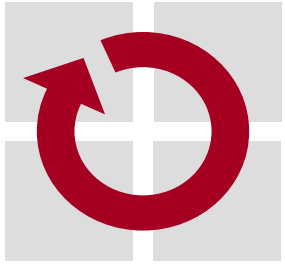
HotSys Praktikum

Anwendungsentwicklung mit eCos

24.11.2010

Martin Hoffmann
Peter Ulbrich
Jürgen Kleinöder

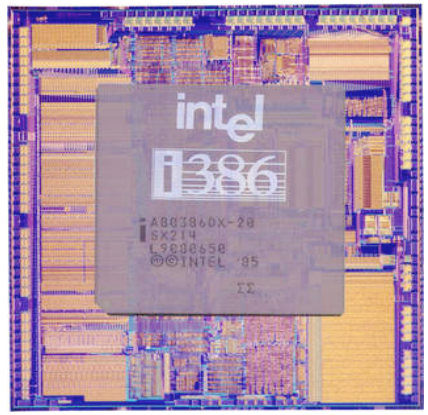
`http://www4.cs.fau.de/~{ulbrich, hoffmann, jklein}`
`{ulbrich, hoffmann, jklein}@cs.fau.de`

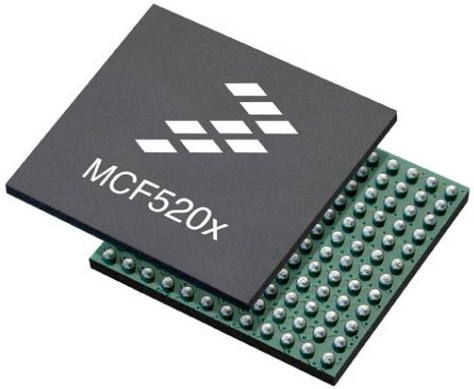


Lehrstuhl für Informatik 4
Verteilte Systeme und Betriebssysteme
Friedrich-Alexander-Universtät Erlangen-Nürnberg

Große Prozessorvielfalt

freescale™ semiconductor

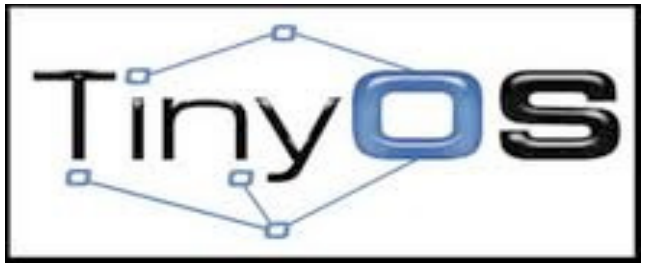


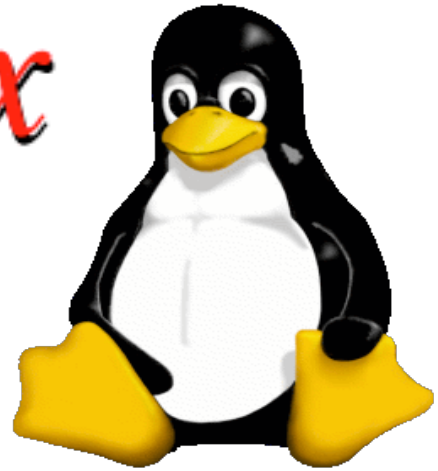


Cortex™
Intelligent Processors by ARM®

ARM.

Noch mehr Betriebssysteme





Effiziente (embedded) Entwicklung

- Aufmerksamkeit auf *funktionale* Aspekte
  - Frühe Fokussierung auf Anwendungsentwicklung
  - Betriebssystemaspekte sind eher nicht-funktional!
- Wiederverwendung
  - Übersichtliche Konfiguration
  - Gute Portabilität
  - Einheitliche Softwareinfrastruktur

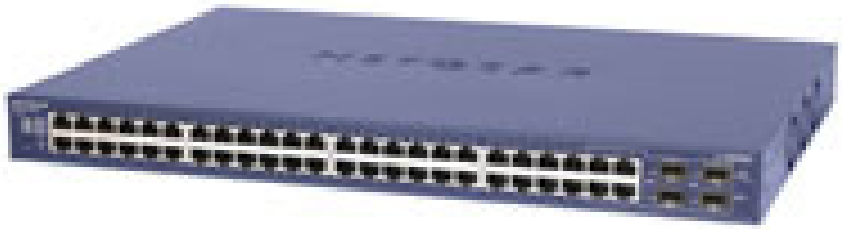
„eCos is an embedded, highly configurable, open-source, royalty-free, real-time operating system...“

http://ecos.sourceware.org

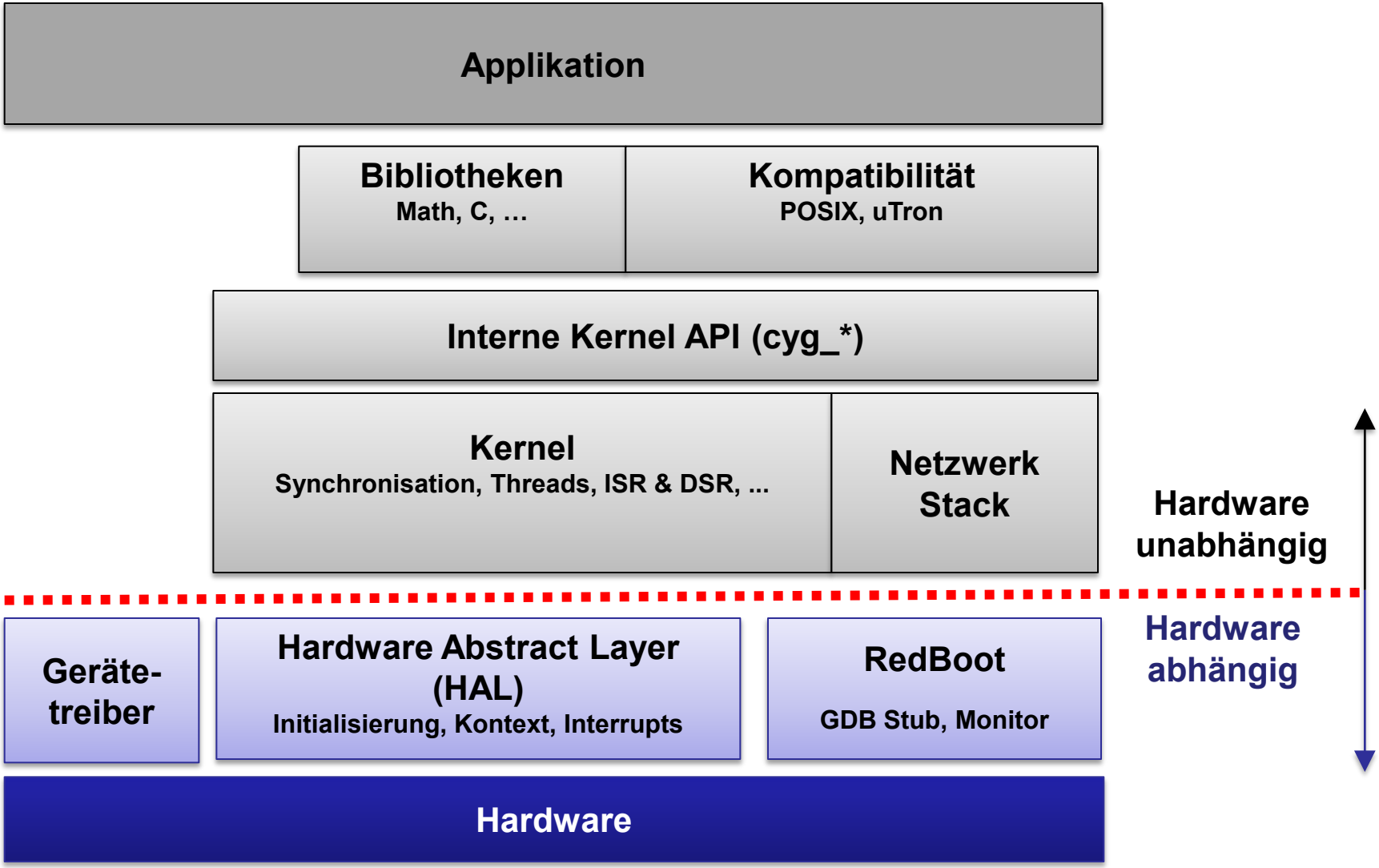
- Ursprünglich von Cygnus entwickelt (1998)
- Primäres Entwurfsziel:
  - „deeply embbeded systems“
  - „high-volume applications“
  - „in consumer electronics, telecommunications, automotive...“
- Zusammenarbeit mit Redhat → Redboot (1999)
- Seit 2004 OpenSource (BSD ähnlich)
- Heute: Maintainer ecoscentric

