www.vscom.de

OpenRISC
User Manual
Edition: April 2010

Vision Systems GmbH
Tel: +49 40 528 401 0
Fax: +49 40 528 401 99
Web: www.visionsystems.de
Support: faq.visionsystems.de

The software described in this manual is furnished under a license agreement and may be used only in accordance with the terms of that agreement.

Copyright Notice

Trademarks

Disclaimer

Vision Systems reserves the right to make changes and improvements to its product without providing notice.

Vision Systems provides this document “as is,” without warranty of any kind, either expressed or implied, including, but not limited to, its particular purpose. Vision Systems reserves the right to make improvements and/or changes to this manual, or to the products and/or the programs described in this manual, at any time.

Information provided in this manual is intended to be accurate and reliable. However, Vision Systems assumes no responsibility for its use, or for any infringements on the rights of third parties that may result from its use.

This product might include unintentional technical or typographical errors. Changes are periodically made to the information herein to correct such errors, and these changes are incorporated into new editions of the publication.

Contents

List of Figures

List of Tables

1. Introduction

The OpenRISC is an ARM9-based RISC industrial embedded computer. The great variety of interfaces like LAN, CF, microSD, USB, CAN[1], I²C, serial interface and digital I/O makes it easy to connect various industrial devices to the OpenRISC.

Compact dimensions and DIN Rail mounting capability make the OpenRISC to a space saving and flexible mounting industrial computer. It is feasible to be installed even in space limited environments.

Due to RISC based architecture the OpenRISC has very small power consumption (6,5 Watt), so fanless heat dissipation is possible. Working in an extended temperature range from -10°C up to 65°C the OpenRISC can be used under harsh industrial conditions. Therefore the OpenRISC is downright designed for industrial automation.

The embedded computer runs full-featured Debian GNU/Linux on ARM operating system. With Debian's repository database it is easy to install and update the free software on the OpenRISC. The OpenRISC is capable to act directly as software development host, Web, Mail, Print and Database server or as desktop computer with X11 window manager and many more.

See the list of features below:

- ARM9 32-bit RISC CPU, 166MHz
- 64MB SDRAM on board
- 4MB Flash Disk on board
- RS232/RS422/RS485 serial ports
- CAN port
- Independent digital I/O channels
- 1 x CF-Slot in True IDE mode (accepts MicroDrives)
- 1 x microSD-Slot
- 2 x USB 2.0 as Host
- MiniPCI-slot for expansion with WLAN, GPS etc.
- 2 x Ethernet interfaces for redundant networking or routing functions
- I²C bus with max. 330kHz clock
- RTC
- Ready-to-Run Debian Linux for ARM operating system
- DIN-Rail and wall-mount installation
- Robust, fanless design
- Wide temperature range -10 to 65°C
- Buzzer, Watchdog Timer

[1] OpenRISC Alena only

OpenRISC Model	**Alekto**	**Alekto LAN**	**Alena**
CPU	ARM9 32-bit RISC CPU, 166MHz	ARM9 32-bit RISC CPU, 166MHz	ARM9 32-bit RISC CPU, 166MHz
RAM	64MB SDRAM	64MB SDRAM	64MB SDRAM
Flash Memory on Board	4MB	4MB	4MB
Serial Interfaces	2 x RS232/RS422/RS485	1 x RS232/RS422/RS485	2 x RS232/RS422/RS485 2 x RS232 only
CAN Interface on board	-	-	1 x
Digital I/O channels	8 x I/O channels	8 x I/O channels	4 x I/O channels 2 x Relays 2 x optically isolated inputs
CF-Slot (True IDE mode)	1 x (also accepts MicroDrives)	1 x (also accepts MicroDrives)	1 x (also accepts MicroDrives)
microSD-Slot	1 x internal slot	1 x internal slot	1 x internal slot
USB	2 x USB 2.0 as Host	2 x USB 2.0 as Host	2 x USB 2.0 as Host
Expansion Slot	MiniPCI-slot	MiniPCI-slot	MiniPCI-slot
Ethernet	2 x	5 x (WAN and 4 x LAN switch)	2 x
I^{2}C bus	1 x	1 x	1 x
RTC	1 x	1 x	1 x
Watchdog Timer	1 x	1 x	1 x
WLAN On/Off Button	-	1 x	-

Table 1: OpenRISC Products Comparison

2. Getting Started

2.1. Connect to OpenRISC via Serial Link

Connect the OpenRISC to the serial port of your PC and start a terminal software (HyperTerminal, ZOC[2], minicom etc) with 115200,8,n,1 settings (no hardware/software handshake is needed. Set the terminal type according to Section 2.2). Insert a CF/microSD-card with one of the preinstalled systems (refer to Section 4). Power your OpenRISC according to Section 3.1. You'll see Linux booting. After the boot procedure you'll be asked to log in. Two users are already added to the system: a super user (root) and an ordinary user (user). For the super user enter:

```
Debian login:  root
Password:  linux
```

For the ordinary user enter:

```
Debian login:  user
Password:  user
```

2.2. Terminal Type

Terminal type is defined in the environment variable `TERM` and is set to `TERM=linux` by default. The terminal type of your terminal application (HyperTerminal, ZOC, minicom etc.) should be set to the same type to interact correctly with the OpenRISC console. If `linux` terminal type is not available in your software, `vt100` can be used instead. To do this add following line to the `~/.bashrc`:

```
export TERM=vt100
```

[2] www.emtec.com

2.3. Configure Network

Now you can configure network interfaces by editing `/etc/network/interfaces`. The IP addresses for `eth0`, `eth1` and `wlan0`[3] are statically assigned by default(see the Listing below).

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
        address 192.168.254.254
        netmask 255.255.255.0

# The secondary network interface
auto eth1
iface eth1 inet static
        address 192.168.253.254
        netmask 255.255.255.0

# The wireless interface
#auto wlan0
iface wlan0 inet static
        address 192.168.127.254
        netmask 255.255.255.0
        wpa-driver wext
        wpa-conf /etc/wpa_supplicant.conf
        post-up echo BLUE > /proc/vsopenrisc/leds
        post-down echo blue > /proc/vsopenrisc/leds
```

Listing 1: /etc/network/interfaces

`post-up` and `post-down` directives switch blue LED on and off signaling that wireless interface is up or down. The wireless interface `wlan0` will be configured with the `wpa_supplicant` utility (see Section 5.1).

Setup gateway and DNS server Assuming your router has IP address 192.168.254.1 the OpenRISC will be configured in the following way:

1. change `eth0` section of `/etc/network/interfaces` file
   ```
   auto eth0
   iface eth0 inet static
   address 192.168.254.254
   netmask 255.255.255.0
   gateway 192.168.254.1
   ```
2. insert following line to the `/etc/resolv.conf`[4]
   ```
   nameserver 192.168.254.1
   ```

[3] to activate wlan0 uncomment the `#auto wlan0` line in `/etc/networking/interfaces`

[4] see `man resolv.conf` for explanations

3. execute `/etc/init.d/networking restart`

2.4. Start Programming

Connect to the OpenRISC either via serial link or network and login as `user`. For the introduction some examples were prepared and placed under `/home/user/examples`. This folder contains following files:

- `ioctls.c` - LEDs, buzzer, reset push button and digital IO usage examples
- `ioctls2.c` - UART and network usage examples
- `rawsrv.c` - raw server application that transfers data from network to serial interface and vice versa
- `wdtimer.c` - Watchdog Timer example
- `vsopenrisc.h` - OpenRISC hardware API header file
- `Makefile` - the makefile to produce examples. Following targets can be created:
  - `all` - creates `ioctls`, `ioctls2` and `rawsrv` executables
  - `doc` - creates doxygen documentation in `doc` folder
  - `clean` - deletes executables
  - `distclean` - executes clean and in addition removes `doc` folder
- `vsopenrisc.doxyfile` - doxygen configuration file

Execute `make` and you'll get three above mentioned executables (see Figure 1).

You can start with an `ioctls` example that reads and changes the LEDs (Power LED, WLAN LED etc.), reads the reset push button status and digital IO registers:

```
./ioctls
```

After that you'll see your LEDs blinking and explaining outputs on your terminal. For further information about software development for the OpenRISC refer to Section 6.

```
user@debian:~/examples$ make
gcc -O0 -g3 -Wall ioctls.c -o ioctls
gcc -O0 -g3 -Wall ioctls2.c -o ioctls2
gcc -O0 -g3 -Wall rawsrv.c -o rawsrv
gcc -O0 -g3 -Wall wdtimer.c -o wdtimer
user@debian:~/examples$ ls -l
total 350
-rw-r--r-- 1 user user 349 Aug 27 14:32 CMakeLists.txt
-rw-r--r-- 1 user user 392 Aug 27 14:57 Makefile
-rwxr-xr-x 1 user user 68577 Aug 27 17:12 ioctls
-rw-r--r-- 1 user user 8865 Aug 27 14:54 ioctls.c
-rwxr-xr-x 1 user user 8568 Aug 27 17:12 ioctls.strip
-rwxr-xr-x 1 user user 91036 Aug 27 17:12 ioctls2
-rw-r--r-- 1 user user 3097 Aug 14 14:01 ioctls2.c
-rwxr-xr-x 1 user user 5352 Aug 27 17:12 ioctls2.strip
-rwxr-xr-x 1 user user 93286 Aug 27 17:12 rawsrv
-rw-r--r-- 1 user user 5682 Aug 14 14:01 rawsrv.c
-rwxr-xr-x 1 user user 8720 Aug 27 17:12 rawsrv.strip
-rw-r--r-- 1 user user 50619 Aug 14 14:01 vsopenrisc.doxyfile
-rw-r--r-- 1 user user 2243 Aug 27 14:50 vsopenrisc.h
-rwxr-xr-x 1 user user 61988 Jul 24 05:44 wdtimer
-rw-r--r-- 1 user user 1042 Aug 14 2008 wdtimer.c
-rwxr-xr-x 1 user user 3868 Jul 24 05:44 wdtimer.strip
user@debian:~/examples$
```

Figure 1: Compilation example

3. Hardware Configuration

3.1. Power Supply

The OpenRISC device is powered by a single 9-30V power supply. A suitable power supply adapter is part of the packaging. Connect the cable to the power jack at the right side of OpenRISC, and put the adapter into the socket. The Power LED on OpenRISC (red) will light. You can connect a power supply of your choice, providing the technical requirements are met.

3.2. Network

The OpenRISC can use all three network interfaces (Ethernet and WLAN) independently. The default IP addresses are:

- 192.168.254.254 for eth0 (first network interface)
- 192.168.253.254 for eth1 (second network interface)
- 192.168.127.254 for wlan0 (wireless network interface)

3.2.1. WLAN Antenna

The connector used for the WLAN Antenna is known as SMA-Reverse. This is a standard type to allow for simple connection of different equipment. Just fit the supplied antenna by carefully screwing it to the connector. You are free to connect a cable and a different antenna of your choice, as long as it is designed for WLAN.

3.2.2. WLAN Configuration

WLAN supports 802.11b/g standard with 54Mbit/s. When the interface goes up, the blue LED will light, when the interface goes down, the blue LED goes dark.

3.2.3. Ethernet

The connectors for Ethernet are the usual RJ45. Simply connect it to your switch or hub. When the connect is done the Link LED on RJ45 (yellow) will light. When data traffic occurs on the network, this LED will blink. It depends on your network whether a 100Mbit or a 10Mbit connect will be established. A 100Mbit net causes the Speed LED on RJ45 (green) to light, otherwise it will remain dark.

OpenRISC Alekto LAN provides quad LAN switch connected to LAN 2 interface instead of single LAN interface.

3.3. MicroDrive

To use MicroDrive its APM functions must be disabled with `hdparm -B 255 /dev/hda`.

3.4. RS422/485 Electrical Configuration

In typical RS422 and RS485 installations certain electric conditions have to be configured. Simply connecting cables is not enough to fulfill the specifications or RS422 and RS485.

For ease of installations the OpenRISC provides these functions for often used parameters. They are activated by placing certain jumpers (see Table 2), internal of the OpenRISC. There is one block of jumpers (see Figure 2) for each serial port (JP5 for Port 1 and JP6 for Port 2). Place a connection cap to activate the function.

Warning: All jumpers are unconnected by default. This is important for use in RS232 mode. Never close any jumper, otherwise communication errors or damage of devices is possible.

Pins	Function of Signals
1-2	Place 120Ω to terminate Tx+/- (Data+/- in RS485 2-wire)
3-4 5-6	Add BIASing function to Tx+/- (mostly required for RS485 2-wire modes)
7-8	Place 120Ω to terminate Rx+/-
9-10	Add BIASing function to Rx+/-
11-12	Add BIASing function to Rx+/-

Table 2: RS422/485 Jumper Configuration

3.4.1. Termination Resistors

The use of long communication lines in RS422 and RS485 mode require the installation of termination resistors. These must match the impedance of the cable. Typical cables in Twisted-Pair configuration have an impedance around 120Ω. In RS422 this resistor has to be placed at the far end from the sender, in RS485 the typical configuration requires one resistor at each end of the cable.

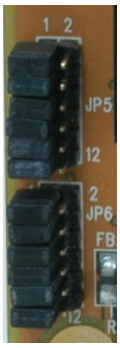


Figure 2: Terminal Resistors

3.4.2. BIAS Function

RS485 requires a BIAS option for the communication lines. This will guarantee stable electrical levels on the cables, even at times when no station is transmitting data. Without BIAS there will be noise on the cable, and sometimes receivers can not detect the first characters of a beginning communication.

3.5. Internal Flash Write Protection

JP1 (see Figure 3) on the CPU board is responsible for protecting first flash partition where RedBoot is installed. When JP1 is closed the RedBoot partition is protected, when opened the partition is writable. Always keep the JP1 closed unless you want to change this partition.

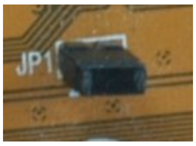


Figure 3: WP Jumper

4. Software Configuration

The OpenRISC comes with a preinstalled Debian GNU/Linux on ARM[5] operating system. The complete system image (see Section 4.7) provides necessary tools and services to start with application development, various services such as web and mail server, Samba server, etc. For office tasks it provides an X-Server with graphical desktop manager and some office software. Table 3 shows free and used space for complete image:

Image	Used	Free
Complete	955MB	797MB

Table 3: 2GB System Image

This image is supplied on the DVD "Software" and can be copied to the CF/microSD card via `vsimgtool` utility (see Appendix C.1) under MS Windows or via `dd` (see Appendix C.2).

4.1. Booting from CF/microSD

There are two files placed under `/boot` directory on the complete image:

- `zImage` - kernel to boot
- `kparam` - kernel parameter

The OpenRISC BIOS (refer to Section 9) is searching for these files to boot the system from an external medium (Mass storage devices, network etc.). Due to the fact that USB devices need more time to be detected it is recommended to increase the Start-Timeout value to at least 5 seconds (refer to 9.5). The kernel parameter will be described below:

root= parameter tells kernel from where to take the root filesystem. If this parameter is not supplied BIOS[6] defines it using the device name containing zImage. So if root filesystem is on the same device/partition as the kernel it is recommended not to define `root=` parameter.

Example:

Specifying `root=/dev/hda1` will mount `/dev/hda1` as root filesystem

Specifying `root=/dev/sda1` will mount `/dev/sda1` as root filesystem

If CF card is inserted into the internal CF-Slot and kparam doesn't supply `root=` parameter, `/dev/hda1` will be mounted as root filesystem

rootwait parameter tells kernel to wait till root device appears. This is important for booting from USB devices, because they will be detected much later than CF. This parameter is integrated into BIOS since version 2.0.

mem= parameter defines the amount of RAM that kernel can use. This parameter is integrated into BIOS since version 2.1, but it can be overridden in the `kparam`.

[5] `http://www.debian.org/ports/arm/`

[6] since BIOS version 2.1

4.2. Swapping and Logging

Due to the fact that the flash memory has a finite number of erase-write cycles it is very important to reduce them. Many applications use logging for information, recovery and debugging purposes, this can lead to frequent flash usage. There are several possibilities to avoid this:

1. use external HDD attached via USB and redirect swapping and logging to it
2. disable swapping[7] (remove swap entry in the `/etc/fstab`) and logging where it is possible
3. redirect the log stream via network

To redirect logged messages that need syslog edit /etc/syslog.conf and change all file destinations to network destinations (see the example). Change

```
*.=info;*.=notice;*.=warn;\
auth,authpriv.none;\
cron,daemon.none;\
mail,news.none -/var/log/messages
```

to

```
*.=info;*.=notice;*.=warn;\
auth,authpriv.none;\
cron,daemon.none;\
mail,news.none @192.168.254.84
```

for further information refer to the `syslog.conf` manpage. To receive the log messages under Linux you can use your existing syslog utility, for Windows you can use Kiwi Syslog Daemon[8] or any other Syslog daemon.

4.3. Activating and Deactivating Services

Some services will be started as daemons at system startup and hence reduce the amount of free memory and increase the boot time. You can use one of the following options to deactivate the unused services at startup or start them only on demand:

- rename links under `/etc/rc2.d`[9] (see Table 4).
- use `sysvconfig` or `sysv-rc-conf`[10] as graphical console programs for the first method
- start the service over `inetd`[11]
- start manually via `/etc/init.d`

[7] To list swapping devices execute `cat /proc/swaps`

[8] www.kiwisyslog.com

[9] For detailed information about filenames see `/etc/rc2.d/README` file

[10] http://sysv-rc-conf.sourceforge.net/

[11] see `man inetd`

Disabled name	Enabled name
K09apache2	S91apache2
K77ntp	S23ntp
K80courier-authdaemon	S20courier-authdaemon
K80courier-mta	S20courier-mta
K80courier-pop	S20courier-pop
K80samba	S20samba
K80sqwebmail	S20sqwebmail
K84openvpn	S16openvpn

Table 4: Enable/disable services

4.4. udev

Since kernel 2.6.26.5 Complete Image uses udev[12] to create devices in `/dev` directory. The network interfaces are not bound to their MACs in order to be able to use only one system image on various devices[13]. To change this behavior in Debian 4.0 Etch uncomment following line in `/etc/udev/rules.d/z45_persistent-net-generator.rules`:

