Freescale Semiconductor
Application Note

Document Number: AN3765
Rev. 0, 12/2008

Porting Linux for the MPC5121e

by: Dave Erazmus, Gene Fortanely, Kalle Odenthal
Austin
Infotainment Multimedia and Telematics
(IMT)

1 Introduction

The Linux BSP for MPC5121e supports the ADS5121 evaluation board. This application note provides guidance for porting the Linux BSP to customer board designs. The first section covers modifications to the u-boot universal boot loader and details the board and processor specific sections of the source code. The second section covers Linux kernel modification and highlights board-specific sources and configuration files involved in porting the kernel to a new design.

This document uses the Linux Target Image Builder (LTIB) running on a Linux-based workstation/PC as the development environment. The appendix gives an example setup procedure for preparing this environment on a Windows PC using a VMware Player to host a Debian Linux distribution on a virtual machine. This may be useful for readers not familiar with Linux software development. Any Linux host able to install and run LTIB can be used with the porting procedures outlined in this application note.

Contents

2 Porting the u-boot Boot-Loader

U-boot is an open-source boot-loader. Two methods for obtaining the u-boot source code are described below. Once the source code is obtained see the u-boot/readme file for information about configuring, building, porting, and debugging u-boot. In addition the book *Embedded Linux Primer* by Christopher Hallinan copyright 2007 is a useful source of information about u-boot and Linux.

2.1 u-boot Source Code from DENX

The most current source code with Freescale MPC5121e support may be obtained from the DENX website. This source code contains the most recent bug-fixes and feature updates.

2.1.1 Verify MPC5121e Support in the u-boot Source Code

Once the source code is obtained on the Linux host machine, verify the MPC5121 support by searching for the MPC5121 in the u-boot installation directory Makefile by entering command:

```
$ grep 5121 Makefile
```

The command output must resemble the following four lines that demonstrate the ADS5121e development board is supported by this u-boot source code:

```
ads5121_config \
ads5121_256_config      \
echo "#define CONFIG_ADS5121_256 1" > $(obj)include/config.h; \
@$(MKCONFIG) -a ads5121 ppc mpc512x ads5121
```

2.1.2 Configure Build Environment

The build environment must be configured appropriately. Add two environment variables and modify the path at the command line or by inserting them into the command-line shell's configuration script.

Add the CROSS_COMPILE environment variable:

```
$ export CROSS_COMPILE=powerpc-e300c3-linux-gnu-
```

Add ARCH environment variable:

```
$ export ARCH=ppc
```

Append path to e300 Power Architecture gcc compiler to search path:

```
$ export
PATH=$PATH:/opt/freescale/usr/local/gcc-4.1.78-eglibc-2.5.78-1/powerpc-e300c3-linux-gnu/bin
```

2.1.3 Build u-boot Image

From the u-boot source code installation directory enter the following commands to clean derived files, configure the source code, and build u-boot:

```
# remove derived files
make clean
# config u-boot to build an ADS5121e development board u-boot image
make ads5121_config
# make an ADS5121e development board u-boot image
make
```

Building produces these files in the u-boot installation directory:

u-boot, u-boot.bin, u-boot.map, and u-boot.srec

2.2 u-boot Source Code from Freescale

The source code with Freescale MPC5121e support can be obtained from www.freescale.com as part of the Linux BSP for Freescale MPC5121eADS distribution. U-boot versions 1.2.0 and 1.3.2 are included as part of the ltib-mpc5121ads-20080528 distribution. See the *Linux BSP for Freescale MPC5121eADS User's Guide* for information about installing, configuring, and building the distribution on the Linux host machine.

2.2.1 Verify MPC5121e Support in Linux BSP

To verify which versions of u-boot are available in the users Linux BSP distribution search for u-boot 5121 specification files by entering this command at the LTIB installation directory:

```
$ find . -name u-boot*5121*.spec
```

The command output must resemble the following two lines that show the u-boot version 1.2.0 and 1.3.2 are available in this Linux BSP:

```
./dist/lfs-5.1/u-boot/u-boot-1.2.0-mpc5121ads.spec
./dist/lfs-5.1/u-boot/u-boot-1.3.2-mpc5121.spec
```

2.2.2 Extracting u-boot Source Code from Linux BSP

u-boot version 1.3.2 source code can be extracted from the installed Linux BSP distribution by entering the following command in the LTIB installation directory:

```
$ ./ltib -p u-boot-1.3.2-mpc5121ads.spec -m prep
```

The LTIB's installation folder is referred to as <ltib>.

u-boot version 1.3.2 source code from the installed Linux BSP distribution is located at:

```
<ltib>/rpm/BUILD/u-boot-1.3.2
```

2.2.3 Build u-boot Image

u-boot version 1.3.2 source code from the installed Linux BSP distribution can be built by configuring the build environment as described in Section 2.1, "U-Boot Source Code from Denx" or by entering the command below in the LTIB installation directory:

```
$ ./ltib -p u-boot-1.3.2-mpc5121.spec -m scbuild
```

Building produces the files below in the u-boot installation directory:

- u-boot
- u-boot.bin
- u-boot.map
- u-boot.srec

2.3 File Organization

After installation, the u-boot directory contains the following sub-directories:

u-boot/api
u-boot/api_examples
u-boot/board
u-boot/common
u-boot/cpu
u-boot/disk
u-boot/doc
u-boot/drivers
u-boot/examples
u-boot/fs
u-boot/include
u-boot/lib_arm
u-boot/lib_avr32
u-boot/lib_blackfin
u-boot/libfdt
u-boot/lib_generic
u-boot/lib_i386
u-boot/lib_m68k
u-boot/lib_microblaze
u-boot/lib_mips
u-boot/lib_nios
u-boot/lib_nios2
u-boot/lib_ppc
u-boot/lib_sh
u-boot/nand_spl
u-boot/net
u-boot/onenand_ipl
u-boot/post
u-boot/tools

2.4 Porting u-boot to a New Freescale MPC5121e-Based Hardware System Board

For porting instructions see the u-boot/readme file. Below is a summary of the procedure for porting the u-boot source code that supports the Freescale MPC5121e processor and the ADS5121e development board.

2.4.1 Modifying u-boot/Makefile and u-boot/MAKEALL Scripts

Edit Makefile and MAKEALL by searching for existing ADS5121e development board support. Modify these portions of the Makefile and MAKEALL scripts to match needs. Here is the portion of the Makefile where support for a new_MPC5121e_board must be inserted as shown.

```
#########################################################################
## MPC512x Systems
#########################################################################
ads5121_config \
ads5121_256_config      \
        : unconfig
        @mkdir -p $(obj)include
        @if [ "$(findstring 256,$@)" ] ; then \
                echo "#define CONFIG_ADS5121_256 1" > $(obj)include/config.h; \
        fi
        @$(MKCONFIG) -a ads5121 ppc mpc512x ads5121

new_MPC5121e_board_config \
        : unconfig
        @mkdir -p $(obj)include
        @$(MKCONFIG) -a new_MPC5121e_board ppc mpc512x new_MPC5121e_board
```

Here is the portion of the MAKEALL where support for a new_MPC5121e_board must be inserted:

```
#########################################################################
## MPC512x Systems
#########################################################################

LIST_512x="             \
 ads5121         \
 new_MPC5121e_board \
"
```

2.4.2 Board Directory

An individual directory for each board supported by u-boot is found in the u-boot/board directory. The directory u-boot/board/ads5121/ contains support for the ADS5121e development board. This sub-directory includes board-specific configuration routines in C language source code files such as flash initialization, DRAM configurations, and external device configurations as ATA, PCI, FPGAs, and so on.

The contents of this directory must be copied to a new directory to form the basis of support for a new Freescale MPC5121e-based hardware system board. These files are to be modified to work with a new board. Detailed knowledge of the new board and the board components are needed to successfully port this code.

Enter these commands to create an initial code base in the directory u-boot/board/new_MPC5121e_board:

- Go to the u-boot/board directory:

```
$ cd u-boot/board
```

- Create a directory to contain the files to support the new board:

```
$ mkdir new_MPC5121e_board
```

- Populate the new directory with the files from the existing ADS5121e development board. Modify these files to support the new_MPC5121e_board.

```
$ cp -r ./ads5121/* ./new_MPC5121e_board
```

2.4.3 Include Directory

The u-boot/include directory contains all header files used by the u-boot. The u-boot/include/configs/ads5121.h file contains board-specific information for the ADS5121e development board. This file must be copied to a new file name and modified to support the new_MPC5121e_board. Enter these commands to create an initial board-specific. Include file for the new_MPC5121e_board:

- Go to the u-boot/include/configs directory:

```
$ cd u-boot/include/configs
```

- Copy the existing ADS5121e development board file to create the new file. Modify this file to support the new_MPC5121e_board.

```
$ cp ads5121.h new_MPC5121e_board.h
```

3 Porting the 2.6.24.6 Linux Kernel to a Customized MPC5121e Board Using LTIB

This chapter shows how to add a new MPC5121e platform to the kernel source tree and deploy it to the LTIB. There is more than one way to add a new platform to the kernel. There are some standarized procedures especially for PowerPC platforms that might facilitate the work. It is recommended to follow those standarized procedures guidelines to maintain clarity and compatibility in the kernel.

To trace the kernel files and functions mentioned in this chapter go to the Linux LXR web page.

NOTE

The kernel /ppc and /powerpc are not the same. The mechanisms specified in the ppc folder are obsolete and used only by an older PowerPC platform source code. Continue using /powerpc.

3.1 PowerPC Software Architecture in the Linux Kernel

To understand the initialization process of the PowerPC kernel two mechanisms have to be discussed, the flat device tree source (.dts file) and the machine description structure (ppc_md structure).

3.1.1 Device Tree Source (.dts) File

The device tree source files are located in kernel folder ./arch/powerpc/boot/dts. Those files describe the device tree of a platform. Each device is represented by a node. Each node can have sub nodes and a

defined set of properties. This information is used by the PowerPC platform specific source code to initialize and drive the hardware (HW) correctly. The .dtb file is the compiled version of the .dts file can also be modified by the bootloader.

Most entries in the .dts are MPC5121e generic and do not have to be changed when porting the kernel to a new platform.

It is important to be familiarized with the platform well enough to change the nodes. Any driver or kernel function may access this file by means of a small API specified in ./include/linux/of.h and ./drivers/of/base.c. Through this API the kernel and the device drivers can search device nodes and read their properties.

Refer to the document ./Documentation/powerpc/booting-without-of.txt in the kernel source tree for detailed information.

3.1.2 PowerPC Machine Description (ppc_md) Structure

The variable ppc_md is a global variable of the type structure machdep_calls (machine description for a platform function calls). This structure is defined in ./include/asm-powerpc/machdep.h. This variable can be seen as an interface to the platform specific code. It consists of a set of pointers to functions that have to be defined by the platform specific code like setup_arch, init, init_IRQ, probe and so on.

By means of a kernel macro the structures entries are set to the platform specific functions at compilation time. The kernel has to know ppc_md to run those functions.

This architecture makes the communication between kernel and platform specific code easy. Figure 1 illustrates how the ppc_md structure works.

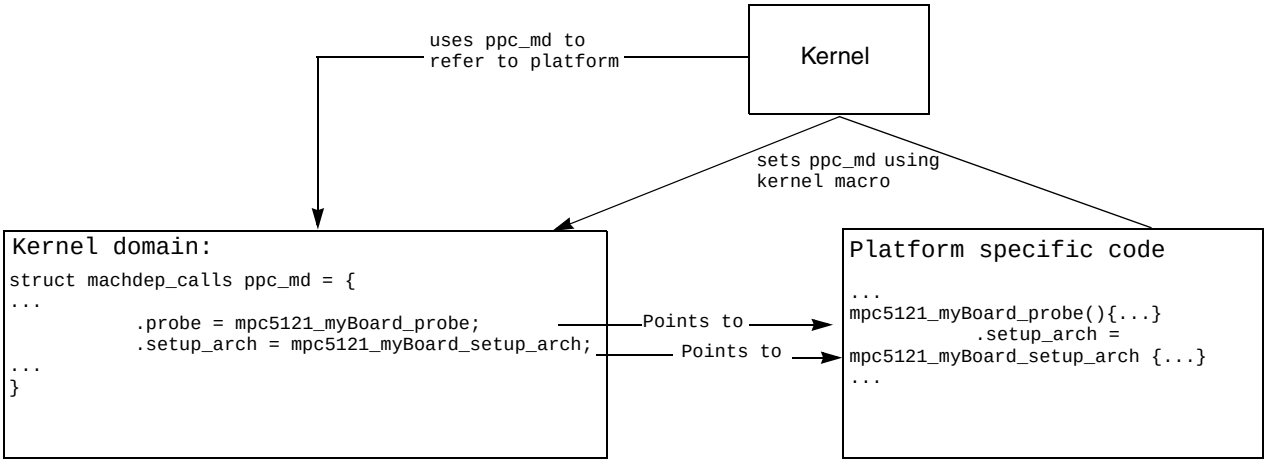


Figure 1. The ppc_md Structure

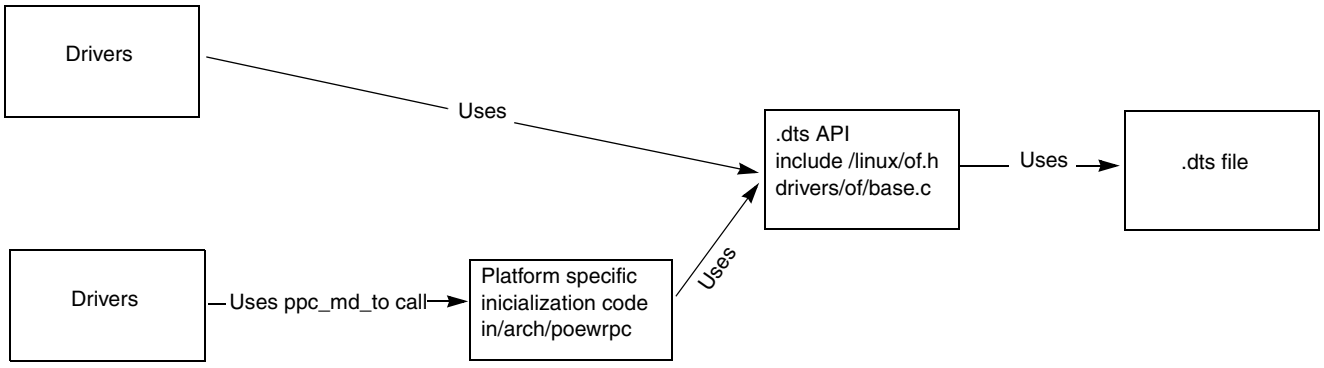


Figure 2. Interaction from Kernel and Drivers with Platform Specific Code and Settings

Figure 1 illustrates the interaction between the kernel, the drivers, and the platform specific code and settings. The kernel uses the global variable ppc_md to call platform specific functions.

