

Future Designs, Inc.

**SOM9DIMM-LPC3250
Linux BSP Manual**

For use with
ARM Touch Screen LCD Kit

Copyright ©2010, Future Designs, Inc., All Rights Reserved

Table of Contents

1. Introduction	4
Copyrights and Limitations	4
Where to Start	4
Required Hardware and Software	4
Required Bootloader	4
2. BSP overview	5
Supported Hardware	5
BSP Components	5
LTIB integration	5
u-boot 1.3.3	5
Linux 2.6.27.8	6
3. Building the Linux System with LTIB	8
Preparing Packages under Fedora 9	8
Downloading LTIB	9
Configuring Super User Access	9
LTIB Configure and Build Process	9
Customizing Devices	12
4. Flash Programming	13
Flashing Linux into SOM9DIMM-LPC3250	13
Flashing U-Boot into SOM9DIMM-LPC3250	14
5. General Questions and Answers	15

Information in this document is provided solely to enable the use of Future Designs products. FDI assumes no liability whatsoever, including infringement of any patent or copyright. FDI reserves the right to make changes to these specifications at any time, without notice. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Future Designs, Inc. 2702 Triana Blvd, Huntsville, AL 35805

NOTE: The inclusion of vendor software products in this kit does not imply an endorsement of the product by Future Designs, Inc.

© 2010 Future Designs, Inc. All rights reserved.

Microsoft, MS-DOS, Windows, Windows XP, Microsoft Word are registered trademarks of Microsoft Corporation.

Other brand names are trademarks or registered trademarks of their respective owners.

FDI PN: M00014

Revision: 1.3, 2/24/2010 9:30:00 AM

Printed in the United States of America

1. Introduction

The SOM9DIMM-LPC3250 Module provides a quick and easy solution for implementing an ARM9EJ-S design by providing the basic functions necessary for a product on an easy to use DIMM module. The DIMM module uses an industry standard 200 pin SO-DIMM interface. These sockets are utilized by virtually every laptop on the market.

This DIMM Module is compatible with FDI's Family of ARM Touch Screen LCD Kits but can also be used for custom platform development or customer applications.

This document details the Linux 2.6.27.8 BSP for the SOM9DIMM-LPC3250 Module. The BSP provides a port of Linux for the DK-TS-KIT platform (which includes an SOM9DIMM-LPC3250 CPU Module).

This document covers the following areas:

- System and target board setup
- Using LTIB to build a complete Linux development system
- Peripheral support in the Linux 3250 BSP
- Board customization

Copyrights and Limitations

The SOM9DIMM-LPC3250 BSP is provided free of charge and with no support from Future Designs, Inc. Portions of the BSP are copyrighted by NXP Semiconductors.

Where to Start

This document assumes you will use the LTIB (Linux Target Image Builder) build system to compile and deploy your system. LTIB handles most of the hard work for you providing an easy to use menu interface.

If you are an advanced user that knows how to set up, build, and deploy a Linux system, you may want to skip to just getting the patch files for u-boot and the Linux kernel.

Required Hardware and Software

To develop Linux for the SOM9DIMM-LPC3250, a host PC running the Linux operating system is required. This BSP and supporting tools have been tested with the Fedora 9 Linux release. Other releases may work fine, but only Fedora 9 has been tested. The LTIB is constantly being updated and may support other platforms.

If installing Fedora 9 on a separate machine is not an option, development can be done under a virtual machine using VMWare. A virtual image of Fedora 9 can be downloaded from <http://www.vmware.com/appliances/directory/1253>.

Required Bootloader

SOM9DIMM-LPC3250's are shipped with the SIBL (Secondary Boot Loader). The SIBL's primary job is to setup the CPU, SDRAM, and load u-boot. This document assumes that the SIBL is intact and already running on the target. Check the website www.teamfdi.com/dk-ts-kit for more details if your board no longer has SIBL running.

