EE445L - Lab 05 Report

Kevin Gilbert
Gilberto Rodriguez

Professor Bard
Lab: Monday/Wednesday 5-6:15

March 3 2014

Requirements Document

Overview

Objectives

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 design 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 LM3S1968 board. There will be 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 the on board switches or off-board switches. A hardware/software interface will be designed that allows 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 exactly what they plan to build. Students are allowed to divide responsibilities of the project however they wish, but, at the time of demonstration, both students are expected to understand all aspects of the design.

Interactions with Existing Systems

The system will use the LM3S1968 board, a solderless breadboard, and the speaker. It will be powered using the USB cable. You may use a +5V power from the lab bench, but please do not power the speaker with a voltage above +5V.

Terminology

1. SSI: Synchronous Serial Interface. Communication standard using a shared clock signal, a data signal, and slave select signals.
2. Linearity: Used in reference to the audio amplifier, linearity determines how the amplifier's output matches to its input. If discrete increments of the input lead to discrete increments in the output of an equal rate of growth at a larger value, then the amplifier has a linear function.
3. Frequency Response: Measure of the audio amplifier's range of inputs to output spectrum.

4. Loudness: How loud a tone is. Determined by the amplitude of the output wave, which is in turn determined by the digital value sent to the DAC (larger values provide a higher output voltage).
5. Pitch: Frequency of a wave.
6. Instrument: Type of sound being played in this lab. Can be adjusted by outputing a non-sine wave. Controlling the voltage over time plot of the wave can create new instrumental sounds.
7. Tempo: The speed at which the song is played. Controlled by how often notes are changed.
8. Envelope: The exponential drop in amplitude of the sound waves over time to provide smoother signals.
9. Melody: The primary sequence of notes to form a song's core.
10. Harmony: Accompanying sequence of notes that form a backdrop for a melody.

Security

The system may include software from StellarisWare 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.

Function Description

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 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 Lab 5 Music Player and Audio Amp switch that allows the operator to control the volume of the music player. 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 of at least 8 different sounds. 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 for(;;){} do nothing loop. If you wish to include OLED output, this output should occur in the foreground. The maximum time to execute one instance of the ISR is 1.798 μs. 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 must be a C data structure to store the sound waveform, or instrument. You are free to design your own format, as long as it uses a formal data structure (i.e., struct). The generated music must sound beautiful utilizing the SNR of the DAC. Although you only have to implement one instrument, it should be easy to change instruments.

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 LM3S1968 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 an 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 backwards jump. Third, all software will be judged according to style guidelines. Software must follow the style described in Section 3.3 of the book. There are three 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 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 to a 8-ohm speaker.

