Application Report
SPRAAA0A–December 2008

Booting DaVinci EVM From NAND Flash

Juan Gonzales

ABSTRACT

Currently, the DaVinci™ evaluation module (DVEVM) supports three boot modes: the DVEVM can boot from NOR (default), NAND, or universal asynchronous receiver/transmitter (UART). NOR Flash offers the advantages of one-byte random access and execute-in-place technology. NAND Flash is not as easy to work with since it requires Flash Translation Layer (FTL) software to make it accessible; however, due to its lower price, speed, and longer life span, many costumers want to design with NAND Flash instead or NOR Flash. This application report describes the process to follow for booting the DaVinci DVEVM from NAND Flash.

Flashwriter source code is not available. If you need to customize a Flash utility for your custom board, please see the DVFlasher utility covered in *Booting and Flashing via the DaVinci TMS320DM644x Serial Interface* (SPRAAI4).

Note: ubl_nand.bin, u-boot-nand.bin and flashwriter_nand.out type files (names may change slightly to reflect version information) are distributed as part of the official Digital Video Software Development Kit (DVSDK) software releases.

Contents

List of Figures

DaVinci, Code Composer Studio are trademarks of Texas Instruments.
Linux. is a trademark of Linux Torvalds in the U.S. and other countries.
Windows is a trademark of Microsoft Corporation in the United States and/or other countries.

1 Prerequisites

Software Required:

- ubl_nand.bin (14 KB first stage boot-loader)
- u-boot-567-nand.bin (second stage boot loader)
- flashwriter_nand.out CCS program
- Code Composer Studio™ version 3.2 or higher
- DaVinci EVM compatible emulator driver for Code Composer Studio
- Serial terminal application such as HyperTerminal, TeraTerm for Windows™ and Minicom or C-Kermit for Linux.™

Hardware Setup Required:

- Linux host workstation for building Linux kernel
- PC workstation for running Code Composer Studio
- IEEE Standard 1149.1-1990, IEEE Standard Test Access Port and Boundary-Scan Architecture (JTAG) Emulator connected to DaVinci EVM
- Connect included RS232 serial cable from *UART0* port of the EVM to either PC workstation or Linux host workstation

Additional Requirements:

- Code Composer Studio 3.2 has been configured and tested with JTAG emulator that works with the DaVinci EVM.
- Make sure you are familiar with the Getting Started Guide (GSG) - in particular Chapter 4.

2 Instructions

These instructions are broken down into four main sections that build on each other. Some may chose to follow only Section 1 and Section 2, at which point they are booting out of NAND and can configure the NAND boot loader to load desired kernels via TFTP from host, and the NFS mount file system from host; this may provide an adequate environment for development. Others may choose to complete all four sections and have not only their boot loader in NAND Flash, but also their desired Linux kernel and file system; thereby, having an environment that closely resembles their final product (no host system).

