ECE 4703: Lab #1

Nicholas Chantre, Cristian Gallardo, Brandon Voci

November 3, 2022

Contents

1 Introduction:

In this laboratory exercise, we familiarize ourselves with the MSP-EXP432P401R (MSP) development board, together with the BOOSTXL-AUDIO daughter card. The MSP board includes a 14-bit Analog-to-Digital Converter (ADC), while the BOOSTXL-Audio (BOOSTXL) card includes a 14-bit Digital-to-Analog Converter (DAC), microphone amplifier, and speaker. After establishing a mixed-signal chain containing an ADC to digital processor to DAC path, we generate a basic audio signal on the MSP board and preform analysis, detailed in **Q1, Q2**. Next, in **Q3**, we utilize an external waveform-generator, to generate a test input to our signal chain and evaluate reconstruction performance. Lastly, we implement a simple voice-recording and playback application, with modifications discussed in **Q4**.

2 Methods:

2.1 Question #1, Question #2:

In the first part of this exercise, we characterize a basic audio signal on the MSP board and BOOSTXL card. Namely, an analog generated saw-tooth waveform with uniform period, digitally processed and mapped to the DAC peripheral. To do this, we execute the lab_sawtooth C program on the MSP board, via the Code-Composer Studio (CCS) development environment. To measure the output of the DAC in real-time and perform basic analysis, we utilize the Analog Discovery 2 digital USB oscilloscope, monitoring the J1.2 pin on the BOOSTXL card. We hence verfiy the shape of the waveform and determine the amplitude, offset, and period. Additionally, we map DAC arguments in C to voltage levels and verify the software-defined sampling rate for our processing system.

2.2 Question #3:

In addition to providing oscilloscope functionality, the Analog Discovery 2 further serves as a waveform generator. We thus use this device to generate a 1 KHz sinusoid with 1V amplitude and offset of 1.65V, as a basic test input for our signal chain. To evaluate the reconstruction performance, we concurrently utilize the oscilloscope functionality, continuing to monitor the J1.2 pin on the BOOSTXL card. We can thus view distortions in the output signal and tune the amplitude and frequency of our test input accordingly.

2.3 Question #4:

In the last part of this exercise, we execute the C program lab1_voicerecorder, once again via CCS. The program uses one of the push-buttons on the MSP board to record audio samples via the microphone and then play them back using a different push button. In our modified program, newly recorded sounds are appended to the end of a buffer, such that audio samples are stored together. To do this we initialize the variable:

```
uint16_t x0 = xlaudio_adc14_to_q15((((int16_t) buffer[bufindex]) << 6)
```

and then add the following into the buffer:

```
buffer[bufindex] = xlaudio_adc14_to_q15(0.75*(x + x0)) >> 6;
```

Another modification made involves playback speed. In particular, in one of the playback modes, the audio sample is played back at twice speed: we wish to playback at half-speed. To do this, we observe that the mechanism for controlling the payback speed is the rate at which the buffer index indexes through the audio sample buffer. Indeed, to play audio at twice speed, we have that our buffer index twice iterates at each sample. Therefore, to play the sample at half-speed, we add the following index divider, prolonging each sample by one index:

```
bufindex2++;
if(bufindex2 == 2){
    bufindex2 = 0;
    bufindex++;
}
```

Note that the second buffer index is initialized at 0 and requires two increments before allowing the primary buffer index to increment, hence giving the effect of half-speed playback.

3 Results:

3.1 Question #1, Question #2:

Q1: After running the sawtooth program, we hear a low-pitch monotonic buzz emitted from the speaker on the BOOSTXL board. The main.c associated with this program uses interrupts to successively increment a level variable, represented as an integral data type, from 0 to a maximum value of 99, before repeating. Thus, the value of the level variable corresponds to the height of the resulting sound wave.

Q2a: As expected, the shape of our waveform is a periodic saw-tooth. Figure 1 shows the resulting digital oscilloscope plot, on the DAC output:

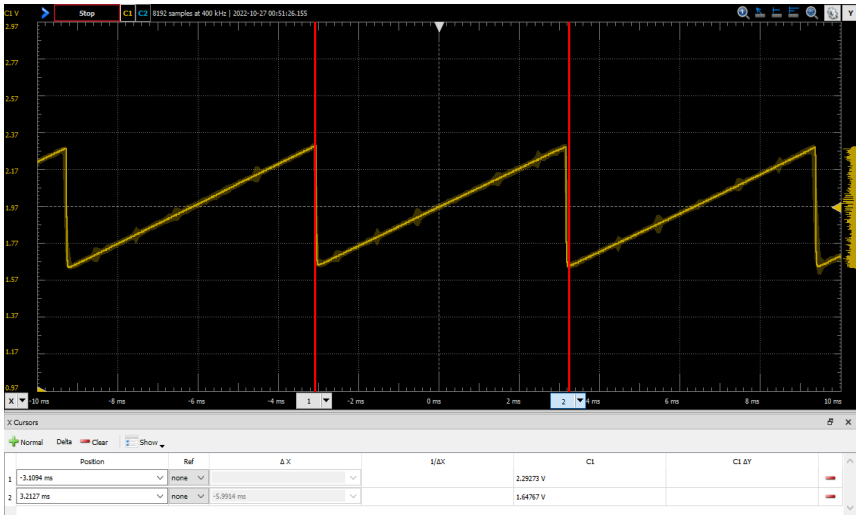

Figure 1: Sawtooth wave observed on DAC output, measured using digital oscilloscope.

From Figure 1, we can further determine the amplitude, offset and period. To determine

the amplitude, let $s(t)$ denote the time-domain representation of our waveform, with maximum value $s_{\max}$ and minimum value $s_{\min}$. The peak-to-peak amplitude, A_{pp} is therefore:

$$A_{pp} = s_{\max} - s_{\min} = 2.29273\text{V} - 1.64767\text{V} = 0.64506\text{V} \approx 0.65\text{V}$$

and the amplitude A is:

$$A = \frac{1}{2}A_{pp} = 0.32253 \approx 0.325\text{V}$$

By inspection of Figure 1, we find that the offset is 1.65V and the period is:

$$T = 3.21270 \text{ ms} - (-3.10940 \text{ ms}) = 6.32210 \text{ ms} = 0.00632 \text{ s}$$

Note that the associated frequency is $f = 1/T = 158.1759$ Hz ≈ 158 Hz.

Q2b: The lowest value of the argument xlaudio_f32_to_dac14 is the 32-bit floating point value $0 \times 0.001 = 0.0$ while the highest value of the argument is $0.001 \times 99 = 0.099$. We thus require a linear mapping between $V \in [1.65V, 2.30V]$ and $F \in [0.0, 0.099]$. Equation (1) captures this mapping:

$$V = \frac{2.30V - 1.65V}{0.099}F + 1.65V = \frac{0.65V}{0.099}F + 1.65V \tag{1}$$

From (1), we generate the following table, describing how arguments of xlaudio_f32_to_dac14 map to voltage measurements:

xlaudio_f32_to_dac14 Argument:	Voltage Measurement:
0.000	$1.650V$
0.025	$1.814V$
0.050	$1.978V$
0.075	$2.140V$
0.099	$2.300V$

Q2c: Based on the C program, 100 points are used to construct the waveform since the level variable increments from 0 to 99. Let $F_s = 16$ KHz denote the sampling frequency. Then the sampling period is $T_s := 1/F_s = 1/16$ KHz and the waveform period, T, is given by:

$$T = 100T_s = \frac{100}{F_s} = \frac{1}{160\text{Hz}} = 0.00625\text{s} = 6.25\text{ms}$$

Indeed, 6.3221 ms $\approx$ 6.25 ms so the measured waveform period matches the period calculated from the sampling rate.

3.2 Question #3:

Q3a: At an approximately 700 mV amplitude input, the sine function appears distortionless. Since the offset of the input signal is 1.65V this gives an amplitude of 700 mV + 1.65 V = 2.35V so it is likely that this voltage value exceeds the full scale range (FSR) for the A/D converter, hence causing the distortions.

Q3b: As seen in the Figures 2-4 below, the Sawtooth wave's distortion begins to increase and then decrease as the sampling frequency increases. In Figures 2 and 3, the oscillation is very fast amongst sampling points. When the frequency reaches 15 KHz, however, we

can see that our result is a visible sine wave. There are a few rough edges throughout the waveform. The sine wave will eventually be extremely smooth once the sampling rate is increased sufficiently. Because of this, it is easy to distinguish waveforms from 1 KHz to 15 KHz. However, once the sampling rate becomes too high, it may begin to miss samples.

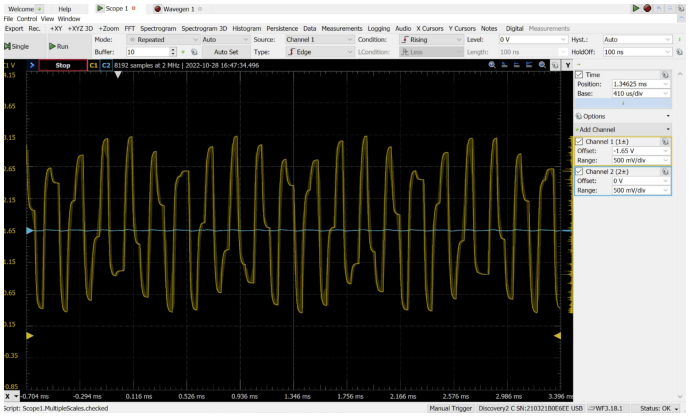

Figure 2: Sawtooth wave at 5 KHz frequency.

