Name: Morris Blaustein

ECE425

Final Project

Music Box

Date: August 20, 2020

Final Project

Writing and Calling Subroutines

Table of Contents

1 Introduction

This final project is a music box implemented in ARM7 assembly on the NXP LPC2148. A music box loops a short sequence of music. It uses 12 notes from the chromatic scale. A physical music box uses a set of pins on a revolving cylinder to pluck the tuned teeth. It has been developed since the 18th century. I wanted to do a project involving music because I am interested in audio signal processing for my career. This music box loops musical notes to the 10-bit digital to analog converter on the LPC2148. The output is a sine wave at frequencies based on notes C4-B4. Music note and length information is stored as double world arrays. The tempo of the music box is 60 beats per minute. Note lengths can be one beat, two beats, and four beats. Output can be observed in simulation by watching “AOUT” in the logic analyzer.

2 Procedure

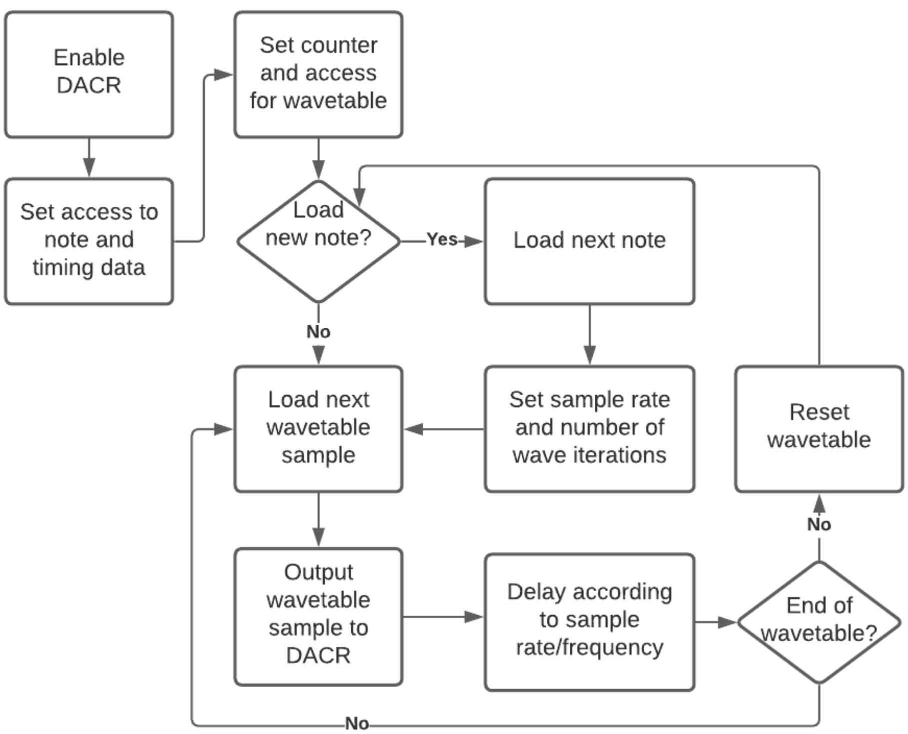

Figure 1 Flowchart for Music Box

3 Testing Strategy

When I began to plan out how I was going to create the music box, I came up with several different options to compensate for not having the actual LPC 2148 microcontroller. When I figured out how to monitor the digital to analog converter output using the logic analyzer and monitoring "AOUT," I decided to use this method to verify my project. Each of the 12 different note frequencies oscillate on the DAC very close to the actual musical note frequency. I compared the two values by calculating frequency from period. I verified the note lengths by making sure the frequency would sustain for either one second, two seconds, or four seconds. It is hard to visually see the results because adjacent notes have similar frequencies. However, when changing from the lowest frequency C4 to the highest frequency B4, the difference in period of each frequency can easily be seen.

To figure out the correct delay count for each note, I stepped through the wave table loop one time to produce a single wave cycle. Then, I compared the output period to the correct period and adjusted the delay count until the periods were as close as possible to the actual note periods. The 10-bit DAC accepts values from 0-1024 and outputs an analog signal 0-3.3V. I created a sine wave table that ranged from 0x0 to 0x3FF.

