Exercise in Embedded Computing: The Basics

Sivan Toledo, Tel-Aviv University

October 13, 2010

General Guidelines

The following guidelines apply to all the exercises, problem sets, and projects in this course. Read them carefully. But submitting the first exercise, you acknowledge that you read and understood them.

- The staff will loan you equipment to perform the exercises. You are responsible for the equipment which it is on loan. Treat it carefully and respectfully.
- In particular, return all the equipment to the staff at the end of each lab session.
- The boards are sensitive to static electricity. Take the following precautions to avoid damage to the boards:
  - Do not touch exposed contacts unless you must. When handling a board, hold it by its sides.
  - When the board is not in use, keep it in its anti-static plastic bag.
  - Avoid wearing synthetic materials to the labs if possible, especially fleece. You must take off fleece clothing before handling the boards. If you take off a synthetic item of clothing, touch a grounded metal before handling a board, to discharge static electricity.
- The USB connectors on the LPC2148 Education Board are fragile. Do not exert force when plugging in USB cables, and make sure that the cable does not apply force on the board or the connector.
- Careless handling of the boards will reduce the participation part of your grade.
- You can access the boards only during lab sessions, so we intend that you use the entire lab session to code, debug, and experiment. If you finish early, implement some of the *additional ideas* that appear at the end of the exercise, or invent your own enhancement. You should normally not leave early unless you have finished the exercise plus a non-trivial additional feature.

- Each student or team must solve each exercise on its own. You may not use without permission any code except for code that the course staff provides you with. In particular, you may not use code from internet web sites or mailing lists, and you may not use or read code by other students in the course except for your team members.
- At the end of each lab session each team must turn in a short report that sumarizes what you accomplished, covering both required parts of the exercise and additional features. The report should not exceed two pages. Turn the report to the teaching assistant and demonstrate the system to him.
- Keep the code (and the report if you typed it) on the web page of one of the team members, as follows:
  - The solution to exercise 1 (this exercise) should be in the directory `~/html/embedded/exercise1`, and so on for the next exercises.
  - Make sure that the staff and the web server can access these files; use the commands `chmod 755 ~` and `chmod 755 ~/html` once, and the command `chmod -R 755 ~/html/embedded` every time you add a solution.
- The public materials for this course, including documentation and exercise templates are available at `/users/courses/embedded`, as well as from `http://courses.cs.tau.ac.il/embedded`

The Exercise

The purpose of this exercise is to familiarize you with the LPC2148 Education Board, with the development environment, and to experiment with digital I/O, timers, and interrupts.

1. Checking that the development environment works.
   (a) Log in and open a terminal (command prompt) window. Create a directory for the exercise, and copy into it all the files from `/users/courses/embedded/exercises/basics`. The copying must be recursive since this directory contains another one with files in it. If you prefer to use Eclipse, start up Eclipse and create an empty C project. Then copy the files into this new project.
   (b) Type `make` and check that the build process created the image file `isr-led-switch.hex`.
   (c) Connect the board to the computer's USB port. There are two USB ports on the board; use the one marked `UART #0`.
   (d) Type `make program`. This should download the binary image to the board and run it. Press the push button marked `P0.14` and check that the LED marked `P0.10` toggles. If it does, you have a working development environment.

2. The program that you compiled and loaded toggles the LED, but the toggling is not reliable. Sometimes the LED toggles a few times when you push the button once, very quickly. You can't see this rapid toggling, but if the number of time the LED toggles is even the button press will appear to do nothing. This happens because the switch is mechanical and its contact bounces, closing and opening the circuit a few times before settling.
   Use the timer to implement *switch debouncing*. Make the code insensitive to the switch in the first 50ms after a valid transition. Use interrupts for full credit, or polling for partial credit. It helps to think of this task as a state machine when you designn the program.

3. The switch on the board is not very prone to bouncing, which makes it challenging to make sure the code works as intended. Explain in your report how you tested your code.

4. Now emulate a bicycle blinker. The LED should be initially off. When the user presses the switch once, the LED turns on. When the user presses the switch again, the LED starts to blink. A third switch turns the LED off. Now the user can turn it back on, etc. Your program is not power-efficient enough for a battery-operated bike blinker; to conserve energy, the processor should sleep when the LED is off. In your program, the processor will execute a loop when the LED is off. Later in the course we'll learn how to make this program energy efficient.

5. Additional ideas:

   (a) Use some of the other 8 LEDs on the board. Some of them are connected to pins that also serve other functions, like our switch, so it is not possible to control them without loosing the other pin function. But a few are not and can be easily used. Implement a rolling light, where one LED is turned on for a short period, then its neighbor, then a third, in a cyclic pattern.

   (b) Next to the `P0.14` push button there is a joystick-type switch. Use it to control the LEDs.

   (c) Pin `P0.7` controls a buzzer. Modify your program to control it rather than an LED, so that it emits a short beep every time the button is pressed. The jumper that controls the buzzer may be in the off position; if so, connect it to allow control of the buzzer.