

Linux[®] Startup Guide For the MPC8349E-MITX-GP Platform

By John Logan

The MPC8349E-MITX-GP platform demonstrates the capabilities of the Freescale PowerQUICC[™] MPC8349E processor. The hardware platform includes the following features:

- Mini-ITX form factor printed circuit board (PCB) with a Gigabit Ethernet port
- High-speed universal serial bus (USB) port
- Serial channels
- Peripheral component interconnect (PCI) slot
- Real-time clock controller
- Flash memory and DDR memory

A pre-installed board support package (BSP) consists of a bootloader (u-boot) and a generic Power Architecture[™] Linux[®] kernel and associated file system. U-boot, the Linux kernel, and the file system reside in flash memory. At power-up, the system runs the u-boot bootloader and is ready to boot Linux using u-boot commands. [Figure 1](#) shows a block diagram of the platform.

Also included is the Linux target image builder (LTIB), which bundles a suite of tools that leverage open source configuration scripts and source code into a single BSP. The source code packages include boot loaders and Linux kernel

Contents

1	Requirements	2
2	MPC8349E-MITX-GP Linux Overview	3
2.1	Bootloader	3
2.2	Linux Kernel	3
2.3	Root File System	3
2.4	Linux Target Image Builder (LTIB)	4
3	System Boot From Flash Memory and Ethernet Setup	4
3.1	System Setup	4
3.2	Mounting A USB Device	7
4	Using LTIB to Customize the Linux System	8
4.1	Installing LTIB	8
4.2	Using LTIB To Modify The System	9
5	Loading the New Linux Build	12
5.1	Using TFTP to Load the Kernel and File System	12
5.2	Setting Up a TFTP Server On a PC	12
5.3	Loading the Kernel and File System Using TFTP	13
6	Network File System (NFS) on the PC	16

sources as well as many user-space source code packages. LTIB also provides a compiler (gcc) and associated tools required to build a complete system.

This application note covers the following topics:

- Brief overview of embedded Linux.
- Start-up demonstration of a complete Linux system with on-board Ethernet and USB connectivity.
- How to install the Linux target image builder (LTIB) supplied with the platform.
- Walkthrough using LTIB to modify the Linux kernel and file system.
- Setups for Linux debugging.

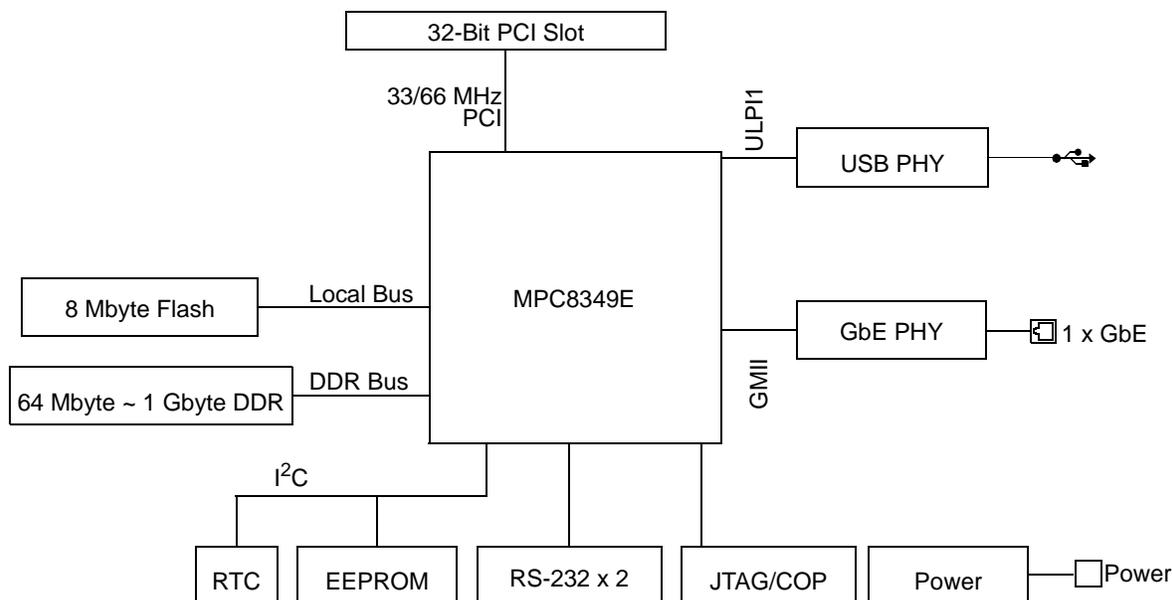


Figure 1. Block Diagram of MPC8349E-MITX-GP

1 Requirements

To use this application note the following equipment is required:

- A MPC8349E-MITX-GP system.
- A PC running Linux natively or through virtualization software such as VMWare.
- A terminal emulation program, for example, minicom, which is commonly pre-installed on most Linux distributions.
- A USB flash memory stick formatted for FAT file systems. Memory capacity of the stick is not important.

This application note was developed on a PC running the Debian Etch Linux distribution on VMWare Workstation virtualization software. In Debian-based systems, minicom can easily be installed using the apt-get system. Enter the following commands on the Linux PC while logged in as root:

```
apt-get update
apt-get install minicom
```

2 MPC8349E-MITX-GP Linux Overview

An embedded Linux system can be broadly split into three components: a bootloader, a Linux kernel, and a root file system. The MPC8349E-MITX-GP is supplied with a tool called Linux target image builder (LTIB) that can be used to build each of these components.

