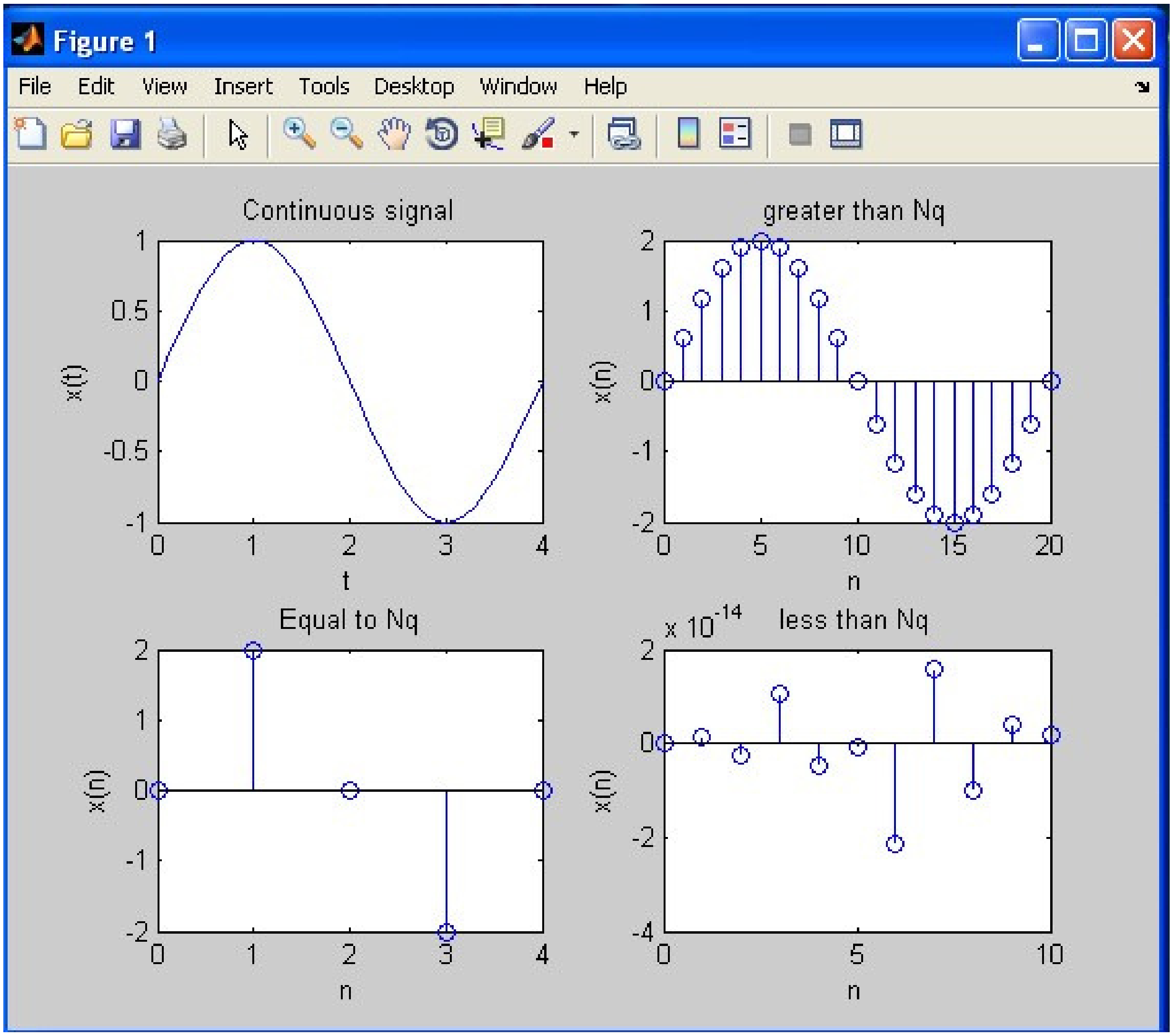
Subject Code	: **06ECL57**	IA Marks	: 25
No. of Practical Hrs/Week: 03		Exam Hours	: 03
Total no. of Practical Hrs.	: 42	Exam Marks	: 50

Error! No table of contents entries found.**EXPERIMENTS USING MATLAB**

1. Verification of Sampling theorem.

```
clc;
T=0.04; % Time period of 50 Hz signal
t=0:0.0005:0.02;
f = 1/T;
n1=0:40;
size(n1)

xa_t=sin(2*pi*2*t/T);

subplot(2,2,1);
plot(200*t,xa_t);
title('Verification of sampling theorem');
title('Continuous signal');
xlabel('t');
ylabel('x(t)');

ts1=0.002;%>niq rate
ts2=0.01;%=niq rate
ts3=0.1;%<niq rate

n=0:20;
x_ts1=2*sin(2*pi*n*ts1/T);
subplot(2,2,2);
stem(n,x_ts1);
title('greater than Nq');
xlabel('n');
ylabel('x(n)');
```

```
 n=0:4;
x_ts2=2*sin(2*sym('pi')*n*ts2/T);
subplot(2,2,3);
stem(n,x_ts2);
title('Equal to Nq');
xlabel('n');
ylabel('x(n)');

n=0:10;
x_ts3=2*sin(2*pi*n*ts3/T);
subplot(2,2,4);
stem(n,x_ts3);
title('less than Nq');
xlabel('n');
ylabel('x(n)');
```



2. Impulse response of a given system

```
clc;
clear all;
close all;
% Difference equation of a second order system
% y(n) = x(n)+0.5x(n-1)+0.85x(n-2)+y(n-1)+y(n-2)

b=input('enter the coefficients of x(n),x(n-1)-----');
a=input('enter the coefficients of y(n),y(n-1)----');
N=input('enter the number of samples of imp response ');

[h,t]=impz(b,a,N);
plot(t,h);
title('plot of impulse response');
ylabel('amplitude');
```

```
xlabel('time index----->N');
disp(h);
grid on;

Output
enter the coefficients of x(n),x(n-1)-----[1 0.5 0.85]
enter the coefficients of y(n),y(n-1)-----[1 -1 -1]
enter the number of samples of imp respons 4
    1.0000
    1.5000
    3.3500
    4.8500
```

Calculation

$$y(n) = x(n)+0.5x(n-1)+0.85x(n-2)+y(n-1)+y(n-2)$$

$$y(n) - y(n-1) - y(n-2) = x(n) + 0.5x(n-1) + 0.85x(n-2)$$

Taking Z transform on both sides,

$$Y(Z) - Z^{-1} Y(Z) - Z^{-2} Y(Z) = X(Z) + 0.5 Z^{-1} X(Z) + 0.85 Z^{-2} X(Z)$$

$$Y(Z)[1 - Z^{-1} - Z^{-2}] = X(Z)[1 + 0.5 Z^{-1} + 0.85 Z^{-2}]$$

But, $H(Z) = Y(Z)/X(Z)$

$$= [1 + 0.5 Z^{-1} + 0.85 Z^{-2}]/ [1 - Z^{-1} - Z^{-2}]$$

By dividing we get

$$H(Z) = 1 + 1.5 Z^{-1} + 3.35 Z^{-2} + 4.85 Z^{-3}$$

$$h(n) = [1\ 1.5\ 3.35\ 4.85]$$

3. Linear convolution of two given sequences.

```
% Linear convolution using conv command
```

Using CONV command.

```
clc;
x1=input('enter the first sequence');
subplot(3,1,1);
stem(x1);
ylabel('amplitude');
title('plot of the first sequence');

x2=input('enter 2nd sequence');
subplot(3,1,2);
stem(x2);
```

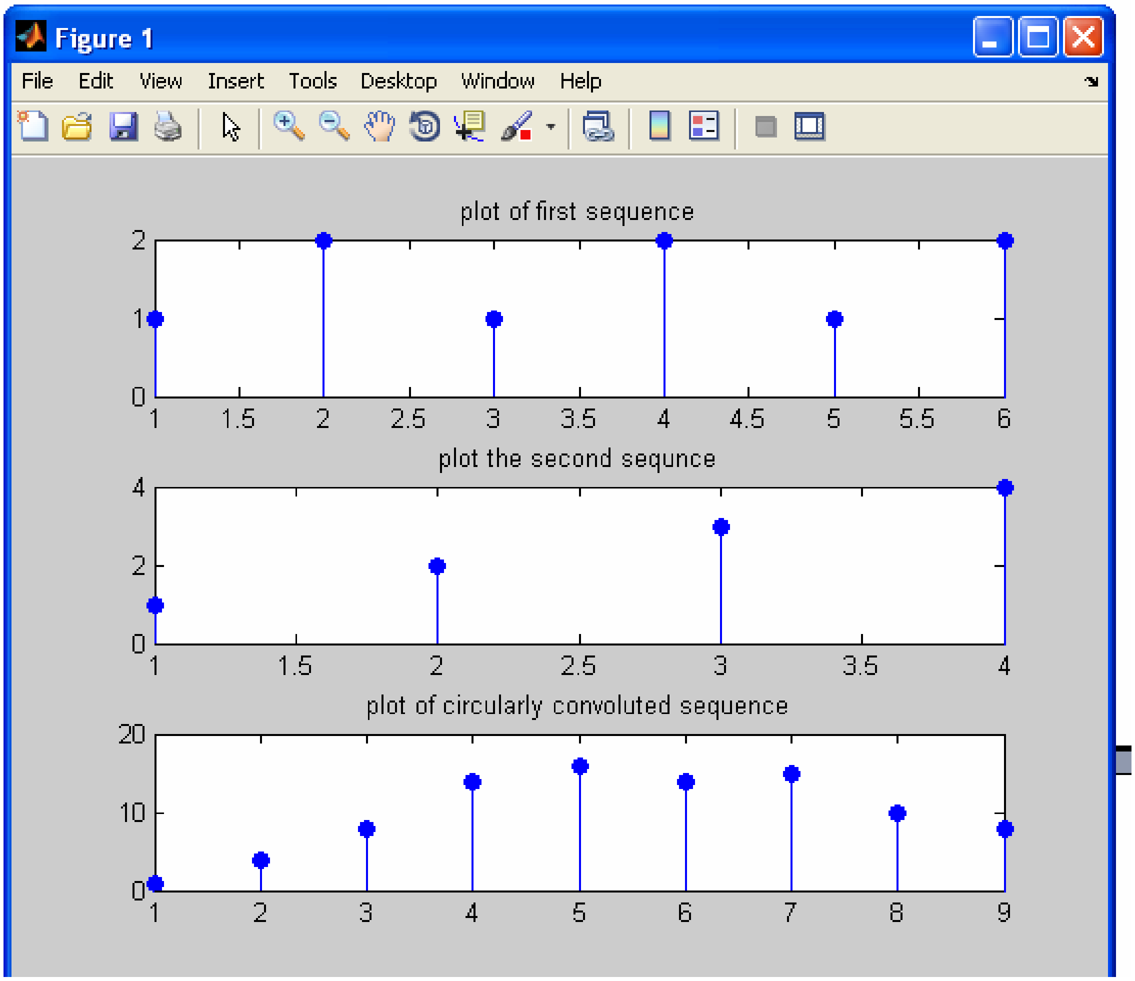
```
ylabel('amplitude');
title('plot of 2nd sequence');

f=conv(x1,x2);
disp('output of linear conv is');
disp(f);
xlabel('time index n');
ylabel('amplitude f');
subplot(3,1,3);
stem(f);
title('linear conv of sequence');
```

```
Output
enter the first sequence[1 2 3]
enter 2nd sequence[1 2 3 4]
output of linear conv is
     1     4    10    16    17    12
```

