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

Department of Electrical Engineering and Computer Science

Faculty Member: Dr. Ahmad Salman Dated: 14/05/2023

Semester: 6th Section: BEE 12C

EE-330 Digital Signal Processing

Lab 11: Introduction to DSP Kit TMS 320C6713 DSK

Group Members

		PLO4 - CLO4		PLO5 - CLO5	PLO8 - CLO6	PLO9 - CLO7
Name	**Reg. No**	**Viva / Quiz / Lab Performance**	**Analysis of data in Lab Report**	**Modern Tool Usage**	**Ethics and Safety**	**Individual and Teamwork**
		5 Marks	**5 Marks**	**5 Marks**	**5 Marks**	**5 Marks**
Danial Ahmad	331388					
Muhammad Umer	345834					
Tariq Umar	334943					

1 Table of Contents

2 Introduction to DSP Kit TMS 320C6713 DSK

2.1 Objectives

The objective of this lab is to introduce the DSP Starter Kit C6713

- Getting started with DSP Kit
- Getting Started with Code Composer Studio
- Basic Code Compilation on DSP Kit
- Working with basic sinusoids on DSP Kit

2.2 Software

Code Composer Studio (CCS) provides an integrated development environment (IDE) for real - time digital signal processing applications based on the C programming language. It incorporates a C compiler, an assembler, and a linker. It has graphical capabilities and supports real - time debugging. The C compiler compiles a C source program with extension `.c` to produce an assembly source file with extension *.asm*. The assembler assembles an *.asm* source file to produce a machine language object file with extension *.obj*. The linker combines object files and object libraries as input to produce an executable file with extension *.out.* This executable file represents a linked common object file format (COFF), popular in Unix - based systems and adopted by several makers of digital signal processors. This executable file can be loaded and run directly on the digital signal processor.

2.3 Lab Report Instructions

All questions should be answered precisely to get maximum credit. Lab report must ensure following items:

- Lab objectives
- MATLAB codes
- Results (graphs/tables) duly commented and discussed
- Conclusion

3 Lab Procedure

3.1 Introduction

The hardware experiments in the DSP lab are carried out on the Texas Instruments TMS320C6713 DSP Starter Kit (DSK), based on the TMS320C6713 floating point DSP running at 225 MHz The basic clock cycle instruction time is 1/(225 MHz)= 4.44 nanoseconds. During each clock cycle, up to eight instructions can be carried out in parallel, achieving up to 8×225 = 1800 million instructions per second (MIPS). The C6713 processor has 256KB of internal memory and can potentially address 4GB of external memory. The DSK board includes a 16MB SDRAM memory and a 512KB Flash ROM. It has an on-board 16-bit audio stereo codec (the Texas Instruments AIC23B) that serves both as an A/D and a D/A converter. There are four 3.5 mm audio jacks for microphone and stereo line input, and speaker and head-phone outputs. The AIC23 codec can be programmed to sample audio inputs at the following sampling rates:

fs = 8, 16, 24, 32, 44.1, 48, 96 kHz

The DSK also has four user-programmable DIP switches and four LEDs that can be used to control and monitor programs running on the DSP.All features of the DSK are managed by the CCS, which is a complete integrated development environment (IDE) that includes an optimizing C/C++ compiler, assembler, linker, debugger, and program loader. The CCS communicates with the DSK via a USB connection to a PC. In addition to facilitating all programming aspects of the C6713 DSP, the CCS can also read signals stored on the DSP's memory, or the SDRAM, and plot them in the time or frequency domains.

