Content

INTRODUCTION

In this project, a real-time audio signal from a mobile phone is processed using the TMS320C6713 Digital Signal Processor to identify the dominant frequency band. The analog audio input is fed into the DSP kit, where the onboard AIC23 codec samples the signal at 8 kHz and stores 256 samples in a buffer.

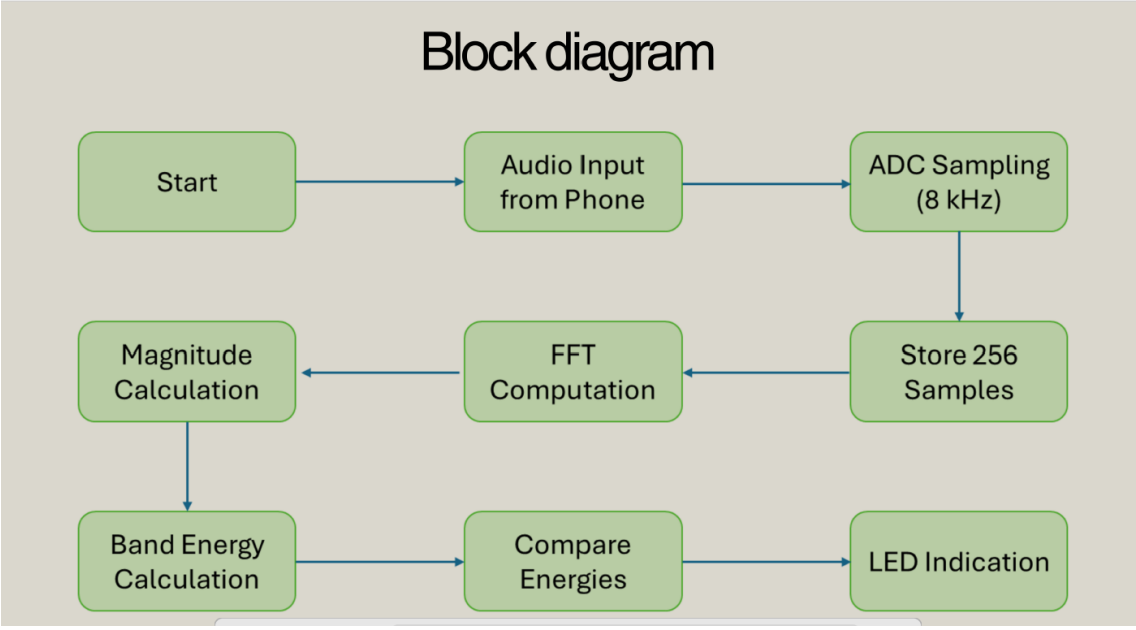
The collected samples are processed using the Fast Fourier Transform (FFT) to convert the signal from the time domain to the frequency domain. The magnitude of the FFT output is calculated, and the frequency spectrum is divided into low (below 300 Hz), mid (300–2000 Hz), and high (above 2000 Hz) frequency bands.

The energy in each band is computed and compared to determine the dominant frequency range. Based on this comparison, the corresponding LED on the TMS320C6713 board is turned ON, providing a real-time visual indication of the dominant frequency content in the input audio signal.

OBJECTIVE

The objective of this project is to implement a real-time audio spectrum visualization system on the TMS320C6713 DSP processor. The audio input is acquired through the AIC23 codec, processed using the Fast Fourier Transform (FFT), and classified into low, mid, and high frequency bands. LEDs on the development board are used to display the dominant frequency band, demonstrating real-time signal analysis and DSP hardware interfacing.

BLOCK DIAGRAM



Fig(a) Audio Frequency Band Detection Flow

EXPLANATION

In this project, real-time frequency analysis of an audio signal is performed using the TMS320C6713 floating-point DSP processor. The analog audio input from a mobile phone is connected to the Line-In port of the DSP board, where the onboard AIC23 stereo codec performs signal conditioning and analog-to-digital conversion. The codec samples the input at a rate of 8 kHz and sends the sampled data to the DSP through the McBSP interface. A buffer of 256 time-domain samples is collected for each analysis frame, which provides a balance between frequency resolution and real-time response. After buffering, the Fast Fourier Transform (FFT) algorithm is applied to convert the signal from the time domain to the frequency domain, leveraging the C6713's hardware-optimized FFT library functions for efficient computation.

