Application Report
SWPA011 – May 2003

Building Linux for the Innovator Development Kit for OMAP™ Platform

Steven Kipisz

ABSTRACT

This report describes how to build and run Linux on the OMAP™ platform using the Texas Instruments OMAP1510 Innovator Development Kit. MontaVista provides a kernel source patch file in open source for Linux on the Innovator and maintains the code. This report describes the steps to obtain the kernel source and the tool chain source, build a tool chain from the GNU sources, and build and install the Linux kernel.

Contents

1 Introduction

Linux for the OMAP1510 Innovator can be built either by obtaining an ARM tool chain from MontaVista or building a tool chain from GNU source code. This application report describes where to obtain the source code and how to build the tool chain and the kernel.

2 Requirements

2.1 Hardware Requirements

- Innovator EVM ES 2.0 or greater
- Innovator Ethernet break-out board
- Serial null-modem cable or serial cable with a null-modem adapter
- JTAG
- PC running Microsoft Windows
- PC running Linux (one PC that dual boots both Windows and Linux will work)

2.2 Software Requirements

Code Composer Studio™ Integrated Development Environment (IDE) 2.1 or greater

Drivers for the required JTAG

Tera Term Pro 2.3 (http://hp.vector.co.jp/authors/VA002416/teraterm.html)

binutils-2.12.1 (ftp://ftp.gnu.org/gnu/binutils)

gcc-2.95.3 (ftp://ftp.gnu.org/gnu/gcc)

glibc-2.2.3 and glibc-linuxthreads-2.2.3 (ftp://ftp.gnu.org/gnu/glibc)

Linux kernel (ftp://ftp.kernel.org/pug/linux/kernel/v2.4)

ARM Linux patch to kernel (http://www.arm.linux.org.uk/developer/v2.4)

A Linux distribution for the PC; RedHat 7.2 was used for building the tool chain and kernel in this application report.

Innovator patch for ARM Linux (ftp://source.mvista.com/pub/omap/2.4)

- Bootloader for Innovator Linux (ftp://source.mvista.com/pub/omap/rrload)

3 Innovator EVM Configuration

There are four switches on the Innovator and four switches on the break-out board. The switches on the break-out board must be set as follows:

SW1-1	OFF

SW1-2	OFF
SW1-3	OFF
SW1-4	ON

The switches on the Innovator can be set for various situations. To boot from iBoot, set SW1-1 to ON and all other switches to OFF, as shown below:

SW2-4	OFF
SW2-3	OFF
SW2-2	OFF
SW2-1	ON

To boot from User Flash 0, set SW2-3 to ON and all others to OFF. To boot from User Flash 1, set SW2-3 and SW2-4 to ON and all others to OFF. See the *Innovator User's Guide* for more details about these switch settings.

4 Obtaining the Kernel and Bootloader

These steps, and the steps to build your tool chain, are performed on a PC running Linux.

The stock Linux kernel does not contain several necessary ARM-specific implementations. Therefore, you must download the stock kernel from a kernel web site, download an ARM-specific patch, and then download the Innovator-specific patch.

Download the following files from their respective web sites:

File	Download from…
linux-2.4.19.tar.gz	ftp://ftp.kernel.org/pub/linux/kernel/v2.4
patch-2.4.19-rmk7.bz2	http://www.arm.linux.org.uk/developer/v2.4
patch-2.4.19-rmk7-omap1.bz2	ftp://source.mvista.com/pub/omap/2.4
rrload-innovator-0301151120.tar.bz2	ftp://source.mvista.com/pub/omap/rrload

Be sure that the correct patch files are downloaded. The names follow the format:

<stock kernel>-<extension>-<extension>-…

so that patch-2.4.19-rmk7-omap1.bz2 requires the stock linux-2.4.19 kernel with the 2.4.19-rmk7 patch applied before it can be applied.

The last file is the boot loader source and is required to load and boot the Linux kernel.

Download the files to the home directory /home/<userid>. For brevity in this document, it is assumed that you are logged on as root and all files are downloaded to the /root directory. If you are not logged on as root, substitute /home/<userid> for /root.

