ECE 2049: MSP430 Hero Lab Report

Due on Monday, February 23rd

Ted Meyer

Introduction

The goal of this assignment were to gain experience using inturrupts and understand the differences between them and simple polling techniques, as well as gaining more experience with a miriad of different peripheral devices, including a small amount of timer work to play music on the buzzer.

Discussion and Results

The main goal of this lab was to understand the importance of inturrupts. Microprocessors generally only have one frame of execution, which is where the previous labs were constrained to. Because of this, doing multiple things at once could lead to backups and race conditions. There were examples of this in the space invaders game; because everything was draw to the screen inside this main operation, button presses could only be read while the screen was no updating. Inturrupts are designed as a solution to this, and implemented at the processor level. They take electrical signals from external devices and use them to halt code execution (they cache the current register values including the instruction pointer) and jump into another routine specified by the inturrupt table. For this lab, both inturrupts and polling were used; because each note had to play for a spicific set amount of time, processor clock speed hiccups would distort the music, and so note playing was triggered by an inturrupt. The capacitive touch pad buttons however do not have the ability to fire an inturrupt, so they were polled instead. Another major topic was how to play notes of a specific frequency. This was in fact quite simple; since the ACLK timer used for the buzzer operates at a frequency of 32768 Hz, in order to play a note at 400 Hz, we simply divide 32768 by 400 to get the number of cycles it should take to oscilate the buzzer. The source used is described in Code Appendix A. For the duration, a simple method of storing the duration as part of the note struct, which each inturrupt would slowly count up to, was used. When the counter and the duration were equal, the pointer to the note struct would be incremented and the counter reset. For game logic, I used a pretty simple control flow. The song I chose was Turkish March by Motzart, and I had programmed 44 notes. To win, a user must hit 30 or more correctly. This was very hard, because it was a fast song. See Code Appendix B for the timer set up function.

Summary and Conclusion

A few things were accomplished in this lab, only some of which were intended. Most importantly of course, inturrupts were used to create a correctly timed series of notes. Though this probably could have been accomplished with precision instruction manipulation at the assembly/bytecode level, doing it with inturrupts is a far more robust and understandable way. It was also an edifying experience in terms of musical knowledge: The fact that two notes one octave apart have half/double frequency was new to me, and this was also the first time in years that I've had to read sheet music.

Code Appendix A

```
void play_note(int freq) {
    P7SEL    |= BIT5;
    P7DIR    |= BIT5;
    TB0CTL    = (TBSSEL__ACLK|ID__1|MC__UP);
    TB0CTL   &= ~TBIE;
    TB0CCR0   = freq?32768/freq:0; // play note of this frequency
    TB0CCTL0 &= ~CCIE;
    TB0CCTL3  = OUTMOD_7;
    TB0CCTL3 &= ~CCIE;
    TB0CCR3   = TB0CCR0/2;
}
```

Code Appendix B

```
void run_timer() {
    TA2CTL = TASSEL_1 + CNTL_0 + ID_1 + MC_1;
    TA2CCR0 = 16;
    // this fires an interrupt once every 1/2048 of a second (0.000488 seconds)
    TA2CCTL0 = CCIE;
}
```