

Lab 2h Audio Recording and Playback

This laboratory assignment accompanies the book, Embedded Microcomputer Systems: Real Time Interfacing, 2nd edition, by Jonathan W. Valvano, published by Thomson Publishing, copyright © 2006.

Goals

- Study ADC conversion, Nyquist Theorem, Valvano Postulate,
- Develop an audio recording system using an electret microphone,
- Develop an audio playback system using a DAC, amplifier and speaker,
- Design and implement various analog and digital filters.

Review

- Operation of the ADC system in the 9S12C32/9S12DP512 data sheet,
- Data sheets on the TLC2272, MAX6225, MAX539, MC34119, electret microphone
- Valvano Chapters 11,12,15 on analog amplifiers, analog low pass filters, ADC's, data acquisition systems, and digital filters.
- Sections 12.4 and 12.5, on noise analysis and DAS

Starter files

- OC, ADC, PWM, DFtest, lpf.xls, EE345L Labs 5 and 6

Background

This experiment will use a microphone and an ADC converter to record audio. A DAC and speaker allow sounds to be played back. You are free to choose any microphone, speaker, ADC precision, DAC precision, and sampling rate you wish. However, I strongly suggest using a 32-ohm speaker and single-supply rail-to-rail electronics so that you run the entire system on the +5V supply from the 9S12 board.

Figure 2.1 shows one possible data-flow graph of the audio recording and playback system. The electret microphone is a low-cost low-fidelity transducer appropriate for voice recording. An analog band-pass filter will be used to improve signal to noise ratio and to prevent aliasing. Hardware triggering is used to start the ADC (PWM output connected to PAD7 input). The electret microphone converts sound to a voltage, the analog amp and filter convert voltage to 0 to +5V analog signal, and the ADC converts analog voltage to an integer. The ADC ISR reads one digital sample, and data is passed to the foreground using a simple buffer called **InBuffer**. You can add an optional digital filtering to improve signal to noise ratio. The sound buffer can be saved as a file on the solid state disk using the file system developed in the last lab. Your system should have the option of playing back data in real time (input directly to output), or to playing back recorded data saved previously on the disk. You will choose a sampling rate slow enough so data can be written as a stream on the serial EEPROM. You are free to implement either 8-bit or 10-bit analog to digital recording.

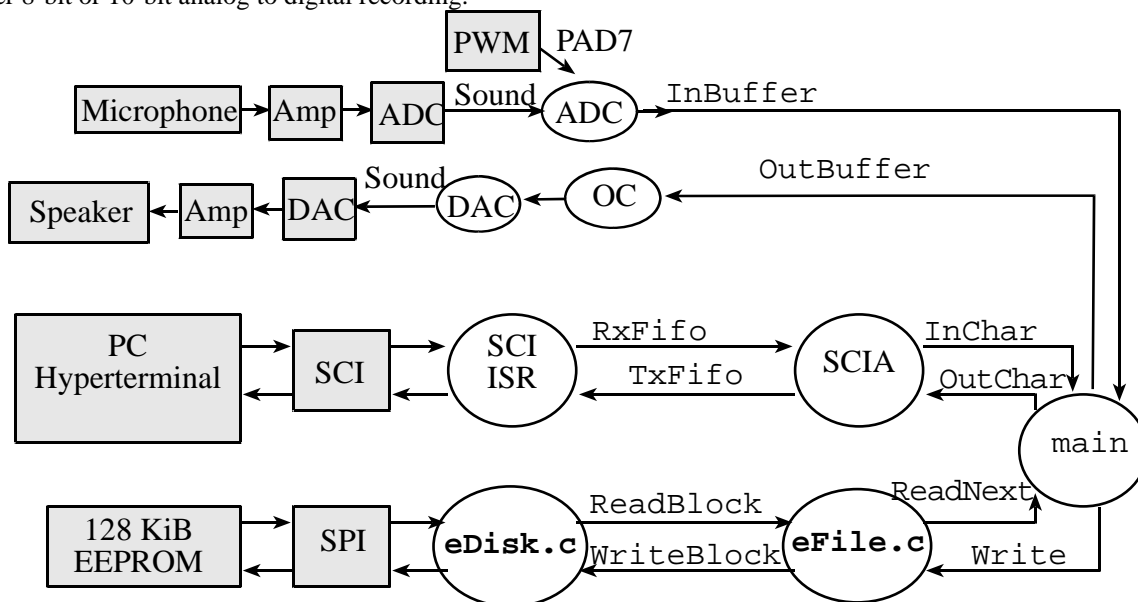


Figure 2.1. Data-flow graph of the data acquisition system.