2. BSP overview

This section gives a brief overview of the BSP contents, features, and supported peripherals. This BSP has been developed to support Linux kernel 2.6.27.8.

The term 'BSP' refers to the files specific to the SOM9DIMM-LPC3250 Linux port. These include the kernel and u-boot patch file(s), pre-built GCC toolchain, and necessary LTIB files to build a Linux distribution.

Not all the files are necessary to build a Linux based system for the SOM9DIMM-LPC3250. Developers are welcome to use their own toolchains or develop a Linux system without using LTIB.

Supported Hardware

This BSP supports the NXP LPC3250 MCU and most of its built-in peripherals. Several external peripherals for the DK-TS-KIT are also supported.

BSP Components

LTIB integration

LTIB (Linux Image Target Builder) provides a convenient method to build the bootloader, kernel image, and root filesystem and then deploy them to your target. LTIB provides an environment that allows easy setup, configuration, and build of the boot loader, Linux kernel, deployment methods, build tools, etc. If LTIB is used to develop a complete Linux system, the work of downloading and installing packages, building the root filesystem, and an assortment of other tasks is greatly simplified.

u-boot 1.3.3

u-boot is the Linux bootloader supported by this BSP. u-boot provides support for booting the Linux kernel through Ethernet or from NAND FLASH.

U-boot relies on the SIBL (Secondary Boot Loader) to pre-initialize the board and memory prior to executing. Once SIBL has initialized the board, it will load and start u-boot from NAND flash.

The SOM9DIMM-LPC3250 version of u-boot supports the following features:

- Persistent configuration of u-boot and boot parameters
- Kernel image boot from NAND FLASH or Ethernet
- Ethernet network configuration

[Linux 2.6.27.8](#)

LPC32xx Architecture Support

The following LPC32xx peripherals are supported as part of the core system architecture:

- Timer 0 (system tick)
- Interrupt controller
- Clock and power manager (Clock query and control for various drivers)
- DMA

LPC32xx Driver Support

The following Linux drivers are provided as part of the BSP:

- Watchdog timer (modified PNX4008 driver)
- Color LCD controller (AMBA CLCD driver)
- SD card controller (modified AMBA driver MMC driver with DMA)
- SMSC Ethernet controller (RMII based)
- Touchscreen controller
- Standard UARTs (up to 4)
- High speed UARTs (up to 3, console on ttyTX0 UART1)
- SPI (up to 2 implemented with the SSP interface)
- I2C (2 dedicated channels, 1 OTG channel)
- USB host with the ISP1301 transceiver
- MTD (NAND SLC controller)

DK-57[V]TS-3250 Board Driver Support

The following additional support is provided for the DK-TS-KIT board:

- PFC8563 RTC clock (I2C)

Some Linux drivers are new to the LPC32xx, while existing drivers were used when possible. The table below shows the kernel CONFIG_ values needed to enable the kernel support for that interface.

Functionality	Driver	Kernel CONFIG_ selection
VFP (hardware float)	ARM926 VFP driver	CONFIG_VFP
MTD (NAND)	LPC32xx NAND SLC driver	CONFIG_MTD_NAND_SLC_LPC32XX
Ethernet 10/100	LPC32xx Ethernet MII driver	CONFIG_LPC32XX_MII
Touchscreen	LPC32xx touchscreen driver	CONFIG_TOUCHSCREEN_LPC32XX
Standard serial ports	8250 UART driver	CONFIG_SERIAL_8250
High speed UARTs	LPC32xx high speed UART driver	CONFIG_SERIAL_HS_LPC32XX
I2C	PNX4008 I2C driver	CONFIG_I2C_PNX
SPI	LPC32xx SSP/SPI driver	CONFIG_SPI_LPC32XX
Watchdog timer	PNX4008 watchdog timer driver	CONFIG_LPC32XX_WATCHDOG
LCD	ARM PL110 driver	CONFIG_FB_ARMCLCD
I2S audio (ALSA) [1]	LPC32xx audio driver	CONFIG_SND_LPC3XXX_SOC
UDA1380 audio CODEC [1]	UDA1380 CODEC driver	CONFIG_SND_LPC3XXX_SOC_UDA1380
USB host	OHCI/PNX drivers	CONFIG_USB_OHCI_HCD
SD/MMC	ARM PL180 driver (modified)	CONFIG_MMC_ARMMMC
RTC	LPC32xx RTC driver	CONFIG_RTC_DRV_LPC32XX
Frame Buffer	ARM PrimeCell PL110 support	CONFIG_FB_ARMCLCD

