Jesse Belanger

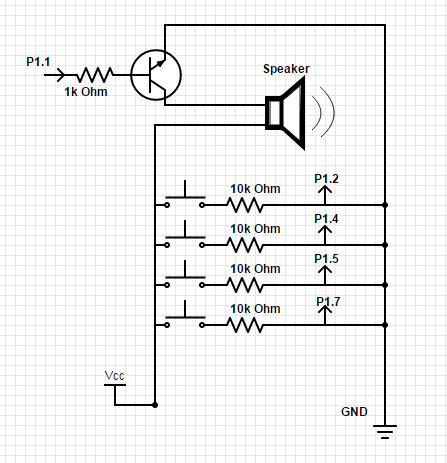
EC 450

Homework 5

3/23/15

**Device Description**

The device consists of a speaker, four buttons, a transistor, and a number of wires and resistors. The layout of the parts on the breadboard is shown in the diagram below:



The speaker is controlled to play music via GPIO pin P1.1, and the sound is amplified using a transistor. User interaction is facilitated through the four buttons on the breadboard, as well as the button on the MSP430 itself. Seven functions are implemented across two modes.

*Off Mode*

No music is playing – neither actively playing nor paused and awaiting a command to resume. This mode is active when the system boots/before any songs have been played, and after a song has fully completed. The buttons provide two functions in this mode:

* **Select**: by pressing the fourth button (the one furthest away from the speaker), you can toggle between the two stored tracks.
* **Play**: by pressing the play button on the MSP430, you can start the playback of your chosen song. This will cause the device to transition to play mode, which will be indicated by the illumination of the green LED.

None of the other buttons offer functionality in this mode.

*Play Mode*

The chosen song is playing, which can mean that it is either actively making music, or it is paused. You enter this mode by pressing the button on the MSP430. Five functions are offered, utilizing each of the device’s five buttons:

* **Pause/Unpause**: the song can be paused while actively playing by pressing the button on the MSP430. The red LED will turn on, and the green LED will turn off, to indicate the change in state. The song can be resumed by pressing the same button again.
* **Restart**: the song can be restarted by pressing the first button on the breadboard (the one closest to the speaker). The song cannot be restarted while paused.
* **Slow Down**: by pressing the second button, the song’s tempo will be cut in half. This can only be done while the song is actively playing, not while it is paused.
* **Speed Up**: by pressing the third button, the song’s tempo will be doubled. Similar to the slow down function, this can only be done during active playback.
* **Reset Tempo**: pressing the fourth button (the same one used for select in off mode) allows you to reset the tempo of the song to default.

**Code Design**

The functionality for this device is mainly spread across two major timers: Timer A and the WDT. Timer A is used exclusively for generating the various notes, while the WDT is used for controlling the playback of the songs and interpreting signals from the buttons. The code is, at its foundation, based off of the example program tone0 that Professor Giles provided.

An important part of my design is the notes header file. In that file I define the frequency for each of the notes, as well as the number of cycles in a quarter note (based on a default tempo of 120 bpm), and the information needed to play the songs. Each song is defined by two arrays of equal length. One contains each of the song’s notes in the order they are to be played, and the other contains the length each note is to be played for, also in order. The length of each note is expressed as a multiplier of the quarter note duration (e.g. an eighth note is represented as ½).

The first few functions in my code are rather straightforward. In my main program, I initialize the default song on boot to be Joy to the World, and set up the LEDs, timers, and buttons to be used. The button handler that is defined to work with the MSP430’s on-board button basically establishes a state machine, which establishes which playback modes can be transitioned to under certain conditions.

The real work happens in the WDT interrupt handler. At the start, it reads each of the buttons for use in the rest of the function. If the device is in off mode, no action has to be taken unless the select button has been pressed, in which case the selection bit is switched, and the song playback variables are set to correspond to the other song.

In play mode, the handler first checks of the various breadboard buttons. If the reset button has been pressed, the note and duration variables are forcefully reset to the data for the first note of whichever song has been playing. The timer is also cleared. If any of the tempo changing buttons have been pressed, a global *tempo* variable is changed to the value corresponding to that tempo (1 for normal, 2 for slow, ½ for fast).

Lastly while in play mode, the handler directs the flow of the song. Each time the interrupt handler is entered, a *duration* variable counting the amount of time to hold a note is decremented. When it gets to zero, if there are still notes left to play in the song, *duration* is set to be the length of a quarter note, multiplied by the multiplier contained in the timing array, and also multiplied by the tempo. To change the note, the TA0CCR0 register is changed to the next note from the notes array. Then a *count* variable is incremented to point at the data for the next note. If the count becomes equal to the length of the arrays, then play mode is turned off and the sound system is shut off.

**Design Comments**

A few comments on quirks, limitations, virtues of my design:

* Every once in a while, the select signal from the breadboard button will not take in the system. That is a hardware rather than software issue. If I press the button in just the right manner for a somewhat specific amount of time, the button works as intended, but if I am not careful, then it can be missed. This is likely because I didn’t do any formal debouncing of the buttons in my code.
* I purposefully designed the device so that the tempo from the most recent playback will carry over through a song restart, or the start of a new song after the end of an old one.
* It was not in my original design to have three buttons to control change in tempo, but I had a lot of trouble transitioning between three speeds using my two buttons. Speeding up would always change the tempo to fastest, and slowing down to slowest, even if you were slowing down from fastest or vice versa. I wrote my original state machine to always go to the normal tempo in the middle, but the WDT samples too quickly for a human reaction, so that never worked. However, the select button was unused in play mode, so I just made that multi-functional.
* I consider it a virtue of this design that it supports three tempos.