################################
Name: Priscilla Usmani- pusmani
Lab section: 3 MW TA: Jay Roldan
Due: 6/05/13
Lab Partner: Abdullah Tamimi- atamimi
################################

Title:

Lab 6- Timers and Interrupts

Purpose:

The purpose of this lab is to implement more codes from previous lab, lab 5. In this lab, we use the interrupts from a timer to implement a similar function. This is then used to drive a piezoelectric speaker at certain frequencies to play a tune. In this assignment we used the interrupts from a timer to implement a music playing program that drives a piezoelectric speaker at specified frequencies for specified durations to play a tune. We are given the main.cpp program that contains the top level program and iterates over any array of notes and duration to play a melody.

Procedure:

In order to prep for this lab, we read the entire lab and read the pdf files from ecommons.
We implemented the required subroutines in assembly to complete this program.

- SetupPort()

This function should set up Port D pin 8 from digital output. In SetupPort we used the knowledge from lab5 to do the virtual addresses. Then we wrote a command to test port D and create a loop and toggle PORTD. Connecting the speaker to the correct pin and listen for the sound.

- SetupTimer()

This function configures timer 1 and the corresponding interrupts, but it should not enable the timer. We set up timer 1, clear T1C0N by putting it in a known state, set the prescaler, clear the counts if it was used. Then we set up timer1 corresponding to the interrupts by setting the interrupt priority, clearing any prior interrupt, enable the interrupts, then connected the speaker in the correct pin and listened for the sound.

- PlayNote(int,int,int)

In this last function, it should play note. We use a delay subrouting for tone duration and silence. This is the main part of the program and contains main.cpp, it holds three variable, size, melody, and duration.

Algorithms and other data:

Pseudo Code:

Push Register

Pop Register

Jump to customized routine by placing a jump at the vector 4 interrupt vector offset

The .global from main.cpp

Start Program

Set up PortD pin 8 for digitial output

- test portD create a loop and toggle PORTD
- connect speaker to the correct pin and listen for sound

Configure Timer 1 and corresponding interrupts, should not enable timer

- set up timer1
- clear t1CON- put it in known state
- set T1CKPS- set the prescalar
- clear tmr1 value- clear the counts if it was used
- set pr1- set the period

Set up timer1 corresponding interrupts

- set t1ip- set the interrupt prioirty
- clear t1if- clear any prior interrupt
- enable t1ie- enable the interrupts

Play note

- set tone frequency and gets the period, checks for zero input using BEQ
- set period
- enables the timer
- disables the timer
- calculates the silence after the tone, full note minus tone duration
- puts registers back in order for return

Skip note

- branch in case zero frequency, delay for 500 miliseconds
- enable time
- disable timer

t1_isr

- program will go to ISR everytime and interrupt occurs
- clear t1if- clear any prior interrupt
- pop registers
- ERET return instruction

What Went Wrong or What Were The Challenges:

The challenge in the program playing the note. It was hard to get the note to play the right melody because our loops weren't working. We couldn't get the loop to work the right way and also understanding push and pop. After reading up on more we learned how to push and pop, and after trials on the loop we were finally able to get it to play the right melody.

Other Information:

How did you go about designing your program?

My partner and I didn't have to design the program much because we had lab6notes. When we opened the program the pseudo code was well written out, it showed all the functions and specific instructions within the functions. Our only job was to figure out how to go about the instructions and working the loops.

How did you convert the frequency to a note duration?

I converted the frequency to a note duration by taking the frequency of the board and multiplying it by the frequency of the note. Then I divided the value by 256*2.

How did you implement the silent part of a note after a tone?

I implemented the silent part of a note after a tone by subtracting the duration of the note from the full note duration then jumping to a delay subroutine.

What sort of bugs did you encounter while writing your program?

We didn't encounter any bugs because we followed the instructions given on lab6notes accurately and was able to compile the program perfectly. Although we had the program play the wrong notes, it wouldn't be considered a bug.

Why do we need the extra ½ when calculating the prescalar register value?

We need an extra ½ when calculating the prescalar register value because only half the time is used for playing a frequency.

Conclusion:

In conclusion, we had made our arduino board play music. Although this was a challenging program we managed to figure out the coding with the help of the lab6notes given by our TA. We got a better understanding in how piezoelectric speakers are used. They work by applying voltage that causes a piezoelectric film to deform. That distortion causes an audio sound by displacing air like normal speakers.