency (Hz)")
ylabel("Gain")
title("Analog filters of order 7")
```



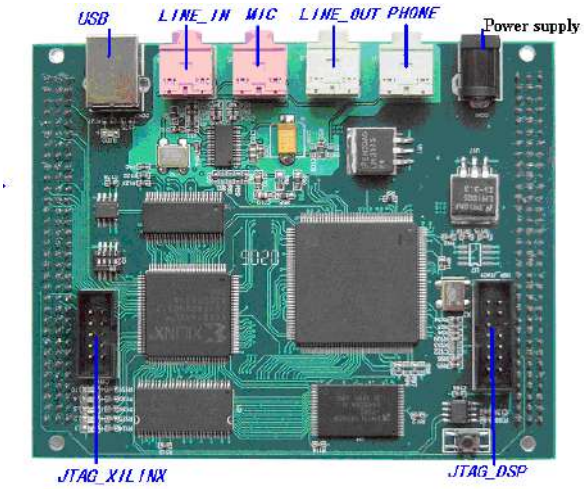

```
// High pass FIR Filter
fcut = 1; //hz
n = 1; // Filter order
hc1 = 1-analpf(n, 'cheb1', [0.1 0], fcut*2*%pi);
hb = 1-analpf(n, 'butt', [0 0], fcut*2*%pi);
hc1.dt = 'c';
hb.dt = 'c';
clf();
[fr, hf] = repfreq(hc1, 0, 15);
plot(fr, abs(hf), 'b')
[fr, hf] = repfreq(hb, 0, 15);
plot(fr, abs(hf), 'r')
legend(["Chebyshev I", "Butterworth"]);
xgrid()
xlabel("Frequency (Hz)")
ylabel("Gain")
title("High Pass Filter")
```



EXPERIMENTS TO BE DONE USING DSP KIT

Introduction of main board:

1. USB2.0 CY7C68013-56PVC, compatible with USB2.0 and USB1.1, including 8051
2. DSP TMS320C6713 TQFP-208 Package Device with, 4 layers board
3. SDRAM MT48LC4M16A2 1meg*16 *4 bank micron
4. FLASH AM29LV800B 8Mbit1Mbyte of AMD
5. RESET chip specialize for reset with button for manually reset
6. POWER power supply externally, special 5V, 3.3V, 1.6V chip for steady voltage with remaining for other devices.
7. EEPROM 24LC64 for download of USB firmware
8. CPLD XC95144XL
9. AIC TLV320AIC23B sampling with 8-96KHZ, 4 channels with interface of headphone.



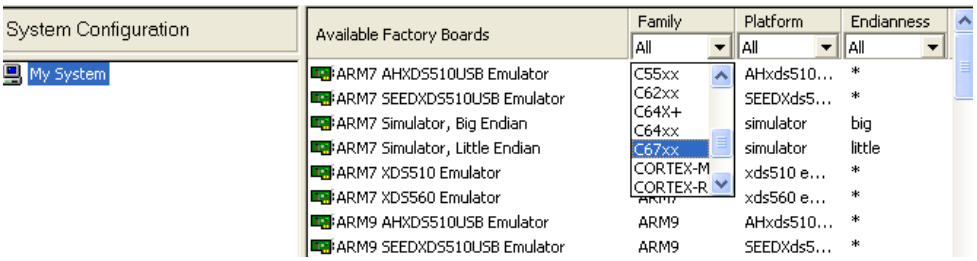
The setup of USB Programmer and emulator in CCS v3.3

1. Install CCS v3.3 software according to the default Custom Install.
2. Install USB Emulator chooses to install directory from CCS v3.3 hat is if CCSv3.3 is installed in C: / CCStudio_v3.3 directory, then install the USB emulator driver in this directory.

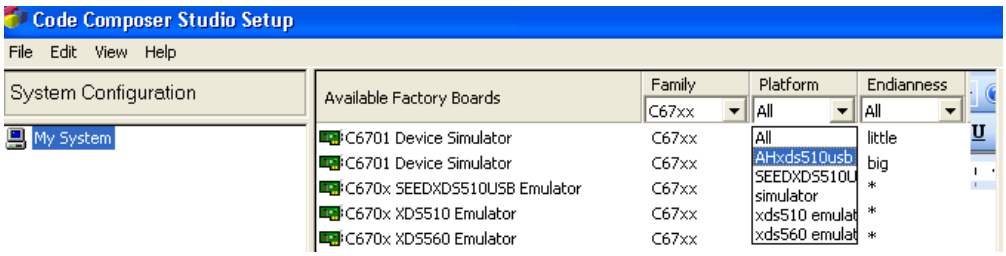
Procedure to Setup Emulator:

1. Open the “Setup CCStudio v3.3”

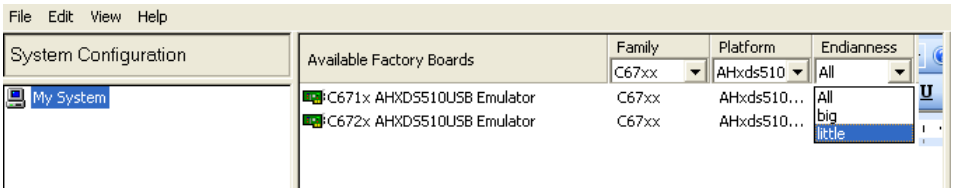
2. Choose c67xx in the “Family”.



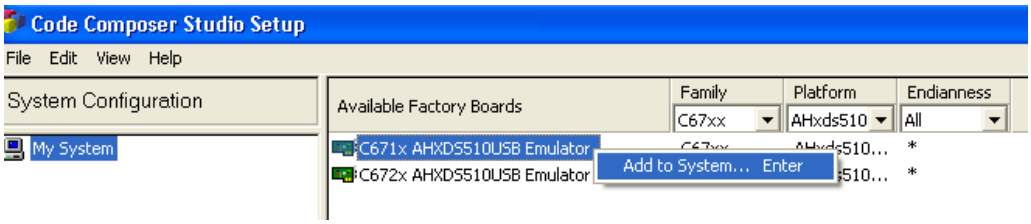
3. Choose AHxds510usb emulator in the “Platform”.



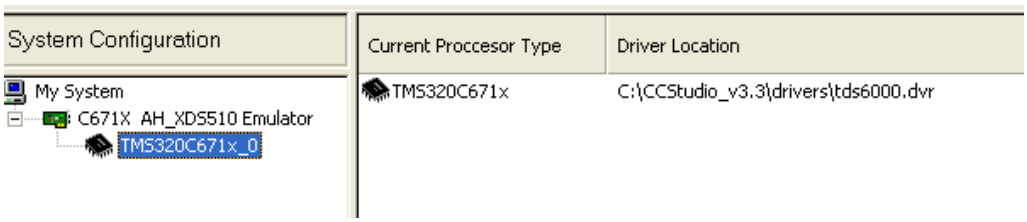
4. Choose little in the “Endianness”.



5. Now you are left with two options under Available Factory Boards, Choose C671X AHXDS510 USB Emulator, right click and “Add to system…”



6 . Now the Emulator and the processor both are selected under “system configuration”.