Figure 2 shows the function of the .dts file and how the kernel accesses the device node information.

3.2 PowerPC Initialization Sequence in the Linux Kernel

The bootloader responsibility is to put the uncompressed and stripped Linux kernel image to a well defined place in the memory. When the bootloader finishes, the instruction pointer is set to the beginning of the Linux kernel code. If the kernel is configured to the PPC32, the kernels starting point is defined in the assembler file ./arch/powerpc/kernel/head_32.S in line 1011. It says start_here. After a few lines of assembler code a statement is found that says:

```
bl      machine_init
```

This statement calls the machine_init function defined in ./arch/powerpc/kernel/setup_32.c. In this function early initialization for debugging are done followed by initialization based on the device tree specified in the .dts file.

The machine_init calls the function probe_machine defined in ./arch/powerpc/kernel/setup-common.c. Here the ppc_md structure is initialized with the platform specific function calls. At the end of machine_init the function returns to head_32.S.

In the function call MMU_init and with the instructions that follows, the memory management unit (MMU) sets up. Later on the the function start_kernel is called. The start_kernel is defined in ./init/main.c. In this function the function call setup_arch and command_line is found and defined in the /arch/powerpc/kernel. In this function various functions of the ppc_md structure are called. Look through the setup_32.c file to find where additional ppc_md functions are called and in what context. This gives an idea of what function must be implemented in the platform specific code.

Figure 3 illustrates the early platform initialization process.

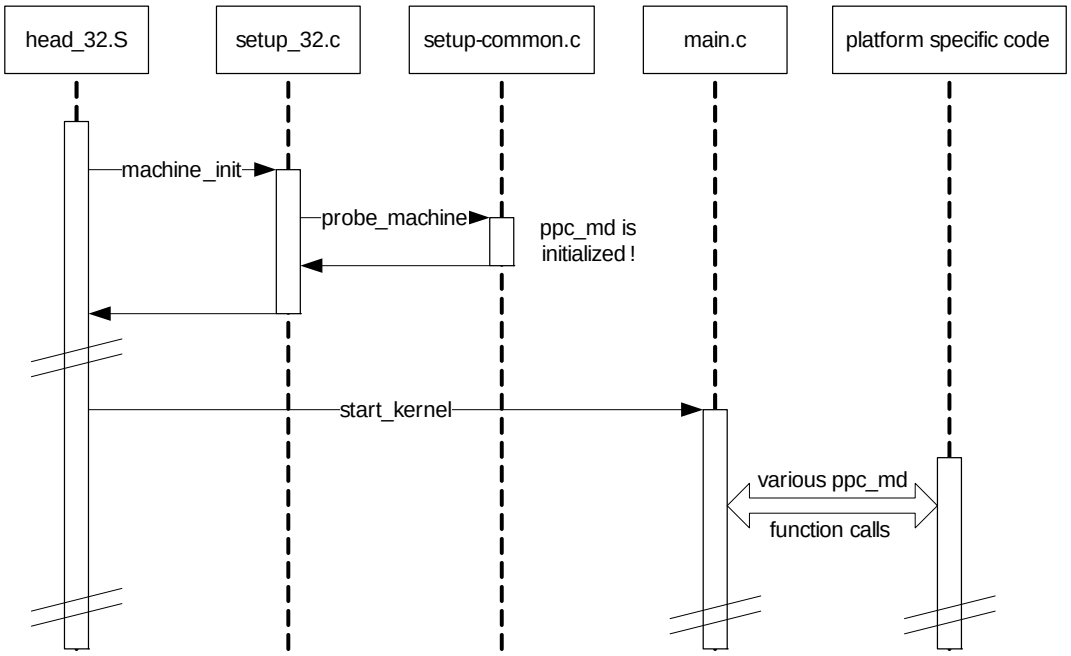


Figure 3. Early initialization Process for PowerPc Platforms

3.3 Create a New MPC5121e Platform in the Linux Source Tree

This section describes how to add a new MPC5121e platform to the Linux source tree and how to build and deploy it using LTIB. You can also cross compile the kernel without LTIB. Refer to Embedded Linux Primer to read how to do this.

The LTIB installation folder is referred to as <ltib>.

Unpack the kernel source to the <ltib>/rpm/BUILD directory and apply recent MPC5121ads kernel patches by running the LTIB command:

<ltib>$ ./ltib –m prep –p kernel

Change the directory to <ltib>/rpm/BUILD/linux-2.6.24.

Paths in this section are relative to <ltib>/rpm/BUILD/linux-2.6.24

Two new files must be created. The new .dts file for the platform and a source code file that implements the functions of the ppc_md structure. Usually the platform specific code may be divided in more than one file to maintain clarity and reusability of the source.

3.3.1 Create a New .dts file

To create the new .dts file, go to folder ./arch/powerpc/boot/dts. It is recommendable to start from existing sources preferably from platforms that are similar to the new platform. In this case the ADS5121 board may be the most similar. Copy the existing mpc5121ads.dts file and give it a new name (for example

my5121board.dts). Edit this new file according to the specifications found in the file booting-without-of.txt. Change the model property to the name of the new board and add entries to the compatible properties.

NOTE

Remember the kernel and the drivers search for compatible nodes that are in the .dts file. If wanting to reuse the Freescale NAND driver (./drivers/mtd/nand/fslnand.c) the compatibility statement can remain in the .dts file. The statement np = of_find_compatible_node (NULL, NULL, "fsl,mpc5121ads-cpld"); in the driver finds the device node.

3.3.2 Create Platform Specific Code

The adequate place for the platform specific code in the kernel source tree for any MPC512x platform is ./arch/powerpc/platforms/512x. Go to this path and take a look at the containing files (disregard the hidden files):

- Makefile
- Kconfig
- mpc5121_usb.c
- mpc512x_pm.h
- clock.c
- mpc5121_pscgpio.c
- mpc512x.h
- mpc5121_spi.c
- pci.c
- mpc512x_pm.c
- mpc512x_pm_test.c
- mpc5121_ads.c
- mpc5121ads_cpld.c

There are two ADS5121 specific files, mpc5121_ads.c and mpc5121ads_cpld.c. This is due to the fact that a lot of the ADS5121 initialization code is MPC5121e generic. The person who wrote those files was aware of the fact that new platforms besides the ADS5121 might be added to the kernel source tree. The not ADS5121 specific files are meant for reuse in new MPC512x platforms.

To get started the mpc5121_ads.c is copied to the mpc5121_myBoard.c .

Edit the file mpc5121_myBoard.c.

At the bottom of the file this statement is found:

```
define_machine(mpc5121_ads) {
        .name                   = "MPC5121 ADS",
        .probe                  = mpc5121_ads_probe,
        .setup_arch             = mpc5121_ads_setup_arch,
        .init                   = mpc5121_ads_declare_of_platform_devices,
        .init_IRQ               = mpc5121_ads_init_IRQ,
        .get_irq                = ipic_get_irq,
        .calibrate_decr         = generic_calibrate_decr,
};
```

What looks like a function is actually a macro defined in ./include/asm-powerpc/machdep.h. This macro provides a machdep_call structure to the kernel deriving the ppc_md variable, and is the hook to the kernel. Here the platform specific functions are defined for the kernel to call.

Read the ./include/asm-powerpc/machdep.h file to see all of the possible functions declared in the machdep_calls structure. Change mpc5121_myBoard.c according to the users platform.

NOTE

When using a CPLD, copy and change the file mpc5121ads_cpld.c. Remember that some ads specific CPLD function declarations are placed in the generic mpc512x.h header.

When finished with the platform specific changes add the new platform to the kernel configuration menu by editing the Kconfig (Kernel Configuration) file in the current ./512x folder. Add the new platform as an option. The CPLD and other configurations can also be added to the Kconfig file.

Entries may look like this:

```
config MPC5121_MYBOARD_CPLD
        bool
        depends on MPC5121_MYBOARD

config MPC5121_MYBOARD
        bool Freescale MPC5121E MYBOARD
        depends on PPC_MULTIPLATFORM && PPC32
        select DEFAULT_UIMAGE
        select WANT_DEVICE_TREE
        select PPC_MPC5121
        select MPC5121_MYBOARD_CPLD
        help
          This option enables support for the MPC5121E MYBOARD board.
        default n
```

NOTE

The following configurations must be set to enable the PowerPC platform mechanisms

- depends on PPC_MULTIPLATFORM && PPC32
- select DEFAULT_UIMAGE
- select WANT_DEVICE_TREE
- select PPC_MPC5121
- select MPC5121_MYBOARD_CPLD

Next, change the Makefile in the current folder ./512x. Open the Makefile and add the new targets to it. The Makefile must resemble the following:

```
#
# Makefile for the Freescale PowerPC 512x linux kernel.
#
obj-y                                := clock.o mpc5121_usb.o mpc5121_pscgpio.o
obj-$(CONFIG_PCI)                            += pci.o
obj-$(CONFIG_SPI)                            += mpc5121_spi.o
obj-$(CONFIG_MPC5121_ADS)                    += mpc5121_ads.o
obj-$(CONFIG_MPC5121_ADS_CPLD)               += mpc5121ads_cpld.o
obj-$(CONFIG_MPC5121_MYBOARD)                += mpc5121_myBoard.o
obj-$(CONFIG_MPC5121_MYBOARD_CPLD)           += mpc5121myBoard_cpld.o
obj-$(CONFIG_PM)                             += mpc512x_pm.o
obj-$(CONFIG_MPC5121_PM_TEST)                += mpc512x_pm_test.o
```

The kernel changes are now finished. Go on to deploy the new platform to LTIB and build it.

3.4 Deploy New Platform to LTIB

To deploy the new platform into LTIB a new platform configuration folder is needed. Those folders are situated in ./config/platform/. Copy or move the existing mpc5121ads folder to a folder named mpc5121myBoard. Change to the new folder and look at its content.

The file main.lkc contains the LTIB menu for this platform. The linux-2.6.xxx.config files are default configuration files for the kernel.

First create a new linux-2.6.24-mpc5121myBoard.config file to provide a new platform by coping the existing linux-2.6.24-mpc5121ads.config file. Edit this file and change the section where the platform is specified and save the file. If you do not need CPLD support remove the CONFIG_MPC5121_MYBOARD_CPLD line.

```
…
#CONFIG_MPC5121_ADS_CPLD=y
#CONFIG_MPC5121_ADS=y
CONFIG_MPC5121_MYBOARD_CPLD=y
CONFIG_MPC5121_MYBOARD=y
…
```

NOTE

Indicate in this file the preselected packages. When creating a new platform it is better to start with a minimal kernel and assembling it step by step. This helps to find failures in the specific platform code.

3.4.1 main.lkc

Open the main.lkc file, edit CONFIG_TITLE, PLATFORM_COMMENT, VENDOR, and PLATFORM at the users own discretion.

Do not change GNUTARCH, LINTARCH, and CFGHOST nor the section where the toolchain is specified. The toolchain for any MPC5121e platform can stay the same.

Scroll down to SYSCFG_DTC_NAME and add the name of the .dts file:

```
…
config SYSCFG_DTC_NAME
    string
    default mpc5121myBoard
…
```

Scroll down to choose a kernel. Add a new kernel:

```
config KERNEL3
        bool Linux 2.6.24.6 for MPC5121MYBOARD
        help
          This is an example kernel that demonstrates how to port the kernel to a new platform.
```

Set the default kernel to the user's kernel:

```
default KERNEL3
```

Link the new linux-2.6.24-mpc5121myBoard.config file to the kernel in the config PKG_KERNEL_PRECONFIG section:

```
…
default linux-2.6.24-mpc5121myBoard.config if KERNEL3
…
```

Save and close the main.lkc file.

3.4.2 default.lkc

Switch to directory ./config/userspace/ and open the file default.lkc. Search the line config E300C3_ARCH.

Add || PLATFORM = mpc5121myBoard to E300C3_ARCH to this section to define the processor architecture for the platform. The name after PLATFORM = must be the platform name assigned in main.lkc.

3.4.3 <ltib>/.config

The new platform is now ready for LTIB. If the mpc5121ads configuration folder was moved to the new mpc5121myBoard configuration folder, the next time the LTIB is run it detects the new configuration folder. It will not find the old one anymore. If the folder was copied, indicate what folder LTIB must use by modifying the platform directory configuration in the file .config. Switch to the <ltib> folder, open the file .config, and change the platform directory configuration.

If the ./ltib –c runs the configuration title assigned in the main.lkc, it will appear in the LTIB configuration window.

3.4.4 Setup LTIB

The selected configurations in the LTIB configuration menu must align to the linux-2.6.24-mpc5121myBoard.config file. Double check all the settings are fine, above all check whether the kernel and its preconfig file are correctly detected.

