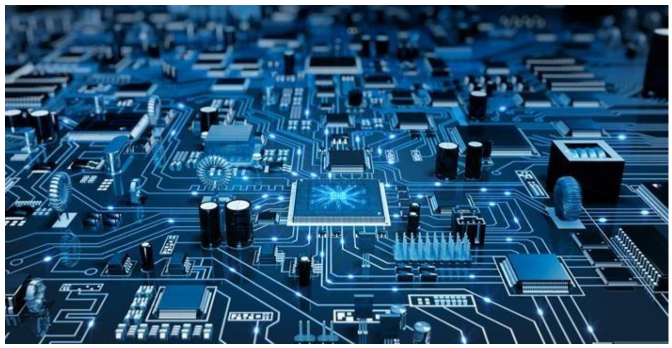
CARNEGIE MELLON UNIVERSITY AFRICA



EMBEDDED SYSTEMS DEVELOPMENT

04-633 A

Learning about Electronics

Due: Monday, January 29, 2024

1 Introduction

1.1 Simulation Software - Embedded Systems Development

The simulation software will allow you to write, debug, and optimize PIC micro MCU applications for firmware product designs.
MPLAB IDE supports an emulator, debugger, and developer mode. The MPLAB IDE allows you to create and edit source code by providing you with a full-featured text editor. Further, you can easily debug source code with the aid of a Build Results window that displays the errors found by the compiler, assembler, and linker when generating executable files.

Proteus 8 Professional Circuit Simulator is a suite widely used to design, test, and debug complete embedded systems inside schematic capture before deploying a physical prototype on a PCB. The circuit simulation is based on the models of electronic components for modeling programmable devices to work with microcontrollers, microprocessors, DSP, and others. The simulation software is officially distributed by LabCenter Electronics Ltd. [1]

1.2 Lab Objectives

- To install and test the simulation software used for PIC microchip micro-controllers and other state-of-the-art circuit design and simulation software such as Proteus 8 Professional.

- To understand the MCU functions on memories, parallel ports, power supply for low-power computing, and clock oscillators using Proteus 8 software and MPLAB X5.50 IDE.

- To understand how to work with the PIC MCUs for Embedded Systems applications.

2 Software Installation Procedure

1. Installation of MPLAB IDE and Proteus Simulation Software

(a) Install MPLAB X5.50 IDE from the official microchip website. [link]

(b) Install the XC-8 compiler version 2.35 [link].

(c) Install Proteus 8 Professional circuit simulator [link].

Kindly ask the Teaching Assistant for assistance with software installation before proceeding to the next steps.

2. Creating a project on MPLAB IDE

(a) Run the downloaded MPLAB IDE X5.50, install the USB options for the microchip, and finish the installation.

(b) Once installed, click on the MPLAB icon on the desktop or search for the software on the startup menu.

(c) Click the **File** option in the menu bar at the top-left portion of the screen and choose to create a **new project**.

(d) Choose the **Embedded Standalone Project**, then **Next**.

(e) Choose the family of the MCU which is an 8-bit mid-range PIC. On the device option, choose the **PIC16F877A**. Then **Next**.

(f) Choose the debugging hardware tool for the project. In the physical lab, you will be provided with PICKit3 devices that will be mounted on your computer via a COMPORT. In this section, choose **PICKit3** and then **Next**.

(g) Select the XC8 (v2.35) compiler for the project. At this point, the installed XC-8 (v2.35) will appear, click on it. Then, **Next**.

(h) Name the project as preferred. Now, save your project in your desired location on the local PC. Choose to set it as the **main project** for the compiler to recognize it and click **Finish**.

3. Creating a file

(a) To create a header file that will contain the configuration files for the PIC 16F877A, On the created project, right-click on source files, choose new, then click *xc8_header.h*. Check *Appendix* to find the contents of the header file. (*To understand the parameters on the configuration file, read the PIC datasheet).*

(b) To create a main file, right-click on the source files in the main project. Choose **New** and select the *main.c* file option.

(c) Write the codes for your project.

3 Checkpoint 1: Blink LEDs

1. On Proteus, identify the following components for the "Blinking LED" exercise.

   - PIC 16F877A
   - At least two LEDs (green, red, or blue)
   - Power and ground sources
   - Oscillator
   - Resistors (470 OHM)
   - Capacitors (22pF)

