

Introduction

This application note discusses Linux operating systems and walks you through the steps of rebuilding the dynamic reconfiguration of Excalibur™ devices with Linux demonstration from the source code to a running board. The demonstration exhibits the capability of Excalibur devices to perform the following operations:

- run embedded Linux
- dynamically load and unload device drivers
- reconfigure the FPGA portion of the Excalibur device without interrupting the operating system or network connectivity



For more information on the Excalibur Linux reconfiguration demonstration, see *AN 278: Excalibur Dynamic Reconfigurations with Linux Demonstration*.

Linux Overview

This section gives an overview of operating systems and Linux.

Operating Systems

The software requirements of an embedded application are often simple enough that there is no need for an actual operating system—one encompassing while loop is all that is needed. However, some of today's embedded applications are quite complex, requiring software to manage a large number of tasks and hardware devices simultaneously. In these situations, the advantages of employing an operating system quickly become apparent. Operating systems typically provide the following five basic services:

- Memory management
- Task scheduling
- Network connectivity
- File system
- Scripting ability

Linux

Linux is quickly becoming one of the more popular operating systems for embedded systems. It has the following desirable characteristics that are helping to promote its popularity:

- *Open source.* The source code of the Linux kernel is available for free under the GNU Public License (GPL). It can be freely modified to optimize or add functionality required for a specific application.
- *Architecture independence / Portability.* Linux is, generally speaking, not dependent on processor architecture. It can be ported to new platforms with relative ease.
- *Extensive networking support.* Linux is designed as a network-centric operating system. A transmission control protocol/internet protocol (TCP/IP) stack is even included with the operating system.
- *Familiarity.* Linux is an environment with which many designers are comfortable, meaning productivity is achieved very quickly.

Embedded Linux versus Desktop Linux

Embedded Linux runs the very same kernel that is run on desktop computers. The difference is that the resources available in an embedded environment are typically much less than those available on a desktop computer. A strength of Linux is that the kernel is relatively small. Hence, when running Linux on an embedded system, only the features, drivers, and utilities that are needed must be installed. The kernel remains essentially the same.

Getting Started

This section involves the following steps:

- [Hardware & Software Requirements](#)
- [Before You Begin](#)
- [Install the Demonstration Files](#)
- [Build the Kernel](#)
- [Build the Applications](#)

Hardware & Software Requirements

To rebuild and run the EPXA1 Linux reconfiguration demonstration, you require the following hardware and software:

- Root access to a Linux PC with Ethernet connectivity.
- GNUPro software development tools (version arm9-020528 or higher)
- Quartus® II software (version 2.2 or higher)
- MontaVista Linux 2.1 Professional Edition or Preview Kit
- MontaVista EPXA1 Linux Support Package (LSP)
- EPXA1 Development Kit

ARMBoot

ARMBoot is the bootloader included with the MontaVista EPXA1 LSP. It is an embedded OS bootloader that is available for free under the GNU Public License (GPL). This version has been ported for use with the EPXA1 Development Board.

MontaVista Linux Professional Edition

MontaVista Linux Professional Edition is a leading cross-development environment for embedded Linux. It is the Linux package on which this reconfiguration demonstration is based. Along with a Linux kernel, Professional Edition provides a large number of libraries, utilities, and drivers. The EPXA1 LSP, available with the Professional Edition package, provides the additional board-specific support for the EPXA1 Development Board.

MontaVista Linux Preview Kit

The Preview Kit is a free version of MontaVista Linux that can be used to test-drive the MontaVista development environment without having to purchase Professional Edition. It provides much of the functionality of Professional Edition, and can be used to rebuild the EPXA1 reconfiguration demonstration.

Before You Begin

Before you begin you must perform the following actions:

- Familiarize yourself with the EPXA1 Linux reconfiguration demonstration
- Erase the flash memory
- Complete the steps in the *MontaVista Quick Start Guide*

Erase the Flash Memory

Before you begin, you must erase all the flash memory on the EPXA1 board to ensure that no residual flash data interferes with the demonstration. To erase the flash memory on the EPXA1 development board, power the board, connect the ByteBlaster II download cable, and type the following commands:

```
exc_flash_programmer -a -e 0↵
```

```
exc_flash_programmer -a -e 1↵
```

MontaVista Quick Start Guide

Before you begin, you must complete the steps described in the *Cross-Development Quick Start Guide*. This document is included in MontaVista Linux Professional Edition and in the MontaVista Linux Preview Kit.

Install the Demonstration Files

The demonstration consists of two zipped tar files: one for the source code, including the kernel, and one for the file system that ultimately resides in flash memory on the target device.

