Getting Started With Linux on the LPC3250 OEM Board

Embedded Artists

Embedded Artists AB

Södra Promenaden 51
SE-211 38 Malmö
Sweden

info@EmbeddedArtists.com
http://www.EmbeddedArtists.com

Disclaimer

Embedded Artists AB makes no representation or warranties with respect to the contents hereof and specifically disclaims any implied warranties or merchantability or fitness for any particular purpose. Information in this publication is subject to change without notice and does not represent a commitment on the part of Embedded Artists AB.

Feedback

We appreciate any feedback you may have for improvements on this document. Please send your comments to support@EmbeddedArtists.com.

Trademarks

All brand and product names mentioned herein are trademarks, services marks, registered trademarks, or registered service marks of their respective owners and should be treated as such.

Table of Contents

1 Introduction

TBD

Something about S1L, documentation…

1.1 TBD

TBD

1.2 Organization of This Document

TBD

- **Chapter 2** – *TBD*
  TBD

1.3 Conventions in This Book

A number of conventions have been used throughout the book to help the reader better understand the content of the book.

`Constant width text` – is used for file system paths and command, utility and tool names.

```
$ This field illustrates user input in a terminal running on the
development workstation, i.e., on the workstation where you edit,
configure and build Linux
```

```
# This field illustrates user input on the target hardware, i.e.,
input given to the terminal attached to the LPC3250 OEM Board
```

```
This field is used to illustrate example code or excerpt from a
document.
```

2 Getting Started

2.1 Introduction

This chapter describes how you get-up-and running with prebuilt images of the u-boot, Linux kernel and root file system. Necessary images can be downloaded from Embedded Artists support site or you can follow the instructions in chapter 3 to build the images yourself.

- `u-boot.bin` – The Universal Bootloader, known as u-boot for short.
- `uImage` – The Linux kernel image.
- `rootfs.jffs2` – A JFFS2 formatted root file system to be stored in NAND flash.

Besides the images you will also need a **MMC/SD card** and a **USB memory stick** in order to follow the instructions below. A **terminal application** is also required to interface towards the board. These instructions will be using the *Tera Term* terminal application.

2.2 Preparation and Setting up the Board

In this section you will setup the board and boot into the Stage 1 boot loader (S1L) where it will be possible to then load the u-boot.

1. Copy the `u-boot.bin` file to the root directory in a FAT formatted MMC/SD card.
2. Copy the `uImage` and `rootfs.jffs` files to the root directory of a USB memory stick.
3. Insert the MMC/SD card in the MMC/SD card slot on the QVGA Base board, see Figure 1.
4. Insert the USB memory stick in the USB host connector, see Figure 1.
5. Connect the USB cable that came with the board to the USB device connector marked UART #0, see Figure 1. Also make sure that the cable is connected to your computer.
6. The board will now power up. Follow the instructions in the User's Manual for the LPC3250 OEM Board to install necessary FTDI USB drivers and identify which COM port that was assigned to the board.
7. Start your terminal application and connect to the COM port associated with the board.

   ***Note**: Make sure the automatic ISP jumpers (marked RST and P2.10 on the base board) are **open**. If not it's possible that a terminal application resets the board. In Figure 1 the jumpers are closed.*

8. Reset the board by pressing the Reset button and the S1L bootloader will boot, see Figure 2 to see how it looks like in Tera Term.
9. You are now ready to continue to the next section and load the u-boot.