[1] I2S is only needed if using I2S daughter card (not currently available).

Some interfaces require additional configurations as shown in the table below. For example, specific I2C ports can be enabled and disabled here to further customize the kernel build.

Functionality	Driver	Kernel CONFIG_ selection
Standard UART enables	8250 UART driver	CONFIG_MACH_LPC32XX_UART5_ENABLE CONFIG_MACH_LPC32XX_UART3_ENABLE CONFIG_MACH_LPC32XX_UART4_ENABLE CONFIG_MACH_LPC32XX_UART6_ENABLE
High speed UART enables	Hispeed UART drivers	CONFIG_MACH_LPC32XX_HSUART1_ENABLE CONFIG_MACH_LPC32XX_HSUART2_ENABLE CONFIG_MACH_LPC32XX_HSUART7_ENABLE
I2C channel enables	PNX4008 I2C driver	CONFIG_MACH_LPC32XX_I2C0_ENABLE CONFIG_MACH_LPC32XX_I2C1ENABLE CONFIG_MACH_LPC32XX_USBOTG_I2C_ENABLE
SPI channel enables	LPC32xx SSP/SPI driver	CONFIG_MACH_LPC32XX_SSP0_ENABLE CONFIG_MACH_LPC32XX_SSP1_ENABLE

The tables below shows the kernel CONFIG_ values needed to enable the kernel support for interfaces specific to a target board. These drivers are usually not related to the LPC32xx microcontroller.

Functionality	Driver	Kernel CONFIG_ selection
SMSC Ethernet PHY	SMSC PHY driver	CONFIG_SMSC_PHY
PFC8563 RTC clock	PCF8563 driver	RTC_DRV_PCF8563

Other options required by the DK-TS-KIT:

Functionality	Driver	Kernel CONFIG_ selection(s)
Enable SOM9DIMM-LPC3250 board	NA	CONFIG_MACH_SOM9DIMM3250
Audio on I2S1 [1]	LPC32xx audio driver	CONFIG_SND_LPC32XX_USEI2S1

[1] I2S is only needed if using I2S daughter card (not currently available).

Board Target

The entire BSP is targeted to the SOM9DIMM-LPC3250 on an DK-TS-KIT. Specific board drivers functions needed by architecture drivers are defined in the kernel file tree at ./arch/arm/mach-lpc32xx/board-arm9dimmm.c

To port the Linux kernel to new boards, only a new board-<platform>.c file is needed with the required architecture functions supported by the board.

3. Building the Linux System with LTIB

This section explains how to configure and build a Linux system for the DK-57TS-LPC3250 kit using LTIB. The following steps that go through an example of setting up LTIB on a new install of Fedora 9 in a VMWare session. The steps can be used to validate other development systems.

Preparing Packages under Fedora 9

LTIB needs to be downloaded and installed on the Linux host machine. Installation can take over an hour (depending on Internet connection and speed of processor). Once installed, all command line tools for configuring and compiling the kernel with the cross compiler tools will be installed and ready to use. To start this process,