The magnitude spectrum is computed from the complex FFT output by taking the square root of the sum of squares of the real and imaginary components. The spectrum is then divided into three perceptually relevant frequency bands: low (<300 Hz), mid (300–2000 Hz), and high (>2000 Hz). For each band, the signal energy is calculated by summing the squared magnitudes, representing the power distribution across the spectrum. By comparing the energy levels of the three bands, the dominant frequency region of the input audio is identified. This classification logic drives the on-board LEDs of the C6713, where each LED corresponds to one of the three frequency bands. As a result, the LEDs provide a real-time visual indication of whether the incoming audio signal is primarily low-pitch, mid-range, or high-frequency in nature.

PROGRAM

```
#include <math.h>
#include <stdio.h>
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#include "dsk6713_led.h"
#define FFT_SIZE     256
#define SAMPLE_RATE  8000
#define PI           3.141592653589793
Uint32 input_sample;
float real[FFT_SIZE];
float imag[FFT_SIZE];
float mag[FFT_SIZE / 2];
float lowEnergy, midEnergy, highEnergy;
DSK6713_AIC23_CodecHandle hCodec;
DSK6713_AIC23_Config config = {
    0x0017,  // Left line input
    0x0017,  // Right line input
    0x01F9,  // Left headphone
    0x01F9,  // Right headphone
    0x0011,  // Analog path
    0x0000,  // Digital path
    0x0000,  // Power down
    0x0043,  // Digital format
    0x0081,  // 8 kHz sample rate
```

```
 0x0001   // Activate
};
void bit_reverse(float *r, float *i)
{
    int j = 0;
    int k;
    int bit;
    float tr;
    for (k = 1; k < FFT_SIZE; k++)
    {
        bit = FFT_SIZE >> 1;
        while (j & bit)
        {
            j ^= bit;
            bit >>= 1;
        }
        j |= bit;
        if (k < j)
        {
            tr = r[k];
            r[k] = r[j];
            r[j] = tr;

            tr = i[k];
            i[k] = i[j];
            i[j] = tr;
```

```
        }
    }
}
void fft_compute(float *r, float *i)
{
    int len, p, k;
    int even, odd;
    float ang, wr, wi, tr, ti;
    for (len = 2; len <= FFT_SIZE; len <<= 1)
    {
        ang = -2.0 * PI / len;
        for (p = 0; p < FFT_SIZE; p += len)
        {
            for (k = 0; k < len / 2; k++)
            {
                even = p + k;
                odd  = even + len / 2;
                wr = cos(ang * k);
                wi = sin(ang * k);
                tr = wr * r[odd] - wi * i[odd];
                ti = wr * i[odd] + wi * r[odd];
                r[odd]  = r[even] - tr;
                i[odd]  = i[even] - ti;
                r[even] = r[even] + tr;
                i[even] = i[even] + ti;
```

```
            }
        }
    }
}
int main(void)
{
    int i;
    float freq;
    DSK6713_init();
    DSK6713_LED_init();
    hCodec = DSK6713_AIC23_openCodec(0, &config);
     printf("FFT Band Energy Project Running\n");
     while (1)
    {
        for (i = 0; i < FFT_SIZE; i++)
        {
            DSK6713_AIC23_read(hCodec, &input_sample);
            real[i] = (float)((Int16)input_sample);
            imag[i] = 0.0;
        }
        bit_reverse(real, imag);
        fft_compute(real, imag);
        for (i = 0; i < FFT_SIZE / 2; i++)
        {
            mag[i] = sqrt(real[i]*real[i] + imag[i]*imag[i]);
        }
```

```
        lowEnergy = midEnergy = highEnergy = 0.0;
        for (i = 1; i < FFT_SIZE / 2; i++)
        {
            freq = (i * SAMPLE_RATE) / (float)FFT_SIZE;
            if (freq < 300.0)
                lowEnergy += mag[i];
            else if (freq < 2000.0)
                midEnergy += mag[i];
            else
                highEnergy += mag[i];
        }
        DSK6713_LED_off(0);
        DSK6713_LED_off(1);
        DSK6713_LED_off(2);
        if (lowEnergy > midEnergy && lowEnergy >
highEnergy)
            DSK6713_LED_on(0);
        else if (midEnergy > lowEnergy && midEnergy >
highEnergy)
            DSK6713_LED_on(1);
        else
            DSK6713_LED_on(2);
        printf("LOW=%f  MID=%f  HIGH=%f\n",
              lowEnergy, midEnergy, highEnergy);
    }
}
```

