Filesystem in NOR or NAND

From Texas Instruments Wiki

After kernel booted, it tries to mount a file system. Using Linux on DaVinci, there are several options where this file system can come from. Options are

- Harddisk (or CompactFlash card)
- MMC/SD card
- NFS (http://en.wikipedia.org/wiki/Network_file_system) (see DVEVM Getting Started Guide, sprue66c.pdf, section 4.3.4 (http://focus.ti.com /general/docs/techdocsabstract.tsp?abstractName=sprue66c))
- Ramdisk (avoid this old model)
- Initramfs, usually a very small part of the kernel image
- (Flash) file system in NOR or NAND

During development and/or debugging phases, it is useful to have a temporary file system (e.g. initramfs, NFS, or removable media) not stored on the target itself. For production ready devices or if file system doesn't change any more, the file system has to be persistent on the target. Besides file system on hard disk, (flash) file system in NOR flash or NAND flash are two options. This article is about how to put file systems in NOR and/or NAND and how to handle them with Linux.

We first assume that system is booted using one of the temporary (development) file systems (initramfs, NFS, or removable media) which kernel mounts as initial file system and which then can be used to create file system in NOR or NAND flash. Once this is done, in second step, support for this temporary (development) file system can be disabled (in bootloader and in kernel) and kernel directly can mount file system in NOR or NAND flash.

Contents

Bootloader

Using DaVinci DM6446 DVEVM, NOR or NAND can't be accessed at the same time (http://linux.omap.com/pipermail/davinci-linux-open-source /2006-October/001151.html) (both connected to chip select CS2 selectable using J4). You have to configure if system should be able to access NOR *or* NAND flash. Seeing both at the same time isn't possible. Therefore, using DM6446 DVEVM, you have to configure the board to start from NOR to deal with NOR flash file system or from NAND to deal with NAND flash file system. This includes having a boot loader (e.g. U-Boot) in the flash you want to use.

The DM355 EVM normally uses NAND booting, so you don't need to configure it specially unless you want to get the boot loader from an SD card.

DM6446 DVEVM configuration

For **NOR**:

- set jumper J4 to FLASH
- set S3-1 to ON and S3-2 to OFF (NOR boot)
- set S3-3 to ON (16-bit CS2 bus width)

For **NAND**:

- set jumper J4 to NAND
- set S3-1 and S3-2 to OFF (NAND boot)
- set S3-3 to OFF (8-bit CS2 bus width)

Having set these configurations, make sure you can completely boot in this configuration. E.g. having a boot loader in NOR or NAND flash (e.g. U-Boot) [1] [2], being able to start the kernel (e.g. using TFTP) and mounting a temporary file system (e.g. Ramdisk or via NFS).

Kernel configuration

Depending on NOR or NAND, you have to set different kernel configuration options.

IDE

Important: the dm6446 (and dm357) can't use the external memory interface (EMIF) for NOR/NAND and, at the same time, access the IDE disk (or CF storage). One common configuration uses Flash (NOR or NAND) just to boot, and uses the IDE interface for everything else. If you're not using Flash at run time, don't bother to configure it into your kernel.

To disable IDE, just make sure that configuration option

```
Device drivers -> ATA/ATAPI/MFM/RLL support
```

is *disabled*.

NOR

Make sure the following kernel options are set (via *make menuconfig*):

```
CONFIG_MTD=y
CONFIG_MTD_PARTITIONS=y

CONFIG_MTD_CHAR=y
CONFIG_MTD_BLKDEVS=y
CONFIG_MTD_BLOCK=y

CONFIG_MTD_CFI=y
CONFIG_MTD_GEN_PROBE=y
CONFIG_MTD_MAP_BANK_WIDTH_1=y
CONFIG_MTD_MAP_BANK_WIDTH_2=y
CONFIG_MTD_MAP_BANK_WIDTH_4=y
CONFIG_MTD_CFI_I1=y
CONFIG_MTD_CFI_I2=y
CONFIG_MTD_CFI_AMDSTD=y
CONFIG_MTD_CFI_UTIL=y

CONFIG_MTD_PHYSMAP=y
CONFIG_MTD_PHYSMAP_START=0x8000000
CONFIG_MTD_PHYSMAP_LEN=0
CONFIG_MTD_PHYSMAP_BANKWIDTH=2
```