1. Download vmplanet.net version of Fedora 9 from <http://www.vmware.com/appliances/directory/1253>
2. Unzip to a folder and run the VMWare image.
3. When Linux boots in the VMWare session, log in as vmplanet with password vmplanet.net
4. If the Security updates available dialog appears, click "Do not warn me again".
5. Go to Applications->System Tools and right click on "Terminal". Select Add this launcher to the panel.
6. Click on the terminal icon that appears at the top. This terminal will be used to enter commands.
7. Using the menu bar at the top, go to System->Administration->Add/Remove Software.
8. Enter "cvs" and click **Find**.
9. Select "cvs-1.11.22-13.fc9 (i386)" and click **Install**.
10. Enter password "vmplanet.net" and select "Remember authorization", but don't select "For this session only". Click on **Authenticate**.
11. Enter "perl-libwww" in the search field and click **Find**. Select "perl-libwww-perl-5.808-7.fc9 (noarch)" and click **Install**.
12. Enter "gcc" in the search field and click **Find**. Select "gcc-4.3.0-8 (i386)" and click **Install**.
13. Repeat the above package install steps using the following table:

Package name:	Package to install:
gcc-c++	gcc-c++-4.3.0-8 (i386)
zlib-devel	zlib-devel-1.2.3-18.fc9 (i386)
rpm-build	rpm-build-4.4.2.3-2.fc9 (i386)
wget	wget-1.11.1-1.fc9 (i386)
ncurses-devel	ncurses-devel-5.6.16.20080301.fc9 (i386)
m4	m4-1.4.10-3.fc9 (i386)
bison	bison-2.3-5.fc9 (i386)
patch	
gettext	gettext-0.17-4.fc9 (i386)
make	make-1:3.81-12.fc9 (i386)
tcl	tcl-1:8.5.1-4.fc9 (i386)
perl-ExtUtils-MakeMaker	perl-ExtUtils-MakeMaker-6.36-20.fc9 (i386)

14. Close Add/Remove Software dialog.

Downloading LTIB

15. Click on the Firefox icon at the top and go to webpage <http://www.ltib.org/pages/netinstall.txt>
16. Right click on the text that appears and click Save As.
17. On the following dialog, save the file netinstall.txt to the newly created ltib. Do this by clicking on "Browse for other folders", click on "vmplanet" on the left and then click **Save**.
18. A Downloads dialog box will appear. Close this dialog. Close Netscape.
19. Go to System->Administration->Users and Groups.
20. Enter password "vmplanet.net" and click **OK**.
21. Click on vmplanet in the list and click **Properties** icon. Select **Groups** tab. Scroll down the list and check "root". Click **OK**. Select File->Exit.

