Lab 5h Music Player and Audio Amp

This laboratory assignment accompanies the book, Embedded Microcomputer Systems: Real Time Interfacing, Third edition, by Jonathan W. Valvano, published by Cengage, copyright © 2010.

Goals
• DAC conversion,
• SPI interface,
• Design a data structure to represent music,
• Develop a system to play sounds.

Review
• Search http://www.maxim-ic.com/ for the MAX539 12-bit DAC data sheet,
• Search http://www.ti.com/ for a data sheet on the TLV5616 12-bit DAC,
• Valvano Section 4.15.3 on periodic interrupts,
• EE445L lecture notes on SPI interfacing,
• 9S12DP512BoardInformation.pdf in Lab manual web site about SPI ports
• Look up bits 4,5,6 in the MODRR register in the 9S12DP512 data sheet
http://users.ece.utexas.edu/~valvano/Datasheets/MC9S12DP512.pdf
• Valvano Section 11.4.1 on DAC parameters,
• Valvano Section 11.4.6 on waveform generation.

Starter files
• **`OC_DP512.zip`** project, Excel files starting with **`dac_`**.

Background

Most digital music devices rely on high-speed digital to analog converters (DAC) to create the analog waveforms required to produce high-quality sound. In this lab, you will create a very simple music player that illustrates a typical application of the DAC. Your goal is to play your favorite song. First, you will interface a 12-bit DAC, and use it to create a sine-wave output. In particular, you will interface either a TI TLV5616 or a Maxim MAX539 12-bit DAC to an SPI port, as shown in Figure 5.1. Please refer to the DAC data sheets for the synchronous serial protocol. During testing, the output of the DAC will be connected to a voltmeter, an oscilloscope or a spectrum analyzer. You are allowed to use any DAC chip you want, as long as it runs on a single +5V supply and has an SPI interface. Many DACs, such as the MAX539 and TLV5616, require a reference voltage. A stable 2.50V reference can be created using a reference chip such as the REF03 or LM4041C. The LM4041CILPR is an adjustable shunt reference that can be powered from the +5V supply, and requires three external resistors to create the 2.50 V reference. Look up in the MAX539/TLV5616 data sheet to find how much current the DAC needs on its Refin input. In the data sheet you will find the input impedance R_{in} of the Refin pin, you can calculate this load current $I_L = 2.5V/R_{in}$. Next, look up page 8 of the LM4041CILPR data sheet to find I_Z (80 µA) and V_{REF} (1.233V). Select R1 and R2 to set the reference output $V_Z = V_{REF}(1+R1/R2)$ to 2.5V. The Rs resistor in Figure 5.1 (data sheet page 15) sets the available current for the shunt reference. Make Rs smaller than $(5-V_Z)/(I_L+I_Z)$. See Figure 15 and Equation 1 of the LM4041CILPR data sheet. Dout on the MAX539 can be left not connected. The TLV5616 has no data output, and the data sheet shows which pins to use for an SPI interface.

The second step is to design a low-level device driver for the DAC. For the MAX539/TLV5616, two 8-bit SPI frames are required to set the DAC output. Next, you will design a data structure to store the sine-wave. The main program will input from switches and allow the operator to select from three pre-defined frequencies.

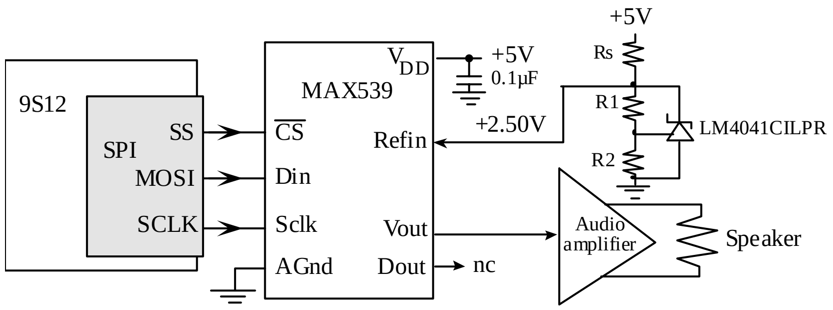

Figure 5.1. Block diagram of the DAC interface (the TLV5616 is similar).

