Lab 5 Music Player and Audio Amp

Duc Tran & Brandon Wong

10/14/2013

1. Overview

1.1. Objectives: Why are we doing this project? What is the purpose?

The objectives of this project are to design, build and test a music player. Educationally, students are learning how to interface a DAC, how to design a speaker amplifier, how to store digital music in ROM, and how to perform DAC output in the background. Your goal is to play your favorite song.

1.2. Process: How will the project be developed?

The project will be developed using the LM3S1968 board. There will be three switches that the operator will use to control the music player. The system will be built on a solderless breadboard and run on the usual USB power. The system may use the on board switches or off-board switches. A hardware/software interface will be designed that allows software to control the player. There will be at least three hardware/software modules: switch input, DAC output, and the music player. The process will be to design and test each module independently from the other modules. After each module is tested, the system will be built and tested.

1.3. Roles and Responsibilities: Who will do what? Who are the clients?

The TAs are our clients, and Duc and Brandon are the engineers. Duc will be adding and modifying the switch interface. Brandon will be doing the same with the timer interface. Both the switch interface and timer interface were used in the previous labs. Together, we will split the load on the main logic and the speaker interface as well as deciding what music to load into the board.

1.4. Interactions with Existing Systems: How will it fit in?

The system will use the LM3S1968 board, a solderless breadboard, and the speaker as shown in Figure 5.1. It will be powered using the USB cable. You may use a +5V power from the lab bench, but please do not power the speaker with a voltage above +5V.

1.5. Terminology: Define terms used in the document.

SSI – alternately called serial peripheral interface (SPI) is used to interface medium-speed I/O devices.
Linearity – a measure of straightness of the static calibration curve.
Frequency response – a standard technique to describe the dynamic behavior of linear systems.
Loudness – characteristic of sound that relates to amplitude.
Pitch – property that allows the ordering of sound on a frequency-related scale.
Instrument – an embedded system that collects information, same as data acquisition system.

Tempo – is the speed or pace of music.
Envelope – a smooth curve outlining its extremes in amplitude.
Melody – main series of notes that stand out enabling you to remember the song.
Harmony – employed to decorate the musical effect of the song.

1.6. Security: How will intellectual property be managed?
The system may include software from StellarisWare and from the book. No software written for this project may be transmitted, viewed, or communicated with any other EE445L student past, present, or future (other than the lab partner of course). It is the responsibility of the team to keep its EE445L lab solutions secure.

2. Function Description

2.1. Functionality: What will the system do precisely?
If the operator presses the play/pause button the music will play or pause. If the operator presses the play/pause button once the music should pause. Hitting the play/pause again causes music to continue. The play/pause button does not restart from the beginning, rather it continues from the position it was paused. If the rewind button is pressed, the music stops and the next play operation will start from the beginning. There is a mode switch that allows the operator to control some aspect of the player. Possibilities include instrument, envelope or tempo.
There must be a C data structure to hold the music. There must be a music driver that plays songs. The length of the song should be at least 30 seconds and comprise of at least 8 different sounds. Although you will be playing only one song, the song data itself will be stored in a separate place and be easy to change. The player runs in the background using interrupts. The foreground (main) initializes the player, then executes **for(;;){}** do nothing loop. If you wish to include OLED output, this output should occur in the foreground. The maximum time to execute one instance of the ISR is xxxx. You will need public functions **Rewind**, **Play** and **Stop**, which perform operations like a cassette tape player. The **Play** function has an input parameter that defines the song to play. A background thread implemented with output compare will fetch data out of your music structure and send them to the DAC.
There must be a C data structure to store the sound waveform, or instrument. You are free to design your own format, as long as it uses a formal data structure (i.e., **struct**). The generated music must sound beautiful utilizing the SNR of the DAC. Although you only have to implement one instrument, it should be easy to change instruments.

2.2. Scope: List the phases and what will be delivered in each phase.
Phase 1 is the preparation; phase 2 is the demonstration; and phase 3 is the lab report. Details can be found in the lab manual.

2.3. Prototypes: How will intermediate progress be demonstrated?
A prototype system running on the LM3S1968 board and solderless breadboard will be demonstrated. Progress will be judged by the preparation, demonstration and lab report.

2.4. Performance: Define the measures and describe how they will be determined.
The system will be judged by three qualitative measures. First, the software modules must be easy to

understand and well-organized. Second, the system must employ an abstract data structures to hold the sound and the music. There should be a clear and obvious translation from sheet music to the data structure. Backward jumps in the ISR are not allowed. Waiting for SSI output to complete is an acceptable backwards jump. Third, all software will be judged according to style guidelines. Software must follow the style described in Section 3.3 of the book. There are three quantitative measures. First, the SNR of the DAC output of a sine wave should be measured. Second, the maximum time to run one instance of the ISR will be recorded. Third, you will measure power supply current to run the system. There is no particular need to optimize any of these quantitative measures in this system.

2.5. Usability: Describe the interfaces. Be quantitative if possible.
There will be three switch inputs. The DAC will be interfaced to a 32-ohm speaker.

2.6. Safety: Explain any safety requirements and how they will be measured.
If you are using headphones, please verify the sound it not too loud before placing the phones next to your ears.

