Ronald Macmaster and Parth Adhia

EID: Rpm953 and Pda375

10/07/2016

EE 445L – Lab 5: Music Player and Audio Amp

1.0: Objectives:

Revised Requirements document

1. Overview

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

Our primary objective for this project is to design, build and test a music player. We are learning how to interface a DAC, design and test a speaker amplifier, store digital music in ROM, and perform DAC output in the background.

Additionally, we will develop two or three low-level device drivers for our TLV5616 DAC to communicate with it via SSI protocol. We will also design a hardware / software interface for a couple of switch inputs. We will measure the supply current necessary to operate our embedded system, and implement our music player with periodic timer interrupts.

Our project will also require us to design data structures to represent music loops and instrument sounds. Finally, we will explore the deeper design principles of signal communication with spectral measurements and observations.

1.2. Process: How will the project be developed?

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 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.

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

Ronald Macmaster and Parth Adhia are the engineers and the Dylan Zika (TA) is the client. We have modified this document to clarify exactly what we plan to build. We are allowed to divide responsibilities of the project however we wish, but, at the time of demonstration, both of us are expected to understand all aspects of the design.

Who will do what?

Ronald Macmaster will be responsible for drafting the requirements document and the lab report. He will also create the software design diagrams that will consist of a call graph and a data-flow graph. He will draft up the interface files for the various software modules and create program skeletons for the various driver files. Finally, he will supervise and contribute to the software development process and submit all of the assignment documentation.

Parth Adhia will be responsible for drafting the hardware design documentation. This will require a schematic diagram of the external system hardware that should be built using the PCB Artist program. He will write a significant portion of the software drivers and assemble the external system hardware circuit. He is responsible for providing the speaker and tactile switches as well. Finally, he will edit and contribute to the various project documentation as fit.

Both engineers will contribute equally to the schematic layout. The schematic layout will require both engineers to research the datasheets for the TLV5616, TPA731, LM4041, and the TM4C123.

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

The system will use the TM4C123 board, a solderless breadboard, and the speaker as shown in Figure 5.1. It will be powered using the USB cable. **We will use a +5V power from the lab bench, and we will not power the TPA731 or the speaker with a voltage above +5V.**

1.5. Terminology: *We herein define the following key terms*:

SSI (Synchronous Serial Interface):

The **SSI** system allows microcontrollers to communicate synchronously with peripheral devices or other controllers over a serial protocol. In an SPI implementation, two devices operate over a communicated and synchronized clock signal.

Linearity:

Linearity a mathematical relationship that implies a signal can be graphically represented as a straight line. One quantitative measure of linearity is the linear regression correlation coefficient. Given a function f(n), the output is linear if f(n+1) – f(n) = f(m+1) – f(m) = dx for all n and n.

Frequency Response:

The **Frequency Response** is a quantitative measure of the system's output signal frequency spectrum in response an input signal. It is often used to describe the dynamic behavior of an LTI system.

Loudness:

The **loudness** of the sound wave is our psychological perception of the wave's physical strength. It is correlated to the magnitude of the sound wave's amplitude.

Pitch:

Pitch is another property of sound that allows the perception of ordering on a frequency-based scale. Pitch provides our means to perceive notes as "higher" or "lower" in comparison to other notes.

Instrument:

An object that is created or adapted in order to create musical sounds. Anything that produces sound can be considered a musical instrument, but we usually reserve this classification for things like a flute, piano, guitar, or violin.

Tempo:

In the world of music, the **tempo** of a song defines the speed of a song. It is usually calculated in beats per minute (bpm).

Envelope:

The **envelope** of a signal is the smooth curve that outlines its amplitude extremes. It can be considered the relationship of amplitude structure to time.

Melody and Harmony:

The **melody** of a song is the combination of rhythm and pitch that gives it its primary identity. You recognize a song by the tune of its melody. The **harmony** of the song is the lower, supporting role to the melody that adds completeness to the song. The melody alone sounds empty without the harmony.

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 will 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 our team to keep its EE445L lab solutions secure. **We will manage our source code over a private git repository hosted by GitHub Inc.**

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 some aspect of the player. **Pressing the mode switch will toggle the music player's playlist feature (switch the song)**. **The tempo of the song can also change through the switch interface.**

There will 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 we 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. Any optional LCD output should occur in the foreground. The maximum time to execute one instance of the ISR is **2.8 microseconds**. We 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. We 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 we 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.

