EE445L – Lab 5: Music Player and Audio Amp

Michael Park and Jack Zhao

03/07/16

1.0 OBJECTIVE

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. 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. Our goal is to play Phantom of the Opera.

1.2. Process: How will the project be developed?

The project will be developed using the TM4C123 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 will use three 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?

EE445L students are the engineers and the TA is the client. Michael will design and implement hardware. He will also write basic software modules to test the hardware. Jack will deisgn and implement the software application.

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

The system will use the TM4C123 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.

1.5. Terminology: Define terms used in the document.

SSI - Synchronous Serial Interface: A device to transmit data with synchronous serial communication protocol.
Linearity - A measure of the straightness of the static calibration curve.
Frequency response – A standard technique to describe the dynamic behavior of linear systems
Loudness – Determined by amplitude of a wave.
Pitch – Determined by frequency of a wave.
Instrument – A embedded system that collects information, same as data acquisition system.
Tempo – Defined by the speed of music.
Envelope – Defines the amplitude vs time.
Melody - Linear succession of musical notes and tones and is a combination of pitch and rhythm
Harmony - The use of simultaneous tones, notes or chords and is referred to as the ‘vertical’ aspect of music

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 plays a horn sound.

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 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 **for(;;){}** 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 xxxx. 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.

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

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.

2.3. Prototypes: How will intermediate progress be demonstrated?

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.

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 three switch inputs. The DAC will be interfaced to a 32-ohm speaker. Between the Speaker and the DAC, there will be an audio amplifier with gain of 1. Switch will use negative logic. DAC will be interfaced with TM4C123 using SSI. DAC will use approximately an 1.5V reference.

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

Will use speaker.

