**LAB 5 REPORT**

**Austin Eberle and Alexander Morgan**

1. **Requirements Document**

1. Overview

1.1. Objectives: Why are we doing this project? What is the purpose?

The objectives of this project are to design, build and test a music player. We will interface a DAC, design a speaker amplifier, store digital music in ROM, and perform DAC output in the background. We will have an array of songs for the user to choose.

1.2. Process: How will the project be developed?

The project will be developed using the TM4C123 or TM4C1294 board. There will be four 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 will use 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.

1.3. Roles and Responsibilities: Who will do what? Who are the clients?

Xander will provide the updated starter code from lab 3 for use in adapting lab 5. Austin will create the initial music structure within memory. Xander will initialize all edge triggered and timer interrupts while Austin provides changes ti suit specific note outputs. Both students will create individual songs within ROM and append them to the song array.

1.4. Interactions with Existing Systems: How will it fit in?

The system will use the TM4C123 or TM4C1294 board, a solderless breadboard, and the speaker as shown in Figure 5.1. 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.

1.5. Terminology: Define terms used in the document.

**SSI** – Synchronous serial interface standard for transferring data between master and slave with a shared clock.

**Linearity** – Error when dealing with step function of digital output versus linear sample of analogue data.

**Frequency response** – Measure of the output spectrum in response to a stimulus.

**Loudness** – Amplitude of a given waveform.

**Pitch** – The frequency at which a wave propagates.

**Instrument** – The signature of a given waveform.

**Tempo** – The amount of time allotted per measure of music. Also known as the beat.

**Envelope** – Slow decay of a waveform as the initial pulse wears off.

**Melody and harmony** – Properties of music that correlate to sounds and tempo coming together.

1.6. Security: How will intellectual property be managed?

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.

2. Function Description

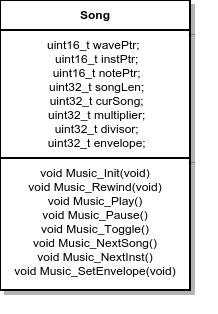
2.1. Functionality: What will the system do precisely?

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 switch that allows the operator to control what instruments is being output by the player. In addition, there is a final button that advances the song currently being played to the next song in the array.

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 436 ms using the standard DAC\_Out function call*.* 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.

C Data Structure Implemented



2.2. Scope: List the phases and what will be delivered in each phase.

Phase 1:

A schematic of the final system will be submitted including all resistors, capacitors, and complex components. In addition, initial module implementation will be listed at this stage.

Phase 2:

We will demonstrate the lab by playing our song in front of the teaching assistant. We will show switch function and its intended result of advancing, pausing, rewinding, or playing a song. If special features such as an envelope or chords are implemented we will add these to the demonstration.

Phase 3:

We will submit a lab report complete with execution measurements and final schematic designs.

2.3. Prototypes: How will intermediate progress be demonstrated?

A prototype system running on the TM4C123 or TM4C1294 board and solderless breadboard will be demonstrated. Progress will be judged by the preparation, demonstration and lab report.

2.4. Performance: Define the measures and describe how they will be determined.

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.

2.5. Usability: Describe the interfaces. Be quantitative if possible.

There will be four switch inputs. The DAC will be interfaced to a 32-ohm speaker. There will be an LCD interface, but this scope of the lab is reserved for reduction based on timing restrictions.

2.6. Safety: Explain any safety requirements and how they will be measured.

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

3. Deliverables

3.1. Reports: How will the system be described?

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

3.2. Audits: How will the clients evaluate progress?

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

3.3. Outcomes: What are the deliverables? How do we know when it is done?

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

**2.0 Measurement Data**

1. DAC Output for Given Digital Input

|  |  |  |
| --- | --- | --- |
| Digital Input | DAC Output Actual | DAC Output Expected |
| 1068 | 1.16 | 1.56 |
| 1737 | 2.03 | 2.54 |
| 1863 | 2.76 | 2.72 |
| 1031 | 1.53 | 1.51 |
| 309 | .46 | .45 |
| 472 | .71 | .69 |
| 1038 | 1.34 | 1.52 |
| 1476 | 2.10 | 2.16 |

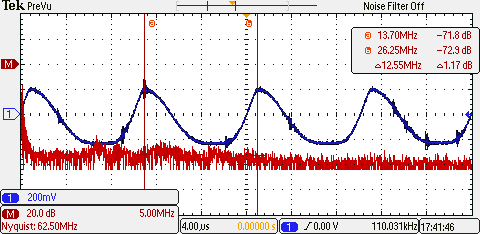
Resolution – 3.2 mV

Range – The range of the 12 bit DAC is typically 0 – 3.3V.

