EECE 544: Embedded Systems Design
Lab 4
Digital Piano using DAC

1. Pre-Lab

- Download, unzip, open, compile and run **Piano_Lab4** into your computer.
- Read section 10.3 in your book.

2. Objective

The objective of this lab is to:

- Learn how to convert digital signal to analog by building a DAC circuit.
- Learn how digital signal can be used to represent sound and music.

3. Overview

Digital music devices use high speed DAC converters to create sound. In this lab you will be creating a simple sound generating system using a DAC circuit. The final system should be capable of creating four different notes with 4-input switches simulating piano keys. The piano keys input will be tested using a 5-bit binary-weighted DAC, which will convert the digital output from 5 PINs to analog output signal using a resistor network. The output from the DAC will be connected to headphones where you can listen to the sounds created by your software. Figure 1 shows a block diagram of the final system.

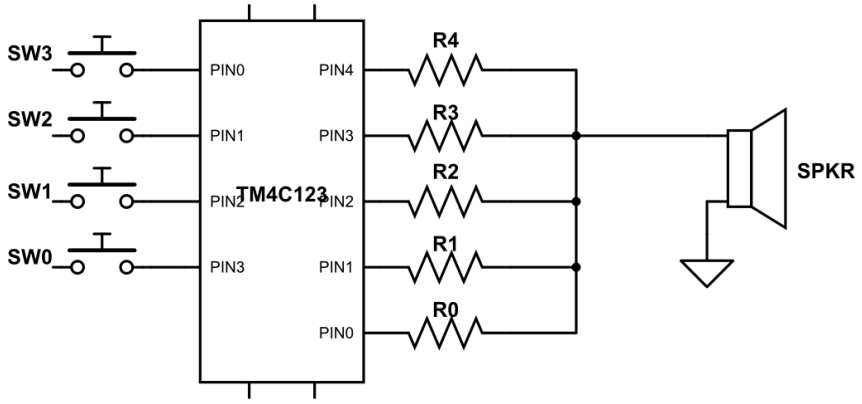

Figure 1

4. Procedure

Part 1: 5-bit binary-weighted DAC

1. Design a 5-bit binary weighted DAC use any combination of resistors. You can create different size resistors using parallel or serial arrangements.
2. From table 1, decide which port pins you will use for the DAC. Two teams should not be using the same DAC pins.

Table 1

DAC bit 4	**PA3**	**PB6**	**PA6**	**PB4**	**PE6**	**PE4**
DAC bit 3	**PA7**	**PB5**	**PA5**	**PB3**	**PE5**	**PE3**
DAC bit 2	**PA6**	**PB4**	**PA4**	**PB2**	**PE4**	**PE2**
DAC bit 1	**PA5**	**PB3**	**PA3**	**PB1**	**PE3**	**PE1**
DAC bit 0	**PA4**	**PB2**	**PA2**	**PB0**	**PE2**	**PE0**

3. Draw the circuit required to interface the DAC to the TM4C123, include your resistor values and connections. Email your design to your TA for approval. (Include any resistors connections, parallel or serial according to the values available in the lab).
4. To simplify connecting the DAC output to your audio jack, figure 2 (you have received an audio jack in lab 2), Solder 24 gauge solid wires to the audio jack connections.

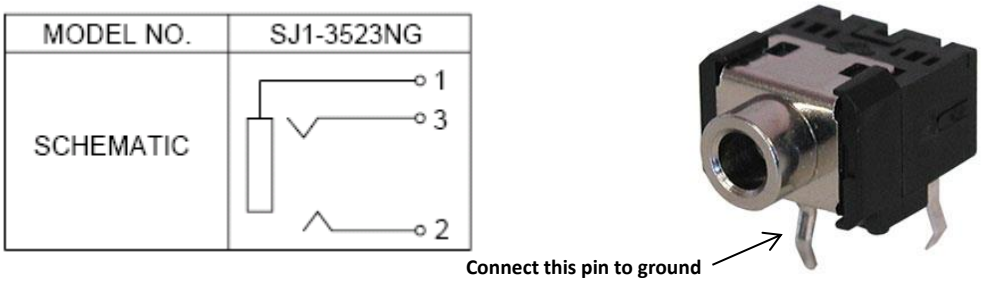

Figure 2

5. Write the device driver for the DAC interface. Include the following functions to implement the interface:
    a. DAC_Init() to initialize the DAC
    b. DAC_Out(unsigned long data) to send the data value to the DAC
    c. Add DAC.h header file containing the prototypes for the DAC functions. Describe how to use the module in the comments of your header file.
6. Before you go any further, test the DAC using the following main program inside lab4.c file.

```
int main(void){
unsigned long Data;
PLL_Init();
DAC_Init();
while(1){
        DAC_Out(Data);
        Data = 0x1F & (Data+1);
        }
}
```

Compile and debug the software in the simulator, then download it to your board.

7. After you download your software, fill out table 2 with the theoretical DAC output voltage and the measured ones using voltmeter.