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 HARDWARE DESIGN

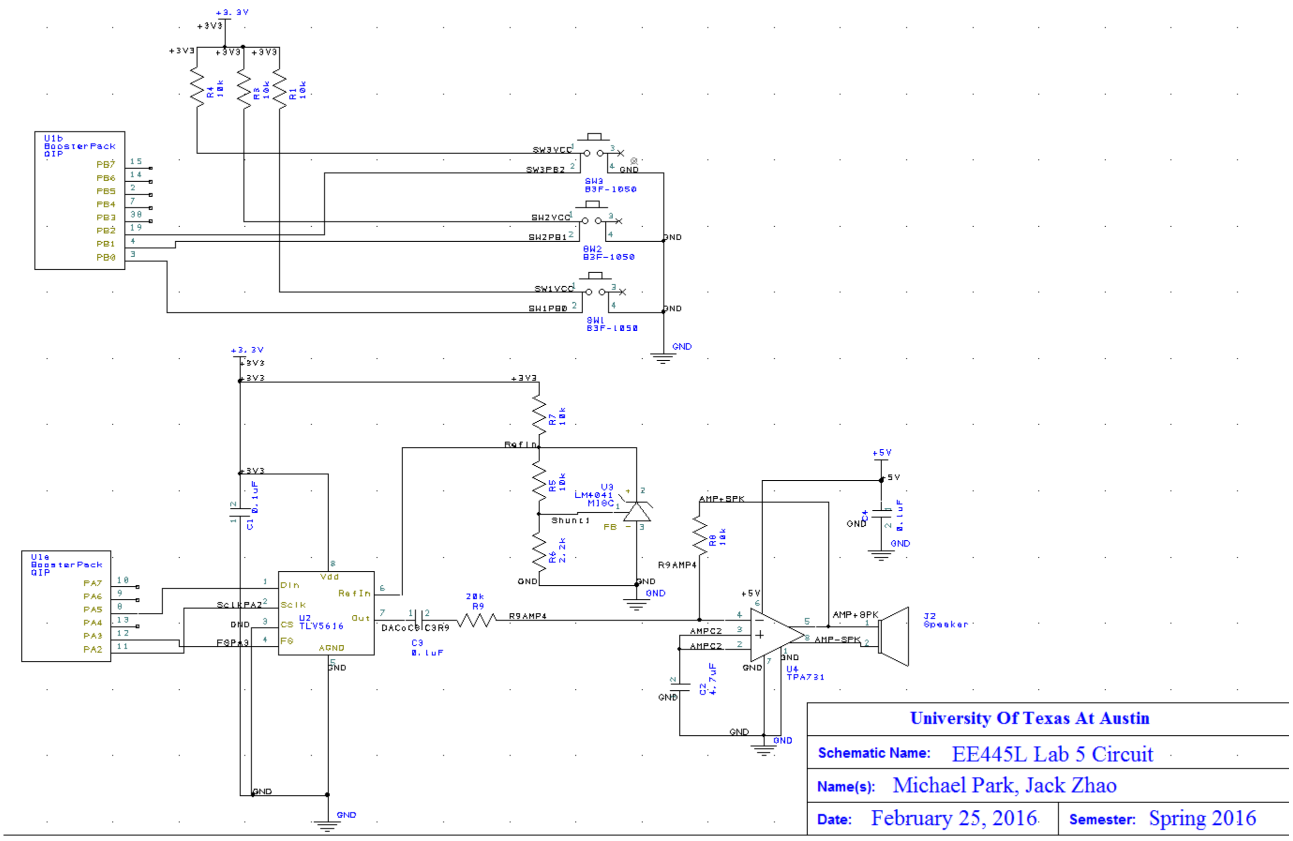

Figure 1: Lab5 hardware schematic

3.0 SOFTWARE DESIGN

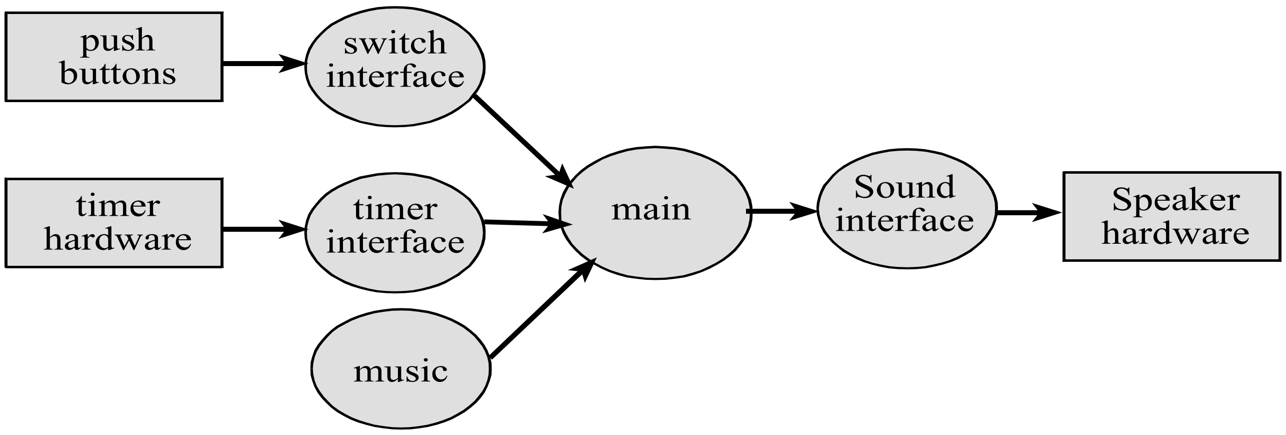

Figure 2: Data flows from the memory and the switches to the speaker

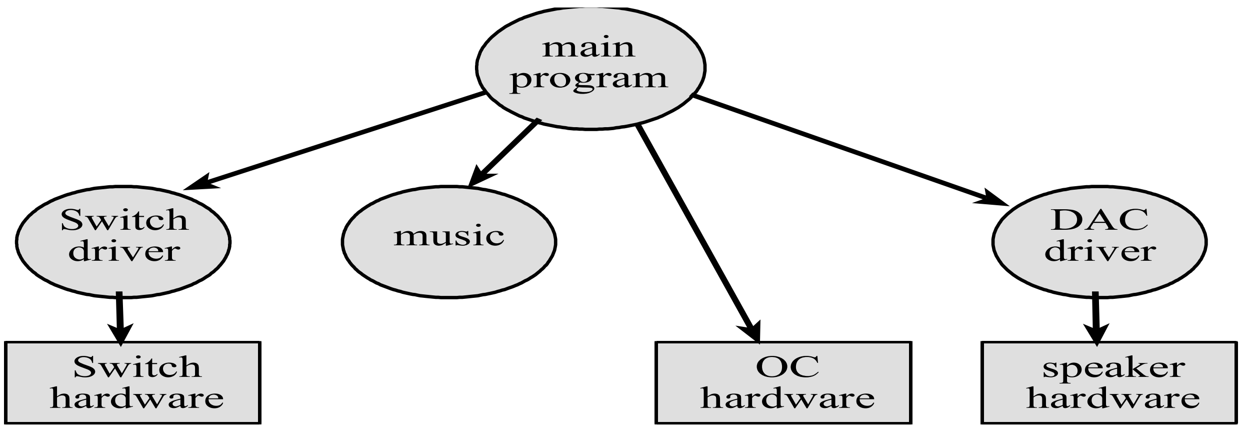

Figure 3: A call graph showing the three modules used by the music player.

4.0 MEASUREMENT DATA

(i) DAC output versus digital input for 8 different digital inputs. Compare the measured data with the expected values. Calculate resolution, range, precision and accuracy of the DAC.

(Measured Output)

intput to DAC	Output Voltage
348	0.295 V
261	0.221 V
400	0.335 V
440	0.373 V
522	0.443 V
660	0.560 V
742	0.630 V
880	0.747 V

(Expected Output)

intput to DAC	Output Voltage
348	0.280 V
261	0.210 V
400	0.322 V
440	0.355 V
522	0.421 V
660	0.532 V

742 - 0.598 V
880 - 0.709 V

DAC
range: 0 – 3.3V
precision: 12bit (0-4095)
resolution: 3.3/4095 = 0.0008058608
accuracy: (0.295 – 0.28)/0.295 = 0.05 -> 5% error. Thus 95% accurate

(ii) Using an oscilloscope and spectrum analyzer, measure the time-domain and frequency-domain outputs from your system at one frequency, like Figure 8.35 in the textbook. Using the spectrum, calculate SNR (ratio of the sinewave output to the largest noise component).

