CECS 447 Spring 2021 Project 1

Music Box and Digital Piano

By

Jesus Franco

2/10/2021

A music box and a digital piano. The music box will allow the user to play three songs consecutively and the digital piano will allow the user to play their own sounds using piano keys and is also equipped with an auto-play mode to play a song.

<u>Introduction for Music Box:</u>
For the Music Box part of this project, the program will play 3 songs of “Marry had a Little Lamb”, “Twinkle Twinkle Little Star”, and “Happy Birthday” in that respected order. If SW1 is pressed on the Launchpad, the music will turn ON/OFF and if SW2 is pressed, then the next song will play. This design will output sound off PA2, which is connected to some apple earphones.

<u>Introduction for Digital Piano:</u>
For the Digital Piano, the program will allow the user to manually press buttons to mimic a piano. There will be 7 buttons and they will be connected to PD3-0 and PC6-4. The sound will be outputting through a R-2R Ladder DAC circuit and pins PB5-0 will be used. The digital piano is also equipped with an auto-play mode as well. If SW1 is pressed on the Launchpad, the music will turn ON/OFF and if SW2 is pressed, then that will change the mode from a piano mode to a auto-play mode.

<u>Port Table for Music Box:</u>

Name	**I/0**	**Port**	**Description**
SW1	Input	PF4	Will turn ON/OFF the music.
SW2	Input	PF0	Will transition to the next song.
Speaker Output	Output	PA2	Will output sound.

<u>Port Table for Digital Piano:</u>

Name	**I/0**	**Port**	**Description**
SW1	Input	PF4	Will turn ON/OFF the music.
SW2	Input	PF0	Will transition to the next mode.
Piano key C4	Input	PD0	Play piano key C4.
Piano key D4	Input	PD1	Play piano key D4.
Piano key E4	Input	PD2	Play piano key E4.
Piano key F4	Input	PD3	Play piano key F4.
Piano key G4	Input	PC4	Play piano key G4.
Piano key A4	Input	PC5	Play piano key A4.
Piano key B4	Input	PC6	Play piano key B4.
R-2R Ladder [5]	Output	PB5	R-2R Ladder bus 5.
R-2R Ladder [4]	Output	PB4	R-2R Ladder bus 4.
R-2R Ladder [3]	Output	PB3	R-2R Ladder bus 3.
R-2R Ladder [2]	Output	PB2	R-2R Ladder bus 2.
R-2R Ladder [1]	Output	PB1	R-2R Ladder bus 1.
R-2R Ladder [0]	Output	PB0	R-2R Ladder bus 0.

Operation for Music Box:

Apple Earphones: For this operation, some alligator clips were used to allow an easy connection of the ground and output coming from the earphone jack.

Operation for Digital Piano:

R-2R Ladder: This circuit was very simple to create from the schematic that was given. The circuit was first tested as a 3 bit DAC and its sole purpose was to check if my R-2R Ladder circuit was correct. Once this was confirmed, added 3 more bits to make the 6 bit DAC circuit was quick.
Piano Keys: The piano keys were setup with internal pull up resistors that were initialized in the PortD and PortC initialization functions.
Audio Amplifier: The audio amplifier allowed the sound to be a bit smoother and louder. It came equipped with a potentiometer that allowed volume adjustments.
Apple Earphones: For this operation, some alligator clips were used to allow an easy connection of the ground and output coming from the earphone jack.

Theory for Music Box:

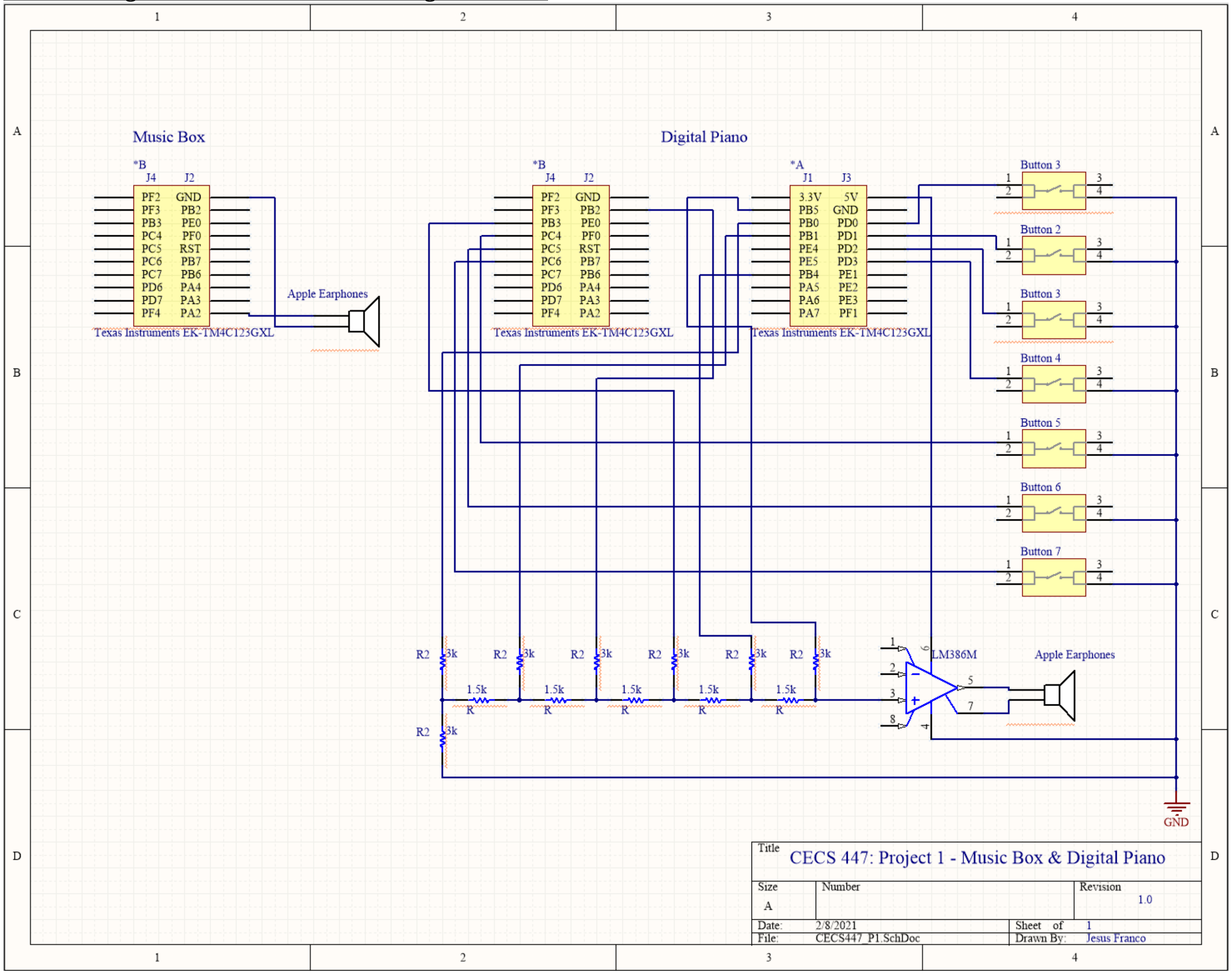
Starting this project was simply using the music example project that was given by the professor and adding some breaking logic to escape the index of an array. Since the earphones that were bought from amazon were not the best, I had to rely on using some old apple earphones that had a decent input jack. I used alligator clips to attach PA2 and the ground pin to the earphone jack in order to output sound. A simple if and else somewhat state machine was used to transition between songs and keeping track of which song was necessary.

Theory for Digital Piano:

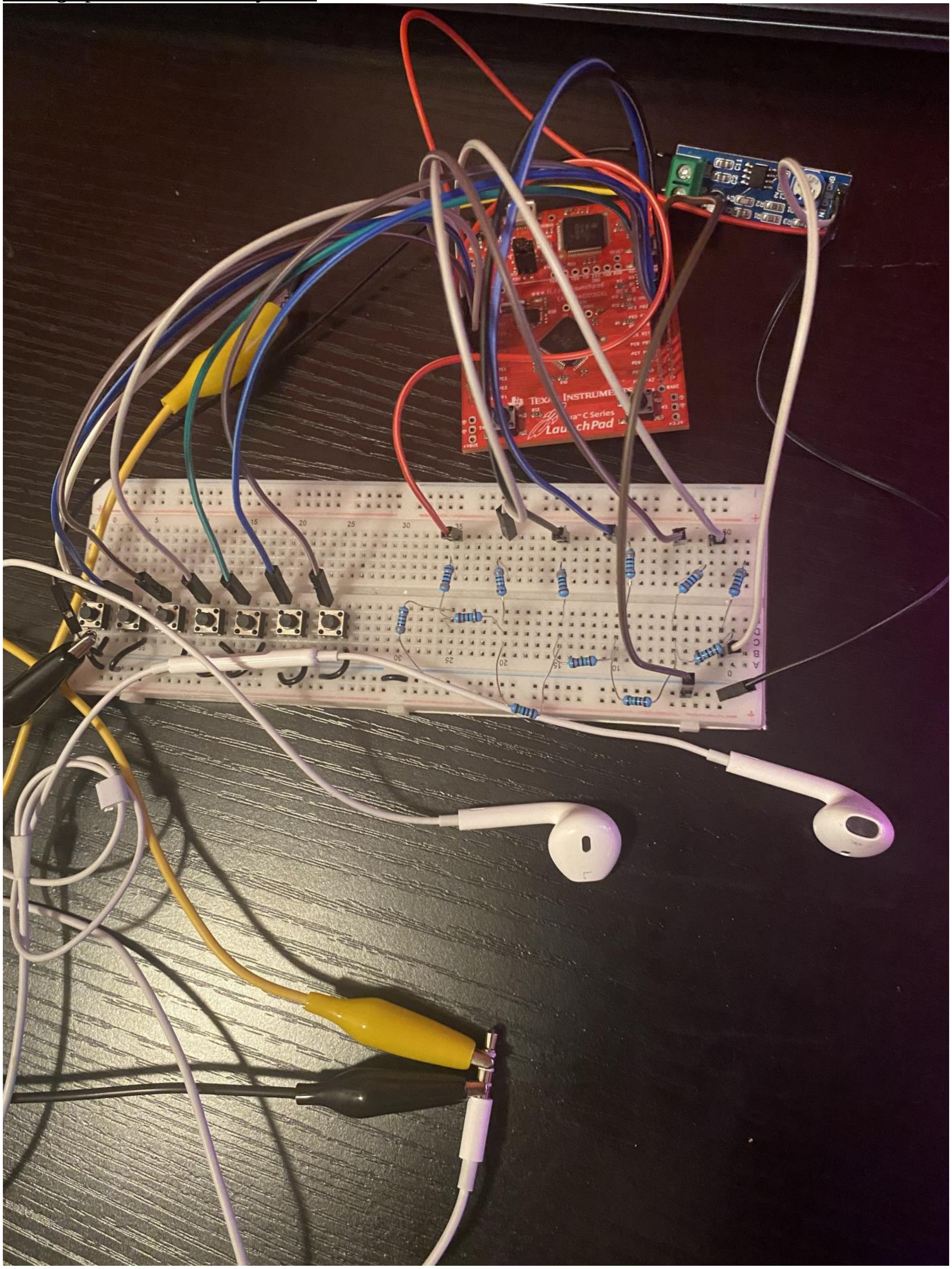
Starting this project was testing the DAC example project that was given by the professor. Testing the DAC example allowed assurance of knowing that the R-2R ladder circuit was correct for a 3 pin DAC output. Knowing this information, it was possible to now create a 6 pin DAC output and test a song. The R-2R ladder uses 1.5k and 3k resistors. Now that a song has been tested, the next part of the logic was to add the piano keys. Since I initialize internal pull up resistors for ports B and C, the switch logic did not require extra resistors. The switches are being sourced from the Launchpad and tied to ground. An audio amplifier is tied to the output of the R-2R ladder, VBUS on the Launchpad, and a ground pin. Some old apple earphones are connected to the end of the audio amplifier to serve as a speaker. Alligator clips are attached to hear a nice smooth sound and the built in potentiometer from the audio amplifier was used to determine the volume control of the digital piano.

Hardware Design:

Circuit Diagram for Music Box and Digital Piano:


Music Box
Digital Piano
Texas Instruments EK-TM4C123GXL
Apple Earphones
J4 J2
PF2 GND PF3 PB2 PB3 PE0 PC4 PF0 PC5 RST PC6 PB7 PC7 PB6 PD6 PA4 PD7 PA3 PF4 PA2
J1 J3
3.3V 5V PB5 GND PB0 PD0 PB1 PD1 PE4 PD2 PE5 PD3 PB4 PE1 PA5 PE2 PA6 PE3 PA7 PF1
Button 3
Button 2
Button 4
Button 5
Button 6
Button 7
R2 3k
1.5k R
LM386M
GND
CECS 447: Project 1 - Music Box & Digital Piano
Size A
Revision 1.0
Date: 2/8/2021
File: CECS447_P1.SchDoc
Sheet 1 of 1
Drawn By: Jesus Franco

Photographs of Hardware System:



Software Design for Music Box:

```
// Name: Jesus Franco
// Student ID: 014046368
// Course Number: CECS 447
// Assignment: Project 1: Music Box (part 1)
// SingASong.c
// Description: This code allows the user to interchange between three songs.
//                                         SW1 will turn ON/OFF the program and SW2 will
transition to
//                                         the next song. PA2 will ouput the sound to some
apple earphones.

//     PF4 -> SW1 to turn ON/OFF music
//   PF0 -> SW2 to transition songs

//   PA2 -> Output Sound

//   Red LED       -> Music is OFF
//     Green LED -> Music is ON

// 1. Header files
#include "tm4c123gh6pm.h"
#include "SysTick.h"
#include "PLL.h"

// 2. Declarations Section

// define music data structure
struct Note {
  unsigned char tone_index;
  unsigned char delay;
};
typedef const struct Note NTyp;

// initial values for piano major tones.
// Assume SysTick clock frequency is 16MHz.
const unsigned long Tone_Tab[] =
// Notes:  C, D, E, F, G, A, B
// Offset: 0, 1, 2, 3, 4, 5, 6
{95556, 85131, 75843, 71586, 63776, 56818, 50619, // C5 Major
 47778, 42566, 37921, 35793, 31888, 28409, 25309, // C6 Major
 23889, 21283, 18961, 17897, 15944, 14205, 12655};// C7 Major

// indexes for notes used in music scores
#define C5 0+7
#define D5 1+7
#define E5 2+7
#define F5 3+7
#define G5 4+7
#define A5 5+7
#define B5 6+7
#define C6 0+2*7
#define D6 1+2*7
#define E6 2+2*7
#define F6 3+2*7
#define G6 4+2*7
#define A6 5+2*7
#define B6 6+2*7

// doe ray mi fa so la ti
// C   D   E  F  G  A  B
NTyp mysong_1[] =
// score table for Mary Had A Little Lamb
{E6, 4, D6, 4, C6, 4, D6, 4, E6, 4, E6, 4, E6, 8,
 D6, 4, D6, 4, D6, 8, E6, 4, G6, 4, G6, 8,
 E6, 4, D6, 4, C6, 4, D6, 4, E6, 4, E6, 4, E6, 8,
 D6, 4, D6, 4, E6, 4, D6, 4, C6, 8, 0, 0 };

NTyp mysong_2[] =
// // score table for Twinkle Twinkle Little Stars
{C6,4,C6,4,G6,4,G6,4,A6,4,A6,4,G6,8,F6,4,F6,4,E6,4,E6,4,D6,4,D6,4,C6,8,
 G6,4,G6,4,F6,4,F6,4,E6,4,E6,4,D6,8,G6,4,G6,4,F6,4,F6,4,E6,4,E6,4,D6,8,
 C6,4,C6,4,G6,4,G6,4,A6,4,A6,4,G6,8,F6,4,F6,4,E6,4,E6,4,D6,4,D6,4,C6,8,0,0};
//
NTyp mysong_3[] =
// score table for Happy Birthday
{//so   so   la   so   doe' ti
   G5,2,G5,2,A5,4,G5,4,C6,4,B5,4,
//// pause so   so   la   so   ray' doe'
   0,4,  G5,2,G5,2,A5,4,G5,4,D6,4,C6,4,
// pause so   so   so'  mi'  doe' ti   la
   0,4,  G5,2,G5,2,G6,4,E6,4,C6,4,B5,4,A5,8,
// pause fa'  fa'   mi'  doe' ray' doe' stop
       0,4,  F6,2,F6,2, E6,4,C6,4,D6,4,C6,8,0,0};

      // Function Prototypes
void Speaker_Init(void);
void Delay(void);
extern void EnableInterrupts(void);
void play_a_song(NTyp scoretab[]);

void PortF_Init(void);
void Sound_Init(unsigned long period);

unsigned char music_flag = 0;
unsigned char next_flag =0;         // play next song

unsigned char song1_flag = 0; // Marry had a Little Lamb song
unsigned char song2_flag = 0; // Twinkle Twinkle Little Star
unsigned char song3_flag = 0; // Happy Birthday

int main(void){
  Speaker_Init(); // PortA initializtion for PA2 for sound
  SysTick_Init(); // Systick initialization
      PortF_Init();            // PortF Initialization for onboard LEDs and switches
      PLL_Init();              // PLL initialization for 50MHz
  EnableInterrupts();  // SysTick uses interrupts
      if(music_flag==1)                          GPIO_PORTF_DATA_R = 0x08;     // music is
ON
      else if(music_flag ==0)        GPIO_PORTF_DATA_R = 0x02; // music is OFF
      song1_flag = 1;   // Marry had a Little Lamb plays initially

  while(1){
            if(music_flag){   // Music flag is active
                  if(song1_flag){   // song1_flag is active
                        song1_flag = 0;   // set song1_flag to 0
                        next_flag = 0;  // set next_flag to 0
                        play_a_song(mysong_1); // plays Marry had a Little Lamb
                        if(next_flag == 1) song2_flag = 1; // if next_flag is active,
goto song2_flag
                        else                                        song1_flag = 1; //
else, stay at song1_flag
                  }

                  else if(song2_flag){ // song2_flag is active
```

```
                        song2_flag = 0;   // set song2_flag to 0
                        next_flag = 0;    // set next_flag to 0
                        play_a_song(mysong_2); // Play Twinkle Twinkle Little Star
                        if(next_flag == 1) song3_flag = 1; // if next_flag is active,
goto song3_flag
                        else                                        song2_flag = 1; //
else, stay at song2_flag
                  }

                  else if(song3_flag){ // song3_flag is active
                        song3_flag = 0;   // set song3_flag to 0
                        next_flag = 0;    // set next_flag to 0
                        play_a_song(mysong_3); // Play Happy Birthday
