Introduction to MATLAB

The Language of Technical Computing

MATLAB is a high-level language and interactive environment for numerical computation, visualization, and programming. Using MATLAB, you can analyze data, develop algorithms, and create models and applications. The language, tools, and built-in math functions enable you to explore multiple approaches and reach a solution faster than with spreadsheets or traditional programming languages, such as C/C++ or Java®. You can use MATLAB for a range of applications, including signal processing and communications, image and video processing, control systems, test and measurement, computational finance, and computational biology.

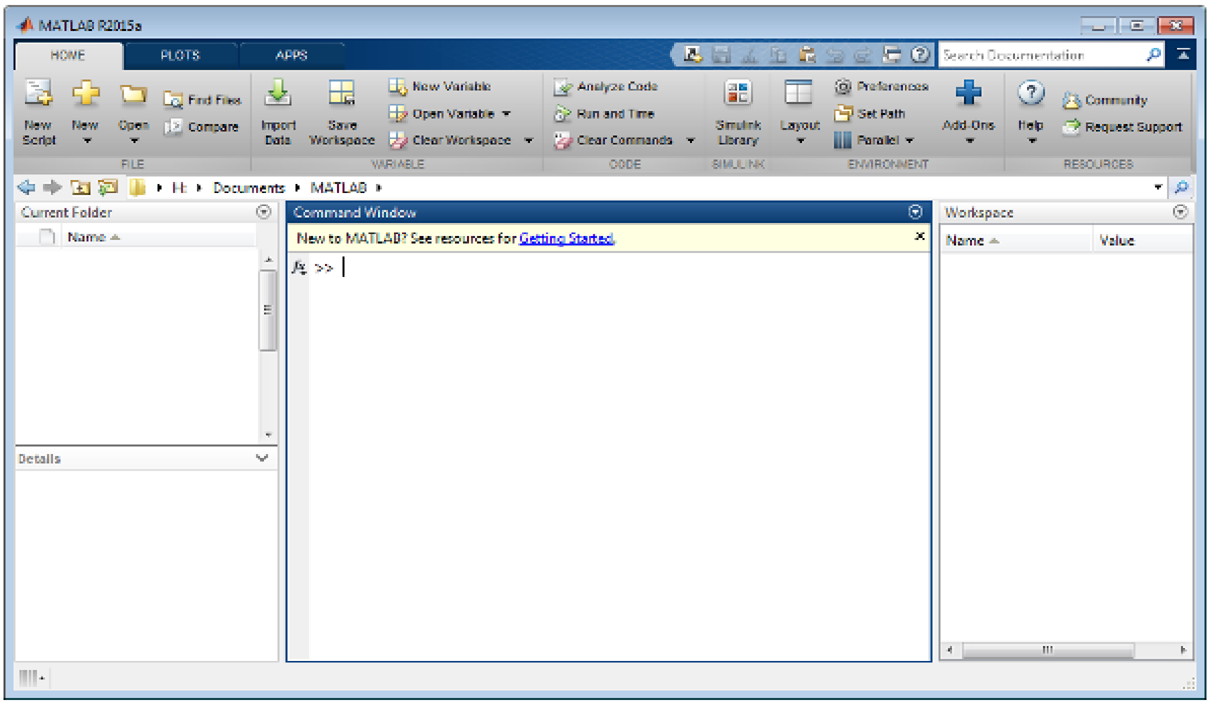
Key Features

• High-level language for numerical computation, visualization, and application development
• Interactive environment for iterative exploration, design, and problem solving
• Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration, and solving ordinary differential equations
• Built-in graphics for visualizing data and tools for creating custom plots
• Development tools for improving code quality and maintainability and maximizing performance
• Tools for building applications with custom graphical interfaces
• Functions for integrating MATLAB based algorithms with external applications and languages such as C, Java, .NET, and Microsoft Excel

MATLAB is an abbreviation for "matrix laboratory."While other programming languages mostly work with numbers one at a time, MATLAB is designed to operate primarily on whole matrices and arrays.

All MATLAB variables are multidimensional arrays, no matter what type of data. A matrix is a two-dimensional array often used for linear algebra.

MATLAB is case-sensitive



New to MATLAB? See resources for Getting Started.

```
>>
```

The desktop includes these panels:

Command Window The window where you type commands and non-graphic output is displayed. A `>>' prompt shows you the system is ready for input. The lower left hand corner of the main window also displays `Ready' or `Busy' when the system is waiting or calculating. Previous commands can be accessed using the up arrow to save typing and reduce errors. Typing a few characters restricts this function to commands beginning with those characters.

Command History Records commands given that session and recent sessions. This window Can be used for reference or to copy and paste commands.

Workspace Shows the all the variables that you have currently defined and some basic information about each one, including its dimensions, minimum, and maximum values. The icons at the top of the window allow you to perform various basic tasks on variables, creating, saving, deleting, plotting, etc. Double-clicking on a variable opens it in the Variable or Array Editor. All the variables that you've defined can be saved from one session to another using File>Save Workspace As (Ctrl-S). The extension for a workspace file is .mat.

Current Directory It is the directory (folder) that MATLAB is currently working in. This is where anything you save will go by default, and it will also influence what files MATLAB can see. You won't be able to run a script that you saved that you saved in a different directory (unless you give the full directory path), but you can run one that's in a sub-directory. The Current Directory bar at the top centre of the main window lets you change directory in the usual fashion The Current Directory window shows a list of all the files in the current directory.

Editor The window where you edit m-files - the files that hold scripts and functions that you've defined or are editing and includes most standard word-processing options and keyboard shortcuts. It can be opened by typing edit in the Command Window. Typing edit myfile will open myfile.m for editing. Multiple files are generally opened as tabs in the same editor window, but they can also be tiled for side by side comparison. Orange warnings and red errors appear as underlining and as bars in the margin. Hovering over them provides more information; clicking on the bar takes you to the relevant bit of text. Also remember that MATLAB runs the last saved version of a file, so you have to save before any changes take effect.

As you work in MATLAB, you issue commands that create variables and call functions MATLAB adds variable to the workspace and displays the result in the Command Window.

EXPERIMENT NO-1

Computation Of N-Point DFT OF GIVEN SEQUENCE

a. **AIM:** To Compute the N point DFT of a given sequence and to plot magnitude and phase spectrum

b. **THEORY:**

Discrete Fourier Transform:

The Discrete Fourier Transform is a powerful computation tool which converts a finite sequence of equally spaced samples of a function into the list of coefficients of a finite combination of complex sinusoids, ordered by their frequencies, that has those same sample values. It can be said to convert the sampled function from its original domain (often time or position along a line) to the frequency domain

The sequence of N complex numbers $x_0, x1, x2..., x_{N-1}$ in time domain is transformed into the sequence of N complex numbers $X_0, X_1..., X_{N\_1}$ of frequency domain by the DFT according to the formula

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{\frac{-j2\pi kn}{N}}$$

k=0,1,2........N-1

The DFT is the most important discrete transform, used to perform Fourier analysis in many practical applications. In digital signal processing, the function is any quantity or signal that varies over time, such as the pressure of a sound wave, a radio signal, or daily temperature readings, sampled over a finite time interval .In image processing, the samples can be the values of pixels along a row or column of a raster image.

The DFT is also used to efficiently solve partial differential equations, and to perform other operations such as convolutions or multiplying large integers.Since it deals with a finite amount of data, it can be implemented in computers by numerical algorithms or even dedicated hardware

Example:

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

We have, $X(k) = \sum_{n=0}^{N-1} x(n) e^{\frac{-j2\pi kn}{N}}$ k=0,1......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) $e^{-j\pi 3/2}$

X(0) = 1+cos(π/2)-j sin(π/2) = 1-j

For k = 2,

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

X(0) = 1+cos(π)-j sin(π) = 1-1 =0

For k = 3,

X(1) = $\sum_{n=0}^{3} x(n)e^{-j3\pi n/2}$ = x(0) + x(1) $e^{-j3\pi/2}$ + x(2) $e^{-j3\pi}$ + x(3) $e^{-j\pi 9/2}$

X(0) = 1+cos(3π/2)-j sin(3π/2) = 1+j

The DFT of the given sequence is,

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

Note: **To find Magnitude of X(k)**

Magnitude= $\sqrt[2]{a^2 + b^2}$ **,** Where a and b are real and imaginary parts of DFT X(k) respectively

To find phase of X(k)

Phase =$\tan^{-1}(\frac{b}{a})$

c. Program:

```
clc;                                          % Clear screen

x1 = input('Enter the sequence:');            %  first sequence

n = input('Enter the length:');               %  the length of DFT

m = abs(fft(x1,n));                           % Compute the DFT and take absolute

disp('N-point DFT of a given sequence:');     % Display the DFT value
disp(m);

N = 0:1:n-1;
```

%magnitude plot

```
subplot(2,2,1);
stem(N,m);
xlabel('Length');
ylabel('Magnitude of X(k)');
title('Magnitude spectrum:');
```

%Phase plot

```
an = angle(fft(x1,n));
disp(an);
subplot(2,2,2);
stem(N, an);
xlabel('Length');
ylabel('Phase of X(k)');
 title('Phase spectrum:');
```

Alternate Method(Formula Implementation)

```
xn=input('enter the seq');                    %  enter first sequence as input
N=input('enter the N_point dft');             %  Enter the number of points of DFT
L=length(xn);                                 %  find the length of input sequence
if N<L
error('N should be greater than L');          %  if N<L generate an error message
end
xn=[xn,zeros(1,N-L)];                         %  zero padding
for k=0:N-1
for n=0:N-1
p=exp((-j*2*pi*n*k)/N);
x2(n+1,k+1)=p;                                %  twiddle factor matrix generation
end
end
xk=xn*x2;                                     %  multiply xn with twiddle factor x2
```

%magnitude plot

```
subplot(2,2,1);
stem(abs(xk));
xlabel('Length');
ylabel('Magnitude of X(k)');
title('Magnitude spectrum:');
```

%Phase plot

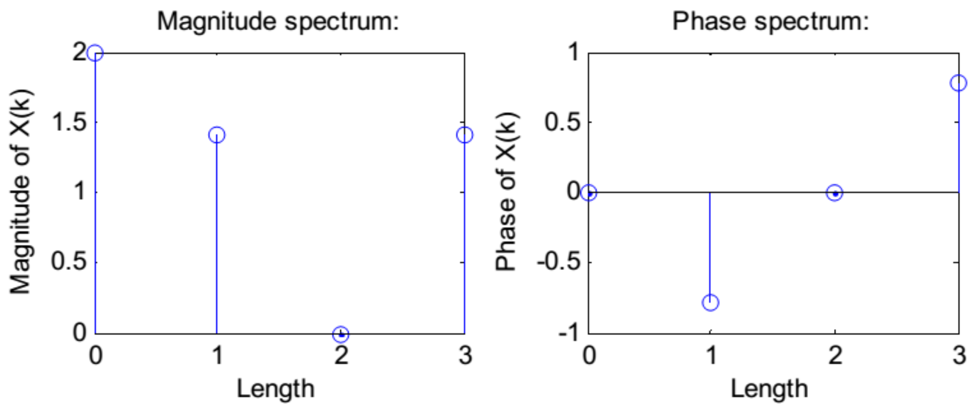
```
an = angle(fft(xk,n));
disp(an);
subplot(2,2,2);
stem(an);
xlabel('Length');
ylabel('Phase of X(k)');
```

```
title('Phase spectrum:')
```

d. **Output:**

```
Enter the sequence: [1 1 0 0]
Enter the length: 4

N-point DFT of a given sequence

[ 2.000     1.4142-0.7854i      0      1.4142+0.7854i ]
```

e. **Graph:**



EXPERIMENT NO-2

Linear convolution and Circular Convolution of two given sequence

2(a) Aim: To determine the Linear convolution of two 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.

If x(n) and y(n) are the two sequences, then convolution z(n) is defined as,

$$y(n)=x(n)*h(n)$$

$$=\sum_{k=0}^{N-1} x(k)h(n-k)$$

Where, N=length[x(n)]+length[h(n)]-1 n=0,1,2 ,3………N-1;

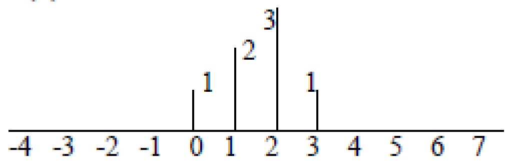
Graphical Interpretation:

• Reflection of h(k) resulting in h(-k)
• Shifting of h(-k) resulting in h(n-k)
• Element wise multiplication of the sequences x(k) and h(n-k)
• Summation of the product sequence x(k) h(n-k) resulting in the convolution value for y(n)

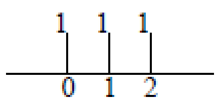
Example:

x(n) = {1, 2, 3, 1}
h(n) = {1, 1, 1}
length(y(n)) = length(x(n)) + length(y(n)) – 1
= 4 + 3 – 1 = 6

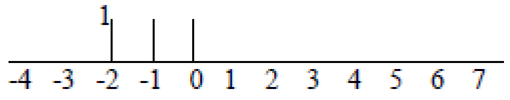
x(k)



h(k)

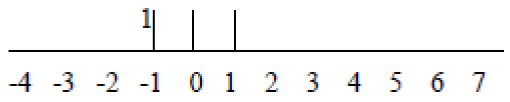


n=0; h(-k)



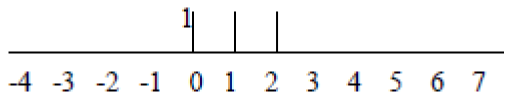
$$y(0) = \sum_{k=-\infty}^{\infty} x(k)\, h(-k) = 1$$

n=1; h(1-k)



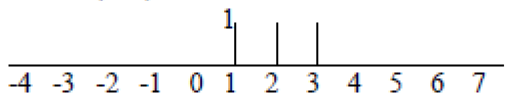
$$y(1) = \sum_{k=-\infty}^{\infty} x(k)\, h(1-k) = 1+2=3$$

n=2; h(2-k)



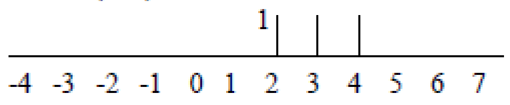
$$y(2) = \sum_{k=-\infty}^{\infty} x(k)\, h(2-k) = 1+2+3=6$$

n=3; h(3-k)



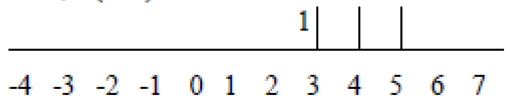
$$y(3) = \sum_{k=-\infty}^{\infty} x(k)\, h(3-k) = 2+3+1=6$$

n=4; h(4-k)

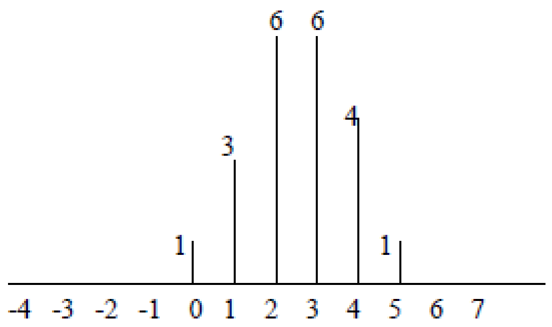


$$y(4) = \sum_{k=-\infty}^{\infty} x(k)\, h(4-k) = 3+1=4$$

n=5; h(5-k)



$$y(5) = \sum_{k=-\infty}^{\infty} x(k)\, h(5-k) = 1$$



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

i. ***Formula implementation***

```
x=input('enter the first sequence');
h=input('enter the second sequence');
n1=length(x);
n2=length(h);
```

```
n=max(n1,n2);
N=n1+n2-1;
x=[x,zeros(1,N-n1)];
h=[h,zeros(1,N-n2)];
for k=1:N
    y(k)=0;
    for i=1:N
        j=k-i+1;
        if (j>0)
            y1(k)=x(i)*h(j);
            y(k)=y(k)+y1(k);
          else
        end
     end
end
stem(y);
disp(y);
```

ii. *Matrix Method*

```
x=input('enter the first sequence');
h=input('enter the second sequence');
n1=length(x);
n2=length(h);
n=max(n1,n2);
N=n1+n2-1;
x=[x,zeros(1,N-n1)];
h=[h,zeros(1,N-n2)];
A=x';
B=h';
temp_seq=A;
for i=1:N-1
    temp_seq=[temp_seq,circshift(A,i)];
end
y=temp_seq*B;
disp(y);
stem(y);
```

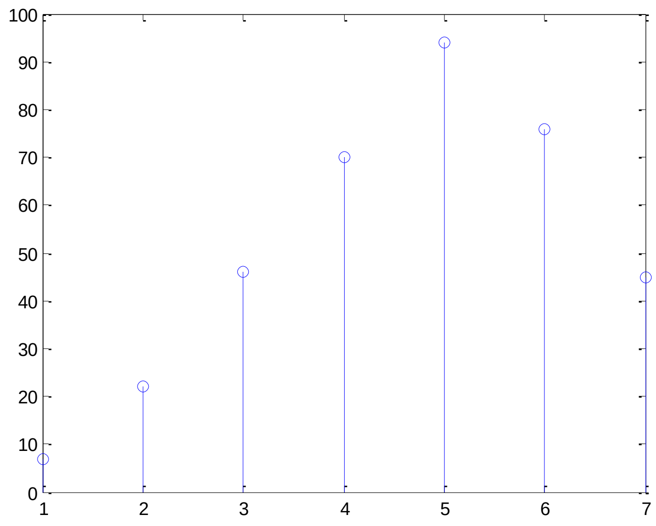
iii. *Using DFT and IDFT*

```
xn=input('enter the first seq');
L1=length(xn);
xm=input('enter the second seq');
L2=length(xm);
N=L1+L2-1;
xn=[xn,zeros(1,N-L1)];
for k=0:N-1
    for n=0:N-1
        p=exp((-j*2*pi*n*k)/N);
        x2(n+1,k+1)=p;
    end
end
```

```
y=xn*x2;
xm=[xm,zeros(1,N-L2)];
for k=0:N-1
    for n=0:N-1
        p=exp((-j*2*pi*n*k)/N);
        x2(n+1,k+1)=p;
    end
end
z=xm*x2;
q=y.*z;
xk=q;
L3=length(xk);
xk=[xk,zeros(1,N-L3)];
for n=0:N-1
    for k=0:N-1
        p=exp((j*2*pi*n*k)/N);
        x2(n+1,k+1)=p;
    end
end
w=(xk*x2)/N;
stem(abs(w));
w
```

OUTPUT:

```
enter the first sequence[1 2 3 4 5]
enter the second sequence[7 8 9]
     7
    22
    46
    70
    94
    76
    45
```



Circular convolution of two given sequences

2(b) Aim: To determine the Circular convolution of two sequences

Theory:

If $x_1(n)$ and $x_2(n)$ are two sequences of length N. Then, the circular convolution $x_3(m)$ is given in the form

$$x_3(m)=\sum_{n=0}^{N-1} x1(n)x2((m-n))\ N$$

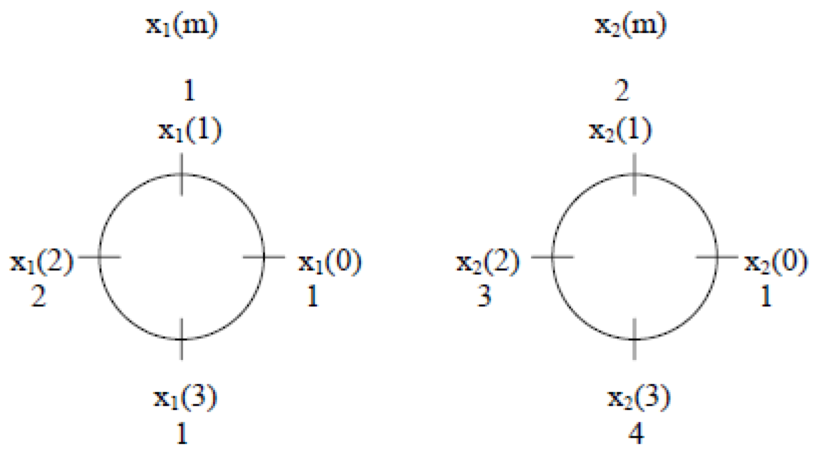
Where, m=0,1,2,3,4…….N-1;

If the given two sequences are of unequal length, the sequences are zero padded to make them equal length.

Example:

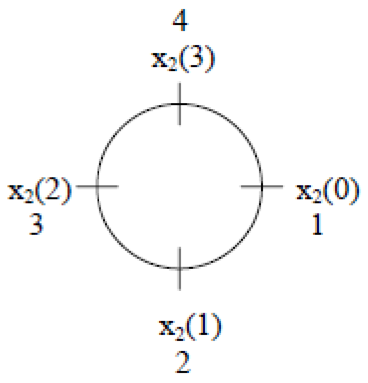
Let's take $x_1(n) = \{1, 1, 2, 1\}$ and
$x_2(n) = \{1, 2, 3, 4\}$

Arrange $x_1(n)$ and $x_2(n)$ in circular fashion as shown below.



To get $x_2(-m)$, rotate $x_2(m)$ by 4 samples in clockwise direction.

$x_2(-m)$

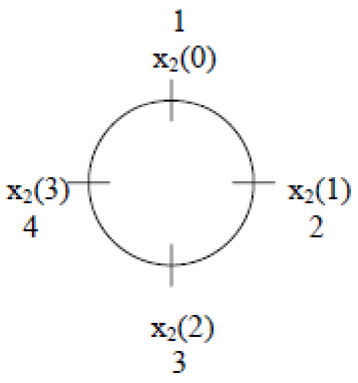


$x_3(0) = x_1(m)\, x_2(-m)$
$= x_1(0)\, x_2(0) + x_1(1)\, x_2(3) + x_1(2)\, x_2(2) + x_1(3)\, x_2(1)$
$= 1 + 4 + 6 + 2$
$x_3(0) = 13$

Keep $x_1(m)$ constant and rotate $x_2(-m)$ once to compute further values

To get $x_3(1)$ rotate $x_2(-m)$ by one sample in anti-clockwise direction

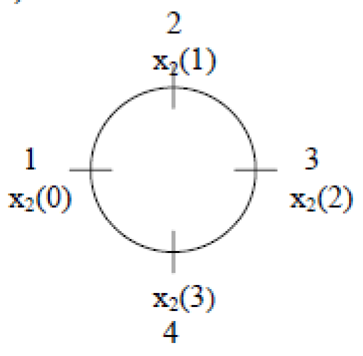
$x_2(1\text{-}m)$



$x_3(1) = x_1(m)\, x_2(1\text{-}m)$
$= x_1(0)\, x_2(1) + x_1(1)\, x_2(0) + x_1(2)\, x_2(3) + x_1(3)\, x_2(2)$
$= 2 + 1 + 8 + 3$
$x_3(1) = 14$

To get $x_3(2)$ rotate $x_2(1\text{-}m)$ by one sample in anti-clockwise direction

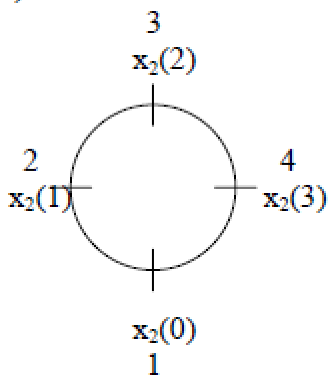
$x_2(2\text{-}m)$



$$\begin{aligned}x_3(2) &= x_1(m)\, x_2(2-m)\\ &= x_1(0)\, x_2(2) + x_1(1)\, x_2(1) + x_1(2)\, x_2(0) + x_1(3)\, x_2(3)\\ &= 3 + 2 + 2 + 4\\ x_3(2) &= 11\end{aligned}$$

To get $x_3(3)$ rotate $x_2(2\text{-}m)$ by one sample in anti-clockwise direction

$x_2(3\text{-}m)$



$$\begin{aligned}x_3(3) &= x_1(m)\, x_2(3-m)\\ &= x_1(0)\, x_2(3) + x_1(1)\, x_2(2) + x_1(2)\, x_2(1) + x_1(3)\, x_2(0)\\ &= 4 + 3 + 4 + 1\\ x_3(3) &= 12\end{aligned}$$

The convoluted signal is,
$x_3(n) = \{13, 14, 11, 12\}$

i. ***Formula implementation***

```
x=input('enter the first sequence');
h=input('enter the second sequence');
n1=length(x);
n2=length(h);
N= max(n1,n2);
x=[x,zeros(1,N-n1)];
h=[h,zeros(1,N-n2)];
for k=1:N
   y(k)=0;
```

```
    for i=1:N
        j=k-i+1;
        if (j>0)
            y1(k)=x(i)*h(j);
            y(k)=y(k)+y1(k);
          else
        end
     end
end
stem(y);
disp(y);
```

ii. *Matrix method*

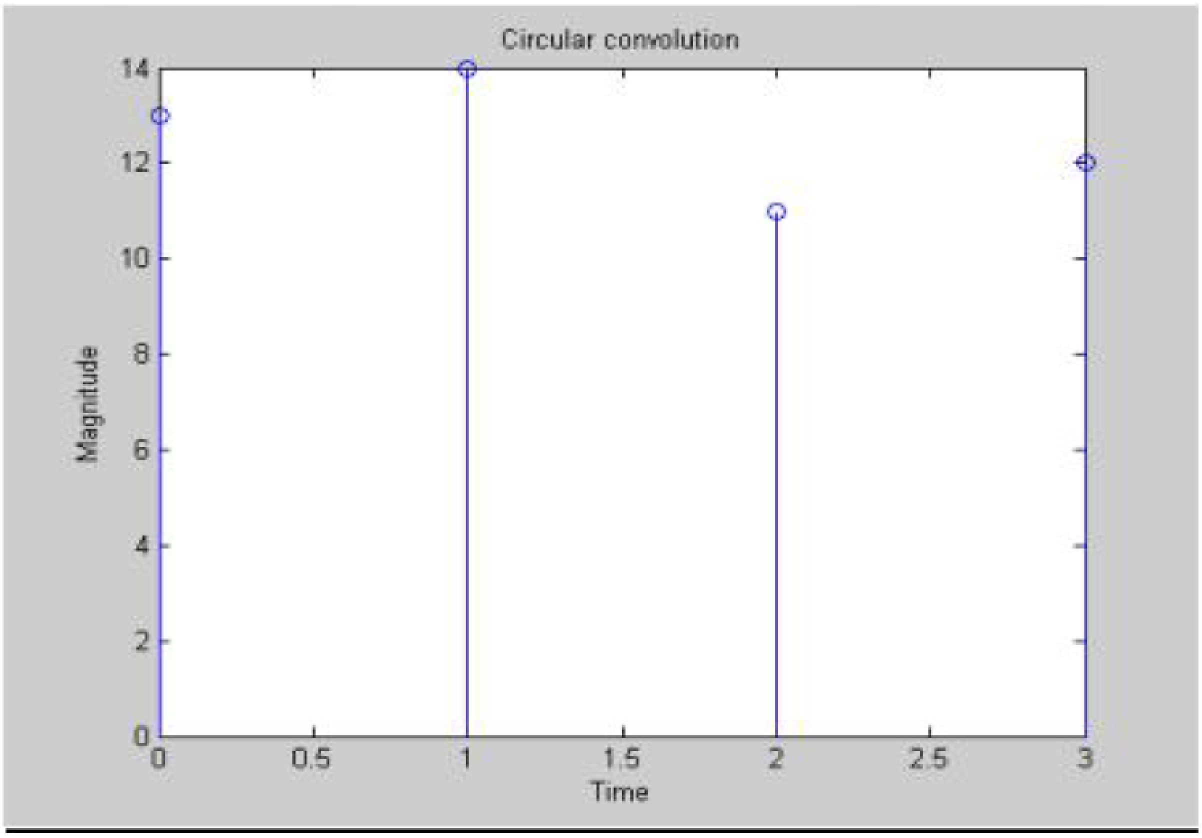
```
x=input('enter the first sequence');
h=input('enter the second sequence');
n1=length(x);
n2=length(h);
n=max(n1,n2);
x=[x,zeros(1,n-n1)];
h=[h,zeros(1,n-n2)];
A=x';
B=h';
temp_seq=A;
for i=1:n-1
temp_seq =[temp_seq,circshift(A,i)];
end
 z=temp_seq*B;
disp(z);
stem(z);
```

iii. *Using DFT and IDFT*

```
xn=input('enter the first seq');
L1=length(xn);
xm=input('enter the second seq');
L2=length(xm);
N=max(L1,L2);
xn=[xn,zeros(1,N-L1)];
for k=0:N-1
    for n=0:N-1
        p=exp((-j*2*pi*n*k)/N);
        x2(n+1,k+1)=p;
    end
end
y=xn*x2;
xm=[xm,zeros(1,N-L2)];
for k=0:N-1
    for n=0:N-1
        p=exp((-j*2*pi*n*k)/N);
        x2(n+1,k+1)=p;
    end
end
z=xm*x2;
q=y.*z;
xk=q;
L3=length(xk);
xk=[xk,zeros(1,N-L3)];
for n=0:N-1
    for k=0:N-1
        p=exp((j*2*pi*n*k)/N);
        x2(n+1,k+1)=p;
    end
end
w=(xk*x2)/N;
stem(abs(w));
w
```

OUTPUT

```
enter the sequence [1 1 2 1]
enter the sequence [1 2 3 4]
z =
13 14 11 12
```



EXPERIMENT NO-3

3(a) AIM: (I) AUTOCORRELATION OF A GIVEN SEQUENCE AND VERIFICATION OF ITS 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. Or equivalently

$$R_{xx}(k) = \sum_{n=-\infty}^{\infty} x(n+k)x(n) \qquad k = 0, \pm1, \pm2, \pm3 \ldots$$

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 = 0:length(Rxx)-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
```

Output:

```
Enter the input sequence: [1 2 3 4] Auto correlated sequence, Rxx:
```

```
4.0000   11.0000   20.0000   30.0000   20.0000   11.0000    4.0000
```

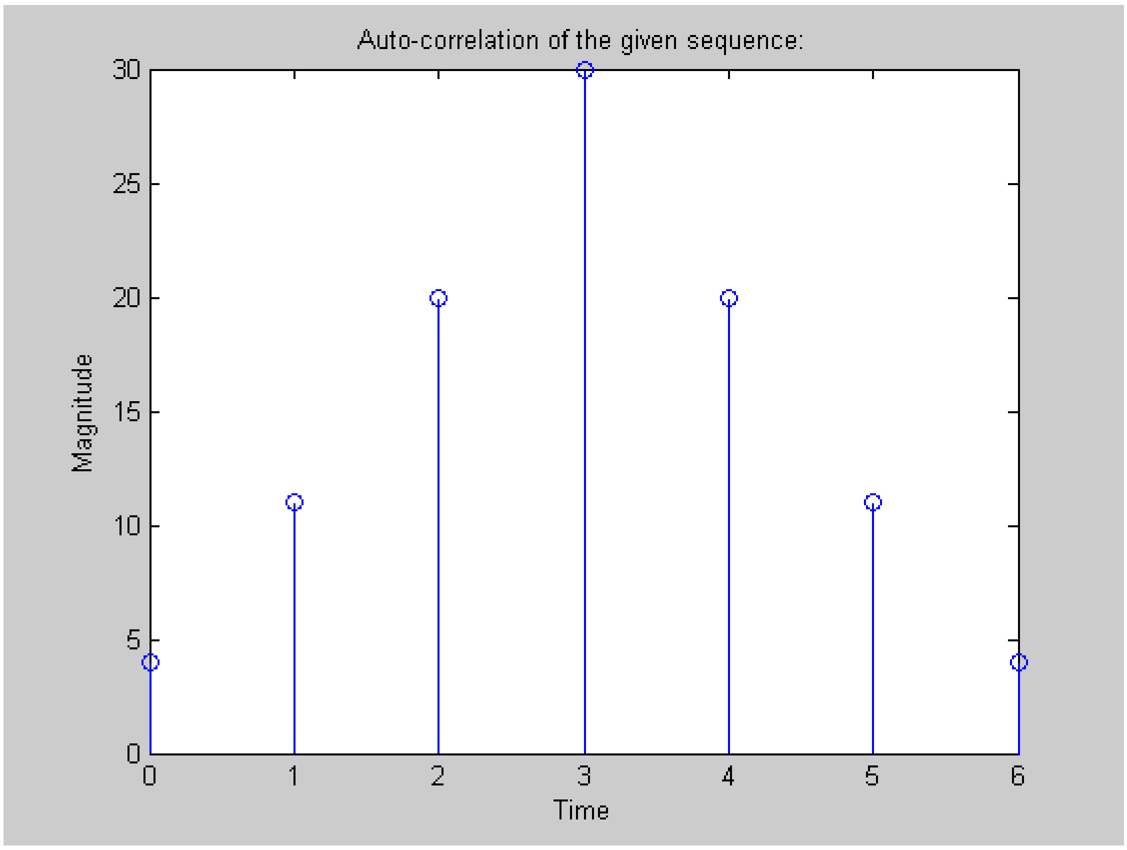
AUTO CORRELATION WITHOIUT USING XCORR:

```
x=input('enter the array'); n=length(x);
a=[zeros(1,n-1),x,zeros(1,n-1)]; b=2*n-1;
for l=1:b
y(l)=0;
for i=1:n
h(l)=x(i)*a(i+l-1);
    y(l)=h(l)+y(l); end
end subplot(2,2,1); stem(x); subplot(2,2,2); stem(y); energy=sum(x.^2); q=length(y);
p=ceil(q/2); if(y(p)==energy)
    disp('energy condition is satisfied'); else
    disp('energy condition is not satisfied'); end
y_left=y(n-1:-1:1);
y_right=y((n+1):1:q);
if (y_left == y_right)
    disp('even sequence');
    else
     disp('not even sequence');
     end
```

OUTPUT:

```
enter the array [ 1 2 3 4]
Auto correlated sequence, Rxx:
4.0000   11.0000   20.0000   30.0000   20.0000   11.0000              4.0000
energy condition is satisfied even sequence
```

Graph:



Properties of Autocorrelation:

1. **Periodicity:** $R_{XX}(k_0) = R_{XX}(0)$ then $R_{XX}(k)$ is periodic with period k_0
2. Autocorrelation function is **symmetric.** i.e. $R_{XX}(m) = R_{XX}(-m)$
3. **Mean square value:** autocorrelation function at k=0, is equal to mean square value of the process. $R_{XX}(0) = E\{|x(n)|^2\} \geq 0$

3(b) AIM: (II) CROSS-CORRELATION OF A GIVEN SEQUENCE AND VERIFICATION OF ITS 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.

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

$R_{xy}(k) = \sum_{n=-\infty}^{\infty} x(n)y(n-k)$ $\quad k = 0, \pm1, \pm2, \pm3 \dots$

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,

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

$R_{xy}(k) = x(k) * y(-k)$ … (1)

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

Program:

```
clc;                                        % Clear screen

x = input('Enter the first sequence:');     % Get the first sequence

y = input('Enter the second sequence:');    % Get the second sequence

Rxy = xcorr(x,y);                           % Returns cross correlated sequence disp('Cross
correlated sequence, Rxy:');                % Display the result

disp(Rxy);                                  % Display the result on command window

t = 0:length(Rxy)-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:'); % Title
```

Output:

```
Enter the first sequence: [1 1 1 1]
```

Enter the second sequence: [1 2 3 4] Cross correlated sequence, Rxy:

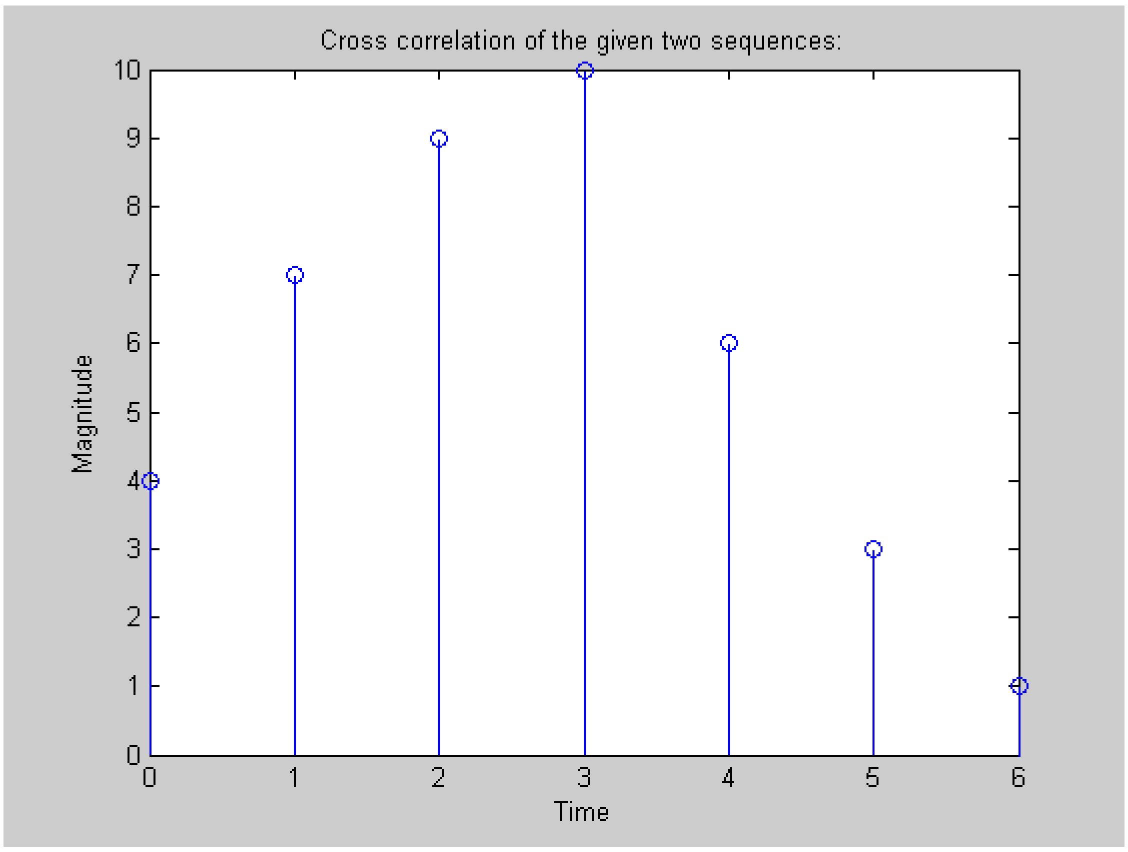
```
4.0000    7.0000    9.0000   10.0000    6.0000    3.0000    1.0000
```

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. $|R_{xy}(-k)| \leq[1/2]\ [R_{xy}(-k) + R_{xy}(-k)]$
5. When $R_{xy}(k) = 0$, $x(n)$ and $y(n)$ are said to be uncorrelated or they said to be statistically independent.

$R_{xy}(k)$ may not be necessarily have a maximum at $k=0$ nor $R_{xy}(k)$ an even function

Graph:



EXPERIMENT NO-4

Plot the spectrum of voice, ECG, EMG and Music.

Aim: To Plot the Spectrum of Practical signal.

Theory: Frequency-domain analysis is a tool of utmost importance in signal processing applications. Frequency-domain analysis is widely used in such areas as communications, geology, remote sensing, and image processing. While time-domain analysis shows how a signal changes over time, frequency-domain analysis shows how the signal's energy is distributed over a range of frequencies. A frequency-domain representation also includes information on the phase shift that must be applied to each frequency component in order to recover the original time signal with a combination of all the individual frequency components.

A signal can be converted between the time and frequency domains with a pair of mathematical operators called a transform. An example is the Fourier transform, which decomposes a function into the sum of a (potentially infinite) number of sine wave frequency components. The 'spectrum' of frequency components is the frequency domain representation of the signal. The inverse Fourier transform converts the frequency domain function back to a time function. The fft and ifft functions in MATLAB allow you to compute the Discrete Fourier transform (DFT) of a signal and the inverse of this transform respectively.

Algorithm

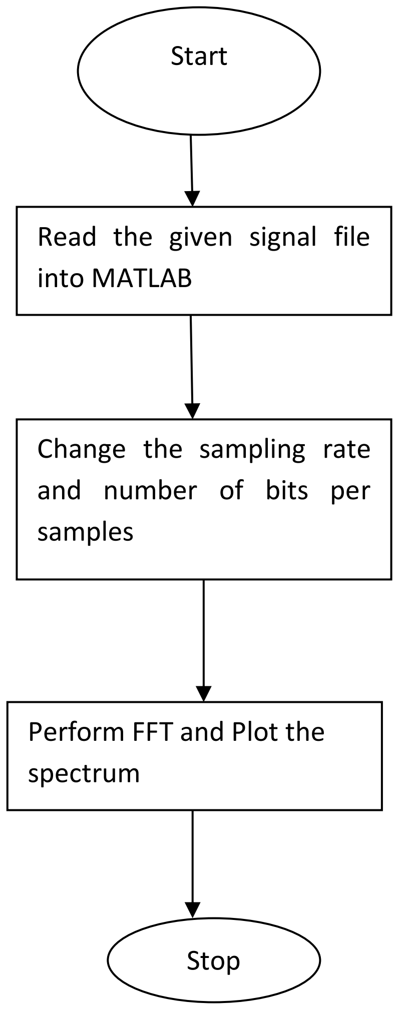
1 Read the data from stored file

2 Store it in a array of 16,32 and 64

3 Call FFT function with array as parameter

4 Plot the Output

Flow chart of Algorithm

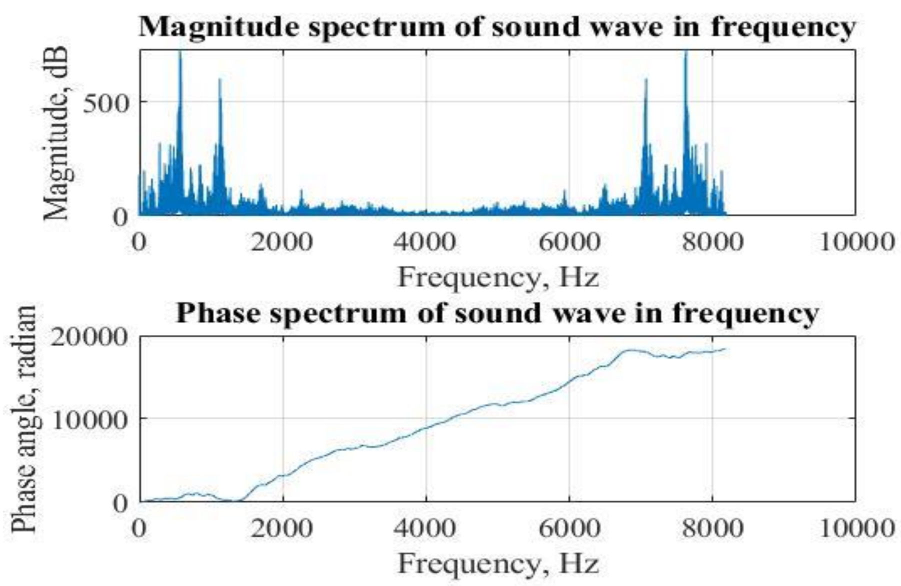

Start
Read the given signal file into MATLAB
Change the sampling rate and number of bits per samples
Perform FFT and Plot the spectrum
Stop

Example Program:

Reading an audio file and Plotting its magnitude and Phase spectrum in MATLAB

