QuickStart Instructions

phyCORE-LPC3250 Rapid Development Kit for Linux

Document No: L-715e_1
Edition: January 28, 2011

	EUROPE	NORTH AMERICA
Address:	PHYTEC Technologie Holding AG Robert-Koch-Str. 39 D-55129 Mainz GERMANY	PHYTEC America LLC 203 Parfitt Way SW, Suite G100 Bainbridge Island, WA 98110 USA
Ordering Information:	+49 (800) 0749832 order@phytec.de	1 (800) 278-9913 sales@phytec.com
Technical Support:	+49 (6131) 9221-31 support@phytec.de	1 (800) 278-9913 support@phytec.com
Fax:	+49 (6131) 9221-33	1 (206) 780-9135
Website:	http://www.phytec.de	http://www.phytec.com

Table of Contents

List of Tables

1 Introduction

This QuickStart provides you with the tools and know-how to get the Linux operating system running on the phyCORE-LPC3250 System on Module. This QuickStart shows you how to do everything from installing the appropriate tools and source, to building custom kernels, to deploying the OS, to exercising the software and hardware.

Please refer to the phyCORE-LPC3250 Hardware Manual for specific information on board-level features such as jumper configuration, memory mapping and pin layout for the phyCORE-LPC3250 System on Module and Carrier Board.

1.1 Rapid Development Kit Documentation

This rapid development kit (RDK) includes the following electronic documentation on the enclosed phyCORE-LPC3250 Rapid Development Kit CD:

- The PHYTEC phyCORE®-LPC3250 Hardware Manual

- Controller User's Manuals and Datasheets

- This QuickStart Instruction guide

1.2 Professional Support Packages Available

PHYTEC backs up our Rapid Development Kits with a Start-Up Guarantee. We invite you to make use of our free Technical Support concerning installation and setup of QuickStart demos until any kit start-up problem you might encounter is resolved. For more in-depth questions, we offer technical support packages for purchase. Please contact our sales team at sales@phytec.com to discuss the appropriate support option if professional support beyond installation and setup is required.

1.3 Overview of this QuickStart Instruction

This QuickStart instruction gives a general Rapid Development Kit description, as well as all of the necessary instructions required to deploy embedded Linux on the phyCORE-LPC3250 in both a standalone configuration, and a networked development configuration. This QuickStart takes you from deploying pre-built images provided in the kit on an SD Card, to building a custom image and deploying everything over the network suitable for the often changing development environment. The structure of this QuickStart is as follows:

Chapter 1 - Introduction: This chapter provides an overview of this QuickStart instruction, system requirements, and kit contents.

Chapter 2 - Getting Started: This chapter guides you through installing the PHYTEC Spectrum CD software and documentation, preparing the phyCORE-LPC3250 hardware and interfacing a host PC, and booting the pre-built images included on the SD Card that comes with the kit.

Chapter 3 - Getting More Involved: This chapter provides more in-depth instructions on installing the Linux build tools, building custom images, deploying images via NAND Flash, SD/MMC Card, TFTP, and NFS.

Appendix A: This chapter provides some additional information that you may find useful while working with Linux on the phyCORE-LPC3250 SOM. This chapter includes links to the NXP Linux BSP page, links to learning more about the Linux Target Image Builder, NAND Flash layout, using DHCP, and enabling Ethernet in the pre-built images.

1.4 Conventions used in this QuickStart

The following is a list of the typographical conventions used in this book.

Bold	Commands or other text that should be typed literally by the user
Bold Italic	File and directory names
Italic	Field and window names or titles, menu items, and other terms that correspond to items on the PC desktop
<u>Underline Blue</u>	Hyperlinks to documents, webpage URLs, or FTPs
<bracket red>	Replace these entries with the text applicable to your system. For instance this may be a directory name, or IP address. The entire entry, including brackets should be replaced by the applicable text.
Fixed Width	Source code listings and examples

Pay special attention to the notes set apart from the text with the following icons:

SUCCESS	Completion of an important part of this QuickStart.
TIP	Useful supplementary information and tips about the topic.
TIME	Estimated completion time for the following chapter.
CAUTION	This warning will help you avoid potential problems.
RESOLVE	Helpful information for troubleshooting and resolving potential problems.

1.5 Kit Contents

The following PHYTEC hardware components are included in the phyCORE-LPC3250 Linux Rapid Development Kit (part number KPCM-040-Linux) and are necessary for completing the instructions in this QuickStart:

- The phyCORE-LPC3250 SOM (PCM-040)

- The phyCORE-LPC3250 Carrier Board (PCM-967)

- Bare PCB Expansion Board (PCM-988) (optional)

- AC adapter supplying 5V/3.2A, center positive

- Ethernet patch cable

- Serial cable (RS-232)

- SOM extraction tool

- Hard copy schematics (optional)

- SD Card with pre-built U-Boot, Linux, and Root File System images

- PHYTEC Kit CD

As an addition the KLCD-011 QVGA (320x240) TFT LCD can be purchased as an add-on for this kit should you require graphics.

1.6 System Requirements

The following system requirements are necessary to complete this QuickStart without errors and problems. Deviations from these requirements may suffice, or may have other work-arounds.

- A network enabled Linux host system running OpenSUSE 11.1 or later[1]

- SD card reader operational under Linux[2]

- Active internet connection

1.7 The PHYTEC Kit CD

The PHYTEC Kit CD contains documentation, datasheets, and demo files for various OSes and develoment environments. For Linux, all demo images are provided on the kit SD card. The latest demo images can be obtained from   http://www.phytec.com/products/linux/bsp-LPC3250.html The remaining components for the Linux kit are all provided from http://ltib.org and are installed as part of this QuickStart Instruction.

If you are interested in evaluating the WinCE BSP, or the Keil or IAR BSPs you can install the PHYTEC Kit CD on a Windows system by running the *setup.exe*. Please refer to the respective QuickStart for each BSP by visiting http://www.phytec.com/products/sbc/ARM-XScale/phyCORE-ARM9-LPC3250.html or by navigating to the QuickStarts folder on the CD under *\PHYTEC\phyCORE-LPC3250\Documentation\QuickStarts*.

---

1. Although other Linux distributions will likely work, this QuickStart is based on working with the openSUSE distribution. The Fedora Core 9 distribution has also been tested to work with some of this QuickStart.
2. If you do not have SD card access under Linux then mounting the root file system on an SD card will not be possible. Some parts of this QuickStart require copying files from Linux onto an SD card. Most of these operations use a FAT32 partition. This can be handled with a Windows machine if SD card access under Linux is unavailable.

2 Getting Started

In this chapter you will set up the phyCORE-LPC3250 Rapid Development Kit to connect to your host Linux PC. Afterwards you will boot the pre-built images that come shipped on the SD card by flashing them to the SOM. At the end of this chapter you will have a fully functioning Linux system on the phyCORE-LPC3250 SOM.

15 min

2.1 Rapid Development Kit Setup

To prepare for the subsequent sections of this QuickStart document, complete the following steps:

1. Insert the SD card that came in the kit in to the SD/MMC card slot connector X15 on the phyCORE-LPC3250 Carrier Board.

2. If you ordered the KLCD-011 add-on, plug this into the LCD connector X26 at this time.

3. Plug the phyCORE-LPC3250 SOM into connector X1 on the Carrier Board if it is not already plugged into the board.

4. Connect the kit supplied serial cable from a free serial port on your Linux host to the bottom DB9 connector P1A on the Carrier Board. This is the UART5 communications channel with the LPC3250 at RS-232 levels.

5. Connect the kit supplied Ethernet cable from the Ethernet connector X7 on the Carrier Board to your network hub, router, or switch. If you do not have an Ethernet connection available you can postpone this step until Chapter 3.7. You will be able to boot the Linux kernel without the need for Ethernet connectivity in this chapter of the QuickStart.

6. Start your favorite terminal software (such as minicom or komport) on your Linux host and configure it for 115200 baud, 8 data bits, no parity, and 1 stop bit (8n1) with no handshake.