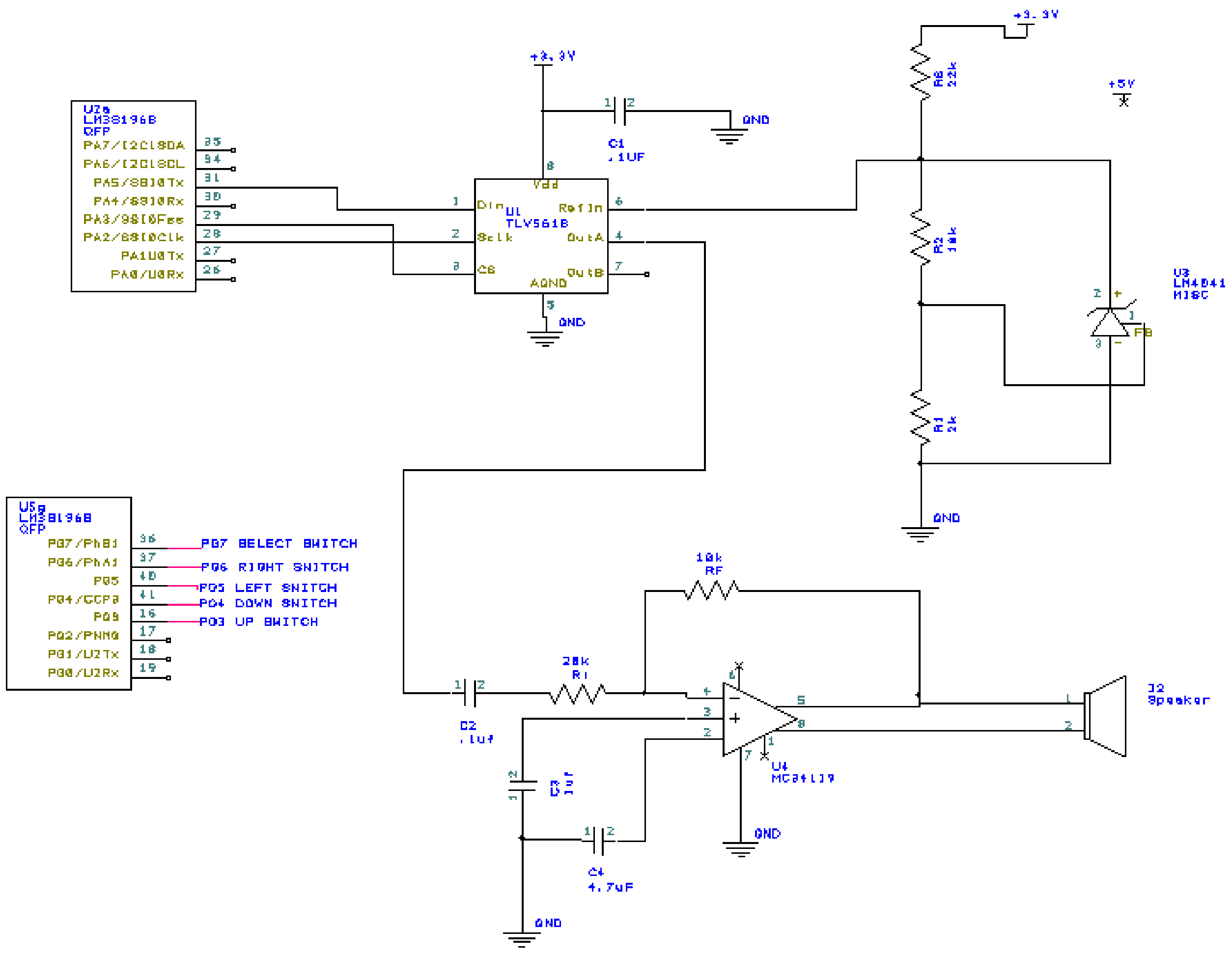
3. Deliverables

3.1. Reports: How will the system be described?
A lab report described below is due by the due date listed in the syllabus. This report includes the final requirements document.

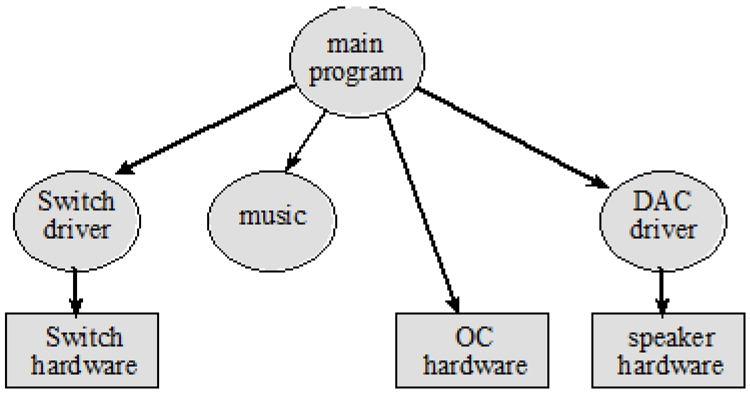
3.2. Audits: How will the clients evaluate progress?
The preparation is due at the beginning of the lab period on October 2, 2013.

