

E.L.L.K. II

**The
Embedded Linux
Learning Kit Version 2
User's Guide**

**Doug Abbott
Intellimetrix**

Copyright Notice

The text and graphics in this manual, its cover, the E.L.L.K. II CD-ROM and its artwork are copyrighted and protected from misuse under local and international laws. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means electronic, mechanical, photocopying, recording, or otherwise without prior written permission of the author.

Limitation of Liability, Intended Use

The Embedded Linux Learning Kit, the kit, is intended as a tool for learning to use Linux in an embedded context. Intellimetrix makes no warranty, representation or guarantee regarding the merchantability, suitability or fitness of the kit, or any of its individual components, for any particular purpose. All software is supplied “as is.”

Updates

In the interest of improving quality and maximizing value to the customer, this manual and the contents of the associated CD-ROM are subject to update and revision at any time. Electronic updates will be posted at www.intellimetrix.us/downloads.htm.

Revision History

7/17/10 – Initial release

9/26/10 – Minor corrections and updates

Contact Information

The author may be contacted at:

Intellimetrix
2311 Ranch Club Rd. #409
Silver City, NM 88061
Phone: +1 575-590-2788
Email: doug@intellimetrix.us
Web: www.intellimetrix.us

Embedded Linux Learning Kit

Preface	Error! Bookmark not defined.
What's in the kit?.....	Error! Bookmark not defined.
Resources.....	Error! Bookmark not defined.
1. Getting Started	4
The Host Development Environment.....	4
Linux installation.....	4
Installing the kit software.....	4
The Terminal Emulator, minicom	5
Networking	7
The target.....	8
What can go wrong?.....	9
The bootloader	10
Resources.....	12
Linux distributions	12
2. Flash Memory and File Systems	Error! Bookmark not defined.
Flash Memory – NAND and NOR.....	Error! Bookmark not defined.
Root file system in flash	Error! Bookmark not defined.
3. Eclipse Integrated Development Environment	Error! Bookmark not defined.
Introduction	Error! Bookmark not defined.
Getting Started	Error! Bookmark not defined.
Java Runtime.....	Error! Bookmark not defined.
Starting Eclipse	Error! Bookmark not defined.
The C Development Environment (CDT).....	Error! Bookmark not defined.
Creating a Project	Error! Bookmark not defined.
The target execution environment	Error! Bookmark not defined.
Remote debugging with GDB.....	Error! Bookmark not defined.
Debugging with CDT	Error! Bookmark not defined.
Debugging with DDD	Error! Bookmark not defined.
Resources.....	Error! Bookmark not defined.
4. Working with Hardware	Error! Bookmark not defined.
ARM I/O Architecture.....	Error! Bookmark not defined.
Accessing I/O from Linux	Error! Bookmark not defined.
5. A Real-World Application	Error! Bookmark not defined.
6. A Thermostat	Error! Bookmark not defined.
Host workstation as debug environment.....	Error! Bookmark not defined.
7. Posix Threads	Error! Bookmark not defined.
Creating a Linux Process – the fork() function	Error! Bookmark not defined.

