AM1808/OMAP-L138 SOM-M1 NAND Flash Programming and Use

Application Note 580

Logic PD // Products
Published: August 2013

Abstract

This document explains how to program and boot the NAND flash on the User Interface (UI) Board included in the AM1808 EVM Development Kit and OMAP-L138 EVM Development Kit.

Revision History

REV	EDITOR	DESCRIPTION	APPROVAL	DATE
A	SK	-Initial Release	JA, SO	08/22/13

Table of Contents

1 Introduction

This application note provides steps for programming and booting NAND flash for use with the AM1808 SOM-M1 and OMAP-L138 SOM-M1. The NAND flash addressed in this document resides on the User Interface (UI) Board included in both the AM1808 EVM Development Kit and OMAP-L138 EVM Development Kit.

1.1 Nomenclature

- This document covers the AM1808 SOM-M1 and OMAP-L138 SOM-M1. Use of “SOM-M1” suggests text that applies to both platforms; information specific to one platform will call out the precise name.
- Use of “EVM Development Kit” suggests text that applies to both the AM1808 EVM Development Kit and OMAP-L138 EVM Development Kit; information specific to one development kit will call out the precise name.

1.2 *AM1808_OMAP-L138_SOM-M1_NAND_Flash_Programming_and_Use_Files* Directory

Accompanying this application note within the *1024612A_AN580_AM1808_OMAP-L138_SOM-M1_NAND_Flash_Programming_and_Use.zip* file is a directory containing software files to be used with the instructions found here. The *AM1808_OMAP-L138_SOM-M1_NAND_Flash_Programming_and_Use_Files* directory should contain the following files that will be referenced throughout this document:

```
rootfs-arago-ram.ext2.gz
rootfs-base.jffs2
ubl_OMAPL138_NAND.bin
u-boot.bin
uImage
```

NOTE: If using the Virtual Machine SDK for the AM1808/OMAP-L138 Linux PSP, these files can be found in the *tftpboot* directory.

1.3 Test Environment

The following environment was used to test the procedures in this document.

1.3.1 Hardware

The part number and revision status of the hardware components used in writing this document are noted below. Please review the most recent schematics and any Product Change Notifications (PCNs) that could affect available features.

Hardware	Model Number (Part Number & Rev)
OMAP-L138 SOM-M1	(1013525 Rev 5)
AM1808 SOM-M1	(1017941 Rev A)
Development Kit Baseboard	(1016660 Rev A)
UI Board	(1013528 Rev 7)

1.3.2 Software

The version numbers of the software components used in writing this document are noted below. Please review the release notes for any software updates that might require changes to the procedures provided within this document.

- Texas Instruments (TI) DaVinci Linux PSP v03-21-00-04[1]
- TI Serial Boot and Flash Loading Utility v2.40[2]
- Logic PD Virtual Machine SDK for the AM1808/OMAP-L138 Linux PSP v03-21-00-04[3]

1.4 Prerequisites

The following items are required to complete the procedures within this document:

- AM1808 EVM Development Kit or OMAP-L138 EVM Development Kit, including:
  - UI Board
  - Null-modem serial cable
  - Ethernet crossover cable

IMPORTANT NOTE: When using the OMAP-L138 EVM Development Kit, the OMAP-L138 SOM-M1 must be connected to the baseboard; the procedures below are not applicable to the TMS320C6748 SOM-M1.

This document also assumes the following:

- The Logic PD Virtual Machine SDK for the AM1808/OMAP-L138 Linux PSP (hereafter, VM) has been downloaded, and the required files are already placed and ready to use.
- If you are using a separate Linux host PC, it is set up with a Linux environment as outlined in the *AM1808/OMAP-L138 Linux User Guide*.[4] Although setting up this Linux environment is a lengthy process, it is necessary to load and use the Linux kernel and root filesystem. The Linux environment can also be used as a platform for further Linux development beyond these procedures.
- Your directory structure follows the structure shown below, and the files are already placed in their respective folders. The commands used in the remainder of this document will reflect the directory structure below. If a different structure is used, the command paths will need to be updated appropriately.

```
$HOME
   +- AM1808_OMAP-L138 (working directory)
      +- setenv.sh
      +- Archive
      +- Docs
      +- Hello
      +- Kernel
          +- linux-03.21.00.04
      +- Images
      +- Recover
          +- sfh_OMAP-L138.exe
          +- NAND
               +- ubl_OMAPL138_NAND.bin
               +- u-boot.bin
          +- SPI_Flash
      +- RootFS
```

