**1.Random wave**

```
#include<stdio.h>
#include<math.h>
#define PI 3.14
#define PTS 128
 float X[PTS];
float YIPTS];
float z[PTS];
float nIPTS]:
void main (){
int 1,j;
for (i = 0; 1 < PTS : 1++)
x[i] = sin (2*PI*i*20/128.0);
y[i]=0.0;
n[il=x[i] + rand () * 10 ;
}
```

**2.Diffrential eq**

```
#include <stdio.h>
#include<math.h>
#define FREQ 400
float y[3]={0,0,0};
float x[3]={0,0,0};
float z[128],m[128],n[128],p[128];
main()
{
int i=0,j;
float a[3]={ 0.072231,0.144462,0.072231};
float b[3]={ 1.000000,-1.109229,0.398152};
for(i=0;i<128;i++)
{
m[i]=sin(2*3.14*FREQ*i/24000);
}
for(j=0;j<128;j++)
{
x[0]=m[j];
y[0] = (a[0] x[0]) +(a[1] x[1] ) +(x[2]*a[2]) - (y[1]*b[1])- (y[2]*b[2]);
z[j]=y[0];
 y[2]=y[1];
y[1]=y[0];
 x[2]=x[1];
 x[1] = x[0};
 }
}
```

3.**Power**

```
#include <stdio.h>
int mains()
{
int num, i, j, x [32];
float num1:
long int sum=0:
print (“\nEnter the number of samples”)
 scanf ("%d", &num) ;
printf "InEnter samples:
for (j=0; j<num;j++)
scanf ("%d",  &x[j]) ;
for (1=0;1<=num; 1++)
{
sum+=x [i]*x [i];
}
num=num* 2
num1 = sum / (float) num;
printf ("\n the Average power of above sampleg ig\n t.2f", num1);
return 0;
}
```

4. **Energy**

```
‡include<stdio.h>
int main ()
{
int num, i, j, x [32]:
long int sum=0:
printf(enter the number or samples:)
scan ("%d", &num)
 printf ("\nEnter samples)
for(j=0;<num; j++)
scanf(“%d”, &x[j])
for(i=0;<num; i++)
{
sum+= tx[i]*x[i];
print ("in the energy of above samples is\n %d",sum)
```

4. **Circularconv**

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

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

**5. DFT**

```
#include<stdio.h>
include<math.h>
int N,k,n,i;
float pi=3.1416,sumre=0, sumim=0,out_real[8]={0.0},
 out_imag[8]={0.0};
 int x[32];
void main(void) {
printf(" enter the length of the sequence\n");
 scanf("%d",&N);
printf(" enter the sequence\n");
 for(i=0;i<N;i++)
scanf("%d",&x[i]);
 for(k=0;k<N;k++)
{
sumre=0;
sumim=0;
for(n=0;n<N;n++)
{
Sumre = sumre+x[n]* cos(2*pi*k*n/N);
Sumim = sumim-x[n]* sin(2*pi*k*n/N); }
out_real[k]=sumre;
out_imag[k]=sumim;
printf("X([%d])=\t%f\t+\t%fi\n",k,out_real[k],out_im ag[k]);
} }
```

### 6.sinwave

```
#include "L138_LCDK_aic3106_init.h"
 #include "math.h"
#define SAMPLING_FREQ 8000
#define PI 3.14159265358979
float frequency = 1000.0;
 float amplitude = 20000.0;
 float theta_increment;
float theta = 0.0;
interrupt void interrupt4(void) // interrupt service routine {
theta_increment = 2*PI*frequency/SAMPLING_FREQ;
 theta += theta_increment;
if (theta > 2*PI) theta -= 2*PI;
output_left_sample((int16_t)(amplitude*sin(theta))); return;
}
int main(void) {
L138_initialise_intr(FS_48000_HZ,ADC_GAIN_0DB,DAC_ATTEN_0DB,LCDK_LINE_INPUT);
while(1); }
```

### 7. Square wave

```
#include "L138_LCDK_aic3106_init.h"
#define LOOPLENGTH 64
int16_t square_table[LOOPLENGTH] =
{10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000,
10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000,
10000, 10000, 10000, 10000, -10000,-10000,-10000,-10000,-10000,-10000,-10000,-10000,
-10000,-10000,-10000,-10000,-10000,-10000,-10000,-10000,
-10000,-10000,-10000,-10000,-10000,-10000,-10000,-10000,
-10000,-10000,-10000,-10000,-10000,-10000,-10000,-10000};

 int16_t loopindex = 0;
interrupt void interrupt4(void) // interrupt service routine {
output_left_sample(square_table[loopindex++]);
 if (loopindex >= LOOPLENGTH)
loopindex = 0; return;
}
int main(void) {
L138_initialise_intr(FS_8000_HZ,ADC_GAIN_0DB,DAC_ATTEN_0DB,LCDK_LINE_INPUT);
while(1); }
```

## 8. Ramp

```
#include "L138_LCDK_aic3106_init.h"
#define LOOPLENGTH 64
int16_t output = 0;
interrupt void interrupt4(void) // interrupt service routine {
output_left_sample(output); output += 2000;
if (output >= 30000)
output = -30000;
return; }
int main(void) {
L138_initialise_intr(FS_8000_HZ,ADC_GAIN_0DB,DAC_ATTEN_0DB,LCDK_LINE_INPUT); while(1);
}
```

## 9. Flanging effect

```
#include "L138_LCDK_aic3106_init.h"
#define BUF_SIZE 24000
uint16_t input,output,delayed;
uint16_t buffer[BUF_SIZE];
int i = 0;
interrupt void interrupt4(void) // interrupt service routine {
input = input_left_sample();
delayed = buffer[i];
output = delayed + input;
buffer[i] = input;
i = (i+1)%BUF_SIZE;
 output_left_sample(output);
return; }
   int main(void) {
int i;
for (i=0 ; i<BUF_SIZE ; i++) {
buffer[i] = 0; }
L138_initialise_intr(FS_48000_HZ,ADC_GAIN_0DB,DAC_ATTEN_0DB,LCDK_MIC_INPUT);
while(1); }
```

## 10. Headphone

```
#include "L138_LCDK_aic3106_init.h" #define GAIN 0.6
#define BUF_SIZE 16000
int16_t input,output,delayed;
int16_t buffer[BUF_SIZE];
int i = 0;
interrupt void interrupt4(void) // interrupt service routine {
input = input_left_sample();
 delayed = buffer[i];
output = delayed + input;
buffer[i] = input + delayed*GAIN; // update the result and store it in buffer i = (i+1)%BUF_SIZE;
output_left_sample(output);
return;
 }
int main(void) {
int i;
for (i=0 ; i<BUF_SIZE ; i++) {
buffer[i] = 0; // buffer filled with zeros }
L138_initialise_intr(FS_48000_HZ,ADC_GAIN_0DB,DAC_ATTEN_0DB,LCDK_MIC_INPUT);
while(1); }
```

## 11. Auto corelation

```
#include<stdio.h>
 #define PTS 4
int x[PTS],y[PTS],n,k;
void main()
{
int atoc[PTS],i,sum;
printf("enter the first sequence\n");
for(i=0;i<PTS;i++)
scanf("%d\n",&x[i]);
 printf("enter the second sequence\n");
 for(i=0; i<PTS; i++)
scanf("%d\n", &y[i]);
for (n=0;n<PTS;n++)
{
sum=0;
for (k=0;k<PTS-n;k++)
{
sum=sum+ (x[k] * x[k+n]); }
atoc[n]=sum;
}
for(i=0;i<PTS;i++) printf("%d\n", atoc[i]);
}
```