```
clear, clc, close all;
load handel.mat; % Load a stored matrix set from matlab
filename = 'handel.wav';
audiowrite(filename,y,Fs); % write the content of matrix into an audio file
[y,fs] = audioread('handel.wav');
N = length(y);                  % Length of vector y, number of samples
Y = fft(y,N);                   % Fourier transform of y
F = ((0:1/N:1-1/N)*fs);         % Frequency vector
w = 2*pi*F;                     % Angular frequency vector
mag_Y = abs(Y);            % Magnitude of the FFT
phaseY = unwrap(angle(Y));  % Phase of the FFT
figure (1);
subplot(2,1,1);
plot(F, mag_Y);
grid on;
set(gca, 'FontName', 'Times New Roman', 'FontSize', 14);
xlabel('Frequency, Hz');
ylabel('Magnitude, dB');
title('Magnitude spectrum of sound wave in frequency');
subplot(2,1,2);
plot (F, phaseY);
grid on;
set(gca, 'FontName', 'Times New Roman', 'FontSize', 14);
xlabel('Frequency, Hz');
ylabel('Phase angle, radian');
title('Phase spectrum of sound wave in frequency');
```

OUTPUT:



EXPERIMENT NO-5

Adding Noise to an audio signal and removing the same using Filter

Aim: To add noise of specific characteristics to audio signal and denoising it using filter

Theory: Noise is unwanted sound considered unpleasant, loud or disruptive to hearing.

Algorithm

1 Read the audio file into Matlab

2 Analyse the characteristics of audio file

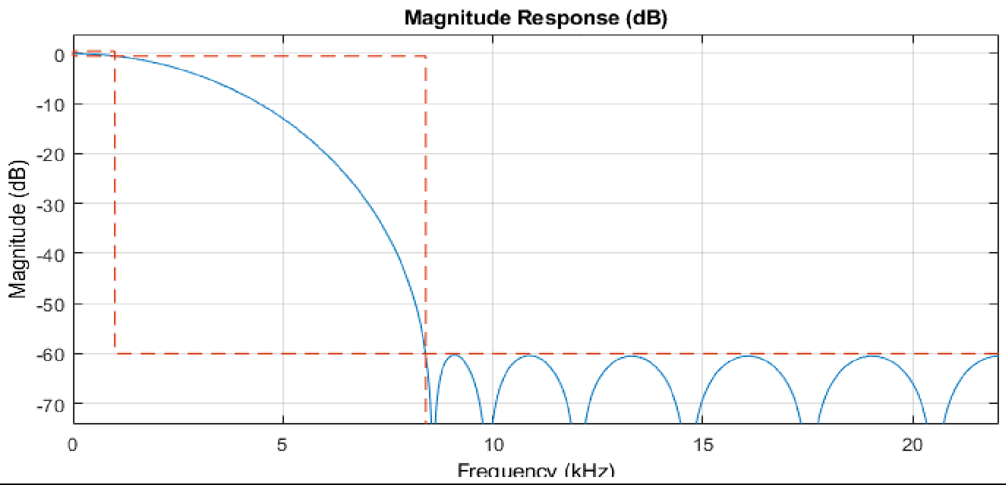
3 Add Noise of specific characteristics to the audio signal

4 Remove noise using Filtering technique

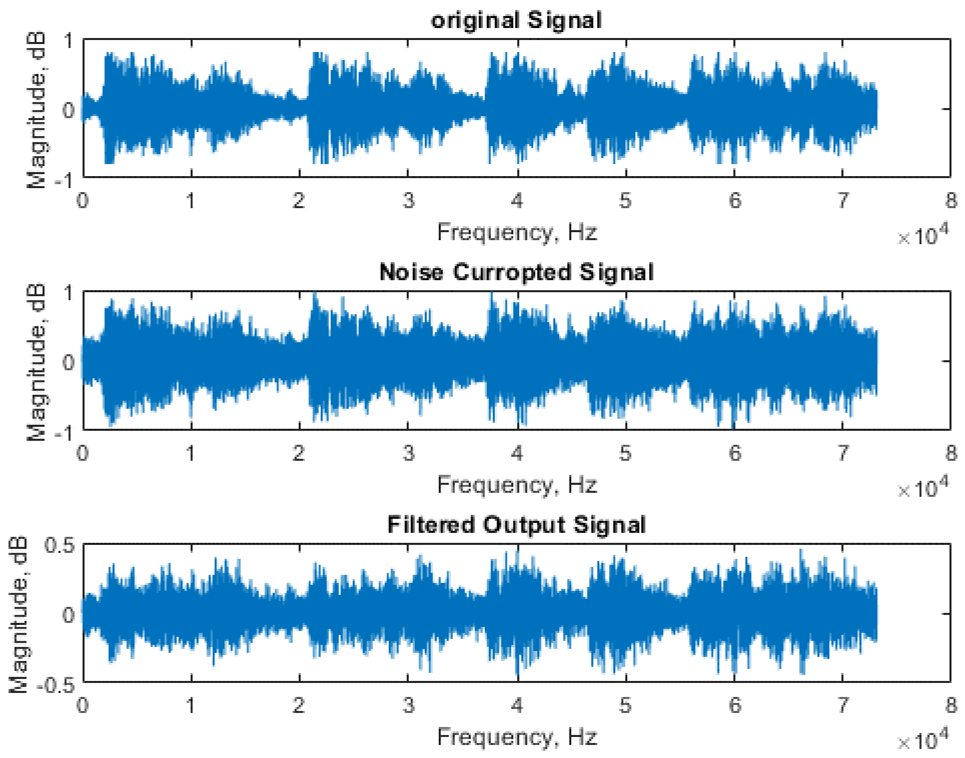
5 Plot the signals

```
clear, clc, close all;
load handel.mat; % Load a stored matrix set from matlab
filename = 'handel.wav';
audiowrite(filename,y,Fs); % write the content of matrix into an audio file

[y,fs] = audioread('handel.wav');

%Filter specifications
Fs=44100;
Fp=1000;
Fst= 8403;
Ap=1;
Ast=95;

df = designfilt('lowpassfir', 'PassbandFrequency', Fp, ...
                'StopbandFrequency', Fst, 'PassbandRipple', Ap, ...
                'StopbandAttenuation', Ast, 'SampleRate', 44100);
fvtool(df);
xn=awgn(y,6,'measured');
yout=filter(df,xn);
subplot(3,1,1)
plot(y);
xlabel('Frequency, Hz');
ylabel('Magnitude, dB');
title('original Signal');
subplot(3,1,2)
plot(xn);
xlabel('Frequency, Hz');
ylabel('Magnitude, dB');
title('Noise Curropted Signal');
subplot(3,1,3)
plot(yout );
xlabel('Frequency, Hz');
ylabel('Magnitude, dB');
title('Filtered Output Signal');
```

<u>OUTPUT</u>



Filter Response


original Signal
Magnitude, dB
Frequency, Hz
×10^4
Noise Curropted Signal
Magnitude, dB
Frequency, Hz
×10^4
Filtered Output Signal
Magnitude, dB
Frequency, Hz
×10^4

EXPERIMENT NO-6

I) SOLVING GIVEN DIFFERENCE EQUATION

AIM: TO solve a given difference equation using Matlab/Simulink

THEORY: A General N^th^ order Difference equations looks like,

$y[n] + a_1 y[n-1] + a_2 y[n-2] + ...+ a_N y[n-N) = b_0 x[n] + b_1 x[n-1] + .. + b_M x[n-M]$. 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,

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

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

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

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

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

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

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

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 i.e., y[0] = 1.5y[-1] - y[-2] + 2x[-2]

= 1.5*2 - 1 + 2*0 y[0]

y[0] = 2

y[1] = 15y[0] - y[-1] + 2x[-1]

= 1.5*2 - 2 + 2*0

y[1] = 1

similarly y[2] = 1.5*y[1] - y[0] + 2*x[0]= 1.5

And so on...

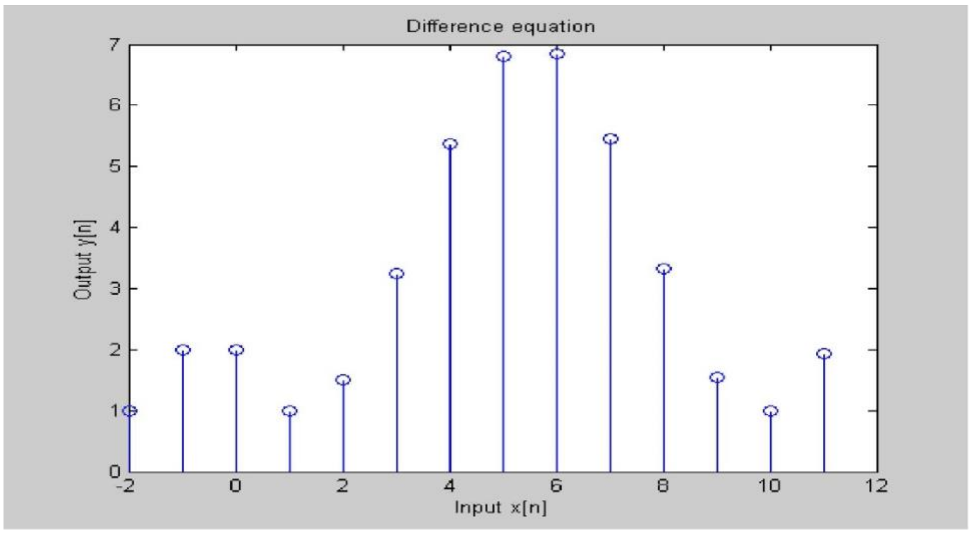
Matlab Program:

```
Clc;                                    % Clear screen
Clear all;                              % Clear all stored variables
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) +2)
y(k)= 1.5*y(k-1)-y(k-2)+2*x(k-2);      % Compute y[n] for various values of n
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]
Output y[n];                            %Name y-axis as Output y[n]

title('Difference equation');
```

a. **Output:**

```
Solution for the given difference equation

Columns 1 through 7

1.0000 2.0000 2.0000 1.0000 1.5000 3.2500 5.3750

Columns 8 through 14

6.8125 6.8438 5.4531 3.3359 1.5508 0.9902 1.9346
```

Graph:



Solving Difference equation using **Simulink:**

Example :consider a second order differential equation defined as:

$$\frac{d^2y}{dt^2} + 2\frac{dy}{dt} + 5y = 1$$

With all initail conditions assumed to be zero

Step 1: Rearrange the given equation to obtain expression for highest derivative

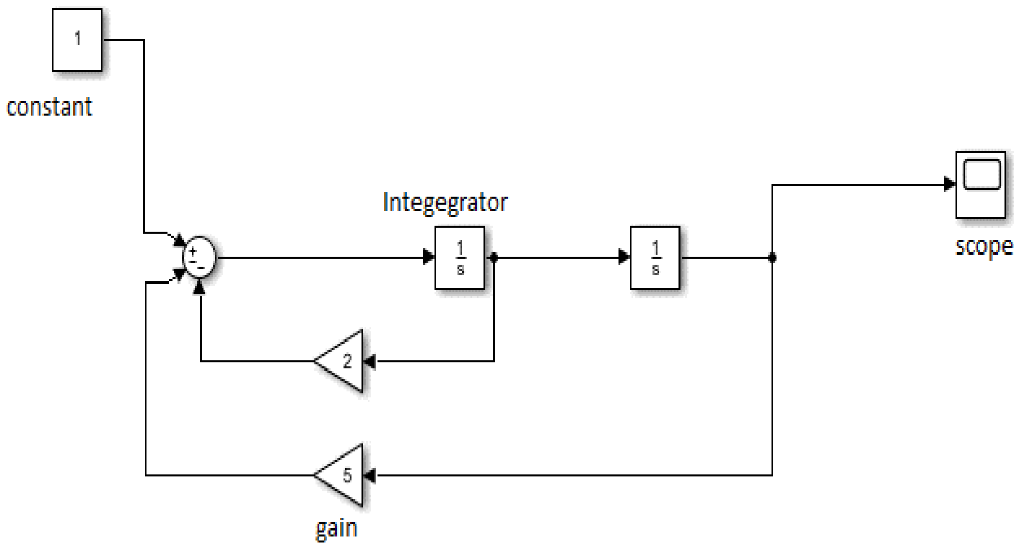
$$\frac{d^2y}{dt^2} = -2\frac{dy}{dt} - 5y + 1$$

Step2: Open Simulink and create a new model file.

Step 3: Open simulink library browser and find the following blocks(sum, integrator, scope, gain and constant), drag and drop the blocks into simulink model file

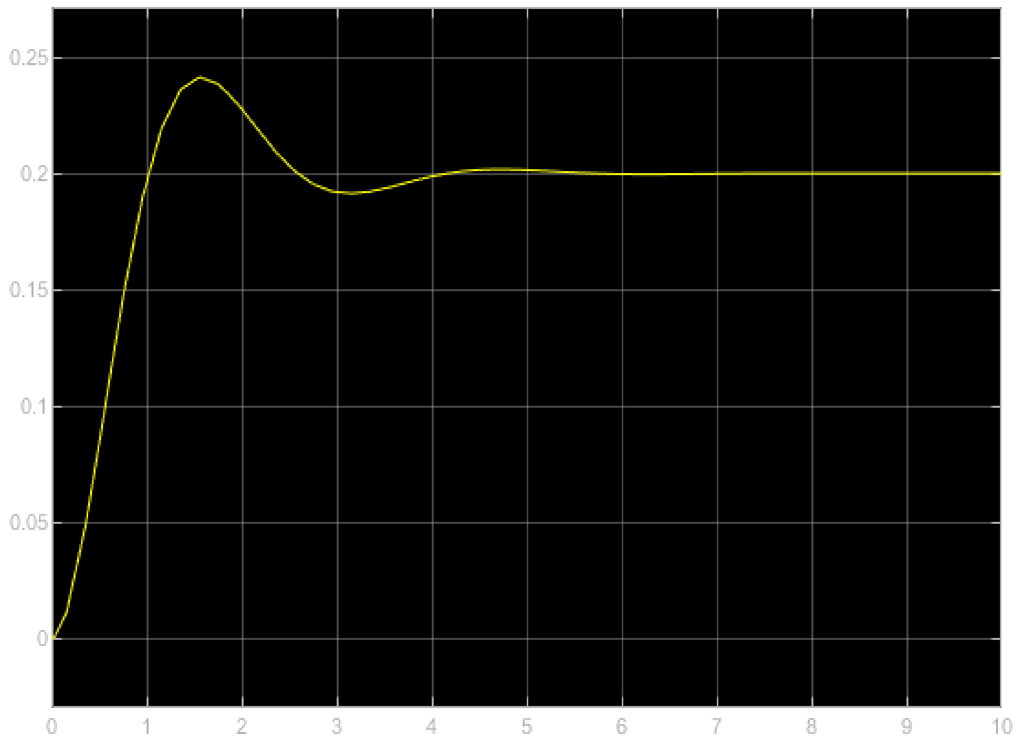
Step 4: Make the conncections as shown in simulink model

Step 5: Press "Run" and double click on scope to see the output



Simulink Model

<u>OUTPUT:</u>



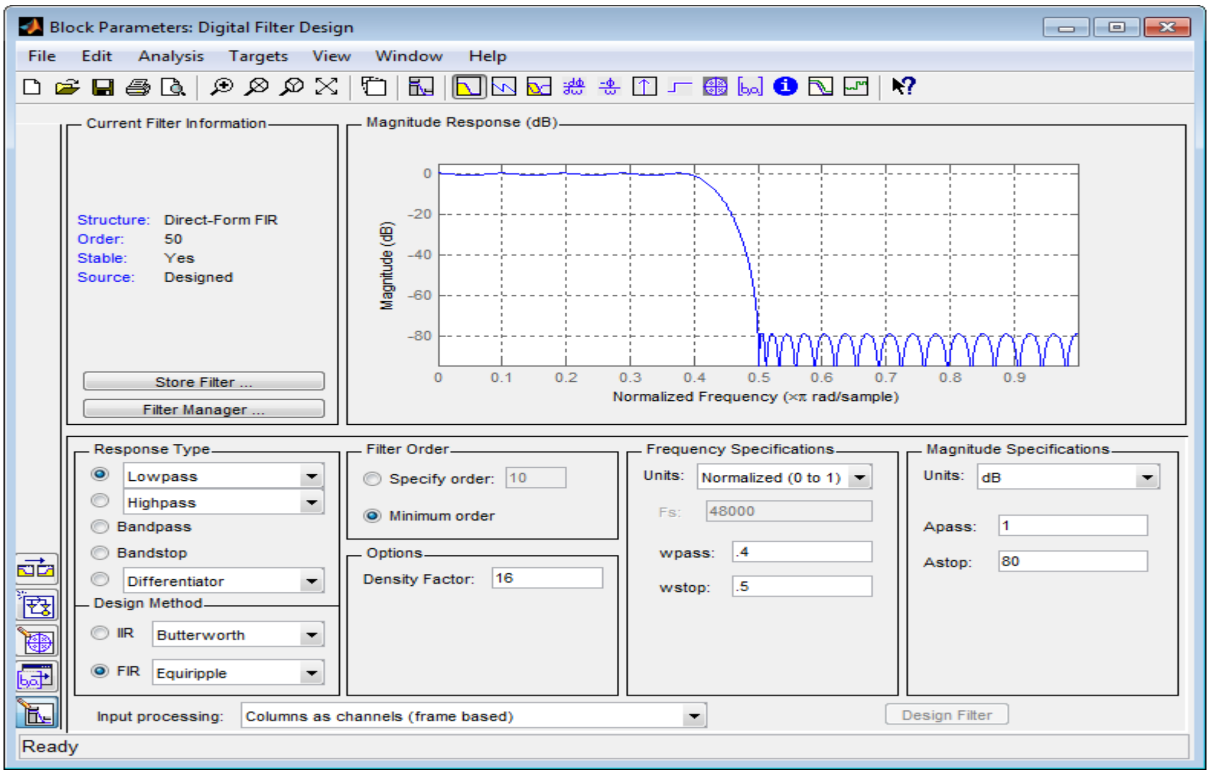
Output of Simulink

EXPERIMENT NO-7

Aim: Design Filter (FIR and IIR) to meet the given specifications using Simulink.

You can use the Digital Filter Design block to design and implement a digital FIR or IIR filter. In this topic, you use it to create an FIR lowpass filter:

1. Open Simulink and create a new model file.
2. From the DSP System Toolbox™ **Filtering library**, and then from the Filter **Implementations library**, click-and-drag a **Digital Filter Design block** into your model.
3. Double-click the Digital Filter Design block.
   The filter designer app opens.
4. Set the parameters as follows, and then click **OK**:
   - **Response Type** = `Lowpass`
   - **Design Method** = `FIR`, `Equiripple`
   - **Filter Order** = `Minimum order`
   - **Units** = `Normalized (0 to 1)`
   - **wpass** = `0.2`
   - **wstop** = `0.5`
5. Click **Design Filter** at the bottom of the app to design the filter. Your Digital Filter Design block now represents a filter with the parameters you specified.
6. From the **Edit** menu, select **Convert Structure**. The **Convert Structure** dialog box opens.
7. Select **Direct-Form FIR Transposed** and click **OK**.


Block Parameters: Digital Filter Design
File Edit Analysis Targets View Window Help
Magnitude Response (dB)
Magnitude (dB)
Normalized Frequency (×π rad/sample)
Store Filter ...
Filter Manager ...
Design Filter

Current Filter Information
Structure: Direct-Form FIR
Order: 50
Stable: Yes
Source: Designed

Response Type: [✓ Lowpass] Highpass Bandpass Bandstop Differentiator
Design Method: IIR Butterworth [✓ FIR Equiripple]
Filter Order: Specify order: 10 [✓ Minimum order]
Options: Density Factor: 16
Frequency Specifications: Units: Normalized (0 to 1); Fs: 48000; wpass: .4; wstop: .5
Magnitude Specifications: Units: dB; Apass: 1; Astop: 80
Input processing: Columns as channels (frame based)
Ready

Steps: In Command Window type **simulink** press enter to open simulink.
Go to **File** and click on **New Model**
Go to **Simulation Library**
From **Discrete Library**
Drag and drop a delay element (Block) from Discrete library.

Delay Block

Drag and drop a Scope from Sink library

Scope Block

From **Math Library**
Drag and drop a summer block

Summer Block

From **Math Library**
Drag and drop a Gain block

Gain Block

From **Math Library**
Drag and drop a product block

Product Block

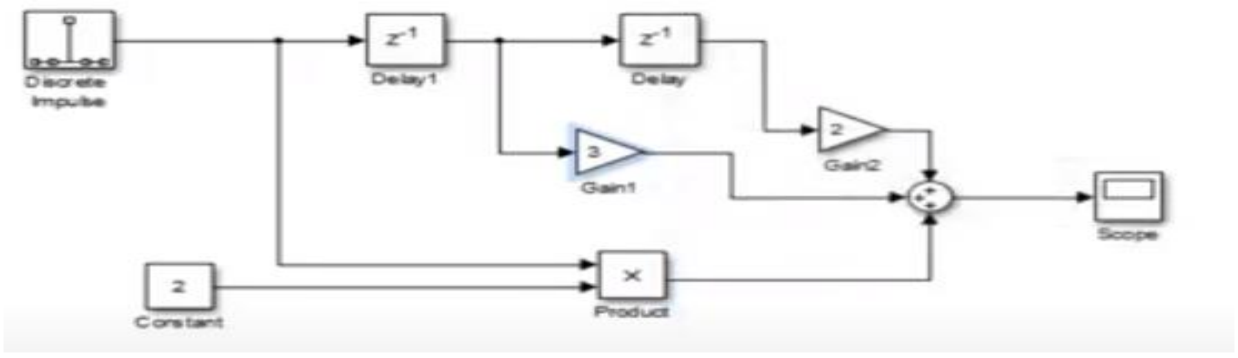
Go to DSP System Toolbox→ Click o Sources
Drag and drop discrete impulse

Discrete Impulse

Drag and drop a constant.

Constant Block

If you need more number of blocks, copy and paste the particular block as many as you needed.
Double click on the block and change the signs and values.
Make the connection as shown in the below figure



Click on run and Double click on Scope to see the output.

Manual of DSP 6713 Starter Kit

Contents

Consignment list

1. A main board of DSP6713 -1No
2. Power Supply 5V, 3A. - 1No
3. Emulator with USB Cable -1No
4. Head Phone with MIC -1No
5. Audio Jack -3No [1+2]
6. Software and VTU Programs CD -1No

7.5th SEM Lab DSP Manual -1No

8. TMSC3206713 and AIC23 Codec Data sheets -1No

Introduction of main board

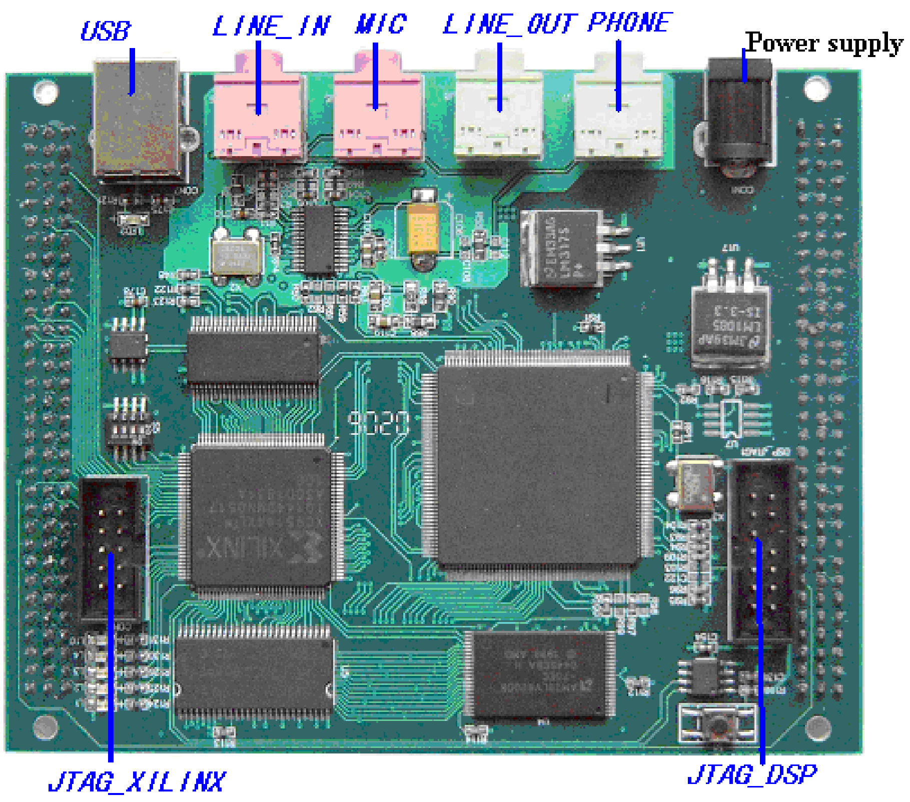
1. USB2.0 CY7C68013-56PVC, compatible with USB2.0 and USB1.1, including 8051
2. DSP TMS320C6713 TQFP-208 Package Device with, 4 layers board
3. SDRAM MT48LC4M16A2 1meg*16 *4 bank micron
4. FLASH AM29LV800B 8Mbit1Mbyte of AMD
5. RESET chip specialize for reset with button for manually reset

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.



Board Picture:

The setup of XDS510 USB emulator in CCS V3.1

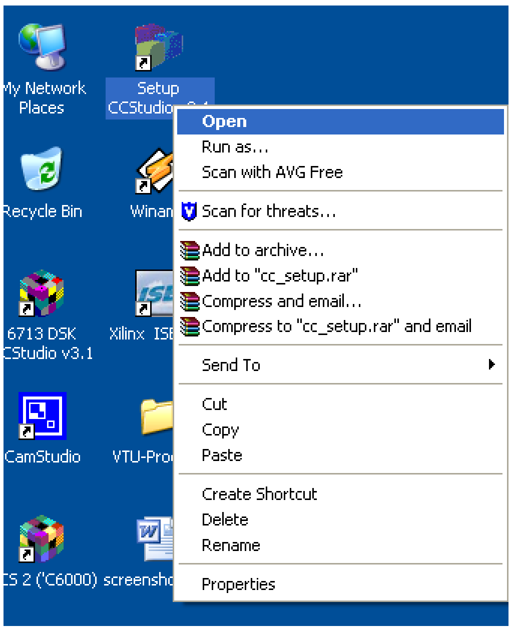
1. Install CCS V3.1 software according to the Custom Install default Custom Install， could change the installation directory. Following is an example of install the software under C disk root directory.
2. Install USB Emulator choose to install directory from CCS v3.1 that is if CCSv3.1 is installed in C: / CCStudio_v3.1 directory, then install the USB emulator driver in this directory.
3. Replace TIXDS510_Connection.xml file
   [From \CCStudio_v3.1\drivers\TargetDB\ connection with the TIXDS510_Connection.xml

   in xds510_setup directory.]

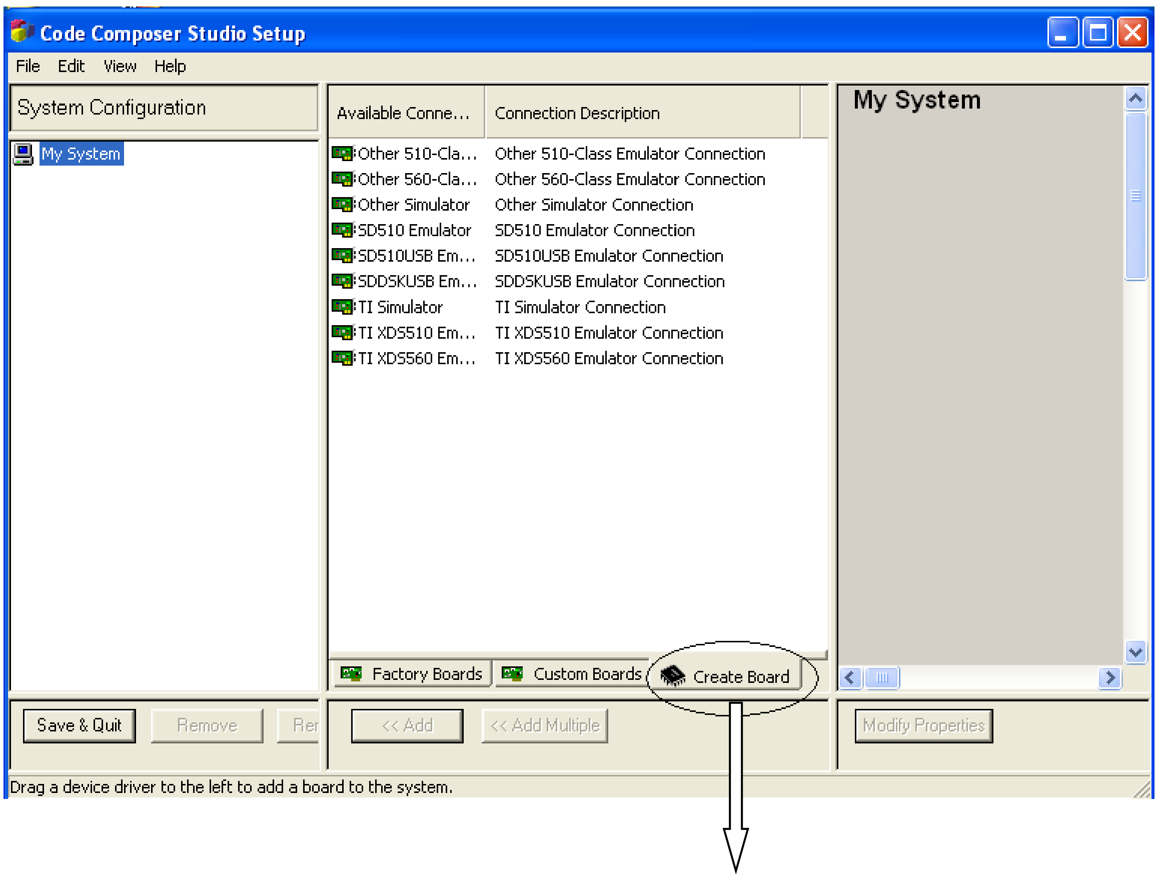
<u>Procedure to Setup Emulator</u>:

Start->Program->Texas Instrument-> Code Composer Studio 3.1 ->Setup Code Composer Studio v3.1, the window of Code Composer Studio Setup will show up.

1. Open the Setup CCStudio v3.1



2. Select Create Board (Marked in Circle, can witness in the below Figure)



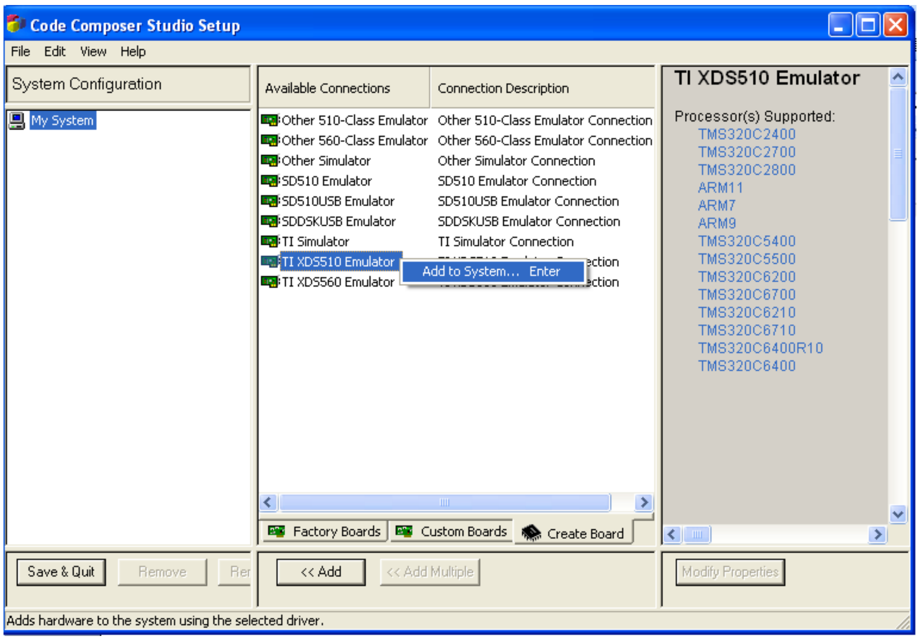
Available Conne...	Connection Description
Other 510-Cla...	Other 510-Class Emulator Connection
Other 560-Cla...	Other 560-Class Emulator Connection
Other Simulator	Other Simulator Connection
SD510 Emulator	SD510 Emulator Connection
SD510USB Em...	SD510USB Emulator Connection
SDDSKUSB Em...	SDDSKUSB Emulator Connection
TI Simulator	TI Simulator Connection
TI XDS510 Em...	TI XDS510 Emulator Connection
TI XDS560 Em...	TI XDS560 Emulator Connection

Drag a device driver to the left to add a board to the system.

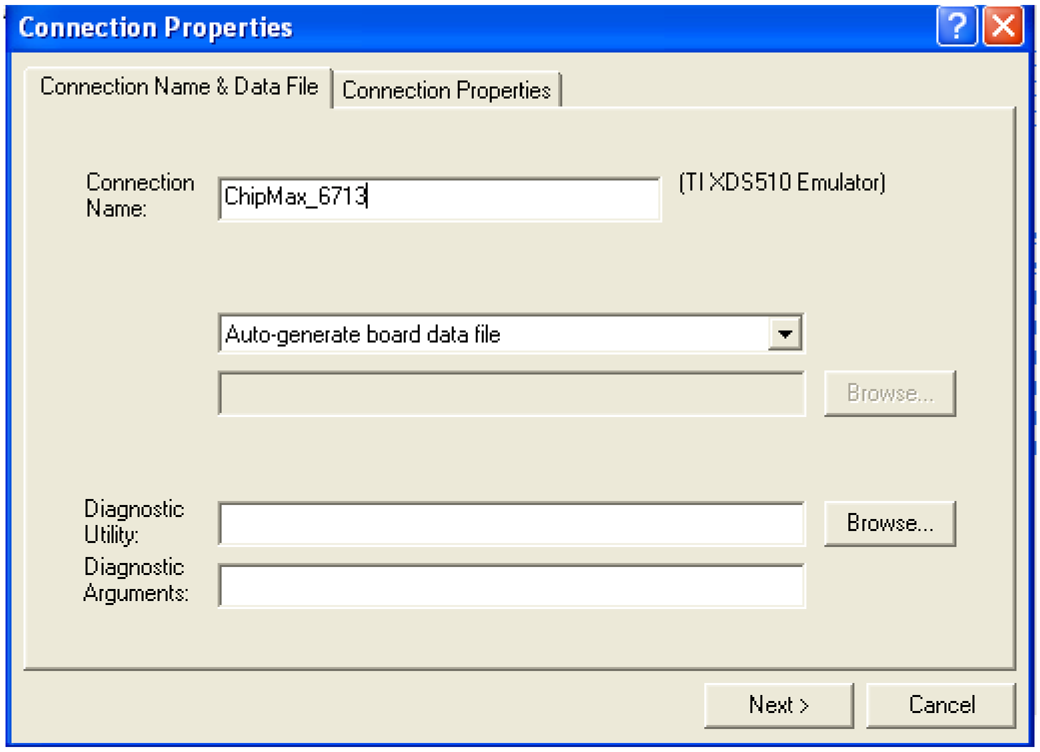
Chose this option

3. Right Click on the TI XDS510 emulator and select add to system.. Enter,

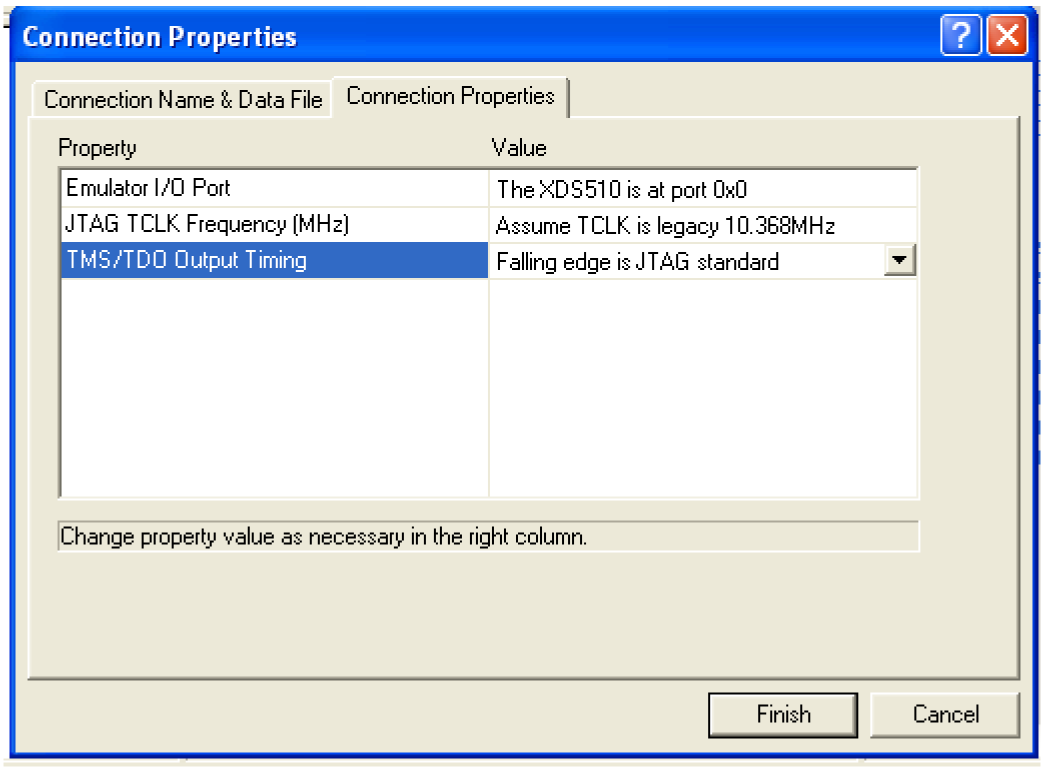
(After selecting that option a Connection Properties window will be opened as shown in step 4).


Code Composer Studio Setup
File Edit View Help
System Configuration
My System
Available Connections
Connection Description
Other 510-Class Emulator | Other 510-Class Emulator Connection
Other 560-Class Emulator | Other 560-Class Emulator Connection
Other Simulator | Other Simulator Connection
SD510 Emulator | SD510 Emulator Connection
SD510USB Emulator | SD510USB Emulator Connection
SDDSKUSB Emulator | SDDSKUSB Emulator Connection
TI Simulator | TI Simulator Connection
TI XDS510 Emulator
Add to System... Enter
TI XDS560 Emulator
TI XDS510 Emulator
Processor(s) Supported:
TMS320C2400
TMS320C2700
TMS320C2800
ARM11
ARM7
ARM9
TMS320C5400
TMS320C5500
TMS320C6200
TMS320C6700
TMS320C6210
TMS320C6710
TMS320C6400R10
TMS320C6400
Factory Boards
Custom Boards
Create Board
Save & Quit
Remove
<< Add
<< Add Multiple
Modify Properties
Adds hardware to the system using the selected driver.

4. Provide Connection Name as ChipMax_6713 and click on Next.



5. Choose the option Falling edge is JTAG Standard in TMS/TDO Output Timing.


Connection Properties

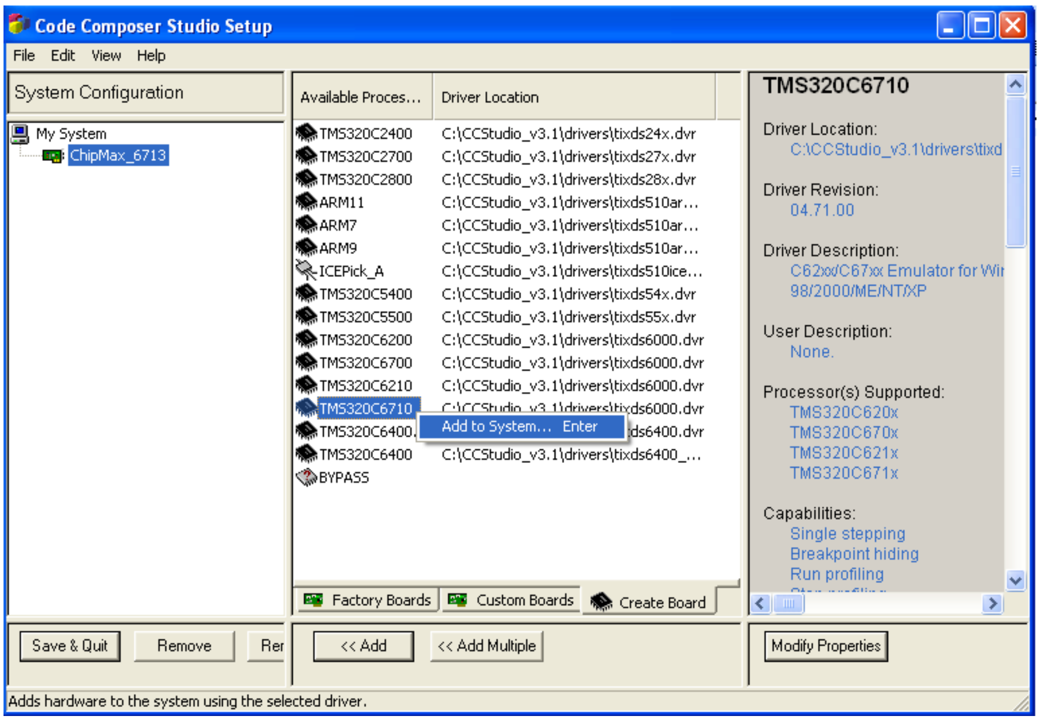
Connection Name & Data File | Connection Properties

Property	Value
Emulator I/O Port	The XDS510 is at port 0x0
JTAG TCLK Frequency (MHz)	Assume TCLK is legacy 10.368MHz
TMS/TDO Output Timing	Falling edge is JTAG standard

Change property value as necessary in the right column.

Finish | Cancel

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


Code Composer Studio Setup

File Edit View Help

System Configuration

My System
ChipMax_6713

Available Proces…	Driver Location
TMS320C2400	C:\CCStudio_v3.1\drivers\tixds24x.dvr
TMS320C2700	C:\CCStudio_v3.1\drivers\tixds27x.dvr
TMS320C2800	C:\CCStudio_v3.1\drivers\tixds28x.dvr
ARM11	C:\CCStudio_v3.1\drivers\tixds510ar…
ARM7	C:\CCStudio_v3.1\drivers\tixds510ar…
ARM9	C:\CCStudio_v3.1\drivers\tixds510ar…
ICEPick_A	C:\CCStudio_v3.1\drivers\tixds510ice…
TMS320C5400	C:\CCStudio_v3.1\drivers\tixds54x.dvr
TMS320C5500	C:\CCStudio_v3.1\drivers\tixds55x.dvr
TMS320C6200	C:\CCStudio_v3.1\drivers\tixds6000.dvr
TMS320C6700	C:\CCStudio_v3.1\drivers\tixds6000.dvr
TMS320C6210	C:\CCStudio_v3.1\drivers\tixds6000.dvr
TMS320C6710	C:\CCStudio_v3.1\drivers\tixds6000.dvr
TMS320C6400	ds6400.dvr
TMS320C6400	C:\CCStudio_v3.1\drivers\tixds6400_…
BYPASS	

Add to System… Enter

Factory Boards | Custom Boards | Create Board

TMS320C6710

Driver Location:
C:\CCStudio_v3.1\drivers\tixd

Driver Revision:
04.71.00

Driver Description:
C62xx/C67xx Emulator for Wi
98/2000/ME/NT/XP

User Description:
None.

Processor(s) Supported:
TMS320C620x
TMS320C670x
TMS320C621x
TMS320C671x

Capabilities:
Single stepping
Breakpoint hiding
Run profiling

Save & Quit | Remove | Rer | << Add | << Add Multiple | Modify Properties

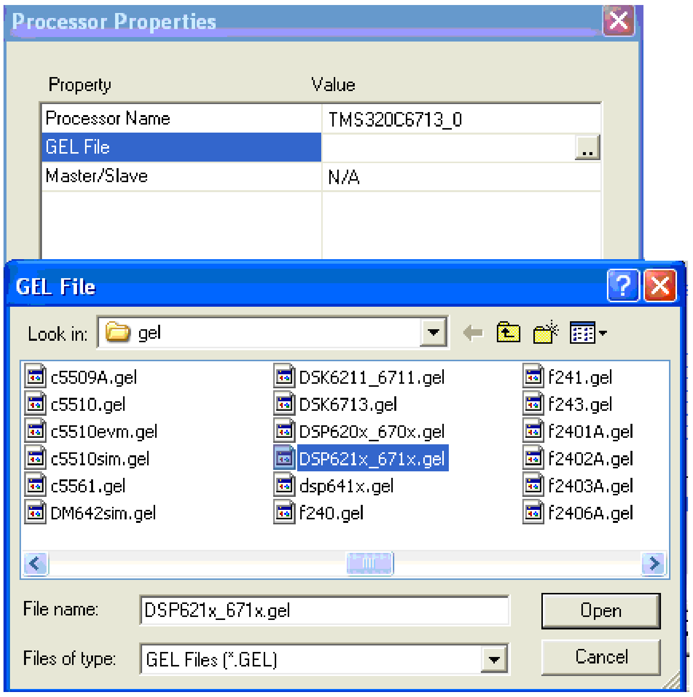
Adds hardware to the system using the selected driver.

7. a) Provide Processor Name as TMS320C6713_0

b) Select GEL File, Click on browse icon and select DSP621x_671x.gel.

c) Select N/A in Master/Slave.

d) Click on Ok.



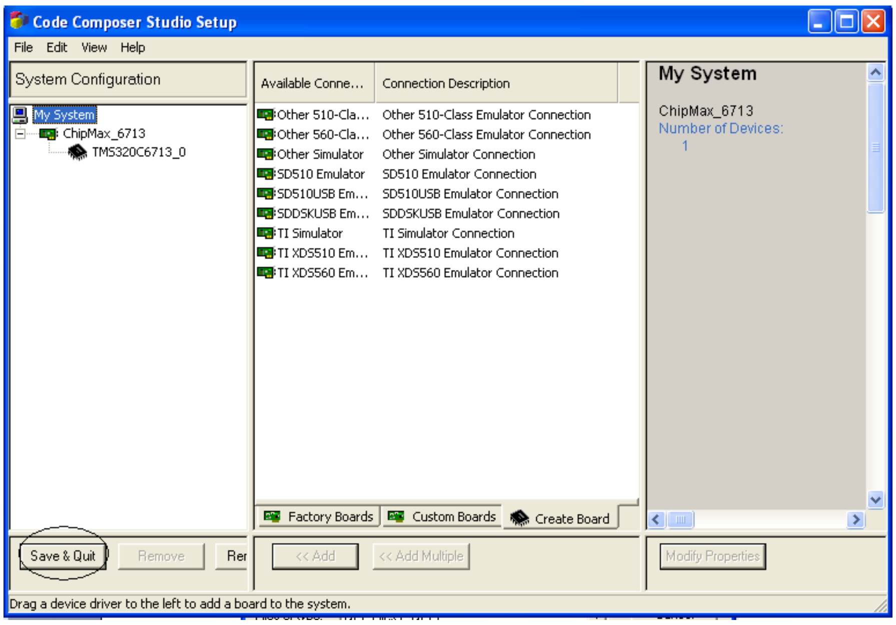
Property	Value
Processor Name	TMS320C6713_0
GEL File	
Master/Slave	N/A

c5509A.gel, c5510.gel, c5510evm.gel, c5510sim.gel, c5561.gel, DM642sim.gel, DSK6211_6711.gel, DSK6713.gel, DSP620x_670x.gel, DSP621x_671x.gel, dsp641x.gel, f240.gel, f241.gel, f243.gel, f2401A.gel, f2402A.gel, f2403A.gel, f2406A.gel

File name: DSP621x_671x.gel

Files of type: GEL Files (*.GEL)

8. Click on Save and quit (highlighted by Circle).



Available Conne...	Connection Description
Other 510-Cla...	Other 510-Class Emulator Connection
Other 560-Cla...	Other 560-Class Emulator Connection
Other Simulator	Other Simulator Connection
SD510 Emulator	SD510 Emulator Connection
SD510USB Em...	SD510USB Emulator Connection
SDDSKUSB Em...	SDDSKUSB Emulator Connection
TI Simulator	TI Simulator Connection
TI XDS510 Em...	TI XDS510 Emulator Connection
TI XDS560 Em...	TI XDS560 Emulator Connection

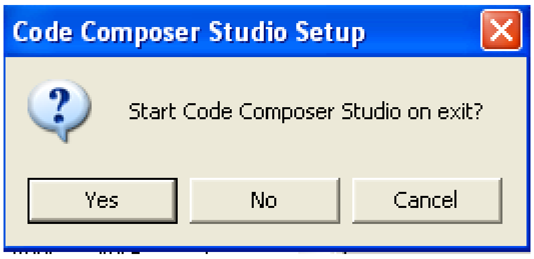
My System

ChipMax_6713

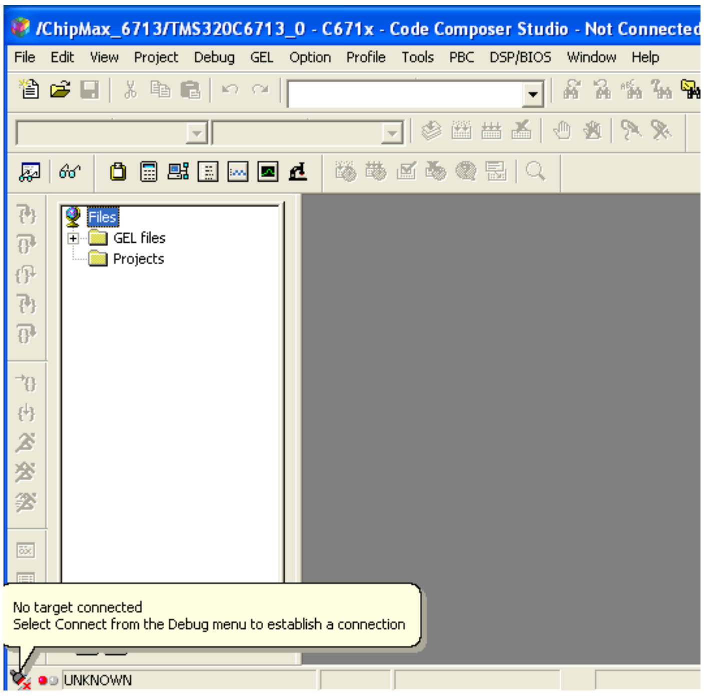
Number of Devices:
1

Drag a device driver to the left to add a board to the system.

9. Click on Yes to start Code Composer Studio.

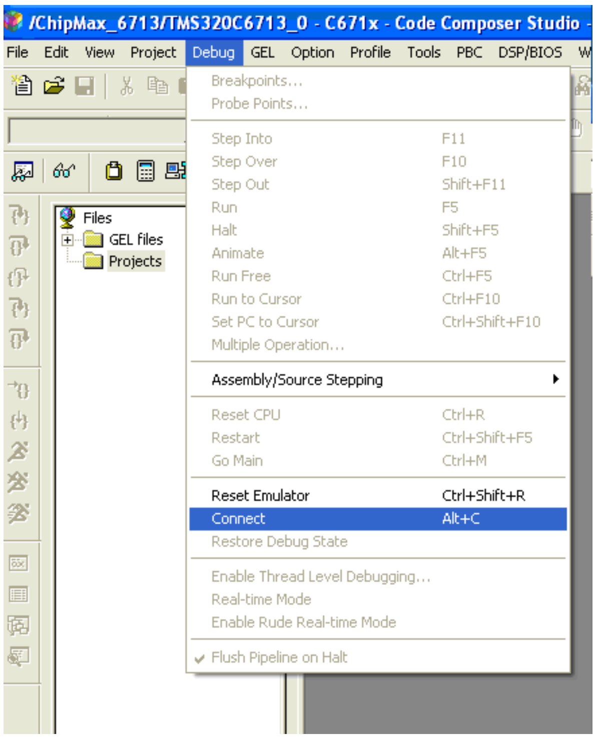

Code Composer Studio Setup

Start Code Composer Studio on exit?

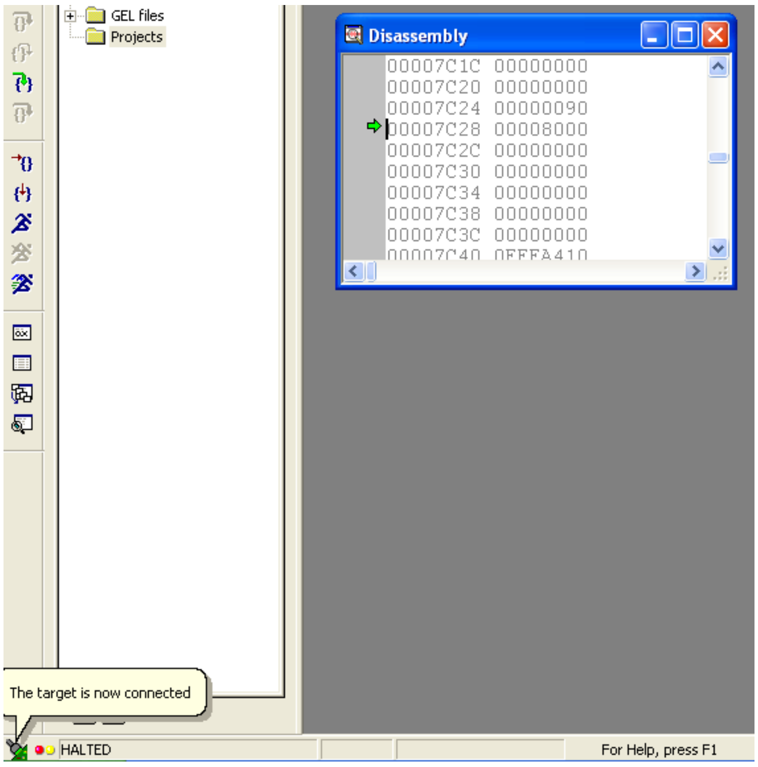

/ChipMax_6713/TMS320C6713_0 - C671x - Code Composer Studio - Not Connected

No target connected
Select Connect from the Debug menu to establish a connection

10. Go to Debug and select the option connect.



11. Now Target is connected

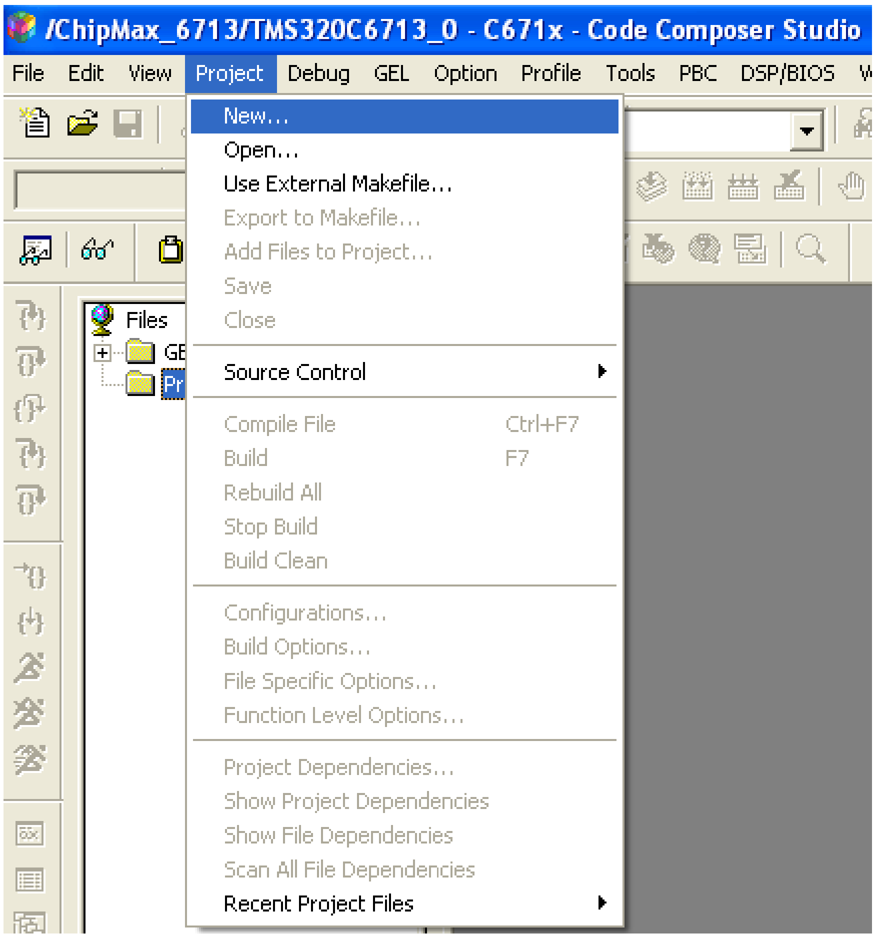


