Neeraj Basu

3/27/15

EC 450

Professor Giles

Homework #5 – Music Player

**Design:**

I began by mapping all my ports to the external buttons using an 8-bit mask. Starting with Joy to the World, which was one octave, I defined the notes that would be used in both the songs. I managed to get all these notes by calculating what half periods amounted to what frequencies and then using a chart to see what frequency mapped to what note. I then stored all of these values into an array called SIZE\_NOTES\_ARRAY. In order to distinguish between notes of the same frequency, I simply added an inaudible frequency between them that would act as the break. I then defined the score using the bit mask values of each note for Joy to the World. I also used a NOTE\_MASK that held a value of 0x1F that was used in order to isolate the bits containing the note from the bits containing the duration.

In order to make sure each note had a different duration, I made an array that contained durations from rests (1 unit) to dotted halves (96 units). These were then also given 8 bit masks and were used in the first half of the 8 bit char that held both duration and note (This will be explained better in a bit).

I defined an array JOY\_TO\_WORLD\_LENGTH that held an array the length of the song and then wrote out the combination of note and duration for each beat of the song. Essentially each note was the length of a char and the first 4 bits held the duration of the song while the second four bits held the actual note. By adding these together I was able to store them all into an array that defined the first song.

\_ \_ \_ \_ | \_ \_ \_ \_

Duration of Note Actual Note

The same process was done for my second song which was the Mario Theme song.

I then created a global variable for the default tempo that essentially acted as the conductor for the Microcontroller. When I wanted to change the tempo of the song to either speed it up or slow it down, the tempo would be scaled up or down respectively depending on the number of button pushes the user inputted. I also established variables that handled what state the play:

0 – song is playing

1 –song is paused

2 –start of a new song

3 - end of a song

I also set up global variable counters that monitored the duration, score and flash. I used timerA as the frequency generator which was responsible for the frequencies that created each note and held the TA0CCTL0 register which was the output for the speaker. I also used the WDT as the conductor of the system which was checking for interrupts at every interval. I then set up the IO pins for each of the pushbuttons.

Functions were written for:

pause\_song: toggled the OUTMOD\_4 between 0 and 4 in order to play or pause a song. This function also shut off the board LED when a song was playing.

Play\_song\_slower: added 0.125 to the default tempo every time the button was pushed.

Play\_song\_faster: subtracted -0.125 from the default tempo every time the button was pushed

Restart\_song: Reset the clock, toggle OUTMOD\_4, reset the tempo to the default, and change the LED to a steady state letting the user know the song is ready to be played.

I also utilized two more pushbuttons each of which corresponded to either JTTW or the Mario theme song. When those buttons were pressed the current song was loaded to whichever song was selected as well as altering the length of the song.

In the interrupt handler I basically aligned whatever buttons where synced with whatever action was needed and called the appropriate function.