Embedded Software Development

Rolfe Bozier

20-Feb-2014

Introduction

- Many of us write software that ends up in an embedded system, but...
  - what is an embedded system exactly?
  - how does my software fit in?
  - what are the ramifications for what we write?
- What is an embedded system?
  - "A software environment often of limited functionality which is created for a special purpose"
- A continuum ranging from tiny 8-bit micro-controllers up to systems approaching general purpose computers
- *Lots* of examples: cars, toys, appliances, printers, cameras, watches, phones, ...

Evolution of embedded systems - 1

- Microprocessor-based systems
- Separate ICs for:
  - CPU, ROM (software), RAM, Interface logic
- Processors: Z80, 68000, 8088+
- Look like desktop computers from 30-40 years ago

Evolution of embedded systems - 2

- Micro-controller based systems
- uC (CPU, ROM, RAM, I/O)
- Single IC containing:
  - CPU, Flash RAM (software), RAM, I/O controllers
- Range from tiny 8-bit devices to medium-sized 32-bit devices

Evolution of embedded systems - 3

- Programmable/custom hardware
- Single IC containing your choice of:
  - CPU[s], ROM (software), RAM, Interface logic
  - You choose clock speed, complexity etc.
  - Common processors: ARM, SPARC, MIPS, 8051, MSP430
- Team up with large external RAM, ROM, Flash for more capable systems
- FPGA - programmable hardware
  - A large array of general purpose logic elements that can be configured to implement processors, memory banks, interfaces etc.
  - extremely flexible
- ASIC – custom hardware
  - take your hardware model and fabricate it in silicon
  - minimize size (cheaper) but higher up-front cost

Examples

- Wireless temperature sensor
  - 2K Flash, 128b RAM, 8-bit @ 3.68MHz
- KVM
  - Xilinx FPGA, Xilinx CPLD
  - Atmel microcontroller (4K Flash, 8-bit @ 8MHz)
  - Renesas microcontroller (128K Flash, 32-bit @ 20MHz)
  - Flash, 2 X SRAM
  - Logic

Examples

- Brother printer (HL-1270N - 1999)
  - ASIC (SPARClite core @66MHz)
  - 1MB ROM (software), 4MB RAM
- Brother printer (HL-1650 - 2001)
  - ASIC (SPARClite core @ 96MHz, bus I/F etc)
  - 16MB ROM (software), 2MB Flash, 8MB RAM
- Canon printer (MG5560 - 2013)
  - ASIC (ARM + “other” cores, ROM (software), bus I/F)
  - 32MB Flash (more software), 128MB RAM

Impact on the developer

- Processor can be limited
  - slow clock speed (< 1GHz)
  - small word size (32, maybe 16 or even 8)
  - probably no FPU
  - no MMU -> no memory protection
  - usually stuck with assembler or C/C++
  - the compiler may have some "issues"
  - performance is going to be slower (maybe an order of magnitude)
  - watch out for operations that have to be emulated
    - divide / modulus
    - floating point
    - operations larger than word size
  - you can finally use `volatile` for its intended purpose!

Impact on the developer

- ROM space limited
  - get rid of unnecessary code
  - restructure code to let the linker drop unrequired objects
  - maybe replace a set of optimised functions with a single, slower version
  - don't drag in unnecessary code from the OS support library
  - forget about the POSIX API

Impact on the developer

- RAM space limited
  - 128B through to 10s of MB
  - no virtual memory
  - no heap (malloc)
  - no high-level languages, no recursion
  - think about how much memory you need
  - think about how much memory you ***really*** need:
    - Global data – pre-allocated at start-up
    - Stack data (local variables, function arguments) grows from one end of memory
    - Heap data (malloc) - if you have one - grows from the other
    - What happens when they collide? There is likely no protection…

Impact on the developer

- System issues
  - small or non-existent caches
  - maybe you don't get cache coherency
- No/limited OS
- Cooperative multitasking rather than pre-emptive multitasking
- Maybe no parallel threads
- No keyboard, mouse, display, file-system, disk, ...
- Debugging much harder
  - Simulate the system instead
  - Dig out the logic analyzer!

Why do it then?

- Cost
  - If you're going to sell millions of units, embedded systems are cheaper
  - Cheaper to move costs from hardware (per unit) to software (one-off)
- Real-time!
- Hardware/library support for useful stuff
  - serial I/O, USB, timers, ADCs, ...
  - high-speed bulletproof I/Fs (PCI, Ethernet, etc.)
- Low footprint (physical size, power requirements)
- Security