CPE 490 Lab 2

## Goals

1. Learn how to use while loops to ensure that main function call is not exited
2. Learn how to use logic and relational statements to control while loops and make simple time delays
3. To understand how to setup the timer peripheral for the dsPIC33
4. How to use the stopwatch and the logic analyzer in MPLAB SIM

## Overview

This lab will allow you to design, write and verify c code.

## Report

Standard report write-up is required.

## Design

Using what you learned from last week and in lecture, design a C program for the dsPIC33FJ256GP710A that will:

1. Never exit the main function call
2. Initialize all SFRs needed to accomplish the function described next
3. Using the lower 8 bits of PORTA as digital outputs, turn the lower 8 pins on for .5 seconds and then off for .5 seconds repeating this pattern indefinitely.

Remember to use header file that was included last week. For help on setting up the required SFR, use what you learned in last week’s lab as well as what you learned in Thursday’s lecture.

## Verification

We will use the MPLAB X SIM simulator like we did last week. The simulator has time based debugging information that can be very useful. In order to get useful information we must tell the simulator what the processor clock speed is. We will set up the simulator to be similar to the Explorer 16 demo board that we will use this semester. To do so:

1. From the menu select -> Run-> Set Project Configuration->Customize
2. In the Categories: box click on Simulator
3. In the Instruction Frequency (Fcyc) box type in 16 while ensuring that the Frequency in entry is MHz. Click Apply and then OK

### Stop Watch

The MPLAB X simulator has a stop watch function that can allow you to time the number of seconds or instruction cycles between break points. A break point can be set by simply clicking the line number of code in the source file.

To view the stopwatch click on the menu choice Window->Debugger->StopWatch. You can reset you code and set two break points. Run you code to the break point and then inspect the Stopwatch window. It will tell you the time it has taken to run code from the reset vector to the breakpoint. You can reset the stopwatch every time you start running by pressing a button for that purpose. **Be patient** the simulation for .5 seconds takes many seconds. Using this function verify that your code has the lower 8 bits of port ‘A’ on for about ½ a second and the off for ½ second and that it repeats.

**Demonstrate to an instructor you code running (in the simulator) and the use of the stop watch to verify that the port outputs are on a period of .5 seconds and off for .5 seconds.**

**Instructor Signature \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

### Logic Analyzer

MPLAB X simulator also has a logic analyzer function.

1. On the menu click Window->Simulator-> Analyzer

You are now set up to use the logic analyzer. There is a problem. When the trace buffer is enabled the simulator really slows down, so it is hard to see a trace that has a period of 1second. In order to see the usefulness of this tool set you delay function to delay 256 instruction cycles instead of the ½ second used before. Build all and run the code for a while, then press the halt button. A square wave signal should be on your logic analyzer when you click on it. The X axis is in instruction cycles.  **Answer the question: What is this period measured and why is it not take exactly 256 instructions?**

**Demonstrate to an instructor your code running (in the simulator) and the use of the logic analyzer, be prepared to state the time that 256 instructions cycles should take to execute.**

16 us. It does not take exactly 256 instructions because a jump instruction in assembly will take more than 1 instruction cycle.

**Instructors Signature \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

### Trace buffer

While not needed for this lab it is possible to see a buffer of previously executed commands prior to a halt command or a breakpoint in a simulation. You must enable the trace function by menu selecting Run->Set Project Configuration->Customize. The project properties window will come up. In the Categories window click on the simulator. In the right hand side of the window select Trace from the option categories. Select the unlock button if needed and then turn the Data Collection Selection to Instruction Trace, then select the apply button and then the OK button. At this point you should have a trace window.