Unterstützte Prozessoren

- Advanced RISC Machines (ARM)
- Fujitsu SPARClite
- Matsushita MN10300
- Motorola PowerPC
- Toshiba TX39
- Hitachi SH3
- NEC VR4300
- MB8683X
- Intel Strong ARM
- Intel x86
- (TriCore)
- ...

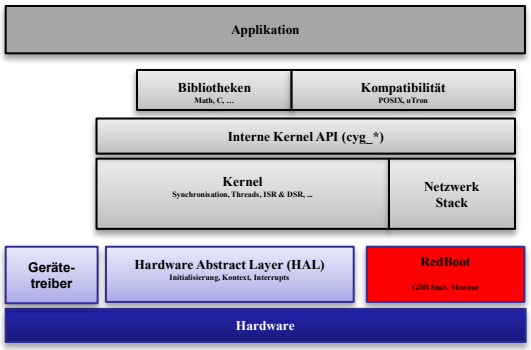


eCos Systemarchitektur



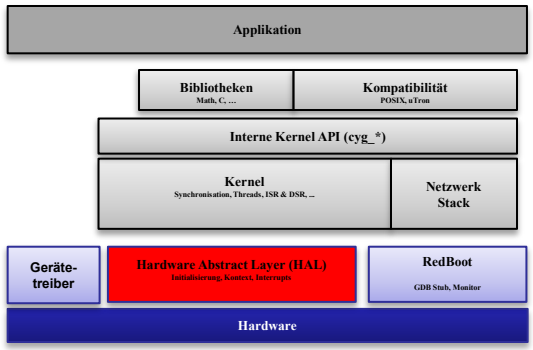
Redboot

- User Interface (RS232, telnet, Tastatur/Monitor)
- Bootloader unterstützt laden per
  - Festplatte, MMC, SD (ext2, FAT16/32)
  - Serielle Schnittstelle (X/Y-Modem)
  - Netzwerk (tftp, http)
  - Flash (JFFS2)
- Debug Monitor (GDB Stub)
  - Remote Debug Protocol per RS232 oder Netzwerk
  - Unterstützt eCos Threads!
  - Teilt sich Netzwerkkarte mit Applikation



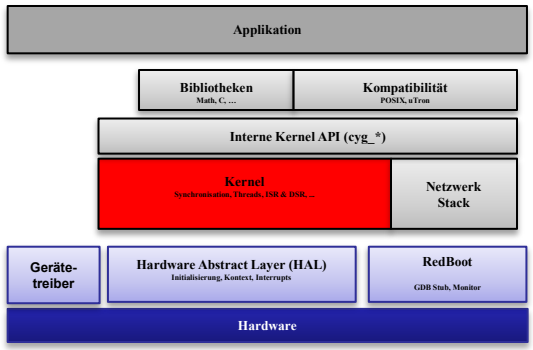
Hardware Abstraction Layer

- Abstrahiert CPU- und plattformspezifische Eigenschaften
  - Kontextwechsel
  - Interruptverteilung
  - CPU Erkennung, Startup
  - Timer, I/O Registerzugriffe
  - Interruptcontroller



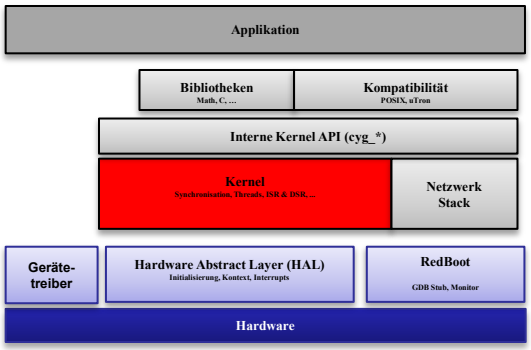
Kernel

- Implementiert in C++ (ermöglicht einweben von Aspekten)
- Feingranular konfigurierbar
  - Verschiedene Scheduling Algorithmen (Bitmap/Multilevel Queue)
  - Zeitscheiben, Präemptiv, prioritätenbasiert
- Synchronisationsstrategien
  - Scheduler Lock
  - Mutex, Semaphore, Condition Variable, Flags
  - Message Boxes



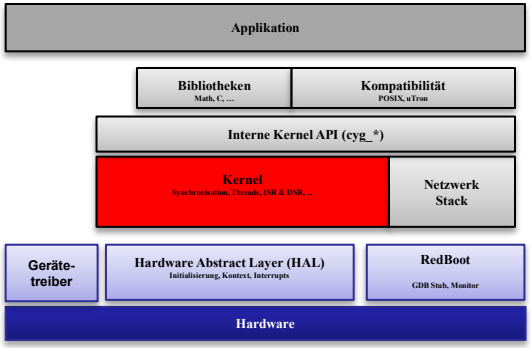
Interruptbehandlung

- Aufgeteilt wie in OoStubs/EzStubs
  - Interrupt Service Routine (ISR)
    - Unverzügliche Ausführung
    - Asynchron
    - Kann DSR anfordern
  - Deferred Service Routine (DSR)
    - Verzögerte Ausführung
    - Synchron



