Product Information

Full information about other Arcom products is available by contacting our Website at:
www.arcom.com

Useful Contact Information

Customer Support US
Tel: 913 549 1000
Fax: 913 549 1002
E-mail: US-support@arcom.com

Customer Support Europe
Tel: +44 (0)1223 412 428
Fax: +44 (0)1223 403 409
E-mail: Euro-support@arcom.com

Sales offices
United States:
Arcom
7500 West 161st Street
Overland Park
KS 66085, USA
Tel: 913 549 1000
Fax: 913 549 1002
E-mail: US-sales@arcom.com

United Kingdom:
Arcom
Clifton Road
Cambridge
CB1 7EA, UK
Tel: 01223 411 200
Fax: 01223 410 457
E-mail: Euro-sales@arcom.com

Sales hotlines
Belgium:
Groen Nummer:
Tel: 0800 7 3192
Fax: 0800 7 3191

Germany:
Kostenlose Infoline:
Tel: 08001 824 511
Fax: 08001 824 512

Italy:
Numero Verde:
Tel: 0800 790841
Fax: 0800 780841

France:
Numero Vert:
Tel: 0800 90 84 06
Fax: 0800 90 84 12

Netherlands:
Gratis Nummer:
Tel: 0800 0221136
Fax: 0800 0221148

Whilst Arcom’s sales team is always available to assist you in making your decision, the final choice of boards or systems is solely and wholly the responsibility of the buyer. Arcom’s entire liability in respect of the boards or systems is as set out in Arcom’s standard terms and conditions of sale.

If you intend to write your own low level software, you can start with the source code supplied by Arcom. This is example code only to illustrate use on Arcom’s products. It has not been commercially tested. No warranty is made in respect of this code and Arcom shall incur no liability whatsoever or howsoever arising from any use made of the code.

© 2003 Arcom Control Systems Ltd.
Arcom Control Systems is a subsidiary of Spectris plc
All trademarks recognized.
Contents

Contents ......................................................................................................................................... 2
Revision History ............................................................................................................................ 3
Preface ............................................................................................................................................ 4
  Definitions ................................................................................................................................. 4
  Disclaimer ................................................................................................................................. 4
  Anti-Static Handling ................................................................................................................ 4
  Packaging ................................................................................................................................. 4
  Acknowledgements .................................................................................................................. 4
  Technical Support .................................................................................................................... 5
Licensing ........................................................................................................................................ 5
  Arcom Embedded Linux Components .................................................................................... 5
  OpenSSH License ...................................................................................................................... 5
Arcom Embedded Linux ................................................................................................................ 6
  Overview ................................................................................................................................. 6
  Further Support ....................................................................................................................... 7
File System .................................................................................................................................... 7
  Journaling Flash File System .................................................................................................. 7
  RAM File System .................................................................................................................... 8
Arcom Embedded Linux Configuration ......................................................................................... 8
  Default Passwords ................................................................................................................... 8
  Keyboard Mapping .................................................................................................................. 8
  Serial Port Configuration ......................................................................................................... 9
  System Startup Scripts (Init Scripts) ....................................................................................... 9
  Making an Application Run Automatically at Boot ............................................................... 9
Network Configuration ................................................................................................................ 10
  IP Address Configuration ....................................................................................................... 10
  Hostname ................................................................................................................................ 10
Wireless Network Configuration .................................................................................................. 10
  Configuring the Wireless Network Device ......................................................................... 11
  Configuring Wireless Equivalent Privacy (WEP) ............................................................... 11
  Configuring TCP/IP over WLAN ...................................................................................... 12
Arcom Embedded Linux Development Kit CD Contents ............................................................. 12
Secure Shell .................................................................................................................................. 12
  Introduction to SSH ............................................................................................................... 12
  Using SSH Commands .......................................................................................................... 13
    The ssh Command .............................................................................................................. 13
    The scp command ............................................................................................................. 14
    The sftp command ............................................................................................................ 15
  Public Key Authentication Methods .................................................................................... 15
  Port Forwarding ...................................................................................................................... 16
  Further Information About SSH ............................................................................................ 16
  Removing Legacy Services ..................................................................................................... 16
Package Management .................................................................................................................. 17
  Removing Packages .............................................................................................................. 17
Adding Packages .................................................................................................................................................. 17
The X Window System ............................................................................................................................................. 17
Window Manager .................................................................................................................................................... 17
Using a Touch Screen via a TSC1 Controller Board .......................................................................................... 18
Installing the Host Environment ....................................................................................................................... 19
Overview of the Host Environment .................................................................................................................... 19
A Simple Example................................................................................................................................................. 20
Common Open Source Build Systems and Cross Compilation ....................................................................... 20
Building Libraries .............................................................................................................................................. 20
Installing Applications and Libraries on the Target .......................................................................................... 21
Custom Build Systems and Cross Compilation ............................................................................................... 21
Complete Example............................................................................................................................................. 22
  Building the shared library ............................................................................................................................... 22
  Building the application ................................................................................................................................. 22
  Installing on the target .................................................................................................................................. 23
Compiling a Kernel .............................................................................................................................................. 23
  Overview ......................................................................................................................................................... 23
  Unpacking, Configuring and Building the Kernel .......................................................................................... 23
  Installing A New Kernel and Modules ........................................................................................................... 24
Host System Configuration ................................................................................................................................ 25
  Configuring a TFTP Server ............................................................................................................................... 25
Appendix A .......................................................................................................................................................... 26
  Software Sources ......................................................................................................................................... 26
Appendix B .......................................................................................................................................................... 28
  Useful Web Links .......................................................................................................................................... 28
Appendix C .......................................................................................................................................................... 29
  Board Specific Information ............................................................................................................................. 29
    VIPER ...................................................................................................................................................... 29
    File System Layout .................................................................................................................................. 29
    RedBoot Boot loader ................................................................................................................................. 29
    Linux Kernel .......................................................................................................................................... 29
    Default Install Flash Footprint .................................................................................................................. 30
    Touch Screen Controller (TSC1) ................................................................................................................. 30
    Intel Integrated Performance Primitives ..................................................................................................... 30
    Reinstalling the default image .................................................................................................................... 31

Revision History

<table>
<thead>
<tr>
<th>Manual</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issue B</td>
<td>8th July 2003</td>
<td>Minor editorial changes</td>
</tr>
</tbody>
</table>
Preface

Definitions
Arcom is the trading name for Arcom Control Systems Inc and Arcom Control Systems Ltd.

Disclaimer
The information in this manual has been carefully checked and is believed to be accurate. Arcom assumes no responsibility for any infringement of patents or other rights of third parties that may result from its use.

Arcom assumes no responsibility for any inaccuracies that may be contained in this document. Arcom makes no commitment to update or keep current the information contained in this manual.

Arcom reserves the right to make improvements to this document and/or product at any time and without notice.

Anti-Static Handling
This development kit contains CMOS devices, which could be damaged in the event of static electricity being discharged through them. At all times, please observe anti-static precautions when handling boards. This includes storing boards in appropriate anti-static packaging and wearing a wrist strap when handling boards.

Packaging
Please ensure that should a board need to be returned to Arcom, it is adequately packed, preferably in the original packing material.

Acknowledgements
Linux is a registered trademark of Linus Torvalds.

Red Hat™ is a registered trademark of Red Hat, Inc. This product contains copies of the Red Hat Linux 7.3 installation CD’s. These are not a product of Red Hat, Inc. and are not endorsed by Red Hat, Inc. They are a product of Arcom and we have no relationship with Red Hat, Inc. The CDs are identical in every respect to a standard Red Hat Linux 7.3 CD set.