3.2 TMS 320 C 6713 Digital Signal Processor

The TMS320C6713 (C6713) is based on the very long instruction word (VLIW) architecture, which is very well suited for numerically intensive algorithms. The internal program memory is structured so that a total of eight instructions can be fetched every cycle. For example, with a clock rate of 225 MHz, the C6713 is capable of fetching eight 32 - bit instructions every 1/(225 MHz) or 4.44 ns. Features of the C6713 include 264 kB of internal memory (8 kB as L1P and L1D Cache and 256 kB as L2 memory shared between program and data space), eight functional or execution units composed of six ALUs and two multiplier units, a 32 - bit address bus to address 4 GB (gigabytes), and two sets of 32 - bit general – purpose registers. The C67xx processors (such as the C6701, C6711, and C6713) belong to the family of the C6x floating - point processors; whereas the C62xx and C64xx belong to the family of the C6x fixed - point processors. The C6713 is capable of both fixed- and floating-point processing.

3.3 Quick Tests

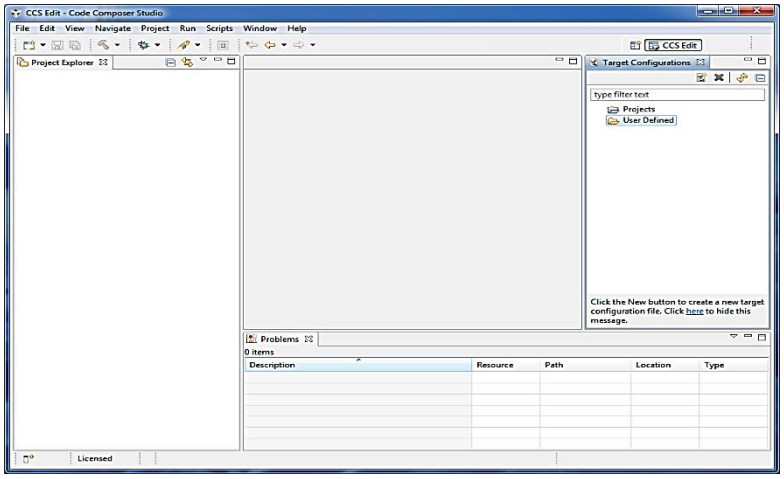
Connect the USB cable from the DSK6713 to the computer and then connect power to the DSK6713. If this is the first time the DSK has been connected to the computer, you may see Windows install the device driver software: You can also check that the DSK is recognized by going into the Device Manager and looking for "SD USB Based Debug Tools". You should see a "Spectrum Digital TMS320C6713 DSK" listed as a correctly installed device. If not, then CCS v6 has probably not been installed correctly.

Now start CCS v6 by double-clicking on the desktop icon (or navigating through the start menu). You may be prompted to select a workspace. If so, choose a directory that you would like to use as your workspace and click "OK". If the TI Resource Explorer window comes up, just close it. Eventually, you should get to the main CCS v6 window, which should look something like this

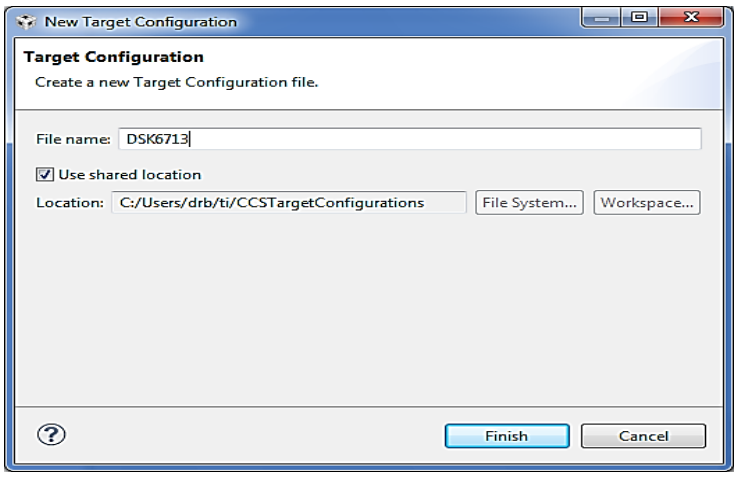
This is the "normal" edit view of CCS v6 where you will manage projects, build and debug your code, and interact with the DSK. Since this is the first time we are using CCS v6, we must set up a "target configuration" for the DSK6713.