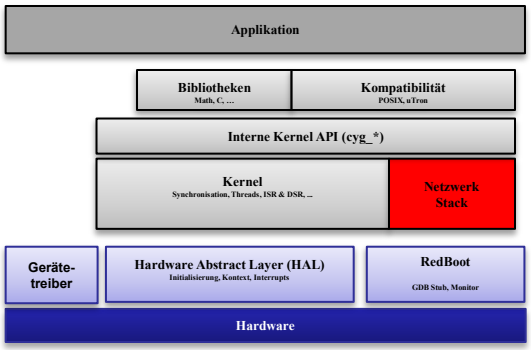
Profiling

- „Kernel Instrumentation“: Tracing von
  - Schedulerereignissen
  - Thread Operationen
  - Interrupts
  - Mutex/Semaphore Operationen
  - Clock Ticks
- Individuell konfigurierbar
- Gprof Unterstützung



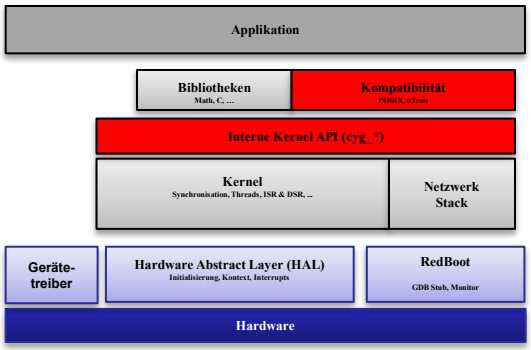
Netzwerkstack

- OpenBSD (deprecated)
- FreeBSD (KAME Projekt)
  - IPv4/6
  - ARP, RARP, ICMP, IGMP,
  - TCP/UDP
  - DHCP, BOOTP, TFTP
  - Multicast addressing/routing
  - Berkeley Packet Filter (BPF)
- Leightweight IP Stack (lwIP von Adam Dunkels)
  - TCP/IP für (deeply) embedded systems
  - Minimum: 10 kB RAM, 40kB ROM



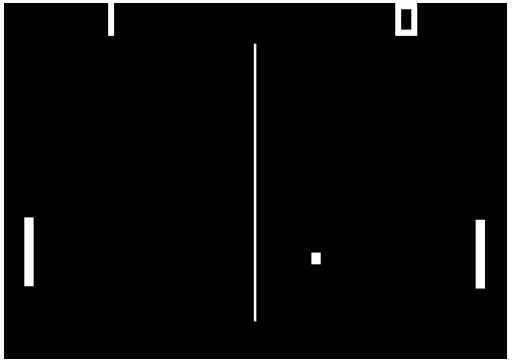
APIs

- Interne Kernel API
  - C-Schnittstelle
  - siehe Dokumentation
- Kompatibilitätsschichten (optional)
  - Posix
    - Scheduling Konfiguration, Pthreads
    - Timer, Semaphore, Message Queues, Signale
  - uITron
    - Reine Schnittstellenbeschreibung ähnlich OSEK oder Posix
    - Weit verbreitet im asiatischen Raum
    - V.a. Unterhaltungselektronik

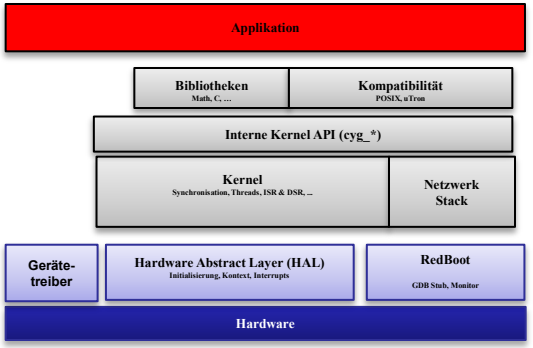


Übungsaufgabe

- **C++** Applikation: „Remote-Pong“ auf x86
  - Steuerung auf entfernten „Clients“ per telnet (Character Mode)
  - Anzeige auf eCos „Server“ über Framebuffer
  - Lernziele:
    - Konfiguration der eCos Library
    - Applikationsentwicklung
      - eCos Threads, Socket-API (~ Linux)
      - Framebuffer schon fertig...
    - GDB/DDD Remote Debugging

