ECE2534 Fall 2012: Lab 1
Introduction to MPLabX and the Cerebot MX7 board

This lab is an individual assignment

Step 1: Hardware and Tools required

In order to complete this lab, you will need a Digilent Cerebot MX7cK board, and a Digilent PmodOled (Organic LED Display) peripheral board.

You can order a board through Digilent's website (www.digilent.com). Enter 'vt2012a' as value code and press submit. The kist costs $175 plus shipping. The following components are included.

- Cerebot MX7cK
- PmodACL, 3-axis Accelerometer
- PmodOLED, Organic LED Graphic Display
- PmodKYPD, 16-Button Keypad
- PmodENC, Rotary encoder
- Pmod1 cable kit
- Pmod2 cable kit
- extra USB cable with micro-USB and USB-A connectors

You will also need to have the Microchip MPLAB X IDE and C32 C compiler installed on your laptop. Download these tools from `http://www.microchip.com/mplabide`. Follow the link 'MPLABX Free Download', and select 'MPLABX IDE v1.30' as well as 'MPLAB XC32 Compiler v1.00'. If you receive a request for a license activation key, leave the key field blank, and select the 'free compiler mode' instead.

Step 2: Read the Safe Handling of Printed Circuit Boards document

This document is distributed with your board and Lab 1. Electrical and computer engineers should set an example for avoiding ESD damage, which is more likely in the winter. I would not hire an electrical or computer engineer that did not understand or use ESD precautions. The Cerebot board and the Pmod peripherals are your personal property.

Step 3: Connecting the PmodOLED Display to the Cerebot board

This lab will make you familiar with the Cerebot hardware and software development tools. To complete this lab, you will need the following documents.

- Cerebot Reference Manual:
  `http://digilentinc.com/Data/Products/CEREBOT-MX7CK/Cerebot_MX7cK_rm.pdf`
- Cerebot Schematics:
  `http://digilentinc.com/Data/Products/CEREBOT32MX7/Cerebot32MX7_sch.pdf`

In addition, you can find documentation on the Pmod modules, including the OLED module, at `http://www.digilentinc.com/Products/Catalog.cfm?NavPath=2,401&Cat=9`.

Connect the PmodOLED module to connector JD on the board. Connector JD is located right next to the RESET button of the Cerebot. The display side of the OLED module should face upward. Make sure that jumper JPD, next to connector JD, is set to the position 3v3.

Step 4: Connecting the Cerebot to your laptop

Read the first 10 pages of the MX7cK Reference Manual. Identify each of the jumpers and connectors mentioned. The Power Select jumper block above the Ethernet connector should be set to the DBG option, which obtains power from the debug USB port. Turn power switch SW1 to the OFF (lower) position, and connect the USB cable to the micro-USB connector next to the Ethernet connector. The USB jack on the Cerebot board is a tight fit, so be careful when attaching or removing the USB cable. Connect the USB cable to a USB port on your laptop and turn SW1 on. Debug driver updates will install when the Cerebot is connected for the first time.

Step 5: Create an MPLABX project with the Lab 1 test files

1. Download the OLEDtest source files from Scholar to a project directory named OLEDtest. Please make a disciplined choice for the directory names. For example, create a main directory 'Design' for all your ECE2534 projects on the C drive of your laptop, and expand the files for Lab1 in a subdirectory 'Lab1'.

2. Start the MPLAB X IDE and review the Quick Start material.

3. Create a new Standalone Project, with the `Family` set to PIC32, and the `Device` set to PIC32MX795F512L (the last device in the list), `Hardware Tools` set to Licensed

Debugger, and Compiler Toolchains set to XC32 (v1.00). The `Project Name` should be set to OLEDtest, with the `Project Location` set to the folder containing the OLEDtest directory. Select the `Set as main project` option, and click on `Finish`.

4. In the upper left Projects window, right-click on `Header Files` for the OLEDtest project and select `Add Existing Item`. Select delay.h, OledChar.h, OledGrph.h, and PmodOLED.h. Right click on `Source Files` and select `Add Existing Item`. This time select delay.c, FillPat.c, main.c, OledChar.c, OledGrph.c, PmodOLED.c.