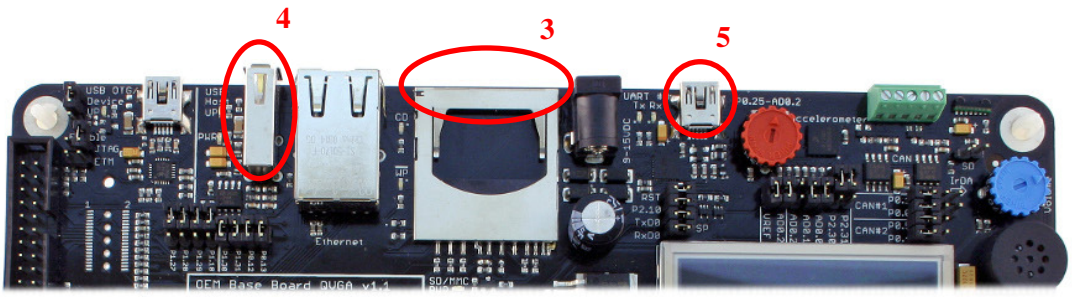

Figure 1 Top Part of QVGA Base Board

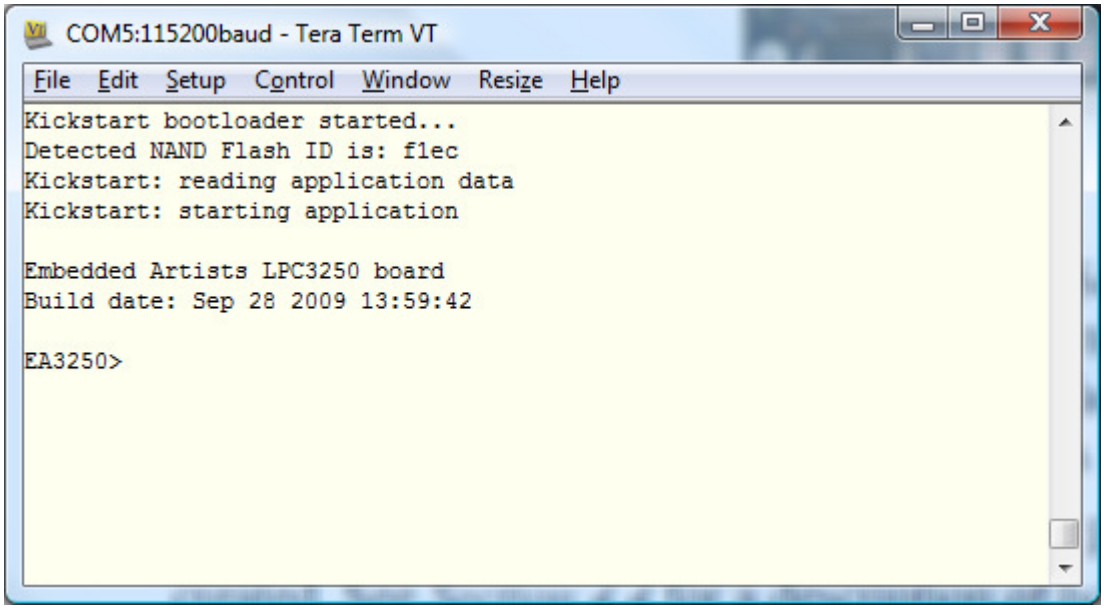

Figure 2 S1L Bootloader Console

2.3 Load and Setup the u-boot

In this section you will load the u-boot from the memory card and store it NAND flash on the target board.

Make sure you have followed the instructions in section 2.2 and that you have output in your terminal application similar to the output shown in Figure 2.

1. Load the u-boot.bin file from the MMC/SD card to SDRAM at address 0x83fa0000.

```
EA3250> load blk u-boot.bin raw 0x83fa0000
```

2. Save the image into NAND flash.

```
EA3250> nsave
```

3. Setup the S1L bootloader to automatically load the u-boot.

```
EA3250> aboot flash raw 0x83fa0000
```

4. Set the boot delay to 2 seconds (the system prompt must be set at the same time).

```
EA3250> prompt EA3250> 2
```

5. Issue the `boot` command to start u-boot.

```
EA3250> boot
```

6. The u-boot will now start and you will see something similar to the output below. **Hit any key** to stop the u-boot from autobooting.

```
U-Boot 2009.03-rc1 (Sep 27 2009 - 14:27:25)

DRAM:  64 MB
NAND:  128 MiB
*** Warning - bad CRC or NAND, using default environment

In:    serial
Out:   serial
Err:   serial
Hit any key to stop autoboot:  0
uboot>
```

7. The warning message is not anything to worry about. It just means that the u-boot environment hasn't been saved to persistent storage (NAND flash).
8. Enter `print` in the u-boot console to see the u-boot environment. Only a portion of the variables are displayed below.

```
uboot> print

bootargs=
bootcmd=run mtdboot

...

ethaddr=00:1a:f1:00:00:00
ipaddr=192.168.5.234
serverip=192.168.5.88
rootpath=/home/user/ltib/rootfs

...
```

9. In order to test the network functionality of Linux you need to change the `ipaddr` variable to an IP address that is valid on your network. In this example we assume that 192.168.0.100 is valid on your network (change it to an address that really is valid on your network).

```
uboot> setenv ipaddr 192.168.0.100
```

10. Now save your changes to the environment. This will also remove the warning message you saw in step 6.

```
uboot> saveenv
```

11. Continue to the next section where you will load the root file system and Linux kernel.

2.4 Load the Root File System and Linux Kernel

In this section you will load the root file system (`rootfs.jffs2`) from the USB memory stick and store it in NAND flash. You will then also load the Linux kernel (`uImage`) from the USB memory stick and boot the kernel.

1. The default u-boot environment has been prepared with a variable named `update_fs` which will load the root file system and store it in NAND flash. The content of the `update_fs` variable is explained in section 4.3.5 , but for now just run the command.

```
uboot> run update_fs

...
NAND write: device 0 offset 0x500000, size 0x400000
 4194304 bytes written: OK
```

2. When the root file system has been stored in NAND flash it is time to load and boot the Linux kernel. A variable named `mtdboot` is available in the default environment. This variable will setup the boot arguments to use a root file system in NAND flash (in an MTD partition), load the kernel and then boot it. The content of the `mtdboot` variable is explained in section 4.3.5 . Run the command.

```
uboot> run mtdboot
```