7. Choose file and click on “save”.

8. Choose file and click on “exit” you will get a wizard as shown bellow

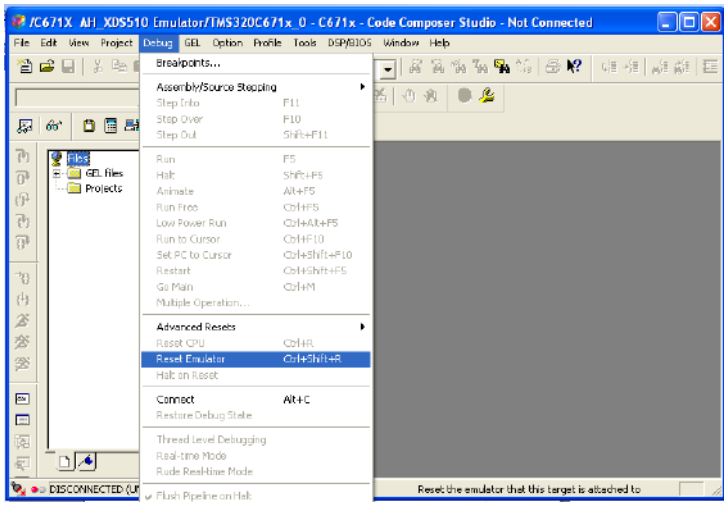
Click on yes.

Then it launches a Code Composer Studio

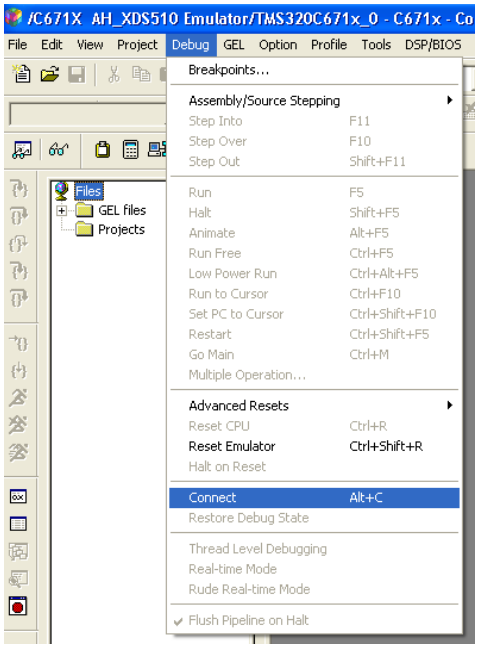
1. Connect the power supply to the board (5V, 3A): make sure that supply is there in the board by pressing reset button.

2. After 2 minutes after power supply is turned on then connects the USB Programmer cable to DSP JTAG connector and Host Computer where CCS 3.3 is installed.

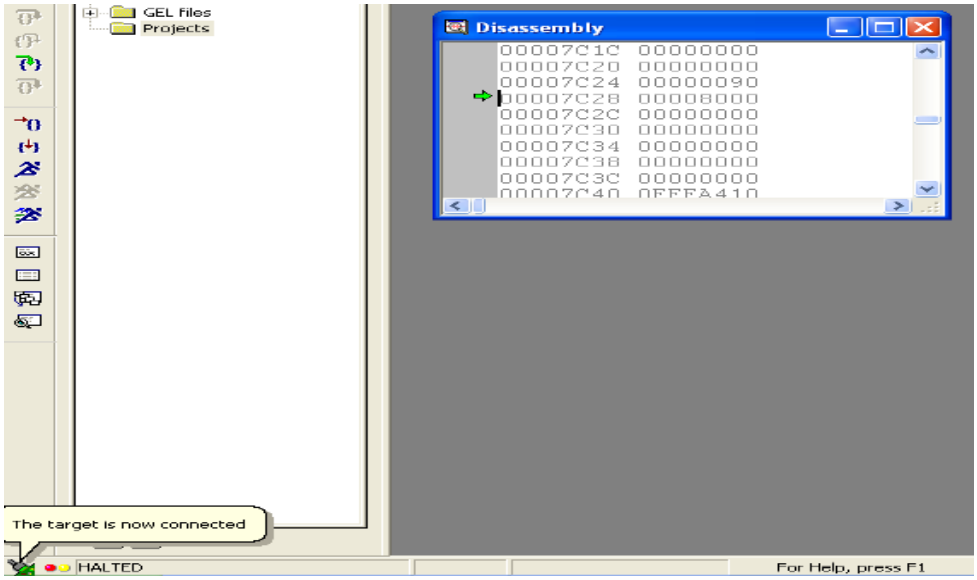
12. Go to Debug and select the Reset Emulator



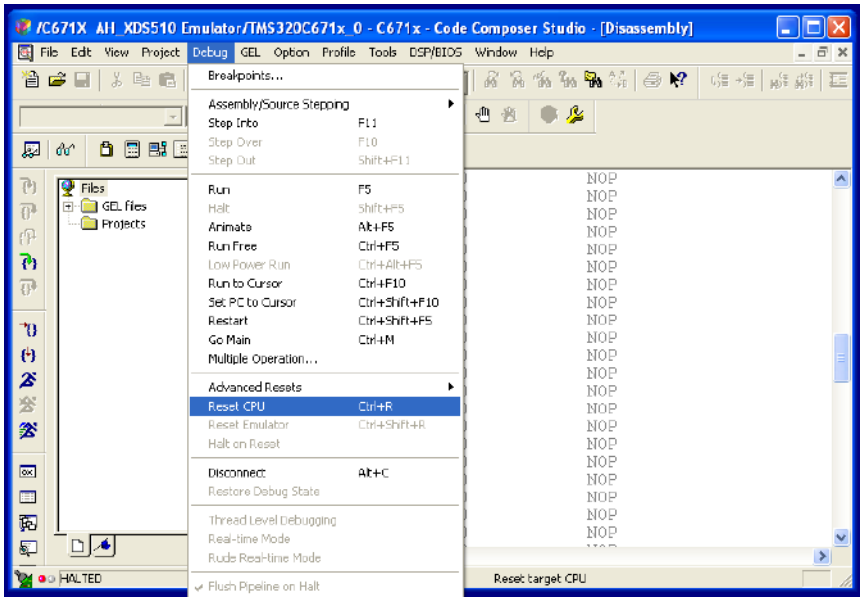
13. Go to Debug and select the option connect by Pressing Reset Button on the board



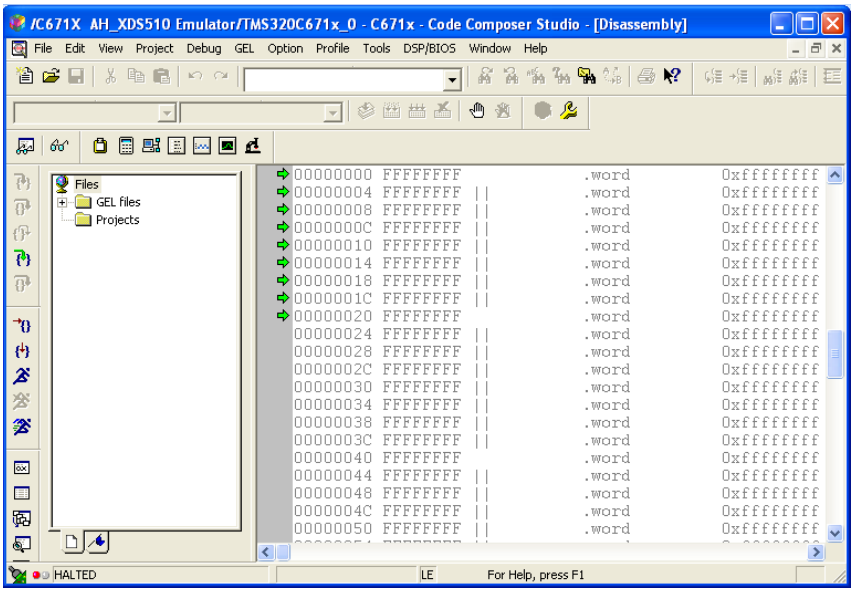
14. Now Target is connected



15 Go to Debug and press Reset CPU that will initialize your memory.



After Pressing Reset the CPU following window should appear in CCS.

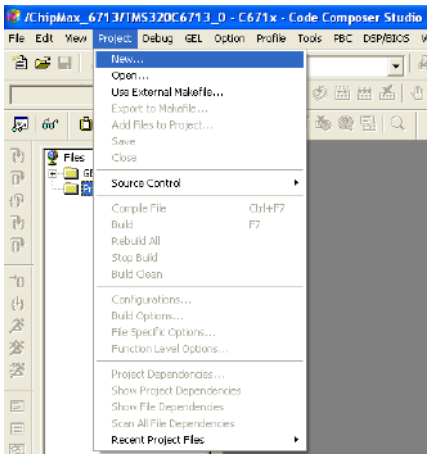


Now Board is ready for working real and non real time programs.

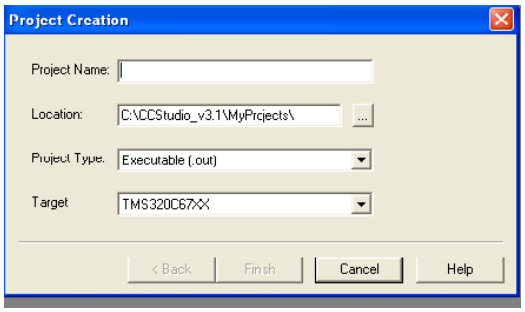
Experiment 1: To compute N- Point DFT of a given sequence using DSK 6713 simulator.

