ECE 2534 – Fall 2015 – Lab 1
Introduction to the Digilent microcontroller board and the MPLAB tools

This lab is an individual assignment.

Introduction
This laboratory exercise will guide you through the process of installing the MPLAB software package and running a simple program on your Digilent microcontroller board. You will need to write a small amount of C language code. The latest boards have the model name **ChipKIT Pro MX7**, but these boards are functionally equivalent to the **Cerebot MX7 cK** boards that we used previously in this course.

Honor Code Requirements
The lab procedure indicates that you may ask other students for assistance up to a certain point in performing the assignment. You must complete all other aspects of the assignment *individually*. Do not discuss any of these aspect of your solution or approach with anyone except for your instructor or a GTA. (You are encouraged to seek help, if needed, from any GTA in the Computer Engineering Lab.) Consider all information that you generate to be proprietary. Copying or using any element of any other person's implementation or solution is a violation of the Virginia Tech Honor Code.

Step 1: Hardware components
The ECE department will lend you a lab kit containing a microcontroller board and other parts. Some of the parts are called peripheral modules, or "Pmods," and they attach to the Digilent board. All of these parts must be returned before the end of the semester. More details are provided at the CEL's web site: https://sites.google.com/a/vt.edu/computer-engineering-labs.

For this assignment, you will only need the following items:
- Digilent board (ChipKIT Pro MX7 or Cerebot MX7 cK)
- PmodOLED (an Organic LED graphic display)
- USB cable with micro-USB and USB-A connectors
- Your laptop computer

Step 2: Safe handling
Read the document "Safe Handling of Printed Circuit Boards," which is available at the CEL web site under Supplementary Documents. Electrical and computer engineers should set an example for avoiding ESD damage, which is more likely in the winter. (Not familiar with "ESD"? Now is a good time to look it up!)

Step 3: Software tools
You will develop code for your microcontroller board using Microchip's MPLAB X Integrated Development Environment (IDE), together with Microchip's C compiler (XC32). These tools are available at no cost, and you need to install them on your laptop. We ask you to download slightly older versions of the tools, which are better for a course such as this than the latest versions. To obtain these tools, go to http://www.microchip.com/mplabide, follow the links "MPLAB X IDE" → "Downloads Archive" and download the following:

- MPLAB X IDE v2.00
  This will give you a large ZIP file that contains a *installer.exe file. Run it to install the IDE on your machine. At the end, you can uncheck the box that will cause a download of the XC compiler.

Near the top of http://www.microchip.com/mplabide, follow the links "MPLAB XC Compilers" → "Downloads Archive" and download the following:

- MPLAB XC32 v1.31
  This gives you a *installer.exe file for the compiler. Run it to do the installation. When prompted, select the options that will cause the XC32 compiler to be installed on your laptop. Don't select anything related to a network license server. If you receive a request for a license activation key, leave the key field blank to use the compiler in "free" mode. It is okay to keep the default settings when prompted about your PATH environment variable. When the "Installation Complete – Licensing Information" box appears, click on Next.

Step 4: Documentation

Get the reference manual and schematics for your Digilent board:

- http://digilentinc.com → chipKIT boards → chipKIT Pro MX7
  Under "Support Documents" you can download the manual and schematics.

Whenever you start using a new Pmod, you should find the documentation for it. For this assignment, follow the link below and find the reference manual and schematics for the PmodOLED (not PmodLED or PmodOLED2):

- http://www.digilentinc.com/Products/Catalog.cfm?NavPath=2,401&Cat=9

Step 5: Connect the display to the board

Connect the PmodOLED to the connector labeled "JD" on the Digilent board, which is located near the RESET button. The smooth display side of the OLED module should face upward, away from the table when the Digilent board is lying flat. Check that jumper JPD, next to connector JD, is set to position 3v3 (instead of 5v0).

Step 6: Connect the board to your laptop

Read sections 1 through 5 of the Digilent board's reference manual. Identify each of the jumpers and connectors mentioned therein. The Power Select jumper block above the Ethernet connector should be set to the DBG ("debug") position, so that the board will obtain power from your laptop through a USB connection.