```
#KERNEL=="eth*|ath*|wlan*|ra*|sta*", DRIVERS=="?*",\
# IMPORT{program}="write_net_rules $attr{address}"
```

To change this behavior in Debian 5.0 Lenny change following line in `/etc/udev/rules.d/75-persistent-net-generator.rules`:

```
KERNEL!="eth*|ath*|wlan*[0-9]|msh*|ra*|sta*|ctc*|lcs*|hsi*",
\ GOTO="persistent_net_generator_end"
```

to

```
KERNEL!="ath*|msh*|ra*|sta*|ctc*|lcs*|hsi*",
\ GOTO="persistent_net_generator_end"
```

`/etc/udev/rules.d/custom.rules` has rules to create symlinks to I2C and RTC devices:

```
KERNEL=="rtc0", SYMLINK+="rtc"
KERNEL=="i2c-0", SYMLINK+="i2c"
```

4.5. Time Zone

The default time zone configured on the system image is Europe/London (GMT+0). BIOS operates with UTC time. So in summer time Debian's clock has one hour difference compared with the time shown in BIOS. It is normal behavior. Use `dpkg-reconfigure tzdata` to change the time zone in Debian according to your geographical position.

[12] http://en.wikipedia.org/wiki/Udev

[13] All relevant settings were removed from `/etc/udev/rules.d/z25_persistent-net.rules` for Debain 4.0 Etch and from `/etc/udev/rules.d/70-persistent-net.rules` for Debian 5.0 Lenny

4.6. Create Swap File

Swap file can be created instead of creating a swap partition. To create it execute following steps:

1. create file with `count` being equal to the desired block size: `dd if=/dev/zero of=/var/swapfile bs=1024 count=131072`
2. `mkswap /var/swapfile`
3. add entry to the `/etc/fstab`: `/var/swapfile none swap sw 0 0`
4. reboot and check if `/var/swapfile` is used by looking at `/proc/swaps`

```
debian:~# cat /proc/swaps
Filename Type Size Used Priority
/var/swapfile file 131064 0 -1
```

4.7. Complete System Image

The complete system image contains lots of programs and libraries. It contains a development environment consisting of the gcc tool chain and vim-tiny text editor. Besides development tasks this image is designed to use the OpenRISC as a server and/or desktop system to accomplish such tasks as mail server, web server, resource sharing and office tasks. For the latter WindowMaker is installed to provide graphical desktop on the OpenRISC.

4.7.1. Program Overview

The complete image provides among others the following utilities:

- Software Development
  - gcc
  - vim-tiny
- Network
  - ssh (server and client)
  - telnet (server and client)
  - vsftpd (server and client)
  - netcat
  - socat
  - Samba client and server
  - Apache2 web server
  - Courier mail server
  - NTP client and server
  - sredird RFC2217 server

- Desktop:
  - WindowMaker desktop
  - xdm server
  - AbiWord
  - dillo or netsurf web browser

4.7.2. GCC

GCC[14] development is a part of the GNU Project, aiming to improve the compiler used in the GNU system including the GNU/Linux variant. The GCC development effort uses an open development environment and supports many other platforms in order to foster a world-class optimizing compiler, to attract a larger team of developers, to ensure that GCC and the GNU system work on multiple architectures and diverse environments, and to more thoroughly test and extend the features of GCC.

4.7.3. Netcat

Netcat[15] is a featured networking utility which reads and writes data across network connections, using the TCP/IP protocol. It is designed to be a reliable "back-end" tool that can be used directly or easily driven by other programs and scripts. At the same time, it is a feature-rich network debugging and exploration tool, since it can create almost any kind of connection you would need and has several interesting built-in capabilities.

4.7.4. Socat

socat[16] is a relay for bidirectional data transfer between two independent data channels. Each of these data channels may be a file, pipe, device (serial line etc. or a pseudo terminal), a socket (UNIX, IP4, IP6 - raw, UDP, TCP), an SSL socket, proxy CONNECT connection, a file descriptor (stdin etc.), the GNU line editor (readline), a program, or a combination of two of these. These modes include generation of "listening" sockets, named pipes, and pseudo terminals.

4.7.5. Samba

Samba[17] is an Open Source/Free Software suite that has, since 1992, provided file and print services to all manner of SMB/CIFS clients, including the numerous versions of Microsoft Windows operating systems. Samba is freely available under the GNU General Public License.
To share an extra directory for users create this directory:

```
mkdir /samba
```

Edit `/etc/samba/smb.conf`:

```
[global]
```

[14] http://gcc.gnu.org/gccmission.html
[15] http://netcat.sourceforge.net/
[16] http://www.dest-unreach.org/socat/
[17] http://us3.samba.org/samba/

```
workgroup = debian
netbios name = debianserver
server string = %h server (Samba %v)
log file = /var/log/samba/log.%m
max log size = 1000
syslog = 0
[SAMBA]
path=/samba
browseable=yes
writeable=yes
valid users = user
admin users = debian
```

Now you need to restart the samba to take the new changes effect:

```
/etc/init.d/samba restart
```

4.7.6. Web Server (Apache2.2)

The Apache HTTP Server Project[18] is an effort to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows NT. The goal of this project is to provide a secure, efficient and extensible server that provides HTTP services in sync with the current HTTP standards. Apache has been the most popular web server on the Internet since April 1996.

In addition to the Apache2 packages the openssl package is installed to enable a SSL certificate creation. Apache2 is already equipped with SSL certificate and is ready to accept https connections needed for example to configure mail server via webadmin tool (see 4.7.7) . To create your own certificate or to install the official certificate take a look at the following paragraph.

How to Configure SSL? To enable https connections, a SSL certificate must be created and then registered by the Apache2 web server:

1. `mkdir /etc/apache2/ssl/`
2. `openssl req $@ -new -x509 -days 365 -nodes -out /etc/apache2/ssl/apache.pem -keyout /etc/apache2/apache.pem`
3. `chmod 600 /etc/apache2/ssl/apache.pem`
4. `cp /etc/apache2/sites-available/default /etc/apache2/sites-available/ssl`
5. `ln -s /etc/apache2/sites-available/ssl /etc/apache2/sites-enabled/ssl`
6. ssl file must be changed:
   ```
   NameVirtualHost *:443
   <VirtualHost *:443>
   SSLEngine On
   SSLCertificateFile /etc/apache2/ssl/apache.pem
   SSLCertificateKeyFile /etc/apache2/apache.pem
   ...
   </VirtualHost>
   ```

[18] http://httpd.apache.org/

7. to activate port 443 for https-Queries edit `/etc/apache2/ports.conf`. Add
   `Listen 443`
8. activate SSL module:
   `a2enmod ssl`
9. restart Apache:
   `apache2ctl restart`

4.7.7. Mail Server (courier)

The Courier mail transfer agent (MTA)[19] is an integrated mail/groupware server based on open commodity protocols, such as ESMTP, IMAP, POP3, LDAP, SSL and HTTP. Courier provides ESMTP, IMAP, POP3, webmail and mailing list services within a single, consistent, framework. Individual components can be enabled or disabled at will. Courier now implements basic web-based calendaring and scheduling services integrated in the webmail module. Advanced groupware calendaring services will follow soon.
The mail server can be conveniently configured via web interface using courierwebadmin tool. It can be started with the following URL
https://*mail-server-address*/cgi-bin/courierwebadmin.
Execute webadmin-tool, setup server name and local domains under *Mail server name and local domains* menu. Then create a user:

`adduser --ingroup users testuser`
login as `testuser`
`cd /home/testuser`
`maildirmake Maildir` (this creates user's maildir folder structure)

Execute webmail-tool
https://*mail-server-address*/cgi-bin/sqwebmail
and log on. From now on one can send and receive e-mails.

4.7.8. NTP

NTP[20] is a protocol designed to synchronize the clocks of computers over a network to a common timebase (usually UTC).

Configuration Client For client configuration edit `/etc/ntp.conf` as follows:

```
### client:/etc/ntp.conf ###############
driftfile /var/lib/ntp/ntp.drift
# NTP-server in LAN
server 192.168.1.1
# Grant access from other NTP-server
restrict 192.168.1.1
# Grant access from localhost (ntpq -p)
restrict 127.0.0.1
# deny access for other peers
```

[19] http://www.courier-mta.org
[20] http://www.ntp.org

```
restrict default notrust nomodify nopeer
###############################################
```

Then restart ntpd.

Configuration Server For server configuration edit `/etc/ntp.conf` as follows:

```
### server:/etc/ntp.conf ########################
driftfile /var/lib/ntp/ntp.drift
# NTP-server
server ptbtime1.ptb.de
server ptbtime2.ptb.de
# Grant access from other NTP-server
restrict ptbtime1.ptb.de
restrict ptbtime2.ptb.de
# Grant access from localhost (ntpq -p)
restrict 127.0.0.1
# grant access for local network
restrict 192.168.1.0 mask 255.255.255.0
# deny access for other peers
restrict default notrust nomodify nopeer
#################################################
```

Then restart ntpd.

4.7.9. sredird

Sredird[21] is a serial port redirector that is compliant with the RFC 2217[22] "Telnet Com Port Control Option" protocol. This protocol lets you share a serial port through the network.

Configuration `sredird` runs only under `inetd`. To configure `inetd` two files must be edited:

- the following line must be appended to `/etc/inetd.conf` to share `/dev/ttyS1`:
  `sredir stream tcp nowait root /usr/sbin/tcpd /usr/sbin/sredird -i 5 /dev/ttyS1 /var/lock/LCK..ttyS1`
- sredir service must be defined in `/etc/services`:
  `sredir 7051`[23]`/tcp`

Four serial ports (`/dev/ttyS1` - `/dev/ttyS4`) have been already configured in `/etc/inetd.conf` and `/etc/services` (ports 7051 - 7054).

Following clients could be used with sredird:

- Serial/IP COM Port Redirector (http://www.tacticalsoftware.com/products/serialip.htm) for Windows-Clients
- cyclades-serial-client (http://freshmeat.net/projects/cyclades-serial-client/) for Linux-Clients

[21] http://freshmeat.net/projects/sredird
[22] http://www.rfc-editor.org/rfc/rfc2217.txt
[23] port 7051 is an example port number

4.7.10. WindowMaker

Window Maker[24] is an X11 window manager originally designed to provide integration support for the GNUstep Desktop Environment. In every way possible, it reproduces the elegant look and feel of the NEXTSTEP[tm] user interface. It is fast, feature rich, easy to configure, and easy to use. It is also free software, with contributions being made by programmers from around the world. Window Maker includes compatibility options which allow it to work with other popular desktop environments, namely GNOME and KDE, and comes with a powerful GUI configuration editor, called WPrefs, which removes the need to edit text-based config files by hand

4.7.11. Connecting to X-Window

The xdm manager[25] lets you connect to the X-Server. Following configuration issues should be considered to enable a remote access to the OpenRISC:

- `#* #any host can get a login window`
  in `/etc/X11/xdm/Xaccess` should be uncommented
- `DisplayManager.requestPort:  0`
  in `/etc/X11/xdm/xdm-config` must be replaced with
  `!  DisplayManager.requestPort:  0`

Xming[26] can be used to connect to the X-Server from MS Windows. After installation, execute XLaunch and configure the proper IP address. The configuration can also be saved. The saved XML-file can look as follows:

```
<?xml version="1.0"?>
<XLaunch xmlns="http://www.straightrunning.com/XmingNotes"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.straightrunning.com/XmingNotes XLaunch.xsd"
WindowMode="Windowed"
ClientMode="XDMCP"
XDMCPHost="192.168.1.66"
Display="0"
Clipboard="true"/>
```

4.7.12. AbiWord

AbiWord[27] is a free word processing program, similar to Microsoft® Word. It is suitable for a wide variety of word processing tasks.

[24] http://www.windowmaker.info
[25] http://en.wikipedia.org/wiki/X_display_manager
[26] http://sourceforge.net/projects/xming
[27] http://www.abisource.com

5. Network Services and Tools Provided by OpenRISC

The OpenRISC can be accessed via Ethernet for remote usage and file sharing. For this purpose there are several services such as telnet, ssh and ftp installed and preconfigured. For WLAN configuration wpa_supplicant and wireless-tools are included in the distribution.

5.1. WLAN Configuration

Wpa_supplicant[28] is a WPA Supplicant for Linux, BSD, Mac OS X, and Windows with support for WEP, WPA and WPA2 (IEEE 802.11i / RSN). Supplicant is the IEEE 802.1X/WPA component that is used in the client stations. It implements key negotiation with a WPA Authenticator and it controls the roaming and IEEE 802.11 authentication/association of the WLAN driver.

5.1.1. Managed Wireless Network (Wpa_supplicant)

Wpa_supplicant uses /etc/wpa_supplicant.conf file for its configuration (see the Listing below).

```
ap_scan=1

# no encryption
network={
  ssid="TEST"
  key_mgmt=NONE
}
# WEP encryption
network={
  ssid="TESTWEP"
  key_mgmt=NONE
  wep_key0=xxxxxxxxxxxxxxxxxxxxx
  wep_tx_keyidx=0
  auth_alg=SHARED
}
# WPA/WPA2 encryption
network={
  ssid="TESTWPA2"
  proto=WPA RSN
  key_mgmt=WPA-PSK
  pairwise=CCMP TKIP
  group=CCMP TKIP
  psk="xxxxxxxxxxxxxx"
}
```

Listing 2: /etc/wpa_supplicant.conf

WLAN interface is automatically configured on system's startup (see Section 2.3). To test the configuration just run:

```
/etc/init.d/networking restart
```

Wpa_supplicant can also be called manually to investigate configuration problems:

[28] http://hostap.epitest.fi/wpa_supplicant/

```
wpa_supplicant -iwlan0 -c/etc/wpa_supplicant.conf -dd -Dwext
```

Further information about configuring the WLAN interface can be taken from wpa_supplicant's manual page (`man wpa_supplicant`) .

5.1.2. Ad-hoc Wireless Network (wireless-tools)

To connect to another node with SSID "Node1" not using encryption execute following commands:

```
ifconfig wlan0 down
iwconfig wlan0 mode ad-hoc key off essid "Node1"
ifconfig wlan0 up
```

For WEP encrypted connection:

```
ifconfig wlan0 down
iwconfig wlan0 mode ad-hoc key restricted key [1] "some WEP key" essid "Node1"
ifconfig wlan0 up
```

5.1.3. WLAN On/Off Button (Alekto LAN only)

OpenRISC Alekto LAN provides a special button to enable/disable the WLAN adapter. Pressing the button will result in button LED going off and WLAN transmit power turning off, so no communication via WLAN is possible. Pressing the button for the second time will result in button LED going on and WLAN transmit power returning to the previous value, so WLAN data transfer is again possible. This functionality will be managed by `wland` service running on start up. The source code for `wland` service is available via our svn repository at http://svn.visionsystems.de/.

5.2. Bluetooth Support

The Bluetooth support is already integrated in the kernel. To use a USB Bluetooth adapter you'll need to install some additional software:

```
apt-get install bluetooth
```

To connect to the Bluetooth network access point (NAP) execute:

1. `hcitool scan`
   you'll get the list of available Bluetooth devices with their addresses

2. select the needed one and connect to it
   `pand -c <bdaddr>`

3. check `/var/log/daemon.log` to verify that the connection is established or execute
   `ifconfig -a`
   if you can see `bnep0` interface the connection is definitely established

4. set up the `bnep0` interface. For example:
   `ifconfig bnep0 192.168.10.1`

5. assuming the NAP has an address 192.168.10.2 try to ping it:
   `ping 192.168.10.2`

To set up the NAP execute:

1. edit `/etc/bluetooth/hcid.conf` and set the security option to auto:
   ```
   # Security Manager mode
   # none - Security manager disabled
   # auto - Use local PIN for incoming connections
   # user - Always ask user for a PIN
   #
   security auto;
   ```
2. execute `/etc/init.d/bluetooth restart` to activate the new configuration
3. `hciconfig hci0 piscan lm master`
   to enable authentication (required by MS Windows) execute:
   `hciconfig hci0 piscan auth lm master`
   the default passkey configured in `/etc/bluetooth/hcid.conf` is “1234”
4. `pand --listen --role NAP`
5. after the client has connected, the `bnep0` must be also set up

To connect to the OpenRISC from MS Windows, use the on-board tools or the software supplied with the Bluetooth device. For the on-board tools:

1. select “Join a Personal Area Network”
2. search for the OpenRISC
3. select it and try to connect. Enter “1234” as passkey
4. configure the network interface

5.3. Telnet

The telnet console can be accessed with following command:

```
telnet ip address
```

This connection can be used only for normal users, not for the super user. To execute super user commands login as user and then execute `su`. For further information see:

`man telnetd` or `man issue.net`

5.4. SSH

To access the OpenRISC via SSH from Linux execute:

```
ssh ip address
```

To access OpenRISC via SSH from Windows you need a ssh-client such as PuTTY[29]. To exchange files several tools could be used:

- scp (Linux) - secure copy tool

[29] http://www.chiark.greenend.org.uk/~sgtatham/putty/

- pscp (Windows) - secure copy tool included in PuTTY distribution
- WinSCP[30] (Windows) - secure copy tool with graphical interface

For further information see:

```
man sshd
```

5.5. FTP

OpenRISC provides vsftpd[31] as FTP-server. The files to be shared must be placed in `/home/ftp`. To access the files execute:

```
ftp ip address
```

The FTP-server is configured for normal user accounts created on the OpenRISC, so login as user "user" with a password "user". For access via graphical FTP-client following programs can be used:

- FileZilla (http://www.filezilla.de/)
- Firefox with FireFTP add-on (http://fireftp.mozdev.org/help.html)

For further information see:

```
man vsftpd
```

5.6. Other Possible Services

Other services such as Samba (see Section 4.7.5) or NFS could also be installed to share resources.

5.6.1. Samba

To install Samba execute:

```
apt-get install samba
```

To access your home directory from Windows, you must register the user in Samba:

```
smbpasswd -a user
```

For further information see:

```
man samba
```

[30] http://winscp.net/eng/index.php

[31] http://vsftpd.beasts.org/

5.6.2. NFS

To install the NFS-server execute:

```
apt-get install nfs-kernel-server
```

For further information see:

```
man nfsd
```

6. Software Development

6.1. Environment

6.1.1. Compile your software directly on the OpenRISC

You can start programming directly on the OpenRISC. The toolchain is already installed and GCC compiler will be invoked in the same way it is done on a desktop Linux. To modify files you can use `vi` or some other editor.

6.1.2. Cross-compile your software on the PC

For more convenience you can also use a PC with Linux running on it. This can be either a directly installed Linux or Linux running in a virtual machine[32] for Windows users. For the compilation of your own applications outside of the OpenRISC you'll need a cross-compiler[33] for the ARM platform. You'll find one on the DVD "Software". Depending on your Debian version you'll need the appropriate version:

1. `toolchain_gcc.4.2.2_libc.2.3.6.tar.bz2` for Debian 4.0 Etch
2. `arm-linux-gcc-4.3.2-gnueabi.tar.bz2` for Debian 5.0 Lenny[34]

Please decompress one of these archives into the `/opt` directory of your PC. To install the toolchain for Debian 4.0 execute:

```
su
mount /dev/cdrom /mnt
cd /opt
tar -xvjf /mnt/development/toolchain_gcc.4.2.2_libc.2.3.6.tar.bz2
```

Add `/opt/arm-linux-gcc-4.2.2/bin` to your `PATH` environment variable:

```
export PATH=/opt/arm-linux-gcc-4.2.2/bin:$PATH
```

Following utilities prepent with `arm-linux-` will be available after extracting the files from the archive (refer to Table 12).

Warning: please note that your software compiled for Debian 4.0 Etch can't be executed on Debian 5.0 Lenny without recompilation due to new ABI.

[32] You can use free VM Software like VMWare Player www.vmware.com or VirtualBox www.virtualbox.org

[33] See http://en.wikipedia.org/wiki/Cross-compile for explanation

[34] EABI (http://en.wikipedia.org/wiki/EABI) interface is mandatory for new Debian versions. Debian 5.0 is the last version where you can choose between old ABI and EABI. The next version will be based only on the new interface.

6.1.3. Integrated Development Environment

For an Integrated Development Environment (IDE) following programs can be used:

- Eclipse (www.eclipse.org). See Appendix D for installation and configuration notes
- Vim[35] (www.vim.org)
- many more

For ease of creating Makefiles the CMake[36] utility can be used. For that purpose the `CMakeLists.txt` file was added to the examples directory. After installing CMake on your OpenRISC directly or on your development host execute:

```
cd /home/user/examples
cmake .
```

After that the Makefile is created. This Makefile has the same targets as the original one.

6.2. Linux Kernel

The OpenRISC uses 2.6 series Linux kernel. The source code for one of thess kernels can be obtained both from DVD "Software" and from Subversion[37] repository at http://svn.visionsystems.de/. For the kernel archive on DVD execute:

```
su
mount /dev/cdrom /mnt
exit
cd /home/user/projects
tar -xvjf /mnt/development/linux-2.6.26-ks8695.tar.bz2
```

To check out the repository to your development host execute:

```
svn co svn://svn.visionsystems.de/linux-2.6.26-ks8695/trunk linux-2.6.26-ks8695/trunk
```

After extracting the kernel sources from the archive or checking out the repository execute

```
cd linux-2.6.26-ks8695/trunk
```

to get to the kernels source tree, where the main `Makefile` is placed.

To configure the kernel via text based graphical user interface Ncurses[38] library must be installed in the development version with all needed headers. After that execute:

```
make menuconfig
```

you'll see the following menu as shown in Figure 4. Change kernel configuration according to your needs and save the configuration. To compile the kernel execute:

```
make
```
[39]

[35] A good tutorial to start programming with Vim `http://heather.cs.ucdavis.edu/~matloff/ProgEdit/ProgEdit.html`

[36] www.cmake.org

[37] http://subversion.tigris.org/

[38] http://www.gnu.org/software/ncurses/ncurses.html

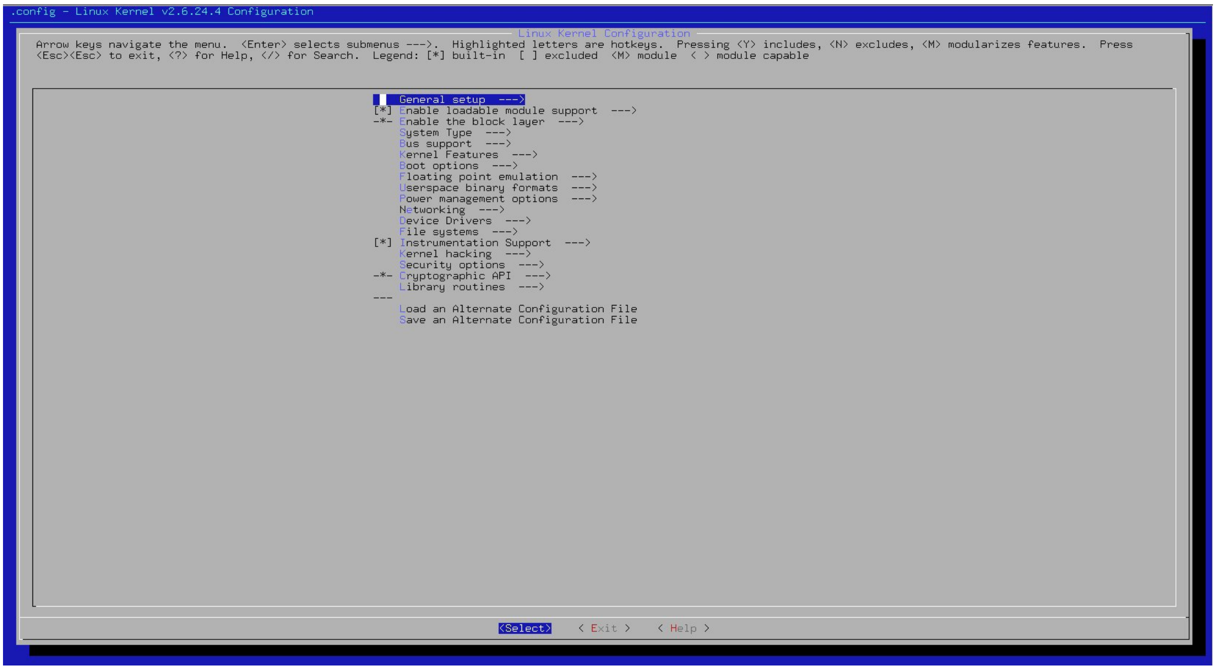
[39] if your development host has more than one CPU core, you can use `make -j` to utilize all of you cores, that will accelerate the compilation process

The compiled kernel image `zImage` will be placed under

`arch/arm/boot/zImage`

To start the OpenRISC with new kernel do the following steps:

1. enter BIOS
2. enter “System Console”
3. configure IP address according to your network (for example 192.168.254.254)
4. execute `nc -l -p 5000 > /var/zImage` this will start Netcat listening on the port 5000
5. on your development host execute `netcat 192.168.254.254 5000 < arch/arm/boot/zImage`
6. after the data transmission mount your CF and copy new kernel
7. `mount /dev/hda1 /mnt`
8. `cp /var/zImage /mnt/boot`
9. `umount /mnt`
10. reboot the OpenRISC



```
.config - Linux Kernel v2.6.24.4 Configuration
                                    Linux Kernel Configuration
 Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes features.  Press
 <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

                                  General setup  --->
                              [*] Enable loadable module support  --->
                              -*- Enable the block layer  --->
                                  System Type  --->
                                  Bus support  --->
                                  Kernel Features  --->
                                  Boot options  --->
                                  Floating point emulation  --->
                                  Userspace binary formats  --->
                                  Power management options  --->
                                  Networking  --->
                                  Device Drivers  --->
                                  File systems  --->
                              [*] Instrumentation Support  --->
                                  Kernel hacking  --->
                                  Security options  --->
                              -*- Cryptographic API  --->
                                  Library routines  --->
                              ---
                                  Load an Alternate Configuration File
                                  Save an Alternate Configuration File

                              <Select>    < Exit >    < Help >
```

Figure 4: Kernel Configuration Menu

6.3. Programming Examples Repository

Some programming examples were prepared to show/test the abilities of OpenRISC hardware. The latest version of this software can be obtained from our repository by executing the following command:

```
svn co svn://svn.visionsystems.de/OpenRISC_Software/trunk OpenRISC_Software/trunk
```

This repository contains three folders:

- examples - some basic programming examples as described in Section 2.4
- hwtest - Hardware Test Utility enables the testing of OpenRISC peripherals (see Section B)
- installation - this folder provides files needed for Debian installation as described in Section 10

6.4. Setup Shared Source Directory

To avoid permanent copying the compiled files, you can share the examples folder on your PC via Samba. Assuming your PC has the IP address 192.168.254.253 and a valid user like “user” from group “user” registered in the Samba-server do:

- On the PC side
  - `cd /home/user`
  - `mkdir openrisc_examples`
  - copy all files from `examples` folder to `openrisc_examples`
  - edit as `su /etc/samba/smb.conf`. Add the `[openrisc]` share:
    ```
    [openrisc]
    comment=OpenRISC programming examples
    path=/home/user/openrisc_examples
    browsable=yes
    writable=yes
    valid users=user
    ```
  - `/etc/init.d/samba restart`
- On the OpenRISC
  - login as `user`
  - `mkdir openrisc_examples`
  - `su`
  - `mount -t cifs -o username=user //192.168.254.253/openrisc /home/user/openrisc_examples`
  - enter password
  - `exit`
  - `cd openrisc_examples`

Now you can edit and compile programs on the PC and execute them on the OpenRISC.

6.5. Debugging

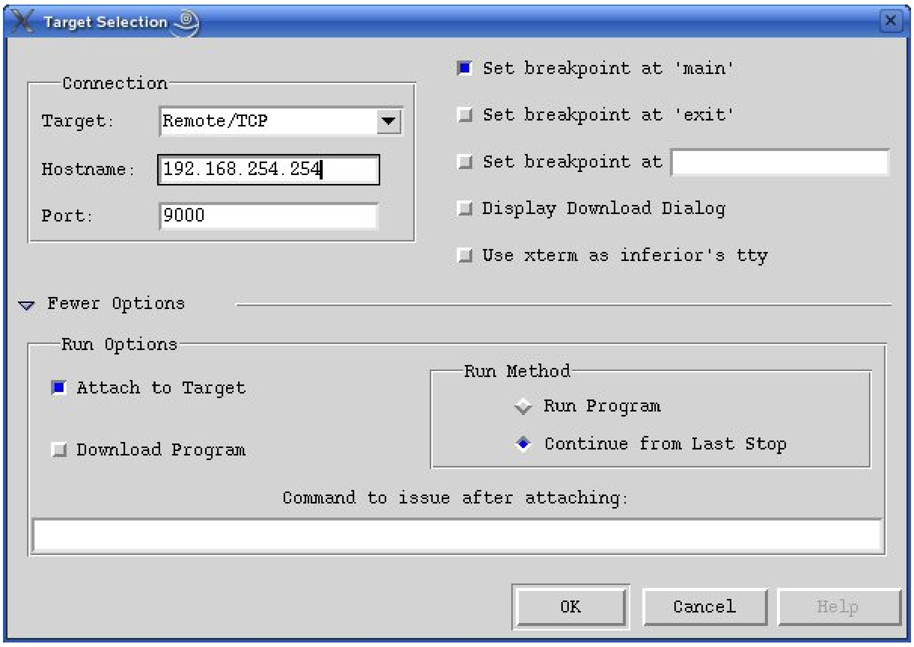
You have the possibility to debug your own applications with the `gdbserver`[40] on the target. To debug your application, start the server with the following command:

```
gdbserver :9000 /home/user/examples/ioctls
```

Make a connection to the server with the Insight[41] debugger:

```
arm-linux-insight ioctls
```

Go to the menu `Target Settings` (see Figure 5) and enter your destination data. Than you can get connected to the target with `Run\Connect to Target`. The rest of the debugging is up to you.



Connection

Target: Remote/TCP

Hostname: 192.168.254.254

Port: 9000

☑ Set breakpoint at 'main'

☐ Set breakpoint at 'exit'

☐ Set breakpoint at

☐ Display Download Dialog

☐ Use xterm as inferior's tty

Fewer Options

Run Options

☑ Attach to Target

☐ Download Program

Run Method

○ Run Program

● Continue from Last Stop

Command to issue after attaching:

Figure 5: Insight: target selection

To get Insight download the source files and compile the code using the instructions given by the Insight developers in their FAQ (http://sourceware.org/insight/faq.php#q-2.2). In the case of OpenRISC take the following steps:

```
host> ls
src/
host> mkdir insight-arm; cd insight-arm
host> ../src/configure --target=arm-linux --program-prefix=arm-linux-
host> make
host> make install
```

You can also use Eclipse to debug your application (see Appendix D.2) .

[40] Please make sure to install `gdb` package: `apt-get install gdb`

[41] http://sourceware.org/insight

7. OpenRISC Hardware API

Such hardware as digital I/O, buzzer, serial interface will be controlled via IOCTL commands. These commands are defined in the `vsopenrisc.h` header file. Almost all IOCTL commands are reflected in the `/proc`-filesystem (see Section 8).

To control the hardware the `/dev/gpio` device must be opened:

```
int fd;

fd = open("/dev/gpio", O_RDWR);
if (fd < 0)
  exit(-1);
```

Listing 3: Open `/dev/gpio`

Some examples for the OpenRISC hardware are provided in the `examples/ioctls.c` and `examples/ioctls2.c`. For the usage in real applications see the source code of the Hardware Test Utility (`hwtest` provided on SVN repository).

7.1. Digital I/O

The OpenRISC provides up to 8 digital input/output channels, 2 optically isolated inputs and 2 relays[42]. The data direction for each I/O channel (not for optically isolated inputs or relays) can be independently set to input (bit value 0) or output (bit value 1). An interrupt for an input channel can also be independently enabled (bit value 1 enables interrupt, bit value 0 disables interrupt for the desired channel). The physical driver operates with 32mA for both high and low level. For digital I/O usage four registers are provided:

- data register - reflected in `/proc/vsopenrisc/gpio_data`
- data direction register - reflected in `/proc/vsopenrisc/gpio_ctrl`
- interrupt mask register - reflected in `/proc/vsopenrisc/gpio_irqmask`
- status register (read only) - reflected in `/proc/vsopenrisc/gpio_change`

In addition the interrupts will be counted for each pin.

Following IOCTL commands are defined in `vsopenrisc.h` to control digital I/O:

- `GPIO_CMD_GET/GPIO_CMD_SET` - set/get data register value
- `GPIO_CMD_GET_CTRL/GPIO_CMD_SET_CTRL` - set/get data direction register value
- `GPIO_CMD_GET_IRQMASK/GPIO_CMD_SET_IRQMASK` - set/get interrupt mask register value
- `GPIO_CMD_GET_CHANGE` - get status register value
- `GPIO_CMD_GET_CHANGES` - get interrupt count (reflected in `/proc/vsopenrisc/gpio_changes`)

[42] OpenRISC Alekto provides 8 digital I/O channels only, OpenRISC Alena provides 4 digital I/O channels, 2 optically isolated inputs and 2 relays

The digital I/O driver also supports `select()` and `poll()` calls. To use this functionality the interrupt must be enabled for the desired pins.

To change the register values special structure and macros will be used (see the Listing below).

```
#define GPIO_BIT_0 0x01 // DIO0 or optically isolated input
#define GPIO_BIT_1 0x02 // DIO1 or optically isolated input
#define GPIO_BIT_2 0x04 // DIO2
#define GPIO_BIT_3 0x08 // DIO3
#define GPIO_BIT_4 0x10 // DIO4
#define GPIO_BIT_5 0x20 // DIO5
#define GPIO_BIT_6 0x40 // DIO6 or relay
#define GPIO_BIT_7 0x80 // DIO7 or relay

struct gpio_struct
{
  unsigned long mask;  // bits to modify
  unsigned long value; // value to set
};
```

Listing 4: GPIO struct

The following Listing shows how to change pins DIO0 and DIO2 to output and then read the control register back. To change the register direction **`struct gpio_struct`** is used because you need to specify what bits are going to changed and to what values (Lines 5-6). To read the register you only need to give **`unsigned long`** variable to the IOCTL command (Line 9).