Procedure to create new Project:

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

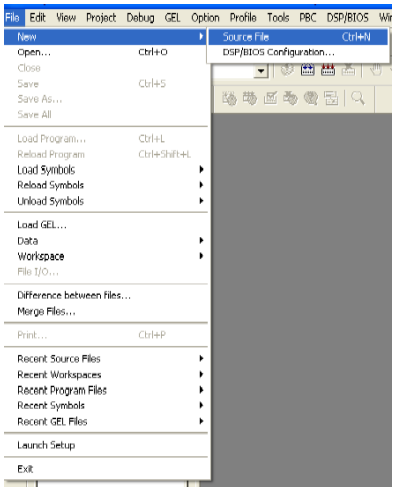


2. Give project name and click on finish.



(**Note:** Location must be c:\CCStudio_v3.3\MyProjects).

3. Click on File →New →Source File, To write the Source Code.

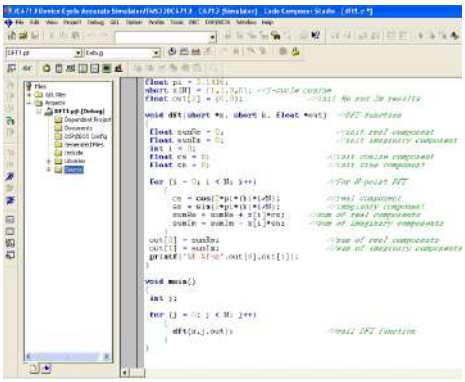


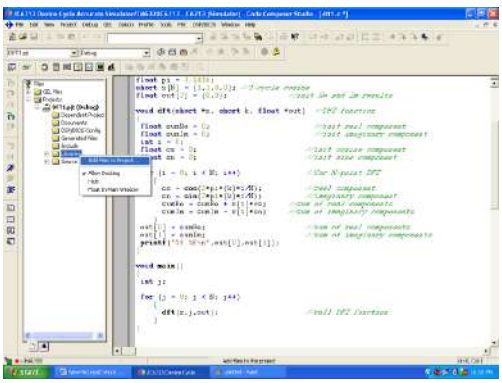
C Code:

```
#include<stdio.h>
#include<math.h>
void main()
{
short N = 4;
short x[4] = {1,2,3,4}; // test data
float pi = 3.1416;
float sumRe = 0, sumIm = 0; // init real/imag components
float cosine = 0, sine = 0; // Initialise cosine/sine components
// Output Real and Imaginary components
float out_real[4] = {0.0}, out_imag[4] = {0.0}; int n = 0, k = 0;
for(k=0 ; k<N ; k++)
{
sumRe = 0;
sumIm = 0;
for (n=0; n<N ; n++)
{
cosine = cos(2*pi*k*n/N);
sine = sin(2*pi*k*n/N);
sumRe = sumRe + x[n] * cosine;
sumIm = sumIm - x[n] * sine;  }
out_real[k] = sumRe;
out_imag[k] = sumIm;
printf("[%d] %f +j %f \n", k, out_real[k], out_imag[k]);
}
}
```

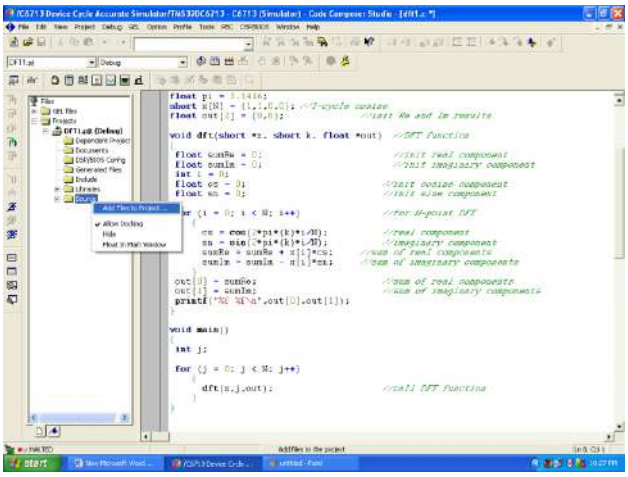
Output : [0] 10.000000 +j 0.000000
[1] -1.999963 +j 2.000022
[2] -2.000000 +j 0.000059
[3] -2.000108 +j -1.999934