3. There will be a lot of output in the terminal application when the kernel boots. Only a portion has been included below.

```
## Booting kernel from Legacy Image at 80100000 ...
   Image Name:   Linux-2.6.27.8
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    1602272 Bytes =  1.5 MB
   Load Address: 80008000
   Entry Point:  80008000
   Verifying Checksum ... OK
   Loading Kernel Image ... OK
OK

Starting kernel ...

Uncompressing
Linux.....................................................................
done, booting the kernel.

ÿLinux version 2.6.27.8 (user@bagvapp) (gcc version 3.4.5) #1 PREEMPT Mon Sep 28
09:51:45 CEST 2009
CPU: ARM926EJ-S [41069264] revision 4 (ARMv5TEJ), cr=00053177
Machine: Embedded Artists LPC3250 OEM board with the LPC3250 Microcontroller
Memory policy: ECC disabled, Data cache writeback
CPU0: D VIVT write-back cache
CPU0: I cache: 32768 bytes, associativity 4, 32 byte lines, 256 sets
CPU0: D cache: 32768 bytes, associativity 4, 32 byte lines, 256 sets
Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 16256
Kernel command line: root=/dev/mtdblock3 rw rootfstype=jffs2 ip=192.168.5.234
ea_ethaddr=00:1a:f1:00:00:00 console=ttyS0,115200n8

...
```

```
mmc0: host does not support reading read-only switch. assuming write-enable.
mmc0: new SD card at address e624
mmcblk0: mmc0:e624 SU256 247040KiB
 mmcblk0: p1
IP-Config: Guessing netmask 255.255.255.0
IP-Config: Complete:
     device=eth0, addr=192.168.5.234, mask=255.255.255.0, gw=255.255.255.255,
     host=192.168.5.234, domain=, nis-domain=(none),
     bootserver=255.255.255.255, rootserver=255.255.255.255, rootpath=
VFS: Mounted root (jffs2 filesystem).
Freeing init memory: 100K
init started: BusyBox v1.11.2 ()
starting pid 296, tty '': '/etc/rc.d/rcS'
Mounting /proc and /sys
Setting the hostname to nxp
Mounting filesystems
scsi 0:0:0:0: Direct-Access     SanDisk  Cruzer           8.02 PQ: 0 ANSI: 0 CCS
sd 0:0:0:0: [sda] 15704063 512-byte hardware sectors (8040 MB)
sd 0:0:0:0: [sda] Write Protect is off
sd 0:0:0:0: [sda] Assuming drive cache: write through
sd 0:0:0:0: [sda] 15704063 512-byte hardware sectors (8040 MB)
sd 0:0:0:0: [sda] Write Protect is off
sd 0:0:0:0: [sda] Assuming drive cache: write through
 sda: sda1
sd 0:0:0:0: [sda] Attached SCSI removable disk
scsi 0:0:0:1: CD-ROM            SanDisk  Cruzer           8.02 PQ: 0 ANSI: 0
mount: mounting usbfs on /proc/bus/usb failed: No such file or directory
Starting syslogd and klogd
Running sysctl
Setting up networking on loopback device:
Setting up networking on eth0:
/etc/rc.d/init.d/network: line 149: udhcpc: not found
Starting inetd:
Starting the boa webserver:
starting pid 359, tty '': '-/bin/sh'
[root@nxp /]#
```

4. Linux is now up-and-running.

3 Using the Linux Target Image Builder

Note: Embedded Artists patches won't be checked in to LTIB until earliest the end of week 42. Meanwhile use the precompiled images or if you are an experienced Linux user download the needed patches from Embedded Artists support site.

3.1 Introduction

The Linux Target Image Builder (LTIB) system will be used to build the u-boot, Linux kernel and root file system. LTIB ease the build and deployment process of several components needed in a Linux system. Besides the bootloader and kernel a lot of needed utilities, modules and libraries are included and will be configured and built automatically by LTIB. For more information about LTIB, go to http://ltib.org.

This chapter describes how you install LTIB and all necessary packages in a Fedora 11 Linux distribution. It will be explained how you can download a Fedora 11 distribution as a VMware virtual appliance and run it in a VMware Player.

Even though you are an experienced user and don't intend to run Fedora 11 you will find information about, for example, packages that are needed by LTIB.

3.2 Setting up a Fedora 11 Distribution

If you are an experienced Linux user and already have your own Fedora distribution (or another Linux distribution) you can skip this section and go to either section 3.2.3 to see which packages that needs to be installed or directly to section 3.3 for instructions of how to install LTIB.

3.2.1 Download and Start the VMware Appliance

1. Download Fedora 11 as a VMware appliance from http://www.bagside.com/bagvapp. When writing this document it was packed in a 7-Zip compressed file named `f11.7z`.
2. You also need 7-Zip to unpack the file. You can download this utility from http://www.7-zip.org.
3. In order to use the VMware appliance you need to install VMware Player. Download it from http://www.vmware.com/products/player/ and follow instructions to install the player.
4. Unpack the `f11.7z` file in a location of your choice.
5. By default this appliance has been setup to run NAT based Ethernet interface. This won't work since we would like to connect to the appliance using TFTP and NFS. Locate the file named `fed11.vmx` and open that file in a text editor.
6. In the fed11.vmx file locate the row below.

```
ethernet0.connectionType = "nat"
```

7. Change from "nat" to "bridged" and save the file (Note: from version 2.5.x it is possible to do these changes from within the VMware Player menu instead of modifying the vmx file).

```
ethernet0.connectionType = "bridged"
```

8. Now double-click in the `fed11.vmx` file to start the VMware Player. If vmx files haven't been associated with VMware Player you can start VMware Player manually and then select the Open command and locate the vmx file.
9. When Fedora has started you will see a Login dialog where you can select which user to login, see Figure 3. Choose the bagside user and enter "bagside" as password.