Linear convolution Using DFT and IDFT / Linear convolution using circular convolution

```
clc;
clear all;
x1=input('enter the first sequence');
x2=input('enter the second sequence');
n=input('enter the no of points of the dft');

subplot(3,1,1);
stem(x1,'filled');
title('plot of first sequence');

subplot(3,1,2);
stem(x2,'filled');
title('plot the second sequnce');

n1 = length(x1);
n2 = length(x2);
m = n1+n2-1; % Length of linear convolution

x = [x1 zeros(1,n2-1)]; % Padding of zeros to make it of
                         % length m
y = [x2 zeros(1,n1-1)];

x_fft = fft(x,m);
y_fft = fft(y,m);
dft_xy = x_fft.*y_fft;
y=ifft(dft_xy,m);

disp('the circular convolution result is ......');
disp(y);

subplot(3,1,3);
stem(y,'filled');
title('plot of circularly convoluted sequence');
```

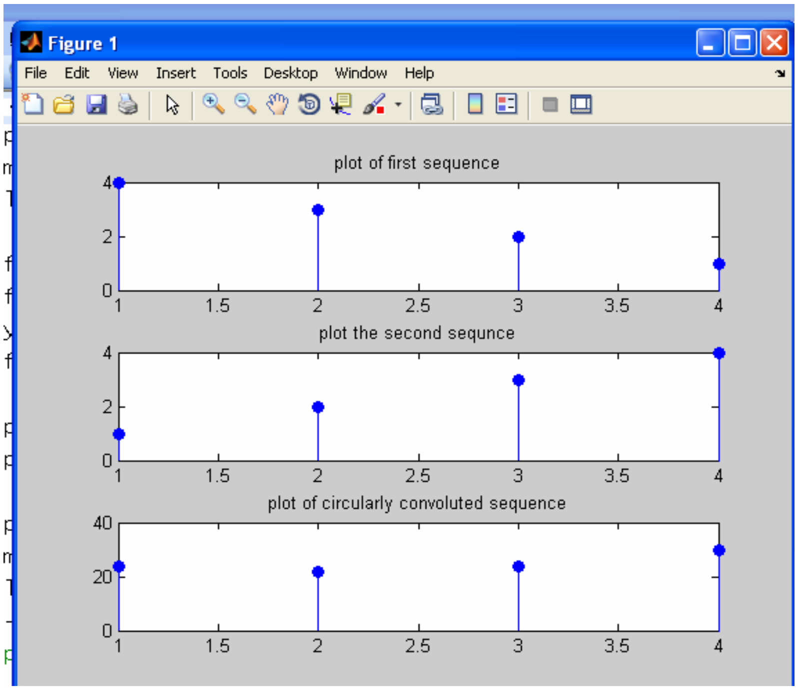
```
Output
enter the first sequence[1 2 1 2 1 2]
enter the second sequence[1 2 3 4]
the circular convolution result is ......
1.0000    4.0000    8.0000   14.0000   16.0000   14.0000
15.0000   10.0000    8.0000
```



4. Circular convolution of two given sequences

```
clc;
clear all;
x1=input('enter the first sequence');
x2=input('enter the second sequence');
n1 = length(x1);
n2 = length(x2);

subplot(3,1,1);
stem(x1,'filled');
title('plot of first sequence');

subplot(3,1,2);
stem(x2,'filled');
title('plot the second sequnce');

y1=fft(x1,n);
y2=fft(x2,n);
y3=y1.*y2;
y=ifft(y3,n);

disp('the circular convolution result is ......');
disp(y);

subplot(3,1,3);
stem(y,'filled');
title('plot of circularly convoluted sequence');
----------*******-------------
```

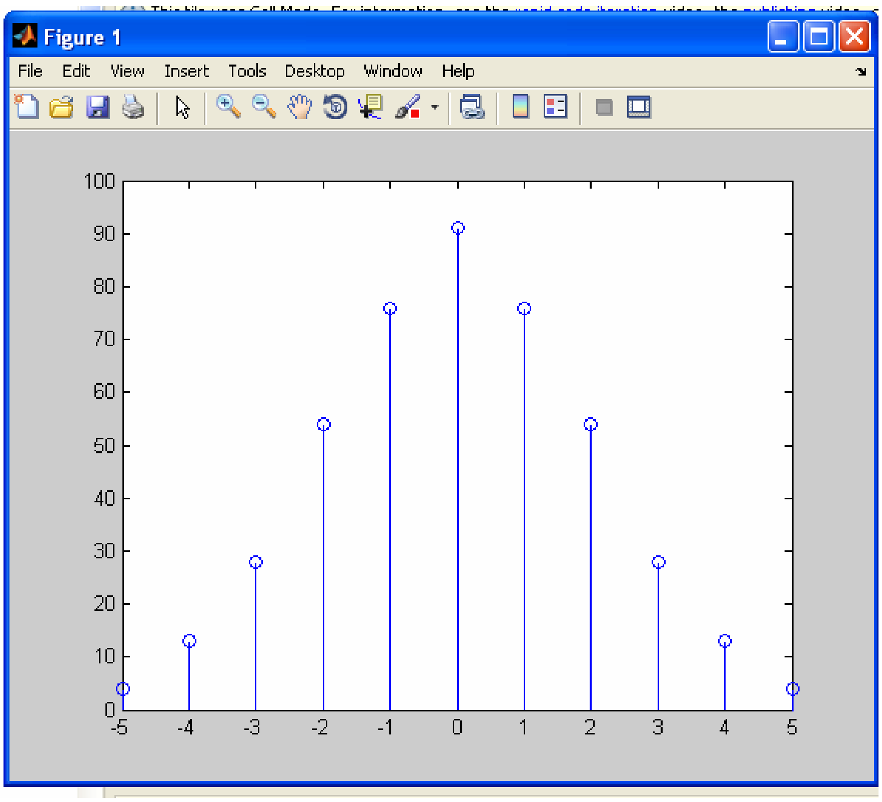
```
Output

enter the first sequence[1 2 3 4]
enter the second sequence[4 3 2 1]
the circular convolution result is ......
    24    22    24    30
```



5. Autocorrelation of a given sequence and verification of its properties.

```
% Read the signal
x=[1,2,3,6,5,4]
% define the axis
n=0:1:length(x)-1
% plot the signal
stem(n,x);
xlabel('n');
% auto correlate the signal
Rxx=xcorr(x,x);
% the axis for auto correlation results
nRxx=-length(x)+1:length(x)-1
% display the result
stem(nRxx,Rxx)

% properties of Rxx(0) gives the energy of the signal
% find energy of the signal
energy=sum(x.^2)

%set index of the centre value
centre_index=ceil(length(Rxx)/2)

% Acces the centre value Rxx(0)
Rxx_0==Rxx(centre_index)
Rxx_0==Rxx(centre_index)

% Check if the Rxx(0)=energy
if Rxx_0==energy
   disp('Rxx(0) gives energy proved');
 else
   disp('Rxx(0) gives energy not proved');
end
Rxx_right=Rxx(centre_index:1:length(Rxx))
Rxx_left=Rxx(centre_index:-1:1)
if Rxx_right==Rxx_left
    disp('Rxx is even');
 else
    disp('Rxx is not even');
```

```
end
x =    1      2      3      6      5      4
n =    0      1      2      3      4      5
nRxx =  -5    -4    -3    -2    -1     0     1     2     3          4
       5

energy =   91
centre_index = 6
Rxx(0) gives energy not proved
Rxx_right =
91.0000   76.0000   54.0000   28.0000   13.0000    4.0000
Rxx_left =
91.0000   76.0000   54.0000   28.0000   13.0000    4.0000
Rxx is even
```



6. Solving a given difference equation.

```
X = [1 2 3 4];
% Compute the output sequences
xcoeff = [0.5 0.27 0.77]; % x(n), x(n-1), x(n-2)… coefficients

y1 = filter(xcoeff,ycoeff,x); % Output of System
% Plot the output sequences
subplot(2,1,1);
plot(n,y1);
ylabel('Amplitude');

----------------**********---------------------

% to find out h(n) of the difference equation
% y(n)-(1/2)*y(n-1) = (1/2)*x(n)+(1/2)*x(n-1)
  For manual calculation of h(n), take the Z transform on both sides,
  find H(Z)=Y(Z)/X(Z). Take inverse Z transform to get h(n)

b=input('enter the coefficients of x(n),x(n-1)-----');
a=input('enter the coefficients of y(n),y(n-1)----');
N=input('enter the number of samples of imp respons');

[h,t]=impz(b,a,N);
plot(t,h);
title('plot of impulse response');
ylabel('amplitude');
xlabel('time index----->N');
disp(h);
grid on;
```

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

```
N = input('Enter the the value of N(Value of N in N-Point DFT)');
x = input('Enter the sequence for which DFT is to be calculated');
n=[0:1:N-1];
k=[0:1:N-1];
WN=exp(-1j*2*pi/N);
nk=n'*k;
WNnk=WN.^nk;
Xk=x*WNnk;
MagX=abs(Xk)            % Magnitude of calculated DFT
PhaseX=angle(Xk)*180/pi % Phase of the calculated DFT
figure(1);
subplot(2,1,1);
plot(k,MagX);
subplot(2,1,2);
plot(k,PhaseX);

-------------*******--------------
OUTPUT
Enter the the value of N(Value of N in N-Point DFT)4
Enter the sequence for which DFT is to be calculated
[1 2 3 4]

MagX =  10.0000    2.8284    2.0000    2.8284


PhaseX =  0  135.0000 -180.0000 -135.0000


DFT of the given sequence is
  10.0000             -2.0000 + 2.0000i  -2.0000 - 0.0000i  -2.0000 -
   2.0000i
```