[1] http://processors.wiki.ti.com/index.php/DaVinci_%28ARM9%29_PSP_Releases#AM18x.2FOMAP-L138.2FDA850
[2] http://processors.wiki.ti.com/index.php/Serial_Boot_and_Flash_Loading_Utility_for_OMAP-L138
[3] http://support.logicpd.com/downloads/1559/
[4] http://support.logicpd.com/downloads/1539/

```
        +- arago
        +- arago-bitbake
        +- arago-deploy
            +- images
                +- arago
                    +- arago-base-image-glibc-ipk-2009.11-
                    arago.rootfs.ext2.gz  (init RAMFS)
                    +- arago-base-image-glibc-ipk-2009.11-
                    arago.rootfs.jffs2   (JFFS2 image)
                    +- arago-base-image-glibc-ipk-2009.11-
                    arago.rootfs.tar.gz   (for NFS)
        +- arago-oe-dev
        +- arago-tmp
        +- downloads
    +- Tools
        +- Flash-Utils
    +- U-Boot
        +- uboot-03.21.00.04
    +- CodeSourcery (cross-compilation tools)

./
  +- tftpboot (files needed for tftp transfer)
    +- rootfs-arago-ram.ext2.gz
    +- rootfs-base.jffs2
    +- uImage
```

2 Kit Communications

The EVM Development Kit comes with U-Boot programmed in the SPI flash. U-Boot presents a command shell to the user via the development kit's debug serial port. This section will explain how to establish a proper serial connection to the EVM Development Kit using Minicom.

For more information on how to install and configure Minicom, please see the *AM1808/OMAP-L138 Linux User Guide*. **NOTE:** The VM comes with Minicom already installed and set up.

2.1 Connect EVM Development Kit to Host PC

1. Ensure all the switches on the S7 DIP switch block on the development kit baseboard are set to the OFF position.

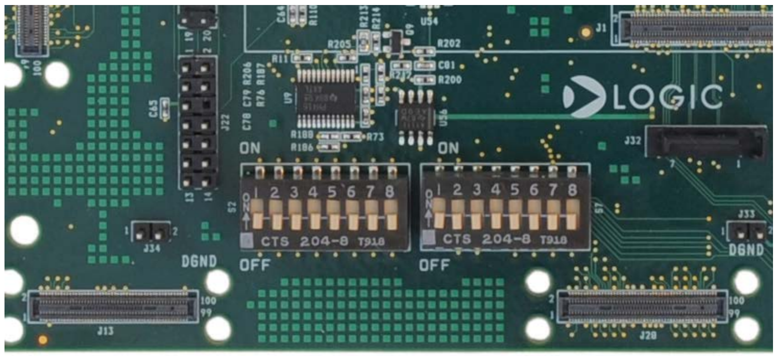
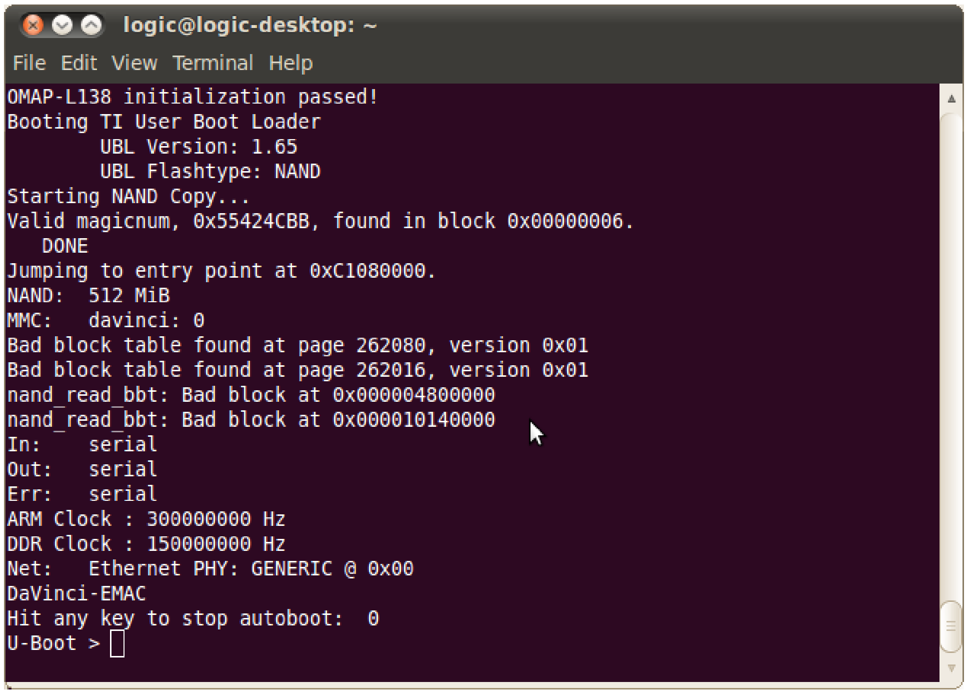


Figure 2.1: EVM Development Kit DIP Switch Settings

2. Connect the serial cable to the serial debug port on the kit baseboard and to an available COM port on your host PC.
3. Use the following command to launch Minicom.

```
host$ sudo minicom
```

