CPE3500: Embedded Digital Signal Processing

Lab Exercise 7: Real-Time Voice Transformer with Audio Effect

Objective:

The purpose of this lab exercise is to implement an audio effect to the voice of a speaker in real time on STM32 microcontrollers. The students will be using Nucleo board and audio daughter board together with external a headset/earphone which they can talk and listen the modified audio.

Introduction:

Microphone and speaker are connected to ADC and DAC of the Nucleo board. Audio signal will be processed in real time by introducing an 'Alien Effect' to the voice signal. ADC, DAC and DMA hardware are utilized to capture and reproduce the voice signal and Timer 6 determines the sampling frequency of the ADC and DAC.

In this exercise students will:

- Learn how to process the audio signal in the memory buffer.
- Create an application to capture audio signals from microphone through ADC and outputs the audio through DAC while modifying it with an 'Alien Effect' in real time.
- Learn how to calculate and plot the frequency spectrum of signals.

Required Equipment:

Personal computer (Windows/Mac OS)

Nucleo-L476RG Board (64-pin)

USB Type-A (or Type-C) to Mini-B cable and Jumper Wires

TI Boostxl-Audio Daughter Board

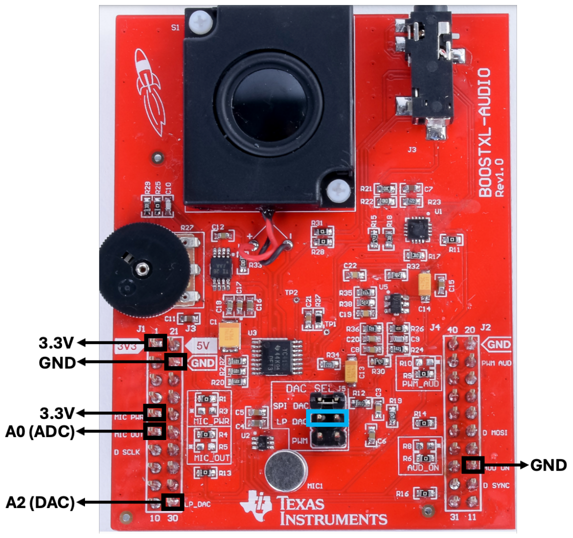
Procedure:

1. Create a new Project in STM32CubeIDE

Create a blank project following the steps that we learned in previous labs.

2. Make Nucleo-Audio Board Connections

Power the audio board (3.3V and GND) through Nucleo board as shown in the connection diagram below. Connect microphone amplifier supply voltage (MIC_PWR) to 3.3V, speaker power amplifier control (AUD_ON) to GND, MIC OUT to Nucleo-A0 and LP_DAC to Nucleo-A2. Finally, switch DAC SEL jumper from SPI DAC to LP DAC as shown below.



TI BOOSTXL-AUDIO Board

3. Configure ADC, DMA, DAC and TIM6 Hardware

Use PA0 port pin on the Nucleo board for the analog input and PA4 (A2 label on the board) for analog output. Configure ADC, DAC, DMA and TIM6 similar with the previous lab.

Set *Prescalar* to 79 and *Counter Period* value to 49 to maintain 20 KHz sampling rate on ADC and DAC.

Make the Data Width as Half Word for both DMAs on ADC and DAC.

Save or generate the code and switch to C perspective to edit the code.

4. Generate Sinusoidal Signal in Matlab

To create an 'Alien Effect', we will use a cosine signal to modulate the audio signal continuously and send the modulated signal out through DAC. We will use 400 Hz cosine signal sampled at 20 kHz.

Use the below line of script to generate a 1-cycle cosine signal at 400 Hz in Matlab.

cosTable=int16(32767*cos(2*pi*[0:49]*400/20000));

Print this array on the command window:

sprintf('%d,', cosTable)

5. Modify the code in STM32CubeIDE

We will create two buffers for ADC and DAC.