Part 2: Setting up a DSK6713 Target Configuration

From the normal view of CCS v6, click on "Window->Show View->Target Configurations" to expose the target configurations panel.

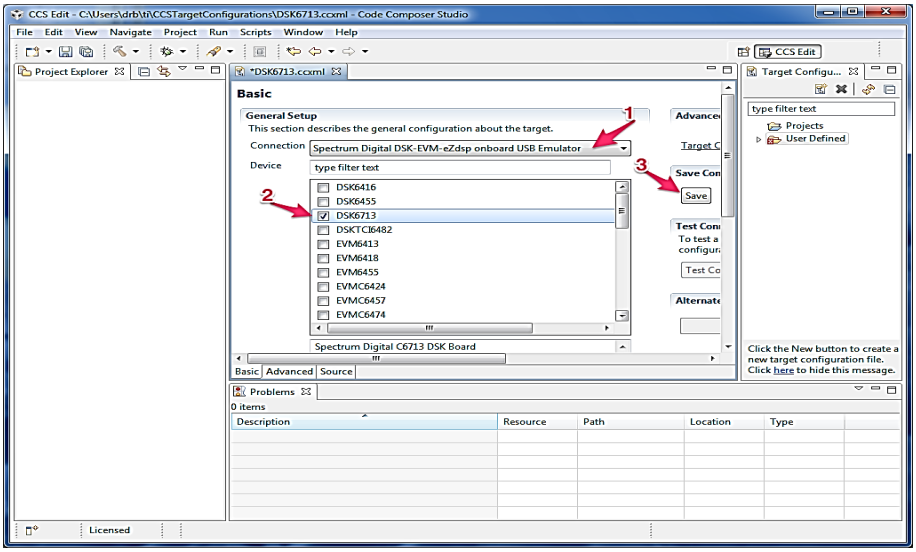


When you first use CCS v6, there won't be any target configurations. So we will set one up for the DSK6713. Note that you only need to do this once. Right click on the "User Defined" folder in the target configurations panel and select "New Configuration".

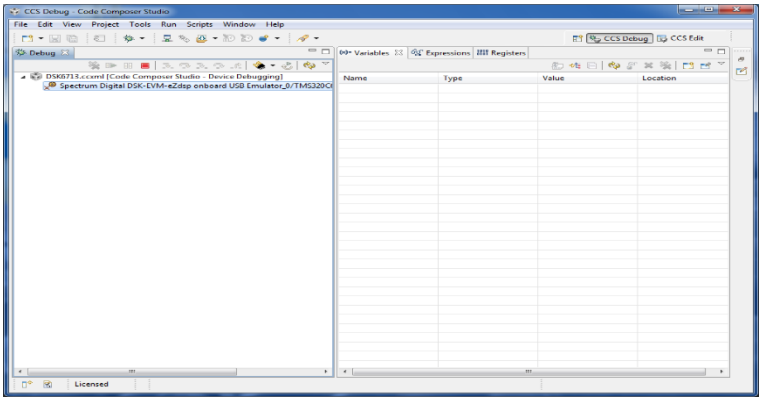


You can name the new target configuration anything you want, but it is a good idea to pick something descriptive. I've named the target configuration DSK6713.ccxml (the extension will be added automatically) and stored it in the default shared location. Click "Finish".

A new panel should appear where you set some options for this DSK6713.ccxml target configuration. In the connection pulldown menu, select "Spectrum Digital DSK-EVM-eZdsp onboard USB Emulator". In the device selector, select "DSK6713"



Now expand the "User Defined" target configurations folder by clicking on the little triangle next to the folder. You should see the new DSK6713.ccxml target configuration in there. Right click on it and select "Launch Selected Configuration". You should see some USB enumeration as CCS v6 talks with the DSK and then you should see something like this window.