```
00007C1C  00000000
00007C20  00000000
00007C24  00000090
00007C28  00008000
00007C2C  00000000
00007C30  00000000
00007C34  00000000
00007C38  00000000
00007C3C  00000000
00007C40  0FFFA410
```

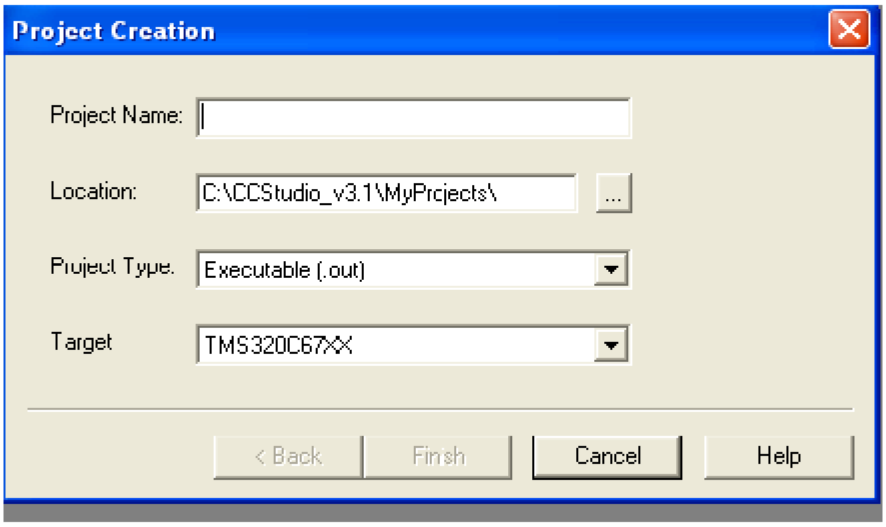
Experiment 1: Linear Convolution

Procedure to create new Project:

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



2. Give project name and click on finish.



Project Name:

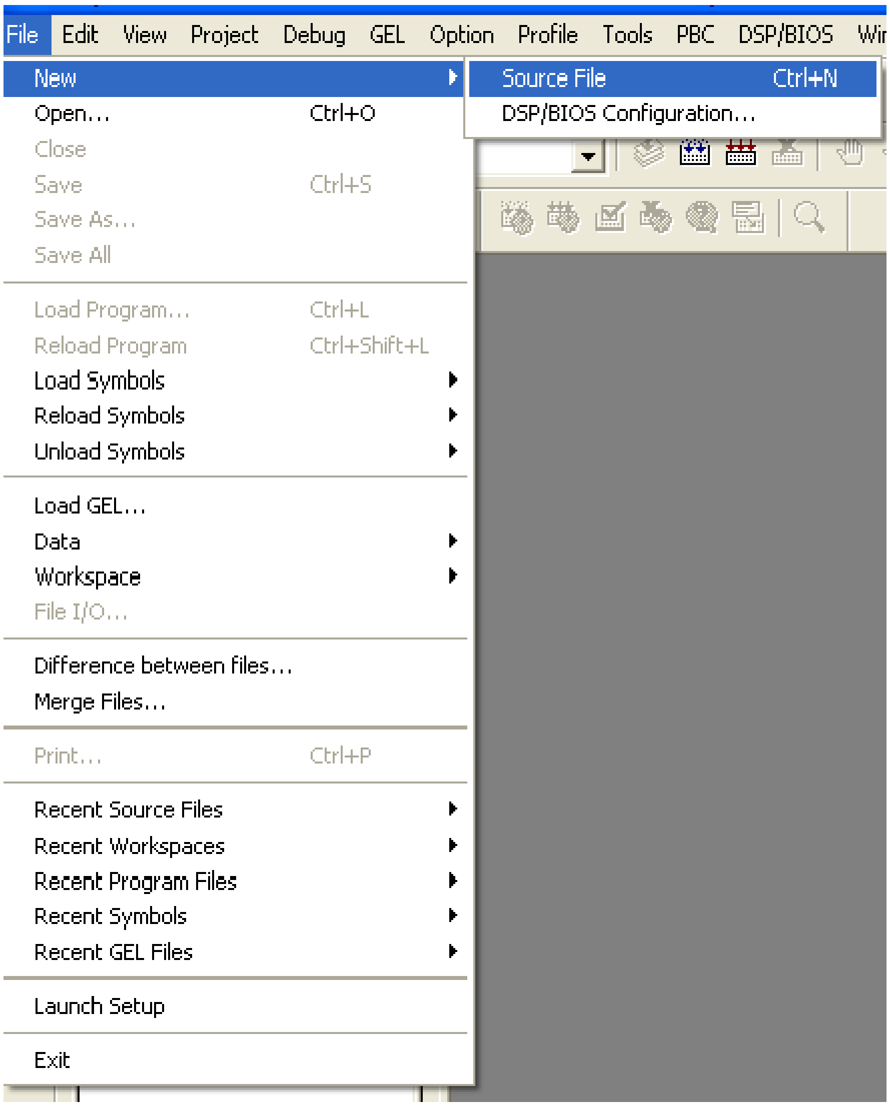
Location: C:\CCStudio_v3.1\MyProjects\

Project Type: Executable (.out)

Target TMS320C67XX

(**<u>Note:</u>** Location must be c:\CCStudio_v3.1\MyProjects).

3. Click on File $\Rightarrow$ New $\Rightarrow$ Source File, To write the Source Code.


File
Edit
View
Project
Debug
GEL
Option
Profile
Tools
PBC
DSP/BIOS
Wir
New
Source File Ctrl+N
DSP/BIOS Configuration...
Open... Ctrl+O
Close
Save Ctrl+S
Save As...
Save All
Load Program... Ctrl+L
Reload Program Ctrl+Shift+L
Load Symbols
Reload Symbols
Unload Symbols
Load GEL...
Data
Workspace
File I/O...
Difference between files...
Merge Files...
Print... Ctrl+P
Recent Source Files
Recent Workspaces
Recent Program Files
Recent Symbols
Recent GEL Files
Launch Setup
Exit

Aim: Linear Convolution of the two given sequences

Mathematical Formula:

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

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

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

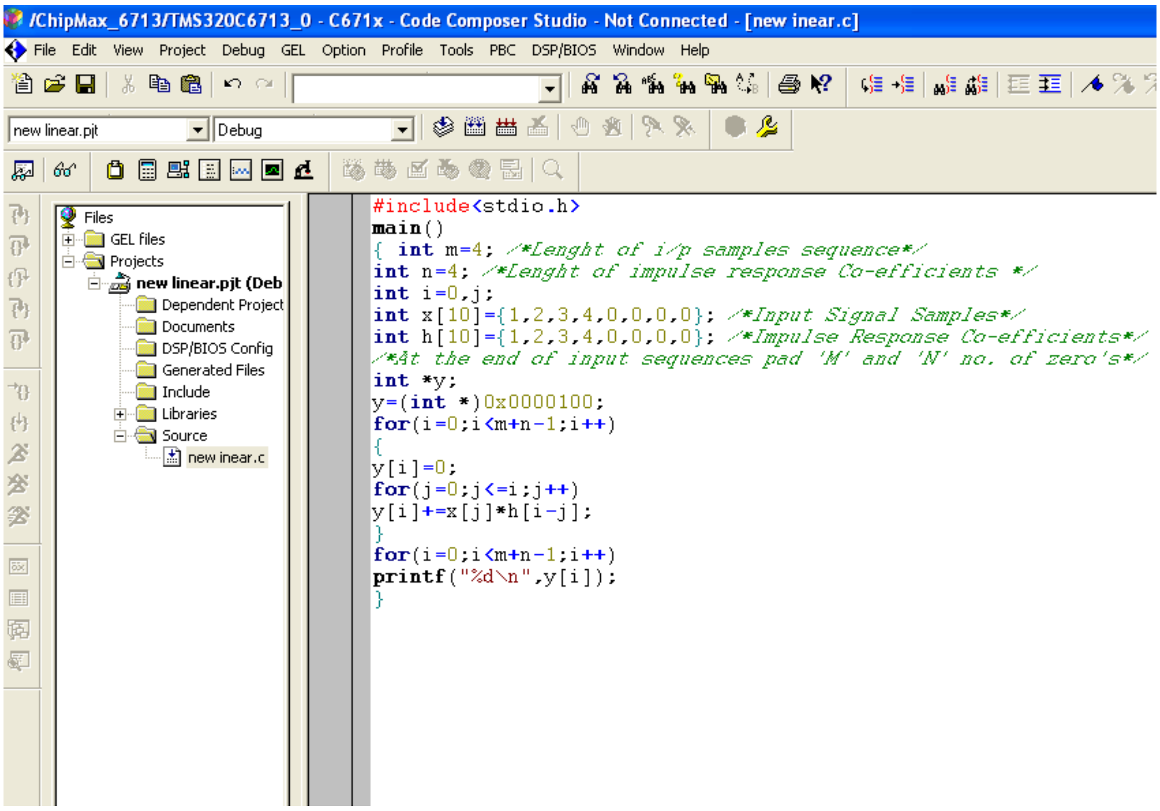
Program:

```
#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;

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]);

}
```

Output:

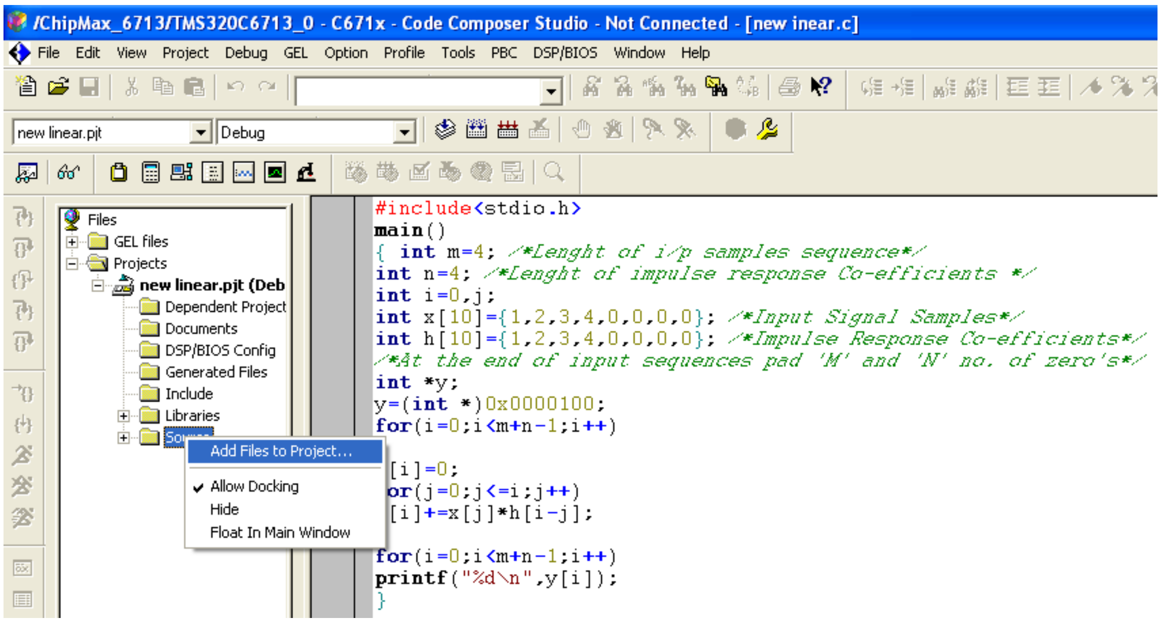
1, 4, 10, 20, 25, 24, 16.

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


/ChipMax_6713/TMS320C6713_0 - C671x - Code Composer Studio - Not Connected - [new inear.c]
File Edit View Project Debug GEL Option Profile Tools PBC DSP/BIOS Window Help
new linear.pjt
Debug
Files
GEL files
Projects
new linear.pjt (Deb
Dependent Project
Documents
DSP/BIOS Config
Generated Files
Include
Libraries
Source
new inear.c

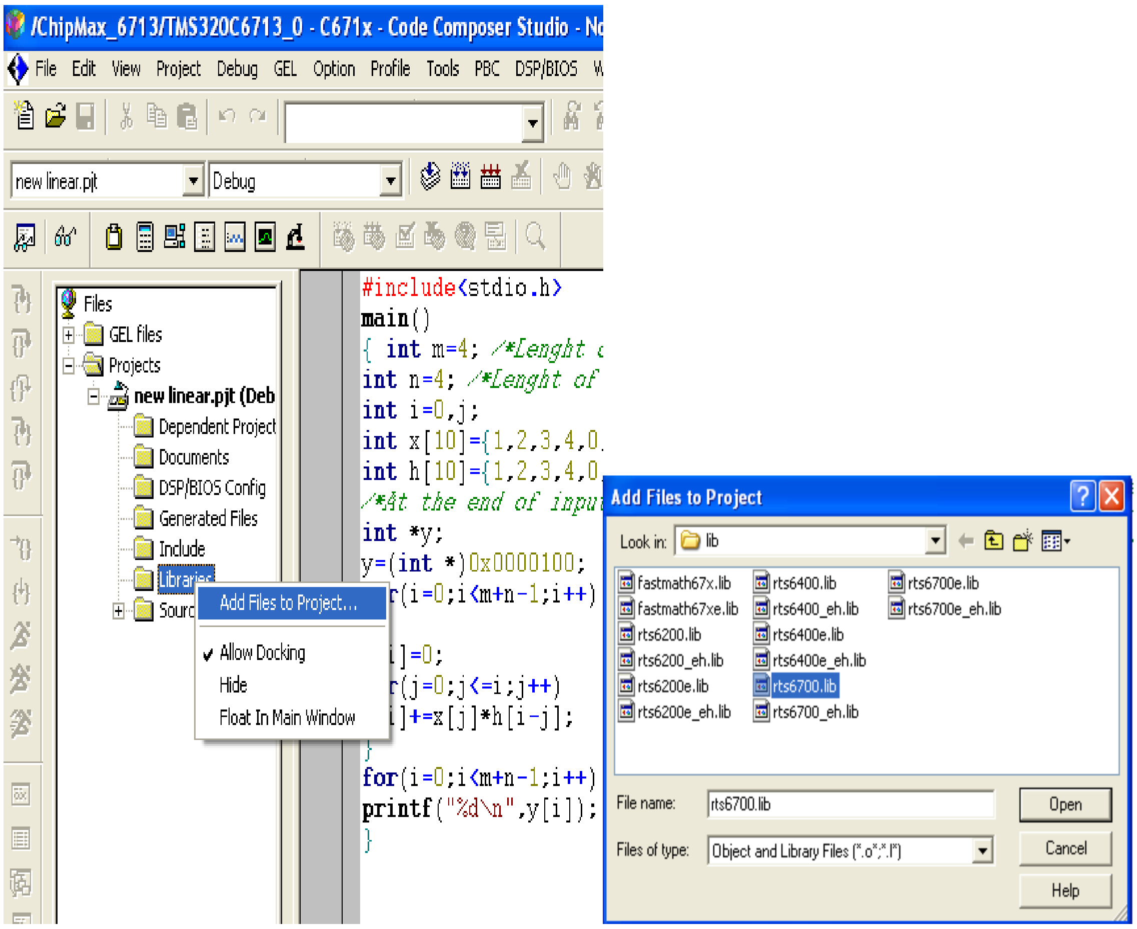
```
#include<stdio.h>
main()
{ int m=4; /*Lenght of i/p samples sequence*/
int n=4; /*Lenght of impulse response Co-efficients */
int i=0,j;
int x[10]={1,2,3,4,0,0,0,0}; /*Input Signal Samples*/
int h[10]={1,2,3,4,0,0,0,0}; /*Impulse Response Co-efficients*/
/*At the end of input sequences pad 'M' and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

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



```
#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;
for(i=0;i<m+n-1;i++)

 [i]=0;
 or(j=0;j<=i;j++)
 [i]+=x[j]*h[i-j];

for(i=0;i<m+n-1;i++)
printf("%d\n",y[i]);
}
```

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.



```
#include<stdio.h>
main()
{ int m=4; /*Lenght c
int n=4; /*Lenght of
int i=0,j;
int x[10]={1,2,3,4,0,
int h[10]={1,2,3,4,0,
/*At the end of inpu
int *y;
y=(int *)0x0000100;
r(i=0;i<m+n-1;i++)

i]=0;
r(j=0;j<=i;j++)
i]+=x[j]*h[i-j];
}
for(i=0;i<m+n-1;i++)
printf("%d\n",y[i]);
}
```

Look in: lib

fastmath67x.lib	rts6400.lib	rts6700e.lib
fastmath67xe.lib	rts6400_eh.lib	rts6700e_eh.lib
rts6200.lib	rts6400e.lib	
rts6200_eh.lib	rts6400e_eh.lib	
rts6200e.lib	rts6700.lib	
rts6200e_eh.lib	rts6700_eh.lib	

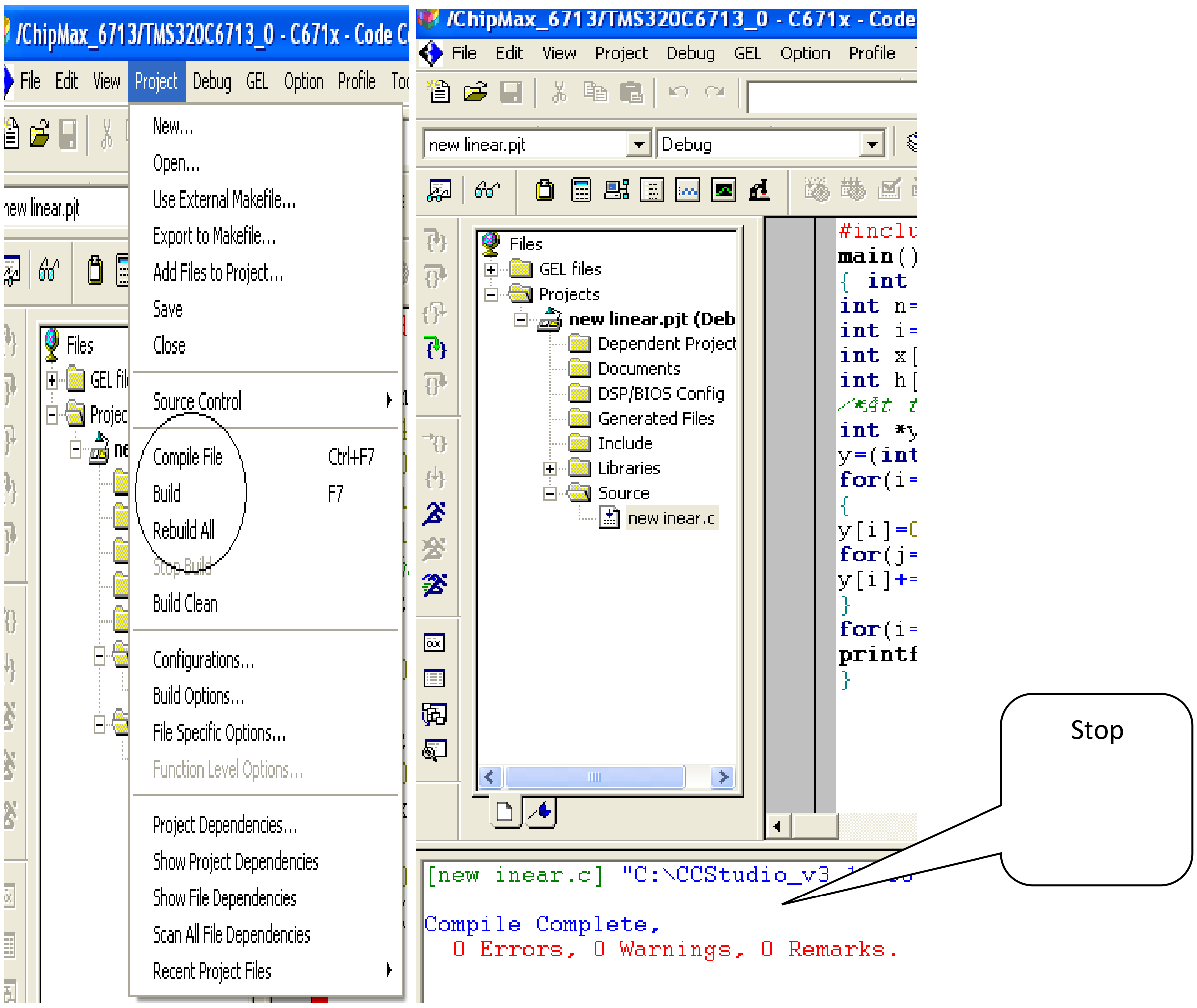
File name: rts6700.lib

Files of type: Object and Library Files (*.o*;*.l*)

7. a)Go to Project to Compile .

   b) Go to Project to Build.

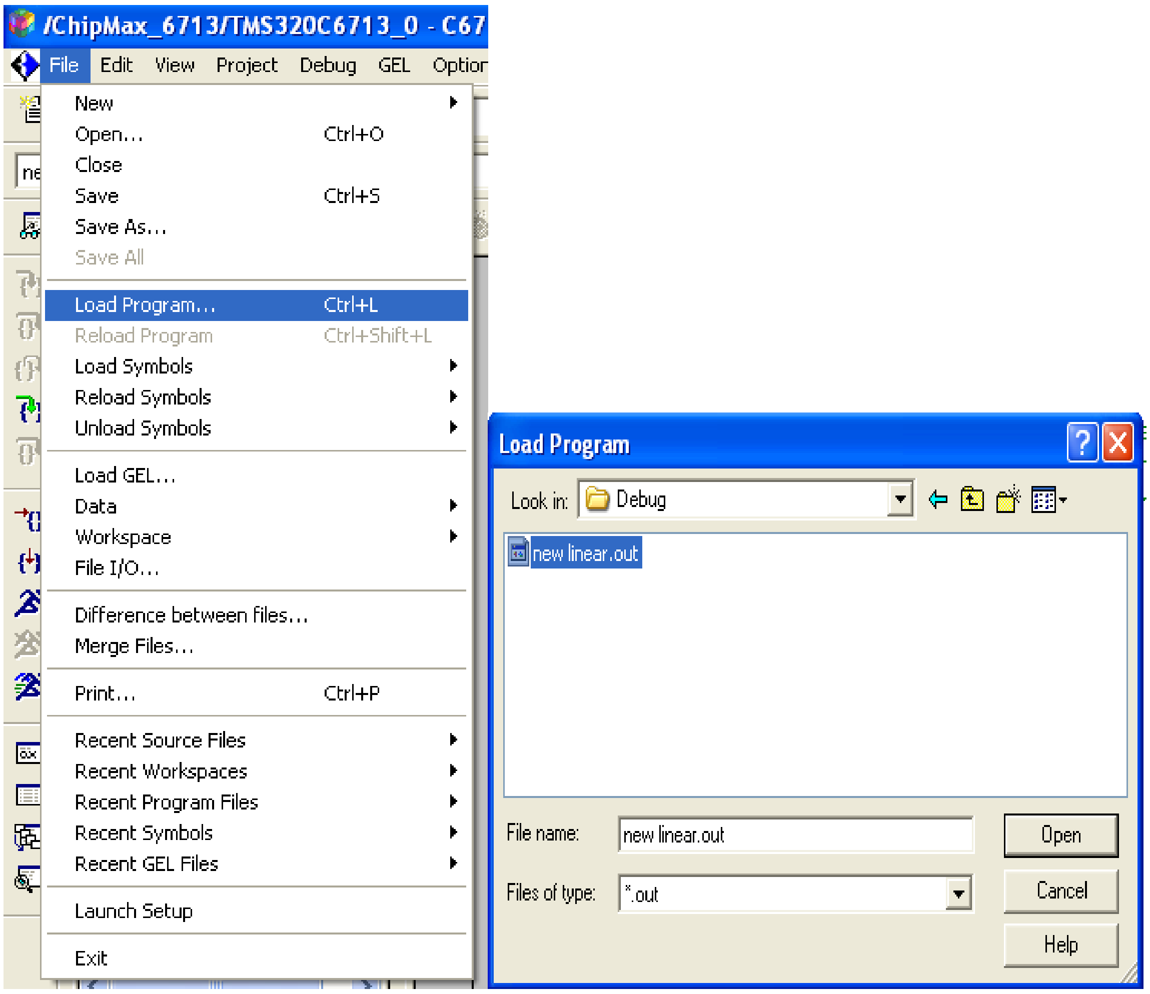
   c) Go to Project to Rebuild All.