If you output a sequence of numbers to the DAC that form a sine-wave, then you should be able to see the wave on an oscilloscope, as shown in Figure 5.2. This measured data was collected using a 12-bit DAC

(MAX539/TLV5616) having a range of 0 to +5 V. The plot on the left was measured with a digital scope (without a speaker attached). The plot on the right shows the frequency response of this data, plotting amplitude (in dB) versus frequency (in kHz). This measured waveform is approximately 2.5+2.5*sin(2π262πt) volts. The two peaks in the spectrum are at DC and 262 Hz (e.g., 20*log(2.5)= 8.0 dB).

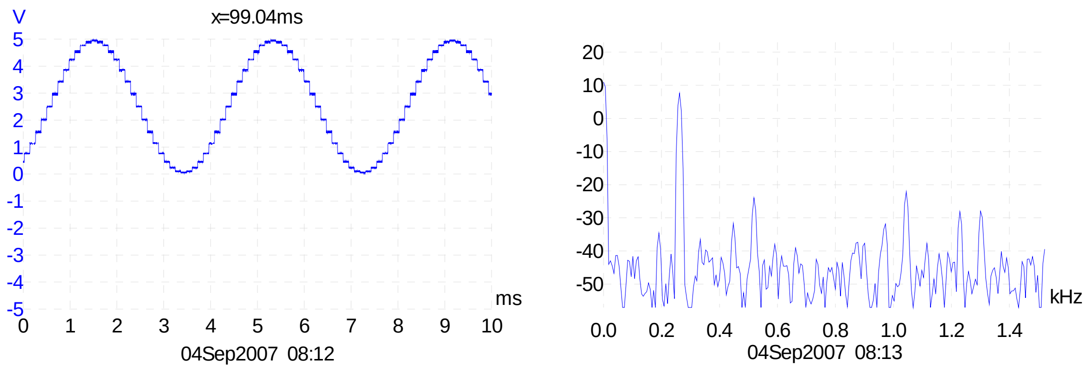

Figure 5.2. Experimental data of a 262 kHz sine-wave. The plot on the right is the Fourier Transform (frequency spectrum dB versus kHz) of the data plotted on the left.

The next step is to convert the DAC analog output to speaker current using a current-amplifying audio amplifier, as shown in Figure 5.3. It doesn't matter what range the DAC is, as long as there is an approximately linear relationship between the digital data and the speaker current. To do this you will have to run the amplifier in its linear range. 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. 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 are some of the hardware factors. Software factors include the DAC output rate and the complexity of the stored music data. Consider using the MC34119 when designing the audio amp, choosing Rf and Ri so the gain is one or less than one.

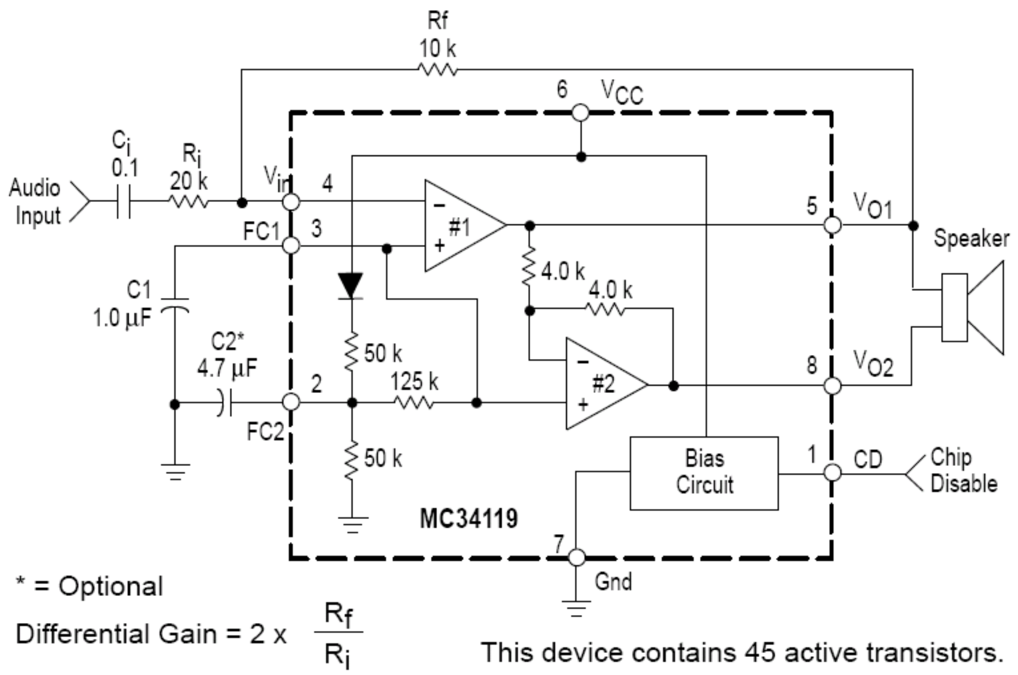

Figure 5.3. The MC34119 is one way to convert DAC voltage into speaker current (ground CD).