ARM and StrongARM are registered trademarks of ARM, Ltd.

Intel and XScale are trademarks or register trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

All other trademarks and copyrights referred to are the property of their respective owners.

This product includes software developed by the University of California, Berkeley and its contributors.
Technical Support
Arcom has a team of technical support engineers who will be able to provide assistance if you have any problems with this product. Please contact

US          us-support@arcom.com     +1 913 549 1000
Europe      euro-support@arcom.com   +44 (0) 1223 412 428

All technical support relating to IBM WebSphere Device Developer and the J9 Java virtual machine should be directed to Arcom.

Licensing

Arcom Embedded Linux Components
The Arcom Embedded Linux Development Kit contains components licensed from different sources. Many of these are Open Source licenses\(^1\). Under certain circumstances you may be required to release source code to any modifications you have made to these modules if you further distribute them. Please consult Appendix A and the directory `/licenses` on the CD to ensure you are familiar with the requirements of each license.

OpenSSH License
OpenSSH is not covered by any restrictive license. It can be used for any and all purposes, and that explicitly includes commercial use. The license for OpenSSH is included in the distribution.

\(^1\) [www.opensource.org](http://www.opensource.org)
**Arcom Embedded Linux**

**Overview**

Arcom Embedded Linux (AEL) is a standard Linux distribution produced by Arcom. It is optimized to fit within the onboard flash of Arcom’s range of single board computers. AEL is based upon the standard Linux kernel and user space tools. Arcom provide free first line technical support for this product.

The minimum target footprint for AEL is a system with 16MBytes RAM and 8MBytes flash memory. Additional RAM and/or flash will allow greater functionality.

The Arcom Embedded Linux distribution consists of:

- A standard Linux kernel, built to support flash memory access
- A number of Linux device drivers (some board specific)
- System & Application libraries
- User-space Applications and Utilities

Some major components are (for versions see Appendix A):

- THTTPD web server
- FTP and Telnet servers and clients
- Bourne Again Shell (BASH)
- OpenSSH (Secure telnet and ftp replacement)
- XFree86/TinyX (on targets where graphics hardware is available)
- Many other standard Linux utilities

Arcom expects that users will want to add specific functionalities to the base system. This is achieved by adding additional libraries and/or applications compiled on a host Linux system. We supply all source code to the kernel, libraries and applications within the Arcom Embedded Linux distribution. This enables customers to completely rebuild the environment from scratch, if they so wish.

AEL is implemented in a space-constrained manner in order to support installation on space-constrained devices such as the onboard flash. However on certain target boards it is possible to install the distribution on IDE compatible media. For example the AEL CD supplied may be used to install to a hard drive or to a CompactFlash device. Consult Appendix C to see if this is possible on your particular target board.

It is not intended that an Arcom Single Board Computer be used to build applications. Arcom recommended that you design and build applications on a host Linux system and download applications to the target system, using FTP or SCP. However, should you wish to develop on the target system, this may be possible with the addition of a hard disk drive (where supported by the target hardware, see Appendix C).

Arcom recommends installing a Red Hat Linux compatible host system, such as that supplied with Arcom Development Kits, for development purposes.
Further Support

This manual provides information about the specifics of the Arcom Embedded Linux distribution as well as tutorials on the key technologies featured.

The Linux RUTE manual (found in /manuals/ on the CD) contains a much more general overview of how to use a Linux system.

The CD also contains a selection of Linux “HOWTO” documents covering common topics. These are found in /manuals/HOWTO/, the full set of “HOWTO” documents is available at www.tldp.org.

The Arcom Quickstart and Technical Manuals can also be found in the /manuals/ directory on the CD.

There are also links to several useful websites in Appendix B.

File System

The exact layout of the file system on an AEL system is dependent on the target board. In general the flash will contain one (or more) smaller partition(s) containing the boot loader and Linux kernel images\(^2\) and one large partition covering the rest of the device containing the root (/) file system. Typically the partition sizes are hard coded into the kernel binary and require rebuilding the kernel to change. On some RedBoot based target boards RedBoot’s Flash Image System (FIS) is used to provide a partition table. In this case the flash can be repartitioned from the boot loader command line. See Appendix C to see if your target supports FIS.

In addition to the flash file systems a RAM based file system will be mounted on /var/tmp.

Consult Appendix C to determine the exact layout for your board.

Journaling Flash File System

The flash is formatted using the Journaling Flash File System (JFFS2). This places a compressed file system on to the flash transparently to the user. Key features of JFFS2 are:

- Directly targeted at flash devices
- Robust
- Consistent across power failure
- No integrity scan (fsck) is required at boot time after normal or abnormal shutdown
- Explicit wear leveling
- Transparent compression

Flash partitions appear as pseudo-block devices with major number 31 which can be mounted using JFFS2.

```
mount -t jffs2 /dev/mtdblock1 mount-point
```

\(^2\) Sometimes the bootloader and kernel are combined into a single flash partition
There are a maximum of 16 partitions numbered 0-15 and corresponding to the block devices /dev/mtdblock0 thru /dev/mtdblock15. In addition each partition has a character device /dev/mtdN which is used to access advanced features of the flash device such as sector locking.

Typically the partitions containing the Linux kernel and bootloader are unformatted partitions containing a pure binary image with no file system. If the target board is booting from a device other than flash it may be possible to place a JFFS2 partition into these partitions.

No special utility is required to make a JFFS2 file system. Simply erase the whole of the partition (with eraseall -j) and mount as normal. This will cause an empty JFFS2 file system to be created.

JFFS2 partitions do not require that an integrity check (fsck) be performed on startup, either after a normal or abnormal shutdown. The supplied /sbin/fsck.jffs2 is a dummy which always succeeds and is present to simplify the boot scripts.

Although JFFS2 is a journaling file system this does not preclude the loss of data. The file system will remain in a consistent state across power failure and will always be mountable. However, if the board is powered down during a write then the incomplete write will be rolled-back on the next boot. Any completed writes will not be affected.

Refer to sources.redhat.com/jffs2 for further information.

**RAM File System**

AEL systems make use of a RAM based file system (tmpfs) mounted on /var/tmp. The contents of this file system are not preserved across reboot. The RAM file system will grow and shrink to accommodate only the size of the files it contains so there is very little overhead.

In order to prevent the RAM file system from using the whole of RAM the file system is constrained to use a maximum of 4MBytes of memory, this can be changed by editing /etc/fstab and changing the size= parameter for /var/tmp.

**Arcom Embedded Linux Configuration**

**Default Passwords**

AEL is supplied configured with two users, the super user (known as root) and a regular user arcom. The default password for both accounts is arcom. To change a users password, login as that user and run the passwd command. To add a user login as root and run the adduser command. For example:

```
adduser abc
```

**Keyboard Mapping**

By default AEL is configured for a US style keyboard. This is controlled by the file /etc/console/keymap.gz which is a symbolic link to us.map.gz in the directory /usr/lib/kbd/keymaps/i386/qwerty/.
If you wish to use another keyboard layout change this link to point to another file. For example, for a UK keyboard:

```
ln -sf /usr/lib/kbd/keymaps/i386/qwerty/uk.map.gz \\
/etc/console/keymap.gz
/etc/init.d/loadkeys start # reload keymap
```

Additional keyboard maps can be found in the `kbd` source package on the CD.