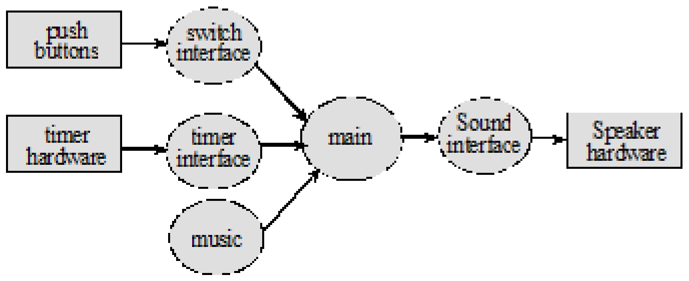
3.3. Outcomes: What are the deliverables? How do we know when it is done?
There are three deliverables: preparation, demonstration, and report. Preparation will be completed as mentioned above. The demonstration will be completed October 7, and the report will be completed on October 14, 2013.

4.0 Hardware Design



5.0 Software Design





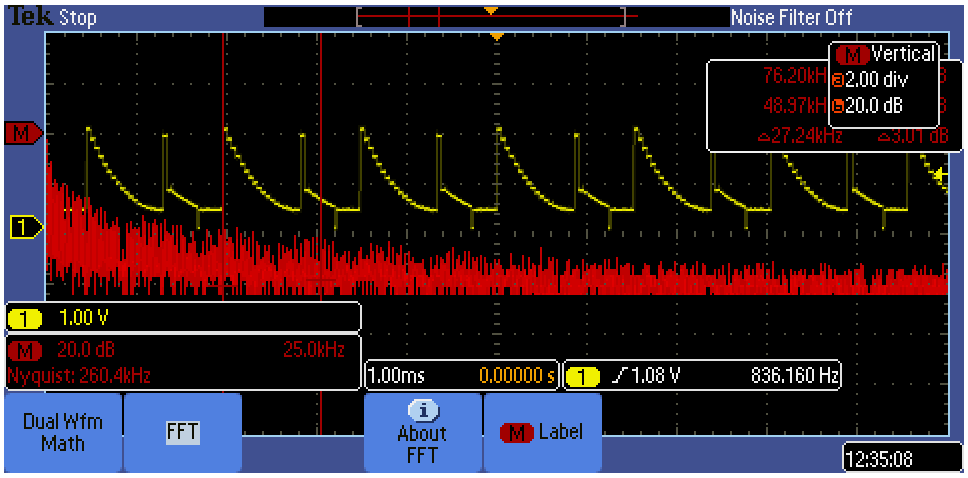
6.0 Measurement Data

Show the data and calculated resolution, range, precision and accuracy (procedure 1)

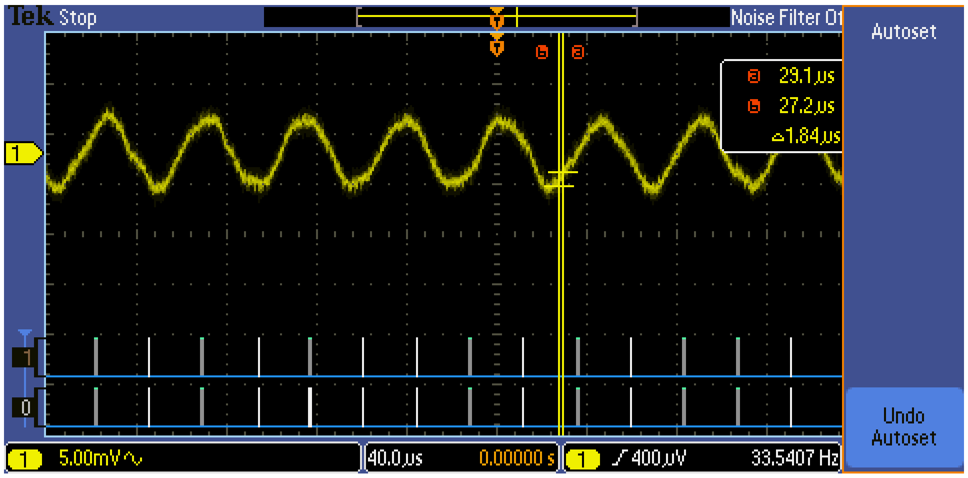
The DAC has 3.3 * 2^-12 volts of resolution with 12-bits. It's range is from 0 volts to 3.3 volts. It's precision is 4096 alternatives. The accuracy is +/- 0.29%.

Show the experimental response of DAC (procedure 2)

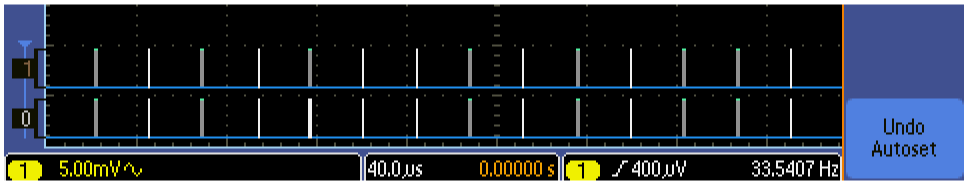
FFT of DAC response



DAC Wave



Show the results of the debugging profile (procedure 3)



We get a maximum time spent in the ISR of 460 ns. With a 50Mhz clock, that makes it 23 bus cycles.

