Lab2
Data Acquisition

Dr. Aly El-Osery

September 4, 2011

The purpose of this lab is to get more familiar with the DSK board, understand the operation of the Codec. By the end of the lab you will be able to input and output signals through the Codec chip as well as see the effects of aliasing.

1 Introduction

TMS320C6713DSK has a AIC23 Stereo Codec 8KHz-96KHz sample rate, 16 to 32 bit. The role of the codec is to act as an analog-to-digital and digital-to-analog converter. Using the codec chip you will be able to input and output signals, and therefore it is important to understand its operation and limitations.

2 Lab

2.1 Part 1

1. Start a new project.
2. Write a code that will allow you to read/write from and to the codec. You can use same code as the one you used in the previous lab to read and write.
3. Set the function generator to output a sinusoidal signal with amplitude 500mV and frequency 1kHz. Connect the function generator to the board.
4. Set the sampling frequency to 96kHz.
5. Connect an oscilloscope to the output of the board.
6. Record and plot the magnitude of the output signal as you vary the frequency from 1kHz to 96kHz. Explain what you are recording.
7. Change the sampling frequency to 8kHz. Vary the frequency and record your observations. Comment on your results.

2.2 Part 2

As mentioned previously, the read function for the AIC23 returns only one channel at a time. By changing the configuration of the MCBSP we can read both channels at the same time. To do that you need to use dsk6713config.h and modify your code to use the new configuration as shown in the sample file, sample.c to read and write the data instead of the standard function. All you have to do is save the file from the link and then include it into your project, also under `build options/compiler/advanced/` select memory model `far`. Now you are reading both channels at the same time where the first 16bits are the left channel and the second 16bits are the right channel.

1. Change the sampling frequency to 96kHz and modify your code to output only every third sample. Vary the frequency and record your observations.