To download the files from the Altera FTP site and install them on your Linux PC, perform the following steps:

1. Connect to <ftp.altera.com> with any FTP client.
2. Login as `anonymous`.
3. Enter your email address as the password.
4. Change to the **outgoing** directory.
5. Set the transfer type to **binary**.
6. Get the following files:
 - **epxa1db.tz**
 - **filesystem.tz**
7. Create a project directory.



The project directory must be accessible by the Linux PC that will run the build.

8. Extract the project source files into this new project directory, by typing the following commands:

```
cd <project directory>↵  
tar -zxvf epxa1db.tz↵
```



To install the file system, you must have root privileges, because the file system includes device nodes in the `/dev` directory.

9. Extract the file system files, by typing the following commands:

```
su
<provide root password>↵
cd <install directory>/epxa1db↵
tar -zxvf filesystem.tz↵
```

Build the Kernel

To perform a kernel build from the kernel source tree, perform the following steps:

- Using the MontaVista Graphical Interface
- Using the Command-line Interface
- Add a Bootable Header to the Kernel Image
- Program the Kernel into Flash
- Boot the New Kernel

Using the MontaVista Graphical Interface

The MontaVista toolset provides a tool for configuring and building the Linux kernel called **targetconf**. Before you run **targetconf**, you must update the **.targetconfrc** file in your home directory. If you have never run **targetconf** before, you must run it once to create **~/.targetconfrc**

Once this file is created, open it in a text editor and between the lines:

```
<PROJECT_DIR_LIST>
```

and

```
</PROJECT_DIR_LIST>
```

Insert the following line:

```
<PROJECT_DIR><install directory>/epxa1db/</PROJECT_DIR>
```



<install directory> is the directory in which you extracted the demonstration source files.

Type **targetconf** to start the **targetconf** GUI. To build the kernel, perform the following steps:

1. Select the **epxa1db** project and click **Open Project**.

2. Choose **Open Config** (File menu), select `epxa1db_demo.cfg`, and click **OK**.
3. (Optional) click **Configure Kernel** to view the configuration for this build.
4. Click **Build Target** to start the build.

Depending on your computer speed, building the kernel can take between 10 and 40 minutes. After build completion, a kernel image, **zImage**, is available in the `<install directory>/kernel/arch/arm/boot` directory.

Using the Command-line Interface

If you wish not to use the MontaVista GUI to build the kernel, it can also be built from a Linux command prompt. To build the kernel from a Linux prompt, type the following commands:

```
cd <install directory>/epxa1db/kernel↵
```

```
make dep zImage↵
```

As with the MontaVista `targetconf` build, a kernel image, **zImage** is created in the `<install directory>/kernel/arch/arm/boot` directory.

Add a Bootable Header to the Kernel Image

For the bootloader, ARMboot, to recognize the kernel image that you've created, a header must be added to the front of the image so that ARMboot can use it. To add this header, use the **mkimage** utility, which is included with the MontaVista toolset. To add an ARMboot header to the kernel image, in one line type the following command:

```
/opt/hardhat/host/bin/mkimage -A arm -O linux -T  
kernel -C none -a 0xa00000 -e 0xa00000 -n linux  
-d <install directory>/kernel/arch/arm/boot  
/zImage zImage.bin↵
```



The exact location of **mkimage** may differ, if you did not install the MontaVista tools in the default location

Program the Kernel into Flash

Now that you have a bootable kernel image, you can burn it into the flash memory on the EPXA1 Development Board so that it can be booted. But first, you must convert it into a **.hex** file that the flash programmer can read. Additionally, because you have already programmed ARMboot into the bottom of flash memory at 0x0, you should give the kernel an offset so as not to overwrite ARMboot. For this example, you will use an offset of 0x30000. Both of these tasks can be accomplished with the **arm-elf-objcopy** command.

To convert the bootable kernel image to **.hex** file format, in one line type the following command:

```
arm-elf-objcopy -I binary -O ihex -change-  
address=0x30000 zImage.bin zImage.hex
```

Now, with the ByteBlaster™ II download cable connected, program the **.hex** file into flash memory, by typing the following host command:

```
exc_flash_programmer -f -p -v zImage.hex
```



If the **exc_flash_programmer** utility fails on your Linux machine, ensure that the ByteBlaster II driver is installed on the machine. Refer to the document *Quartus II Installation & Licensing for UNIX and Linux Workstations* for instructions on installing this driver.