                        if(next_flag == 1) song1_flag = 1; // if next_flag is active,
goto song1_flag
                        else                                        song3_flag = 1; //
else, stay at song3_flag
                  }
            }// end music_flag
            Delay(); // .01sec delay
  }// end while loop
}// end main

// This function will play a song given from the song array
void play_a_song(NTyp scoretab[])
{
      unsigned char i=0, j;

      while (scoretab[i].delay) {
            if(!music_flag || next_flag) break; // will break out of the function
            if (!scoretab[i].tone_index)
                  SysTick_stop(); // silence tone, turn off SysTick timer
            else {
                  SysTick_Set_Current_Note(Tone_Tab[scoretab[i].tone_index]);
                  SysTick_start();
            }
            // tempo control:
            // play current note for duration
            // specified in the music score table
            for (j=0;j<scoretab[i].delay;j++){
                  if(!music_flag || next_flag) break; // will break out of the function

                  Delay();
            }
            SysTick_stop();
            i++;  // move to the next note
      }

      // pause after each play
      for (j=0;j<15;j++)
            Delay();
}

// Subroutine to wait 0.01 sec for 50MHz system clock
void Delay(void){
      unsigned long volatile time;
      time = 500000; // .01 sec
  while(time){
            time--;
  }
}

// Make PA2 an output, enable digital I/O, ensure alt. functions off
```

```
void Speaker_Init(void){ volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x01;           // 1) activate clock for Port A
  delay = SYSCTL_RCGC2_R;           // allow time for clock to start
                                    // 2) no need to unlock PA2
  GPIO_PORTA_PCTL_R &= ~0x00000F00; // 3) regular GPIO
  GPIO_PORTA_AMSEL_R &= ~0x04;      // 4) disable analog function on PA2
  GPIO_PORTA_DIR_R |= 0x04;         // 5) set direction to output
  GPIO_PORTA_AFSEL_R &= ~0x04;      // 6) regular port function
  GPIO_PORTA_DEN_R |= 0x04;         // 7) enable digital port
      GPIO_PORTA_DR8R_R |= 0x04;        // 8) optional: enable 8 mA drive on PA2 to
increase the voice volumn
}

// PortF_Init for initializing the onboard switches/LEDs and the interrupts
void PortF_Init(void){
      volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x00000020;     // Acativate clock for port F
  delay = SYSCTL_RCGC2_R;           // delay
  GPIO_PORTF_LOCK_R = 0x4C4F434B;   // unlock GPIO PortF PF0
  GPIO_PORTF_CR_R |= 0x1F;          // allow changes to PF4-0
      GPIO_PORTF_DIR_R &= ~0x11;                // PF4 & PF0 as inputs
  GPIO_PORTF_DIR_R |=  0x0E;              // PF3-0 as outputs
  GPIO_PORTF_AFSEL_R &= ~0x11;            // disable alt funct on PF4 & PF0
  GPIO_PORTF_DEN_R |= 0x1F;               // enable digital I/O on PF4 & PF0
  GPIO_PORTF_PCTL_R &= ~0x000F0000; // configure PF4 as GPIO
  GPIO_PORTF_AMSEL_R &= ~0x11;      // disable analog functionality on PF4 & PF0
  GPIO_PORTF_PUR_R |= 0x11;               // enable pull-up on PF4 & PF0
      GPIO_PORTF_IS_R &= ~0x11;                 // PF4 & PF0 is edge-sensitive
  GPIO_PORTF_IBE_R &= ~0x11;                    // PF4 & PF0 are not both edges
      //GPIO_PORTF_IBE_R |= 0x11;                     // PF4 & PF0 are both edges
  GPIO_PORTF_IEV_R &= ~0x11;              // PF4 & PF0 falling edge event
  GPIO_PORTF_ICR_R = 0x11;                // clear flag4
  GPIO_PORTF_IM_R |= 0x11;                // arm interrupt on PF4 & PF0
  NVIC_PRI7_R |= (NVIC_PRI7_R&0xFF00FFFF)|0x00A00000; // priority 5
  NVIC_EN0_R |= 0x40000000;               // enable interrupt 30 in NVIC
}

// PortF Handler for the switches
void GPIOPortF_Handler(void){
// SW1: Turn music ON/OFF
// SW2: Changes to the next song
  if(GPIO_PORTF_RIS_R&0x01){          // SW2 touched
    GPIO_PORTF_ICR_R = 0x01;          // acknowledge flag0
            next_flag = 1;                                                      //
set next_flag to 1
      }
  if(GPIO_PORTF_RIS_R&0x10){          // SW1 touched
    GPIO_PORTF_ICR_R = 0x10;          // acknowledge flag4
            music_flag ^= 1;                                              // toggle
music_flag
            song1_flag = 1;                                                     //
initially start at song1
            song2_flag = 0;                                                     // do
not play song2 initially
            song3_flag = 0;                                                     // do
not play song2 initially
      }
      if(music_flag==1)                          GPIO_PORTF_DATA_R = 0x08;     // music is
ON
      else if(music_flag ==0)       GPIO_PORTF_DATA_R = 0x02; // music is OFF
}
```