Safety

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

Deliverables

Reports

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

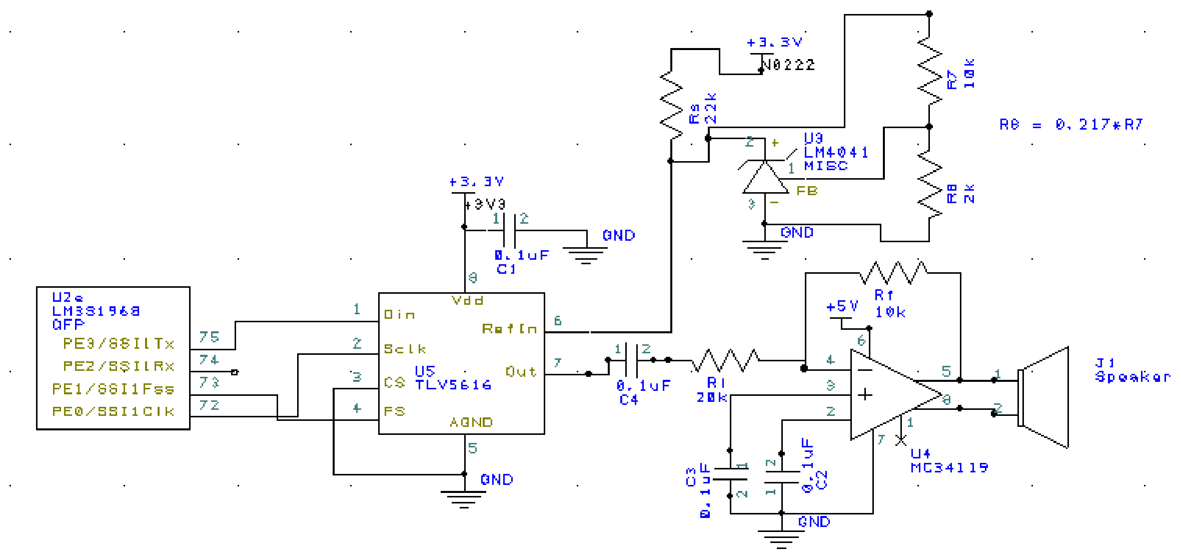
Audits

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

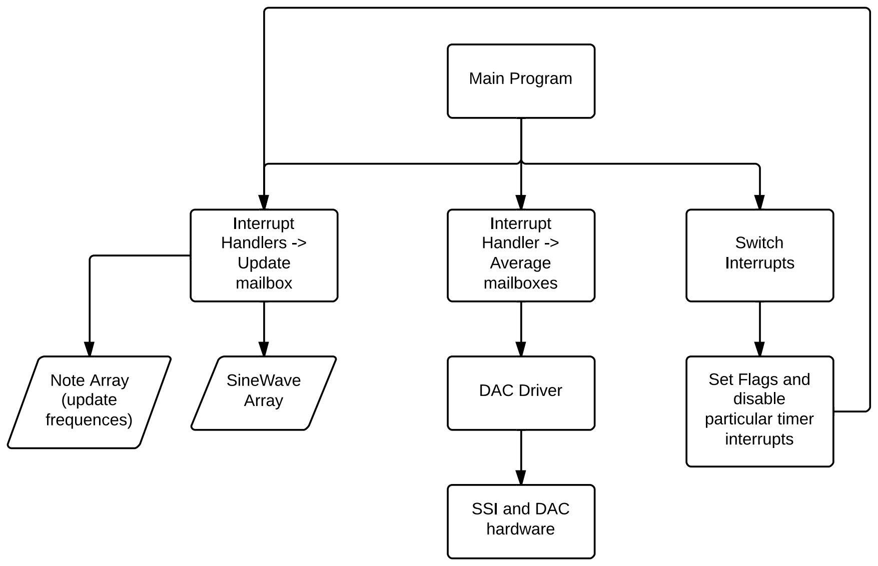


Software Design

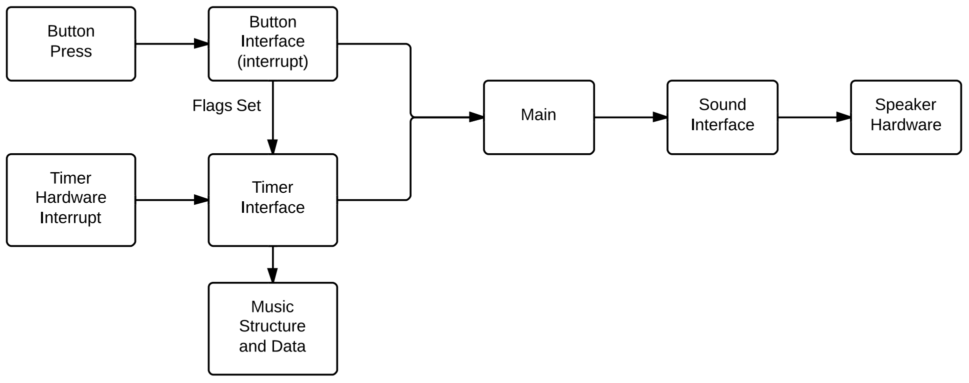
Music Data Structure

tNote is the music data strucutre. An array of tNotes contains a song, with each element containing both the frequency of the note, and the duration that the note lasts.

Call Graph


Main Program
Interrupt Handlers -> Update mailbox
Interrupt Handler -> Average mailboxes
Switch Interrupts
Note Array (update frequences)
SineWave Array
DAC Driver
Set Flags and disable particular timer interrupts
SSI and DAC hardware

Data Flow Graph



Measurement Data

DAC percision: 4096
DAC range: 0 V to 3 V
DAC resolution: 366 μV
DAC accuracy: 2%
Current without music: .102A
Current with music: .145A

ISR Timer0A (sine wave 1): 1.798 μs
ISR Timer0B (frequency update/note select): 679 ns
ISR Timer1A (sine wave 2): 1.218 μs
ISR Timer1B (average sign waves to DAC_ Out): 938 ns
The ISR lengths can be seen in detail throug the logic analyser results at the end of this document.

