ECE 445L Lab 4

12-bit DAC, SPI, Music player, audio amplifier

This laboratory assignment accompanies the book, [*Embedded Systems: Real-Time Interfacing to ARM Cortex M Microcontrollers, ISBN-13: 978-1463590154*](https://www.amazon.com/Embedded-Systems-Real-Time-Interfacing-Microcontrollers/dp/1463590156), by Jonathan W. Valvano, copyright © 2024.

# Table of Contents

[Table of Contents 1](#_Toc173657006)

[Team Size 2](#_Toc173657007)

[Goals 2](#_Toc173657008)

[Review 2](#_Toc173657009)

[Starter Files 2](#_Toc173657010)

[Required Hardware 2](#_Toc173657011)

[Lab Overview 3](#_Toc173657012)

[Preparation 3](#_Toc173657013)

[Procedure 7](#_Toc173657014)

[Deliverable 1 7](#_Toc173657015)

[Deliverable 2 8](#_Toc173657016)

[Deliverable 3 8](#_Toc173657017)

[Deliverable 4 8](#_Toc173657018)

[Deliverable 5 8](#_Toc173657019)

[Deliverable 6 8](#_Toc173657020)

[Deliverable 7 8](#_Toc173657021)

[Deliverable 8 (15pts Extra Credit) 9](#_Toc173657022)

[Lab Checkout 9](#_Toc173657023)

[Lab Report 9](#_Toc173657024)

[Hint (Audio waveform basics) 10](#_Toc173657025)

[Hint (System Call Graphs) 13](#_Toc173657026)

# Team Size

The team size for this lab is **2**.

# Goals

* Understand Digital to Analog Converters (DACs) and voltage references,
* Create a simple SPI/SSI interface,
* Design data structures to represent music,
* Develop systems to play sounds.

# Review

* Search <http://www.ti.com/> for a data sheet on the TLV5616CP and TLV5618CP 12-bit DAC,
* Valvano Section 6.2 on periodic interrupts using the timer,
* Valvano Section 7.5 on SSI interfacing,
* Valvano Section 8.4 on DAC parameters and waveform generation.

# Starter Files

* Example projects:
  + PeriodicTimer0AInts\_xxx project,
  + Max5353\_xxx Excel files starting with *dac\_*.
* Starter project:
  + Lab 4 template provided on the GH Classroom repo.
  + LM4041 voltage reference design XLSX in resources folder.

# Required Hardware

|  |  |  |  |
| --- | --- | --- | --- |
| Parts | Datasheet | Price | Source (**price source)** |
| EK-TM4C123GXL | [EK-TM4C123GXL datasheet](https://github.com/ECE445L/ECE445L-Lab5/blob/main/resources/TM4C_Datasheet.pdf) | $16.99 | **TI** |
| 8Ω or 32Ω speaker | N/A | N/A | EER Checkout Desk |
| Resistors and capacitors | N/A | N/A | EER Checkout Desk |
| Switches | N/A | N/A | EER Checkout Desk |
| TLV5616CP 12-bit DAC or TLV5618ACP 12-bit DAC | [TLV5616 datasheet](https://github.com/ECE445L/ECE445L-Lab5/blob/main/resources/part_datasheets/tlv5616.pdf) | $9.61 | EER Checkout Desk  Or **Mouser**, Digikey |
| LM4041CILPR shunt diode  or LM4041C12ILP | [LM4041C datasheet](https://github.com/ECE445L/ECE445L-Lab5/blob/main/resources/part_datasheets/lm4041c.pdf) | $0.78 | EER Checkout Desk  Or **Mouser**, Digikey |
| TPA731D audio amp  if can’t get MC34119P | [TPA731 datasheet](https://github.com/ECE445L/ECE445L-Lab5/blob/main/resources/part_datasheets/lm4041c.pdf) | $2.54 | EER Checkout Desk  Or **Mouser**, Digikey |
| MC34119P (discontinued) | [MC34119 datasheet](https://github.com/ECE445L/ECE445L-Lab5/blob/main/resources/part_datasheets/mc34119.pdf) | N/A | EER Checkout Desk |

You can use TPA731D or MC34119P (not both)

You can use LM4041CILPR or LM4041C12ILP (not both)

You can use TLV5616CP (single) or TLV5618ACP (dual) DAC (not both)

# Lab Overview

Many embedded systems require the generation of analog signals. One example is a digital music player. This device relies on digital to analog converters (DACs) to create high-quality waveforms. In this lab you will use a 12-bit DAC to create a sine-wave output.