Introducing Threads.....	Error! Bookmark not defined.
Thread Attributes	Error! Bookmark not defined.
Synchronization—Mutexes	Error! Bookmark not defined.
The thermostat with threads	Error! Bookmark not defined.
Linux Device Drivers	Error! Bookmark not defined.
The Low Level I/O API.....	Error! Bookmark not defined.
Changes required in thermostat.c.....	Error! Bookmark not defined.
Debugging threaded programs.....	Error! Bookmark not defined.
Resources	Error! Bookmark not defined.
8. Networking.....	Error! Bookmark not defined.
Sockets	Error! Bookmark not defined.
The Server Process.....	Error! Bookmark not defined.
The Client Process	Error! Bookmark not defined.
Socket Attributes.....	Error! Bookmark not defined.
A Simple Example.....	Error! Bookmark not defined.
The Server	Error! Bookmark not defined.
The Client.....	Error! Bookmark not defined.
Try it Out.....	Error! Bookmark not defined.
A Remote Thermostat.....	Error! Bookmark not defined.
Multiple monitor threads.....	Error! Bookmark not defined.
Resources	Error! Bookmark not defined.
9. An Embedded Web Server	Error! Bookmark not defined.
Background on HTTP.....	Error! Bookmark not defined.
A Simple embedded web server	Error! Bookmark not defined.
Dynamic web content.....	Error! Bookmark not defined.
Forms and the POST method.....	Error! Bookmark not defined.
Build and Try it.....	Error! Bookmark not defined.
A “Real” Embedded Web Server.....	Error! Bookmark not defined.
Resources	Error! Bookmark not defined.
10. Configuring and Building the Kernel	Error! Bookmark not defined.
The kernel source tree.....	Error! Bookmark not defined.
Configuring the kernel	Error! Bookmark not defined.
The patch utility.....	Error! Bookmark not defined.
make xconfig	Error! Bookmark not defined.
Building the kernel	Error! Bookmark not defined.
Resources	Error! Bookmark not defined.
11. Booting the kernel over the network.....	Error! Bookmark not defined.
Setting up a TFTP server.....	Error! Bookmark not defined.
Creating a bootable image.....	Error! Bookmark not defined.
Booting the image.....	Error! Bookmark not defined.

U-boot Scripting.....	Error! Bookmark not defined.
12. The LCD Driver	Error! Bookmark not defined.
So what's a device driver?.....	Error! Bookmark not defined.
Linux Device Drivers	Error! Bookmark not defined.
The Low Level I/O API.....	Error! Bookmark not defined.
Driver internal structure.....	Error! Bookmark not defined.
The console and framebuffer devices	Error! Bookmark not defined.
ANSI terminal escape sequences	Error! Bookmark not defined.
Thermostat display.....	Error! Bookmark not defined.
ncurses library.....	Error! Bookmark not defined.
Qtopia.....	Error! Bookmark not defined.
Resources.....	Error! Bookmark not defined.
13. BusyBox.....	Error! Bookmark not defined.
Resources.....	Error! Bookmark not defined.
14. The Final Steps: Linux Initialization and Flash File Systems.....	Error! Bookmark not defined.
Linux Initialization.....	Error! Bookmark not defined.
Create a YAFFS File System and Load to NAND flash.....	Error! Bookmark not defined.
Starting from Scratch – the “Supervivi” boot loader....	Error! Bookmark not defined.
“JTAGing” the NOR flash.....	Error! Bookmark not defined.
15. The Next Steps.....	Error! Bookmark not defined.
Web Resources	Error! Bookmark not defined.
Mini2440 Resources	Error! Bookmark not defined.
Commercial sources of embedded Linux.....	Error! Bookmark not defined.
Appendix A: Bootloader Commands.....	Error! Bookmark not defined.
Information Commands.....	Error! Bookmark not defined.
Memory Commands.....	Error! Bookmark not defined.
NOR Flash Memory Commands.....	Error! Bookmark not defined.
NAND Flash Memory Commands.....	Error! Bookmark not defined.
Execution Control Commands	Error! Bookmark not defined.
Download Commands	Error! Bookmark not defined.
Environment Variable Commands	Error! Bookmark not defined.
Environment Variables	Error! Bookmark not defined.
Appendix B: Target board memory map	Error! Bookmark not defined.
General Purpose I/O Port B (GPB)	Error! Bookmark not defined.
General Purpose I/O Port G (GPG).....	Error! Bookmark not defined.

1. Getting Started

The Host Development Environment

In many cases, the target computer on which an embedded application runs is severely limited in terms of its computing resources. It probably doesn't have a full-sized display or keyboard. It may have at most a few megabytes of mass storage in the form of a flash file system, hardly enough to contain a compiler much less a decent integrated development environment (IDE). Thus embedded development usually requires at least two computers—the target on which the embedded program will run and a development workstation on which the embedded program is written and compiled. Before we begin working with our embedded Linux environment, we'll have to set up an appropriate development workstation.