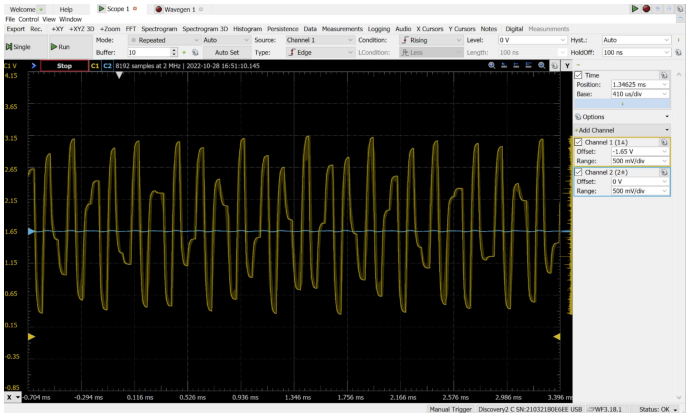

Figure 3: Sawtooth wave at 10 KHz frequency.

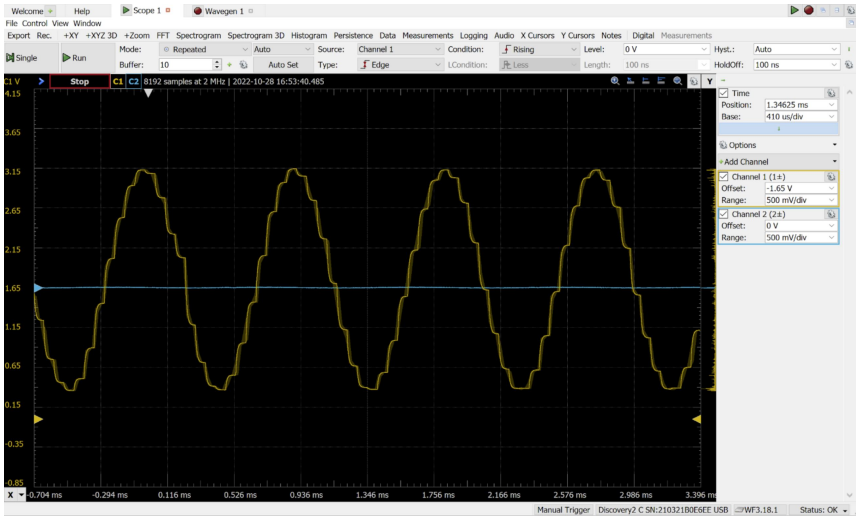

Figure 4: Sawtooth wave at 15 KHz frequency.

3.3 Question #4:

Q4a: Having implemented our described modifications to the main.c file for 'lab1voicerecorder' , we recorded our first voice sample and played the sample back to then hear the other modes play the recording at a reduced noise. For each recording sample that followed and added itself onto our buffer, we could clearly hear the overlap of audios(new and prior). As more samples were added, the amplitude noticeably decreased for our signal as it gradually faded.

Q4b: For part b of question 4 we edited the existing code to have PLAYBACK2 play at 1/2 speed instead of at double. To accomplish this, we made use of a simple counter that we would use to control when we incremented the buffer index(bufindex). In inspecting the provided code for PLAYBACK2, we discovered that the code was iterating over every 2 indexes through our buffer(ex: 1, 3, 5 ...). In order to configure PLAYBACK2 to play at 1/2 speed we wanted each iteration of audio played to play the same sample of audio data twice(ex: 1, 1, 2, 2 ...). Through incrementing the counter every iteration, we were able to keep track of when the same buffer index value had been used twice and then increment our buffer index while also resetting our counter to repeat the same behavior. The result of these changes was a stretched and slowed audio sample.

4 Conclusions:

In this lab we were able to gain exposure to the MSP-EXP342 micro controller and experiment with applications that provided insight into its data storage ability of audio signals and their manipulation using bit shifting operations as well as basic arithmetic. Through using the ADC/DAC functionality of our micro controller we gained exposure and an understanding to how analog inputs are converted digitally and how digital representations are converted into analog values. Additionally, we were able to interact with the peripherals of

this micro controller and begin to explore the wide range of signal processing functionalities that our AUDIO BOOST XL Kit offers.