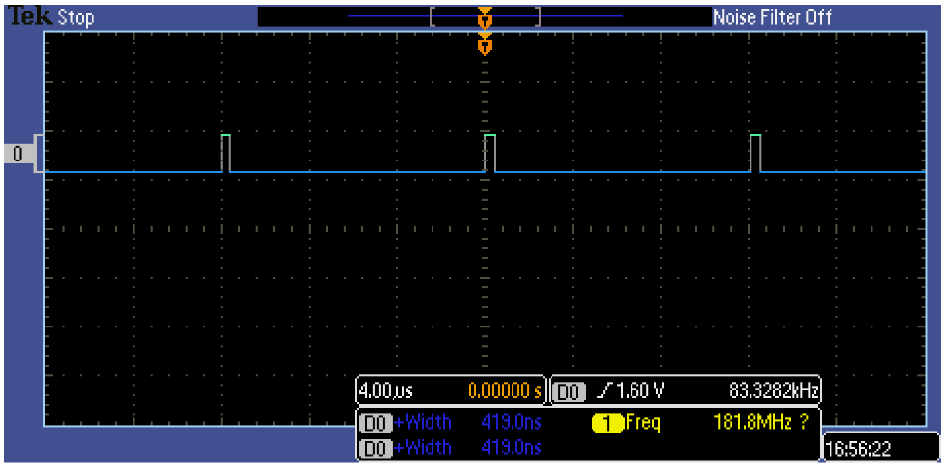
Analysis and Discussion

Problem 1. Briefly describe three errors in a DAC.

Gain error is a shift in the slope of the V_{out} versus digital input static response. Offset error is a shift in the V_{out} versus digital input static response. A linearity error is formed when the growth of the functions becomes nonmonotonic and begins decrease.

Problem 2. Calculate the data available and data required intervals in the SSI/DAC interface. Use these calculations to justify your choice of SSI frequency.

SSI wait interval: 419ns. Taking a midrange frequency of 440Hz, this would give us $419E-9*440 = 18.436E-5$*. This is the minimum rate at which the SSI must be sending data. Our SSI clock is set to 3MHz, which is well above this value.*



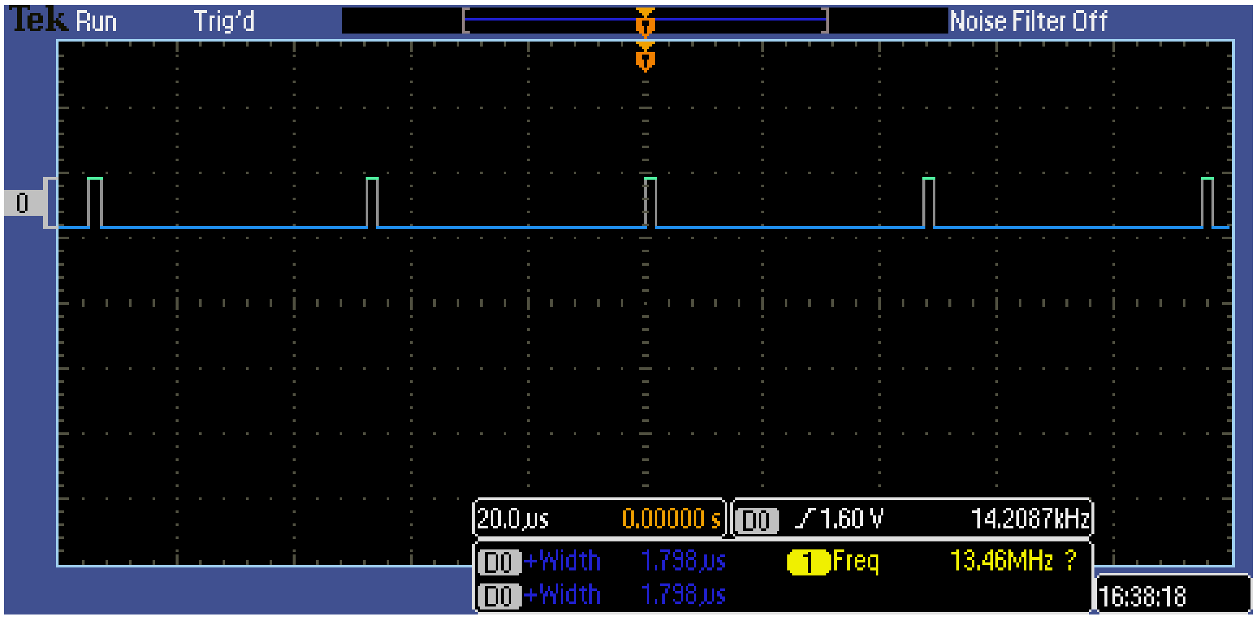
Problem 3. How is the frequency range of a spectrum analyzer determined?