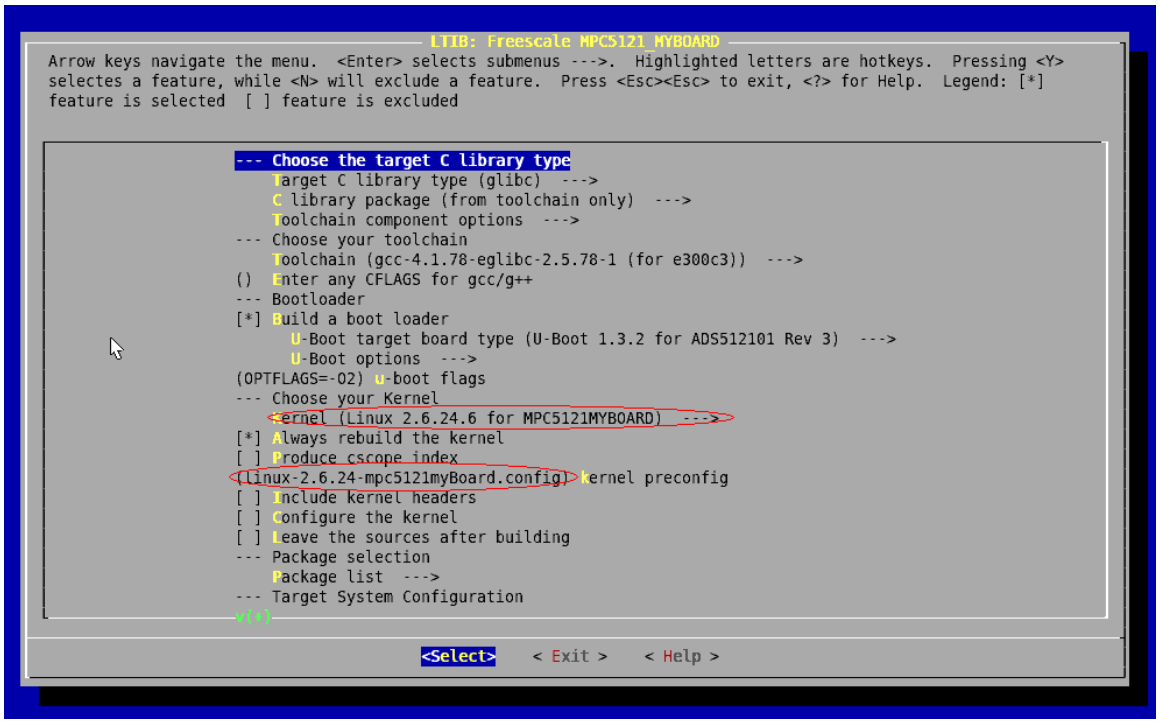


Figure 4. LTIB Configuration Menu for mpc5121myBoard Platform

Select rebuild the kernel to build the kernel and configure the kernel option to see the kernel configuration.

3.4.5 Build Kernel Image

To build and deploy the kernel save and exit the LTIB menu.

NOTE

Since the platform is a new platform, LTIB decompresses and rebuilds all the selected packages. It is not likely all packages compile at the first try. This may be caused by the host machine setup. Some packages need programs installed in the host computer, certain versions of bash and so on. There is no generic way to overcome those failures. Read the LTIB's error messages carefully and make changes as necessary. In Appendix A common errors and their solutions are found.

Appendix A Package Fixes

A.1 hotplug

The hotplugs Makefile uses a syntax that uses round brackets {}. This syntax is not supported by all shells. If LTIB fails compiling a package it automatically leaves the source code in the rpm/BUILD directory.

Change the listed section of the Makefile in rmp/BUILD/hotplug-2004_03_29/:

```
for F in etc/hotplug/{*.{agent,rc},hotplug.functions} ; do \
        ${INSTALL_SCRIPT} $$F $(etcdir)/hotplug ; \
            done
for F in etc/hotplug/{blacklist,usb.{user,hand,dist}map} ; do \
            ${INSTALL_DATA} $$F $(etcdir)/hotplug ; \
            done
```

The section of the Makefile in rmp/BUILD/hotplug-2004_03_29/ is replaced:

```
for F in etc/hotplug/*.agent ; do \
            ${INSTALL_SCRIPT} $$F $(etcdir)/hotplug ; \
            done
for F in etc/hotplug/*.rc; do \
            ${INSTALL_SCRIPT} $$F $(etcdir)/hotplug ; \
            done
for F in etc/hotplug/hotplug.functions ; do \
            ${INSTALL_SCRIPT} $$F $(etcdir)/hotplug ; \
            done
for F in etc/hotplug/blacklist; do \
        #    ${INSTALL_DATA} $$F $(etcdir)/hotplug ; \
        #    done
for F in etc/hotplug/usb.usermap} ; do \
        #    ${INSTALL_DATA} $$F $(etcdir)/hotplug ; \
        #    done
for F in etc/hotplug/usb.handmap} ; do \
        #    ${INSTALL_DATA} $$F $(etcdir)/hotplug ; \
        #    done
for F in etc/hotplug/usb.distmap ; do \
        #    ${INSTALL_DATA} $$F $(etcdir)/hotplug ; \
        #    done
```

A.2 glib2

The glib2 requires the gettext package installed on the host. Install that package.

Appendix B Linux Development Environment Setup Example

This section presents one possible development environment for the Linux MPC5121e development. For those not set up for Linux cross-development, this is a way to get started.

The development environment has three parts:

- The virtual machine
- VMware Player
- MPC5121e Linux BSP

This example setup was originally created on a notebook PC running Microsoft Windows XP Professional Edition. The entire installation procedure may take a few hours to complete.

B.1 Creating a Virtual Machine

For this example use the VMware Player. This a free desktop virtualization application that allows to run a desktop Linux distribution in a virtual machine under Windows. To create the virtual machine used by the VMware Player visit the EasyVMX website.

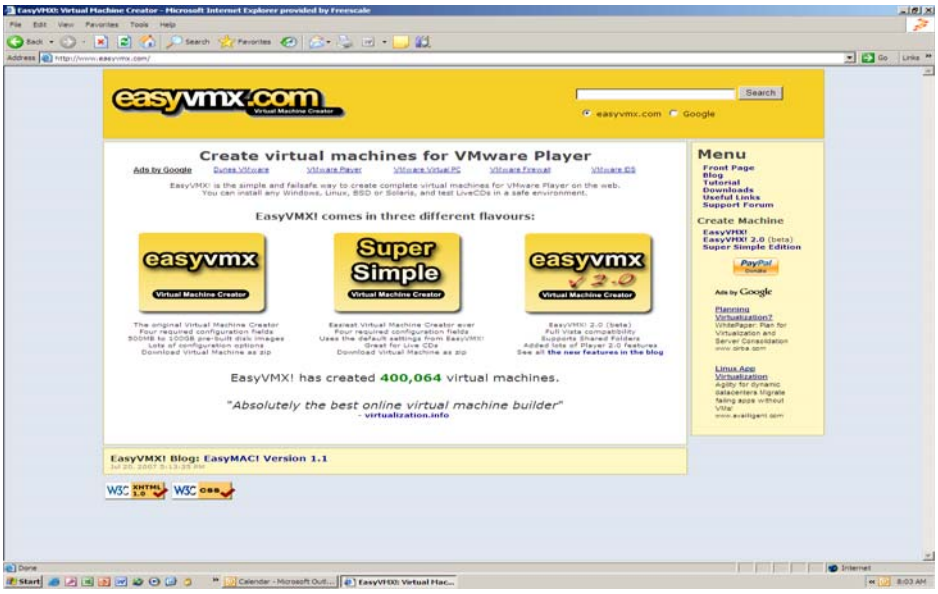


Figure B-1. Web Page

Click on the button  left of the Window in Figure B-1 and a new webpage Basic Configuration appears on the screen.

B.2 Virtual Hardware

Under the option Virtual Hardware enter a name of choice.

CAUTION

For those who are using a different host operating system be sure to make the appropriate choice on the Select GuestOS menu as shown in Figure B-2.

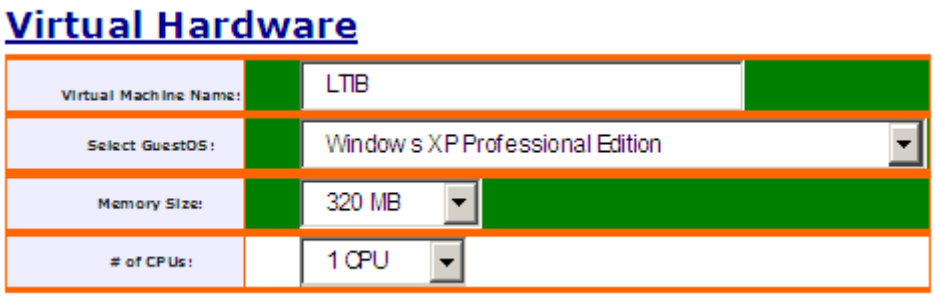


Figure B-2. Virtual Hardware

B.3 Hard Disk Drives

Scroll down the page to the Hard Disk Drive options. Select Disk Size 10 GB from the menu Disk #1. To allow the virtual machine to use up to 10 GB of hard disk space, see Figure B-3, .

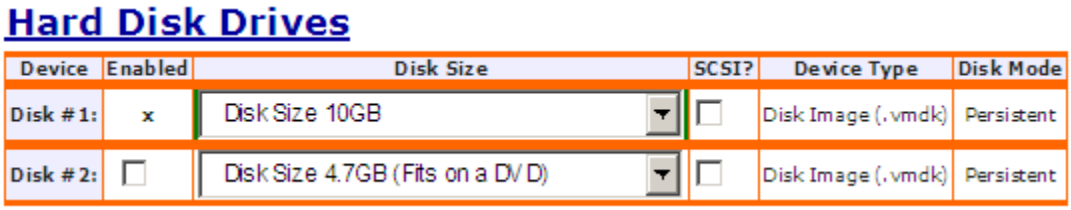


Figure B-3. Hard Disk Drives

B.4 Network Configuration

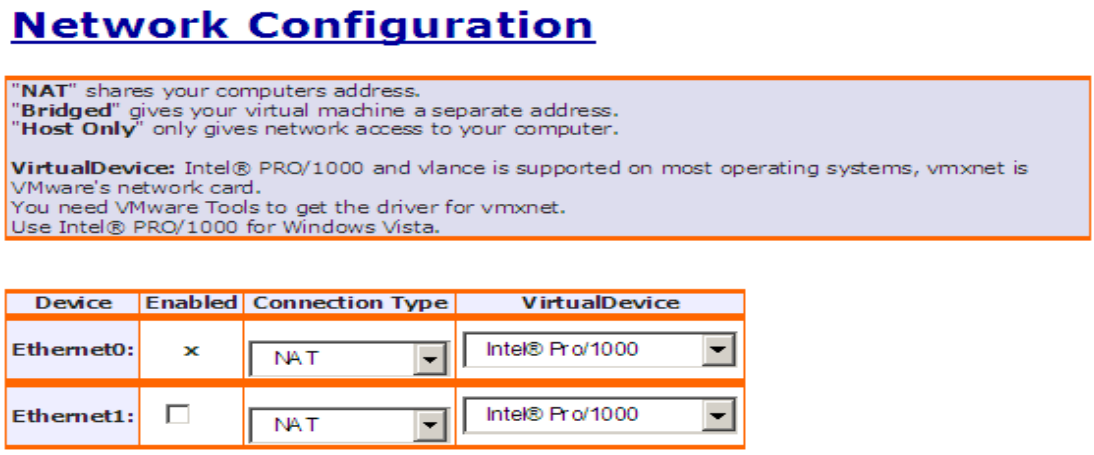


Figure B-4. Network Configuration

Use the default network setting.

B.5 Sound and I/O-Ports Configuration

See Figure B-5 Sound and I/O-Ports Configuration. To enable USB auto connects select Enable USB Autoconnect option in the USB menu as shown in Figure B-4.

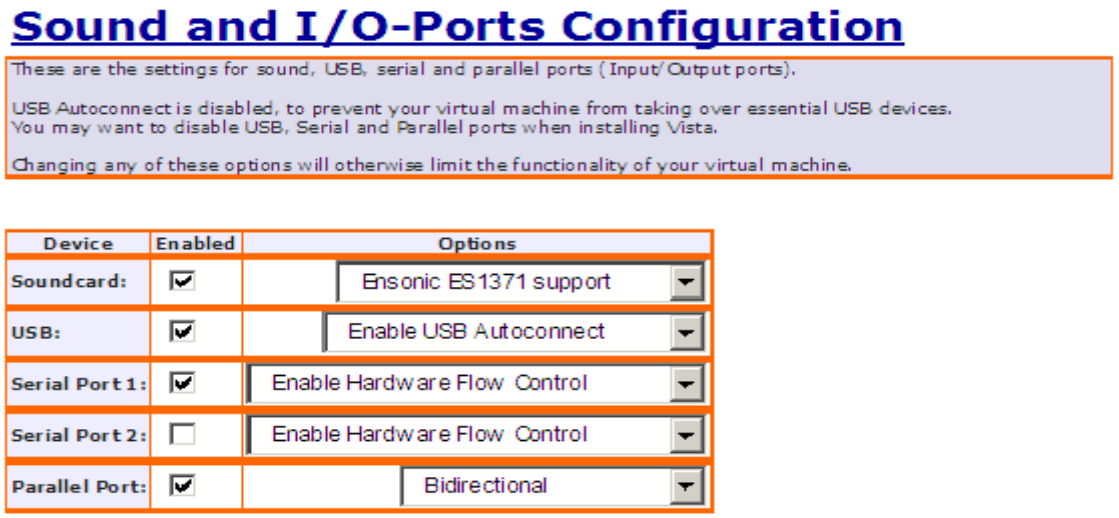
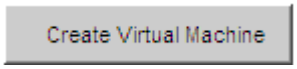


Figure B-5. Sound and I/O-Ports Configuration

Click on the button  and a new webpage appears as shown in Figure B-5.

B.6 Download Virtual Machine

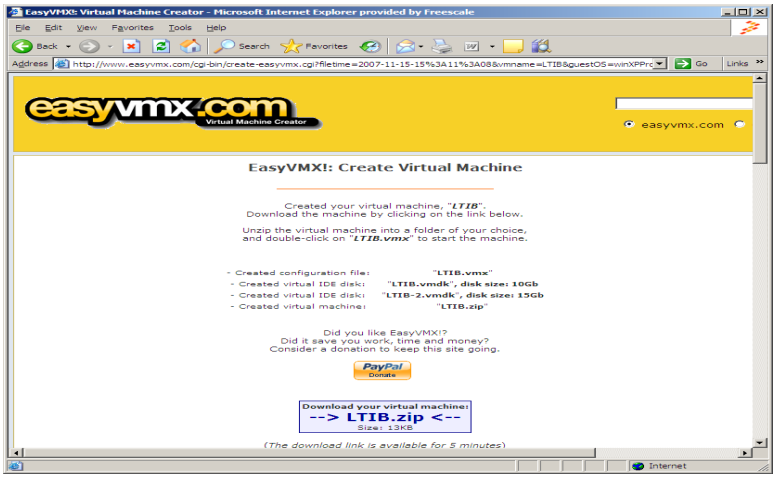


Figure B-6. EasyVMX Web Page

Finally, click on the button  to download the virtual machine. Save it on the hard disk and unzip it.

The virtual machine is now successfully created.

B.7 Install VMware Player

Download the VMware Player at the following VMware website.