**Serial Port Configuration**

The default install of AEL is configured to run a login session on ttyS0 (COM1). This can cause problems if your application wishes to use ttyS0 since the two will conflict. Normally this will manifest itself as the serial port changing baud rate at unusual times. To remove the console edit `/etc/inittab` and remove the line which reads:

```
T0:23:respawn:/sbin/getty -L ttyS0 115200 vt100
```

Then signal init to reload its configuration

```
kill -HUP 1
```

**System Startup Scripts (Init Scripts)**

Arcom Embedded Linux uses a System V type init process. Scripts are placed in `/etc/init.d/` with symbolic links for each runlevel in `/etc/rc?.d/`. The `?` may be replaced with one of the following characters:

<table>
<thead>
<tr>
<th>Character</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>Startup</td>
<td>Run once at boot time</td>
</tr>
<tr>
<td>0</td>
<td>Halt</td>
<td>Run on system shutdown</td>
</tr>
<tr>
<td>1</td>
<td>Single</td>
<td>Run on entering single user mode</td>
</tr>
<tr>
<td>2</td>
<td>Normal</td>
<td>Serial only console</td>
</tr>
<tr>
<td>3</td>
<td>Normal</td>
<td>Serial and VGA console</td>
</tr>
<tr>
<td>4</td>
<td>Normal</td>
<td>VGA only console</td>
</tr>
<tr>
<td>5</td>
<td>Normal</td>
<td>VGA only console</td>
</tr>
<tr>
<td>6</td>
<td>Reboot</td>
<td>Run when rebooting</td>
</tr>
</tbody>
</table>

The default runlevel is level 3 for targets with graphics hardware and 2 for others.

When the runlevel changes the `K*` scripts in the `/etc/rc?.d/` directory corresponding to the new runlevel are executed in alphanumerical order (with an argument of `stop`). Then the `S*` scripts in the same directory are executed in alphanumerical order (with an argument of `start`).

**Making an Application Run Automatically at Boot**

If you want an application to run automatically on boot then you need to do the following.

1. Write a script (e.g. `someapp`) that will run your application and put it in the directory `/etc/init.d/`. 
2. Make the script executable
3. chmod +x /etc/init.d/someapp
4. Make a symbolic link in /etc/rcX.d (e.g. at runlevel 3 /etc/rc3.d/S99someapp) that points to the script in /etc/init.d. Using 99 ensures that you application will be started after all other services.
   ```bash
   ln -s /etc/init.d/someapp /etc/rcX.d/S99someapp
   ```

**Network Configuration**

**IP Address Configuration**

In the default install of Arcom Embedded Linux the network device will be configured to obtain an IP address automatically via DHCP (Dynamic Host Configuration Protocol). If you are not running a DHCP server on your network or if you wish to force a static IP address then you can reconfigure the device by editing /etc/config/network/interfaces/eth0. This file contains several fields that describe the device configuration. Table 2 describes these fields.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEVICE</td>
<td>Ethernet interface that this file configures</td>
<td>DEVICE=eth0</td>
</tr>
<tr>
<td>USEDHCP</td>
<td>Should this interface use DHCP (yes or no)</td>
<td>USEDHCP=yes</td>
</tr>
<tr>
<td>IPADDR</td>
<td>IP address of this interface (if USEDHCP=no)</td>
<td>IPADDR=192.168.1.1</td>
</tr>
<tr>
<td>NETMASK</td>
<td>Network mask for this interface (if USEDHCP=no)</td>
<td>NETMASK=255.255.0.0</td>
</tr>
</tbody>
</table>

In addition the default gateway is configured by placing `GATEWAY=[IP ADDR]` in /etc/config/network/gateway.

You can view the current Ethernet configuration by running the `ifconfig` tool and the current default gateway with `route`.

**Hostname**

The default hostname for a board is the type of board. To change this edit the file /etc/config/network/hostname. The command `hostname` can be used to examine the current hostname.

**Wireless Network Configuration**

This section briefly describes how to configure wireless networking (WLAN) to access a network via a Wireless Access Point under Arcom Embedded Linux. Further details are available on the Linux WLAN website at http://www.linux-wlan.org/.

Wireless support has been tested under Arcom Embedded Linux using a Buffalo 'AirStation' WLI-CF-S11G CompactFlash card. The AirStation card is based on the popular PRISM chip set and any PRISM2, 2.5 or 3 based card should work. Cards based on other chip sets may or may not work.
Configuring the Wireless Network Device

In order to connect to a wireless network you need to obtain several settings from your network administrator.

- **SSID** - Network identifier. e.g. "ABC-Wireless-Network"
- Radio channel to use (1-14 depending on local legislation)
- If your access point is configured to use Wired Equivalent Privacy (WEP) encryption then you will need the shared keys used by your access point.

The basic WLAN configuration is performed in the file `/etc/wlan/wlan.conf`. In this file you configure the SSID of the network you wish to use and the channels you wish to scan for access points. You should ensure that the following values are set to the desired values:

- **WLAN_DEVICES** - A space separated list of the WLAN devices you wish to configure. Typically there is only one i.e. `wlan0`.
- **WLAN_SCAN** - Set to "y" to scan for peers at startup.
- **ChannelList** - A colon (:) separated list of the channels (in hexadecimal) to scan for peers. You may scan this to the channel used by your network or you can leave it as the default, which is to scan all channels. For example use "03:0A" to scan channels 3 and 10, or "0B" to scan channel 11.
- **SSID_wlan0** - Sets the SSID for the wlan0 device. SSID_ \* should be set for each device listed in WLAN_DEVICES. For example you might set SSID_wlan0 to "ABC-Wireless-Network".

Configuring Wireless Equivalent Privacy (WEP)

WEP is available in both 64 and 128 bit variants. In order to configure WEP you will need to obtain the shared WEP keys used by your wireless access point. Each access point typically has 4 keys configured; you will need to know the keys themselves as well as which key is the default.

The privacy settings for each wireless network are configured in the file `/etc/wlan/wlancfg-SSID` where `SSID` is the value given to SSID_device in wlan.conf, for example `/etc/wlan/wlancfg-ABC-Wireless-Network`. You can create this file by copying the file `/etc/wlan/wlancfg-DEFAULT`.

---

3 Confusingly these are also known as 40 and 104 bit WEP respectively.
4 In fact, if no wlancfg-SSID file is found, then wlancfg-DEFAULT will be read instead.
dot11PrivacyInvoked Set this to “true” if you wish to enable WEP, set it to “false” otherwise.

AuthType This setting can be either "opensystem" or "sharedkey". It must be "sharedkey" to use WEP.

dot11WEPDefaultKeyID This setting specifies which of the 4 WEP keys to use by default; it must be the same as on the access point. Note that this setting is zero based whilst many access points number the keys starting from 1.

dot11WEPDefaultKeyn These settings specify the four WEP keys (n = 0..3). Each key is either 5 (for 64 bit WEP) or 13 bytes (for 128 bit WEP) colon separated bytes in hexadecimal. For example 01:A3:12:42:5E.

Configuring TCP/IP over WLAN
In order to configure the TCP/IP interface you must edit the file /etc/pcmcia/network.opts.

If your network provides a DHCP server then you can set the DHCP line to =y. Otherwise you may configure the interface statically by setting the IPADDR and NETMASK variables. Other variables such as GATEWAY and DNS_1 may be configured if you wish.