2. After you have identified all the needed components, connect them according to Figure 1 below.

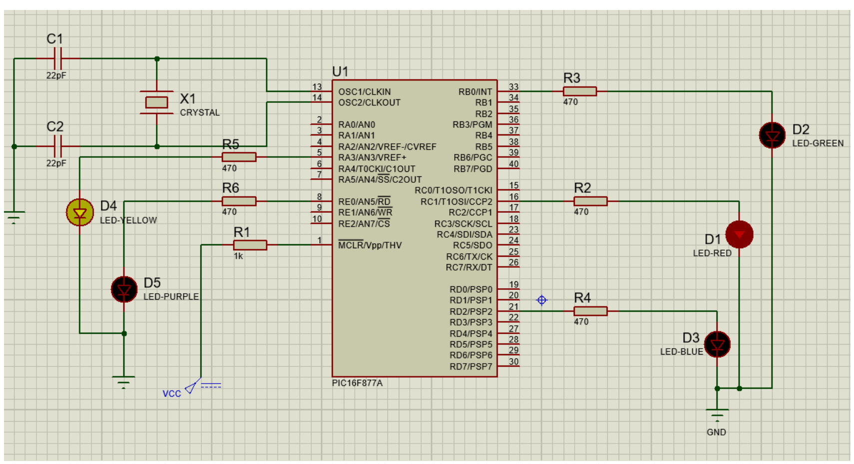


Figure 1: Led Blink Schematic

3. Create a project in MPLAB and write the code to blink the LEDs simultaneously.

4. Compile the working project in MPLAB, copy the '.hex' file into proteus,

and run the simulation.

5. Connect the physical hardware and upload the code.

4 Checkpoint 2: Digital I/O

1. On Proteus, identify the following components for the “Digital I/O” exercise.

   - PIC 16F877A
   - At least two LEDs (green, red, or blue)
   - Power and ground sources
   - Oscillator
   - Push button
   - Resistors (470 OHM)
   - Capacitors (22pF)

2. After you have identified all the needed components, connect them according to Figure 1 below.

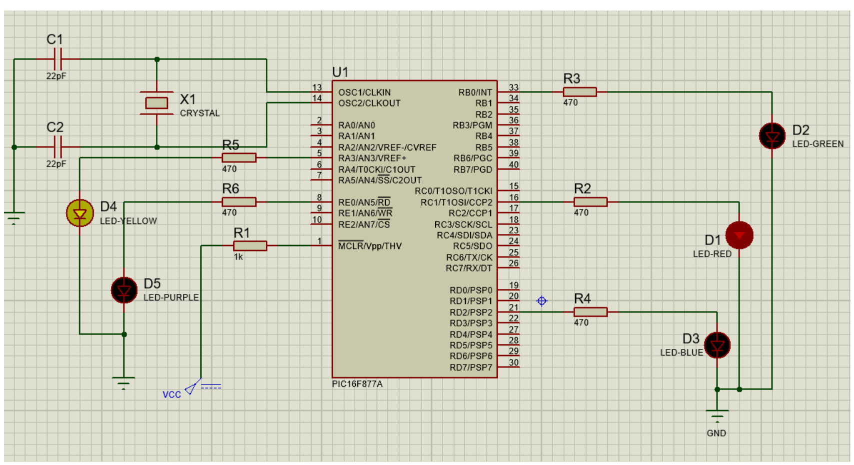


Figure 2: Led Blink Schematic

3. Create a project in MPLAB and write the code to put on the LED when the button is pressed and put off the LED when the button is pressed.

4. Compile the working project in MPLAB, copy the ‘.hex’ file into proteus, and run the simulation.

5. Connect the physical hardware and upload the code.

5 Submission

1. Submit to GitHub and Canvas
2. Fill in a short description of the project and the repository in a README file at the base of your repository.
3. To submit, create an issue on GitHub titled Lab1-Submission. In the comments section include your name, AndrewID, and the commit hash you want to submit. Also, be sure to submit the link to your repository to Canvas.

6 Submission Requirements

- Proteus design (15 points)
- Source codes (15 points)
- Code style (10 points)
- Demo (10 points)

7 Appendix

CONFIGURATION FILES

```
#pragma config FOSC = HS // Oscillator Selection bits (HS oscillator)
#pragma config WDTE = OFF// Watchdog Timer Enable bit (WDT disabled)
#pragma config PWRTE = OFF // Power−up Timer Enable bit (PWRT disabled)
#pragma config BOREN = ON // Brown−out Reset Enable bit (BOR enabled)
#pragma config LVP = OFF // Low−Voltage (Single −Supply) In−Circuit Serial Programming Enable bit (RB3 is digital I/O, HV on MCLR must be used for programming)
#pragma config CPD = OFF // Data EEPROM Memory Code Protection bit (Data EEPROM code protection off)
#pragma config WRT = OFF // Flash Program Memory Write Enable bits (Write protection off; all program memory may be written to by EECON control)
```

```
#pragma config CP = OFF // Flash Program Memory Code
 Protection bit (Code protection off)
```

References

[1] M. Farooq, “Full working portable circuit simulation application proteus 8 professional,” Nov 2014. [Online]. Available: https://www.itechsoul.com/full-working-portable-circuit-simulation-application-proteus-8-professional