5. Click on `Debug -> Debug Main Project`. The application will be compiled, linked, downloaded and run. You should see the message "ECE2534", "Micro is fun" (it really is), and a high-speed counter appear on the OLED display.

Step 6: Identify the mapping of LEDs and Buttons on the Cerebot

The objective of the second part of this lab is to access the LED and the Pushbuttons on the Cerebot board, through software executing on the PIC32. This will require you to figure out what ports and package pins are being used for each component.

PIC32 pins connected to buttons must be configured as digital inputs, and pins connected to LEDs must be configured as digital outputs. Pages 8 and 9 of the Cerebot MX7cK Reference Manual indicate the PIC32 internal ports and bits that Digilent has selected for the buttons and LEDs.

Do not confuse PIC32 internal ports A through G with the Cerebot 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 Cerebot Reference Manual clarifies the mapping from PIC32 package pins to Cerebot Connector pins, and the PIC32 Port bit.

Identify the ports and pins for each LED and each button.

Step 7: Download the PIC32 Peripheral Library Guide

To shield you from low-level programming, the PIC32 comes with a C library that lets you access I/O ports of the microcontroller. This library, the plib, must be used in this lab. The Scholar website contains a link to the documentation of plib, the PIC32 Peripheral Library Guide. Chapter 10 (I/O Port Library) is relevant to this lab. You do not need to take any new actions to download and install the plib library; the library is part of the MPLAB XC32 compiler.

You will need to write four functions to access the LEDs and the Buttons. The `initButtons()` and `initLEDs()` functions should call the appropriate peripheral library functions to set the relevant PIC32 ports and pins as digital outputs and inputs, respectively. The `getButtonState()`

and `setLEDState()` functions should call the appropriate library functions to read and write the relevant PIC32 ports and bits.

Step 8: Develop the Reaction Timer Application

Using what you've learned on the LED/Button ports, the plib, and the use of MPLABx, now proceed developing the following reaction timer.

1. Download the ReactionTime source (.c and .h) files from Scholar to a project directory named ReactionTime.

2. Create a new MPLAB X project named ReactionTime following the same method as before, except there will be additional header files (Button.h, LED.h) and source files (Button.c, LED.c).

3. Study the Button.c, LED.c, Button.h, LED.h files. The files are incomplete, and need additional code and statements at locations where you see the string "??".

4. You also need to complete the application in main.c according to the following specifications. The application to be implemented in main() should measure a user's reaction time. The instruction strings `Press BTN 1 when` and `LED 1 turns on` should first be written to the first and second lines of the OLED. After a pseudo- random delay of up to 5 seconds, LED LD1 is turned on. The number of milliseconds between LD1 turning on and button BTN1 being pressed is calculated and displayed on the OLED using the string `Reaction time is` on the first line, and `# ms` on the second line. Pressing BTN1 again has no effect, while pressing button BTN2 provides a reset to repeat the process. If BTN1 is pressed before LD1 turns on, the application is reset.

The lab comes with a reference solution that enables you to verify your hardware. The reference solution is a binary (hex) dump of a working reaction time meter. To run the reference solution, you need to create an MPLAb project of 'prebuilt' type. Configure the project as before (same PIC32, same debugger), but select the type as a Prebuilt project, and use the solution.hex file as project implementation.

Step 9: Validate your lab in the CEL using the Lab1 Validation Sheet

Validation (or at least entering into the validation queue) is due by 10:00 pm on Thursday Sep 20. Return the signed validation sheet to your instructor at the start of the next class period. Note that CEL GTA's will give priority to validations on Thursday Sep 20.

Step 10: Submit your code on Scholar

Create a ZIP archive of all ReactionTime source code used in Lab 1 (i.e. only .c and .h files), and submit it to Scholar before 10:00 pm on Friday Sep 21. The ZIP archive should be named

`<Last name>_<First name>_Lab_<Lab number>.zip`

In Windows, this is accomplished by selecting your files, right click on one of them, and choose Send to and then Compressed (zipped) folder.

Be sure to write suitable comments in your code.

Note: The Grade Sheet text file included on Scholar is intended to serve as a checklist. You don't need to do anything with the Grade Sheet, which will be used by the GTAs as a template for grade feedback available on Scholar.