You will interface a TI TLV5616 or TLV5618 12-bit DAC to an SSI port. Note that you are allowed to use any DAC chip you want if it runs on a single +3.3V supply and has an SSI interface. Additionally, you will create a voltage reference circuit because most DACs require a voltage reference. Once the DAC is able to produce an analog signal based on commands from the TM4C it will be time to interface an audio amplifier in order to drive a speaker.

# Preparation

Preparation is performed before or during the W/TH lab session.

1. Requirements Review
   1. Refresh your knowledge of audio waveforms. It is recommended that you read the hint on audio waveforms if you are unfamiliar with the topic.
   2. Read the requirements document included in the Lab05Report. Be prepared to answer questions about what you are implementing to the TA’s.
      * What are the minimum requirements for the system?
      * What opportunities for extra features exist?
   3. Understand that you can and **should** edit the requirements document to reflect your design, and to solidify the team’s understanding of what the lab’s goals are.
   4. Understand that the performance score of this lab is **NOT** based on loudness, but sound quality. The quality of the music will depend on both hardware and software factors.
      * Hardware factors include the precision of the DAC, the linearity of the audio amp, the frequency response of the audio amp and the dynamic range of the speaker.
      * Software factors include the DAC output rate, the complexity of the stored music data, the jitter of the DAC output, and the signal the software generates.
2. Software Setup
   1. Understand that in this lab you will create an SSI module and drivers for the DAC. These drivers like most code in this class, **MUST** be written at a low level.
   2. Create the two header files DAC.h and Switch.h. Define at least two functions for the SSI/DAC interface and two functions for the switch interface.
   3. Design the data structure you will use to store the song. This data structure should minimally contain notes and their duration, but can include additional information such as rests, envelopes, instruments, or other data to reproduce the song.
      * Find the sheet music that you would like to play, and consider how this sheet music would be represented in your data structure.
      * Consider how you would include the actual constants needed to define the song.
   4. Consider what the System Call graph and Data Flow graph will look like. Examples of these graphs are shown in the hints at the end of this document.
3. Prepare the schematic
   1. You must draw a rough draft of the circuit for this lab in KiCad. You will find the TLV5616CP/TLV5618ACP, TPA731D/MC34119, and LM4041C symbol, footprint, and model files in the hardware folder, or on websites such as mouser and SnapEDA. To begin you should start by adding these parts to your schematic, then continue reading to see how to wire your circuit. Your schematic will Include:
      * Bypass capacitors for ICs on their power pin(s) sized appropriately (i.e. 0.1uF)
      * A DAC and its relevant SPI, reference, and output pins connected properly
      * A voltage reference circuit and its configuration resistors
      * An audio amplifier and its required passive components powered by VBUS
      * Two or more buttons as required.
      * Relevant signal names and pin numbers.
4. Hardware Setup
   1. Be prepared to show your TA that you can assemble the hardware for this lab; Before leaving prep, you must collect all hardware needed to perform this lab as listed below:
      * A DAC, such as the TLV5616/TLV5618
      * A Voltage Reference, such as the LM4041
      * An Audio Amplifier, such as the MC34119P/TPA731
      * A speaker, either 8 or 32 Ohms
      * Additional resistors, capacitors, and jumpers
      * (Optional) A pop-tart or small cereal box to mount the speaker to
   2. Consider the DAC used in this lab and its wiring. The TLV5616 has no digital data output, and the data sheet shows which pins to use for an SPI interface. Consider which TM4C123 pins you will connect to DIN, SCLK, and CS bar (active low) of the TLV5616 to. A screen shot of the TLV documentation is shown below. This datasheet can be found in resources/part\_datasheets of the starter project.