Figure 2.2 shows one possible call-graph. Each of the modules has separate header and implementation files. Notice the main program does not directly access the ADC DAC SCI or SPI hardware.

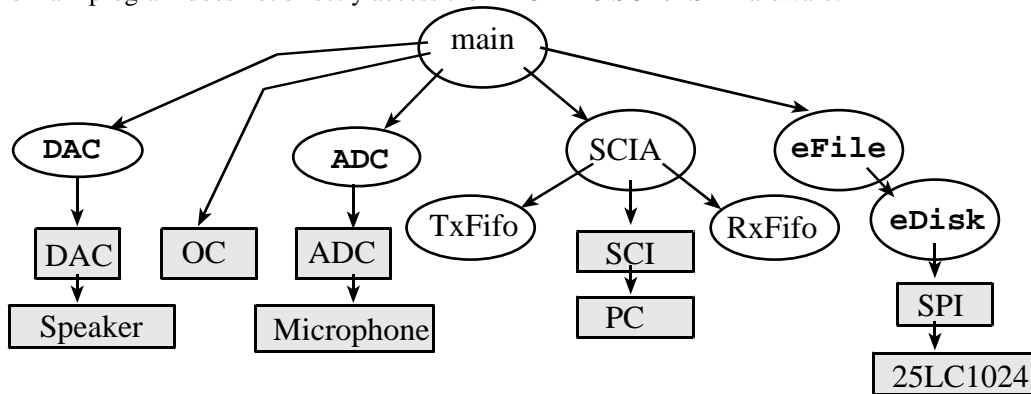


Figure 22.2. One possible call-graph of the data acquisition system.

Preparation (do this before your lab period)

1. You will design an analog interface between the electret microphone and the ADC. The DC component will be removed by a high-pass filter, but you will use a 2.50V reference chip (REF03 or MAX6225) to create an analog ground at 2.50 V relative to the digital ground. This way, the sound information, $v(t)$ can be presented to the ADC as a voltage equal to $2.50+v(t)$. You will design a low pass filter to reject frequencies above $\frac{1}{2} f_s$. You will want an amplifier gain so that the ADC swings from about 1 to 4 volts for normal speaking voices. For now, you can pick any gain around 100. You will adjust the gain later during the testing portion of the lab.

2. Write a simple program to test the real-time data acquisition. Set the AFFC, ETRIG, and ASCIE bits in the ATDCTL2 register. If ETRIGLE and ETRIGP bits are clear, an ADC sample will be started on the falling edge of PAD7. Write software that creates a digital squarewave at the desired sampling rate and connect this signal to PAD7. Setting the ASCIE bit will arm the ASCIF flag, so an interrupt occurs after the ADC sample sequence is complete. With AFFC=1, reading the result register will automatically clear the ASCIF flag, acknowledging the interrupt. You write to the ATDCTL5 register only once, in the initialization, specifying the right/left mode and the channel number. Using hardware ADC synchronization requires two 9S12 pins (a PWM output and a PAD7 input), but the resulting sampling jitter is extremely low. In fact, the sampling jitter will only depend on the stability of squarewave. You can use a PWM channel to create this squarewave, which requires no software overhead, and has a clock jitter less than the E clock itself. The system samples the ADC in real-time using interrupt synchronization and fills a large RAM-based buffer. This program runs continuously filling the buffer over and over. Don't add any SCI I/O, so you can use the debugger to test this simple system. Add debugging instruments so that the operation of the real-time 9S12 software can be observed on a logic analyzer or oscilloscope. With this technique the ADC is started by hardware (falling edge of PAD7). As long as the ADC ISR runs before the next sample is requested, the sampling jitter depends only on this hardware clock.

3. Rebuild the hardware from EE345L Labs 5 and 6. In particular, design the interface between the 9S12 and a DAC (EE345L lab 5). Design the analog amplifier between the DAC and the 32-ohm speaker (EE345L lab 6).

4. Extend the software system in preparation 2 so that it also outputs each sample to the DAC. Use one periodic output compare interrupt so that both the input sampling and output playback are in real time.