Next, use bunzip2 to unzip the two patch files and the boot loader.

```
[root@localhost]# bunzip2 patch-2.4.19-rmk7.bz2
[root@localhost]# bunzip2 patch-2.4.19-rmk7-omap1.bz2
[root@localhost]# bunzip2 rrload-innovator-0301151120.tar.bz2
```

To keep the ARM Linux kernel separate from the build machine kernel, create an ARM-Linux subdirectory off of /usr/src. Go to the /usr/src/arm-linux directory and untar the kernel and the boot loader. Then apply the patch to patch the kernel.

```
[root@localhost]# cd /usr/src
[root@localhost]# mkdir arm-linux
[root@localhost]# cd arm-linux
[root@localhost]# tar –xzf /root/linux-2.4.19.tar.gz
[root@localhost]# tar –xf /root/rrload-innovator-0301151120.tar
[root@localhost]# cd linux-2.4.19
[root@localhost]# patch –p1 </root/patch-2.4.19-rmk7
[root@localhost]# patch –p1 </root/patch-2.4.19-rmk7-omap1
```

Before building the tools, you must set up some files in the kernel source tree. The kernel must be configured for Innovator, and a version file must exist in the include path. While still in the /usr/src/arm-linux/linux-2.4.19 directory perform the following:

```
[root@localhost]# make ARCH=arm innovator_config
[root@localhost]# make ARCH=arm oldconfig
[root@localhost]# make ARCH=arm include/linux/version.h
```

After the second make command, a message appears prompting you to include a workaround for missed FPGA interrupts. Type Y and press Enter to continue. The message is shown below.

```
*
* TI OMAP Implementations
*
 TI Innovator/OMAP1510 (CONFIG_OMAP_INNOVATOR [Y/n/?]
   Include  workaround for missed FPGA interrupts (CONFIG_INNOVATOR_MISSED_IRQS)
[N/y/?] (NEW)
```

5 Building the Tool Chain

Download the following files from their respective web sites:

File	Download from…
binutils-2.12.1.tar.bz2	ftp://ftp.gnu.org/gnu/binutils
gcc-2.95.3.tar.gz	ftp://ftp.gnu.org/gnu/gcc
glibc-2.2.3.tar.gz	ftp://ftp.gnu.org/gnu/glibc
glibc-linuxthreads-2.2.3.tar.gz	ftp://ftp.gnu.org/gnu/glibc

Again, the following steps assume that you are logged on as root and have downloaded the files to /root. Substitute /root with /home/<userid> if the files were downloaded there.

5.1 Create the Build Directories

The tool chain must be built into directories separate from the source. The build directories must not reside within the source tree. The GNU tools have not been extensively tested where the source and object directories are the same.

Under /usr/src/arm-linux, create a build directory and then individual directories for each tool within the build directory.

```
[root@localhost]# cd /usr/src/arm-linux
[root@localhost]# mkdir build
[root@localhost]# cd build
[root@localhost]# mkdir binutils-2.12.1
[root@localhost]# mkdir gcc-2.95.3
[root@localhost]# mkdir glibc-2.2.3
```

5.2 Build binutils

1. Change the directory to /root and use bunzip2 to unzip the binary utilities. Change the directory to /usr/src/arm-linux and untar the file with the tar command. The subdirectory /usr/src/arm-linux/binutils-2.12.1 is then created and contains the source tree.

```
[root@localhost]# cd /root
[root@localhost]# bunzip2 binutils-2.12.1.tar.bz2
[root@localhost]# cd /usr/src/arm-linux
[root@localhost]# tar –xf /root/binutils-2.12.1.tar
```

2. Chose the prefix for the new tool chain. This is a directory where the tool chain resides. This application report uses /opt/arm-linux for the new tool chain. The directory is created by the installation of the tools.