Turn power switch SW1 to the OFF (lower) position, and then gently connect the USB cable to the micro-USB connector on the Digilent board that is next to the large Ethernet connector. (On my board, the label "DEBUG" is printed next to this micro-USB connector.) This is a tight fit, and the small connecter is only glued to the board, so be careful when attaching or removing the cable. Next, connect the other end of the cable to a USB port on your laptop. Turn SW1 to the ON position. Your Windows platform should automatically install device drivers when the board is powered up the first time.

Step 7: Compile and run a simple program on your Digilent board

a) With your board connected to your laptop and powered on, start MPLAB X IDE.
b) On the Start page, choose the "Learn & Discover" tab. Under the "Getting Started" header, choose the "Quick Start" option. (If you see any information labeled "important" for MPLAB IDE V8 users, you can safely ignore it. We are using MPLAB X IDE, which is a different tool altogether.)
c) Look over the QuickStart guide, and follow the instructions there to create a new Standalone Project. At the "Select Device" step, choose `32-bit MCUs (PIC32)` as the family, and choose `PIC32MX795F512L` as the device. At the "Select Tool" step, choose "CerebotMX7cK" under "Licensed Debugger." A serial number that corresponds to your board should appear after the board name. At the "Select Compiler" step, choose "XC32 (v1.31)."

   You may see a message similar to "Licensed Debugger Not Found" if your board is not connected or powered. It may also happen because of driver hiccups in Windows. In such cases, the usual fix is to power-cycle the board.

d) At the step called "Select Project Name and Folder", type "`OLEDtest`" for our first Project Name. As your Project Location, specify a folder where you can easily keep track of code for the different software "projects" that you will create for this course. For example, you might type `C:\MPLABXProjects`. As you make changes in the Project Name and Project Location boxes, notice that MPLAB X automatically updates the Project Folder box. For the names given here, `C:\MPLABXProjects\OLEDtest.X` should appear in the Project Folder box. As the course progresses, it is best for you to create a new MPLAB project for each new assignment.
e) Place a check in the box "Set as main project" box, and click "Finish".
f) The system should now create the folders described in the previous 2 steps. Into your `OLEDtest.X` folder, copy the source files that were provided to you as part of this assignment (from the "Step7" zip file). The names of the files are ChrFont0.c, delay.c, delay.h, FillPat.c, main.c, OledChar.c, OledChar.h, OledGrph.c, OledGrph.h, PmodOLED.c, and PmodOLED.h. (All of these files except main.c were supplied by Digilent.)
g) In the "Projects" window of MPLAB X, expand the `OLEDtest` project. Right-click on the `Header Files` folder and select `Add Existing Item` and then select your file delay.h. Repeat for all other *.h files listed in the previous step. You can Control-Left Click file names to select multiple files.
h) Repeat the previous step to populate the `Source Files` folder with all of the *.c files from the "Step7" archive.
i) Turn on *Additional warnings* under File→Project Properties→xc32-gcc→Preprocessing and Messages.
j) It is time for the moment of truth. Verify that your Digilent board is still on, and connected to the laptop. Click on `Debug` → `Debug Main Project`. Things should begin to happen. MPLAB X should "build" your "project", which means that it will compile and link the source files specified above, storing the result into a *.elf file on your laptop. The message "BUILD SUCCESSFUL" should appear. The system should start up its "Licensed Debugger" and then "program" the "target", which means that machine-language code is transferred from your newly created *.elf file into the memory of the PIC32 microcontroller on the Digilent board.

The message "Loading completed" should appear. Eventually, the OLEDtest program should begin to run on the board; you should see the messages "ECE 2534" and "Fall 2015" on the OLED display, along with a fast-changing count value.

k) Find a pushbutton switch that is labeled BTN1 on your Digilent board. Whenever you press the button, an LED labeled LD1 should turn on. Releasing the switch should cause the LED to go off right away.

l) *If everything works* as described in the previous steps, then you are ready to proceed with the rest of this assignment. If you don't see the OLED messages, or if LED1 doesn't turn on when you press BTN1, then go through the procedure again carefully. *Seek help if you can't get things to work. Up to this point, it is acceptable to ask other students for assistance.*

m) Notice that you can see and edit the source files by double-clicking on *.h and *.c files at the left. Try making a simple change in main.c, such as changing "Fall 2015" to "Micro is Fun". Recompile the program with your changes, and run it on the Digilent board. You should see the different message on the OLED display.

n) You have the option of compiling all of your project code from scratch by selecting Run → Clean and Build Main Project.

o) When finished, click on `Debug` → `Finish Debugger Session`. Notice that the OLED is no longer being updated. You can then exit MPLAB X, and turn off your Digilent board.

