

Introduction to embedded operating systems - eCos® 2008

eCos is a registered trademark of eCosCentric Limited

Øyvind Harboe, General Manager, Zylin AS



eCos overview

- Not Linux
- Does not require MMU
- Not 8 bit library OS
- Requires 10-100s of kBytes of memory, uCLinux megabytes of RAM
- A raft of deeply embedded operating systems out there
- eCos is open source, free(as in speech, not beer) and widely deployed(growing)



eCos demo

- ZY1000 running web server



GCC toolchain

- eCos only supports the GCC toolchain
- GCC is the “gold standard” in compilers – a must have
- 32 bit programming model
- Open source and free
- Binaries of GCC freely available for any mature CPU
- It is possible to build GCC yourself, but it is not for the faint of heart
- See <http://www.zylin.com/gccbinary.html>



GCC C/C++ libraries

- All standard C functions available
- C++ STL(including iostreams & pthread exceptions) supported
- Many POSIX C functions available(e.g. sockets, fileio)
- C/C++ skills from PC development reusable with eCos HAL in place



GDB debugger

- GDB is the “GCC debugger”
- GDB is a “low-level library”, not a GUI
- Various graphical frontends exist
- Insight
- DDD
- Eclipse CDT



Get a JTAG/hardware debugger

- As little as 100EUR
- Pay peanuts get monkeys
- 2000-4000 EUR for new targets
- Zylin provides the Goldilocks JTAG debugger: ZY1000
- Excellent hardware debugger. Oriented towards more mature targets.



eCos HALs

- HAL = BSP
- Supports “all” deeply embedded CPUs
- Most importantly requires a GCC toolchain
- A HAL for a CPU + PCB means writing a slight variant on existing HAL. It is not hard. Ca. 1000-2000 lines of code + drivers. Experience helps a lot.
- Best practice: get hardware with an eCos HAL, alternatively get help from an experienced eCos engineer for the first mile
- With the HAL in place, things are relatively straightforward. POSIX API, pthreads or eCos API



Drivers

- eCos comes with many drivers
- In addition to standard drivers, rolling your own is not hard
- No need for a driver for your hardware if your application accesses the hardware directly



eCos modules

- TCP/IP – web servers
- USB
- Serial ports
- Timers
- Compression
- GUI (FLTK)
- Just about anything you can think of and more
- POSIX code compiles with few if any modifications to code



eCos kernel

- All RTOS functions
- Threads
- Mutex
- Interrupts
- SMP (symmetric multiprocessing)
- etc.



eCos fileio

- JFFS2 (flash filing system)
- ROM
- FAT
- Roll your own(e.g. /tftp).
- Naming inspired by Linux(mount points and forward slashes)
- etc.



eCos synthetic target

- Non-sequitor...
- eCos can run in thread under Linux
- Useful for testing/development



RedBoot bootloader

- Not for the end-user
- Geek command line for uploading new applications to flash/RAM
- Includes GDB communication protocol support
- Definitely useful when familiarizing oneself with eCos
- Useful when testing new target & flash drivers



Custom bootloader

- Likely you're better off writing your own
- RedBoot is nothing more than a small eCos app
- A typical case for a complete custom bootloader; ~500 lines of code.
- eCos provides all the bootloader guts, you just have to piece it together in a way that makes sense for your application
- Add production features?
- Self tests?
- User friendly firmware upgrades?



Install eCos

- Install Cygwin or Linux
- <http://www.andlinux.org/> under Windows?
- Install eCos tools
- Fetch latest eCos source from CVS repository
- Ignore eCos releases, use CVS HEAD
- eCos web pages are dated. The project is most definitely alive.



eCos development flow

- Enable relevant options and modules using ecosconfig
- ecosconfig GUI is preferred by some
- Build eCos – happens rarely
- Link eCos lib w/your target using your own makefiles
- If you need to modify eCos, consider making a separate eCos repository that you keep in your own version control system



eCos hello world build demo

- Show how to build a hello app
- Run it



eCos 3.0 coming up

- More of a marketing ploy
- Already available from CVS HEAD
- Probably there will be some new features announced.
More better Coretex? More C++ features?
- It makes little sense to to pile on features with deeply embedded targets
- eCos really is quite stable and does not change much



Writing custom HAL

- <http://www.zylin.com/ecosha.html>
- Put your HAL in your own repository
- Stored in your version control system
- Take a snapshot of the official eCos repository and upgrade as needed
- Commit toolchain binaries to version control



Zylin AS

Embedded services

2008

Øyvind Harboe, General Manager, Zylin AS