Measurements of current required to run the system, with and without the music playing
Without music the current required to sun the system is: 0.094A
With music the current required to sun the system is: 0.106A

7.0 Analysis and Discussion

1) Briefly describe three errors in a DAC.

Gain error is a shift in the slope of Vout (Out A) versus digital input static response. Offset error is a shift in Vout versus digital input static response. Linearity the a measure of the straightness of the static calibration curve. Worst case of nonlinearity is nonmonotonic, which mean the analog voltage output decreases at any particular digital input.

2) Calculate the data available and data required intervals in the SSI/DAC interface. Use these calculations to justify your choice of SSI frequency.

Data available should overlap data required, hence set-up time and hold time must satisfy. Datasheet show SCLK pulse width low of 25ns, setup time of 10ns and hold time of 20ns. fssi = fbus/(CPSDVSR *(1+SCR)) => so 3MHz

3) How is the frequency range of a spectrum analyzer determined?

The lower frequency range of a spectrum analyzer is determine by the SB(sideband) noise of oscillator. The span of frequency range also depend on the sweep-time.

4) Why did we not simply drive the speaker directly from the DAC? I.e., what purpose is the MC34119?

The signal from the DAC is too small for the speaker to output. Therefore, we pass it through the MC34119 amplifier so that we get a larger signal that can be heard.