For preparation, we will deliver an updated requirements document, schematic diagram of our hardware system, and the call and data-flow graph that models our software design. We will also demonstrate possession of the necessary hardware components and reasonable progress on the various software modules. The software will be written and compiled by the preparation date.

For checkout demonstration, we will begin by demonstrating the features of our DAC Driver. We will also present our completed music player software and system. The music player will be stand-alone (powers on and off) and the speaker will be placed in a box.

The final documentation for this project will be written up in a lab report and submitted over Canvas for grading by Midnight on Friday, October 7th.

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 Kerrigan and Ritchie's book, *The C Programming Language*. 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, we 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 an 8-ohm speaker.

The TM4C123 and TLV5616 will interface through a custom SPI protocol. The SSI Clock will be set to 10Mhz during operation. The output of the TLV5616 will flow into the TPA731, and the TPA731 will amplify the output to our speaker.

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. Connecting or disconnecting wires on the protoboard while power is applied may damage the board.

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.

The final lab report is due on Friday, October 7th at Midnight. This report will include the final requirements document. The outline of the lab report is as follows:

A) Objectives (final requirements document)

B) Hardware Design (External hardware schematic to interface TLV5616)

C) Software Design (Updated software modules, Call graph, and Data-flow graph)

D) Measurement Data (Data and calculated resolution, range, precision, and accuracy of the DAC, experimental response of the DAC, debugging profile, +5V voltage and RMS voltage, and current required with and without music playing.)

E) Analysis and Discussion (short answers to a couple design questions)

3.2. Audits: How will the clients evaluate progress?

Clients will be presented with a preparation the week before before the final checkout. The lab report will be presented at conclusion of the project.

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

There are three deliverables: preparation, demonstration, and report. **Most of our work and design will be documented in the final lab report.**

2.0 Hardware Design:

Our system will interact with a TLV5616 DAC and TPA731 AC Amplifier. The final ouput will be exerted on an 8-Ohm speaker. The three tactile switches will be implemented with positive logic, and the speaker will be interfaced through the amplifier. The TM4C123 and TLV5616 will communicate over an SSI interface with a shared clock speed of 10Mhz. The frame format specifies the clock will remain a logical high during the idle phase. The SSI FIFO will begin the shifting process after the first active clock edge on both communication ends.

