Homework 2: Song Composer Report

Submitted: October 20, 2017

Sabbir Ahmed

1 Description

This project utilizes the AVR Butterfly through the terminal to allow a user to create up to 4 songs at a time, along with a title and author, and play them back.

This document serves as the final report for the project. The report details the implementation of the project using C and specific AVR libraries to handle memory allocation and user interface with the AVR system. The usage of the program along with its constraints are included as well.

2 Implementation

The project was implemented using C and the `AVR/io.h` , `AVR/pgmspace.h` and `util/delay.h` libraries included in AVR-GCC. `U0_UART` was included with the project prompt to direct all interfaces to the AVR hardware.

2.1 Conversion of User Inputs and Notes

Notes comprise of an ASCII character, (A-G, and R) and a duration of the note (0-31). They are converted from user inputted strings to `uint8_t` by left-shifting the integer value mapped to the ASCII character by 5 bits and adding the duration.
Unpacking the notes is done by right-shifting by 5 bits to retrieve the ASCII character or by bitwise logical and-ing with 0x1F (0b0001 1111) to retrieve the duration.

2.2 Playing Notes

Notes are played by the unpacked ASCII characters and durations. Each ASCII characters are associated with a frequency described in `music.h` . The frequency is used to compute the half period and the number of iterations for the Port B pin to set on and off. The following snippet demonstrates the playing of individual notes:

```
/*
 * Plays individual notes from the songs. */
void play_note(uint8_t letter_ascii, uint8_t quarters) {

    unsigned int num_iter, half_periods, freq;

    // assign the frequency associated with the ASCII
    // character note
    freq = get_frequency(letter_ascii);
    // compute the half period; 500000 / freq
    half_periods = FREQ_TO_MS / freq;
    // compute the number of iterations;
    // (freq / 4) * duration
    num_iter = freq / 4 * quarters;

    // loop for num_iter times to play a note
    for (int i = 0; i < num_iter; ++i) {

        // set port B pin 5 low, and delay for
        // half_period ms
        PORTB = (PORTB5_SPEAKER_MASK & 0x00);
        delay_ms(half_periods);

        // set port B pin 5 high, and delay for
        // half_period ms
        PORTB = (PORTB5_SPEAKER_MASK & 0xFF);
        delay_ms(half_periods);

    }

}
```

2.3 Memory Management

Since the memory on the chip was limited, the data on the program space was forced to be under 1 kB. Extra steps had to be taken to prevent memory overflow. The `PROGMEM` macro was utilized in appropriate instances to explicitly allocate constant variables to the program space. Strings for displaying menu prompts and user alerts were reused across the program to minimize the number of string data type variables created.

3 Usage

The project depends on user inputs from the terminal. RealTerm or a similar serial terminal program is required to interface the board after uploading the compiled binaries. Once in the terminal, the Main Menu is displayed, prompting the user for a choice of 4 options: List, Play, New and Exit.

3.1 List Menu

The List option displays all the song titles in the database (represented by the `char song_title_list[ NUMBER_OF_SONGS ][ USER_LINE_MAX ]` array). The database starts empty when the program is initialized.

3.2 Play Menu

The Play option allows the user the play any songs created. The Play Menu consists of two options for locating the song to play: by title and by the index in the database. If the user chooses to search the song by the title, they are required to input their query in the proceeding prompts. The user's query is then compared with all the titles in the database. The program then plays the song with the title scoring the highest when being compared with the query. If the latter option is chosen, the user is prompted for a valid index of the song to be played.

3.3 Create Menu

The Create Menu is displayed when the New option is selected. This feature allows the user to add their own song notes along with their desired song title and location of the database for storing the song.

3.4 Exit

The Exit option breaks out of the loop and immediately terminates the program.

3.5 Invalid Choices

The program will continue displaying the Main Menu until a valid choice is inputted by the user.

4 Code

The C scripts used for the implementation has been attached alongside the report.

4.1 main.c

Driver file for Homework 2: Song Composer. This script contains the declarations and implementations of the menu functions and other functions to handle user inputs and outputs.

4.2 music.h

Contains the declarations for the functions in music.c and main.c

4.3 music.c

Contains the implementation of all the functionalities related to parsing user inputted song notes, packing and unpacking song notes, and utilizing the ports of the AVR to produce the notes via its speaker.