Lab 5 : Introduction to Code Composer Studio and the C6713 DSK.

September, 2006

1 Overview

The purpose of this lab is to familiarize you with the C6713 DSK and Code Composer Studio (CCS). This lab involves generating and plotting a sine wave in CCS.

2 DSP development system overview

A PC is required to run Code Composer Studio which is required to compile and download code to(run on)the DSP.

2.1 DSP Board highlights

- Texas Instruments TMS320C6713 DSP operating at 225 MHz
- An AIC23 stereo codec
- 8 MB of synchronous DRAM
- 512 KB of non-volatile Flash memory
- 4 user accessible LEDs and DIP switches
- Configurable boot options
- Standard expansion connectors for daughter card use
- JTAG emulation through on-board JTAG emulator with USB host interface or external emulator
- Single voltage power supply (+5V)

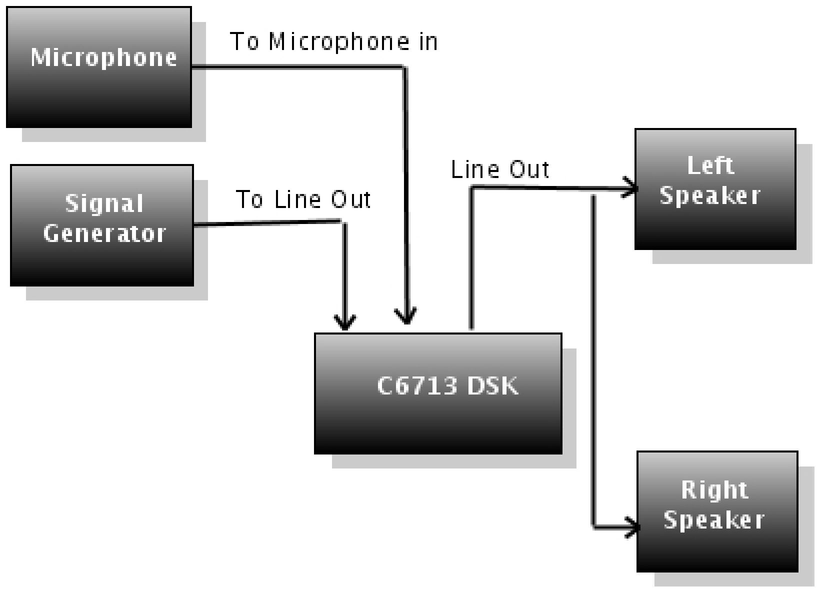


Figure 1: Typical lab setup

2.2 Functional Overview of DSP Board

The DSP on the 6713 DSK interfaces to on-board peripherals through a 32-bit wide EMIF (External Memory Interface). The SDRAM, Flash and CPLD are all connected to the bus. EMIF signals are also used for daughter cards. The DSP interfaces to analog audio signals through an on-board AIC23 codec and four 3.5 mm audio jacks (microphone input, line input, line output, and headphone output). The codec can select the microphone or the line input as the active input. The analog output is sent to both the line out and headphone out connectors. The line out has a fixed gain, while the headphone out allows for an adjustable gain. connectors.

A programmable logic device called a CPLD (Complex Programmable Logic Device)is used to implement logic that ties the board components together. The DSK includes 4 LEDs and a 4 position DIP switch which allow for interactive feedback.

2.3 Software

In this lab, CCS is used to write, compile and download code on to the C6713 DSK. CCS includes a C compiler, an assembler and a linker. It also has graphical abilities and supports real-time debugging.

