AMD Alchemy™ Processors
Building a Root File System for Linux®
Incorporating Memory Technology Devices

1.0 Scope

This document outlines a step-by-step process for building and deploying a Flash-based root file system for Linux® on an AMD Alchemy™ processor-based development board, using an approach that incorporates Memory Technology Devices (MTDs) with the JFFS2 file system.

Note: This document describes creating a root file system on NOR Flash memory devices, and does not apply to NAND Flash devices.

1.1 Journaling Flash File System

JFFS2 is the second generation of the Journaling Flash File System (JFFS). This file system provides a crash-safe and powerdown-safe Linux file system for use with Flash memory devices. The home page for the JFFS project is located at http://developer.axis.com/software/jffs.

1.2 Memory Technology Device

The MTD subsystem provides a generic Linux driver for a wide range of memory devices, including Flash memory devices. This driver creates an abstracted device used by JFFS2 to interface to the actual Flash memory hardware. The home page for the MTD project is located at http://www.linux-mtd.infradead.org.

2.0 Building the Root File System

Before being deployed to an AMD Alchemy platform, the file system must first be built on an x86 Linux host PC. The primary concern when building a Flash-based root file system is often the size of the image. The file system must be designed so that it fits within the available space of the Flash memory, with enough extra space to accommodate any runtime-created files, such as temporary or log files.

Before moving on to the deployment phase, the root file system should be tested using either an NFS-based or hard disk-based configuration to ensure its functionality.

2.1 OpenEmbedded Root File System

A root file system generated by the OpenEmbedded (OE) toolkit is often suitable for deployment onto a Flash device without any major modifications. The *AMD Alchemy™ Au1200™ OE Development Kit* is configured to build a BusyBox-based system by default, and it is not recommended that this be changed unless absolutely necessary.

To ensure no space is wasted, include only those packages necessary for operation of the system in the platform's OE meta-package before building the image. To follow the steps for deployment, use the generated root file system located in the travis-dev-kit/tmp/rootfs directory if using the directory structure outlined in the *AMD Alchemy™ DBAu1200™ Developent Board OpenEmbedded Quick Start Guide* (Publication ID 40351).

2.2 MontaVista Root File System

The MontaVista Pro toolkit includes a very large pre-built root file system. This file system must be reduced using an iterative process until its footprint fits into the available Flash memory. It is highly recommended that the file system be deployed using NFS during the reduction process, as any changes can be tested immediately.

2.2.1 BusyBox

The first step in reducing the footprint of the MontaVista root file system is to switch from the default Bash shell and coreutils to a BusyBox-based system. BusyBox is a single multi-call binary that replaces and replicates the basic and most-used functionality of the shell and many of the necessary utilities for running a Linux system. The BusyBox binary is approximately 1 MB in size, and can be built to rely on no external libraries except for the C library, libc.

The BusyBox source code and documentation can be obtained at http://www.busybox.net. While BusyBox advocates the use of uClibc, a smaller version of the C library, it is entirely possible, and often desirable, to build BusyBox to use the pre-compiled C library contained within the original MontaVista root file system.

2.2.2 Removing Unnecessary Packages

The most time consuming portion of reducing the MontaVista footprint is removing the packages that are not necessary for the target system. This can be done in a top-down manner, by removing a subset of the unnecessary packages and then testing the file system, or in a bottom-up manner by starting with only BusyBox and adding additional necessary packages one-at-a-time and resolving dependency errors that arise. The method chosen depends largely on developer preference.

In either case, it is helpful to use the ldd tool (ensure a MIPS-based tool is used (e.g., mips_nfp_le-ldd)) to determine the library dependencies of a dynamically-linked binary executable.

2.2.3 Removing Debug Information

The MontaVista root file system binaries and libraries are built with debugging symbols included. While this is beneficial when debugging software, it can also greatly increase the size of the executable. Once the file system has been reduced to its necessary functionality, the debugging information can be removed from the included binaries, further reducing the footprint of the file system. This is done using the strip tool, as shown in the following example, which removes the debugging symbols from the C library:

```
# mips_nfp_le-strip -d libc-2.3.2.so
```

3.0 Deploying a JFFS2 File System

Once the file system has been built on the x86 host PC and tested on the target platform, the file system must then be converted into a JFFS2 image and written to Flash.