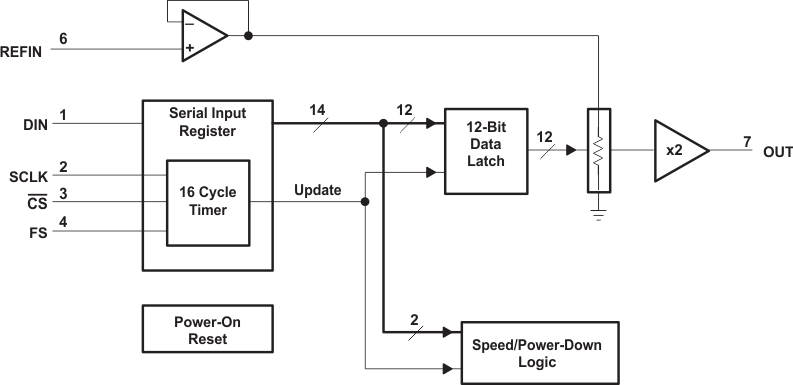


Figure 4.1. Block diagram of the DAC interface for the TLV5616. See the [datasheet](https://github.com/ECE445L/ECE445L-Lab5/blob/857eec15afeb5871c7a20e70286310ff80c41f41) to find which pins connect to DIN, SCLK, and CS bar.

* 1. Consider the Voltage reference required for this lab. As many DACs need a reference voltage, we must create one. The LM4041C12ILP is a fixed shunt reference. The LM4041CILPR is an adjustable shunt reference that can be used to create a variety of precise reference voltages. There is an excel sheet in the resources folder of the repo that you may use to help with the design process. Consider the following for your voltage reference circuit:
     + The reference voltage needed. Any fixed voltage from 1.20 to 1.50V is ok
     + The maximum current draw of devices requiring the reference voltage
       - HINT: Look up in the TLV5616 data sheet to find how much current the DAC needs on its REF input. In the data sheet you will find the input impedance Rin of the REF pin. This Rin can be used to calculate the load current IL = 1.5V / Rin.
     + The power source for the LM4041C. In this case the 3V3 Rail
     + The minimum cathode current of the LM4041C (IZ\_min = 80 µA)
     + The feedback voltage of the LM4041C (VREF = 1.233V)
  2. If you have the LM4041CI, calculate the resistor values for the reference circuit using the information found in the prior step. Current through R1+R2 will be IREF =VREF/(R1+R2). Select R1 and R2 to set the reference output. VZ = VREF (1 + R2 / R1) = 1.50V. The RS resistor in Figure 4.1 (Figure 14-3 of the datasheet) sets the available current for the shunt reference. Make RS ≤ (3.3 - VZ) / (IL + IREF + IZ).