The compiler compiles a C source program with extension .c to produce

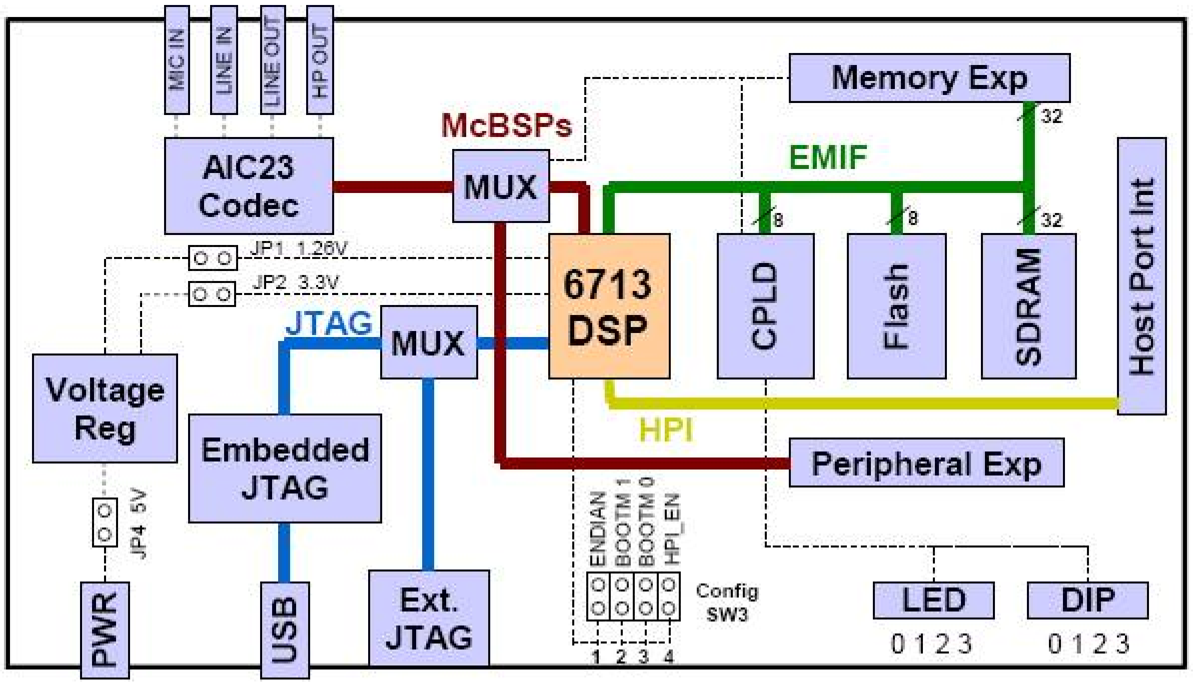


Figure 2: C6713 DSK block diagram

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 to produce an executable file with extension.out.This executable file represents a linked common object file format(COFF).This executable file can be loaded and run directly on the C6713 DSP.

3 Setting up the system

Follow the instructions below to ensure that CCS and the C6713 DSK are set up properly.

- Launch CCS from the icon on the desktop.
- In the GEL menu, select Check DSK → Quick test.
- The message diplayed is **Switches:15 Board Revision:1 CPLD Revision:2**. This assumes the four DIP switches (0,1,2,3) are in the up position. Depress switch 3 and run the quick test again. The message now displayed should be **Switches:7 Board Revision:1 CPLD Revision:2**.

4 Programming the C6713 DSK

We now write a program to generate a sine wave at 1KHz using a lookup table. This program uses the AIC23 codec in the 6713 Board Support Library to generate a tone. The tone is generated based on the state of DIP switch 0. When the switch is depressed, the tone is generated and `LED#0` turns on. The program also creates a buffer to store the output data in memory. As long as `DIP#0` is in the on position, the program outputs a tone and a graph. This program uses an infinite loop to poll the DIP switch.

4.1 What the program does

The main loop of the code writes each data point in the sine wave table out to the codec using the AIC23 codec package of the BSL. Each write function sends a single 16 bit sample to the codec. In this case the same data is sent out twice, once to the left channel and once to the right channel. The codec is configured to accept data at a rate of 48,000 stereo samples per second. Since the sine table is 48 entries long, the resulting output wave will be a 1KHz sine wave with the same output on both the left and right channels.