5. Extend the Lab 1 system (add additional commands) so audio data can be saved as files. When recording data, you will create a file, open it for writing, then continuously stream the data directly into the disk until the disk is full or the operator stops the recording. Add commands so data previously recorded can be played back on the speaker. You can either play the sound buffer back once or loop it so that it continuously plays the same buffer over and over. The following commands illustrate the types of features you need to develop (feel free to implement the syntax however you wish) (in addition to the existing commands from Lab 1):

- r m open the file called "m", and stream audio data into the file
- o y playback the contents of audio file "y"
- = copies input sound directly to output (this will cause feedback)
- s stops the output sound or stop sound recording

Lab 2h Audio Recording and Playback

Lab 2h.3

You must use the file system developed in Lab 1, rather than saving sounds as unformatted data using just the **eDisk** functions.

Procedure (do this during your lab period)

1. *Gain adjustment:* Perform these tests before connecting the circuit to the 9S12. Connect the microphone with its amplifier and observe the analog output on an oscilloscope. Verify the analog signal is centered around 2.5 V relative to digital ground. Adjust the gain so that the voltage is about 1 to 4 volts for normal speaking.

2. *Dynamic analog circuit test:* Perform these tests before connecting the circuit to the 9S12. Disconnect the microphone, and connect a sine-wave signal generator in its place. Make sure the voltage level of the signal generator is within range, so that the inputs and outputs of your analog circuit are not saturated (approximately $2.5 + 2 \sin(2\pi ft)$ at the ADC). Record the sine-wave amplitudes of the input and output voltages. Collect measurements for about ten different frequencies. Make sure you choose frequencies above the LPF cutoff and below the HPF cutoff. Calculate the gain at each frequency. Plot the gain versus frequency response of your audio input circuit.

3. *Real-time data acquisition test.* Use the simple programs written in preparations 2 and 4 to test the data acquisition components of the system. Again, generate a sinusoidal waveform (approximately $2.5 + 2 \sin(2\pi ft)$ at the ADC) with an adjustable frequency, **f**. First connect the analog waveform to a scope and verify the voltage range is between 0 and +5V. If you connect the signal generator to the analog amplifier input (rather than directly to the ADC), you are less likely to damage the 9S12. **VOLTAGES OUTSIDE THE 0 to 5V RANGE WILL DAMAGE THE 9S12.** Disconnect the speaker to prevent feedback. Perform this test dumping data into a memory buffer without SCI I/O, so you can use all the debugging features of Metrowerks. Stop the program at the end of a fill-cycle and capture the digital data using the debugger. Plot the collected digital data using a program like MatLab or Excel. Compare qualitatively three sets of data

Analog signal at the ADC input measured with a scope and a spectrum analyzer

Collected digital data plotted with a program like MatLab or Excel

Analog signal at the DAC output measured with a scope and a spectrum analyzer

Using MatLab or Excel, perform an FFT on the collected digital data and discuss the frequency components that exist in the signal. Compare the spectrum at the ADC input (measured with a spectrum analyzer), the spectrum of the digital samples (MatLab or Excel), and the spectrum at the DAC output (measured with a spectrum analyzer). Perform the measurements at three frequencies, one below $1/10 f_s$, one about $1/4 f_s$ and one above $1/2 f_s$.

4. *Test the audio recording and playback system.* Test the overall system. Take repeated measurements of the period of the squarewave on PAD7 using a high speed scope to PAD7 and estimate the ADC sampling jitter.

Deliverables (exact components of the lab report)

A) Objectives (1/2 page maximum)

B) Hardware Design

Circuit diagram of the microphone, speaker and EEPROM interfaces

C) Software Design (printout of these software components)

1) Low level ADC interface (**ADC.c** and **ADC.h** files)

2) Low level DAC interface (**DAC.c** and **DAC.h** files)

3) Main program used to test the data acquisition (preparations 2 and 4)

4) Main program used to record and playback sound (preparation 5)

D) Measurement Data

1) Gain versus frequency of the input analog amplifier (procedure 2)

2) Three analog waveforms and three spectrums below $1/10 f_s$ (procedure 3)

3) Three analog waveforms and three spectrums at about $1/4 f_s$ (procedure 3)

4) Three analog waveforms and three spectrums above $1/2 f_s$ (procedure 3)

5) Estimate the sampling jitter and compare to the approach used in EE319K and EE345L (procedure 4)

E) Analysis and Discussion (1 page maximum)

Checkout (show this to the TA)

You should be able to demonstrate the proper operation of audio recording and playback system. Bring your recorded data, and be prepared to discuss the Nyquist Theorem and spectrum analysis.