2.1 Configure DVEM Hardware for NAND Boot Mode

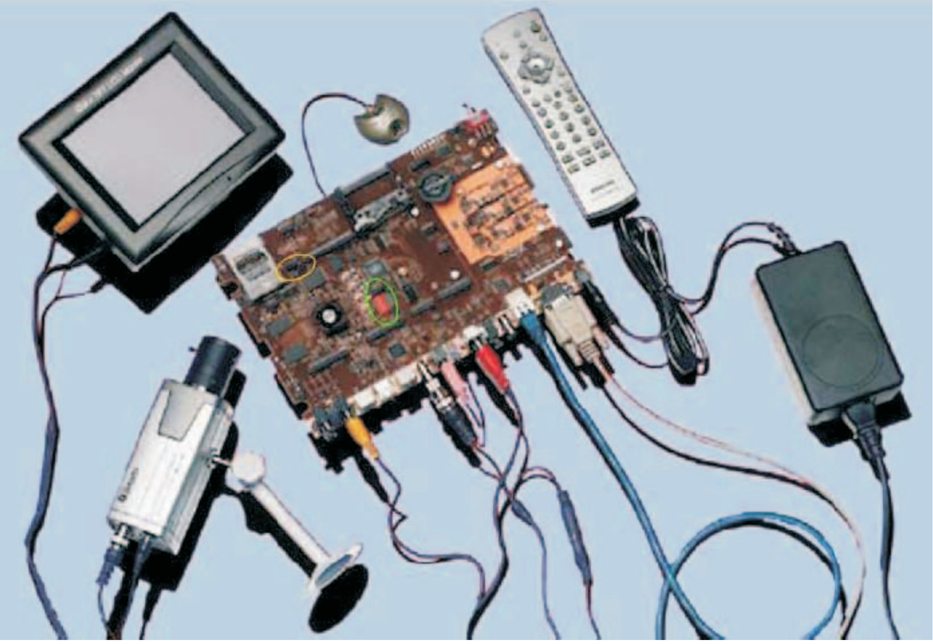


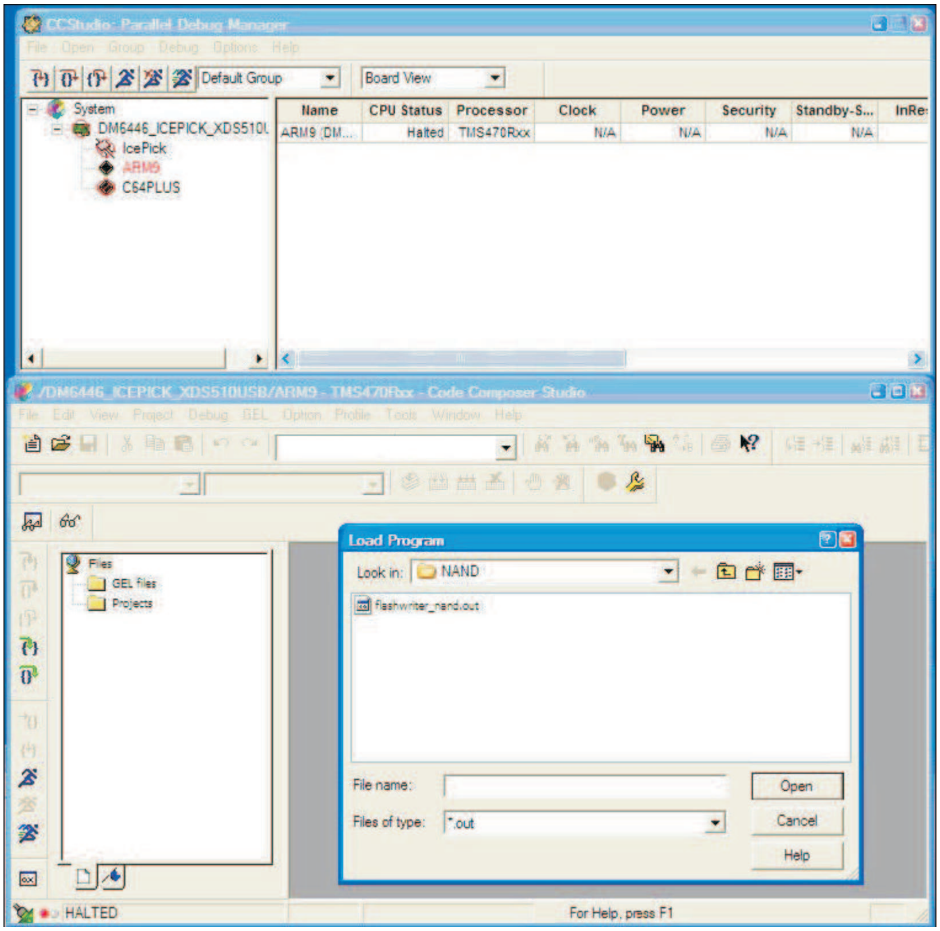
Figure 1. TMDXEVM6446, DVEVM

1. Set J4 jumper (circled in yellow in Figure 1) to NAND.
2. Set S3 red switch (circled in green in Figure 1) to 0000111110, where 1=on and 0 =off.

2.2 Use Code Composer Studio to Load First and Second Stage NAND Boot Loaders

This section requires completion of Section 2.1.

1. Open the ARM-side Code Composer Studio debugger. From the menu, click on File → Load Program and open the flashwriter_nand.out image as in Figure 2:


CCStudio: Parallel Debug Manager
File Open Group Debug Options Help
Default Group
Board View
System
DM6446_ICEPICK_XDS510U
IcePick
ARM9
C64PLUS
/DM6446_ICEPICK_XDS510USB/ARM9 - TMS470Rxx - Code Composer Studio
File Edit View Project Debug GEL Option Profile Tools Window Help
Files
GEL files
Projects
Load Program
flashwriter_nand.out
File name:
Open
Cancel
Help
HALTED
For Help, press F1

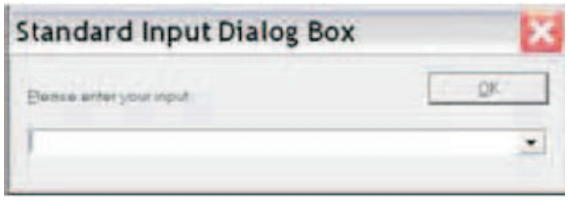
Name	CPU Status	Processor	Clock	Power	Security	Standby-S...	InRe:
ARM9 (DM...	Halted	TMS470Rxx	N/A	N/A	N/A	N/A	

Look in: NAND

Files of type: *.out

Figure 2. Loading flashwritter_nand.out

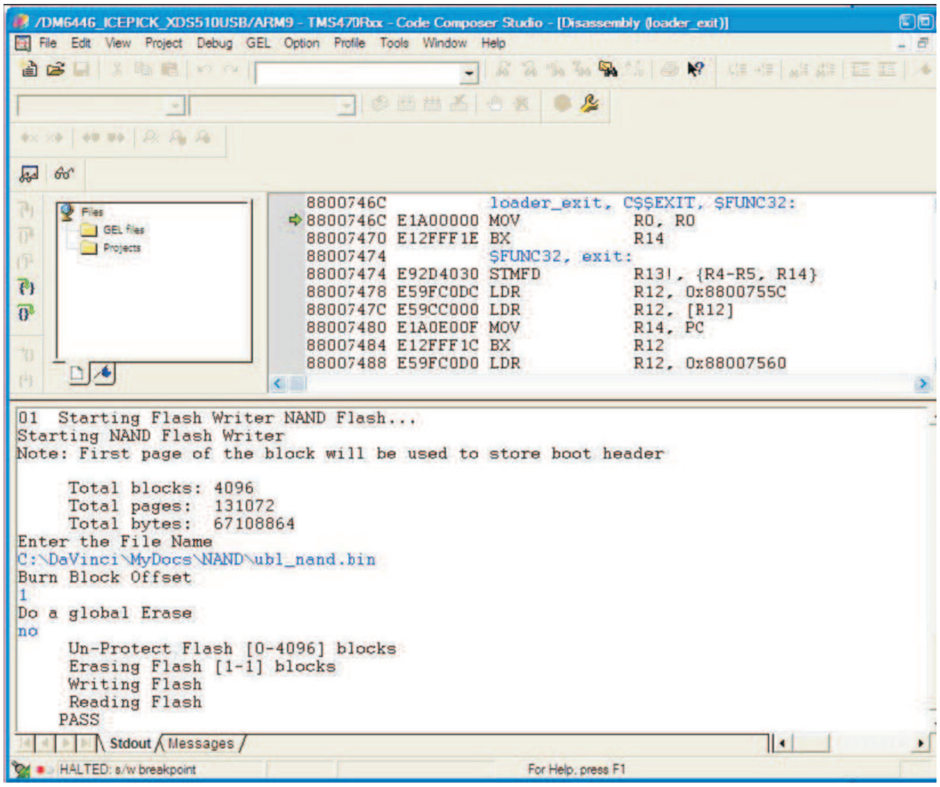
2. Run flashwriter_nand.out by either clicking Debug → Run or hit the F5 key. A dialog box will show as in Figure 3. Enter the local Windows path to the first stage boot loader file, such as `c:\temp\ubl_nand.bin`. Then, it will now ask you for a second parameter, the offset; enter “1” and press return. It will now ask you for a third parameter, global erase; enter “y” and press return.