3.1 Add MTD Support to the Linux Kernel

The Linux kernel must be configured to support MTD and JFFS2. Since the root file system will be mounted using JFFS2, all of the kernel options must be enabled as built-ins, not as modules. For a Linux 2.4 kernel on a AMD Alchemy™ DBAu1550™ development board, enable the following options:

```
Memory Technology Devices (MTD) --->
       <*> Memory Technology Device (MTD) Support
       <*> MTD Partitioning Support
       RAM/ROM/Flash chip drivers --->
              <*> Detect flash chips by Common Flash Interface (CFI) probe
              <*> Support for AMD/Fujitsu Flash Chips
       Mapping drivers for chip access --->
              <*> Db1550 MTD Support
File systems --->
       <*> Journalling Flash File System v2 (JFFS2) support
```

For a Linux 2.6 kernel on a AMD Alchemy™ DBAu1200™ development board, enable the following options:

```
Device Drivers --->
       Memory Technology Devices (MTD) --->
              <*> Memory Technology Device (MTD) support
              [*] MTD partitioning support
              RAM/ROM/Flash chip drivers  --->
                     <*> Detect flash chips by Common Flash Interface (CFI) probe
                     <*> Support for AMD/Fujitsu flash chips
              Mapping drivers for chip access --->
                     <*> AMD Alchemy Pb1xxx/Db1xxx/RDK MTD support
File systems --->
       Miscellaneous filesystems --->
              <*> Journalling Flash File System v2 (JFFS2) support
```

The MTD mappings for the AMD Alchemy development boards can be found within the Linux source tree in drivers/mtd/maps/. For the examples above, the actual filenames are db1550-flash.c and alchemy-flash.c, respectively.

These MTD mappings divide the Flash memory into separate regions of differing size as shown in Figure 3-1. These regions become separate MTD partitions when Linux is run. In the case of the AMD Alchemy development boards, the regions are arranged as follows.

3 MB	mtdblock2	Kernel image (raw binary)
1 MB	mtdblock1	YAMON (raw binary)
Size -4 MB	mtdblock0	Root file system (JFFS2)

Figure 3-1. MTD Partition Mapping

Therefore, on a DBAu1200 development board, which has 64 MB of Flash, the lower 60 MB of Flash is available for use as the root file system, while the upper 4 MB is reserved for the YAMON boot loader and a 3 MB binary kernel image. Note, however, that YAMON uses the upper 256 kB of Flash to store its environment variables, so this space is not made accessible by MTD.

3.2 Obtain MTD Tools

In order to build a JFFS2 image, the mkfs.jffs2 tool must be built from the MTD toolkit. Obtain the MTD source directly from the project's CVS server:

```
# cvs -d :pserver:anoncvs@cvs.infradead.org:/home/cvs login
      password: anoncvs
# cvs -d :pserver:anoncvs@cvs.infradead.org:/home/cvs co mtd
```

Alternately, obtain daily snapshots from ftp://ftp.uk.linux.org/pub/people/dwmw2/mtd/cvs/.

Once the MTD source has been obtained, build the mkfs.jffs2 utility (the actual drivers have already been incorporated into the AMD Alchemy Linux kernel source tree, and need not be built):

```
# cd mtd/util
# make mkfs.jffs2
```

Once the binary is built, it may be copied to any location within the PATH.

```
# cp mkfs.jffs2 /usr/sbin
```

3.2.1 Prepare Root File System

Before booting the root file system, ensure the /etc/fstab file is configured to use an MTD-based root file system. Create or replace the existing entry for the root mount point (/) in the /etc/fstab file with an entry similar to the following:

```
/dev/mtdblock0   /   jffs2   defaults,sync,noatime   1 1
```

Unless using devfs or udev, MTD device nodes must also be added to the /dev/ directory of the root file system. This can be done using the MAKEDEV script included in the MontaVista root file system:

```
# cd rootfs/dev
# ./MAKEDEV mtd
# ./MAKEDEV mtdblock
```

3.3 Create JFFS2 Image

Once the mkfs.jffs2 tool has been built, it can be used to build a JFFS2 binary image out of the root file system created in Section 2.0 on page 1. The following example creates an image for a DBAu1200 development board from a file system located in the directory “rootfs”.

```
# mkfs.jffs2 -p0x3C00000 -drootfs -orootfs.jffs2 -l -q -v
```

The image created, rootfs.jffs2, will be 60 MB (0x3C00000 bytes), which is the size of the available MTD partition as discussed in Section 3.1 beginning on page 2. This size declaration should be adjusted according to the available space on the target platform.