8: CODE

```
//Filename: MusicModule.c
//Author: Duc Tran, Brandon Wong
//Initial Creation Date: October 2, 2013
```

```
//Description: Handle music player
//Lab Number: W 2-3:30
//TA : Omar & Mahesh
//Date of last revision :October 14, 2013
//Hardware Configuration : Lab5_artist.sch

#define WHOLE_NOTE 64000
#define HALF_NOTE 32000
#define QUARTER_NOTE 16000
#define EIGHTH_NOTE 8000
#define QUARTER_NOTE_PAUSE 100
#define SIXTEENTH_NOTE 4000

#define EIGHTH_TRIPLE_NOTE 5333
#define EIGHTH_TRIPLE_NOTE2 5533
#define EIGHTH_TRIPLE_NOTE3 5733
#define EIGHTH_TRIPLE_NOTE4 5933
#define EIGHTH_TRIPLE_NOTE5 6133
#define EIGHTH_TRIPLE_NOTE6 6333



#define DOTTED_SIXTEENTH_NOTE 12000


#define FINAL_NOTE 65000


#define REST 0
#define PAUSE 0
#define QUIET 1
#define C_2 11945   // 65.406 Hz
#define DF_1 11274   // 69.296 Hz
#define D_1 10641   // 73.416 Hz
#define EF_1 10044   // 77.782 Hz
#define E_1 9480   // 82.407 Hz
#define F_1 8948   // 87.307 Hz
#define GF_1 8446   // 92.499 Hz
#define G_1 7972   // 97.999 Hz
#define AF_1 7525   // 103.826 Hz
#define A_1 7102   // 110.000 Hz
#define BF_1 6704   // 116.541 Hz
#define B_1 6327   // 123.471 Hz
#define C_1 5972   // 130.813 Hz
#define DF0 5637   // 138.591 Hz
#define D0 5321   // 146.832 Hz
#define EF0 5022   // 155.563 Hz
#define E0 4740   // 164.814 Hz
#define F0 4474   // 174.614 Hz
#define GF0 4223   // 184.997 Hz
#define G0 3986   // 195.998 Hz
#define AF0 3762   // 207.652 Hz
#define A0 3551   // 220.000 Hz
#define BF0 3352   // 233.082 Hz
#define B0 3164   // 246.942 Hz
#define C0 2986   // 261.626 Hz
#define DF 2819   // 277.183 Hz
```

```
#define D 2660   // 293.665 Hz
#define EF 2511   // 311.127 Hz
#define E 2370   // 329.628 Hz
#define F 2237   // 349.228 Hz
#define GF 2112   // 369.994 Hz
#define G 1993   // 391.995 Hz
#define AF 1881   // 415.305 Hz
#define A 1776   // 440.000 Hz
#define BF 1676   // 466.164 Hz
#define B 1582   // 493.883 Hz
#define C 1493   // 523.251 Hz
#define DF1 1409   // 554.365 Hz
#define D1 1330   // 587.330 Hz
#define EF1 1256   // 622.254 Hz
#define E1 1185   // 659.255 Hz
#define F1 1119   // 698.456 Hz
#define GF1 1056   // 739.989 Hz
#define G1 997   // 783.991 Hz
#define AF1 941   // 830.609 Hz
#define A1 888   // 880.000 Hz
#define BF1 838   // 932.328 Hz
#define B1 791   // 987.767 Hz
#define C1 747   // 1046.502 Hz
#define DF2 705   // 1108.731 Hz
#define D2 665   // 1174.659 Hz
#define EF2 628   // 1244.508 Hz
#define E2 593   // 1318.510 Hz
#define F2 559   // 1396.913 Hz
#define GF2 528   // 1479.978 Hz
#define G2 498   // 1567.982 Hz
#define AF2 470   // 1661.219 Hz
#define A2 444   // 1760.000 Hz
#define BF2 419   // 1864.655 Hz
#define B2 395   // 1975.533 Hz
#define C2 373   // 2093.005 Hz




struct Player{
        unsigned short Pitch;
        unsigned short Duration;
        //const struct Player *Next[8];

        //const struct Player *Song;
        //unsigned short SinTable[256];
};

typedef const struct Player SongType;
typedef SongType *SongPtr;

typedef const struct Player MusicStateType;
const struct Player *Ptc;
```

```
//MusicStateType MarioTheme[58]={

//		{E1, SIXTEENTH_NOTE},
//		{E1, EIGHTH_NOTE},
//		{E1, SIXTEENTH_NOTE},
//		{REST, SIXTEENTH_NOTE},
//		{C, SIXTEENTH_NOTE},
//		{E1, EIGHTH_NOTE},
//		{G1, QUARTER_NOTE},
//		{G, QUARTER_NOTE},

//		{C, EIGHTH_NOTE},
//		{REST, SIXTEENTH_NOTE},
//		{G, SIXTEENTH_NOTE},
//		{REST, EIGHTH_NOTE},
//		{E, EIGHTH_NOTE},
//		{REST, SIXTEENTH_NOTE},
//		{A, SIXTEENTH_NOTE},
//		{REST, SIXTEENTH_NOTE},
//		{B, SIXTEENTH_NOTE},
//		{REST, SIXTEENTH_NOTE},
//		{BF, SIXTEENTH_NOTE},
//		{A, EIGHTH_NOTE},

//		{G, EIGHTH_TRIPLE_NOTE},
//		{E1, EIGHTH_TRIPLE_NOTE},
//		{G1, EIGHTH_TRIPLE_NOTE},
//		{A1, EIGHTH_NOTE},
//		{F1, SIXTEENTH_NOTE},
//		{G1, SIXTEENTH_NOTE},
//		{REST, SIXTEENTH_NOTE},
//		{E1, SIXTEENTH_NOTE},
//		{REST, SIXTEENTH_NOTE},
//		{C, SIXTEENTH_NOTE},
//		{D1, SIXTEENTH_NOTE},
//		{B, SIXTEENTH_NOTE},
//		{REST, EIGHTH_NOTE},

//		{C, EIGHTH_NOTE},
//		{REST, SIXTEENTH_NOTE},
//		{G, SIXTEENTH_NOTE},
//		{REST, EIGHTH_NOTE},
//		{E, EIGHTH_NOTE},
//		{REST, SIXTEENTH_NOTE},
//		{A, SIXTEENTH_NOTE},