The serial port is used to transmit data to the codec at a much slower rate than the DSP can process data. It accepts data 16 bits at a time and shifts them out slowly one at a time. The write function returns a 1 if the write is completed successfully or a 0 if the serial channel is busy. The **while()** loop around the writes waits while the serial port is busy so program can be synchronized to the data rate of the codec.

```
// sinegraph.c
// The C6713 Board Support Library(BSL) has several
// modules, each of which has it's own include file.
// The file dsk6713.h must be used in every program
// that uses the BSL. This example also includes
// dsk6713_led.h and dsk6713_dip.h because it uses
// the LED and DIP control on the board

#include "dsk6713.h"
#include "dsk6713_aic23.h"
#include "dsk6713_led.h"
#include "dsk6713_dip.h"

//table index
short loop = 0;
//gain factor
short gain = 10;
//output buffer
Int16 out_buffer[256]
//size of buffer
const short BUFFERLENGTH = 256;
//counter for buffer
int i = 0;

// Codec configuration
DSK6713_AIC23_Config config = { \
    0x0017,  /* 0 DSK6713_AIC23_LEFTINVOL\
    0x0017,  /* 1 DSK6713_AIC23_RIGHTINVOL\
    0x00d8,  /* 2 DSK6713_AIC23_LEFTHPVOL\
    0x00d8,  /* 3 DSK6713_AIC23_RIGHTHPVOL\
    0x0011,  /* 4 DSK6713_AIC23_ANAPATH\
    0x0000,  /* 5 DSK6713_AIC23_DIGPATH\
    0x0000,  /* 6 DSK6713_AIC23_POWERDOWN\
    0x0043,  /* 7 DSK6713_AIC23_DIGIF\
    0x0081,  /* 8 DSK6713_AIC23_SAMPLERATE\
    0x0001   /* 9 DSK6713_AIC23_DIGACT\
    };
// Lookup table
Int16 sine_table[48] = {
    0x0000, 0x10b4, 0x2120, 0x30fb, 0x3fff, 0x4dea,
    0x5a81, 0x658b, 0x6ed8, 0x763f, 0x7ba1, 0x7ee5,
    0x7ffd, 0x7ee5, 0x7ba1, 0x76ef, 0x6ed8, 0x658b,
    0x5a81, 0x4dea, 0x3fff, 0x30fb, 0x2120, 0x10b4,
    0x0000, 0xef4c, 0xdee0, 0xcf06, 0xc002, 0xb216,
    0xa57f, 0x9a75, 0x9128, 0x89c1, 0x845f, 0x811b,
    0x8002, 0x811b, 0x845f, 0x89c1, 0x9128, 0x9a76,
    0xa57f, 0xb216, 0xc002, 0xcf06, 0xdee0, 0xef4c
};

Uint32 fs = DSK6713_AIC23_FREQ_48KHZ;

// main() - Main code routine, initializes BSL and
// runs LED application
void main()
{
        DSK6713_AIC23_CodecHandle hCodec;

  // Initialize the board support library, must be first
  // BSL call
  DSK6713_init();

  // Initialize the LED and DIP switch modules of the BSL
  DSK6713_LED_init();
  DSK6713_DIP_init();

  // Start the codec
  hCodec = DSK6713_AIC23_openCodec(0, &config);

// DIP Switch API
// DSK6713_DIP_get(Uint32 dipNum)
// Return value 0   Specified switch is off
// Return value 1   Specified switch is on

//infinite loop
while(1)
{
        if (DSK6713_DIP_get(0) == 0)
        {
        //turn LED#0 on
        DSK6713_LED_on(0);
        out_buffer[i] = sine_table[loop];
        // while(return_value is not zero)
        // see DSK6713_AIC23_write(...)
        // send data to left channel
        //output every Ts SW0
```

```
   while  (!DSK6713_AIC23_write(
                      hCodec,  sine_table[loop]*gain));
          // send data to right channel
   while  (!DSK6713_AIC23_write(
                      hCodec,  sine_table[loop]*gain));

          i++;
          if(i==BUFFERLENGTH)  i=0;
          //check for end of table
          if (++loop > 47) loop = 0;
                  }
                  //LED#0 off
                  else DSK6713_LED_off(0);
          }       //end of while(1)
}
```

The array `out_buffer[]` stores the sine data for plotting within CCS The statement **while(1)** within the function **main** creates an infinite loop. When dip switch `#0` is pressed, `LED#0` turns on and the sinusoid is generated.

The loop index is incremented until the end of the table is reached, after which it is re-initialized to zero.

The following two commands are used to initialize and shut down the audio codec and are found at the beginning and end of all programs that use the BSL codec module.

```
DSK6713_openCodec()
```

returns a handle that is passed to each of the other codec functions.

```
hCodec = DSK6713_AIC23_openCodec(0, &config);
```

opens the codec and

```
hCodec = DSK6713_AIC23_closeCodec(0, &config);
```

closes the codec.

4.2 Create a project in CCS

1. Type the code and save as `sinegraph.c`
2. Create a project in CCS (Project → New). Save the project as `sinegraph.pjt`
3. Add sinegraph.c to the project.
4. Add the required library files to project (ti/c6000/dsk6713/lib/).
5. Scan file dependencies (Project → Scan file dependencies).

6. Set the appropriate compiler options (Project → Build options). The following compiler options are suggested.

   - For the 'Basic' category:
     target version : C670x
     gen. debug info : full
     opt speed vs size: speed most critical
     prog level opt :none
   - For 'Feedback' category:
     interlisting : opt/c and ASM(-s)

   - For 'Preprocessor' category :
     Define symbols:`CHIP_6713`

4.3 Building and Running the Project

The project can now be built and run.

