**Microprocessor Systems Lab 2**

Interrupt and Timer ISRs

Shane Conaboy

Justin Yost

ECSE-4790 Microprocessor Systems

September 28, 2015

**Introduction**

This lab investigated how to generate both external and internal interrupts on the 8051 microcontroller. For the external interrupt, a push button was wired to an input port and an interrupt would be thrown when it was pushed. Timer overflows were used to generate the internal interrupts. By changing the system clock settings to obtain different speed in conjunction with modifying the timer settings, a stopwatch-like program that was accurate to one-tenth of a second was created using internal interrupts. At the end, both parts of the lab were integrated to make a game based off of human reaction time.

**Procedure**

The first part of the lab involved wiring a pushbutton to trigger an external interrupt on the microcontroller. The pushbutton was wired to P0.2 since the UART was enabled, taking up pins P0.0 and P0.1. When the button was pressed, the program would set a global variable, *interrupt\_detected*, to 1. The I/O was handled in the *main* function, which would output to the terminal that the interrupt was detected. To handle debouncing, a capacitor was added in parallel to the output.

The next part of the lab was to modify the speed of the system clock and use it to generate timer overflows that would track time accurately to one-tenth of a second. The full calculations for this section are shown in Appendix A. The first attempt to generate the stopwatch was using Timer0 in 13-bit mode with a system clock of 4.1472 Mhz. To generate 0.1 seconds, Timer0 would need to overflow 50.625 times. As an approximation, 51 overflows was chosen, which gave an error of 0.375 overflows per overflow, or 0.375 seconds per second.

The second, more accurate stopwatch was created with Timer1 in 16-bit mode, a clock rate of 4.1472 MHz, and preloading Timer1 each overflow so it would not count its entire range of values. The preload value was 64512, which was set every interrupt, and 405 overflows at this preload would give exactly 0.1 seconds. The calculations can be found in Appendix A.

The last part of the lab was to create a game using the previous two parts. The game would tell the user to press the pushbutton when “GO!” is displayed after a random delay, and it would measure the reaction time of the player using the stopwatch program in part 2 to keep track of time and the external interrupt in program in part 1. The last five of the player’s attempts were displayed on the screen, along with the average.

**Results**

Part 1 of the lab was implemented with little problems. After the debounce circuit was added, the screen would display “Interrupt Detected” only once upon each press of the button. Part 2 of the lab was much trickier because of the clock settings.

Obtaining the inaccurate timer in Part 2 was straightforward since everything was already set up in the given code. To generate the accurate timer, first selecting the external crystal as the system clock was tried, however this caused problems with the baud rate. Next, changing the multiply and divide values of the PLL was attempted. Even with changing the expected frequency filter to the correct range, the PLL would never initialize properly and the program would hang. At this point, time was running short, so the method of preloading the timer was chosen. After performing the calculations, the values were set and the timer was able to achieve a 0.1 second accuracy.

Developing the game was a lot like lab 1 and mostly required text positioning. After computing the random delay, the timers would count up to that point before resetting, after which they would begin counting the reaction time of the player. The times were shifted through an array, and the average of the five stored values was computed every trial. As an add-on the lab, a penalty was implemented that would automatically give a 5 seconds to the current trial if the player pressed the button before the screen displayed “GO!” No major problems were encountered during the implementation of the game.

**Conclusion**

By creating a timing-based game using both external and internal interrupts, different interrupt, timer, and clock settings of the 8051 microcontroller were explored. The most time was spent on trying to get the clock settings correct so the program would run properly, and this part was never implemented in the way it was originally intended. Future suggestions would be to explore this area of the 8051 more to gain a better understanding. Otherwise, the lab was successfully completed.

**Appendix A: Timing Calculations**

Inaccurate clock:

Accurate Clock:

Timer1 in 16-bit mode 🡪 GCF of 4.1472E5 and 65536 is 1024

**Appendix B: Circuit Schematic**

**Appendix C: Code**