Please enter your input

Figure 3. Standard Input Dialog Box

The process may take a few seconds; to verify, Figure 4 shows a screen capture of what a successful screen log would look like:

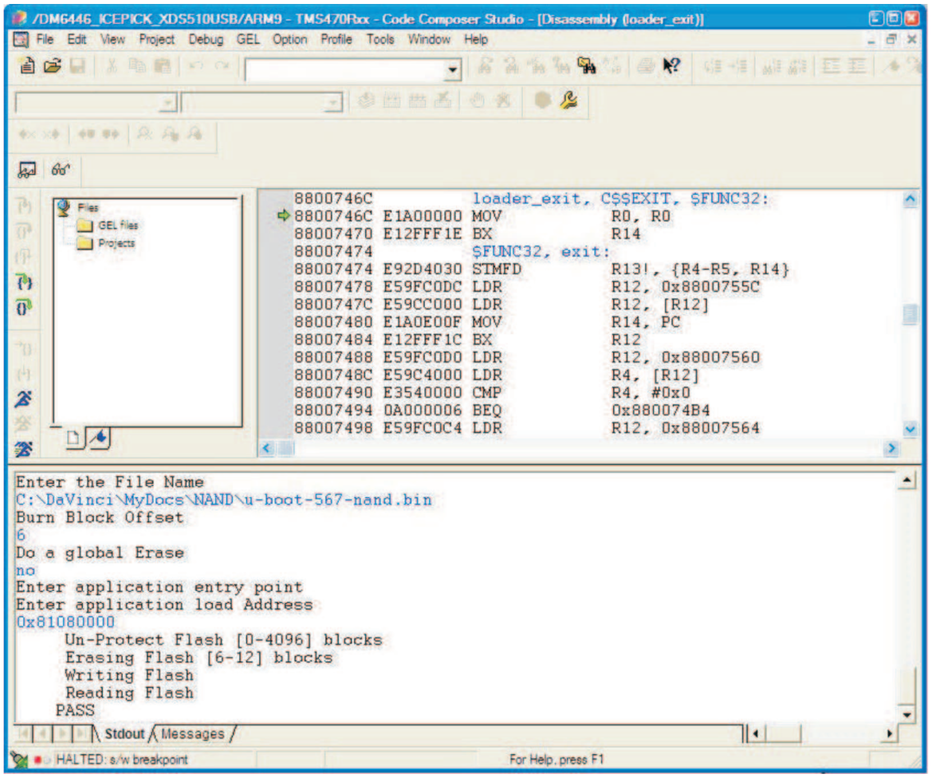

/DM6446_ICEPICK_XDS510USB/ARM9 - TMS470Rxx - Code Composer Studio - [Disassembly (loader_exit)]
File Edit View Project Debug GEL Option Profile Tools Window Help
Files
GEL files
Projects
Stdout
Messages
HALTED: s/w breakpoint
For Help, press F1

```
8800746C          loader_exit, C$$EXIT, $FUNC32:
8800746C E1A00000 MOV              R0, R0
88007470 E12FFF1E BX               R14
88007474          $FUNC32, exit:
88007474 E92D4030 STMFD            R13!, {R4-R5, R14}
88007478 E59FC0DC LDR              R12, 0x8800755C
8800747C E59CC000 LDR              R12, [R12]
88007480 E1A0E00F MOV              R14, PC
88007484 E12FFF1C BX               R12
88007488 E59FC0D0 LDR              R12, 0x88007560
```

```
01  Starting Flash Writer NAND Flash...
Starting NAND Flash Writer
Note: First page of the block will be used to store boot header

     Total blocks: 4096
     Total pages:  131072
     Total bytes:  67108864
Enter the File Name
C:\DaVinci\MyDocs\NAND\ubl_nand.bin
Burn Block Offset
1
Do a global Erase
no
     Un-Protect Flash [0-4096] blocks
     Erasing Flash [1-1] blocks
     Writing Flash
     Reading Flash
    PASS
```

Figure 4. Successful ubl_nand.bin Flashing Screen

3. Load the flashwriter_nand.out program again by selecting File → Reload Program from the Code Composer Studio menu and run it by either clicking Debug → Run or hit the F5 key. Again, a dialog box will show as in Figure 3. Enter the local Windows path to the second stage boot loader file, such as `c:\temp\u-boot-567-nand.bin`. Then, it will ask you for a second parameter, the offset; enter *6* and press return. It will now ask you for a third parameter, global erase; enter *n* and press return. After that, it will ask you for a fourth parameter, application entry point; enter *0x81080000* and press return. For the fifth parameter, application load address; enter *0x81080000* and press return This process may take several minutes; to verify, Figure 5 shows a screen capture of what a successful log should look like.

