CPE3500: Embedded Digital Signal Processing

Lab Exercise 6: Audio Record-Playback using Ping-Pong Buffer

Objective:

The purpose of this lab exercise is to introduce audio interface (microphone and speaker) for STM32 microcontrollers. The students will be using audio daughter board which houses an on-board microphone with front-end amplifier and a low-profile speaker with volume control slider.

Introduction:

Microphone and speaker are connected to ADC and DAC of the Nucleo board. Audio recording and playback is performed using a ping-pong buffer where half of the input buffer is being processed and copied to output buffer while the other half of the input buffer is being filled with new samples. DMA is utilized to automatically stream the data between the peripheral and memory. Timer 6 determines the sampling frequency of the ADC and DAC.

In this exercise students will:

- Learn how to interface the audio card to Nucleo board.
- Create a simple application to capture audio signals from microphone through ADC and outputs recorded audio through DAC using HAL functions.
- Learn how to control record-playback with pushbutton and external interrupt.

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

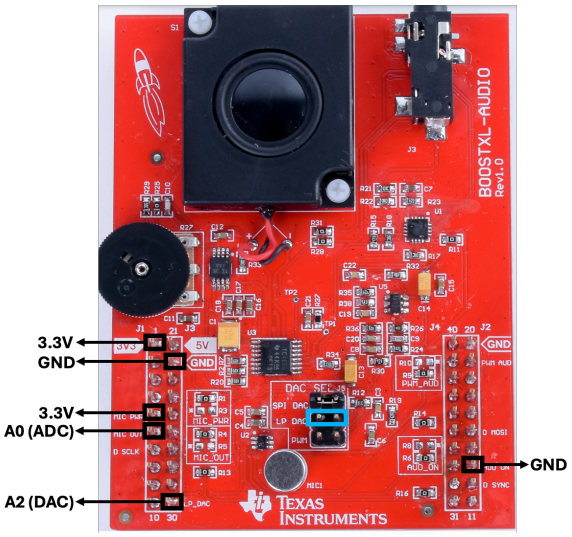
Procedure:

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

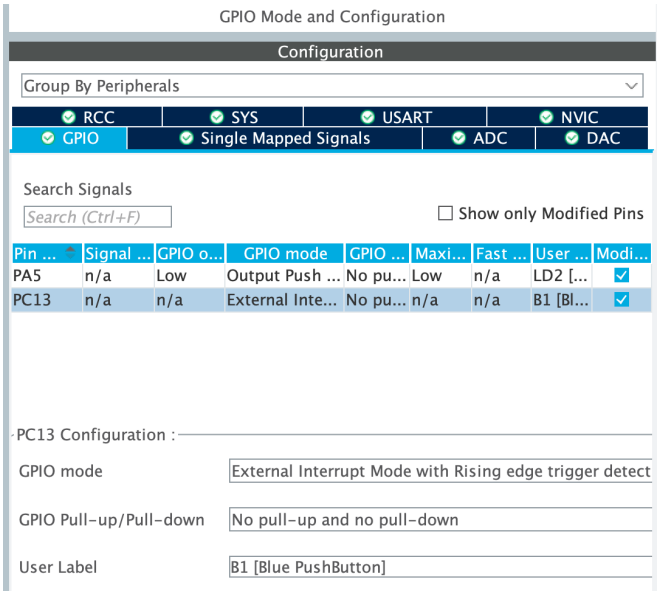
3. Configure ADC, DMA, DAC and TIM6 Hardware

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.

Click on GPIO and modify PC13 (where the blue push button is connected) configuration. Set *GPIO mode* to *External Interrupt Mode with Rising edge trigger detection*.



Click on NVIC tab and enable EXTI Line [15:10] interrupts.

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

We will create two buffers for ADC and DAC.

Inside /* USER CODE BEGIN PD */ section, define the buffer size as 20000 for 1 sec audio recording and playback:

```
#define BUFFER_HALFSIZE 10000
#define BUFFER_SIZE 20000
```

Inside /* USER CODE BEGIN PV */ section, create the buffers:

```
uint16_t adc_buffer[BUFFER_SIZE];
uint16_t dac_buffer[BUFFER_SIZE];
```

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

```
HAL_TIM_Base_Start(&htim6);
```

Inside /* USER CODE BEGIN 4 */, define the GPIO External Interrupt callback function to start ADC with DMA after push button release detection.

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
	HAL_ADC_Start_DMA(&hadc1, adc_buffer, BUFFER_SIZE);
}
```

As described in Lab 5, create two callback functions for ADC half and full buffer complete events.

Addionally, include two more callback functions for DAC half and full buffer complete events as below:

```
void HAL_DAC_ConvHalfCpltCallbackCh1(DAC_HandleTypeDef *hdac)
{
}
void HAL_DAC_ConvCpltCallbackCh1(DAC_HandleTypeDef *hdac)
{
}
```

Inside the ADC conversion complete callback function, stop ADC DMA to stop audio recording and start DAC DMA for playback including the following lines:

```
HAL_ADC_Stop_DMA(&hadc1);
HAL_DAC_Start_DMA(&hdac1, DAC_CHANNEL_1, dac_buffer, BUFFER_SIZE, DAC_ALIGN_12B_R);
```

Once, DAC conversion is fully completed, DAC DMA should be stopped with the following line to end playback:

```
HAL_DAC_Stop_DMA(&hdac1, DAC_CHANNEL_1);
```

This code will record audio from the microphone for 1 sec and immediately play backs through the speaker. Recording-Playback sessions are activated with push button press and release action.

Task-1:

Start debugging the code. Test your code by pressing the pushbutton and saying sample words.

Take a screenshot of the code piece that includes the callback functions. Record a short video which demonstrates audio record/playback is working.

Task-2:

Place a breakpoint to the related line and record audio by saying “Hello” to the microphone. After audio recording is completed, export the buffer data and plot it in Matlab.

Note: Microphone output has a DC value. Therefore, you should subtract the mean value of your signal to display the signal in a more meaningful way.

Task-3:

Modify ADC callback functions to copy the input buffer to output buffer in such a way that the playback will output the backwards!

Test your code by saying random words as well as palindrome words (words that sounds the same backwards).

Take a screenshot of the code piece that includes the callback functions. Record a short video of audio record/playback.

Lab Exercise 6 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 code piece that includes the callback functions.
- Task-2: Insert the Matlab plot for the recorded audio signal with proper labels and title.
- Task-3: Insert the screenshot of the code piece that includes the modified callback functions.
- Conclusion (1-paragraph)

Attach video files in for Task1 and Task3 along with your report!