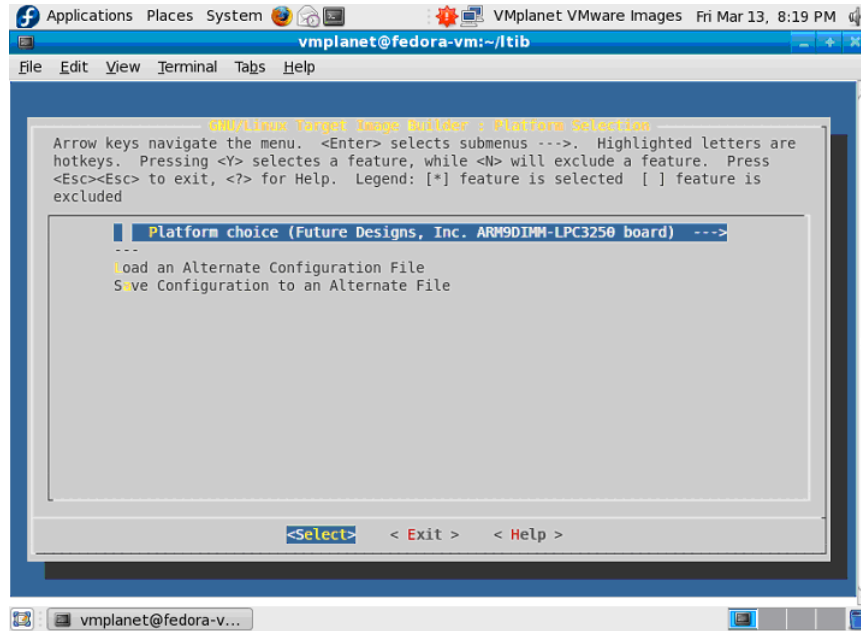
Configuring Super User Access

22. Go back to the terminal and enter the command "su" and enter the password "vmplanet.net"
23. Enter the command "/usr/sbin/visudo"
24. You are now in a vi editor. Add the line "vmplanet ALL = NOPASSWD: /bin/rpm, /opt/ltib/usr/bin/rpm" to the end of the file by pressing SHIFT-G, press i, and typing in the text. When done, press ESC and then type ":wq" to save and quit.
25. Enter command "exit" to leave super user mode.

LTIB Configure and Build Process

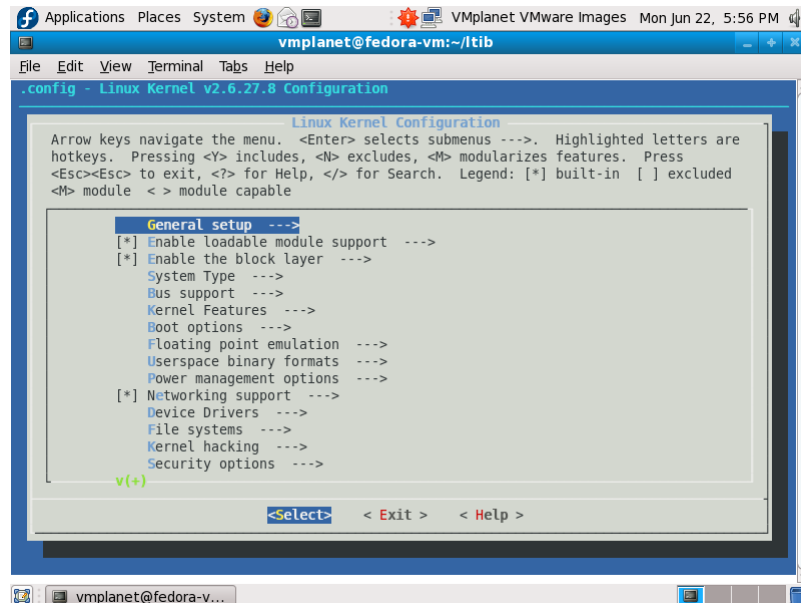
26. Enter command "perl netinstall.txt"
27. When asked "Do you want to continue ?", press ENTER.
28. When asked "Where do you want to install LTIB ? (/home/vmplanet/ltib), press ENTER.
29. Several files will be downloaded and the message, "LTIB download complete, your ltib installation has been placed in /home/vmplanet/ltib, to complete the installation, run the following commands:"
30. Enter command "cd ltib"
31. Enter command "./ltib" and you should see the message "Installing host support packages. This only needs to be done once per host, but may take up to an hour to complete ..."
32. Wait a very long time for the downloads to occur.

33. When the Platform Selection dialog appears, select “Future Designs, Inc. SOM9DIMM-LPC3250 board.” Select **Exit** and **Save**.



34. The LPC32xx and SOM9DIMM-LPC3250 platform and associated packages will be automatically downloaded and installed from the Internet.