Any modern PC will work just fine as a development host. Minimum requirements are: a Pentium class processor, 1 Gbyte of RAM for graphical operation, and at least 10 Gbytes of disk for a “workstation” class Linux installation. Of course, more RAM and disk are always better. You will need an asynchronous serial port, which can be a USB to serial converter, and a network interface.

Linux installation

Since this is not a beginners' guide, we won't say much about installing Linux itself. Pick your favorite Linux distribution—Fedora, Ubuntu, Debian, Suse, whatever. Any fairly recent distribution should work just fine. I'm partial to Fedora and currently run Fedora 11 as a guest machine under Sun Microsystems' VirtualBox.

I'm also partial to graphical desktop environments and my favorite is KDE. Throughout this chapter and to a lesser extent in the rest of the manual, you'll find references to the “start menu”. In Fedora KDE this is a stylized “f” icon in the lower left corner of the screen that behaves like the Windows start menu. In recent Fedora releases, the start menu is organized as a matrix of cutsie icons. I prefer the traditional list style, which you can select by right clicking the “f” icon and selecting **Switch to Classic Menu Style**.

Installing the kit software

1. Insert and mount the CD ROM, normally at `/media/EmbeddedLinux` for Fedora distributions. Note that the CD will probably auto mount.
2. `cd` to your home directory
3. Use the `su` command to become root user
4. Execute `/media/EmbeddedLinux/install_tools`

The script assumes the CD is mounted at `/media/EmbeddedLinux`. If your mount point is different, execute:

`/<your mount point>/install_tools <your mount point>`

The `install_tools` script installs the ARM cross-compilation tool chain along with a couple of other tools and a root file system for the target board. It also changes the permissions on some directories so that you as a normal user can write to them. When `install_tools` finishes, it will prompt you to exit the root user shell and execute `/media/EmbeddedLinux/install`.

The `install` script assumes that the directories `/usr/local/eclipse`, and `/usr/src/arm` don't exist and it doesn't check to see if the software it's trying to install already exists on your system. It copies and unpacks a number of files into different locations. The ARM cross-compilation tool chain is installed in `/usr/local/arm/4.3.2/bin`, the Eclipse IDE is in `/usr/local/eclipse`, and the Linux kernel sources are installed in `/usr/src/arm/linux-2.6.32.2-mini2440`. Add `/usr/local/arm/4.3.2/bin` to your `PATH`.

When the installation is finished, you'll find three new subdirectories under your home directory:

- `busybox-1.13.3/` – source distribution for the BusyBox utility that is described in detail in chapter 14
- `factory_images/` -- the original kernel, root filesystem, and boot loader images as currently stored in the target's flash. It may be necessary to re-flash one or more of these if something goes drastically wrong.
- `target_fs/` -- this is the target board's root file system that will be mounted over NFS. `target_fs/home/` contains sample code in three subdirectories:
 - `include/` -- some header files needed for access to the target hardware
 - `src/` -- source code for the examples in this manual
 - `Linux Sample Code/` -- examples provided by the Mini2440 vendor

The Terminal Emulator, minicom

`minicom` is a fairly simple Linux application that emulates a dumb RS-232 terminal through a serial port. This is what we'll use to communicate with the target board.

There are a number of `minicom` configuration options that we need to change to facilitate communication with the target.

In a shell window as root user, enter the command `minicom -s`. If you're running `minicom` for the first time you may see the following warning message:

WARNING: Configuration file not found. Using defaults

You will be presented with a configuration menu. Select Serial port setup. By default, `minicom` communicates through the modem device, `/dev/modem`. We need to change that to talk directly to one of the PC's serial ports Type "A" and replace the word "modem" with either "ttyS0" or "ttyS1", where `ttyS0` represents serial port COM1 and `ttyS1` represents COM2. However, if your host only has USB ports and you're using a USB to serial converter, the correct device is most likely "ttyUSB0."

Type “E” followed by “T” to select 115200 baud. Make sure both hardware and software flow control are set to “no”. These are toggles. Typing “F” or “G” will toggle the corresponding value between “yes” and “no”. Figure 1.1 shows the final serial port configuration.