4 Results

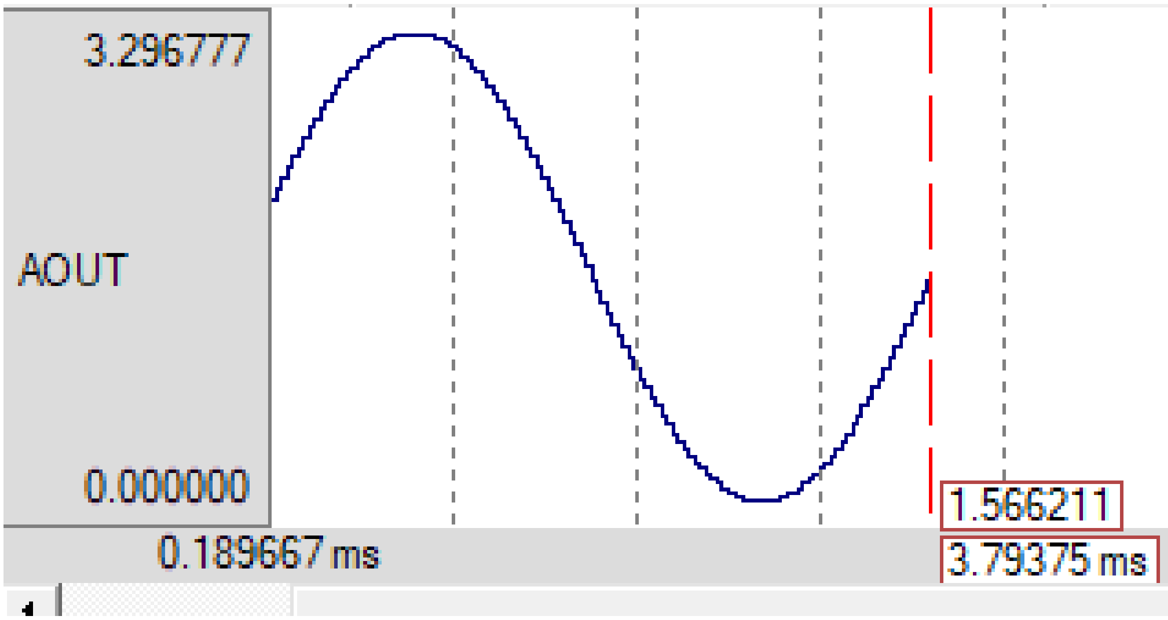

Figure 2 One Wave Iteration of C4

Figure 2 shows that the period of C4 in the music box is 3.79ms. The C4 note is actually 261 Hz and has a period 3.83ms.

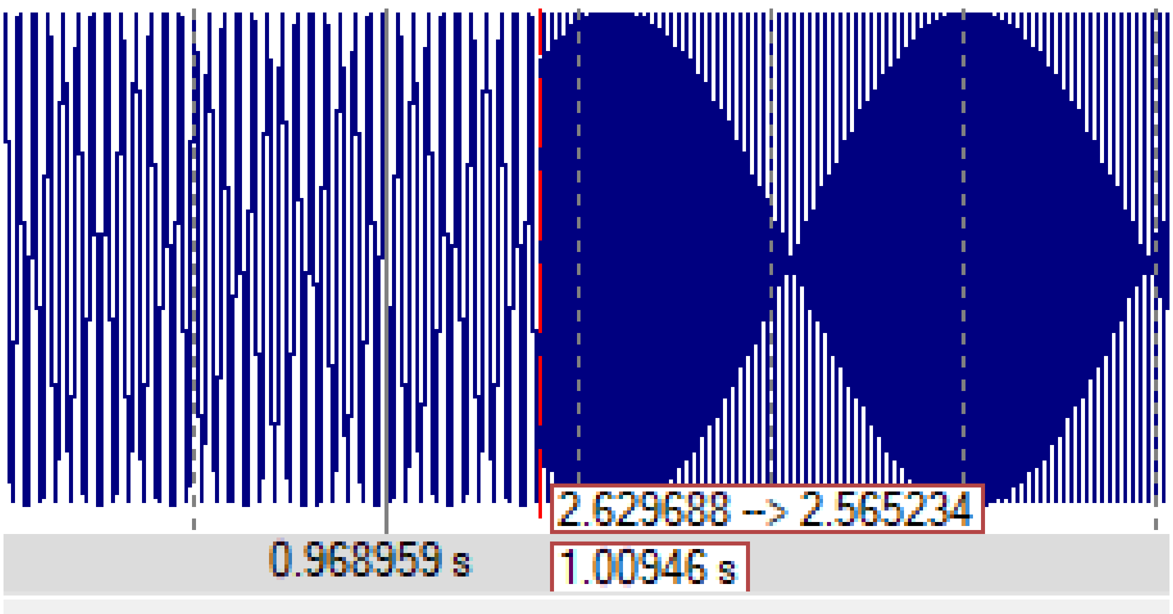

Figure 3 Changing notes at 1 second.