1. Enter the source code and save the file with **".C"** extension.

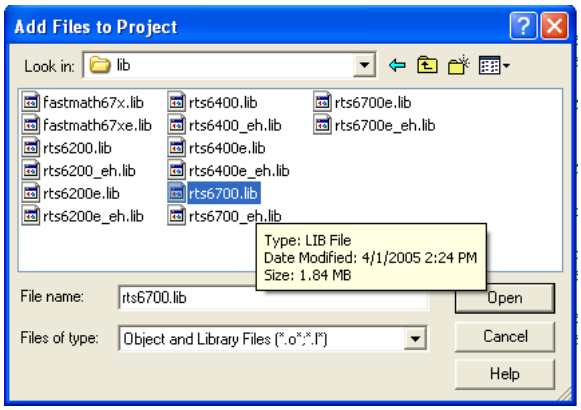




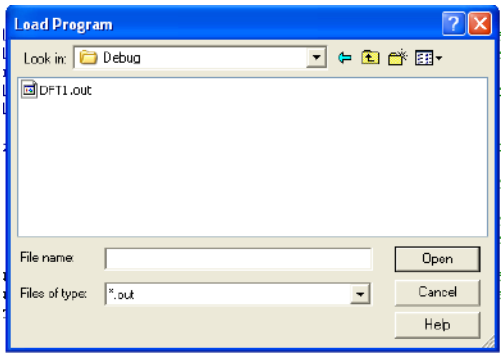
2. Right click on source, Select add files to project .. and Choose "**.C** " file Saved before.



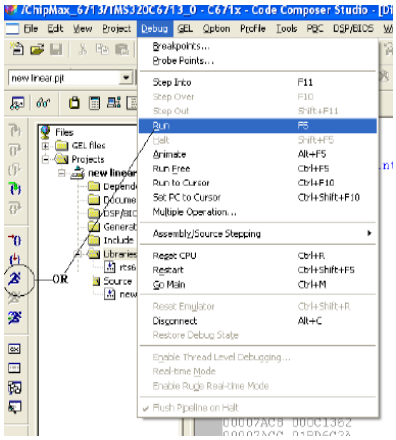
3. Right Click on libraries and select add files to project and choose C:\CCStudio_v3.3\C6000\cgtools\lib\rts6700.lib and click open.



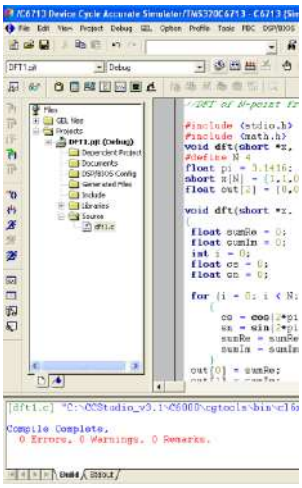
4. Go to file and load program and load **".out"** file into the board..

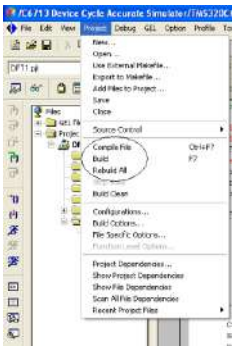


5. Go to Debug and click on run to run the program.

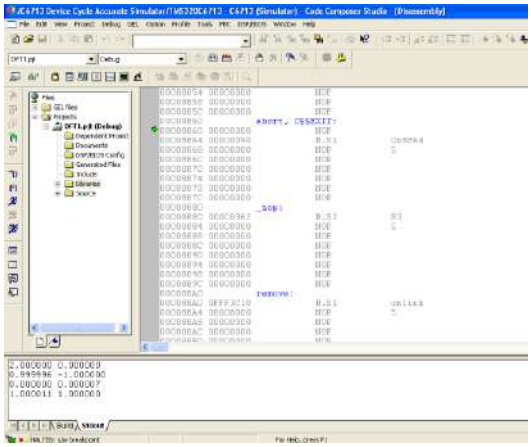


6. a) Go to Project to Compile .
b) Go to Project to Build.
c) Go to Project to Rebuild All.





7. Observe the output in output window.

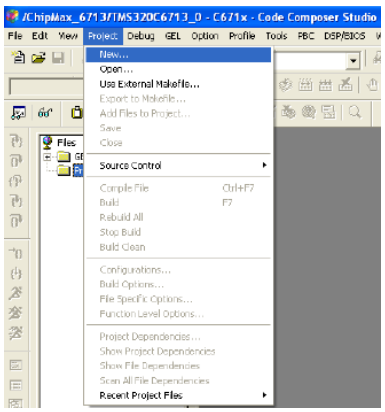


Experiment 2: To compute Linear Convolution of two given sequence using DSK 6713 simulator.