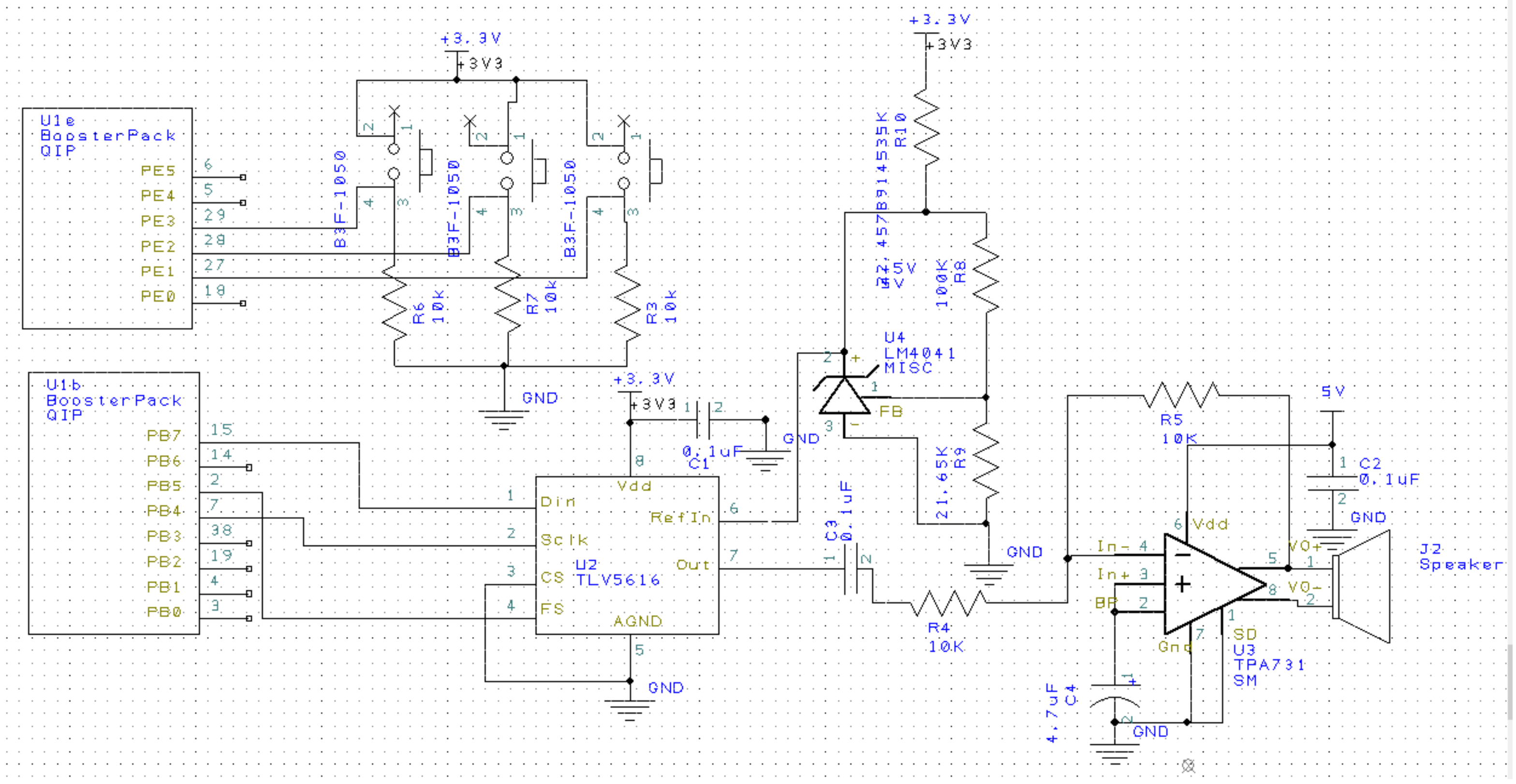


Figure 2.1: *Hardware schematic for our music player system. The TLV5616 is the DAC interfaced through SSI. The DAC output is directed into the TPA731 amplifier which is then delivered to the 8-Ohm speaker.*

3.0: Software Design:

We have decided to proceed with a design similar to the original software architecture presented in the lab manual. All of our required and additional functionality can be implemented using four basic modules — switch control, music management, time management, and DAC / SSI Control. Submitted along with this report are the interface and driver files (.h and .c) for our software modules (DAC, Music, Timer, and Switch).

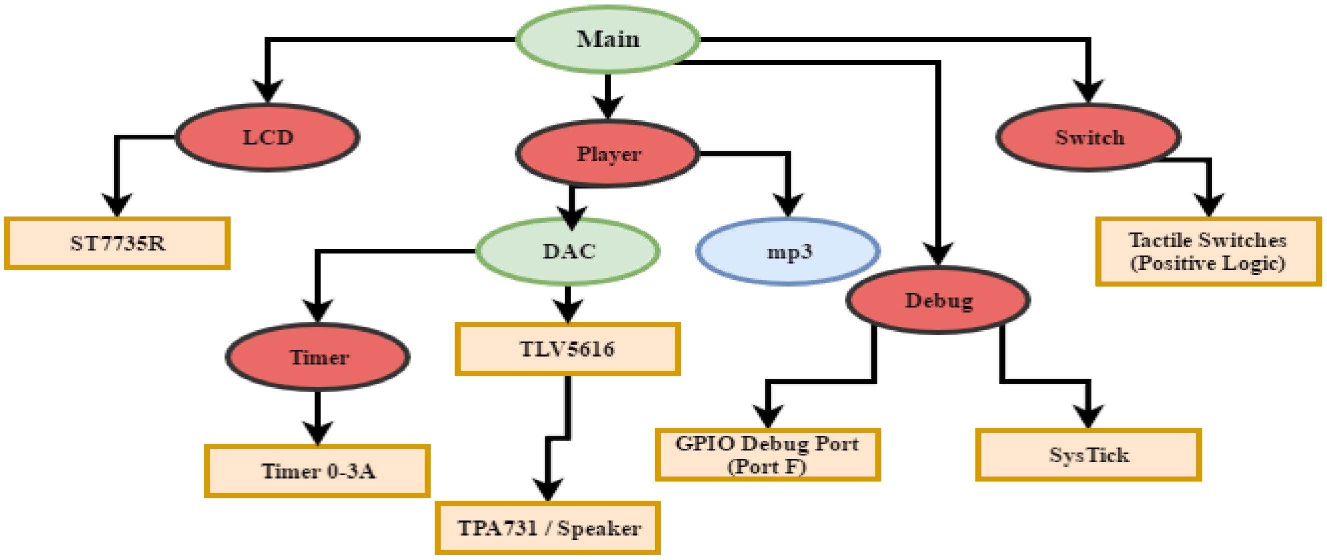


Figure 3.1: *Call graph for our Music Player system. A main driver program manages hardware through secondary software modules. The DAC implementation is abstracted through an interface*