Boot the New Kernel

Now that the new kernel is in flash, you can boot it, but you must first change a parameter in ARMboot that specifies where to get the kernel. To boot the new kernel, perform the following steps:

1. Reset the EPXA1 development board.
2. In the terminal, stop ARMboot by hitting any key.
3. Change the bootcmd parameter in ARMboot with the following commands

```
setenv bootcmd bootm 0x40030000  
saveenv
```

4. Boot Linux by typing the command:

```
bootd
```

The new kernel now boots on the board from flash memory, but the root file system is still mounted over NFS.

Copy Target File System to Flash Memory

The next step is to transfer the target's root file system from an network file system (NFS) share into the target's flash memory, making it completely autonomous.

The root file system that you want to put into flash memory is not the same file system that you are now connecting to over NFS. You are currently mounting the default MontaVista file system stored on the host, which is far too big to put into flash memory. The file system for this demonstration is much smaller and contains all of the files that you need to run the demonstration.

To copy the file system to flash memory you must make the file system stored on the host accessible via NFS. This is similar to how the default MontaVista file system was made available in the *MontaVista Quick Start Guide*. To copy the file system to flash memory, perform the following steps:

1. Login as root
2. Add the following line in **/etc/exports** (this line may vary if you did not install the file system in the default location):

```
<install directory>/epxa1db/filesystem*  
    (rw,no_root_squash,no_all_squash) ↵
```

3. Reset the NFS daemon with the following command:

```
/etc/rc.d/init.d/nfs restart↵
```

4. Log onto the EPXA1 Development Board as root using the terminal. Once logged on, type the following commands:

```
mkdir /tmp/fs↵  
mkdir /tmp/flash↵  
mount -t nfs -o rsize=1024,wsiz=1024  
    137.57.193.134:<install directory>/epxa1db/filesystem  
    /tmp/fs↵  
mount -t jffs2 /dev/mtdblock0 /tmp/flash  
rm -a /tmp/flash/*↵
```



Ensure that the `rm` command is never run anytime the currently mounted root file system is in flash memory, otherwise the system will crash.

```
cp -a /tmp/fs/* /tmp/flash
```



This command will take several minutes to complete.



All of the above commands must be entered as one line.

The root file system is now located in the flash memory, although Linux is still using the one mounted over NFS on the host. To change this you must change a parameter in ARMboot. To mount the root file system stored in flash, perform the following steps:

1. Reset the EPXA1 Development Board.
2. In the terminal, stop ARMboot by hitting any key.
3. Change the `bootargs` parameter in ARMboot with the following commands:

```
setenv bootargs mem=31M console=ttyUA0,57600
root=/dev/mtdblock0 ip=<ip addr>:<host
ip>:<default gw>:<net mask>:epxa1:eth0:off
```



The `ip=` parameter is not necessary if you are using dynamic host configuration protocol (DHCP).



The above command must be entered as one line.

4. Type `bootd` to boot Linux.

The EPXA1 Development board is now an autonomous Linux platform that contains both its own kernel and root file system on-board. You may want to take a snapshot of the flash memory for easy re-programming of the board at a later time. To save the flash memory contents into `.hex` files, run the following host commands with the ByteBlaster II cable connected:

```
exc_flash_programmer -r -e 0 demo0.hex
exc_flash_programmer -r -e 1 demo1.hex
```

Now at any time, the entire system can be restored with the following commands:

```
exc_flash_programmer -f -p -v -e 0 demo0.hex
exc_flash_programmer -f -p -v -e 1 demo1.hex
```

You can now run the EPXA1 reconfiguration demonstration as described in *AN 278: EPXA1 Linux Reconfiguration Demonstration*.

Build the Applications

The demonstration includes the source code for the applications, which you can also rebuild. To rebuild the applications, run the following commands on the host:

```
cd <install-dir>/epxa1db/applications↵  
make↵  
make install↵
```

When building the applications, you may encounter the following error:

```
Error: Line '' at line number 3 in data file <file>.txt  
is not valid
```

If you encounter this error, open the file specified in the error message and remove the empty line at the end of the file. This is a software version issue and depends on the version of software that you are using.

When building the applications, some files in the target file system are updated, therefore the target file system must be re-copied into flash memory, before you try to download the applications.

Summary

The EPXA1 Linux reconfiguration demonstration shows the capabilities and flexibility of the Excalibur devices. You can use the demonstration as a basis for developing embedded systems that require a high-performance processor running Linux and the configurability of an FPGA.



101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
<http://www.altera.com>
Applications Hotline:
(800) 800-EPLD
Literature Services:
lit_req@altera.com

Copyright © 2003 Altera Corporation. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, mask work rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services. All rights reserved.



I.S. EN ISO 9001

For more information on Excalibur devices, see the *Excalibur Device Hardware Reference Manual*.