2.1 Bootloader

The bootloader is a small program that resides in flash memory and runs when the MPC8349E processor is powered up or resets. Its main roles are to allow loading of the Linux kernel and/or file system into system RAM and initiate booting (starting) the kernel. The bootloader supplied with the MPC8349E-MITX-GP is the Universal Bootloader (u-boot), an open source program that supports a wide range of processors and boards. It configures the processor buses, registers, and memory map and provides a command line interface through the serial port. It includes a wide range of commands and features to make developing and booting an embedded Linux system easy. U-boot is pre-programmed into the flash memory on the MPC8349E-MITX-GP.

2.2 Linux Kernel

The Linux kernel contains the core of the operating system; it provides services and functions in four main areas:

- Process management and communications
- Memory system management
- Hardware device management
- System calls

The Linux kernel is the link between user programs and the hardware resources of an embedded system. It contains the low-level device drivers required for any peripherals on the processor (for example drivers for the MPC8349 Ethernet controllers) and any peripherals connected to the system (for example, peripherals on the PCI bus). A full description of the kernel architecture and features are beyond the scope of this application note. Many available books describe the kernel and its workings in detail. The version of the kernel shipped with the MPC8349E-MITX-GP may vary over time as new versions are tested.

2.3 Root File System

A file system is a method of storing and managing data. A Linux system requires at least one file system to operate, and it uses a hierarchical directory structure for file storage. The file system can reside on a local storage device such as a hard drive, volatile or non-volatile memory, or remotely on another device or board acting as a file server using a network protocol such as NFS for access. The file system contains a mix of configuration files, programs, scripts, and data files. File systems must be mounted before they can be used. Typically, the root file system is mounted at /, the root directory. Other file systems, for example a file system stored on a USB memory stick, can be mounted at different directories, as demonstrated later in this application note.

2.4 Linux Target Image Builder (LTIB)

Developers can use LTIB to build the u-boot boot loader, configure the Linux kernel, select user-space packages and options for a root file system, and perform many other tasks required in a Linux environment. LTIB is an open-source tool and is licensed under the GNU General Public License. It uses open-source technologies such as the RPM package management tool and Linux Kernel Config. Taking advantage of open source technologies helps Freescale keep LTIB up to date with changes to Linux in the open source community.

Freescale provides LTIB as a component of its 2.6 kernel Linux BSPs. It is available through the following sources:

- CD-ROM in the MPC8349E-MITX-GP kit
- Downloadable from the Freescale web site listed on the back cover of this document
- Downloadable from <http://www.bitshrine.org>, an open source web site.

Developers can download updates such as new BSP components, software patches, and updates for the LTIB tool. For this application note, the version of LTIB supplied on the CD-ROM in the MPC8349E-MITX-GP kit is used.

3 System Boot From Flash Memory and Ethernet Setup

Out of the box, a user can quickly have an embedded Linux system running in minutes. The 10-minute demo presented in this section shows how to boot Linux, configure Ethernet connectivity, and mount a USB memory stick.

3.1 System Setup

System setup proceeds as follows:

1. Connect a serial cable (supplied in MPC8349E-MITX-GP kit) from the MPC8349E-MITX-GP to the PC.
2. Connect the MPC8349E-MITX-GP power supply, but do not power up the board yet.
3. Connect an Ethernet cable (supplied in the MPC8349E-MITX-GP kit) between the MPC8349E-MITX-GP board and the PC.
4. Configure the Ethernet port on your PC to IP address 192.168.1.1, netmask 255.255.255.0. On a Linux machine, the port can be configured by typing the command `ifconfig eth0 192.168.1.1`, assuming you are using Ethernet port eth0.
5. Start the terminal emulation program on the PC.
For example, if using minicom, type `minicom`. You may need to be logged in as root, depending on your particular distribution.
6. Configure the terminal program for 115200 baud, 8 data bits, no parity, 1 Stop bit.
7. Power up the MPC8349E-MITX-GP. The u-boot bootloader on the board starts. Figure 2 shows the boot display.
8. Execute the `run flashramboot` command to transfer a fully functional Linux kernel from flash memory to DDR RAM and boot.

A complete file system is also decompressed from flash memory into RAM. A standard Linux login prompt appears, and you can log in and start to use standard Linux commands such as `ls` and `ifconfig` to use the system.

9. Log in using the username `root` and the password `root`.
10. Enter the `ifconfig` command.

Figure 3 shows a screenshot after a user logs into the system and enters the `ifconfig` command. Notice that Ethernet port `eth0` does not have an assigned IP address. This is because the Linux kernel booted on the MPC8349E-MITX-GP board has `dhcp` support enabled by default, so it expects the PC or network to supply it with an IP address at startup. Because we are using a direct connection to a PC with a static IP address assigned, the board should have `dhcp` disabled and have its own static IP address assigned at bootup. Later, we show how to change the configuration to provide this behavior using `LTIB`.