Figure 3 shows a note change from C4 to B4. It occurs at one second because the note length of C4 in this instance is 1 beat. At 60 beats per minute, one beat is one second.

Note	Actual Frequency	Music Box Frequency
C4	261Hz	266Hz
C#4	277Hz	275Hz
D4	293Hz	293Hz
D#4	311Hz	313Hz
E4	329Hz	329Hz
F4	349Hz	347Hz
F#4	369Hz	368Hz
G4	392Hz	390Hz
G#4	415Hz	416Hz
A4	440Hz	446Hz
A#4	466Hz	469Hz
B4	493Hz	500Hz

Figure 4 Table Comparing actual note frequencies to music box note frequencies

This music box was not able to produce the exact frequency for every note. The delay times used produced the frequency closest to the actual frequency.

5 Analysis/Conclusions

The final project was helpful in accumulating everything I've learned from ECE 425 into one program. One part of the program is not implemented efficiently. This happens when setting delay counts when a new note is loaded. For each of the 12 possible notes, I used the same compare statement before each conditional LDR. I think a better way to do this is to use CMPNE for each one after the first compare so that it will only execute if it hasn't seen a match yet.

I wanted to hear the output of the music box on speakers. I thought that I could output the wave table samples to a file and create a wave file using those samples in MATLAB. I realized that this music box works by changing the delay count for each frequency to implement correct timing. This means the sample rate of updating to the DAC changes accordingly, which is not suitable for a wave audio file. Seeing the output of the DAC on the Logic Analyzer was helpful in finding delay times and verifying the project. This project helped me understand how music is created with digital electronics.