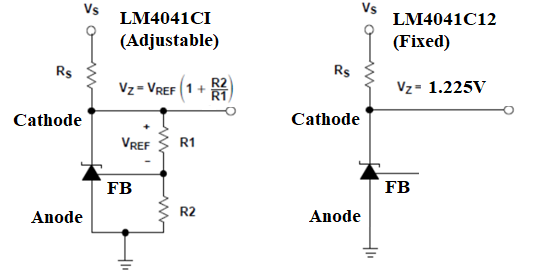


Figure 4.2. Shunt voltage reference

* 1. Consider the audio amplifier you will use. The audio amplifier is required to increase the strength (Equivalently to reduce the source impedance) of the DAC so that the signal can successfully drive the speaker. Consider using either the TPA731 or MC34119 when designing the audio amp. Figures 4.3a/b show these audio amplifiers interfaced with a speaker and an audio input coming from the DAC. Both audio amplifiers require a network of passive components. Some hints on constructing your circuit are below.
     + When choosing RF and RI, make sure that the resulting gain less than or equal to one (gain = 2 \* RF / RI).
       - HINT: The range of the DAC matters less than there being an approximately linear relationship between the digital data and the speaker current. This means that we should choose a gain such that we remain in the linear region, and 1 is a safe choice to do this.
     + CI should be ceramic with a range of 0.1 µF to 0.47 µF.
     + CB should be tantalum with a range of 1.0 µF to 4.7 µF.
     + CS should be ceramic with a range of 0.1 µF to 0.47 µF.
     + The MC34119 and TPA731 can be powered with either +3.3V or +5V, however a larger supply voltage will typically have better performance and louder sound.
     + Refer to the data sheet for more information on passive sizing.
     + **NOTE: You will need to determine what to do with the shutdown signal.**

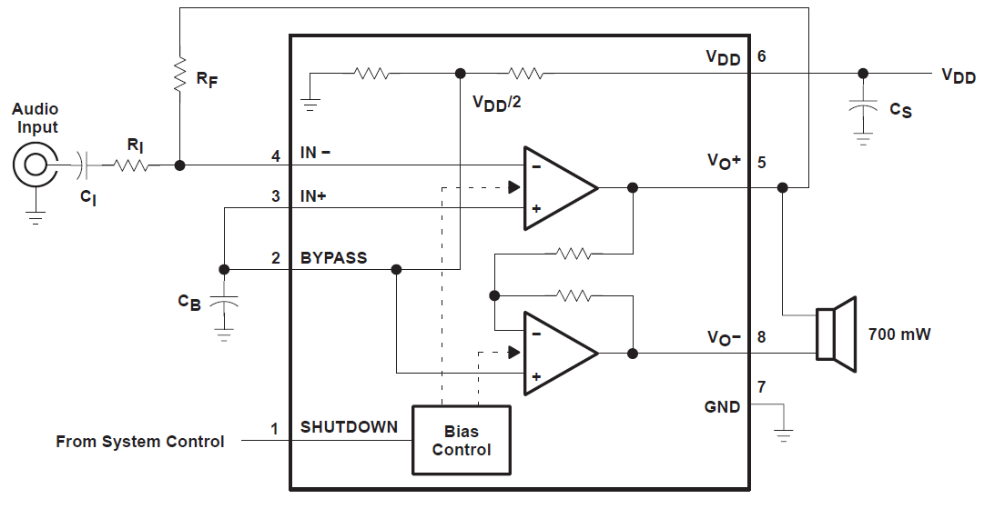


Figure 4.3a. The TPA731 is one way to convert DAC voltage into speaker current (ground SHUTDOWN to enable the amp). VDD is +5V VBUS.

A diagram of a circuit

Description automatically generated

Figure 4.3b. The MC34119P is another way to convert DAC voltage into speaker current (ground CD to enable the amp).

# Procedure

Procedure is performed during the W/TH lab session.

1. Write the C file for the DAC interface. Look very carefully at the four Freescale SPI modes possible. Only one of these four modes matches exactly the shape and polarity of the clock needed by the TLV5616/TLV5618. The function DAC\_Init() initializes the SSI protocol, and the function DAC\_Out() sends a new data value to the DAC. Create separate DAC.h and DAC.c files. Write a second low-level device driver for the two or three switches, creating separate Switch.h and Switch.c files.
2. Design and write the music device driver software. Create separate Music.h and Music.c files. Place the data structure format definition in the header file. For example, you could implement a Music\_Play() function that takes as an input parameter a pointer to a song data structure. Add minimally intrusive debugging instruments to allow you to visualize when interrupts are being processed.
3. Build the SSI/DAC hardware including voltage reference. Use simple main programs to debug the SSI/DAC interface. Experimentally measure the 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.
4. Write and debug the music system. Cut up a box, placing the speaker inside, and notice how much better it sounds. Pins 5 and 8 will have the sound signal, but these two signals will be 180 degrees out of phase (so the difference between pins 5 and 8 will be AC sound, with DC=0), as shown in Figure 4.4. Using a dual channel scope measure the outputs on pins 5 and 8 (like Figure 4.4).