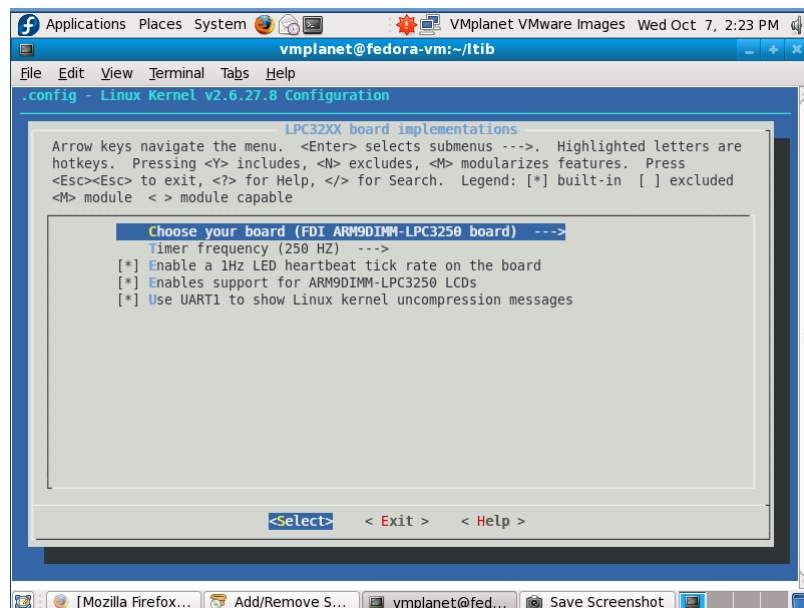
35. Eventually, the following LTIB configuration menu will appear:



36. Before continuing, several parameters must be checked. Most options are already selected.

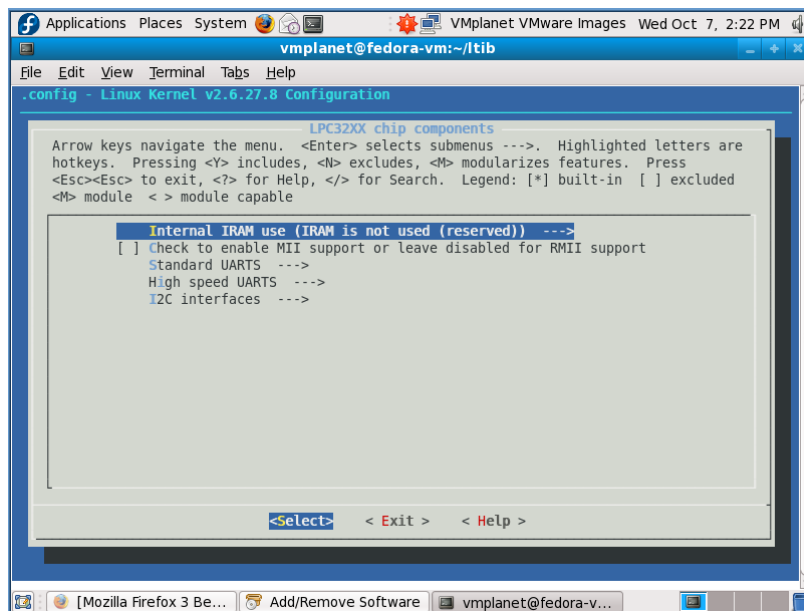
- 1) Select “System Type” by scrolling down and hitting ENTER
- 2) Scroll down to “LPC32XX board implementations” and press ENTER
- 3) The Timer frequency should be 250 Hz.
- 4) Make sure “FDI SOM9DIMM-LPC3250 board” is selected. When “Enable a 1Hz LED heartbeat tick rate” and “Enables support for the SOM9DIMM-LPC3250 LCD” appear, select them as well.

5) Also select “Use UART1 to show Linux kernel uncompressed messages”



6) When done, right arrow and select Exit.

7) Now select “LPC32xx chip components” and the following will appear:



8) “Use IRAM for the LCD frame buffer” and “Check to enable MII support” should be left unchecked.

9) Under “Standard UARTS” should have UART5 only selected (for now).

10) “High speed UARTS” should also only have “Enable high speed UART1” selected.

11) Under “I2C interfaces”: “Enable I2C0”, “Enable I2C1”, and “Enable the USB OTG I2C peripheral” should all be selected.

37. Leave most of the rest of the options alone. Changing options can cause the code to not compile correctly. If you later find you have entered an option incorrectly, enter the command “Itib –configure” and select the “Configure kernel” option to return to this set of menus.