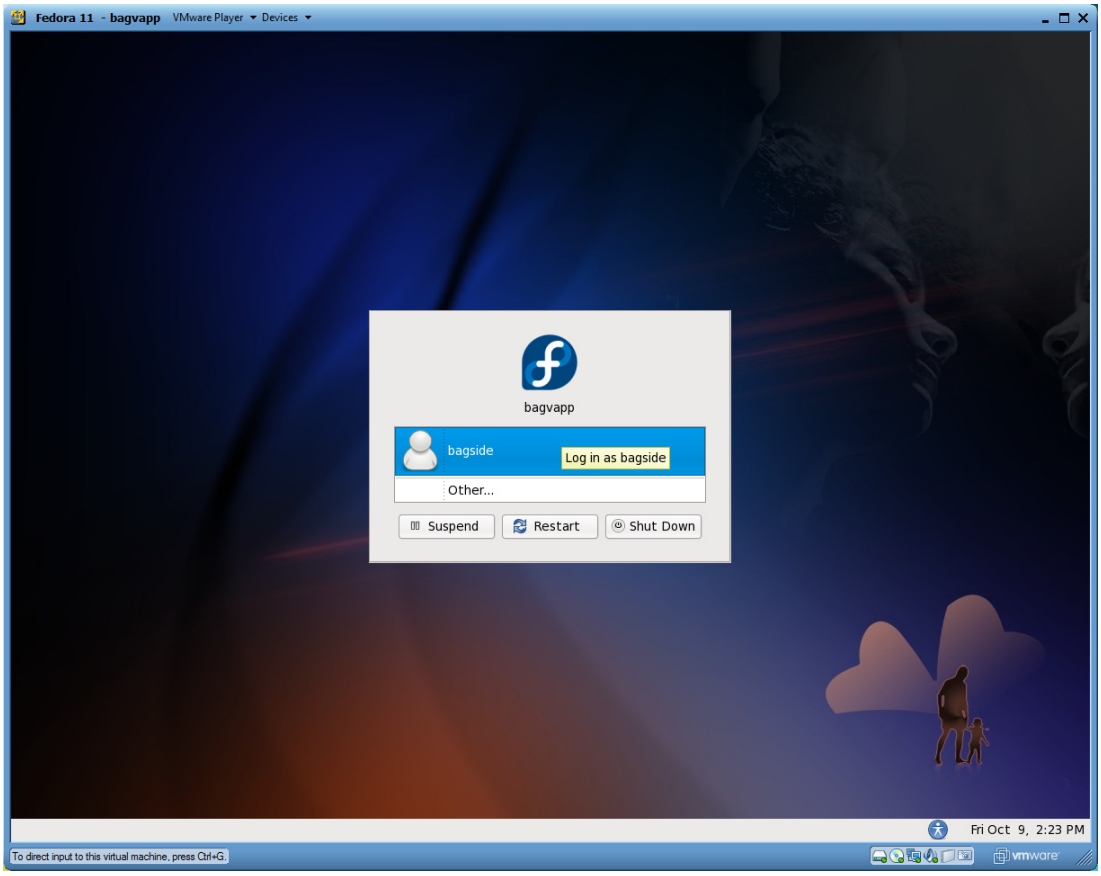

Figure 3 Fedora Login Screen

3.2.2 Adapt Fedora and Setup a New User

1. By default the keyboard layout is a US Layout. If you have a different keyboard layout change it by following these instructions.
    a. Go to the System → Preferences → Keyboard menu
    b. Click on the Layouts tab
    c. Click the "Add" button
    d. Select your keyboard layout (country) and click the "Add" button.
    e. Select which layout to be the default layout. You can also remove the layouts you don't intend to use.
    f. Click the "Close" button
2. To have consistent instructions we will create a new user named "user".
    a. Open a terminal application: Applications → System Tools → Terminal

b. Run the `adduser` command in the terminal. You need to use the `sudo` command to get administrator privileges and will be asked for a password. The password for your current user is "bagside".

```
$ sudo adduser user
```

c. Give a new password to the user "user" by using the `passwd` command. Select the password of your choice (a password you will remember).

```
$ sudo passwd user
```

3. Now we also need to give administrator rights to this user and also prepare rights needed by LTIB.
   a. Start the visudo tool to edit the sudoers file (where rules are specified)

```
$ sudo visudo
```

   b. The sudoers file will be opened. Locate the lines that looks like below.

```
root      ALL=(ALL)    ALL
bagside   ALL=(ALL)    ALL
```

   c. Press the 'i' key on your keyboard to enter insert mode and add the lines below after the line starting with "bagside". The second line is needed by LTIB and means that no password will be requested when using the rpm command.

```
user      ALL=(ALL)    ALL
user      ALL=(ALL)    NOPASSWD: /bin/rpm, /opt/ltib/usr/bin/rpm
```

   d. Press the ESC key on your keyboard to exit insert mode.
   e. Enter ":wq" to save your changes and exit visudo

```
:wq
```

4. Now you need to login as the new user.
   a. Go to System → Log out
   b. Click the "Log Out" button.
   c. When you are at the Login dialog select "user" and enter the password you selected in step 2 to login "user".
5. When you have logged in as "user" you probably need to change the keyboard layout for this user as described in step 1 above.
6. Since a terminal application is used often it is convenient to add shortcuts to this application.
   a. Go to Applications → System Tools
   b. Right-click on Terminal and select "Add this launcher to panel"
   c. Right-click again on Terminal and select "Add this launcher to desktop"

3.2.3 Install Necessary Packages

LTIB will require a number of packages to be installed in your Linux distribution before you can actually use LTIB. If these packages haven't been installed LTIB will usually complain and list the packages that are missing. LTIB could also fail without listing any package. The instructions below install the packages that are missing in the Fedora distribution we are using.

Please note that we don't display the output and progress of an installation of a package in the instructions below. You will, for example, be asked if it is ok to download a package. Answer 'y' on these questions.

1. Install a CVS client in order to checkout LTIB files

```
$ sudo yum install cvs
```

2. LTIB is using Perl and need some Perl related modules

```
$ sudo yum install perl-libwww-perl
$ sudo yum install perl-ExtUtils-MakeMaker
```

3. A few other packages are also needed by LTIB

```
$ sudo yum install tcl
$ sudo yum install zlib-devel
$ sudo yum install rpm-build
$ sudo yum install ncurses-devel
$ sudo yum install bison
$ sudo yum install gettext
```

3.2.4 Setup a TFTP Server

This section describes how you setup a TFTP server in Fedora and makes it accessible from other computers on your network. The TFTP server can be used to transfer files to the target board, for example, download the kernel image by the u-boot.