7. Plug the kit supplied 5V power adapter into the power input connector X10 on the Carrier Board. You will see something similar to the following in the terminal output window:

```
Phytec LPC3250 board
Build date: Dec 4 2008 09:45:09

phy3250>
```

8. This is the output of the Stage 1 Loader. Test that your terminal session is working correctly by typing **info** at the Stage 1 Loader prompt and hitting **Enter**. You will see a bunch of system information dumped to the terminal window if everything is working correctly. If you are unable to type anything at the Stage 1 Loader prompt, check that you have setup the terminal software correctly per step 6 and that handshaking has been disabled.

You are now ready to begin booting the pre-built images that come with your SD card.

2.2 Booting the Pre-built Images

The following pre-built images are included on the SD card contained in the kit.

1. ubt-qs.bin -- Das U-Boot 1.3.3-1, universal bootloader

2. uimg-qs -- Linux kernel with LCD support

3. rootfs -- Linux root file system in jffs2 format suitable for placement in NAND Flash (Apache Web Server installed).

In this part of the QuickStart U-Boot, the Linux kernel image, and the root file system images will all be written to the on-board NAND Flash.

1. Return to the Stage 1 Loader prompt in your terminal output window and type **ls** to get a list of files on the SD card. You should see the files listed above.

2. Execute the following command to load the U-Boot binary image from the SD card into SDRAM at 0x83fc0000:

   ```
   phy3250>load blk ubt-qs.bin raw 0x83fc0000
   ```

3. Execute the following command to save the U-Boot image to NAND Flash:

   ```
   phy3250>nsave
   ```

4. Execute the following command to configure the auto boot settings to automatically fetch the U-Boot binary from flash into SDRAM at 0x83fc0000 and then execute it:

   ```
   phy3250>aboot flash raw 0x83fc0000
   ```

5. Execute the following command to change the Stage 1 Loader prompt to "phy3250-linux>" and the boot delay to 2 seconds:

   ```
   phy3250>prompt phy3250-linux> 2
   ```

6. Execute the following command to load the Linux kernel image into SDRAM at 0x80100000:

   ```
   phy3250-linux>load blk uimg-qs raw 0x80100000
   ```

7. Now type **boot** at the Stage 1 Loader prompt. You will see something similar to the following output:

   ```
   U-Boot 1.3.3 (Feb 10 2009 - 15:44:53)

   DRAM:  64 MB
   NAND:  64 MiB
   *** Warning - bad CRC or NAND, using default environment

   In:    serial
   Out:   serial
   Err:   serial
   uboot>
   ```

8. The bad CRC warning is normal. This is indicative that the current environment variables that U-Boot attempted to read from NAND Flash were bad, or non-existent. Since the board is new the environment variables haven't been written to NAND yet, and are non-existent. This CRC warning will come up until a **saveenv** command is executed in U-Boot. This will be done shortly.

9. The next step is to use U-Boot to write the Linux kernel to flash. Execute the flash_kernel command to write the kernel to flash:

```
uboot> run flash_kernel
```

10. Reboot the board by pressing the reset button S1 on the Carrier Board. Press **any key** in your terminal window to stop the auto boot process, arriving at a Stage 1 Loader prompt.

Make sure you wait for the autoboot message to come up in the terminal window before hitting **any key** to stop the boot process. This is especially important when the boot jumper (JP21) is installed and the board attempts to boot over UART5 after reset. Pressing a key during the UART5 boot will fool the boot loader into thinking a UART boot is occurring, causing an invalid boot process, and the board will reset.

11. Load the Linux file system image into SDRAM and boot U-Boot again:

```
phy3250-linux>load blk rootfs raw 0x80100000
phy3250-linux>boot
```

Note that the loading process will take several seconds. No status messages are printed to the screen during this time.

12. U-Boot will come up again and stop at the prompt. Execute the flash_rootfs command to write the root file system image to flash:

```
uboot> run flash_rootfs
```

13. Configure the boot command to tell U-Boot to boot the Linux kernel image from NAND Flash:

```
uboot> setenv bootcmd ${boot_nand}
```

14. Now save the environment variables to NAND Flash. This will get ride of that CRC error that comes up every time U-Boot boots and tries to read the non-existent environment variables in NAND Flash.

```
uboot> saveenv
```

15. Reboot the board again by pressing the system reset button S1 on the Carrier Board. If everything was done correctly the board should boot completely into Linux, arriving at a root prompt. Note that the first time the board is booted it will takes a little while for the SSH server to generate new keys. Subsequent boots should be faster.

You have successfully booted the pre-built images included on the SD card.

3 Getting More Involved

In this chapter you will install the tools required to build an embedded Linux system for the phyCORE-LPC3250 SOM. You will then be guided through building the boot loader, Linux image, and root file system. Later on in the chapter you will learn how to deploy the images to the SOM using a variety of methods. This chapter will also guide you through configuring your host Linux PC for rapid development via TFTP and NFS. When you are finished with this chapter you will know how to build Linux images from scratch, add a variety of useful software components, and deploy the images using a variety of methods.

1-3 hr

3.1 Installing the Linux Target Image Builder (LTIB)

The Linux Target Image Builder (LTIB) is a tool that integrates the build and configuration of the mutliple software packages and components required for a typical embedded Linux distribution. LTIB greatly reduces the complexity of gathering, configuring, and building all of these required components.

For the phyCORE-LPC3250 Linux package, the primary components are:

- Das U-Boot -- the universal bootloader

- Linux kernel -- the compressed kernel image

- Linux file system -- the filesytem containing modules, executables, configuration files, etc...

- arm-(vfp)-linux-gnu toolchain -- cross-toolchain built to support the ARM926EJ-S processor (with VFP)

LTIB is reponsible for building U-Boot, the Linux kernel, and the root file system. The ARM cross-toolchain is not built by LTIB. LTIB retrieves it as a separate, pre-built component.

Installing LTIB is a very easy process. To get started you must download the ***netinstall*** script from http://ltib.org. This can be done using the *wget* command. Open a new shell window and execute the following sequence of commands in your base home directory:

1. Change to your base home directory by typing **cd ~** at the shell prompt.

2. Create a subdirectory for LPC3250 related development and change into the new directory:

   ```
   ~> mkdir lpc3250
   ~> cd lpc3250
   ```

3. Download the netinstall script:

   ```
   ~/lpc3250> wget http://ltib.org/pages/netinstall.txt
   ```

4. Run the **netinstall** script:

```
~/lpc3250> perl netinstall.txt
```

RESOLVE

The netinstall script gets LTIB from a CVS repository. Before the script can attempt to run, the CVS package needs to be installed:

```
~/lpc3250> sudo zypper install cvs
```

5. You will be prompted for an install path. For the purposes of this document the install path is */home/<username>/lpc3250/ltib/.* To avoid confusion later on in this QuickStart you should use the exact same directory. If you decide to use something else, remember to change all references to this path when appropriate.

SUCCESS

From here LTIB will download from ltib.org and install automatically. Depending on the speed of your host machine and internet connection, this may take several minutes, or several hours. Be aware that LTIB will not show its progress while it is installing; be patient. After install has completed the build will return to a prompt. You are now ready to begin configuring LTIB for a phyCORE-LPC3250 Linux distribution build.

3.2 Building U-Boot, the Linux Kernel, and Root File System

LTIB completely automates building U-Boot, the Linux kernel, and the root file system. LTIB allows configuration of the bootloader, kernel, and installed packages through a menu driven system. To begin the build process:

1. Return to the shell window used to install LTIB. Change directories to the LTIB install path and execute LTIB like so:

```
~/lpc3250> cd ltib
~/lpc3250/ltib> ./ltib
```

2. A menu will come up instruction you to select the platform. Use the cursor keys to navigate the menu. Choose the *Phytec 3250 board with the NXP LPC32XX SoC* from the list. Select *Exit* and say *Yes* to saving the new configuration.

3. The LTIB configuration menu will appear allowing you to configure all of the build options LTIB controls. Select *Exit* and say *Yes* to saving the new configuration.

RESOLVE

Before the ltib script will run, you must install gcc:

```
~/lpc3250> sudo zypper install gcc
```

LTIB depends on some additional packages which will be listed when you attempt to run the ltib script after gcc has been installed. Install all packages that are listed. Multiple packages can be installed with one command in the following format:

```
~/lpc3250> sudo zypper install <package1> <package2> <package3>
```

You will likely encounter a build failure notifying you that you do not have sudo permissions to execute rpm commands as root without a password. For the LTIB install, this is required. Follow the instructions given in the error output to enable sudo permissions to execute rpm commands. You must run visudo as root. You can either do this by using sudo:

`~/lpc3250> sudo /usr/sbin/visudo`

And type in the root password. Or you can do this by opening a root shell and typing:

`/home/<username>/lpc3250 # visudo`

Either method will allow you to edit the sudoers file.

