California State University

EGCP 450

Spring 2023

Lab 4

Professor Michael Turi

Zaid frayeh

INTRODUCTION:

This experiment I have been advised to make a Digital Piano using hardware components and software, my design was based on a Digital to Analog Converter (DAC) that would take in bits to output them to resistors which would output a specific key note in the shape of a sign wave.

Procedure/Discussion:

I started by building my hardware to know which ports I was using I used 12,6,3,1.5 ohm resistors for my DAC which also used 4 bits. I used port 4 for my outputs pins 0 1 2 3.
And port 5 as my input pins 4 5 6 for my pushbuttons. As well as a headphone jack to see the output with headphones and using a visual analyzer.

I then started designing my DAC to ensure that it was working correctly.
My DAC consisted of 4 bits which were converted from 4 bits of digital output to analog using the Resistor network that I have built. I have tested this DAC using a voltmeter and made sure the voltage output was within the range of the expected output voltages which is called static testing, I had 16 different values for the DAC with 16 different outputs.
DAC initialized my Output port 4 and had a function to output the different Digital bit values.

Voltage values:

DECIMAL VALUE	VOLTAGE (V) lab	Voltage Theoreticle
0	35.6m	0
1	247.6m	0.22
2	460.7m	0.44
3	0.672	0.66
4	0.864	0.88
5	1.075	1.1
6	1.288	1.32
7	1.499	1.54
8	1.667	1.76
9	1.878	1.98
10	2.089	2.2
11	2.301	2.42
12	2.491	2.64
13	2.701	2.86
14	2.913	3.08
15	3.123	3.3

I needed a function to check which button was being pressed so I made a function called piano which checked which button was being pressed. The source file Piano initialized my input port 5 and I used pins 4,5,6 because my first pins had some problems and where giving other values.
The way I fixed this problem was by using pins 4,5,6 which would give me a value of 0x70 in hexadecimal after I would get the value of which button is pressed I would divide it by 16 in order to have it as the first 3 bits where being used and from there the function would return the button being pressed to the main.c.

My main function was in a constant loop that would get the value of the Button being pressed from PIANO.C
Where I had an if- else statement to which button is being received once it confirms the value is either 0 , 1, 2,4 .

	button
0	Not pressed
1	4th pin
2	5th pin
4	6th pin

Once main confirms which button it sends out to the SysTick->load which is the reload value that we need to set so that it counts down from a specific number in order to play a specific note which was calculated earlier on paper.

Here are my values :
A note :

3000000/440=6818.18
6818/32=> 213 counts

B note :

3000000/494=6072.87
6073/32=> 190 counts

F note :

3000000/349=8595.9
8596/32=> 269 counts

My main also would set the SysTick load to 0 if no button was pressed to set the timer off that would insure no sound would be playing if no button was being pressed.

The SysTick_Handler would compare which reload value is being fed into the timer from main once it figures out which reload value is being fed it sends out to the sound_play the note to be played.