1. Install a TFTP server.

```
$ sudo yum install tftp-server
```

2. Open the TFTP server configuration file.

```
$ sudo gedit /etc/xinetd.d/tftp &
```

3. The file will be opened in a text editor. Locate the `server_args` variable which contains the directory that will be used as a root directory by the TFTP server.

```
server_args     = -s /var/lib/tftpboot
```

4. Change it to instead use the user's home directory as root.

```
server_args     = -s /home/user
```

5. Now it is time to enable the TFTP server. Run the commands below to do this.

```
$ sudo /sbin/chkconfig tftp on
$ sudo /sbin/service xinetd start
```

6. By default Fedora has setup strict firewall rules that may prohibit access to the TFTP server over the network. We take a pretty drastic approach in these instructions and completely disable iptables.

```
$ sudo /etc/init.d/iptables stop
$ sudo /etc/init.d/ip6tables stop
```

7. We also need to configure SELinux.
   a. Go to System → Administration → SELinux Management
   b. Enter "bagside" as password.
   c. Select "Process Domain" and type tftp in the filter box and press Enter.
   d. Select the tftpd and then click on the "Permissive" button.
8. In order for the TFTP server to access the files in the `/home/user` directory we must add execute permissions (list content) to the `user` directory.

```
$ cd /home
$ sudo chmod a+x user
```

9. You are now ready to use the TFTP server.

3.2.5 Setup a NFS Server

A NFS (network file system) mounted root file system is quite convenient to use during development of a Linux system. The actual root file system will then be located on the development computer and not on the target board, but the target board gets access to the file system using the NFS protocol.

This section describes how to setup a NFS server in Fedora.

1. The Fedora distribution we are working with already has a NFS server installed, but we need to make a part of the file system accessible by editing the `/etc/exports` file.

```
$ sudo gedit /etc/exports
```

2. Add the following line to the opened file (note that it is only one line). Also note that if you are not using the 192.168.x.x network (IP addresses in this address range) you need to change this part of the line.

```
/home/user/ltib/rootfs
    192.168.0.0/255.255.0.0(rw,no_root_squash,no_subtree_check,sync)
```

3. Stop the NFS server.

```
$ sudo /sbin/service nfs stop
```

4. Start the NFS server (it will now use the changed `exports` file).

```
$ sudo /sbin/service nfs start
```

3.3 Setup an Ubuntu 9.04 Distribution

TBD

3.3.1 Download and Start the VMware Appliance

TBD

3.3.2 Install Necessary Packages

TBD

3.3.3 Setup a TFTP Server

TBD

3.3.4 Setup a NFS Server

TBD

3.4 Install LTIB and Build the Images

This section describes how you install LTIB, selects the configuration applicable for the Embedded Artists LPC3250 OEM Board and starts the build process where the u-boot, Linux kernel and root file system will be built.

1. Open a web browser and go to http://ltib.org
2. Click on the Download link in the left menu (below the Resources title).
3. In the "Quick install" section you will find a link to the netinstall.txt file. Right-click on this file and save it in your home directory (/home/user).
4. Open up a terminal application and run the netinstall script.

```
$ cd /home/user
$ perl netinstall.txt
```

5. Select 'Y' to continue the installation.
6. Click Enter to use the default installation directory.
7. When LTIB files have been downloaded change directory.

```
$ cd ltib
```

8. Now start LTIB configuration. The first time you run the configuration it will take quite a long time since a lot of packages must be downloaded.

```
$ ./ltib
```