Steps For Execution

1.Connect the mobile phone audio output to the Line-In input of the TMS320C6713 DSK board.

2.Power ON the DSP board and initialize the onboard AIC23 codec.

3.Open Code Composer Studio (CCS) and create a new DSP project.

4.Configure the sampling rate to 8 kHz and enable data transfer through the McBSP interface.

5.Include necessary DSP libraries (codec driver, FFT library, and board support files).

6.Set up a buffer to collect 256 audio samples from the codec.

7. Continuously sample the analog audio signal and store the digitized values in the buffer.

8.Apply FFT to the 256-sample frame to convert the signal to the frequency domain.

9.Compute the magnitude of the FFT output from real and imaginary components.

10.Divide the magnitude spectrum into low, mid, and high-frequency regions.

11.Calculate the energy in each band by summing the squared magnitudes of the respective FFT bins.

12.Compare the energies to determine the dominant frequency band.

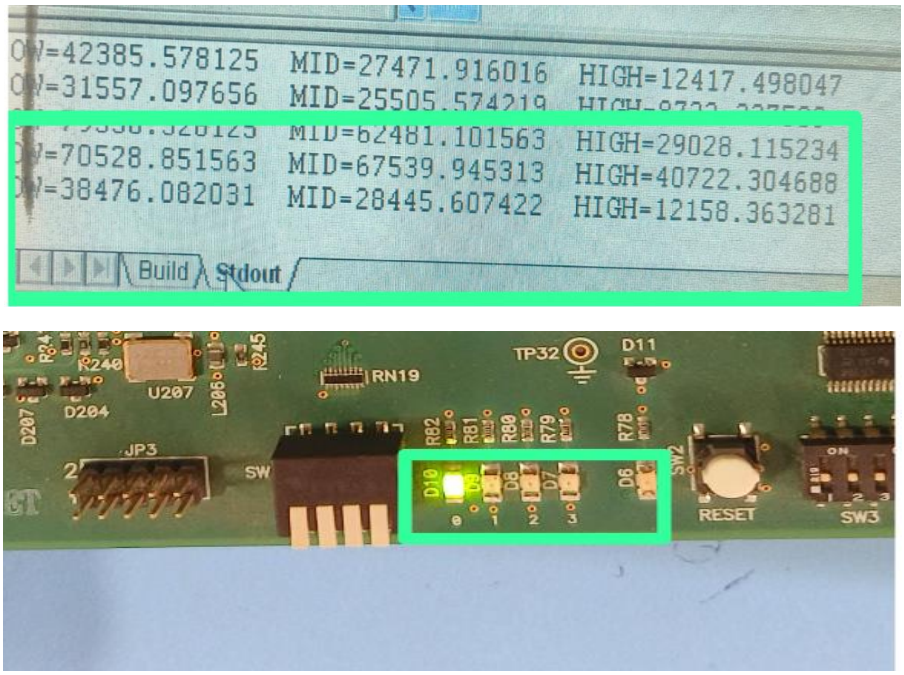
13.Turn ON the LED corresponding to the dominant band (low/mid/high).

14.Repeat the entire sampling and analysis process continuously for real-time operation.