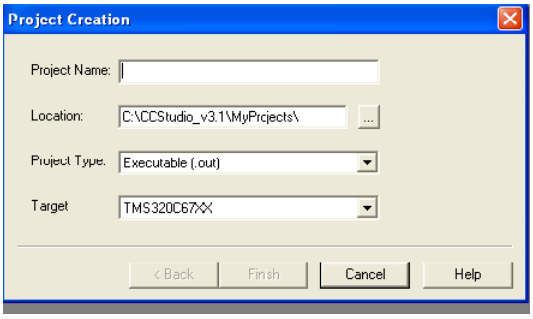
Linear Convolution

Procedure to create new Project:

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

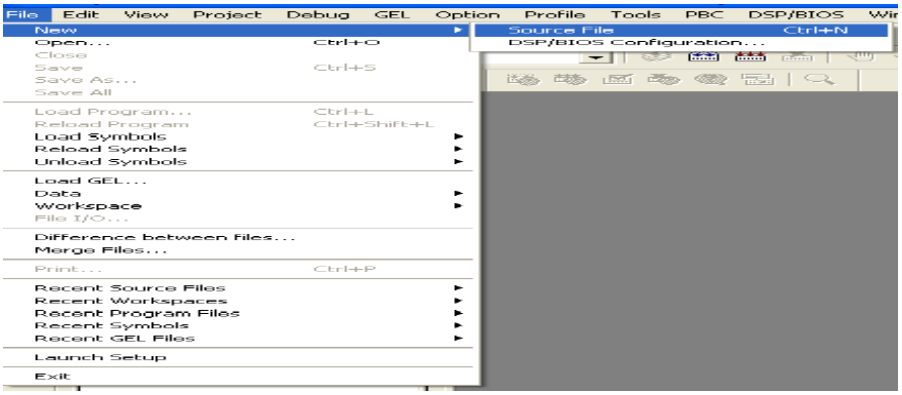


2. Give project name and click on finish.



(Note**:** Location must be c:\CCStudio_v3.3\MyProjects).

3. Click on File -→New Source-→ File, To write the Source Code.



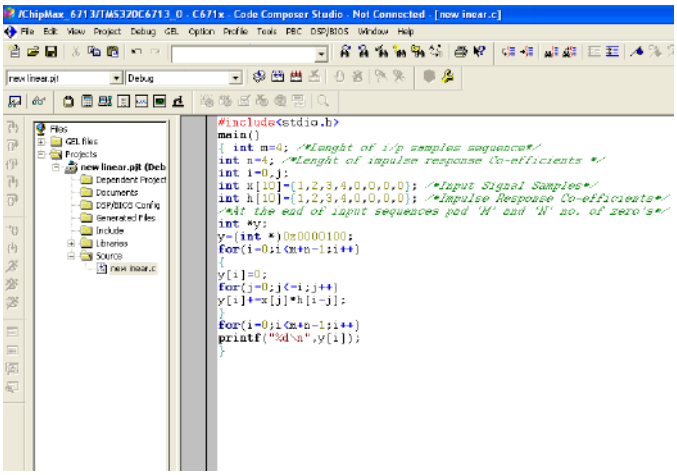
Program:

```
#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[10];
printf("Linear Convolution Result: ");
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\t",y[i]);
}
```

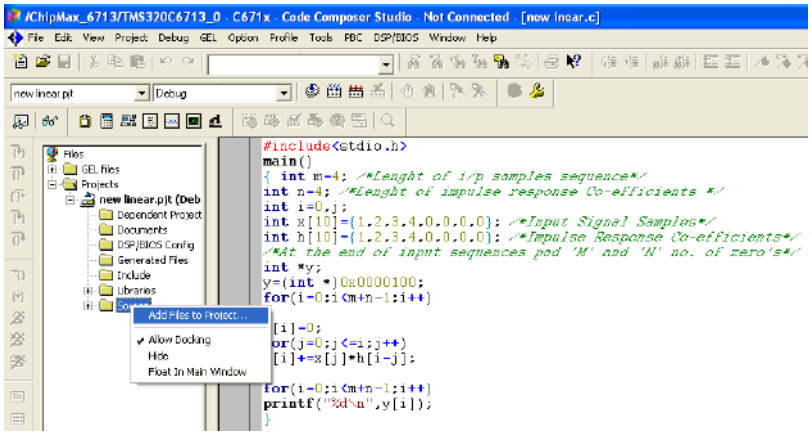
Result:

Linear Convolution Result: 1 4 10 20 25 24 16

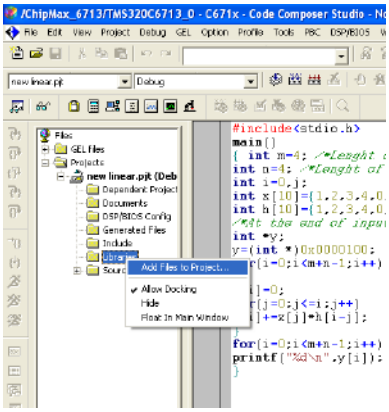
4. Enter the source code and save the file with "**.C**" extension.

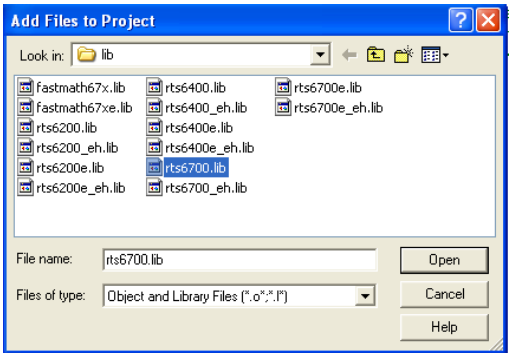


5. Right click on source, Select add files to project .. and Choose "**.C** " file Saved before.

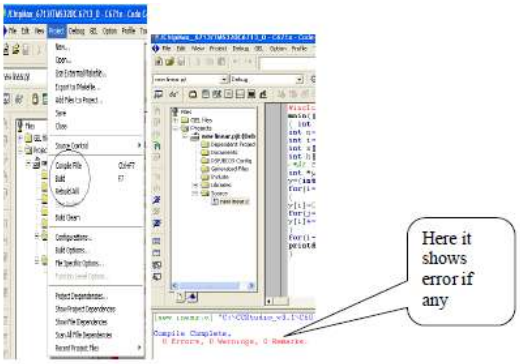


6. Right Click on libraries and select add files to Project.. and choose C:\CCStudio_v3.3\C6000\cgtools\lib\rts6700.lib and click open.