3. Go to the /usr/src/arm-linux/build/binutils-2.12.1 directory to configure and then build the binary utilities with the commands shown below. Make sure the configure command is entered all on one line (in this document it appears as two lines).

```
[root@localhost]# cd /usr/src/arm-linux/build/binutils-2.12.1
[root@localhost]# ../../binutils-2.12.1/configure –-target=arm-linux
 –-prefix=/opt/arm-linux
                                                    .
                                                    .
[root@localhost]# make
                                                    .
                                                    .
[root@localhost]# make install
```

4. The new binary utilities can be found in /opt/arm-linux/bin.

5. Add the path to the binary utilities into your current path. If you are using the bash shell, use the export command.

```
[root@localhost]# export PATH=/opt/arm-linux/bin:$PATH
```

5.3 Build Bootstrap gcc

The gcc built here is just enough of the compiler without libraries so that the kernel or the GNU C libraries can be built.

1. Make sure that /opt/arm-linux/bin is in the current path. See Section 5.2, *Build binutils*, step 5.

2. To build gcc, the directories /opt/arm-linux/arm-linux/include/asm and /opt/arm-linux/arm-linux/include/linux must point to the corresponding kernel directories. Create the soft link from the development tools to the Innovator linux kernel with the following commands:

```
[root@localhost]# cd /opt/arm-linux/arm-linux
[root@localhost]# mkdir include
[root@localhost]# cd include
[root@localhost]# ln –s /usr/src/arm-linux/linux-2.4.19/include/asm asm
[root@localhost]# ln –s /usr/src/arm-linux/linux-2.4.19/include/linux linux
```

3. Change the directory to /usr/src/arm-linux and unzip and untar the gcc-2.95.3.tar.gz file with the tar command.

```
[root@localhost]# cd /usr/src/arm-linux
[root@localhost]# tar –xzf /root/gcc-2.95.3.tar.gz
```

4. A configuration file must be modified so that the compiler is built without libc includes. Edit the /usr/src/arm-linux/gcc-2.95.3/gcc/config/arm/t-linux file and append –Dinhibit_libc –D__gthr_posix_h to the line starting with TARGET_LIBGCC2_CFLAGS =. The top of the file is shown below, after appending the flags.

```
# Just for these, we omit the frame pointer since it makes such a big
# difference.  It is then pointless adding debugging.
TARGET_LIBGCC2_CFLAGS = -fomit-frame-pointer -fPIC –Dinhibit_libc –D__gthr_posix_h
LIBGCC2_DEBUG_CFLAGS = -g0
                                    .
                                    .
```

5. Go to the /usr/src/arm-linux/build/gcc-2.95.3 directory to configure and build the GNU C compiler with the commands below. Be sure the configure command is entered all on one line (in this document it appears as two lines).

```
[root@localhost]# cd /usr/src/arm-linux/build/gcc-2.95.3
[root@localhost]# ../../gcc-2.95.3/configure –-target=arm-linux –-prefix=/opt/arm-linux
 --disable-threads –-disable-shared –-enable-languages=c
                                    .
                                    .
[root@localhost]# make
                                    .
                                    .
[root@localhost]# make install
```

6. If the error ‘configure: error: installation or configuration problem: C compiler cannot create executables’ appears, it is perfectly acceptable. The configure script for the compiler was built with older tools and the C compiler is built with newer tools that try to build more than just the C compiler.

7. The new gcc compiler can be found in /opt/arm-linux/bin.

5.4 Build glibc

1. Make sure /opt/arm-linux/bin is in the current path. See Section 5.2, *Build binutils*, step 5.

2. Change the directory to /usr/src/arm-linux and unzip and untar the glibc-2.2.3.tar.gz file with the tar command. Then change to the newly created glibc-2.2.3 directory and unzip and untar the glibc-linuxthreads-2.2.3.tar.gz file.

```
[root@localhost]# cd /usr/src/arm-linux
[root@localhost]# tar –xzf /root/glibc-2.2.3.tar.gz
[root@localhost]# cd glibc-2.2.3
[root@localhost]# tar –xzf /root/glibc-linuxthreads-2.2.3.tar.gz
```

3. Go to the /usr/src/arm-linux/build/glibc-2.2.3 directory to configure and then make the GNU C libraries with the commands shown below. Remember to enter the configure command all on one line.

```
 [root@localhost]# cd /usr/src/arm-linux/build/glibc-2.2.3
[root@localhost]# ../../glibc-2.2.3/configure arm-linux –-build=i386-pc-linux-gnu
  --prefix=/opt/arm-linux/arm-linux –-enable-add-ons=linuxthreads
                                    .
                                    .
[root@localhost]# make
                                    .
                                    .
[root@localhost]# make install
```

