Part d) Design and write the piano keyboard device driver software. These routines facilitate the use of the four piano keys. Include at least two functions that implement the piano keyboard interface. For example, you could implement the function **Piano_Init()** to initialize the switch inputs, and the function **Piano_In** that returns a logical key code for the pattern of switches that are pressed. Place all code that directly accesses the four switches in the **Piano.c** code file. The **Piano.h** header file contains the prototypes for public functions. Add comments that describe what it does in the **Piano.h** file and how it works in the **Piano.c** file.

Part e) Design and write the sound device driver software. The input to the sound driver is the pitch of the note to play. SysTick interrupts will be used to set the time in between outputs to the DAC. Add minimally intrusive debugging instruments to allow you to visualize when interrupts are being processed. Include at least two functions that implement the sound output. For example, you could implement the function **Sound_Init()** to initialize the data structures, calls **DAC_Init**, and initializes the SysTick interrupt. You could implement a function **Sound_Play(note)** that starts sound output at the specified pitch. Once the software calls play, the sound continues until the software explicitly calls the driver again to change pitch or turn off the sound. In order to turn the sound off, you could implement a third function, **Sound_Off()** or simply call the play function with a special parameter, e.g., **Sound_Play(0)**. Place all code that implements the waveform generation in the **Sound.c** code file. The **Sound.h** header file contains the prototypes for public functions. This driver contains the SysTick ISR, and since the ISR is private to the driver, do NOT add a prototype for the ISR in the header file. Add comments that describe what the software does in the **Sound.h** file and how the software works in the **Sound.c** file. To set the pitch, the play function will write to the **RELOAD** register, without completely initializing SysTick.

Part f) Write a main program to implement the four-key piano. Make heartbeats on PF3, PF2, PF1 so you can see your program is running. Debug the system first in the simulator then on the real board using the TExaS oscilloscope. In the simulator you can add the heartbeats to the logic analyzer. When debugging on the real board you can observe the heartbeats by looking at the color-LED, or connecting PD3 and using the TExaS scope. When no buttons are pressed, the output should be quiet and the DAC output should be 0. Each of the four switches is associated with a separate pitch (C, D, E, G) as listed in Table 13.2. Only one button should be pressed at a time. The sound lasts until the button is released. Figure 13.4 shows a data flow graph of the Lab 13 system.

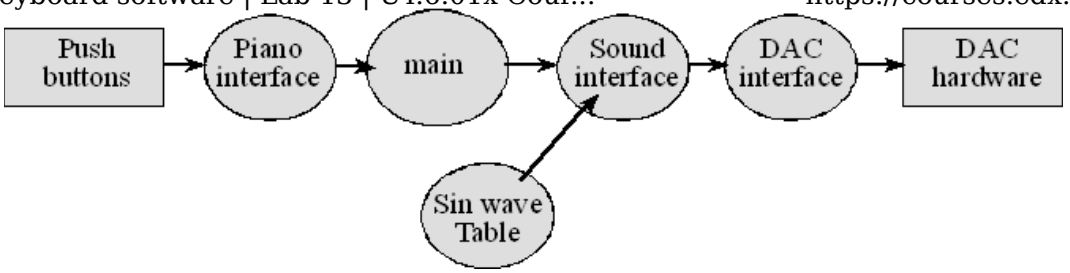

Figure 13.4. Data flows from the memory and the switches to the speaker.

Figure 13.5 shows a possible call graph of the Lab 13 system. Dividing the system into modules allows for concurrent development and eases the reuse of code.

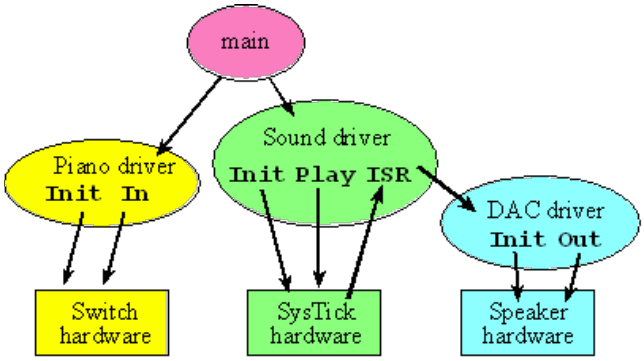

Figure 13.5. A call graph showing the three modules used by this digital piano. Notice the SysTick hardware calls the SysTick ISR.

This lab is much too complex to type it all in and expect the system to run. Since there are four modules (DAC, piano, sound, and main) it makes sense to debug each module separately. Start with the DAC and then the piano, because they are lowest level. As with the other labs, you should first complete Lab 13 in simulation. During the simulation grading I will automatically set the inputs and check your outputs. Simulation grading checks for the approximate frequency but not the shape of the wave. However, the real board grader does evaluate the shape of the wave, so you will need to implement a sinusoid. As always, watch the command window to see what the grader is looking for.

GRADING IN SIMULATION

DR. JONATHAN VALVANO: Hi, in this video, we'll

Help

0:00 / 1:57 1.0x

show you how to earn your grade in Lab 13 Simulation mode.

We begin in edX and capture the four digit number, Control-C, Copy.

And then we go over to Keil, and since we're running in Simulation mode,

we want to make sure the options are using The Simulator.

So there's the Simulator.

And we build it.

And then we debug.

And now we're running in the debugger in Simulation mode.

DR. RAMESH YERRABALLI: We can use the Logic Analyzer to view the DACOUT.

And we do that by populating it.

Here we type in DACOUT, and hit Enter.

And we select it.

Make sure it's Analog