```
 1 ...
 2 struct gpio_struct gpio_val;
 3 unsigned long val;
 4
 5 gpio_val.mask = (GPIO_BIT_0 | GPIO_BIT_2);
 6 gpio_val.value = (GPIO_BIT_0 | GPIO_BIT_2);
 7
 8 ioctl(fd, GPIO_CMD_SET_CTRL, &gpio_val);
 9 ioctl(fd, GPIO_CMD_GET_CTRL, &val);
10 ...
```

Listing 5: GPIO usage Example

7.2. Buzzer

The OpenRISC provides a buzzer for acoustic signaling. You can manipulate it via the IOCTL-calls or via the Proc-Filesystem. Following IOCTL commands are defined in the vsopenrisc.h to control the buzzer:

- `GPIO_CMD_GET_BUZZER/GPIO_CMD_SET_BUZZER` - turn on/off the buzzer
  (reflected in `/proc/vsopenrisc/buzzer`)
- `GPIO_CMD_GET_BUZZER_FRQ/GPIO_CMD_SET_BUZZER_FRQ` - get/set the modulated signal delay. So that the buzzer will be turned on for the delay specified and the it will be turned off for the same delay and so on till this mode will be turned off
  (reflected in `/proc/vsopenrisc/buzzer_frq`)

7.3. LEDs

The OpenRISC provides three configurable LEDs:

- red LED (power LED)
- blue LED (can be used to signal WLAN link)
- green LED (user LED)
- WLAN button LED (OpenRISC Alekto LAN only)

Following IOCTL commands are defined in the vsopenrisc.h to control the LEDs:

- `GPIO_CMD_GET_LEDS/GPIO_CMD_SET_LEDS` - get/set LED combination (reflected in `/proc/vsopenrisc/leds`)
- `GPIO_CMD_SET_LED_POWER` - turn on/off the power LED
- `GPIO_CMD_SET_LED_BLUE` - turn on/off the blue LED
- `GPIO_CMD_SET_LED_GREEN` - turn on/off the green LED
- `GPIO_CMD_SET_LED_BTN_WLAN` - turn on/off the WLAN button LED (OpenRISC Alekto LAN only)

7.4. Serial Interfaces

RS232/422/485 mode switching The serial ports can operate in one of the three modes RS232, RS422 or RS485[43] (see Section 3.4 for electrical configuration issues). These modes will be controlled through the EPLD circuit. Following commands are be used to switch the modes:

- `TIOCGEPLD` - get EPLD mode
- `TIOCSEPLD` - set EPLD mode

These commands use the following structure to store the desired parameters:

```
struct epld_struct
{
  unsigned long port;
  unsigned long reg_shift;
  unsigned long value;
};
```

The field `value` can have one of the following values:

- `EPLD_RS232` - RS232 mode
- `EPLD_RS422` - RS422 mode
- `EPLD_RS485_ART_4W` - RS485 mode 4-wire transmit control by ART
- `EPLD_RS485_ART_2W` - RS485 mode 2-wire direction control by ART
- `EPLD_RS485_ART_ECHO` - RS485 mode 2-wire direction control by ART with echo

[43] See `include/asm/arch/serial.h` to see what EPLD capabilities each port has

- EPLD_RS485_RTS_4W - RS485 mode 4-wire transmit control by RTS
- EPLD_RS485_RTS_2W - RS485 mode 2-wire direction control by RTS
- EPLD_RS485_RTS_ECHO - RS485 mode 2-wire direction control by RTS with echo
- EPLD_CAN - CAN mode
- EPLD_PORTOFF - shutdown the port

RS485 transmission control In RS485 the line driver for transmitting must be disabled (tri-stated) when the device does not send data. In a 2-wire configuration this is known as data direction change, with 4-wire it is called line contention. The following modes are provided by OpenRISC:

- RTS: if this mode is set user software is responsible to turn transmitter on/off by toggling the RTS signal (RTS 1 turns sender on, RTS 0 turns sender off)
- ART (Automatic Receive Transmit): The sender will be automatically turned on/off by hardware. The turning off occurs after the stop bit was sent (recommended mode).

RS485 receive control There are two modes to handle own transmitted messages in 2-wire mode:

- with echo (EPLD_RS485_RTS_ECHO, EPLD_RS485_ART_ECHO): both outgoing and incoming messages will be received by application
- without echo (EPLD_RS485_ART_2W, EPLD_RS485_RTS_2W): application receives only incoming messages

Baud rate generation interface The OpenRISC provides full support for the 16C950 UART baud rate generation. This allows the user to use the serial interfaces with arbitrary speeds up to 3,6Mbit/s. To use this capability the current baud rate must be set to B38400 and the custom divisor to the negative value of the desired baud rate. In the source code below the baud rate will be set to 500000bit/s, so the custom divisor was set to -500000 (see the lines 26-39) and the current baud rate to 38400 (see the lines 41-66).

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <sys/types.h>
 4 #include <sys/time.h>
 5 #include <sys/ioctl.h>
 6 #include <sys/stat.h>
 7 #include <unistd.h>
 8 #include <termios.h>
 9 #include <fcntl.h>
10 #include <linux/serial.h>
11 #include <string.h>
12
13 int main (int argc, char **argv)
14 {
15   int fd, ret;
16   struct termios ser_termios;
17   struct serial_struct ss_st;
18
19   fd = open("/dev/ttyS1", (O_RDWR | O_NOCTTY));
20   if(fd < 0)
21   {
    perror("open");
    return -1;
  }

  // set custom divisor to -500000 to achieve 500000bit/s
  if (ioctl(fd, TIOCGSERIAL, &ss_st)<0)
  {
    perror("TIOCGSERIAL");
    return -1;
  }
  ss_st.custom_divisor = -500000;
  ss_st.flags |= ASYNC_SPD_CUST;

  if (ioctl(fd, TIOCSSERIAL, &ss_st)<0)
  {
    perror("TIOCSSERIAL");
    return -1;
  }

  // set baud rate to 38400bit/s to activate custom divisor
  ret = tcgetattr(fd, &ser_termios);
  if(ret <0 )
  {
    perror("getattr");
    return -1;
  }
...
  ret = cfsetispeed(&ser_termios, B38400);
  if (ret < 0)
  {
    perror("ispeed");
   return -1;
  }
  ret = cfsetospeed(&ser_termios, B38400);
  if (ret < 0)
  {
    perror("ospeed");
    return -1;
  }
  ret = tcsetattr(fd, TCSANOW, &ser_termios);
  if (ret < 0)
  {
    perror("getattr");
    return -1;
  }
...
  close(fd);
  return 0;
}
```

Every time the baud rate will be changed a special baud rate generation function will be invoked in the serial driver (`drivers/char/serial.c`) namely `serial16C950_get_divisors()`. This routine calculates appropriate values for the clock prescaler register (CPR), times clock register (TCR) and 16-bit divisor supported by 16C950 UART. If the routine couldn't find the appropriate values the baud rate 9600bit/s will be set. To disable this functionality comment the following definition in `drivers/char/serial.c`:

```
#define ENABLE_16C950_BAUD_GENERATION_FEATURES
```

7.5. CAN

The OpenRISC Alena has an integrated CAN interface based on the SJA1000 chip. CAN bus can be accessed via several APIs:

- VSCAN API
- LinCAN driver

It is possible to use CANopen[44] library from CanFestival[45] with both APIs.

The CAN interface and the fourth serial interface are sharing the same connector. When opening the CAN device the EPLD will be automatically switched to CAN mode. After closing the device the former mode will be restored.

VSCAN API For VSCAN API see the VSCAN_Manual.pdf.

LinCAN driver To write application using LinCAN[46] driver you'll need two header files: `can.h` and `canmsg.h`. The `"/dev/can0"` device must be opened in order to communicate with the driver. In addition to the standard IOCTL commands `GET_CAN_STATUS` command was added to get drivers state (see Listing 6).

There are two LEDs to indicate CAN transfer status: Data and Error. Data LED indicates incoming and outgoing data transfer, Error indicates one of the error states described in Listing 6. If bus error occurs (Listing 6 Line 8) the Error LED lights permanently, if any other error occurs (Listing 6 Lines 1-7) Error LED will be flashing. Bus error has the highest priority, i.e. if it has occurred no matter in what state CAN was or what error happens as next the Error LED will light permanently. Error LED will be turned off only when closing the port or reading the status via `GET_CAN_STATUS` IOCTL.

```
1 #define CAN_IOCTL_FLAG_RX_FIFO_FULL   (1<<0)
2 #define CAN_IOCTL_FLAG_TX_FIFO_FULL   (1<<1)
3 #define CAN_IOCTL_FLAG_ERR_WARNING    (1<<2)
4 #define CAN_IOCTL_FLAG_DATA_OVERRUN   (1<<3)
5 #define CAN_IOCTL_FLAG_UNUSED         (1<<4)
6 #define CAN_IOCTL_FLAG_ERR_PASSIVE    (1<<5)
7 #define CAN_IOCTL_FLAG_ARBIT_LOST     (1<<6)
8 #define CAN_IOCTL_FLAG_BUS_ERROR      (1<<7)
```

Listing 6: CAN Status Register Values

See `hwtest/module_can.c` for usage example.

[44] www.can-cia.org

[45] www.canfestival.org

[46] http://sourceforge.net/projects/ocera/

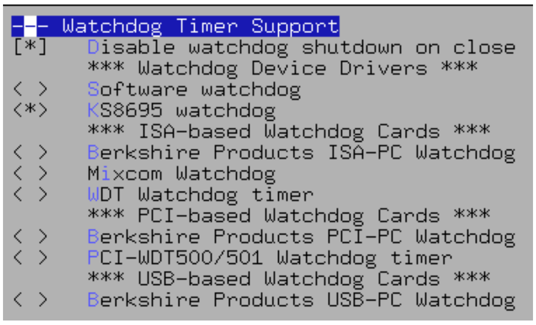
7.6. Watchdog Timer

The OpenRISC provides a watchdog timer[47] (WDT) that can work in intervals from 1 second to maximal 171 seconds. The access to the WDT occurs via `/dev/watchdog` device. After starting the WDT it should be turned off or reloaded after the critical part otherwise the system will reboot. Table 5 shows the most important IOCTLs to control WDT.

IOCTL	Description
`WDIOC_SETTIMEOUT`	set timeout and start the timer
`WDIOC_GETTIMEOUT`	get timeout
`WDIOC_SETOPTIONS`	enable (`WDIOS_ENABLECARD`) disable (`WDIOS_DISABLECARD`) the timer
`WDIOC_KEEPALIVE`	reload the timer

Table 5: Watchdog Timer IOCTLs

The WDT driver provides a special option called "Disable watchdog shutdown on close" to prevent stopping the timer on WDT descriptor close (see Figure 6). This is a compile-time option. It is deactivated by default, so you have to build your own kernel if you need it.



```
--- Watchdog Timer Support
[*]   Disable watchdog shutdown on close
      *** Watchdog Device Drivers ***
< >   Software watchdog
<*>   KS8695 watchdog
      *** ISA-based Watchdog Cards ***
< >   Berkshire Products ISA-PC Watchdog
< >   Mixcom Watchdog
< >   WDT Watchdog timer
      *** PCI-based Watchdog Cards ***
< >   Berkshire Products PCI-PC Watchdog
< >   PCI-WDT500/501 Watchdog timer
      *** USB-based Watchdog Cards ***
< >   Berkshire Products USB-PC Watchdog
```

Figure 6: Watchdog Timer Support

See `examples/wdtimer.c` for WDT usage example.

7.7. I^{2}C

I^{2}C[48] (Inter-Integrated Circuit) is a multi-master serial computer bus. In the OpenRISC integrated I^{2}C controller is already supported by the mainline kernel. A good starting point on how to use I^{2}C is the `hwtest` utility (see Section B.6). `test_LCD()` routine from `hwtest/module_i2c.c` shows how to communicate with an I^{2}C device. Additional information can be found in the Linux kernel documentation `Documentation/i2c`. I^{2}C platform device initialization is described in `arch/arm/mach-ks8695/board-vsopenrisc.c`. It shows how RTC is attached to the I^{2}C bus.

7.8. Read Hardware Parameters like MAC Address, Serial Number etc.

Such parameters as MAC addresses, serial number etc. are stored in flash and can be accessed via special structure:

[47] For further information see `drivers/watchdog/ks8695_wdt.c`

[48] http://en.wikipedia.org/wiki/I2C

```
struct _param_hw                // v1.0
{
        byte pad[1];
        ushort szram;           // direct in MB
        ushort szflash2;
        ushort szflash1;        // direct in MB
        byte mac2[6];
        byte mac1[6];
        ushort biosid;
        char prddate[11];       // as a string ie. "01.01.2006"
        ulong serialnr;
        ulong hwrev;
        ulong magic;
} __attribute__ ((packed));
```

See `hwtest/hwtest.c->param_read_hw_params()` routine on how to read the parameters from flash.

7.9. WLAN Button (Alekto LAN Only)

OpenRISC Alekto LAN provides an additional button. By default this button is used to control WLAN card's transmit power. This button is just connected to GPIO line and will be controlled by `wland` service (refer to Section 5.1.3). To use it for other needs just disable the `wland` service. Following ioctl command for `/dev/gpio` is provided to get button's state:

`GPIO_CMD_GET_BTN_WLAN`

return value is `0` - for not pressed and `1` - for pressed.

8. `/proc`-Extensions for the OpenRISC

In the `/proc/vsopenrisc` directory reside several files to manipulate the OpenRISC hardware:

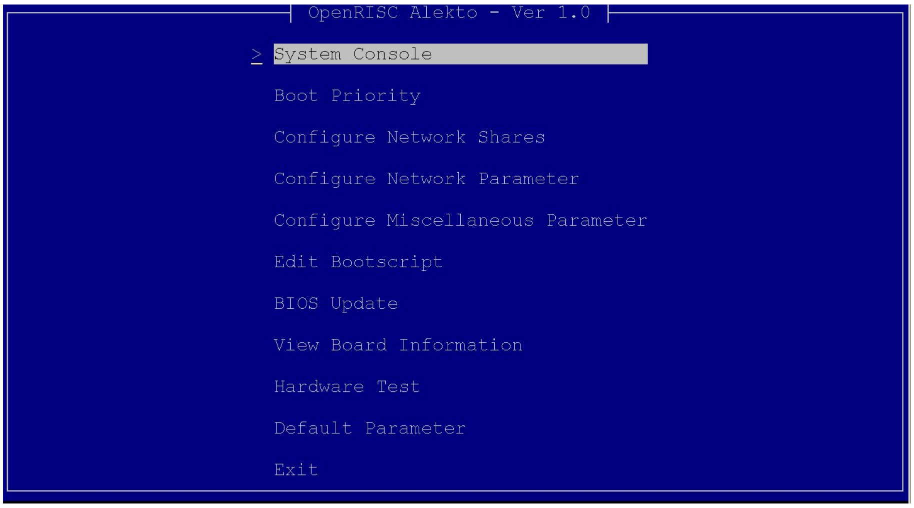
- `epld_ttyS*` - configure serial driver e.g. rs232, rs422, rs485. For exact values execute
  `echo /proc/vsopenrisc/epld_ttySn` (n=1..4)
  For detailed description see 7.4
- `gpio_*` - set and read digital IO channels. For detailed description see 7.1
- `leds` - set and read LEDs values. For detailed description see 7.3
- `buzzer, buzzer_frq` - configure the buzzer. For detailed description see 7.2
- `reset` - reboot the OpenRISC. To execute the hardware reset
  `echo 1 > /proc/vsopenrisc/reset`

Examples:

`echo 1 > /proc/vsopenrisc/buzzer`	turn buzzer on
`echo 0 > /proc/vsopenrisc/buzzer`	turn buzzer off
`echo 0x000001f4 > /proc/vsopenrisc/buzzer_frq`	this will activate the buzzer for 500ms and then deactivate it for 500ms and so on
`cat /proc/vsopenrisc/leds`	get the current LED status
`echo GREEN > /proc/vsopenrisc/leds`	turn the green LED on
`echo green > /proc/vsopenrisc/leds`	turn the green LED off
`echo rs422 > /proc/vsopenrisc/epld_ttyS2`	sets the mode of the second port to rs422
`echo 0x05 0x05 > /proc/vsopenrisc/gpio_ctrl`	set DIO0 and DIO2 to output

9. BIOS

BIOS (Basic Input Output System) lets you configure your OpenRISC e.g. configure how to boot, set up date, time and so on. To get into the BIOS, press ’ESC’ during the system start and you’ll enter the following menu (see Figure 7).



```
OpenRISC Alekto - Ver 1.0

> System Console
  Boot Priority
  Configure Network Shares
  Configure Network Parameter
  Configure Miscellaneous Parameter
  Edit Bootscript
  BIOS Update
  View Board Information
  Hardware Test
  Default Parameter
  Exit
```

Figure 7: BIOS: main menu

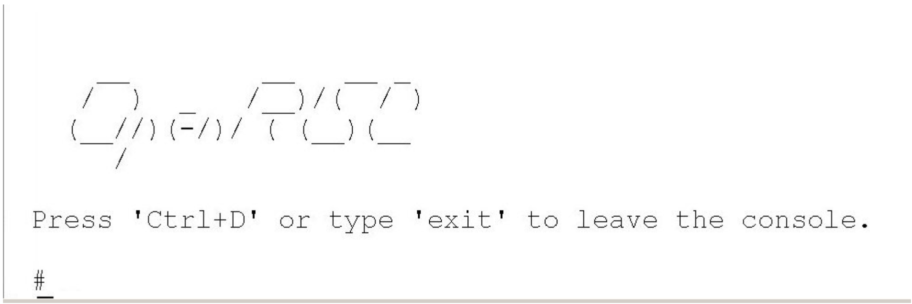
To connect to the BIOS via Telnet press reset push button for a while till WLAN LED turns on and the following output appears (see Figure 8). Now you can connect to the BIOS via network by using the IP address specified under “Configure Network Parameter” (see 9.4).

```
Press ’Esc’ to enter setup or ’Tab’ for a bootlist
```

Figure 8: BIOS Prompt for Bootlist

9.1. System Console

Choosing this menu item, you'll get to the system console. To return to the main menu press 'Ctrl+D' or execute `exit` (see Figure 9). You can use the console to mount CompactFlash, copy the kernel (see 10.11), partition the disk and so on.