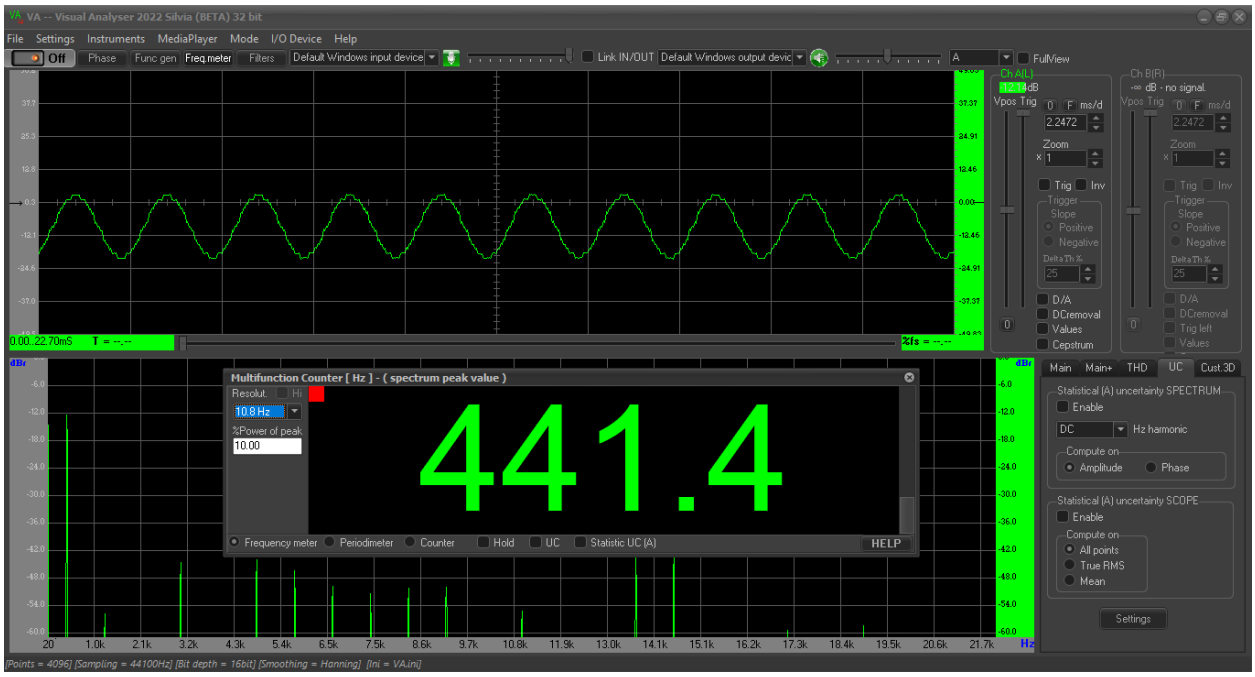
The Sound file initializes the systick timer as well as the DAC it also introduced the values for the sin wave which was sound_Init. my sound_play gets the note to be played from the Systick.

Once sound_play gets the note it sends out the sin wave to the DAC then my DAC outputs those values to the resistors which gives us a sound.

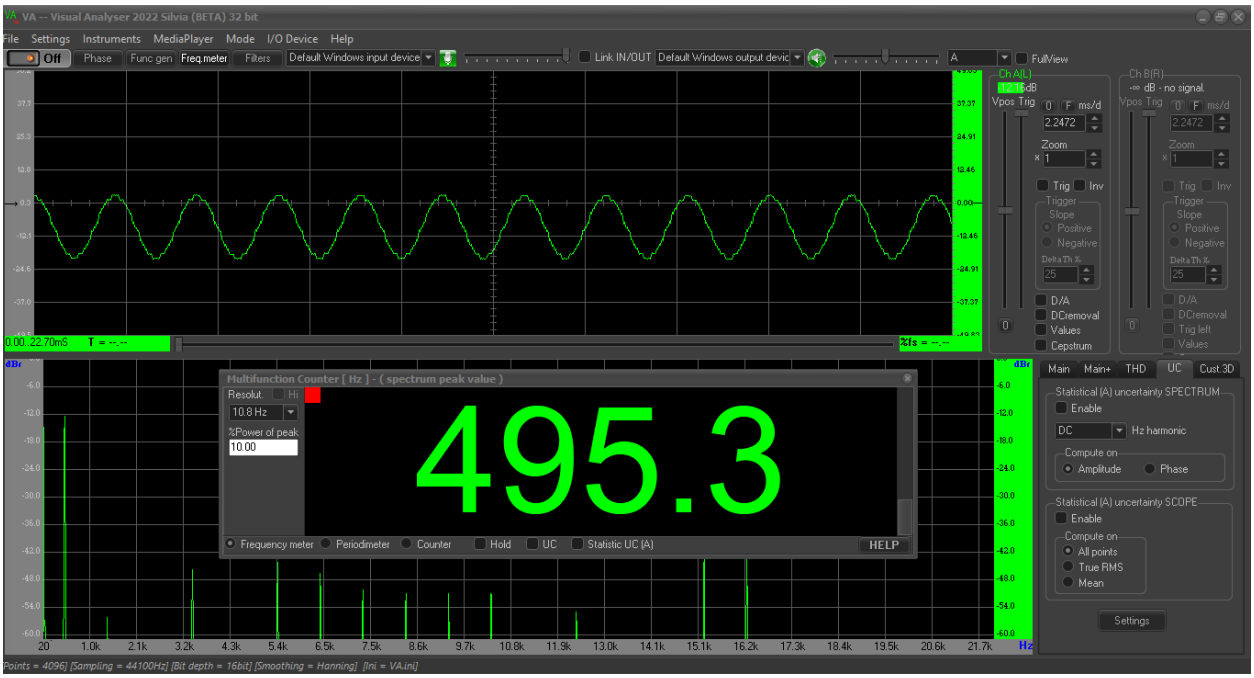
Sound play also needed to realize that when no button is pressed it sends a value of zero to the DAC which insured that whenever I'm not pressing any button it would not have any wave sent out to the DAC.