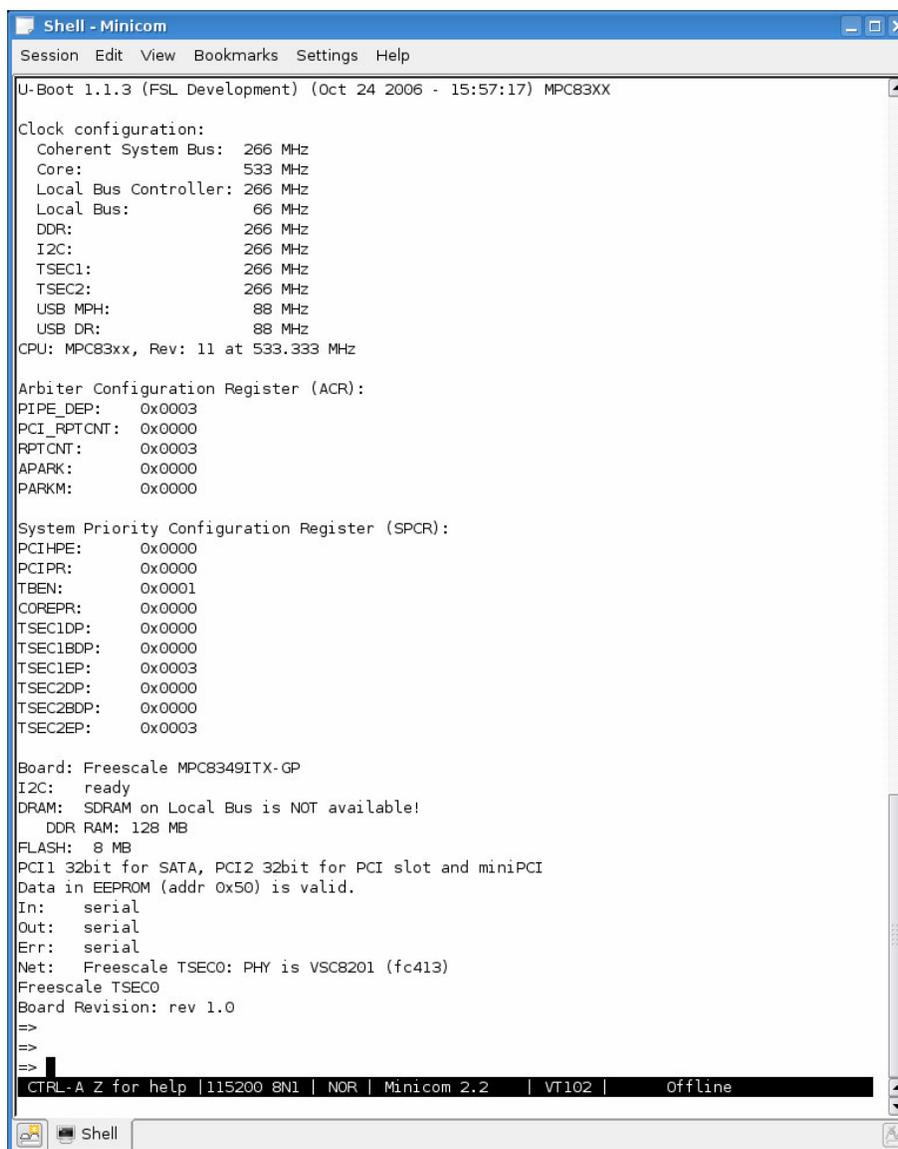


Figure 2. U-Boot Startup Screen Using Minicom Terminal Emulator in Linux



Figure 3. Linux Login Display

11. Assign an IP address to the MPC8349E-MITX-GP Ethernet port using the standard Linux `ifconfig` command:

```
ifconfig eth0 192.168.1.2
```

Full TCP/IP networking is now available and can be configured using standard Linux commands such as `route` and `netstat`. The Busybox package is also included, allowing telnet, tftp and other networking protocols in addition to many other Linux tools (ping, vi, and so on).

12. Try pinging between the board and the PC. In the terminal window, enter the command:

```
ping 192.168.1.1
```

If the connection is successful, the terminal window display should look like that shown in [Figure 4](#).

13. Stop the ping command using the CTRL-Z keyboard sequence.

```

Shell - Konsole
Session Edit View Bookmarks Settings Help

~ # ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1): 56 data bytes
84 bytes from 192.168.1.1: icmp_seq=0 ttl=128 time=1.7 ms
84 bytes from 192.168.1.1: icmp_seq=1 ttl=128 time=1.7 ms
84 bytes from 192.168.1.1: icmp_seq=2 ttl=128 time=1.6 ms
84 bytes from 192.168.1.1: icmp_seq=3 ttl=128 time=1.6 ms
84 bytes from 192.168.1.1: icmp_seq=4 ttl=128 time=1.7 ms
84 bytes from 192.168.1.1: icmp_seq=5 ttl=128 time=1.7 ms
84 bytes from 192.168.1.1: icmp_seq=6 ttl=128 time=1.8 ms

--- 192.168.1.1 ping statistics ---
7 packets transmitted, 7 packets received, 0% packet loss
round-trip min/avg/max = 1.6/1.6/1.8 ms
~ # █
    
```

Figure 4. Result Of Successful ping Command

3.2 Mounting A USB Device

USB connectivity also works out of the box, allowing connection of USB peripherals such as memory devices. For example, to mount a USB flash memory stick, insert the flash memory stick into the mini USB socket on the back of the MPC8349E-mITX-GP using the supplied miniA-to-typeA converter. When the memory stick is detected, its details are displayed in the terminal window. The memory stick can then be mounted into the Linux file system using the following commands:

- `mkdir usbdrv`. Creates a directory where the memory stick can be mounted.
- `mount -t msdos /dev/sda1 usbdrv`. Mounts the memory stick into the file system.

The contents of the memory stick can now be viewed using the `ls usbdrv` command. [Figure 5](#) shows the terminal display when a 1 Gbyte memory stick with MS-DOS formatting is mounted. In this case, the memory stick contains 1 folder (`backup`) and 1 file (`training.ppt`).

As this 10-minute demonstration has shown, with a few simple commands, you can boot a complete Linux system, enable Ethernet communications, and mount a USB peripheral.

4 Using LTIB to Customize the Linux System

In the 10-minute demo, we noticed that the default Linux system stored in flash memory has dhcp support enabled. This section explains how to use LTIB to set a static IP address for the board. Also, it discusses how to add extra packages to the file system.