Note: DaVinci supports only *16MByte* NOR. So even if your NOR chip is physically larger (the original EVM boards used 32 MB parts, newer ones use 16 MB parts), you will only get 16MByte. While booting, this results in message

```
Reducing visibility of 32768KiB chip to 16384KiB
```

NAND

Make sure the following kernel options are set (via *make menuconfig*):

```
CONFIG_MTD=y
CONFIG_MTD_PARTITIONS=y

CONFIG_MTD_CHAR=y
CONFIG_MTD_BLKDEVS=y
CONFIG_MTD_BLOCK=y

CONFIG_MTD_NAND=y
CONFIG_MTD_NAND_VERIFY_WRITE=y
CONFIG_MTD_NAND_IDS=y
CONFIG_MTD_NAND_DAVINCI=y
```

Note: while the DaVinci NAND driver is now in mainline kernels, 4-bit ECC support (for DM355 and so on) is not merged yet.

File systems

Because of the special physical behavior of flash, special file systems for these devices are developed. See the Linux-MTD (http://www.linux-mtd.infradead.org/) website for the most current information about mainline filesystems like JFFS2 and UBIFS, and associated flash-related userspace tools.

Journaling flash file system 2 (JFFS2) (http://sourceware.org/jffs2/) is part of the standard kernel and mostly used for NOR devices.

While JFFS2 can be used for NAND devices as well, Yet Another File System 2 (YAFFS2) (http://www.yaffs.net/) was specifically designed for NAND devices. See jffs vs. yaffs comparison (http://www.yaffs.net/comparison-yaffs-vs-jffs) for more infos. YAFFS2 is currently not part of the standard kernel though. Using recent git kernel, you can easily patch it to use YAFFS2.

Recent mainline kernels (2.6.27+) add support for UBIFS (http://www.linux-mtd.infradead.org/doc/ubifs.html), which can be thought of as a JFFS2 successor with significant improvements in scalability and NAND support.

JFFS2

As JFFS2 is part of the standard git kernel, only thing to configure kernel to be able to use JFFS2 is to enable in make menuconfig:

```
CONFIG_JFFS2_FS=y
CONFIG_JFFS2_FS_DEBUG=0
CONFIG_JFFS2_FS_WRITEBUFFER=y
CONFIG_JFFS2_ZLIB=y
CONFIG_JFFS2_RTIME=y
```

Note: Regarding JFFS2 see Create a JFFS2 Target Image and Put JFFS2 Image to Flash articles as well.

UBIFS

UBIFS is also part of the standard GIT kernel. To configure it, you need to support UBI (in the MTD section) as well as UBIFS (in the filesystem section):

```
CONFIG_MTD_UBI=y
CONFIG_UBIFS_FS=y
```

You will also want the userspace tools for UBI and UBIFS.

YAFFS2

As mentioned above, YAFFS2 is not part of the (git) kernel. Follow the download instruction on YAFFS2 download page (http://www.yaffs.net /node/346) how to use cvs do get recent YAFFS2 sources.

Once you downloaded the sources, patch the kernel with YAFFS2 easily using *patch-ker.sh* part of the downloaded YAFFS2 sources.

Note: The installation instructions for YAFFS2 on YAFFS2 page (http://www.yaffs.net/howto-incorporate-yaffs) are *outdated*. Once downloaded the source using cvs, follow the instructions given in file *README-linux-patch* how to use patch script *patch-ker.sh*.

Then, configure your kernel:

```
CONFIG_YAFFS_FS=y
CONFIG_YAFFS_YAFFS1=y
CONFIG_YAFFS_YAFFS2=y
CONFIG_YAFFS_AUTO_YAFFS2=y
CONFIG_YAFFS_SHORT_NAMES_IN_RAM=y
```

File system preparation

MTD handling

Now, you should be able to boot a kernel with NAND or NOR and the file system you selected.

