Put JFFS2 Image to Flash

Often times it is useful to write the file system for your device into a permanent storage media such as NOR or NAND flash. Below you will find steps on how to write a JFFS2 file system into NOR or NAND flash on the DVEVM board.

Contents

Prerequisites

- A file system image saved as either a tarball file or a JFFS2 image.
  - NOTE: If you are writing the file system using U-Boot the file system image must be a JFFS2 image. The tarball image can only be written using an already booting kernel with some intermediate file system (e.g. ram disk, NFS or hard drive file system)
- A terminal application for communicating with the DVEVM. i.e. minicom or hyperterminal.
- A copy of the u-boot bootloader running on the target.
  - If you do not have U-Boot running on the target please see the RBL, UBL and host program article for how to get the u-boot bootloader installed on your board first.

Writing a JFFS2 Image with U-Boot

The primary benefit of writing the JFFS2 image using U-Boot is that you do not need an intermediate file system in order to boot Linux to access the Flash device. This only works with JFFS2 image.

Additional Requirements

- A tftp or NFS server to download the file system image from. In this example tftp is used.
- The start location for your file system partition.
  - For information on determining the start location please see the get FFS partition configuration page.
- An ethernet connection to the target board.

Common Steps

The following steps are the same whether you are using NOR or NAND Flash. After following these steps you may then go to the NOR or NAND section for additional instructions based on your Flash type.

- Transfer the JFFS2 image to the target board

For boards using dhcp use:

```
dvevm# setenv bootfile rootfs.jffs2
dvevm# dhcp
```

For boards using a static ip use:

```
dvevm# tftp 0x80700000 rootfs.jffs2
```

You should see output similar to:

```
TFTP from server xxx.xxx.xxx.xxx; our IP address is xxx.xxx.xxx.xxx
Filename 'rootfs.jffs2'.
Load address: 0x80700000
Loading: #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
```

```
         ################################################################
         ################################################################
         ############################################################
done
Bytes transferred = 2617068 (27eeec hex)
```

Record the bytes transferred and the load address for later use. In this case the load address is 0x80700000 and the bytes transferred is 0x27eeec.

NOR Flash

The following steps should be followed when using NOR flash devices.

- Unprotect the Flash for writing. You will need to know:
  1. The start address of Flash. For this example the start address is

0x2000000. Refer to the Technical Reference documentation for your board to determine the Flash start address for your board.

  1. The offset to the beginning of the file system partition.
  2. The size of the file system image as reported above.

In this example the start of our flash is at 0x2000000, the offset to the beginning of the file system partition is 0x440000 and the size of the file system image is 0x27eeec. The command to enable writing would look like:

```
dvevm# protect off 0x2440000 +0x27eeec
```

You should see output similar to:

```
dvevm# protect off 0x2440000 +0x27eeec
Un-Protected 40 sectors
```

- Erase the flash where the file system image will be copied. In this example this is the flash from 0x2440000 (Our flash base plus offset to the file system partition) for a length of 0x27eeec. The command would look like:

```
dvevm# erase 0x2440000 +0x27eeec
```

You should see output similar to:

```
dvevm# erase 0x2440000 +0x27eeec
Erasing sector 68 ... done.
Erasing sector 69 ... done.
Erasing sector 70 ... done.
.
.
.
Erasing sector 106 ... done.
Erasing sector 107 ... done.
 Erase Operation Completed.
Erased 40 sectors
```

- Copy the file system image to Flash using the cp.b command. In this example we are copying from address 0x80700000 (from the transfer above) to the start of our flash file system partition the size of the file system image. The command would look like:

```
dvevm# cp.b 0x80700000 0x2440000 0x27eeec
```

You should see output similar to:

```
dvevm# cp.b 0x80700000 0x2440000 0x27eeec
Copy to Flash...\done
```

The file system image has now been written to Flash. You may proceed to the Setting Kernel to Boot using Flash section below for information on booting the JFFS2 file system.