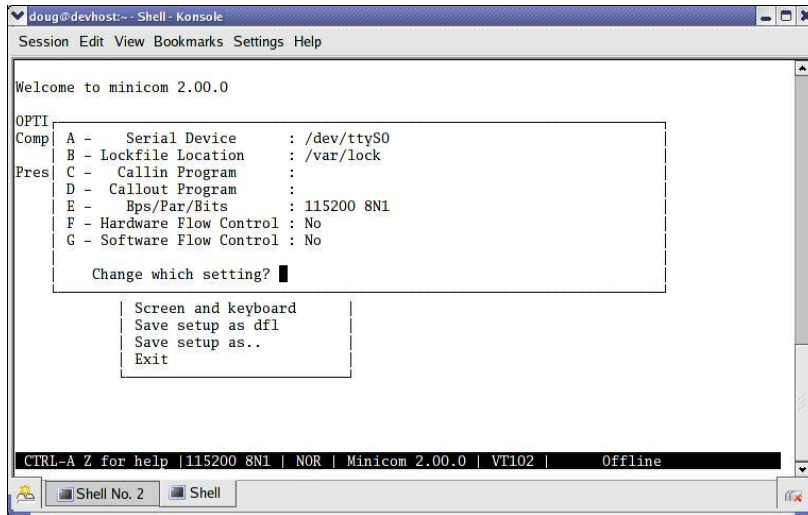


Figure 1.1: minicom serial port setup

Type <Enter> to exit Serial port setup and then select Modem and dialing. Here we want to delete the modem’s Init string and Reset string since they’re just in the way on a direct serial connection. Type “A” and backspace through the entire Init string. Type “B” and do the same to the Reset string.

Type <Enter> to exit Modem and dialing and then select Screen and keyboard. Type “B” once to change the Backspace key from BS to DEL.

Type <Enter> to exit Screen and keyboard. Select Save setup as dfl (default). Then select Exit.

You will probably have to change the permissions on the file /dev/ttyS0 (or /dev/ttyS1 if that’s the one you chose in Serial port setup) to allow the group and world to read and write the device. And of course, you must be root user to do this.

Modern Linux distributions such as the recent Fedora releases run a daemon called udev that dynamically builds the /dev directory at each boot based on information in “rules” files. This means that the permissions on /dev/ttyS0 (S1) revert to the default at boot time. A simple way around this

without having to understand `udev` is to put the appropriate `chmod` command in the script file `/etc/rc.d/rc.local`, which is the last initialization script executed at boot up.

Networking

Your workstation is probably configured to get a network address via DHCP (Dynamic Host Configuration Protocol). But in this case, to keep things simple, we're going to specify fixed IP addresses for both ends.

If you're using KDE, there's a nice graphical menu for changing network parameters. From the Start menu, select Administration -> Network Configuration¹. You'll be asked for the root password. In the Network Configuration dialog box, select the Devices tab (it's the default), select `eth0` and click Edit. In the Ethernet Device dialog, General tab, unselect the "Automatically obtain IP address settings with..." box. Now enter the fixed IP address. Network address 192.168.1 is a good choice here because it's one of a range of network addresses reserved for local networks. Select node 2 for your workstation. Enter the Subnet Mask as shown in Figure 1.2. You may also want to set the Default Gateway Address and DNS nodes if you're connected to a network.