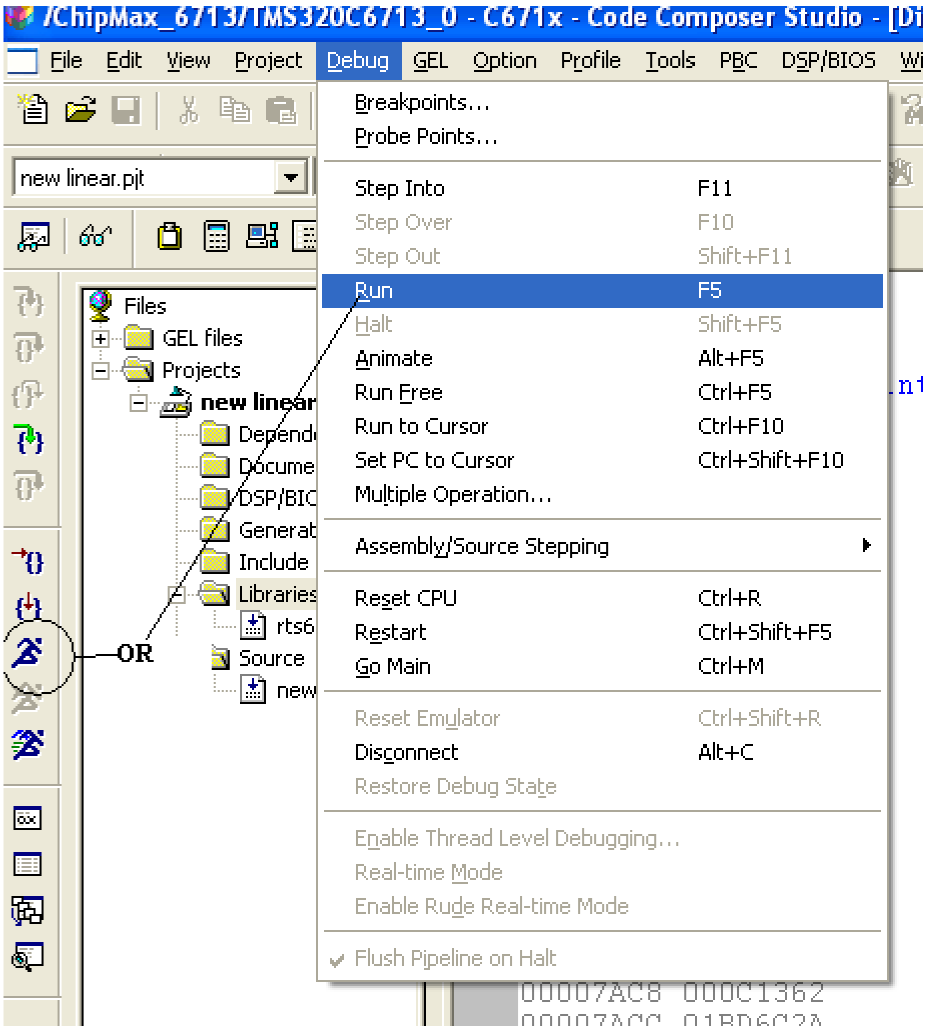
```
#inclu
main()
{ int
int n=
int i=
int x[
int h[
/*at t
int *y
y=(int
for(i=
{
y[i]=(
for(j=
y[i]+=
}
for(i=
printf
}
```

```
[new inear.c] "C:\CCStudio_v3

Compile Complete,
  0 Errors, 0 Warnings, 0 Remarks.
```

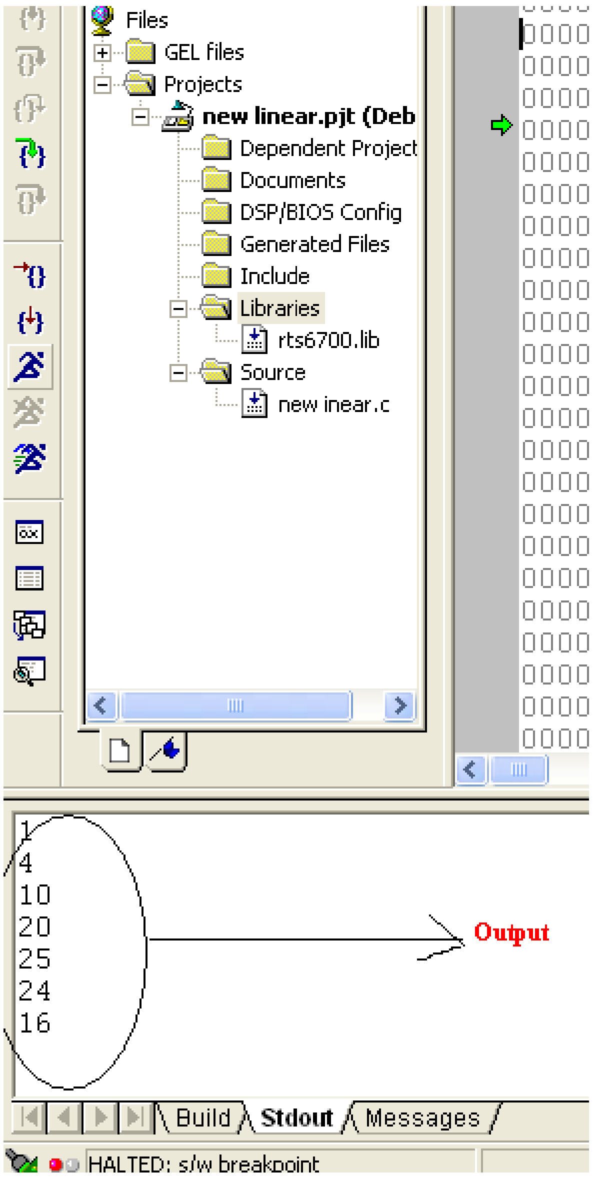
8. Go to file and load program and load **“.out”** file into the board..



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



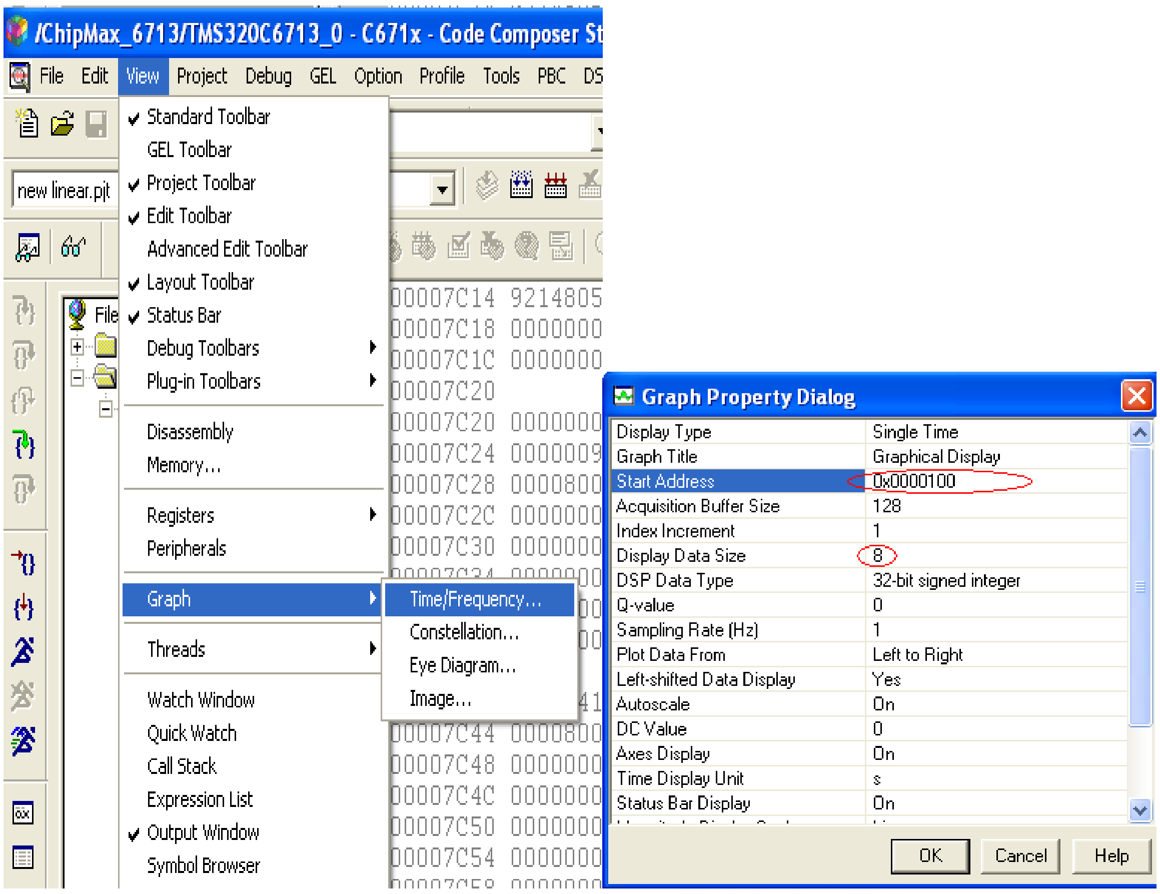
10. Observe the output in output window.



```
1
4
10
20
25
24
16
```

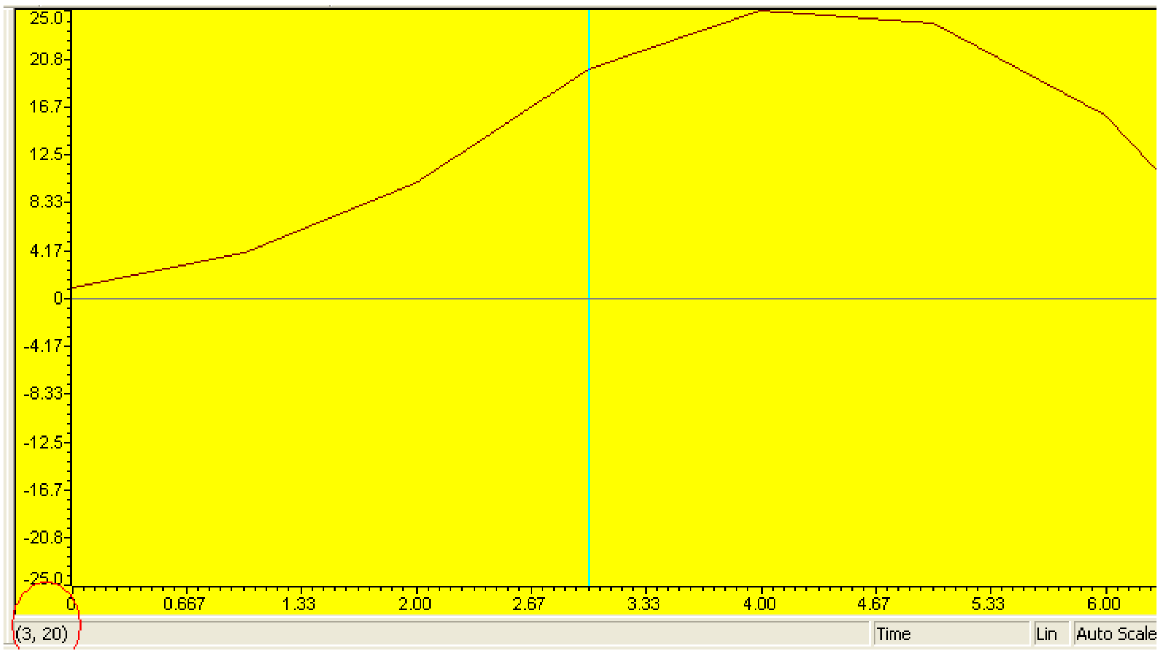
HALTED: s/w breakpoint

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.



Display Type	Single Time
Graph Title	Graphical Display
Start Address	0x0000100
Acquisition Buffer Size	128
Index Increment	1
Display Data Size	8
DSP Data Type	32-bit signed integer
Q-value	0
Sampling Rate (Hz)	1
Plot Data From	Left to Right
Left-shifted Data Display	Yes
Autoscale	On
DC Value	0
Axes Display	On
Time Display Unit	s
Status Bar Display	On

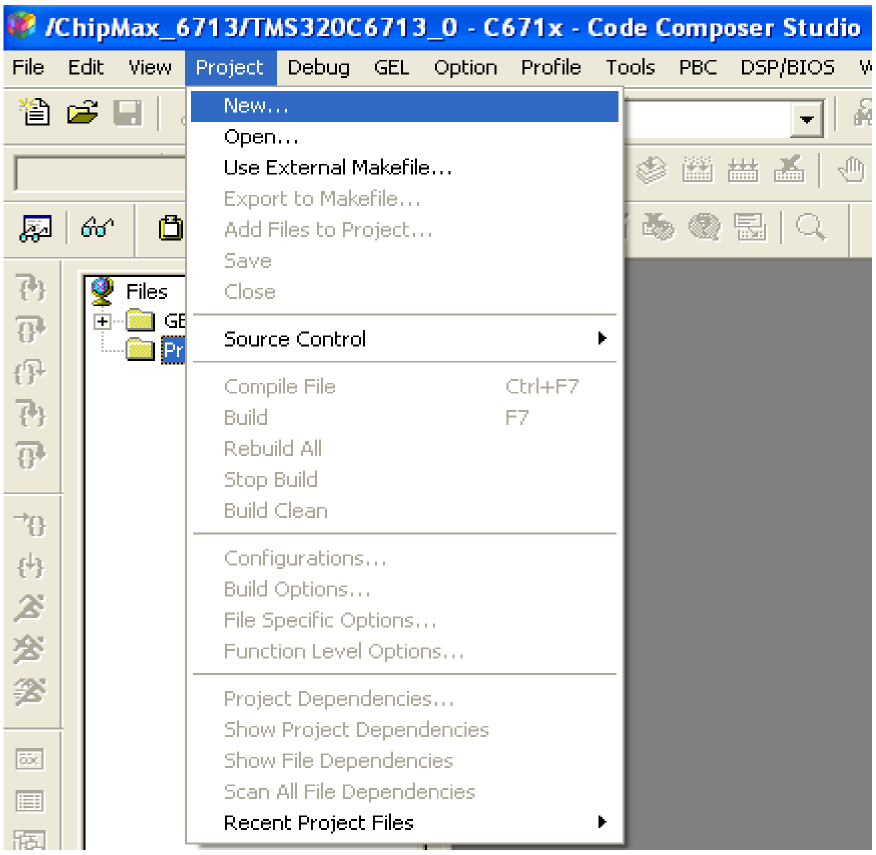
12. Green line is to choose the point, Value at the point can be seen (Highlighted by circle at the left corner).



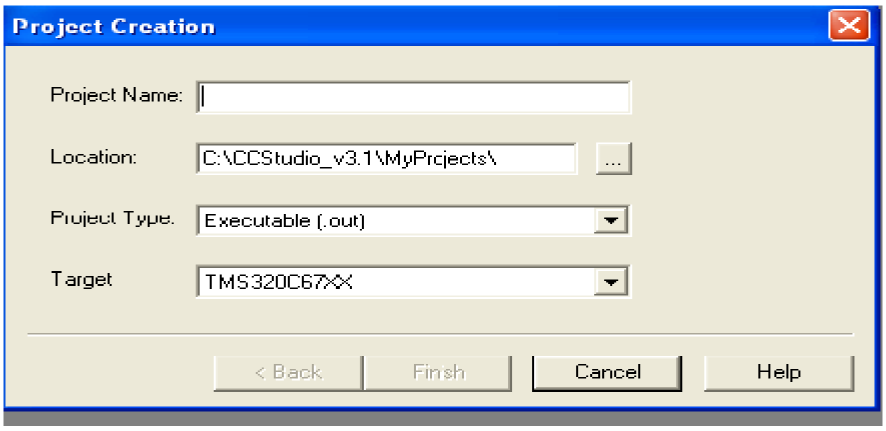
Circular Convolution

Procedure to create new Project:

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



2. Give project name and click on finish.


Project Creation

Project Name:

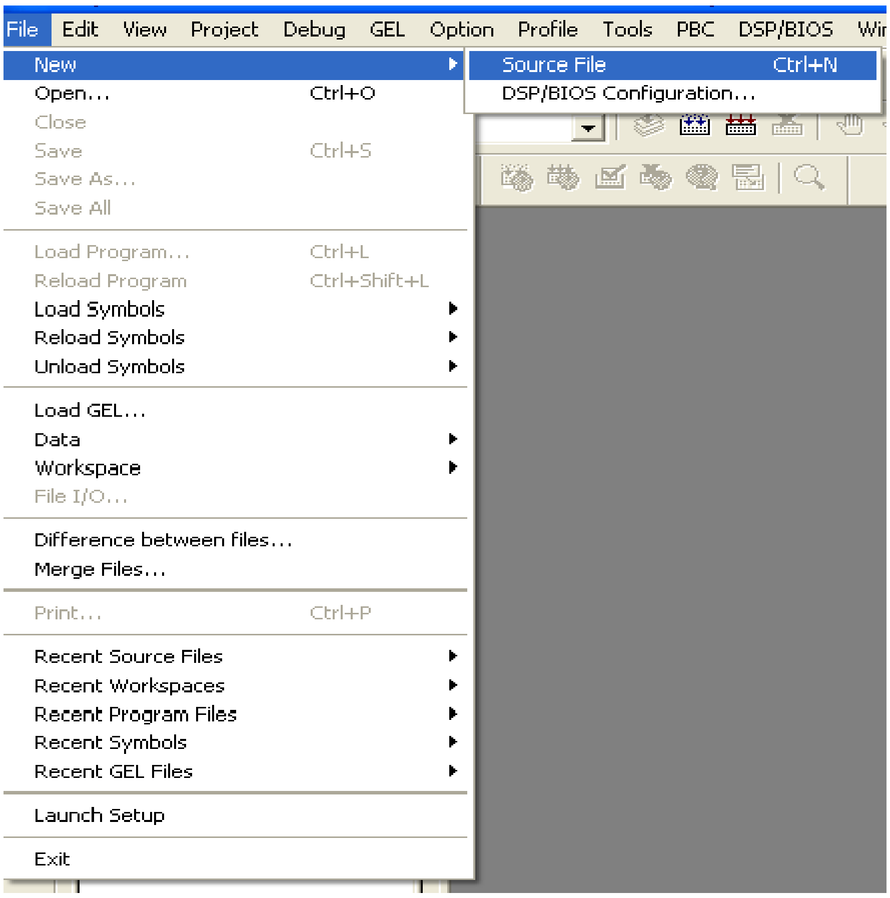
Location: C:\CCStudio_v3.1\MyPrcjects\

Project Type: Executable (.out)

Target TMS320C67XX

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

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



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,

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

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

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

Program:

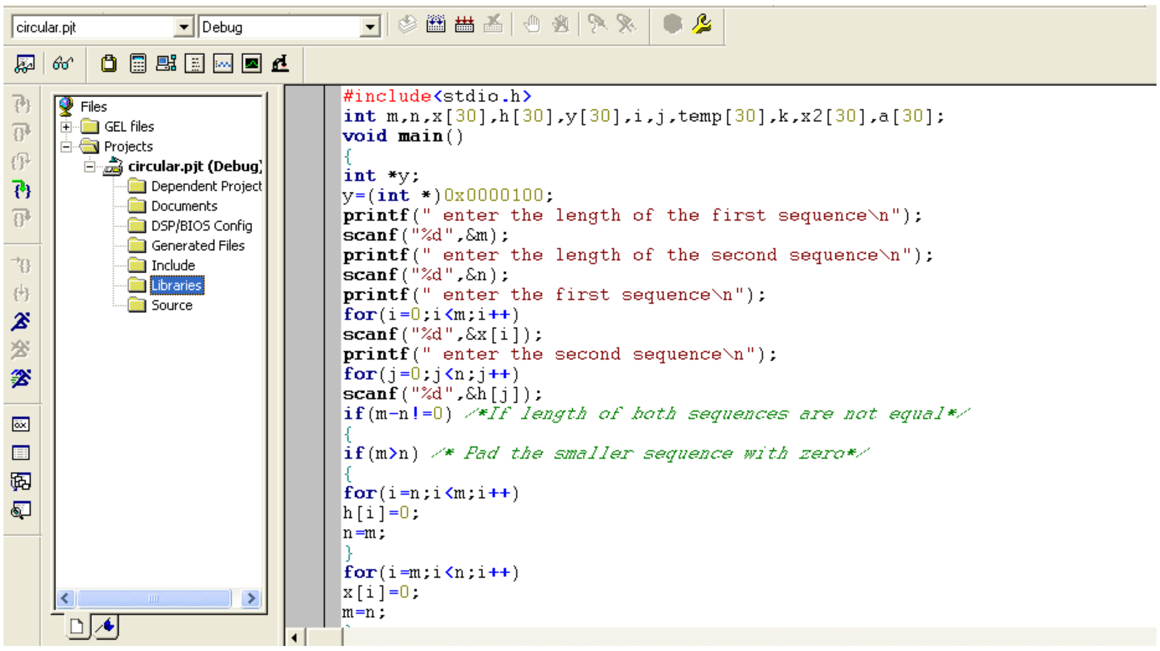
```
#include<stdio.h>

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;

/*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 ",y[i]);
```

}

Output:

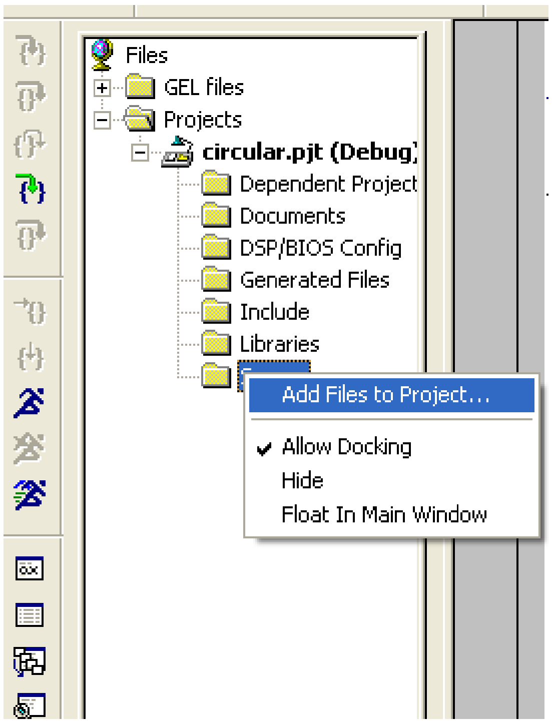
```
enter the length of the first sequence

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
```

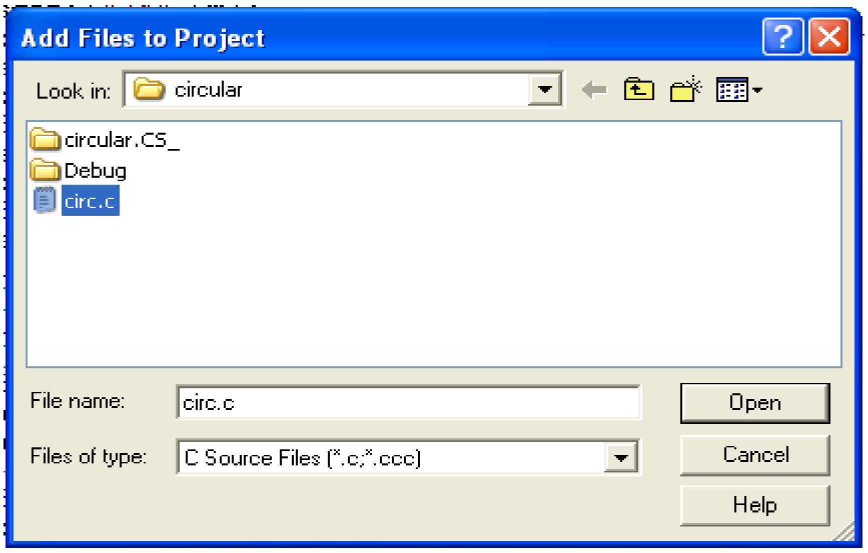
4. Enter the source code and save the file with **“.C”** extension.



```
#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;
```

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



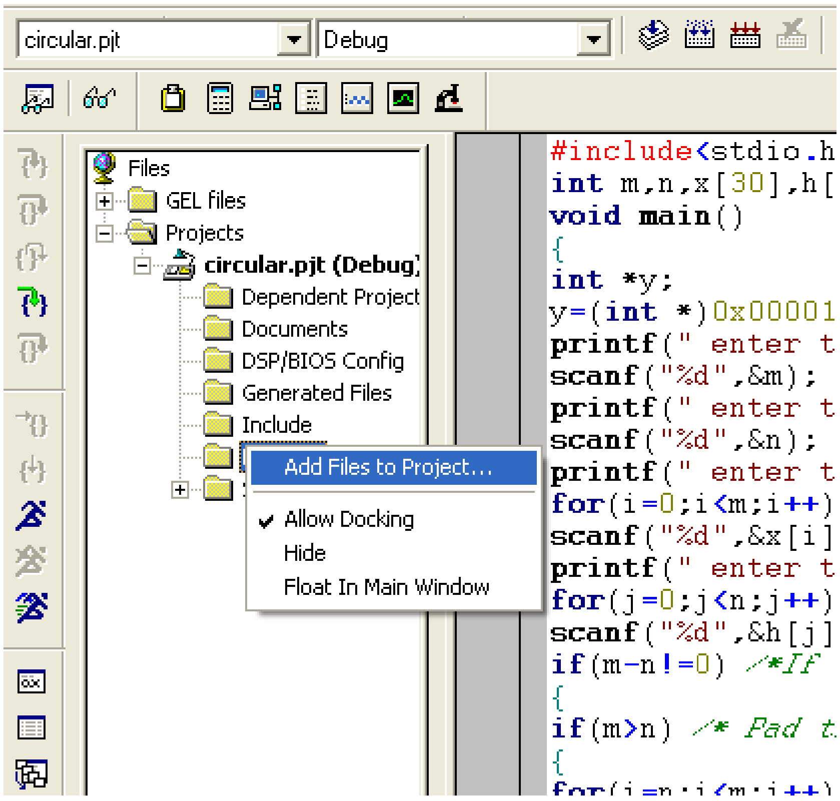


Look in: circular

File name: circ.c

Files of type: C Source Files (*.c;*.ccc)

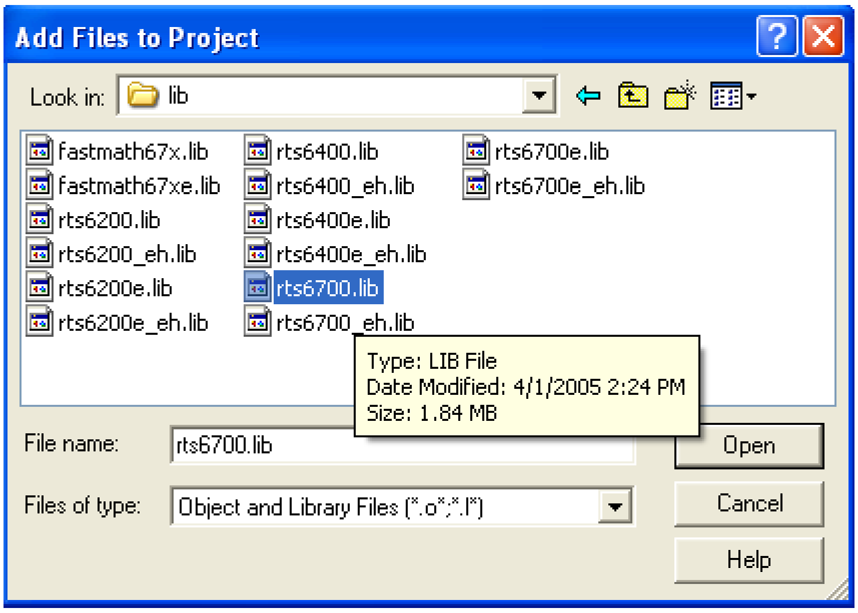
6. Right Click on libraries and select add files to Project.. and choose C:\CCStudio_v3.1\C6000\cgtools\lib\rts6700.lib and click open.



circular.pjt

Debug

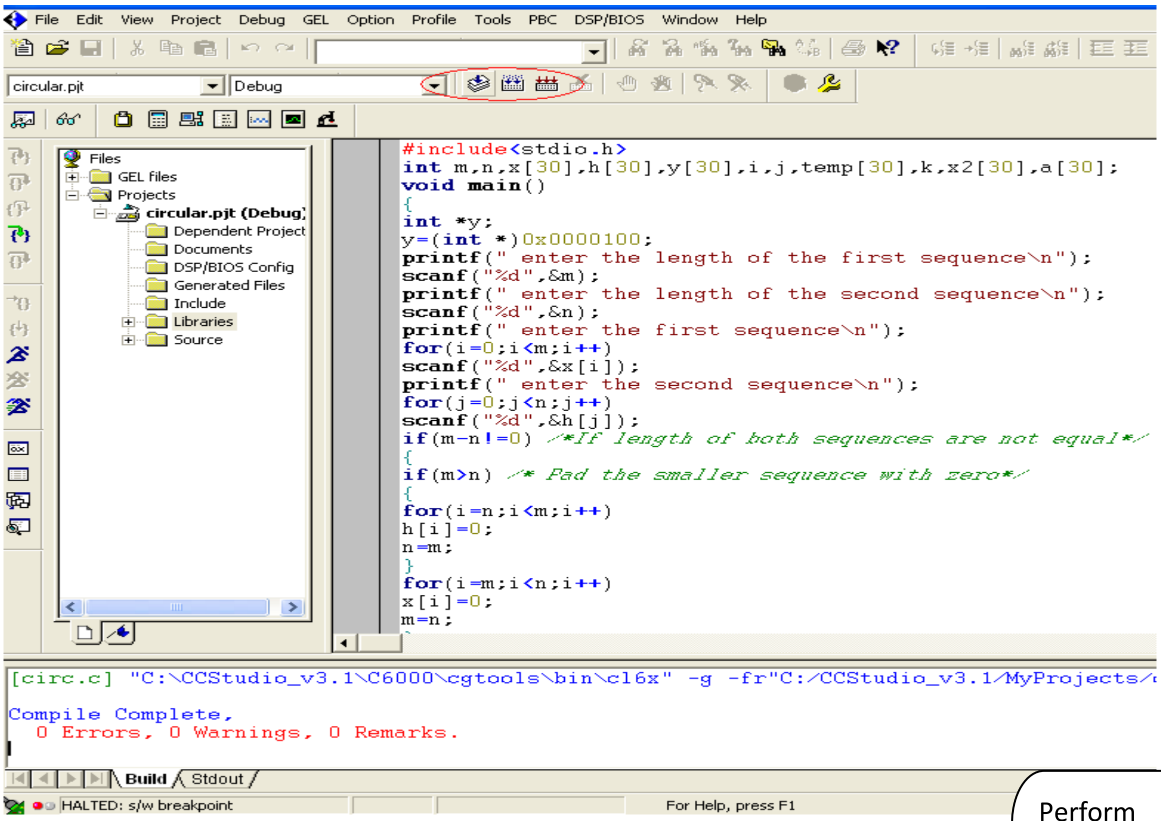
```
#include<stdio.h
int m,n,x[30],h[
void main()
{
int *y;
y=(int *)0x00001
printf(" enter t
scanf("%d",&m);
printf(" enter t
scanf("%d",&n);
printf(" enter t
for(i=0;i<m;i++)
scanf("%d",&x[i]
printf(" enter t
for(j=0;j<n;j++)
scanf("%d",&h[j]
if(m-n!=0) /*If
{
if(m>n) /* Pad t
{
for(i=n;i<m;i++)
```



7. a) Go to Project to Compile .

b) Go to Project to Build.

c) Go to Project to Rebuild All.

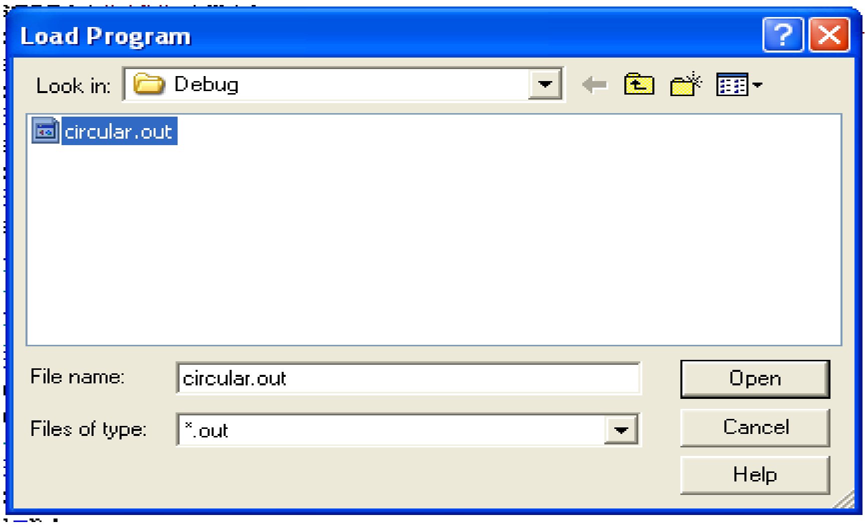


```
#include<stdio.h>
int m,n,x[30],h[30],y[30],i,j,temp[30],k,x2[30],a[30];
void main()
{
int *y;
y=(int *)0x0000100;
printf(" enter the length of the first sequence\n");
scanf("%d",&m);
printf(" enter the length of the second sequence\n");
scanf("%d",&n);
printf(" enter the first sequence\n");
for(i=0;i<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;
```

```
[circ.c] "C:\CCStudio_v3.1\C6000\cgtools\bin\cl6x" -g -fr"C:/CCStudio_v3.1/MyProjects/

Compile Complete,
  0 Errors, 0 Warnings, 0 Remarks.
```

Perform FFT and Plot the spectrum

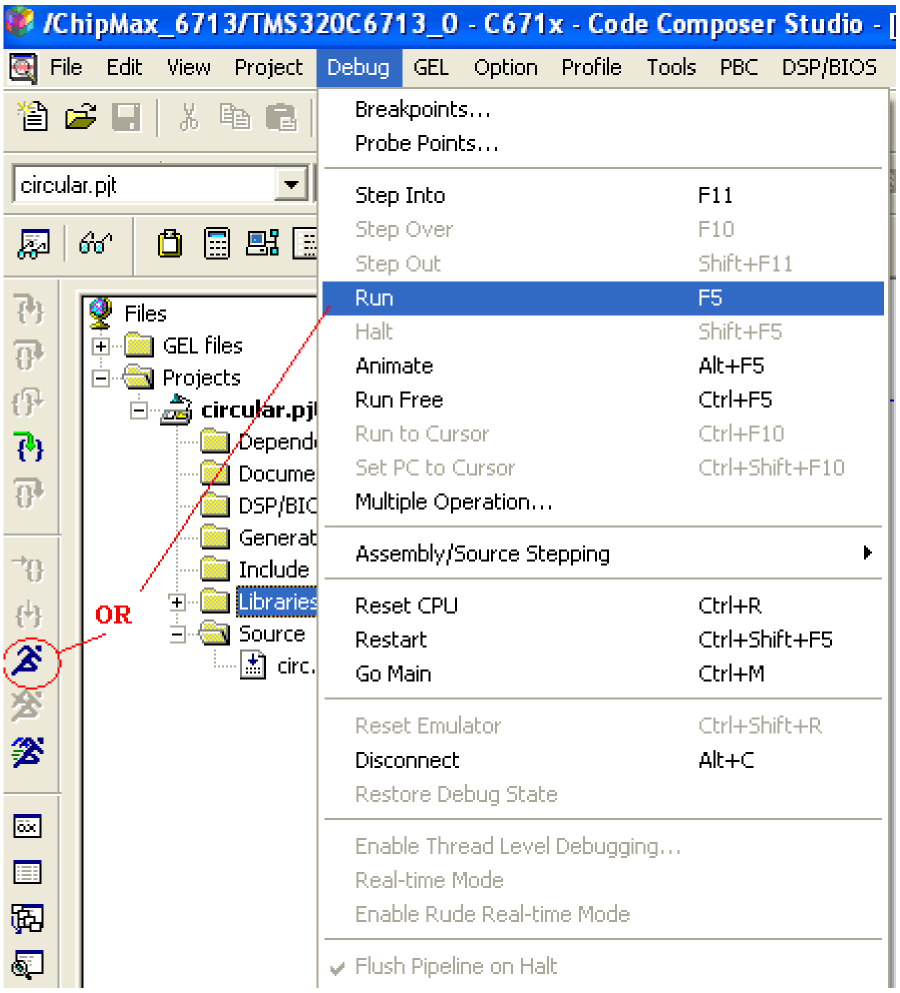
8. Go to file and load program and load **".out"** file into the board..



