Jonathan Lam
Prof. Kirtman
ECE 251
Computer Architecture
4 / 25 / 19

Homework 2: Pic Programming

Implementation of given demo program

The connections necessary for the implementation of the given program were a high signal to master clear (MCLR), power and ground to VDD and VSS, a 4MHz oscillator and two capacitors on the oscillator inputs, and eight output LEDs from the PORTC (RC0..7) pins. Figures 1 and 2 are images of the (full) implementation.

The timing of the program was verified using an oscilloscope. The 4MHz clock was running at 4.0MHz as expected, and the frequency of the LSB was 1.992Hz (i.e., one HIGH-LOW period every ~0.5s, or a count every ~0.25s). This was verified with the debugging stopwatch tool in the MPLAB IDE, which counted 251006 instruction cycles (1μs per instruction cycle) per count. (Specifically, L3 takes 10μs per iteration, L2 takes 100*L3+4=1004μs per iteration, and L3 takes 250*L2+6=251006μs per iteration, leading to a 1000000/251006=3.984Hz counting signal, or 1.992Hz "signal" for the LSB.) No attempt was made to fix the slight offset.

Implementation of two control switches

First, the lowermost two TRISB bits were set high to make RB0 and RB1 inputs. (PORTB was chosen because had a similarly easy setup as PORTC.) RB0 was used as the start/stop input, and RB1 was used as the count by two input.

To implement start/stop behavior, the final two nop commands in L3 were replaced with a check for the start/stop input bit. If the start/stop bit (RB0) was not set, execution would continue and the code would goto L1, restarting the iteration count. (As long as RB0 stayed low, execution would always hit this goto and restart the iteration count before COUNT would update.) If the start/stop bit was set, then the goto statement would be skipped. Because the btfss command used to iterate this conditional behavior takes two instruction cycles on a skip and two nops were removed, timing was preserved.

To implement count by two behavior, btfsc was used to increment COUNT a second time if RB1 was set. If RB1 was not set, then the second increment would be skipped. This adds an additional (trivial) 2μs to each count timer.

Refactoring delay into subroutine

The L2 and L3 inner loops, which were used simply to iterate through a certain number of nop commands, were moved into a subroutine called DELAY. Every iteration of L2 runs for approximately 1ms (1004μs), so the number of iterations of L2, which is read from the W

register on the start of the DELAY subroutine, is roughly the number of milliseconds of delay. To recreate the same 0.25s delay, I manually set .250 into the W register before calling the DELAY subroutine.

Figure 1: Annotated photo of implementation

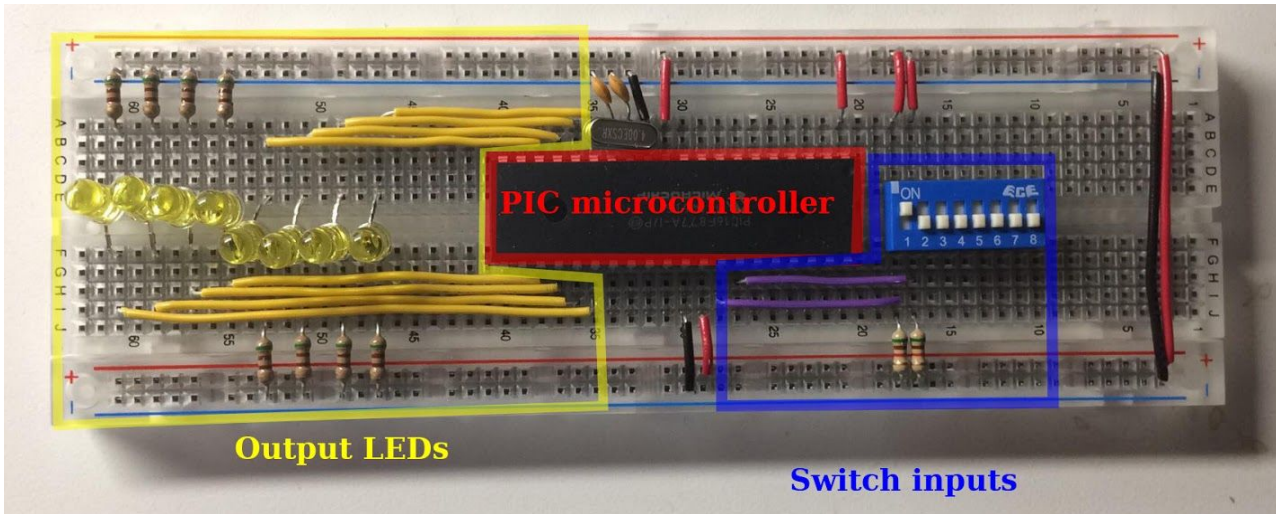

Figure 2: Schematic of implementation