Arcom Embedded Linux Development Kit CD Contents

Table 3. Development Kit CD Contents

<table>
<thead>
<tr>
<th>Directory Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>packages</td>
<td>AEL source and binaries</td>
</tr>
<tr>
<td>examples</td>
<td>Various examples</td>
</tr>
<tr>
<td>host</td>
<td>Host Environment</td>
</tr>
<tr>
<td>licenses</td>
<td>Copies of common licenses used by the software included in the Development Kit</td>
</tr>
<tr>
<td>manual</td>
<td>Arcom and 3rd party documentation</td>
</tr>
<tr>
<td>reference</td>
<td>Board reference documentation</td>
</tr>
</tbody>
</table>

Secure Shell

Introduction to SSH

SSH (The Secure SHell) is a secure replacement for several common Internet protocols such as the Berkley r* tools (rlogin, rsh, rexec), ftp and telnet all of which have security flaws when used in a non-trusted network environment (primarily the plaintext exchange of passwords across a non-trusted network). It features several enhancements over these tools, such as:

- All traffic sent across the network (critically this includes passwords) is encrypted using
strong encryption

- Prevent spoofing and man in the middle attacks using host keys
- Tunneling of arbitrary connections through an SSH pipe, known as *port forwarding* (in particular X11 forwarding)
- Enhanced authentication methods that improve upon normal password based mechanisms

The server also benefits from SSH especially if it is running a number of services. If you use *port forwarding*, otherwise insecure protocols (for example, POP) can be encrypted for secure communication with remote machines. SSH makes it relatively simple to encrypt different types of communication normally sent insecurely over public networks.

For more information on SSH and it’s advantages and usage please visit [www.openssh.org](http://www.openssh.org).

A large number of client and server programs can use the SSH protocol, including many open source and freely available applications. Several different SSH client versions are available for almost every major operating system in use today.

Red Hat Linux includes the OpenSSH server (*openssh-server*) and client (*openssh-clients*) packages, as well as the general OpenSSH package (*openssh*) that must be installed for either of them to work. Please see the documentation for your host system for instructions on installing and deploying OpenSSH on your Red Hat Linux system.

There are also several SSH clients available for non-Linux system, including Microsoft Windows platforms, such as

- PuTTY (available in `/host/windows` on the Development Kit CD and at [http://www.chiark.greenend.org.uk/~sgtatham/putty](http://www.chiark.greenend.org.uk/~sgtatham/putty)), a Windows version of the ssh program

### Using SSH Commands

**The ssh Command**

The `ssh` command allows you to remotely login to a machine. For example to login to the machine `ael.example.net` you would use the command

```bash
ssh ael.example.net
```

The first time you login to a machine you will get a message similar to the following (with a different fingerprint).  

```
The authenticity of host ‘ael.example.net (10.2.4.14)’ can’t be established.  
Are you sure you want to continue connecting (yes/no)?
```

This message is intended to give you the opportunity to verify that the machine you are logging into is really the machine you are expecting by confirming that the host’s cryptographic key is
correct. If you wish you may verify that the fingerprint is correct before typing yes to continue logging in.

You can obtain the host fingerprint of a system by asking the administrator or by logging in at the console and running the command (there are likely to be several ssh_host_*key.pub files, examine them all)

```
ssh-keygen -l -f /etc/ssh/ssh_host_key.pub
```

Once you have verified the fingerprint of a system it will be written to ~/.ssh/known_hosts so you will not be asked this question again. If the remote hosts fingerprint changes for any reasons then ssh will give you an error, if you determine the change is legitimate (for example the remote operating system may have been reinstalled, causing the SSH server’s host key to change) then you may edit ~/.ssh/known_hosts to remove the incorrect fingerprint.

After you have accepted the remote system’s fingerprint you be asked to authenticate yourself. Normally this will be by entering your password, however see below for alternative authentication methods such as public key authentication.

The above ssh command would login to the remote machine using the current username from your local host. If you wish to login as a different user, for example as the user arcom, then you may use one of the two forms below.

```
ssh arcom@ael.example.net
ssh -l arcom ael.example.net
```

All of the commands in the SSH suite accept both the user@host and -l user forms for specifying a login user. The examples in this manual use the user@host form.

As well as allowing you to login to a remote machine, ssh can run a command on a remote machine without the need to login and run it manually. To accomplish this simply append the command to the ssh command line. For example, to examine the contents of the /bin/ directory on a remote system you might do the following.

```
ssh arcom@ael.example.net ls /bin/
```

When you use ssh to run a remote command you should be careful to quote shell meta characters which you wish to be passed to the remote system rather than processed locally.

**The scp command**

The scp command is similar to the regular cp command, except it uses the SSH protocol and allows for the source and/or destination file to be located on a remote system. You can specify a remote file by using the user@host:file syntax. The user@host part is the same as that used for the ssh command above. If you leave file blank or give a relative path (one which doesn’t start with /) then the default is the remote users home directory. If you forget the colon (‘:’) then you will copy to or from a local file named user@host, this is rarely required.

For example the following would copy my-file to /home/arcom on the remote system ael.example.net.
scp my-file arcom@ael.example.net:

The following would copy the same file to /etc/ on the remote host.

scp my-file arcom@ael.example.net:/etc/my-file

To retrieve a remote file you reverse the order of the operands. The following would retrieve /home/arcom/my-file from the remote system.

scp arcom@ael.example.net:my-file .

The sftp command
The sftp command behaves like the regular ftp command, except it uses the SSH protocol to provide strong authentication and encryption which regular ftp lacks. You can specify a user and hostname using the user@host syntax described above.

sftp arcom@ael.example.net

Public Key Authentication Methods
In addition to regular password authentication, SSH also offers public key authentication. Public key authentication works by generating a public/private key pair. The public part may be passed around freely in order to transfer it to a remote system running SSH. The private part must remain a secret since anyone possessing the private key can login to any system configured with the public key. Typically a private key designated for interactive use will have a pass-phrase that adds an extra layer of security\(^5\), however a private key which is designated for non-interactive use (such as in a script on a remote system) will not.

Public/private key pairs are generated using the ssh-keygen utility.

The ssh-keygen utility will ask for a destination (press enter to accept the default) and a pass-phrase (which may be blank). If you accepted the default location then you will now have a DSA\(^6\) public/private key pair in ~/.ssh/id_dsa and ~/.ssh/id_dsa.pub.

The file id_dsa is your private key and you should keep this safe. Your public key is in id_dsa.pub you can copy this to a remote machine and append it to the file ~/.ssh/authorized_keys to enable public key authentication on that host.

Once you have configured the remote host then you can continue to use all the SSH utilities as normal. The authorized_keys file can also be used to restrict the commands that may be run when that key is used. This can allow you to restrict the commands that can be run by an interactive script (where the key will have an empty pass-phrase). Consult the SSH documentation for more details, see “Further Information About SSH” for pointers to SSH documentation.

\(^5\) Consult the section “Further Information About SSH” for pointers to details of the ssh-agent command which will allow you to only enter your pass-phrase once per session.

\(^6\) SSH can also use RSA keys by passing rsa rather than dsa to ssh-keygen. RSA keys are stored in id_rsa and id_rsa.pub.
**Port Forwarding**

SSH has the ability to tunnel TCP/IP connections from the local machine to the remote host and vice-versa. This can be useful to provide a secure wrapper around an insecure protocol.