4.1 Installing LTIB

To install LTIB, perform the following steps:

1. Insert the Linux BSP CD supplied with the MPC8349E-MITX-GP into the PC CD drive and mount the disk. Typical commands to mount a CD are:

```
mkdir /mnt/cdrom. Create a directory for mounting the CD.
```

```
mount -t iso9660 /dev/cdrom /mnt/cdrom. Mount the CD
```

You must be logged in as a superuser (or root) to run these commands successfully.

2. Run the install script on the CD using the following command:

```
/mnt/cdrom/ltib-mpc8349e-mitx-gp/install
```

You should not use superuser to run the install script; LTIB issues a warning if you attempt it.

The installer displays license information and prompts you to type `yes` or `no` to accept or reject the license. It then asks for an installation directory. Typically, a subdirectory of the user's home directory should be used. By default, LTIB is installed in the `ltib-mpc8349itx-gp-20061024` subdirectory of the current working directory.

The installer copies various packages to the PC's hard drive.

3. When copying has completed, move to the directory where LTIB was copied and type

```
./ltib
```

LTIB finishes installing its packages and builds a complete BSP for the MPC8349E-MITX-GP (u-boot bootloader, Linux kernel, and root file system image). Depending on your Linux distribution, you may need to install some additional packages to enable LTIB to run. It verifies that all packages it requires are installed and lists any missing packages. For example, on Debian systems, `zlib-dev` and `ncurses-dev` need to be installed. These packages are all freely available for download. Refer to your particular Linux distribution documentation for details on installing packages (for Debian systems, the `apt-get` command can be used to install additional packages)

4.2 Using LTIB To Modify The System

LTIB provides a set of scripts and a graphical front end to allow the user to modify the Linux kernel and file system. When all modifications are complete, LTIB compiles and builds the kernel and file system and outputs them in a range of user-configurable formats.

We now use LTIB to set a static IP address for the Ethernet port on the MPC8349E-MITX-GP, remove the dhcp option, and add a new package (a simple "Hello World" package) to the file system, as follows:

1. Start LTIB with command `./ltib --configure`.

A graphical menu system appears so that you can choose kernel and file system options. [Figure 6](#) shows this top-level menu.

2. Select option `Package List` using arrow keys and return key to show a list of available packages. From here, you can select from a huge range of Linux applications and libraries and add them to the file system. Scroll down to package `hello world`. When you press the space bar to select the package, an asterisk appears beside the package name. See [Figure 7](#).
3. To open a new list of settings, select the Options menu just below the `Target System Configuration` option on the top-level menu.
4. To open a list of options relating to the IP address, select `Network setup`. Note that the `eth0` interface is configured for `dhcp`. See [Figure 8](#).
5. Remove `dhcp` support by selecting `get network parameters using dhcp` and pressing space bar. A list of static IP address options is listed for `eth0`.
6. Select each option, press return, and type in an IP address. IP address settings for this demo are as follows (see [Figure 9](#)):
 - IP Address 192.168.1.10
 - Netmask 255.255.255.0
 - Broadcast address 192.168.1.1
 - Gateway address 192.168.1.1
 - Nameserver IP address 192.168.1.1
7. When all IP addresses set, exit to the top-level menu and then exit from the top-level menu. A dialog box asks `Do you wish to save your new configuration?`
8. Select `yes` and press enter.

LTIB begins building the kernel and file system. Depending on your system, this may take several minutes. On completion, a `Build Succeeded` message appears on the PC screen display.

