CM506 – Small Embedded Systems

Debugging

Dr Alun Moon

Department of Computer and Information Science

Lecture 7

when you have eliminated the impossible, whatever remains, however improbable, must be the truth

— Sherlock Holmes

Why Debugging?

- Once you have a program you can download into the ARM hardware, it has passed through the build process.
  - it is syntactically correct – it has compiled
  - all variable and function calls are resolved in memory – it has linked
- Any further errors occur at runtime. (The compiler can't help you here)
- Debugging isolates and identifies where the program is erroneous
  - not doing something it should
  - doing something it shouldn't

Identify the Problem

- What are the symptoms?
  - stick to what it is (is not) doing
  - don't try to guess cause yet
- What are the related sections of code?
  - Be comprehensive, initialisation, input/output, interrupts
  - be alert, the cause may be elsewhere
- Try not get sucked into one narrative of what is wrong, it closes off other potential sources, be broad-minded as to causes.

The Debugger

The Debugger is a valuable tool for examining the actual state of a program on the machine.

- variable examination – what are the actual contents
- variable manipulation – can change variables contents
- memory and register examination
- code execution – flow of the program

Watch lists

- lists of variables to monitor
- displays their value when the program is halted
- whether the variable is in scope (is visible) for the line of code

Program Execution

The debugger can control the execution of the program, running and halting the program.

- Single step – executes a line of source code (C or assembly)
- Step into – steps into a function call
- Step over – steps over a function as it it was a single statement
- run.

Breakpoints

A breakpoint is a predetermined point in the program where the execution halts.

- good for stopping before a critical section to step through
- good for stopping at a "waypoint" so check state of variables
- good for checking if a line of code is executed – is the breakpoint hit.

1. Identify the symptoms
2. Isolate relevant sections of code
    1. examine in detail, execution and variables
    2. be comprehensive (is the interrupt flag cleared)
3. Step back – problem might be elsewhere
4. Have I covered everything?