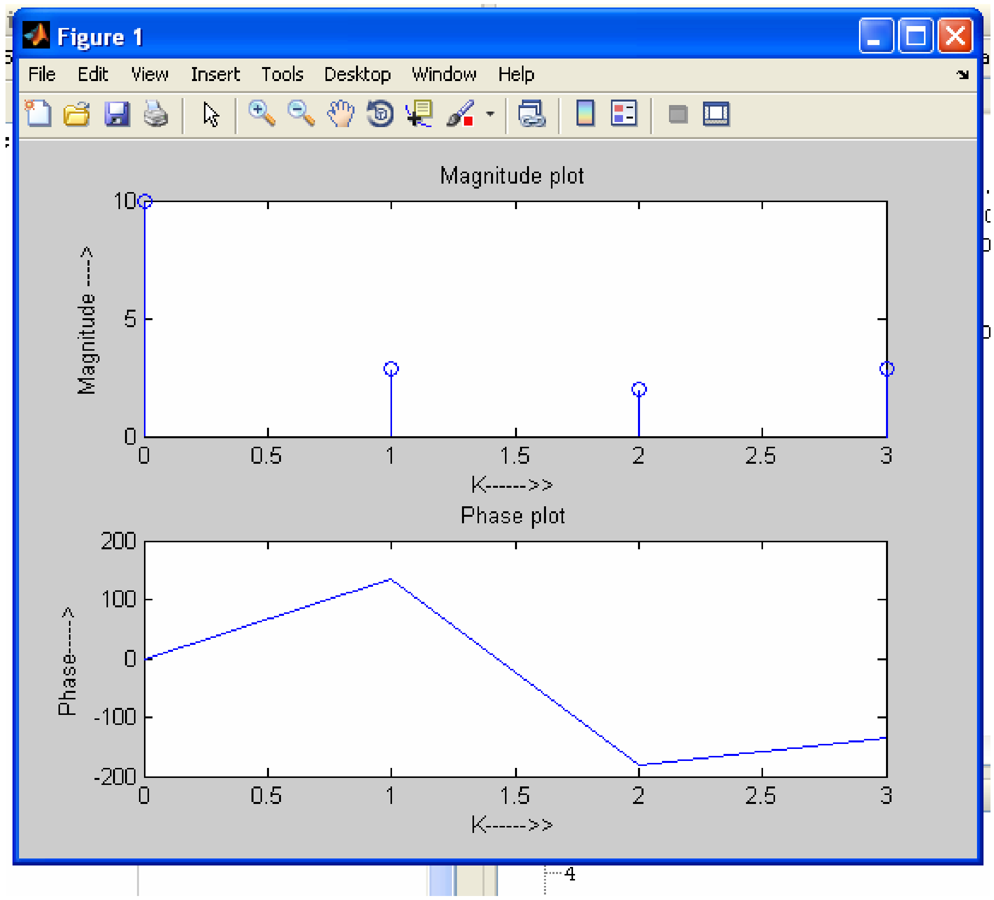


Figure 1
File Edit View Insert Tools Desktop Window Help
Magnitude plot
Magnitude ---->
10
5
0
0 0.5 1 1.5 2 2.5 3
K----->>
Phase plot
Phase----->
200
100
0
-100
-200
0 0.5 1 1.5 2 2.5 3
K----->>

8. Circular convolution of two given sequences using DFT and IDFT

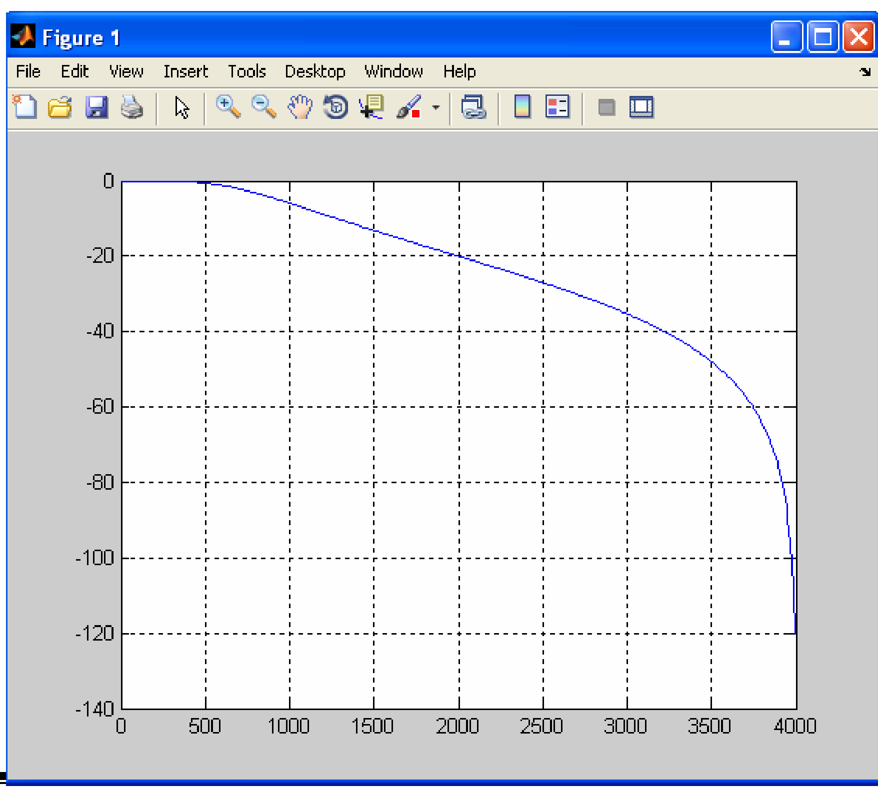
```
clc; % Program for circular convolution
clear all;
x1=input('enter the first sequence');
x2=input('enter the second sequence');
n=input('enter the no of points of the dft');

subplot(3,1,1);
stem(x1,'filled');
title('plot of first sequence');

subplot(3,1,2);
stem(x2,'filled');
title('plot the second sequnce');

y1=fft(x1,n);
y2=fft(x2,n);
y3=y1.*y2;
y=ifft(y3,n);

disp('the circular convolution result is ......');
disp(y);

subplot(3,1,3);
stem(y,'filled');
title('plot of circularly convoluted sequence');
```

Output

```
enter the first sequence[4 3 2 1]
enter the second sequence[1 2 3 4]
enter the no of points of the dft4
the circular convolution result is ......
    24    22    24    30
```

9. Design and implementation of IIR BUTTERWORTH filter to meet given specifications.

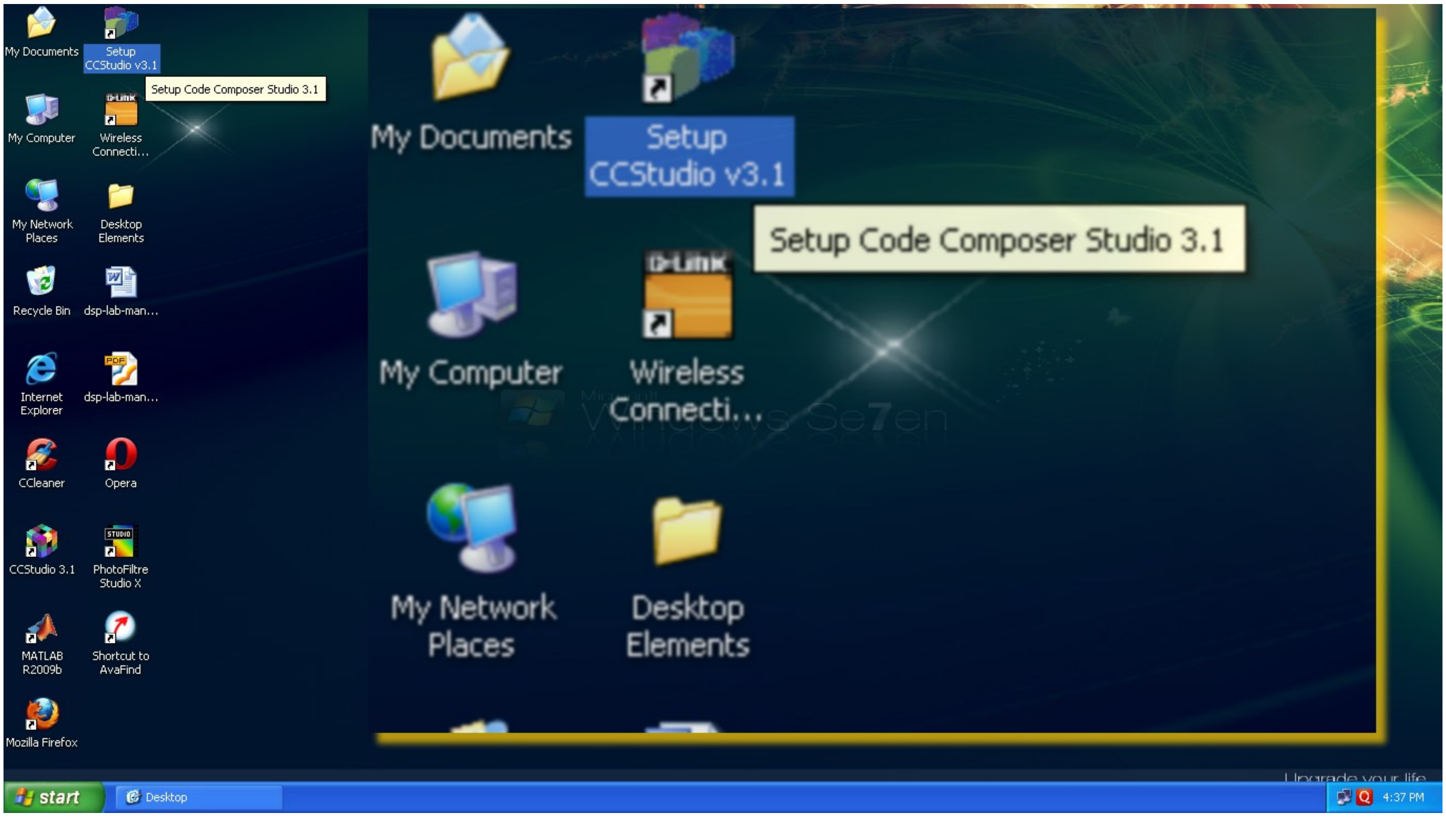
```
clc;
clear all;
close all;
wp=500;  % Enter the pass band frequency
ws=2000; % Enter the stop band frequency

Rp=3;     % Enter the pass band ripple
Rs=20;    % Enter the stop band attenuation
Fs=8000;  % Enter the sampling frequency
Fn=Fs/2;  % Normalized sampling frequency

% Find the order n and cutt off frequency
[n,wc]=buttord(wp/Fn,ws/Fn,Rp,Rs);
% Find the filter co-efficients
 [b,a]=butter(n,wc);
disp(n)
% Plot the frequency response
[h,f]=freqz(b,a,512,8000);
plot(f,20*log10(abs(h)))
grid;
```