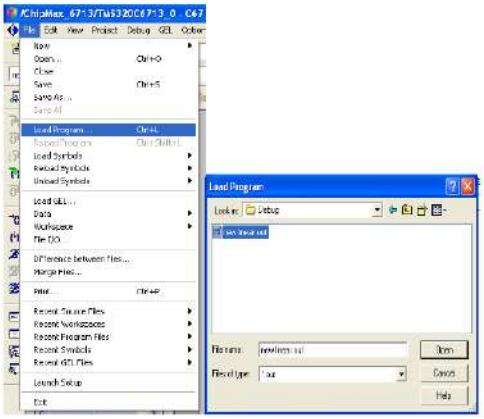




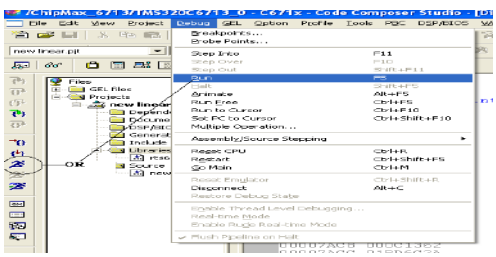
7. a)Go to Project to Compile .
   b) Go to Project to Build.
   c) Go to Project to Rebuild All.



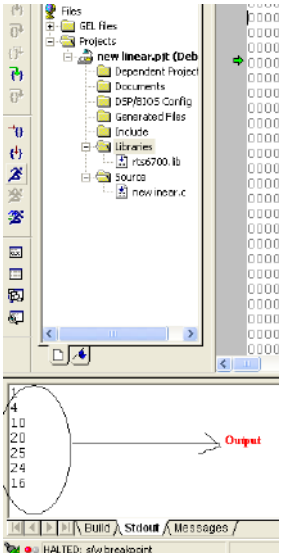
8. Go to file and load program and load **".out"** file into the board.



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



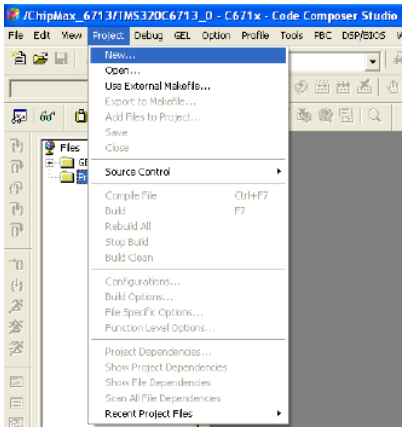
10. Observe the output in output window.



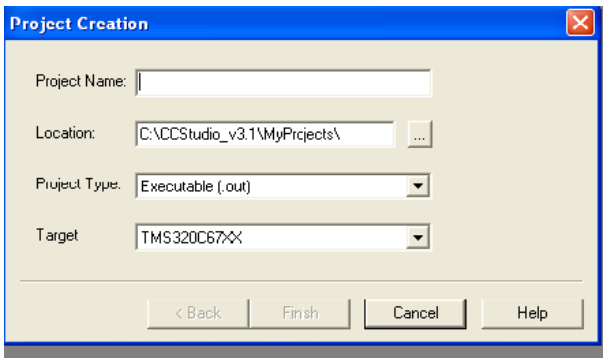
Experiment 3: To compute Circular Convolution of two given sequence using DSK 6713 simulator.

Procedure to create new Project:

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

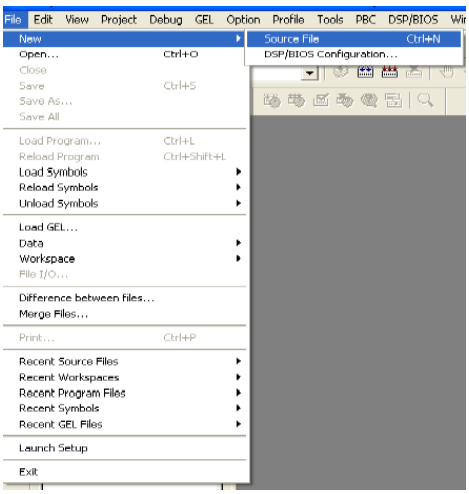


2. Give project name and click on finish.



(**Note:** Location must be c:\CCStudio_v3.3\MyProjects).

3. Click on File →New --> Source File, to write the Source Code.



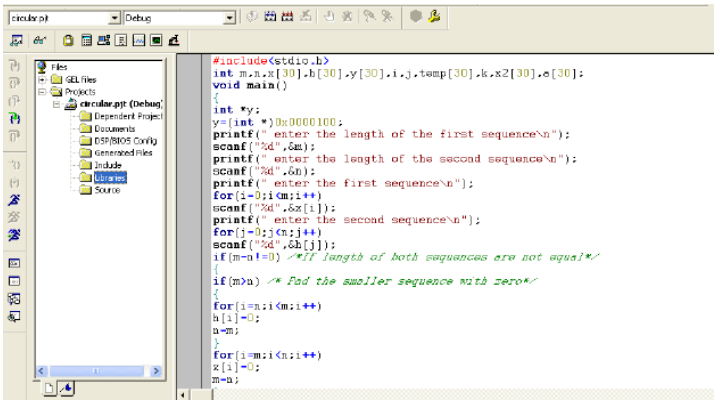
Program:

```
#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 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;
/*circular shift*/
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 \t",y[i]);
}
```

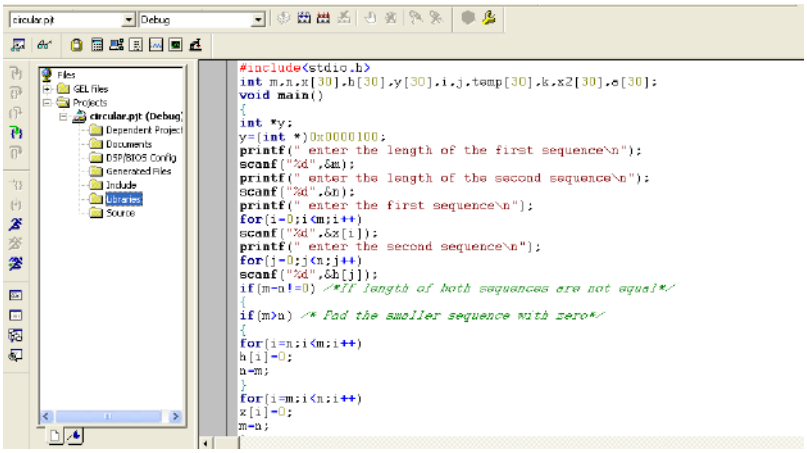
Output:

enter the length of the first sequence
4
enter the length of the second sequence
4
enter the first sequence
4 3 2 1
enter the second sequence
1 1 1 1
the circular convolution is
10 10 10 10

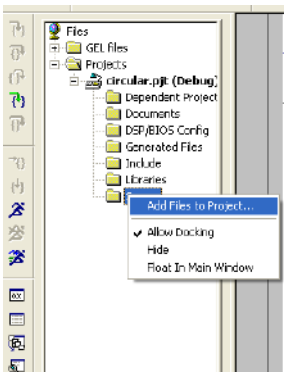
Enter the source code and save the file with "**.C**" extension.