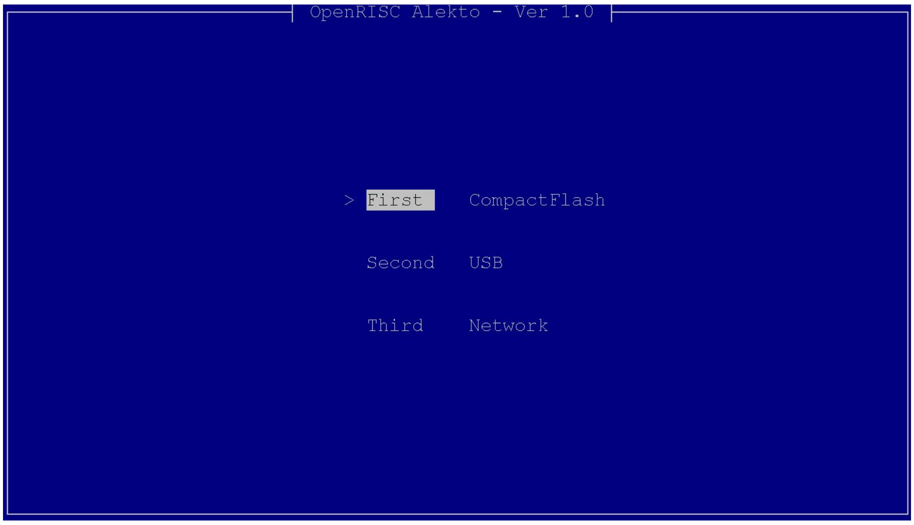
```
    __        __   __ _
   /  )  _   /__)/(  / )
  (__//)(-/)/ ( (__)(__
     /

Press 'Ctrl+D' or type 'exit' to leave the console.

#_
```

Figure 9: BIOS: System Console

9.2. Boot Priority

In this menu item you can change the boot priority (see Figure 10). You can choose between following sources:

- CompactFlash
- USB
- Network (Windows share, Samba)



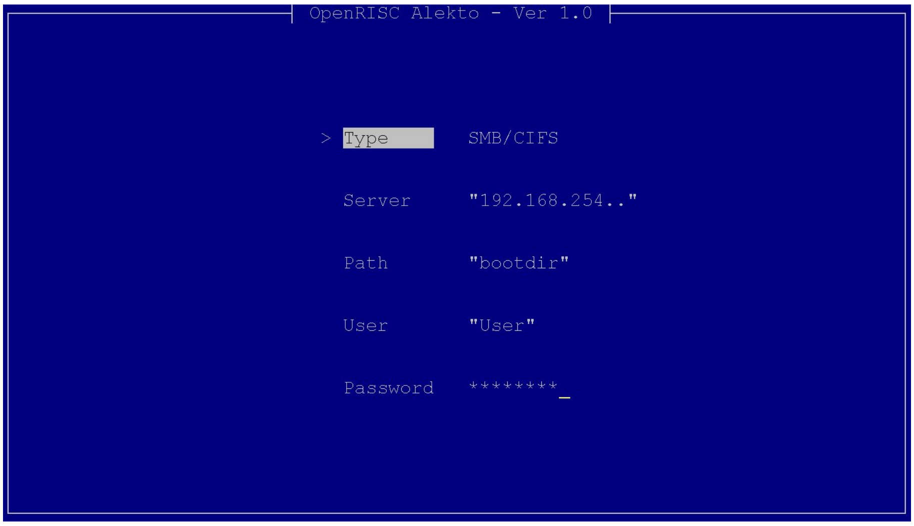
```
OpenRISC Alekto - Ver 1.0

> First    CompactFlash

  Second   USB

  Third    Network
```

Figure 10: BIOS: Boot Priority

9.3. Configure Network Shares

Here you can configure a Windows share (SMB) to boot from (see Figure 11). To use this feature, you should set up following parameters:

- Type
- Server IP address
- Samba directory name
- User name
- Password



```
                      OpenRISC Alekto - Ver 1.0

                    > Type        SMB/CIFS

                      Server      "192.168.254.."

                      Path        "bootdir"

                      User        "User"

                      Password    ********_
```

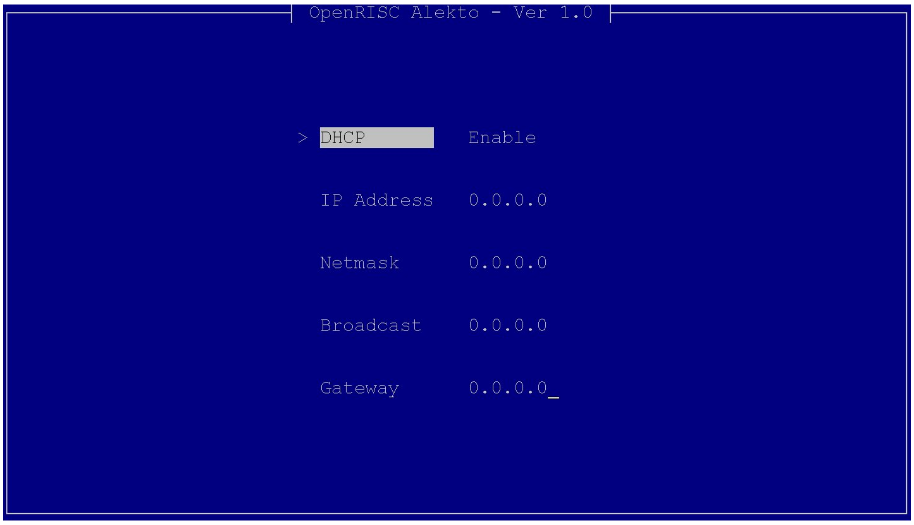
Figure 11: BIOS: Configure Network Shares

9.4. Configure Network Parameter

Here you can configure the network parameters (see Figure 12). You can choose between getting IP Address via DHCP or to assigning it statically. For the latter you should configure following parameters:

- IP Address
- Netmask
- Broadcast
- Gateway

Warning: please note that these network parameters are applicable only to BIOS not the system BIOS is booting.



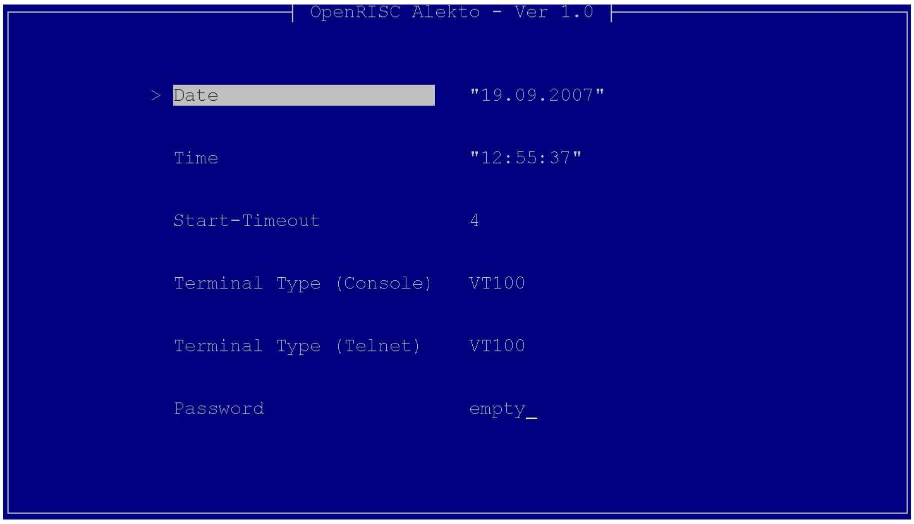
```
OpenRISC Alekto - Ver 1.0

> DHCP          Enable

  IP Address    0.0.0.0

  Netmask       0.0.0.0

  Broadcast     0.0.0.0

  Gateway       0.0.0.0_
```

Figure 12: BIOS: Configure Network Parameter

9.5. Configure Miscellaneous Parameter

Following parameters can be set up here (see Figure 13):

- Date
- Time
- Start-Timeout[49] - the time in seconds during that the BIOS or bootlist could be accessed.
- Terminal Type (Console/Telnet) - emulation type to choose by terminal software such as Hyperterminal, ZOC etc while connecting via serial link or telnet. The OpenRISC can emulate three types of terminals:
  - Linux
  - ANSI
  - VT100
- Password - BIOS protecting password



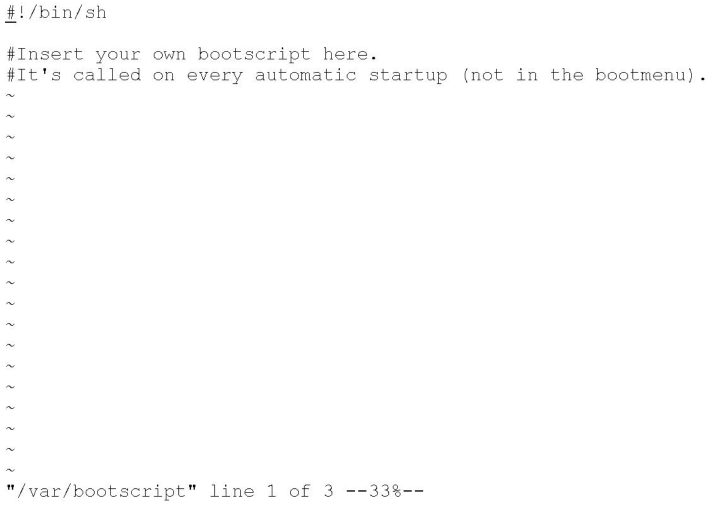
```
                    OpenRISC Alekto - Ver 1.0

          > Date                        "19.09.2007"

            Time                        "12:55:37"

            Start-Timeout               4

            Terminal Type (Console)     VT100

            Terminal Type (Telnet)      VT100

            Password                    empty_
```

Figure 13: BIOS: Configure Miscellaneous Parameter

[49] for devices with internal microSD card reader the timeout should be set to at least 5 seconds

9.6. Edit Bootscript

The bootscript can be edited here (see Figure 14):

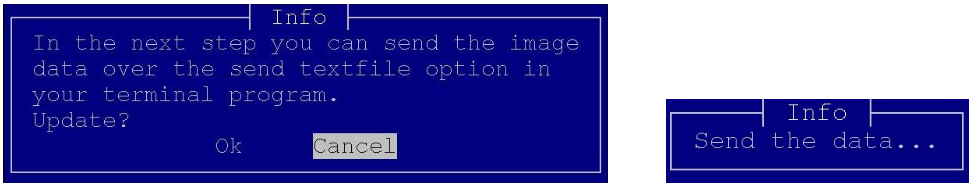


```
#!/bin/sh

#Insert your own bootscript here.
#It's called on every automatic startup (not in the bootmenu).
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"/var/bootscript" line 1 of 3 --33%--
```

Figure 14: BIOS: Edit Bootscript

9.7. BIOS Update

BIOS update can be done either via serial connection or via network connection[50].



```
Info
In the next step you can send the image
data over the send textfile option in
your terminal program.
Update?
```

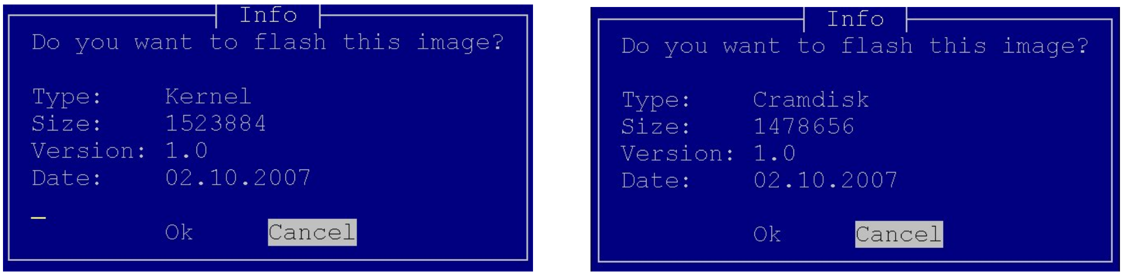
```
Info
Send the data...
```

(a) Update prompt

(b) Waiting for data

Figure 15: BIOS: Update

Update procedure via serial connection In the dialog shown in Figure 15(a) choose "Ok". "Send the data..." prompt appears (see Figure 15(b)). Select the appropriate *.b64 file and send it via the "Send Text File" functionality of your terminal software.



```
Info
Do you want to flash this image?

Type:     Kernel
Size:     1523884
Version:  1.0
Date:     02.10.2007
```

```
Info
Do you want to flash this image?

Type:     Cramdisk
Size:     1478656
Version:  1.0
Date:     02.10.2007
```

(a) Kernel

(b) Filesystem

Figure 16: BIOS: Update (Images)

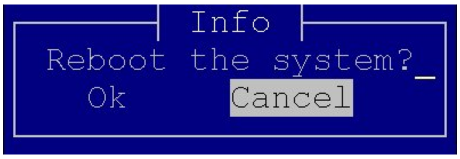
The *.b64 file consists of two files:

- Kernel - Linux kernel image (zImage)
- Cramdisk - filesystem image

After transmission you can choose between installing both kernel (see Figure 16(a)) and filesystem image (see Figure 16(b)) or only one of them.

Warning: do not power off or reset the OpenRISC during the flashing of the images!!!

The last step is to reboot the system (see Figure 17).



```
Info
Reboot the system?_
```

Figure 17: BIOS: Update (Reboot System)

[50] The network option is available since BIOS version 2.1

Update procedure via network connection Push the reset button till the WLAN LED is on. Send the appropriate *.b64 file via `netcat`[51] or `socat` (`socat` should be used if `netcat` closes connection as soon as the update image is transferred) as in the examples below:

```
netcat -w 600 192.168.254.254 23 < OpenRISC-2.2.bin.b64
```

or

```
socat -,ignoreeof TCP:192.168.254.254:23 < OpenRISC-2.2.bin.b64
```

As a result you'll see following output in your shell window (see Figure 18). The characters before this output are echo and terminal sequences for BIOS menu.

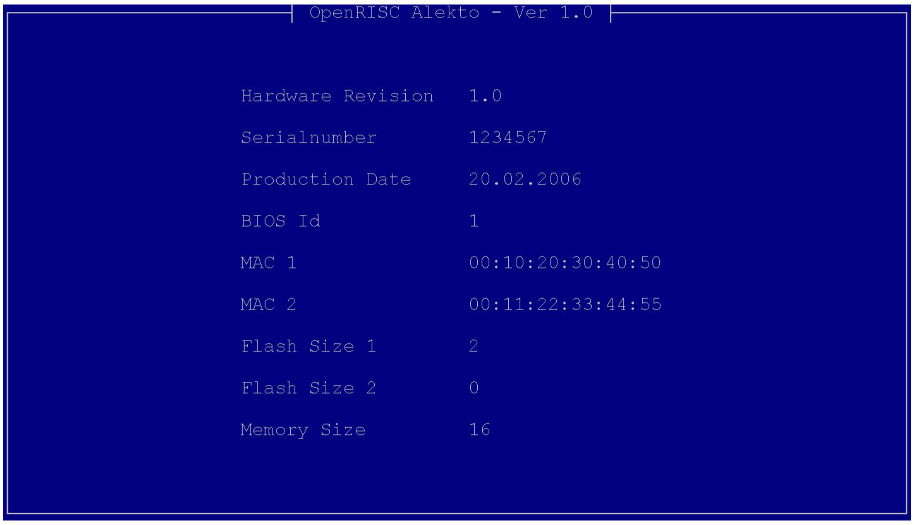
```
Update BIOS
Receiving update file
...............................................................................
...............................................................................
...............................................................................
...............................................................................
...............................................................................
Extracting binary images
Flashing image
Type:  Kernel
Size:  2012752
Version:  2.1
Date:  28.10.2008
O.K.
Flashing image
Type:  Cramdisk
Size:  1687552
Version:  2.1
Date:  28.10.2008
O.K.
BIOS updated.  The system will be restarted...
```

Figure 18: BIOS: Update via Network

[51] on some systems `netcat` binary is called `nc`. Windows version `nc.exe` is provided on the DVD "Software"

9.8. View Board Information

Shows some board information (see Figure 19)



```
OpenRISC Alekto - Ver 1.0

Hardware Revision   1.0
Serialnumber        1234567
Production Date     20.02.2006
BIOS Id             1
MAC 1               00:10:20:30:40:50
MAC 2               00:11:22:33:44:55
Flash Size 1        2
Flash Size 2        0
Memory Size         16
```

Figure 19: BIOS: View Board Information

9.9. Hardware Test

The **hwtest** program will be invoked for one cycle with the parameters listed in Table 6. For detailed information about **hwtest** refer to Appendix B.

`size`	4096	`modem`	
`cycles`	1	`sero`	/dev/ttyS1
`confirm`		`seri`	/dev/ttyS2
`serial`		`io`	
`net`		`buz`	
`gpio`		`led`	
`mpci`		`sleep`	1
`cf`		`testrtc`	
`i2c`		`eplddev`	/dev/ttyS1
`epld`		`eplddev`	/dev/ttyS2
`usb`		`usbmntdev`	/dev/sda1

Table 6: BIOS: hwtest parameters

9.10. Default Parameter

Sets all parameters to factory settings.

9.11. Exit

Leaves BIOS. If any parameters were changed, you'll be asked to save the changes.

10. Debian Installation

The installation of the Debian 5.0.3 Lenny[52] will be described in this section. The installation instructions for other Debian versions are similar with this one. Your system should meet the following requirements:

- BIOS 2.6 or later
- Serial Terminal (Emulation) at 115200bps, 8N1
- Installation source: CD/DVD-ROM on USB or Windows share (Samba)
- Installation target: CompactFlash card or USB Mass Storage device[53]

Due to the fact that OpenRISC is not officially supported by arm-debian port there is no suitable kernel in the distribution. So you have to provide boot image files to the OpenRISC to be able to start the Debian installer. These files (see the description below) can be downloaded from our SVN repository.