For this QuickStart, the following edit was added to the very end of the sudoers file:

`<username> ALL = NOPASSWD: /bin/rpm, /opt/ltib/usr/bin/rpm`

Now restart the install process by executing the `perl netinstall` script again.

LTIB will begin to download and build U-Boot, the kernel, and root file system. The entire process may take several minutes, or several hours depending on the speed of your host system and internet connection. The build process may pause at several points as package sources are downloaded from the internet. This is normal.

If all goes well LTIB will return to a prompt with a *Build Succeeded* message. Once this process has finished U-Boot, the Linux kernel, and the root file system have all been built and assembled and are ready to be deployed on the phyCORE-LPC3250 Rapid Development Kit. If you received a Build Succeeded message you can skip the following paragraphs and table and continue to the next section.

If the build process fails then restart the LTIB configuration menu and check that the default settings are correct. See Table 3-1 below for default configuration options. The table only contains the items which must be checked. To restart the LTIB configuration menu, type the following:

`~/lpc3250/ltib> ./ltib --configure`

Note that LTIB will only enter the configuration menu the first time after install when typing just **./ltib** at the prompt. Thereafter typing just **./ltib** at the prompt will rebuild modified source. To enter the configuration menu the command line switch **--configure** must be used, as shown above.

Table 3-1. LTIB Configuration Menu Default Settings

LTIB Option	Setting
System features	`[*] cache target rpms`
Target C library type	`(X) glibc`
C library package	`(X) from toolchain only`

Toolchain component options	`[*] libc shared libraries`
	`[*] c++ shared libraries`
	`[*] libgcc*.so*`

Toolchain	`(X) ARMv5te gcc-4.1.2,Multi-lib,` `gnueabi/glibc-2.5-nptl-3`
Enter any CFLAGS for gcc/g++	`-02 -fsigned-char -mfloat-abi=softfp` `-mfpu=vfp`
bootloader choice	`(X) u-boot 2009.03 for the Phytec 3250` `board`
u-boot flags	`<empty>`
kernel	`(X) Linux 2.6.34 for LPC3250/Phytec` `3250`
Always rebuild the kernel	`[*] (checked)`
Produce cscope index	`[ ] (unchecked)`
Kernel preconfig	`linux-2.6.34-phy3250.config`
Include kernel headers	`[ ] (unchecked)`
Configure the kernel	`[ ] (unchecked)`
Leave the kernel sources after build-ing	`[ ] (unchecked)`

Package list	`Check only:`
	`[*] alsa-utils`
	`[*] busybox`
	`[*] dropbear ssh client/server`
	`[*]     non-blocking random device`
	`[*]     disable reverse host lookups`
	`[*]     disable X11 forwarding`
	`[*] module dependencies`
	`[*] mtd-utils`
	`[*] Skeleton base files`

Target System Configuration Options	`Check only:`
	`[*] start inetd`
	`[*] start dropbear ssh server`
	`[*] start networking`
	`[*] start syslogd/klogd`

Target Image Generation Options	`Target image: (NFS only)`

3.3 Setting up TFTP

TFTP is a "trivial" file transfer protocol used to transfer files across networks. Although TFTP is not absolutely required for working with Linux on the phyCORE-LPC3250, it is highly recommended during the building and development phase. TFTP allows the phyCORE-LPC3250 to be

configured to always load the latest build of the Linux kernel image from your host machine, without the need to continually reflash the target board or update kernel image in some other time consuming fashion.

The setup of the TFTP server will only be described for the openSUSE 11.3 distribution, although it will likely apply to earlier versions of openSUSE. Because of the vast number of Linux distributions out there, describing the setup of a TFTP server for other Linux distributions is out of the scope of this document. Please refer to one of the many guides available on the internet for your particular distribution.

To begin setup:

1.   Start *YaST* and enter the root password.

2.   Under the *Software* section on the left in the *YaST Control Center*, click *Software Management*.

3.   In the *Search* field, type **tftp** and click *Search*.

4.   Check *tftp* and *yast2-tftp-server* under the *Package* list.

5.   Click *Accept* in the lower right hand corner to install the packages.

Make sure you resolve dependencies required to complete the package install if prompted to do so.

6.   After installation has completed, close the Software Management window, then close and re-open YaST to refresh the list of YaST components.

7.   Under the Network Services section in the YaST Control Center, click TFTP Server.

8.   Click the *Enable* radio button and leave the *Boot Image Directory* as */tftpboot*.

9.   Check the *Open Port in Firewall* checkbox to prevent the firewall from blocking the TFTP server access.

10. Click OK to accept the configuration.

Your TFTP server should now be enabled and ready to use.

3.4 Setting up NFS

NFS allows you to mount a remote network disk as if it were a local disk thus providing easy sharing of files over a network. For this QuickStart NFS is used to access the root file system on the host Linux machine. That is, the root file system for the phyCORE-LPC3250 SOM will actually be located on the remote host Linux machine. This enables easy access and modificatios to the root file system during development. Although NFS is not absolutely required for working with Linux on the phyCORE-LPC3250, it will greatly decrease development time.

The setup of the NFS server will only be described for the openSUSE 11.1 distribution, although it will likely apply to earlier versions of openSUSE. Because of the vast number of Linux distributions out there, describing the setup of a NFS server for other Linux distributions is out of the scope of this document. Please refer to one of the many guides available on the internet for your particular distribution.

To begin setup:

1. Start *YaST*.

2. Under the *Software* section on the left in the *YaST Control Center*, click *Software Management*.

3. In the *Search* field, type **nfs** and click *Search*.

4. Check the *nfs-kernel-server* and *yast2-nfs-server* packages and click *Accept* to install.

RESOLVE

Make sure you resolve dependencies required to complete the package install if prompted to do so.

5. After installation has completed, close the Software Management window then close and re-open YaST to refresh the list of YaST components.

6. Under the Network Services section in the YaST Control Center, click NFS Server.

7. Under *Firewall* choose *Open Port in Firewall*. Click *Next*.

8. Click the *Add Directory* button and type the complete path to your LTIB root file system. For this QuickStart the following path was used: ***/home/cday/lpc3250/ltib/rootfs***.

9. If the *Add Host* window does not appear automatically, then click the *Add Host* button and type the IP address of the phyCORE-LPC3250 SOM into the *Host Wild Card* box. Make sure you select a free IP address available on your network. Ask your network administrator for assistance if you are unsure of how to obtain a free IP address.

10. In the *Options* box type the following configuration options:
    rw,no_root_squash,sync,no_subtree_check

11. Click *OK* and then click *Finish* for the settings to take effect. Close the YaST Control Center.

SUCCESS

Your NFS is ready to be accessed by the SOM.

3.5 Overview of the Boot Process

Before proceeding into the following sections it is helpful to first have an understanding of the entire boot process and the software components required to boot and run Linux on the phyCORE-LPC3250.

There are six software components that are involved in bringing Linux to life on the phyCORE-LPC3250. A brief description of each is given below.

1. LPC3250 on-chip bootstrap software -- This software component is part of the LPC3250 SoC itself. It is the very first piece of software that is executed after a system reset/power cycle. The bootstrap software is capable of booting code from several sources, including UART, SPI, external memory bus, and NAND Flash.

2. Kickstart Loader (KS) -- This software component is stored in the NAND Flash at block 0 and is loaded into IRAM and executed by the on-chip bootstrap software. The Kickstart Loader does some GPIO initialization and then proceeds to load the Stage 1 Loader into IRAM and transfer execution to it.

3. Stage 1 Loader (S1L) -- This software component is stored in the NAND Flash and is loaded into IRAM and executed by the Kickstart Loader. The S1L does additional low level initiaization, including SDRAM, and allows interaction with the outside world via a command line driven interface over a terminal session. The S1L is responsible for loading U-Boot into SDRAM and transfering execution to it.

4. Das U-Boot -- This software component is one of the leading bootloaders used to boot the Linux kernel. It is feature rich supporting environment variables, NAND Flash, and Ethernet (TFTP) to name a few. U-Boot is responsible for loading the Linux kernel into SDRAM and transfering execution to it.