9. After a while a configuration menu will appear, see Figure 4. Hit Enter to select Platform.

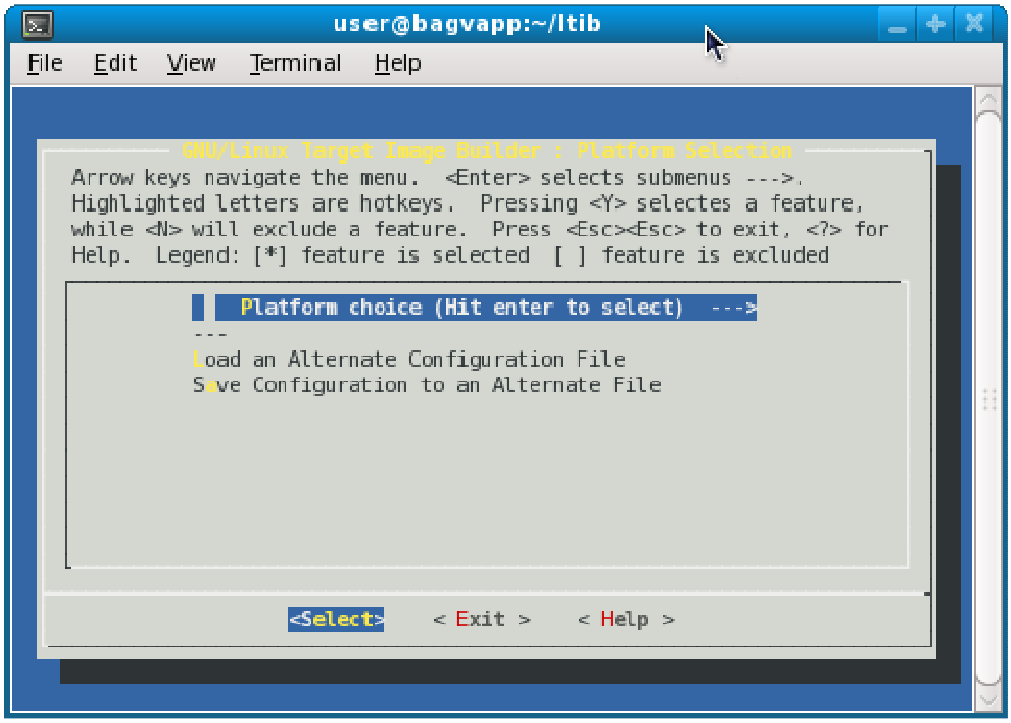

Figure 4 LTIB Platform Selections Menu

10. In the platform choice menu select “Embedded Artists LPC3250 OEM Board with the NXP LPC32XX SoC”.

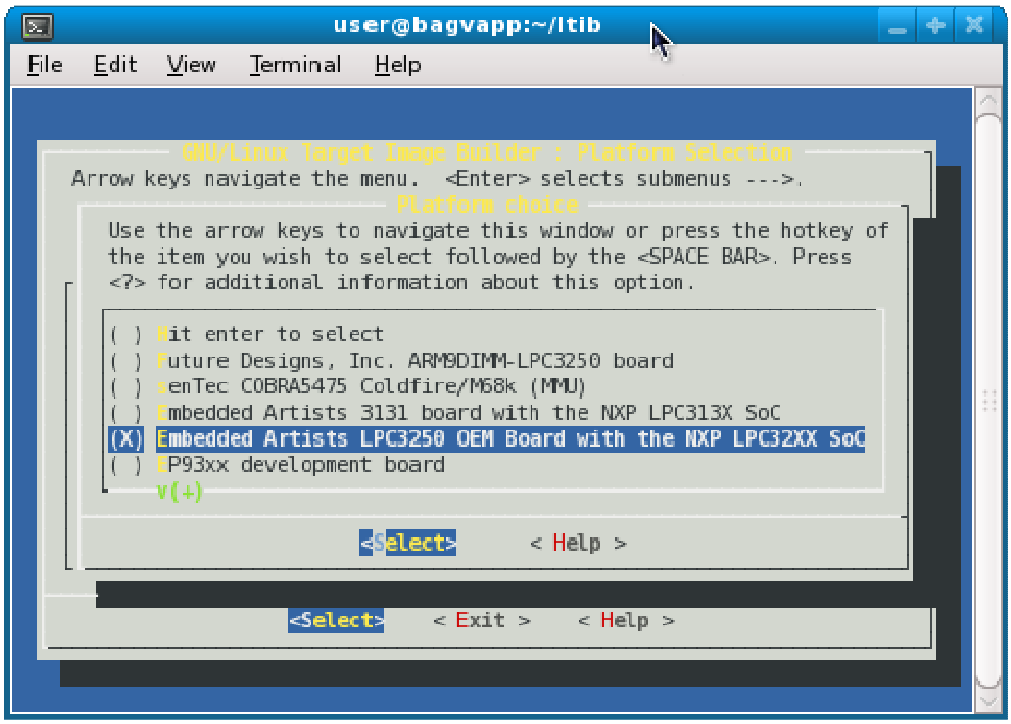

Figure 5 LTIB Platform Choice Menu

11. Click the “Exit” button.
12. Select “Yes” when asked to save the configuration.
13. A new configuration menu will appear, see Figure 6. For now just click the Exit button and go with the default configuration.

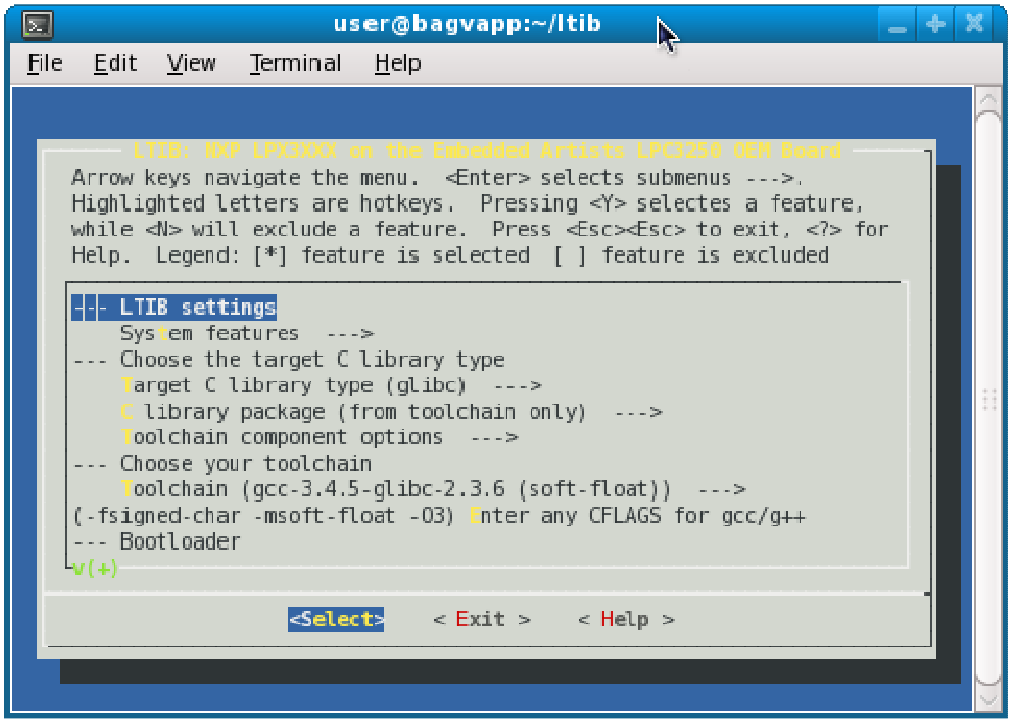

Figure 6 LTIB Platform Configuration Menu

14. Select “Yes” when asked to save the configuration. LTIB will now download necessary packages, build u-boot, build the Linux kernel and create a root file system.
15. If everything builds successfully you will have something similar to the example below in your terminal.

```
…
Filesystem stats, including padding:

   Total size            = 9160k
   Total number of files = 455

Started: Wed Sep 30 15:39:38 2009
Ended:   Wed Sep 30 15:46:01 2009
Elapsed: 383 seconds

Build Succeeded
```