Inside /* **USER CODE BEGIN PD** */ section, define the buffer size and half buffer size as 2000 and 1000 respectively. Also define the cosine table length as 50:

#define COS_TABLE_LEN 50

Inside /* **USER CODE BEGIN PV** */ section, create the buffers adc_buffer and dac_buffer as 16-bit integer.

In the same field, create the cosine table. Copy the numbers generated in Matlab and paste it in C code in the example below:

```
static int16_t cos_table[COS_TABLE_LEN] = {
        32767, 32509, 31738, 30466, 28714, 26509, 23886, 20886, 17557, 13952, 10126,
6140, 2057, -2057, -6140, -10126, -13952, -17557, -20886, -23886, -26509, -28714, -30466, -
31738, -32509,-32767,-32509,-31738,-30466,-28714,-26509,-23886,-20886,-17557,-13952,-
10126, -6140, -2057, 2057,6140,10126,13952,17557,20886,23886,26509,28714,30466,31738,
32509
};
```

Inside /* **USER CODE BEGIN 2** */ section, start TIM, ADC and DAC.

Create the callback functions for ADC half and full buffer complete events.

Inside the ADC callback functions, call **processBuffer** function for each half of the buffer to process the data:

```
void HAL_ADC_ConvHalfCpltCallback(ADC_HandleTypeDef *hadc)
{
    processBuffer(&adc_buffer[0], &dac_buffer[0], BUFFER_HALFSIZE);
}
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef *hadc)
{
    processBuffer(&adc_buffer[BUFFER_HALFSIZE], &dac_buffer[BUFFER_HALFSIZE],
BUFFER_HALFSIZE);
}
```

Create the **processBuffer** as below to multiply (modulate) the samples of the audio signal in the buffer with the cosine signal.

```
void processBuffer(uint16_t *inBuffer, uint16_t *outBuffer, uint16_t size)
{
        int32_t x;
        uint8_t gain = 2;
        static uint16_t x_prev = 0;
        uint8_t nFrames = BUFFER_HALFSIZE/COS_TABLE_LEN;

        for(uint8_t fr=0; fr<nFrames; fr++)
        {
                for(uint8_t i=0; i<COS_TABLE_LEN; i++)
                {
                        x = (int32_t)inBuffer[i+fr*COS_TABLE_LEN];
                        x = x - x_prev;
                        x = x * (int32_t)cos_table[i];
                        x = x >> (15-gain);
                        outBuffer[i+fr*COS_TABLE_LEN] = (uint16_t)(x + 2048);
                        x_prev = inBuffer[i+fr*COS_TABLE_LEN];
                }
        }
}
```

This code will modify the captured audio signal with an ‘Alien Effect’ and generate the output audio in real time.

Task-1:

Start debugging the code. Test your code by talking to the microphone and listen from the speaker.

Modify the configuration file to change the sampling frequency to 40 kHz and update the cosine array elements for the new sampling frequency. Test the code and comment on your observations.

Task-2:

For 20 kHz sampling rate, modify the code to capture 1 sec of audio signal. Disable processBuffer and place a breakpoint to the related line inside the callback function. After running the code, say "Hello" to the microphone. After audio recording is completed, export the buffer data and plot it in Matlab.

Repeat the same process while processBuffer function is enabled. Export the modified "Hello" signal and plot it in Matlab.

Task-3:

Use readBinFileAndPlotFFT.mlx file in Matlab to calculate the Fourier Transform of the recorded signals in Task-2 and plot the frequency spectrum.

Comment on the results!

Lab Exercise 7 Report:

Prepare a lab report (as single pdf file) consisting of the followings and upload it to the D2L dropbox.

- Cover Sheet
- Task-1: Insert the screenshot of the modified code pieces and configurations, and comment on the results.
- Task-2: Insert screenshot of the code modified and the Matlab plots for the recorded audio signal with and without the Alien effect. Put proper labels and titles.
- Task-3: Insert the Matlab plots for the frequency spectrums of the two audio signals. Include your comments.
- Conclusion (1-paragraph)