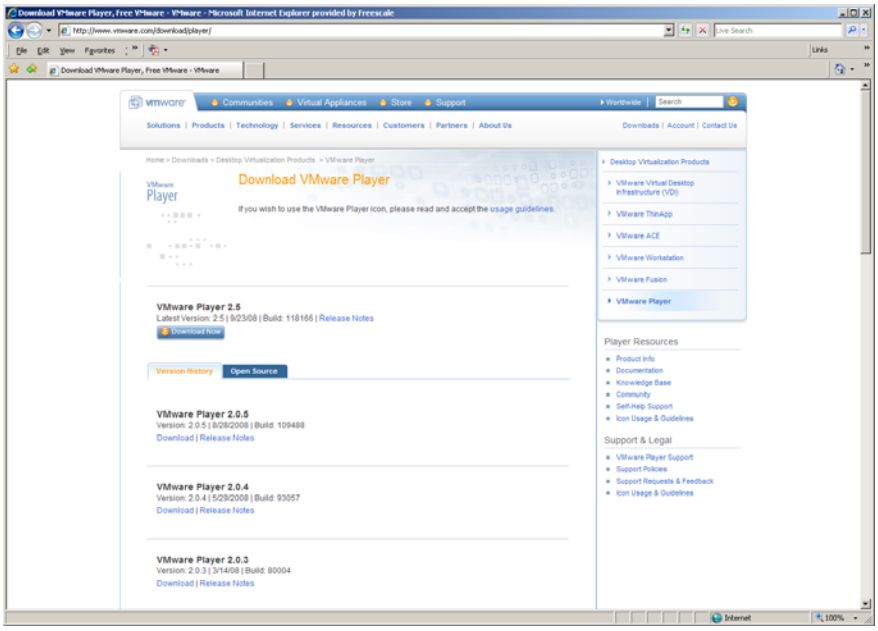


Figure B-7. VMware Website

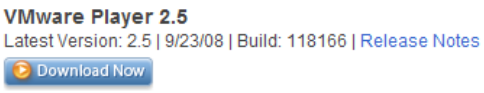


Figure B-8. VMware Installer

Click the Download Now button. Complete the free registration and save the installer to the hard disk drive. Run the installer and when it completes, reboot the computer.

B.8 Install Debian Linux in the Virtual Machine

There are several choices for a Linux distribution. For the purpose of this example the Debian distribution is used. Make sure to have a Debian net installation CD. If the user does not have one, visit the Debian website to download a net installation ISO image and burn it on a CD. Version 4.0 or later will suffice.

Start the VMware Player. A window appears as shown in Figure B-26. Click on the Open icon and select the folder where the virtual machine files from EasyVMX are placed.

Figure B-9. VMware Player

B.9 Open Virtual Machine

Select the .vmx file created at EasyVMX. Insert the Debian net installation CD into the CDROM and click the Open button on the Open Virtual Machine window.

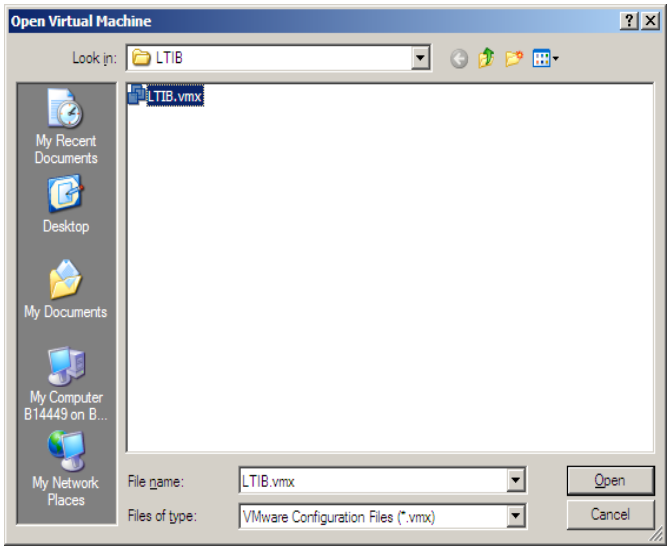


Figure B-10. Open Virtual Machine Window

The virtual machine boots and the Debian net installation begins.

Figure B-11. LTIB VMware

Linux installation begins with a startup screen as show in Figure B-29. Click the mouse inside the VMware window to direct all mouse and keyboard input to the VMWare while the installation programs are running. Press the Enter key on the keyboard and the Linux installation continues.

B.10 Installing Linux

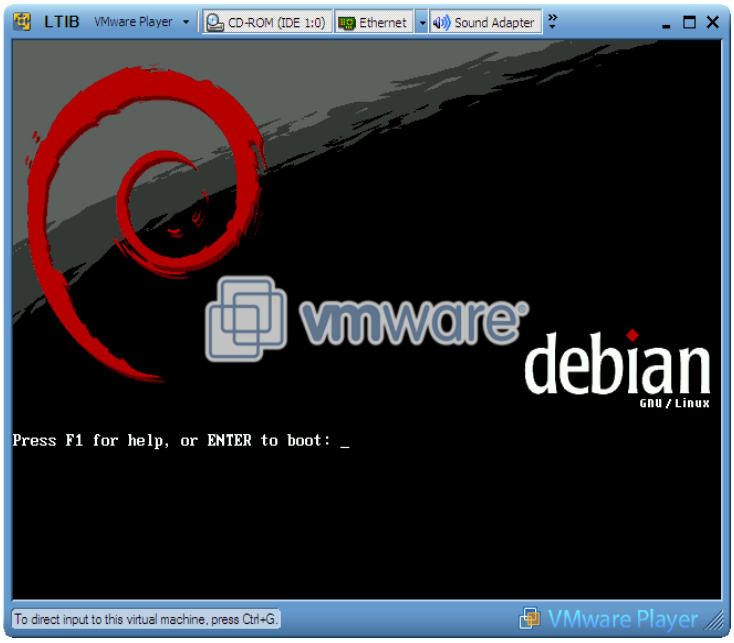


Figure B-12. Debian Web Page

The following three windows appear in the order shown below to gather localization information. Choose the appropriate field using the up or down arrow key on your keyboard. Then, press the Enter key to go to the next window.

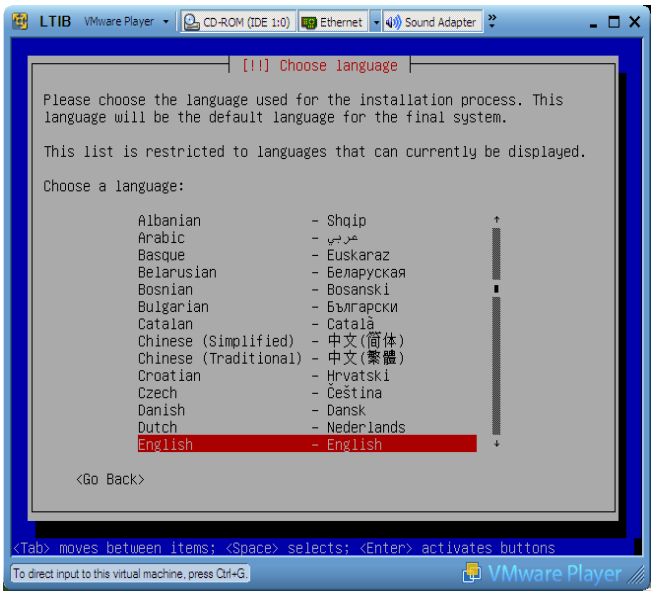


Figure A

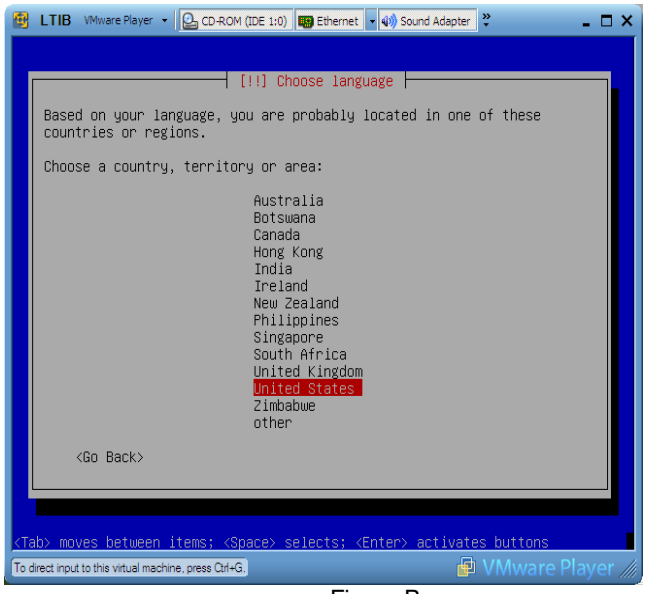


Figure B

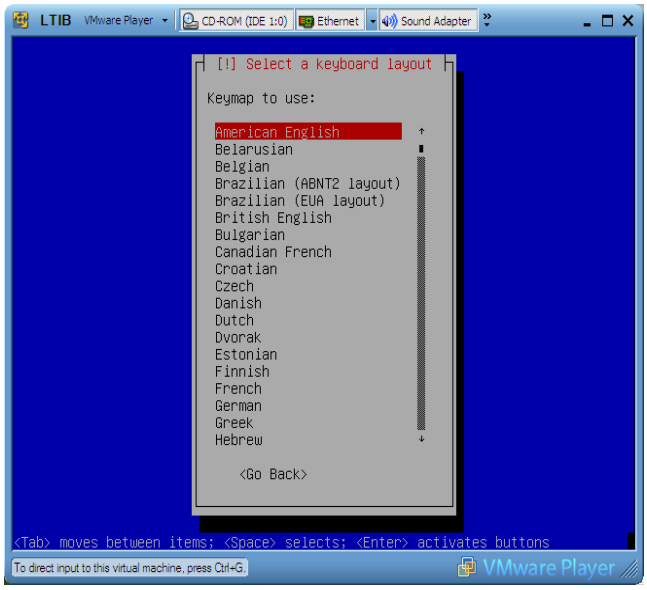


Figure C

Figure B-13. Localization Options

The installation wizard scans the CD and loads additional components as shown in Figures D and E.

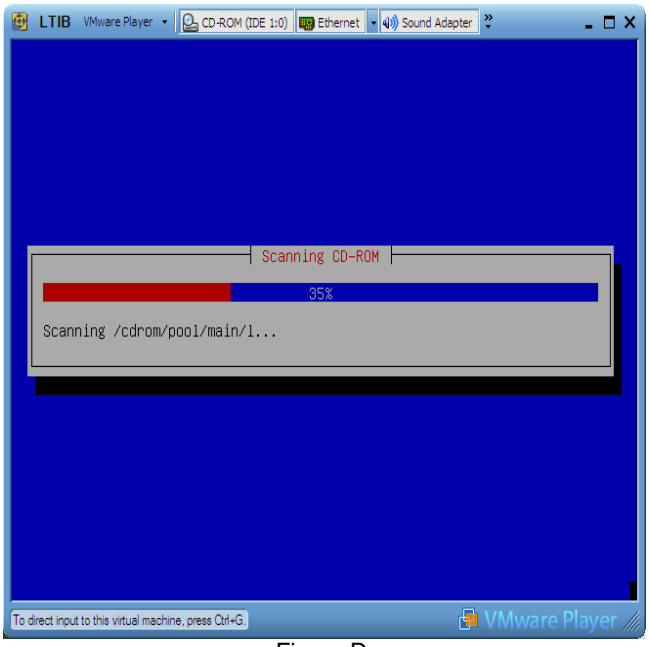


Figure D

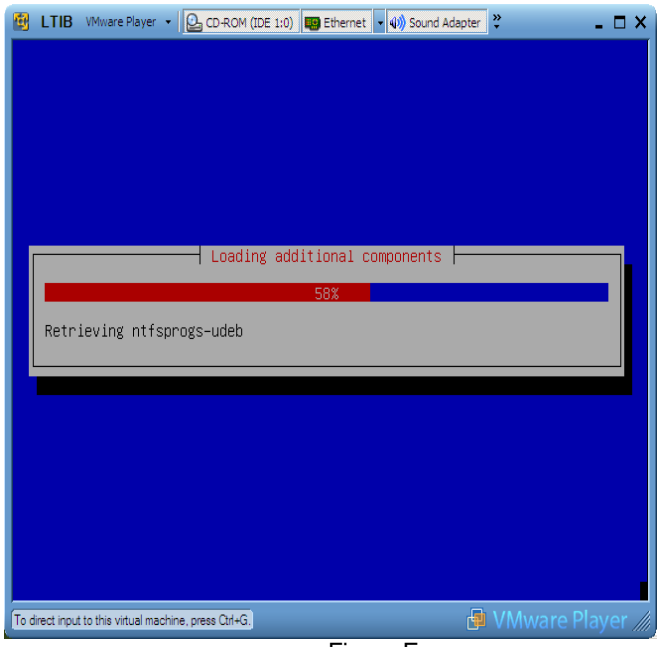


Figure E

Figure B-14. Wizard Scan

Next is the network configuration stage. Enter in each window a hostname and local domain for this system.