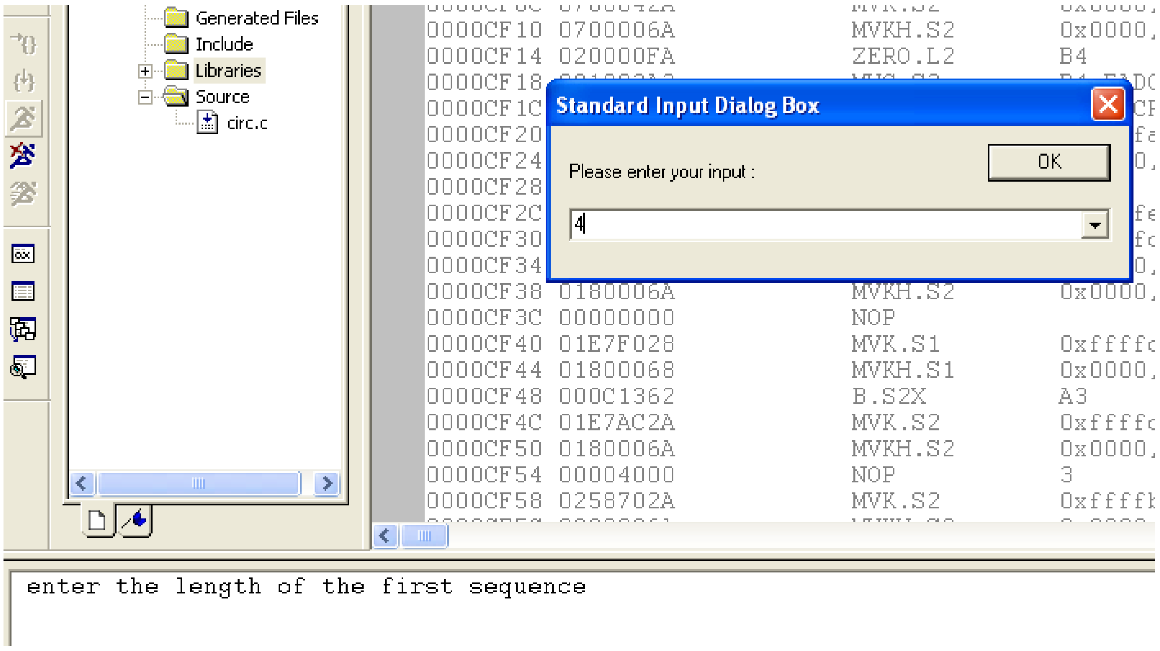
File name: circular.out

Files of type: *.out

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



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

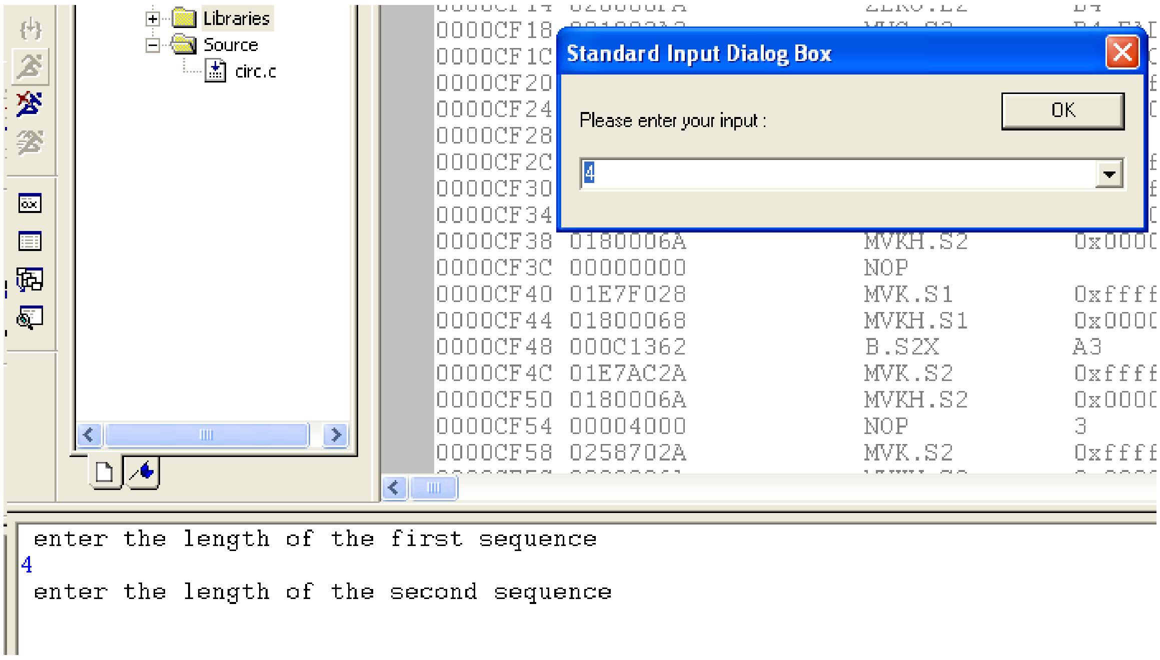


```
0000CF10  0700006A        MVKH.S2    0x0000,
0000CF14  020000FA        ZERO.L2    B4
0000CF18
0000CF1C
0000CF20
0000CF24
0000CF28
0000CF2C
0000CF30
0000CF34
0000CF38  0180006A        MVKH.S2    0x0000,
0000CF3C  00000000        NOP
0000CF40  01E7F028        MVK.S1     0xffffc
0000CF44  01800068        MVKH.S1    0x0000,
0000CF48  000C1362        B.S2X      A3
0000CF4C  01E7AC2A        MVK.S2     0xffffc
0000CF50  0180006A        MVKH.S2    0x0000,
0000CF54  00004000        NOP        3
0000CF58  0258702A        MVK.S2     0xffffk
```

Please enter your input :

4

```
enter the length of the first sequence
```

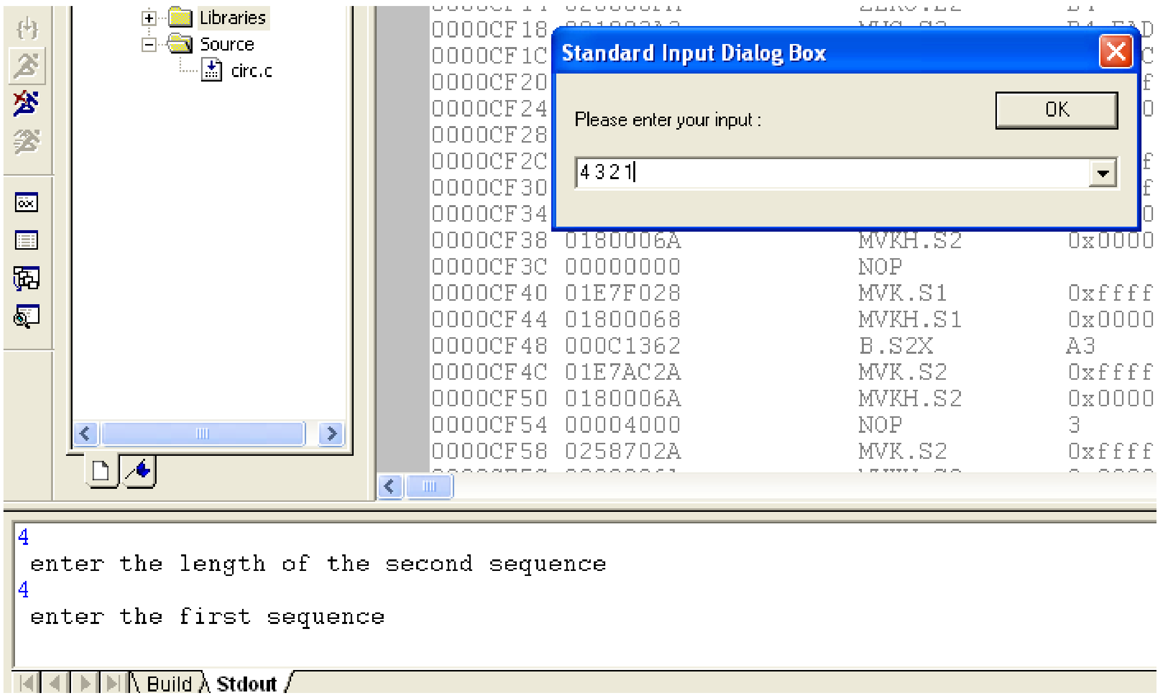


```
0000CF18
0000CF1C
0000CF20
0000CF24
0000CF28
0000CF2C
0000CF30
0000CF34
0000CF38  0180006A        MVKH.S2    0x000C
0000CF3C  00000000        NOP
0000CF40  01E7F028        MVK.S1     0xffff
0000CF44  01800068        MVKH.S1    0x000C
0000CF48  000C1362        B.S2X      A3
0000CF4C  01E7AC2A        MVK.S2     0xffff
0000CF50  0180006A        MVKH.S2    0x000C
0000CF54  00004000        NOP        3
0000CF58  0258702A        MVK.S2     0xffff
```

Please enter your input :

4

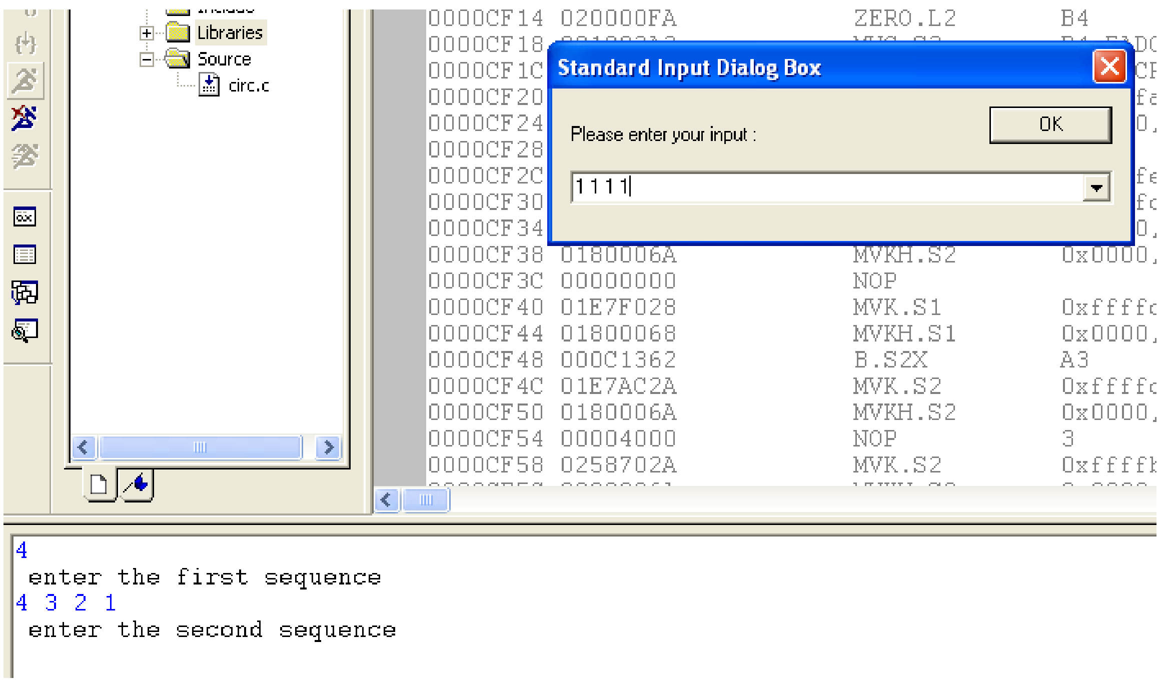
```
enter the length of the first sequence
4
enter the length of the second sequence
```



```
0000CF38  0180006A    MVKH.S2    0x0000
0000CF3C  00000000    NOP
0000CF40  01E7F028    MVK.S1     0xffff
0000CF44  01800068    MVKH.S1    0x0000
0000CF48  000C1362    B.S2X      A3
0000CF4C  01E7AC2A    MVK.S2     0xffff
0000CF50  0180006A    MVKH.S2    0x0000
0000CF54  00004000    NOP        3
0000CF58  0258702A    MVK.S2     0xffff
```

```
4 3 2 1
```

```
4
 enter the length of the second sequence
4
 enter the first sequence
```

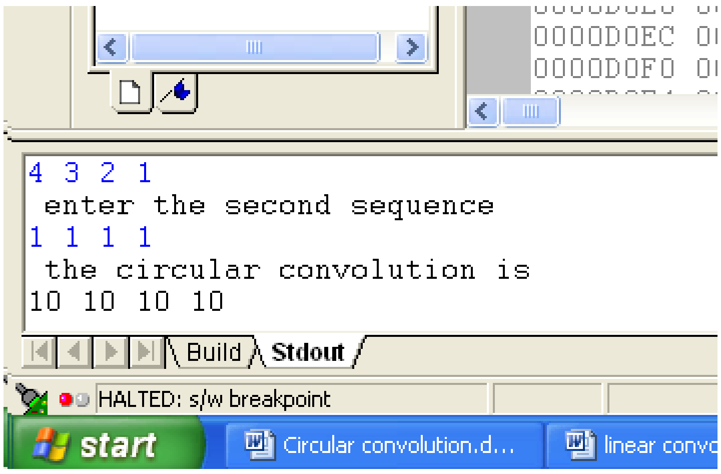


```
0000CF14  020000FA    ZERO.L2    B4
0000CF38  0180006A    MVKH.S2    0x0000,
0000CF3C  00000000    NOP
0000CF40  01E7F028    MVK.S1     0xffffc
0000CF44  01800068    MVKH.S1    0x0000,
0000CF48  000C1362    B.S2X      A3
0000CF4C  01E7AC2A    MVK.S2     0xffffc
0000CF50  0180006A    MVKH.S2    0x0000,
0000CF54  00004000    NOP        3
0000CF58  0258702A    MVK.S2     0xffffk
```

```
1 1 1 1
```

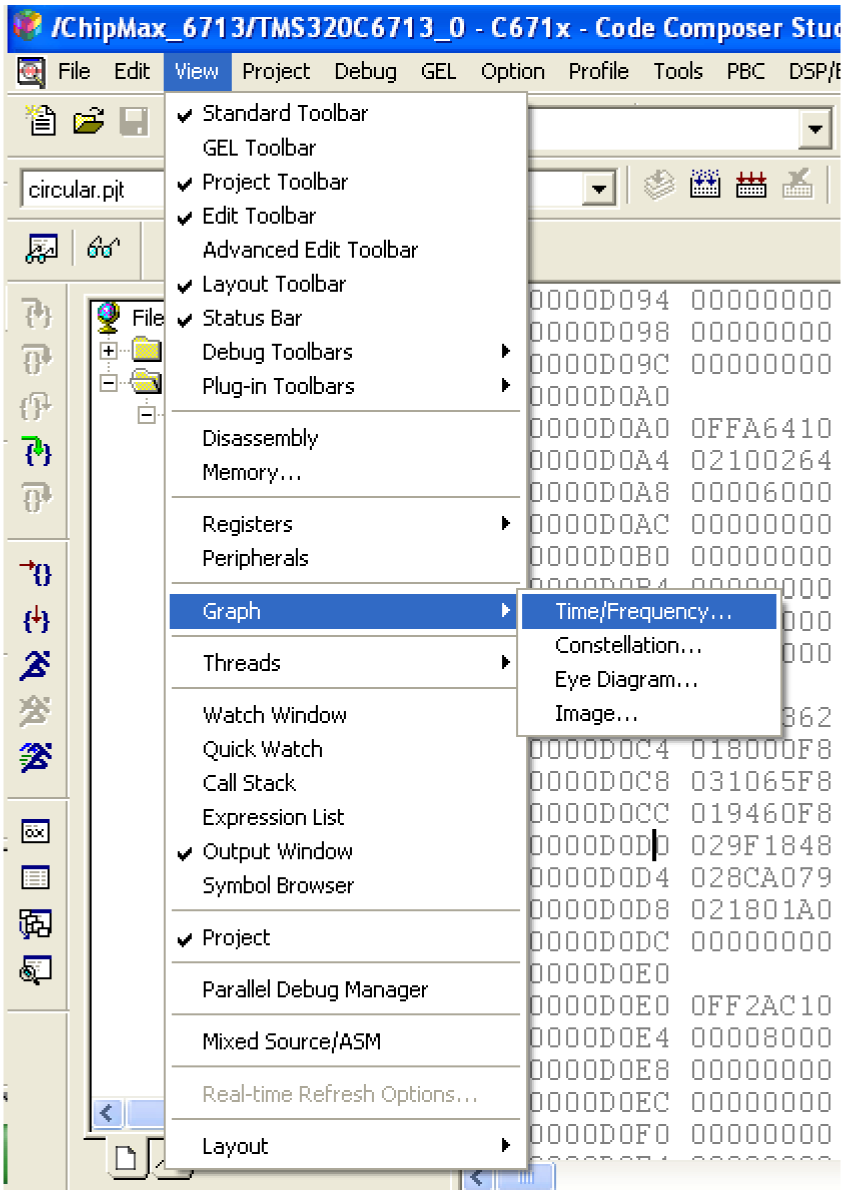
```
4
 enter the first sequence
4 3 2 1
 enter the second sequence
```

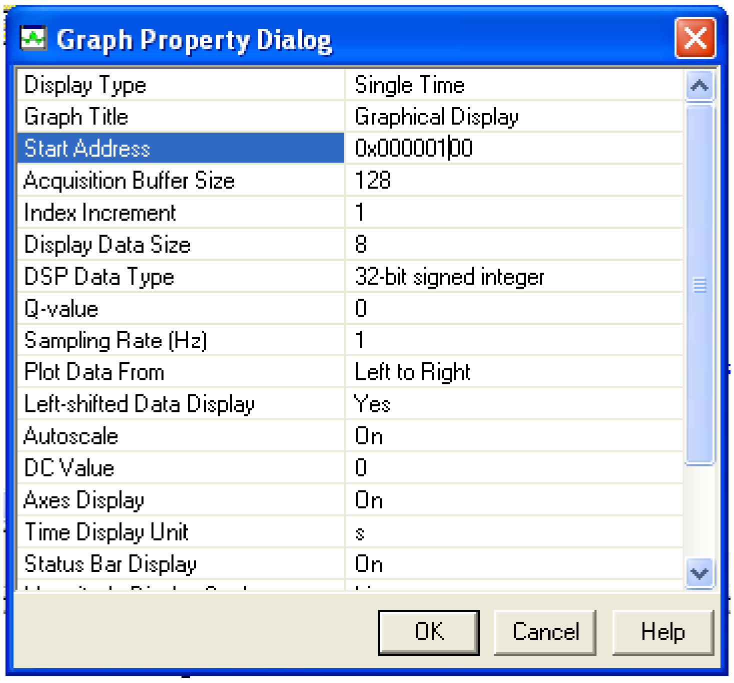
The corresponding output will be shown on the output window as shown below



```
4 3 2 1
 enter the second sequence
1 1 1 1
 the circular convolution is
10 10 10 10
```

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.

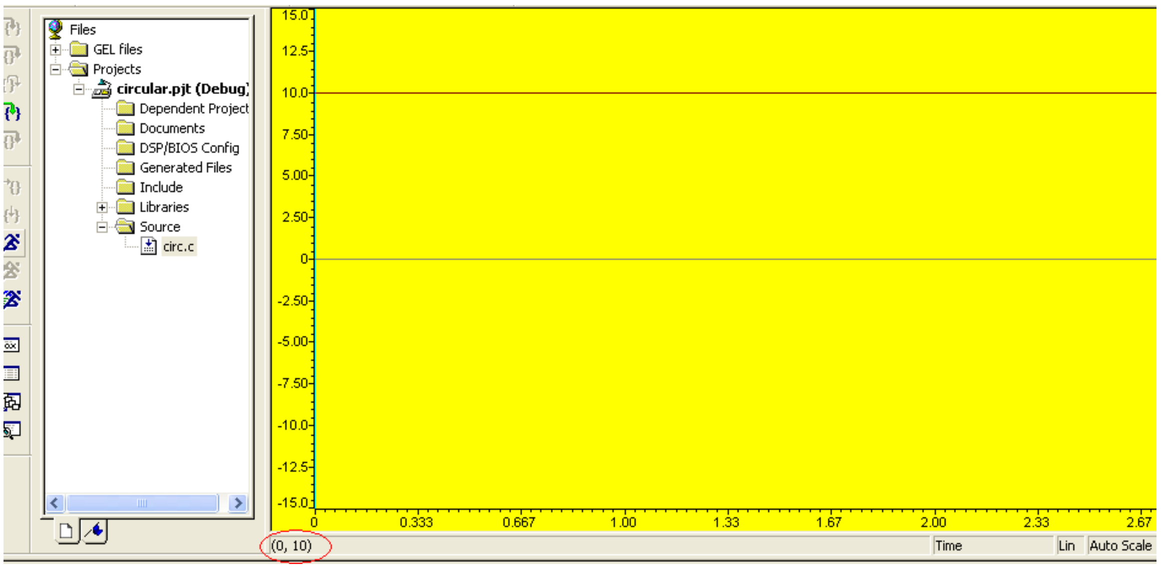




Graph Property Dialog

Display Type	Single Time
Graph Title	Graphical Display
Start Address	0x00000100
Acquisition Buffer Size	128
Index Increment	1
Display Data Size	8
DSP Data Type	32-bit signed integer
Q-value	0
Sampling Rate (Hz)	1
Plot Data From	Left to Right
Left-shifted Data Display	Yes
Autoscale	On
DC Value	0
Axes Display	On
Time Display Unit	s
Status Bar Display	On

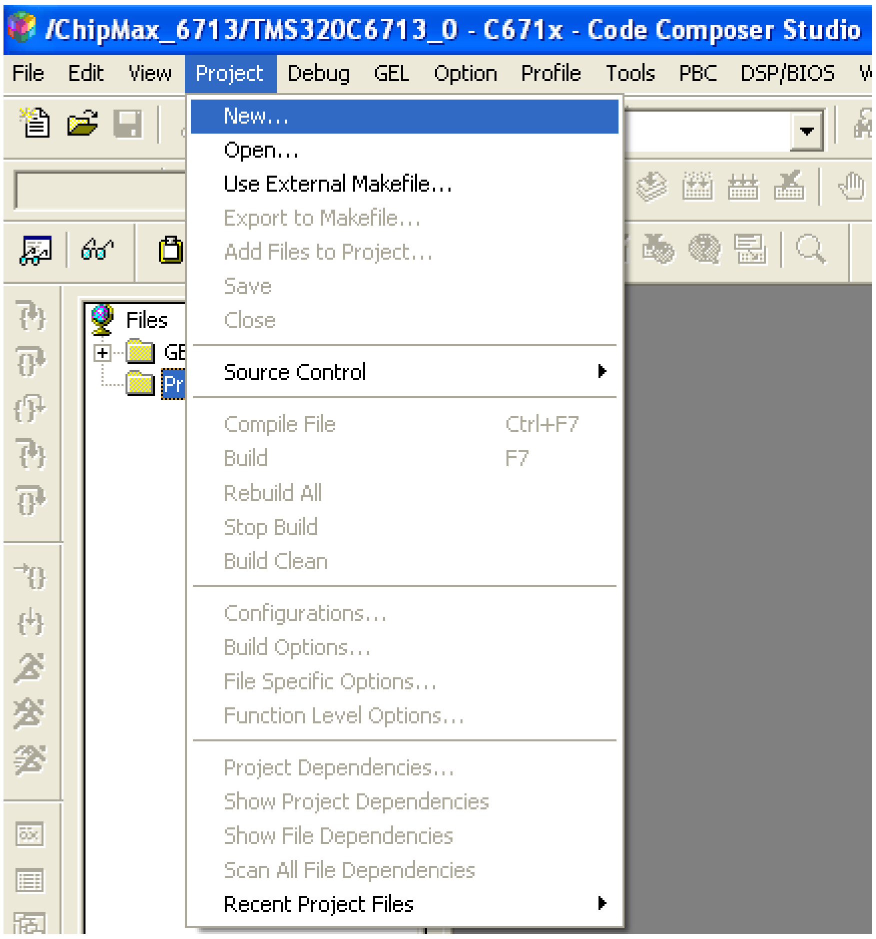
12. Green line is to choose the point, Value at the point can be seen (Highlighted by circle at the left corner).



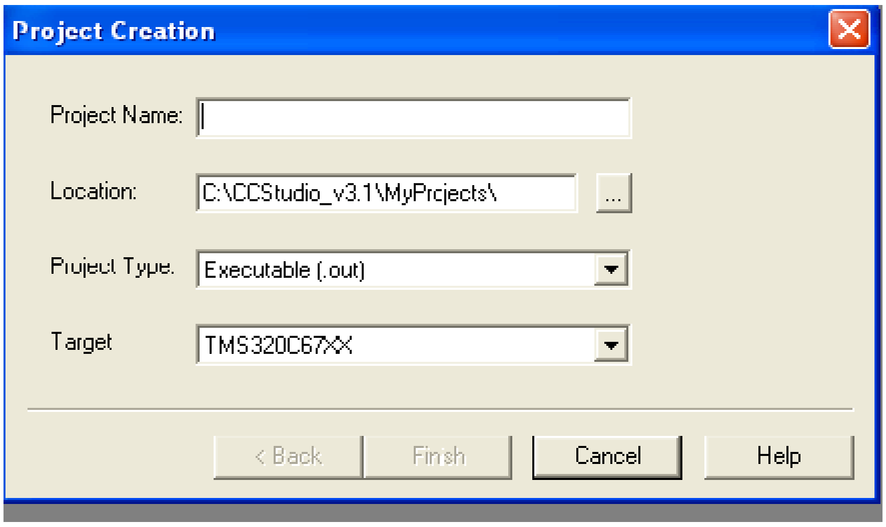
Experiment 2: N-Point DFT

Procedure to create new Project:

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



2. Give project name and click on finish.


Project Creation

Project Name:

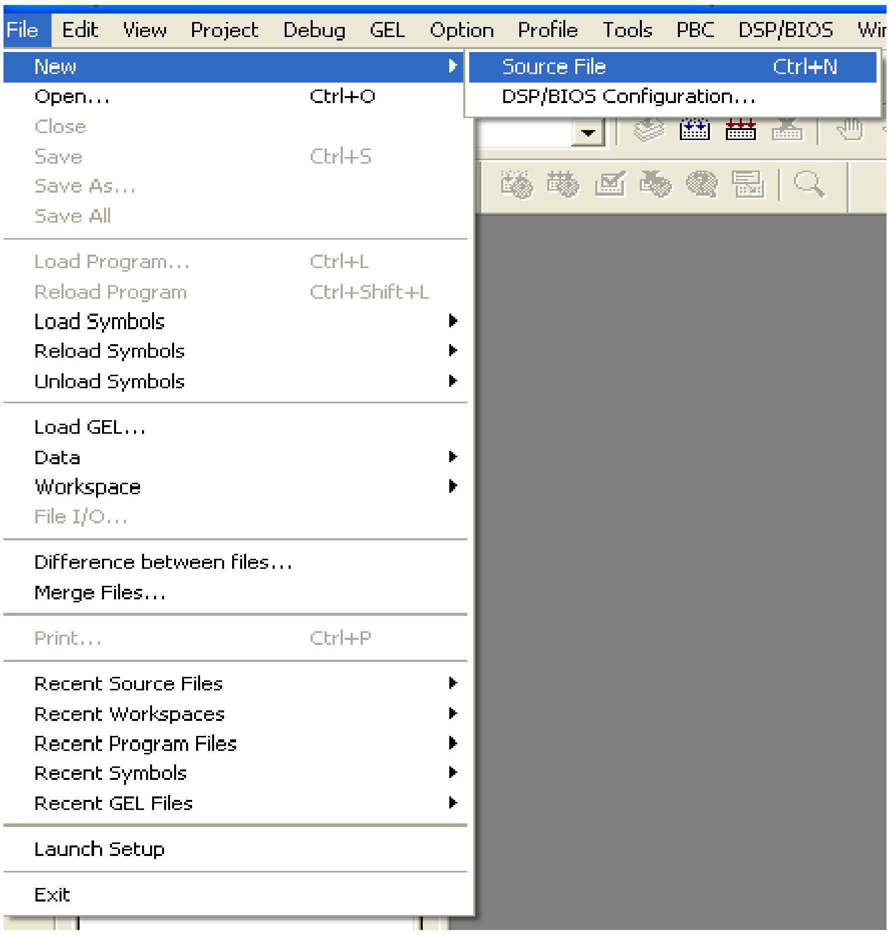
Location: C:\CCStudio_v3.1\MyPrcjects\

Project Type: Executable (.out)

Target TMS320C67XX

(**<u>Note:</u>** Location must be c:\CCStudio_v3.1\MyProjects).

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



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:

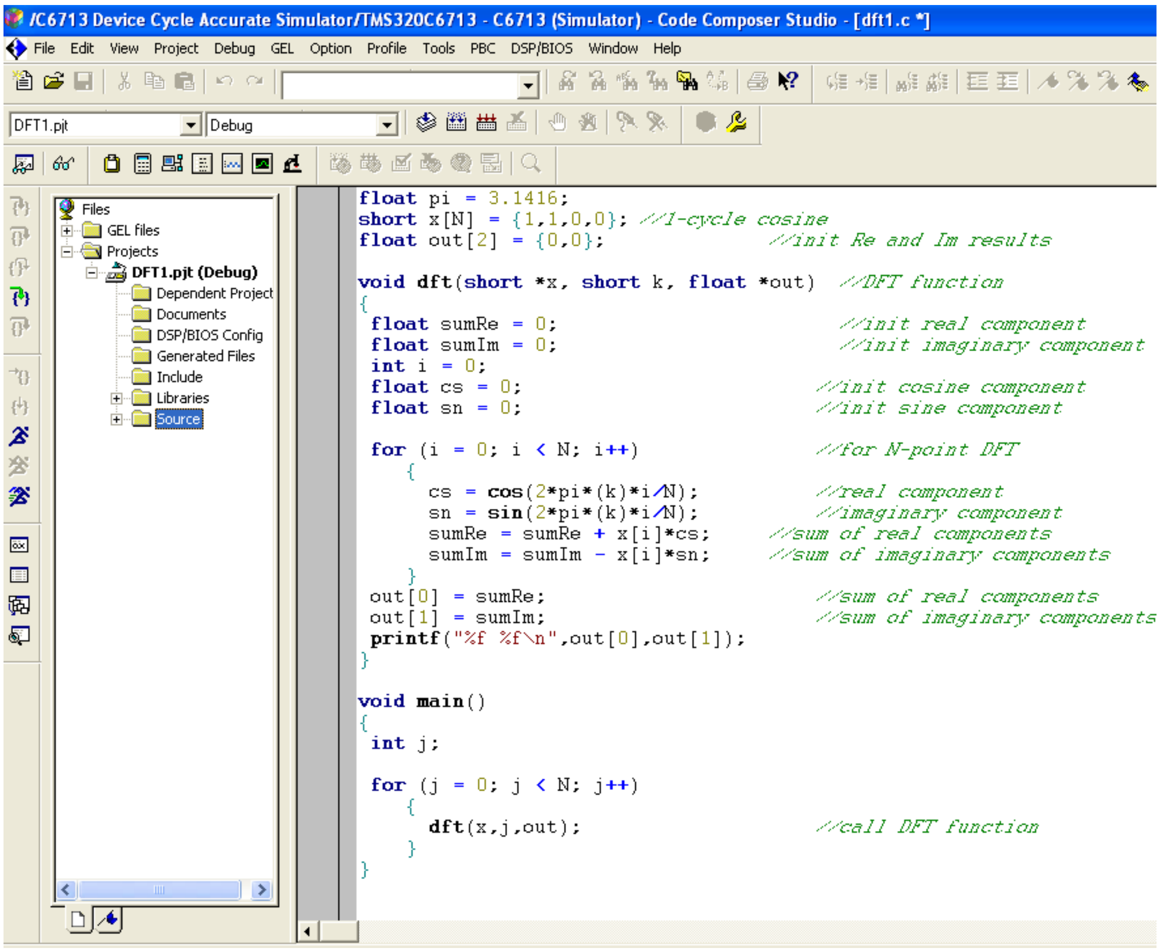
```
//DFT of N-point from lookup table. Output from watch window

#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};                               //initialize Re and Im results


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
out[1] = sumIm;                             //sum of imaginary components
printf("%f %f\n",out[0],out[1]);
}

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

Output:

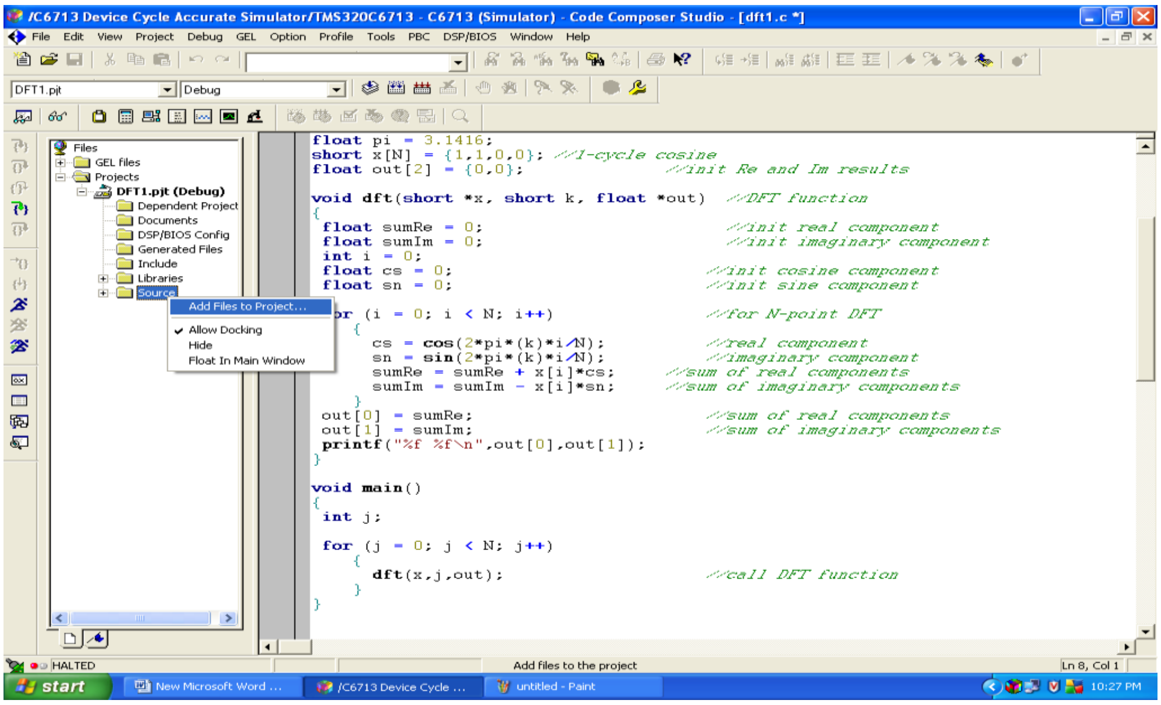
```
2.000000 0.000000

0.999996 -1.000000

0.000000 0.000007

1.000011 1.000000
```

4. Enter the source code and save the file with “.C” extension.


/C6713 Device Cycle Accurate Simulator/TMS320C6713 - C6713 (Simulator) - Code Composer Studio - [dft1.c *]
File Edit View Project Debug GEL Option Profile Tools PBC DSP/BIOS Window Help
DFT1.pjt
Debug
Files
GEL files
Projects
DFT1.pjt (Debug)
Dependent Project
Documents
DSP/BIOS Config
Generated Files
Include
Libraries
Source

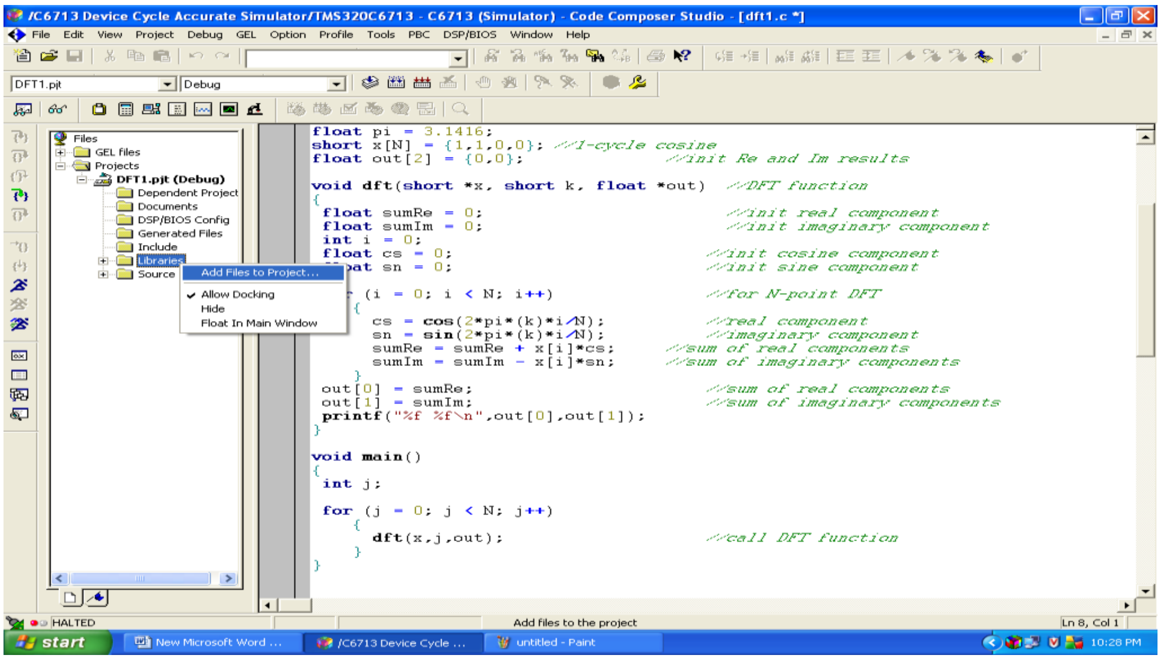
```
float pi = 3.1416;
short x[N] = {1,1,0,0}; //1-cycle cosine
float out[2] = {0,0};                //init Re and Im results

void dft(short *x, short k, float *out)  //DFT function
{
 float sumRe = 0;                         //init real component
 float sumIm = 0;                         //init imaginary component
 int i = 0;
 float cs = 0;                          //init cosine component
 float sn = 0;                          //init 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
 out[1] = sumIm;                        //sum of imaginary components
 printf("%f %f\n",out[0],out[1]);
}

void main()
{
 int j;

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

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

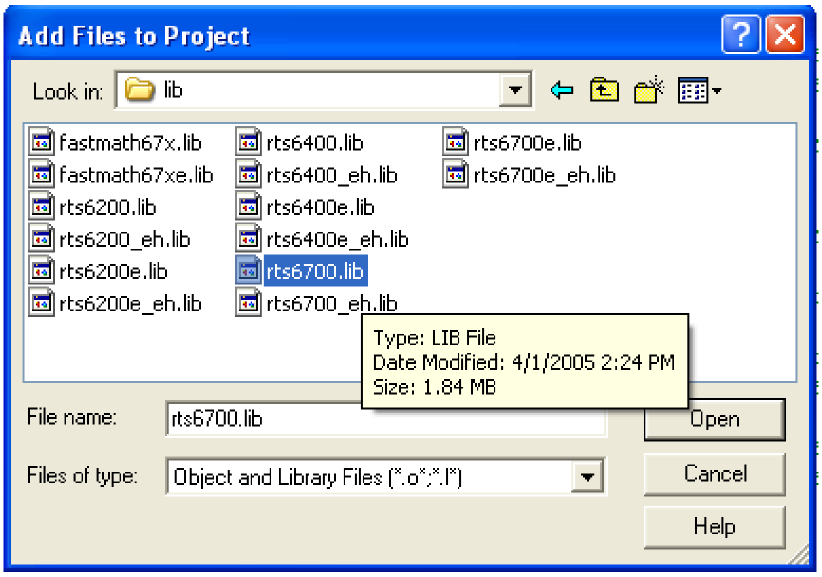


```
float pi = 3.1416;
short x[N] = {1,1,0,0}; //1-cycle cosine
float out[2] = {0,0};              //init Re and Im results

void dft(short *x, short k, float *out)  //DFT function
{
 float sumRe = 0;                  //init real component
 float sumIm = 0;                  //init imaginary component
 int i = 0;
 float cs = 0;                     //init cosine component
 float sn = 0;                     //init 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
 out[1] = sumIm;                   //sum of imaginary components
 printf("%f %f\n",out[0],out[1]);
}

void main()
{
 int j;

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

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.



```
float pi = 3.1416;
short x[N] = {1,1,0,0}; //1-cycle cosine
float out[2] = {0,0};              //init Re and Im results

void dft(short *x, short k, float *out)  //DFT function
{
 float sumRe = 0;                  //init real component
 float sumIm = 0;                  //init imaginary component
 int i = 0;
 float cs = 0;                     //init cosine component
 float sn = 0;                     //init 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
 out[1] = sumIm;                   //sum of imaginary components
 printf("%f %f\n",out[0],out[1]);
}

void main()
{
 int j;

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


Add Files to Project

Look in: lib

fastmath67x.lib
fastmath67xe.lib
rts6200.lib
rts6200_eh.lib
rts6200e.lib
rts6200e_eh.lib
rts6400.lib
rts6400_eh.lib
rts6400e.lib
rts6400e_eh.lib
rts6700.lib
rts6700_eh.lib
rts6700e.lib
rts6700e_eh.lib

Type: LIB File
Date Modified: 4/1/2005 2:24 PM
Size: 1.84 MB

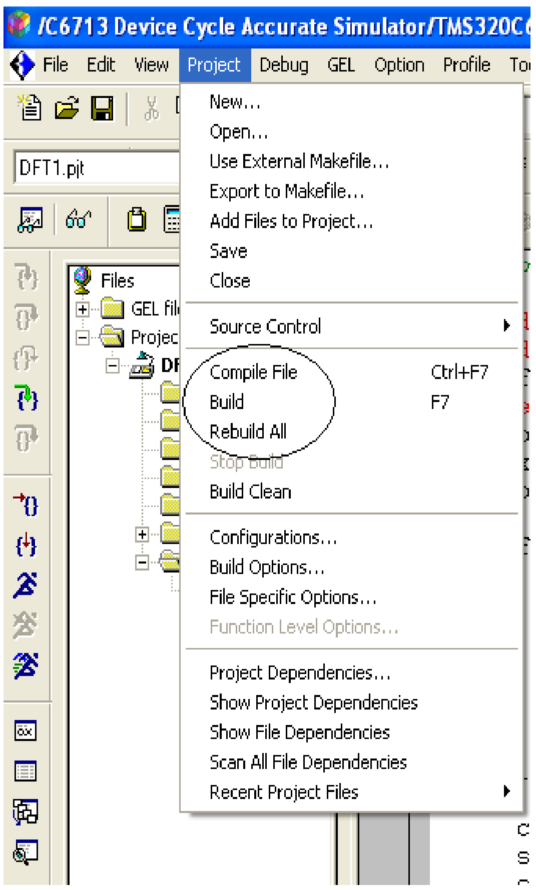
File name: rts6700.lib

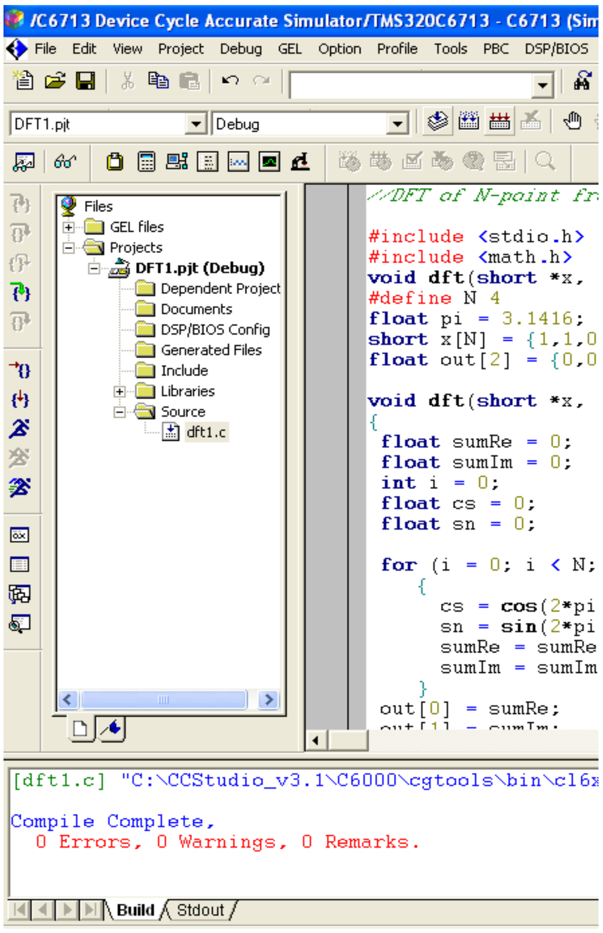
Files of type: Object and Library Files (*.o*;*.l*)

7. a) Go to Project to Compile .

b) Go to Project to Build.

c) Go to Project to Rebuild All.