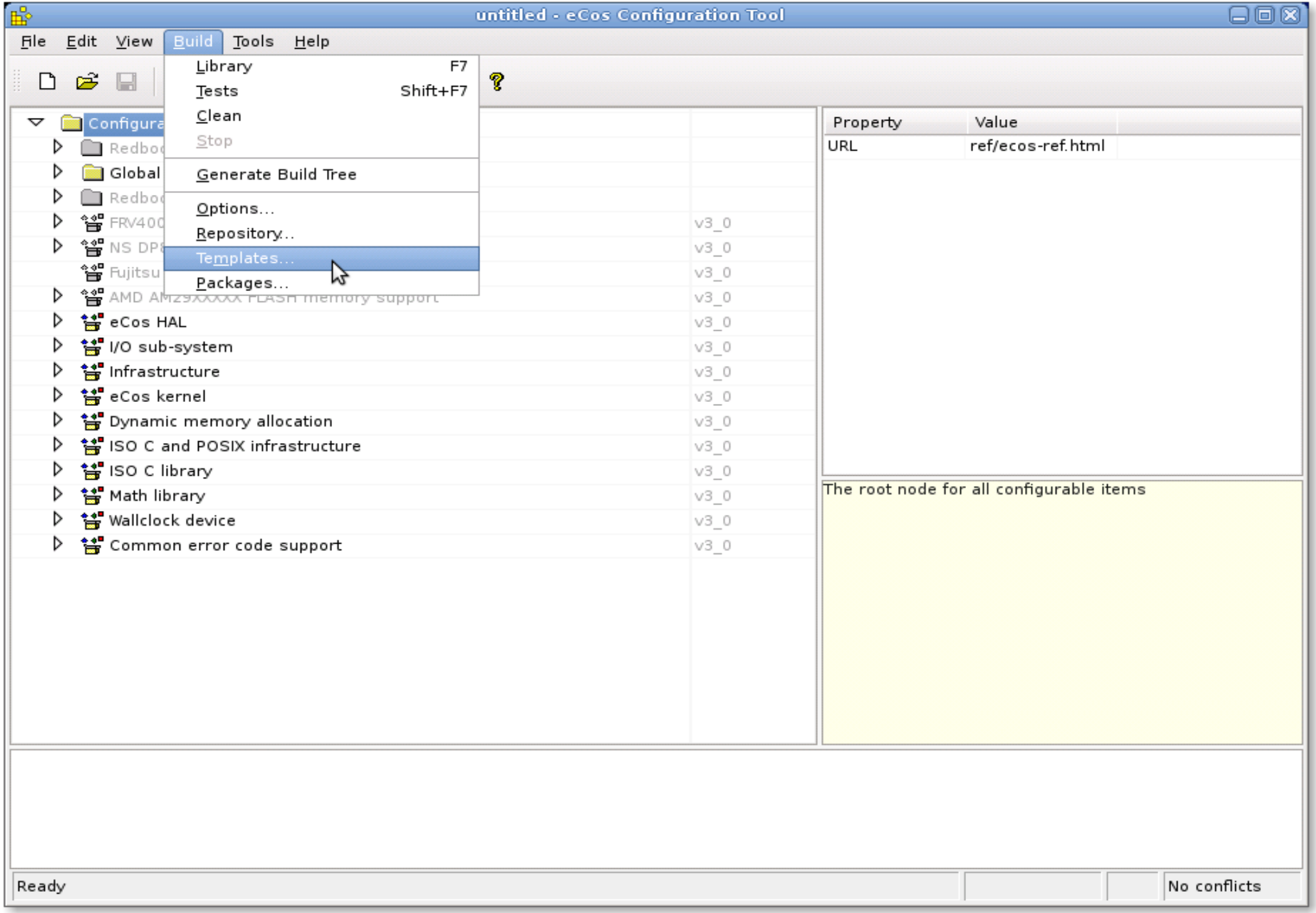


Dokumentation
http://ecos.sourceware.org/docs-latest/

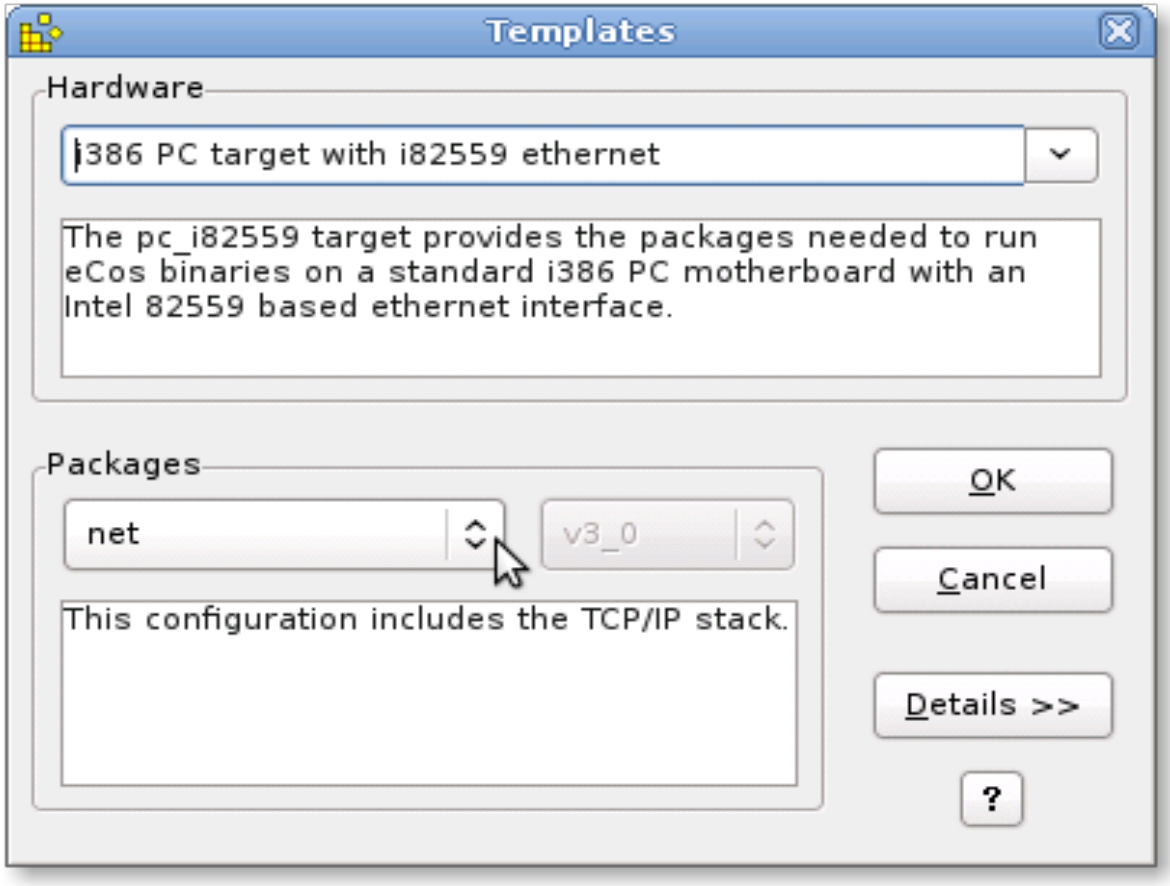


Konfiguration

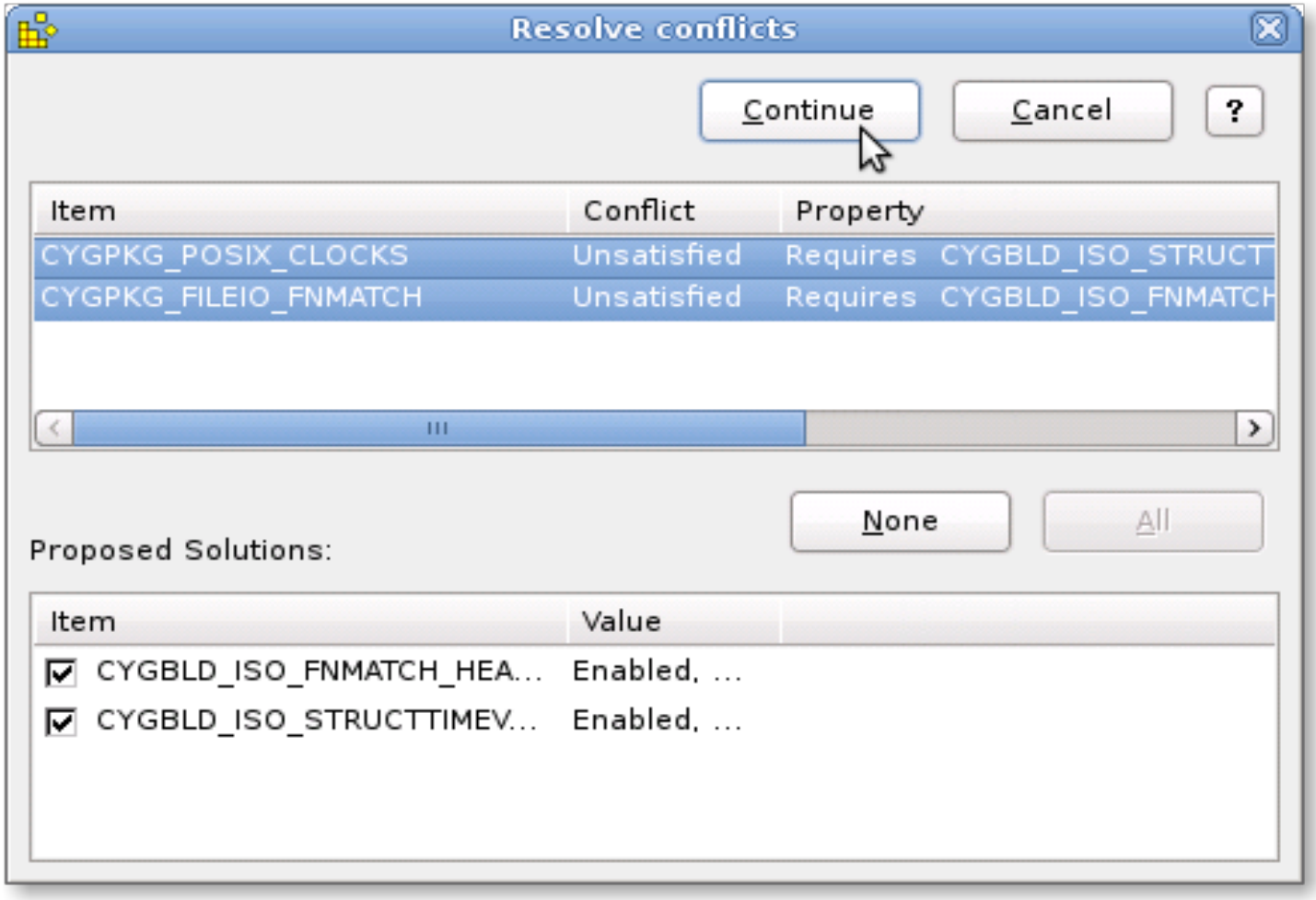
- CDL Component Definition Language
- Grafische Oberfläche (configtool)
- Kommandozeilenwerkzeug (ecosconfig)
- Ablaufbeispiel
  - 0. source ecosenv.sh (Setzt nötige Umgebungsvariablen)
  - 1. Auswahl eines Templates (z.B. i386 with i82559 Ethernet)
  - 2. Hinzufügen des Framebuffer Support (Add Package...)
  - 3. Auflösen der Abhängigkeiten (automatisch)
  - 4. Einstellen der Konfigurationsparameter
    - Speicherlayout (Boot from RAM, Load into HighMem > 2 MB)
    - Netzwerkeinstellungen
  - 5. Speicher der Konfiguration → Tool erzeugt Systemheader
  - 6. Bauen der Library -> Build Library