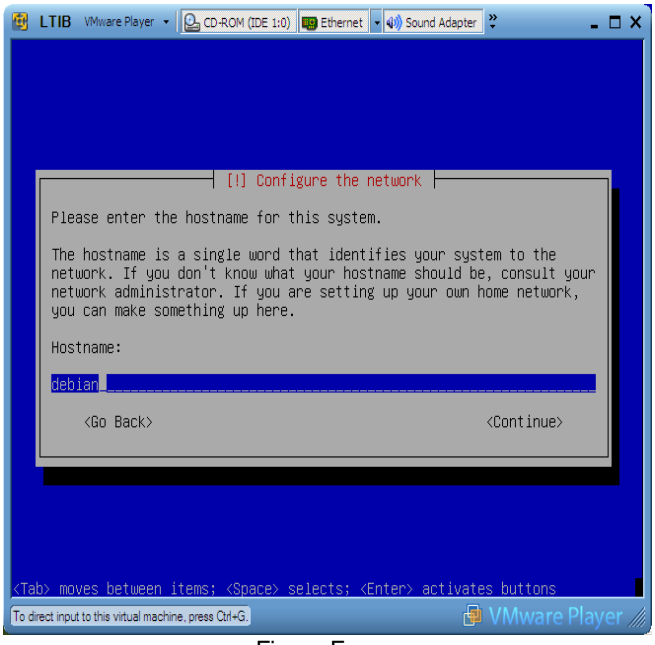


Figure F

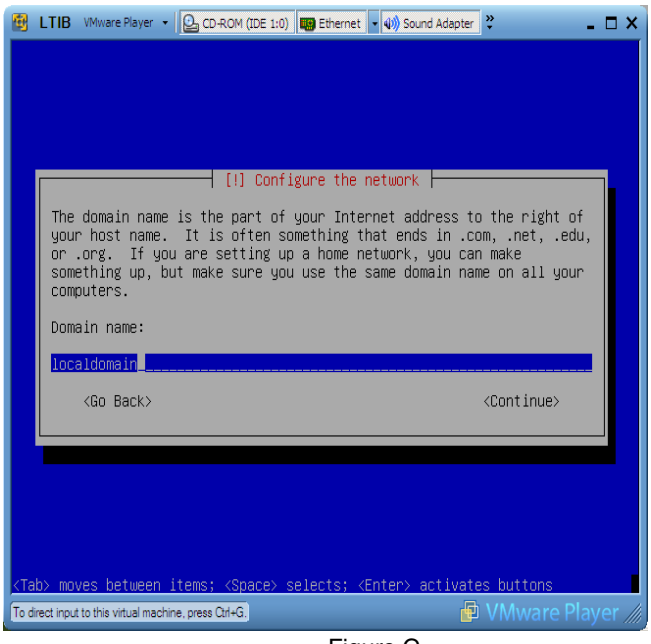


Figure G

Figure B-15. Networking Configuration Stage

The disk partition stage is next. Select the options as shown in Figures H through I below.

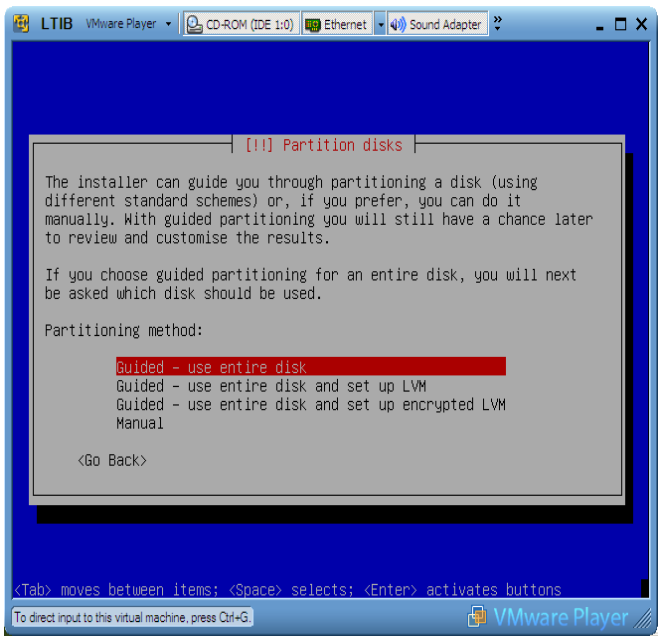


Figure H

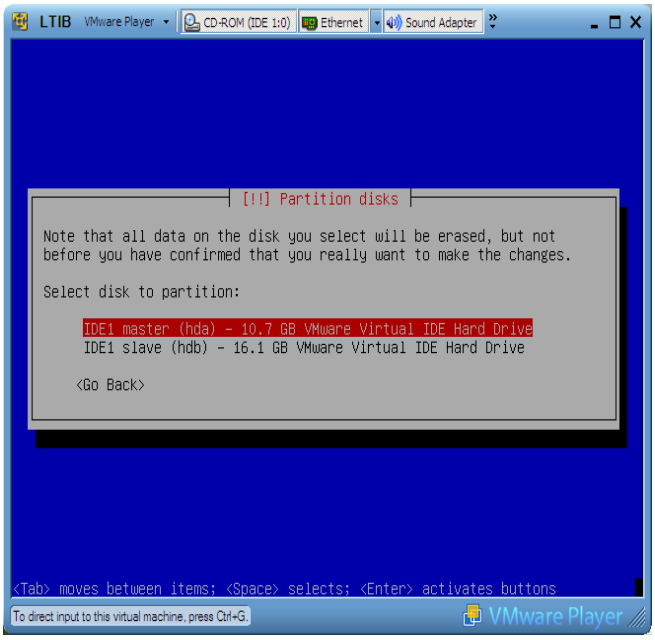


Figure I

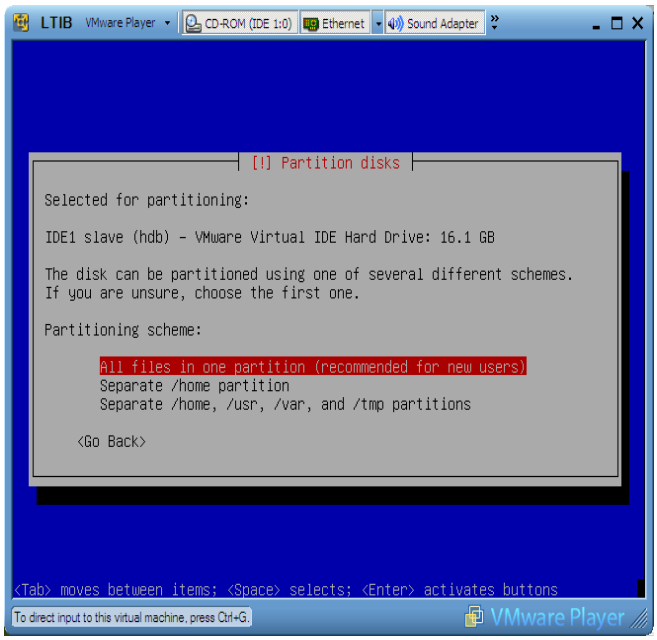


Figure J

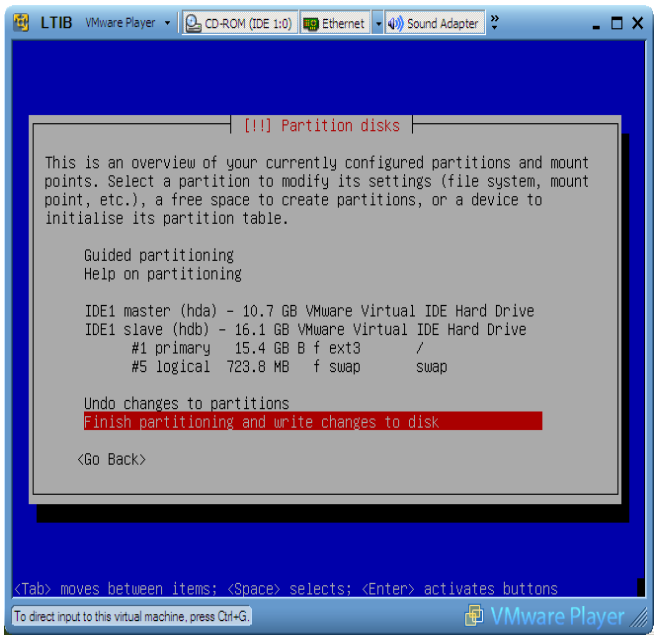


Figure K

Figure B-16. Disk Partition Stage

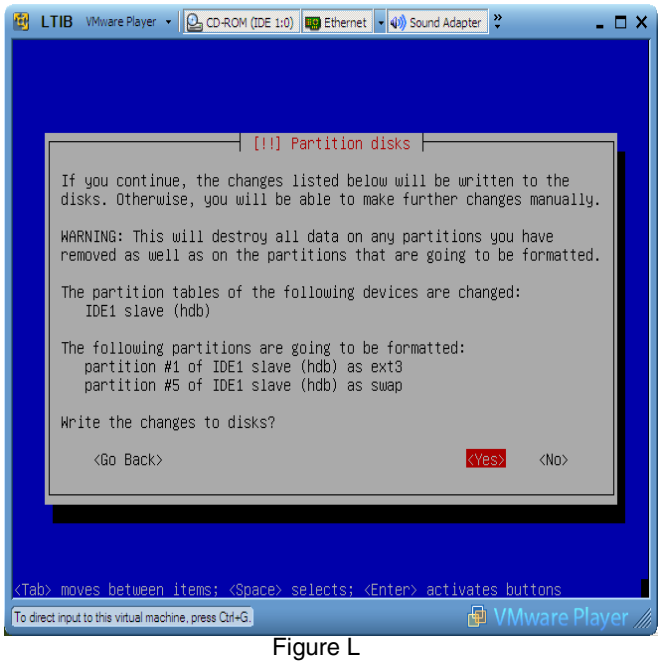


Figure L

Figure B-17. Disk Partition Stage

This is a summary of partitioning options for Figure B-16 and Figure B-17:

Figure H: Guided use entire disk VMWare IDE hard drive

Figure I: IDE master (hda) 10.7GB

Figure J: All files in one partition (recommended for new users)

Figure K: Finish partitioning and write changes to disk

Figure L: Yes

B.11 Time Zone

Select your time zone.

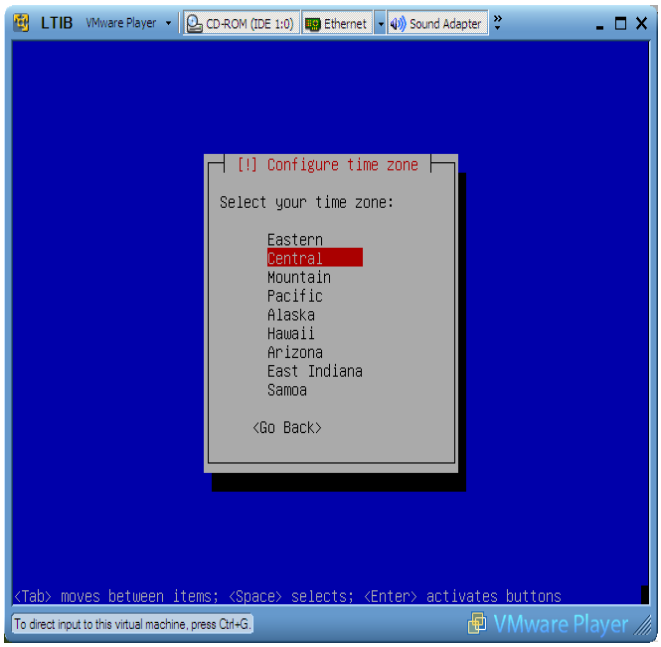


Figure B-18. Time Zone

B.12 Users and Passwords

Figure B-19 and Figure B-20 (M through R) set up the root, user passwords, and the login name. Make sure to remember both the root and user password. These passwords are used later to log in to the system and upgrade the user status to super user when necessary.