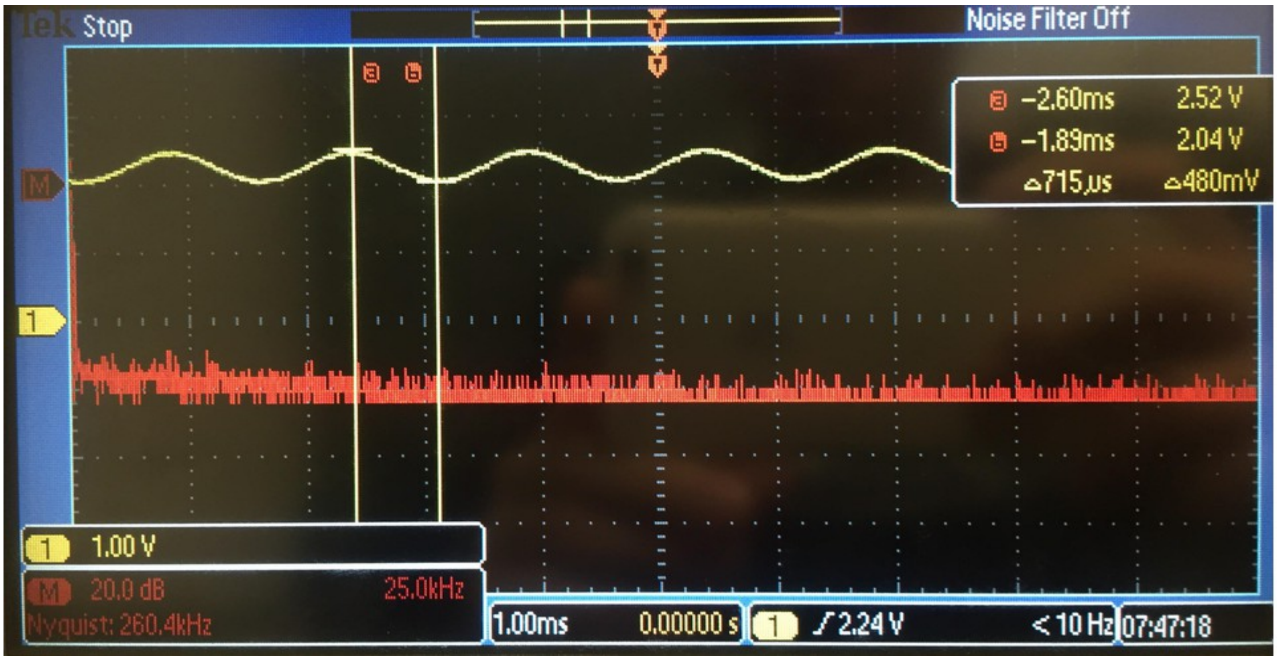

Figure 4: time-domain

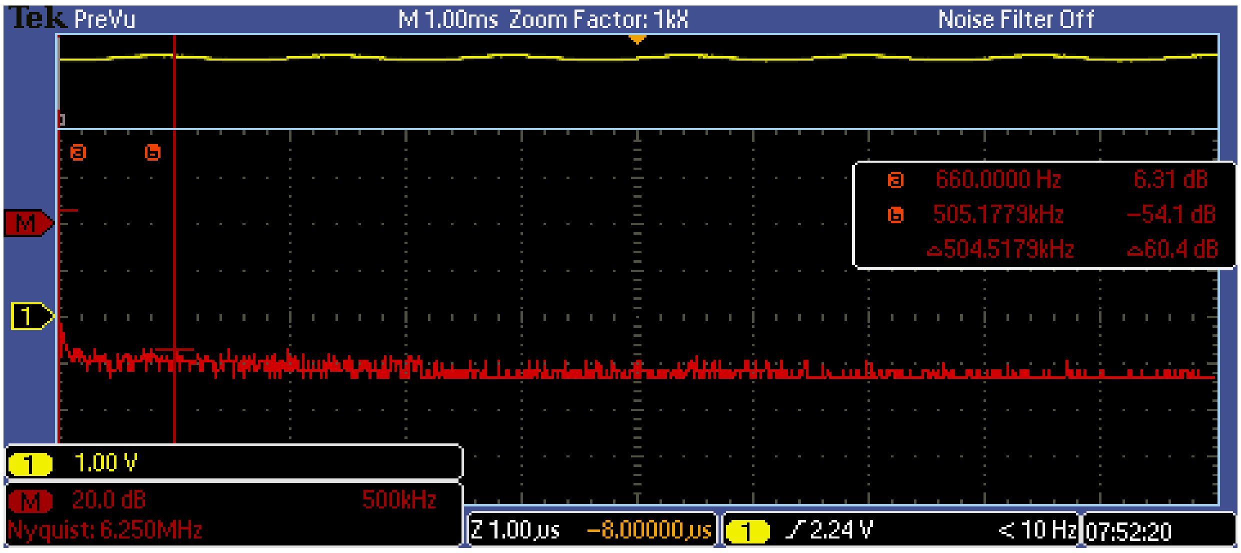

Figure 5: frequency-domain

SNR = 6.31dB - (-54.1dB) = 6.31 + 54.1dB = 60.41 dB

(iii) Using debugging instruments, measure the maximum time required to execute the periodic interrupt service routines. In particular, create a debugging profile to measure the percentage processor time required to play the song.

