Som Wakdikar - sw38937
Kevin Tong - kyt259
17710

Lab Report

Objectives:

Overview: The objectives of this project are to design, build and test a music player. Educationally, students are learning how to interface a DAC, how to create a speaker amplifier, how to store digital music in ROM, and how to perform DAC output in the background. Your goal is to play your favorite song.

Process: The project will be developed using the TM4C123 board. There will be two or three switches that the operator will use to control the music player. The system will be built on a solderless breadboard and run on the usual USB power. The system may use onboard switches or off-board switches. A hardware/software interface will be designed that allows the software to control the player. There will be at least three hardware/software modules: switch input, DAC output, and the music player. The process will be to design and test each module independently from the other modules. After each module is tested, the system will be built and tested.

Roles and Responsibilities: EE445L students are the engineers and the TA is the client. Students are expected to make minor modifications to this document in order to clarify precisely what they plan to build. Students are allowed to divide the responsibilities of the project however they wish, but, at the time of the demonstration, both students are expected to understand all aspects of the design.

Interactions with Existing Systems: The system will use the TM4C123 board, a solderless breadboard, and the speaker as shown in Figure 5.1. It will be powered using a USB cable. You may use a +5V power from the lab bench, but please do not power the TPA731/MC34119 or the speaker with a voltage above +5V. The particular system created and tested for this lab is powered by 3.3 V from the TM4C123 board.

Terminology: Definitions for the terms SSI, linearity, frequency response, loudness, pitch, instrument, tempo, envelope, melody, and harmony can be found in the textbook.

Security: The system may include software from TivaWare and from the book. No software written for this project may be transmitted, viewed, or communicated with any other EE445L student past, present, or future (other than the lab partner of course). It is the responsibility of the team to keep its EE445L lab solutions secure.

Functionality: If the operator presses the play/pause button the music will play or pause. If the operator presses the play/pause button once the music should pause. Hitting the play/pause again causes the music to continue. The play/pause button does not restart from the beginning, rather it continues from the position it was paused. If the rewind button is pressed, the music stops, and the next play operation will start from the beginning. There is a mode switch that allows the operator to control some aspects of the player. This switch switches songs from one to the other.

There must be a C data structure to hold the music. There must be a music driver that plays songs. The length of the song should be at least 30 seconds and comprise at least 8 different frequencies. Although you will be playing only one song, the song data itself will be stored in a separate place and be easy to change. The player runs in the background using interrupts. The foreground (main) initializes the player, then executes while(1){} do nothing loop. If you wish to include LCD output, this output should occur in the foreground. The maximum time to execute one instance of the ISR is 3.417 us. The maximum sampling jitter is 35567. You will need public functions Rewind, Play, and Stop, which perform operations like a cassette tape player. The Play function has an input parameter that defines the song to play. A background thread implemented with output compare will fetch data out of your music structure and send them to the DAC. There are two additional aspects added to the notes of the songs. The first aspect includes an envelope, which crescendos down each note. The second aspect is creating harmony with another wave before outputting to the DAC.

Scope: Phase 1 is the preparation; phase 2 is the demonstration, and phase 3 is the lab report. Details can be found in the lab manual.

Prototypes: A prototype system running on the TM4C123 board and solderless breadboard will be demonstrated. Progress will be judged by the preparation, demonstration, and lab report.

Performance: The system will be judged by three qualitative measures. First, the software modules must be easy to understand and well-organized. Second, the system must employ abstract data structures to hold the sound and the music. There should be a clear and obvious translation from sheet music to the data structure. Backward jumps in the ISR are not allowed. Waiting for SSI output to complete is an acceptable backward jump. Third, all software will be judged according to style guidelines. Software must follow the style described in Section 3.3 of the book (note to students: you may edit this sentence to define a different style format). There are four quantitative measures. First, the SNR of the DAC output of a sine wave should be measured. Second, the maximum time to run one instance of the ISR will be recorded. Third, you will measure the maximum jitter of the DAC outputs. Fourth, you will measure the power supply current to run the system. There is no particular need to optimize any of these quantitative measures in this system.

Usability: There will be three switch inputs. The DAC will be interfaced with a 32-ohm speaker.

Safety: If you are using headphones, please verify the sound is not too loud before placing the phones next to your ears.

Reports: A lab report described below is due by the due date listed in the syllabus. This report includes the final requirements document.

Audits: The preparation is due at the beginning of the lab period on the date listed in the syllabus.

Outcomes: There are three deliverables: preparation, demonstration, and report.

Hardware Design:

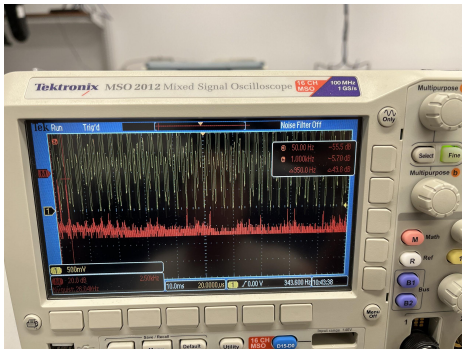
Uploaded on GitHub.

Software Design:

Uploaded on Github.

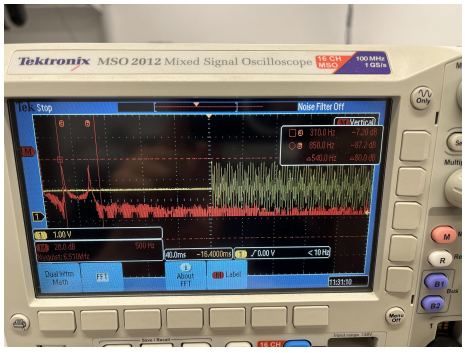
Deliverables:

Deliverable 1:
Resolution: 3.0/(2^12) = 7.324 x 10^-4
Range: 3.0 V
Precision: 2^12 = 4096
Accuracy: 0.97 (varies due to noise on the reference voltage and output DAC voltage)
Time domain and frequency domain outputs at 1kHz.