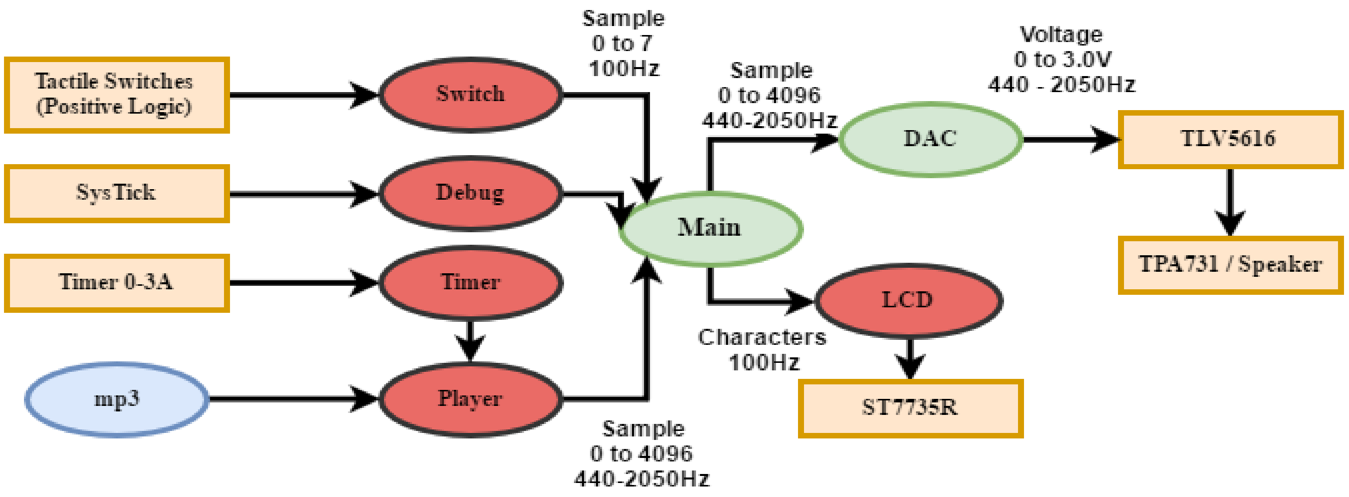

Tactile Switches (Positive Logic)
Switch
Sample 0 to 7 100Hz
SysTick
Debug
Timer 0-3A
Timer
mp3
Player
Sample 0 to 4096 440-2050Hz
Main
Sample 0 to 4096 440-2050Hz
DAC
Voltage 0 to 3.0V 440 - 2050Hz
TLV5616
TPA731 / Speaker
LCD
Characters 100Hz
ST7735R

Figure 3.2: *Data-flow graph for our Music Player system. Hardware service requests are from the timer and switch interfaces. The main driver background logic outputs the sound data to the DAC interface.*

4.0: Measurement Data:

4.1 DAC Test and Calculations (Range, Precision, Accuracy)

Show the data and calculated resolution, range, precision and accuracy (procedure 3)

Table 4.1: DAC Voltage Measurements in Volts. (Expected vs. Actual). Vref = 1.50V

DAC Output	DAC Voltage (Expected) [V]	DAC Voltage (Actual) [V]
0	0	0
512	0.375	0.4
1024	0.75	0.76
1536	1.125	1.12
2048	1.5	1.52
2560	1.875	1.92
3072	2.25	2.24
3584	2.625	2.64

Calculations:

Resolution = range / resolution = 3.0V / 4096 = **0.732 mV**

Range = Vmax – Vmin = 3.0V – 0.0V = **3.0V**

Precision = number of alternatives = **4096 (12-Bit precision)**

Accuracy = (sum(X_{actual} – $X_{expected}$)) / N =

(0 + 0.025 + 0.01 + -0.005 + 0.02+ 0.045 + -0.01 + 0.015)V / 8 = **0.125V**

DAC Linearity Test

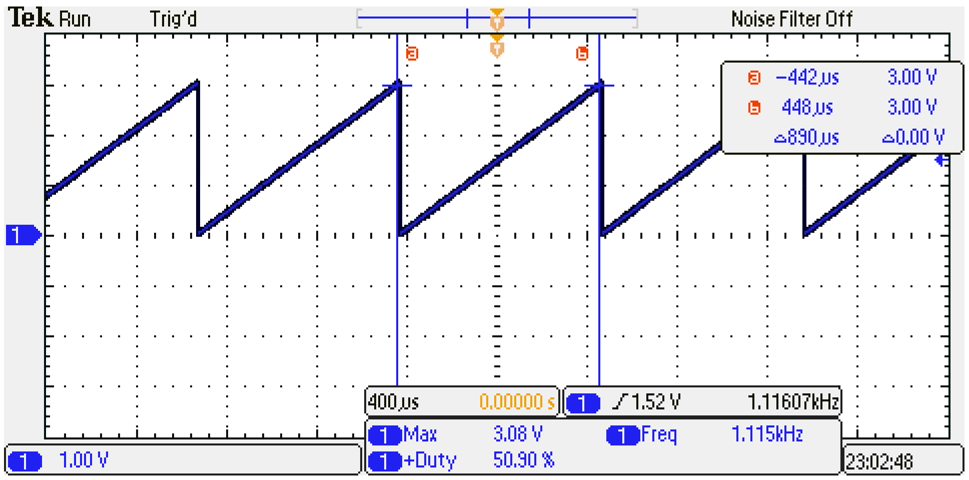


Figure 4.1: *Time response of our DAC output for a single frequency. Note played is C1 (2093Hz) with a period of ~475us. The Time response is a sine wave.*

4.2 DAC Experimental Response (Time response and Frequency Spectrum)

Show the experimental response of DAC (procedure 4) including SNR

Time Response

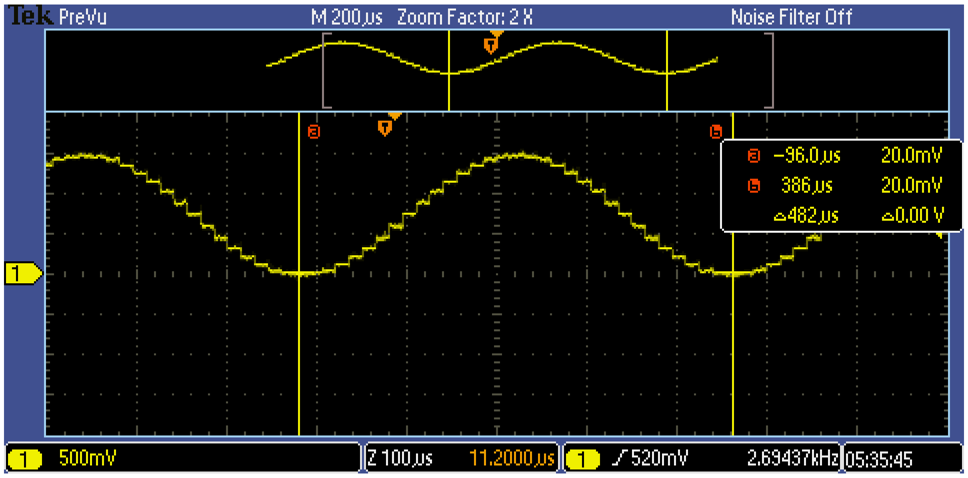


Figure 4.2: *Time response of our DAC output for a single frequency. Note played is C1 (2093Hz) with a period of ~475us. The Time response is a sine wave.*