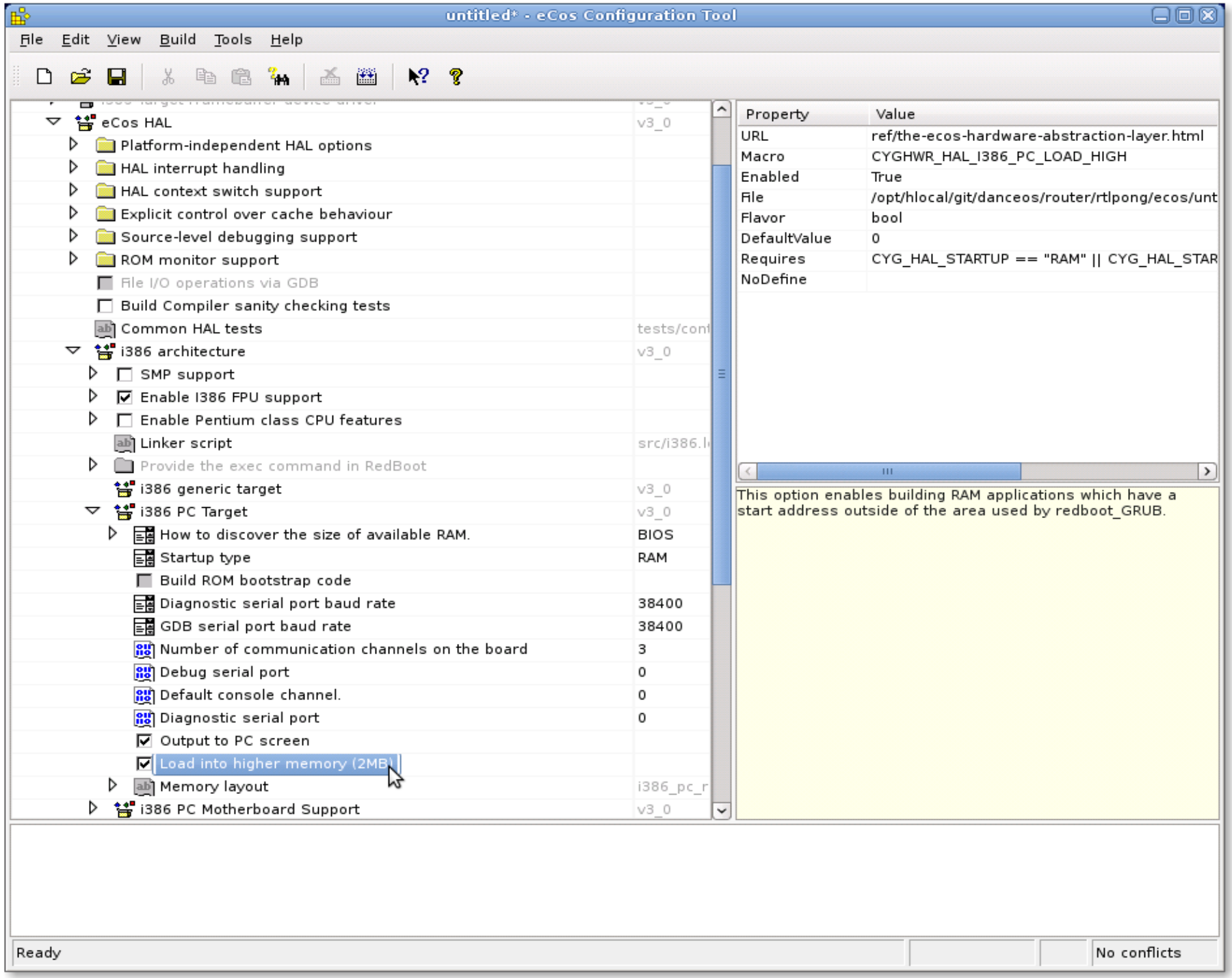

untitled - eCos Configuration Tool
File
Edit
View
Build
Tools
Help
Library F7
Tests Shift+F7
Clean
Stop
Generate Build Tree
Options...
Repository...
Templates...
Packages...
Configura
Redboo
Global
Redboo
FRV400
NS DP8
Fujitsu
AMD AM29XXXXX FLASH memory support
eCos HAL
I/O sub-system
Infrastructure
eCos kernel
Dynamic memory allocation
ISO C and POSIX infrastructure
ISO C library
Math library
Wallclock device
Common error code support
v3_0
Property
Value
URL
ref/ecos-ref.html
The root node for all configurable items
Ready
No conflicts