NAND Flash

To load a JFFS2 filesystem image to NAND flash with u-Boot you must use a version of u-Boot which supports the ".trimffs" option to the "nand write" command. This command is supported in u-Boot-2011.09, but not in the version that ships with the TI SDK. For more information, see this TI E2E forum post: http://e2e.ti.com/support/dsp/sitara_arm174_microprocessors/f/416/p/140272/519195.aspx#519195

To enable the command make sure to add support in your board configuration in u-Boot. The board configuration is found at u-Boot-2011.09/include/configs/your_board.h. To add support for the command, add "#define CONFIG_CMD_NAND_TRIMFFS" directly after the line which reads "#define CONFIG_CMD_NAND."

Once the "nand write.trimffs" command is enabled, you can transfer your filesystem to RAM as shown above with TFTP, over UART via YMODEM with the "loady" command, or with a JTAG emulator such as the XDS560v2.

Once the filesystem image is stored in RAM, erase the NAND pages which will contain your filesystem:

```
dvevm# nand erase 0x600000 0x1FA00000
```

where 0x600000 is the NAND offset at which the filesystem is to be stored (should match your kernel MTD partition table), and 0x1FA00000 is either the size of the filesystem, or the remaining bytes available in NAND.

You may then copy the filesystem image to NAND with the following command:

```
dvevm# nand write.trimffs 0xC0700000 0x600000 ${file_size}
```

where 0xC0700000 is the RAM address where the image is currently stored, and 0x600000 is the NAND offset at which the filesystem is to be stored.

If you're running into problems, remember to consider your own environment when following the steps. For instance, if your jffs2 file system is large and doesn't seem to be loading, is it overwriting something else, say U-Boot and that is why your system is freezing? In this case you'll want to transfer the filesystem images in several chunks. You can split the image on your development Linux workstation with the "split" command.

The steps in this link have been verified to work on a dm6467t: http://processors.wiki.ti.com/index.php/DM6467_EVM_Installation#Writing_Kernel_and_Filesystem_Image_to_NAND_Flash Please remember that the addresses and filesizes are examples.

Writing Flash from Linux

It is possible to use the Linux MTD subsystem to write the JFFS2 file system. While this method requires the use of an intermediate file system it has the benefit of not requiring calculation of the start of the file system partition. This can example can be used with .tar.gz file or a JFFS2 image. Tarball is preferred way, though.

When using Linux, interaction with the Flash device is controlled through the MTD subsystem. The MTD subsystem exports two device nodes per Flash partition. These are:

1. /dev/mtd# - A Character device that is used to access the raw flash device.
2. /dev/mtdblock# - A block device used to access the disk/block established in the raw flash.

In the above device nodes the '#' sign represents the partition number. For example partition 0 would have the device nodes /dev/mtd0 and /dev/mtdblock0

Additional Requirements

- You need a kernel with JFFS2 and Flash support enabled.
  - For details on how to enable kernel JFFS2 support see the JFFS2 kernel configuration page.
  - For details on how to enable kernel flash support refer to the Flash configuration in the Kernel page.
- A root file system mounted from somewhere other than Flash. For this example we will use NFS.
  - For more information on setting up an NFS file system please refer to the "Exporting a Shared File System for Target Access" section of the Getting Started Guide for your board.
- A version of the MTD Utilities for the target board to be used in writing the file system image to Flash.

Common Steps

The following steps are the same whether you are using a JFFS2 image or a tarball image. After following these steps you may then go to the JFFS2 or tarball section for additional instructions based on your image type.

To write a JFFS2 image you will use the MTD utilities to erase and copy the file system image into the Flash. The utilities that will be used for both NOR and NAND are:

- flash_eraseall - Erases an entire MTD device, which in this case is a partition

The steps to write the JFFS2 image are

NOTE: The sample output below is for NOR Flash. You should see similar output when using NAND Flash.

- Determine the MTD device for the file system partition of your Flash device

```
target$ cat /proc/mtd
```

You should see output similar to:

```
target$ cat /proc/mtd
dev:    size   erasesize  name
mtd0: 00020000 00010000 "bootloader"
mtd1: 00020000 00010000 "params"
mtd2: 00400000 00010000 "kernel"
mtd3: 00bc0000 00010000 "filesystem"
```

In this example the file system partition of our flash device is /dev/mtd3.