5. Linux Kernel -- This is the monolithic kernel that is the Linux Operating System.

6. Root File System -- This software component contains all of the data associated with particular software components installed on the system, configuration files, device nodes, etc... that are available to a user when the operating system is running.

After a reset the normal boot procedure becomes:

1. On-chip bootstrap executes and boots the Kickstart Loader.

2. Kickstart Loader executes and boots the Stage 1 Loader.

3. Stage 1 Loader executes and boots Das U-Boot.

4. Das U-Boot executes and boots the Linux kernel.

The Kickstart Loader, Stage 1 Loader, and Das U-Boot are relatively stable components that do not require frequent modifications or rebuilds. During development these components are always flashed to the SOM because of their static, stable nature.

The Linux kernel and root file system can change often during system development and debug. Because of this it is desirable to have these components loaded dynamically onto the target SOM until they've reached a mature, stable status that is unlikely to change. Dynamic loading and access of the Linux kernel and root file system is done via TFTP and NFS, as explained in the previous sections of this document.

Now that a better understanding of the boot process and components required to bring Linux up on the phyCORE-LPC3250 has been presented, you are ready to place U-Boot into the NAND Flash, followed by booting the Linux kernel for the first time with the newly compiled kernel and root file system from Chapter 3.2.

3.6 Placing U-Boot into NAND Flash

The phyCORE-LPC3250 SOM comes pre-flashed with the Kickstart and Stage 1 Loaders, so you do not need to flash these software components. U-Boot does not come pre-flashed to the SOM and must be flashed in order to boot Linux.

After U-Boot is flashed to the board, the system can be booted by fetching the latest kernel image (uImage) over TFTP and mounting the latest root file system over NFS.

At later points in this QuickStart you will be provided instructions for flashing the kernel and root file system to the board. However, during development of your target system the need to rebuild the kernel and file system will occur frequently. TFTP and NFS accelerate the development process. Reflashing the newest kernel and root file system to the SOM after every new build would be very cumbersome and time consuming. Das U-Boot, unlike the kernel and file system, will probably not need to be reconfigured after it has been built. For this reason U-Boot is the only image that is initially flashed to the SOM in the development process.

If you followed Chapter 2.2 and booted the pre-built images shipped on the SD Card, then you've already flashed U-Boot to the SOM. Unless you have custom needs for U-Boot it is unlikely that you need to flash the version you built in Chapter 3.2. If you do have custom U-Boot requirements, or are interested in seeing the version you built in action, follow the steps below:

To begin flashing U-Boot to the SOM:

1. Copy the *u-boot.bin* file built in the previous steps of this QuickStart from the root file system */boot* directory to your SD card. You can use the SD card that came with the kit. For this QuickStart *u-boot.bin* is located at the following location: ***/home/<user-name>/lpc3250/ltib/rootfs/boot/u-boot.bin***.

2. Place the SD card into the SD/MMC connector X15 on the phyCORE-LPC3250 Carrier Board.

3. Switch to minicom, or the applicable terminal software you've chosen for this QuickStart.

4. Press the reset button S1 on the phyCORE-LPC3250 Carrier Board for a clean reboot. If you've previously flashed U-Boot to the board make sure you hit the space bar to stop the boot process in the Stage 1 Loader.

5. You will see something similar to the following in your terminal output window:

    ```
    Phytec LPC3250 board
    Build date: Dec 4 2008 09:45:09

    phy3250>
    ```

6. Type **ls** at the prompt to get a list of the files on the SD card. You should see the ***U-BOOT.BIN*** binary that you copied to the SD card in the list of files.

7. Type the following at the prompt to load the u-boot.bin binary into SDRAM for subsequent writing to NAND Flash:

    ```
    phy3250>load blk u-boot.bin raw 0x83fc0000
    ```

8. Type **nsave** to write the image to flash.

9. Configure the autoboot settings by typing the following:

    ```
    phy3250>aboot flash raw 0x83fc0000
    ```

10. Configure the prompt string and boot delay (2 seconds) by typing the following:

    ```
    phy3250>prompt phy3250-linux> 2
    ```

11. Erase the environment variables that were saved to NAND Flash (stored in blocks 125 through 128) when you flashed the U-Boot binary that was included on the SD Card:

    ```
    phy3250>erase 125 4 0
    ```

12. Cycle power, or press the RESET button S1 to reset the board. You should see something similar to the output below on your terminal window.

    ```
    Phytec LPC3250 board
    Build date: Dec  4 2008 09:45:09
    Autoboot in progress, press any key to stop...

    U-Boot 1.3.3 (Feb 10 2009 - 15:44:53)

    DRAM:  64 MB
    NAND:  64 MiB
    *** Warning - bad CRC or NAND, using default environment

    In:    serial
    Out:   serial
    Err:   serial
    uboot>
    ```

    The bad CRC warning is normal. This is indicative that the current environment variables that U-Boot attempted to read from NAND Flash were bad, or non-existent. Since you erased the environment variables that were stored in the previous step, they are non-existent. This CRC warning will come up until a **saveenv** command is executed in U-Boot. This will be done shortly in the next section when U-Boot is configured to boot the Linux kernel.

SUCCESS

> You have successfully placed U-Boot into the NAND Flash and booted the system.

3.7 Configuring U-Boot to Boot Linux over TFTP

With U-Boot flashed to the board you are ready to boot the system over TFTP and NFS for the first time. To do this U-Boot must be configured with several environment variable parameters to direct it where to boot the Linux images from, and to tell Linux how to mount the root file system.

Execute the following steps to configure U-Boot:

1. Reset the board by cycling power or pressing the reset button S1 on the Carrier Board.

2. At the U-Boot prompt execute the following set of commands:

    ```
    uboot> setenv bootfile uImage
    uboot> setenv bootcmd 'tftpboot; bootm'
    uboot> setenv bootdelay 1
    uboot> setenv loadaddr 0x80100000
    ```

3. Execute the following U-Boot commands to configure the TFTP server IP address:

   ```
   uboot> setenv serverip <tftp server IP>
   ```

4. Now configure the boot arguments that are passed to the Linux kernel during boot:

   ```
   uboot> setenv bootargs console=ttyS0,115200n8 root=/dev/nfs rw nfsroot=<
   NFS server IP address>:/home/<username>/lpc3250/ltib/rootfs init=/sbin/i
   nit ethaddr=${ethaddr} ip=<SOM IP address>
   ```

5. Set the IP address of the SOM to whatever IP you've chosen from the free pool of IP addresses:

   ```
   uboot> setenv ipaddr <SOM IP address>
   ```

6. Now type **saveenv** at the U-Boot prompt to save the environment variables to NAND Flash. This will prevent you from having to type the environment variables into U-Boot after every reset and it will also remove the CRC warning error that you received the first time U-Boot was booted.

7. At this point the system is almost ready to boot. The last step is to copy the kernel image built in Chapter 3.2 to the tftpboot directory. Since the */tftpboot* directory is likely owned by root and restricted to writes only by the owner you will need to modify this to allow writing by normal users. Do this by executing the following commands as root:

   ```
   # chown <username> /tftpboot
   # chgrp users /tftpboot
   ```

8. Now copy the kernel image to the tftpboot directory by executing the following command:

   ```
   ~/lpc3250> cp /home/<username>/lpc3250/ltib/rootfs/boot/uImage /tftpboot
   ```

   Note that after LTIB completes the build process the kernel image is placed in the root file system under the *boot/* directory as the file *uImage*.

9. You are now ready to boot the system for the first time. If everything has been set up correctly the system should boot all the way to a root prompt upon reset. Press the reset button S1 on the Carrier Board to boot the system into Linux.

> Given that you are likely to rebuild the kernel several times during your development process it is a good idea to add a command to the LTIB configuration menu that copies the newly built kernel image to your TFTP boot directory after every build. You can do this by entering the LTIB configuration menu (typing **./ltib --configure** in the base LTIB directory) and selecting the T*arget Image Generation Options* menu item, and then the *Run a command after building the target imag*e option and using **cp /home/*<username>*/lpc3250/ltib/rootfs/boot/uImage /tftpboot** as the argument.