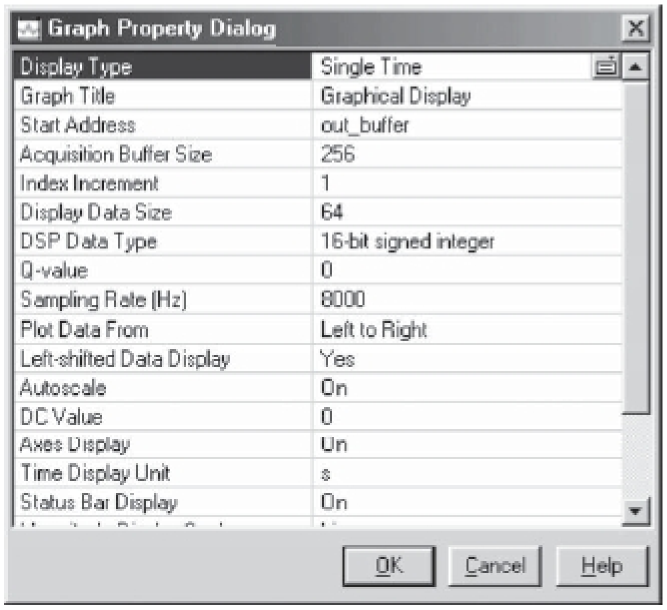
1. Select Project → Rebuild All or press the toolbar with the three down arrows.This compiles and assembles the source file(s). The resulting object files are then linked with the library files. This creates an executable file `sinegraph.out`. that can be loaded into the C6713 processor and run. The building process causes all the dependent files to be included (in case one forgets to scan for all the file dependencies).

2. Select File → Load Program in order to load `sinegraph.out`to the DSK. It should be in the folder `sinegraph\Debug`.Select Debug → Run or use the toolbar with the running man. Connect a speaker to the LINE OUT connector on the DSK. Press the dip switch `#0`.

Plotting with CCS The output buffer is updated continuously every 256 points .CCS can be used to plot the current output data stored in the buffer `out_buffer`.

1. Select View→Graph→Time/Frequency. Change the Graph Property Dialog so that the options are as indicated in figure 3. The starting address of the output buffer is `out_buffer`.The other options can be left as default.

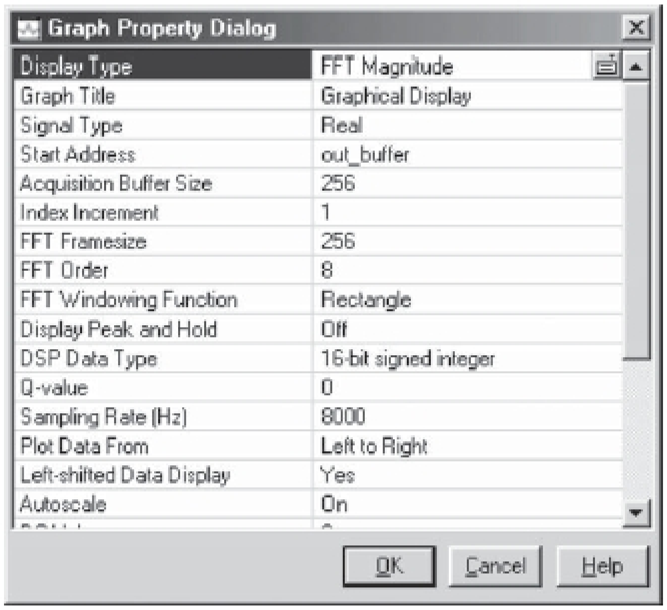
2. Choose a fast Fourier transform (FFT) order so that the frame size is 2 order. Press OK and verify that the FFT magnitude plot is as shown (figure 4) .The spike at 1000 Hz represents the frequency of the sinusoid generated.



Graph Property Dialog

Display Type	Single Time
Graph Title	Graphical Display
Start Address	out_buffer
Acquisition Buffer Size	256
Index Increment	1
Display Data Size	64
DSP Data Type	16-bit signed integer
Q-value	0
Sampling Rate (Hz)	8000
Plot Data From	Left to Right
Left-shifted Data Display	Yes
Autoscale	On
DC Value	0
Axes Display	On
Time Display Unit	s
Status Bar Display	On

Figure 3: Time domain graph property dialog



Graph Property Dialog

Display Type	FFT Magnitude
Graph Title	Graphical Display
Signal Type	Real
Start Address	out_buffer
Acquisition Buffer Size	256
Index Increment	1
FFT Framesize	256
FFT Order	8
FFT Windowing Function	Rectangle
Display Peak and Hold	Off
DSP Data Type	16-bit signed integer
Q-value	0
Sampling Rate (Hz)	8000
Plot Data From	Left to Right
Left-shifted Data Display	Yes
Autoscale	On

Figure 4: Frequency domain graph property dialog