Exercise in Embedded Computing: USB Devices

Sivan Toledo, Tel-Aviv University

December 21, 2009

This exercise exposes you to the firmware of USB devices. **This exercise has a lot of room for improvement**.

1. Copy the project from `exericses/usb`, build `test-usb-serial` and upload it to the LPC2148 Education Board. The code uses the USB port of the LPC2148 as a virtual serial port.

2. Run the code while connected to the LPC2148's UART through the USB-to-serial bridge that we use to program the microcontroller.

3. Plug in a second USB cable to the connector marked "USB". Check whether the device prints out diagnostic messages to the UART.

4. When you plug in a USB device, the host enumerates it and loads an appropriate driver, if there is one. Under Linux, the host also prints out diagnostic messages to the system log. You can view the latest messages using the `dmesg` utility. Run it; you should be able to see that the host detected the device correctly.

5. Open a connection to the new virtual serial port (`dmesg` should show you its name) using the utility `cu`. Type text and watch what happens. You can also inspect `main` to figure out what the code is doing with the data it receives.

6. Can you run `cu` at different Baud rates and still communicate with the device?

7. Hack the USB driver in `usb.c` so that it does not return the device descriptor when the host requests it. Check the `dmesg` output; the messages you see are helpful in debugging the USB driver. (The dirver we have works, but it is not very good; for example, it could use DMA but it does not; if you add DMA support, you will need to debug it.)

8. Fix the driver so that it works. Implement a small command language that will allow the host to control the LCD. Commands from the host should be able to turn the backlight on and off and to display messages on the display. The commands should be received through the virtual serial port. Test the code using `cu`.

9. Now build and test `test-usb-mouse`.

10. Additional ideas:

    (a) Add functionality to the mouse firmware. For example, use the P0.14 button as an extra mouse button. This requires some understanding of the mouse HID specification.

    (b) Modify the mouse device into a keyboard.

    (c) Modify the mouse device into a composite device that contains both a mouse and a keyboard.

    (d) Write a Linux-side command-line utility that will control the board through the virtual serial port. The utility can inspect `/usb/bus/usb` to determine the name of the serial port. (**WARNING: for some reason, /proc/bus/usb does not seem to work in the school.**)

    (e) Implement a custom USB HID device that controls the LCD and a Linux or Windows utility that communicates with it.