3.8 Placing the Kernel into NAND Flash

Placing the kernel into NAND Flash allows booting the system without the need for the TFTP hosted kernel image. This is the most common place to put the kernel in a standalone application. Normally development is done using a TFTP hosted kernel image until the configuration has become more stable and is unlikely to change frequently. Once stable, the kernel image can be moved to NAND Flash.

Placing the kernel image into the NAND Flash is accomplished using the U-Boot NAND write commands. The kernel is loaded into SDRAM using TFTP and then flashed to the board. To begin:

1. Copy the *uImage* kernel image file from the LTIB *rootfs/boot* directory to your TFTP-BOOT directory. For this QuickStart the *uImage* file is located at */home/<user-name>/lpc3250/ltib/rootfs/boot/uImage*.

2. Reboot your board by pressing the reset button and stop the boot process in U-Boot by hitting any key at the *Hit any key to stop autoboot* message.

3. Load the kernel image into SDRAM using the *tftpboot* command in U-Boot as follows:

   ```
   uboot> tftpboot 0x80100000 uImage
   ```

4. Erase the area of NAND Flash reserved for the kernel image partition:

   ```
   uboot> nand erase 0x204000 0x400000
   ```

   The *0x204000* is the offset of the beginning of kernel image partition, and the *0x400000* is the size of the partition.

5. The next step is to write the kernel image into NAND Flash. This is done using the *nand write.jffs2* command in U-Boot. This command takes three parameters:

   ```
   nand write.jffs2 <addr> <offset> <size>
   ```

   Where `<addr>` is the address of the source data (0x80100000 in this case), `<offset>` is the offset in NAND Flash to begin writing data, and `<size>` is the amount of data to write. The offset will always be 0x204000. The size will depend on the size of your kernel image you've created. The U-Boot tftpboot command will report the size of the image transfered after it is completed. You should see something similiar to the following when the tftpboot tranfer is completed:

   ```
   Bytes transferred = 1701080 (19f4d8 hex)
   ```

   In this instance the `<size>` field is 0x19f4d8. This value should be rounded up to the nearest block boundary. Block sizes are 16KB (16384 bytes). This means the above image is 1701080/16384 = 103.8 blocks. Rounding this up to 104 blocks yields 1703936 bytes, or 0x1a0000.

6. Now write the image to flash, remembering to use the correct size for your image:

   ```
   uboot> nand write.jffs2 0x80100000 0x204000 <image size>
   ```

7. Next, configure U-Boot to boot the kernel image from NAND Flash. See below for an example:

   ```
   uboot> setenv bootcmd  'nboot.jffs2 0x80100000 0x0 0x204000; bootm'
   ```

8. Lastly, save the environment variables using the **saveenv** command and reboot the board.

> The system should boot into Linux, fetching the kernel image from NAND.

SUCCESS

3.9 Placing the Root File System into NAND Flash

Placing the root file system into NAND Flash allows booting the system without the need for the NFS hosted root file system. This is the most common place to put the root file system in a standalone application. Normally development is done using an NFS hosted root file system until the configuration has become more stable and is unlikely to change frequently. Once stable, the root file system can be moved to NAND Flash.

Placing the root file system into the NAND Flash requires creating a jffs2 version of the root file system. Fortunately this task is very easily accomplished using LTIB. After the jffs2 root file system has been created, U-Boot can be used to write the file system image to the reserved root file system partition in the NAND Flash.

To place the root file system into NAND Flash, follow these steps:

1. Return to your shell window and start the LTIB configuration menu:

   ```
   ~/lpc3250/ltib>./ltib --configure
   ```

2. Scroll down to the *Target Image Generation Options* menu item and hit Enter.

3. Scroll down to the Target Image: (NFS only) option and hit Enter. Select *jffs2* from the list of options.

4. A new list of options will appear to allow you to configure the jffs2 image created. Scroll down to the *jffs2 erase block size in KB (NEW)* option and change the erase block size from 64KB to 16KB to match the size of the erase blocks for the small page NAND Flash on the SOM.

5. Now exit from the *Target Image Generation Options* menu, returning to the main LTIB configuration menu.

6. Exit from LTIB, saving the new configuration.

7. LTIB will build the new kernel and file system. When the build is completed a new file called rootfs.jffs2 will available in the root LTIB directory. For this QuickStart the file is located at ***/home/<username>/lpc3250/ltib/rootfs.jffs2***. This is the root file system in a JFFS2 image format suitable for placement into the NAND Flash.

8. Copy the **rootfs.jffs2** image and the **uImage** file to your TFTPBOOT directory.

9. Reboot your board by pressing the reset button and stop the boot process in U-Boot by hitting any key at the *Hit any key to stop autoboot* message.

10. Load the image file into SDRAM using the *tftpboot* command in U-Boot as follows:

    ```
    uboot> tftpboot 0x80100000 rootfs.jffs2
    ```

11. Erase the area of NAND Flash reserved for the root file system partition:

    ```
    uboot> nand erase 0x604000 0x39fc000
    ```

    The *0x604000* is the offset of the beginning of root file system partition, and the *0x39fc000* is the size of the partition.

12. The next step is to write the root file system image into NAND Flash. This is done using the nand write.jffs2 command in U-Boot. This command takes three parameters:

    ```
    uboot> nand write.jffs2 <addr> <offset> <size>
    ```

    Where `<addr>` is the address of the source data (0x80100000 in this case), `<offset>` is the offset in NAND Flash to begin writing data, and `<size>` is the amount of data to write. The offset will always be 0x604000. The size will depend on the size of your root file system image you've created. The U-Boot tftpboot command will report the size of the image transfered after it is completed. You should see something similiar to the following when the tftpboot tranfer is completed:

    ```
    Bytes transferred = 3719168 (38c000 hex)
    ```

    In this instance the `<size>` field is 0x38c000. Note that the size should always be a multiple of the erase block size (16kB or 0x4000 hex). You should round up the image size if this is not the case. If the JFFS2 erase block size was set correctly in the LTIB configuration menu, the image size will automatically fall on a block size boundary, and no rounding is required.

13. Now write the image to flash, remembering to use the correct size for your image. This process will take a few minutes to complete:

    ```
    uboot> nand write.jffs2 0x80100000 0x604000 <image size>
    ```

14. Next, configure the kernel command line to tell Linux to use the NAND partition as the root file system. The *root* option must be set to */dev/mtdblock4 rw* and the *rootfstype* must be set to *jffs2*. The */dev/mtdblock4* refers to the NAND partition used for root file system. See below for an example:

    ```
    uboot>
    setenv bootargs console=ttyS0,115200n8 root=/dev/mtdblock4 rw rootfstype
    =jffs2 init=/sbin/init ip=<SOM IP address> ethaddr=${ethaddr}
    ```

15. Lastly, save the environment variables using the **saveenv** command and reboot the board.

The system should boot into Linux with the root file system in NAND Flash.

SUCCESS

3.10 Placing the Root File System on an SD/MMC Card

Placing the root file system on an SD/MMC card allows booting the system without the need for the NFS hosted root file system. Although not as convenient for development purposes, placing the root file system on an SD/MMC card is one possible solution for standalone applications.

Placing the root file system on an SD/MMC card is straightforward. The process requires formatting an SD/MMC card for the ext2 file system and then copying the NFS root file system created by LTIB onto the SD card. The process will be explained for the openSUSE 11.1 distribution using a USB-to-SD/MMC card reader. The process may vary slightly depending your distribution/platform. There are numerous helpful internet resources that can be found via a simple search to help you if the process described below doesn't apply 100% to your system.

Note that all data on this SD/MMC card will be lost when following this procedure.

To place the root file system on an SD/MMC card, follow these steps:

1. Insert an SD/MMC card into your SD/MMC card slot on your Linux host. Open a shell window as root and type **mount** to get a list of mounted devices. You should see the SD/MMC card show up in the list as something similar to this:

    `/dev/sdc1 on /media/disk type ext2 (rw,nosuid,nodev)`

    If you are unsure which item in the list is the SD/MMC card you can type **mount** before inserting the card and again afterwards and then look for the new item in the list.