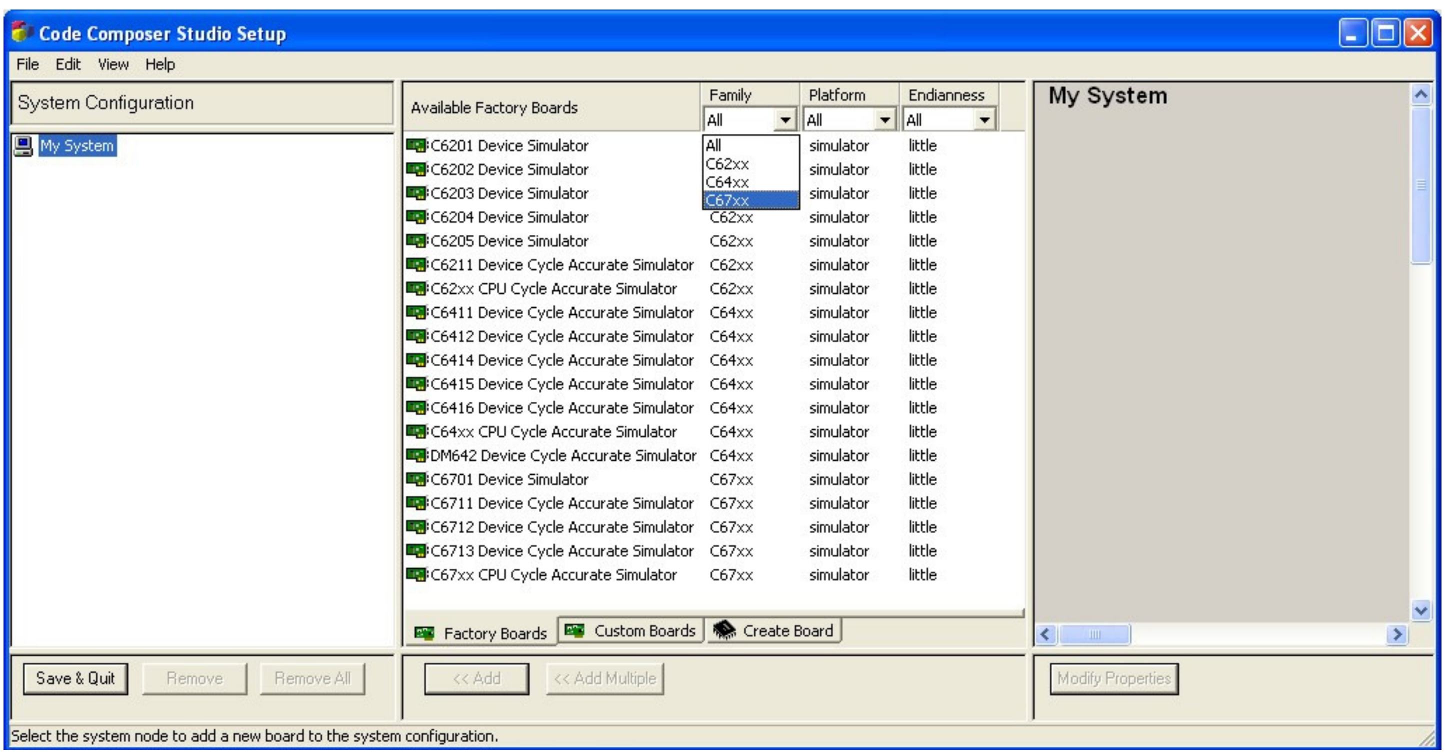
Part B: EXPERIMENTS USING DSP PROCESSOR

Procedure for execution in TMS3206713 Simulator

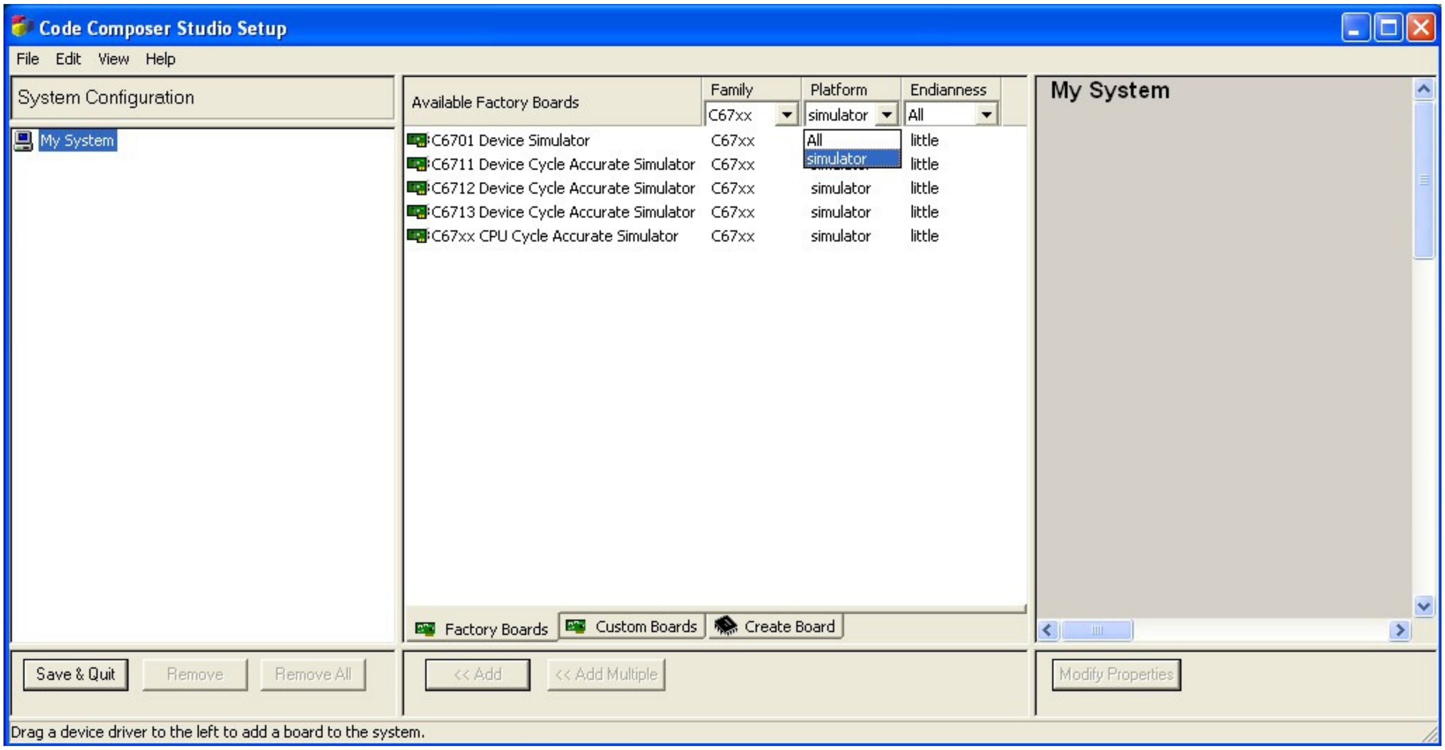
- Open CCS Studio Setup3.1



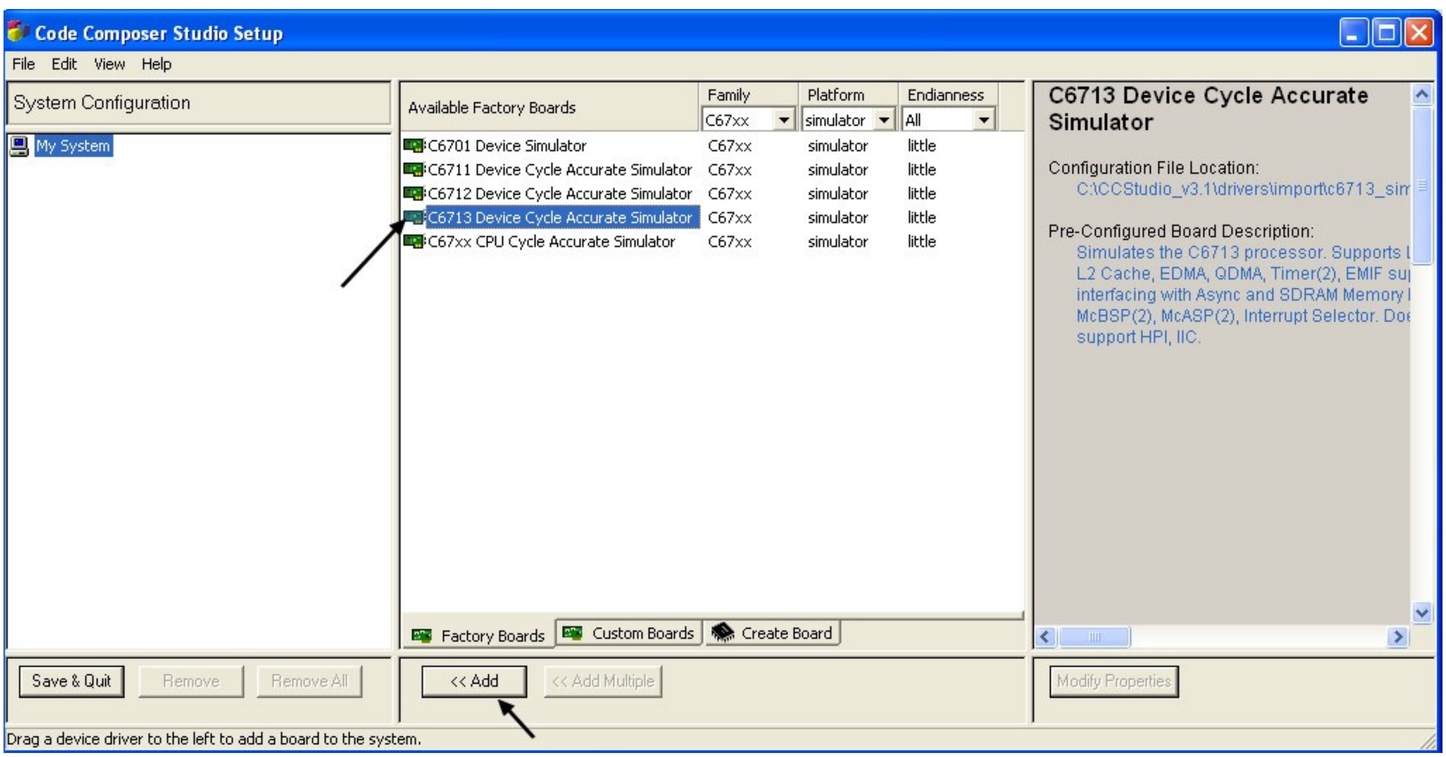
- Select Family→ 67xx



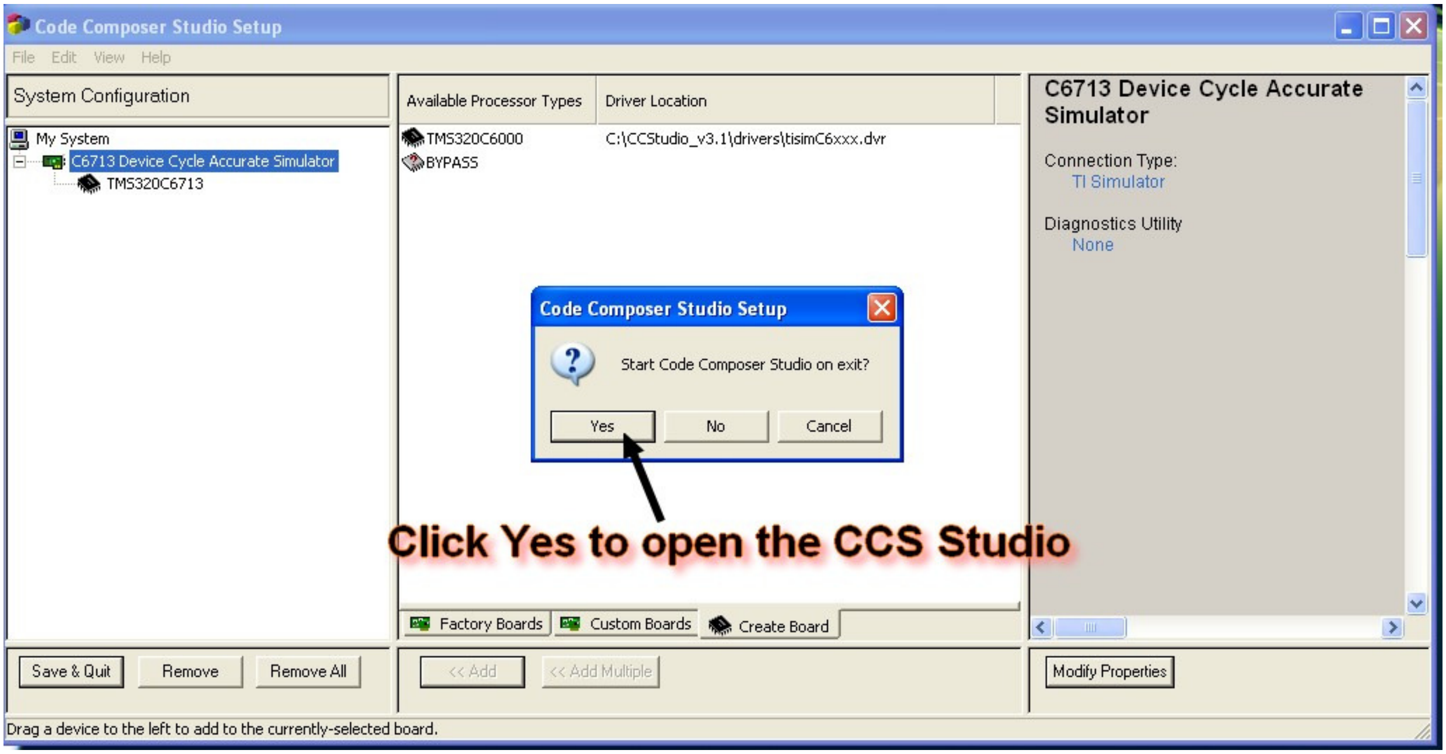
- Platform → Simulator



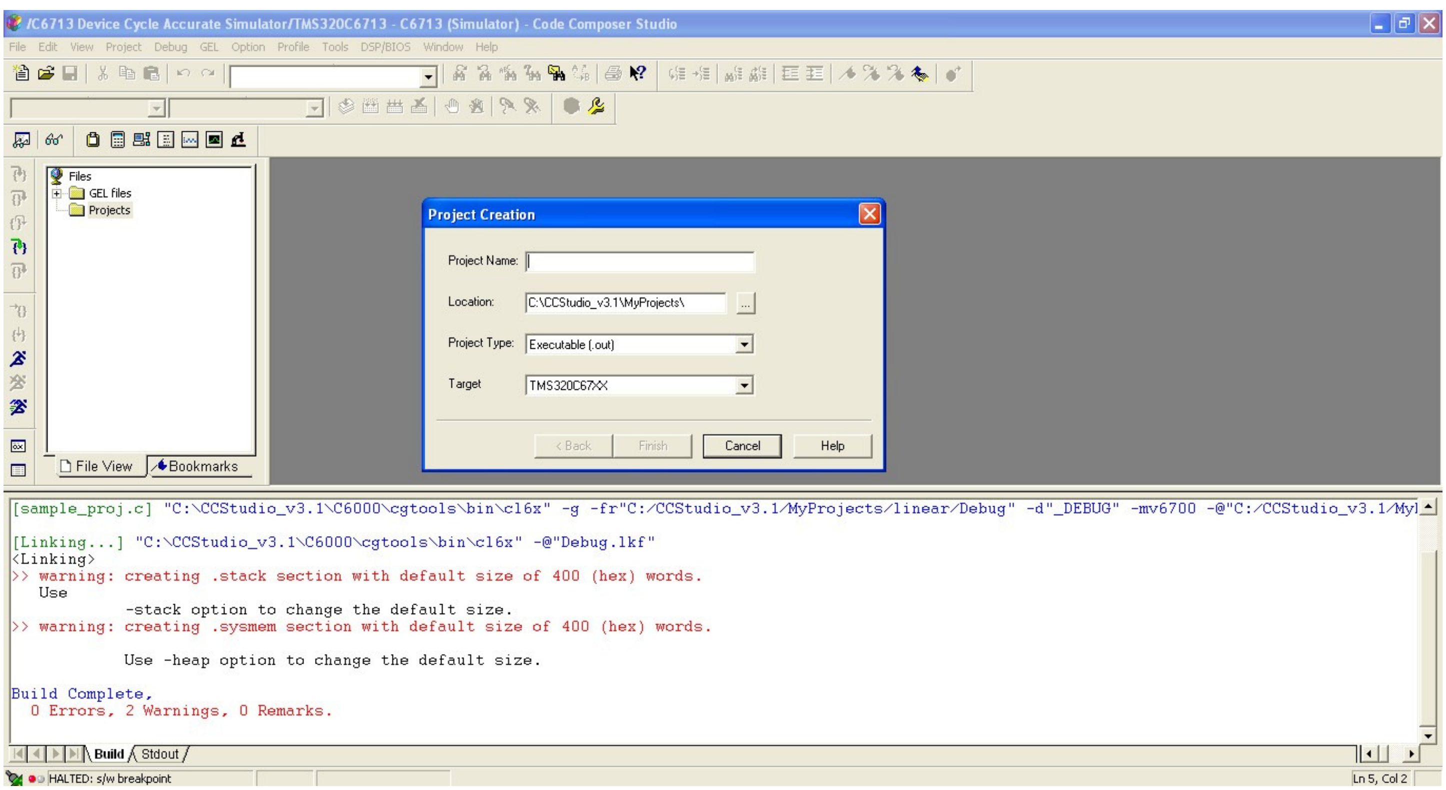
- Select 6713 Device cycle accurate simulator