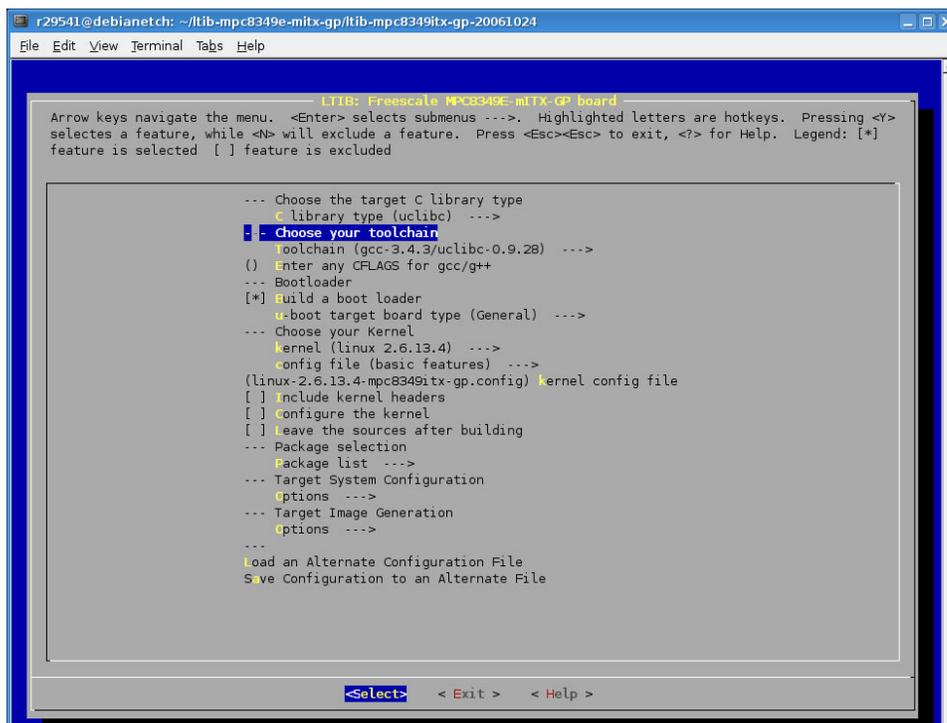


Figure 6. LTIB Top-Level Menu

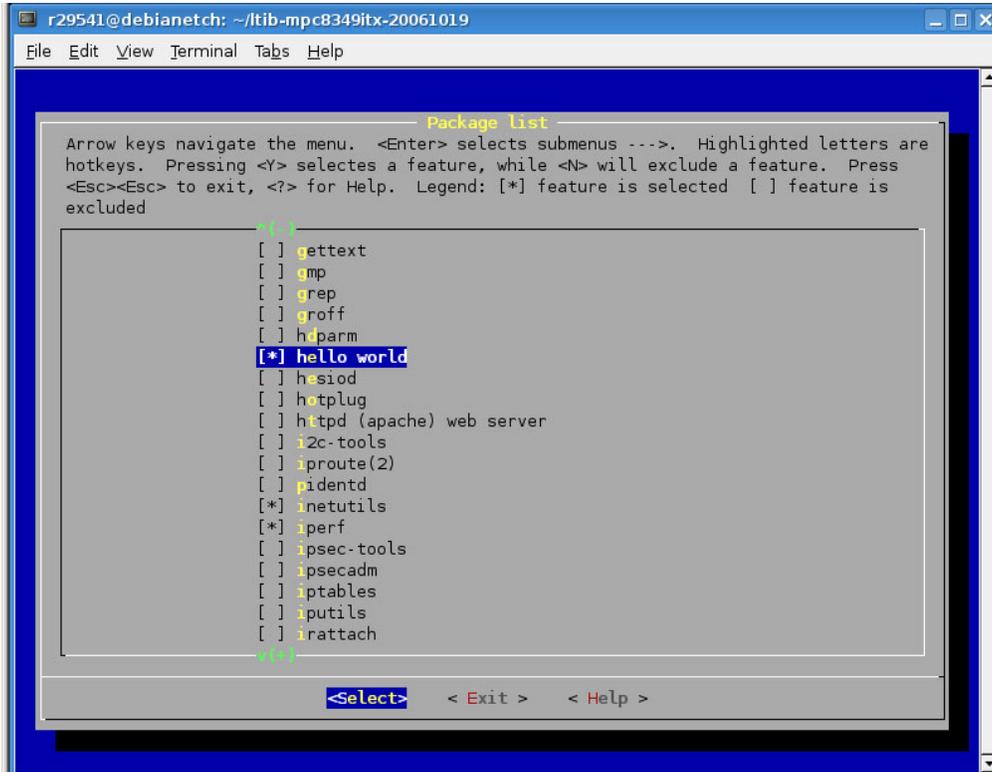


Figure 7. LTIB Package List Menu With “hello world” Selected

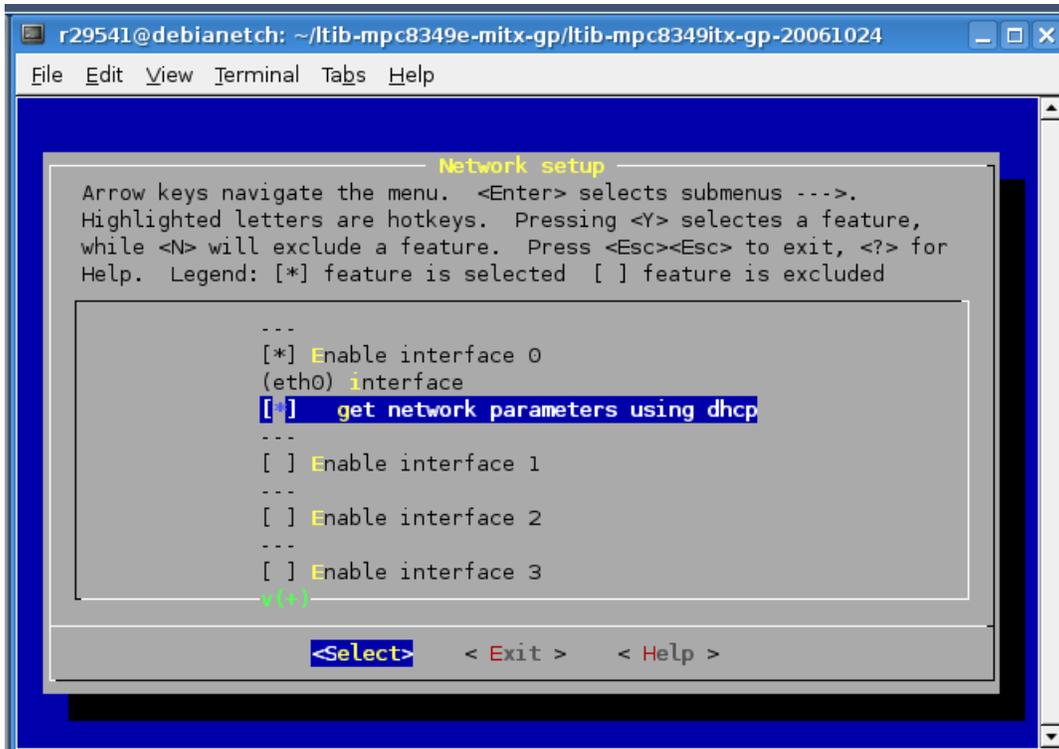


Figure 8. Network Settings With DHCP Enabled

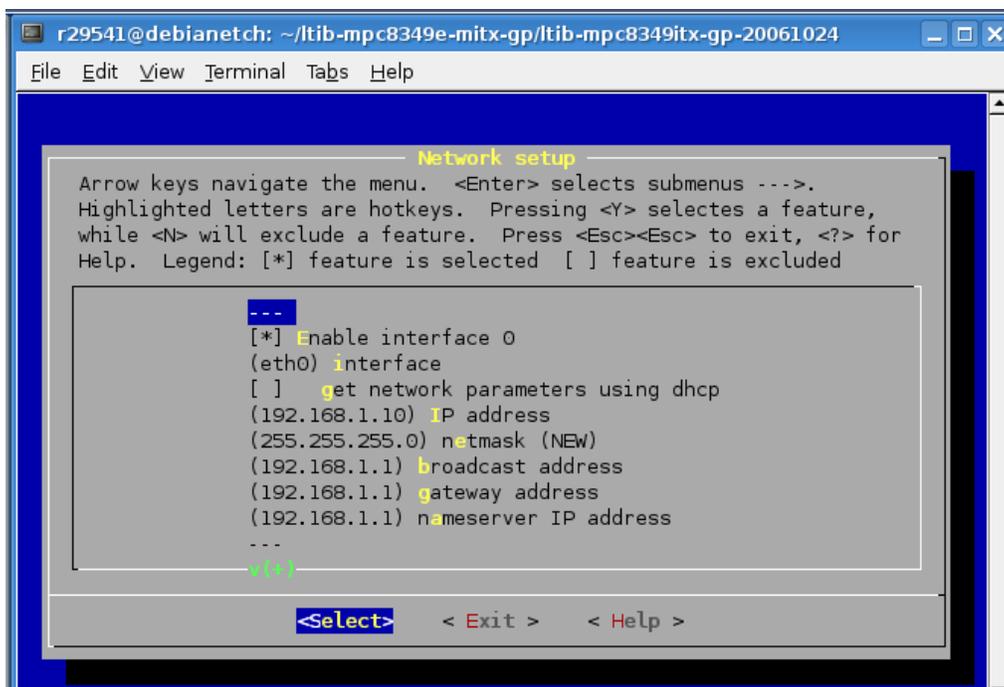


Figure 9. Network Settings With Static IP Addressing

5 Loading the New Linux Build

In the demo discussed in Section 3, “System Boot From Flash Memory and Ethernet Setup,” both the Linux kernel and file system are loaded from flash memory into system RAM. This is typical in a completed design, but it is not the only option for debugging a system. Because we have just modified the file system, we need to find another way to load it into system memory. For example, the kernel and file system can be loaded from a server or PC through Ethernet using the TFTP protocol. This is how we load our modified system.

5.1 Using TFTP to Load the Kernel and File System

Trivial File Transfer (TFTP) is a simple file transfer protocol. U-boot has built-in TFTP support, so it can load files into RAM through the MPC8349E-MITX-GP Ethernet port. Therefore, the Linux kernel and/or root file system can be loaded and decompressed. First, you must set up a TFTP server on your PC.

5.2 Setting Up a TFTP Server On a PC

The PC must have a TFTP server running to allow TFTP transfers between a PC and the MPC8349E-MITX-GP board. For Linux, TFTP is a popular choice. There are many ways to install TFTP servers on Linux machines. On the Debian system, the following procedure was used to install TFTP (all commands entered when logged in as root):

1. Install TFTP and related packages using the following commands on the PC:

```
apt-get update
apt-get install tftpd
apt-get install xinetd
```

2. Create or edit a file using the vi editor (`vi /etc/xinetd.d/tftp`) and add the following entry:

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

3. Create a `/tftpboot` directory using the following commands:

```
mkdir /tftpboot
chmod -R 777 /tftpboot
chown -R nobody /tftpboot
```

4. Start TFTP using this command:

```
/etc/init.d/xinetd start
```

5.3 Loading the Kernel and File System Using TFTP