The spectrum analyer uses a fast Fourier transform (FFT) to compute the discrete Fourier transform (DFT), which transforms a waveform into the components of its frequency spectrum and input signal.

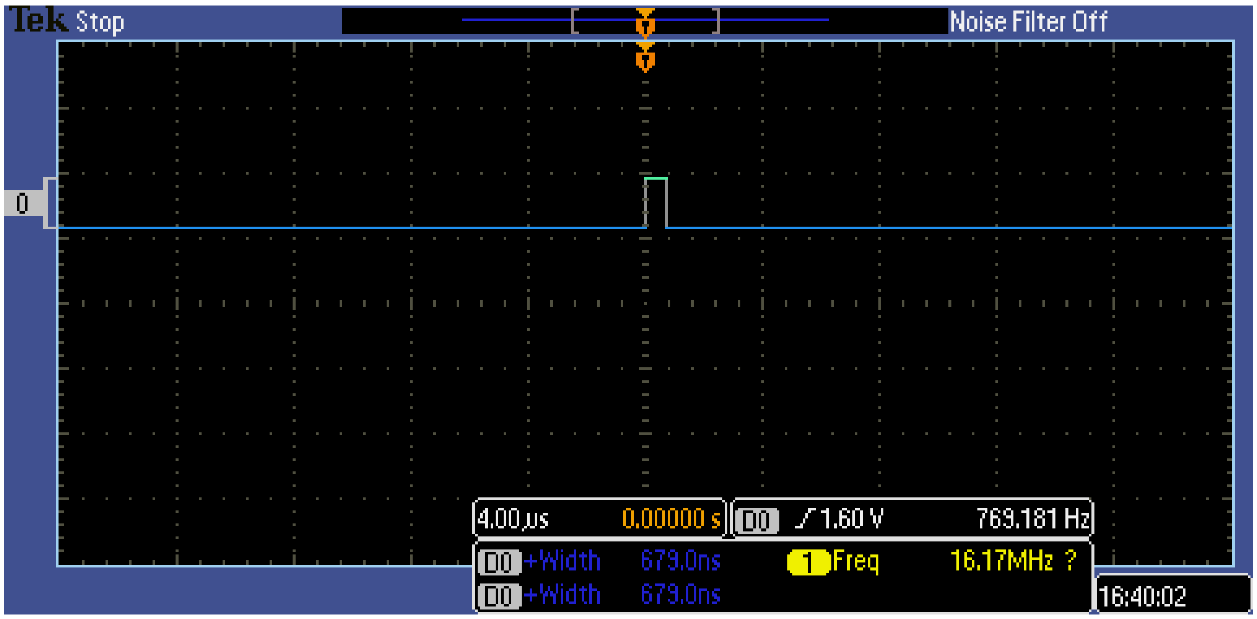
Problem 4. Why did we not simply drive the speaker directly from the DAC? I.e., what purpose is the MC34119?

We cannot drive the speaker directly from the DAC because the DAC does not provide enough current to power the speakers.