4. Creating the libraries takes awhile. It took approximately 21 minutes on an 800-MHz Pentium III notebook computer.

5. The new GNU C library can be found in /opt/arm-linux/arm-linux.

5.5 Build gcc With Threads and Additional Languages

1. Edit the /usr/src/arm-linux/gcc-2.95.3/gcc/config/arm/t-linux file and remove the append shown in Section 5.3, *Build Bootstrap gcc*, step 5. Specifically, remove –Dinhibit_libc –D__gthr_posix_h so that the top of the file appears as follows:

```
# Just for these, we omit the frame pointer since it makes such a big
# difference.  It is then pointless adding debugging.
TARGET_LIBGCC2_CFLAGS = -fomit-frame-pointer -fPIC
LIBGCC2_DEBUG_CFLAGS = -g0
```

2. Edit the /usr/src/arm-linux/gcc-2.95.3/libchill/basicio.c file and add a #define PATH_MAX 4095 just before the #ifndef PATH_MAX test. A section of the file is shown below.

```
                        …
#include <stdlib.h>

#include “fileio.h”

#define PATH_MAX 4095
#ifndef PATH_MAX
#ifdef _POSIX_PATH_MAX
#define PATH_MAX _POSIX_PATH_MAX
#else
                        …
```

3. Go to the /usr/src/arm-linux/build/gcc-2.95.3 directory and remove all files from the earlier build by using rm –rf *. Configure, build, and install gcc with threads and additional languages using the steps below.

```
 [root@localhost]# cd /usr/src/arm-linux/build/gcc-2.95.3
[root@localhost]# rm –rf *
[root@localhost]# ../../gcc-2.95.3/configure –-target=arm-linux –-prefix=/opt/arm-linux
  --host=i386-pc-linux-gnu –-enable-languages=c,c++
                                    .
                                    .
[root@localhost]# make
                                    .
                                    .
[root@localhost]# make install
```

4. The complete set of GNU compilation tools can be found in /opt/arm-linux/bin.

6 Building the Linux Kernel

All Linux kernel configuration and compilations are in the /usr/src/arm-linux/linux-2.4.19 directory. Go to that directory before performing any of the following steps. Also, the path to the ARM-Linux tool chain must be in the path. Make sure that /opt/arm-linux/bin is in the current path (see Section 5.2, *Build binutils*, step 5.)

6.1 Kernel Configuration

To use the Innovator on a network and have it obtain an IP address via BOOTP, leave the configuration alone. To have it obtain an IP address via DHCP, the default configuration must be changed before compiling the kernel. The steps are:

- On the command line, type: make ARCH=arm menuconfig
- Using the down arrow key, scroll down to Networking Options → and press Enter.
- Scroll down to IP: DHCP support and press the space bar to select it.
- Scroll down to IP: BOOTP support and press the space bar to deselect it.
- Press the right arrow key to exit this menu.

NOTE: The command line passed into the kernel that describes boot parameters must be configured into the kernel here. Usually, the boot loader passes in the command. At the time of this writing, however, the boot loader does not. To set the command, scroll up to General setup → and press Enter. Scroll down to Default kernel command string: and press Enter. Enter any kernel command parameters and then press Enter. See Section 9, *Configuring rrload*, steps 6 and 7, for more information on kernel command parameters. Also, if an IP address is obtained at boot time via DHCP, BOOTP, or RARP, the command line must also include IP=DHCP, IP=BOOT, or IP=RARP.

- Press the right arrow key to exit the Main menu.
- Be sure Yes is highlighted, and then press Enter when prompted to save the new configuration.

