Professor Fearing EECS192 Debugging Checklist v1.01 Spring 2015

Debugging Commandments

Anything that can go wrong will go wrong. Which means that you need to follow this checklist every time you want to debug. (adapted from `www-inst.eecs.berkeley.edu/~ cs150/fa98/labs/debug/debug.html`)

1. Collect documentation.
   Don't use processor bandwidth remembering design details. Save processor bandwidth for observations and questions. Have schematics, board layout, wiring diagrams, data sheets, documented code and any needed manuals open and ready use.

2. Visually Inspect the Board/Setup
   A quick check to verify that chips are still plugged in, no wires have broken, or pins bent, is worth a minute, before you start, but is unlikely to find most bugs.

3. Check Software Functionality
   Is the computer system and network functioning? Are your files missing pieces or corrupted? (You do have memory stick backups, right?) Verify that the software tools are working as expected. Test a known working simulation/example; do the tools still work?

4. Check Hardware Functionality
   Are all cables, oscilloscope and logic probes working? (You can check the oscilloscope probes by using the Calib output on the front of the oscilloscope. You should see a nice square wave). Is test equipment set up for desired sweep rate, volts-per-division, etc. (Please notify your TA about bad cables so other students won't have to find them the hard way). Do you really have power supply voltages correct (Verify with scope)? Are grounds tied together for battery, CPU, power supply, and test equipment, or are you relying on the AC power line to provide a common ground? Verify that basic functionality by testing a known good circuit.

5. Check System Essentials
   Is the clock running at the expected frequency? Do voltage levels seem reasonable (using oscilloscope to measure)?

6. Divide and Conquer

   Isolate modules and debug each module independently.

7. Scientific Debugging

   The four keys to scientific debugging are:

   (a) Model Expected Behavior. What do you expect to see? Simulators tell you what you have, not what you need to see.

   (b) Measure the real system's behavior using oscilloscope. Where are the differences from your model significant?

   (c) Hypothesize possible errors.

   (d) Controllability and Observability. Force module inputs to verify hypotheses, and observe outputs. Are outputs consistent with your model, or is something not functioning as expected? (Various test cases- DC input, AC input, single stepping).

8. Understand Component Devices
   Are you sure you understood the data sheet correctly on that part?

9. Know your Software
   Is the software working as you expect it to? Is there a mismatch between your expectations and the result? Are there hidden settings/state which changed from a previous session? Are you using the right commit?

10. Know Your Test Equipment
    The digital scope will give nonsense if the sweep rate is too high or too low. When in doubt about a signal, verify that you have proper voltage levels and timing using the oscilloscope.

11. Take a Break/Get a New Perspective
    Debugging is mentally taxing, and we tend to focus in narrowly on particular suspect areas. Taking a break opens the mind to a wider range of possibilities and can get you out of a rut.