Then, to be able to use kernels MTD (MemoryTechnologyDevices) (http://www.linux-mtd.infradead.org/) subsystem from user space, you need some device nodes in your file system:

```
crw-rw-rw- 1 root root 90,0 Jul 25 2002 /dev/mtd0
brw-rw-rw- 1 root disk 31,0 Jul 25 2002 /dev/mtdblock0
...
crw-rw-rw- 1 root root 90,6 Jul 25 2002 /dev/mtd3
brw-rw-rw- 1 root disk 31,3 Jul 25 2002 /dev/mtdblock3
```

If you don't have these yet, use *mknod (http://www.linuxcommand.org/man_pages/mknod1.html)* command to create them.

If you have more than one partition in your flash device (depends on NOR and NAND kernel configuration), you have to create additional device nodes mtd**x** and mtdblock**x** for each additional partition. Note that the minor for mtdx nodes has to be an even number.

Use /proc/mtd to get information on how many partitions are currently configured by the kernels flash driver. For NAND on DVEVM you might get e.g.:

```
> cat /proc/mtd
dev:    size    erasesize   name
mtd0: 03b00000 00004000 "User Space"
```

This has only one partition and a mtd0/mtdblock0 device node pair is sufficient.

For NOR and NAND on DVEVM you will get e.g.:

```
> cat /proc/mtd
dev:    size    erasesize   name
mtd0: 00040000 00010000 "bootloader"
mtd1: 00010000 00010000 "params"
mtd2: 00200000 00010000 "kernel"
mtd3: 00db0000 00010000 "filesystem"
```

This has four partitions, but you most probably don't want to touch bootloader, params and kernel using MTD subsystem (cause you handle it with U-Boot). Thus, a mtd3/mtdblock3 device node would be sufficient here.

Note: You always need a pair mtdx/mtdblockx of device nodes to access your flash file system. The mtdx is used to access the raw flash device, the mtdblockx is used to access the disk/block established in the raw flash.

Then, you need MTD User modules (http://www.linux-mtd.infradead.org/doc/general.html) to erase and prepare your NOR or NAND file system. E.g. using NAND on DVEVM:

```
> flash_eraseall -j /dev/mtd0
Erasing 16 Kibyte @ 3afc000 -- 100 % complete. Cleanmarker written at 3afc000.
```

E.g. using NOR on DVEVM:

```
> flash_eraseall -j /dev/mtd3
Erasing 64 Kibyte @ da0000 -- 99 % complete. Cleanmarker written at da0000.
```

Note: Repeat *flash_eraseall -j /dev/mtd**x*** on all devices you want to use.

Note: The *flash_eraseall -j* is the same for NOR or NAND devices if you use *JFFS2*.

Note: Use the *-j* option only if you will use JFFS2 for this partition. For YAFFS2 omit the -j option and simply use *flash_eraseall /dev/mtdx*.

Note: To get MTD User modules, use git snapshot (*snapshot* link) at right side of MTD User modules git repository (http://git.infradead.org/?p=mtd-utils.git;a=summary).

Mount

After all this preparation, you should be able to mount your new created file system in flash. Assuming you have a directory /mnt/nand or /mnt/nor to mount the new (empty) file system to, do the following.

For NOR e.g.:

```
mount -t jffs2 /dev/mtdblock3 /mnt/nor
```

For NAND e.g.:

```
mount -t yaffs2 /dev/mtdblock0 /mnt/nand
```

Note: Replace **0** or **3** above with the block number you want to use.

Note: Above assumes that you want to use JFFS2 for NOR and YAFFS2 for NAND. You can use each file system for NAND *and* NOR as well, thus JFFS2 is recommended for NOR and YAFFS2 for NAND.

Inital file system

If you don't want to boot from the file system established above in NOR or NAND, everything above is sufficient. Maybe you want to add the mount command to a startup script so it is done automatically at boot up.