    As a note before continuing to the next step, the `/dev/sdc1` entry is the device link to the SD/MMC card partition 1. The SD/MMC can have multiple partitions, in which case each partition would show up as `/dev/sdc2, /dev/sdc3`, etc... We will use the *fdisk* command to setup the card with one Linux ext2 partition. To do this fdisk operates on the raw disk data accessed at `/dev/sdc` (note that NO number is appended to this). If your SD/MMC card shows up as `/dev/sdd1` for instance, then you will access the partition table/raw disk data at `/dev/sdd`.

Note that it is imperative you select the correct `/dev` entry when using fdisk. Selecting the wrong disk, such as your primary hard disk could result in irreparable damage to the data on your system. In the following sections replace all occurrences of `<sd?>` with the appropriate entry for you SD card. As en example, for the system this QuickStart was tested on, `<sd?>` would be replaced by `sdc`.

2. Temporarily open a shell as root by typing **su** at the prompt and entering the root password. You need root access to use fdisk.

3. Umount the disk by typing the following:

    `# umount /dev/<sd?>1`

4. Invoke *fdisk* to edit the partition table by typing the following at the prompt:

    `# fdisk /dev/<sd?>`

    When fdisk starts you should see a new prompt that says `Command (m for help):`

5. Type **p** to print the current SD/MMC partition table. As a method to further validate you are operating on the correct /dev entry, ensure that the disk size reported by fdisk matches the SD/MMC card you're using.

6. Use the **d** option in fdisk to delete all partitions on the disk.

7. After all partitions have been deleted, use the **n** option to create a new partition.

8. Select **p** for primary.

9. Select **1** for partition number.

10. Hit **enter** to select 1 as the first cyclinder.

11. Hit **enter** to select the last available cyclinder on the device (default).

12. Select **w** to write the new partition table to the SD/MMC card. The new partition will have been created as Linux type partition. Fdisk will automatically exit back to a prompt after writing the partition table to the disk.

13. Now make the ext2 file system by using the *mkfs.ext2* tool like so:

    ```
    # mkfs.ext2 /dev/<sd?>1
    ```

If your Linux distribution automatically remounted the disk after creating the new partition table you may have to unmount it again using the umount command as was done at step #3 above before executing the *mkfs.ext2* command.

14. Now remount the card by issuing a *mount* command. Alternatively if your distribution auto mounts the SD/MMC card, you can remove the card and reinsert it to mount it. To mount the card via the *mount* command, type the following:

    Make a mount point:
    ```
    # mkdir /mnt/sdmmc
    ```

    Mount the card:
    ```
    # mount -t auto /dev/<sd?>1 /mnt/sdmmc
    ```

    If you've removed the card and reinserted it you must use the mount point that was automatically created by your distribution. In step #1 above you can see this mount point is `/media/disk` instead of the manually created mount point `/mnt/sdmmc`.

15. You are now ready to copy the LTIB generated root file system to the SD/MMC card. To do this, type the following:

    ```
    # cp -R /home/<username>/lpc3250/ltib/rootfs/* /mnt/sdmmc
    ```

The SD/MMC card should be ready to use and mount as the root file system. Now you must configure U-Boot with the appropriate kernel command line to mount the SD/MMC card as the root file system instead of NFS, or some other mount point. To begin:

1. Umount the card and remove it from your Linux host machine and then insert it into the SD/MMC card connector X15 on the phyCORE-LPC3250 Carrier Board.

2. Cycle power on the Carrier Board, or press the reset button S1. When U-Boot begins booting hit any key to stop the boot process in U-Boot. You will come to a boot prompt that looks like this:

    ```
    uboot>
    ```

3. Type **printenv** to get a dump of the current U-Boot environment variables.

4. Change the bootargs environment variable so the rootfs option points to the ext2 root file system copied onto the SD/MMC card.

```
uboot> setenv bootargs
console=ttyS0,115200n8 root=/dev/mmcblk0p1 init=/sbin/init
ethaddr=${ethaddr} ip=<SOM IP address>
```

Change this so the root option is /dev/mmcblk0p1 by typing the following:

```
uboot> setenv bootargs  'console=ttyS0,115200n81 root=/dev/mmcblk0p1
ip=<SOM IP address> init=/sbin/init'
```

5. Now type **saveenv** to save the new environment variables configuration.

6. Type **run bootcmd** to boot the system, or hit the system reset button S1. If the procedure was done correctly the system should boot to a prompt.

3.11 Building Custom Images

The Apache Web Server is an open source HTTP server that is ideal for embedded designs. In addition to providing static content, Apache also integrates easily with dynamic content pages driven by PHP or Perl.

3.11.1 Building the Apache Web Server

1. Open a shell window and start the LTIB configuration menu:

```
~/lpc3250/ltib> ./ltib --configure
```

2. Scroll down to the *Package list* option menu item and hit *Enter*. This will bring up a menu of all of the packages that are available with LTIB.

3. Scroll down to the *httpd (apache) web server* package and hit the *spacebar* to select it. An asterix should appear next to the entry.

4. Exit from the *Target System Configuration Options* menu and then exit from LTIB, saving your changes. LTIB will begin building the new packages and installing them into the root file system.

5. After the build has completed you must make the updated root file system available to the SOM. If you are using NFS to host the root file system on your Linux machine then nothing further needs to be done. You can simply reboot the SOM to use the new root file system with the newly intsalled software packages. If you are using the SD/MMC card, or NAND Flash to host the root file system, then the new *ltib/rootfs* needs to be copied to the SD/MMC card, or the *rootfs.jffs2* image needs to be reflashed to NAND. Please refer to Chapter 3.10 and Chapter 3.9 respectively for instructions on updating the root file system.

3.11.2 Testing the Apache Web Server

1. If you haven't already rebooted the SOM after buliding the new root file system, then reboot the SOM by pressing the reset button S1 on the Carrier Board.

2. At the root prompt, start the Apache Web Server by typing the following:

```
[root@nxp /]# /usr/bin/httpd
```

3. On the host PC, open a web browser and enter the IP address of your SOM into the address bar. If everything is operating correctly you should see a page appear similar to the one below:
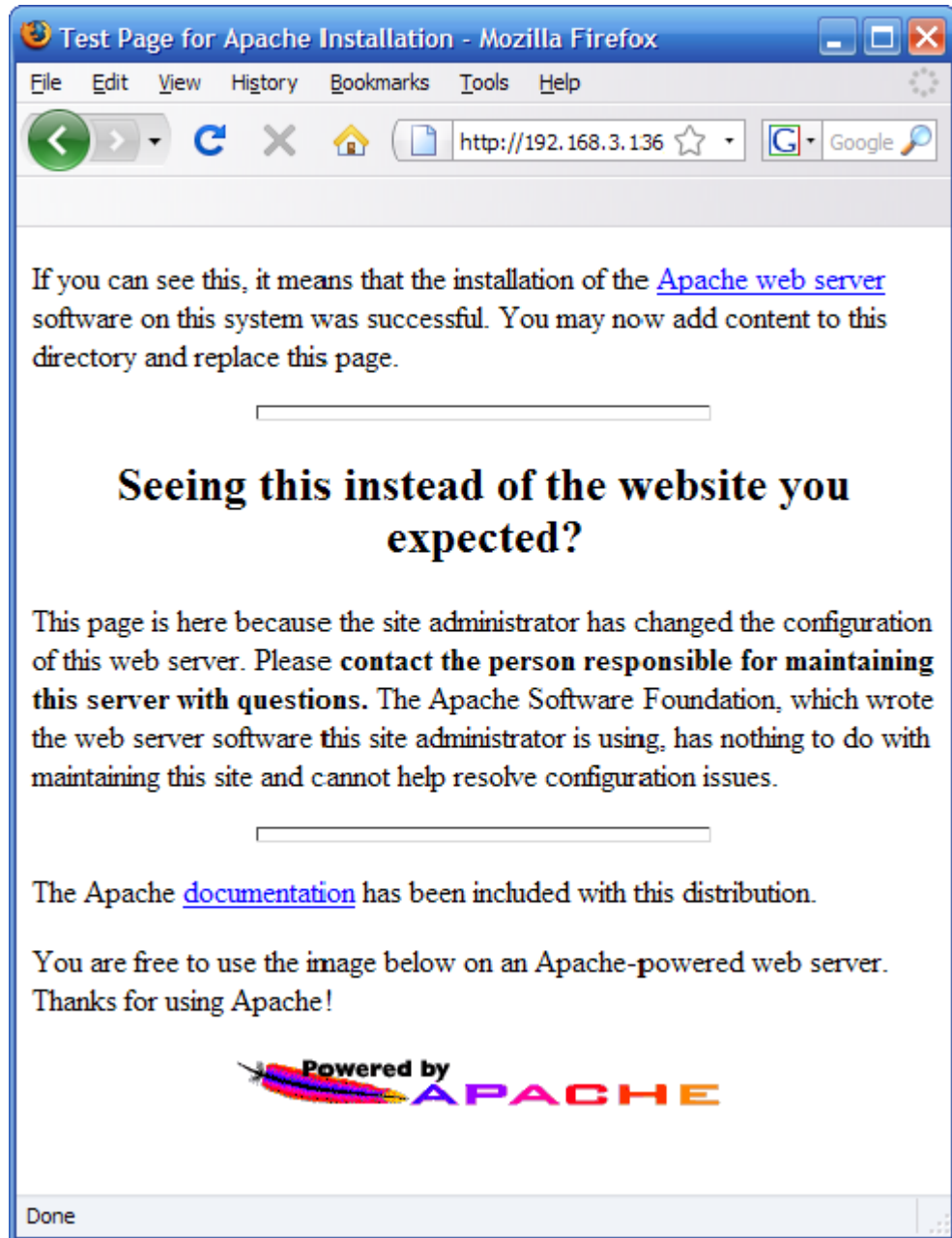
SUCCESS

