1. Introduction

The purpose of this lab is to explore the design of firmware using timer driven interrupts. Timer interrupts will be used to create tones on the Piezo speaker and to play 2 tunes. For an A-Level project, external interrupts must be used to drive the keypad and the Piezo speaker will be driven by a TOC Timer.

2. Prerequisites

- StmCubeIDE is installed on your machine.
- Your MSOE Development Board with the following components assembled:
  - STM32 Microcontroller
  - LED Bar Array (with drivers)
  - LCD Display (with drivers)
  - SysTick Counter (with drivers)
  - Wired and functional 4x4 Keypad
  - A working Piezo speaker

3. Activities

3.1 Create Lab4Keypad Project

- Copy your working Lab4HexKeys Project within CubeIDE in the ProjectExplorer tab by copy-and-pasting.
- Name your project Lab5Tune
- Start a fresh main.c file

3.2 Piezo Speaker API

- Write a piezo(.h/.c) API
  - “PiezoInit” method to configure the appropriate perpherals
  - “Play” method that includes the frequncy(Hz) and length of time(ms) to play the tone
  - Either Interrupts or TOC should be used to produce the piezo square wave.
- Hints:
  - The Piezo speaker is on PB4.
  - The general purpose timers run at 16MHz by default.
  - Use TIM3 to generate the pulse purpose. (Helpful if TOC is used)

3.3 Create a music(.c/.h) API

- Using your piezo API create call which allow you to play music notes.
  - Notes have duration(1/16, 1/4, 1/2, and whole)
  - Notes have an assigned tone (A,A#,B,C,C#,D,D#,E,F,F#,G,G#,Rest)
  - A struct of enums can be used to package the information related to a note.
  - By making an array of notes you can compose a song.
- An interrupt driven timer should be used to stop the playing each note and transition to the next note.

3.4 Keypad API (.c/.h) API (Interrupt Driven)

- Rework your Keypad API making it interrupt driven.
- Note that Algorithm #2 for keypad scanning is required for interrupts on all keys to be detected.

3.5 Create tune player in main.c

1. Begin with a welcome screen on the LCD.
2. Prompt for the Beats per minute(BPM) on the LCD.
   a) Note this will set be used to set the duration of a quarter note.
   b) Entered digits should appear on the screen as they are typed.
3. Give the user the option to play the scale or one/several custom tunes.
   a) The interface deliver song options is up to the student but should be clean and clear.
4. While the song is playing:
   a) the title should display on line 1 of the LCD.
   b) the index of current playing note and the note should display on line 2 of the LCD. (Note name and duration)
5. The user should be able to manipulate the playing of the tune while it is playing.
   a) The hash key should pause.
   b) The astrix key should play.
   c) The 0 key should play the current tune backwards.
   d) The 1 key should pause the current playing tune and prompt for a new BPM.
   e) The A key should stop and set the scale of quarter notes as the current tune.
   f) The B-D keys can be used of other custom tunes if you wish.
   g) Note: You should be able to change songs in the middle while another song is playing.

4. Deliverables

- All code should include a header block with:
  - Your name
  - Course number and section
  - Assignment name
  - File name
  - List of any dependencies.
  - Description
- Print out you document code in the following order: (*italics implies optional*)
  - piezo.h
  - *piezo.c*
  - music.h
  - music.c
  - keypad.h
  - keypad.c
  - *player.h*
  - *player.c*
  - main.c
- Print in Light mode with Line numbers, filenames, and time and date using CubeIDE or Notepad++
- Staple together packet in the top left, in order, with rubric cover sheet.

- Phase one target – Timer Interrupt for tone generation – (No timers) (Week9Lab)
- Phase two target – Interrupt driven keypad (Week10Lab)
- Final packet and demonstration due at the beginning of Week11 Lab.

5. Grading Levels

D-Level – Tone Generation Polling
C-Level – Application with Tone Generation Using Interrupts
B-Level – Application with Tone Generation and Keypad Utilizing Interrupts
A-Level – Full Application with Keypad Utilizing Interrupts and Tone Generation Using TIM3 TOC Functions