For example to create a secure tunnel from the local machine to a mail server running remotely on port 25 you could do the following.

```bash
ssh -L 2525:localhost:25 user@mail.example.net
```

Since the `localhost` part refers to localhost from the perspective of the remote host anyone connecting to port 2525 on your local host would be forwarded through the SSH tunnel to port 25 on mail.example.net.

You do not have to use `localhost`. For example the following would allow anyone with access to your local machine to access `mail.example.net`, which we assume is secured behind the example.net firewall.

```bash
ssh -L 2525:mail.example.net:25 firewall.example.net
```

In addition to forwarding a port on the local machine to a port on a host accessible to the remote machine you can also forward a port on the remote machine to any machine accessible from your local machine. For example to tunnel port 2525 on the remote machine to a mail server on your local network.

```bash
ssh -R 2525:mail.example.net:25 ael.example.net
```

Anyone connecting to port 2525 on `ael.example.net` would now be forwarded to `mail.example.net` on your local network.

**Further Information About SSH**

The above are simple examples of what can be achieved using SSH. Further information on the use of SSH can be found on the OpenSSH website at [www.openssh.org](http://www.openssh.org).

**Removing Legacy Services**

The default install of Arcom Embedded Linux includes telnet and ftp daemons that are made obsolete by the use of SSH. They remain in the distribution in order to maintain backwards compatibility. However, we recommend you remove them unless you explicitly require them. You can remove the telnet and ftp servers using the `dpkg` packaging tool.

```bash
dpkg -r ftpd
dpkg -r telnetd
```

See “Package Management" on page 20 for more details.

It is also possible to restrict access to these services to certain IP addresses or hosts without disabling them completely using `tcpd`. Consult the `tcpd(8)` and `host_access(5)` man pages for details by typing `man tcpd` on your host system.
Package Management

A default installation of AEL can contain several optional packages such as the J9 Java Virtual Machine, the OpenBSD Secure Shell (SSH), Web and FTP Servers. If your application does not require one or more of these and you wish to free some space in the flash then you may remove some packages. Conversely if you require a package that is not installed by default you may wish to add packages to the board.

AEL uses the dpkg package management system. A list of installed packages can be found in /var/lib/dpkg/status.

Removing Packages

Packages can be removed using the dpkg tool. For example to remove the ftphd package:

dpkg -r ftphd

Adding Packages

Additional software components in .deb packages can be found on the Development Kit CD in the /packages directory.

Transfer the required packages to a temporary directory on the target.

scp/mnt/cdrom/packages/ntp/ntpd_4.1.1-1_arm.deb
root@target.example.net:/tmp

On the target install the package

dpkg -i /tmp/ntpd_4.1.1-1_arm.deb

The X Window System

Arcom Embedded Linux makes use of the TinyX Server that is designed for embedded systems and makes use of the Linux frame buffer driver. If your target board has graphics capabilities then it will have been preinstalled with the TinyX server and a small number of other utilities.

You can start the X server on the primary frame buffer device by running the command startx. If you wish to run X at a different video mode to the board default then you can use the fbset utility to set the mode before running startx. The list of modes available is stored in the file /etc/fb.modes on the target and includes a selection of standard VGA modes and modes corresponding to various flat panels shipped by Arcom. For example to run X at 800x600 at 75Hz

```
fbset 800x600-75
startx
```

Window Manager

By default AEL comes installed with the matchbox window manager. The matchbox window manager is specifically designed to require very few resources and aims to target systems with
little screen real estate and limited input mechanisms (such as a touch screen). In order to support these aims all windows remain maximized at all times (unlike other window managers which allow you to arbitrarily choose position and size).

AEL does not come supplied with any other window managers, however you may choose to compile and install another window manager. In order to use a different window manager you can set the environment variable \texttt{X\_WINDOW\_MANAGER} to the full path to the window manager binary

\begin{verbatim}
export X_WINDOW_MANAGER=/usr/bin/another-wm
startx
\end{verbatim}

Using a Touch Screen via a TSC1 Controller Board

Included in many AEL Development Kits is a touch screen and a TSC1 touch screen controller board. If your Development Kit is supplied with these then your target board will have been preinstalled with the TSC1D touch screen controller daemon software. The X server included in the Development Kit includes an input driver that will automatically make use of the TSC1D daemon if it is running. You can start the daemon by typing \texttt{tsc1d \&} before launching X.

If you obtained your touch screen and TSC1 separately from your Development Kit then before you can use the touch screen you must calibrate it using the \texttt{tsc1cal} program. Development kits that are supplied with a TSC1 should already have been calibrated (however if you reinstall the system you may need to recalibrate). You can calibrate your touch screen by performing the following steps.

- Connect the TSC1 controller board to your target board. In the default configuration Arcom Embedded Linux runs a serial login on ttyS0 (COM1) and it is therefore recommended to connect the TSC1 controller to ttyS1 (COM2). Some Development Kits that are supplied with an ICE Enclosure containing the processor board and the TSC1 controller may use a different serial port. Please consult Appendix C on page 29 for more details.

- Ensure that the tsc1d touch screen daemon is not running by typing \texttt{killall tsc1d}

Run the touch screen calibration program \texttt{tsc1cal}. If you are not using the default serial port then pass the correct \texttt{--device /dev/ttyS\$n} parameter.

\begin{verbatim}
tsc1cal --device /dev/ttyS1 --generate /etc/tsc1d.conf
\end{verbatim}

Follow the onscreen instructions by pressing the squares as they appear. When you have finished tsc1cal will exit and display a “Calibration String”, a string of five integers separated by commas. The \texttt{--generate} option also causes it to write your configuration to the correct location.

Now the \texttt{tsc1d} has been calibrated for your touch screen. You can start it in the background by typing \texttt{tsc1d \&}, and then run X by typing \texttt{startx}.

For more information on \texttt{tsc1d} and \texttt{tsc1cal} please consult the tsc1d README file located in the tsc1d source, found in \texttt{/packages/tsc1d} on the Development Kit CD.

Arcom Embedded Linux Host Environment

In order for an application to run on a target board it must be cross-compiled on a host system (an X86 based Linux PC).
Cross compilation is a mechanism that allows applications to be linked against the libraries that will be present on the target board rather than those which are available on the host system, which may be incompatible with the target. In the case of non-X86 based targets cross-compilation also facilitates the generation of suitable machine code for the target architecture.

The Arcom Embedded Linux Host Environment provides a cross compile environment as well as a variety of other useful tools.

The host environment requires that Perl be installed on the host system. Almost every modern Linux distribution includes Perl. Consult your host distribution documentation if it is not installed.

### Installing the Host Environment

All the packages for the host environment are available on the Development Kit CD. This includes: C compiler, C++ compiler, linker, libraries and include files.

To install the host environment follow the following commands as root on your host system:

1. Mount the Development Kit CD. It is assumed in the following steps that it's mounted on
   ```
   mount /mnt/cdrom
   ```
2. Install the Arcom Embedded Linux Host Environment. This will install the host tools and libraries into `/opt/arcom` on your host system.
   ```
   /mnt/cdrom/install
   ```
3. Ensure that `/opt/arcom/bin` is in the path. Either
   a. Place it temporarily in the path for the current shell.
      ```
      export PATH="/opt/arcom/bin:$PATH"
      ```
   or
   b. Add it to default PATH for all users by adding the above command to `/etc/profile`. To add for a single user add the above line to `.bash_profile` in their home directory. A new shell will need to be started for this to take effect.