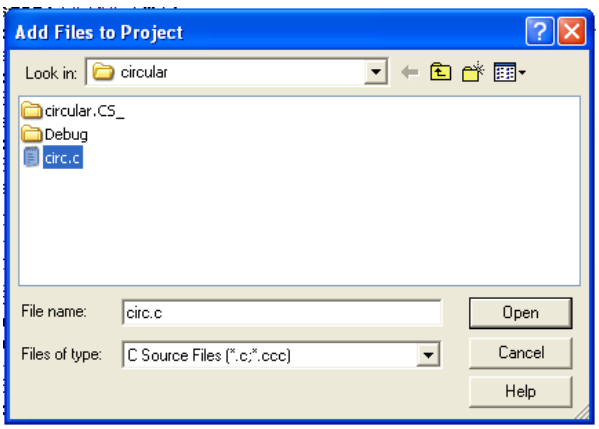


4. Right click on source, Select add files to project .. and Choose "**.C** " file Saved before.

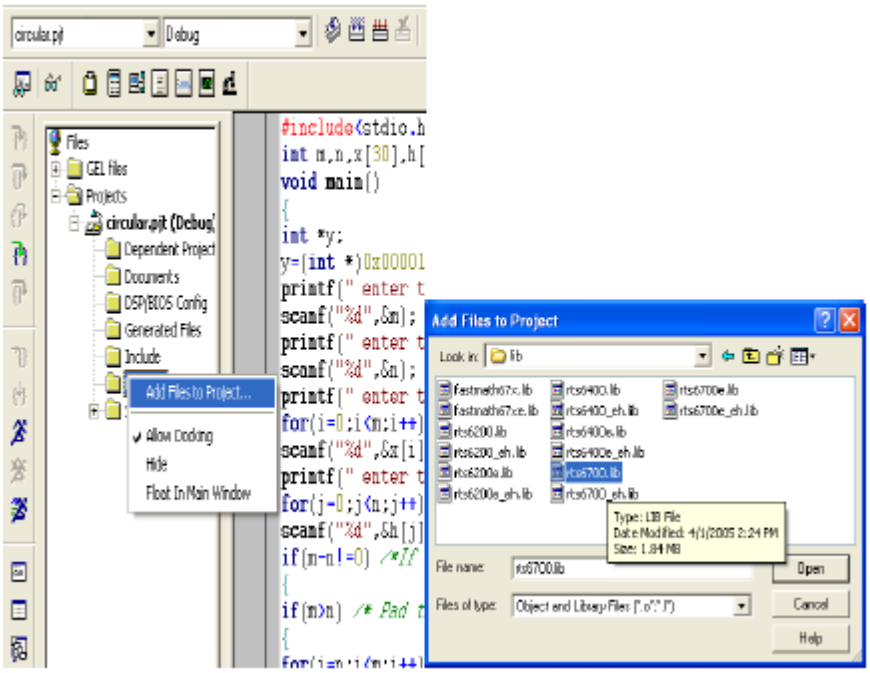


5. Right click on source, Select add files to project .. and Choose "**.C** " file Saved before.

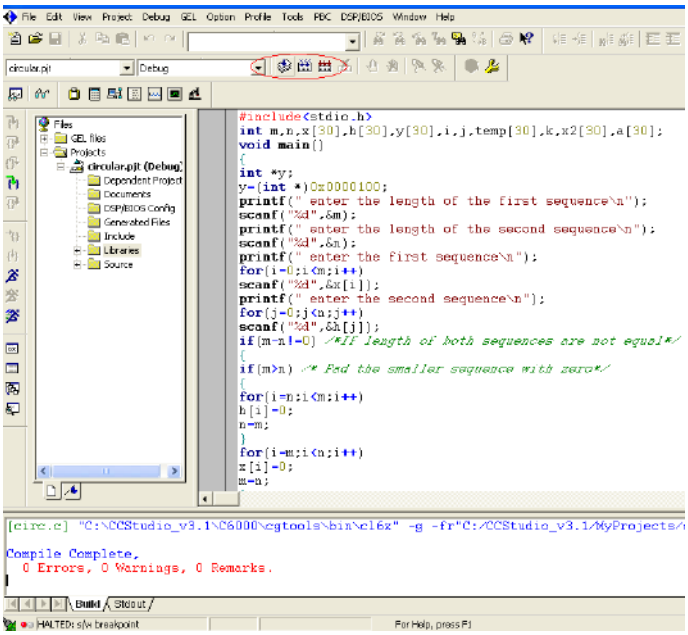




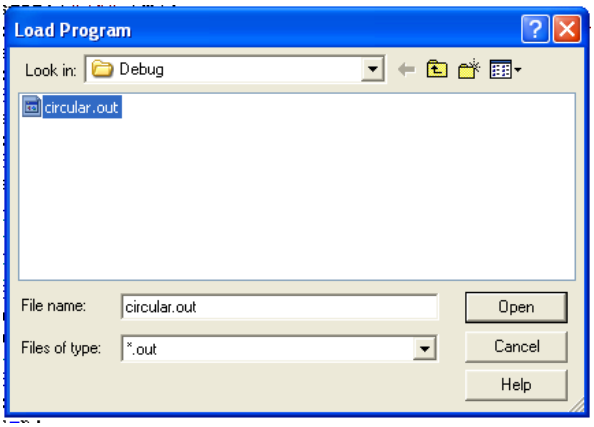
6. Right Click on libraries and select add files to Project.. and choose C:\CCStudio_v3.3\C6000\cgtools\lib\rts6700.lib and click open.