When we use LTIB to modify the system, a Linux kernel image file is generated and placed at:

```
ltib-mpc8349itx-gp-20061024/rootfs/boot/uImage.
```

A root file system is also created (in the `ltib-mpc8349itx-gp-20061024/rootfs` directory), and a compressed image file of the file system is stored at:

```
ltib-mpc8349itx-gp-20061024/rootfs.ext2.gz.uboot
```

We use TFTP and these two files to restart our modified system, as follows:

1. Copy the kernel and file system images to the `tftpboot` directory on the PC using commands:

```
cp rootfs/boot/uImage /tftpboot
cp rootfs.ext2.gz.uboot /tftpboot
```

These commands assume you are in the LTIB install directory. Refer to [Figure 10](#).

2. Connect the MPC8349E-MITX-GP board to the PC as described in [Section 3, “System Boot From Flash Memory and Ethernet Setup.”](#)
3. Start minicom as described in [Section 3, “System Boot From Flash Memory and Ethernet Setup.”](#)
4. Power up the MPC8349E-MITX-GP board.
u-boot starts and a prompt appears in minicom.
5. Enter the following commands at the prompt in minicom:

```
setenv ipaddr 192.168.1.10
setenv serverip 192.168.1.1
setenv gatewayip 192.168.1.1
setenv tftp_path ''
saveenv
```

These commands modify various environment variables in u-boot relating to IP addresses and the TFTP directory path. Refer to [Figure 11](#).

U-boot uses environment variables to configure various options such as IP addresses. To list all the environment variables, enter the `printenv` command. Then the `help` command displays all u-boot commands. A full online manual that completely describes u-boot is available at www.denx.de.

6. Enter the `run tftpramboot` command at the prompt in minicom.

The display shows the board downloading the kernel (`uImage` file) and file system (`rootfs.ext2.gz.uboot` file) from the PC using tftp protocol, booting the kernel, and decompressing the file system. [Figure 12](#) shows the download in progress.

7. After successful download and boot, a standard Linux login prompt appears and you can log in and start to use standard Linux commands. Log in using the username `root` and password `root`.
8. After logging in, enter the `ifconfig` command.