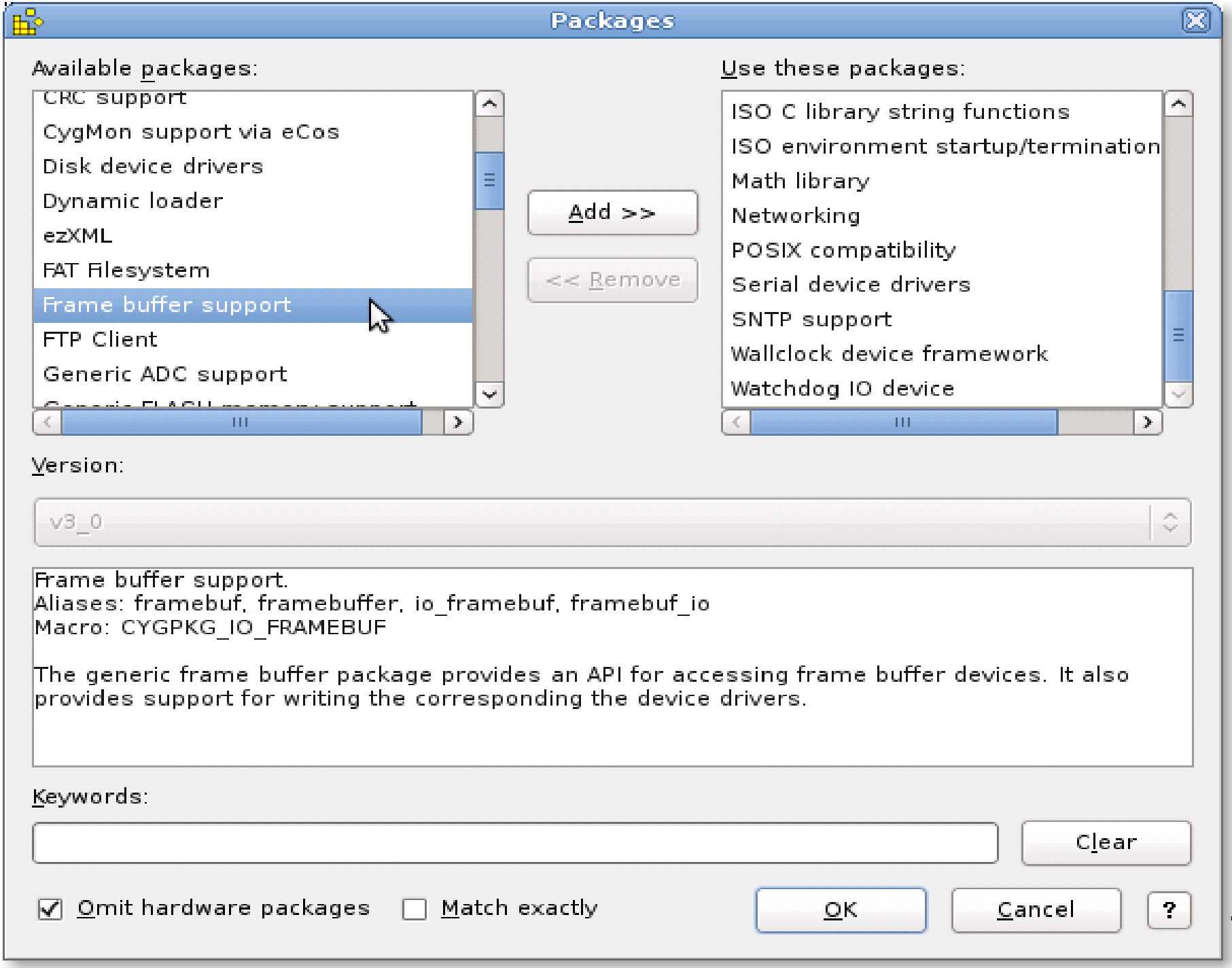
Konfiguration



Konfiguration




untitled* - eCos Configuration Tool
File Edit View Build Tools Help
eCos HAL v3_0
Platform-independent HAL options
HAL interrupt handling
HAL context switch support
Explicit control over cache behaviour
Source-level debugging support
ROM monitor support
File I/O operations via GDB
Build Compiler sanity checking tests
Common HAL tests tests/conf
i386 architecture v3_0
SMP support
Enable I386 FPU support
Enable Pentium class CPU features
Linker script src/i386.l
Provide the exec command in RedBoot
i386 generic target v3_0
i386 PC Target v3_0
How to discover the size of available RAM. BIOS
Startup type RAM
Build ROM bootstrap code
Diagnostic serial port baud rate 38400
GDB serial port baud rate 38400
Number of communication channels on the board 3
Debug serial port 0
Default console channel. 0
Diagnostic serial port 0
Output to PC screen
Load into higher memory (2MB)
Memory layout i386_pc_r
i386 PC Motherboard Support v3_0
Property Value
URL ref/the-ecos-hardware-abstraction-layer.html
Macro CYGHWR_HAL_I386_PC_LOAD_HIGH
Enabled True
File /opt/hlocal/git/danceos/router/rtlpong/ecos/unt
Flavor bool
DefaultValue 0
Requires CYG_HAL_STARTUP == "RAM" || CYG_HAL_STAR
NoDefine
This option enables building RAM applications which have a start address outside of the area used by redboot_GRUB.
Ready
No conflicts


Packages
Available packages:
CRC support
CygMon support via eCos
Disk device drivers
Dynamic loader
ezXML
FAT Filesystem
Frame buffer support
FTP Client
Generic ADC support
Add >>
<< Remove
Use these packages:
ISO C library string functions
ISO environment startup/termination
Math library
Networking
POSIX compatibility
Serial device drivers
SNTP support
Wallclock device framework
Watchdog IO device
Version:
v3_0
Frame buffer support.
Aliases: framebuf, framebuffer, io_framebuf, framebuf_io
Macro: CYGPKG_IO_FRAMEBUF
The generic frame buffer package provides an API for accessing frame buffer devices. It also provides support for writing the corresponding the device drivers.
Keywords:
Clear
Omit hardware packages
Match exactly
OK
Cancel
?

- eCos Installation: /proj/i4ezs/P_HotSys/...
- Quellbaum: ~/pong
  ~/pong/ecos/ecos.ecc
  ~/pong/CMakeLists.txt

