

Lab 1

Introduction to TI's TMS320C6713 DSK DSP Board

This laboratory introduces you to the TMS320C6713 DSK board module with:

- An overview of the functional blocks of the board
- Code Composer Studio (CCS)
- Writing, compiling, and running a simple program
- Learning how to generate a tone
- Learning how to read the DIP switches

1 Introduction

The TI's TMS320C6713 DSK is designed and optimized to perform digital signal processing operations. For short this DSP will be referred to as 'C6713'. The family of this DSP is referred to as 'C6x' or 'C6000'. 'C6713' is a high performance 32-bit floating-point DSP.

The basic operation in digital signal processing is solving the following equation:

$$y[n] = \sum_{k=0}^M b_k x[n-k] - \sum_{k=1}^N a_k y[n-k]$$

The DSP has to be able to perform the above operation very efficiently and very fast. For example, for a 100-tap FIR filter, where $M = 99$ and $N = 0$, the DSP will have to be able to store 99 samples, and perform 100 multiplication and 100 summation operations between every two samples. The DSP implements multiply-accumulate (MAC) hardware with circular addressing capabilities very efficiently.

Some of the TMS320C6713DSK features shown in Figure 1 are

- 225 MHz TMS320C6713 Floating Point DSP
- AIC23 Stereo Codec 8KHz-96KHz sample rate, 16 to 32 bit samples, MIC IN, LINE-IN, LINE-OUT, and HP-OUT (headphone output) jacks. Communication with the AIC23 is carried over two multichannel buffered serial ports (MCBSPs): the MCBSP0 is used as unidirectional channel to send a 16-bit control word to the AIC23, the MCBSP1 is used a bidirectional channel to send and receive audio data.
- Four position user DIP Switches and four LEDs
- 1800 million instructions per second (MIPS) and 1350 MFLOPS

- Software board configuration through registers implemented in CPLD
- Standard expansion connectors for daughter card
- The maximum allowable input signal level at the LINE IN is 1 Vrms. However, the C6713 has a voltage divider circuit with a gain of 0.5 which allows for a maximum of 2 Vrms. Above this level, input signals will be distorted.

TI's Code Composer Studio is a development tool that will be used to program the DSP. The version that comes with the board includes a chip support library (CSL) and a board support library (BSL). The chip support library (CSL) provides a C-language interface for configuring and controlling on-chip peripherals. The BSL provides a C-language interface for configuring and controlling all on-board devices.

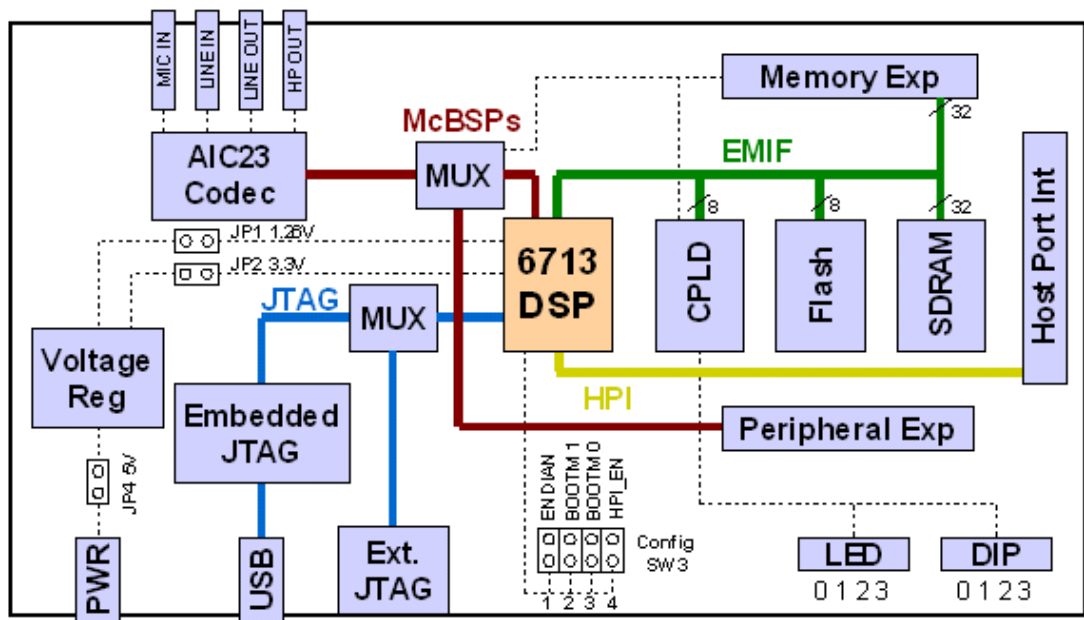


Figure 1: Key features of the C6713 DSK

2 Lab

2.1 Part 1

1. Use the provided guideline document to create a new project.
2. Discretize the continuous-time sine wave given by

$$y(t) = \sin 2\pi f_0 t$$

Assume a sampling rate of $f_s = 8$ KHz, and $f_0 = 1$ KHz.