Hints

- 1) This is a long lab with many parts, so start early.
- 2) Don't try to complete the experiment in one fell swoop. Debug the ADC and DAC modules without the SCI interpreter; this way, you can use the Metrowerks debugger.
- 3) See Valvano Chapter 15 for information about digital filters.
- 4) Please do not use regular op amps that require ± 12 V supplies. An advantage of the single supply rail-to-rail op amp is that the output of your analog amplifier will remain within the 0 to +5V ADC range at all times.
- 5) You are going to need PM1 and PM0 to implement the CAN network in the next lab, so please leave these two pins free.
- 6) A stable 2.50V reference can be created using a reference chip such as the REF03 or MAX6225. The REF03 reference can be powered from the +5V supply, but the MAX6225 requires a supply voltage (called V_{in}) above 8 V. To use the MAX6225 you will need to connect MAX6225 V_{in} (pin 2) directly to the wall-wart (pin 40, labeled V_{in} on 9S12C32 docking module.)
- 7) Both the DAC and the EEPROM require an SPI interface. There are two approaches that could work in this lab. Since the two devices are not accessed at the same time you could connect MOSI from 9S12 to both serial inputs, connect SCLK output to both clock inputs, but use separate 9S12 outputs to connect to the two CS inputs. This way you can activate one device at a time. If the DAC is accessed in the background and the EEPROM in the foreground, you will need to make the foreground accesses to the EEPROM atomic (so the DAC outputs do not occur in the middle of EEPROM outputs). Disabling interrupts will cause time jitter on the DAC outputs. The second approach is to use SPI for the EEPROM, and use three regular digital outputs for the DAC. You will have to create the CS, clock, and data signals manually by outputting high and low values to the three output pins.
- 8) Expect a lot of feedback noise when the ADC input is fed directly to the speaker output. Higher quality recordings will be possible if the ADC is sampled without simultaneously activating the DAC. Thus, when performing procedure part 3, you should collect the data with the speaker disconnected.
- 9) This is an optional part. At each sample time you could sample the ADC four times and add the four results, giving you a 12-bit number compatible with the 12-bit DAC. This method is appropriate only in situations when the noise is above the ADC resolution. If the noise has a zero mean, then (according to the Central Limit Theorem) the sum of four samples will have a better S/N ratio than a single sample alone. If you were to take four samples of a signal without noise, then you would get the exact same result each time, and the sum of the four samples would have the exact same S/N ratio as a single sample.
- 10) There are lots of ways to store 10-bit data into as bytes on the disk (you can do it however you wish).
 - Option 1. Discard two bits and simply store each sample as an 8-bit number
 - Option 2. Break 10 bits into two 5-bit parts. Store 0x40+top 5 bits, Store 0x60+bottom 5 bits
 - Option 3. Pack 4 ADC samples (40 bits) into 5 bytes (40 bits)
 - Option 4. Store the difference between samples as an 8-bit signed number (-128 to +127)
- 11) Metrowerks has a limit of 32 files. If your project goes over the 32 file limit, then consider combining multiple modules into larger files.
- 12) Think about which is the more appropriate to use in this case:
 - 1) SCI using busy-wait synchronization or
 - 2) SCI using interrupt synchronization.The interrupt implementation reduces the possibility of lost SCI input data, but causes time jitter on the DAC process. There is no definitive answer, there are advantages of each. Which is more important?
- 13) Please avoid using an 8-ohm speaker (rather use a 32-ohm speaker). The system can drive an 8-ohm speaker, but the regulator on the docking module gets hot. Similarly, I suggest you do NOT use external power sources for this class (until we get to the robot competition), because you could make a mistake (wrong voltage or backwards) and destroy the board. If you want high quality sound output, you could use headphones.
- 14) Think about when to enable interrupts at the beginning of execution. I like to initialize all modules, then enable interrupts. Worry about critical sections when using shared global memory or shared I/O devices.
- 15) The data sheet says writing a 1 to SCF should clear SCF. However, this C command did not seem to clear SCF, **ATDSTAT0 = 0x80**; Fortunately, fast clear mode automatically clears SCF simply by reading the ADC data.