National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Electrical Engineering

Faculty Member: Dr. Wajahat Hussain **Dated:** 7-3-2107

Course/Section: BEE6-B **Semester:** 6th Semester

EE-330 Digital Signal Processing

Lab4: Audio Processing using DSP Kit TMS 320C6713 DSK

Name	Reg. no.	Report Marks / 10	Lab Quiz-Viva Marks / 5	Total / 15
Abdullah Bin Asif	111596			
Saad Iqbal	111394			
Usman Iqbal	111393			

National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Objectives

The objective of this lab is to explore some more features of Code Composer Studio (CCS) that is time domain and frequency domain plots of audio signal in addition to that we will also do real time processing of audio input.

- Time domain and frequency Domain Plots in CCS
- Real Time processing of Audio Signal
- Working with basic sinusoids on DSP Kit

1. LAB TASK NO.1

Modify program sine8_buf.c to generate a sine wave with a frequency of 3000 Hz. Verify your result using an oscilloscope connected to the LINE OUT socket on the DSK as well as using Code Composer to plot the 32 most recently output samples in both the time and frequency domains.

CCS Waveform:

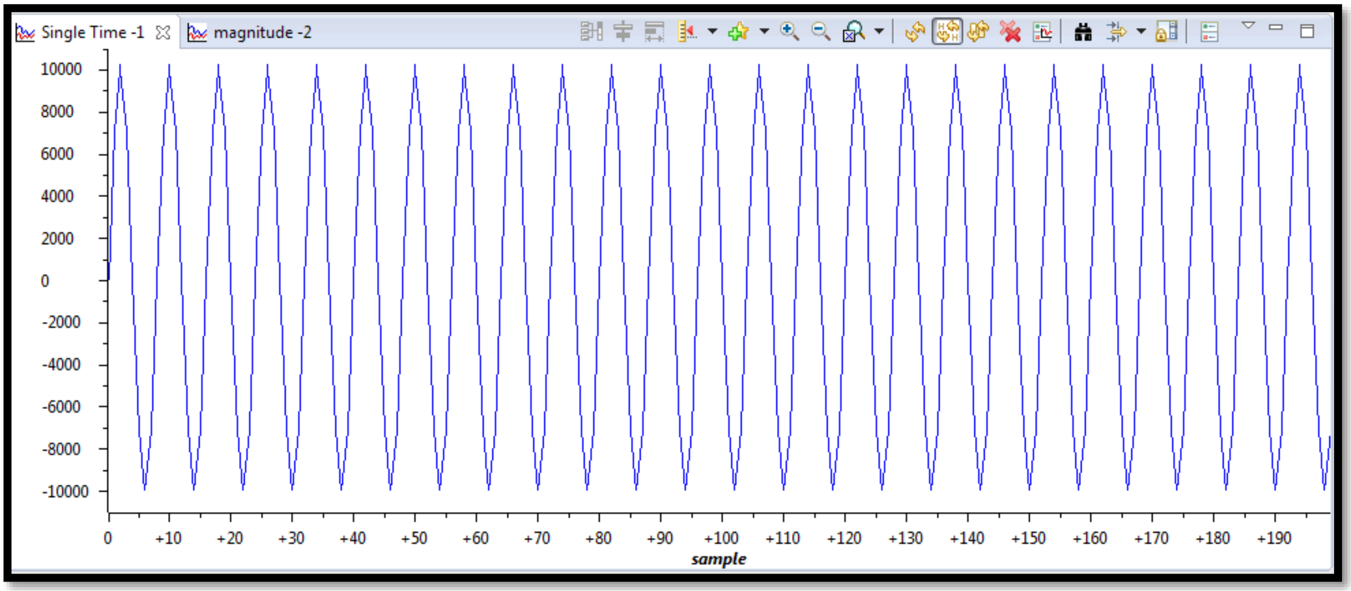

Figure No. 1

CCS FFT Plot:

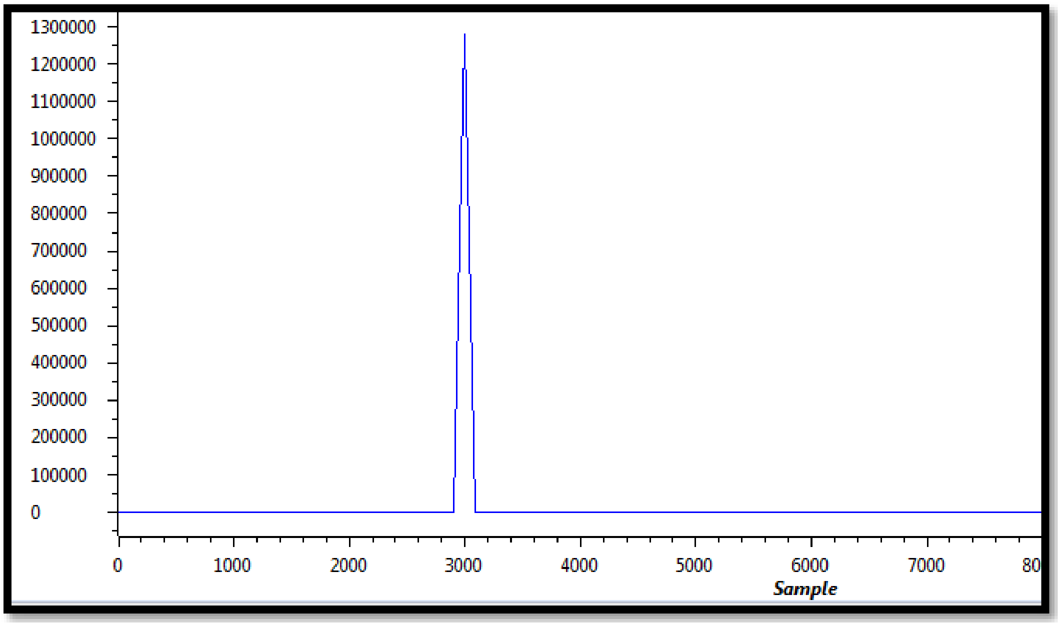

Figure No. 2

2. LAB TASK NO.2

Write polling - based program such that when DIP switch #3 is pressed down, LED #3 turns on and a 500 - Hz cosine wave is generated for 5 seconds.

CCS Code:

```
//sine8_LED.c  sine generation with DIP switch control

#include "dsk6713_aic23.h"                    //codec support
Uint32 fs = DSK6713_AIC23_FREQ_8KHZ;   //set sampling rate
#define DSK6713_AIC23_INPUT_MIC 0x0015
#define DSK6713_AIC23_INPUT_LINE 0x0011
Uint16 inputsource=DSK6713_AIC23_INPUT_MIC;//select input
#define LOOPLENGTH 16
short loopindex = 0;                    //table index
short gain = 10;                                        //gain factor
short sine_table[LOOPLENGTH]={1000,914,669,309,-104,-500,-809,-978,-978,-809,-500,-
104,309,669,914,1000}; //sine values
int i;
void main()
{
 comm_poll();                           //init DSK,codec,McBSP
 DSK6713_LED_init();                                 //init LED from BSL
 DSK6713_DIP_init();                                 //init DIP from BSL
 while(1)                                          //infinite loop
 {
  if(DSK6713_DIP_get(3)==0)                     //=0 if DIP switch #0 pressed
  {
   DSK6713_LED_on(3);                                //turn LED #0 ON
   for (i=1;i<40000;i++){
        output_left_sample(sine_table[loopindex++]*gain); //output sample
        if (loopindex >= LOOPLENGTH) loopindex = 0; //reset table index
   }
  }
  else DSK6713_LED_off(3);                       //turn LED off if not pressed
 }                         //end of while(1) infinite loop
}                           //end of main
```

3. LAB TASK NO.3

Write an interrupt - driven program that maintains a buffer containing the 128 most recent input samples read at a sampling frequency of 16 kHz from the AIC23 codec, using the MIC IN socket on the DSK. Halt the program and plot the buffer contents using Code Composer.