### Overview of the Host Environment

The majority of the cross compilation tools are the same as their native compilation counterparts but with an additional prefix that specifies the target system. In the case of X86 boards this prefix is `i386-linux-` and in the case of ARM and XScale based boards it is `arm-linux-`. For example the native C compiler is `gcc` and the cross C compiler for XScale boards is `arm-linux-gcc`.

The following cross compilation tools are provided:
Table 4. Tools available in the Host Environment

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ar</td>
<td>Manage archives (static libraries)</td>
</tr>
<tr>
<td>as</td>
<td>Assembler</td>
</tr>
<tr>
<td>c++, g++</td>
<td>C++ compiler</td>
</tr>
<tr>
<td>cpp</td>
<td>C preprocessor</td>
</tr>
<tr>
<td>gcc</td>
<td>C compiler</td>
</tr>
<tr>
<td>gdb</td>
<td>Debugger</td>
</tr>
<tr>
<td>ld</td>
<td>Linker</td>
</tr>
<tr>
<td>nm</td>
<td>Lists symbols from object files</td>
</tr>
<tr>
<td>objcopy</td>
<td>Copies and translates object files</td>
</tr>
<tr>
<td>objdump</td>
<td>Display information about object files</td>
</tr>
<tr>
<td>ranlib</td>
<td>Generate indexes to archives (static libraries)</td>
</tr>
<tr>
<td>readelf</td>
<td>Displays information about ELF files</td>
</tr>
<tr>
<td>size</td>
<td>List object file section sizes</td>
</tr>
<tr>
<td>strings</td>
<td>Print strings of printable characters from files (usually object files).</td>
</tr>
<tr>
<td>strip</td>
<td>Remove symbols and sections from object files (usually debugging information)</td>
</tr>
</tbody>
</table>

A Simple Example

Compiling a simple C application is simply a matter of using the cross compiler instead of the regular compiler.

```bash
arm-linux-gcc -o example -Wall -g -O2 example.c
```

Common Open Source Build Systems and Cross Compilation

The majority of Open Source software available uses configure scripts as part of their build process. Passing `--host=arm-linux` on the command line in addition to your normal options is often all that is required. (Host in this context refers to the system that the final application will run on. Newer configure scripts will accept a `--target=arm-linux` option as an alternative.)

```bash
./configure --host=arm-linux [other options]
```

Note that not all configure scripts follow this behavior. In particular, build systems not generated with the `autoconf` and `automake` tools will likely not behave as expected. Fortunately a great many open source projects do use these tools.

Building Libraries

Building libraries is similar to building applications. The libraries must be configured and built to run on the target board (i.e. the `--prefix` should be `/usr` so that libraries expect to be installed to `/usr/lib/`). However the library and headers must also be installed on the build system in `/opt/arcom/TARGET/lib` and `/opt/arcom/TARGET/bin` (where target is the cross

---

7 In these examples we will use the arm-linux cross compiler. For X86 based boards you would substitute i386-linux everywhere you see arm-linux.
compile prefix, such as arm-linux or i386-linux) so that the cross compiler and linker can use them.

With a standard automake and autoconf build system this can be achieved with

```
make install prefix=/opt/arcom/arm-linux
```

Alternatively you could install to /tmp/myapp-tmp and move the libraries and headers into /opt/arcom/TARGET by hand.

**Installing Applications and Libraries on the Target**

Installing an application or library that uses the automake tool is normally achieved by calling the install target

```
make install
```

However this would cause the application or library to be installed into the host file system, potentially causing enormous damage to the system.\(^8\) It is wise to perform the build and install of applications for the target as a non-root user on the host system, since this can prevent the worst disasters.

Fortunately automake provides a variable called DESTDIR that is used as the base directory for installation. Normally DESTDIR would be an empty string but we can define it in order to install into a temporary directory

```
make install DESTDIR=/tmp/target-install
```

Now /tmp/target-install can be pruned of any files, such as documentation and static libraries (*.a), which you do not wish to install on the target and transferred to the board. In addition you may wish to use the strip command to remove unnecessary symbols from any application binaries or libraries.

```
cd /tmp/target-install
arm-linux-strip --strip-unneeded ./usr/bin/app
arm-linux-strip --strip-unneeded ./lib/libtmp.so
```

Often, build systems also have an install-strip target that behaves exactly like the install target but also calls the correct strip utility for you.

**Custom Build Systems and Cross Compilation**

Unfortunately there are projects that use build systems that differ from the de-facto standard discussed above. There is no simple recipe for building applications with non-standard build systems. Consult any build instructions, README files etc and study the build system makefiles\(^9\) etc.

Normally you can edit the makefile to prefix all references to tools mentioned in

---

\(^8\) Consider the consequences of replacing libc on an X86 system with a libc cross compiled for an ARM system.

\(^9\) The RUTE Linux Tutorial (on the development kit CDROM) contains sections on the use of make and makefiles.
Table 4 with the correct cross compilation prefix. Often a makefile will define variables such as CC and CXX to contain the C and C++ compilers respectively. Generally setting environment variables to override these before running make will meet with some success.

CC=arm-linux-gcc  CXX=arm-linux-c++  make

As described above you must take care to install to a temporary directory so as not to overwrite your build system’s native libraries and binaries. A custom build system may not make use of the DESTDIR variable. Again, consult any documentation you may have and examine the makefile for install targets or similar.

**Complete Example**

Included on the Development Kit CD is a trivial example application that utilizes a shared library. The two source tarballs may be found in /examples/c/trivial/trivial-app-1.0.tar.gz and /examples/c/trivial/libtrivial-1.0.tar.gz. This example uses a standard autoconf\[10\]/automake\[11\]/libtool\[12\] build system so the steps presented here will be mostly applicable to real applications using the same build system.

**Building the shared library**

Unpack the library distribution

```
tar xzf libtrivial-1.0.tar.gz
    cd libtrivial-1.0
```

Configure the library. Note the use of the --host option.

```
./configure --prefix=/usr --host=arm-linux
    make
```

Now install the library on the host (build) system.

```
    make install prefix=/opt/arcom/arm-linux
```

Install for the target (this will be transferred to the target later)

```
    make install-strip DESTDIR=/tmp/trivial-app
```

**Building the application**

Unpack the application distribution

```
tar xzf trivial-app-1.0.tar.gz
    cd trivial-app-1.0
```

Configure the application. Again, note the use of the --host option.

```
./configure --prefix=/usr --host=arm-linux
    make
```

---

Install for the target

```
make install-strip DESTDIR=/tmp/trivial-app
```

**Installing on the target**
Generate the tarball to be installed based on the installed files. For a real application you may wish to remove files (such as documentation) that are not required on the target from the temporary directory before building the tarball.

```
cd /tmp/trivial-app
tar czvf ../trivial-app.tar.gz *
cd ..
```

Transfer the installation tarball to the target board (target.example.net) with scp

```
scp trivial-app.tar.gz root@penguin.example.net:/tmp
```

On the target board (as root)

```
cd /
tar xzvf /tmp/trivial-app.tar.gz
```

Update the shared library cache

```
ldconfig
```

Run the application

```
trivial-app
```

And it should display

```
This is a trivial application.
This is a trivial function in a trivial shared library.
```

---

**Compiling a Kernel**

**Overview**

This section contains basic instructions on how to unpack, configure, build and install a new Linux kernel onto your target board. For more detailed instructions and troubleshooting please consult the Linux Kernel HOWTO on the Development Kit CD at manual/HOWTO/Kernel-HOWTO.pdf.