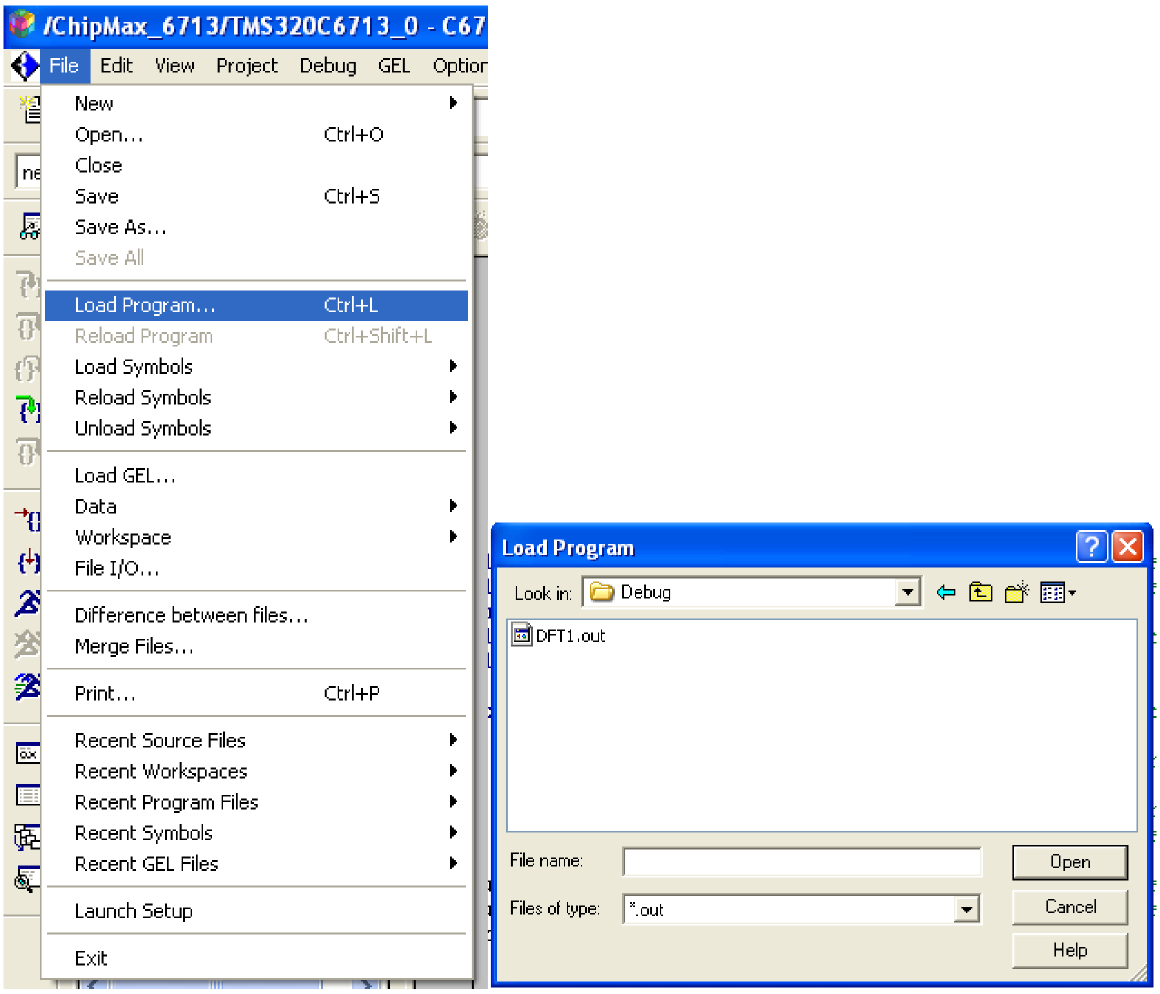


```
//DFT of N-point fr

#include <stdio.h>
#include <math.h>
void dft(short *x,
#define N 4
float pi = 3.1416;
short x[N] = {1,1,0
float out[2] = {0,0

void dft(short *x,
{
 float sumRe = 0;
 float sumIm = 0;
 int i = 0;
 float cs = 0;
 float sn = 0;

 for (i = 0; i < N;
     {
        cs = cos(2*pi
        sn = sin(2*pi
        sumRe = sumRe
        sumIm = sumIm
     }
 out[0] = sumRe;
```

```
[dft1.c] "C:\CCStudio_v3.1\C6000\cgtools\bin\cl6x

Compile Complete,
  0 Errors, 0 Warnings, 0 Remarks.
```

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

Change the sampling rate and number of



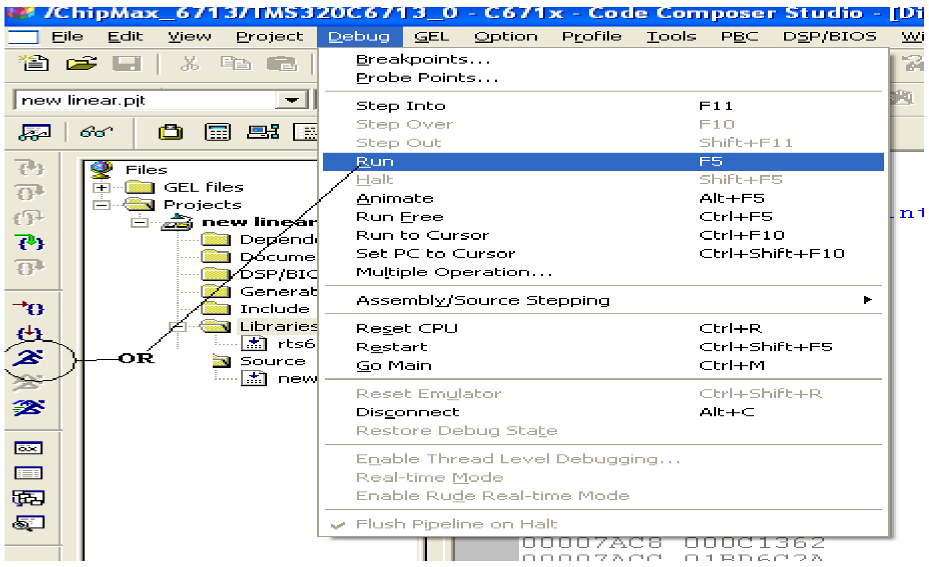
Look in: Debug

DFT1.out

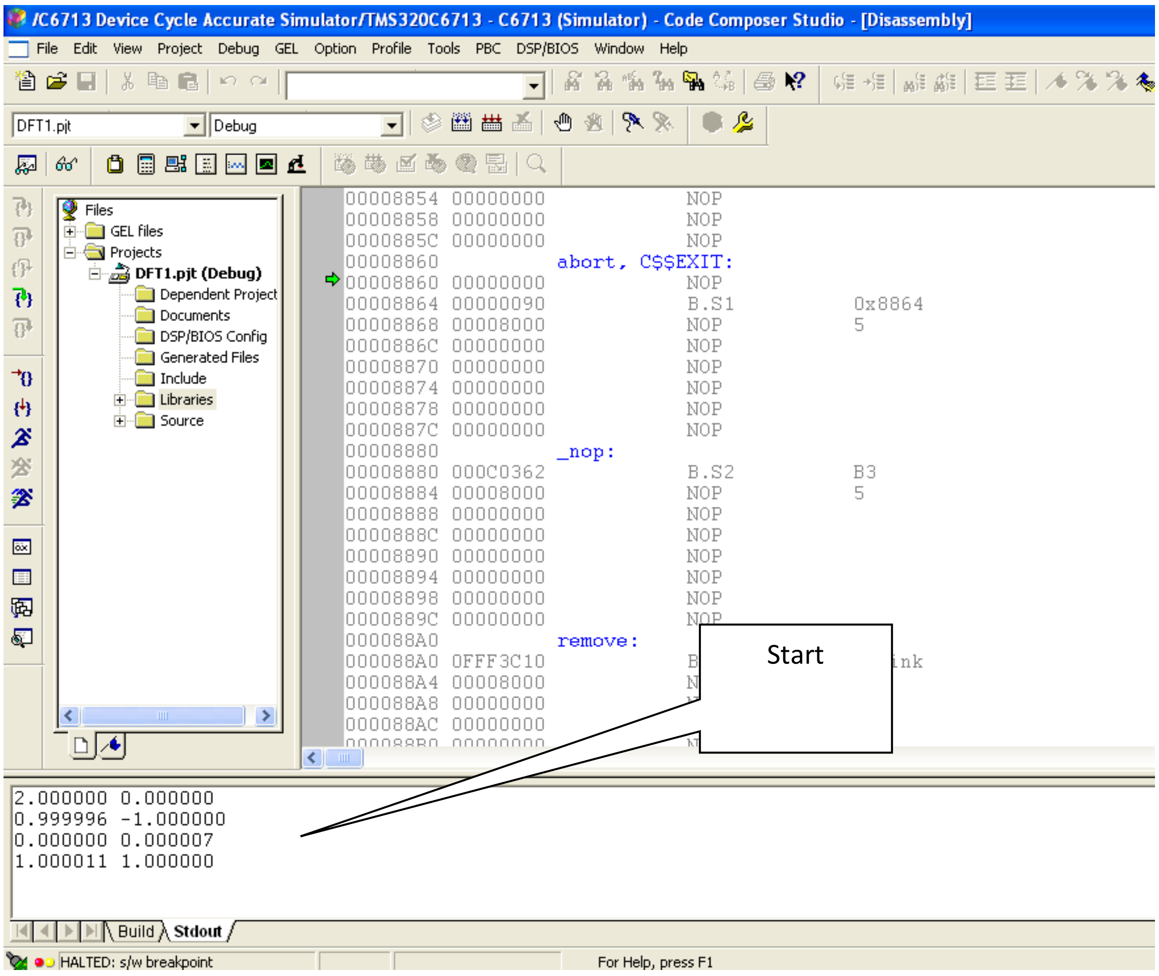
File name:

Files of type: *.out

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



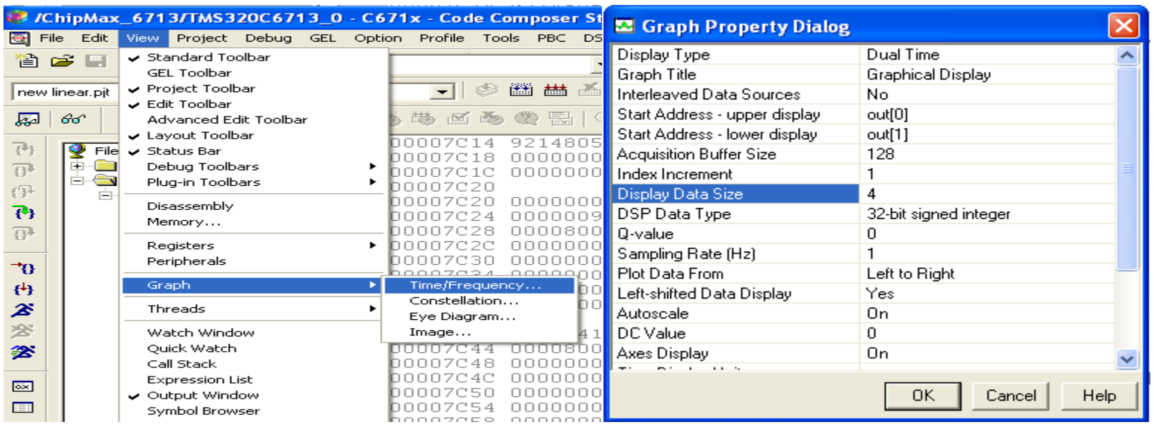
10. Observe the output in output window.


/C6713 Device Cycle Accurate Simulator/TMS320C6713 - C6713 (Simulator) - Code Composer Studio - [Disassembly]
File Edit View Project Debug GEL Option Profile Tools PBC DSP/BIOS Window Help
DFT1.pjt
Debug
Files
GEL files
Projects
DFT1.pjt (Debug)
Dependent Project
Documents
DSP/BIOS Config
Generated Files
Include
Libraries
Source
Start
Build
Stdout
HALTED: s/w breakpoint
For Help, press F1

```
00008854 00000000          NOP
00008858 00000000          NOP
0000885C 00000000          NOP
00008860          abort, C$$EXIT:
00008860 00000000          NOP
00008864 00000090          B.S1        0x8864
00008868 00008000          NOP         5
0000886C 00000000          NOP
00008870 00000000          NOP
00008874 00000000          NOP
00008878 00000000          NOP
0000887C 00000000          NOP
00008880          _nop:
00008880 000C0362          B.S2        B3
00008884 00008000          NOP         5
00008888 00000000          NOP
0000888C 00000000          NOP
00008890 00000000          NOP
00008894 00000000          NOP
00008898 00000000          NOP
0000889C 00000000          NOP
000088A0          remove:
000088A0 0FFF3C10          B           ink
000088A4 00008000          N
000088A8 00000000
000088AC 00000000
```

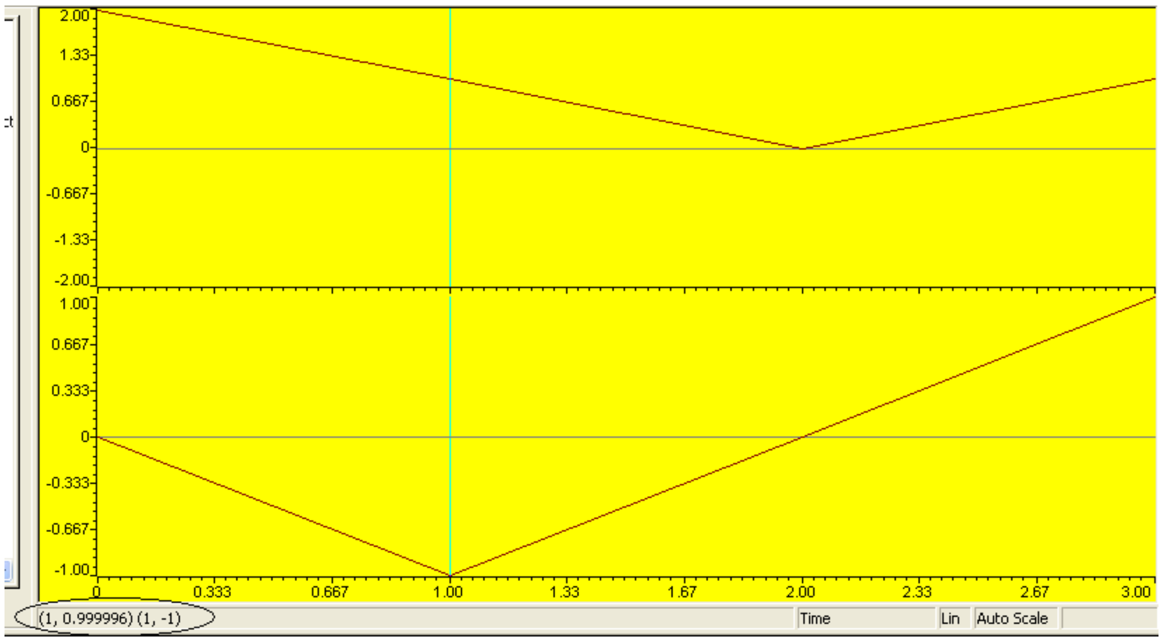
```
2.000000 0.000000
0.999996 -1.000000
0.000000 0.000007
1.000011 1.000000
```

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.



Display Type	Dual Time
Graph Title	Graphical Display
Interleaved Data Sources	No
Start Address - upper display	out[0]
Start Address - lower display	out[1]
Acquisition Buffer Size	128
Index Increment	1
Display Data Size	4
DSP Data Type	32-bit signed integer
Q-value	0
Sampling Rate (Hz)	1
Plot Data From	Left to Right
Left-shifted Data Display	Yes
Autoscale	On
DC Value	0
Axes Display	On

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

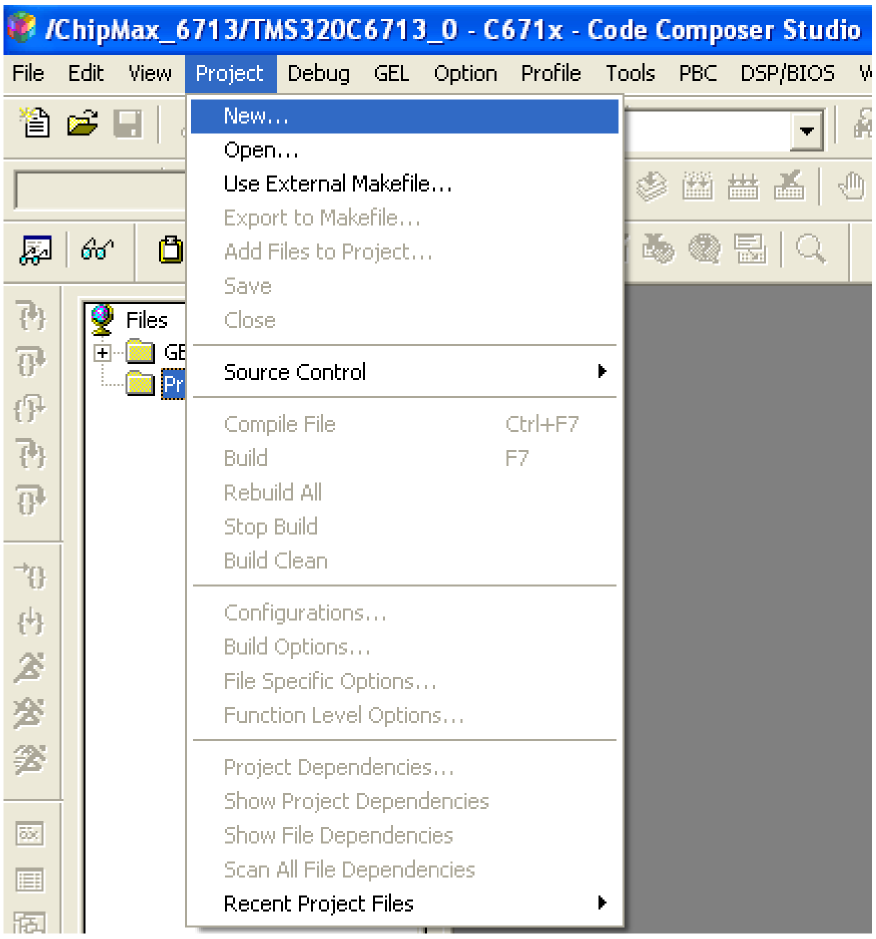

2.00
1.33
0.667
0
-0.667
-1.33
-2.00
1.00
0.667
0.333
0
-0.333
-0.667
-1.00
0
0.333
0.667
1.00
1.33
1.67
2.00
2.33
2.67
3.00
Time
Lin
Auto Scale

(1, 0.999996) (1, -1)

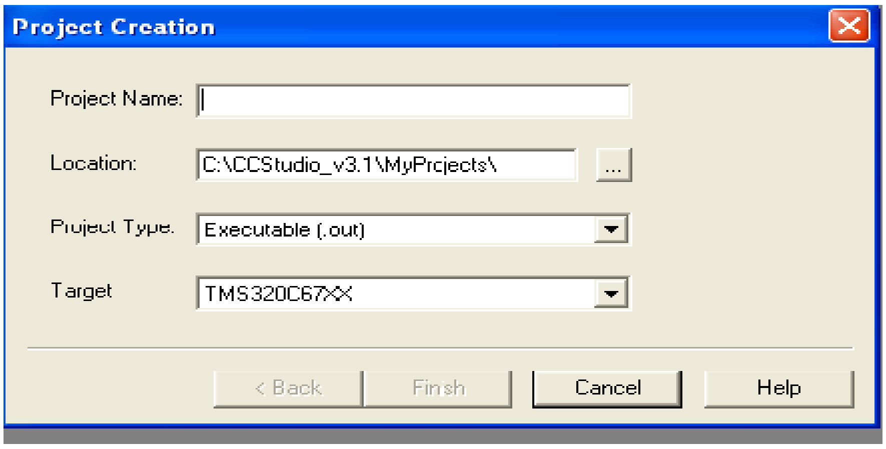
Experiment 3: Realizing FIR Filter

Procedure to create new Project:

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



2. Give project name and click on finish.


Project Creation

Project Name:

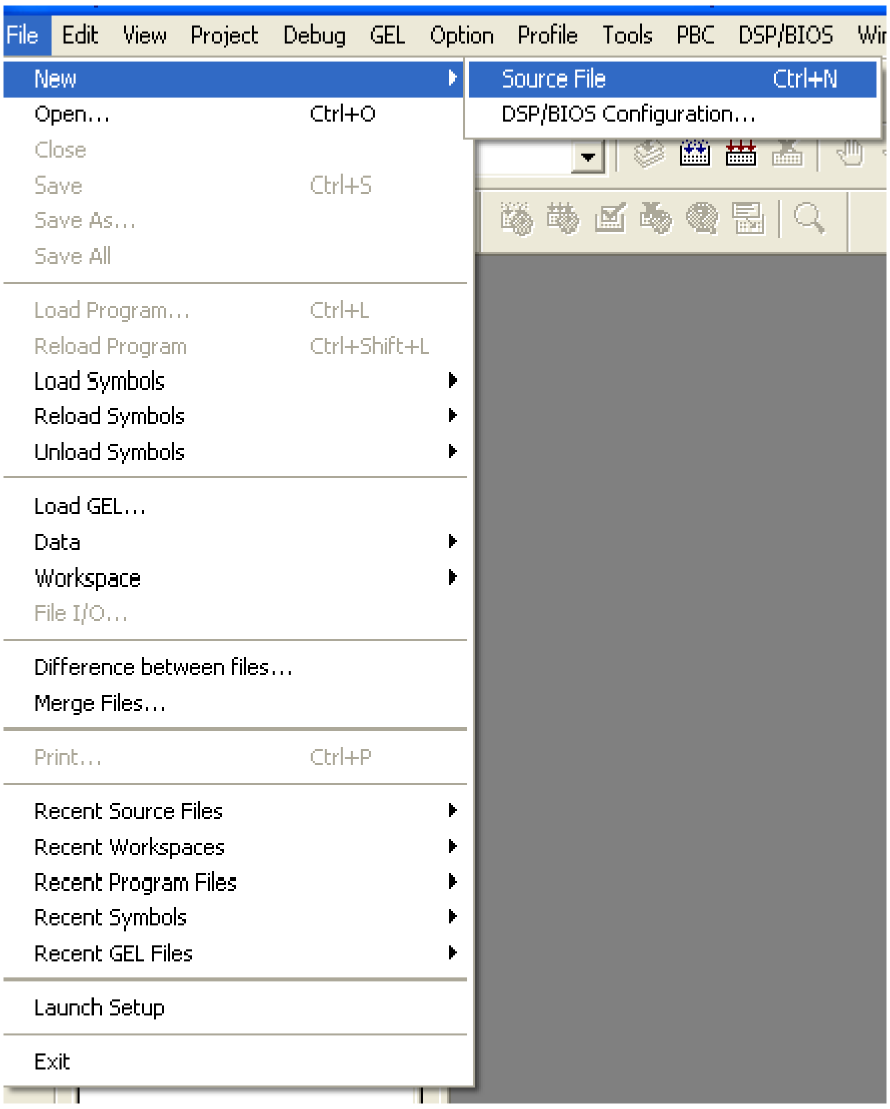
Location: C:\CCStudio_v3.1\MyProjects\

Project Type: Executable (.out)

Target: TMS320C67XX

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

3. Click on File and select New and then Source File, To write the Source Code.



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 $w(n) = w(-n) \neq 0 \quad \text{for } |n| \leq [(N-1)/2]$

$= 0 \qquad \text{for } |n| > [(N-1)/2]$

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

$$h(n) = h_d(n)\, w(n) \quad \text{for} \quad |n| \le [(N-1)/2]$$

$$= 0 \quad \text{for} \quad |n| > [(N-1)/2]$$

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:

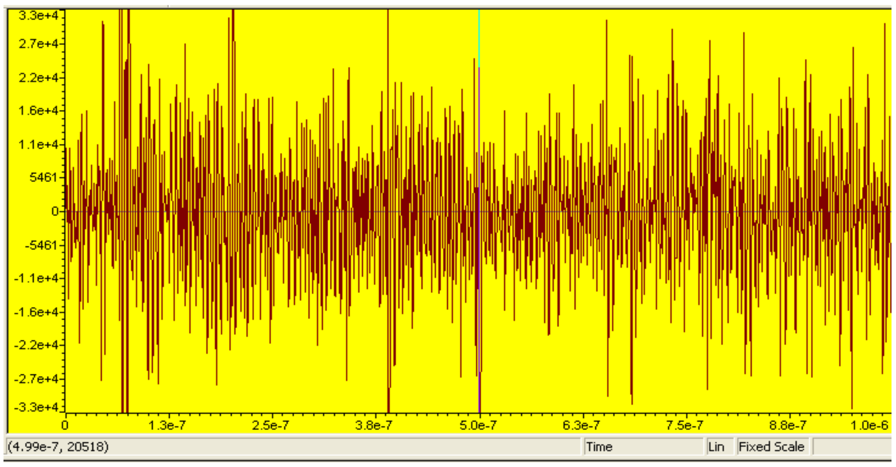
```
#include <stdio.h>

#include "c6713dsk.h"

#include "master.h"

#include "aic23cfg.h"

#include "dsk6713_aic23.h"

#include <std.h>

#include <swi.h>

#include <log.h>

#include <c6x.h>

#include <csl.h>

#include <csl_mcbsp.h>
```

```
/* Length of sine wave table */

#define SINE_TABLE_SIZE  48


// Delta

/*float filter_Coeff[] ={0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,

0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,

0.00,0.00,0.00,0.00,0.00,0.00,1.00,0.00};*/


// Low Pass Filter

/*float  filter_Coeff[]  ={2.715297634171146e-003,  2.315819802942924e-004,-1.244493581373643e-002, 7.244364917221401e-003,1.207341716154354e-002, 2.734134166585232e-003,

-1.941706440790678e-002,                    -1.729098218843226e-002,1.773008730568675e-002, 4.091495174059349e-002,2.113436751136944e-003, -6.788468549771730e-002,

-6.059440700570791e-002,                    8.970313256448266e-002,3.014572949374625e-001, 4.019009454299968e-001,3.014572949374625e-001, 8.970313256448266e-002,

-6.059440700570791e-002,                    -6.788468549771730e-002,2.113436751136944e-003, 4.091495174059349e-002,1.773008730568675e-002, -1.729098218843226e-002,

-1.941706440790678e-002,                    2.734134166585232e-003,1.207341716154354e-002, 7.244364917221401e-003,-1.244493581373643e-002, 2.315819802942924e-004,

2.715297634171146e-003};*/


// High Pass Filter

float    filter_Coeff[]    ={3.294316420702696e-004,3.800020076486443e-003,9.822200806739014e-003,1.517265313889167e-002,

1.323547007544908e-002,2.635896986048919e-004,-1.808215737734512e-002,-2.666833013269758e-002,

-1.155354962270025e-002,2.448211866656400e-002,5.534101055783895e-002,4.424359087198896e-002,

-2.922329551555757e-002,-1.473332022689261e-001,-2.574625659073934e-001,6.976203109125493e-001,
-2.574625659073934e-001,-1.473332022689261e-001,-2.922329551555757e-002,4.424359087198896e-002,

5.534101055783895e-002,2.448211866656400e-002,-1.155354962270025e-002,-2.666833013269758e-002,

-1.808215737734512e-002,2.635896986048919e-004,1.323547007544908e-002,1.517265313889167e-002,

9.822200806739014e-003,3.800020076486443e-003,3.294316420702696e-004};


// Pre-generated sine wave data, 16-bit signed samples

int sinetable[SINE_TABLE_SIZE] = {

0x0000, 0x10b4, 0x2120, 0x30fb, 0x3fff, 0x4dea, 0x5a81, 0x658b,

0x6ed8, 0x763f, 0x7ba1, 0x7ee5, 0x7ffd, 0x7ee5, 0x7ba1, 0x76ef,

0x6ed8, 0x658b, 0x5a81, 0x4dea, 0x3fff, 0x30fb, 0x2120, 0x10b4,

0x0000, 0xef4c, 0xdee0, 0xcf06, 0xc002, 0xb216, 0xa57f, 0x9a75,

0x9128, 0x89c1, 0x845f, 0x811b, 0x8002, 0x811b, 0x845f, 0x89c1,

0x9128, 0x9a76, 0xa57f, 0xb216, 0xc002, 0xcf06, 0xdee0, 0xef4c

};

DSK6713_AIC23_Config config = { \

0x0017,  /* 0 DSK6713_AIC23_LEFTINVOL  Left line input channel volume */ \

0x0017,  /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume */\

0x00d8,  /* 2 DSK6713_AIC23_LEFTHPVOL  Left channel headphone volume */  \

0x00d8,  /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */ \

//          0x0014  micin with 0dB boost

0x0014,  /* 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 InitWait =1;

Int32 data;

Int32 Logger[1024];

Int32 LoggerIndex =0;


float in_buffer[100];


main(){

int i;

LED=0x0;


// Filter Initialization

for( i = 0 ; i < 100; i++ ) in_buffer[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;
```

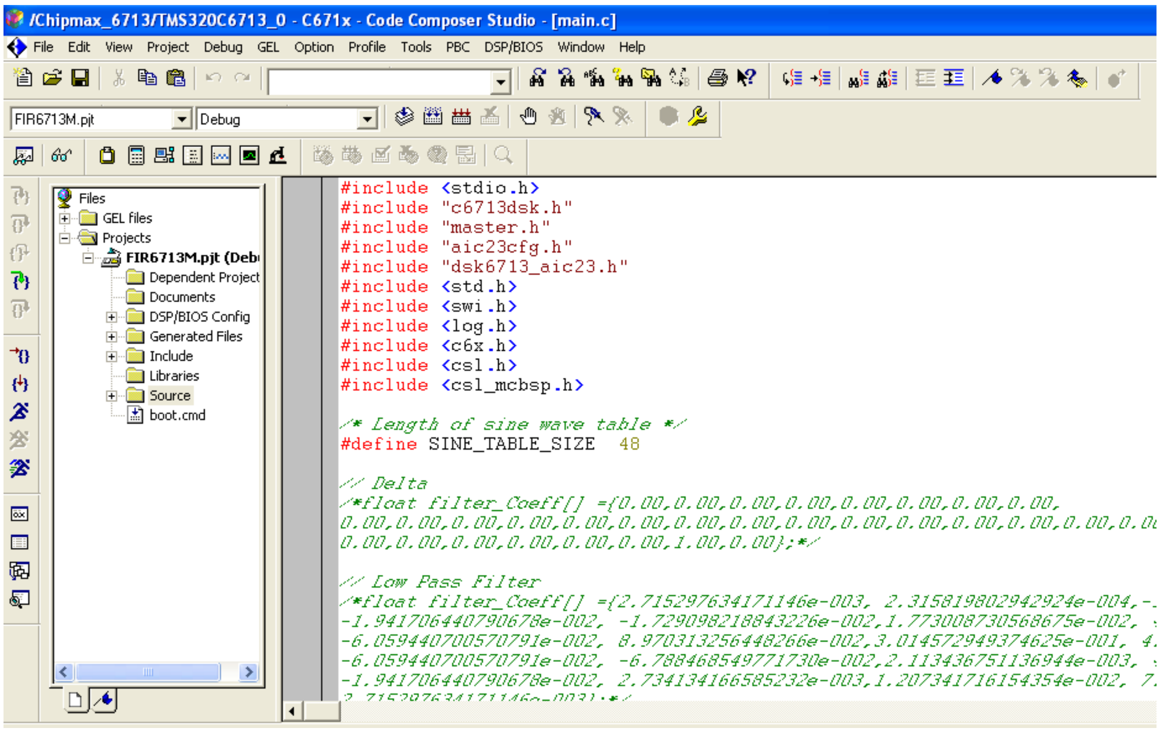
```
LED       = cc;


// To Shift Pattern

if (cc == 0x03) cc = 0x05;

else if (cc == 0x05) cc = 0x06;

else if (cc == 0x06) cc = 0x03;

else cc = 0x03;

//To count Binary

//cc++; if (cc>0x07) cc = 0x00;


// TO Toggle LED

// *cc ^= 0x07; if ((cc !=0x00) && (cc !=0x07)) cc = 0x07;

}


setpll200M(){


}


void read(){

int i = 0;

float result;

data=MCBSP_read(DSK6713_AIC23_DATAHANDLE);


if(data>=32768) data= data|0xffff0000;

for( i = 0; i <= 29; i++) in_buffer[i] = in_buffer[i+1];

in_buffer[30]=((float)(data))/16;
```

```
result = 0;

for( i = 0 ; i <= 30; i++ ) result += filter_Coeff[i] * in_buffer[i];

data = (Int32)(result*512);

//data = (Int32)(in_buffer[30]*16);

Logger[LoggerIndex++] = data;

if (LoggerIndex == 1024)

LoggerIndex = 0;

}


void write(){

if (InitWait<1000){

InitWait++;

MCBSP_write(DSK6713_AIC23_DATAHANDLE, 0);

MCBSP_write(DSK6713_AIC23_DATAHANDLE, 0);

}

else{

MCBSP_write(DSK6713_AIC23_DATAHANDLE, data);

MCBSP_write(DSK6713_AIC23_DATAHANDLE, data);

}

}
```

Output:

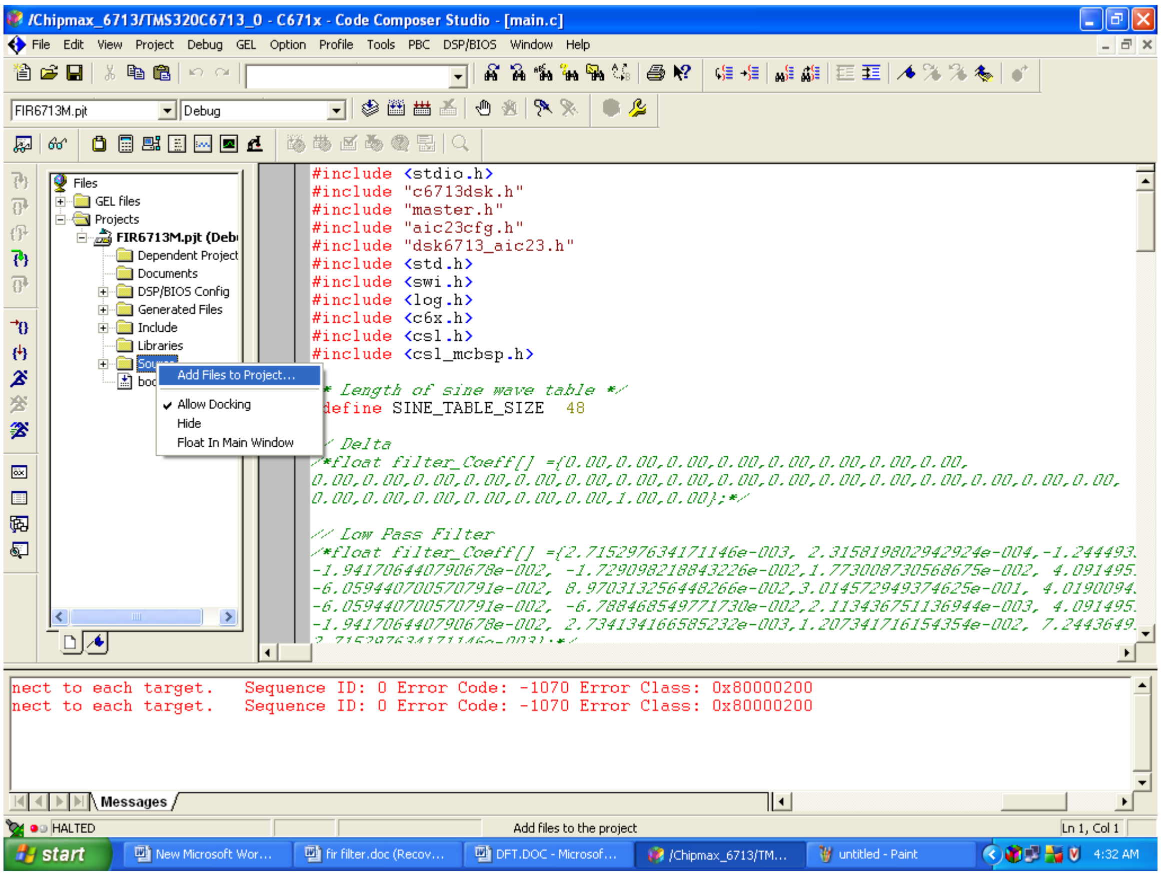


4. Enter the source code and save the file with main.c extension.


/Chipmax_6713/TMS320C6713_0 - C671x - Code Composer Studio - [main.c]
File Edit View Project Debug GEL Option Profile Tools PBC DSP/BIOS Window Help
FIR6713M.pjt
Debug
Files
GEL files
Projects
FIR6713M.pjt (Debu
Dependent Project
Documents
DSP/BIOS Config
Generated Files
Include
Libraries
Source
boot.cmd

```
#include <stdio.h>
#include "c6713dsk.h"
#include "master.h"
#include "aic23cfg.h"
#include "dsk6713_aic23.h"
#include <std.h>
#include <swi.h>
#include <log.h>
#include <c6x.h>
#include <csl.h>
#include <csl_mcbsp.h>

/* Length of sine wave table */
#define SINE_TABLE_SIZE  48

// Delta
/*float filter_Coeff[] ={0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.0
0.00,0.00,0.00,0.00,0.00,0.00,1.00,0.00};*/

// Low Pass Filter
/*float filter_Coeff[] ={2.715297634171146e-003, 2.315819802942924e-004,-
-1.941706440790678e-002, -1.729098218843226e-002,1.773008730568675e-002,
-6.059440700570791e-002, 8.970313256448266e-002,3.014572949374625e-001, 4.
-6.059440700570791e-002, -6.788468549771730e-002,2.113436751136944e-003,
-1.941706440790678e-002, 2.734134166585232e-003,1.207341716154354e-002, 7.
2.715297634171146e-003};*/
```

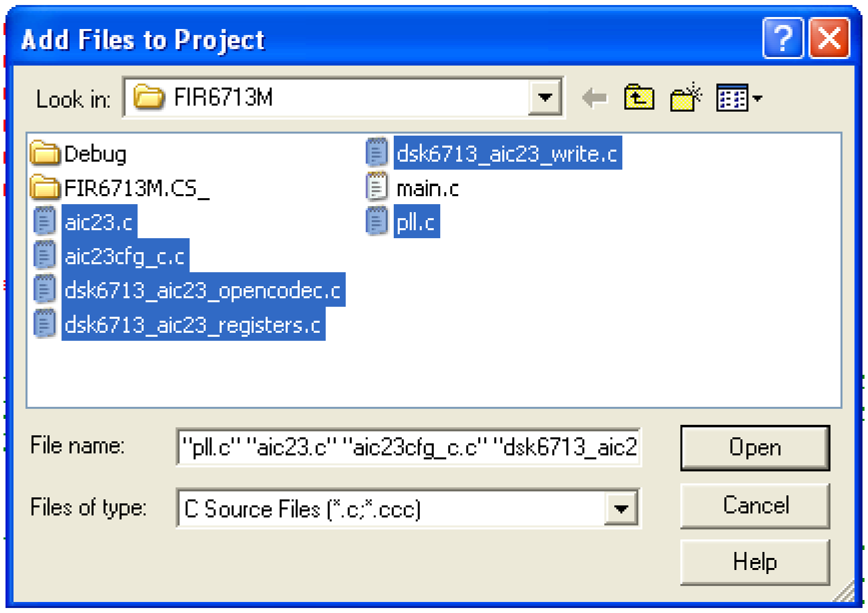
5. Right click on source, Select add files to project and Choose main.c file Saved before.



```
#include <stdio.h>
#include "c6713dsk.h"
#include "master.h"
#include "aic23cfg.h"
#include "dsk6713_aic23.h"
#include <std.h>
#include <swi.h>
#include <log.h>
#include <c6x.h>
#include <csl.h>
#include <csl_mcbsp.h>

* Length of sine wave table */
define SINE_TABLE_SIZE   48

/ Delta
/*float filter_Coeff[] ={0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,
0.00,0.00,0.00,0.00,0.00,0.00,1.00,0.00};*/

// Low Pass Filter
/*float filter_Coeff[] ={2.715297634171146e-003, 2.315819802942924e-004,-1.244493
-1.941706440790678e-002, -1.729098218843226e-002,1.773008730568675e-002, 4.091495
-6.059440700570791e-002, 8.970313256448266e-002,3.014572949374625e-001, 4.0190094
-6.059440700570791e-002, -6.788468549771730e-002,2.113436751136944e-003, 4.091495
-1.941706440790678e-002, 2.734134166585232e-003,1.207341716154354e-002, 7.2443649
2.715297634171146e-003};*/
```

```
nect to each target.   Sequence ID: 0 Error Code: -1070 Error Class: 0x80000200
nect to each target.   Sequence ID: 0 Error Code: -1070 Error Class: 0x80000200
```

6. Add the other supporting .c files which configure the audio codec.



Look in: FIR6713M

Debug
FIR6713M.CS_
aic23.c
aic23cfg_c.c
dsk6713_aic23_opencodec.c
dsk6713_aic23_registers.c
dsk6713_aic23_write.c
main.c
pll.c

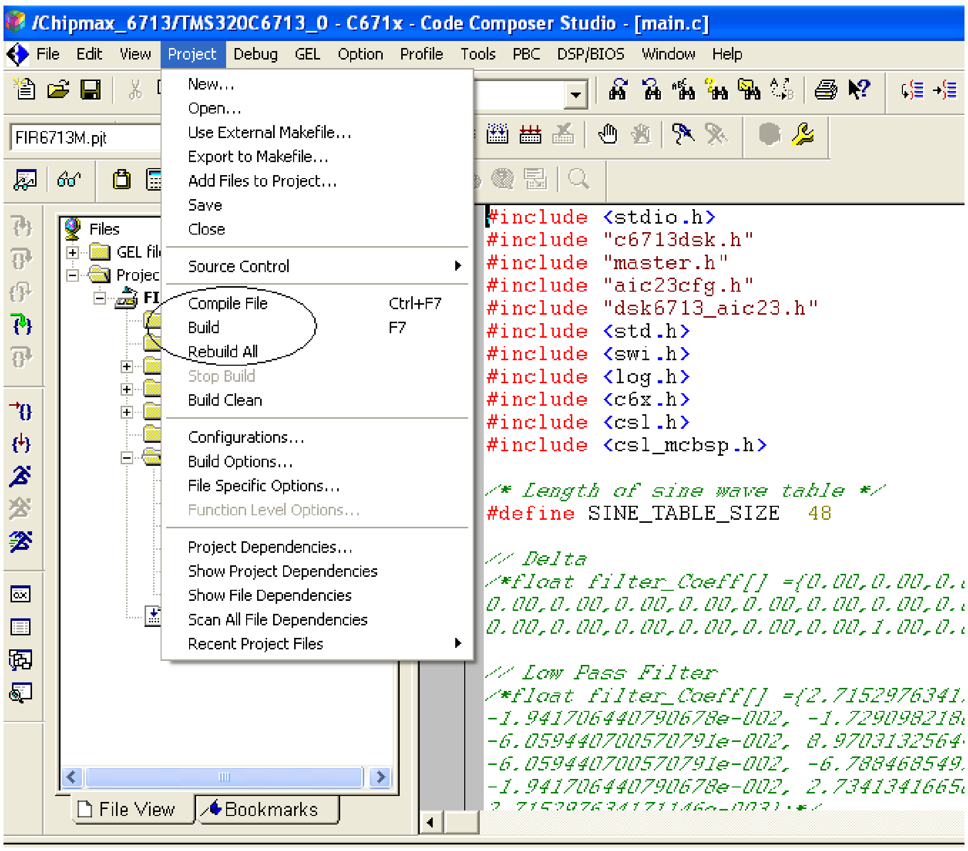
File name: "pll.c" "aic23.c" "aic23cfg_c.c" "dsk6713_aic2

Files of type: C Source Files (*.c;*.ccc)

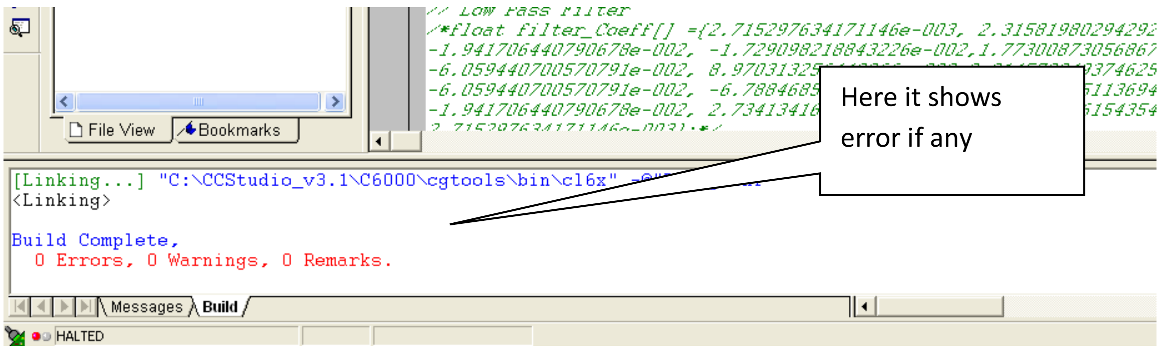
7. 7. a) Go to Project to Compile.

b) Go to Project to Build.

c) Go to Project to Rebuild All.