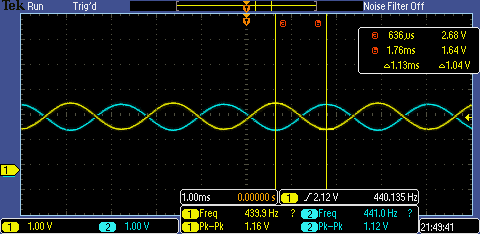


Figure 4.4. Two channel recording of pins 5 and 8. DC component is 2.14V, peak to peak is 1.16V (amplitude of the sound), and the frequency is 440.135 Hz.

## Deliverable 1

Using KiCad, create a schematic for your design. Include a screenshot in the space below.

## Deliverable 2

If you organized the system differently than those in Figure 4.10 and 4.11, then draw its data flow and call graphs. Otherwise, reproduce these figures. Figures 4.10 and 4.11 can be found in the hint section.

## Deliverable 3

Configure your system to your system to output a single note at one frequency. Then measure and include images of the following in your report:

* The time-domain output (Using an oscilloscope)
* Frequency-domain output (Using a spectrum analyzer, or the FFT mode of the oscilloscope)

Finally calculate the range, precision, and accuracy of the DAC.

Be sure to include in your report the data measurements and calculated parameters of the DAC.

## Deliverable 4

Using the spectrum (Frequency-domain output) from deliverable 3, calculate the SNR of your audio circuit. The SNR is the ratio of the sinewave output to the largest noise component. For more information see Figure 8.35 in the textbook. To calculate the SNR:

1. Find the peak in the spectrum created by the note and measure its amplitude
2. Find the highest non-signal peak in the spectrum
3. If the amplitudes of the peaks are measured in db, the difference of amplitude of these peaks is the SNR measured in db. If the amplitude of the peaks are in another unit such as volts/joules, divide the amplitude of these peaks to get the SNR as a ratio.

If you have the SNR in db, you need to convert it to an SNR as a ratio. Once you have a ratio, convert this into equivalent number of bits (ENOB) of the audio circuit. Record in your lab report the resolution (ENOB) of your audio circuit.

## Deliverable 5

Using debugging instruments, measure the maximum time required to execute the periodic interrupt service routine(s). Create a debugging profile to measure the percentage processor time required to play the song. Adjust the interrupt rate to guarantee no data is lost. Include in your report:

* The logic analyzer plots you used to profile the ISR(s) in your system.
* The maximum execution time of every ISR in your system.
* Calculate the percentage of time spent in ISR(s).

## Deliverable 6

Use your Lab 2 code to measure jitter of the DAC output. Note that you should measure jitter at the line just before your software outputs data via SPI. Include in your report:

* The measured jitter
* The reason(s) why the jitter is/isn’t zero.

## Deliverable 7

Remove the USB cable and carefully power your system using a lab power supply connected to the +5V line. Set the voltage to +5V and measure the required current to run the system with and without playing music, in addition measure the RMS voltage on the +5V line, which is a measure of power line noise. Take a measurement with and without the music playing. **Double check the positive and negative connections before turning it on. If you are at all unsure about this measurement, ask your TA for help.** Include in your report:

* The total current by the system when it is playing sound
* The total current by the system when it is **NOT** playing sound
* The RMS noise on the 5V line when it is playing sound.
* The RMS noise on the 5V line when it is **NOT** playing sound

## Deliverable 8 (15pts Extra Credit)

You may (for a +5% bonus) create multiple sine-waves at the same time. This way, you can play music containing melody and harmony. For this bonus you will use two sine-wave generators and add them together in hardware or software; be careful not to overflow and cause clipping. You will need three interrupts: one for outputting the sine-wave for the melody, one for outputting the sine-wave for the harmony, and a third to interpret the music (updating the frequencies and envelopes for the other two.) You will have to add the two sine-waves together in software.