You have successfully served your first webpage using the Apache Web Server.

TIP

The default directory out of which documents are served is ***/usr/htdocs***. This can be modified by editing the ***/usr/conf/httpd.conf*** configuration file, along with a vast number of other settings to configure the Apache Web Server.

3.11.3 Building Qtopia

Qtopia is a graphical environment for Linux on embedded devices. Qtopia provides the functionality needed to develop advanced GUI applications. Using the touchscreen, a user can interact with Qtopia to run applications or games on the phyCORE-LPC3250. If you have the PHYTEC KLCD-011, you can build and install Qtopia by following these instructions:

CAUTION

Qtopia can take up to two hours to build, even with a fast connection and a fast PC. Make sure everything else can build successfully before attempting to build Qtopia.

1. Install additional packages on the host machine that are required to build Qtopia.

    In OpenSuse 11.3, use Zypper or YaST to install the following packages and their dependencies to resolve Qtopia's dependencies:

    *   xorg-x11-devel

    *   libjpeg-devel

    *   zlib-devel

    *   libuuid-devel

2. Open a shell window and start the LTIB configuration menu
    ```
    ~/lpc3250/ltib> ./ltib --configure
    ```

3. Scroll down to the *Package list* option menu item and hit **Enter**. This will bring up a menu of all of the packages that are available with LTIB.

4. Scroll down to the *Qtopia* menu item and hit **Enter**.

5. Select the *Qtopia version* menu item and hit **Enter**. Use the arrow keys to highlight *Qtopia 2* and hit **Enter**.

6. Scroll down to the *Use touchscreen* item and press the **spacebar** to select it. An asterisk should appear next to the entry.

7. Exit from the *Qtopia* menu, then exit from the *Package list* menu. Scroll down to the *Target System Configuration Options* menu item and hit **Enter**.

8. Scroll down the list and select the *start Qtopia* menu item

9. Exit from the *Target System Configuration Options* menu and then exit LTIB, saving your changes. LTIB will begin building the new packages and installing them into the root file system.

10. After the build has completed you must make the updated root file system available to the SOM. If you are using NFS to host the root file system on your Linux machine then you can simply reboot the SOM to use the new root file system with the newly installed software packages. If you are using the SD/MMC card, or NAND Flash to host the root file system, then the new ltib-qs/rootfs needs to be copied to the SD/MMC card (see Chapter 3.10), or the rootfs.jffs2 image needs to be reflashed to NAND. (see Chapter 3.9)

11. If Qtopia built and installed successfully, it will run automatically on next boot. You must calibrate the touchscreen before you can interact with Qtopia. (see Chapter 3.11.4)

3.11.4 Touchscreen Calibration

The TSLIB package can be used to test the touchscreen support and to calibrate the touchscreen for other tslib supported applications. The following procedure needs to be completed for the touchscreen to work with Qtopia. Please follow the instructions below on how to setup tslib and perform a calibration sequence.

These instructions are from http://www.lpclinux.com/Forums/ForumLPC32xx0008

1. Boot to the Linux kernel and locate the device information for the touchscreen device. This can be done by viewing the ***/proc/bus/input/devices*** file:

    ```
    [root@nxp /]# cat /proc/bus/input/devices
    I: Bus=0019 Vendor=0001 Product=0002 Version=0100
    N: Name="lpc32xx-ts"
    P: Phys=lpc32xx/input0
    S: Sysfs=/class/input/input0
    U: Uniq=
    H: Handlers=mouse0 event0
    B: EV=b
    B: KEY=400 0 0 0 0 0 0 0 0 0
    B: ABS=1000003
    ```

2. Create a device node for the device if it doesn't already exist. You can get the device node major and minor numbers using the following sequence and the SysFs information above.

    ```
    [root@nxp /]# cd /sys/class/input/input0/event0/
    [root@nxp event0]# cat dev
    13:64
    [root@nxp event0]# mknod /dev/input/event0 c 13 64
    [root@nxp event0]#
    ```

3. Setup the following environment variables for tslib:

    ```
    [root@nxp event0]# export TSLIB_TSDEVICE=/dev/input/event0
    [root@nxp event0]# export TSLIB_CALIBFILE=/etc/pointercal
    [root@nxp event0]# export TSLIB_CONFILE=/etc/ts.conf
    ```

4. Create the file ***/etc/ts.conf*** with the following lines:

    ```
    module_raw input-raw
    ```

```
module pthres pmin=1
module variance delta=100
module dejitter delta=10
```

5. Run ts_calibrate to calibrate the touchscreen. The calibration data will be saved to
    /etc/pointercal

```
[root@nxp /]# ts_calibrate
xres = 240, yres = 320
Took 5 samples...
Top left : X =  244 Y =  180
Took 4 samples...
Top right : X =  809 Y =  176
Took 3 samples...
Bot right : X =  801 Y =  843
Took 2 samples...
Bot left : X =  232 Y =  839
Took 3 samples...
Center : X =  516 Y =  510
-10.378296 0.246892 0.003720
-9.088440 -0.000006 0.331812
Calibration constants: -680152 16180 243 -595620 0 21745 65536
```

6. The ts_test and ts_print applications can also be executed to test the touchscreen.

> You are now ready to start using Qtopia. Tap anywhere on the touchscreen and follow the
> steps that appear on the touchscreen to configure Qtopia.

SUCCESS

3.11.5 Adding your Own Packages

LTIB provides an extensive list of packages that can be built for the target. The basic process for
including a package in the build process is as follows:

1. Opening a shell window and starting the LTIB configuration menu.

2. Scrolling down to the *Package list* menu item.

3. Scrolling through the list of available packages and selecting the desired packages with
    the spacebar.

4. Exiting LTIB and saving changes to begin the build process.

Once the build process is completed, the new package will be built and installed into the root file
system. Several packages allow you to do some configuration in LTIB for system startup. To
access this configuration select the *Target System Configuration Options* from the LTIB menu after

you've selected the packages to include in the build. For instance, after selecting the openssh package a new option appears in the *Target System Configuration Options* that allows you to automatically start the openssh server on boot.

Deselecting a package after it has been built does not remove it from the root file system install. To remove a package that has been built, use the LTIB `-m  clean` option on the command line. For instance, to remove the openssh package once built and installed, execute the following:

```
~/lpc3250/ltib> ./ltib -p openssh -m clean
```

Where the -p option specifies the package to remove.

Note that many packages have dependencies that are automatically selected when you select the package. As soon as you select the package in the package list, LTIB automatically selects the dependencies. If you deselect a package, you must manually go through and deselect all dependencies. Also note that some of the dependencies may have dependencies of their own that are automatically selected. When removing a package remember that all dependencies must be removed (unless they are required by other packages) to fully remove a package. You can find package dependencies by examining the *ltib/config/userspace/packages.lkc* file.