Interrupts = 95 cycles

Main = 271 cycles

95/271 = 0.35; Percentage of time required to play song = 35%

(iv) Measurements of current required to run the system, with and without the music playing

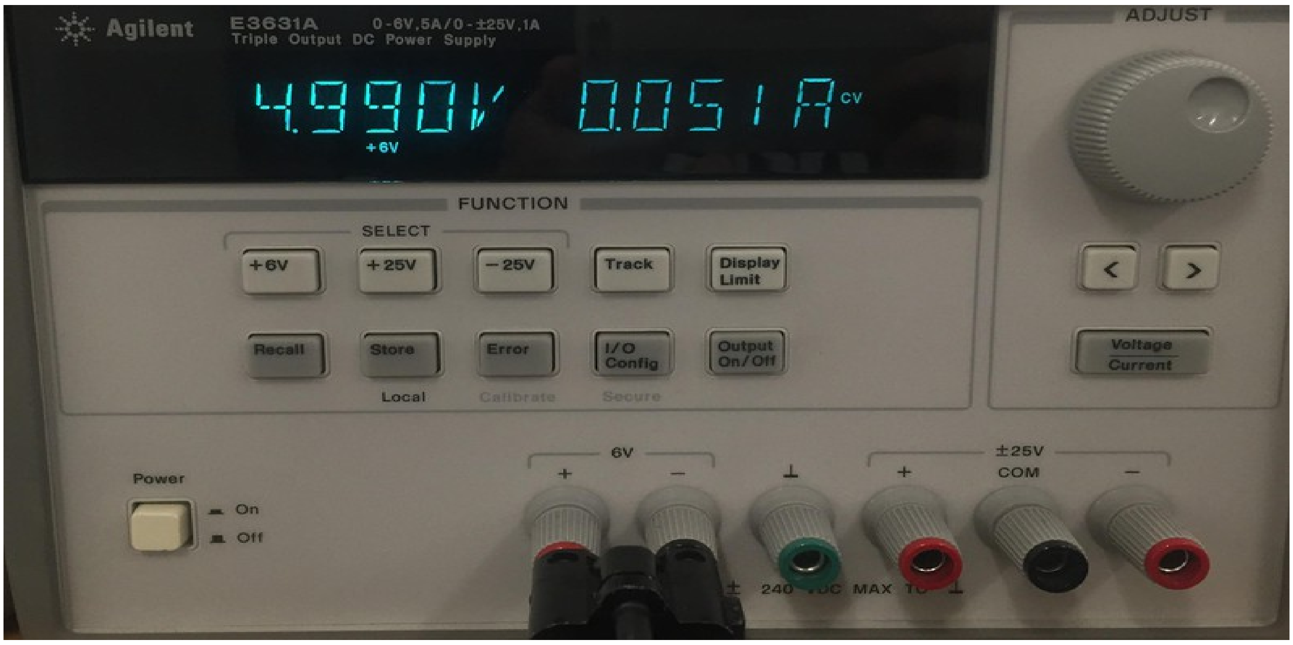

Figure 6: current required without the music playing

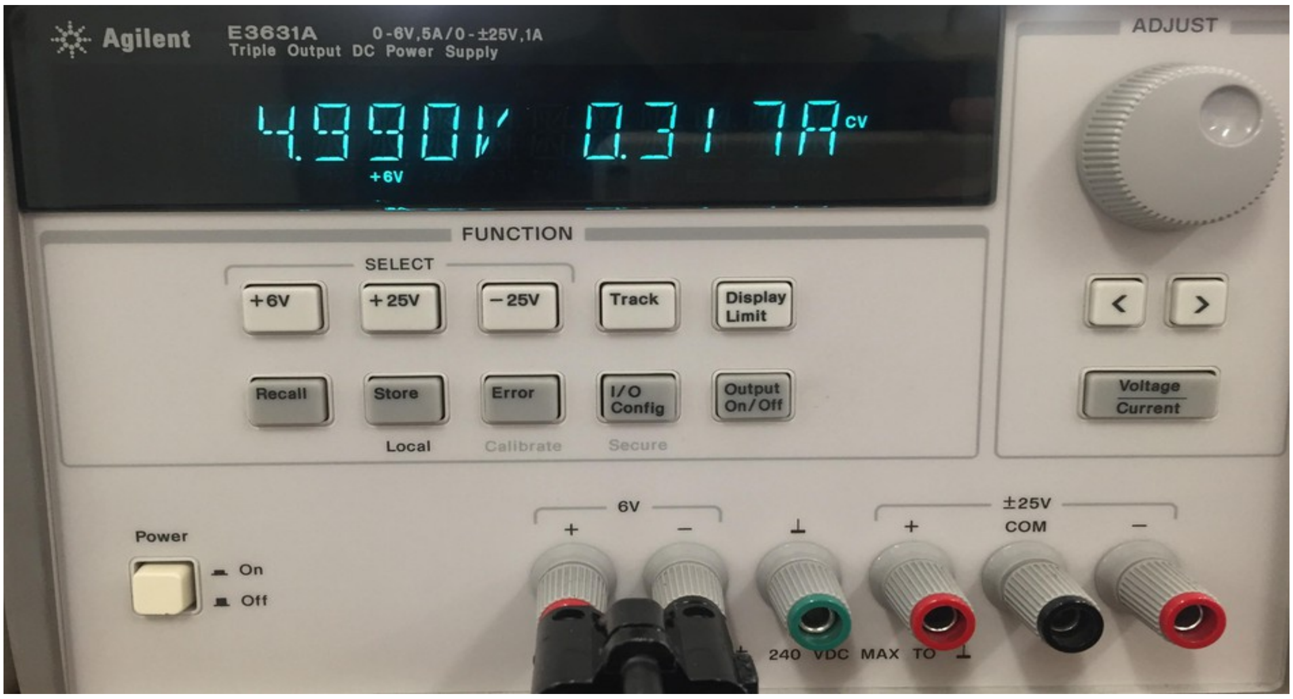

Figure 7: current required with the music playing

5.0 ANALYSIS AND DISCUSSION

1) Briefly describe three errors in a DAC.

(i) DAC gain error- A DAC gain error is a shift in the slope of the Vout versus the digital input static response. This will affect the tuning of our notes.

(ii) DAC offset error- A DAC offset error is a shift in the Vout verses digital input static response. This affects the initial values of Vout and affects the range.

(iii) DAC linearity error- A DAC linearity error is the maximum deviation at any point in the transfer function, of the output voltage level to the ideal voltage level. This means that our outputs will not be shaped perfectly linearly which will affect the shape of our sine wave

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

Data Required: (cp – tsu, cp +th) = (25-8, 25+5) = (17, 30)ns

For this lab which uses the TM4C123 and TLV5616, the maximum baud rate of the TM4C123 (initialized to 50 MHz) gives a setup and hold time (based on the baud rate, 1 system bus period and 2 system bus periods respectively) above the minimum for the DAC chip(8MHz). Therefore, we can use the maximum baud rate without any trouble.

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

The frequency range of the spectrum analyzer is determined through a process of mapping out differences in dB with frequncy. The graph is created through a process of fourier transform which will convert data in the time domain to the frequency domain. In terms of our spectrum analyzer, it uses a FFT which is a fast fourier transform to map out time domain into the frequency domain.

4) Why did we not simply drive the speaker directly from the DAC? I.e., what purpose is the TPA731?

The TPA731 is an audio amplifier. The reason we cannot drive the speaker directly from the DAC is that there is not enough current outputted from the DAC to actually drive the speaker to emit a sound. Once we implement the Audio Amp circuit, the current that will pass into the speaker input is increaesed enough to power the speaker.