Frequency Spectrum

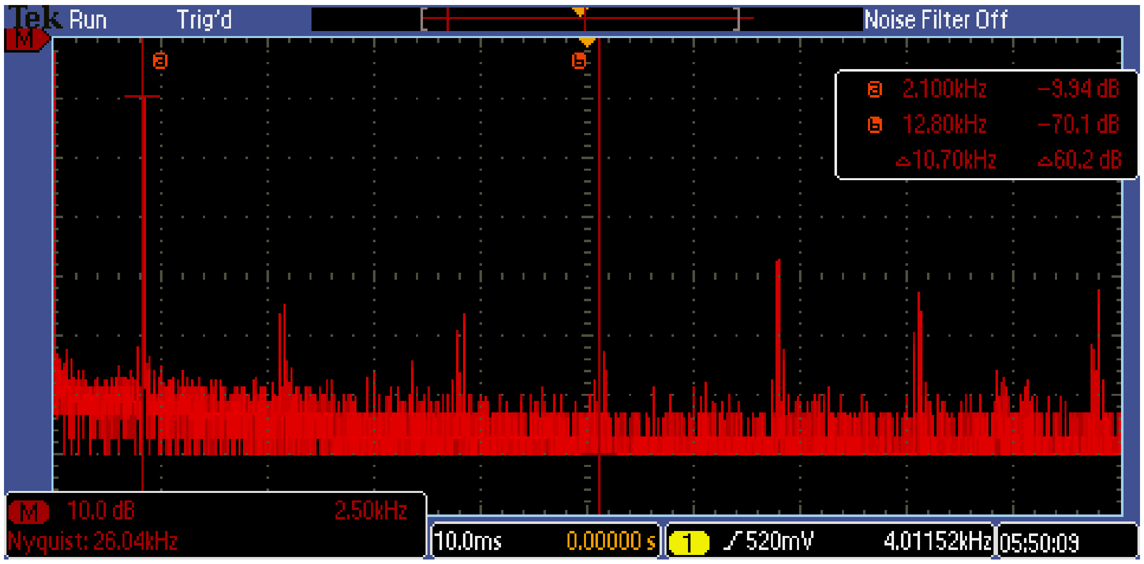


Figure 4.3: *Frequency Spectrum of our DAC output for a single frequency. Note played is C1 (2093Hz) with a period of ~475us. The Frequency Spectrum shows a dominant signal at 2050Hz and a dominant noise signal at 12800 Hz.*

Signal to Noise Ratio:

We have a signal level of **-9.94dB** and a noise level of **-70.1dB.** Thus, our signal to noise ratio (SNR) is as follows: SNR = signal level – noise level = -9.94dB – (-70.1dB) = **60.16 dB**

4.3 ISR Debugging Profiles (Tempo, Melody, Harmony)

Show the results of the debugging profile (procedure 5)

Tempo Thread

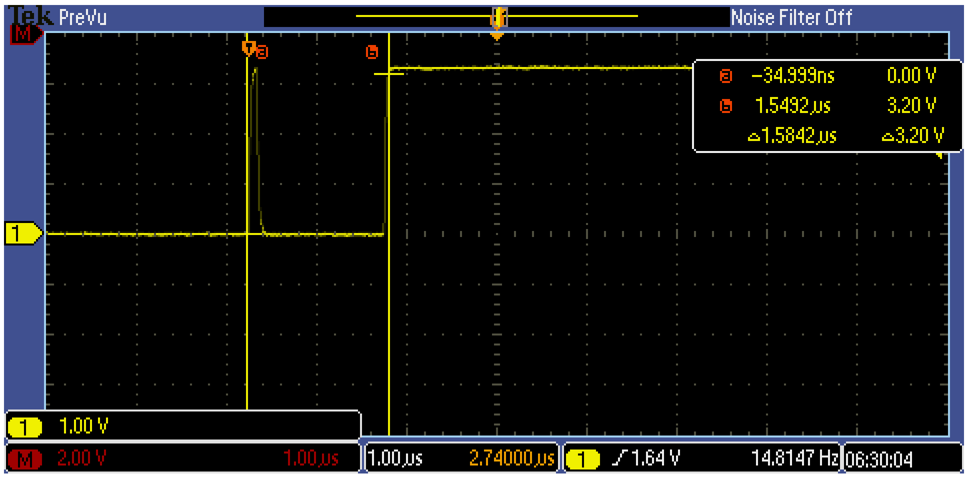


Figure 4.4: *Profile of our Tempo thread. This ISR will update the song cursor and change the frequency of the other three timers. That will cause the notes to change.*