- Erase the file system partition

```
target$ flash_eraseall -j /dev/mtd3
```

Here we use the -j option to tell the flash_eraseall command to format the device for jffs2. You should see output similar to:

```
target$ flash_eraseall -j /dev/mtd3
Erasing 64 Kibyte @ bb0000 -- 99 % complete. Cleanmarker written at bb0000.
```

Writing a JFFS2 Image

The following sections detail how to write the JFFS2 image into NOR and NAND flash. After following the steps for your device you may proceed to the Setting Kernel to Boot using Flash section below for information on booting the JFFS2 file system.

NOR Flash

To write the JFFS2 image to NOR Flash you will also need the following MTD utilities:

- flashcp - Copies the file system image to Flash

- Copy the JFFS2 file system image to Flash

```
target$ flashcp <image dir>/rootfs.jffs2 /dev/mtd3
```

NAND Flash

To write the JFFS2 image to NAND Flash you will also need the following MTD utilities:

- nandwrite - writes the file system image to Flash

- Copy the JFFS2 file system image to Flash

```
target$ nandwrite -p /dev/mtd3 <image dir>/rootfs.jffs2
```

Writing a tarball Image

The following section details how to write a .tar.gz file system image into NOR or NAND flash. After following these steps you may proceed to the Setting Kernel to Boot using Flash section below for information on booting the JFFS2 file system.

- Mount the Flash file system partition using the block device node.

```
target$ mkdir /mnt/flash
target$ mount -t jffs2 /dev/mtdblock3 /mnt/flash
```

The mount command uses the "-t jffs2" option to indicate that the device should be mounted as a jffs2 file system.

- Untar the contents of the tarball file system image to the Flash device

```
target$ cd /mnt/flash
target$ tar xzf <image dir>/rootfs.tar.gz
```

- Unmount the Flash file system partition

```
target$ cd /
target$ umount /mnt/flash
```

Setting Kernel to Boot Using JFFS2

This section covers setting the kernel bootargs in u-boot to boot from a Flash partition populated with a JFFS2 file system image. The following requirements must be met in order to boot the JFFS2 file system from a Flash partition:

Requirements

- A kernel with Flash support and JFFS2 support
  - For information on enabling Flash support in the TI kernel please refer to the Flash configuration in the Kernel page
  - For information on enabling JFFS2 support in the TI kernel please refer to the JFFS2 kernel configuration page
- A Flash device with a partition populated with a JFFS2 file system.

Command Line Parameters

In order to boot the JFFS2 file system in Flash you must add the following kernel parameters to the kernel boot arguments defined in the "bootargs" environment variable in U-Boot.

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

The above bootargs tell the kernel that the root file system is on MTD partition 3 and that the file system type is JFFS2.

You should now have a kernel which boots with a JFFS2 root file system in Flash.

{{

1. switchcategory:MultiCore=

- For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum
- For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum

Please post only comments related to the article **Put JFFS2 Image to Flash** here.

Keystone=

- For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum
- For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum

Please post only comments related to the article **Put JFFS2 Image to Flash** here.

C2000=*For technical support on the C2000 please post your questions on The C2000 Forum. Please post only comments about the article **Put JFFS2 Image to Flash** here.*

DaVinci=*For technical support on DaVincoplease post your questions on The DaVinci Forum. Please post only comments about the article **Put JFFS2 Image to Flash** here.*

MSP430=*For technical support on MSP430 please post your questions on The MSP430 Forum. Please post only comments about the article **Put JFFS2 Image to Flash** here.*

OMAP35x=*For technical support on OMAP please post your questions on The OMAP Forum. Please post only comments about the article **Put JFFS2 Image to Flash** here.*

OMAPL1=*For technical support on OMAP please post your questions on The OMAP Forum. Please post only comments about the article **Put JFFS2 Image to Flash** here.*

MAVRK=*For technical support on MAVRK please post your questions on The MAVRK Toolbox Forum. Please post only comments about the article **Put JFFS2 Image to Flash** here.*

*For technical s please post yo questions at http://e2e.ti.cor Please post on comments abo article **Put JFF Image to Flas*

}}

Links

Amplifiers & Linear
Audio
Broadband RF/IF & Digital Radio
Clocks & Timers
Data Converters
DLP & MEMS
High-Reliability
Interface
Logic
Power Management
Processors

- ARM Processors
- Digital Signal Processors (DSP)
- Microcontrollers (MCU)
- OMAP Applications Processors

Switches & Multiplexers
Temperature Sensors & Control ICs
Wireless Connectivity