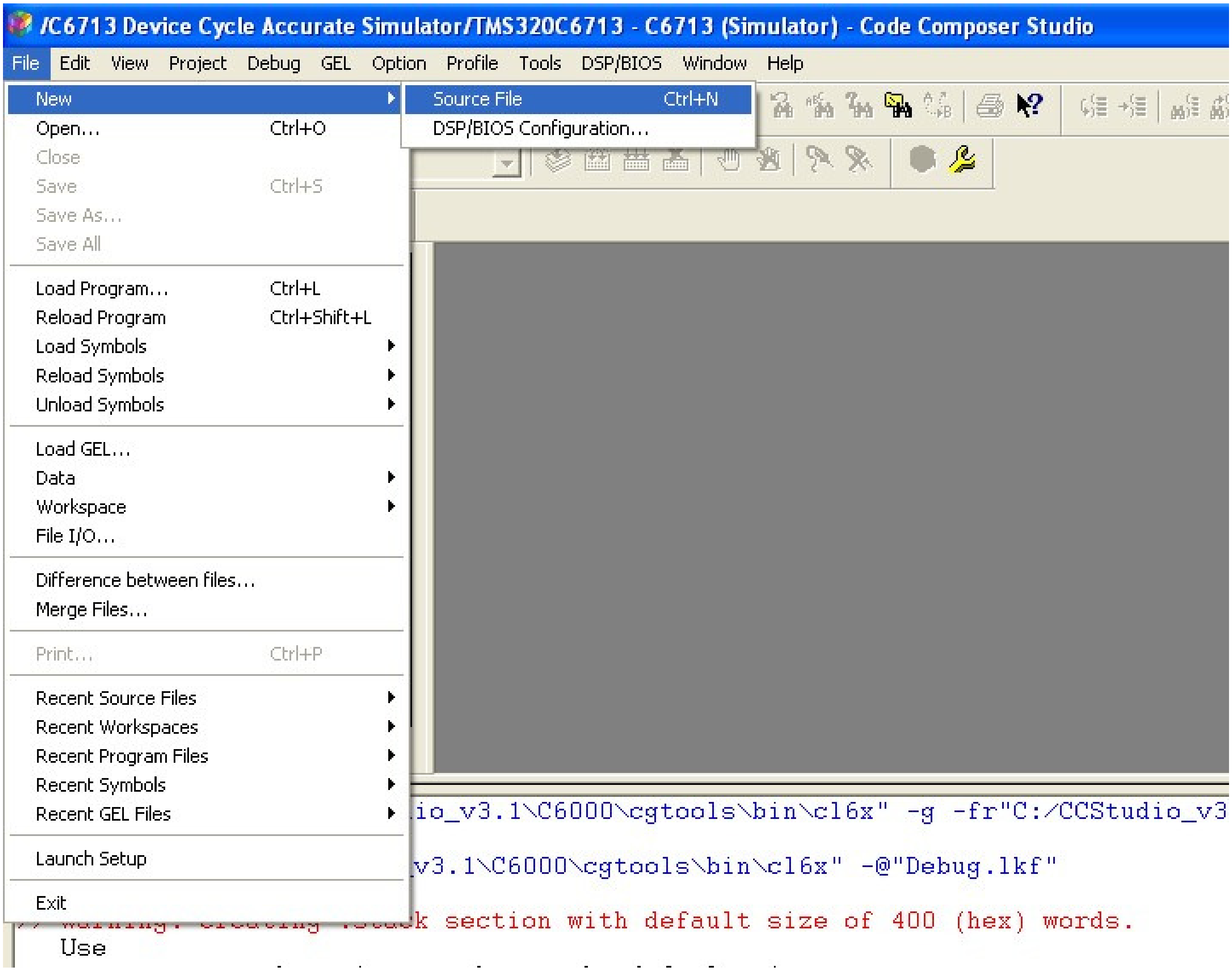
- Select Little endian. If little endian is not selected, building/linking error can occur. Add it to the left panel. Save and quit.



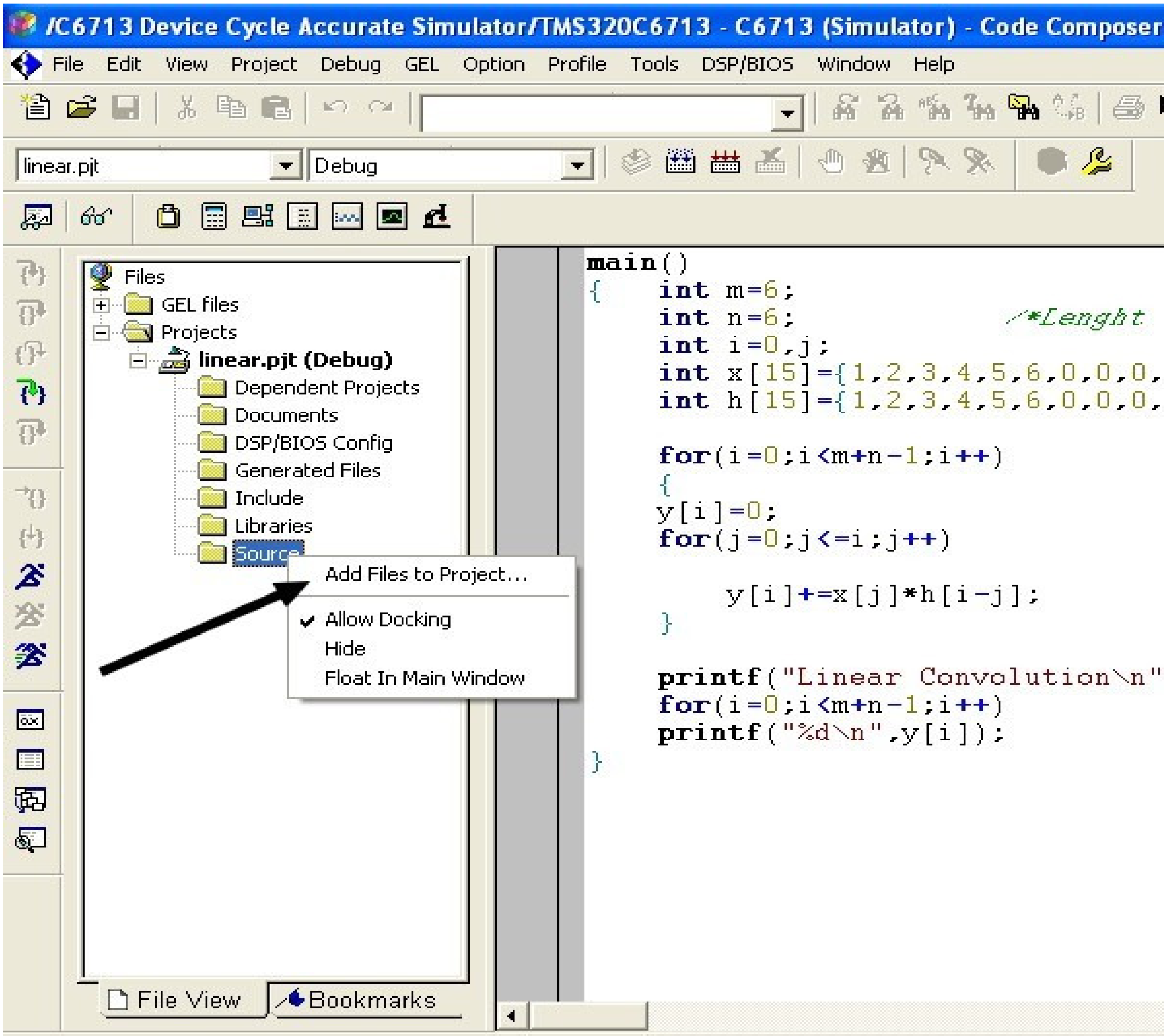
- Project→New→Project Name→ Location(Location of project) → Project type (.out Executable) → Target (TMS320C67xx)



- Write the code in a new source file. Save it in the project folder with .C file format.