2. Modify your code to read the signal from the function generator and output this signal to both channels.

```
//======== sample.c ==============
/*
 *  Created by Aly El-Osery
 *  Last modified: 08/31/10
 */

#include "lab2cfg.h"
#include <dsk6713.h>
#include <dsk6713_aic23.h>
#include "dsk6713config.h"
#include <math.h>
#define pi 4*atan(1) //3.141592653589


Uint32 fs = DSK6713_AIC23_FREQ_48KHZ;

int counter=0;
int i;

void main()
{
    DSK6713_AIC23_CodecHandle hCodec;

    /* Initialize the board support library, must be called first */
    DSK6713_init();

    /* Start the codec */
    hCodec = DSK6713_AIC23_openCodec(0, &config);
        /* Configure codec control McBSP */

    /* Configure codec data McBSP */
    MCBSP_config(DSK6713_AIC23_DATAHANDLE, &AIC23CfgData);

    /* Reset the AIC23 */
    DSK6713_AIC23_rset(0, DSK6713_AIC23_RESET, 0);

    /* Configure the rest of the AIC23 registers */
    DSK6713_AIC23_config(0, &config);

    /* Clear any garbage from the codec data port */
    if (MCBSP_rrdy(DSK6713_AIC23_DATAHANDLE))
        MCBSP_read(DSK6713_AIC23_DATAHANDLE);

    /* Start McBSP2 as the codec data channel */
    MCBSP_start(DSK6713_AIC23_DATAHANDLE, MCBSP_XMIT_START |
     MCBSP_RCV_START | MCBSP_SRGR_START | MCBSP_SRGR_FRAMESYNC, 220);
    DSK6713_AIC23_setFreq(hCodec, fs);

    for (;;)
    {
      while(!(DSK6713_AIC23_read(hCodec, &AIC_data.uint)));

      /* You may need to add code here */

      while(!(DSK6713_AIC23_write(hCodec, AIC_data.uint)));
    }
}
```

```
//======== dsk6713config.h ==============
//
// General definitions for the configuration of the AIC23
// Original by:
// Digital Signal Processing and Applications with the
// TMS320C6713 and TMS320C6416 DSK
// Rulph Chassaing and Donald Reay
//
// Modified by Aly El-Osery
// Modification data 08/2010
//
//
#define LEFT  1                    //data structure for union of 32-bit data
#define RIGHT 0                    //into two 16-bit data
union {
  Uint32 uint;
  short channel[2];
  } AIC_data;

DSK6713_AIC23_Config config = { \
    0x0017,   /* 0 LEFTINVOL  Left line input channel volume */ \
    0x0017,   /* 1 RIGHTINVOL Right line input channel volume */\
    0x01f9,   /* 2 LEFTHPVOL  Left channel headphone volume */  \
    0x01f9,   /* 3 RIGHTHPVOL Right channel headphone volume */ \
    0x0011,   /* 4 ANAPATH    Analog audio path control */      \
    0x0000,   /* 5 DIGPATH    Digital audio path control */     \
    0x0000,   /* 6 POWERDOWN  Power down control */             \
    0x0043,   /* 7 DIGIF      Digital audio interface format */ \
    0x0081,   /* 8 SAMPLERATE Sample rate control */            \
    0x0001    /* 9 DIGACT     Digital interface activation */   \
};
   // This is needed to modify the BSL's data channel McBSP configuration
MCBSP_Config AIC23CfgData = {
        MCBSP_FMKS(SPCR, FREE, NO)              |
        MCBSP_FMKS(SPCR, SOFT, NO)              |
        MCBSP_FMKS(SPCR, FRST, YES)             |
        MCBSP_FMKS(SPCR, GRST, YES)             |
        MCBSP_FMKS(SPCR, XINTM, XRDY)           |
        MCBSP_FMKS(SPCR, XSYNCERR, NO)          |
        MCBSP_FMKS(SPCR, XRST, YES)             |
        MCBSP_FMKS(SPCR, DLB, OFF)              |
        MCBSP_FMKS(SPCR, RJUST, RZF)            |
        MCBSP_FMKS(SPCR, CLKSTP, DISABLE)       |
        MCBSP_FMKS(SPCR, DXENA, OFF)            |
        MCBSP_FMKS(SPCR, RINTM, RRDY)           |
        MCBSP_FMKS(SPCR, RSYNCERR, NO)          |
        MCBSP_FMKS(SPCR, RRST, YES),

        MCBSP_FMKS(RCR, RPHASE, SINGLE)         |
        MCBSP_FMKS(RCR, RFRLEN2, DEFAULT)       |
        MCBSP_FMKS(RCR, RWDLEN2, DEFAULT)       |
        MCBSP_FMKS(RCR, RCOMPAND, MSB)          |
        MCBSP_FMKS(RCR, RFIG, NO)               |
        MCBSP_FMKS(RCR, RDATDLY, 0BIT)          |
        MCBSP_FMKS(RCR, RFRLEN1, OF(0))         | // This changes to 1 FRAME
        MCBSP_FMKS(RCR, RWDLEN1, 32BIT)         | // This changes to 32 bits per frame
        MCBSP_FMKS(RCR, RWDREVRS, DISABLE),

        MCBSP_FMKS(XCR, XPHASE, SINGLE)         |
        MCBSP_FMKS(XCR, XFRLEN2, DEFAULT)       |
```

```
        MCBSP_FMKS(XCR, XWDLEN2, DEFAULT)          |
        MCBSP_FMKS(XCR, XCOMPAND, MSB)             |
        MCBSP_FMKS(XCR, XFIG, NO)                  |
        MCBSP_FMKS(XCR, XDATDLY, 0BIT)             |
        MCBSP_FMKS(XCR, XFRLEN1, OF(0))            | // This changes to 1 FRAME
        MCBSP_FMKS(XCR, XWDLEN1, 32BIT)            | // This changes to 32 bits per frame
        MCBSP_FMKS(XCR, XWDREVRS, DISABLE),

        MCBSP_FMKS(SRGR, GSYNC, DEFAULT)           |
        MCBSP_FMKS(SRGR, CLKSP, DEFAULT)           |
        MCBSP_FMKS(SRGR, CLKSM, DEFAULT)           |
        MCBSP_FMKS(SRGR, FSGM, DEFAULT)            |
        MCBSP_FMKS(SRGR, FPER, DEFAULT)            |
        MCBSP_FMKS(SRGR, FWID, DEFAULT)            |
        MCBSP_FMKS(SRGR, CLKGDV, DEFAULT),

        MCBSP_MCR_DEFAULT,
        MCBSP_RCER_DEFAULT,
        MCBSP_XCER_DEFAULT,

        MCBSP_FMKS(PCR, XIOEN, SP)                 |
        MCBSP_FMKS(PCR, RIOEN, SP)                 |
        MCBSP_FMKS(PCR, FSXM, EXTERNAL)            |
        MCBSP_FMKS(PCR, FSRM, EXTERNAL)            |
        MCBSP_FMKS(PCR, CLKXM, INPUT)              |
        MCBSP_FMKS(PCR, CLKRM, INPUT)              |
        MCBSP_FMKS(PCR, CLKSSTAT, DEFAULT)         |
        MCBSP_FMKS(PCR, DXSTAT, DEFAULT)           |
        MCBSP_FMKS(PCR, FSXP, ACTIVEHIGH)          |
        MCBSP_FMKS(PCR, FSRP, ACTIVEHIGH)          |
        MCBSP_FMKS(PCR, CLKXP, FALLING)            |
        MCBSP_FMKS(PCR, CLKRP, RISING)
    };
```