After doing this lab and showing it to Professor Michael he pointed out to me that the SysTick_handler was doing A lot of work and it didn't need to, I have tried to update my code in order to make it do less work but I had trouble trying to communicate with the SysTick_Handler to make it send out the sin wave that was in the sound file so I kept it the same way since it was working and having a good waveform after analyzing the output using the Virtual Analyzer.
No other issues were faced after doing this LAB.
This lab is straight forward by following the steps assigned to us.
These waveforms show that my Lab is working correctly and shows the frequency of the notes that are to be played

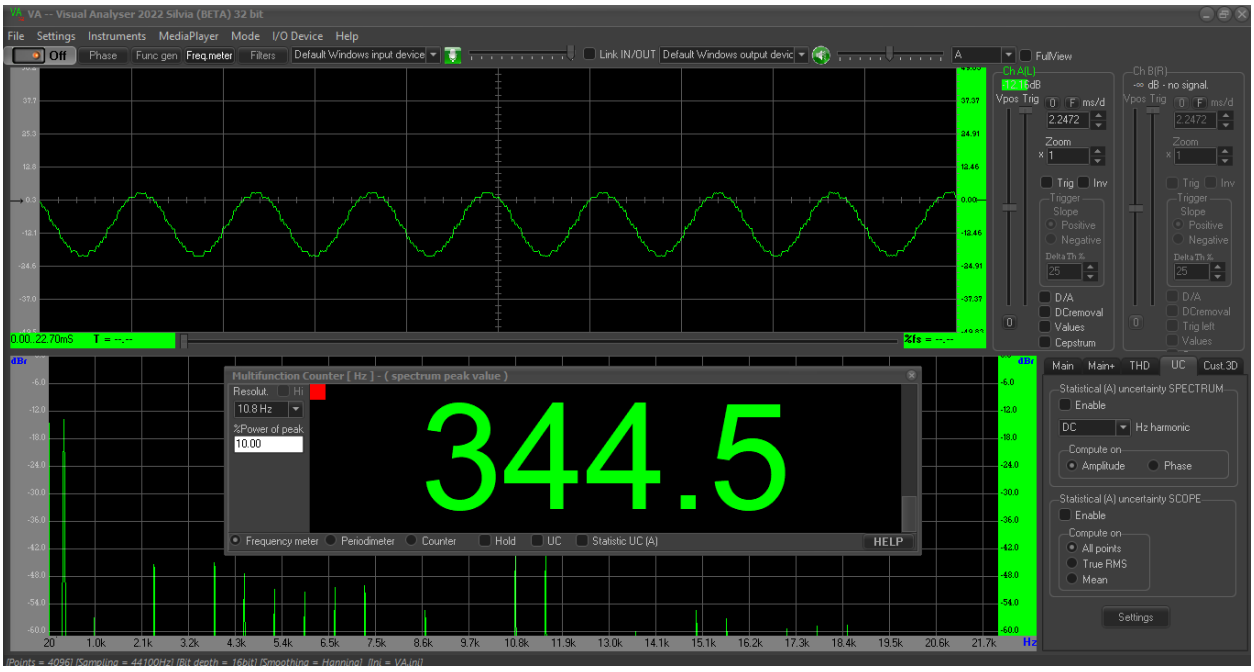
A note (actual value is 440Hz)