Melody Thread

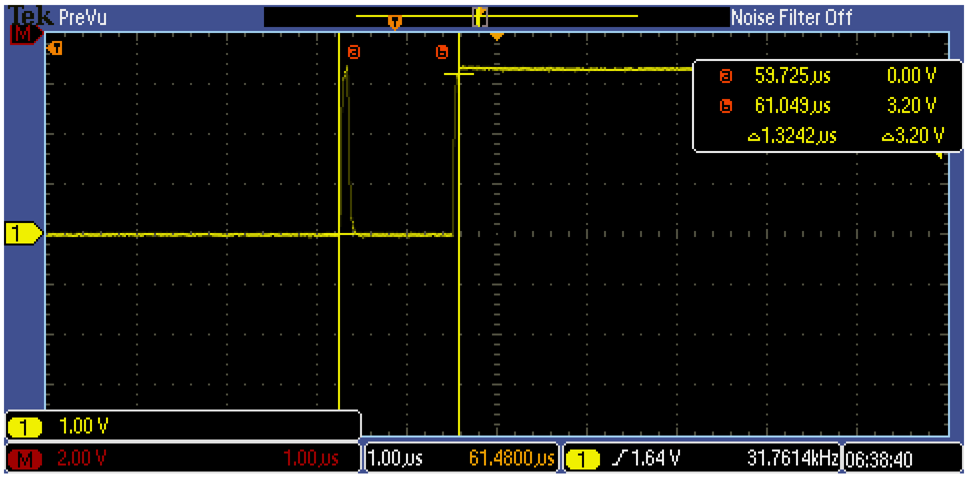


Figure 4.5: *Profile of our Melody thread. The ISR outputs the melody notes to the DAC with the DAC_Out() function. The data is read from an instrument wave and melody mp3.*

Harmony Thread

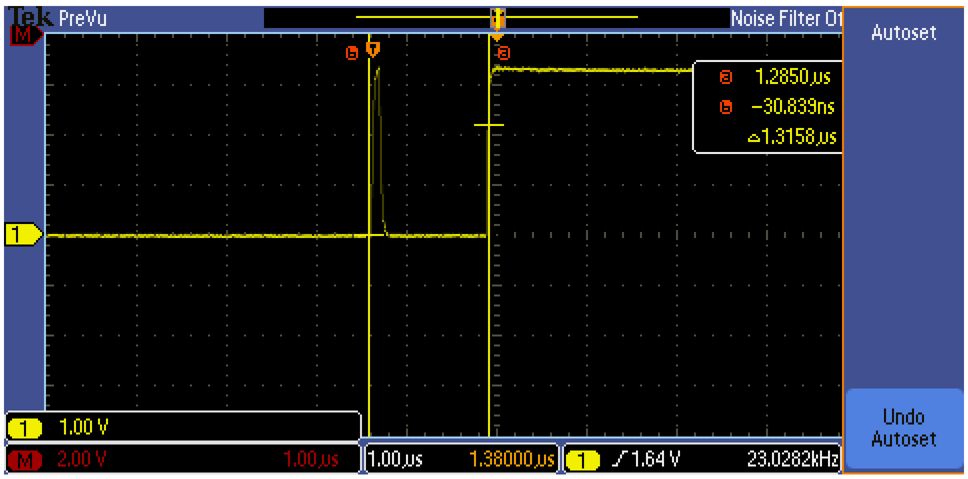


Figure 4.6: *Profile of our Harmonyy thread. The ISR outputs the harmony notes to the DAC with the DAC_Out() function. The data is read from an instrument wave and harmony mp3.*

Performance Analysis

Our maximum thread execution time is determined by the longest executing background thread, the tempo interrupt. That thread takes approximately **1.549 us** to complete. To calculate the processor execution percentage spent on playing the song, we need to add up the total time spent in all three threads. Given a single thread, the time spent in it every second is (t_{exec}) * (f_{thread}). Processor % execution time = (1.32us * 23.028kHz) + (1.32us * 31.761kHz) + (1.58us * 14.815Hz) = 0.0723 = **7.23%**