//		{REST, SIXTEENTH_NOTE},
//		{B, SIXTEENTH_NOTE},
//		{REST, SIXTEENTH_NOTE},
//		{BF, SIXTEENTH_NOTE},
//		{A, EIGHTH_NOTE},


//		{G, EIGHTH_TRIPLE_NOTE},
//		{E1, EIGHTH_TRIPLE_NOTE},
//		{G1, EIGHTH_TRIPLE_NOTE},
//              {A1, EIGHTH_NOTE},
//              {F1, SIXTEENTH_NOTE},
//              {G1, SIXTEENTH_NOTE},
//              {REST, SIXTEENTH_NOTE},
//              {E1, SIXTEENTH_NOTE},
//              {REST, SIXTEENTH_NOTE},
//              {C, SIXTEENTH_NOTE},
//              {D1, SIXTEENTH_NOTE},
//              {B, SIXTEENTH_NOTE},
//              {REST, EIGHTH_NOTE},

//};


MusicStateType MarioTheme[352]={

    {E1, SIXTEENTH_NOTE},   //1
    {E1, EIGHTH_NOTE},
    {E1, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {C, SIXTEENTH_NOTE},
    {E1, EIGHTH_NOTE},
    {G1, QUARTER_NOTE},
    {G, QUARTER_NOTE},

    {C, EIGHTH_NOTE},    //2
    {REST, SIXTEENTH_NOTE},
    {G, SIXTEENTH_NOTE},
    {REST, EIGHTH_NOTE},
    {E, EIGHTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {A, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {B, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {BF, SIXTEENTH_NOTE},
    {A, EIGHTH_NOTE},

    {G, EIGHTH_TRIPLE_NOTE},    //3
    {E1, EIGHTH_TRIPLE_NOTE},
    {G1, EIGHTH_TRIPLE_NOTE},
    {A1, EIGHTH_NOTE},
    {F1, SIXTEENTH_NOTE},
    {G1, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {E1, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {C, SIXTEENTH_NOTE},
    {D1, SIXTEENTH_NOTE},
    {B, SIXTEENTH_NOTE},
    {REST, EIGHTH_NOTE},

    {C, EIGHTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {G, SIXTEENTH_NOTE},
    {REST, EIGHTH_NOTE},
    {E, EIGHTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {A, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {B, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {BF, SIXTEENTH_NOTE},
    {A, EIGHTH_NOTE},


    {G, EIGHTH_TRIPLE_NOTE},
    {E1, EIGHTH_TRIPLE_NOTE},
    {G1, EIGHTH_TRIPLE_NOTE},
    {A1, EIGHTH_NOTE},
    {F1, SIXTEENTH_NOTE},
    {G1, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {E1, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {C, SIXTEENTH_NOTE},
    {D1, SIXTEENTH_NOTE},
    {B, SIXTEENTH_NOTE},
    {REST, EIGHTH_NOTE},

    {REST, EIGHTH_NOTE},
    {G1, SIXTEENTH_NOTE},
    {GF1, SIXTEENTH_NOTE},
    {F1, SIXTEENTH_NOTE},
    {EF1, EIGHTH_NOTE},
    {E1, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {AF, SIXTEENTH_NOTE},
    {A, SIXTEENTH_NOTE},
    {C, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {A, SIXTEENTH_NOTE},
    {C, SIXTEENTH_NOTE},
    {D1, SIXTEENTH_NOTE},

    {REST, EIGHTH_NOTE},
    {G1, SIXTEENTH_NOTE},
    {GF1, SIXTEENTH_NOTE},
    {F1, SIXTEENTH_NOTE},
    {EF1, EIGHTH_NOTE},
    {E1, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {C1, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {C1, SIXTEENTH_NOTE},
    {C1, QUARTER_NOTE},

    {REST, EIGHTH_NOTE},
    {G1, SIXTEENTH_NOTE},
    {GF1, SIXTEENTH_NOTE},
    {F1, SIXTEENTH_NOTE},
    {EF1, EIGHTH_NOTE},
    {E1, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {AF, SIXTEENTH_NOTE},
    {A, SIXTEENTH_NOTE},
    {C, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {A, SIXTEENTH_NOTE},
    {C, SIXTEENTH_NOTE},
    {D1, SIXTEENTH_NOTE},

    {REST, EIGHTH_NOTE},
    {EF1, EIGHTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {D1, SIXTEENTH_NOTE},
    {REST, EIGHTH_NOTE},
    {C, QUARTER_NOTE},
    {REST, QUARTER_NOTE},

    {C, SIXTEENTH_NOTE},
    {C, EIGHTH_NOTE},
    {C, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {C, SIXTEENTH_NOTE},
    {D1, EIGHTH_NOTE},
    {E1, SIXTEENTH_NOTE},
    {C, EIGHTH_NOTE},
    {A, SIXTEENTH_NOTE},
    {G, QUARTER_NOTE},

    {C, SIXTEENTH_NOTE},
    {C, EIGHTH_NOTE},
    {C, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {C, SIXTEENTH_NOTE},
    {D1, EIGHTH_NOTE},
    {E1, SIXTEENTH_NOTE},
    {REST, HALF_NOTE},

    {C, SIXTEENTH_NOTE},
    {C, EIGHTH_NOTE},
    {C, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {C, SIXTEENTH_NOTE},
    {D1, EIGHTH_NOTE},
    {E1, SIXTEENTH_NOTE},
    {C, EIGHTH_NOTE},
    {A, SIXTEENTH_NOTE},
    {G, QUARTER_NOTE},