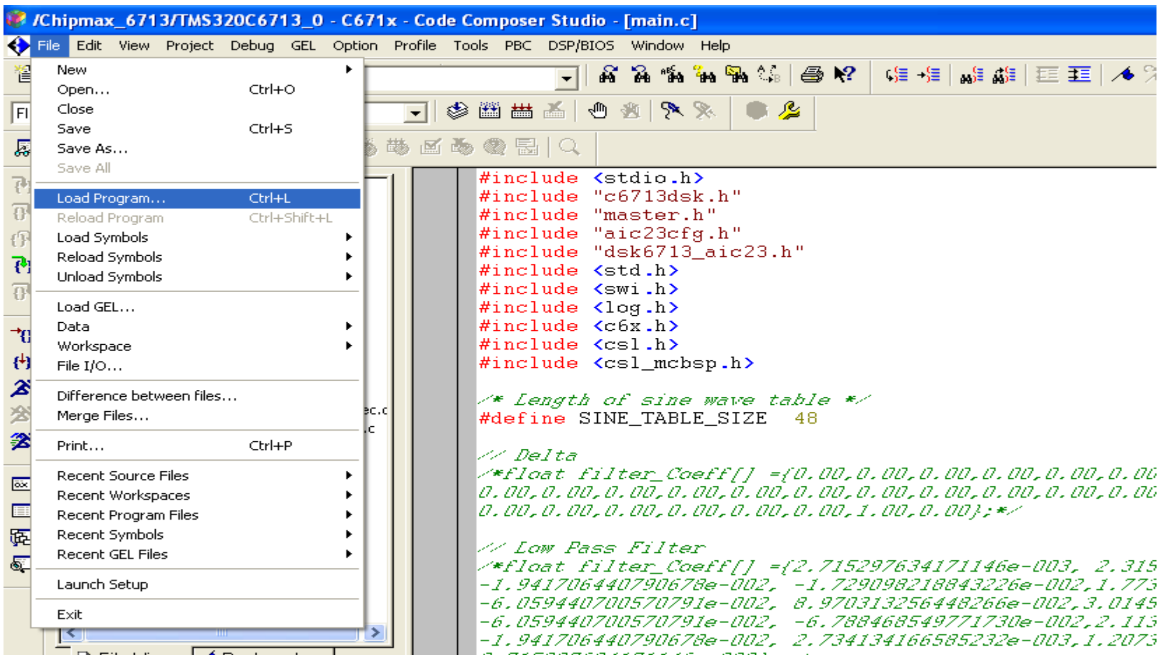


```
#include <stdio.h>
#include "c6713dsk.h"
#include "master.h"
#include "aic23cfg.h"
#include "dsk6713_aic23.h"
#include <std.h>
#include <swi.h>
#include <log.h>
#include <c6x.h>
#include <csl.h>
#include <csl_mcbsp.h>

/* Length of sine wave table */
#define SINE_TABLE_SIZE   48

// Delta
/*float filter_Coeff[] ={0.00,0.00,0.
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.
0.00,0.00,0.00,0.00,0.00,0.00,1.00,0.

// Low Pass Filter
/*float filter_Coeff[] ={2.7152976341
-1.94170644079067 8e-002, -1.729098218
-6.059440700570791e-002, 8.9703132564
-6.059440700570791e-002, -6.788468549
-1.941706440790678e-002, 2.7341341665
```

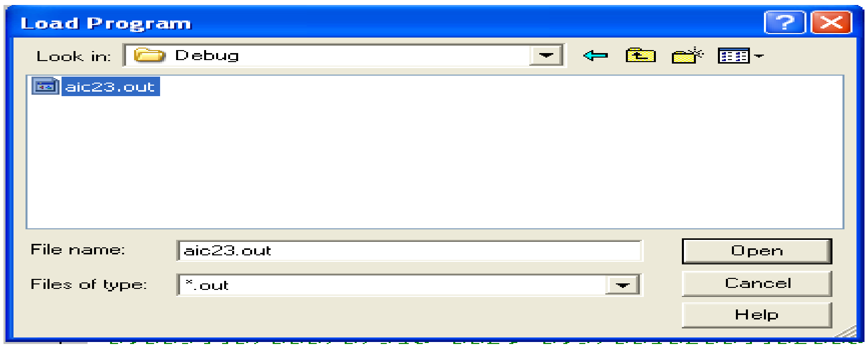


```
[Linking...] "C:\CCStudio_v3.1\C6000\cgtools\bin\cl6x"
<Linking>

Build Complete,
  0 Errors, 0 Warnings, 0 Remarks.
```

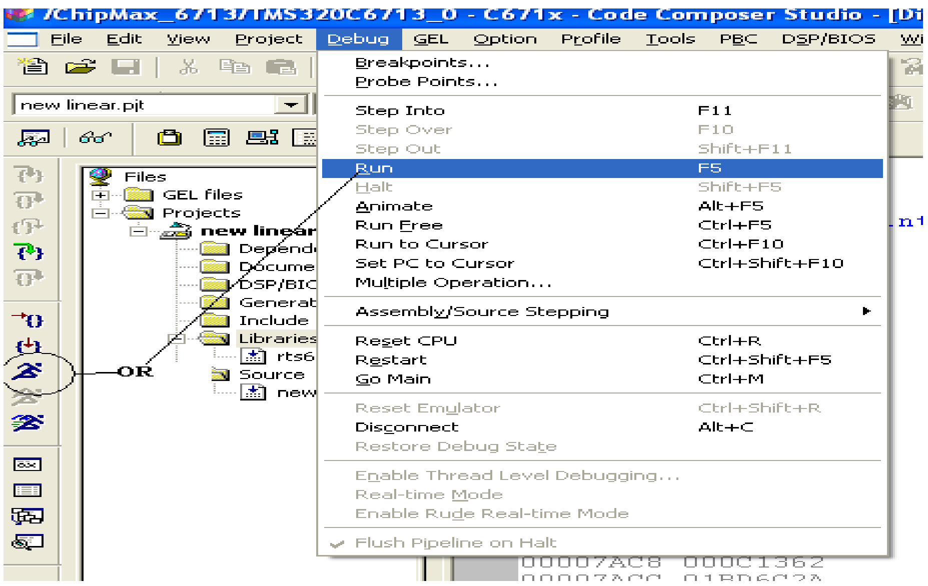
8. Go to file and load program and load **“.out”** file into the board.



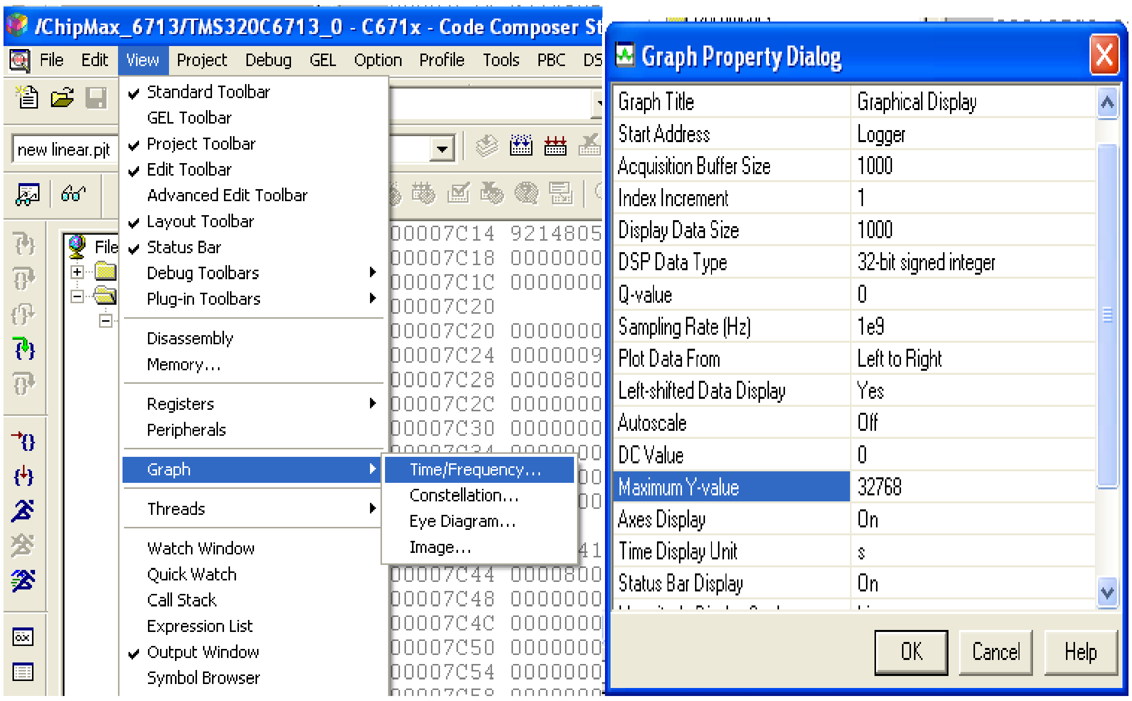
```
#include <stdio.h>
#include "c6713dsk.h"
#include "master.h"
#include "aic23cfg.h"
#include "dsk6713_aic23.h"
#include <std.h>
#include <swi.h>
#include <log.h>
#include <c6x.h>
#include <csl.h>
#include <csl_mcbsp.h>

/* Length of sine wave table */
#define SINE_TABLE_SIZE   48

// Delta
/*float filter_Coeff[] ={0.00,0.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,0.00,0.00,0.00,1.00,0.00};*/

// Low Pass Filter
/*float filter_Coeff[] ={2.715297634171146e-003, 2.315
-1.941706440790678e-002, -1.729098218843226e-002,1.773
-6.059440700570791e-002, 8.970313256448266e-002,3.0145
-6.059440700570791e-002, -6.788468549771730e-002,2.113
-1.941706440790678e-002, 2.734134166585232e-003,1.2073
```



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

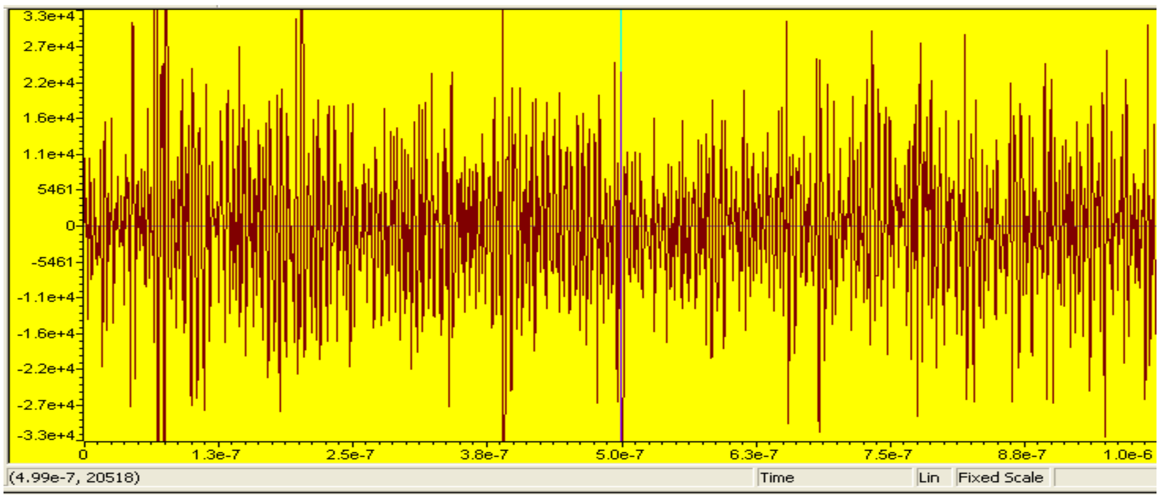


10. To see the Graph go to View and select time/frequency in the Graph and give the correct Start address provided in the program, Display data can be taken as per user.

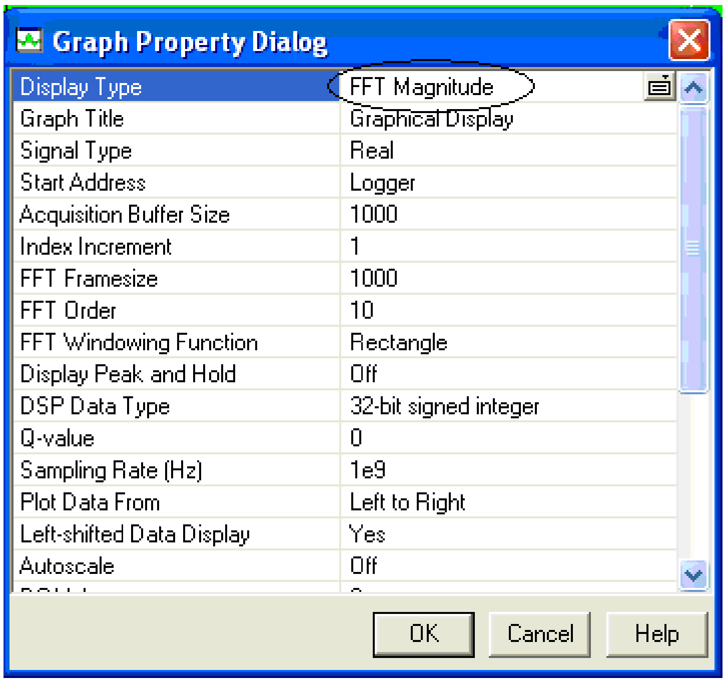


Graph Title	Graphical Display
Start Address	Logger
Acquisition Buffer Size	1000
Index Increment	1
Display Data Size	1000
DSP Data Type	32-bit signed integer
Q-value	0
Sampling Rate (Hz)	1e9
Plot Data From	Left to Right
Left-shifted Data Display	Yes
Autoscale	Off
DC Value	0
Maximum Y-value	32768
Axes Display	On
Time Display Unit	s
Status Bar Display	On

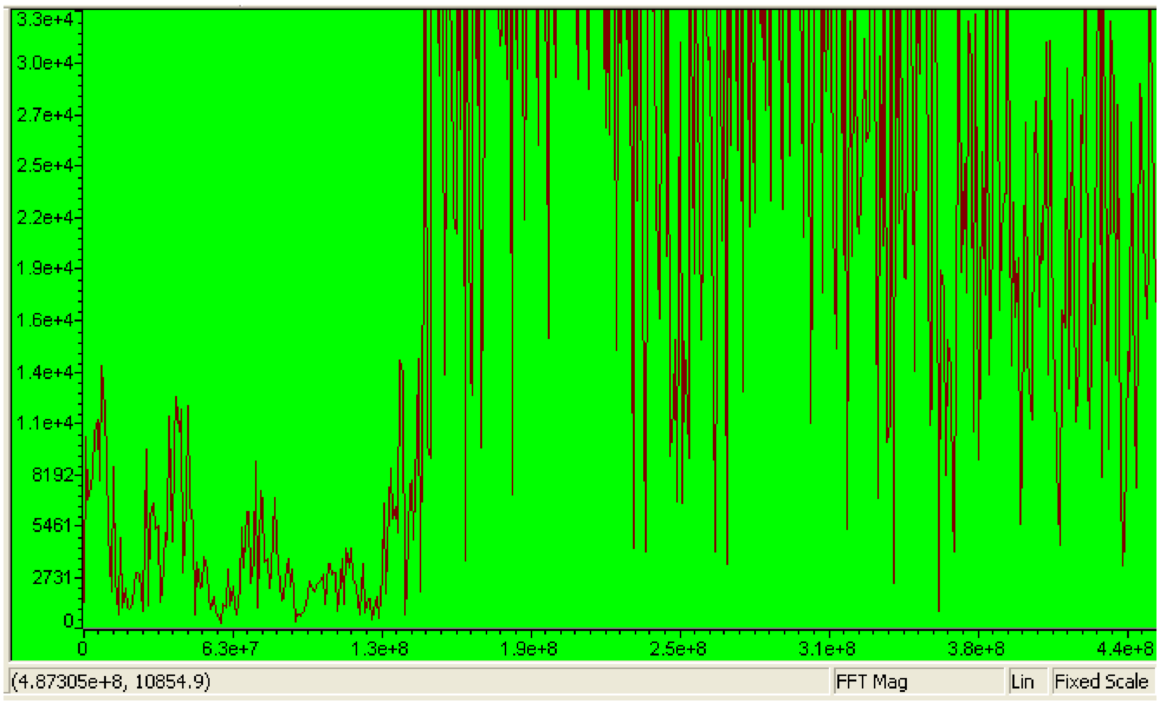
11. Green line is to choose the point, Value at the point can be seen (Highlighted by circle at the left corner).



12. In the graph, chose FFT magnitude as display type we will get Graph B

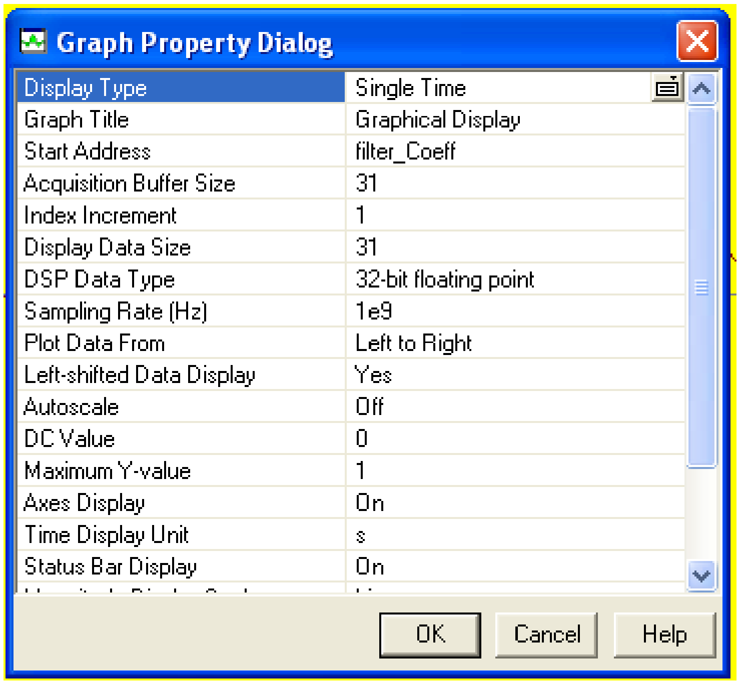


Display Type	FFT Magnitude
Graph Title	Graphical Display
Signal Type	Real
Start Address	Logger
Acquisition Buffer Size	1000
Index Increment	1
FFT Framesize	1000
FFT Order	10
FFT Windowing Function	Rectangle
Display Peak and Hold	Off
DSP Data Type	32-bit signed integer
Q-value	0
Sampling Rate (Hz)	1e9
Plot Data From	Left to Right
Left-shifted Data Display	Yes
Autoscale	Off



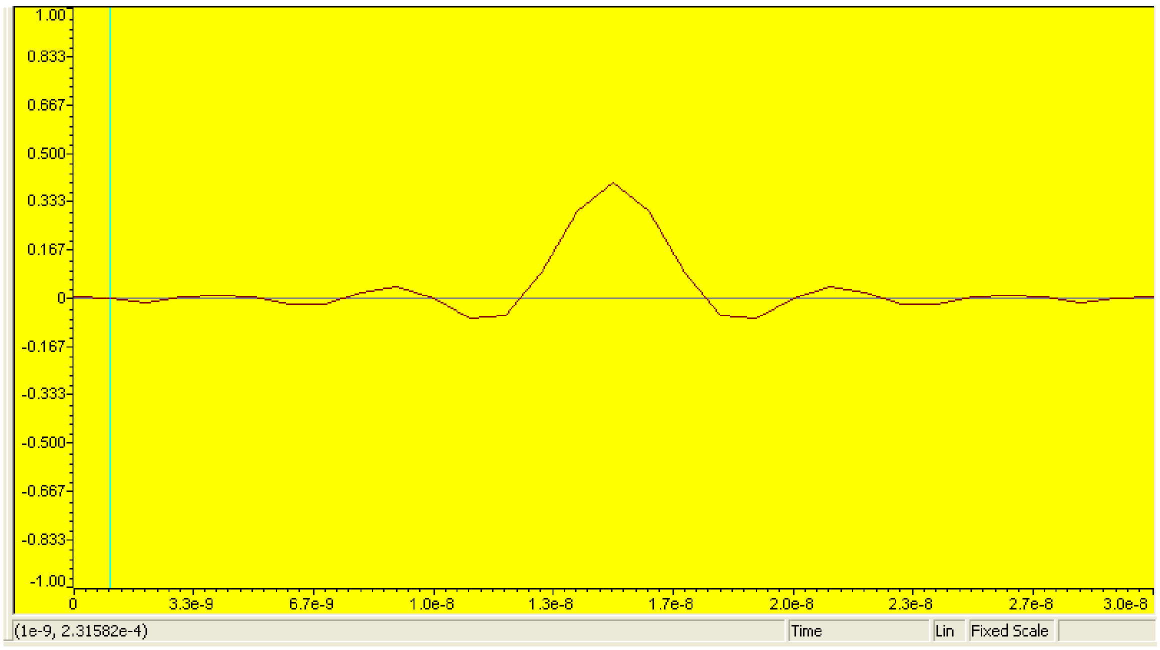
Graph 2:FFT magnitude of FIR filter.

1. The impulse response of FIR filters.



Graph Property Dialog

Display Type	Single Time
Graph Title	Graphical Display
Start Address	filter_Coeff
Acquisition Buffer Size	31
Index Increment	1
Display Data Size	31
DSP Data Type	32-bit floating point
Sampling Rate (Hz)	1e9
Plot Data From	Left to Right
Left-shifted Data Display	Yes
Autoscale	Off
DC Value	0
Maximum Y-value	1
Axes Display	On
Time Display Unit	s
Status Bar Display	On



Experiment 4: Audio Applications

Aim: - To Perform Audio applications such as to plot a time and frequency display of microphone plus a cosine using DSP. Read a wav file and match with their respective spectrograms.

Theory: Spectrogram with RTDX using MATLAB This version of project makes use of RTDX with MATLAB for transferring data from the DSK to the PC host. This section introduces configuration file(.CDB) file and RTDX with MATLAB.

This project uses source program spectrogram_rtdx_mtl.c that runs on the DSK which computes 256 point FFT and enables an RTDX output channel to write/send the resulting FFT data to the PC running MATLAB for finding the spectrogram. A total o f N/2 (128 points)are sent. The (.CDB) configuration file is used to set interrupt INT11. From this configuration file select Input/Output and RTDX. Right click on properties and change the RTDX buffer size to 8200. Within CCS, select tools, RTDX ,Configure to set the host buffer size to 2048(from 1024). An input signal is read in blocks of 256 samples. Each block of data is then multiplied with a hamming window of length 256 points. The FFT of the windowed data is calculated and squared. Half of the resulting FFT of each block of 256 points is then transferred to the PC running MATLAB to find the specrtrogram.

Spectrogram_rtdx_mtl.c Time-Frequency analysis of signals Using RTDX-MATLAB

```
 #include "dsk6713_aic23.h"              //codec-DSK support file
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ; //set sampling rate
#include <rtdx.h>   //RTDX support file
#include <math.h>
#include "hamming.cof" //Hamming windowcoefficients
 #define PTS 256 //# of points for FFT
#define PI 3.14159265358979
typedef struct {float real,imag;} COMPLEX;
void FFT(COMPLEX *Y, int n); //FFT prototype
float iobuffer[PTS],iobuffer1[PTS],a[PTS]; //input and output buffer
float x1[PTS]; //intermediate buffer
short i; //general purpose index variable

int j, k,l, curr_block = 0; //index variables
short buffercount = 0; //number of new samples in iobuffer
short flag = 0; //set to 1 by ISR when iobuffer full
COMPLEX w[PTS]; //twiddle constants stored in w
COMPLEX samples[PTS]; //primary working buffer
 RTDX_CreateOutputChannel(ochan); //create output channel C6x->PC

main() { for (i = 0 ; i<PTS;I++)

{
```

```
w[i].real = cos(2*PI*i/512.0); //Re component of twiddle constants
w[i].imag =-sin(2*PI*i/512.0); //Im component of twiddle constants
}
 comm_intr(); //init DSK, codec, McBSP

while(!RTDX_isOutputEnabled(&ochan)) //wait for PC to enable RTDX

puts("\n\n Waiting . . . "); //while waiting
for(l=0;l<256;l++)
 a[l]=cos(2*3.14*1500*l/8000);
for(k=0;k<5000;k++) //infinite loop

{
while (flag == 0) ; //wait until iobuffer is full
 flag = 0; //reset flag
for (i = 0 ; i < PTS; i++) //swap buffers
{ iobuffer1[i]=iobuffer[i]+a[i];
samples[i].real=h[i]*iobuffer1[i]; //multiply by Hamming window coeffs

 iobuffer1[i] = x1[i]; //process frame to iobuffer
}


for (i = 0 ; i < PTS ; i++)
samples[i].imag = 0.0; //imag components = 0
FFT(samples,PTS); //call C-coded FFT function
 for (i = 0 ; i < PTS; i++) //compute square of FFT magnitude

{

x1[i] = (samples[i].real*samples[i].real + samples[i].imag*samples[i].imag)/16; //FFT data
scaling
}
 RTDX_write(&ochan, x1, sizeof(x1)/2); //send 128 samples to PC
 } //end of infinite loop
} //end of main


interrupt void c_int11() //ISR
{ output_sample((short)(iobuffer[buffercount])); //out from iobuffer
iobuffer[buffercount++]=(short)(input_sample()); //input to iobuffer
 if (buffercount >= PTS) //if iobuffer full
{
 buffercount = 0; //reinit buffercount
flag = 1;  //reset flag
}
}
```

<u>FFT.c C callable FFT function in C</u>

```
#define PTS 256       //# of points for FFT

typedef struct {float real,imag;}COMPLEX;


externCOMPLEX w[PTS]; //twiddle constants stored in w



void FFT(COMPLEX *Y, int N)        //input sample array, # of points

{

COMPLEX temp1,temp2;        //temporary storage variables
int i,j,k;            //loop counter variables


int upper_leg, lower_leg;       //index of upper/lower butterflyleg

int leg_diff;       //difference between upper/lower leg

int num_stages = 0;     //number ofFFT stages (iterations)
int index, step;            //index/step through twiddle constant
i = 1;                      //log(base2) of N points= # ofstages
do
{
num_stages +=1;
i = i*2;
}while (i!=N);
leg_diff = N/2;             //difference between upper&lower legs
step = 512/N;               //step between values in twiddle.h
for (i = 0;i < num_stages; i++)             //for N-point FFT
{
index = 0;
for (j = 0; j < leg_diff; j++)
{
for (upper_leg = j; upper_leg < N; upper_leg += (2*leg_diff))
{

lower_leg = upper_leg+leg_diff;
temp1.real = (Y[upper_leg]).real + (Y[lower_leg]).real;
temp1.imag = (Y[upper_leg]).imag + (Y[lower_leg]).imag;
temp2.real = (Y[upper_leg]).real - (Y[lower_leg]).real;
temp2.imag = (Y[upper_leg]).imag - (Y[lower_leg]).imag;
(Y[lower_leg]).real = temp2.real*(w[index]).real -temp2.imag*(w[index]).imag;
(Y[lower_leg]).imag = temp2.real*(w[index]).imag
+temp2.imag*(w[index]).real;

(Y[upper_leg]).real = temp1.real;
(Y[upper_leg]).imag = temp1.imag;
}
index += step;
}
leg_diff = leg_diff/2;
step *= 2;


}
 j = 0;
for (i = 1; i < N-1;i++)   // bit reversal for resequencing data

{

k=N/2;
while (k <= j)
{
j = j - k;
k = k/2;
}
j = j + k;
if (i<j)
{
temp1.real = (Y[j]).real;
temp1.imag = (Y[j]).imag;
(Y[j]).real = (Y[i]).real;
(Y[j]).imag = (Y[i]).imag;
(Y[i]).real = temp1.real;
(Y[i]).imag = temp1.imag;
}
}
return;
}
```

Spectrogram_RTDX.m For spectrogram plot using RTDX with MATLAB

```
clc;
ccsboardinfo %board info
cc=ccsdsp('boardnum',0); %set up CCS object
reset(cc); %reset board
visible(cc,1); %for CCS window
enable(cc.rtdx); %enable RTDX
if ~isenabled(cc.rtdx);
```

```
error('RTDX is not enabled')
end
cc.rtdx.set('timeout',50); %set 50sec timeout for RTDX
open(cc,'spectrogram1.pjt'); %open CCS project
load(cc,'./debug/spectrogram1.out'); %load executable file
run(cc); %run program
configure(cc.rtdx,2048,1); %configure one RTDX channel
open(cc.rtdx,'ochan','r'); %open output channel

pause(3) %wait for RTDX channel to open
enable(cc.rtdx,'ochan'); %enable channel from DSK
isenabled(cc.rtdx,'ochan');
M = 256; %window size
N = round(M/2);
B = 128; %No. of blocks(128)
fs = 8000; %sampling rate
t=(1:B)*(M/fs); %spectrogram axes generation
f=((0:(M-1)/2)/(M-1))*fs;
set(gcf,'DoubleBuffer','on');
y = ones(N,B);
column = 1;
set(gca,'NextPlot','add');
axes_handle = get(gcf,'CurrentAxes');
set(get(axes_handle,'XLabel'),'String','Time (s)');
set(get(axes_handle,'YLabel'),'String','Frequency (Hz)');
set(get(axes_handle,'Title'),'String','\fontname{times} \bf Real-Time Spectrogram');

set(gca,'XLim', [0 4.096]);
set(gca,'YLim', [0 4000]);
set(gca,'XLimMode','manual');
set(gca,'YLimMode','manual');
for i = 1:32768
w=readmsg(cc.rtdx,'ochan','single'); %read FFT data from DSK
w=double(w(1:N));
y(:, column) = w';


imagesc(t,f,dB(y)); %plot spectrogram
column = mod(column, B) + 1;
end
halt(cc); %halt processor
close(cc.rtdx,'ochan'); %close channel
clear cc %clear object
```

Procedure:
1. Create a new project with name spectrogram.pjt.
2. Open “spectrogram.cdb” from given CD and save it in your new project folder.
3. Copy the following files from the CD to your new project folder
1) c6713dskinit . c

2) FFT.c
3) spectrogram_rtdx_mtl.c
4) c6713dskinit . h
5) hamming.cof
6) spectrogram_RTDX.m
4. Add “spectrogram.cdb”, “c6713dskinit.c” and “spectrogram_rtdx_mtl.c” to the current project.
5. Add the library file “dsk6713bsl.lib” to the current project

Path -----------“C:\CCStudio\C6000\dsk6713\lib\dsk6713bsl.lib”
5. Set the following compiler options.
Select Project and choose Build options.
Select the following for compiler option with Basic (for category):
(1) c671x{mv6710} (for target version)
(2) Full symbolic debug (for Generate Debug info)
(3) Speed most critical(for Opt Speed vs. Size)
(4) None (for Opt Level and Program Level Opt)

Select The Preprocessor Category and Type for Define Symbols{d}: CHIP_6713, and from Feedback category, select for Interlisting: OPT / C and ASM{-s}

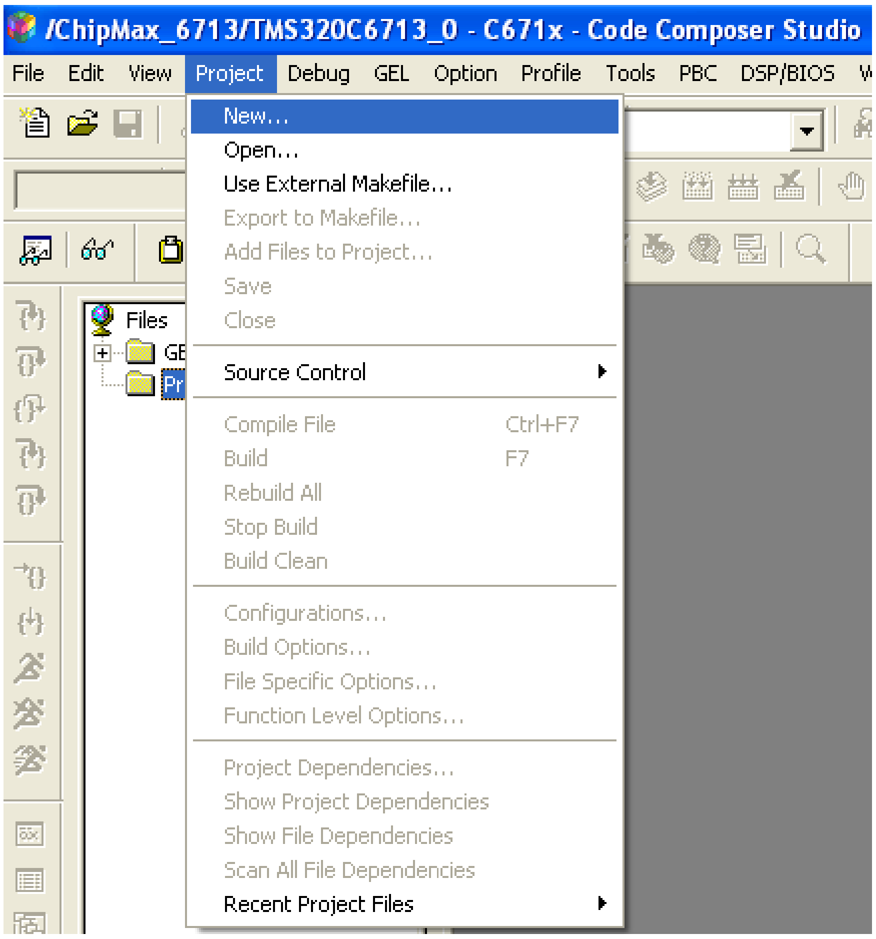
6 Build project.
7. Close CCS
8. Open MATLAB and Run spectrogram_RTDX.m . within MATLAB ,CCS will enable RTDX and will load and run the COFF(.out) executable file. Then MATLAB will plot the spectrogram of an input signal.

Result:- Thus Audio application is performed and spectrogram of an input signal is plotted using Matlab.

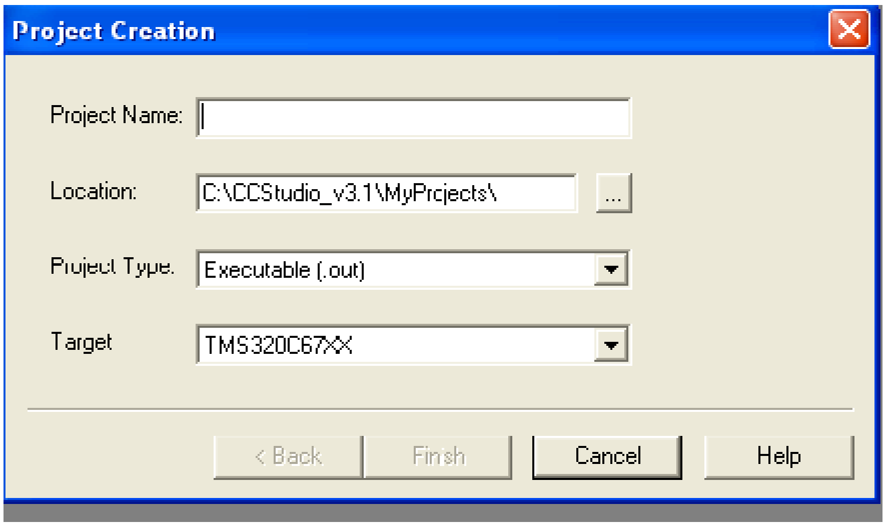
Experiment 5: Noise Removal

Procedure to create new Project:

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



2. Give project name and click on finish.


Project Creation

Project Name:

Location: C:\CCStudio_v3.1\MyProjects\

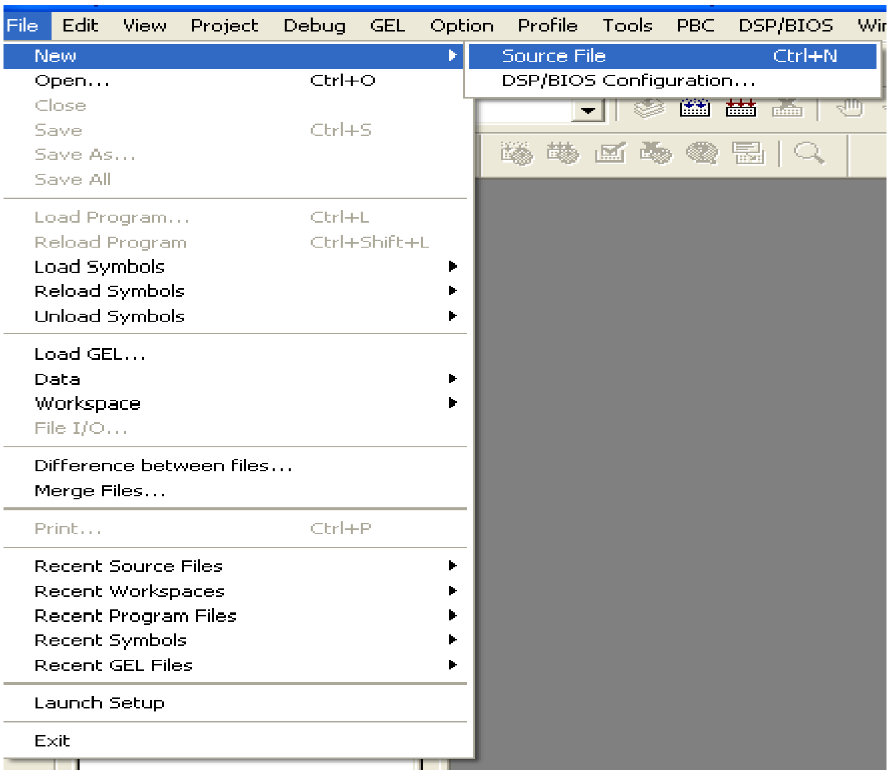
Project Type: Executable (.out)

Target TMS320C67XX

< Back | Finish | Cancel | Help

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

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



AIM: Noise removal in a given mixed signal.