16. You will find the Linux kernel and u-boot images at the following location.

```
/home/user/ltib/rootfs/boot/uImage
/home/user/ltib/rootfs/boot/u-boot.bin
```

17. The JFFS2 formatted root file system image will be located here.

```
/home/user/ltib/rootfs.jffs2
```

18. The complete root file system is located in the following directory.

```
/home/user/ltib/rootfs/
```

19. You are now ready to deploy your Linux system to the target board. Chapter 2 describes a way of deploying the system using a USB memory stick. Chapter 4 describes the u-boot in more detail and also presents more booting options.

4 Universal Boot Loader - u-boot

4.1 Introduction

TBD

4.2 Console / Environment

The u-boot supports a command line interface usually accessed via a terminal application, such as Tera Term, connected to the serial port associated with the development board. The command line interface allow you to manually type in boot commands or update the environment variables that can later be used as boot options.

4.2.1 Commands

It is possible to discover which commands are available by using the `help` command.

```
uboot> help
```

When issuing the `help` command a list of all the available commands will be presented. These commands are the ones that have been selected to be supported when configuring the u-boot. If more information is needed about a specific command type `help` followed by the name of the command. The example below gives you more information about the `setenv` command.

```
uboot> help setenv
```

Below is a list of some of the commands used to modify, list and execute variables in the u-boot environment.

- `printenv` – This command will print the u-boot environment.
- `setenv` – This command is used to set the value of an environment variable. If the variable doesn't exist when calling `setenv` it will be created.
- `saveenv` – This command will save any changes done to the environment and must be called after `setenv` has been used in order for the changes to be saved persistently.
- `run` – execute the commands found in an environment variable

```
uboot> setenv serverip 192.168.0.110
uboot> saveenv
uboot> run mtdboot
```

4.2.2 Network Related Variables

The u-boot environment contains a number of variables that are network related, i.e., related to communication over a network. Make sure these variables are correctly setup for your network.

- `ethaddr` – Specifies the Ethernet/MAC address that will be assigned to the development board. The address will also be forwarded to the Linux kernel via the `ea_ethaddr` argument in the boot argument list, sections 4.3.4 4.3.5 and 4.3.6 contain example of its usage. Make sure your board gets a unique address.

- `ipaddr` – Specifies the IP address that will be assigned to the development board. The address will be forwarded to the Linux kernel via the `ip` argument in the boot argument list, sections 4.3.4 4.3.5 and 4.3.6 contain example of its usage. Make sure this address is a valid and unique address on your network. If you instead would like to use dynamically assigned IP addresses see ***section XXX***.
- `serverip` – This variable specifies the IP address of the TFTP server used when downloading images using TFTP. Set it to the IP address of the computer running your TFTP server. ***(Something about how to retrieve the IP address in a Linux distribution)***.
- `netmask` – Defines a mask used to divide your network into subnets. On most office and home networks the netmask is set to 255.255.255.0 which means that the 3 first octets of the IP address is fixed while the last can vary.

4.2.3 Boot Related Variables

The variables listed below are all related to the boot process.

- `bootargs` – This variable the boot arguments sent to the Linux kernel. It usually contains settings for the console and where to find the root file system.
- `bootcmd` – This variable contains the boot command(s) that will be run during auto booting.
- `bootdelay` – This variable defines the delay in seconds until an autoboot will take place. Autoboot can be cancelled by hitting any key during boot.

4.3 Booting Options

Setting up a booting option means specifying from which source the Linux kernel should be loaded as well as specify where to find the root file system.

4.3.1 Kernel from USB Memory Stick

Most computers today have a USB connection and most operating systems support USB and have device drivers for USB mass storage devices. This makes it simple to use a USB memory stick to transfer boot images from the development computer to the development board.

For this purpose the `usb` and `fatload` commands are used to access the memory stick and transfer the images to the development board. In the default environment there is a variable named `loadkernel_usb` which illustrates how to load the kernel from a USB memory stick.

```
loadkernel_usb=usb start; fatload usb 0 $(loadaddr) uImage; usb stop
```

1. First the USB interface must be initialized with the `usb start` command.
2. The kernel image (`uImage`) is then loaded via the USB interface, device 0 and to the load address specified by the `loadaddr` variable.
3. The final step is to stop the USB interface.

Note: *For some USB memory sticks you might also need to specify which partition to use, not only which device to use. In the example below the* `loadkernel_usb` *variable is updated to also specify partition 0 on device 0. Please note how the backslash character needs to be used before the semicolon when updating a variable.*

```
uboot> setenv loadkernel_usb usb start\;fatload usb 0:0 $(loadaddr)
uImage\; usb stop
uboot> saveenv
```

If you would like to load the kernel from a USB memory stick make sure that the `loadkernel` variable contains the content of the `loadkernel_usb` variable.

```
uboot> setenv loadkernel $(loadkernel_usb)
uboot> saveenv
```

4.3.2 Kernel from TFTP Server

During development of the Linux kernel it is convenient to use the Trivial File Transfer Protocol (TFTP) to download a newly created kernel. The development cycle will be much shorter compared to when you would need to copy the kernel image to, for example, a USB memory stick.