Generally two types of Debian installer are provided:

- CD-ROM Installer
- Network Installer

For Debian 4.0 Etch you'll find only CD-ROM Installer `etch/initrd.gz`. For Debian 5.0 Lenny both installer are supplied `lenny/initrd.gz-cdrom` and `lenny/initrd.gz-net`. The desired installer must be renamed to `initrd.gz` when copying to a mass storage device or Windows share. You can get the latest install DVDs at http://www.debian.org/distrib/

10.1. Preparing Boot Image Files

Copy the downloaded files to the root directory of your USB mass storage device. These files are:

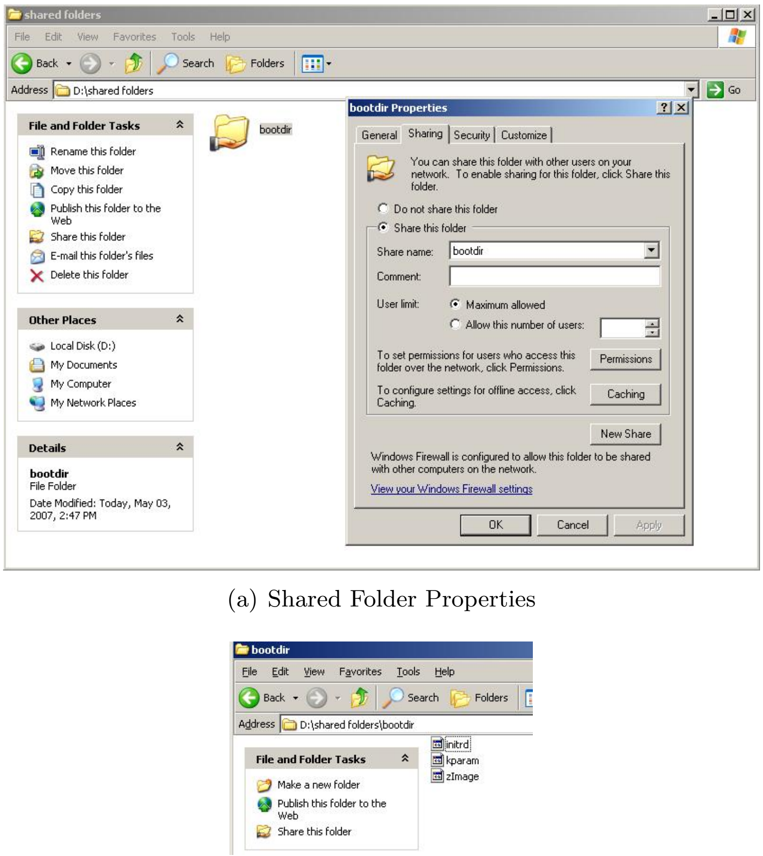
- `initrd.gz` - RAM disk image with Debian Installer
- `zImage` - Linux kernel
- `kparam` - Kernel parameter like `root=/dev/ram` describing that kernel should use RAM disk as root filesystem (refer to Section 4.1)

Alternatively you can use a Windows share (Samba) (see Figure 20). It can be made in four steps:

1. create shared folder. For example D:\shared folders\bootdir (see Figure 20(a))
2. copy the image files to this folder (see Figure 20(b))
3. enter BIOS and set the Network Parameter (see 9.4)
4. configure your BIOS to access the network share directory (see 9.3)

[52] For detailed information about Debian Installation visit http://www.debian.org/releases/stable/arm

[53] 1GB or more recommended



bootdir
File Folder
Date Modified: Today, May 03, 2007, 2:47 PM

You can share this folder with other users on your network. To enable sharing for this folder, click Share this folder.

○ Do not share this folder
● Share this folder

Share name: bootdir
Comment:
User limit: ● Maximum allowed
○ Allow this number of users:

To set permissions for users who access this folder over the network, click Permissions.

To configure settings for offline access, click Caching.

Windows Firewall is configured to allow this folder to be shared with other computers on the network.

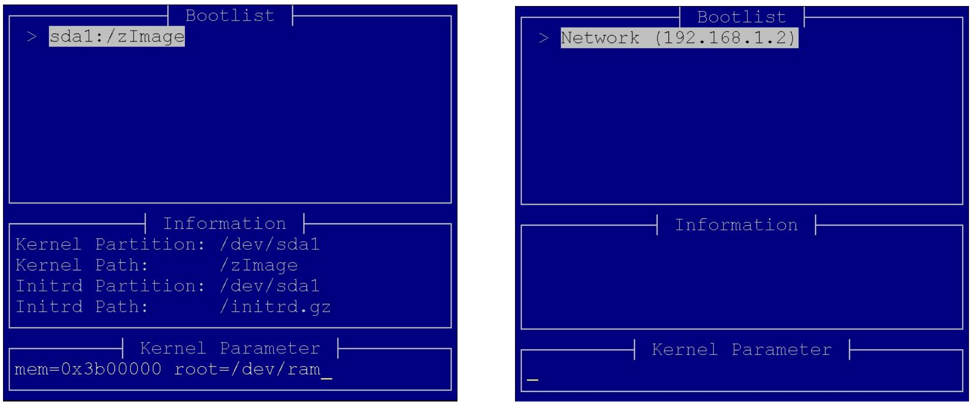
(a) Shared Folder Properties

(b) Boot Image Files

Figure 20: Windows Share

10.2. Starting Debian Installer

Exit BIOS and save changes if required. Press ’Tab’ to access bootlist and choose USB Mass Storage Device (see Figure 21(a)) or Network (see Figure 21(b)). The Installer will be put into low memory mode (see Figure 22). Press ’Enter’ and choose your country in the next dialog.



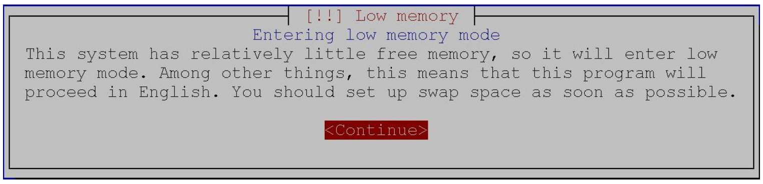
```
Bootlist
 > sda1:/zImage

Information
Kernel Partition: /dev/sda1
Kernel Path:      /zImage
Initrd Partition: /dev/sda1
Initrd Path:      /initrd.gz

Kernel Parameter
mem=0x3b00000 root=/dev/ram_
```

(a) Mass storage

```
Bootlist
 > Network (192.168.1.2)

Information

Kernel Parameter
_
```

(b) Windows share

Figure 21: Bootlist

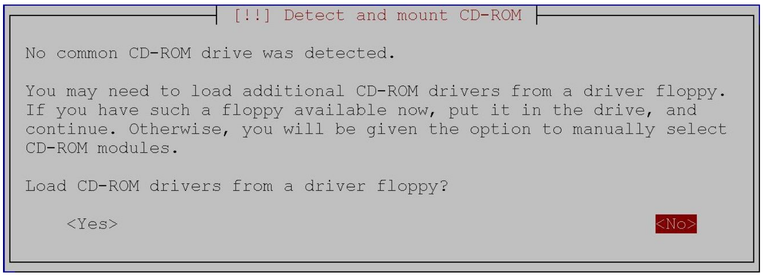


```
[!!] Low memory
Entering low memory mode
This system has relatively little free memory, so it will enter low
memory mode. Among other things, this means that this program will
proceed in English. You should set up swap space as soon as possible.

<Continue>
```

Figure 22: Low Memory Mode

10.3. Choosing CD-ROM

Debian Installer will try to find a device driver for your USB CD-ROM. This could fail as for Debian Etch installer. To select the device manually answer with 'No' (see Figure 23). In the second dialog say 'Yes' to manually select your CD-ROM module. In the third dialog select 'none'. In the fourth dialog press 'Enter' to select `/dev/cdrom` device file to access the CD-ROM. The Debian Installer will now scan your CD-ROM.

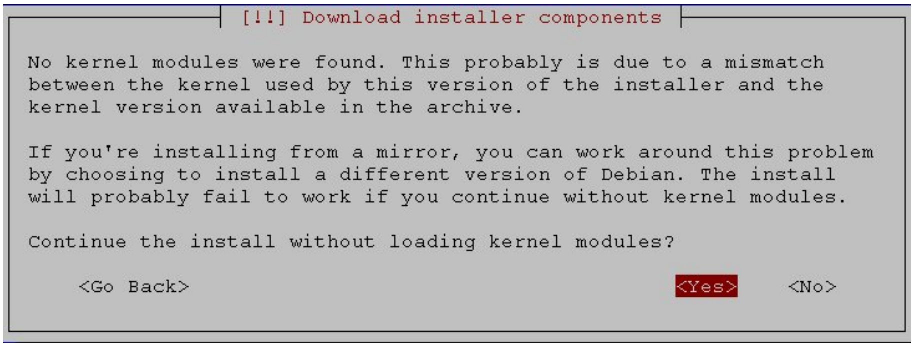


```
[!!] Detect and mount CD-ROM

No common CD-ROM drive was detected.

You may need to load additional CD-ROM drivers from a driver floppy.
If you have such a floppy available now, put it in the drive, and
continue. Otherwise, you will be given the option to manually select
CD-ROM modules.

Load CD-ROM drivers from a driver floppy?

    <Yes>                                                  <No>
```

Figure 23: Detect and mount CD-ROM

10.4. Load Installer Components from CD

First you'll see a dialog with the statement that the kernel modules were not found (see Figure 24). It is due to the issue that all needed modules were compiled into the kernel. So answer with 'Yes'. Additional components will be loaded.

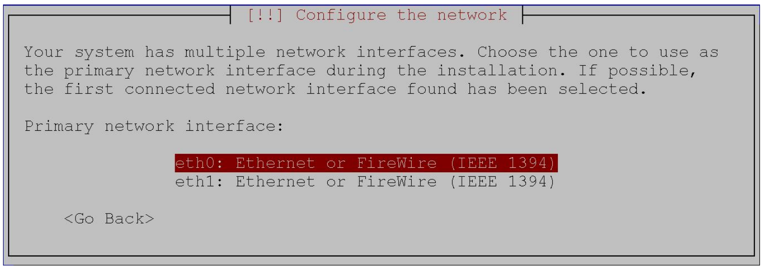


```
[!!] Download installer components

No kernel modules were found. This probably is due to a mismatch
between the kernel used by this version of the installer and the
kernel version available in the archive.

If you're installing from a mirror, you can work around this problem
by choosing to install a different version of Debian. The install
will probably fail to work if you continue without kernel modules.

Continue the install without loading kernel modules?

    <Go Back>                                   <Yes>    <No>
```

Figure 24: No Modules Found Warning

10.5. Network Configuration

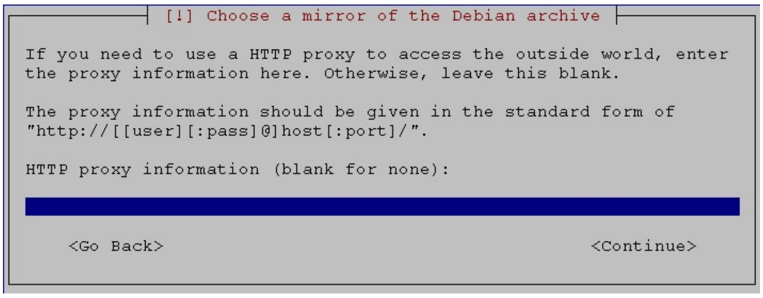
In the next dialog the network configuration will be executed (see Figure 25). Choose the network interface to which the Ethernet cable is connected to. In the next two dialogs enter host and domain names.



```
[!!] Configure the network

Your system has multiple network interfaces. Choose the one to use as
the primary network interface during the installation. If possible,
the first connected network interface found has been selected.

Primary network interface:

              eth0: Ethernet or FireWire (IEEE 1394)
              eth1: Ethernet or FireWire (IEEE 1394)

    <Go Back>
```

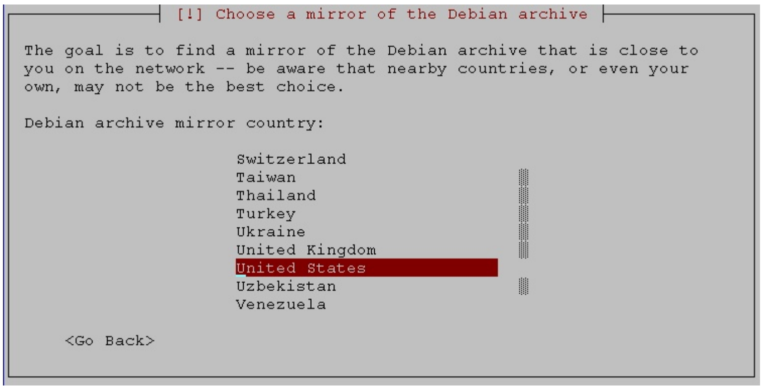
Figure 25: Network configuration

In the case of network installation you'll be asked to configure package mirror as in Figures 26 and 27.



```
[!] Choose a mirror of the Debian archive

If you need to use a HTTP proxy to access the outside world, enter
the proxy information here. Otherwise, leave this blank.

The proxy information should be given in the standard form of
"http://[[user][:pass]@]host[:port]/".

HTTP proxy information (blank for none):

    <Go Back>                                          <Continue>
```

Figure 26: Proxy Configuration

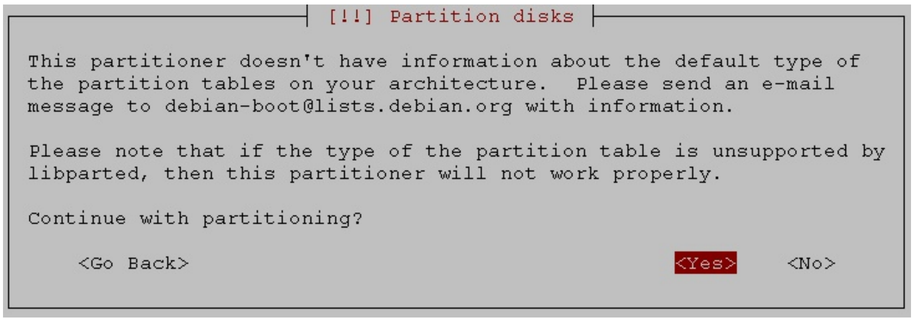


```
[!] Choose a mirror of the Debian archive

The goal is to find a mirror of the Debian archive that is close to
you on the network -- be aware that nearby countries, or even your
own, may not be the best choice.

Debian archive mirror country:

                     Switzerland
                     Taiwan
                     Thailand
                     Turkey
                     Ukraine
                     United Kingdom
                     United States
                     Uzbekistan
                     Venezuela

    <Go Back>
```

Figure 27: Mirror Configuration

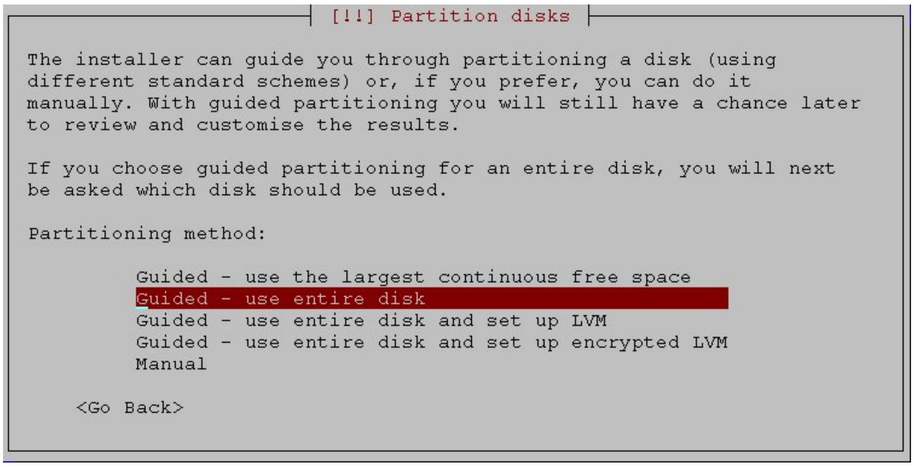
10.6. Partition Disks

After scanning your target storage media, you'll be asked to continue with partitioning (see Figure 28). Answer with 'Yes'. In the second dialog select 'Guided - use entire disk' as in Figure 29.



```
[!!] Partition disks

This partitioner doesn't have information about the default type of
the partition tables on your architecture.  Please send an e-mail
message to debian-boot@lists.debian.org with information.

Please note that if the type of the partition table is unsupported by
libparted, then this partitioner will not work properly.

Continue with partitioning?

    <Go Back>                                         <Yes>    <No>
```

Figure 28: Partition Disks



```
[!!] Partition disks

The installer can guide you through partitioning a disk (using
different standard schemes) or, if you prefer, you can do it
manually. With guided partitioning you will still have a chance later
to review and customise the results.

If you choose guided partitioning for an entire disk, you will next
be asked which disk should be used.

Partitioning method:

         Guided - use the largest continuous free space
         Guided - use entire disk
         Guided - use entire disk and set up LVM
         Guided - use entire disk and set up encrypted LVM
         Manual

    <Go Back>
```

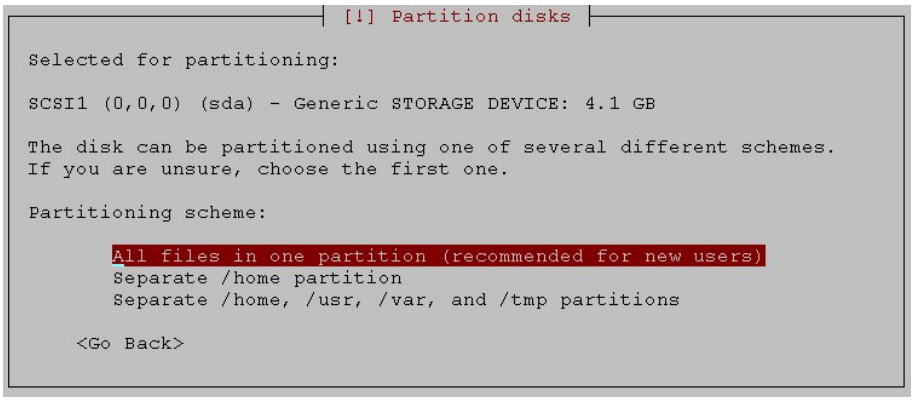
Figure 29: Partitioning Method

In the next dialog confirm your target disk and then select 'All files in one partition' scheme as in Figure 30.