6 Appendix

```
;Morris Blaustein

;ECE 425

;CSUN

;8/21/20


;FINAL PROJECT - MUSIC BOX

;This music box loops a song to the DAC on the NXP LPC2148

;The output is a sine wave at frequencies based on notes C4-B4

;Music note and length information is stored as double word arrays

;The tempo of the music box is 60 beats per minute. Note lengths can be one beat, two beats, and four beats.


;Output can be observed in simulation by watching "AOUT" in the logic analyzer


DACR    EQU         0xE006C000

PINSEL0 EQU     0xE002C000


        AREA    musicbox,CODE,READONLY

        ENTRY


        LDR         r3,=0x00080000          ;enable DACR

        LDR         r2,=PINSEL0

        STR         r3,[r2]

        LDR         r1,=0x00001000

        LDR         r0,=DACR

        STR         r1,[r0]
```

```
            ADR     r10,note                ;access to note value and timing
            ADR     r3,time
            MOV     r12,#0

li          LDR     r4,=0x80                ;128, sine wave counter
            LDR     r7,=sine                ;access to sine wave table
            CMP     r12,#0                  ;determine if enough wave cycles
have occured

            LDREQ r9,[r10],#4               ;load next note

            CMPEQ   r9,#0xC                 ;check if at end of music
            ADREQ r10,note                  ;reset note and time to beginning of data if
at end
            ADREQ r3,time
            BEQ     li

            CMP     r12,#0
            CMPEQ   r9,#0xB                 ;check if note is B
            LDREQ r11,=0x23                 ;delay for updating each sample to DACR
            LDREQ r5,=0x1F4                 ;number of wave cycles of B frequency in
one second

            CMP     r12,#0
            CMPEQ   r9,#0xA                 ;check if note is A#
            LDREQ r11,=0x26
            LDREQ r5,=0x1D5
```

```
CMP     r12,#0
CMPEQ   r9,#0x9                 ;check if note is A
LDREQ r11,=0x28
LDREQ r5,=0x1BE

CMP     r12,#0
CMPEQ   r9,#0x8                 ;check if note is G#
LDREQ r11,=0x2B
LDREQ r5,=0x1A0

CMP     r12,#0
CMPEQ   r9,#0x7                 ;check if note is G
LDREQ r11,=0x2E
LDREQ r5,=0x186

CMP     r12,#0
CMPEQ   r9,#0x6                 ;check if note is F#
LDREQ r11,=0x31
LDREQ r5,=0x170

CMP     r12,#0
CMPEQ   r9,#0x5                 ;check if note is F
LDREQ r11,=0x34
LDREQ r5,=0x15B

CMP     r12,#0
```

```
CMPEQ       r9,#0x4                         ;check if note is E
LDREQ r11,=0x37
LDREQ r5,=0x149

CMP         r12,#0
CMPEQ       r9,#0x3                         ;check if note is D#
LDREQ r11,=0x3A
LDREQ r5,=0x139

CMP         r12,#0
CMPEQ       r9,#0x2                         ;check if note is D
LDREQ r11,=0x3E
LDREQ r5,=0x125

CMP         r12,#0
CMPEQ       r9,#0x1                         ;check if note is C#
LDREQ r11,=0x42
LDREQ r5,=0x113

CMP         r12,#0;
CMPEQ       r9,#0x0                         ;check if note is C
LDREQ r11,=0x45
LDREQ r5,=0x10A

CMP         r12,#0
LDREQ r2,[r3],#4                ;if time to change notes, load next note
timing
```

```
            MULEQ       r12,r5,r2                   ;multiply frequency times note
length (1,2,4)
            SUB         r12,r12,#1                  ;decrement from wave cycle count

l           MOV         r8,r11
            LDR         r5,[r7],#4                  ;load next sine wave table sample
            LSL         r6,r5,#6                    ;shift left 6 bits to format for DACR
            STR         r6,[r0]                     ;store value to DACR
            SUB         r4,r4,#1                    ;decrement sine wave table counter

delay   SUB         r8,#1                       ;delaying according to frequency
            CMP         r8,#0

            BNE         delay
            CMP         r4,#0
            BNE         l                                   ;branch if not at end of sine
wave table
            B           li                                  ;branch to process next wave
iteration

stop    B           stop

            ; 128 sample sine wave table
sine    DCD         0x200,0x219,0x232,0x24b,0x263,0x27c,0x294,0x2ac
            DCD         0x2c3,0x2da,0x2f1,0x306,0x31c,0x330,0x344,0x357
            DCD         0x369,0x37a,0x38b,0x39a,0x3a9,0x3b6,0x3c3,0x3ce
            DCD         0x3d8,0x3e1,0x3e9,0x3f0,0x3f5,0x3f9,0x3fd,0x3fe
            DCD         0x3ff,0x3fe,0x3fd,0x3f9,0x3f5,0x3f0,0x3e9,0x3e1
```

```
            DCD         0x3d8,0x3ce,0x3c3,0x3b6,0x3a9,0x39a,0x38b,0x37a
            DCD         0x369,0x357,0x344,0x330,0x31c,0x306,0x2f1,0x2da
            DCD         0x2c3,0x2ac,0x294,0x27c,0x263,0x24b,0x232,0x219
            DCD         0x200,0x1e6,0x1cd,0x1b4,0x19c,0x183,0x16b,0x153
            DCD         0x13c,0x125,0x10e,0xf9,0xe3,0xcf,0xbb,0xa8
            DCD         0x96,0x85,0x74,0x65,0x56,0x49,0x3c,0x31
            DCD         0x27,0x1e,0x16,0xf,0xa,0x6,0x2,0x1
            DCD         0x0,0x1,0x2,0x6,0xa,0xf,0x16,0x1e
            DCD         0x27,0x31,0x3c,0x49,0x56,0x65,0x74,0x85
            DCD         0x96,0xa8,0xbb,0xcf,0xe3,0xf9,0x10e,0x125
            DCD         0x13c,0x153,0x16b,0x183,0x19c,0x1b4,0x1cd,0x1e6

            ;12 notes to be played by music box
            ;C=0 C#=1 D=2 D#=3 E=4 F=5 F#=6 G=7 G#=8 A=9 A#=0xA B = 0xB
note    DCD         0x0,0xB,0x1,0xA,0x4,0x5,0x6,0x7,0x8,0x9,0xA,0xB,0xC

            ;note lengths
            ;1=1 second(1 beat), 2=2 seconds (2 beats), 4=4 seconds (4 beats)
time    DCD         0x1,0x1,0x1,0x1,0x1,0x1,0x1,0x1,0x1,0x1,0x1,0x1

            END
```