An Arcom Embedded Linux kernel source tree is supplied on the Arcom Development Kit CD in the /packages/linux directory. It consists of an upstream tarball and a patch containing Arcom's modifications consult Appendix A for the correct versions and filenames for your target board. The patch contains essential support for the on board Flash and other peripherals.

In order to cross compile the kernel you will need the cross compilation environment installed. This is described in “Installing the Host Environment”.

**Unpacking, Configuring and Building the Kernel**

The kernel must be recompiled on your host system.

Mount the CD
Copy the kernel source from the CD (to for example /usr/local/src/)

```
cp /mnt/cdrom/packages/linux/ \ 
   linux_VERSION.orig.tar.gz /usr/local/src

cp /mnt/cdrom/packages/linux/linux_REVESION.diff.gz \ 
   /usr/local/src
```

Unpack the source code

```
cd /usr/local/src

tar xzf linux_VERSION.orig.tar.gz
```

Apply the patch

```
cd kernel-VERSION

cat ../linux_REVISION.diff.gz | patch -p1
```

Select the default configuration

```
make TARGET_config
```

Where TARGET is the name of the board. Typically ‘viper’, ‘gxx’ or ‘pegasus’. Consult Appendix C to determine the correct default configuration for your target board.

Configuring the kernel

```
make menuconfig
```

Navigate the menu system to configure any user changes then save and exit.

Build the source dependency information

```
make dep
```

Build the kernel image

```
make clean

make zImage # arm & XScale platforms only

make bzImage # X86 platforms only
```

If your kernel configuration includes modules then build the modules. (All of the supplied configurations are modular).

```
make modules
```

**Installing A New Kernel and Modules**

If you are using modules then you will need to install them. Modules can be installed to an alternate prefix on your host by adding `INSTALL_MOD_PATH=<prefix>` to the modules_install line. This should be used to install the module tree to a temporary directory before copying to the target. Without this option they will be installed to `/lib/modules/VERSION`, which could potentially overwrite your host system’s modules and could potentially damage your host system.

```
make modules_install INSTALL_MOD_PATH=/tmp/ael-modules
```

Your new kernel image can be found relative to the build directory as `arch/arm/boot/zImage` (for arm or XScale targets) or `arch/ARCH/boot/bzImage` (for X86 targets). It should be installed using either the `flashboot` program or the RedBoot FIS Commands. Please consult Appendix C.
For a more detailed guide to building a custom kernel please consult the Linux Kernel HOWTO on the CD /manuals/HOWTO/Kernel-HOWTO.pdf.

Host System Configuration

Configuring a TFTP Server

Several target boards support the use of TFTP to load images onto the board (primarily those targets which use RedBoot as the bootloader). In order to use this functionality you must have a TFTP server available on your network. This section outlines how to configure a TFTP server on a Red Hat Linux system; other Linux distributions will be similar.

The first step is to install the tftp-server package. This should be available on your distribution media or from the redhat.com website. It makes to install the tftp package as well since it provides a tftp client that you can use to test your server. Once you have obtained the tftp-server package you can install it by logging in as the root user and typing

```
rpm –ivh <RPM filename>
```

By default the RPM package installs the TFTP server with a configuration that leaves it disabled. To enable it edit the file `/etc/xinetd.d/tftp` and modify the line which reads

```
disable = yes
```

To read

```
disable = no
```

A typical example of the file is

```
service tftp
{
    socket_type = dgram
    protocol = udp
    wait = yes
    user = root
    server = /usr/sbin/inet.tftpd
    server_args = -s /tftpboot
    disable = no
    per_source = 11
    cps = 100 2
}
```

This specifies that the TFTP server root will be `/tftpboot`, meaning that all filenames will be relative to this directory.

Now that you have configured tftp correctly then you need to restart xinetd

```
/etc/init.d/xinetd restart
```

You can test the tftp server by creating a file in `/tftpboot` and then downloading it using the tftp client

```
echo “Hello, This is a test file” > /tftpboot/Hello
tftp <IP Address>
tftp> get Hello
Received 28 bytes in 0.0 seconds
tftp> quit
```
Note: Some Red Hat systems install a default firewall that blocks ports 67 and 68 for UDP traffic. TFTP requires that these ports are open to clients. You can examine your firewall configuration using the command `ipchains -L`. Please consult your distribution documentation if you need to modify the firewall rules.

## Appendix A

### Software Sources

The following lists some of the components included in Arcom Embedded Linux.

The location on the supplied CD is given for the component source and any Arcom patches applied to that source code. Relevant licenses are also noted.

The source for a component consists of an upstream source tarball named `PACKAGE_VERSION.orig.tar.gz` and a patch containing Arcom’s modifications named `PACKAGE_VERSION-REVISION.diff.gz`. Where `VERSION` is the upstream version number and `REVISION` is Arcom’s revision of the component. A small number of packages do not include a patch since there are no modifications to upstream (this is common when Arcom is the upstream author).

Source and binaries for a given component can normally be found on the Development Kit CD in the directory `/packages/PACKAGE/`. Source code to any open source components of Arcom Embedded Linux that are not included on the CD will be supplied by Arcom upon request.

<table>
<thead>
<tr>
<th>Software Component</th>
<th>Version</th>
<th>Location of source code and binaries</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>License(s)</td>
</tr>
</tbody>
</table>

Copies of the more widespread licenses are included on the CD in the directory `/licenses`. They are also listed in Appendix C.

For less widespread licenses please consult the source distribution for that package.

### Table 5. Major Components of Arcom Embedded Linux.

<table>
<thead>
<tr>
<th>Software Component</th>
<th>Version</th>
<th>Location of source code and binaries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux Kernel</td>
<td>See Appendix C</td>
<td>/packages/linux</td>
</tr>
<tr>
<td>GNU Libc Library</td>
<td>2.2.4</td>
<td>/packages/glibc</td>
</tr>
<tr>
<td>GNU GCC</td>
<td>3.2.1</td>
<td>/packages/gcc-installer¹³</td>
</tr>
</tbody>
</table>

¹³ The GCC installer creates packages from libraries installed by the AEL host environment. The binaries are contained in `/host/RPMS/i386/gcc-arm-linux-cross-VERSION.rpm` and the corresponding source is in `/host/SRPMS/gcc-arm-linux-cross-VERSION.rpm`
<table>
<thead>
<tr>
<th>Package</th>
<th>Version</th>
<th>License</th>
<th>Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bash</td>
<td>2.05</td>
<td>GPL</td>
<td>/packages/bash</td>
</tr>
<tr>
<td>Modutils</td>
<td>2.4.12</td>
<td>GPL</td>
<td>/packages/modutils</td>
</tr>
<tr>
<td>TinyLogin</td>
<td>0.80</td>
<td>GPL</td>
<td>/packages/tinylogin</td>
</tr>
<tr>
<td>BusyBox</td>
<td>0.60.2</td>
<td>GPL</td>
<td>/packages/busybox</td>
</tr>
<tr>
<td>Linux Ftpd</td>
<td>0.17</td>
<td>BSD</td>
<td>/packages/linux-ftpd</td>
</tr>
<tr>
<td>Netkit Ftp</td>
<td>0.17</td>
<td>BSD</td>
<td>/packages/netkit-ftp</td>
</tr>
<tr>
<td>thttpd</td>
<td>2.19</td>
<td>BSD like</td>
<td>/packages/thttpd</td>
</tr>
<tr>
<td>SysVInit</td>
<td>2.78</td>
<td>GPL</td>
<td>/packages/sysvinit</td>
</tr>
<tr>
<td>pppd</td>
<td>2.4.0</td>
<td>Mixed GPL/ BSD</td>
<td>/packages/ppp</td>
</tr>
<tr>
<td>Sysklogd</td>
<td>1.4</td>
<td>GPL</td>
<td>/packages/sysklogd</td>
</tr>
<tr>
<td>XFree86/TinyX</td>
<td>4.1.0</td>
<td>MIT</td>
<td>/packages/xfree86</td>
</tr>
</tbody>
</table>
Appendix B