Software Design for Digital Piano:

```
// Name: Jesus Franco
// Student ID: 014046368
// Course Number: CECS 447
// Assignment: Project 1: Digital Piano (part 2)
// DAC3Bit.c
// Description: This code allows the user to interchange between a piano mode
//                                    or a auto-player mode. SW1 will turn ON/OFF the
program and SW2
//                                          will change the mode from intially a piano mode
to a auto-play
//                                          mode. PortsB will output a 6 bit DAC to ouput
the sound to an audio
//                                          amplifier which is connected to some apple
earphones. PortC and PortD
//                                          will serve as piano keys inputs to allow the
user to do sound.

//     PD3-0 & PC6-4 -> Piano buttons

//     PF4 -> SW1 to turn ON/OFF music
//   PF0 -> SW2 to change mode

//   PB5-0 -> DAC outputs

//   Red LED       -> Music is OFF
//     Green LED -> Music is ON

#include "tm4c123gh6pm.h"
#include "Sound.h"
#include "SwitchLed.h"
#include "PLL.h"

// 2. Declarations Section

// define music note data structure
struct Note {
  unsigned long tone_index;
  unsigned char delay;
};

typedef const struct Note NTyp;

// Constants
#define TONE_DURATION 2 // each tone uses the same duration
#define NUM_VALs  64    // Assume 6-bit DAC is used, that will give 64 values for one
period.

// initial values for piano major tones.
// Assume SysTick clock frequency is 50MHz.
const unsigned long tonetab[] =
// C, D, E, F, G, A, B
// 1, 2, 3, 4, 5, 6, 7
{95556, 85131, 75843, 71586, 63776, 56818, 50619, // C5 Major
 47778, 42566, 37921, 35793, 31888, 28409, 25309, // C6 Major
 23889, 21283, 18961, 17897, 15944, 14205, 12655};// C7 Major

// index definition for tones used in happy birthday.
#define G6 4+7
#define A6 5+7
#define B6 6+7
#define C7 0+14
```

```
#define D7 1+14
#define E7 2+14
#define F7 3+14
#define G7 4+14

// note table for Happy Birthday
// doe ray mi fa so la ti
// C   D   E  F  G  A  B
NTyp happybirthday[] =
{
// so   so   la   so   doe' ti
   G6,2,G6,2,A6,4,G6,4,C7,4,B6,4,
// pause so   so   la   so   ray' doe'
   0,4,  G6,2,G6,2,A6,4,G6,4,D7,4,C7,4,
// pause so   so   so'  mi'  doe' ti   la
   0, 4, G6,2,G6,2,G7,4,E7,4,C7,4,B6,4,A6,8,
// pause fa'  fa'   mi'  doe' ray' doe'  stop
       0,4,  F7,2,F7,2, E7,4,C7,4,D7,4,C7,8, 0,0
};

void play_a_song(NTyp notetab[]);
void Delay(void);
void PortF_Init(void);

#define C4_K 95420 // 262 Hz
#define D4_K 85034 // 294 Hz
#define E4_K 75758 // 330 Hz
#define F4_K 71633 // 349 Hz
#define G4_K 63776 // 392 Hz
#define A4_K 56818 // 440 Hz
#define B4_K 50607 // 494 Hz

// basic functions defined at end of startup.s
extern void DisableInterrupts(void); // Disable interrupts
extern void EnableInterrupts(void);  // Enable interrupts
extern void WaitForInterrupt(void);  // low power mode

unsigned long input_D, input_C, previous, previous2; // Inputs for PortD and PortC inputs
unsigned char music_flag = 0;
       // music ON/OFF
unsigned char next_flag =0;
             // play next mode keyboard/auto-play
unsigned char keyboard_flag = 0;
// keyboard flag to be in keyboard mode
unsigned char play_flag = 0;
       // play flag to be in auto-play mode

int main(void){
      PLL_Init();       // PLL initialization for 50MHz
      PortF_Init(); // Port F initialization for onboard switches and LEDs
  DAC_Init();   // Port B initialization for 6 bit DAC
      PortD_Init();     // Port D initialization for inputs PD3-0
      PortC_Init(); // Port C initialization for inputs PC6-4

      // output is music is on/off
      if(music_flag==1)                          GPIO_PORTF_DATA_R = 0x08; // green LED is
ON
      else if(music_flag ==0)       GPIO_PORTF_DATA_R = 0x02; // red LED is OFF

  while(1){
            if(music_flag){ // if ON

                  if(keyboard_flag){   // if in keyboard mode
```

```
                        keyboard_flag = 0; // set keyboard_flag back to 0
                        next_flag = 0;          // set next_flag to 0

                        input_D = Switch_In_D()&0x0F; // inputs for portD
                        input_C = Switch_In_C()&0x70; // inputs for portC

                        if((input_D&&(previous==0)) || (input_C && (previous2 ==0))){ //
just pressed
                              EnableInterrupts();
                              if(Switch_In_D()&0x01){Sound_Init(C4_K/64);}      // C4
                              else if(Switch_In_D()&0x02){Sound_Init(D4_K/64);} // D4
                              else if(Switch_In_D()&0x04){Sound_Init(E4_K/64);} // E4
                              else if(Switch_In_D()&0x08){Sound_Init(F4_K/64);} // F4
                              else if(Switch_In_C()&0x10){Sound_Init(G4_K/64);} // G4
                              else if(Switch_In_C()&0x20){Sound_Init(A4_K/64);} // A4
                              else if(Switch_In_C()&0x40){Sound_Init(B4_K/64);} // B4
                        }

                        if((previous&&(input_D==0))|| (previous2&&input_C==0)){ // just
released
                              DisableInterrupts();    // stop sound
                        }

                        previous = input_D;  // previous state for inputs D
                        previous2 = input_C; // previous state for inputs C
                        Delay10ms();                      // remove switch bounce

                        if(next_flag == 1) play_flag = 1;                // if next_flag
is active, goto auto-play mode
                        else                                        keyboard_flag = 1; //
else stay at keyboard mode

                  } // end keyboard flag
                  else if(play_flag){
                        EnableInterrupts();
                        play_flag = 0; // set play_flag back to 0
                        next_flag = 0; // set next_flag back to 0
                        play_a_song(happybirthday); // play happy birthday

                        if(next_flag == 1) keyboard_flag = 1;     // if next_flag is
active, goto keyboard mode
                        else                                        play_flag = 1;
      // else stay at auto-play mode
                  } //end play_flag
            } // end music flag
      } // end while loop
} // end main

// PortF_Init for initializing the onboard switches/LEDs and the interrupts
void PortF_Init(void){
      volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x00000020;     // Acativate clock for port F
  delay = SYSCTL_RCGC2_R;           // delay
  GPIO_PORTF_LOCK_R = 0x4C4F434B;   // unlock GPIO PortF PF0
  GPIO_PORTF_CR_R |= 0x1F;          // allow changes to PF4-0
      GPIO_PORTF_DIR_R &= ~0x11;                // PF4 & PF0 as inputs
  GPIO_PORTF_DIR_R |=  0x0E;              // PF3-0 as outputs
  GPIO_PORTF_AFSEL_R &= ~0x11;            // disable alt funct on PF4 & PF0
  GPIO_PORTF_DEN_R |= 0x1F;               // enable digital I/O on PF4 & PF0
  GPIO_PORTF_PCTL_R &= ~0x000F0000; // configure PF4 as GPIO
  GPIO_PORTF_AMSEL_R &= ~0x11;      // disable analog functionality on PF4 & PF0
  GPIO_PORTF_PUR_R |= 0x11;               // enable pull-up on PF4 & PF0
      GPIO_PORTF_IS_R &= ~0x11;                 // PF4 & PF0 is edge-sensitive
  GPIO_PORTF_IBE_R &= ~0x11;                    // PF4 & PF0 are not both edges
  GPIO_PORTF_IEV_R &= ~0x11;              // PF4 & PF0 falling edge event
  GPIO_PORTF_ICR_R = 0x11;                // clear flag4
  GPIO_PORTF_IM_R |= 0x11;                // arm interrupt on PF4 & PF0
  NVIC_PRI7_R |= (NVIC_PRI7_R&0xFF00FFFF)|0x00A00000; // priority 5
  NVIC_EN0_R |= 0x40000000;               // enable interrupt 30 in NVIC
}

// PortF Handler for the switches
void GPIOPortF_Handler(void){
// SW1: Turn ON/OFF
// SW2: Switch mode

  if(GPIO_PORTF_RIS_R&0x01){          // SW2 touched
    GPIO_PORTF_ICR_R = 0x01;          // acknowledge flag0
            next_flag = 1;                                                      //
set next_flag to 1
      }
  if(GPIO_PORTF_RIS_R&0x10){          // SW1 touched
    GPIO_PORTF_ICR_R = 0x10;          // acknowledge flag4
            music_flag ^= 1;                                              // toggle
music_flag
            keyboard_flag = 1;                                            //
initially keyboard mode should be set
            play_flag= 0;
      // auto-play mode should not initially start
      }

      // output is music is on/off
      if(music_flag==1)                               GPIO_PORTF_DATA_R = 0x08; // green LED is
ON
      else if(music_flag ==0)       GPIO_PORTF_DATA_R = 0x02; // red LED is OFF
}

// This function will play a song
// Inputs: note table for happy birthday song
void play_a_song(NTyp notetab[])
{
      unsigned char i=0, j;

      while (notetab[i].delay) {
            if(!music_flag || next_flag) break; // escape the function if music_flag == 0
or next_flag == 1
            if (!notetab[i].tone_index)
                  Sound_stop(); // silence tone, turn off SysTick timer
            else {
                  Sound_Init(tonetab[notetab[i].tone_index]/NUM_VALs);
            }

            // tempo control: play current note for specified duration
            for (j=0;j<notetab[i].delay;j++){
                  if(!music_flag || next_flag) break; // escape the function if
music_flag == 0 or next_flag == 1
                  Delay();
            }

            Sound_stop();
            i++;  // move to the next note
      }

      // pause after each play
      for (j=0;j<15;j++)
            Delay();
```