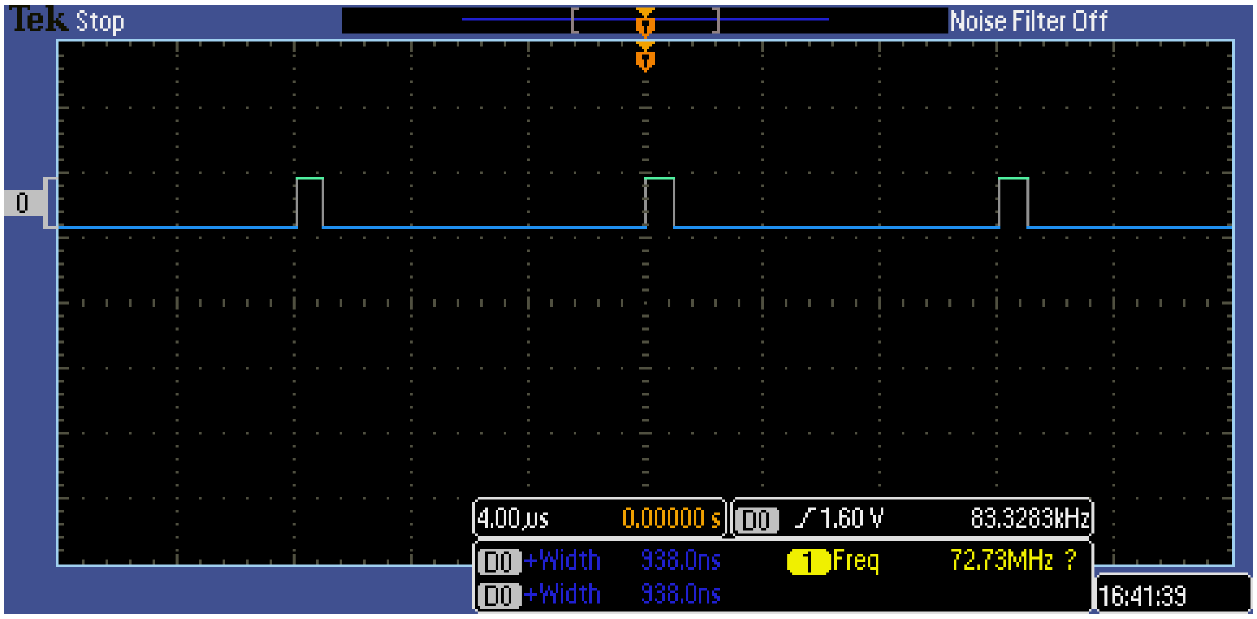
Logic Analyser Images



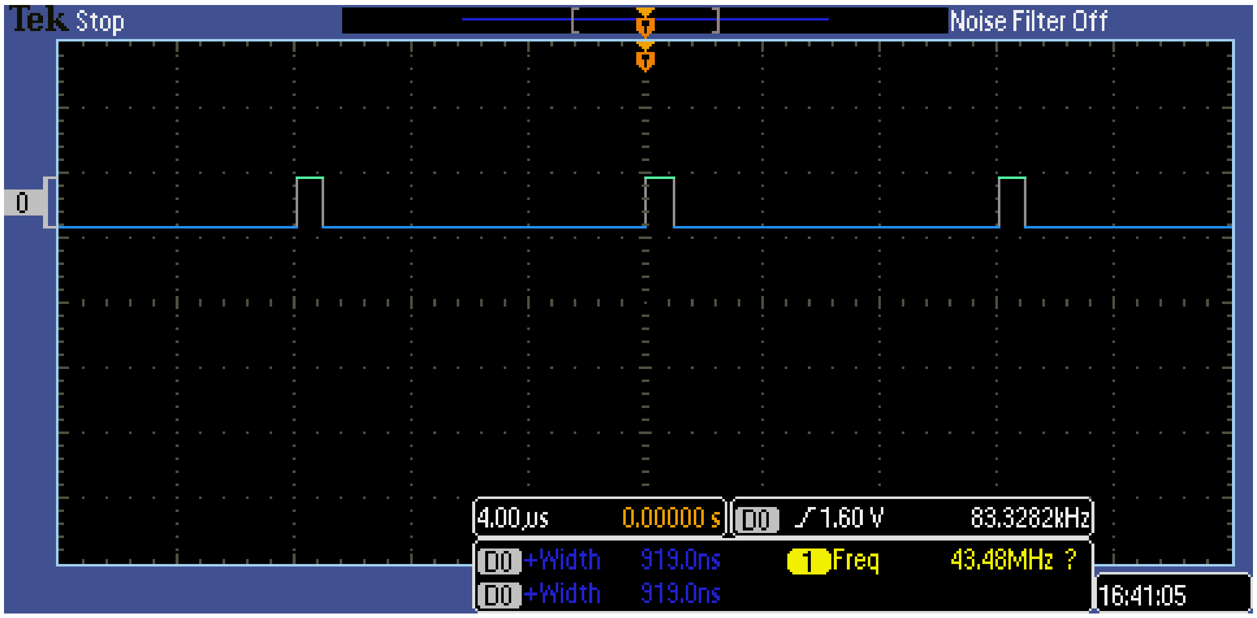
Timer0A_ISR



Timer0B_ISR



Timer1A_ISR



Timer1B_ISR