You will design a data structure to store the sound waveform. You are free to design your own format, as long as it uses a formal data structure (i.e., `struct`). Compressed data occupies less storage, but requires runtime calculation. On the other hand, a complete list of points will be simpler to process, but requires more storage than is available on the 9S12. Next, you will organize the music software into a device driver. Although you will be playing only one song, the song data itself will be stored in the main program, and the device driver will perform all the I/O and interrupts to make it happen. 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. The last step is to write a main program that inputs from three binary switches and performs the three public functions.

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 Figures 5.2 and 5.4. The **loudness** of the tone is determined by the amplitude of the wave. The **pitch** is defined as the frequency of the wave. Table 5.1 contains frequency values for the notes in one octave.

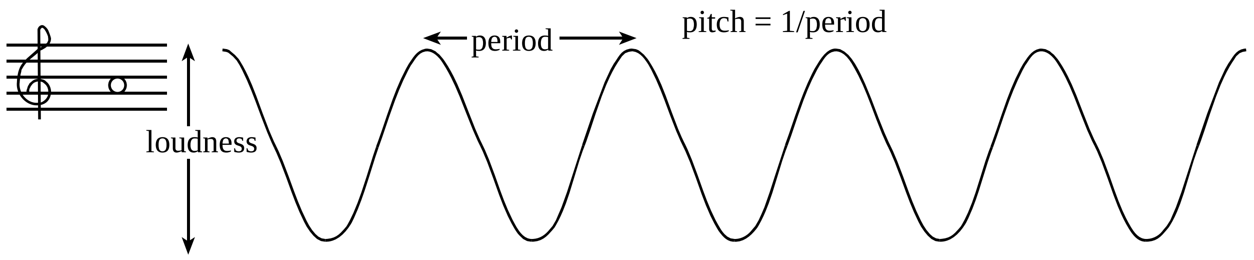

Figure 5.4. A sine-wave generates a pure tone.

Note	frequency
C	523 Hz
B	494 Hz
B^b	466 Hz
A	*440 Hz*
A^b	415 Hz
G	392 Hz
G^b	370 Hz
F	349 Hz
E	330 Hz
E^b	311 Hz
D	294 Hz
D^b	277 Hz
C	262 Hz

Table 5.1. Fundamental frequencies of standard musical notes. The frequency for 'A' is exact.

The frequency of each note can be calculated by multiplying the previous frequency by $\sqrt[12]{2}$. You can use this method to determine the frequencies of additional notes above and below the ones in Table 6.1. There are twelve notes in an octave, therefore moving up one octave doubles the frequency. Figure 5.5 illustrates the concept of **instrument**. 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.

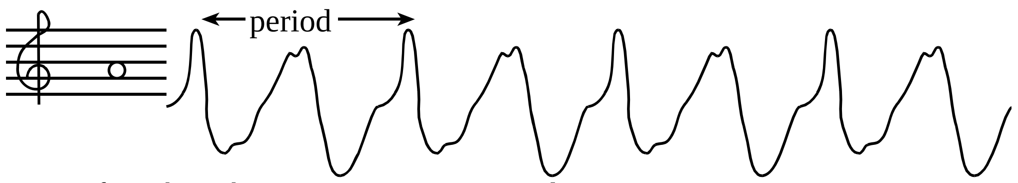

Figure 5.5. A waveform shape that generates a trumpet sound.

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 ½ 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 versus time. A very simple envelope is illustrated in Figure 5.6. The 9S12 has plenty of processing power to create these types of waves.

