LAB 1: Programming and basic wiring of the PIC24 processor family.

This lab consists of the following tasks:

- Familiarization with the operation of the programmer and the MPLABX IDE.
- Programming the PIC processor.
- Wiring the clock, power and ground, and I/O pins.
- Observing and timing the behavior of the resulting circuit.

We will cover some of the necessary parts in discussion, but, since we won't finish discussion before the lab, you may need to do some things on your own.

You will need to know how to create a project, how to build the project, and how to simulate the project. These operations can all be done on your own computer before arrival in the lab. Only programming (flashing) the processor will have to be done in the lab.

Before you come to lab you should build the project, and, using the MPLABX simulator, you should have spent some time simulating the operation of the program so that you understand how it behaves.

You will program PIC processors in this lab using one of the programming stations. You will need to build (that is, **assemble**) the first program, "lab1a.s" and have the program ready for programming your microcontroller. Follow the lab instructions and place your microcontroller in the programmer (note: if you place it in the programmer 180 degrees off, your chip will likely be destroyed, so a measure of care is required here.)

Program the PIC processor (flash the internal flash program memory.) Remove the processor from the programmer.

The test circuit for this lab consists of hooking up the power and ground, the two capacitors, plus a resistor to tie the reset pin high (inactive), as well as the LED and current-limiting resistor. The schematic shows an additional connector (SV1), but that is optional.

Insert the processor in your test circuit. Install the rest of your circuit. When it has been installed and checked carefully (mis-application of power and ground here can damage the processor.) Triple-check, and get your lab partner to examine what you've done carefully to make sure the circuit is set up. Apply power (3.3V) and observe its behavior, carefully timing the flashing of the LED.

To time the period of the LED (actually, half the period, as the code takes the same amount of time on as it does off, and that is one pass through the code) you can measure either frequency (count blinks for a fixed period of time, say a minute or so) or period using a stopwatch.

You'll need to measure the frequency to within one-half of one percent. If you are measuring a frequency of 16MHz, for example, your measurement of frequency must be accurate to within 80KHz. To get this accuracy measuring

the frequency of the blinks you'll need to count the number of blinks over 200 seconds (so you know the number of blinks in one second accurate to plus or minus 1).

To obtain the same accuracy using the period, you need to measure the period accurately to one-half of one percent. (Since the period is close to 1 second, we have the relationship between frequency and period errors – from calculus – viz: since $f = \frac{1}{P}$, $\Delta f \approx -\frac{1}{P^2}\Delta P$, where P is period and f is frequency. Put in $P = 1$ second.)

Demonstrate the operation of your circuit for the TA and turn in answers to the following questions:

1. What is the rate at which the lights blink? RB2 blinks fastest, RB3 blinks at half the rate (twice the period), RB4 at one-fourth the rate, and RB9 blinks at 1/128th the rate of RB2. You may need to use RB3 if RB2 flashes too quickly for an accurate count. Remember, one period of the LED is twice the time required to get through the loop.

2. Look in the lab1a.s program and locate the delay loop. This is the part of the program that separates (in time) the occurrences when the "mov w2,PORTB" instruction is executed. Count the number of machine instructions the delay loop comprises (execution-wise). From this, compute the machine instruction cycle frequency by simply dividing the number of instructions by the time you measured for those instructions. Note that the clock timings (via the "stopwatch") using the debugger to run lab1a.s are given in "instruction cycles", (a mythical concept, as many instructions take different times to execute) which are exactly 2 oscillator clock periods in length. The oscillator runs at a nominal frequency of 32MHz. It won't be exact.

3. Based on your answers to the previous questions, and the fact that the instruction cycle frequency is supposedly 16 MHz, how accurate is the clock generator compared with the theoretical 16 MHz?)

4. If you want, you can search for the string "FRCPLL" and change it to "FRC". What happens to the clock rate if you do?

The .1 microfarad capacitors between power (Vdd) and ground near the power pins of the processor are "decoupling" capacitors. They protect the power from being affected by transients due to the processor switching. **Always put decoupling capacitors in your circuit, even if not shown!** They may be marked "104". The other capacitor is the one that is part of the internal voltage regulator which provides the power for the processor core (which runs at a lower frequency than the external parts of the chip.)