3. Generate a lookup table for the sine wave in MATLAB and copy it as an array in your C-code.
4. Write the code to cycle through the array and send the value to the codec. Below is a skeleton file that might be useful, Program 1.
5. Connect your headphones or speakers to the output of the codec. Do you hear a tone? if not, what do you think the problem is?
6. Experiment with sending the signal to the right channel then to the left channel, then to both.

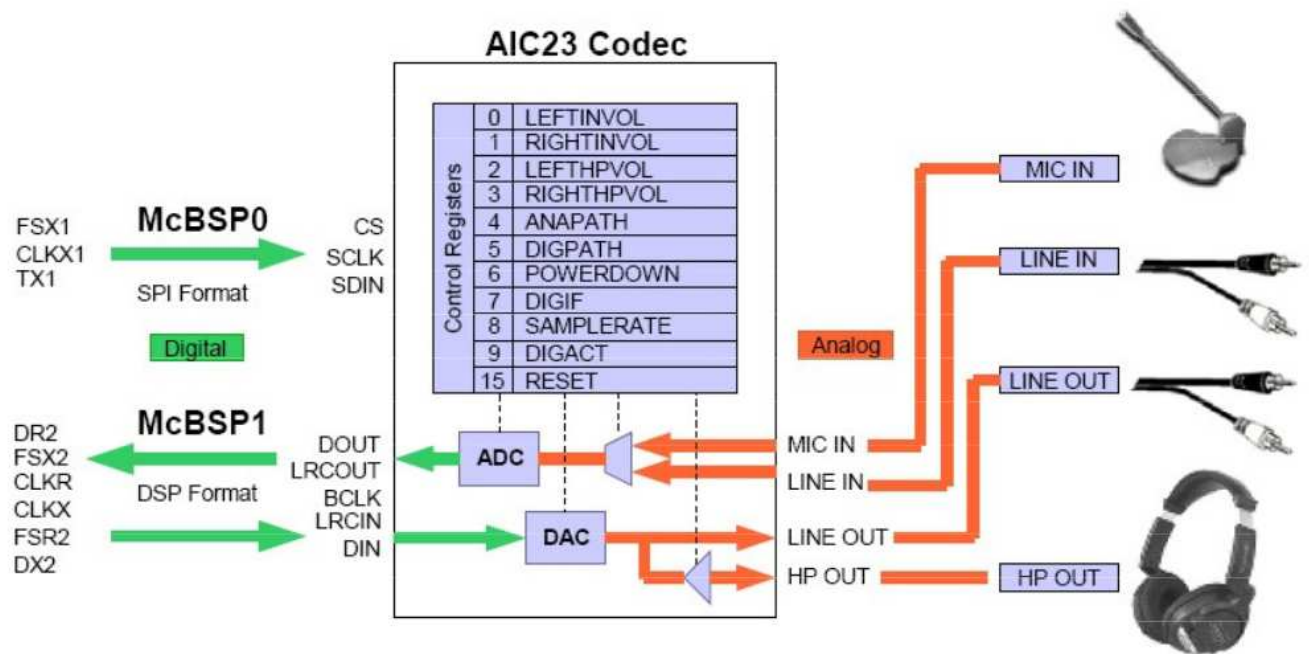


Figure 2: DSK6713/AIC23 interface

2.2 Part 2

1. Write a program to generate the sine wave using the sin function. Make sure that you will keep the angle from overflowing. Below is a skeleton file that might be useful, Program 1.
2. Compile, load and run the program.
3. Use the watch window to change the frequency to 500Hz, to do so you will need to declare the frequency variable as `volatile`. Then highlight the frequency variable, right-click and select Add to Watch Window. The option `volatile` tells the DSP to reload the frequency every time it is needed.

Program 1 Code parts, sample.c

```
/* Created by Aly ElOsery * Last modified : 08/25/2013) */

#include "dsk6713.h"
#include "dsk6713_aic23.h"
#include <math.h>

    Uint32 f_s=DSK6713_AIC23_FREQ_8KHZ; // set sampling rate
    #define LEFT 1
    #define RIGHT 0

    union{
    Uint32 data;
    short channel[2];
    } AIC23_data;

/* YOU MAY NEED SOME MORE CODE HERE */

    void main ()
    {

        comm_poll();

        // Uncomment the following line if using DIP switches
        // DSK6713_DIP_init ();
        //
        // Use the function DSK6713_DIP_get(n) to read the nth DIP switch

    while (1)
    {

        /* YOU MAY NEED SOME MORE CODE HERE */

        /* YOU MAY NEED SOME MORE CODE HERE */

    }
}
```