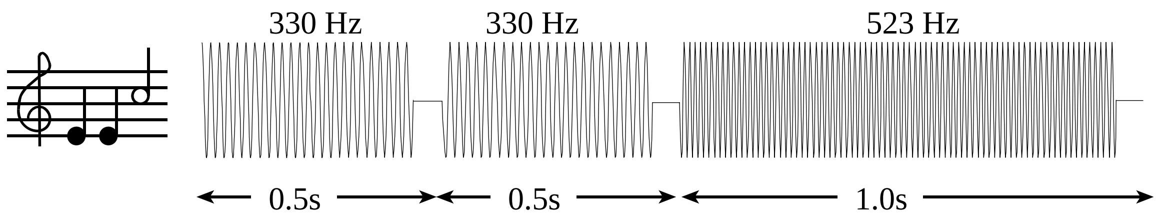

Figure 5.6. You can control the amplitude, frequency and duration of each note (not drawn to scale).

The smooth-shaped envelope, as illustrated in Figure 5.7, causes a less staccato and more melodic sound. This type of sound generation may be difficult to produce in real-time on the 9S12.

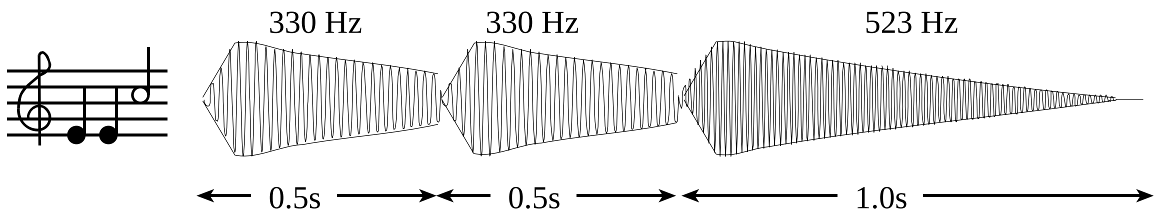

Figure 5.7. The amplitude of a plucked string drops exponentially in time.

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 5.8 plots the mathematical addition of a 262 Hz (low C) and a 392 Hz sine wave (G), creating a simple chord.

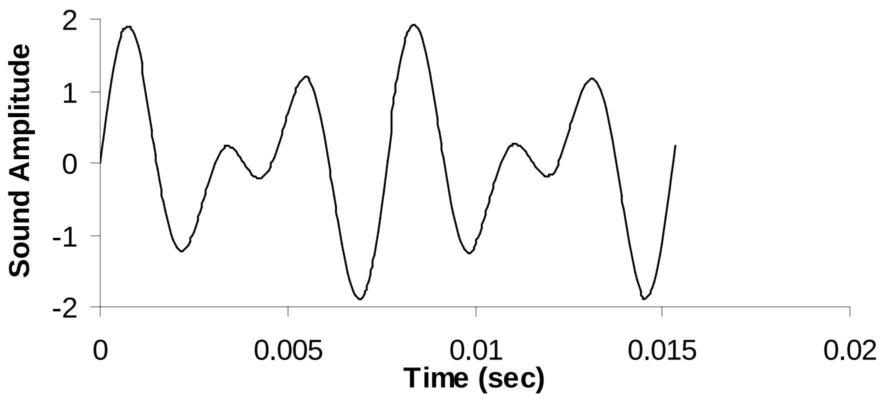

Figure 5.8. A simple chord mixing the notes C and G.

Preparation (do this before your lab period)

1. Draw the circuit required to interface the DAC to the 9S12 SPI port. Include signal names and pin numbers. The bypass capacitor on the +5 V supply of the DAC can be any value from 0.01 µF to 0.22 µF. Draw the circuit required to interface three push button switches. Design the audio amplifier that runs on the +5V power from the board.

2. Write a low-level device driver for the SPI interface. Include two functions that implement the SPI/DAC interface. The function **`DAC_Init()`** initializes the SPI 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 three switches, creating separate **`Switch.h`** and **`Switch.c`** files.

3. 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.

4. Write a main program to run the entire system.

A "syntax-error-free" hardcopy listing for the 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 lab. Document clearly the operation of the routines. Figure 5.9 shows one possible data flow graph of the music player.