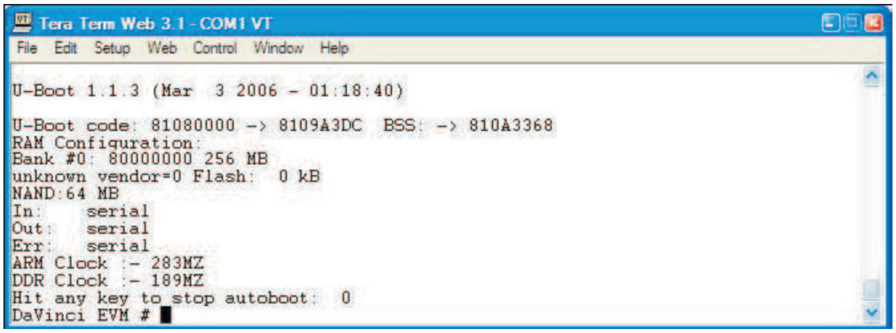

/DM6446_ICEPICK_XDS510USB/ARM9 - TMS470Rxx - Code Composer Studio - [Disassembly (loader_exit)]
File Edit View Project Debug GEL Option Profile Tools Window Help
Files
GEL files
Projects
Stdout
Messages
HALTED: s/w breakpoint
For Help, press F1

```
8800746C          loader_exit, C$$EXIT, $FUNC32:
8800746C E1A00000 MOV               R0, R0
88007470 E12FFF1E BX                R14
88007474          $FUNC32, exit:
88007474 E92D4030 STMFD             R13!, {R4-R5, R14}
88007478 E59FC0DC LDR               R12, 0x8800755C
8800747C E59CC000 LDR               R12, [R12]
88007480 E1A0E00F MOV               R14, PC
88007484 E12FFF1C BX                R12
88007488 E59FC0D0 LDR               R12, 0x88007560
8800748C E59C4000 LDR               R4, [R12]
88007490 E3540000 CMP               R4, #0x0
88007494 0A000006 BEQ               0x880074B4
88007498 E59FC0C4 LDR               R12, 0x88007564
```

```
Enter the File Name
C:\DaVinci\MyDocs\NAND\u-boot-567-nand.bin
Burn Block Offset
6
Do a global Erase
no
Enter application entry point
Enter application load Address
0x81080000
     Un-Protect Flash [0-4096] blocks
     Erasing Flash [6-12] blocks
     Writing Flash
     Reading Flash
    PASS
```

Figure 5. Successful u-boot-567-nand.bin Flashing Screen

4. Verify that NAND u-boot is working properly. Power down DVEVM, disconnect the JTAG emulator and make sure the DVEVM is configured as in Section 2.1. Also, make sure the terminal application settings are 115200 baud rate, 8-bit data, one stop bit, no parity and no flow control. Turn on the DVEVM and you should be greeted with a u-boot prompt on the serial terminal as in Figure 6.


Tera Term Web 3.1 - COM1 VT
File Edit Setup Web Control Window Help

```
U-Boot 1.1.3 (Mar  3 2006 - 01:18:40)

U-Boot code: 81080000 -> 8109A3DC  BSS: -> 810A3368
RAM Configuration:
Bank #0: 80000000 256 MB
unknown vendor=0 Flash:  0 kB
NAND:64 MB
In:    serial
Out:   serial
Err:   serial
ARM Clock :- 283MZ
DDR Clock :- 189MZ
Hit any key to stop autoboot:  0
DaVinci EVM # █
```

Figure 6. Serial Terminal Application Showing Successful u-boot is Flashed

At this point, you can configure the u-boot parameters to boot a desired Linux kernel stored in the host workstation via the TFTP and NFS mount file system on the host work station. This configuration provides a good development environment as it allows you to quickly make and test changes to applications (working tree could be NFS mounted) and the Linux kernel (can try different Linux kernel builds by simply copying the desired one to TFTP directory on host). For information on how to configure u-boot parameters, see the *Booting Linux Kernel and Mounting a File System on DaVinci EVM* document from the DaVinci Ease of Use Series.

2.3 Build Linux Kernel With NAND Support

This section is independent of Section 2.1 and Section 2.2. However, it assumes that you have gone through the Getting Started Guide (GSG) included in the DaVinci EVM kit, in particular the *DVEVM Software Setup* chapter in which you learned the process of building a Linux kernel. This section follows from that point and assumes the same directory structure defined in the GSG.

1. Go to the directory (on the host Linux workstation) where the Linux support package (LSP) is found:

```
host$cd/home/user/working/lps/ti-davinci
I
```

2. Launch the Linux kernel configuration utility:

```
host$makeARCH=armCROSS_COMPILE=arm_v5t_le-xconfig
g
```

3. Go to *ATA/ATAPI/MFM/RLL Support* (under *Device Drivers*) and uncheck the main box. This disables the ATA drivers for the HDD so that the NAND file system is included instead. Refer to Figure 7.