- Write the code in a new source file. Save it in the project folder with .C file format.
- Add this to the project. Project will be having .pjt extension. Right click on .pjt file created, add the .c file you have written.



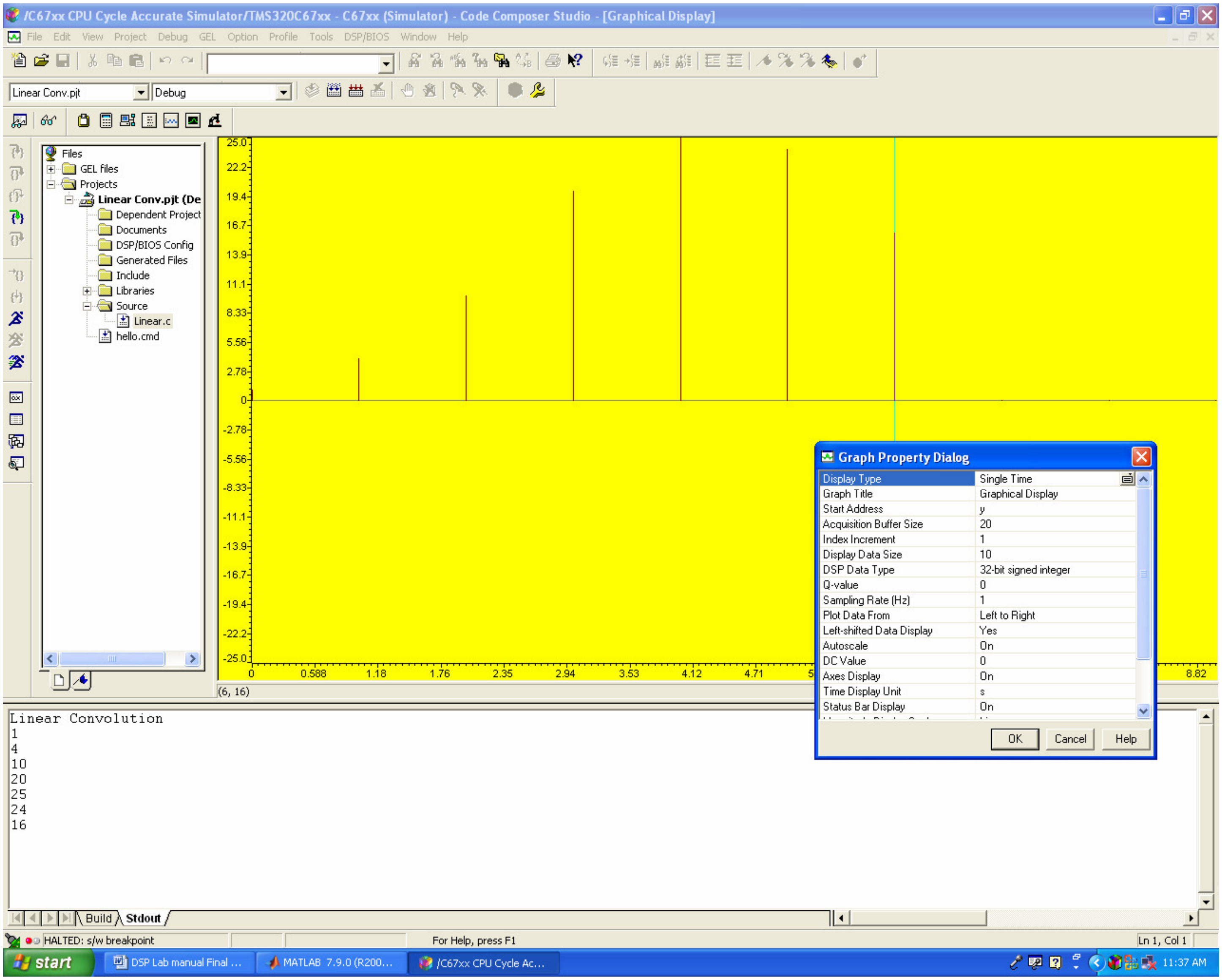
- Two other files are to be added to project. One is library file (*.lib) and other is Linker command file (*.cmd)
- Add rts6713.lib C:\CCStudio_v3.1\C6000\cgtools\librts6700.lib
- Add hello.cmd C:\CCStudio_v3.1\tutorial\dsk6713\hello1\hello.cmd
- Debug→ Build
- File→ Load Program(Often this is the most comman mistake to forget this..!) Load the .out file which is in the DEBUG folder of the project folder. Select this and open.
- Debug → Debug Run

Procedure for execution in TMS320DSK6713 kit

- CCS Studio Setup v3.1 → Family (67xx) → Platform (dsk) → Endianness → Little endian→ Add it to panel. Click on 6713dsk, save and quit.
- Connect the power card to the DSP kit.
- Connect the data cable → USB from PC
- After getting the project window, DEBUG→ CONNECT.
- Rest of the procedure is same as compared to simulator running.

1. Linear convolution of two given sequences.

```
 /* prg to implement linear convolution */
#include<stdio.h>
#include<math.h>
int y[20];

main()
{ int m=6;                              /*Lenght of i/p samples sequence*/
  int n=6;             /*Lenght of impulse response Co-efficients */
  int i=0,j;
  int x[15]={1,2,3,4,5,6,0,0,0,0,0,0};   /*Input Signal Samples*/
  int h[15]={1,2,3,4,5,6,0,0,0,0,0,0};   /*Impulse     Response     Co-
efficients*/

  for(i=0;i<m+n-1;i++)
  {
  y[i]=0;
  for(j=0;j<=i;j++)

      y[i]+=x[j]*h[i-j];
  }

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



Verification using matlab

```
x = [1,2,3,4];
 y = [1,2,3,4];
output = conv(x,y)

output =

     1     4    10    20    25    24    16
```

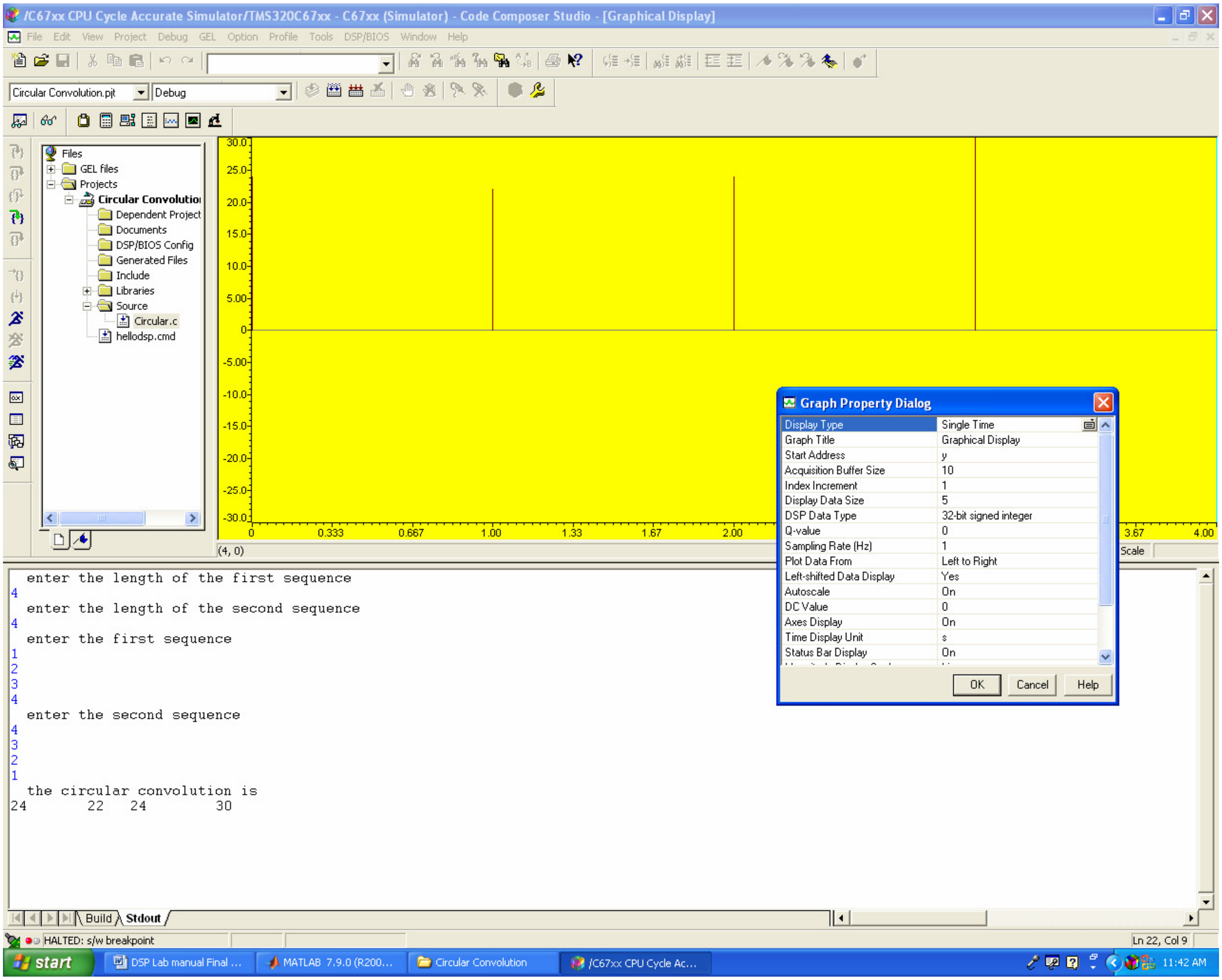
2. Circular convolution of two given sequences.