Per „ cmake . “ werden automatisch Makefiles erzeugt

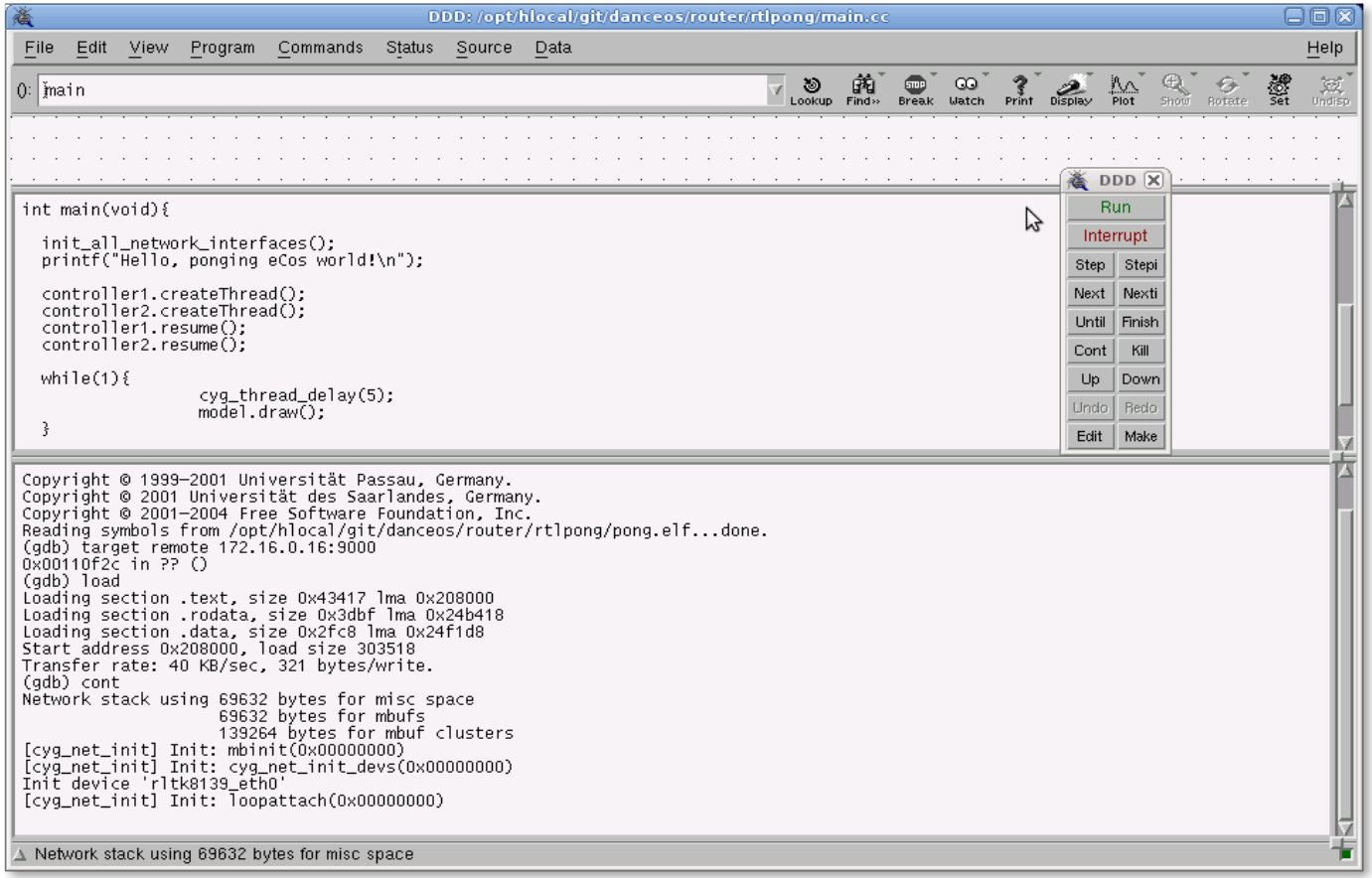
und sogar Projektdateien für Eclipse oder CodeBlocks:

cmake –G“Eclipse CDT4 – Unix Makefiles“ –D CMAKE_BUILD_TYPE=Debug .

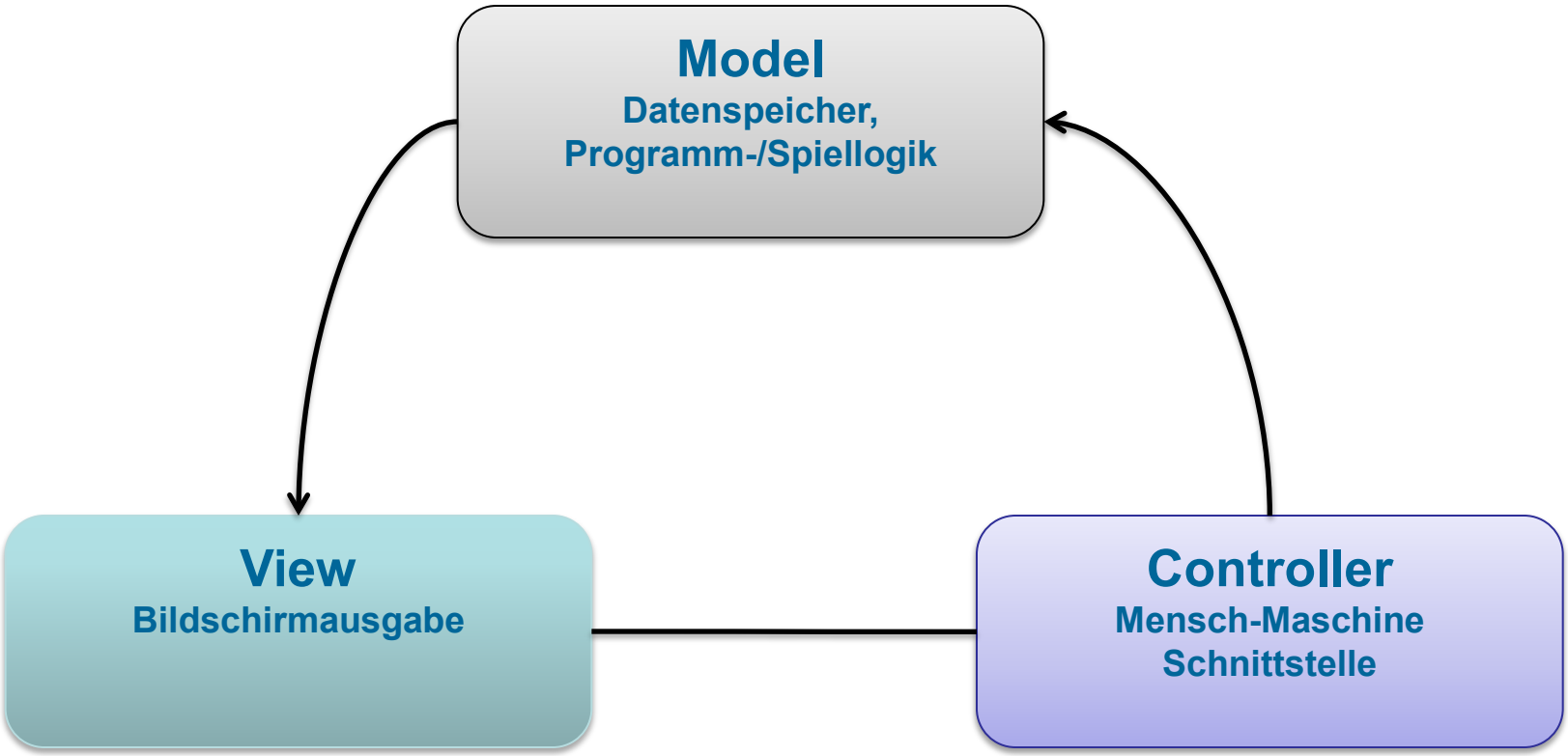
cmake –G“CodeBlocks – Unix Makefiles“ –D CMAKE_BUILD_TYPE=Debug .