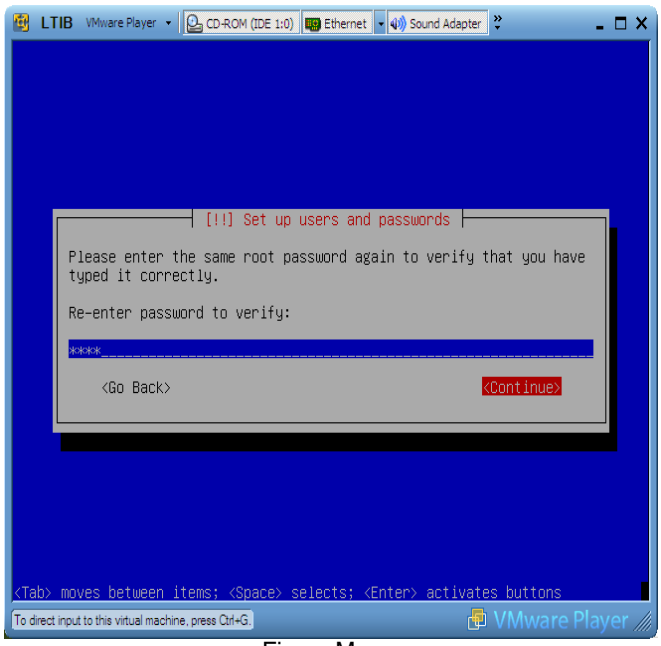


Figure M

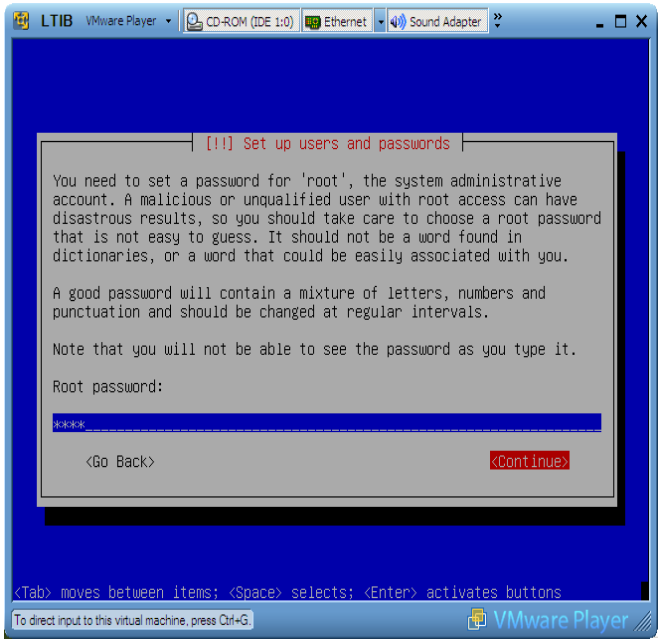


Figure N

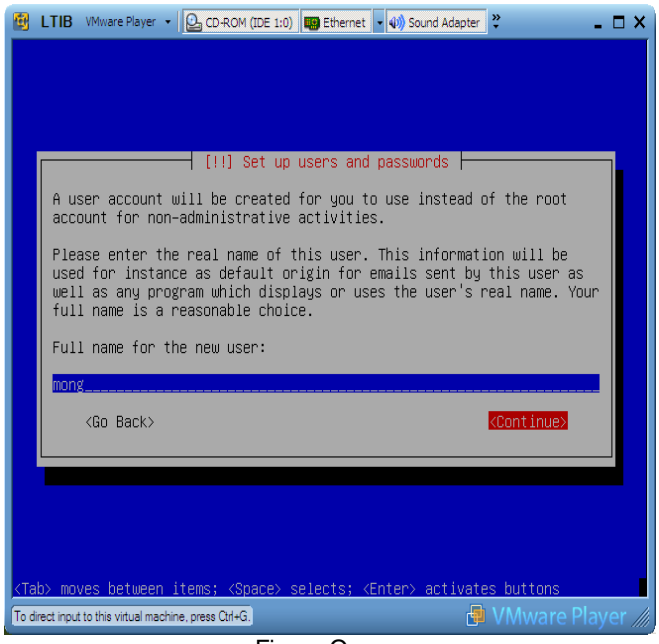


Figure O

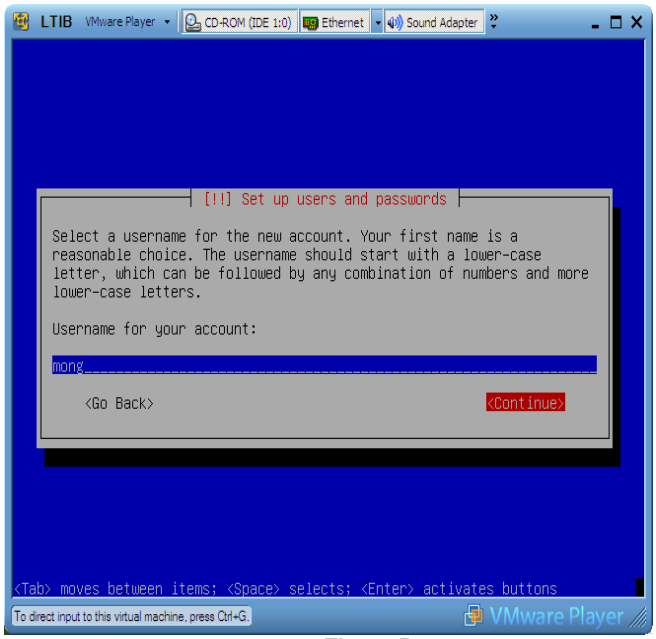


Figure P

Figure B-19. Users and Passwords

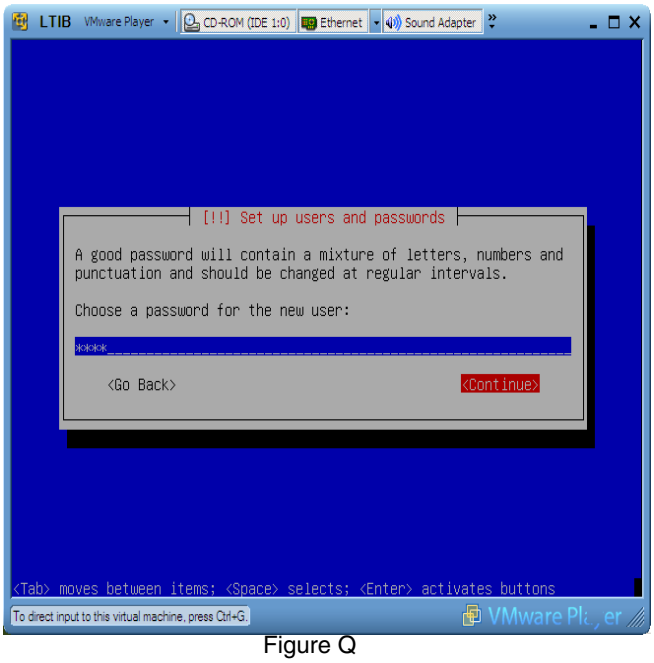


Figure Q

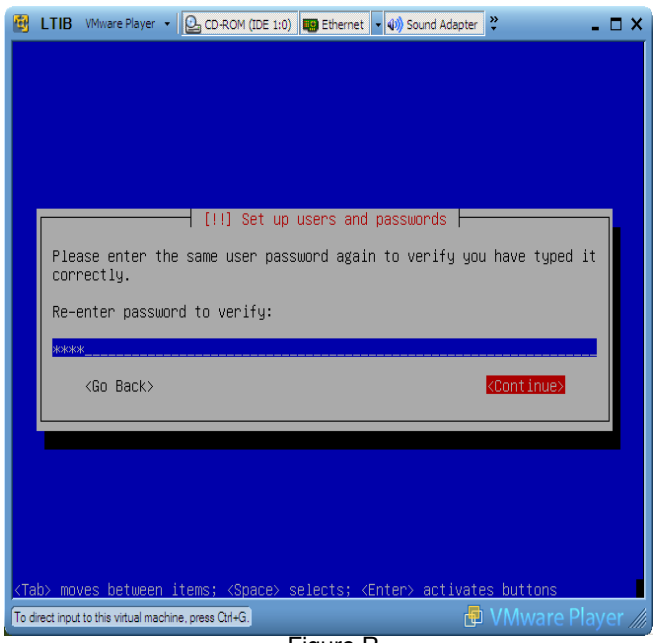


Figure R

Figure B-20. Users and Passwords

At this point the Debian installer installs the Linux base system shown in Figure B-21.

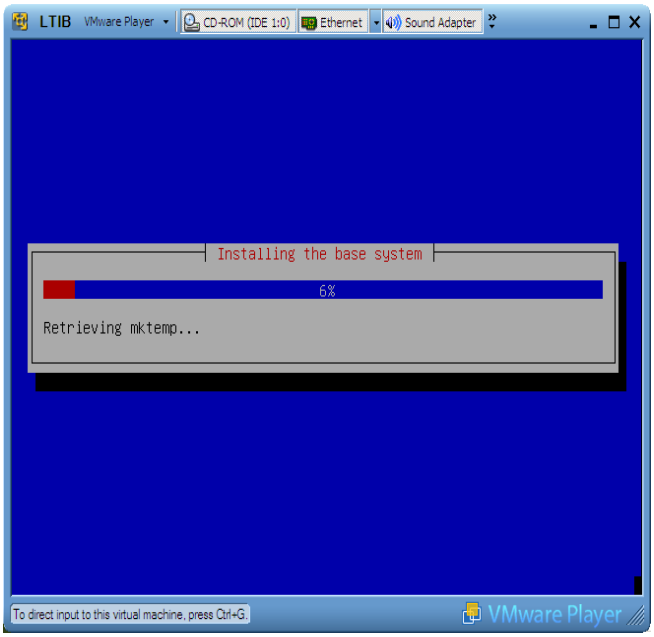


Figure B-21. Base System Installer

B.13 Configure Package Manager

The windows shown in Figures S to V are the package manager configurations. Make sure to enter the correct HTTP proxy information (if applicable) in the text entry box shown in Figure V.

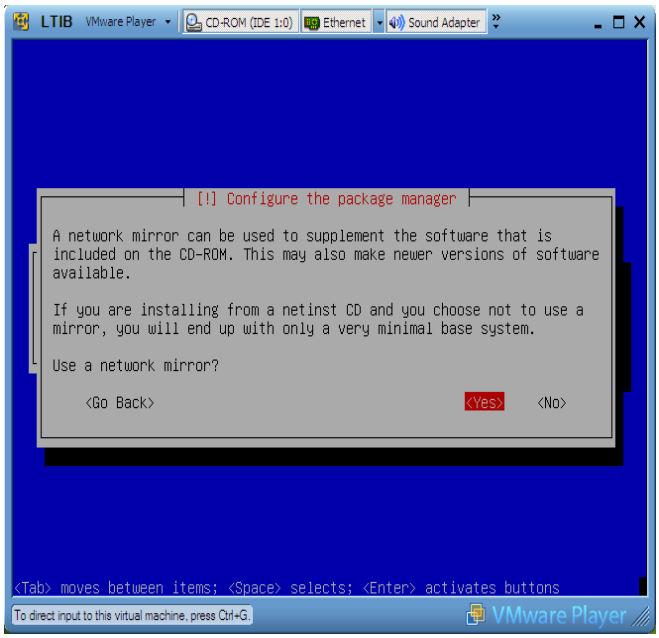


Figure S

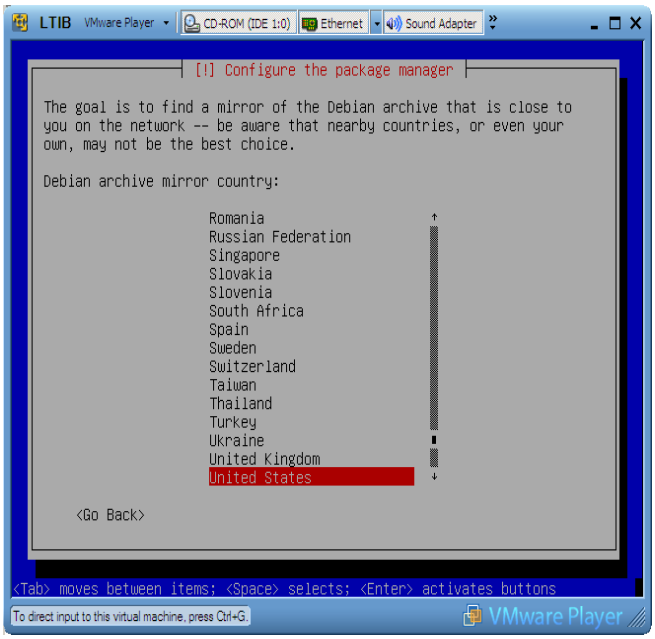


Figure T

Figure B-22. Package Manager Configuration

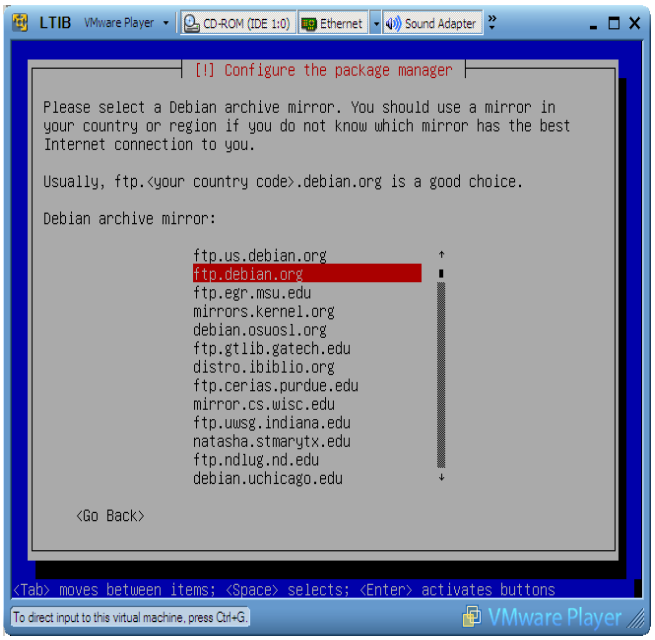


Figure U

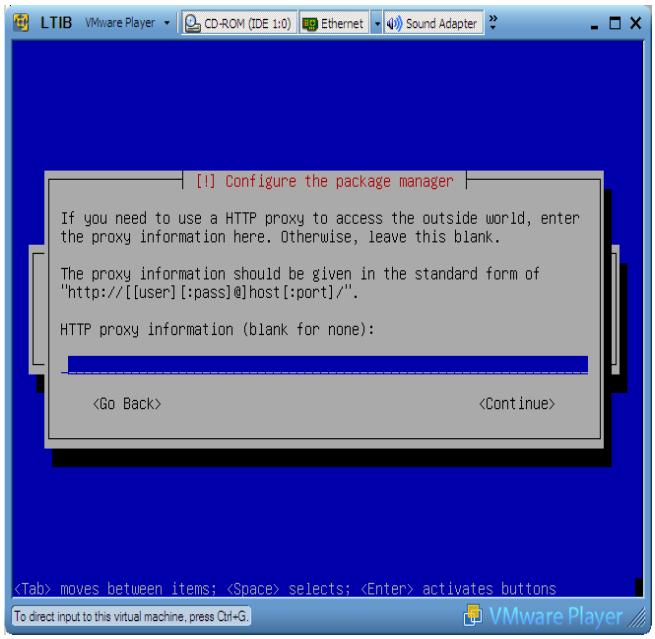


Figure V

Figure B-23. Package Manager Configuration

The Debian installer then scans the mirror site and configures the applications.

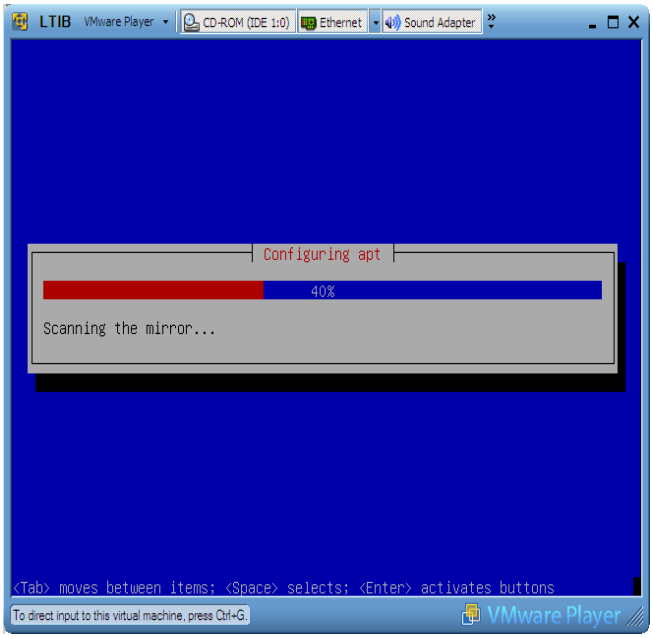


Figure B-24. Configuring

If participating in the package usage survey select Yes or No.

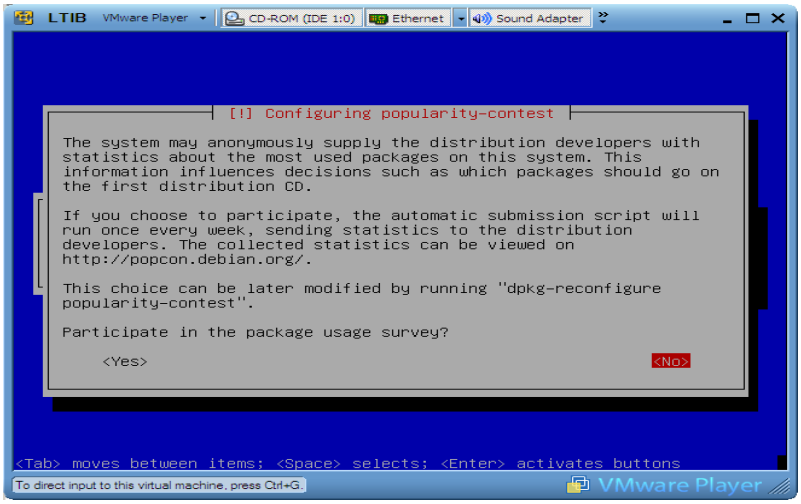


Figure B-25. Survey

B.14 Software Selection

Select desktop environment and standard system in Figure B-26.