But if you want to use NOR or NAND file system established above as an initial file system the kernel mounts at startup, you have to ensure some additional points.

Content of file system

You need a complete boot file system in NOR or NAND file system. This comprises e.g. device nodes, init program and scripts etc.

Easiest way is copy content of your development files system (Ramdisk, NFS) to flash file system. Cause you have to ensure that device nodes, hard and soft links etc are not destroyed while file system is copied, using tar is a good way. E.g. tar your target file system, make the tar file containing your file system available on the target (e.g. via NFS) and then mount flash file system:

```
▷ mount -t jffs2 /dev/mtdblock3 /mnt/nor
▷ nfs mount to e.g. /mnt/nfs
▷ cd /mnt/nor
▷ tar -xvf /mnt/nfs/ramdisk_content.tar
```

Command line options

Now that root file system is in flash, you can disable temporary development file system in kernel command line options:

For NOR use e.g.:

```
... root=/dev/mtdblock3 rootfstype=jffs2 ...
```

For NAND use e.g.:

```
... root=/dev/mtdblock0 rootfstype=yaffs2 ...
```

A complete command line option to set in the UBoot prompt would then be e.g.:

```
setenv bootargs console=ttyS0,115200n8 root=/dev/mtdblock0 rootfstype=yaffs2 rw mem=32M ip=off
```

Note: If you want to reduce kernel size, don't forget to remove options for temporary file systems (e.g. NFS or Ramdisk) from your kernel once mounting persistent file system in NOR or NAND.

Engage in the TI E2E Community Ask questions, share knowledge, explore ideas and help solve problems with fellow engineers	*For technical support please post your questions at http://e2e.ti.com. Please post only comments about the article **Filesystem in NOR or NAND** here.*

Links

| | Amplifiers & Linear (http://www.ti.com/lsds/ti/analog/amplifier_and_linear.page) Audio (http://www.ti.com/lsds/ti/analog/audio/audio_overview.page) Broadband RF/IF & Digital Radio (http://www.ti.com | DLP & MEMS (http://www.ti.com/lsds/ti/analog/mems/mems.page) High-Reliability (http://www.ti.com/lsds/ti/analog/high_reliability.page) Interface | Processors (http://www.ti.com/lsds/ti/dsp/embedded_processor.page)
 ■ ARM Processors (http://www.ti.com/lsds/ti/dsp/arm.page)
 ■ Digital Signal Processors (DSP) (http://www.ti.com | Switches & Multiplexers (http://www.ti.com/lsds/ti/analog/switches_and_multiplexers.page) Temperature Sensors & Control ICs (http://www.ti.com/lsds/ti/analog/temperature_sensor.page) Wireless Connectivity (http://focus.ti.com/wireless/docs/wirelessoverview.tsp?familyId=2003& |

/lsds/ti/analog/rfif.page) Clocks & Timers (http://www.ti.com/lsds/ti /analog/clocksandtimers /clocks_and_timers.page) Data Converters (http://www.ti.com/lsds/ti /analog/dataconverters /data_converter.page)

(http://www.ti.com /lsds/ti/analog /interface /interface.page) Logic (http://www.ti.com /lsds/ti/logic /home_overview.page) Power Management (http://www.ti.com /lsds/ti/analog /powermanagement /power_portal.page)

/lsds/ti/dsp/home.page)
- Microcontrollers (MCU) (http://www.ti.com/lsds/ti /microcontroller /home.page)
- OMAP Applications Processors (http://www.ti.com/lsds/ti /omap-applications- processors/the-omap- experience.page)

sectionId=646&tabId=2735)

Retrieved from "http://processors.wiki.ti.com/index.php?title=Filesystem_in_NOR_or_NAND&oldid=15115"

Categories: Linux | File system | Flash

- This page was last modified on 23 August 2009, at 21:59.
- This page has been accessed 5,456 times.
- Content is available under Creative Commons Attribution-ShareAlike unless otherwise noted.