Select 'msdos' for partition table type as in Figure 31

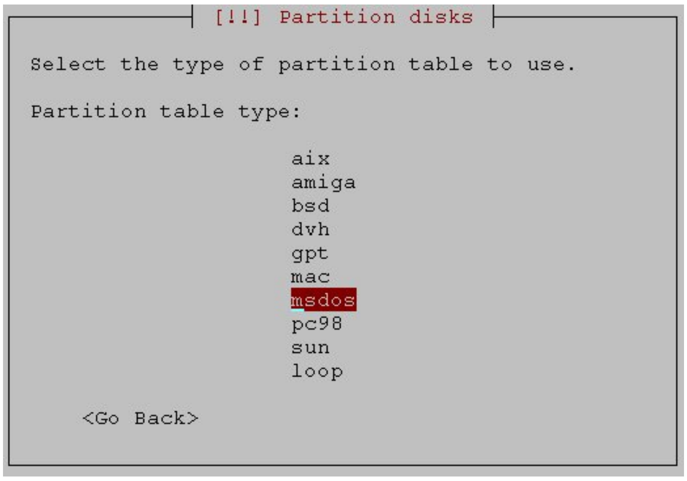
You'll get the following partition table as in Figure 32. It is very important that you leave swap partition during the installation. You can then change your partitions so you have only one partition and swap as a swap file (see Section 4.6). Select 'Finish partitioning and write changes to disk'. In the next dialog select 'Yes' to format both partitions.

After doing this select 'Finish partitioning and write changes to disk'. In the next dialog select 'Yes' to format both partitions.



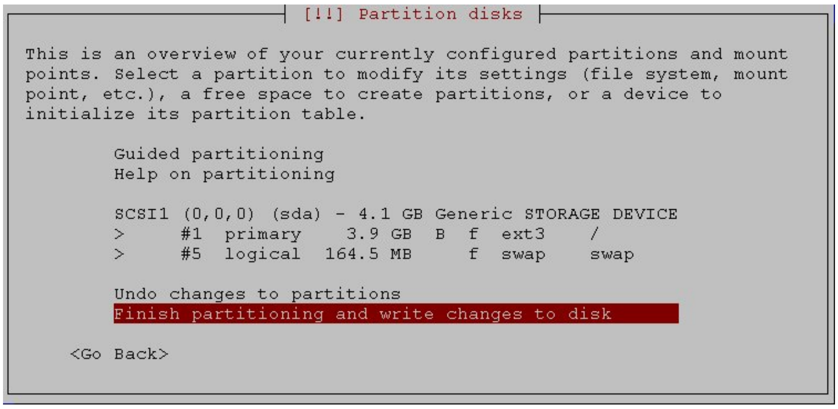
```
                     [!] Partition disks

Selected for partitioning:

SCSI1 (0,0,0) (sda) - Generic STORAGE DEVICE: 4.1 GB

The disk can be partitioned using one of several different schemes.
If you are unsure, choose the first one.

Partitioning scheme:

       All files in one partition (recommended for new users)
       Separate /home partition
       Separate /home, /usr, /var, and /tmp partitions

    <Go Back>
```

Figure 30: Partitioning Scheme



```
              [!!] Partition disks

Select the type of partition table to use.

Partition table type:

                    aix
                    amiga
                    bsd
                    dvh
                    gpt
                    mac
                    msdos
                    pc98
                    sun
                    loop

     <Go Back>
```

Figure 31: Partition Table Type



```
                        [!!] Partition disks

This is an overview of your currently configured partitions and mount
points. Select a partition to modify its settings (file system, mount
point, etc.), a free space to create partitions, or a device to
initialize its partition table.

        Guided partitioning
        Help on partitioning

        SCSI1 (0,0,0) (sda) - 4.1 GB Generic STORAGE DEVICE
        >     #1  primary    3.9 GB  B  f  ext3    /
        >     #5  logical  164.5 MB     f  swap    swap

        Undo changes to partitions
        Finish partitioning and write changes to disk

    <Go Back>
```

Figure 32: Partition Table

10.7. Setting Passwords

After partitioning the disks and setting up the clock, you'll be asked to set a password for 'root' and create a new user.

10.8. Install the Base System

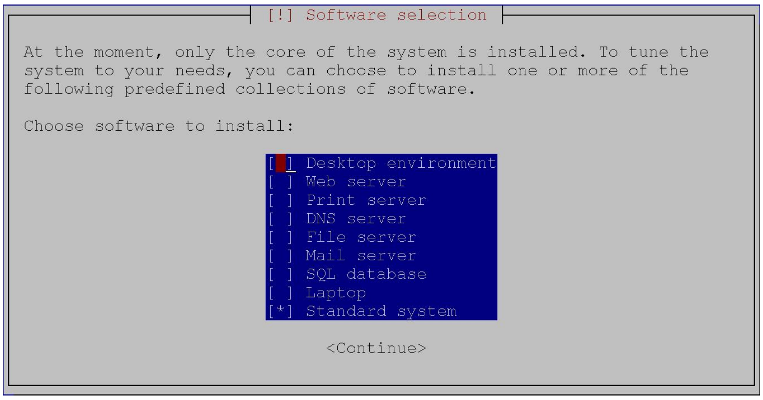
As the OpenRISC is not officially supported by Debian, there is no kernel shipped with the distribution. Answer with 'Yes', when you'll be asked to install without a kernel.

10.9. Configure the Package Manager

In the case of a CD-ROM installer you'll be asked if you want to download and install updates from a network mirror. The network mirror will be configured as described in Subsection 10.5. After configuring the package manager, you'll be asked to participate in the package usage survey.

10.10. Software Selection

In this dialog you can select which component groups will be installed on your system (see Figure 33).



```
[!] Software selection

At the moment, only the core of the system is installed. To tune the
system to your needs, you can choose to install one or more of the
following predefined collections of software.

Choose software to install:

                    [ ] Desktop environment
                    [ ] Web server
                    [ ] Print server
                    [ ] DNS server
                    [ ] File server
                    [ ] Mail server
                    [ ] SQL database
                    [ ] Laptop
                    [*] Standard system

                         <Continue>
```

Figure 33: Software Selection

10.11. Finish the Installation

Due to the lack of the official support by Debian there is no boot loader (see Figure 34). After reboot, enter BIOS and go to the console. Mount the first partition:

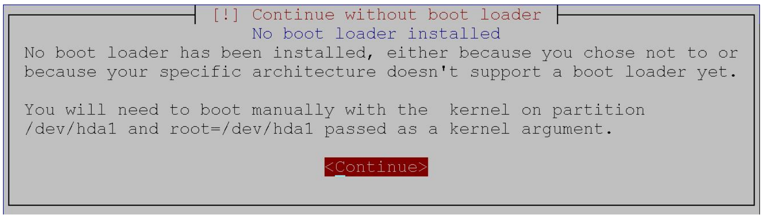
```
mount /dev/hda1 /mnt
```

Copy zImage either from your host or USB mass storage device into the /boot directory and execute:

```
echo "mem=59M root=/dev/hda1" > /boot/kparam
```

Exit BIOS and enter the bootlist by pressing 'Tab'. You can now boot from /dev/hda1.
To enable the console port edit /etc/inittab, so that tty1-tty6 are commented and console is configured for 115200bps:

```
...
# Note that on most Debian systems tty7 is used by the X Window System,
# so if you want to add more getty's go ahead but skip tty7 if you run X.
#
#1:2345:respawn:/sbin/getty 38400 tty1
#2:23:respawn:/sbin/getty 38400 tty2
#3:23:respawn:/sbin/getty 38400 tty3
#4:23:respawn:/sbin/getty 38400 tty4
#5:23:respawn:/sbin/getty 38400 tty5
#6:23:respawn:/sbin/getty 38400 tty6
# Example how to put a getty on a serial line (for a terminal)
#
T0:2345:respawn:/sbin/getty -L console 115200 linux
#T1:23:respawn:/sbin/getty -L ttyS1 9600 vt100
...
```



```
[!] Continue without boot loader
No boot loader installed
No boot loader has been installed, either because you chose not to or
because your specific architecture doesn't support a boot loader yet.

You will need to boot manually with the  kernel on partition
/dev/hda1 and root=/dev/hda1 passed as a kernel argument.
```

Figure 34: Continue without Boot Loader

A. Debian Maintenance Notes

A.1. Debian Package Management

Debian uses following utilities for the package management[54] :

- dpkg - the main package management program.
- APT - the Advanced Package Tool. It provides the `apt-get` program. `apt-get` allows a simple way to retrieve and install packages from multiple sources using the command line. Unlike `dpkg`, `apt-get` does not understand .deb files, it works with the packages proper name and can only install .deb archives from a source specified in `/etc/apt/sources.list`. `apt-get` will call `dpkg` directly after downloading the .deb archives from the configured sources.
- aptitude - a package manager for Debian GNU/Linux systems that provides a frontend to the apt package management infrastructure. `aptitude` is a text-based interface using the curses library, it can be used to perform management tasks in a fast and easy way.

To find a package execute:

`apt-cache search` *`pkg name`*

To view info such as version, dependencies, installed size etc. execute:

`apt-cache show` *`pkg name`*

To install packages execute:

`apt-get install` *`pkg1 pkg2`* `...` (su rights needed)

To update the list of package known by your system execute:

`apt-get update` (su rights needed)

To upgrade all the packages on your system

`apt-get upgrade` (su rights needed)

To remove packages from your system execute:

`apt-get remove` *`pkg1 pkg2`* `...` (su rights needed)

To install a package that is not contained in the repository download the *.deb file and execute:

`dpkg -i` *`pkg file name`* (su rights needed)

One of the on-line repositories is already configured, to use the enclosed DVD "Debian" uncomment the first repository entry:

```
# deb cdrom:[Debian GNU/Linux 4.0 r0 _Etch_ - Official arm DVD Binary-1 20070407-18:08]/ etch contrib
main
deb http://security.debian.org/ etch/updates main contrib
deb-src http://security.debian.org/ etch/updates main contrib
deb http://ftp.de.debian.org/debian stable main
```

[54] for detailed information visit http://www.debian.org/doc/FAQ/ch-pkgtools.en.html

A.2. Keep a Track on Disk Usage

To get the list of all installed packages with its installed sizes execute:

```
dpkg-query -W -f'${Package}\t${Installed-Size}\t${Status}\n' | awk '/installed/ { print
$2 "\t" $1 }'
```

To estimate the file space usage execute:

```
du -h
```

To get a HTML output execute:

```
durep -w /tmp/web/
```

For detailed information see the manpages for du and durep.
To estimate free disk space execute:

```
df -h
```

B. Hardware Test Utility: hwtest

The test utility for the OpenRISC hardware is provided in both BIOS and preconfigured Debian CF-images and consists of the following test modules:

- Network
- MiniPCI
- Serial
- GPIO
- CompactFlash
- I2C and RTC
- EPLD
- USB
- CAN
- WLAN Button

Three modes can be chosen for the test execution :

1. userless - executes all tests and shows the statistics at the end
2. fully interactive - the user has to acknowledge each test
3. half interactive - all of the automatic tests such as Network, MiniPCI, Serial, EPLD, USB, CAN and CompactFlash will be executed without user acknowledgment but GPIO and I2C would ask for the acknowledgment

The parameters can be specified in the command line or stored in the configuration file[55]. Without any parameter, the program will not execute any test. For the list of common parameters see Table 7.

For detailed information about options execute:

```
hwtest --help
```

[55] To use configuration file execute hwtest `--cfg=hwtest.conf`

`--cfg <file name>`	configuration file
`--cycles <number>`	number of test cycles. Default: infinite
`--size <bytes>`	test file size in bytes. Default: 1048576 bytes
`--mode <mode>`	`uless` for userless, `fint` for fully interactive, `hint` for half interactive. Default: `uless`
`--failstop`	stop testing after the first failure
`--verbose`	verbose output
`--net <params>`	network test module with its parameters
`--mpci <params>`	MiniPCI test module with its parameters
`--serial <params>`	serial test module with its parameters
`--gpio <params>`	GPIO test module with its parameters
`--cf <params>`	CompactFlash test module with its parameters
`--i2c <params>`	I2C and RTC test module with its parameters
`--epld <params>`	EPLD test module with its parameters
`--usb <params>`	USB test module with its parameters
`--can <params>`	CAN test module with its parameters
`--wlanbtn <params>`	WLAN button test with its parameters

Table 7: Common parameters and test modules

B.1. Network Test

Two interfaces must be connected with each other for the network test (either with a patch or crossover cable). A test file will be sent as raw ethernet packets. To see the packet content use the `--verbose` flag. The size of the ethernet packet (in bytes) can be defined with `--nblock` option. The delays between two packets will be defined with the `--ndelay` option (default 100000 microseconds).

Usage example:

```
hwtest --cycles=1 --size=4096 --net --verbose
```

executes the network test with a 4096 byte test file in verbose mode

B.2. MiniPCI Test

During the test `/proc/bus/pci/devices` will be searched for the presence of the WLAN card. If the WLAN card is found the name of the chip will be printed. After that, the WLAN environment will be searched for available participants.

Usage example:

```
hwtest --cycles=1 --mpci
```

executes the MiniPCI test

B.3. Serial Test

To test the serial port you will need a special null-modem cable. Following pinout will be used (see Table 8 and Figure 35 without USB-CAN connector):

TX	↔	RX
RX	↔	TX
RTS	↔	CTS, RI
CTS, RI	↔	RTS
DTR	↔	DSR, CD
DSR, CD	↔	DTR
GND	↔	GND

Table 8: Serial Test Cable Wiring

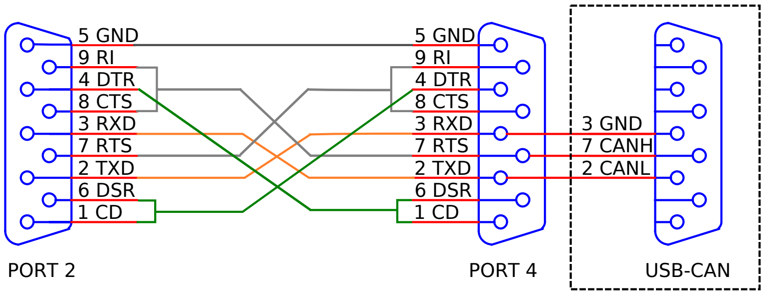


Figure 35: Serial Test Cable Wiring

The test file will be transferred from the first interface to the other and vice versa. This test can be also used to simultaneously test the serial and USB interfaces. To do this you must connect the USB port with the serial one using a USB-to-Serial adapter based on the FTDI chip. The serial test module has its own parameters `--seri` and `--sero` to configure the interfaces. With the `--sblock` parameter, the serial write block size can be configured. By default it is set to 64 bytes. The `--rtscts` option enables hardware handshake. The `--modem` option enables the test of the modem status pins.

Usage examples:

```
hwtest --cycles=1 --size=4096 --serial --seri=/dev/ttyS1 --sero=/dev/ttyS2
```
executes the serial test for on-board serial interfaces

```
hwtest --cycles=1 --size=4096 --serial --sblock=200 --rtscts
```
executes the serial test for default on-board serial interfaces (`/dev/ttyS1` and `/dev/ttyS2`) using hardware handshake and write block size of 200 bytes

```
hwtest --cycles=1 --size=4096 --serial --seri=/dev/ttyS1 --sero=/dev/ttyUSB0
```
executes the serial test using a USB-to-Serial adapter based on the FTDI chip

B.4. GPIO Test

The GPIO test module consists of following tests:

- IO test (will be activated with `--io` option)
- poll(), select() and interrupt functionality test (will be activated with `--poll` option)
- buzzer test (will be activated with `--buz` option)
- LEDs test (will be activated with `--led` option)

The I/O pins must be connected with each other for the IO test (using 4,7k resistors for example) (see Table 9). The optical isolated input channels will be connected to relays by Alena without any resistors(see Table 10).

DIO0	↔	DIO1			
DIO2	↔	DIO3	DIO2	↔	DIO3
DIO4	↔	DIO5	DIO4	↔	DIO5
DIO6	↔	DIO7			

Table 9: I/O Pin Connections for Alekto (left) and Alena (right)

DI0-G	↔	DO6-B
DI0-S	↔	DO6-C
DI1-G	↔	DO7-B
DI1-S	↔	DO7-C

Table 10: Optically Isolated Input Channels and Relays

Usage examples:

```
hwtest --cycles=1 --gpio --io --poll --buz --led --mode=fint
```

executes the complete GPIO test in full interactive mode

```
hwtest --cycles=1 --gpio --io
```

executes only the IO test in userless mode

B.5. CompactFlash Test

The CompactFlash test must be started from the BIOS or USB mass storage device, because the CF card must be unmounted. A test file will be written to the CF card during the test and then compared with the original one. After that, the test file will be removed from the CF card.

Usage example:

```
hwtest --cycles=1 --size=4096 --cf
```

executes the CompactFlash test with 4096 bytes big test file

B.6. I2C and RTC Test

A time stamp will be read from the Real Time Clock using I2C bus during the RTC test . The second time stamp will be read after the delay in seconds specified by `--sleep` option (default value is 1 second) and then compared. The test can be activated/deactivated with `--testrtc` and `--testlcd` options.

The LCD display (EAT123A-I2C) must be connected to the OpenRISC for the I2C test. The program connects to the display and "O.K." string must be visible on the display.

Usage examples:

```
hwtest --cycles=1 --i2c --testrtc --testlcd
```

executes the RTC and LCD tests

```
hwtest --cycles=1 --i2c --testrtc
```
executes the RTC test only

```
hwtest --cycles=1 --i2c --testlcd
```
executes the LCD test only

B.7. EPLD Test

Each UART has its own EPLD to switch between RS232, RS422 and RS485 modes. Current configuration will be acquired and then switched to RS232 or RS422 depending on what the current configuration was. The EPLDs will be chosen using serial port devices configured in `--eplddev` option.