B note (actual value is 494 Hz)



F note (actual value is 349 Hz)



Resolution, range and precision :

	Theoretical	LAB
Range	0-3.3 v	35.6m – 3.123 v
Resolution	3.3/(2^4 -1) =0.22	3.123-35.6m/(2^4 -1) =0.206
Precision	2^4 = 16	16

Code :

Main.c :

```
#include <stdint.h>
#include "msp432p401r.h"
#include "SOUND.h"
#include "PIANO.h"
#include "SysTickInts.h"

void main(void){

     uint8_t PIANO = 0;


      PIANO_INIT();
      Sound_Init();


      EnableInterrupts();
      WaitForInterrupt();


      while(1) {
       PIANO = PIANO_IN();

                                    if(PIANO == 0x00)
                                    {
                                        SysTick->LOAD=0;

                                    }
                                    else if(PIANO == 0x01)
                                    {
                                        SysTick->LOAD=269;

                                    }

                                    else if(PIANO == 0x02)
                                    {
                                        SysTick->LOAD = 190;
                                    }

                                    else if(PIANO == 0x04)
                                    {
                                        SysTick->LOAD = 213;
                                    }

      }
};
```

Dac.h :

```
#ifndef DAC_H_
#define DAC_H_


void DAC_INIT(void);
void DAC_OUT(uint32_t data);


#endif /* DAC_H_ */
```

Dac.c :

```
#include <stdint.h>
#include "SOUND.h"
#include "msp432p401r.h"
#include "DAC.h"
#define DAC_out (*((volatile uint8_t *)0x40004C23))


void DAC_INIT(void)
{
            //output pins for dac port 4 pins 0-3
            P4SEL0 &= ~0x0F;
            P4SEL1 &= ~0x0F;
            P4DIR  |= 0x0F;

}
void DAC_OUT( uint32_t data )
{

    DAC_out=data;
}
```

Sound.h :

```
#ifndef SOUND_H_
#define SOUND_H_
void Sound_Init(void);
void Sound_play(uint8_t note);

#endif /* SOUND_H_ */
```

Sound.c :

```
#include <stdint.h>
#include "msp432p401r.h"
#include "SOUND.h"
#include "DAC.h"
#include "SysTickInts.h"

    int sound = 0;
    int indx = 0;

const uint8_t wave[32]= {
   8,9,11,12,13,14,14,15,15,15,14,
   14,13,12,11,9,8,7,5,4,3,2,
   2,1,1,1,2,2,3,4,5,7};

void Sound_Init(void){
    DAC_INIT();
    SysTick_Init(0);
}
void Sound_play(uint8_t note){


    if(note == 1)      {
            indx = (indx + 1) &0x1f ;
            DAC_OUT(wave[indx]);
                       }

    else if(note == 2) {

            indx = (indx + 1) &0x1f ;
            DAC_OUT(wave[indx]);

                        }

    else if (note == 4) {

           indx = (indx + 1) &0x1f ;
            DAC_OUT(wave[indx]);

                        }

    else if(note == 0)  {
            DAC_OUT(0);

                        }

}
```

Piano.h :

```
#ifndef PIANO_H_
#define PIANO_H_

void PIANO_INIT(void);

int PIANO_IN( void);


#endif /* PIANO_H_ */
```

Piano.c :

```
#include <stdint.h>
#include "msp432p401r.h"
#include "piano.h"
#define piano_Input (*((volatile uint8_t *)0x40004C40))
//input port 5 pins 5.4 5.5 5.6 because other pins are giving wrong values
void PIANO_INIT(void){
            P5SEL0 &= ~0x70;
            P5SEL1 &= ~0x70;
            P5DIR  &= ~0x70;
}
int PIANO_IN(void){
        uint32_t input;
        input = (piano_Input&0x70);
        input= input/16;
        return input;

}
```