Note: The file system will be built to be 60 MB in size regardless of the actual size of the contents of the rootfs directory. The extra space is left as available writeable space.

3.4 Download JFFS2 Image to Flash

The root file system can be programmed into Flash using any standard Flash programming method, but for this example YAMON is used. Before YAMON can be used to program the image, it must be converted to S-record format using the objcopy tool. To convert the DBAu1200 development board image, created in Section 3.3, the command uses the following parameters:

```
# objcopy -O srec -I binary --adjust-vma=0xBC000000 rootfs.jffs2 rootfs.srec
```

Either the host or cross-compiler objcopy tool may be used, as the image is in raw binary format. The --adjust-vma argument is used to relocate the S-record to the address of the beginning of the MTD partition. For the default mappings of all AMD Alchemy development boards, the beginning of the first MTD partition is the beginning of Flash. The address for the beginning of Flash for any AMD Alchemy development board is found in the corresponding address mapping on the hardware CD-ROM shipped with the board.

Place the resultant rootfs.srec file into the TFTP root directory of the TFTP server YAMON has access to. Then use YAMON to erase the target area and program the image:

```
YAMON> erase 0xBC000000 0x3C00000
YAMON> load /rootfs.srec
```

The first argument to the erase command is the starting location of the erase operation, which is the starting location of the MTD partition, which is equivalent to the value given in the --adjust-vma argument given to objcopy. The second argument to the erase command is the total size of the image, which is the value given in the -p argument to mkfs.jffs2. It is important to double-check these values to ensure that the boot vector (0xBFC00000), which contains the YAMON boot-loader, is not erased.

3.5 Booting Linux with MTD Root File System

For the Linux kernel to recognize MTD as its root file system, the kernel arguments “root=/dev/mtdblock0 rw rootfstype=jffs2” must be specified. This can either be specified as part of the default kernel command line in the kernel configuration, or directly from the YAMON command line as follows:

```
YAMON> load /vmlinux.srec
YAMON> go . root=/dev/mtdblock0 rw rootfstype=jffs2
```

© 2006 Advanced Micro Devices, Inc. All rights reserved.
The contents of this document are provided in connection with Advanced Micro Devices, Inc. (“AMD”) products. AMD makes no representations or warranties with respect to the accuracy or completeness of the contents of this publication and reserves the right to make changes to specifications and product descriptions at any time without notice. No license, whether express, implied, arising by estoppel or otherwise, to any intellectual property rights is granted by this publication. Except as set forth in AMD’s Standard Terms and Conditions of Sale, AMD assumes no liability whatsoever, and disclaims any express or implied warranty, relating to its products including, but not limited to, the implied warranty of merchantability, fitness for a particular purpose, or infringement of any intellectual property right.

AMD’s products are not designed, intended, authorized or warranted for use as components in systems intended for surgical implant into the body, or in other applications intended to support or sustain life, or in any other application in which the failure of AMD’s product could create a situation where personal injury, death, or severe property or environmental damage may occur. AMD reserves the right to discontinue or make changes to its products at any time without notice.

www.amd.com

TRADEMARKS
AMD, the AMD Arrow logo, and combinations thereof, and AMD Alchemy, Au1200, DBAu1200, and DBAu1550 are trademarks of Advanced Micro Devices, Inc.

Linux is a registered trademark of Linus Torvalds.

Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

One AMD Place • P.O. Box 3453 • Sunnyvale, CA 94088-3453 USA • Tel: 408-749-4000 or 800-538-8450 • TWX: 910-339-9280 • TELEX: 34-6306