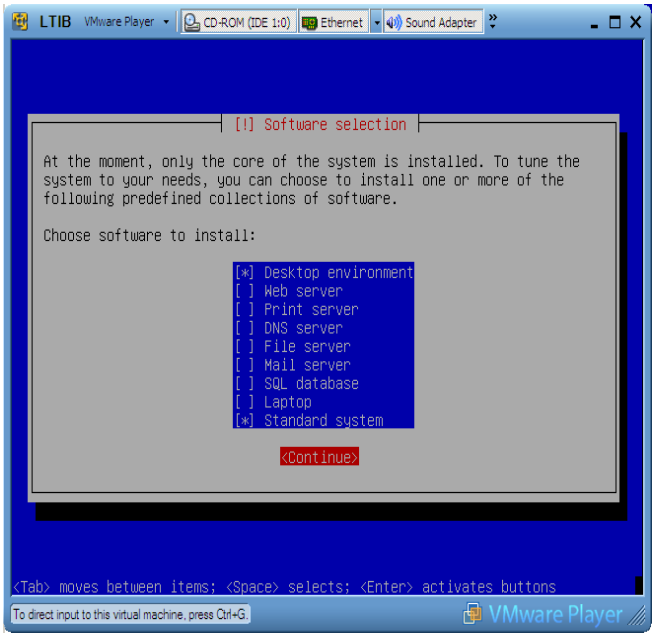


Figure B-26. Software Selection

The selected packages become installed. This process may take more than an hour depending on the mirror site and the network connection.

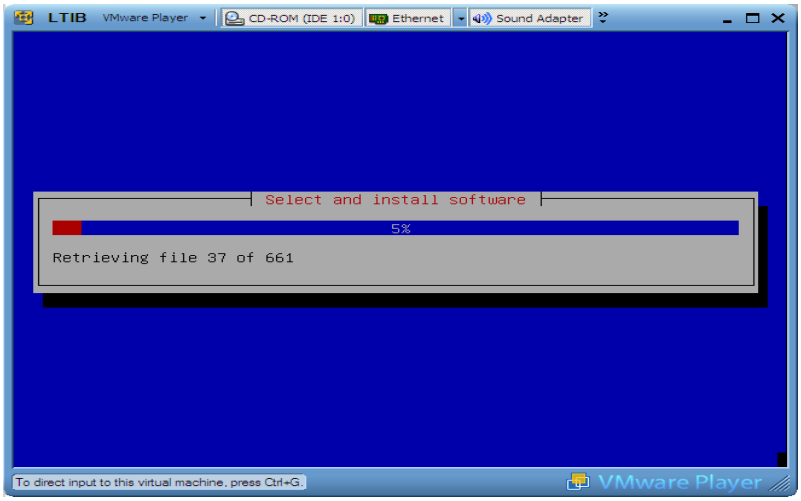


Figure B-27. Software Installation

Select Yes for the option shown in Figure B-28.

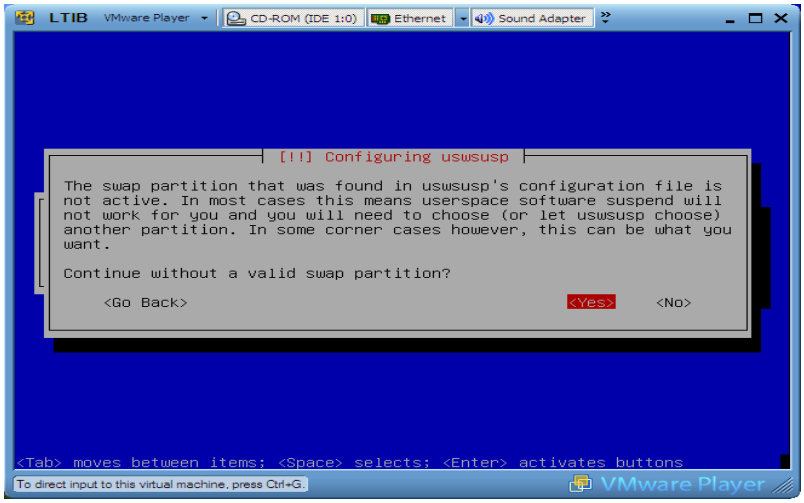


Figure B-28. Swap Partition

B.15 Configuring XServer Xorg

Select the video modes supported by the host machine. Notice the highest resolution checked is used as the default for the X session login screen.

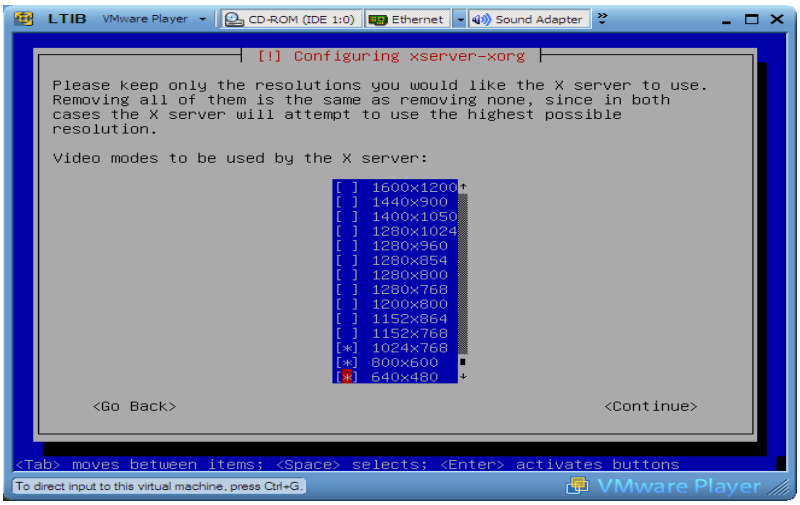


Figure B-29. Screen Resolution

B.16 Install the GRUB Bootloader

Debian now installs the GRUB boot loader on the hard disk. Select Yes in the first dialog box and then Continue.

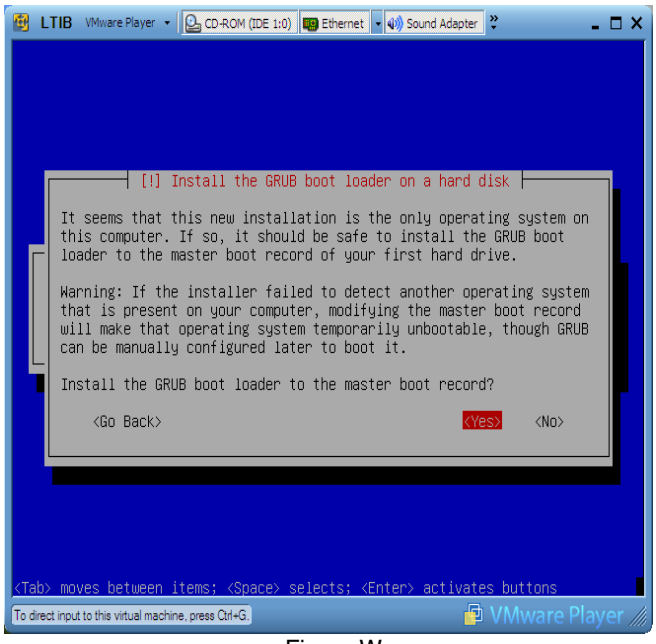


Figure W

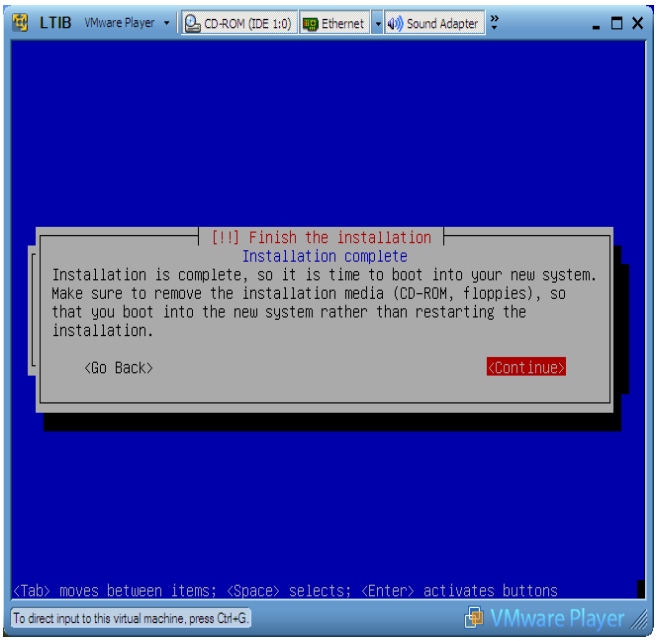


Figure X

Figure B-30. Finishing Installation

B.17 Finishing Installation

After the Debian installer is finished it ejects the installation CD and reboots the virtual machine into the newly installed OS. Use the username and password selected earlier in the installation process to now log in.

B.18 Install Freescale Version of LTIB for the MPC5121e

If you do not have the Freescale version of LTIB for the mpc5121e visit the Freescale website www.freescale.com and download a copy. It is easier to do this from within the virtual machine with one of the web browsers included in Debian Linux. If downloading the file outside of the virtual machine place it on a USB memory stick and copy it into the VM.

In the following instructions the **$** sign to indicate commands entered in a Linux shell command window and the => arrow to indicate commands entered at the u-boot on your MPC5121e ADS board.

B.19 Installing the BSP

Figure A

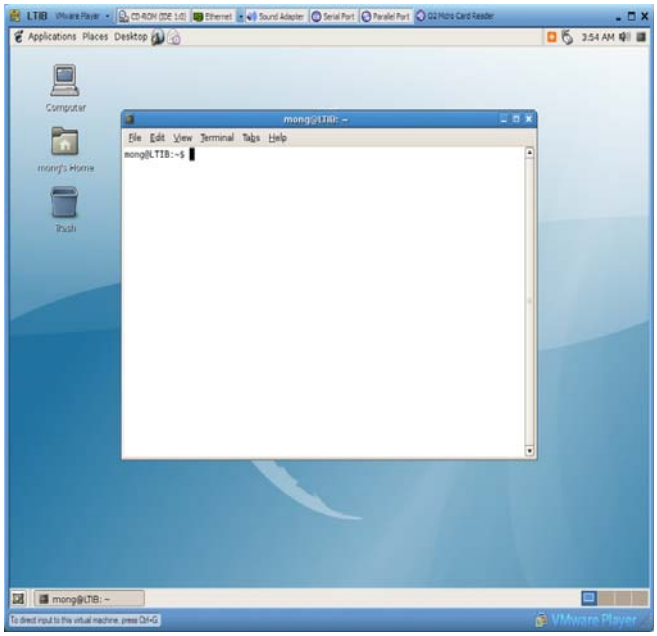


Figure B

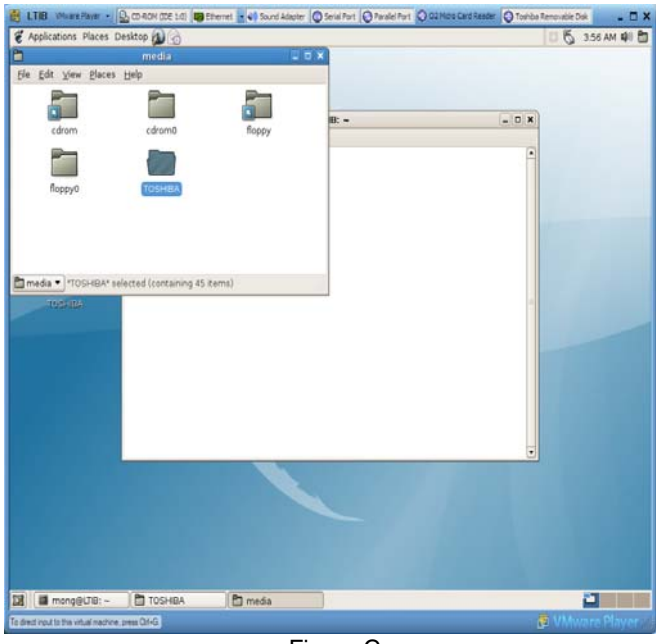


Figure C

Figure B-31. Installing LTIB

Select Bash from the desktop menu and a Bash command shell window appears see Figure B-31. Enter the following commands in sequence in the command shell window.

Table Linux Instructions:

`$ mkdir LTIBISO` — Creates a directory

`$ su` — Upgrades to super user to use root commands

`Password` — Enters the root password

`$ mount -o loop bsp-filename LTIBISO` — The full path to the LTIB install file downloaded earlier. The text bsp-filename in the mount command must be replaced by the full path and filename of the downloaded LTIB installed image file.

`$ exit` — Returns to non-root status

`$ LTIBISO/install` — Installs the BSP software

The installation requests where to install LTIB (/home/myusername). Accept the default location in the home directory or select another and press enter to continue.

B.20 Running LTIB

When LTIB is running, it writes information into /opt/freescale/pkgs and /opt/freescale/ltib. Make sure that LTIB has permission to write into these directories.

```
# /etc/sudoers
#
# This file MUST be edited with the visudo command as root.
#
# See the man page for details on how to write a sudoers file.
#

Defaults env_reset

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL) ALL
```

Add the following command in the file /etc/sudoers.

```
myusername ALL = NOPASSWD: /usr/bin/rpm, /opt/freescale/ltib/usr/bin/rpm
```

Enter the LTIB install directory and run LTIB.

`$ ./ltib` — Run LTIB

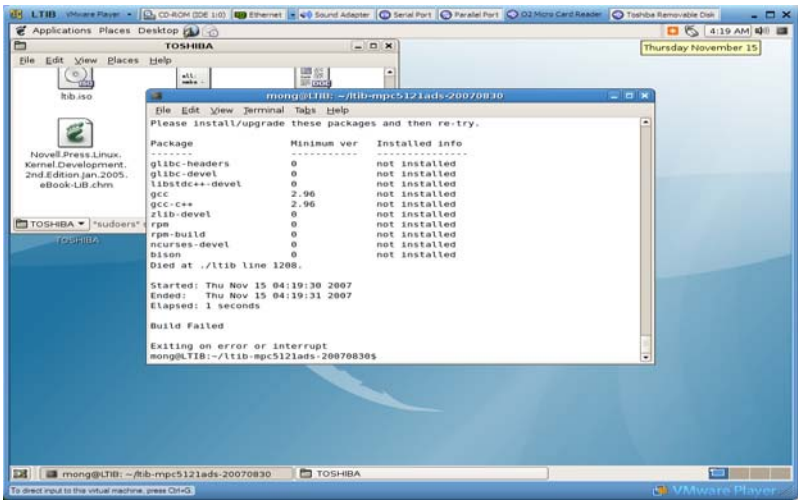


Figure B-32. Required Packages

The first time through, the LTIB detects if all the required software packages are present in the system. It generates a list of any missing software components needed for this process as shown in Figure B-32.