Step 8: Learn about LEDs and switches on the Digilent board

The PIC32 microcontroller supports several general-purpose input/output (GPIO) ports, known as PORTA through PORTG. Through these ports you can access such things as LEDs and pushbutton switches on the Digilent board.

As you may have noticed by now, the Digilent board contains four LEDs that are labeled LD1 to LD4 on the board, and three pushbutton switches that are labeled BTN1 to BTN3. You can write software to read the states of the switches, and to turn the LEDs on and off. Before you can write the software, however, you need to find out which particular GPIO ports and pins are wired to those external components. You also need to learn the procedure for configuring the GPIO registers. Many of these details are given in section 4 of the microcontroller board's reference manual. Refer to that manual now, and complete the following table. (This is for your reference only. You do not need to hand in this table.)

Device	***PIC32 GPIO port***	***Port bit***
BTN1	PORTG	6
BTN2		
BTN3		
LED1		
LED2		
LED3		
LED4		

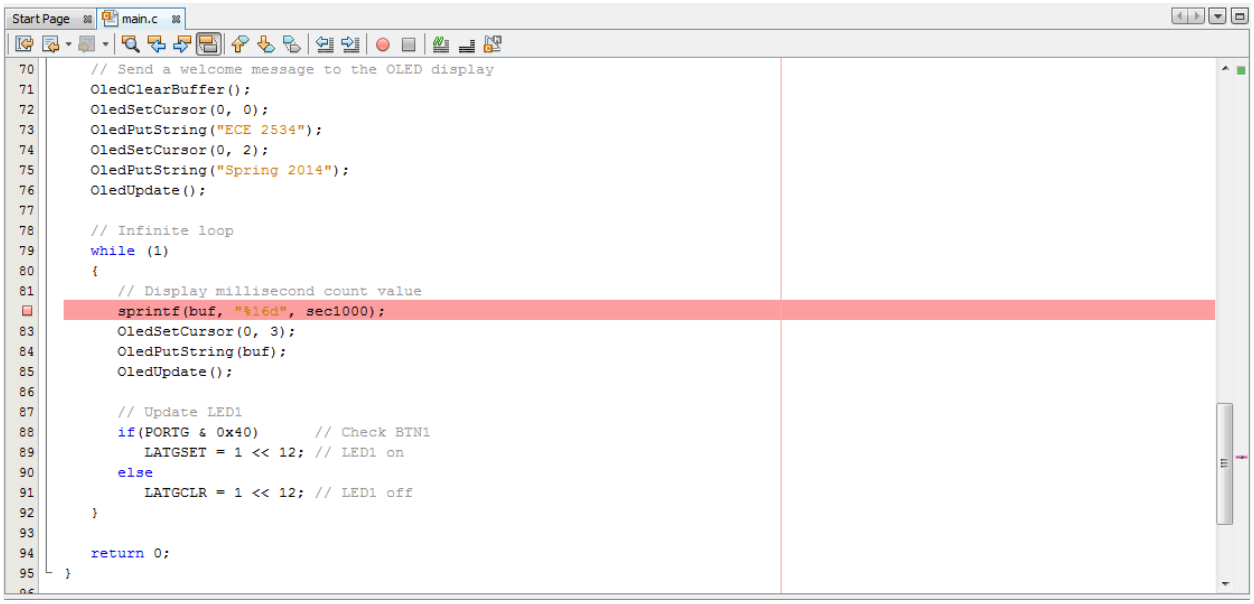
Do not confuse PIC32 GPIO ports A through G with the Digilent board Pmod connectors JA thru JF. They are unrelated. Similarly, the 32 bit positions within an internal port are unrelated to PIC32 package pins or Pmod connector pins. Appendix C in the board's Reference Manual clarifies the mapping from PIC32 IC package pins to the board's connector pins, and to PIC32 port bits.