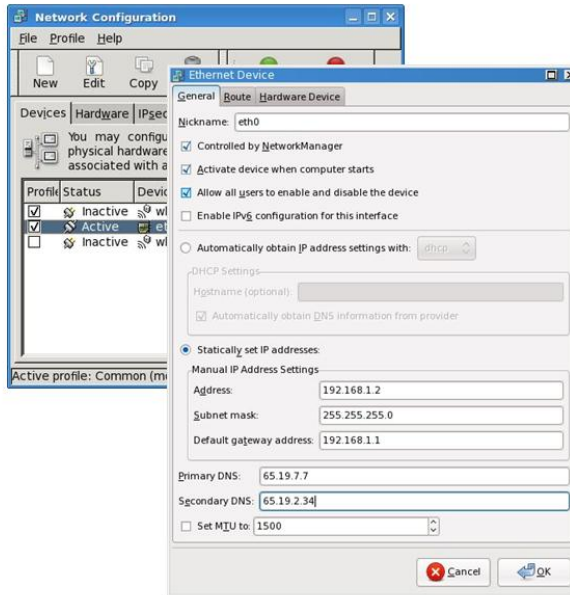


Figure 1.2: Host network configuration

¹ Configuration options tend to move around with each new release of Fedora. This is where it's located in Fedora 11.

Alternatively you can just go in and directly edit the network device parameters file. Network configuration parameters are found in `/etc/sysconfig/network-scripts/` where you should find a file named something like `ifcfg-eth0` that contains the parameters for network adapter 0. You might want to make a copy of this file and name it `dhcp-ifcfg-eth0`. That way you'll have a DHCP configuration file for future use if needed. Now open the original file with an editor (as root user of course). It should look something like listing 1.1a. Delete the line `BOOTPROTO=dhcp` and add the four new lines shown in listing 1.1b. Strictly speaking, the Gateway entry is only necessary if the workstation is connected to a network with Internet access.

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
```

```
DEVICE=eth0
ONBOOT=yes
IPADDR=192.168.1.2
NETMASK=255.255.255.0
GATEWAY=192.168.1.1
BROADCAST=192.168.1.255
```

Listing 1.1a: ifcfg-eth0

Listing 1.1b: revised ifcfg-eth0

We'll use NFS (Network File System) to download executable images to the target. That means we have to "export" one or more directories on the workstation that the target can mount on its file system. Exported directories are specified in the file `/etc/exports`. Initially this file is present but empty. As root user, open it with an editor and insert the following line:

```
/home/<your_home_name> (rw, sync)
```

This makes your home directory visible on the network where other nodes can mount parts of it to their local file system.

Finally, we have to start the NFS server on the workstation. It may turn out that NFS is started automatically on boot up. Go to the start menu, Administration -> Services Service Management. Scroll down to nfs. If it's not running or enabled then enable and start it. If it is running, you'll want to restart it so that it rereads the `exports` file.

That's it for configuring the workstation. Let's move on to the target board.

The target

If you haven't done so already, unpack the target board, power supply and cables. The LCD panel is mounted to the board with standoffs making a very compact configuration. Unfortunately, it covers the LEDs, A/D pot, and pushbuttons. Unscrew the panel from the standoffs and use the supplied nuts to reattach the panel and its own PCB.

Connect the serial cable to the workstation port you selected in minicom setup and plug the other end into the target. Connect the network port using either the supplied crossover cable to connect directly between the workstation and the target, or a standard Ethernet cable to connect to a hub. Figure 1.3 shows the layout of the target board. Check that the Boot Select switch is positioned away from the edge of the board. This selects NAND boot.

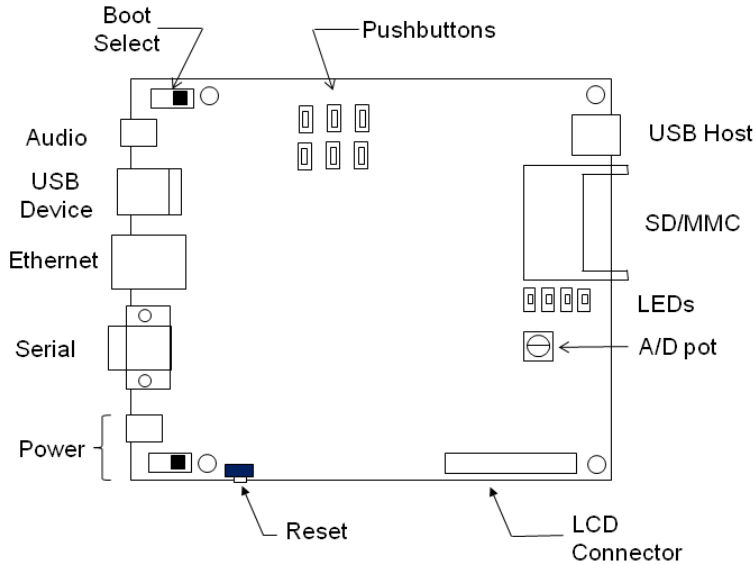


Figure 1.3: Target board layout
(not to scale)

Start up minicom in a terminal window. Now power up the target by moving the power switch toward the edge of the board. The target's bootloader boots the Linux kernel from NAND flash and mounts the root filesystem from the host workstation over NFS. Log in as root user. There is no password. Try a few simple Linux shell commands to prove that it really is working.

What can go wrong?

It's not unusual to encounter difficulties when bringing up an embedded target board such as the ELLK. The most common problems fall into two broad categories. The first is the serial port. Make sure the baud rate is set correctly. This is generally not a problem because 115 kbaud seems to be the default for minicom. A more common problem is not turning off hardware flow control.

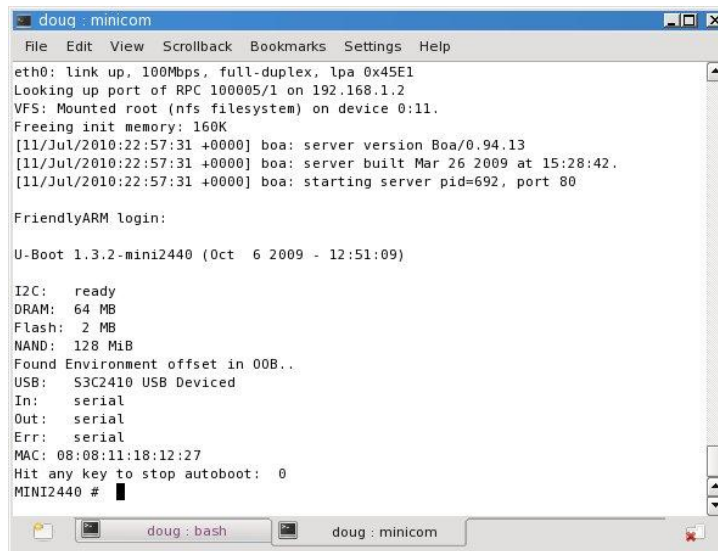
Common networking problems include having Security Enhanced Linux enabled and/or the firewall turned on. This will prevent the target from NFS mounting its root file system. As a starting point, disable SELinux and turn off the firewall. Later on you can configure either of these features to allow NFS mounting but still provide some level of protection.

Make sure the IP address is correct. And double check that you're using the right network cable – a crossover cable if you're connecting directly between the target and workstation, or a straight through cable if you're going through a hub or switch.

The bootloader

The target employs a 2-stage boot process that we'll explore in detail later. The final stage is an Open Source bootloader called Das U-Boot or just U-Boot that resides in NAND flash. Press the reset button, and quickly type a key or two. I usually hit the space bar.

Instead of executing its auto boot sequence, U-Boot presents its own prompt after printing some basic information about the board as illustrated in Figure 1.4.



```
doug : minicom
File Edit View Scrollback Bookmarks Settings Help
eth0: link up, 100Mbps, full-duplex, lpa 0x45E1
Looking up port of RPC 100005/1 on 192.168.1.2
VFS: Mounted root (nfs filesystem) on device 0:11.
Freeing init memory: 160K
[11/Jul/2010:22:57:31 +0000] boa: server version Boa/0.94.13
[11/Jul/2010:22:57:31 +0000] boa: server built Mar 26 2009 at 15:28:42.
[11/Jul/2010:22:57:31 +0000] boa: starting server pid=692, port 80

FriendlyARM login:

U-Boot 1.3.2-mini2440 (Oct 6 2009 - 12:51:09)