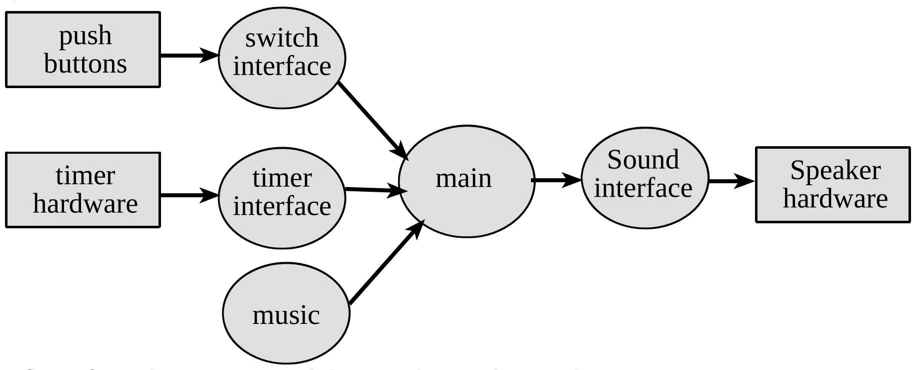

Figure 5.9. Data flows from the memory and the switches to the speaker.

Figure 5.10 shows a possible call graph of the system. Dividing the system into modules allows for concurrent development and eases the reuse of code.

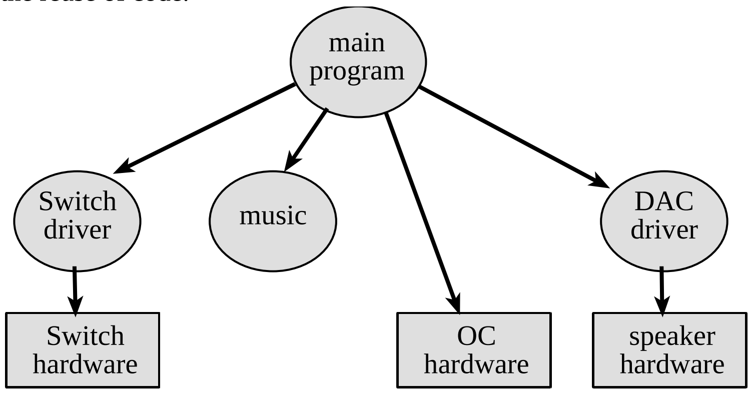

Figure 5.10. A call graph showing the three modules used by the music player.

Procedure (do this during your lab period)

1. Use simple main programs to debug the SPI/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.
2. Using an oscilloscope and spectrum analyzer, measure the time-domain and frequency-domain outputs from your system at one frequency, like Figure 5.2.
3. Using debugging instruments, measure the maximum time required to execute the periodic interrupt service routines. In particular, create s debugging profile to measure the percentage CPU time required to play the song. Adjust the interrupt rate to guarantee no data are lost.
4. Debug the music system.
5. Remove the wall wart power cable and carefully power your system using a lab power supply (not on the +5V pin on the protoboard but the 2-pin male header labeled J1-PWR on the board itself). Set the voltage to +6V and measure the required current to run the system. 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.

Deliverables (exact components of the lab report)

A) Objectives (1/2 page maximum)

B) Hardware Design

Detailed circuit diagram of all hardware attached to the 9S12 (preparation 1)

C) Software Design (a pdf file of the software should be uploaded to blackboard)

Draw pictures of the data structures used to store the sound data
If you organized the system different than Figure 5.9 and 5.10, then draw its data flow and call graphs

D) Measurement Data
Show the data and calculated resolution, range, precision and accuracy (procedure 1)
Show the experimental response of DAC (procedure 2)
Show the results of the debugging profile (procedure 3)
Measurements of current required to run the system, with and without the music playing

E) Analysis and Discussion (give short answers to these questions)
1) Briefly describe three errors in a DAC.
2) Using setup and hold, explain how you selected the maximum SPI frequency.
3) How is the frequency range of a spectrum analyzer determined?
4) Why did we not simply drive the speaker directly from the DAC? I.e., what purpose is the MC34119?

Checkout (show this to the TA)

You should be able to demonstrate the three functions **`Rewind`**, **`Play`** and **`Stop`**. 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.

A pdf file of your software must be uploaded, and graded for style at a later time.

Do you want your machine to sound better than everyone else's? Extra credit bonus

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 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 similar to Figure 5.5. To get extra credit, these envelopes must have shapes that sound pretty and are independent of pitch. Notice in Figure 5.5 that the decay slope of the envelopes for 330 and 523 Hz are the same. A sinusoidal envelope sound like a violin.

Simple Music Theory	http://library.thinkquest.org/15413/theory/theory.htm
Free sheet music	http://www.8notes.com/piano/

Do an internet search on "midi format specification", they may give you ideas on how to encode music digitally.