Section 12 of the PIC32 Reference Manual provides many more details of GPIO port operation. Notice that PIC32 pins connected to pushbuttons must be configured (by your software) as digital inputs, and pins connected to LEDs must be configured as digital outputs. Examples of input/output configuration were given to you in the previous step. Examine main.c, which demonstrates how to configure and use a GPIO port for interfacing with one of the board's pushbuttons and one of the LEDs.

Step 9: Learn some debugging tricks

MPLAB provides many aids to help you debug your code. After your code compiles (an important first step!), you can use MPLAB to set breakpoints in your code, single-step through your program as it runs on the PIC32, observe the values of variables, and other things. All of this is possible because MPLAB X IDE interfaces through USB with a special in-circuit debugger IC on the Digilent board. It is this IC that controls the operation of the PIC32, causing program execution to start and stop based on commands that you send from MPLAB.

Try out the debugging capability with the OLEDtest program. A brief introduction is given here, and more details can be found in section 4.17 of the MPLAB X IDE User's Guide. When viewing main.c, for example, you can set a breakpoint by clicking on the line number of the source code as shown below:



The example above places a breakpoint at line 82 of main.c. The next time you run the program, such as by selecting `Debug` → `Debug Main Project` or by some other means, the program will temporarily halt just before the PIC32 executes the code at any breakpoint. From there you can continue or single-step through the code by clicking one of the following icons near the top of

the MPLAB interface:

You can move the cursor and hover over these icons to find out what they do.

Whenever the processor is halted in debug mode, you can also hover over a variable, and MPLAB will try to report the current value for that variable. Try this for sec1000, buf, and PORTG, while experimenting with the icons shown above.

You can open a "watch window" by selecting `Window→Debugging→Watches`, and you can add a variable to the watch list by right-clicking the variable and selecting "New watch". Whenever the PIC32 is halted, MPLAB will try to report the current values of all variables in the watch list. Try this capability for yourself by adding a new variable to the program, maybe an integer named `i`, that is incremented on each pass through the `while(1)` loop. The ability to halt the CPU occasionally and observe variable values will be extremely useful when you begin to debug your own code.

After you have set a breakpoint, right-click the breakpoint's orange box and then select `Breakpoint→Properties` and then under `Pass count` enter a count value that is greater than 0. This is the number of times that the PIC32 will execute the breakpoint line before halting. This can be extremely helpful in debugging, because you can tell the processor to execute a particular line of code 10 or 1000 times before halting, for example.

Finally, one way to remove a breakpoint is to left-click on the breakpoint's orange box at the left. MPLAB X IDE provides many other debugging capabilities, and you can read about them in the reference manual.

Step 10: Write some code

Follow the procedures that you learned earlier in this assignment to create a new MPLAB X project called Lab1. Into your new project folder, copy the same source files that you used earlier in this assignment. Verify that you can compile and run the program from within your new MPLAB X project.

Modify the given program as follows. The only file that you need to change is main.c:

a) In place of the text "Fall 2015", display your name on the OLED. You are limited to 16 characters, so use appropriate abbreviations to observe this limit, if needed.
b) The original version of the program causes LED1 to be on whenever BTN1 is pressed, and to be off when BTN1 is released. Change the program so that all 4 of the LEDs are on when exactly 1 of the switches BTN1 or BTN2 is pressed. All 4 LEDs should be off whenever BTN1 and BTN2 are both released, and all 4 LEDs should be off whenever BTN1 and BTN2 are both pressed. (For this assignment, you do not need to write code to perform *switch debouncing*. That topic will be introduced separately.)
c) Add comments to main.c to describe your additions. Also change the comment header at the top of the file to give your name, and to give a brief description of program operation. As

always in this course, good coding style is required.

Do *not* use peripheral library (Plib) functions in the new code that you write. Instead, read and write from/to GPIO registers directly, as in the example code that you are given.

Step 11: Submit your code to Scholar

Create a ZIP archive that contains all of your Lab1 source files (all *.c and *.h files) from Step 10. Use the following file name, and upload it to Scholar before the announced submission deadline:

`<Last name>_<First name>_Lab<lab number>.zip`

You do not have to go the CEL to have your lab validated. When it is time to grade your work, the grader will download your zip file from Scholar, and will compile and run your program on a Digilent board.

Take care to upload the correct files. After submitting to Scholar, you can download what you submitted and verify that they are correct.