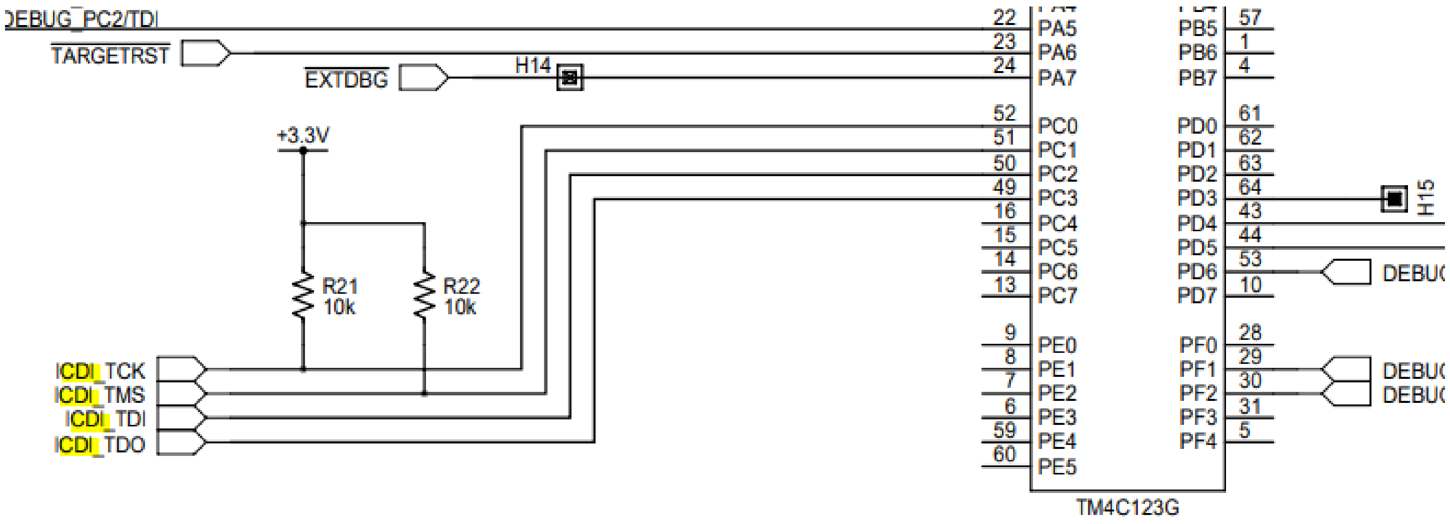
```
}

// Subroutine to wait 0.01 sec
// Inputs: None
// Outputs: None
void Delay(void){
      unsigned long volatile time;
      time = 3.125*727240*20/91;  // 0.01sec
  while(time){
            time--;
  }
}
```