I2C:   ready
DRAM:  64 MB
Flash: 2 MB
NAND:  128 MiB
Found Environment offset in 00B..
USB:   S3C2410 USB Deviced
In:    serial
Out:   serial
Err:   serial
MAC:   08:08:11:18:12:27
Hit any key to stop autoboot:  0
MINI2440 #
```

Figure 1.4: U-Boot startup screen

Enter the command `printenv` to see the environment variables. There are a lot of them. Here are some of the more significant environment variables:

```
bootdelay=3
```

```
baudrate=115200
bootcmd=nand read 32000000 kernel 267000; bootm 32000000
stdin=serial
stdout=serial
stderr=serial
ethaddr=08:08:11:18:12:27
bootargs=console=ttySAC0,115200 root=/dev/nfs
nfsroot=192.168.1.2:/home/target_fs ip=192.168.1.50
```

Note that the last two lines above are actually one line in U-Boot. The default variables are:

bootdelay – seconds to wait before proceeding with autoboot

baudrate – for the serial port

bootcmd – command executed if the autoboot sequence is not interrupted

bootargs – command line passed to the kernel when it boots

stdin, stdout, stderr – direct standard Linux file descriptors to the serial port

ethaddr – sets the board’s MAC address. Note that all ELLK boards are delivered with the same default MAC address

bootargs – command line arguments passed to the Linux kernel when it boots. Information in **bootargs** includes:

- console device and, because it’s a serial port, baud rate
- root filesystem type, in this case NFS
- where the NFS root filesystem is located
- the IP address of this device

Normally, a MAC address is “burned” into an Ethernet device when it is manufactured using numbers obtained from the IEEE that guarantee that every MAC address in the world is unique. In practice, all that’s required is that every MAC address on a particular *network* be unique.

If you have two or more ELLK boards on the same network, you’ll have to change the MAC address on all but one of them. You can change the address by executing the U-Boot command:

```
setenv ethaddr nn.nn.nn.nn.nn.nn
```

where “nn” represents any arbitrary hex digits.

In the **bootargs** variable, the parameter **nfsroot** shows that the root filesystem is found on the host workstation at **/home/target_fs**. If you check your **/home** directory, you’ll find that **target_fs** is in fact a link to **/home/<your_home_dir>/target_fs**. It’s done this way so that the boot parameters are independent of your user name and home directory. **target_fs** is located in your home directory because that’s where you’ll be developing code examples.

U-Boot has a very extensive command set, much of which is detailed in Appendix A. For now, type **help** to see a complete list.

Note incidentally that U-boot does print a line that says “Hit any key to stop autoboot:”. The default delay is three seconds, which seems sufficient to get in before the autoboot takes off.

Resources

Linux distributions

There are something over 300 Linux “distributions” floating around the Internet. Many of them are designed to serve some specific purpose and, in fact, there are a number of embedded distributions. The website <http://iso.linuxquestions.org/> claims to have 1120 versions of 346 “distros” available for download.

In most cases, downloads are available as ISO image files. An ISO file is an exact image of a CD or DVD. Note that burning an ISO image to a CD is not the same as copying a file. Many popular CD burning programs have an option for burning ISO images. Some of the more popular distributions include:

Fedora -- the Fedora project is the community-supported successor to the free Red Hat Linux distributions that ended with Red Hat 9.

Debian -- the Debian project is known for its adherence to the Unix and free software philosophies, and for its abundance of options — the current release includes over fifteen thousand software packages for eleven computer architectures.

Suse -- the SUSE distribution is currently owned by Novell and is available in both an open source version and a commercial version. The link here is to openSUSE.

Ubuntu -- Ubuntu was developed in South Africa based on Debian GNU/Linux and has become quite popular in recent years. It emphasizes usability, regular releases, ease of installation, and freedom from legal restrictions. The name comes from the Zulu and Xhosa concept of *ubuntu*, which means “humanity towards others”.

Gentoo -- Gentoo is designed to be modular, portable, easy to maintain, flexible, and “optimized for the user's machine.” All tools and utilities are built from source code, but for convenience, several large packages are also available as precompiled binaries.

Various versions of all these distributions are available from LinuxQuestions.org.