C-programming code  
///////////////////////////////////////  
#include "tonecfg.h"  
#include <dsk6713.h>  
#include "dsk6713\_aic23.h"  
  
#include <csl.h>  
#include "cpee\_student\_diffeq.h"  
#include "dsk6713\_dip.h"  
#include "dsk6713\_led.h"  
#include <math.h>  
#define maxdouble 1.7976931348623158e+308  
#define Uint16 unsigned short  
/\* Codec configuration settings \*/  
DSK6713\_AIC23\_Config config = {  
0x0017, // 0 DSK6713\_AIC23\_LEFTINVOL Left line input channel volume  
0x0017, // 1 DSK6713\_AIC23\_RIGHTINVOL Right line input channel volume  
0x01F9, // 2 DSK6713\_AIC23\_LEFTHPVOL Left channel headphone volume  
0x01F9, // 3 DSK6713\_AIC23\_RIGHTHPVOL Right channel headphone volume  
0x0011, // 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  
0x0001, // 8 DSK6713\_AIC23\_SAMPLERATE Sample rate control  
0x0001 // 9 DSK6713\_AIC23\_DIGACT Digital interface activation  
};  
  
long double output=0;  
long double input;  
long double delaylength = 0, delaylength2 = 0, delaylength3 = 0, delaylength4 = 0;

long double delay[10000], delay2[10000], delay3[10000], delay4[10000];

//random number delays for chorus effect, 250 ~10ms  
double chorusdelay[100]={  
685, 797, 781, 450, 650, 752, 630, 691, 546, 686,  
431, 555, 507, 568, 550, 495, 799, 771, 455, 771,  
594, 495, 655, 467, 425, 770, 565, 562, 578, 722,  
484, 637, 745, 728, 621, 657, 580, 589, 577, 489,  
481, 459, 664, 527, 413, 684, 799, 426, 608, 704,  
671, 436, 428, 477, 498, 445, 787, 527, 423, 787,  
417, 671, 630, 732, 459, 636, 428, 731, 411, 445,  
743, 499, 547, 676, 709, 644, 440, 504, 448, 792,  
629, 562, 445, 751, 795, 616, 650, 741, 711, 742,  
511, 589, 439, 784, 709, 617, 592, 760, 546, 773  
};

//center frequencies of BPF for auto-wah  
double wahfreqsweep[50]={  
500, 520, 540, 560, 580, 600, 620, 640, 660, 680,  
700, 720, 740, 760, 780, 800, 820, 840, 860, 880,  
900, 920, 940, 960, 980, 1000, 1020, 1040, 1060, 1080,  
2000, 2020, 2040, 2060, 2080, 3000, 3020, 3040, 3060, 3080, 4000, 4020, 4040, 4060, 4080,  
5000  
};

//input samples and output samples for left and right channels  
Uint32 left\_input,right\_input,left\_output,right\_output;  
Int16 distortint, predistort;  
int i = 0, ii=0, iii=0, iv=0, d=0, d2=0, freqcount = 0, freqcount2 = 1250, k = 13;

double yn1=0,yn2=0,xn1=0,xn2=0;  
int x, y, on=1, c=128; // on/off variable and initial threshold c  
Int16 echo(Int16 channel,Uint32 delaylength);  
Int16 flanger(Int16 channel);  
Int16 chorus(Int16 channel);  
Int16 reverb(Int16 channel,Uint32 delaylength);  
Int16 wah\_wah(Int16 input);  
intfreqsweep();  
int sweep();  
intdistortionthresh(int);  
Int16 distortion(Int16 channel);

Int16 echo(Int16 channel,Uint32 delaylength){output = 0;input = channel;output = input + .3\*delay[i] + .5\*delay2[ii] + .7\*delay3[iii]+ .9\*delay4[iv];delay[i] = input;delay2[ii] = input;delay3[iii] = input;delay4[iv] = input;i++;ii++;iii++;iv++;if (i>=delaylength) i = 0;if (ii>=delaylength\*.9) ii=0;if (iii>=delaylength\*.7) iii=0;if (iv>=delaylength\*.5) iv=0;return (Int16) output;}