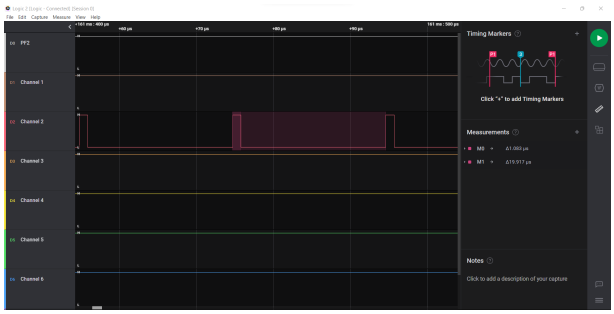
SNR: -5.70 + 55.5 = 49.8 dB
ENOR = 7.98 = 8 bits

Deliverable 2:



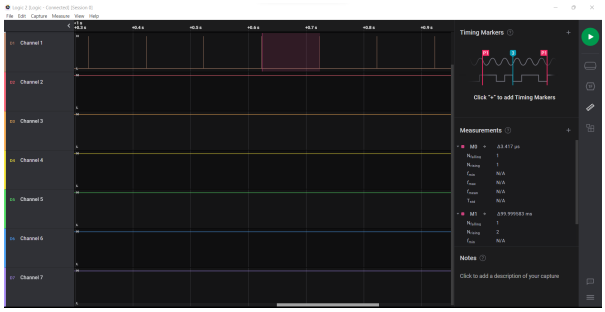
SNR = -7.20 - 87.20 = 80.0 dB
ENOB =12.99 = 13 bits

Max time to execute Timer1A ISR: 1.083 us
Period for Timer1A: 20 us
Percent time in ISR: 1.083 us / 20 us = 0.05415 * 100 = 5.415 %
Timer1A ISR Jitter: 2



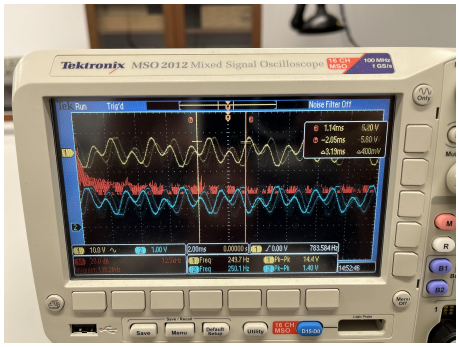
Deliverable 3:

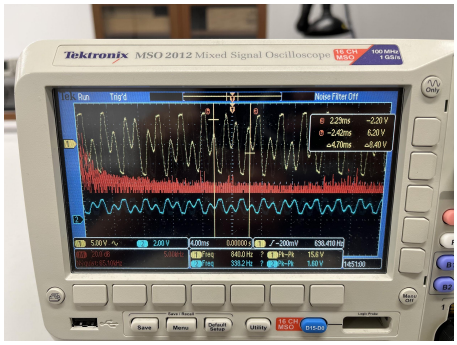
Max time for Timer0A ISR execution: 3.417 us
Timer0A Period: 1 second
Percent time in ISR: 3.417 us / 1 s = 3.417 * 10^-6 * 100 = 0.0003417%



Deliverable 4:
Jitter for Timer0A ISR: 35567
The reason for this jitter is other ISRs such as edge triggered for buttons or the TIMER1A ISR interrupting it.

Deliverable 5:
Pictures of dual channel outputs (pins 5 and 8):





5V line without music playing: 5.00 V
5V line with music playing: 5.04 V

Deliverable 6:
Current without sound: 0.083 A

Current with sound: 0.151 A
(noise measurements in Deliverable 5)

Analysis:

Briefly describe three errors in a DAC.
Gain- error: Shift in slope of Vout vs the digital input static response.
Non-monotonicity: Worse case of non-linearity, one where
Non-linearity: The difference between one value corresponding to one digital input vs each consequent digital input is different.

Calculate the data available and data required intervals in the SSI/DAC interface. Use these calculations to justify your choice of SSI frequency.
We chose the SSI frequency of 10 MHz. The setup time for the DAC is 8ns and the hold time is 5ns. However, we used the datasheet to calculate the SSI frequency. It can be seen that from the datasheet, up to a 20 MHz clock speed can be used. This is 50 ns, which satisfies the setup and hold time requirements. However, we used the 10 MHz clock speed as this is 100 ns, which has a higher margin for satisfying setup and hold time.

Why did you use Freescale mode 1 and not mode 0 (bits 7,6 of SSI1_CR0_R)?
This was done to set the SPO and SPH numbers for the clock to the correct values.

How is the frequency range of a spectrum analyzer determined?
The frequency range is determined by the local oscillator frequency range and the Intermediate Frequency filter's central frequency. The local oscillator is an oscillator used with a mixer to change the frequency of a signal. In a spectrum analyzer, the input signal and local oscillator signal are combined in the mixer which is then filtered by the IF filter's center frequency and displayed to produce the frequency range..

Notice the audio amplifier had a voltage gain of 1. Why did we not simply drive the speaker directly from the DAC? In other words, what purpose is the TPA731/MC34119?
The DAC only outputs the voltage and very little current. Thus, the speakers do not receive enough power to actually hear from the DAC. An audio amplifier is used to drive the speaker and is done using OP-AMPs to maximize voltage swings at low voltages. This is done using the feedback resistor.