    {E1, SIXTEENTH_NOTE},   //17
    {E1, EIGHTH_NOTE},
    {E1, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {C, SIXTEENTH_NOTE},
    {E1, EIGHTH_NOTE},
    {G1, QUARTER_NOTE},
    {G, QUARTER_NOTE},

    {C, EIGHTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {G, SIXTEENTH_NOTE},
    {REST, EIGHTH_NOTE},
    {E, EIGHTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {A, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {B, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {BF, SIXTEENTH_NOTE},
    {A, EIGHTH_NOTE},

    {G, EIGHTH_TRIPLE_NOTE},
    {E1, EIGHTH_TRIPLE_NOTE},
    {G1, EIGHTH_TRIPLE_NOTE},
    {A1, EIGHTH_NOTE},
    {F1, SIXTEENTH_NOTE},
    {G1, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {E1, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {C, SIXTEENTH_NOTE},
    {D1, SIXTEENTH_NOTE},
    {B, SIXTEENTH_NOTE},
    {REST, EIGHTH_NOTE},

    {C, EIGHTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {G, SIXTEENTH_NOTE},
    {REST, EIGHTH_NOTE},
    {E, EIGHTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {A, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {B, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {BF, SIXTEENTH_NOTE},
    {A, EIGHTH_NOTE},


    {G, EIGHTH_TRIPLE_NOTE},
    {E1, EIGHTH_TRIPLE_NOTE},
    {G1, EIGHTH_TRIPLE_NOTE},
    {A1, EIGHTH_NOTE},
    {F1, SIXTEENTH_NOTE},
    {G1, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {E1, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {C, SIXTEENTH_NOTE},
    {D1, SIXTEENTH_NOTE},
    {B, SIXTEENTH_NOTE},
    {REST, EIGHTH_NOTE},

    {E1, SIXTEENTH_NOTE},   //22
    {C, EIGHTH_NOTE},
    {G, SIXTEENTH_NOTE},
    {REST, EIGHTH_NOTE},
    {AF, EIGHTH_NOTE},
    {A, SIXTEENTH_NOTE},
    {F1, EIGHTH_NOTE},
    {F1, SIXTEENTH_NOTE},
    {A, QUARTER_NOTE},

    {B, EIGHTH_TRIPLE_NOTE},
    {A1, EIGHTH_TRIPLE_NOTE},
    {A1, EIGHTH_TRIPLE_NOTE},
    {A1, EIGHTH_TRIPLE_NOTE},
    {G1, EIGHTH_TRIPLE_NOTE},
    {F1, EIGHTH_TRIPLE_NOTE},
    {E1, SIXTEENTH_NOTE},
    {C, EIGHTH_NOTE},
    {A, SIXTEENTH_NOTE},
    {G, QUARTER_NOTE},

    {E1, SIXTEENTH_NOTE},
    {C, EIGHTH_NOTE},
    {G, SIXTEENTH_NOTE},
    {REST, EIGHTH_NOTE},
    {AF, EIGHTH_NOTE},
    {A, SIXTEENTH_NOTE},
    {F1, EIGHTH_NOTE},
    {F1, SIXTEENTH_NOTE},
    {A, QUARTER_NOTE},

    {B, SIXTEENTH_NOTE},
    {F1, EIGHTH_NOTE},
    {F1, SIXTEENTH_NOTE},
    {F1, EIGHTH_TRIPLE_NOTE},
    {E1, EIGHTH_TRIPLE_NOTE},
    {D1, EIGHTH_TRIPLE_NOTE},
    {C, SIXTEENTH_NOTE},
    {G, EIGHTH_NOTE},
    {G, SIXTEENTH_NOTE},
    {C0, QUARTER_NOTE},

    {E1, SIXTEENTH_NOTE},
    {C, EIGHTH_NOTE},
    {G, SIXTEENTH_NOTE},
    {REST, EIGHTH_NOTE},
    {AF, EIGHTH_NOTE},
    {A, SIXTEENTH_NOTE},
    {F1, EIGHTH_NOTE},
    {F1, SIXTEENTH_NOTE},
    {A, QUARTER_NOTE},

    {B, EIGHTH_TRIPLE_NOTE},
    {A1, EIGHTH_TRIPLE_NOTE},
    {A1, EIGHTH_TRIPLE_NOTE},
    {A1, EIGHTH_TRIPLE_NOTE},
    {G1, EIGHTH_TRIPLE_NOTE},
    {F1, EIGHTH_TRIPLE_NOTE},
    {E1, SIXTEENTH_NOTE},
    {C, EIGHTH_NOTE},
    {A, SIXTEENTH_NOTE},
    {G, QUARTER_NOTE},

    {E1, SIXTEENTH_NOTE},
    {C, EIGHTH_NOTE},
    {G, SIXTEENTH_NOTE},
    {REST, EIGHTH_NOTE},
    {AF, EIGHTH_NOTE},
    {A, SIXTEENTH_NOTE},
    {F1, EIGHTH_NOTE},
    {F1, SIXTEENTH_NOTE},
    {A, QUARTER_NOTE},

    {B, SIXTEENTH_NOTE},
    {F1, EIGHTH_NOTE},
    {F1, SIXTEENTH_NOTE},
    {F1, EIGHTH_TRIPLE_NOTE},
    {E1, EIGHTH_TRIPLE_NOTE},
    {D1, EIGHTH_TRIPLE_NOTE},
    {C, SIXTEENTH_NOTE},
    {G, EIGHTH_NOTE},
    {G, SIXTEENTH_NOTE},
    {C0, QUARTER_NOTE},

    {C, SIXTEENTH_NOTE},    //30
    {C, EIGHTH_NOTE},
    {C, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {C, SIXTEENTH_NOTE},
    {D1, EIGHTH_NOTE},
    {E1, SIXTEENTH_NOTE},
    {C, EIGHTH_NOTE},
    {A, SIXTEENTH_NOTE},
    {G, QUARTER_NOTE},

    {C, SIXTEENTH_NOTE},
    {C, EIGHTH_NOTE},
    {C, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {C, SIXTEENTH_NOTE},
    {D1, EIGHTH_NOTE},
    {E1, SIXTEENTH_NOTE},
    {REST, HALF_NOTE},

    {C, SIXTEENTH_NOTE},
    {C, EIGHTH_NOTE},
    {C, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {C, SIXTEENTH_NOTE},
    {D1, EIGHTH_NOTE},