Anwendungsentwicklung

- Hochladen und Debugging per {ddd,gdb} pong.elf
- target remote {redboot-ip}:9000
- load
- b main
- cont



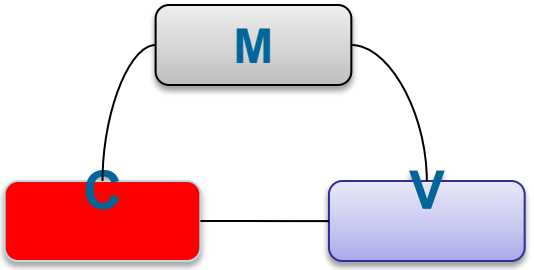
Entwurfsmuster MVC



MVC - Controller

- Verwaltet Schnittstelle zum Spieler (Tastatur, TCP, RS232, ...)
- Nimmt Tastendrücke entgegen
- Gibt „Kommando“ an das Modell weiter
  - → „Hey Modell! Linker Schläger einen Schritt runter!“
  - → z.B. Model.leftRacketDown()...

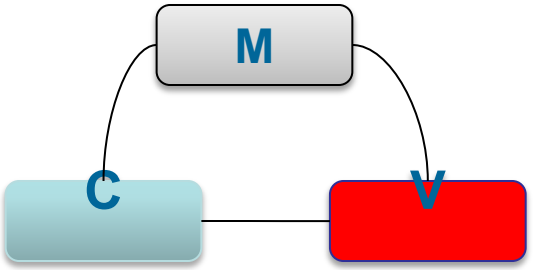
- Kümmert sich nicht um weitere Berechnungen



MVC - View

- Zeichnet Spielfeld, Schläger, Ball, etc.
- Erhält Position und Formen aus dem Modell
- Hilfsklasse „Viewable“
  - Abstrahiert eCos Framebuffer API
  - Legacy C → C++
  - drawHLine(), drawVLine(), fillBlock()
  - Viewable::screen_width, Viewable::screen_size

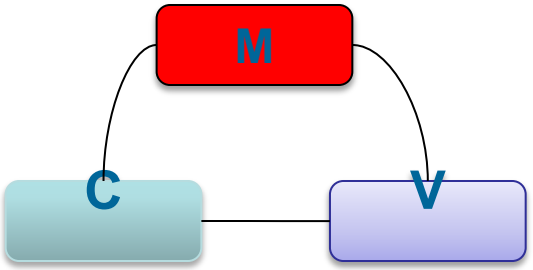
- Wann und wer aktiviert Neuzeichnen?

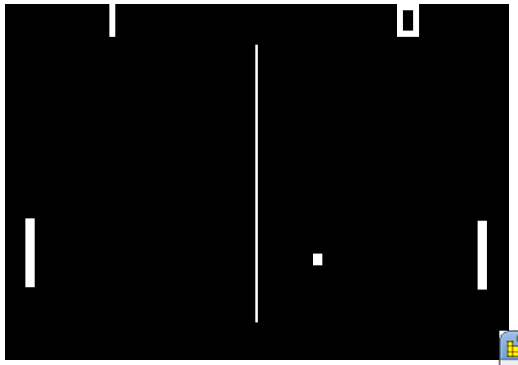


MVC - Model

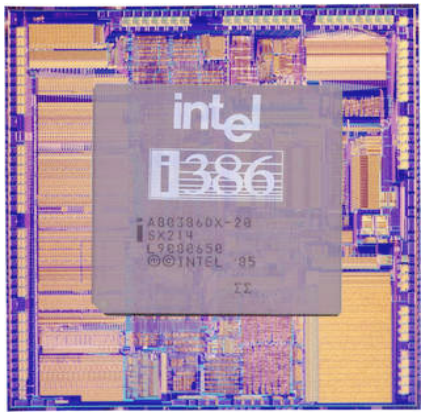
- Speichert Zustand der einzelnen Objekte
  - Ballposition, Ballrichtung, Ballgeschwindigket
  - Schlägerposition, Schlägergeschwindigkeit
  - Spielstand
- Implementiert Spielelogik, Physik

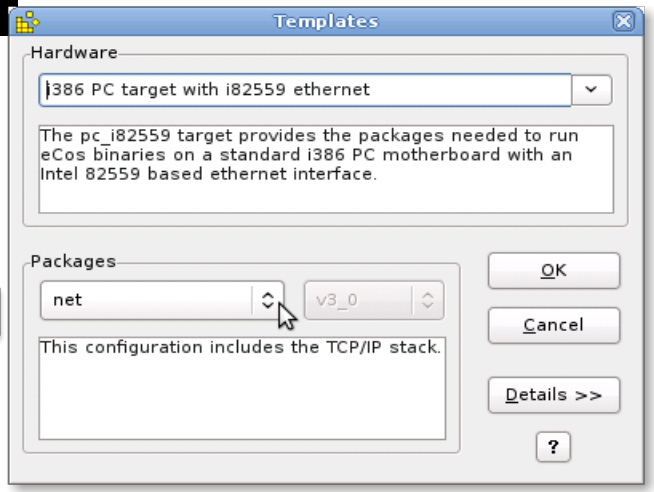
- Wann berechnet man mögliche Kollisionen?
- Wie erkennt man Kollisionen des Balls...
  - ...an den Wänden?
  - ...am Schläger?
- Wie realisiert man ein Abprallen des Balles?

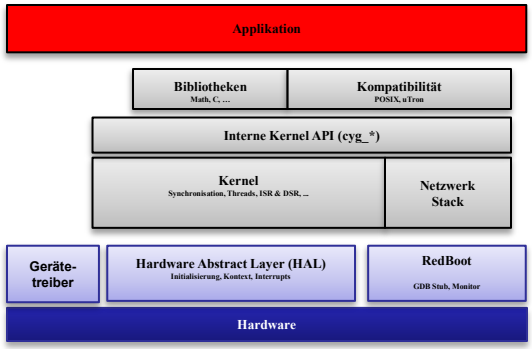


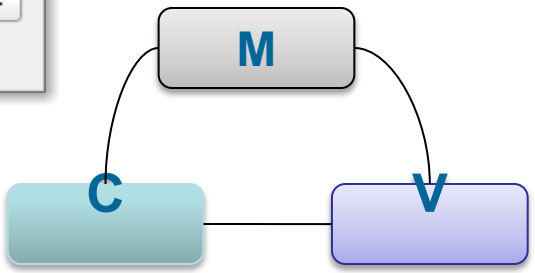


Viel Spaß!


intel
i386


Templates
Hardware
i386 PC target with i82559 ethernet
The pc_i82559 target provides the packages needed to run eCos binaries on a standard i386 PC motherboard with an Intel 82559 based ethernet interface.
Packages
net
v3_0
This configuration includes the TCP/IP stack.
OK
Cancel
Details >>
?


Applikation
Bibliotheken
Math, C, ...
Kompatibilität
POSIX, uTron
Interne Kernel API (cyg_*)
Kernel
Synchronisation, Threads, ISR & DSR, ...
Netzwerk Stack
Geräte-treiber
Hardware Abstract Layer (HAL)
Initialisierung, Kontext, Interrupts
RedBoot
GDB Stub, Monitor
Hardware


M
C
V