Once you've successfully built a new file system, it can be deployed to the target in one of the several ways outlined earlier in this QuickStart: via NFS, via SD/MMC, or via NAND Flash. You must update you root file system to use the new packages built in this section. If you're using NFS then the updated root file system is already handled by the act of building new images. If you're using SD/MMC or NAND Flash as the target location for the root file system then you must either copy the new *ltib/rootfs* directory to the SD/MMC card, or flash the new *rootfs.jffs2* image to the NAND Flash.

Appendix A: Helpful Hints and Tips

A.1 Where to Find More Information

There are several resources available beyond this QuickStart to facilitate you in working with Linux on the phyCORE-LPC3250. You may find some of the following links helpful:

- The NXP maintained Linux BSP page with quickstart, manual, FAQ, and current list of BSP issues -- http://www.standardics.nxp.com/support/software/lpc32xx.bsp.linux/

- LTIB FAQ -- http://ltib.org/autodocs/LtibFaq.html

- LTIB training guide -- http://www.ltib.org/pages/LTIB_generic_v1.4_-_version_6.4.1.pdf

- LPCLinux Community -- http://www.lpclinux.com

A.2 NAND Flash Layout

The default NAND Flash populating the kit version of the phyCORE-LPC3250 SOM is 64MB in size, containing 4096 blocks, each of which are 16KB in size. See Table 4-1 below for a summary of the NAND Flash layout.

Table 4-1. NAND Flash Layout

Start Block	End Block	# of Blocks	Size	Description
0	0	1	16KB	Kickstart Loader
1	24	24	384KB	Stage 1 Loader
25	124	100	1.56MB	Das U-Boot
125	128	4	64KB	Das U-Boot Environment Variables
129	384	256	4MB	Linux Kernel
385	4095	3711	57.98MB	Linux File System

Any one of the sections can be erased using the Stage 1 Loader, or Das U-Boot. When using the Stage 1 Loader, use the *erase* command. The arguments you must provide are the start block, number of blocks, and whether or not to erase bad blocks. In general you should never erase bad blocks unless you know what you're doing. See below for an example of erasing the U-Boot environment variables from the NAND:

```
phy3250>erase 125 4 0
```

To erase the same area using U-Boot you use the *nand erase* command. The arguments you must provide are start offset and erase size. You must do some conversions into hex based on the start block, and number of blocks, with each block being 16KB (0x4000 hex) in size. Execute the following command to erase the U-Boot environment variables using U-Boot:

```
uboot> nand erase 0x1f4000 0x10000
```

It may be useful to erase sections of the NAND Flash like this during development. Please avoid erasing the Kickstart and Stage 1 Loaders. Without these, the boards will not boot and a special recovery procedure must be used to restore them. Please refer to the following PHYTEC Knowledge Base article on restoring the Stage 1 and Kickstart loaders: http://www.phytec.com/kb/index.php?article=302

A.3 Using DHCP Instead of a Static IP

If your network has a DHCP server (as most networks do), you may find it useful to be able to use DHCP to assign an IP address to the SOM instead of using a static IP address. When using DHCP there are two areas that must be addressed: U-Boot and the Linux kernel.

U-Boot can be configured to use DHCP instead of a static IP. Instead of using the *tftpboot* command in U-Boot, the *dhcp* command can be used instead. In this case the *ipaddr* environment variable does not need to be set to a static IP, but is set automatically by U-Boot once the DHCP server assigns the SOM an IP address. See below for two examples that demonstrate using the *tftpboot* command to fetch a file over TFTP, and the *dhcp* command to fetch a file over TFTP.

To fetch a file (say uImage) over TFTP using the *tftpboot* command:

```
uboot> setenv bootfile uImage
uboot> setenv loadaddr 0x80100000
uboot> setenv ipaddr <SOM IP address>
uboot> setenv serverip <TFTP server IP address>
uboot> tftpboot
```

To fetch a file (say uImage) over TFTP using the *dhcp* command:

```
uboot> setenv bootfile uImage
uboot> setenv loadaddr 0x80100000
uboot> setenv serverip <TFTP server IP address>
uboot> dhcp
```

As can be seen from the two examples above, with the *tftpboot* command you must set the SOM *ipaddr* environment variable. With the *dhcp* command you do not need to do this, and instead the SOM is automatically assigned an IP address.

> **CAUTION**
>
> If you decide to use the dhcp option in U-Boot at some point during development while still using NFS to host the root file system, you must make sure that your NFS server allows a connection from the IP address assigned by the DHCP server to the SOM. You can ensure the DHCP server always assigns the same IP address to a particular SOM by configuring the DHCP server to assign a chosen IP address to any connecting device with a matching MAC ID. The MAC ID can be found on the green sticker on the top of the SOM.

On top of configuring U-Boot for DHCP operation, Linux can be configured to use DHCP also. This is done by using the *ip* command line option and specifying *dhcp* as the parameter. Throughout this QuickStart the *ip* command line option has always been set to a static IP address. Change this to say *ip=dhcp* instead and Linux will query the DHCP server for an IP address. See below for two examples of the kernel command line configured for static IP and for DHCP.

For static IP:

```
uboot> setenv bootargs console=ttyS0,115200n8 root=/dev
mtdblock4 rw rootfstype=jffs2 init=/sbin/init
ethaddr=${ethaddr} ip=<SOM IP address>
```

For DHCP:

```
uboot>
setenv bootargs console=ttyS0,115200n8 root=/dev/mtdblock4 rw rootfstype=jffs2
 init=/sbin/init ethaddr=${ethaddr} ip=dhcp
```

A.4 Enabling Ethernet in the Provided Images

The default images shipped on the SD card included in the kit have both the Dropbear SSH Server and Apache Web Server installed in the root file system -- both of these features require Ethernet connectivity. The included U-Boot image, however, does not enable the Ethernet interface in Linux by default. This is to facilitate booting the board without having to have an Ethernet connection available. If you want to enable Ethernet when using these default images, simply change the kernel command line in U-Boot to include an *ip* command line option. See below for examples on configuring for DHCP or static IP.

The U-Boot image included on the SD card uses the following bootargs environment variable setting for the kernel command line:

```
bootargs=console=ttyS0,115200n8 root=/dev/mtdblock4 rw rootfstype=jffs2 init=/
sbin/init ethaddr=${ethaddr}
```

To set a static IP, use the following setting:

```
bootargs=console=ttyS0,115200n8 root=/dev/mtdblock4 rw rootfstype=jffs2 init=/
sbin/init ethaddr=${ethaddr} ip=<static IP address>
```

To use DHCP for a dynamic IP, use the following setting:

```
bootargs=console=ttyS0,115200n8 root=/dev/mtdblock4 rw rootfstype=jffs2 init=/
sbin/init ethaddr=${ethaddr} ip=dhcp
```

If you want to bring the interface up temporarily in Linux without having to enable it in U-Boot, use the ifconfig command as shown below:

```
[root@nxp /]# ifconfig eth0 up <static IP address>
```

A.5 Testing SSH Server

SSH provides a secure shell connection to a remote system. SSH can be extremely useful to securely access and configure a remote target machine. It can also provide the ability to open multiple shell sessions to the same target. Dropbear SSH Server is the server component to allow SSH connections to the target system.

The first time the SOM is rebooted the SSH server will spend a short amount of time generating new keys during bootup, dumping the following message: `Generating keys for the dropbear ssh server.` Wait until the SOM boots to a root prompt.

To test the SSH server open a shell on your host Linux PC and execute the following command:

```
~/lpc3250> ssh <SOM IP address> -l root
```
SSH will connect to the SOM's SSH server and you will be notified that the authenticity of the host cannot be established. Type **yes** to indicate that you want to continue connecting. Next you will be prompted for the root password. Type **root** in as the password. See the output below for comparison.

```
~/lpc3250> ssh 192.168.3.136 -l root
The authenticity of host '192.168.3.136 (192.168.3.136)' can't be established.
RSA key fingerprint is c5:0b:79:bc:56:85:a1:bf:b1:05:1d:20:1c:07:74:6c.
Are you sure you want to continue connecting (yes/no)? yes
"Warning: Permanently added '192.168.3.136' (RSA) to the list of known hosts.
root@192.168.3.136's password:
[root@nxp /root]#
```

Subsequent connections to the SOM will not warn you about the authenticity of the host, since accepting the connection the first time added the RSA key to the list of known hosts.

You have successfully connected to the target via a secure shell connection.