    {E1, SIXTEENTH_NOTE},
    {C, EIGHTH_NOTE},
    {A, SIXTEENTH_NOTE},
    {G, QUARTER_NOTE},

    {E1, SIXTEENTH_NOTE},   //33
    {E1, EIGHTH_NOTE},
    {E1, SIXTEENTH_NOTE},
    {REST, SIXTEENTH_NOTE},
    {C, SIXTEENTH_NOTE},
    {E1, EIGHTH_NOTE},
    {G1, QUARTER_NOTE},
    {G, QUARTER_NOTE},

    {E1, SIXTEENTH_NOTE},
    {C, EIGHTH_NOTE},
    {G, SIXTEENTH_NOTE},
    {REST, EIGHTH_NOTE},
    {AF, EIGHTH_NOTE},
    {A, SIXTEENTH_NOTE},
    {F1, EIGHTH_NOTE},
    {F1, SIXTEENTH_NOTE},
    {A, QUARTER_NOTE},

    {B, EIGHTH_TRIPLE_NOTE},
    {A1, EIGHTH_TRIPLE_NOTE},
    {A1, EIGHTH_TRIPLE_NOTE},
    {A1, EIGHTH_TRIPLE_NOTE},
    {G1, EIGHTH_TRIPLE_NOTE},
    {F1, EIGHTH_TRIPLE_NOTE},
    {E1, SIXTEENTH_NOTE},
    {C, EIGHTH_NOTE},
    {A, SIXTEENTH_NOTE},
    {G, QUARTER_NOTE},

    {E1, SIXTEENTH_NOTE},
    {C, EIGHTH_NOTE},
    {G, SIXTEENTH_NOTE},
    {REST, EIGHTH_NOTE},
    {AF, EIGHTH_NOTE},
    {A, SIXTEENTH_NOTE},
    {F1, EIGHTH_NOTE},
    {F1, SIXTEENTH_NOTE},
    {A, QUARTER_NOTE},

    {B, SIXTEENTH_NOTE},
    {F1, EIGHTH_NOTE},
    {F1, SIXTEENTH_NOTE},
    {F1, EIGHTH_TRIPLE_NOTE},
    {E1, EIGHTH_TRIPLE_NOTE},
    {D1, EIGHTH_TRIPLE_NOTE},
    {C, SIXTEENTH_NOTE},
    {G, EIGHTH_NOTE},
    {G, SIXTEENTH_NOTE},
    {C0, QUARTER_NOTE},

    {C, DOTTED_SIXTEENTH_NOTE},
    {G, DOTTED_SIXTEENTH_NOTE},
    {E, EIGHTH_NOTE},
    {A, EIGHTH_TRIPLE_NOTE},
    {B, EIGHTH_TRIPLE_NOTE2},
    {A, EIGHTH_TRIPLE_NOTE3},
    {AF, EIGHTH_TRIPLE_NOTE4},
    {BF, EIGHTH_TRIPLE_NOTE5},
    {AF, EIGHTH_TRIPLE_NOTE6},

    {G, FINAL_NOTE},
    {REST, WHOLE_NOTE}
};


MusicStateType Carol[]={
// carol of the bells
//1
  {C,    QUARTER_NOTE},
  {B,    EIGHTH_NOTE},
  {C,    EIGHTH_NOTE},
  {A,    QUARTER_NOTE_PAUSE},
  {QUIET,PAUSE},
//2
  {C,    QUARTER_NOTE},
  {B,    EIGHTH_NOTE},
  {C,    EIGHTH_NOTE},
  {A,    QUARTER_NOTE_PAUSE},
  {1,PAUSE}
};

MusicStateType BirthdaySong[25]={

                {C0, EIGHTH_NOTE},
                {C0, EIGHTH_NOTE},
                {D, QUARTER_NOTE},
                {C0, QUARTER_NOTE},
                {F, QUARTER_NOTE},

                {E, HALF_NOTE},
                {C0, EIGHTH_NOTE},
                {C0, EIGHTH_NOTE},
                {D, QUARTER_NOTE},
                {C0, QUARTER_NOTE},

                {G, QUARTER_NOTE},
                {F, HALF_NOTE},
                {C0, EIGHTH_NOTE},
                {C0, EIGHTH_NOTE},
                {C, QUARTER_NOTE},

                {A, QUARTER_NOTE},
                {F, QUARTER_NOTE},
                {E, QUARTER_NOTE},
                {D, QUARTER_NOTE},
                {BF, EIGHTH_NOTE},

                {BF, EIGHTH_NOTE},
                {A, QUARTER_NOTE},
                {F, QUARTER_NOTE},
                {G, QUARTER_NOTE},
                {F, QUARTER_NOTE},
};

unsigned short PauseIndex;
unsigned long Duration;
unsigned short Index;
/****************Music_Init***************
 Description: Initializes stepper motor
 Input: none
 Output: none  */
void Music_Init(void){
		PauseIndex = 0;
		Index = 0;
		Ptc = &MarioTheme[0];
		Duration = Ptc->Duration;
}
/****************Music_Play***************
 Description:Play music
 Input: none
 Output: none  */
#define TIMER0_TAILR_R          (*((volatile unsigned long *)0x40030028))
void Music_Play(void){
		if(Index == 352){
				Index = 0;
		}
		Ptc = &MarioTheme[Index];
		TIMER0_TAILR_R = Ptc->Pitch;
		Duration = Ptc->Duration;
		Index = Index+PauseIndex;
}
/****************Music_Rewind***************
 Description: Rewind music to beginning
 Input: none
 Output: none  */
void Music_Rewind(void){
		Ptc = &MarioTheme[0];
		TIMER0_TAILR_R = 0;
		Duration = Ptc->Duration;
		Index = 0;
}
/****************Music_Stop***************
 Description: Stop music, next play will be from the beginning
 Input: none
 Output: none  */
void Music_Stop(void){
		PauseIndex =0;
		Ptc = &MarioTheme[0];
		TIMER0_TAILR_R = 0;
		Duration = Ptc->Duration;
		Index = 0;
}
/****************Music_Pause***************
 Description: Pause music, next play will be where it left off
 Input: none
 Output: none  */
void Music_Pause(void){
		PauseIndex ^=1;
		TIMER0_TAILR_R  = 0;
}
```