6.2 Kernel Compilation

From the command line, enter the following commands. The export statement is needed by mkimage so that it selects the ARM-Linux tools.

```
[root@localhost]# make ARCH=arm CROSS_COMPILE=arm-linux- dep
[root@localhost]# make ARCH=arm CROSS_COMPILE=arm-linux- clean
[root@localhost]# make ARCH=arm CROSS_COMPILE=arm-linux- zImage
[root@localhost]# export DSPLINUX_ARCH=TI925DC_EVM
[root@localhost]# /usr/src/arm-linux/rrload/mkimage --LAddr  10C08000 --EAddr 10C08000
arch/arm/boot/compressed/vmlinux vmlinux.rr
[root@localhost]# make ARCH=arm CROSS_COMPILE=arm-linux- modules
[root@localhost]# make ARCH=arm CROSS_COMPILE=arm-linux- INSTALL_MOD_PATH=<root fs>
modules_install
```

The last command installs the modules into a subdirectory that contains the Innovator root file system. Ignore any depmod errors that occur. These are caused by the Makefile attempting to run the host depmod command on the ARM-based modules.

The file vmlinux.rr is the kernel in a format that is understood by the rrload boot loader that is built next.

7 Building the Bootloader

Change the directory to /usr/src/arm-linux/rrload and edit the bsp.defs file so that it reflects the correct name of the tool chain prefix. The last line of the file changes from

export PREFIX=arm_920t_le-

to

export PREFIX=arm-linux-

Now, from the command line type make. This builds the setup program and the bootloader, rrload. Copy these two files and the vmlinuz.rr file to the Windows-based PC so that they can be installed on the Innovator.

8 Installing the Bootloader

Linux requires that a Linux-aware bootloader reside in User Flash 0 or 1. This loader is currently rrload. This differs from the TI bootloader or iBoot that resides in boot flash. Before loading the Linux operating system, the rrload bootloader must be installed on the Innovator.

8.1 Installing rrload Using JTAG

This method requires Code Composer Studio™ IDE 2.x and a compatible JTAG device connected to a PC running the Windows operating system.

1. Power off the Innovator.
2. Connect the JTAG to the Innovator and to the PC.
3. Connect the serial cable to the Innovator and to the PC.
4. Configure the Innovator to boot from User Flash 0 by flipping switch SW2-3 to the ON position and switches SW2-1, SW2-2, and SW2-4 to the OFF position.
5. Start up Tera Term and configure it for 115K bits per second, 8 bits, no parity, 1 stop bit, and no flow control.
6. Power on the Innovator.
7. Start up Code Composer Studio™ IDE.
8. From the Parallel Debug Manager window, select Open and the first CPU (the ARM).
9. From the File menu item, select Load Program and load the setup program.
10. Either press the F5 key, or select Debug and then select Run. This initializes the system.
11. Wait 5 to 10 seconds and then halt the CPU by selecting Debug, and then Halt.
12. Load the rrload program.
13. Select Debug, and then Run.

14. The serial terminal on the PC now shows the following:

```
+-------------------------------------+
|           Welcome to the            |
|         DSPLinux Bootloader         |
|                                     |
|     rrload v4.8 by RidgeRun, Inc    |
|     platform: OMAP1510_INNOVATOR    |
+-------------------------------------+

MAIN MENU
---------
  1. Load [comp] from I/O port...
  2. Store RAM [comp] to Flash...
  3. View/Edit Params...
  4. Boot Kernel/filesystem (boot_auto)
  5. CmdLine Mode
  6. Dump memory

  r. Run Default Boot Cmd
  E. Erase [comp] from Flash...

  Which?
```

15. The bootloader is now loaded and running in SDRAM. To transfer it to flash, first erase the flash by selecting E, and then press Enter. The serial terminal now shows the following:

```
Erase [comp] from Flash
-----------------------
  0. --to MAIN--
  1. Erase [BootLoader]
  2. Erase [Params]
  3. Erase [Kernel]
  4. Erase [FileSys]
  5. Erase ALL

  Which?
```

16. Enter 1 and press Enter to erase the bootloader.
17. Select Yes at the prompt, “Are you sure?”
18. Enter 2 and press Enter to erase the parameters.
19. Select Yes at the prompt, “Are you sure?”
20. Enter 3 and press Enter to erase the kernel area.
21. Select Yes at the prompt, “Are you sure?”
22. Select 0 to go back to the Main menu.
23. Select 2 to store a RAM component to flash. The serial terminal shows the following:

```
Store RAM [comp] to Flash
-------------------------
  0. --to MAIN--
  1. Store [Bootloader]
  2. Store [Params]
  3. Store [Kernel]
  4. Store [FileSystem]

  Which?
```

24. Select 1 to store the bootloader from RAM to user flash. The serial terminal shows a series of periods until the store is complete.