You may (for another +5% bonus) create sine-waves with envelopes like Figure 4.8. To get extra credit, these envelopes must have shapes that sound pretty and are independent of pitch. Notice in Figure 4.8 that the decay slope of the envelopes for 330 and 523 Hz are the same. i.e. The envelopes are not frequency dependent. A sinusoidal envelope sounds like the bowing action on a violin.

You may (for another +5% bonus) develop a technique to support multiple notes with a single timer. In general, develop and implement a technique that allows more notes than timers.

Include in your report, which –if any—extra credit deliverables you have done. For each extra credit deliverable, tell what functions in your code were created/modified to achieve the effect.

# Lab Checkout

The lab checkout is performed during the M/T lab session.

You should be able to demonstrate the three functions as described in the requirements document. The TA will ask you to connect your DAC output to an oscilloscope and spectrum analyzer, and ask you questions about the frequency spectrum of your output. You should be prepared to discuss alternative approaches and be able to justify your solution.

# Lab Report

The lab report shall be submitted by the Friday after the second (W/Th) lab section.

You should complete the Lab04Report.docx file with your data and answers then submit the completed file to Canvas.

# Hint (Audio waveform basics)

When attempting to create a pleasing sound, it may help to understand the basics of audio signals. The sound we hear is nothing but pressure waves in the air, normally generated by movement and vibration.

Table 4.1 contains frequency values for the notes in one octave. The frequency of each note can be calculated by multiplying the previous frequency by 21/12. You can use this method to determine the frequencies of additional notes above and below the ones in Table 4.1. There are twelve notes in an octave, therefore moving up one octave doubles the frequency.

If you output a sequence of numbers to the DAC that form a sine-wave, then you will hear a continuous tone on the speaker, as shown in Figure 4.5. The loudness of the tone is determined by the amplitude of the wave. The pitch is defined as the frequency of the wave.

|  |  |
| --- | --- |
| Note | Frequency |
| C | 523Hz |
| B | 494Hz |
| B♭ | 466Hz |
| A | 440Hz |
| A♭ | 415Hz |
| G | 392Hz |
| G♭ | 370Hz |
| F | 349Hz |
| E | 330Hz |
| E♭ | 311Hz |
| D | 294Hz |
| D♭ | 277Hz |

Table 4.1. Fundamental frequencies of standard musical notes. The frequency for A is exact.

A diagram of a period

Description automatically generated

Figure 4.5. A sine-wave generates a pure tone.

Figure 4.6 illustrates the concept of timbre. You can define the type of sound by the shape of the voltage versus time waveform. Brass instruments have a very large first harmonic frequency.

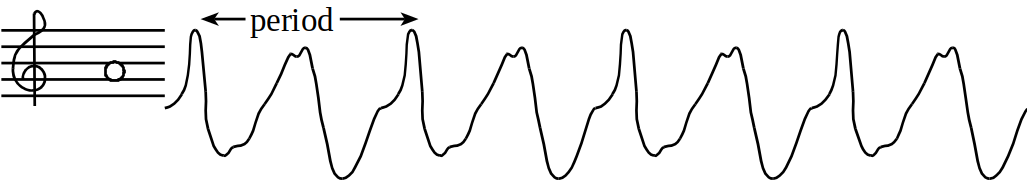


Figure 4.6. Timbre of a A node.

The tempo of the music defines the speed of the song. In 2/4, 3/4, or 4/4 music, a beat is defined as a quarter note. A moderate tempo is 120 beats/min, which means a quarter-note has a duration of 1/2 second. A sequence of notes should be separated by pauses (silences) so that each note is heard separately. The envelope of the note defines the amplitude of the note over time. A very simple envelope is illustrated in Figure 4.7.