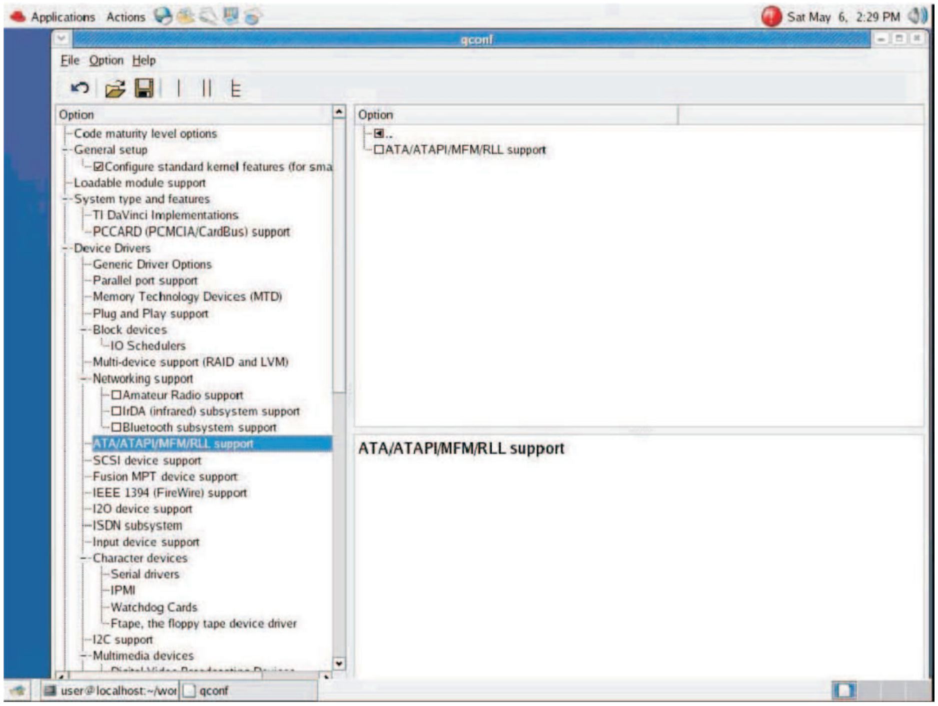
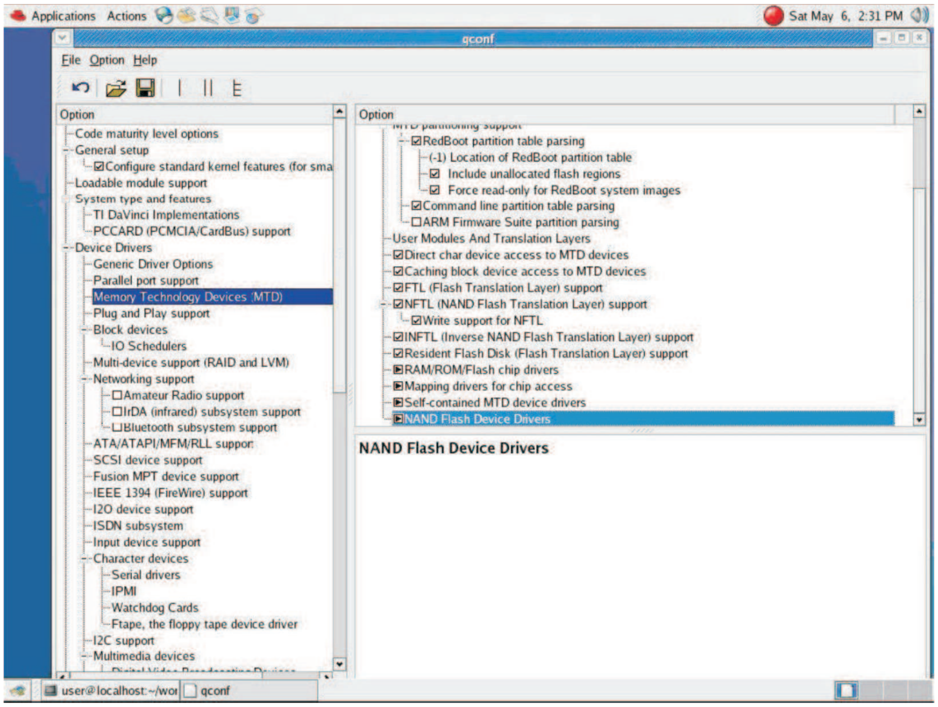