38. For now, select Exit and enter Yes to save your configuration (if asked).
39. LTIB will immediately start compiling. If all goes well, the following screen should appear.

```

Applications Places System VMplanet VMware Images Mon Jun 22, 6:41 PM
vmplanet@fedora-vm:~/ltib/rootfs/boot
File Edit View Terminal Tabs Help
Filesystem stats, including padding:

    Total size           = 15532k
    Total number of files = 503

Your ramdisk exceeds the old default size of 4096k, you may need to
set the command line argument for ramdisk_size in your bootloader
allowing 10% free this gives 17085k . For instance, for u-boot:

setenv bootargs root=/dev/ram rw ramdisk_size=17085

creating an ext2 compressed filesystem image: rootfs.ext2.gz
genext2fs: Running in LTIB backwards compatibility mode: -i -> -N
creating a uboot ramdisk image: rootfs.ext2.gz.uboot
Image Name:   uboot ext2 ramdisk rootfs
Created:      Mon Jun 22 18:30:10 2009
Image Type:   ARM Linux RAMDisk Image (gzip compressed)
Data Size:    4312290 Bytes = 4211.22 kB = 4.11 MB
Load Address: 0x00000000
Entry Point:  0x00000000

Started: Mon Jun 22 17:50:46 2009
Ended:   Mon Jun 22 18:30:11 2009
Elapsed: 2365 seconds

Build Succeeded

[vmplanet@fedora-vm ltib]$
vmplanet@fedora-v...

```

Customizing Devices

40. Before continuing, the device table file needs to be edited to add the high speed serial port and MMC card to the configuration. Using your editor, open file “~/ltib/bin/device_table.txt”. Add the following line under the “#Normal system devices” group or at the end if they are not already in the list.

```
/dev/ttyTX c 666 0 0 204 196 0 1 3
```

41. Change the following line from:

```
/dev/console c 666 0 0 5 1 - - -
```

To the following line:

```
/dev/console c 666 0 0 204 196 - - -
```

42. Uncomment the following line from:

```
#/dev/input/event c 644 0 0 13 64 0 1 3
```

To the following line:

```
/dev/input/event c 644 0 0 13 64 0 1 3
```

43. Also add the following lines under the “#MTD stuff” group or at the end (if not already in the list):

```
/dev/mmcblk0 b 640 0 0 179 0 - - -
```

```
/dev/mmcblk0p1 b 640 0 0 179 1 - - -
```

44. Save and exit.
45. Enter command “cd ~/ltib” and force recompile of the rootfs’s devices again by entering the command “./ltib -p dev -f”. The software has now complete compiled and is ready for flashing.

4. Flash Programming

Flash programming can be done in a number of ways, but one of the most dependable methods is using the J-Link and J-Link Commander software.

To flash the file built using LTIB, find the following files and copy to a place that the J-Link Commander software can access (e.g., flash drive and/or network file system):

- Itib/rootfs/boot/ulmage ← Linux kernel
- Itib/rootfs.ext2.gz ← Root file system file
- Itib/rootfs/boot/u-boot.bin ← U-boot binary executable

Flashing Linux into SOM9DIMM-LPC3250

- 1) Connect JTAG to SOM9DIMM-LPC3250's J3 port using JTAG and mini-adapter.
- 2) Plug in the other side to the PC and install J-Link driver if not already done.
- 3) Connect serial port to RS232 port (P4).
- 4) Open terminal program on PC to 115200 baud (8 bits, no parity, 1 stop bit).
- 5) Power on unit.
- 6) Press space bar any time before the screen boots Linux. If a boot occurs accidentally, just reboot and try again.
- 7) You should now be sitting at a prompt that says "uboot>"
- 8) Now open SEGGER's J-Link Commander program and enter the following commands
 - a. `h` ← this halts the processor
 - b. `speed 8000` ← This makes the J-Link communications go much faster.
 - c. `loadbin <path>\rootfs.ext2.gz, 0x81800000` ← this loads the root file system into memory.
 - d. `loadbin <path>\uImage, 0x80100000` ← This loads the Linux kernel into memory.
 - e. `g` ← This let's the LPC3250 run again
- 9) Go back to the terminal program and enter the following commands:
 - a. `nand erase 0x80000 0x200000` ← Erase Linux kernel space in NAND
 - b. `nand write.jffs2 0x80100000 0x80000 0x200000` ← write Linux kernel from memory into NAND.
 - c. `nand erase 0x280000 0x600000` ← Erase root file system in NAND
 - d. `nand write.jffs2 0x81800000 0x280000 0x600000` ← Write root file system in NAND.
- 10) You are done. Reset and the unit should boot properly now.