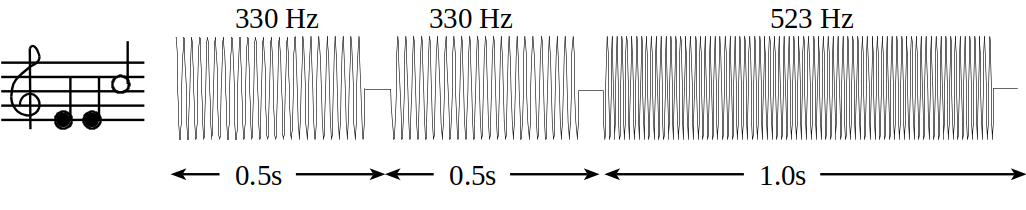


Figure 4.7. You can control the amplitude, frequency and duration of each note (not drawn to scale).

The smooth-shaped envelope, as illustrated in Figure 4.8, causes a less staccato and more melodic sound. The ARM Cortex M4 has plenty of processing power to create these types of waves.

A drawing of a cone

Description automatically generated

Figure 4.8. The amplitude of a plucked string drops exponentially in time.

A chord is created by playing multiple notes simultaneously. When two piano keys are struck simultaneously both notes are created, and the sounds are mixed arithmetically. You can create the same effect by adding two waves together in software, before sending the wave to the DAC. You can produce this effect by using two interrupts and adding two waves together in software. Figure 4.9 plots the mathematical addition of a 262Hz (low C) and a 392Hz sine wave (G), creating a simple chord.

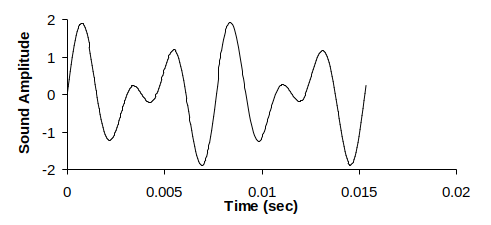


Figure 4.9. A simple chord mixing the notes C and G.

# Hint (System Call Graphs)

1. A “syntax-error-free” software is required as preparation. The TA will check off your listing at the beginning of the lab period. You are required to do your editing before lab. The debugging will be done during the lab. Document clearly the operation of the routines. Figure 4.10 shows one possible data flow graph of the music player.

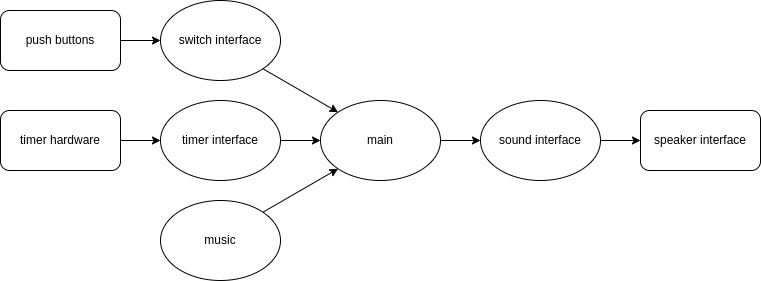


Figure 4.10. Data flows from the memory and the switches to the speaker.

Figure 4.11 shows a possible call graph of the system. Dividing the system into modules allows for concurrent development and eases the reuse of code.

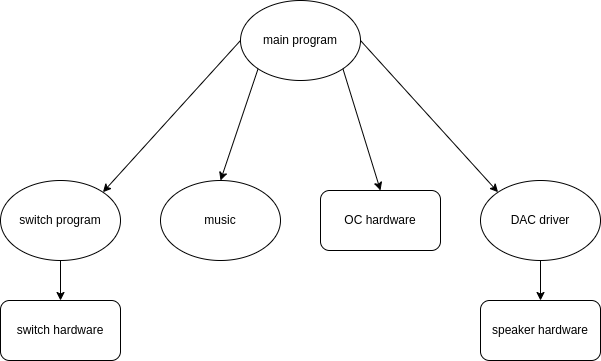


Figure 4.11. A call graph showing the three modules used by the music player.