# Lab2
# Data Acquisition

Dr. Aly El-Osery

September 5, 2017

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 using polling and interrupts.

## 1 Lab Part 1

### 1.1 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.

### 1.2 Lab

1. Start a new project.
2. Write a code that will allow you to read/write from and to the codec.
3. What is the maximum input voltage to the codec? **confirm value with TA before proceeding**.
4. Set the sampling frequency to 96kHz.
5. Set the function generator to output a sinusoidal signal with amplitude 500mV and frequency 1kHz. Connect the function generator to the board.
6. Connect an oscilloscope to the output of the board.
7. Output the input signal on both the left and the right channel using
   (a) `output_sample` function, then using
   (b) `output_left_sample` and `output_right_sample` functions

   are both approaches the same, why or why not?

8. Record and plot the magnitude of the output signal as you vary the frequency from 1kHz to 96kHz. Explain what you are recording. What is the 3dB cutoff of the filter you are observing.

9. Change the sampling frequency to 8kHz. Vary the frequency and record your observations. Comment on your results. What is the 3dB cutoff of the filter you are observing.

# 2 Lab Part 2

## 2.1 Introduction

The codec (see Figure 1) is connected to the DSK6713 through the multichannel buffered serial port (McBSP) interface. The block diagram of the McBSP is shown in Figure 2.

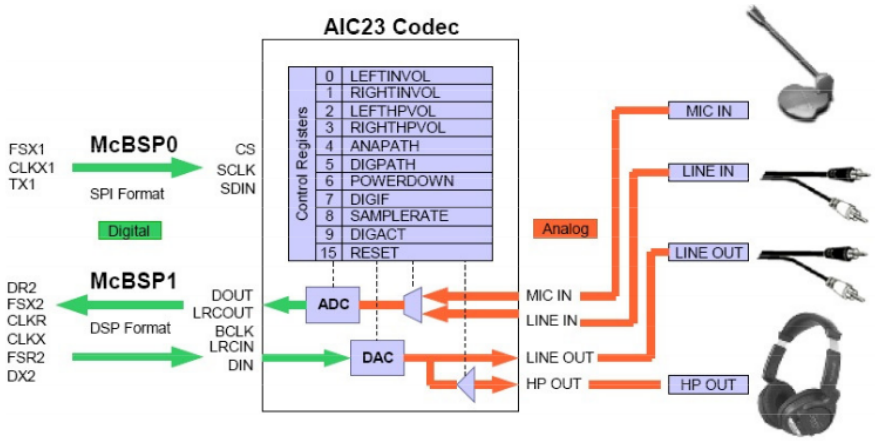


Figure 1: DSK6713/AIC23 interface

Transmit (Tx)

1. The CPU or DMA writes a 32-bit word in parallel into the Data Transmit Register (DXR)
2. XRDY flag ready is cleared
3. When transmit frame synch (FSX) goes high, data is serially shifted out of the XSR, then parallel transfer of DXR into XSR
4. XRDY is set high
5. The serial port transmitter sends an interrupt request (XINT) to the CPU when the XRDY Flag makes a transition from 0 to 1, if interrupts are setup through the Serial Port Control Register (SPCR)

Receive (Rx)

1. FSR goes high, the received bits are shifted serially intro the RSR

2. When an element with the configured number of bits is received, the 32-bit RSR is transferred in parrallel to the Receive Buffer Register (RBR) it is empty.

3. The RBR is copied to the DRR if it is empty.

4. The RRDY bit in the SPCR is set to 1 when the RBR is moved to the DRR, and it is cleared when the DRR is read.

5. When the RRDY transitions from 0 to 1, the McBSP generates a CPU interrupt request (RINT) the SPCR is setup to do so.

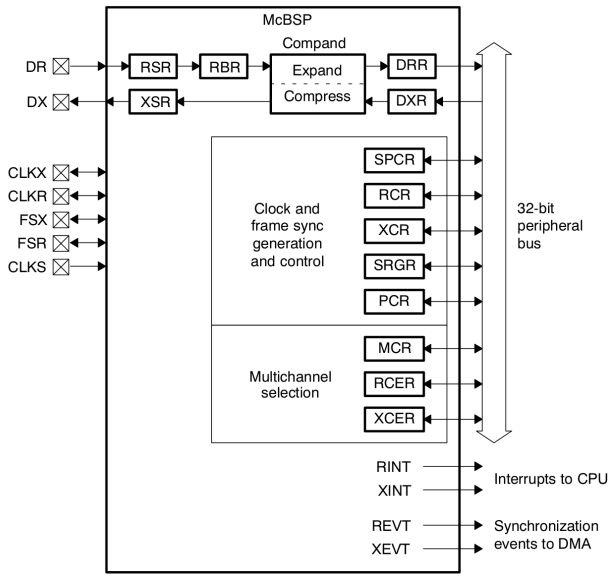


Figure 2: Multichannel Buffered Serial Port (McBSP)

## 2.2 Lab

1. Start a new project.

2. Write a code that will allow you to read/write from and to the codec. This time you will have an empty infinite loop in the `main` function, and will create an interrupt routine for the read and write, e.g.,

```
1    interrupt void c_int11(void)
2    {
3        \\insert your code below
4    }
```

   This time you will need the file `Vectors_intr.asm` instead of `Vectors_poll.asm` as part of your project and `comm_intr()` instead of `comm_poll()` . Verify that you are able to read a signal from the function generator and pass it through to the output.

3. Change the sampling frequency to 32kHz. Vary the frequency and record your observations. Comment on your results. What is the 3dB cutoff of the filter you are observing.