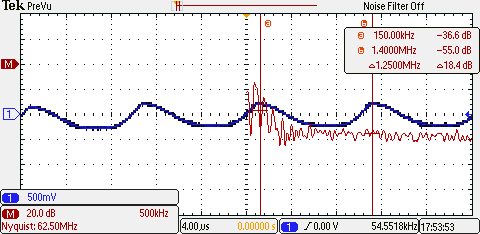
Precision – 12 bits of precision, but we used a shortened version of the sine wave in order to account for the poor signal to noise ratio.

Accuracy – Our average accuracy for the DAC output despite some slightly bad readings was ~92%.

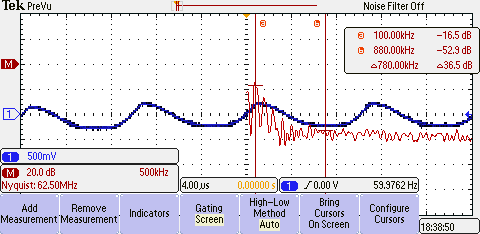
2.0 Time Domain and Frequency Domain Graph Output



Fixed Scale of FFT (Limited O-scope FFT range between 0 and 500 kHz)

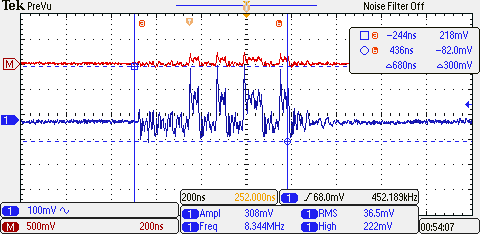


Signal to Noise Ratio



Rate of Interrupt Service Routine

The ISR within our program originally output directly to the DAC without waiting for a ready signal. However, in order to get an ISR reading of a realistic system we measured the ISR by calling the provided DAC\_Out function. By toggling a bit at the beginning and end of the ISR we can see a time length of about 436 ns.



Power Used By the System

When music is not playing the system idles at 83 mA. However, when music is playing and power is delivered to the speaker the system uses 124 mA.

1. **Analysis and Discussion**
2. Briefly describe three errors in a DAC.

**Linearity Error** – Natural analog output forms a continuous curve as levels rise and fall. Unfortunately even at the lowest step we see that our DAC output is a step function as voltages rise and fall. This produces error in the curve created.

**SNR error** – Our DAC is subject to outside oscillations and incident waves. As a result we can clearly see a ratio between the signal created and the noise surrounding the output.

**Offset Error** – The DAC has a secified range of outputs at its disposal, but even when the input is set at zero we can see that a small offset is present at the system’s output. For an ideal input of zero we receive a small output error.

**Gain Error** – The opposite end of the output spectrum shows a small error when a full scale input is applied to the DAC. This error is very dependent upon the systems reference voltage. Our shunt regulator helps to minimize this error.

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

According to the data sheet found at <http://www.ti.com/lit/ds/symlink/tlv5616.pdf>, the equation for choosing the frequency of the SSI/DAC interface is:

f{SCLKmax} = 1 / (t{wH(min)} + t{wL(min)} ) = 20 MHzWith this upper bound in mind we chose a frequency of 8 MHz based on the clock divider.

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

There are several ways of ascertaining the frequency range of a spectrum analyzer. The first deals with the instruments IF filter while the second is based on the fast fourier transform and its sampling properties. If a mixer is present we can look at the characteristics of an IF filter if one is present. The measurement frequency range is determined by the center frequency of the IF filter and the local oscillator. An FFT analyzer computes the discrete fourier transform in order to obtain the frequencies present within a waveform. The waveform must be sampled at twice the frequency as per the Nyquist Theory.

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

The TM4C123 does not provide enough current from an I/O port for driving a large component such as the speaker. To drive the speaker for this lab we use the VCC line out from the board connection and amplify the signal further just before the speaker is reached.