CCS Code:

```
//sine8_buf.c sine generation with output stored in buffer
#include "DSK6713_AIC23.h"                    //codec support
Uint32 fs=DSK6713_AIC23_FREQ_16KHZ;       //set sampling rate
#define DSK6713_AIC23_INPUT_MIC 0x0015
#define DSK6713_AIC23_INPUT_LINE 0x0011
Uint16 inputsource=DSK6713_AIC23_INPUT_MIC; // select input
#define LOOPLENGTH 8
#define BUFFERLENGTH 128
int loopindex = 0;                                          //table index
int bufindex = 0;                  //buffer index
int out_buffer[BUFFERLENGTH];          //output buffer
short gain = 10;

interrupt void c_int11()               //interrupt service routine
{
  short out_sample;
 // possible sampling rates: 8, 16, 24, 32, 44, 48, 96 kHz
  out_sample =input_sample();
  output_sample(out_sample);  //output sample value
  out_buffer[bufindex++] = out_sample;    //store in buffer
  if (bufindex >= BUFFERLENGTH) bufindex = 0; //check for end of buffer
  return;                                               //return from interrupt
}

void main()
{
  comm_intr();           //initialise DSK
  while(1);    //infinite loop
}
```

CCS Waveform:

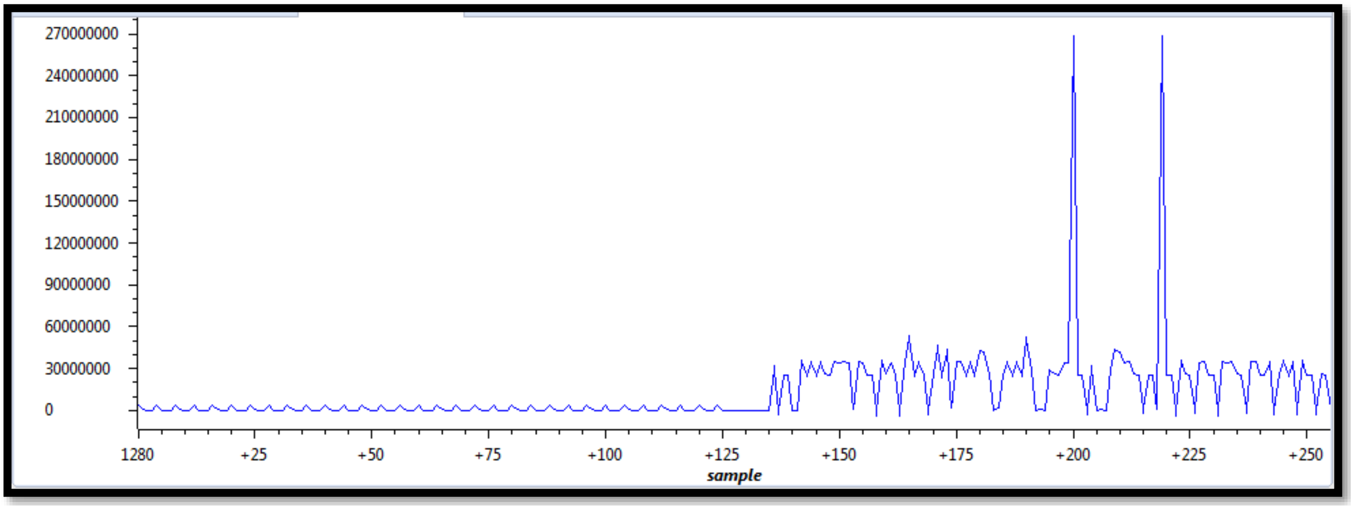

Figure No. 3

Conclusion:

In this lab we studied the fundamental concepts of signal processing using the DSK kit. We learned to generate a tone signal with a specific frequency, also we observed the FFT of a signal. Also the effects on the waveform were seen when the sampling rate was changed. In the last part we took an input from the user and displayed the waveform on the interface.