Example:

```
hwtest --cycles=1 --epld --eplddev=/dev/ttyS1 --eplddev=/dev/ttyS2
```
executes the EPLD test for both serial ports

B.8. USB Test

For USB test USB mass storage device should connected to the USB port. The device must be properly detected and mounted. `--usbmntdev` option defines which device to mount.

Usage examples:

```
hwtest --cycles=1 --usb --usbmntdev=/dev/sda1
```
executes the USB test for the first detected mass storage device

```
hwtest --cycles=1 --usb --usbmntdev=/dev/sda1 --usbmntdev=/dev/sdb1
```
executes the USB test for the first and the second detected mass storage devices

B.9. CAN Test

The CAN test will be made between the internal CAN interface and the VScom USB-CAN device. At first the USB-CAN sends one frame with the following data 0xDEADBEEF00000001, after internal interface receives this frame it send the same frame in return. There is only one parameter `--canpseed` to configure the CAN baudrate.

```
hwtest --cycles=1 --can --canspeed=20000
```
executes CAN test at 20Kbit/s

B.10. WLAN Button Test

WLAN Button test checks if WLAN button and WLAN button LED are functioning properly. Starting the test will let the WLAN button LED light on. The button must be pressed during the via –wbtimeout specified time slot. After pressing the button the LED will go off and the test is successful.

```
hwtest --cycles=1 --wlanbtn --wbtimeout=5
```

executes WLAN button test for 5 seconds

B.11. All-in-one Test for Alena

It is possible to test all serial ports, CAN and GPIO without reattaching the cables. Following components will be needed to make such a test:

- modified USB cable for USB-CAN (see Figure 36) so that GND and VCC wires will switched through relays and the outer GND is also isolated
- modified null-modem cable to connect PORT2, PORT4 and USB-CAN together (see Figure 35). Ports 1 and 3 will be connected with the same cable but without connector for USB-CAN
- special board for GPIO tests, that connects optically isolated inputs to bidirectional channels (see Figure 37)
- special `--canprod` flag must be activated to use this features

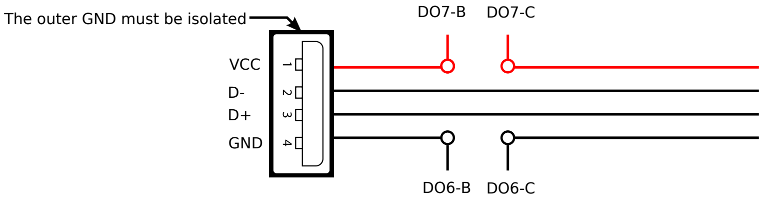


Figure 36: Modified USB cable

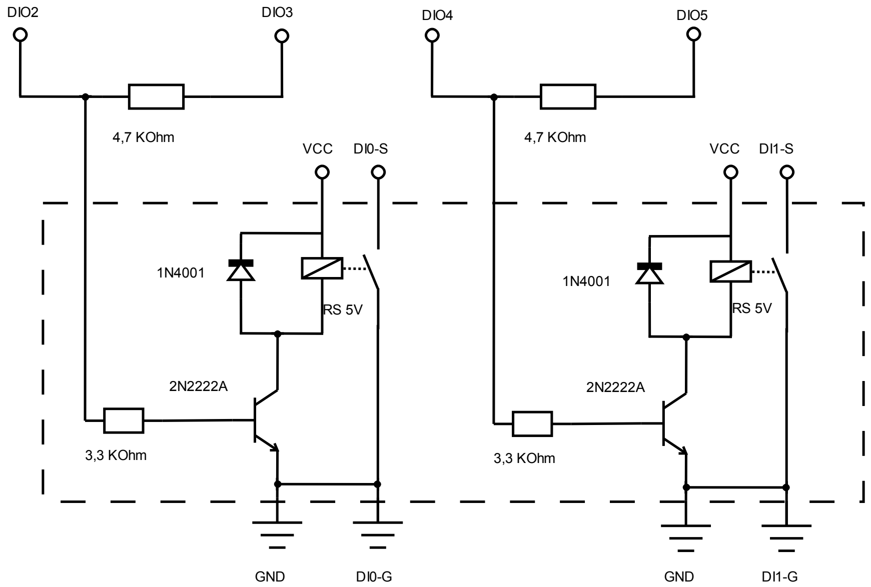


Figure 37: Special Board for GPIO Tests

Connect all components as described above. With the following command GPIO, CAN and serial tests will be executed:

```
hwtest --cycles=2 --size=4096 --can --canspeed=20000 --canprod --serial --seri=/dev/ttyS2
--sero=/dev/ttyS4 --gpio --io --poll
```

CAN test must be executed with low speed like 20000 bit/s, because the cable has no termination resistors. All other tests can be made without further modifications.

B.12. Build Notes

hwtest can be built either with GNU make or CMake. Both `Makefile` and `CMakeLists.txt` are supplied. The command line parameter parser was generated using GNU gengetopt[56]. To be able to generate the parser for changed options in `hwtest.ggo` one has to install GNU gengetopt tool.

[56] http://www.gnu.org/software/gengetopt/

C. Managing System Images

C.1. VS Image Tool: VSImgTool

VSImgTool[57] was developed to make/copy images from/to the flash drives connected to a Windows host. This utility can make the whole image or only part of it. Refer to Table 11 for parameters.

Warning: please note that you must have administrator privileges to properly execute VSImgTool.

`--list`	list available flash drives with their size in bytes
`source destination`	copy image from source to destination
`source destination custom_size`	copy only part of the image from source to destination

Table 11: VSImgTool call parameters

C.1.1. Burn System Image to CF/microSD Card

1. Insert CF/microSD card in your card reader and execute
   ```
   vsimgtool --list
   ```
   you should see the similar output:
   ```
   >vsimgtool.exe --list
   VSImgTool 1.0
   Available devices:
   PhysicalDrive2 Size:  1024450560 bytes
   ```
2. to copy the complete image to the CF/microSD card on the PhysicalDrive2 execute
   ```
   vsimgtool.exe 26082008_etch_complete.bin PhysicalDrive2
   ```
   you should see the following output:
   ```
   >vsimgtool.exe 26082008_etch_complete.bin PhysicalDrive2
   VSImgTool 1.0
   Image source:  26082008_etch_complete.bin
   Size:  1008451584 bytes
   Removable destination:  PhysicalDrive2
   Size:  1024450560 bytes
   The data on the destination will be lost.  Are you sure you want to continue?  [yes/no]
   >
   ```
3. answer with "yes" and the copying process will begin
4. at the end of the copying process you should see the elapsed time and the size of the burned/-copied image

If your CF/microSD card has bigger capacity than the system image, you can just insert it in the card reader and change the partition with some partition manager like Gparted[58] on your Linux host.

[57] VSImgTool can be found on the DVD "Software" under `tools`

[58] http://en.wikipedia.org/wiki/Gparted

C.1.2. Make An Image From CF/microSD Card

`vsimgtool` *`drive_name`* `image.bin`

where *`drive_name`* is the name showed by `vsimgtool --list`.

For 256MB and 512MB CF/microSD cards there are two custom sizes prepared:

`vsimgtool PhysicalDrive2 image.bin 256`
will copy exactly $244 \cdot 1024 \cdot 1024 = 255852544$ bytes

`vsimgtool PhysicalDrive2 image.bin 512`
will copy exactly $488 \cdot 1024 \cdot 1024 = 511705088$ bytes

`vsimgtool PhysicalDrive2 image.bin 519192576`
will copy exactly 519192576 bytes

C.2. dd

To copy a system image to a CF/microSD card the DVD "Software" must be mounted (for example to `/mnt`). If USB card reader is used the CF/microSD could be found under `/dev/sdx` devices (for example `/dev/sda`, `/dev/sdb` etc). If CF card is attached to IDE channel then CF card could be found under `/dev/hdx` (for example `/dev/hda`, `/dev/hdb` etc). Please note, that the system images provided on the DVD "Software" are complete images of a CF/microSD card not of the single partition and must be applied to the device like `/dev/sda` and not `/dev/sda1`. For the example below it will be assumed that DVD "Software" is mounted to `/mnt`, the CF/microSD card is assigned to `/dev/sda` device and the system image is `27032008_etch_complete.bin`:

```
su
mount /dev/cdrom /mnt
exit
dd if=/mnt/images/27032008_etch_complete.bin of=/dev/sda bs=4096
```

To create an image from the CF/microSD card execute following:

```
dd if=/dev/sda of=/home/user/image.bin
```

D. Eclipse

D.1. Installation Notes

Eclipse is an open source community whose projects are focused on building an extensible development platform, runtimes and application frameworks for building, deploying and managing software across the entire software life cycle. In Debian run:

```
apt-get install eclipse
```

to install it. Additionally you'll need to download and install Java 2 SDK or Java 2 JRE from Sun Microsystems[59]. It can happen that Eclipse doesn't find Java binary, in this case execute:

```
eclipse -vm java
```

To get working with C projects you'll need the CDT[60] and Embedded CDT[61] plug-ins to compile and debug your software for the OpenRISC.

If you'd like to checkout Kernel or examples from our Subversion repository, use doxygen etc. you could also find these plug-ins as useful:

- Subclipse (subclipse.tigris.org)
- Eclox (home.gna.org/eclox)
- Gengetopt Eclipse (ggoeclipse.sourceforge.net)
- CMake Editor(www.cthing.com/CMakeEd.asp)

[59] java.sun.com/javase
[60] planeteclipse.net/cdt/
[61] www.zylin.com/embeddedcdt.html

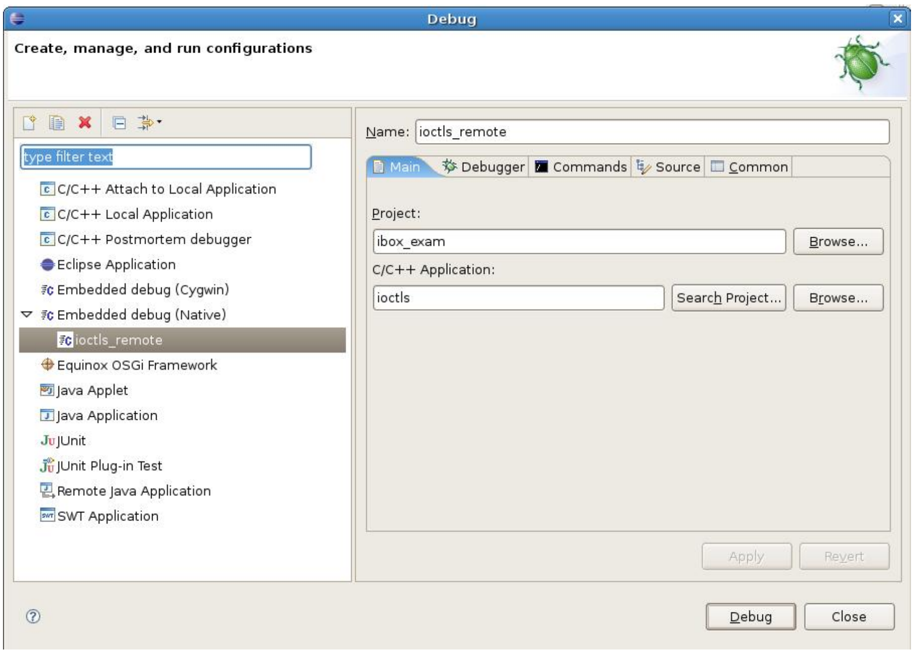
D.2. Debugging

Assuming your OpenRISC has IP address 192.168.1.66. Start your `ioctls` executable on the OpenRISC with following command:

```
gdbserver :9000 ioctls
```

On the PC side:

- open `examples` project (the .project and .cdtproject file are already created)
- Select Run->Debug
- Double click Embedded debug (Native). After that you'll get new configuration entry
- select the project corresponding to yours (see Figure 38)
- select C/C++ Application to `ioctls`
- choose Debugger tab (see Figure 39)
- GDB debugger: `arm-linux-gdb`
- choose Commands tab (see Figure 40)
- 'Initialize' commands: `target remote 192.168.1.66:9000`
- 'Run' commands: `c`
- Apply
- Debug



Debug

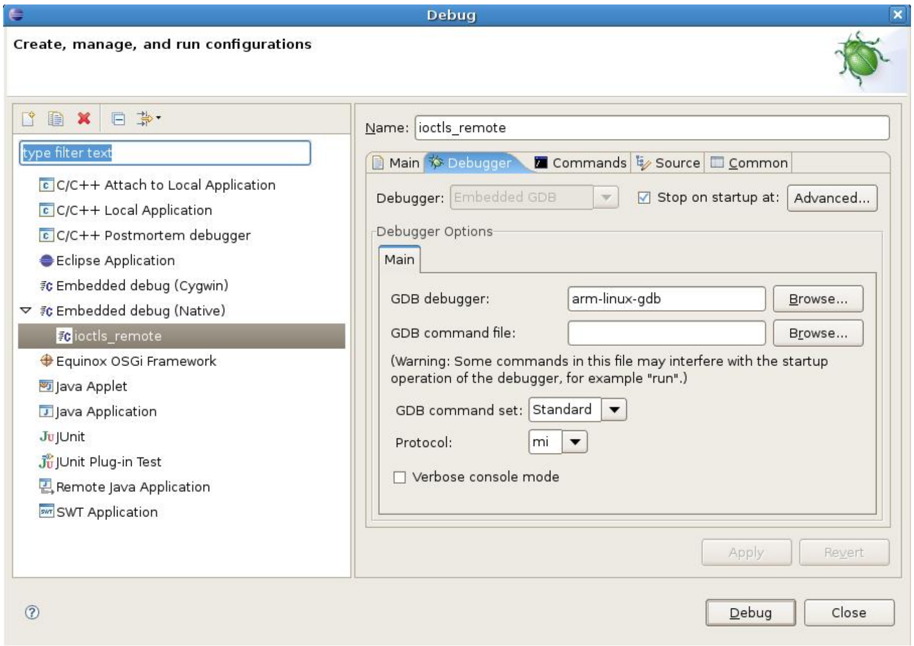
Create, manage, and run configurations

Name: ioctls_remote

Project: ibox_exam

C/C++ Application: ioctls

Figure 38: Eclipse debug: Main tab



Create, manage, and run configurations

Name: ioctls_remote

Debugger: Embedded GDB

☑ Stop on startup at:

GDB debugger: arm-linux-gdb

GDB command file:

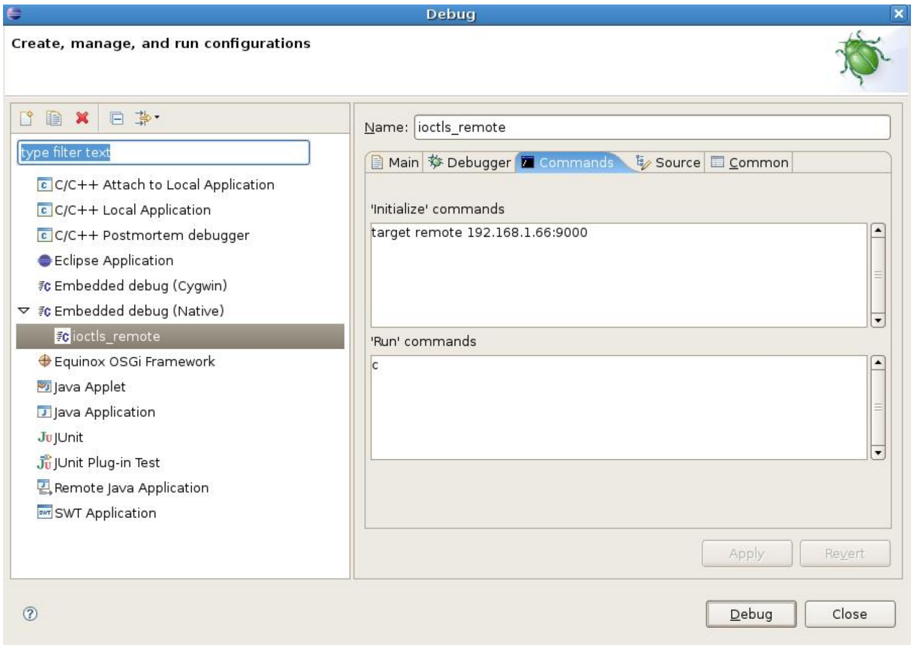
(Warning: Some commands in this file may interfere with the startup operation of the debugger, for example "run".)

GDB command set: Standard

Protocol: mi

□ Verbose console mode

Figure 39: Eclipse debug: Debugger tab



Create, manage, and run configurations

Name: ioctls_remote

'Initialize' commands

```
target remote 192.168.1.66:9000
```

'Run' commands

```
c
```

Figure 40: Eclipse debug: Commands tab

E. Cross-Compiler Tools

Name	Description
ar	Creates and updates static library files
as	Assembler
g++	C++ compiler
cpp	C preprocessor
gcc	C compiler
ld	Linker
nm	Lists symbols from object files
objcopy	Copies and translates object files
objdump	Displays information about object files
ranlib	Generates indexes to archives (static libraries)
readelf	Displays information about ELF files

Table 12: Toolchain items

F. DVDs Content

The OpenRISC will be shipped with two DVDs:

- DVD "Debian" is the Debian for ARM distribution
- DVD "Software" includes documentation, development tools and sources:
  - development: arm-linux toolchain and kernel source
  - documentation: manuals and data sheets
  - images: CF/microSD images with preinstalled Debian for ARM Linux
  - tools: various Windows tools useful for administrating and maintaining the OpenRISC like vsimgtool, netcat etc.

G. Frequently Asked Questions (FAQ)

There is dedicated website http://faq.visionsystems.de to handle the FAQ concerning OpenRISC and other products provided by VScom. The customer question will be posted to the support team for approval and if approved appears in the corresponding category.

Index