Flashing U-Boot into SOM9DIMM-LPC3250

- 1) Connect JTAG to SOM9DIMM-LPC3250's J3 port using JTAG and mini-adaptor.
- 2) Plug in the other side to the PC and install J-Link driver if not already done.
- 3) Connect serial port to RS232 port (P4).
- 4) Open terminal program on PC to 115200 baud (8 bits, no parity, 1 stop bit).
- 5) Power on unit. (A previous version of code may start loading, ignore).
- 6) Now open SEGGER's J-Link Commander program and enter the following commands
 - a. `h` ← this halts the processor
 - b. `speed 8000` ← This makes the J-Link communications go much faster.
 - c. `rx 200` ← This resets the processor
 - d. `loadbin <path>\u-boot.bin, 0x81FC0000` ← this loads u-boot into execution memory.
 - e. `loadbin <path>\u-boot.bin, 0x81000000` ← This loads a copy of u-boot to program.
 - f. `setpc 0x81FC0000` ← Setup to run u-boot.
 - g. `g` ← This let's the LPC3250 run again
- 7) Press space bar any time before the screen boots Linux. If a boot occurs accidentally, just reboot and try again, repeating the steps in part 6.
- 8) Go back to the terminal program and enter the following commands:
 - a. `nand erase 0x20000 0x20000` ← Erase u-boot from NAND
 - b. `nand write.jffs2 0x81000000 0x20000 0x20000` ← write u-boot copy in memory to NAND flash.

5. General Questions and Answers

Where is the source code?

LTIB downloads (if needed), patches, compiles, and then erases the source each time you build. If RPM files are available, it will use those files and skip the build process. The source and patch files are downloaded in /opt/ltib/pkg and linked in ltib/rpm/SOURCES. The sources are unpacked, compiled, and deleted in the directory ltib/rpm/BUILD.

The source code keeps disappearing, what can I do to keep it around?

Use the command `./ltib -p <package> -m prep` to download and open up the source in the ltib/rpm/BUILD directory. Build the files with the command `./ltib -p <package> -m scbuild`.

The build for u-boot keeps getting in the way when I change the kernel configuration and I want it stay built. How do I do that?

To keep u-boot from constantly rebuilding, just edit the file ltib/dist/lfs-5.1/u-boot/u-boot-common.tmpl. Change the line `rm -rf $RPM_BUILD_ROOT` under `%Clean` to be `%rm -rf $RPM_BUILD_ROOT`. This disables the cleaning process on u-boot so it is built and stays built. Note that if you need to clean u-boot and start fresh, you will need to uncomment this line first.

Where do I work on the helloworld application?

The hello world application can be unpacked and prepared with the command `./ltib -p helloworld -m prep`. The directory ltib/rpm/BUILD/helloworld-1.1/ will now appear. Modify the hello.c file and the Makefile as necessary. In the Makefile, the install command copies the final executable into the root filesystem.

Where can I get updates to this document and additional information?

Visit the site www.teamfdi.com/dk-ts-kit and/or post messages at www.teamfdi.com/forum/.
NOTE: The forum requires you first to request access by emailing to support@teamfdi.com.

Where is more information about LTIB?

LTIB is kept at <http://ltib.org/> and you should check out the FAQ at <http://ltib.org/documentation-LtibFaq>.