Int16 chorus(Int16 channel){input = channel;d2++;if (d2>=1250){d++;d2=0;}if(d>=100) d=0;output = input + (Int16) (.5\*delay[i]);delay[i] = input;i++;if (i>=chorusdelay[d]) i = 0;//minimize distortion from addition of delayed signalsif (output>maxdouble) output = .7\*maxdouble;if (output<-maxdouble) output = -.7\*maxdouble;return (Int16) output;}

Int distortionthresh(int x){float y, xc = x/c;y = x \* (1 - b \* xc \* xc);if (x>c) y = a\*c; // force the threshold valuesif (x<-c) y = -a\*c;return ((int) y);}Int16 distortion(Int16 channel){output = (Int16) distortionthresh((int) channel);return ((Int16) (4\*output));}

Int16 wah\_wah(Int16 input)  
{  
Int16 freq;  
double y;  
freq = freqsweep();  
y = .7\*input - 1.164\*cos(2\*PI\*(freq/16000))\*xn1 + .567\*xn2 +  
1.9\*cos(2\*PI\*(freq/16000))\*yn1 - .9025\*yn2;  
xn2 = xn1;  
xn1 = input;  
yn2 = yn1;  
yn1 = y;

return (Int16) y;  
}

//sweeps center frequency of band pass filter for auto-wah  
intfreqsweep(){  
Int16 freq;  
if(!--freqcount2){  
if (!control){  
freq = wahfreqsweep[freqcount++];  
if (freqcount>50) control = 1;  
}  
else if (control){  
freq = wahfreqsweep[freqcount--];  
if (freqcount==0) control = 0;  
}  
freqcount2 = 1250;  
}  
return freq;  
}

main()  
{DSK6713\_AIC23\_CodecHandle hCodec;  
int check;  
CSL\_init();  
DSK6713\_init();  
DSK6713\_DIP\_init();  
DSK6713\_LED\_init();

hCodec = DSK6713\_AIC23\_openCodec(0, &config);

DSK6713\_AIC23\_setFreq(hCodec,DSK6713\_AIC23\_FREQ\_16KHZ);  
student\_diffeq\_initialize(BPfreq);  
IRQ\_globalDisable();  
while(1)  
{  
while (!DSK6713\_AIC23\_read(hCodec, &left\_input));  
while (!DSK6713\_AIC23\_read(hCodec, &right\_input));  
while (!DSK6713\_AIC23\_write(hCodec, left\_output));  
while (!DSK6713\_AIC23\_write(hCodec, right\_output));  
//moves selected band from left to right}  
if (state==0){//echo effect state  
if (DSK6713\_DIP\_get(0)==0){  
predistort = echo(left\_input, 2500);  
distortint = distortion(predistort);  
left\_output = distortint;}  
else left\_output = echo(left\_input, 2500);  
DSK6713\_LED\_on(0);  
DSK6713\_LED\_off(1);  
DSK6713\_LED\_off(2);  
DSK6713\_LED\_off(3);}

else if (state==1){//chorus effect state  
  
if (DSK6713\_DIP\_get(0)==0){  
predistort = chorus(left\_input);  
distortint = distortion(predistort);  
left\_output = distortint;}  
else left\_output = chorus(left\_input);  
DSK6713\_LED\_off(0);  
DSK6713\_LED\_on(1);  
DSK6713\_LED\_off(2);

DSK6713\_LED\_off(3);}  
  
else if (state==4){  
  
if (DSK6713\_DIP\_get(0)==0){  
predistort = wah\_wah(left\_input);  
distortint = distortion(predistort);  
left\_output = distortint;}  
else left\_output = wah\_wah(left\_input);  
DSK6713\_LED\_off(0);  
DSK6713\_LED\_off(1);  
DSK6713\_LED\_off(2);  
DSK6713\_LED\_on(3);}  
else if (state ==5) state =0;  
//since guitar single is mono both output signals are the same in case of stereo out  
right\_output=left\_output;  
};  
}