The following packages were not included in the default Debian Linux installation. Use the Synaptic Package Manager (available in the Debian desktop menus) to locate and install the missing packages.

- glibc-headers — For one to five search for gcc and g++
- glibc-devel — Selects gcc, gcc 3.4 and 4.1 , g++ and g++ 4.1
- libstd++.devel
- gcc
- gcc-c++
- zlib-devel — Search zlib and select zliblg-dev
- rpm
- rpm-build
- ncurses-devel — Search ncurses and select libncurses-dev
- bison

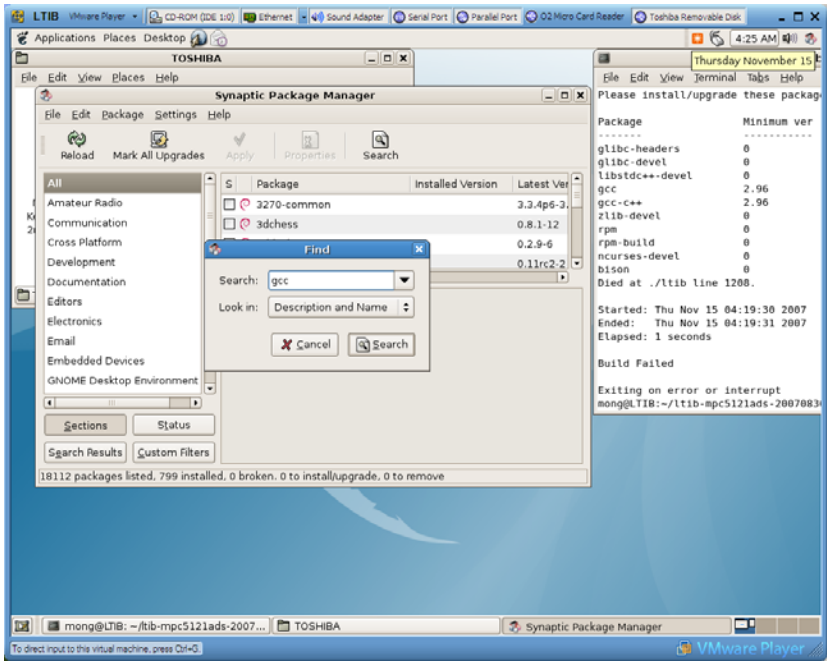


Figure B-33. Package Manager

Once all the missing software components are installed, run LTIB again.

```
$ ./ltib
```

When LTIB has successfully completed, the following message is displayed, see Figure B-34.

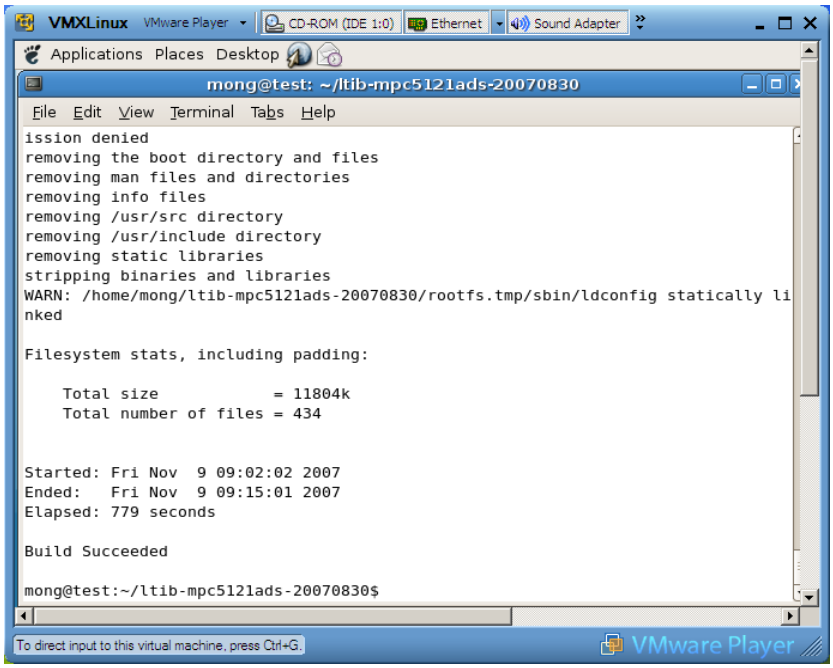


Figure B-34. LTIB Insallation Finished

B.21 Host Setup

Use the synaptic package manager to install the TFTP and NFS servers if they are not present. For NFS, install the NFS kernel server.

Create a TFTP directory for tftp transfers between the host and the target.

`$ su` — Upgrades to super user for using root commands

`Password` — Enters the root password

```
$ cd /
$ mkdir –p /tftpboot
```

Create a symbolic link to an exportable directory.

```
$ ln – s /home/myusername/litb-install-directory/rootfs /tftpboot/rootfs
```

Copy the kernel, boot loader, device trees, and root file system image to the /tftpboot directory

`$ su` — Upgrades to super user for using root commands

`Password` — Enters the root password

```
cp /home/myusername/litb-install-directory/rootfs /boot/uImage /tftpboot
cp /home/myusername/litb-install-directory/rootfs/boot/mpc5121ads.dtb /tftpboot
cp /home/myusername/litb-install-directory/rootfs /boot/u-boot.bin /tftpboot
```

B.22 Setup Host IP Address

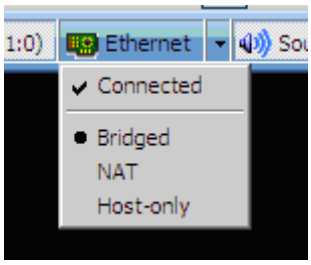


Figure B-35. Ethernet Configuration

In the ethernet option of the VMware player, make sure it is Connected and set to Bridged mode. If the host operating system obtains the IP address via the DHCP, the VM also obtains its IP address via the DHCP. If the network does not allow this manually set an IP address for the VM as follows:

`$ su` — Upgrades to super user to use root commands

`Password` — Enters the root password

```
$ ifconfig eth0 192.168.1.100 netmask 255.255.255.0 up
```

B.23 Set-up TFTP and NFS Server

Edit the file /etc/exports and add the following line. If the export files do not exist, create one.

```
/tftpboot/rootfs *(rw,no_root_squash,async)
```

NOTE

Make sure there are no spaces between the * and the (.

Edit the file /etc/xinetd.d/tftp to enable the tftp. If the tftp file does not exist, create one.

If the directory xinetd.d does not exist, most likely xinetd is not installed. If necessary use the Synaptic Package Manager to install xinetd.

```
service tftp
{
        disable             = no
        socket_type         = dgram
        protocol            = udp
        wait                = yes
        user                = root
        server              = /usr/sbin/in.tftpd
server_args                 = /tftpboot
}
```

B.24 Restart TFTP and NFS Server

Restart the TFTP and NFS server on the host as follows:

`$ su` — Upgrades to super user to use root commands

`Password` — Enters the root password

```
$ /etc/init.d/xinetd restart
$ /etc/init.d/nfs-kernel-server restart
```

If the target development board cannot connect to either tftp or NFS, use this restart process s a trouble-shooting method.

B.25 Target Development Board Setup

- Connect the board power supply
- Connect the ADS board to the network via the Ethernet port.
- Connect the ADS board to the host machine via a serial port. The console serial port on the ADS board is a DB9-type connector and must be labeled UART_PORT_0.
- Start a serial communication program of choice on the host PC. Use port settings, 115200 baud, 8 data bits, no parity, one stop bit, and no flow control.
- Power up the ADS board and make sure the u-boot banner appears in the serial terminal window. Press a key to halt the auto-boot process:

```
U-Boot 1.3.4 (August 27, 2008 - 11:00:02) MPC512X
CPU:   MPC5121e rev. 2.0, Core e300c4 at 399.999 MHz, CSB at 199 MHz
Board: ADS5121 rev. 0x0400 (CPLD rev. 0x05)
I2C:   ready
DRAM:  512 MB
FLASH: 64 MB
PCI:   Bus Dev VenId DevId Class Int
In:    serial
Out:   serial
Err:   serial
```

```
Net:   FEC ETHERNET
Type -- run jffs2boot -- to boot Linux
Hit any key to stop autoboot:  0
=>
```

B.26 U-Boot Configuration

The ADS board may have a Linux kernel and a file-system stored in flash. When rebuilding and reconfiguring the kernel, boot the image over NFS while working on it. The u-boot command to accomplish this is run net_nfs. Check the network settings on the board before doing this. Use the print command to display current environment variable settings. Use the setenv and saveenv commands to accordingly change the environment. If the network supports DHCP the u-boot can obtain an address with the dhcp command. Change the settings shown below to match the network.

```
=>setenv ipaddr 192.168.1.101
=>setenv serverip 192.168.1.100
=>setenv netmask 255.255.255.0
=>setenv gatewayip 192.168.1.254

=>setenv  rootpath /tftpboot/rootfs
=>setenv  bootfile /tftpboot/rootfs/boot/uImage
=>setenv  fdtfile /tftpboot/rootfs/boot/mpc5121adsrev4.dtb
=>setenv  fdt_addr_r 3000000

=>saveenv
```

Use the following environment as a reference point if encountering any errors booting Linux.

```
bootdelay=5
baudrate=115200
loads_echo=1
hostname=ads5121
loadaddr=400000
u-boot_addr_r=200000
kernel_addr_r=300000
ramdisk_addr_r=500000
u-boot_addr=FFF00000
fdt_addr=FC2C0000
ramdisk_addr=FC300000
ramdiskfile=ads5121/uRamdisk
netdev=eth0
consdev=ttyPSC0
nfsargs=setenv bootargs root=/dev/nfs rw nfsroot=${serverip}:${rootpath}
ramargs=setenv bootargs root=/dev/ram rw
addip=setenv bootargs ${bootargs} ip=${ipaddr}:${serverip}:${gatewayip}:${netmas
k}:${hostname}:${netdev}:off panic=1
addtty=setenv bootargs ${bootargs} console=${consdev},${baudrate}
flash_nfs=run nfsargs addip addtty;bootm ${kernel_addr} - ${fdt_addr}
flash_self=run ramargs addip addtty;bootm ${kernel_addr} ${ramdisk_addr} ${fdt_a
ddr}
net_nfs=tftp ${kernel_addr_r} ${bootfile};tftp ${fdt_addr_r} ${fdtfile};run nfsa
rgs addip addtty;bootm ${kernel_addr_r} - ${fdt_addr_r}
net_self=tftp ${kernel_addr_r} ${bootfile};tftp ${ramdisk_addr_r} ${ramdiskfile}
;tftp ${fdt_addr_r} ${fdtfile};run ramargs addip addtty;bootm ${kernel_addr_r} $
{ramdisk_addr_r} ${fdt_addr_r}
load=tftp ${u-boot_addr_r} ${u-boot}
update=protect off ${u-boot_addr} +${filesize};era ${u-boot_addr} +${filesize};c
```

```
p.b ${u-boot_addr_r} ${u-boot_addr} ${filesize}
upd=run load update
ethact=FEC ETHERNET
ethaddr=00:1e:59:7b:4e:eb
mynet=set ipaddr 192.168.0.11;set gatewayip 192.168.0.1;set netmask 255.255.255.
0;set serverip 192.168.0.199;tftp 400000 bkupflash40DEBUG.img;autoscr 400000
kernel_addr=0xff900000
preboot=echo Type -- run jffs2boot -- to boot Linux
fdtflashaddr=0xffec0000
kernelflashaddr=0xffc40000
consoledev=ttyPSC0
bmp_addr=0xffe40000
bootcmd=run jffs2boot
rootpath=/tftpboot/rootfs
fdt_addr_r=3000000
u-boot=/tftpboot/ADS5121_40_u-boot_1_3_4.bin
bootfile=/tftpboot/rootfs/boot/uImage
fdtfile=/tftpboot/rootfs/boot/mpc5121adsrev4.dtb
filesize=3000
fileaddr=100000
gatewayip=10.81.111.254
netmask=255.255.252.0
ipaddr=10.81.111.123
serverip=10.81.108.42
jffs2boot=setenv bootargs console=ttyPSC0,115200 root=/dev/mtdblock1 rw rootfsty
pe=jffs2 mem=256M;cp.b 0xffec0000 3000000 200000;bootm 0xffc40000 - 3000000
othbootargs=mem=512M
stdin=serial
stdout=serial
stderr=serial
```

B.27 Boot Over NFS

Enter the following command at the u-boot prompt to boot Linux over NFS:

```
=> run net_nfs
```

After the boot messages have scrolled by, the terminal must display a shell command prompt.

`-sh-2.05b# ls` — Enter ls to make sure the system works

```
bin      dev      home     linuxrc  opt      root     sys      usr
boot     etc      lib      mnt      proc     sbin     tmp      var
```

B.28 Replacing U-Boot

The process is straightforward, if necessary to update the u-boot on the board to a newer version.

Place the new u-boot binary file in the /tftpboot directory on the host PC.

On the target, make sure the u-boot environment variable matches the filename of the u-boot binary in the /tftpboot directory.

```
=> print u-boot
u-boot=/tftpboot/ADS5121_40_u-boot_1_3_4.bin
```

Run the load command to download the u-boot binary.

```
=> run load
Using FEC ETHERNET device
TFTP from server 10.81.108.79; our IP address is 10.81.111.123
Filename '/tftpboot/ADS5121_40_u-boot_1_3_4.bin'.
Load address: 0x200000
Loading: #########################################
done
Bytes transferred = 208704 (32f40 hex)
=>
```

If this succeeds, run the update command to flash the new version of the u-boot.

```
=> run update
Un-Protected 1 sectors

. done
Erased 1 sectors
Copy to Flash... done
=>
```

THIS PAGE IS INTENTIONALLY BLANK

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

freescale™
semiconductor

AN3765
Rev. 0
12/2008