Noise removal: In the real time systems every signal will get corrupted by the noise. To analyze the noise we need to add the noise, a noise will be generated and added to the signal. A noise of 3KHz is added to a tone of 400KHz then the noise is removed by using an high pass filter.

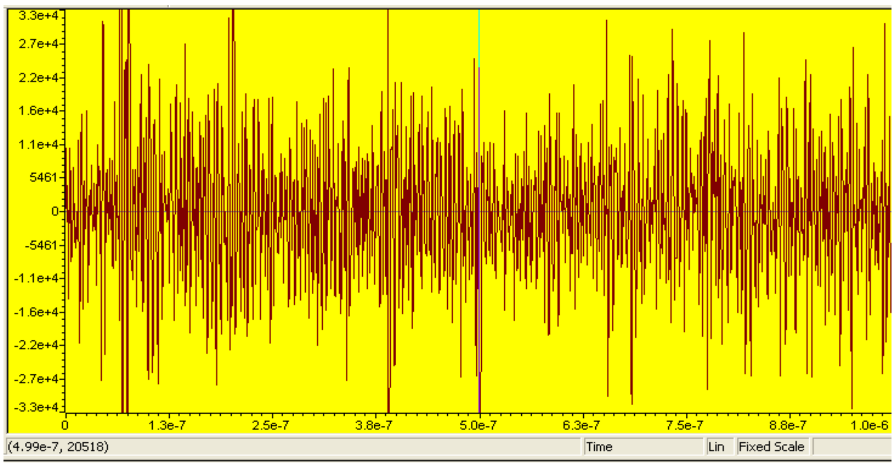
Program:

```
#include <stdio.h>

#include "c6713dsk.h"

#include "master.h"

#include "aic23cfg.h"

#include "dsk6713_aic23.h"

#include <std.h>

#include <swi.h>

#include <log.h>

#include <c6x.h>

#include <csl.h>

#include <csl_mcbsp.h>


/* Length of sine wave table */

#define SINE_TABLE_SIZE  48


// Delta

/*float filter_Coeff[] ={0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,

0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,

0.00,0.00,0.00,0.00,0.00,0.00,1.00,0.00};*/


// Low Pass Filter

/*float filter_Coeff[] ={2.715297634171146e-003, 2.315819802942924e-004,-1.244493581373643e-
002, 7.244364917221401e-003,1.207341716154354e-002, 2.734134166585232e-003,
-1.941706440790678e-002, -1.729098218843226e-002,1.773008730568675e-002,
4.091495174059349e-002,2.113436751136944e-003, -6.788468549771730e-002,

-6.059440700570791e-002, 8.970313256448266e-002,3.014572949374625e-001,
4.019009454299968e-001,3.014572949374625e-001, 8.970313256448266e-002,

-6.059440700570791e-002, -6.788468549771730e-002,2.113436751136944e-003,
4.091495174059349e-002,1.773008730568675e-002, -1.729098218843226e-002,

-1.941706440790678e-002, 2.734134166585232e-003,1.207341716154354e-002,
7.244364917221401e-003,-1.244493581373643e-002, 2.315819802942924e-004,

2.715297634171146e-003};*/


// High Pass Filter

float filter_Coeff[] ={3.294316420702696e-004,3.800020076486443e-003,9.822200806739014e-
003,1.517265313889167e-002,

1.323547007544908e-002,2.635896986048919e-004,-1.808215737734512e-002,-
2.666833013269758e-002,

-1.155354962270025e-002,2.448211866656400e-002,5.534101055783895e-
002,4.424359087198896e-002,

-2.922329551555757e-002,-1.473332022689261e-001,-2.574625659073934e-
001,6.976203109125493e-001,

-2.574625659073934e-001,-1.473332022689261e-001,-2.922329551555757e-
002,4.424359087198896e-002,

5.534101055783895e-002,2.448211866656400e-002,-1.155354962270025e-002,-
2.666833013269758e-002,

-1.808215737734512e-002,2.635896986048919e-004,1.323547007544908e-
002,1.517265313889167e-002,

9.822200806739014e-003,3.800020076486443e-003,3.294316420702696e-004};


// Pre-generated sine wave data, 16-bit signed samples

int sinetable[SINE_TABLE_SIZE] = {

0x0000, 0x10b4, 0x2120, 0x30fb, 0x3fff, 0x4dea, 0x5a81, 0x658b,

0x6ed8, 0x763f, 0x7ba1, 0x7ee5, 0x7ffd, 0x7ee5, 0x7ba1, 0x76ef,

0x6ed8, 0x658b, 0x5a81, 0x4dea, 0x3fff, 0x30fb, 0x2120, 0x10b4,
0x0000, 0xef4c, 0xdee0, 0xcf06, 0xc002, 0xb216, 0xa57f, 0x9a75,

0x9128, 0x89c1, 0x845f, 0x811b, 0x8002, 0x811b, 0x845f, 0x89c1,

0x9128, 0x9a76, 0xa57f, 0xb216, 0xc002, 0xcf06, 0xdee0, 0xef4c

};

DSK6713_AIC23_Config config = { \

0x0017,  /* 0 DSK6713_AIC23_LEFTINVOL  Left line input channel volume */ \

0x0017,  /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume */\

0x00d8,  /* 2 DSK6713_AIC23_LEFTHPVOL  Left channel headphone volume */  \

0x00d8,  /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */ \

//          0x0014  micin with 0dB boost

0x0014,  /* 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 InitWait =1;

Int32 data;

Int32 Logger[1024];

Int32 LoggerIndex =0;


float in_buffer[100];
```

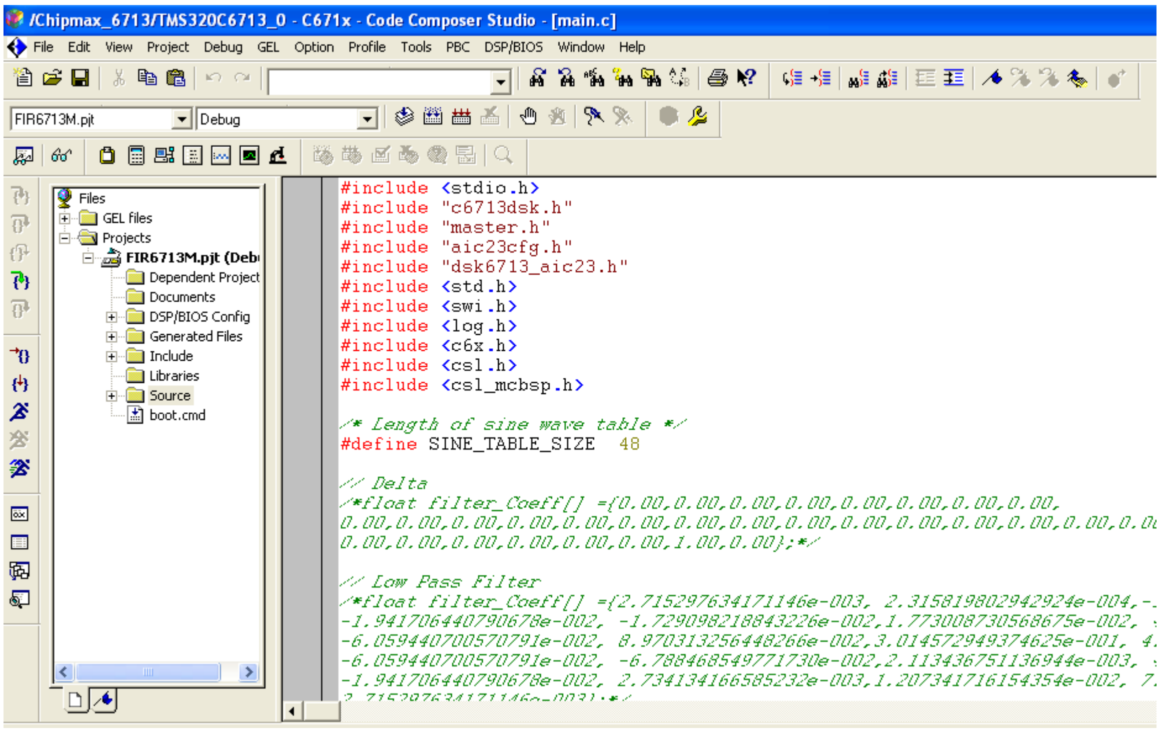
```
main(){

int i;

LED=0x0;


// Filter Initialization

for( i = 0 ; i < 100; i++ ) in_buffer[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;

//To count Binary

//cc++; if (cc>0x07) cc = 0x00;


// TO Toggle LED
// *cc ^= 0x07; if ((cc !=0x00) && (cc !=0x07)) cc = 0x07;

}


setpll200M(){


}


void read(){

int i = 0;

float result;

data=MCBSP_read(DSK6713_AIC23_DATAHANDLE);


if(data>=32768) data= data|0xffff0000;

for( i = 0; i <= 29; i++) in_buffer[i] = in_buffer[i+1];

in_buffer[30]=((float)(data))/16;


result = 0;

for( i = 0 ; i <= 30; i++ ) result += filter_Coeff[i] * in_buffer[i];

data = (Int32)(result*512);

//data = (Int32)(in_buffer[30]*16);

Logger[LoggerIndex++] = data;

if (LoggerIndex == 1024)

LoggerIndex = 0;

}


void write(){

if (InitWait<1000){
```

```
InitWait++;

MCBSP_write(DSK6713_AIC23_DATAHANDLE, 0);

MCBSP_write(DSK6713_AIC23_DATAHANDLE, 0);

}

else{

MCBSP_write(DSK6713_AIC23_DATAHANDLE, data);

MCBSP_write(DSK6713_AIC23_DATAHANDLE, data);

}

}
```

<u>Output</u>:

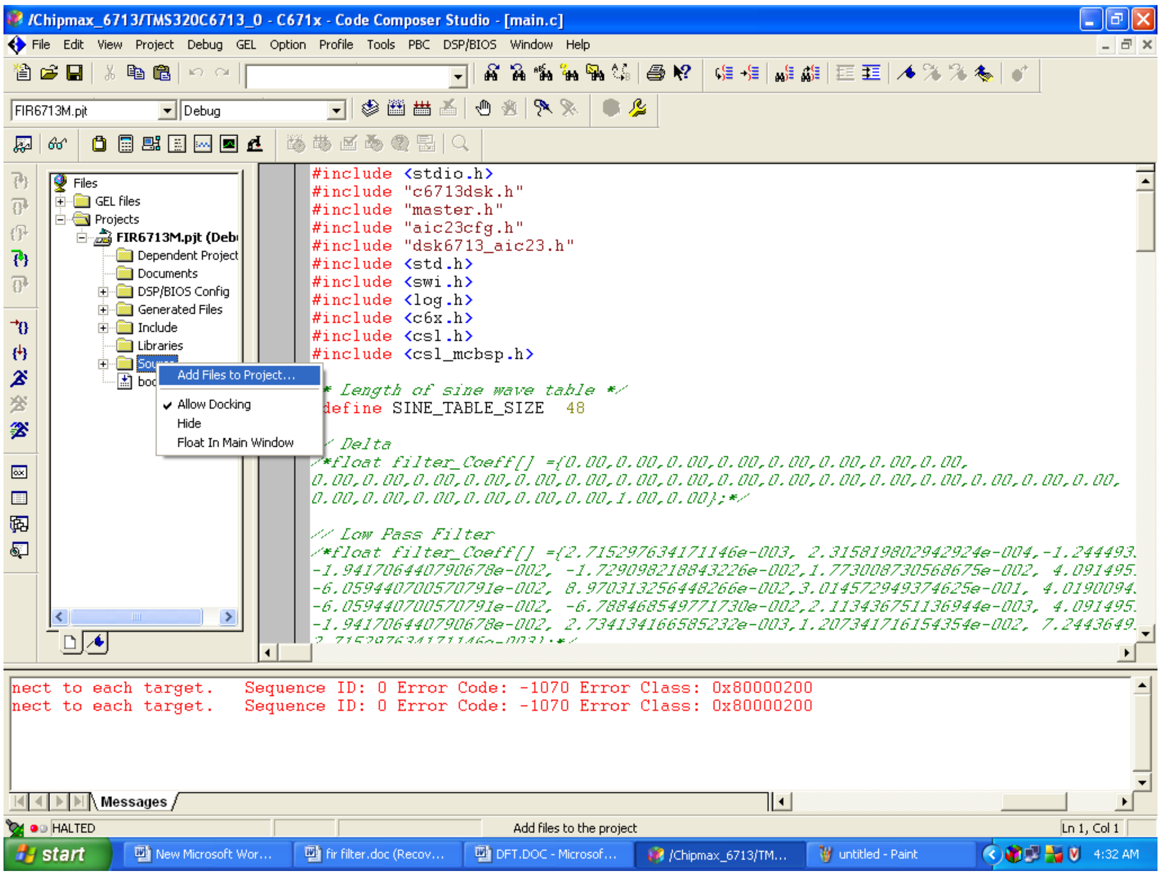


4. Enter the source code and save the file with main.c extension.


/Chipmax_6713/TMS320C6713_0 - C671x - Code Composer Studio - [main.c]
File Edit View Project Debug GEL Option Profile Tools PBC DSP/BIOS Window Help
FIR6713M.pjt
Debug
Files
GEL files
Projects
FIR6713M.pjt (Debi
Dependent Project
Documents
DSP/BIOS Config
Generated Files
Include
Libraries
Source
boot.cmd

```
#include <stdio.h>
#include "c6713dsk.h"
#include "master.h"
#include "aic23cfg.h"
#include "dsk6713_aic23.h"
#include <std.h>
#include <swi.h>
#include <log.h>
#include <c6x.h>
#include <csl.h>
#include <csl_mcbsp.h>

/* Length of sine wave table */
#define SINE_TABLE_SIZE  48

// Delta
/*float filter_Coeff[] ={0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.0
0.00,0.00,0.00,0.00,0.00,0.00,1.00,0.00};*/

// Low Pass Filter
/*float filter_Coeff[] ={2.715297634171146e-003, 2.315819802942924e-004,-
-1.941706440790678e-002, -1.729098218843226e-002,1.773008730568675e-002,
-6.059440700570791e-002, 8.970313256448266e-002,3.014572949374625e-001, 4.
-6.059440700570791e-002, -6.788468549771730e-002,2.113436751136944e-003,
-1.941706440790678e-002, 2.734134166585232e-003,1.207341716154354e-002, 7.
2.715297634171146e-003};*/
```

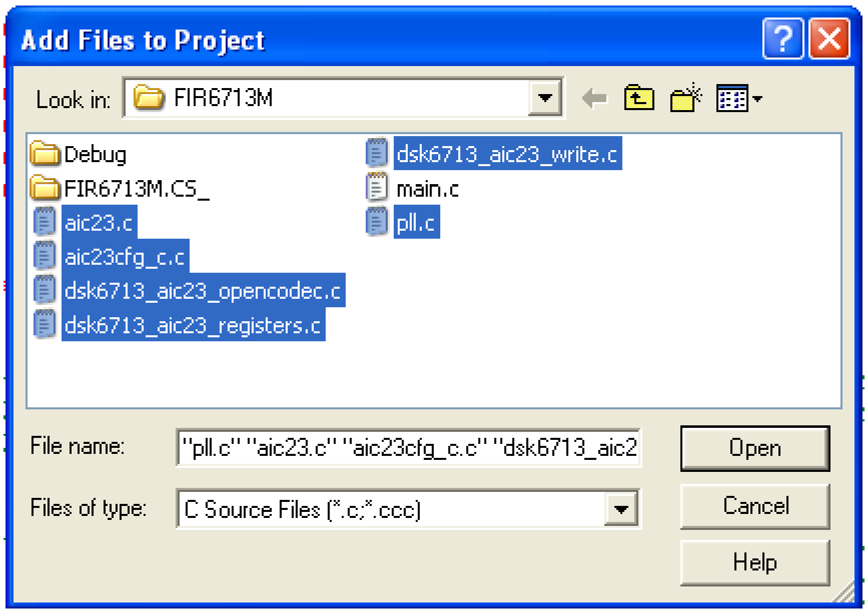
5. Right click on source, Select add files to project and Choose main.c file Saved before.



```
#include <stdio.h>
#include "c6713dsk.h"
#include "master.h"
#include "aic23cfg.h"
#include "dsk6713_aic23.h"
#include <std.h>
#include <swi.h>
#include <log.h>
#include <c6x.h>
#include <csl.h>
#include <csl_mcbsp.h>

* Length of sine wave table */
define SINE_TABLE_SIZE  48

/ Delta
/*float filter_Coeff[] ={0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,
0.00,0.00,0.00,0.00,0.00,0.00,1.00,0.00};*/

// Low Pass Filter
/*float filter_Coeff[] ={2.715297634171146e-003, 2.315819802942924e-004,-1.244493
-1.941706440790678e-002, -1.729098218843226e-002,1.773008730568675e-002, 4.091495
-6.059440700570791e-002, 8.970313256448266e-002,3.014572949374625e-001, 4.0190094
-6.059440700570791e-002, -6.788468549771730e-002,2.113436751136944e-003, 4.091495
-1.941706440790678e-002, 2.734134166585232e-003,1.207341716154354e-002, 7.2443649
2.715297634171146e-003};*/
```

```
nect to each target.   Sequence ID: 0 Error Code: -1070 Error Class: 0x80000200
nect to each target.   Sequence ID: 0 Error Code: -1070 Error Class: 0x80000200
```

6. Add the other supporting .c files which configure the audio codec.



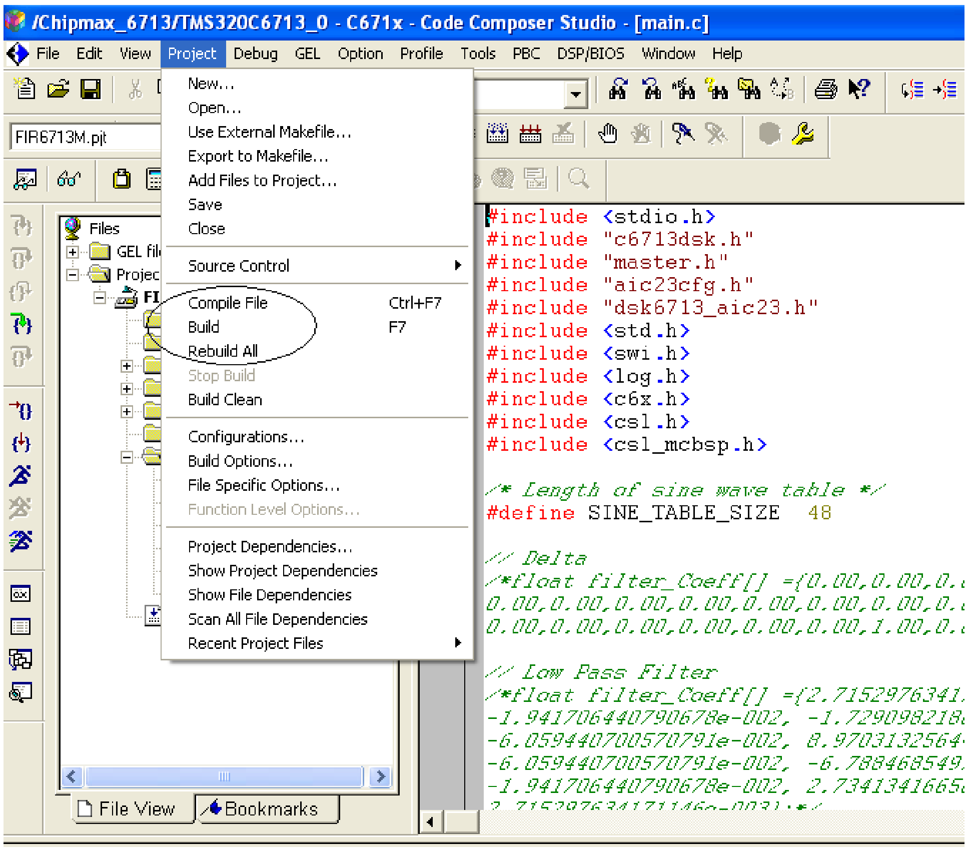
File name: "pll.c" "aic23.c" "aic23cfg_c.c" "dsk6713_aic2

Files of type: C Source Files (*.c;*.ccc)

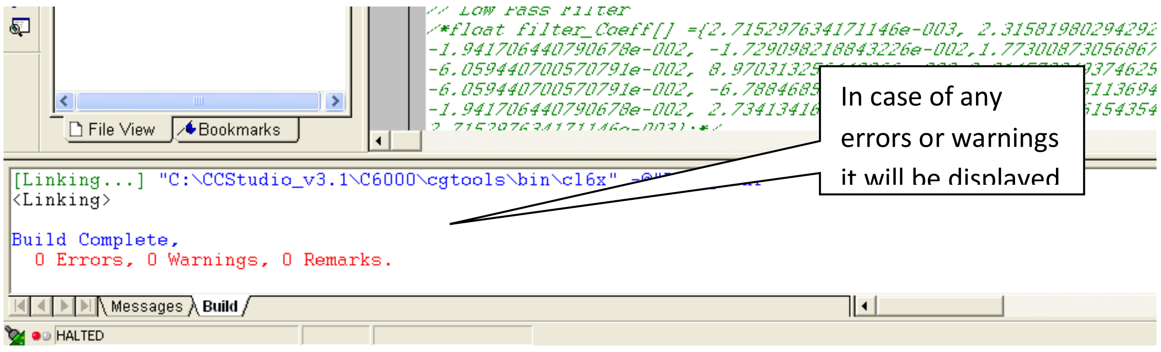
7. 7. a) Go to Project to Compile.

b) Go to Project to Build.

c) Go to Project to Rebuild All.



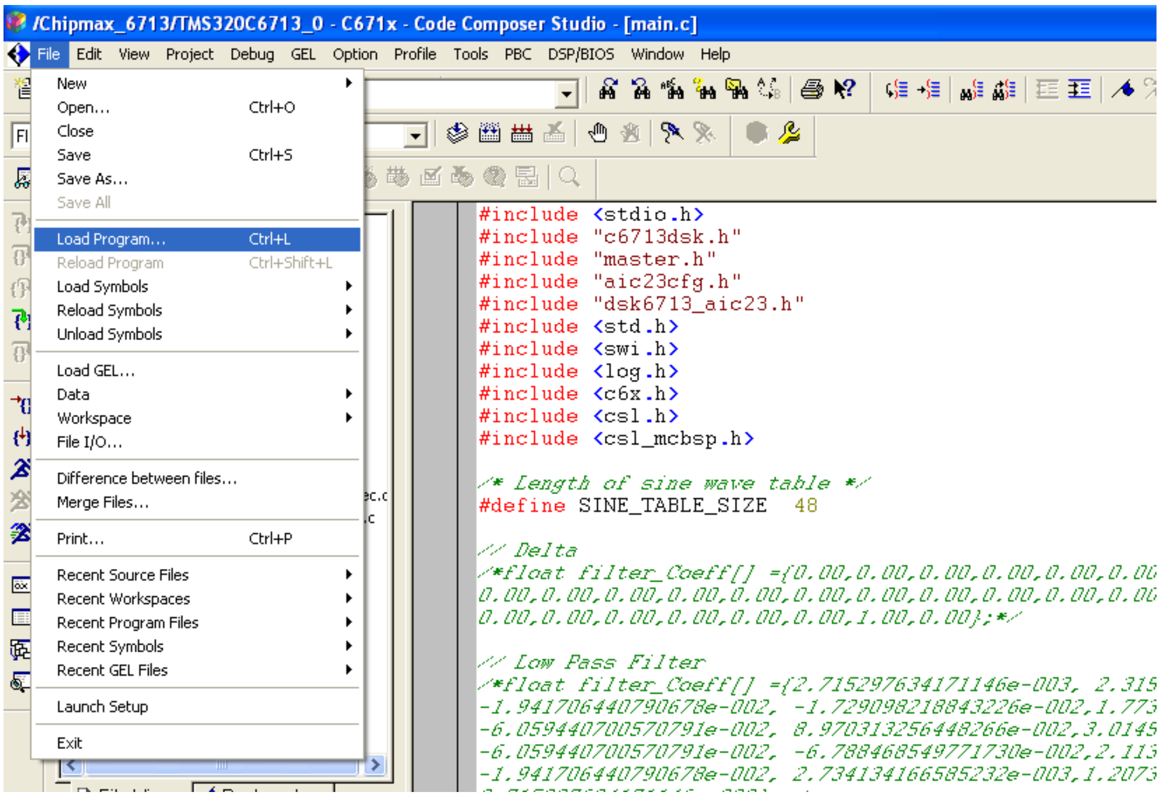
```
#include <stdio.h>
#include "c6713dsk.h"
#include "master.h"
#include "aic23cfg.h"
#include "dsk6713_aic23.h"
#include <std.h>
#include <swi.h>
#include <log.h>
#include <c6x.h>
#include <csl.h>
#include <csl_mcbsp.h>

/* Length of sine wave table */
#define SINE_TABLE_SIZE   48

// Delta
/*float filter_Coeff[] ={0.00,0.00,0.
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.
0.00,0.00,0.00,0.00,0.00,0.00,1.00,0.

// Low Pass Filter
/*float filter_Coeff[] ={2.7152976341
-1.9417064407906788e-002, -1.729098218
-6.0594407005707910e-002, 8.9703132564
-6.0594407005707910e-002, -6.788468549
-1.9417064407906788e-002, 2.7341341665
```



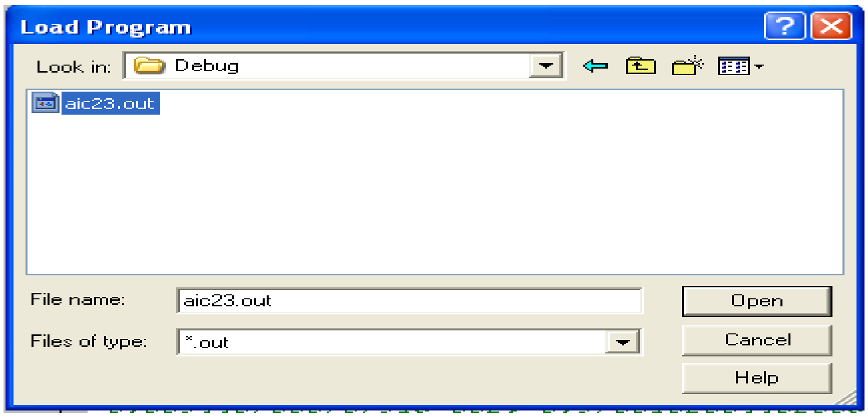
```
// Low Pass Filter
/*float filter_Coeff[] ={2.7152976341711460e-003, 2.31581980294292
-1.9417064407906788e-002, -1.7290982188432260e-002,1.77300873056867
-6.0594407005707910e-002, 8.97031325                              374625
-6.0594407005707910e-002, -6.7884685                              5113694
-1.9417064407906788e-002, 2.73413416                              5154354
```

```
[Linking...] "C:\CCStudio_v3.1\C6000\cgtools\bin\cl6x" -@"
<Linking>

Build Complete,
  0 Errors, 0 Warnings, 0 Remarks.
```

8. Go to file and load program and load **<u>".out"</u>** file into the board.



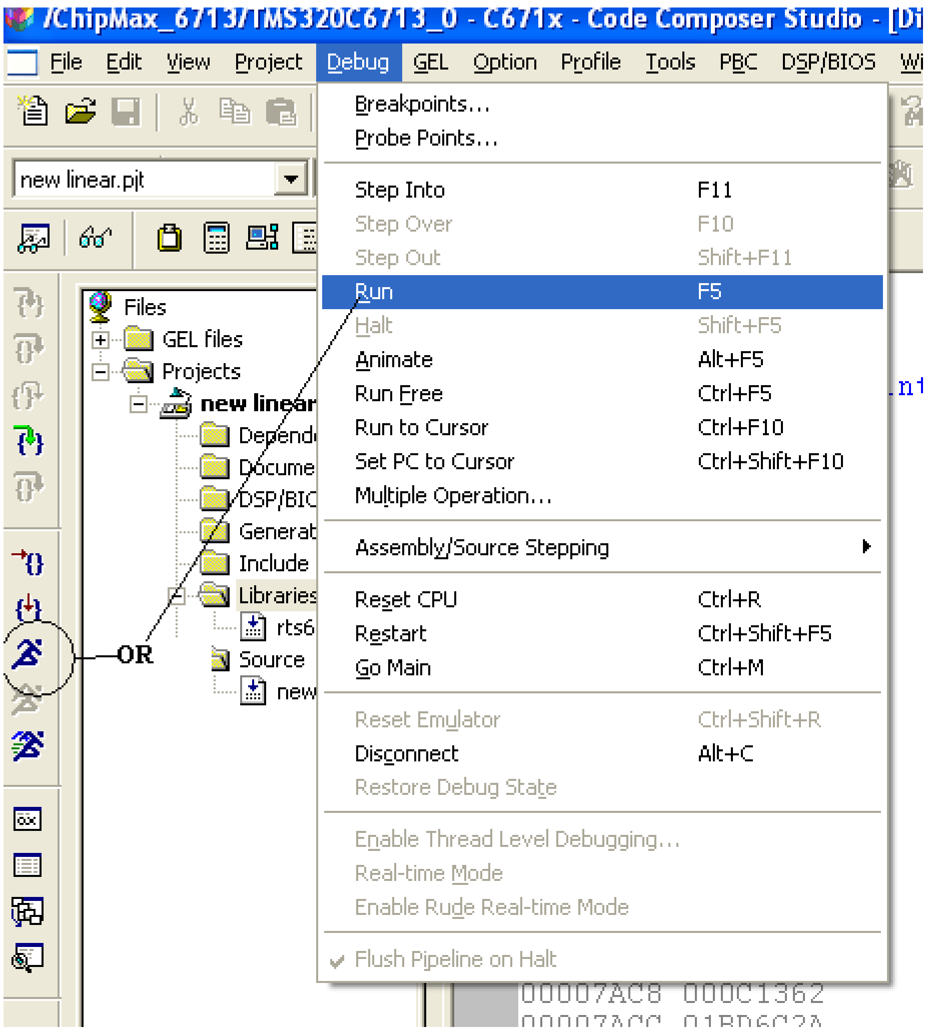
```
#include <stdio.h>
#include "c6713dsk.h"
#include "master.h"
#include "aic23cfg.h"
#include "dsk6713_aic23.h"
#include <std.h>
#include <swi.h>
#include <log.h>
#include <c6x.h>
#include <csl.h>
#include <csl_mcbsp.h>

/* Length of sine wave table */
#define SINE_TABLE_SIZE   48

// Delta
/*float filter_Coeff[] ={0.00,0.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,0.00,0.00,0.00,1.00,0.00};*/

// Low Pass Filter
/*float filter_Coeff[] ={2.715297634171146e-003, 2.315
-1.941706440790678e-002, -1.729098218843226e-002,1.773
-6.059440700570791e-002, 8.970313256448266e-002,3.0145
-6.059440700570791e-002, -6.788468549771730e-002,2.113
-1.941706440790678e-002, 2.734134166585232e-003,1.2073
```



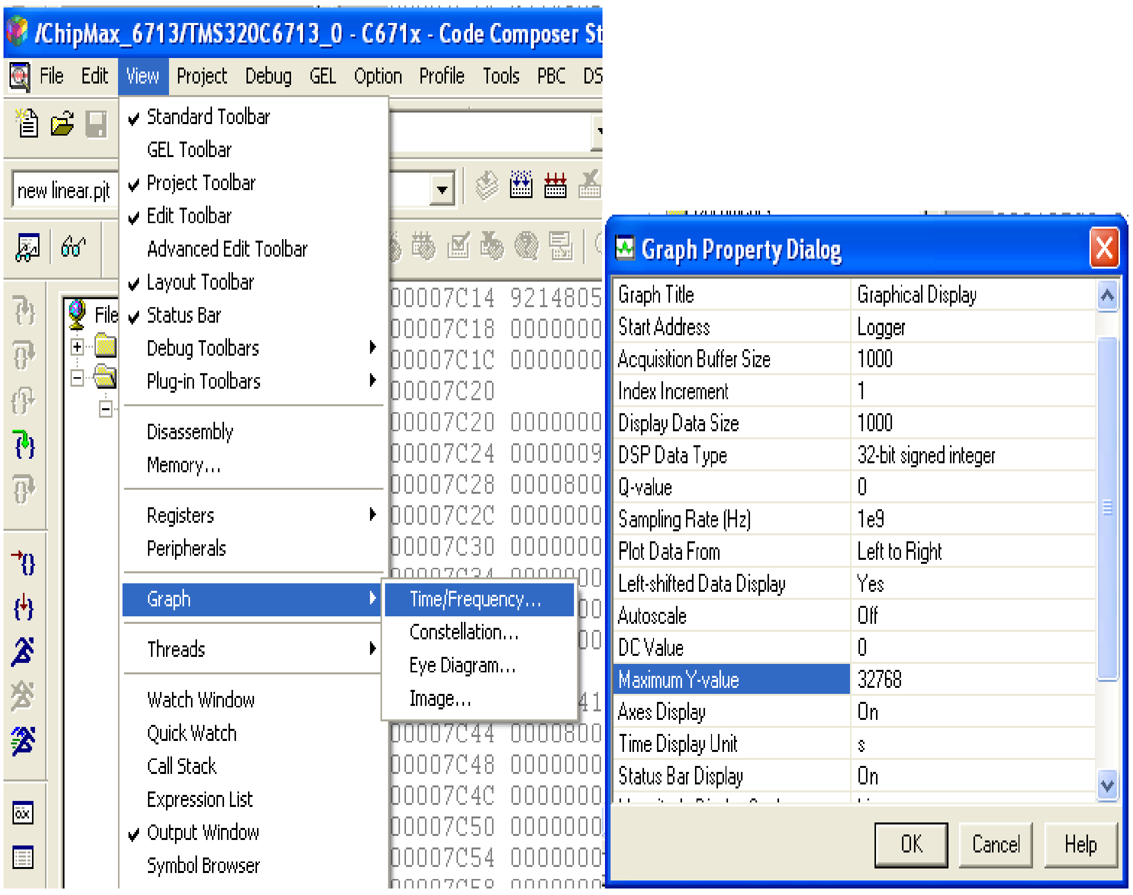
File name: aic23.out

Files of type: *.out

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

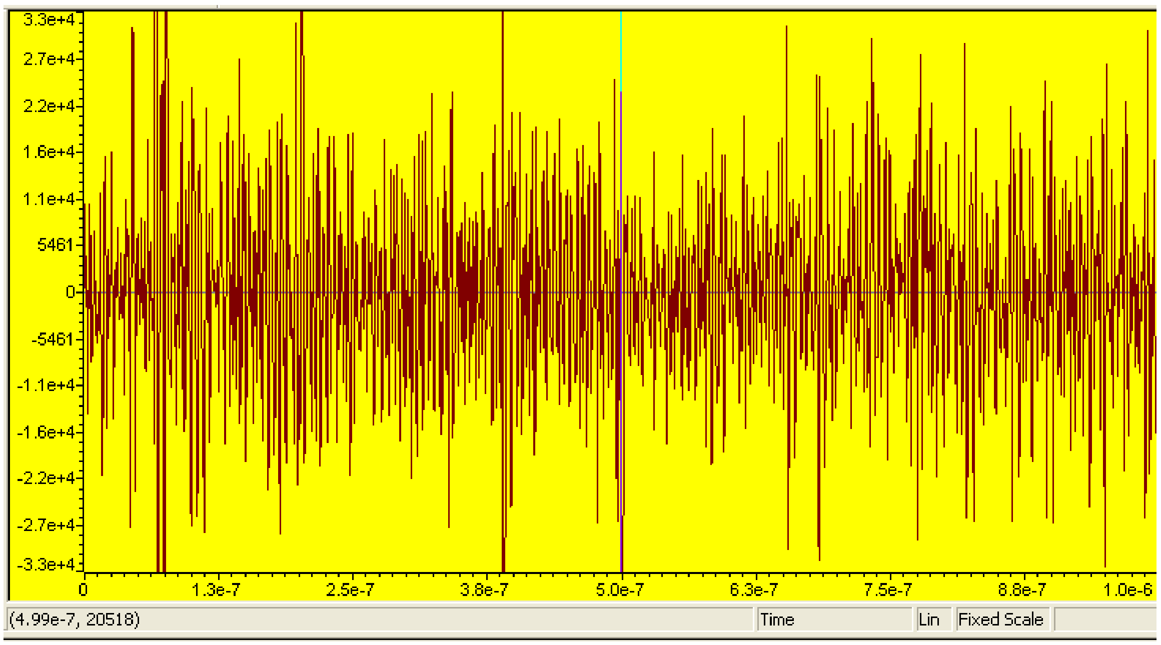

/ChipMax_6713/TMS320C6713_0 - C671x - Code Composer Studio - [Di
File Edit View Project Debug GEL Option Profile Tools PBC DSP/BIOS Wi
new linear.pjt
Files
GEL files
Projects
new linear
Depend
Docume
DSP/BIO
Generat
Include
Libraries
rts6
Source
new
OR
Breakpoints...
Probe Points...
Step Into F11
Step Over F10
Step Out Shift+F11
Run F5
Halt Shift+F5
Animate Alt+F5
Run Free Ctrl+F5
Run to Cursor Ctrl+F10
Set PC to Cursor Ctrl+Shift+F10
Multiple Operation...
Assembly/Source Stepping
Reset CPU Ctrl+R
Restart Ctrl+Shift+F5
Go Main Ctrl+M
Reset Emulator Ctrl+Shift+R
Disconnect Alt+C
Restore Debug State
Enable Thread Level Debugging...
Real-time Mode
Enable Rude Real-time Mode
Flush Pipeline on Halt
00007AC8 000C1362

10. 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.



Graph Title	Graphical Display
Start Address	Logger
Acquisition Buffer Size	1000
Index Increment	1
Display Data Size	1000
DSP Data Type	32-bit signed integer
Q-value	0
Sampling Rate (Hz)	1e9
Plot Data From	Left to Right
Left-shifted Data Display	Yes
Autoscale	Off
DC Value	0
Maximum Y-value	32768
Axes Display	On
Time Display Unit	s
Status Bar Display	On

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



References

[1] S. J. Orfanidis, Introduction to Signal Processing, online book, 2010, available from: http://www.ece.rutgers.edu/~orfanidi/intro2sp/

[2] R. Chassaing and D. Reay, Digital Signal Processing and Applications with the TMS320C6713 and TMS320C6416 DSK, 2nd ed., Wiley, Hoboken, NJ, 2008.

[3] D. R. Brown III, 2009 Workshop on Digital Signal Processing and Applications with the TMS320C6713 DSK, Parts 1 & 2, available online from:
http://spinlab.wpi.edu/courses/dspworkshop/dspworkshop_part1_2009.pdf
http://spinlab.wpi.edu/courses/dspworkshop/dspworkshop_part2_2009.pdf

[4] N. Dahnoun, "DSP Implementation Using the TMS320C6711 Processors," contained in the Texas Instruments "C6000 Teaching Materials" CD ROM, 2002-04, and available online from TI: http://www.ti.com/ww/cn/uprogram/share/ppt/c6000/Chapter1.ppt
http://www.ti.com/ww/cn/uprogram/share/ppt/c6000/Chapter2.ppt
http://www.ti.com/ww/cn/uprogram/share/ppt/c6000/Chapter3.ppt

[5] B. W. Kernighan and D. M. Ritchie, The C Programming Language, 2nd ed., Prentice Hall, Englewood Cliffs, NJ, 1988. S. P. Harbison and G. L. Steele, C: A Reference Manual, Prentice Hall, Englewood Cliffs, NJ, 1984. A. Kelly and I. Pohl, A Book on C, 2nd ed., Benjamin/Cummings, Redwood City, CA, 1990. GNU gcc, http://gcc.gnu.org/ DJGPP - Windows version of GCC, http://www.delorie.com/djgpp/ GCC Introduction, http://www.network-theory.co.uk/docs/gccintro/ 1 TMS320C6713 DSK AND CODE COMPOSER STUDIO 19

[6] C.R. Sullivan. “Extending the Karplus-Strong Algorithm to Synthesize Electric Guitar Timbres with Distortion and Feedback,” Computer Music J., 14, 26, (1990).