Conclusion:

For project 1 part 1, which is the Music Box, it was a simple task. The only difficult thing about it was that I couldn’t escape the array where the music notes were at. Thankfully thanks to the professor’s help, I was able to escape because of the mistake of making a global variable become a local variable. Outside of this issue, the rest of this assignment was very simple and easy to finish.

For project 1 part 2, which is the Digital Piano, I had difficulty with some weird errors. At first I was getting power cycling issues which kept messing up my board and only allowed me to load my board once. This issue was because I did not initialize my PortC functions correctly. Luckily thanks to Andrew in our class, he was able to figure out the issue that Nicole and I had. Andrew figured out that my PortC_init() function was causing the issues because of this following schematic:



Here we see that PC3-0 were the main cause of power cycling issues because of their role towards the ICDI. I fixed my initialization functions and my code was finally able to load properly. Before I would have had to delete the memory from my Launchpad using the LM Flash Programmer and I was only able to load it up once before it crashed again. This also did not allow me to enter the debugger to continue to fix my code. My next issue was a minor but quite annoying one. I was not able to access my port F interrupts and that was because there was a DisableInterrupts() function in the main that did not allow any interrupts to happen. Removing that line allowed me to use the interrupts and finish this assignment. This project was not so difficult while looking back but it sure was hectic because of these minor issues.