Table 2

bit4 bit3 bit2 bit1 bit0	Theoretical DAC voltage	Measured DAC voltage
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
21		
22		
30		
31		

Part 2: Piano keyboard device software

1. Design the piano keyboard, use table 3 to decide which port pins you will be using for the piano keyboard inputs.

Table 3

Piano Key 3	**PA5**	**PD3**	**PE3**	**PD4**	**PE4**	**PA6**
Piano Key 2	**PA4**	**PD2**	**PE2**	**PD3**	**PE3**	**PA5**
Piano Key 1	**PA3**	**PD1**	**PE1**	**PD2**	**PE2**	**PA4**
Piano Key 0	**PA2**	**PD0**	**PE0**	**PD1**	**PE1**	**PA3**

2. Design and write the piano keyboard device driver software. Include the following functions:
    - Piano_Init() to initialize the switches inputs
    - Piano_In() to return the digital value representing the pressed key.
    - Place all code that accesses the three switches in the piano.c file.
    - Add a header file “piano.h” containing all prototypes of the functions used in the piano.c file

Part 3: Sound device driver software

1. Design and write the sound device driver software. You will need a sound wave data structure to output the sound. You can use the following link to create your table: http://www.daycounter.com/Calculators/Sine-Generator-Calculator.phtml
   You are free to use the size of the table you want.
2. Place your data structure inside the sound.c file.
3. Include the following functions in your sound.c file:
   - Sound_Init() to initialize the data structure, calls the DAC_Init, and initialize the SysTick interrupt.
   - Sound_Play(note) that starts the sound output at the specified frequency (pitch). The input to the sound_play function will be the RELOAD value that will update the SysTick timer. When you need to output a new note you should write to the NVIC_ST_RELOAD_R without completely initializing SysTick.
4. Chose four notes from table 4 to play by your piano keys

Table 4

Note	Frequency
A	440 Hz
B	494 Hz
C	262 Hz
D	294 Hz
E	330 Hz
F	349 Hz

The frequency of the output sound wave f_{wave} can be determined using the note frequency f_{note} from table 4, multiplied by the size of your sine wave table. For example, if your sine wave size was 16, and your piano switch is playing note A, the output frequency $f_{wave} = f_{note} * 16$.

5. Add a sound.h header file to describe all functions' prototypes. Include description of what sound.c is doing.
6. The SysTick timer will be used to set the time between outputs to the DAC. It will output one value from the sine wave table at every interrupt.
7. Debug your ISR and interrupt. One approach to debugging is to create a sine wave with a constant frequency as shown in figure 3. Run the SysTick periodic interrupts and output one DAC value each interrupt. Add a debugging monitor to your interrupt service routine by setting a digital output pin high at the start of the ISR and clear it at the end of the ISR. Connect one channel of a logic analyzer to the digital output and one channel to the DAC output. The toggling output will show you when the interrupts are occurring. Observe the relationship between software and hardware. This debugging approach is called "heart beat".

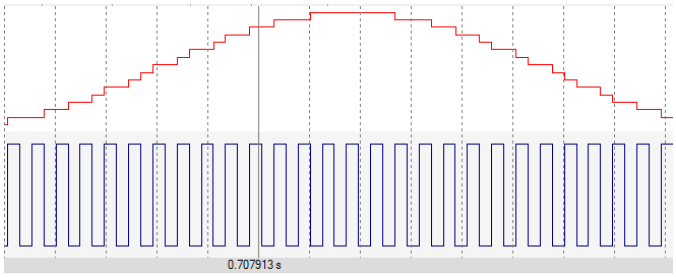

Figure 3

Part 4: The main program.

1. Write a main program to implement the 4-key piano. Implement the piano by making calls to the functions you created in part 1-3
2. Download the program to your Tiva LaunchPad
3. Connect the output from your DAC and the debugging output to a logic analyzer and take a snap shot of the result.
4. Your program should run such as, only one switch is pressed at any time. When one switch is pressed the DAC should output a sine wave at the first note frequency, when another switch is pressed another note is played and so on. When no switch is pressed the DAC should output no sound.
5. The structure of your project should be similar to figure 4.

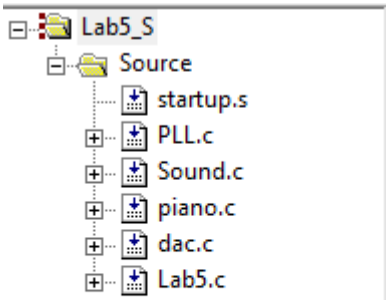

Figure 4

5. Demonstration

When demonstrating the program, you are expected to explain each line of code if asked. Each student in a team will be asked a different question to demonstrate his or her understanding of the project at hand.

Deliverables

1. Circuit diagram showing the DAC circuit and any other connections
2. Software design flowchart
3. A snapshot of the sound wave with the heartbeat signal
4. Theoretical and measured DAC voltage, include calculations of the resolution, range, precision and accuracy.
5. All source files with your names and comments. Your program will be graded for good documentation.