The bootloader is now stored in User Flash 0. Exit Code Composer Studio™ IDE and restart the Innovator. The serial terminal shows the same boot menu as in Section 8.1, *Installing rrload Using JTAG,* step 14.

9 Configuring rrload

The Linux bootloader requires some initial setup to properly load the kernel. This setup information is stored in a user flash sector called Params. All commands are entered via the serial terminal.

1. Power up the Innovator. If the boot_auto parameter was previously set, quickly press Enter to avoid autobooting the kernel.
2. Erase any existing parameters by selecting E from the Main menu, and then selecting 2 from the Erase menu. Answer Yes to the “Are you sure?” question.
3. Select 0 to go back to the Main menu.
4. Select 3 to View/Edit Parameters. The following appears on the serial terminal

```
View/Edit Params
----------------
  0. --to MAIN--
  1. ui mode [cmd|menu]  : menu
  2. default boot cmd    :
  3. I/O load format     : rrbin
  4. I/O load port       : serial
  5. console port        : serial

 DSPlinux Settings
  6. Kernel cmdline      :
  7. Device MAC          : 00:e0:be:ef:fa:ce

 rrload TFTP Settings
  8. Server IP    (tftp) : 192.168.1.15
  9. Server MAC   (tftp) : 00:01:02:3A:66:25
  a. Device IP    (tftp) : 192.168.1.125
  b. Kernel Path  (tftp) : linux.rr
  c. FileSys Path (tftp) : romdisk.img.rr
  d. OSD Img Path (tftp) : logo.img.rr

 OnScreenDisplay (OSD) Logo Settings
  e. Enable OSD          :

  Edit Which?
```

5. To automatically jump to the kernel when the system is powered on, select 2 and then type boot_auto for the new value.

6. The kernel command line must be set to override the default value stored in the kernel. This command line contains the path to the root file system and a command that informs the kernel not to use an initial ramdisk. Select 6 and then enter the kernel command line. The following example shows a command line that assumes the root file system is mounted via NFS from the workstation at 128.247.77.208, exporting a root file system /armtarget.

```
new val = noinitrd root=/dev/nfs rw nfsroot=128.247.77.208:/armtarget,nolock
```

7. If the root file system is located on the user flash, a possible command line is:

```
new val = noinitrd root=/dev/mtdblock3
```

8. Also, console=ttyS0,115200n8 can be appended so that console messages and login are on the serial terminal. If there is no telnet server configured for logging into the Innovator, then the console must be on the serial port, as shown below.

```
new val = noinitrd root=/dev/nfs rw nfsroot=128.247.77.208:/armtarget,nolock
console=ttyS0,115200n8
```

9. Store the parameters to the flash by selecting 0 to go back to the Main menu, then 2 to go to the Store menu, and then 2 to store the parameters to flash.

NOTE: At the time of this writing, the kernel command line was not passed into the kernel properly. It is designed to work in future versions.

10 Installing the Kernel

All commands are entered through Tera Term on the Windows-based PC.

1. Power up the Innovator. If the boot_auto parameter was previously set, quickly press Enter to avoid autobooting the kernel.
2. From the rrload main menu, select 1 to load a component from the I/O port. The following appears on the serial terminal:

```
Load [comp] from -serial- I/O
---------------------------------
  0. --to MAIN—
  1. Load [BootLoader]
  3. Load [Kernel]
  4. Load [FileSystem]
  5. Load [OSD Logo] (part of params)

  Which?
```