```
#include<stdio.h>
 #include<math.h>
 int m,n,x[30],h[30],y[30],i,j,temp[30],k,x2[30],a[30];
 void main()
 {
   printf("  enter the length of the first sequence\n");
   scanf("%d",&m);
   printf("  enter the length of the second sequence\n");
   scanf("%d",&n);
   printf("  enter the first sequence\n");
   for(i=0;i<m;i++)
   scanf("%d",&x[i]);
   printf("  enter the second sequence\n");
   for(j=0;j<n;j++)
   scanf("%d",&h[j]);
   if(m-n!=0)        /*If length of both sequences are not equal*/
   {
      if(m>n)        /* Pad the smaller sequence with zero*/
      {
         for(i=n;i<m;i++)
           h[i]=0;
        n=m;
      }
      for(i=m;i<n;i++)
      x[i]=0;
      m=n;
   }
   y[0]=0;
   a[0]=h[0];
   for(j=1;j<n;j++)                    /*folding h(n) to h(-n)*/
   a[j]=h[n-j];
     /*Circular convolution*/
  for(i=0;i<n;i++)
    y[0]+=x[i]*a[i];
  for(k=1;k<n;k++)
```

```
 {
    y[k]=0;
    /*circular shift*/
    for(j=1;j<n;j++)
      x2[j]=a[j-1];
    x2[0]=a[n-1];
    for(i=0;i<n;i++)
    {
       a[i]=x2[i];
       y[k]+=x[i]*x2[i];
    }
 }
 /*displaying the result*/
 printf("  the circular convolution is\n");
 for(i=0;i<n;i++)
 printf("%d \t",y[i]);

 }
--------------*********--------------
```

Verification of circular convolution using matlab

```
x1=[1 2 3 4];
x2=[4 3 2 1];
n = 4;

X1=fft(x1,n);
X2=fft(x2,n);
Y=X1.*X2;
y=ifft(Y,n);
disp(y)
y = 24 22 24 30
```



/C67xx CPU Cycle Accurate Simulator/TMS320C67xx - C67xx (Simulator) - Code Composer Studio - [Graphical Display]
Graph Property Dialog
Display Type Single Time
Graph Title Graphical Display
Start Address y
Acquisition Buffer Size 10
Index Increment 1
Display Data Size 5
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
enter the length of the first sequence
4
enter the length of the second sequence
4
enter the first sequence
1
2
3
4
enter the second sequence
4
3
2
1
the circular convolution is
24 22 24 30
HALTED: s/w breakpoint

3. Computation of N- Point DFT of a given sequence

```
#include<stdio.h>
#include<math.h>

void main()
{
     short N = 8;
     short x[8] = {1,2,3,4,5,6,7,0}; // test data
     float pi = 3.1416;
     float sumRe = 0, sumIm = 0; // init real/imag components
     float cosine = 0, sine = 0; // Initialise cosine/sine components

          // Output Real and Imaginary components

     float out_real[8] = {0.0}, out_imag[8] = {0.0};
     int n = 0, k = 0;

     for(k=0 ; k<N ; k++)
     {
          sumRe = 0;
          sumIm = 0;
          for (n=0; n<N ; n++)
          {
               cosine = cos(2*pi*k*n/N);
               sine   = sin(2*pi*k*n/N);
               sumRe = sumRe + x[n] * cosine;
               sumIm = sumIm - x[n] * sine;
          }

          out_real[k] = sumRe;
          out_imag[k] = sumIm;
          printf("[%d] %7.3f %7.3f \n", k, out_real[k], out_imag[k]);
     }

}
```

Output

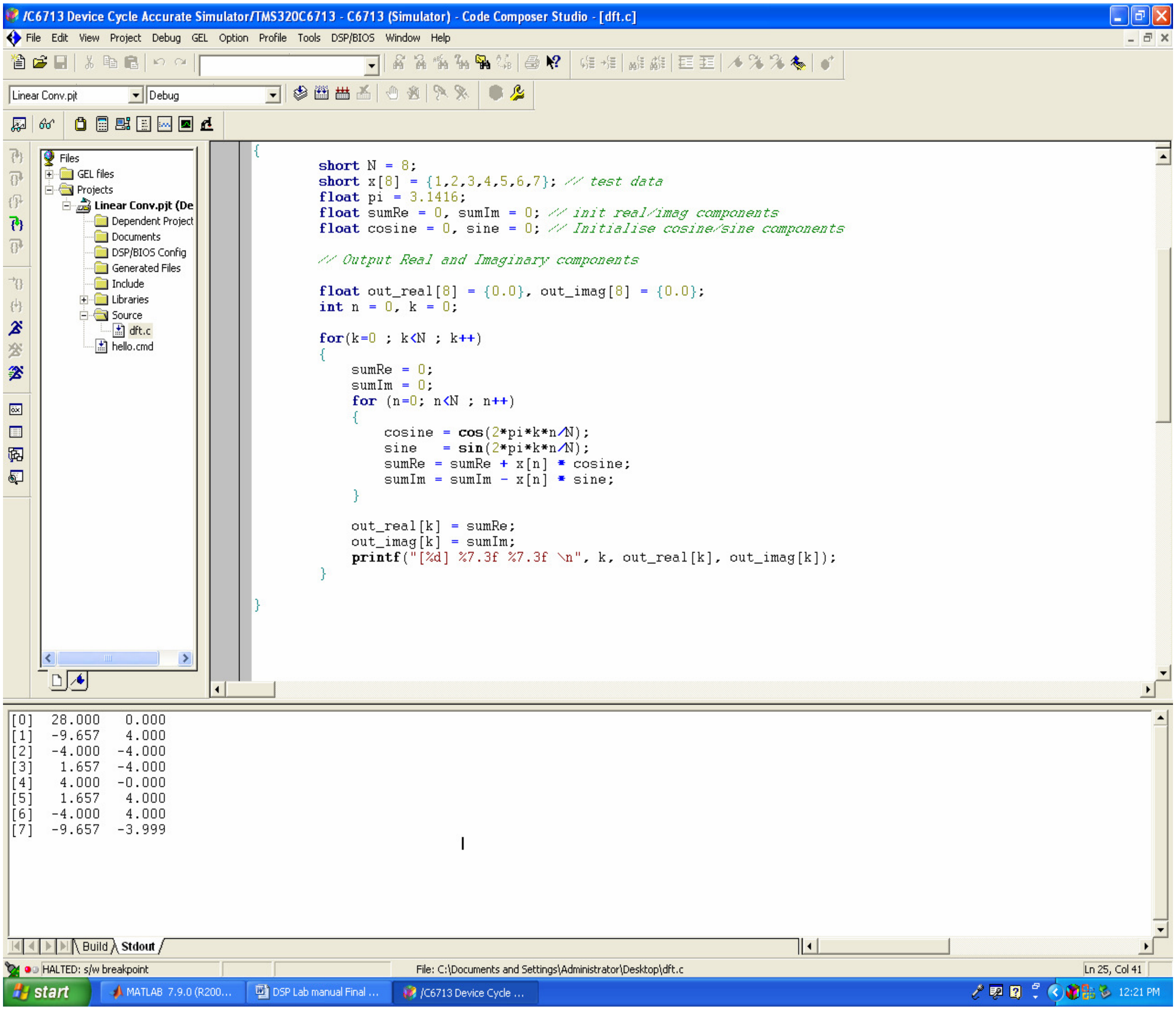
```
[0]  28.000   0.000
[1]  -9.657   4.000
[2]  -4.000  -4.000
[3]   1.657  -4.000
[4]   4.000  -0.000
[5]   1.657   4.000
[6]  -4.000   4.000
[7]  -9.657  -3.999
```

Verification in matlab

```
x = [1,2,3,4,5,6,7,0]
fft(x)
```

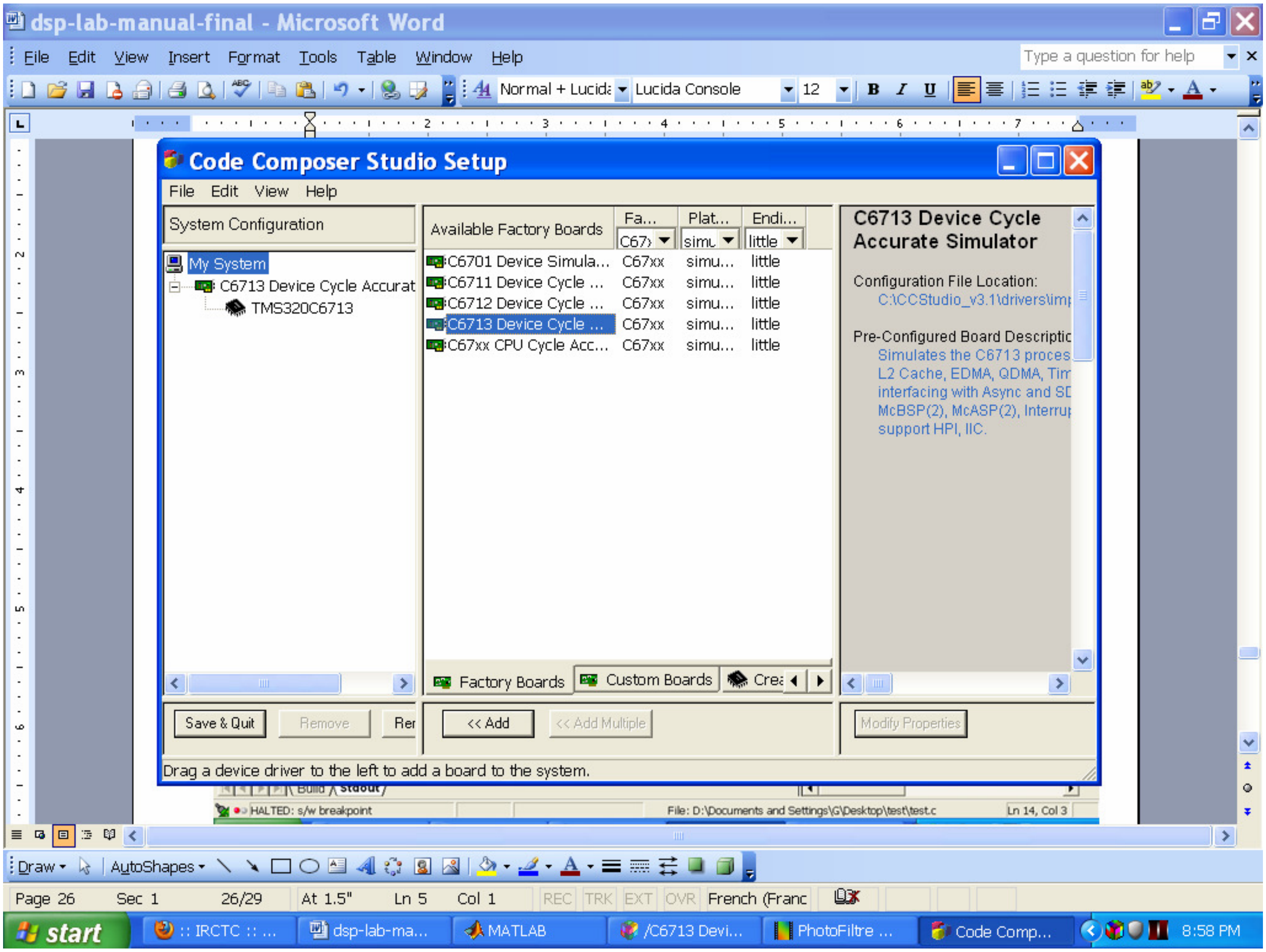
Output

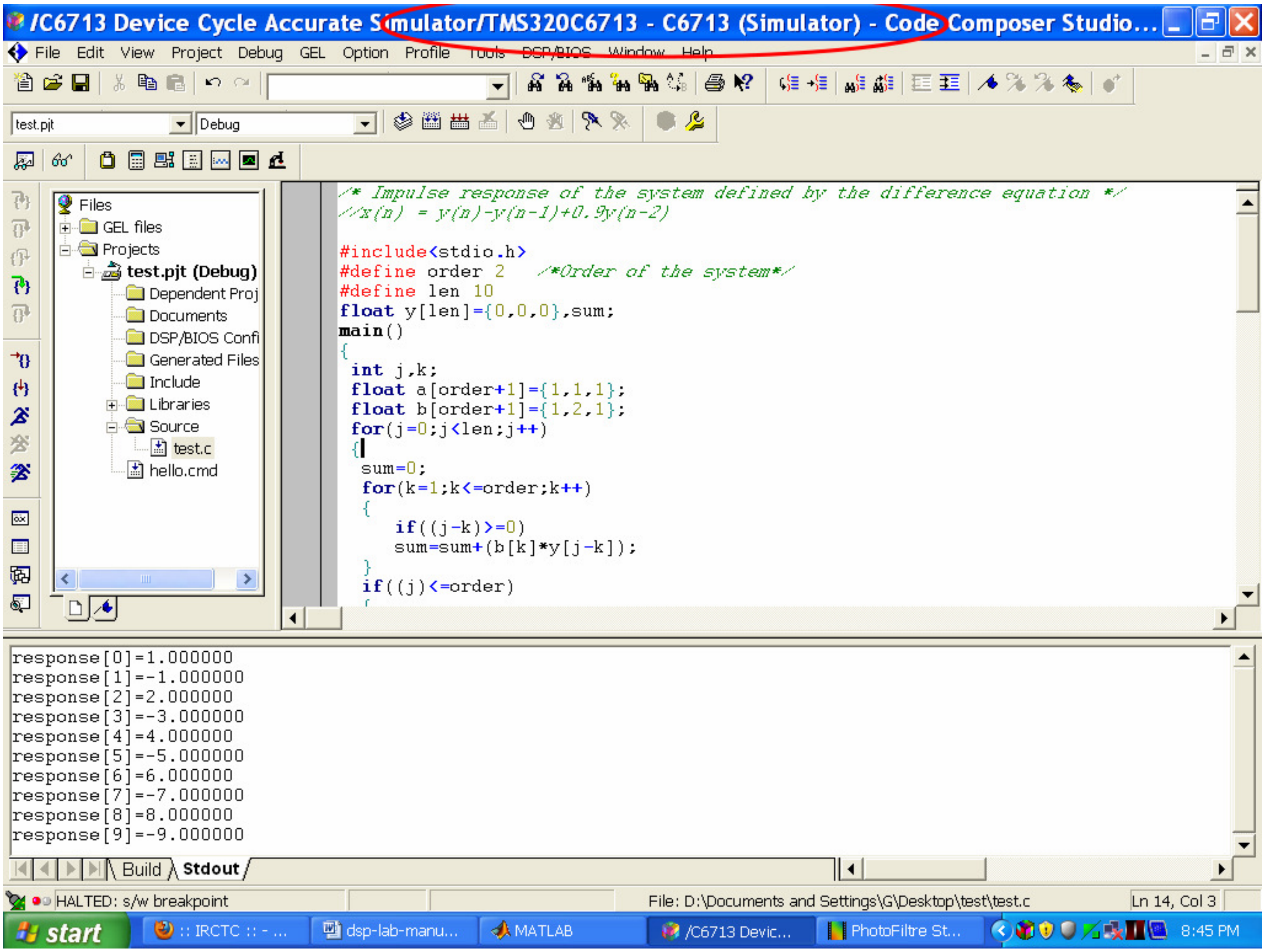
```
  Columns 1 through 4