The `eth0` Ethernet interface now has a static IP address assigned.

9. Enter the `ls /usr/bin` command. Note that there is a file labelled `hello` listed.

10. To run this file, enter the `/usr/bin/hello` command. See [Figure 13](#).

At this point, you have modified the system, downloaded it to the board, booted the system, and verified that the changes work.

```

r29541@debianetch: ~/ltib-mpc8349itx-20061019
File Edit View Terminal Tabs Help
Filesystem stats, including padding:

  Total size           = 49780k
  Total number of files = 566

Your ramdisk exceeds the old default size of 4096k, you may need to
set the command line argument for ramdisk_size in your bootloader
to at least 49780k .  For instance, for u-boot:

setenv bootargs root=/dev/ram rw ramdisk_size=49780

creating an ext2 compressed filesystem image: rootfs.ext2.gz
creating a uboot ramdisk image: rootfs.ext2.gz.uboot
Image Name:   uboot ext2 ramdisk rootfs
Created:     Wed Mar 14 20:14:13 2007
Image Type:  PowerPC Linux RAMDisk Image (gzip compressed)
Data Size:   12352166 Bytes = 12062.66 kB = 11.78 MB
Load Address: 0x00000000
Entry Point: 0x00000000

Started: Wed Mar 14 19:43:37 2007
Ended:   Wed Mar 14 20:14:18 2007
Elapsed: 1841 seconds

Build Succeeded

r29541@debianetch:~/ltib-mpc8349itx-20061019$ ls
bin      dist      ltib      RELEASE_INFO  rootfs.ext2.gz.uboot  tmp
config  doc       ltib_bashrc  rootfs      rpm
COPYING host_config.log  README     rootfs.ext2.gz  rpmdb
r29541@debianetch:~/ltib-mpc8349itx-20061019$ cp rootfs/boot/uImage /tftpboot
r29541@debianetch:~/ltib-mpc8349itx-20061019$ cp rootfs.ext2.gz.uboot /tftpboot
r29541@debianetch:~/ltib-mpc8349itx-20061019$

```

Figure 10. Copying Kernel and File System


```

r29541@debianetch: ~
File Edit View Terminal Tabs Help
ifconfig
eth0      Link encap:Ethernet  Hwaddr 08:00:3E:03:01:10
          inet addr:192.168.1.10 Bcast:192.168.1.255 Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:15 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1380 (1.3 KiB)  TX bytes:0 (0.0 B)
          Base address:0x4000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

~ # ls /usr/bin
[
[
[[
awk
basename
bunzip2
bzip2
cal
chvt
clear
cmp
crontab
cut
deallocate
hello
nc
tee
wc
dirname
hellocpp
od
telnet
wget
du
hexdump
openvt
test
which
dumpleases
hostid
passwd
tftp
who
env
id
patch
time
whoami
expr
install
readlink
top
xargs
fdformat
killall
realpath
tr
yes
find
last
renice
traceroute
free
logger
reset
tty
ftpget
logname
sort
uniq
ftpput
md5sum
strings
unzip
head
mkfifo
tail
uptime
~ # /usr/bin/hello
hello world
loop count = 0
loop count = 1
loop count = 2
loop count = 3
loop count = 4
loop count = 5
loop count = 6
loop count = 7
loop count = 8
loop count = 9
hello this is the end
~ #

```

Figure 13. Modified System In Operation

6 Network File System (NFS) on the PC

The preceding section describes how to download the kernel and the file system into the board using tftp protocol. This section describes how to download the kernel using tftp with the file system remaining on the PC. The board can then access the kernel remotely using the NFS protocol. NFS allows the MPC8349E-MITX-GP board to access a root file system on a remote PC through an Ethernet connection. This setup can be useful if you want to add/remove files to the file system during the debugging phase of a project. You can simply add/delete files on the remote PC and they are instantly available to your embedded system. The first step is to ensure that your PC has an NFS server installed. There are many ways to install NFS servers on Linux machines. On the Debian system used during the writing of this app note, the following procedure was used to install NFS (all commands entered when the user is logged in as root):

1. Enter the following commands into a terminal window:

```
apt-get update
```

```
apt-get install portmap
apt-get install nfs-common
apt-get install nfs-kernel-server
```

These commands download and install an NFS server and support software.

2. Edit the `/etc/exports` file on the PC and add the following text:

```
/nfs/root1 192.168.1.*(rw,no_root_squash)
```

This step configures the NFS server to accept requests from IP addresses `192.168.1.*` for files in the `/nfs/root1` directory and its subdirectories. Therefore, the file system must be located in this directory.

3. Enter the following commands to start the NFS server:

```
/etc/init.d/portmap start
/etc/init.d/nfs-kernel-server start
/etc/init.d/nfs-common start
```

4. When LTIB is first installed, a root file system is created in the `ltib-mpc8349itx-gp-20061024/rootfs` subdirectory. Rather than move this complete file system to `/nfs/root1`, create a symbolic link to it using the following commands:

```
mkdir /nfs
ln -s <directory where you installed ltib>/ltib-mpc8349itx-gp-20061024/rootfs
/nfs/root1
```

If you now enter the `ls /nfs/root1` command, you see the contents on the file system at `<directory where you installed ltib>/ltib-mpc8349itx-gp-20061024/rootfs`. Later, if you want to use a different file system, you can change the symbolic link to point to the new file system. This is much easier than copying the new file system, changing u-boot variables, and so on.

5. Enter the following commands in minicom:

```
setenv rootpath /nfs/root1
saveenv
```

This step sets an environment variable in u-boot, which sets the NFS access path.