If you get an error, then either CCS v6 wasn't installed correctly or you haven't set up the target configuration correctly.

Now click on "Run->Connect Target" (or press Ctrl+Alt+C). You should then see a window something like this The message "GEL StartUp Complete" in the Console means that CCS v6 is successfully talking with the DSK.

3.4 Sine Wave Generation Using Eight Points with DIP Switch Control

This example generates a sinusoidal analog output waveform using a table – lookup method. More importantly, it illustrates some of the features of CCS for editing source files, building a project, accessing the code generation tools, and running a program on the C6713 processor. The C source file *sine8_LED.c* listed in Figure 1.2 is included in the folder *sine8_LED.*

3.4.1 Program Description

The operation of program *sine8_LED.c* is as follows. An array, sine_table , of eight 16 - bit signed integers is declared and initialized to contain eight samples of exactly one cycle of a sinusoid. The value of sine_table[i] is equal to

$1000\sin(2\pi i/8)$ *for* $i \rightarrow 1, 2, 3, \ldots, 7$

Within function *main()* , calls to functions *comm_poll()* , *DSK6713_LED_init()* , and *DSK6713_DIP_init()* initialize the DSK, the AIC23 codec onboard the DSK, and the two multichannel buffered serial ports (McBSPs) on the C6713 processor. Function *comm_poll()* is defined in the file *c6713dskinit.c* , and functions *DSK6713_LED_init()* and *DSK6713_DIP_init()* are supplied in the board support library (BSL) file dsk6713bsl.lib .

The program statement while(1) within the function *main()* creates an infinite loop. Within that loop, the state of DIP switch #0 is tested and if it is pressed down, LED #0 is switched on and a sample from the lookup table is output. If DIP switch #0 is not pressed down, then LED #0 is switched off. As long as DIP switch #0 is pressed down, sample values read from the array sine_table will be output, and a sinusoidal analog output waveform will be generated via the left - hand channel of the AIC23 codec and the LINE OUT and HEADPHONE sockets. Each time a sample value is read from the array sine_table, multiplied by the value of the variable gain, and written to the codec, the index, loopindex , into the array is incremented and when its value exceeds the allowable range for the array(LOOPLENGTH - 1), it is reset to zero.

```
//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 8
short loopindex = 0; //table index
short gain = 10; //gain factor
short sine_table[LOOPLENGTH]=
{0,707,1000,707,0,-707,-1000,-707}; //sine values
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(0)==0) //if DIP #0 pressed
{
DSK6713_LED_on(); //turn LED #0 ON
output_left_sample(sine_table[loopindex++]*gain); //output
if (loopindex >= LOOPLENGTH) loopindex = 0; //reset index
}
else DSK6713_LED_off(0); //else turn LED #0 OFF
} //end of while(1)
} //end of main
```

Each time the function output_left_sample() , defined in source file *C6713dskinit.c* , is called to output a sample value, it waits until the codec, initialized by the function comm_poll() to output samples at a rate of 8 kHz, is ready for the next sample. In this way, once DIP switch #0 has been pressed down it will be tested at a rate of 8 kHz. The sampling rate at which the codec operates is set by the program statement

Uint32 fs = DSK6713_AIC23_FREQ_8KHZ;

One cycle of the sinusoidal analog output waveform corresponds to eight output samples and hence the frequency of the sinusoidal analog output waveform is equal to the codec sampling rate (8 kHz) divided by eight, that is, 1 kHz.

4 Conclusion

In this lab, we introduced the DSP Starter Kit C6713. We learned how to get started with the kit, including how to connect it to a computer and how to install the necessary software. We also learned how to compile basic code on the kit and how to work with basic sinusoids.

We learned that the DSP Starter Kit C6713 is a powerful tool that can be used to develop embedded systems. The kit includes a variety of features that make it easy to get started with embedded development, including a powerful DSP processor, a variety of peripherals, and a comprehensive development environment.