Figure 7. Disabling ATA Driver Support

4. Go to the NAND Flash Device Drivers option under Memory Technology Devices (MTD). See Figure 8.

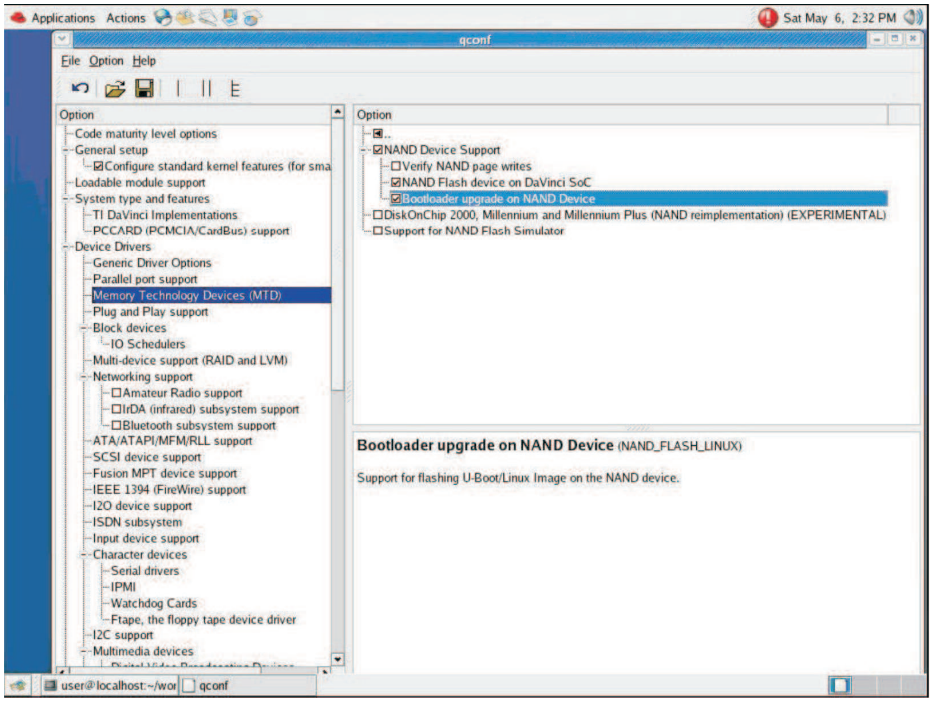


Option
MTD partitioning support
☑RedBoot partition table parsing
(-1) Location of RedBoot partition table
☑ Include unallocated flash regions
☑ Force read-only for RedBoot system images
☑Command line partition table parsing
□ARM Firmware Suite partition parsing
User Modules And Translation Layers
☑Direct char device access to MTD devices
☑Caching block device access to MTD devices
☑FTL (Flash Translation Layer) support
☑NFTL (NAND Flash Translation Layer) support
☑Write support for NFTL
☑INFTL (Inverse NAND Flash Translation Layer) support
☑Resident Flash Disk (Flash Translation Layer) support
▣RAM/ROM/Flash chip drivers
▣Mapping drivers for chip access
▣Self-contained MTD device drivers
▣NAND Flash Device Drivers

NAND Flash Device Drivers

Figure 8. Selecting NAND Flash Device Drivers

5. Make sure that the NAND Flash Device on the DaVinci System-on-Chip (SoC) option has a check and not a dot. Also, check the Bootloader upgrade on the NAND Device as well. See Figure 9.



Option
▣..
☑NAND Device Support
□Verify NAND page writes
☑NAND Flash device on DaVinci SoC
☑Bootloader upgrade on NAND Device
□DiskOnChip 2000, Millennium and Millennium Plus (NAND reimplementation) (EXPERIMENTAL)
□Support for NAND Flash Simulator

Bootloader upgrade on NAND Device (NAND_FLASH_LINUX)

Support for flashing U-Boot/Linux Image on the NAND device.

Figure 9. Enabling NAND Support

6. Log in as *user* (if you haven't already done so) prior to building the Linux Kernel in the next step (similar to the steps in the GSG).

   ```
   host$ suuser
   ```

7. Build the Linux kernel with this command:

   ```
   host$makeARCH=armCROSS_COMPILE=arm_v5t_le-uImage
   ```

At this point, you have a Linux kernel that has NAND Flash driver support instead of ATA HDD support. Section 2.4 shows you how to use this Linux Kernel to write any desired Linux Kernel image to NAND Flash.

2.4 Use the Linux Kernel With NAND Support to Write the Desired Linux Kernel to NAND Flash

This section requires completion of Section 2.1, Section 2.2, and Section 2.3. Keep in mind that these sections refer to two Linux kernels: one with NAND Flash support produced in Section 2.3 one that you will burn into NAND Flash. These two Linux kernels could be the same, but they could also be different (e.g., perhaps you want HDD support on the DVEVM, but want to store kernel on NAND Flash instead of default NOR Flash because you envision your end product not having any NOR Flash). To this end, this section uses the Linux kernel produced in Section 2.3 and the original kernel shipped with DVEVM that has HDD support. Since you will be copying over these Linux Kernels, it is suggested that a copy of each be stored in a safe place.

1. Log in as *user* on the host machine and place the Linux kernel to be burned to NAND Flash under the file system that will be NFS mounted to target the DVEVM.

   ```
   host$cp/tftpboot/uImage~/workdir/filesys/opt
   ```

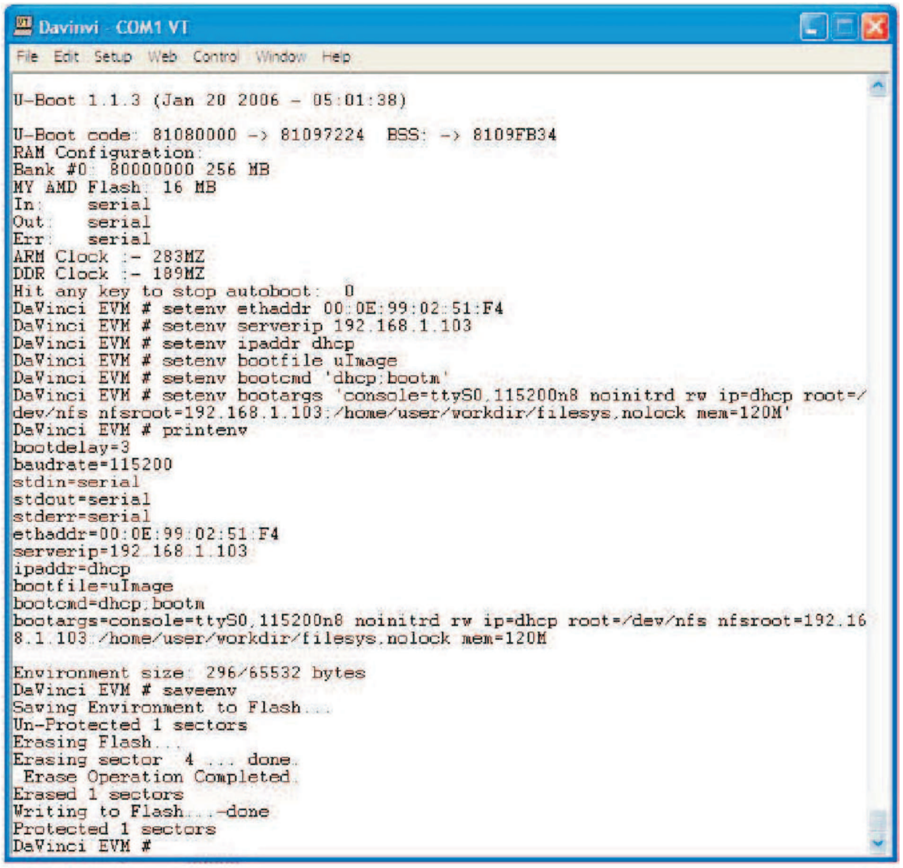
2. Log in as *user* on the host machine and copy Linux kernel image with NAND support to the TFTP directory and change permissions to the image file.