4.4 Amplified Speaker Output (TPA731 Response)

Amplifier to Speaker Signal

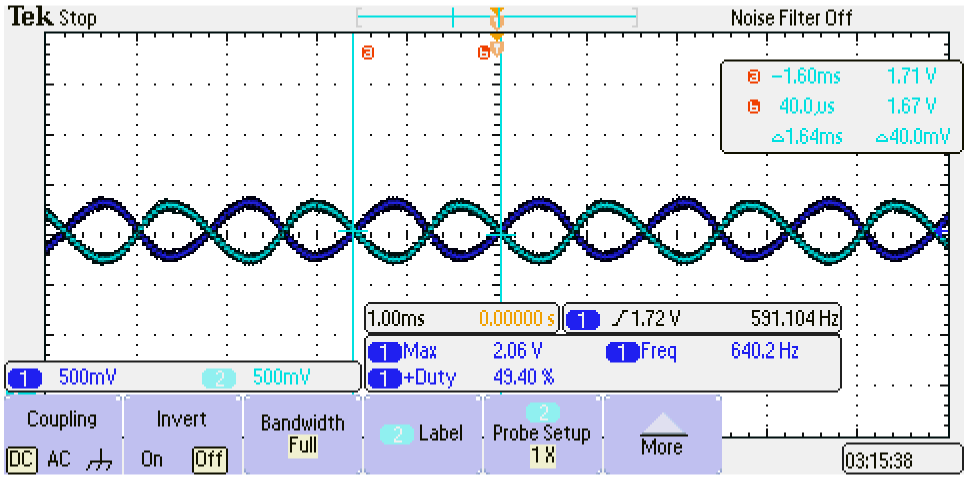
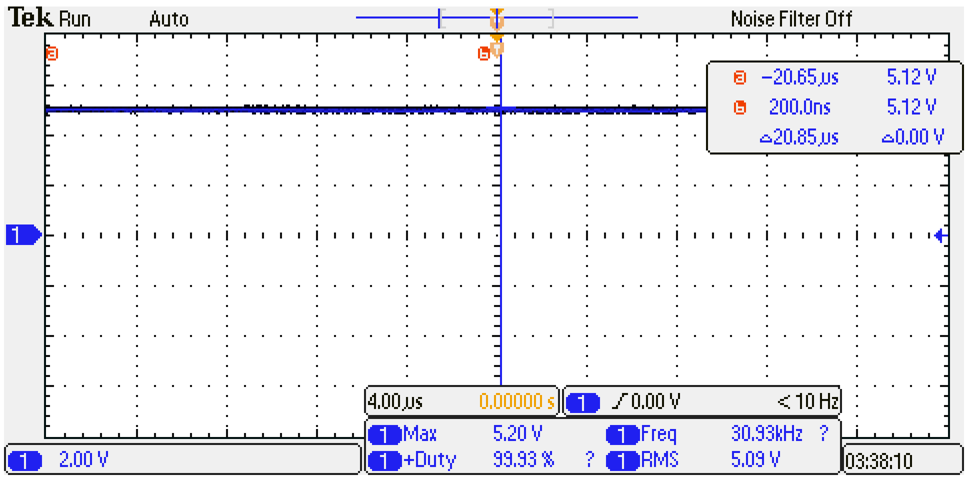


Figure 4.7: *Two channel recording of pins 5 and 8. DC component is 1.5V (about ½ of +3V power). The frequency of our sound wave is about 609 Hz.*

4.5 Supply and Power Analysis

Show +5V voltage, voltageRMS (which will be very small) and current, with and without the music playing

5V Power Supply Voltage



<u>Figure 4.7:</u> *The 5V power supply voltage from our TM4C123 microcontroller.* ***Vrms = 5.09V***.

Current Measurements:

(with music playing, without music playing) = (131mA, 90mA).

Note: Current measurements include the current required to power the LCD.

5.0: Analysis and Discussion:

1) Briefly describe three errors in a DAC.

Three errors a DAC may experience are gain error, offset error, and non-linearity or non-monotonicity. Gain error describes an unexpected shift in the slope of V_{out} vs. D_{in}. Offset error describes an unexpected shift in the y-intercept of V_{out} vs. D_{in}. Finally, non-linearity and non-monotonicity occur when the DAC relationship isn't consistently linear or monotonic. Precisely, the DAC experiences nonlinearity if there exist n and m where f(n+1)-f(n) != f(m+1)-f(m).

2) 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 TLV5616 data sheet, the setup time and hold time for our digital input are 8ns and 5ns respectively. Using our SSI frequency of 10MHz, the period of the SSI clock is 100ns. Our D_{in} signal from the microcontroller is synched with the SSI clock, so our **data available** interval is **100ns**. The data sheet does not specify a **data required** interval, but it illustrates it as ~0ns on the timing diagram or approximately the time required for a falling clock edge. Thus, our data available range just has to exceed $t_h + t_s$ = **13ns**. Our choice of a 10Mhz SSI frequency is justified because data available > (data required + setup time + hold time).

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

Frequency range denotes the range of frequencies that the spectrum analyzer can identify without aliasing. It is determined by the sampling frequency of the spectrum analyzer. According to the Nyquist theorem, the range of available frequencies at a sampling frequency of f_s is from **-½ f_s to ½ f_s**. We can sometimes identify this range by looking at the range of frequencies available on the spectrum analyzer's x-axis for our input signal.

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

Our DAC cannot supply the necessary power (current and voltage combo) necessary to drive the speaker. We use the TPA731 to amplify the power supplied to the speaker and also increase the volume of our music. The volume of the speaker is proportional to the amount of current the speaker receives.