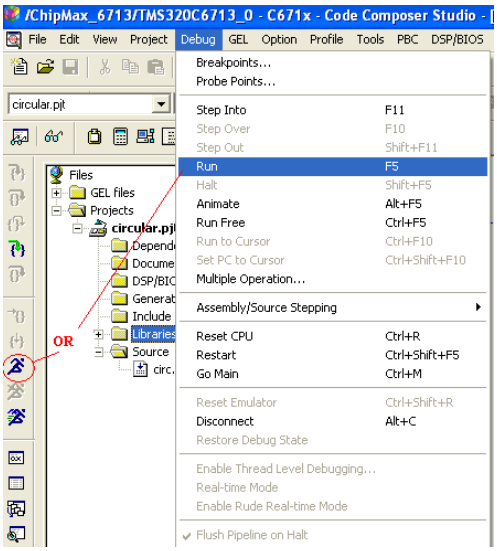
7. a) Go to Project to Compile .
   b) Go to Project to Build.
   c) Go to Project to Rebuild All.



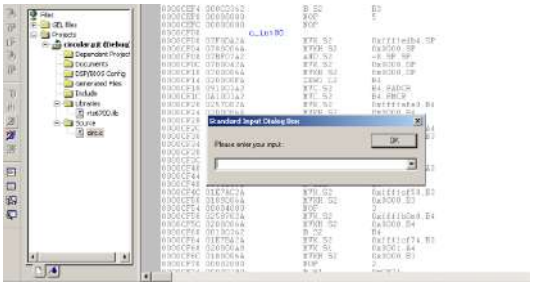
8. Go to file and load program and load **".out"** file into the board.

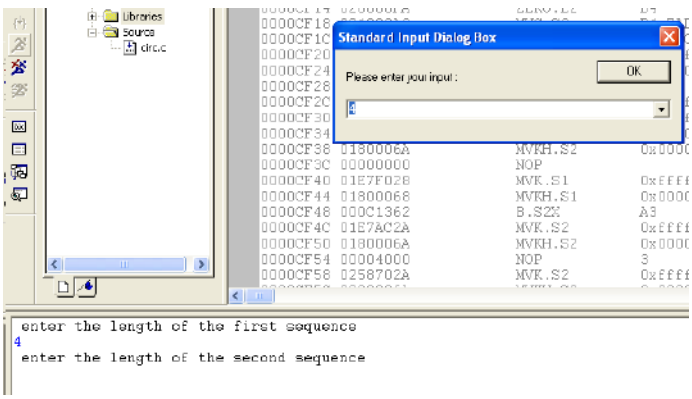


9. Go to Debug and click on run to run the program.

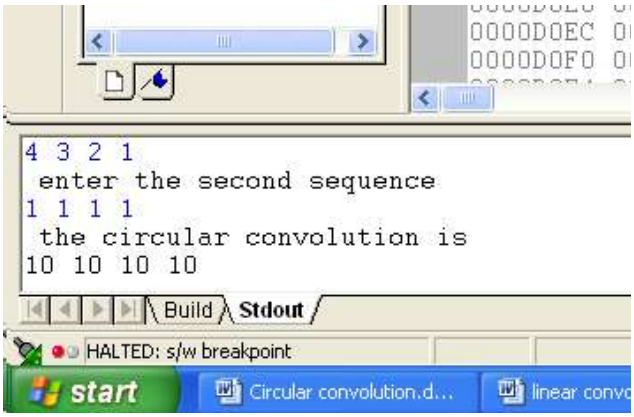


10. Enter the input data to calculate the circular convolution.





The corresponding output will be shown on the output window as shown below



DIGITAL SIGNAL PROCESSING LAB VIVA QUESTIONS

(21EC42)

1) Mention basic blocks of DSP and DSP Applications
2) Mention advantages and disadvantages of DSP
3) Discuss the effect of undersampling for the given signal
4) Discuss the effect of Nyquist sampling for the given signal
5) Discuss the effect of oversampling for the given signal.
6) Define Linear Convolution
7) Define Circular convolution
8) Explain how Circular convolution is performed using Linear Convolution
9) Define Autocorrelation of Signal
10) Define Cross-correlation of two signal
11) Explain properties of Correlation.
12) Obtain the circular convolution of the following sequences $x(n)$ = {1,2,1}; $h(n)$ = {1, -2,2}
13) Define DFT and IDFT
14) How many multiplications and additions are required to compute N –point DFT using radix – 2 FFT?
15) Distinguish between DFT and DTFT?
16) What is zero padding? What are its uses?
17) What is twiddle factor?
18) What is meant by in – place computation?
19) What are the differences and similarities between DIT and DIF.
20) Distinguish between linear convolution and circular convolution?
21) What are the differences between Overlap – add and Overlap – save method?
22) State the properties of DFT?
23) How will you perform linear convolution using circular convolution?
24) Find the circular convolution of $x(n)$ = {1,2,3,4} with $h(n)$ = {1,1,2,2}?
25) State Parseval's relation with respect to DFT?
26) Explain Radix – 2 DIF FFT algorithm. Compare it with DIT – FFT algorithms.
27) Compute the linear convolution of finite duration sequences $h(n)$ = {1,2} and $x(n)$ = {1, 2, -1, 2, 3, -2, -3, -1, 1, 1, 2, -1} by Overlap add method?
28) Perform the linear convolution of the sequence x(n) = {1, -1, 1, -1} and h(n) = {1,2,3,4} using DFT method
29) Compare Butterworth with Chebyshev filters?
30) What is known as pre warping in digital filters?
31) List the properties of Chebyshev filter?

32) Draw the direct form structure of IIR filter?

33) Why do we go for analog approximation to design a digital filter?

34) What is the advantage of direct form II realization when compared to direct form I realization?

35) Why the Butterworth response is called a maximally flat response?

36) . Mention the advantages of cascade realization?

37) Write the properties of Butterworth filter?

38) What is bilinear transformation?

39) What are the advantages and disadvantages of bilinear transformation?

40) Distinguish between recursive realization and non recursive realization?

41) What is meant by impulse invariance method of designing IIR filter?

42) Give the expression for location of poles of normalized Butterworth filter?

43) What are the parameters that can be obtained from Chebyshev filter specification?

44) Explain the procedure for designing analog filters using the Chebyshev approximation.

45) What are advantages and disadvantages of FIR filter?

46) What is the reason that FIR filter is always stable?

47) State the condition for a digital filter to be causal and stable?

48) Compare Hamming window with Kaiser window.

49) What is the principle of designing FIR filter using frequency sampling method?

50) What is window and why it is necessary?