   ```
   host$cp~/workdir/lsp/ti_davinci/arch/arm/boot/uImage/tftpboot
   host$chmod a+r/tftpboot/uImage
   ```

3. Configure u-boot parameters so that the DVEVM uses TFTP to load Linux kernel from the host machine and the NFS mount file system from the host machine. This is discussed in detail in the *Booting Linux Kernel and Mounting a File System on DaVinci EVM* document from the DaVinci Ease of Use Series, but the short version will be provided here. Note that all the remaining steps in this section take place on the terminal applications window as opposed to the host workstation. With the terminal application running, power on the DVEVM and press any key on your terminal window to stop u-boot's autoboot sequence. At the u-boot prompt, type the following commands. See Figure 10.

   Note: Replace 00:0E:99:02:51:F4 with the MAC address found on your DVEVM board and all instances of 192.168.1.103 with the IP address on your host workstation

   ```
   EVM # setenv ethaddr 00:0E:99:02:51:F4
   EVM # setenv serverip 192.168.1.103
   EVM # setenv ipaddr dhcp

   EVM # setenv bootfile uImage

   EVM # setenv bootcmd ‘dhcp;bootm’

   EVM # setenv bootargs ‘console=ttyS0,115200n8 noinitrd rw ip=dhcp root=/dev/nfs
   nfsroot=192.168.1.103:/home/user/workdir/filesys,nolock mem=120M’

   EVM # saveenv
   ```



```
U-Boot 1.1.3 (Jan 20 2006 - 05:01:38)

U-Boot code: 81080000 -> 81097224  BSS: -> 8109FB34
RAM Configuration:
Bank #0: 80000000 256 MB
MY AMD Flash: 16 MB
In:    serial
Out:   serial
Err:   serial
ARM Clock :- 283MZ
DDR Clock :- 189MZ
Hit any key to stop autoboot:  0
DaVinci EVM # setenv ethaddr 00:0E:99:02:51:F4
DaVinci EVM # setenv serverip 192.168.1.103
DaVinci EVM # setenv ipaddr dhcp
DaVinci EVM # setenv bootfile uImage
DaVinci EVM # setenv bootcmd 'dhcp;bootm'
DaVinci EVM # setenv bootargs 'console=ttyS0,115200n8 noinitrd rw ip=dhcp root=/
dev/nfs nfsroot=192.168.1.103:/home/user/workdir/filesys,nolock mem=120M'
DaVinci EVM # printenv
bootdelay=3
baudrate=115200
stdin=serial
stdout=serial
stderr=serial
ethaddr=00:0E:99:02:51:F4
serverip=192.168.1.103
ipaddr=dhcp
bootfile=uImage
bootcmd=dhcp;bootm
bootargs=console=ttyS0,115200n8 noinitrd rw ip=dhcp root=/dev/nfs nfsroot=192.16
8.1.103:/home/user/workdir/filesys,nolock mem=120M

Environment size: 296/65532 bytes
DaVinci EVM # saveenv
Saving Environment to Flash...
Un-Protected 1 sectors
Erasing Flash...
Erasing sector  4 ... done.
 Erase Operation Completed.
Erased 1 sectors
Writing to Flash...-done
Protected 1 sectors
DaVinci EVM #
```

Figure 10. Booting Linux Kernel via TFTP and Root Mounting File System From NFS

4. Cycle the power on the DVEVM and log in as root on your terminal window. The DVEVM should be running the Linux Kernel with NAND support from the host workstation's /tftpboot directory. You can verify this by invoking the following command on the terminal application window and making sure this device is found. See Figure 11.

   ```
   $ ls /dev/mtd0
   ```

5. Format NAND Flash device. See Figure 11.

   ```
   $ ftl_format /dev/mtd0
   ```

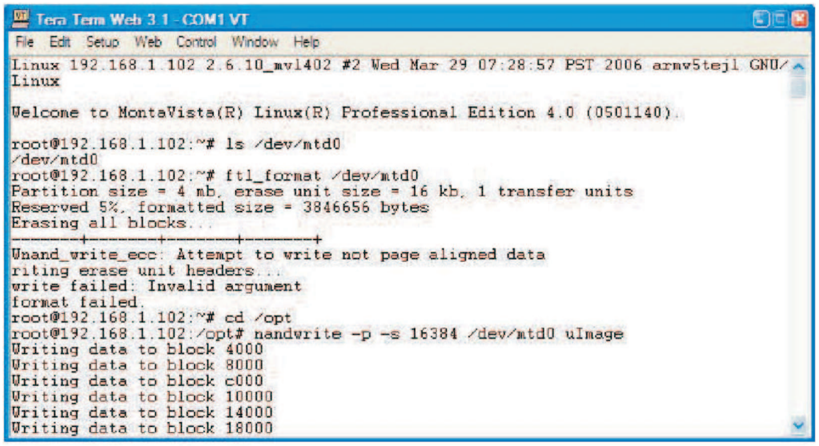
6. Go to the directory where the Linux kernel to be burned to NAND Flash is stored using terminal application window. See Figure 11.