Useful Web Links

General Linux Information
www.linux.org

Linux Kernel
www.kernel.org

ARM (and XScale) Linux Kernel
www.arm.linux.org.uk

GNU GCC
www.gnu.org/software/gcc

GNU GCJ
www.gnu.org/software/gcc/java

Linux Documentation project
www.tldp.org

The BSD License
www.opensource.org/licenses/bsd-license.html

GNU General Public License (GPL)
www.gnu.org/copyleft/gpl.html

GNU Lesser General Public License (LGPL)
www.gnu.org/copyleft/lgpl.html

The MIT License
www.opensource.org/licenses/mit-license.html

RUTE: The Rute User’s Tutorial and Exposition
http://rute.sourceforge.net

eCos and RedBoot
http://sources.redhat.com/ecos/docs-latest
http://sources.redhat.com/ecos/docs-latest/ref/redboot.html
Appendix C

Board Specific Information

VIPER
The VIPER is an ultra low power PC/104 compatible single board computer based on the Intel 400MHz PXA255 XScale processor. The PXA255 is an implementation of the Intel XScale micro-architecture combined with a comprehensive set of integrated peripherals including, a flat panel graphics controller, DMA controller, interrupt controller, real time clock and multiple serial ports. The VIPER board offers a long list of features making it ideal for power sensitive embedded communications and multimedia applications.

File System Layout

Table 6. VIPER File System Layout

<table>
<thead>
<tr>
<th>Mount Point</th>
<th>Location</th>
<th>File System</th>
<th>Mount Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>/</td>
<td>mtd3</td>
<td>jffs2</td>
<td>Read/Write</td>
<td>15.25 Megabytes</td>
</tr>
<tr>
<td>/var/tmp</td>
<td>RAM</td>
<td>tmpfs</td>
<td>Read/Write</td>
<td>4 Megabyte Limit</td>
</tr>
</tbody>
</table>

The onboard flash is partitioned using RedBoot FIS as follows:

Table 7. Default Flash Layout

<table>
<thead>
<tr>
<th>Partition #</th>
<th>Block Device</th>
<th>Size (k)</th>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>/dev/mtdblock0</td>
<td>124</td>
<td>0</td>
<td>Partition Table</td>
</tr>
<tr>
<td>1</td>
<td>/dev/mtdblock1</td>
<td>4</td>
<td>124k</td>
<td>RedBoot configuration</td>
</tr>
<tr>
<td>2</td>
<td>/dev/mtdblock2</td>
<td>640</td>
<td>128k</td>
<td>Kernel</td>
</tr>
<tr>
<td>3</td>
<td>/dev/mtdblock3</td>
<td>15616</td>
<td>768k</td>
<td>Root File System</td>
</tr>
</tbody>
</table>

RedBoot Boot loader
The VIPER uses the RedBoot boot loader. For more details please consult the board technical manual and eCos Reference Manual (both in /manuals/ on the Development Kit CD).

Linux Kernel
The version of the Linux kernel used on the VIPER is 2.4.19-rmk7-pxa1-arcom1. This is kernel version 2.4.19 plus patches for ARM (rmk7) and XScale (pxa1) support from www.arm.linux.org.uk and support for Arcom’s boards (arcom1).

The kernel sources can be found on the Development Kit CD in /packages/linux.

    Upstream       linux_2.4.19.orig.tar.gz
    Arcom Patch    linux_2.4.19-rmk7+pxa1+arcom1+viper.1.diff.gz

Once you have unpacked and patched the kernel source you can configure the default VIPER kernel with make viper_config.

The kernel is stored in a RedBoot FIS partition called ‘kernel’. To update the kernel you must first load the kernel into RAM using Xmodem or TFTP then write it to the flash.

Load kernel via TFTP
1. Configure TCP/IP network stack
   ip –l <VIPER IP ADDRESS> -h <TFTP SERVER IP ADDRESS>

2. Load the kernel image
   load –r –b 0x200000 zImage

Load kernel via Xmodem

1. Configure RedBoot to receive a file via Xmodem
   load –r –b 0x200000 –m xmodem

2. Now use your serial terminal to send the zImage using Xmodem. Under minicom use Ctrl-A S.

Once the kernel has been loaded you can use the fis create command to update the kernel flash partition. The location of the kernel in RAM and the size is remembered from the load command

   fis create kernel

Answer yes to overwrite the existing kernel.

Default Install Flash Footprint
The basic AEL install uses about 7.9 Megabytes of physical flash. This leaves about 8.1 Megabytes free on a 16M board. This is about 16 Megabytes assuming a typical 2:1 compression ratio. Additional space can be made available by removing unused packages (see “Package Management” on page 20).

The default flash image is available on the Development Kit CD as /images/ArcomEmbeddedLinuxViper-3.1.img.

You can reinstall it using the following steps.

Configure TCP/IP network stack
   ip –l <VIPER IP ADDRESS> -h <TFTP SERVER IP ADDRESS>

Load image via TFTP
   load –r –b 0x200000 ArcomEmbeddedLinuxViper-3.1.img

Now that the flash image is loaded into RAM you can write it to the flash using the following command.

   fis write -f 0x60000000 –b 0x200000 –l 0x1000000

Touch Screen Controller (TSC1)
If your VIPER ICE Enclosure was supplied with a touch-screen then the TSC1 supplied will be attached to /dev/ttyS2 (COM3).

Intel Integrated Performance Primitives
Intel supplies a set of code modules optimized for the XScale processor.

These include basic functionality including:

- signal, image, speech and audio processing
• vector manipulation
• matrix math

In addition there exist modules providing primities useful in specific application areas such as audio, video and speech codecs (MP3 (MPEG-1 Audio Layer 3), MPEG-4, H263, JPEG, GSM-AMR and G723.1) and cryptography (DES, Triple DES, SHA1 and RSA).

For more information on Intel IPP please see

http://developer.intel.com/design/pca/application4processors/swsup/IPPv30.htm

Reinstalling the default image
The default development kit image can be found in /images/ArcomEmbeddedLinuxViper-3.1.img on the development kit CD. This is a raw 16M flash image that can be loaded via TFTP using Red Boot and burnt into flash. Before you can do this you should configure a TFTP server, see “Configuring a TFTP Server” on page 25.

Once you have a tftp server running you can copy the flash image into the /tftpboot/ directory on the TFTP server

```
cp /mnt/cdrom/images/ArcomEmbeddedLinuxViper-3.1.img /tftpboot/
```

Now power on the VIPER board and press Control-C to drop to a RedBoot prompt. Once here you need to configure an IP address for the board

```
ip -l <IP address> -h <TFTP server IP address>
```

Next load the image; this may take some time depending on your network

```
load -r -b 0x200000 ArcomEmbeddedLinuxViper-3.1.img
```

Now unlock the flash and write the image

```
fis unlock -f 0 -l 0x20000
fis write -f 0 -b 0x200000 -l 0x1000000
```