SysTickints.h :

```
#ifndef __SYSTICKINTS_H__
#define __SYSTICKINTS_H__

void SysTick_Init(uint32_t period);
void SysTick_Handler(void);

#endif
```

SysTickints.c :

```
#include <stdint.h>
#include "SOUND.h"
#include "PIANO.h"
#include "DAC.h"
#include "msp432p401r.h"

void DisableInterrupts(void); // Disable interrupts
void EnableInterrupts(void);  // Enable interrupts
long StartCritical (void);    // previous I bit, disable interrupts
void EndCritical(long sr);    // restore I bit to previous value
void WaitForInterrupt(void);  // low power mode
volatile uint32_t Counts;
volatile uint32_t note;

void SysTick_Init(uint32_t period) {
long sr = StartCritical();

Counts = 0;

SysTick->CTRL = 0;                   // disable SysTick during setup
SysTick->LOAD = period - 1;          // maximum reload value
SysTick->VAL = 0;                    // any write to current clears it
SCB->SHP[3] = (SCB->SHP[3]&0x00FFFFFF)|0x40000000;    // priority 2
SysTick->CTRL = 0x00000007;          // enable SysTick with no interrupts

EndCritical(sr);
}

void SysTick_Handler(void){
                                    if(SysTick->LOAD == 213)
                                        {
                                            Sound_play(4);
                                        }
                                        else if(SysTick->LOAD == 190)
                                        {
                                             Sound_play(2);
                                        }

                                        else if(SysTick->LOAD == 269)
                                        {
                                            Sound_play(1);
                                        }

                                        else if(SysTick->LOAD == 0)
                                        {
                                            Sound_play(0);
                                        }
}
```

Improvements:

Since I have started with the design on my notes and every input, output was followed from the steps that were provided from the professor no improvements needed to be made because my code was working correctly.

Analysis:

Using a 64 sin wave sample would definitely improve quality slightly since we are using a 4 bit there is an infinite number of representations in the sine wave that we can output to the DAC using more numbers to represent that sine wave would have an improved quality and less distortion between the points because we have more points to output to the DAC but it would still need higher power to analyze the data

Using a 6-bit DAC would also improve quality of the waveform because we would have a better percision and better resolution if we would use a 6 bit DAC it can represent 2^6 = 64 different voltage levels, compared to a 4-bit DAC that can represent only 2^4 = 16 different voltage levels.
This would also imply to the resolution since would be dividing by a bigger number it would make it a smaller number and that number is basically the smallest distinguishable change in input to make a change in output.
It would also be better if we use a 64 sine wave sample with that 6 bit.

	4 bit	6 bit
Resolution	3.3/(2^4 -1) =0.22	3.3 /(2^6 -1) =0.0523
Precision	2^4 = 16	2^6=64

Getting a higher Bit DAC or a higher sampling rate will both have a higher improvement to the waveform

Using a 6 bit DAC would need the resistors 1.5,3 ,6, 12,24 ,48 ohm resistors

Bit	Resistor (ohm)
0	48
1	24
2	12
3	6
4	3
5	1.6

Implementation would be almost the same instead of initializing 4 bits for the ports I would initialize 6 bits on port 5.

Conclusion:

This lab introduced me the first time to the DAC I have always knows what it meant but never did I know how to worked, this taught me a lot about these kind of applications since I am in the music industry this gave me more knowledge about speakers and what are the numbers that I see on the speakers and why high end speakers are expensive.
More importantly is how to design a DAC and how its implemented, DAC's are widely used in video, audio applications but most importantly in control and instrument measurement systems these systems need to be as accurate as possible.
It also showed me what the terms precision and resolution and what they actually meant.
This lab also showed me the hardware side of the DAC.
It also showed me how a SysTick handler works and that it gets called automatically from the hardware.
I believe this lab showed me the way of thinking procedure and how things basically work but higher and more sophisticated applications would need more research and more precise values and as well more powerful resources in order to achieve our goal.

Reference:

material provided by the professor.
Brainstorming with David Mouser.