  28.0000    -3.5000 + 7.2678i  -3.5000 + 2.7912i  -3.5000 + 0.7989i

  Columns 5 through 7

  -3.5000 - 0.7989i  -3.5000 - 2.7912i  -3.5000 - 7.2678i
```


/C6713 Device Cycle Accurate Simulator/TMS320C6713 - C6713 (Simulator) - Code Composer Studio - [dft.c]
File Edit View Project Debug GEL Option Profile Tools DSP/BIOS Window Help
Linear Conv.pjt
Debug
Files
GEL files
Projects
Linear Conv.pjt (De
Dependent Project
Documents
DSP/BIOS Config
Generated Files
Include
Libraries
Source
dft.c
hello.cmd
{
short N = 8;
short x[8] = {1,2,3,4,5,6,7}; // test data
float pi = 3.1416;
float sumRe = 0, sumIm = 0; // init real/imag components
float cosine = 0, sine = 0; // Initialise cosine/sine components
// Output Real and Imaginary components
float out_real[8] = {0.0}, out_imag[8] = {0.0};
int n = 0, k = 0;
for(k=0 ; k<N ; k++)
{
sumRe = 0;
sumIm = 0;
for (n=0; n<N ; n++)
{
cosine = cos(2*pi*k*n/N);
sine = sin(2*pi*k*n/N);
sumRe = sumRe + x[n] * cosine;
sumIm = sumIm - x[n] * sine;
}
out_real[k] = sumRe;
out_imag[k] = sumIm;
printf("[%d] %7.3f %7.3f \n", k, out_real[k], out_imag[k]);
}
}
[0] 28.000 0.000
[1] -9.657 4.000
[2] -4.000 -4.000
[3] 1.657 -4.000
[4] 4.000 -0.000
[5] 1.657 4.000
[6] -4.000 4.000
[7] -9.657 -3.999
Build
Stdout
HALTED: s/w breakpoint
File: C:\Documents and Settings\Administrator\Desktop\dft.c
Ln 25, Col 41
start
MATLAB 7.9.0 (R200...
DSP Lab manual Final ...
/C6713 Device Cycle ...
12:21 PM

4. Impulse response of first order and second order system

```
/* Impulse response of the system defined by the difference
equation */
//x(n) = y(n)-y(n-1)+0.9y(n-2)
```

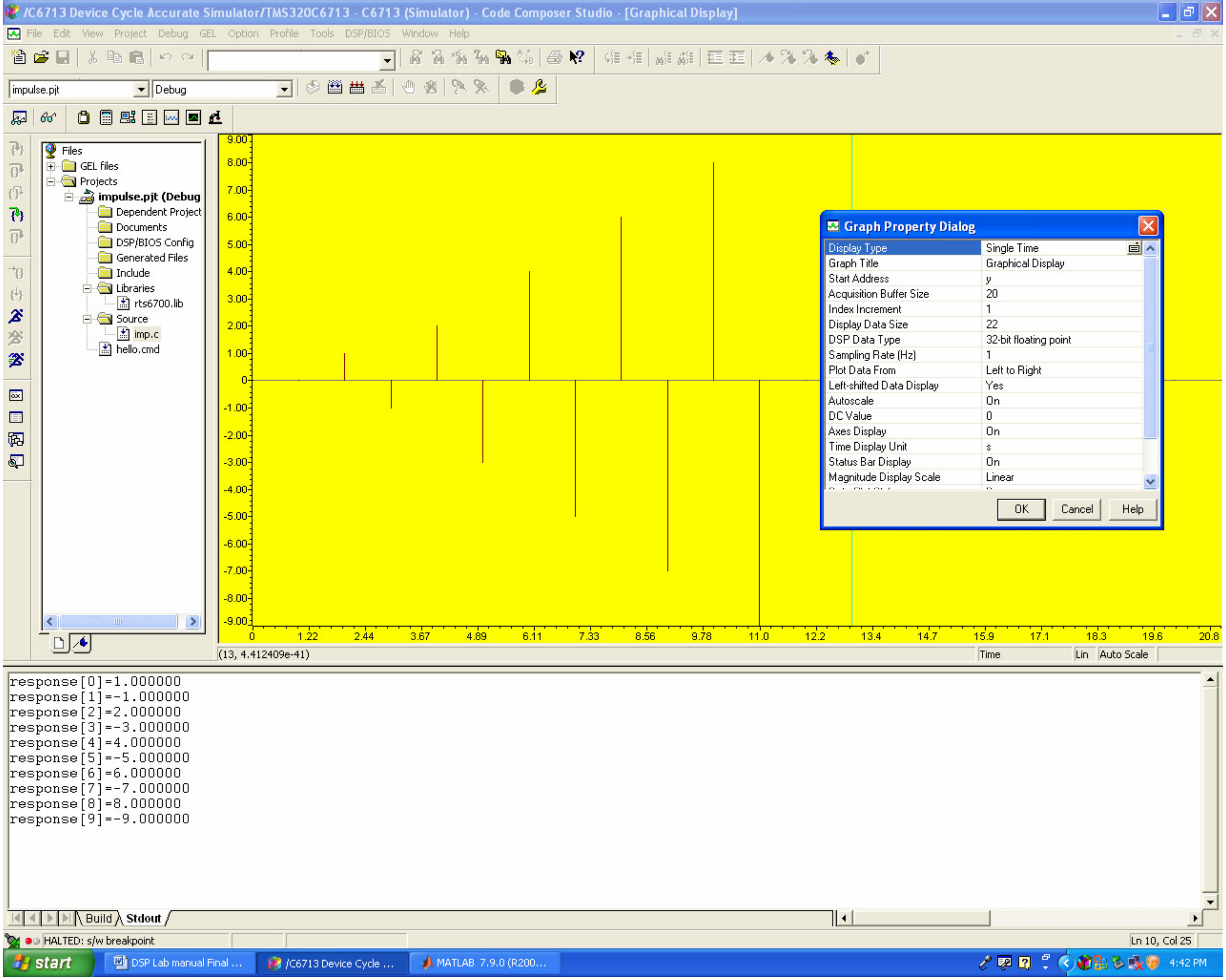




Please select DEVICE CYCLE ACCURATE SIMULATOR

```
/*Impulse response of the system
 y[n] + a1 y[n-1] + a2 y[n-2] + .. = b0 x[n] + b1 x[n-1]
 + b2 y[n-2] + ..
 Example :
 1 y[n] + 1 y[n-1] + 1 y[n-2] = 1 x[n] + 2 x[n-1] + 1 y[n-2]
 */
 #include<stdio.h>
 #define order 2    /*Order of the system*/
 #define len 10     /*Length of the output pulses*/
 float y[len]={0,0,0},sum;
 main()
 {
  int j,k;
  float a[order+1]={1,1,1};
 /* y coefficients - may change in accordance with the difference
 equation */

  float b[order+1]={1,2,1};
 /* x coefficients - may change in accordance with the difference
 equation */

  for(j=0;j<len;j++)
  {
        sum=0;
        for(k=1;k<=order;k++)
        {
           if((j-k)>=0)
           sum=sum+(b[k]*y[j-k]);
        }
        if((j)<=order)
        {
           y[j]=a[j]-sum;
        }
        else
        {
           y[j]=-sum;
        }
        printf("response[%d]=%f\n",j,y[j]);
```

```
 }
}
```



OUTPUT:

```
response[0]=1.000000
response[1]=-1.000000
response[2]=2.000000
response[3]=-3.000000
response[4]=4.000000
response[5]=-5.000000
response[6]=6.000000
response[7]=-7.000000
response[8]=8.000000
response[9]=-9.000000
```

Matlab verification of impulse response of LTI system

```
xcoeff = [1 1 1]
ycoeff = [1 2 1]
imp_resp = filter(xcoeff,ycoeff,[1 zeros(1,9)])


imp_resp =

1    -1     2    -3     4    -5     6    -7     8    -9
```

REFERENCE BOOKS:

1. **Digital signal processing using MATLAB -** Sanjeet Mitra, TMH, 2001
2. **Digital signal processing using MATLAB** - J. G. Proakis & Ingale, MGH, 2000
3. **Digital Signal Processors**, B. Venkataramani and Bhaskar, TMH,2002