The `tftpboot` command is used when downloading images from a TFTP server. Before using this command you have to make sure the network related variables described in section 4.2.2 are correctly setup for your network. In the default environment there is a variable named `loadkernel_tftp` which illustrates how to load the kernel from a TFTP server.

```
loadkernel_tftp=tftpboot $(loadaddr) uImage
```

If you would like to load the kernel from a TFTP server make sure that the `loadkernel` variable contains the content of the `loadkernel_tftp` variable.

```
uboot> setenv loadkernel $(loadkernel_tftp)
uboot> saveenv
```

4.3.3 Kernel Stored in NAND Flash

The LPC3250 OEM Board comes with a large NAND flash that can host the Linux kernel for fast access without the need for a USB memory stick or network access.

Before the kernel can be loaded from NAND flash the NAND flash must be updated with the kernel. For this purpose the default environment has been setup with a variable named `update_kernel`.

```
update_kernel=run loadkernel; nand erase $(nand_kernel_off)
$(nand_kernel_sz); nand write.jffs2 $(loadaddr) $(nand_kernel_off)
$(nand_kernel_sz)
```

1. The `loadkernel` variable is used by the `update_kernel` variable to load the kernel to SDRAM (address given by `loadaddr`). Section 4.3.1 and section 4.3.2 describe how the `loadkernel` variable can be setup to load the kernel from either a USB memory stick or a TFTP server.
2. The second step is to erase the part of the NAND flash that will be used to store the kernel. The offset into the NAND flash as well as the maximum size of the kernel is given by the variables `nand_kernel_off` and `nand_kernel_sz`.
3. The last step is to write the kernel image to NAND flash.

In the default environment there is a variable named `loadkernel_nand` which illustrates how to load the kernel from NAND flash.

```
loadkernel_nand=nboot.jffs2 $(loadaddr) 0x0 $(nand_kernel_off)
```

1. The `nboot` command is used to load the kernel from NAND device `0x0` and offset `nand_kernel_off` to SDRAM at address `loadaddr`.

If you would like to load the kernel from NAND flash make sure that the `loadkernel` variable contains the content of the `loadkernel_nand` variable.

```
uboot> setenv loadkernel $(loadkernel_nand)
uboot> saveenv
```

4.3.4 Root File System NFS Mounted

During the development phase of your Linux system it is convenient to be able to easily update the root file system without having to transfer the file system to the development board. Using NFS (Network File System) will allow you to do this.

1. Make sure you have NFS setup on your development computer as described in section 3.2.5 or section 3.3.4 .
2. Setup the `rootpath` variable in the u-boot environment to point to the location of your exported root file system directory.

```
rootpath=/home/user/ltib/rootfs
```

3. Use the `nfsboot` variable to boot Linux. This variable will setup the `bootargs` variable in a way where Linux use a NFS mounted root file system.

```
nfsboot=setenv bootargs root=/dev/nfs rw nfsroot=$(serverip):$(rootpath)
ip=$(ipaddr) ea_ethaddr=$(ethaddr) console=ttyS0,115200n8;run
loadkernel;bootm $(loadaddr)
```

4.3.5 Root File System in NAND Flash

The root file system can be stored in the large NAND flash available on the LPC3250 OEM Board. This is typically the place where the root file system will be placed after the development phase allowing the Linux system to be more stand-alone.

1. The first step is to update the NAND flash with the root file system. Two variables have been setup to load the root file system from either a TFTP server or from a USB memory stick. See the note in section 4.3.1 about possible problems with USB memory sticks.

```
loadrootfs_tftp=tftpboot $(loadaddr) rootfs.jffs2
loadrootfs_usb=usb start;fatload usb 0 $(loadaddr) rootfs.jffs2;usb stop
```

2. Set the `loadrootfs` variable to the content of either the `loadrootfs_tftp` or `loadrootfs_usb` variable as in the example below.

```
uboot> setenv loadrootfs $(loadrootfs_usb)
uboot> saveenv
```

3. The `update_fs` variable has been setup to update the NAND flash with the root file system. First the `loadrootfs` variable will be used to load the root file system to SDRAM. Then the NAND flash will be erased at the offset specified by

nand_rootfs_off and size specified by nand_rootfs_sz. The last step is to write the file system to NAND flash.

```
update_fs=run loadrootfs; nand erase $(nand_rootfs_off)
$(nand_rootfs_sz); nand write.jffs2 $(loadaddr) $(nand_rootfs_off)
$(nand_rootfs_sz)
```

4. To actually update the NAND flash run the update_fs variable.

```
uboot> run update_fs
```

5. Use the mtdboot variable to boot Linux. This variable will setup the bootargs variable in a way where Linux looks for the root file system in NAND flash. See ***section XXX*** for an explanation of how the NAND flash is partitioned.

```
mtdboot=setenv bootargs root=/dev/mtdblock3 rw rootfstype=jffs2
ip=$(ipaddr) ea_ethaddr=$(ethaddr) console=ttyS0,115200n8; run
loadkernel;bootm $(loadaddr)
```

4.3.6 Root File System on MMC/SD Card

TBD

5 Accessing Peripherals in Linux

5.1 Introduction

TBD

5.2 Display

TBD

5.3 Touch Screen

TBD

5.4 Network

TBD

5.5 Memory Card

TBD

5.6 USB Host

TBD

5.7 LEDs and Buttons

TBD