6. Connect MPC8349E-MITX-GP to PC as described in [Section 3, “System Boot From Flash Memory and Ethernet Setup.”](#)
7. Start minicom as described in [Section 3, “System Boot From Flash Memory and Ethernet Setup.”](#)
8. Power up the MPC8349E-MITX-GP board.
u-boot starts and a prompt appears in minicom
9. At the prompt in minicom, enter the `run tftpnfsboot` command.
The display shows the board downloading the kernel (file `uImage`) and connecting to the file system on the PC. Alternatively, you can use the `run flashnfsboot` command, which boots the kernel stored in flash memory and uses the file system on the PC through NFS.
After successful download and boot, a standard Linux login prompt appears and you can log in and start to use standard Linux commands.
10. Log in using the username `root` and the password `root`.

You can now modify, add, or remove files in the file system and see the changes on the board. For example, trying adding a file on the PC side and accessing it on the board, as follows:

1. On a PC, open a terminal window and move to the file system directory:

```
cd /nfs/root1
```

2. Check the contents of the directory using the `ls` command.
3. Create a very simple text file containing the phrase `Hello World!`, as follows (see [Figure 14](#)):

```
echo Hello World! > textfile
```

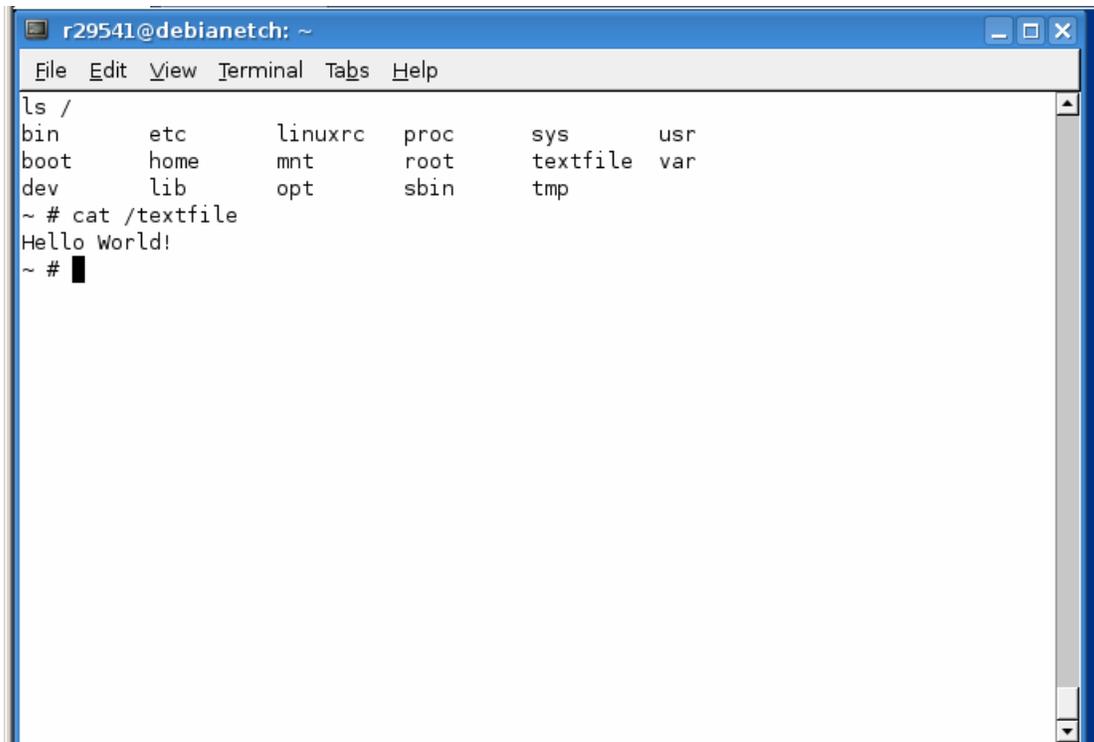
4. Use the `ls` command to verify that the new file is created with the name `textfile`.
5. In the minicom window, list all files in the file system root directory using the `ls` command.
6. Use the `cat /textfile` command to print the file contents. See [Figure 15](#).

```

r29541@debianetch: /nfs/root1
File Edit View Terminal Tabs Help
r29541@debianetch:~$ cd /nfs/root1
r29541@debianetch:/nfs/root1$ ls
bin  dev  home  linuxrc  opt  root  sys  usr
boot  etc  lib  mnt      proc  sbin  tmp  var
r29541@debianetch:/nfs/root1$ echo Hello World! > textfile
r29541@debianetch:/nfs/root1$ ls
bin  dev  home  linuxrc  opt  root  sys      tmp  var
boot  etc  lib  mnt      proc  sbin  textfile  usr
r29541@debianetch:/nfs/root1$

```

Figure 14. Creating a Text File On the PC

A terminal window titled 'r29541@debianetch: ~' with a menu bar containing 'File', 'Edit', 'View', 'Terminal', 'Tabs', and 'Help'. The terminal output shows a directory listing of the root directory, followed by a command to cat a file and its contents.

```
r29541@debianetch: ~  
File Edit View Terminal Tabs Help  
ls /  
bin      etc      linuxrc  proc     sys      usr  
boot    home    mnt      root     textfile var  
dev      lib      opt      sbin     tmp  
~ # cat /textfile  
Hello World!  
~ #
```

Figure 15. Accessing Text File Using Minicom

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 Hong Kong Ltd.
 Technical Information Center
 2 Dai King Street
 Tai Po Industrial Estate
 Tai Po, N.T., Hong Kong
 +800 2666 8080
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

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc., 2007. All rights reserved.

