CSCI 345 Laboratory IV

October 1, 2003

Objective

This week you are to test your up-down counter from last week, and then to start working on the next exercise for the course, which is to use the keyboard and touchscreen. The goal for the laboratory session on 10/1 is to read characters from a keyboard and to display their ASCII code on the hex displays of the RC200.

This project will make use of the PAL so far as possible, but you will also learn to work below the PAL for those devices that are not supported by "generic PAL" features.

Once again, you are to work in groups of two or three during the laboratory session. Do the work for the session in one person's account, and then make the directory tree containing the laboratory workspace available to the other members of your team to copy.

Project Specifications

The program you are going to develop is to display ASCII characters as they are typed on a keyboard. First you will develop code to display the hex values of ASCII characters as they are typed on the seven segment displays. Then you are to develop a version of the program that displays the characters themselves on the LCD display of the touchscreen.

Lab Activities

1. Create a Workspace For This Project
2. Share this Laboratory with Your Group
3. USE RC200 Macros for LED I/O
4. Read from Keyboard, Write to Hex Displays
5. Write Characters to the LCD
6. Submit a Report of Your Lab Activities

Create a Workspace For This Project

Create a DK Workspace named "Laboratory IV" for this project in your "My Projects" directory. Add a new project named "LED Buttons" to this workspace, and add a Handel-C source file named *led_buttons.hcc* to the project. Configure the project settings for EDIF so that USE_RC200E is defined, and so the program will be linked to the *stdlib, pal_rc200e,* and *rc200e* Handel-C libraries. The chip part number, in case you somehow forgot, is XC2V1000-4FG456. Don't forget to set up the commands (*cd* and *edifmake_rc200*) and outputs (EDIF) on the Build Commands tab. You will not be able to use the PAL simulator for this project, so there is no need to configure the Debug build configuration.

In your source file, define either the symbol PAL_TARGET_CLOCK_RATE or RC200_TARGET_CLOCK_RATE to a meaningful value, like 50MHz. (There are three clocks on the RC200, which operate at frequencies of 50 MHz, 24.567 MHz, and 25.175 MHz, as shown on page 23 of the *RC200 Hardware Reference Manual*.) Include the necessary header files, *pal_master.hch,* and *pal_rc200.hch.* Write a *main()* function that does nothing much, maybe just increments a counter in an endless loop, and make sure you can compile and build the project.

Share this Laboratory with Your Group

When the project is set up, make the files and folders for this entire laboratory accessible to the other members of your group.

To do this, first log out of your account so Windows will write everything out to the H: drive. Then log back in, and use Windows Explorer to navigate to the workspace directory (Laboratory IV) on the H: drive, right-click on it, and click on the Properties menu. Select the Security tab, and give the other members of your group read access to this folder and all subfolders. You will also need to give the members of your group permission to browse all your folders above the Laboratory IV directory. Test that you have this set up right by logging out of your account, having another member of the group log in and copy the workspace from the original student's account on the H: drive to the other student's account on the C: drive.

USE RC200 Macros for LED I/O

The header file, *pal_rc200.hch*, defines macros for accessing the various devices on the RC200 board. Actually, this header file just defines macro expressions for PAL handles to the various devices, and it includes another header file that makes external references to the macros that actually perform the I/O operations. The macros themselves are defined in the *pal_rc200e.hcl* library you are linking to your code.

Find the *pal_rc200.hch* header file on your system and examine it with a text editor (*vim* or the DK editor). Write as simple a program as you can that will turn the two LEDs on and off as the buttons are pressed and released. Download and test your code. Because you are not using the PAL, you will not be able to simulate this code.

Read from Keyboard, Write to Hex Displays

For this step, we will revert to using the PAL for all I/O, which will allow us to simulate the code as well as to download it.

Create another project in your Laboratory IV workspace named Keyboard-Hex, and add a source file named *keyboard_hex.hcc* to it. You can start *keyboard_hex.hcc* by copying the code from *led_buttons.hcc* if you want to, but you still have to configure the project's build settings.

> **Shortcut: When you are working on the dialog box for the project settings, you can click on the LED Buttons project in the "Object" pane on the left to see what the settings were there, and do copy-paste.**

You will need another library for this project, *pal_keyboard.hcl*. You will also need to include *pal_keyboard.hch* in your source code. To read from the keyboard, you must run the function *PalKeyboardRun()* in parallel with calls to *PalKeyboardEnable()* and *PalKeyboardReadASCII()*. Your job is to figure out the parameters to use for these various functions.

- You will need a variable of type pointer to PalKeyboard.
- Remember, that you can get the actual clock rate for your *main()* function by using the macro PAL_ACTUAL_CLOCK_RATE once you have included *pal_master.hch*.

- Handles to the PS/2 ports (mouse and keyboard) are accessible using the PAL macro *PalPS2PortCT()*. An argument of 0 gives a handle to the mouse, and an argument of 1 gives a handle for the keyboard.

You *could* look at the Keyboard program in the sample code provided by Celoxica, but that probably wouldn't be as good a learning experience as getting the code to work on your own.

When you simulate the code, you can enter input values for the PS/2 Keyboard port in the PAL Virtual Platform window, but you have to enter the scan codes that the keyboard generates, and you have to enter them as decimal numbers. You can look these scan codes up on the web. A good resource that I found is at Adam Chapweske's "PIC micro-Resources" page. Microsoft also publishes documentation on scan codes at their Keyboard Support page. I've made a PDF copy of their *Keyboard Scan Code Specification* Word document that you can download from here. As Chapweske's pages indicate, you need to use scan code set number 2 (of three different ones used by different types of keyboards). For example, the sequence of scan codes 28, 240, 28, 89, 28, 240, 28, 240, 89 would represent the sequence, press A (0x1C), release A (0xF01C), press right-shift (0x59), press A (0x1C), release A (0xF01C), release right-shift (0xF059), that is, typing the characters, 'a' and 'A', resulting in the hex display of those two characters, 0x61, and 0x41. I've found that the first four scan codes entered seem to be ignored by the simulator. And it's not necessary to enter the release codes.

Be sure your code works correctly when you download it to an RC200E with a keyboard attached to it.

> *Caution: Unplug the RC200E before connecting or disconnecting any peripheral device such as a keyboard or mouse.*

Write Characters to the LCD

For this part of the Laboratory, you are to create a third project called Keyboard-LCD containing a source file named *keyboard_lcd*, which reads from the keyboard and writes each character to the touchscreen's LCD. You can use the Keyboard sample code from Celoxica as a model; we will examine an alternative approach in the next lab.

Submit a Report of Your Lab Activities

Submit copies of *led_buttons.hcc, keyboard_hex.hcc,* and *keyboard_lcd.hcc* by email by Wednesday October 8. Be sure your code is properly documented and formatted.