3. Select 3 to load the kernel.
4. From Tera Term, select File and then Send File.
5. Check the Option Binary checkbox at the bottom of the dialog.
6. Browse and select the Linux kernel file, vmlinuz.rr.
7. The serial terminal shows a series of periods as the kernel is downloaded.
8. After the kernel has downloaded, select 0 to return to the Main menu.
9. Now either boot the new kernel or save it to flash. For information on how to boot the kernel, see Section 11, *Booting the Kernel*.
10. To save the kernel to flash, erase the old contents of flash by selecting E to erase a component, and then 3 to erase the old kernel. Answer Yes to the “Are you sure?” prompt. Select 0 to return to the Main menu. Select 2 to store a component from RAM to flash, and then select 3 to store the kernel to flash.
11. After the new kernel has been stored in flash, it is time to boot the kernel.

11 Booting the Kernel

Power on the Innovator. If the boot_auto parameter was selected, the new kernel boots up with messages displayed on the LCD. If the boot_auto parameter was not selected, the boot loader displays the Main menu on the serial terminal.

Select 4 to boot the new kernel. Depending on which device was configured for a console, either the host Tera Term session or the Innovator LCD displays boot up messages.

12 Root File System

Although it is beyond the scope of this application report to explain how to build a root file system, it may be desirable to boot the Innovator with one and see it run. Several ARM-based Linux root file systems can be found on the Internet. One such file system is maintained by The Wearable Group, at Carnegie Mellon University. This file system can be used to test the kernel on the Innovator. A set of steps to boot to the root file system over NFS follows. It is assumed that the Linux workstation has loop support built into the kernel and NFS server support (see the kernel documentation accompanying the distribution).

NOTE: This root file system is for testing purposes only. It is based on a different C-runtime library than the one built in this paper. Programs built with the tool chain used in this application report probably will not work with this root file system.

1. Start a browser and go to http://www.wearablegroup.org/software/ramdisk.
2. Download the file ramdisk_ks.gz. The following instructions assume that the file is downloaded to /root.
3. This file is typically used as a RAM disk but is not used as one in this example; instead, it is expanded and copied to an NFS mounted file system. To use it this way, it must first be unzipped with gunzip, mounted using the loop file system, and then the contents copied to an NFS exported directory. In the example below, it is assumed that the exported directory is /data/target.

```
[root@localhost]# cd /root
[root@localhost]# gunzip ramdisk_ks.gz
[root@localhost]# mount –o ro,loop=/dev/loop0 ramdisk_ks /mnt
[root@localhost]# mkdir /data/target
[root@localhost]# cp –R /mnt/* /data/target
```

4. Be sure that /data/target is exported by NFS and that NFS is running on the Linux workstation. See the Linux distribution documentation for this information.
5. Build the Innovator-Linux kernel so that the console is on ttyS0 at 115200 baud. Also be sure that the nfsroot points to the Linux workstation with the NFS exported root file system. A sample command follows. The IP address of the Linux workstation replaces x.x.x.x.

```
noinitrd console=ttyS0,115200n8 root=/dev/nfs rw nfsroot=x.x.x.x:/data/target,nolock
```

6. From Tera Term, load and boot the kernel. rrload uses 115200 for a baud rate, whereas the kernel console uses 9600. After booting the kernel, reconfigure Tera Term for 9600 baud to be able to log in.
7. Because the root file system assumes PCMCIA support, some error messages regarding PCMCIA drivers not being found are displayed during boot up. This is normal, because the Innovator does not support PCMCIA.
8. If you are using Tera Term, a Linux prompt to log in appears. Log in as root, with no password.
9. Commands such as cp, tar, ls, and so on, can be used to demonstrate that the kernel and the root file system work.

Summary

This application report explained the steps to download, configure, build, and install an ARM-Linux tool chain. It then went on to discuss how to download, configure, build, and install a kernel and boot loader on the Innovator platform.

References

1. Innovator Development Kit for the Texas Instruments OMAP1510 Platform Deluxe Model User's Guide, Document Number 715-0003-203.
2. The Wearable Group, Carnegie Mellon University, http://www.wearablegroup.org/software/ramdisk
3. The ARM Linux Project, http://www.arm.linux.org.uk
4. MontaVista Software, http://www.mvista.com, ftp://source.mvista.com

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third–party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265