   ```
   $ cd /opt
   ```

7. Write Linux Kernel image to NAND Flash. See Figure 11.

   ```
   $ nandwrite -p -s 16384 /dev/mtd0 uImage
   ```

Note: Offset 0x4000 (16384) is block 2 of your NAND Flash /dev/mtd0 memory.

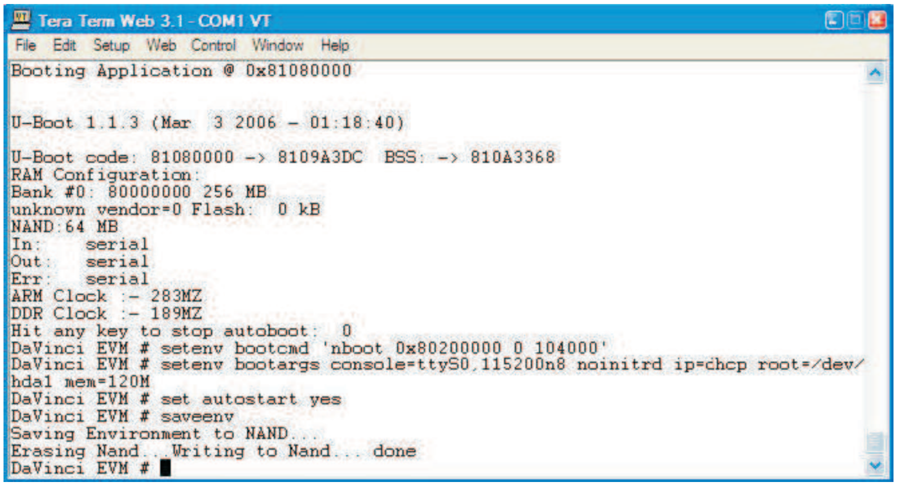

Tera Term Web 3.1 - COM1 VT
File Edit Setup Web Control Window Help

```
Linux 192.168.1.102 2.6.10_mvl402 #2 Wed Mar 29 07:28:57 PST 2006 armv5tejl GNU/
Linux

Welcome to MontaVista(R) Linux(R) Professional Edition 4.0 (0501140).

root@192.168.1.102:~# ls /dev/mtd0
/dev/mtd0
root@192.168.1.102:~# ftl_format /dev/mtd0
Partition size = 4 mb, erase unit size = 16 kb, 1 transfer units
Reserved 5%, formatted size = 3846656 bytes
Erasing all blocks...
--------+--------+--------+--------+
Wnand_write_ecc: Attempt to write not page aligned data
riting erase unit headers...
write failed: Invalid argument
format failed.
root@192.168.1.102:~# cd /opt
root@192.168.1.102:/opt# nandwrite -p -s 16384 /dev/mtd0 uImage
Writing data to block 4000
Writing data to block 8000
Writing data to block c000
Writing data to block 10000
Writing data to block 14000
Writing data to block 18000
```

Figure 11. Writing Linux Kernel to NAND Flash

8. Cycle power to your DVEVM and press any key to stop u-boot’s auto-boot sequence. In this step, the u-boot environment variables are set so that the u-boot knows where to find the Linux Kernel image in NAND Flash as well as configure the DVEVM to load this Linux kernel (as opposed to the one via TFTP) and use the file system in DVEVM HDD (as opposed to NFS mounting the file system from a host workstation). This configuration is what will most likely be encountered in the final product. At the u-boot prompt, type the following commands. See Figure 12.

```
EVM # setenv bootcmd 'nboot 0x80200000 0 104000'
EVM # setenv bootargs console=ttyS0,115200n8 noinitrd ip=dhcp root=/dev/hda1 mem=120M
EVM # setenv autostart yes
EVM # saveenv
```


Tera Term Web 3.1 - COM1 VT
File Edit Setup Web Control Window Help

```
Booting Application @ 0x81080000


U-Boot 1.1.3 (Mar  3 2006 - 01:18:40)

U-Boot code: 81080000 -> 8109A3DC  BSS: -> 810A3368
RAM Configuration:
Bank #0: 80000000 256 MB
unknown vendor=0 Flash:  0 kB
NAND:64 MB
In:    serial
Out:   serial
Err:   serial
ARM Clock :- 283MZ
DDR Clock :- 189MZ
Hit any key to stop autoboot:  0
DaVinci EVM # setenv bootcmd 'nboot 0x80200000 0 104000'
DaVinci EVM # setenv bootargs console=ttyS0,115200n8 noinitrd ip=dhcp root=/dev/
hda1 mem=120M
DaVinci EVM # set autostart yes
DaVinci EVM # saveenv
Saving Environment to NAND...
Erasing Nand...Writing to Nand... done
DaVinci EVM # █
```

Figure 12. Reconfiguring u-boot to Load Linux Kernel From NAND Flash

9. Cycle power to your DVEVM and you should now have a boot loader (u-boot) and Linux kernel stored in NAND Flash.

3 References

- *Booting and Flashing via the DaVinci TMS320DM644x Serial Interface* (SPRAAI4)

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 TI 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. Information of third parties may be subject to additional restrictions.

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.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Clocks and Timers	www.ti.com/clocks	Digital Control	www.ti.com/digitalcontrol
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Telephony	www.ti.com/telephony
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265