OUTPUT

i) Low frequency band(20-300Hz)

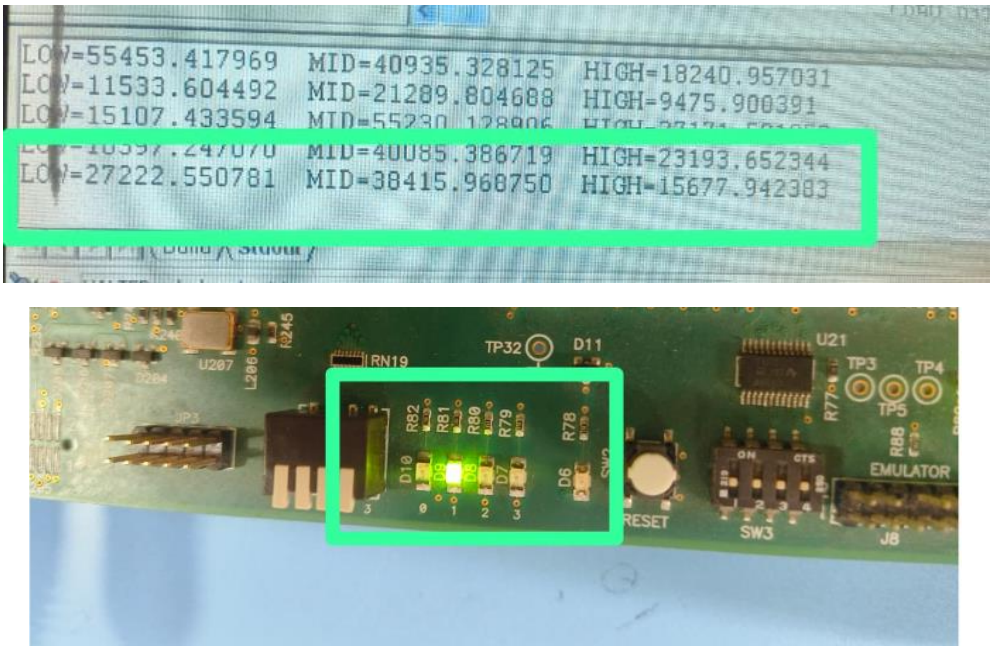
In this case, the dominant audio energy is concentrated in the low-frequency range (20–300 Hz). The DSP identifies the low band as the highest magnitude segment. Accordingly, **LED(0)** is activated to indicate the low-frequency dominance.



Fig(i). Low energy band is high. Led 0 is on

ii)Medium frequency band(300-2000Hz)

Here, the audio signal contains higher energy in the medium-frequency band (300–2000 Hz). The FFT magnitude values show that the medium band has the highest energy compared to the others. Therefore, **LED(1)** turns ON to represent medium-band dominance.



Fig(ii). Medium energy band is high. Led 1 is ON

Application: Baby Cry Detection

This FFT-based audio band energy detection system can be effectively used for baby cry detection and monitoring. A baby's cry typically contains strong energy in the mid and high frequency ranges (approximately 500 Hz to 3000 Hz), which makes it distinguishable from normal background sounds such as fan noise or low-frequency disturbances.

In this application, a microphone placed near the baby continuously captures audio signals. These audio signals are processed in real time by the DSP using FFT to convert them from the time domain to the frequency domain. The frequency spectrum is then divided into low, mid, and high frequency bands, and the energy in each band is calculated.

When a baby cries, the mid-frequency and high-frequency energy becomes dominant compared to the low-frequency band. The system detects this dominance and can trigger a visual indication (LED), an alarm, or a notification system. For example, the mid-frequency LED can indicate a normal cry, while high-frequency dominance may indicate distress or discomfort.

This approach enables continuous, automatic, and non-intrusive monitoring of infants, making it useful in homes, hospitals, and childcare centers. With further enhancements such as sound classification and wireless alerts, the system can be extended into a smart baby monitoring solution.

ADVANTAGES

- Enables real-time detection of baby cries
- Automatic and non-intrusive monitoring without human supervision
- Effectively distinguishes baby cries from low-frequency background noise
- Low-cost and simple implementation using FFT and LEDs
- Can be extended to alarms or mobile notifications
- Suitable for homes, hospitals, and childcare centers
- Low power consumption due to efficient DSP processing

DISADVANTAGES

- Cannot accurately distinguish baby cries from other similar high-frequency sounds
- Performance may degrade in noisy environments
- Uses fixed frequency ranges, which may not cover all cry variations
- No sound classification or emotion detection
- Limited accuracy due to absence of machine learning
- Only provides indication, not detailed analysis