4. Power on the EVM Development Kit. If U-Boot is located in SPI flash, you should see output similar to that included below, meaning you have successfully established a serial connection to the EVM Development Kit.



```
OMAP-L138 initialization passed!
Booting TI User Boot Loader
        UBL Version: 1.65
        UBL Flashtype: NAND
Starting NAND Copy...
Valid magicnum, 0x55424CBB, found in block 0x00000006.
   DONE
Jumping to entry point at 0xC1080000.
NAND:  512 MiB
MMC:   davinci: 0
Bad block table found at page 262080, version 0x01
Bad block table found at page 262016, version 0x01
nand_read_bbt: Bad block at 0x000004800000
nand_read_bbt: Bad block at 0x000010140000
In:    serial
Out:   serial
Err:   serial
ARM Clock : 300000000 Hz
DDR Clock : 150000000 Hz
Net:   Ethernet PHY: GENERIC @ 0x00
DaVinci-EMAC
Hit any key to stop autoboot:  0
U-Boot >
```

NOTE: If you see a *BOOTME* message instead of the output above, the S7 DIP switches may not be set to the correct position. Power off the kit, make sure all switches are in the OFF position, and then power on the kit again.

3 Load User Bootloader and U-Boot into NAND Flash

This section will explain how to load the User Bootloader (UBL) and U-Boot on to the SOM-M1. The instructions below will feature the OMAP-L138 SOM-M1; however, filenames can be adjusted for the AM1808 SOM-M1 as needed.

If you are using the VM, you may skip to Section 3.3, as all the necessary files have already been placed.

3.1 Erase NAND Flash

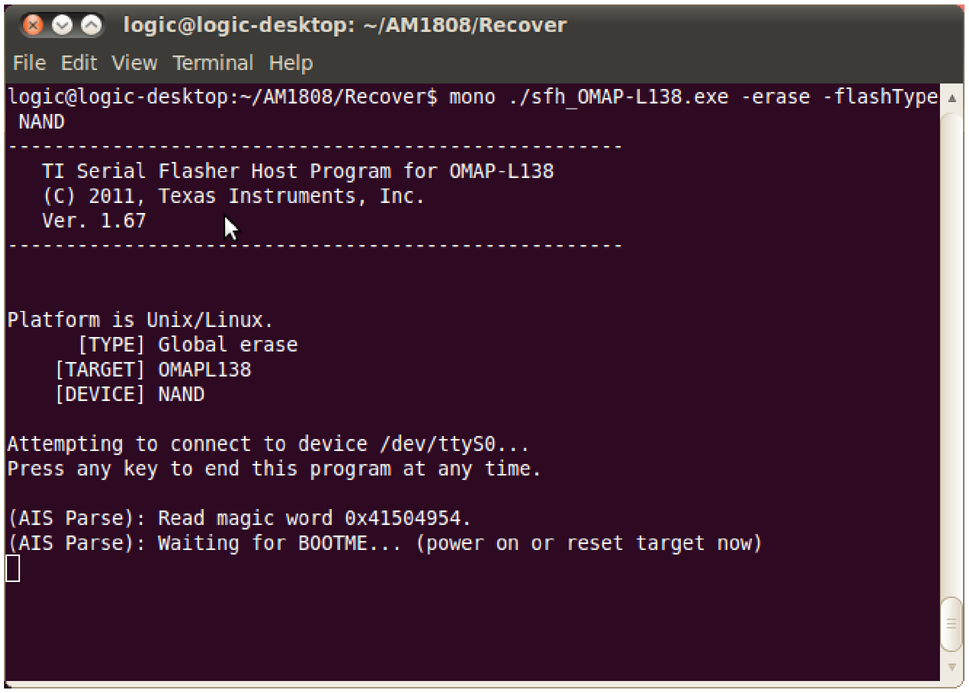
It is necessary to first erase the NAND flash to clean the memory before programming new data into it. Follow the steps below to erase the NAND flash.

1. Attach the UI Board to the EVM Development Kit Baseboard as shown in the appropriate QuickStart Guide. The NAND flash is located at J30 on the UI Board.
   - *Zoom AM1808 EVM QuickStart Guide*[5]
   - *Zoom OMAP-L138 EVM QuickStart Guide*[6]
2. With the EVM Development Kit powered off, set switches 7 and 8 on the S7 DIP switch block on the development kit baseboard to the ON position. Ensure all other DIP switches are set to the OFF position. This will permit the Serial Boot and Flash Loading Utility (hereafter, Flash Loading Utility) operation to work in UART mode from a host PC command window.
3. Connect the null-modem serial cable to the serial debug port on the development kit and to the COM1 port on your host PC. **NOTE:** The Flash Loading Utility assumes it will be communicating with the COM1 port, so it is important that you connect the serial cable to this port on your host PC.
4. Open a terminal and navigate to the directory containing the Flash Loading Utility. **NOTE:** This must be the only terminal window open when completing this step; ensure Minicom is not running during this procedure.
5. Erase the flash.

```
host$ mono ./sfh_OMAP-L138.exe -erase -flashType NAND
```

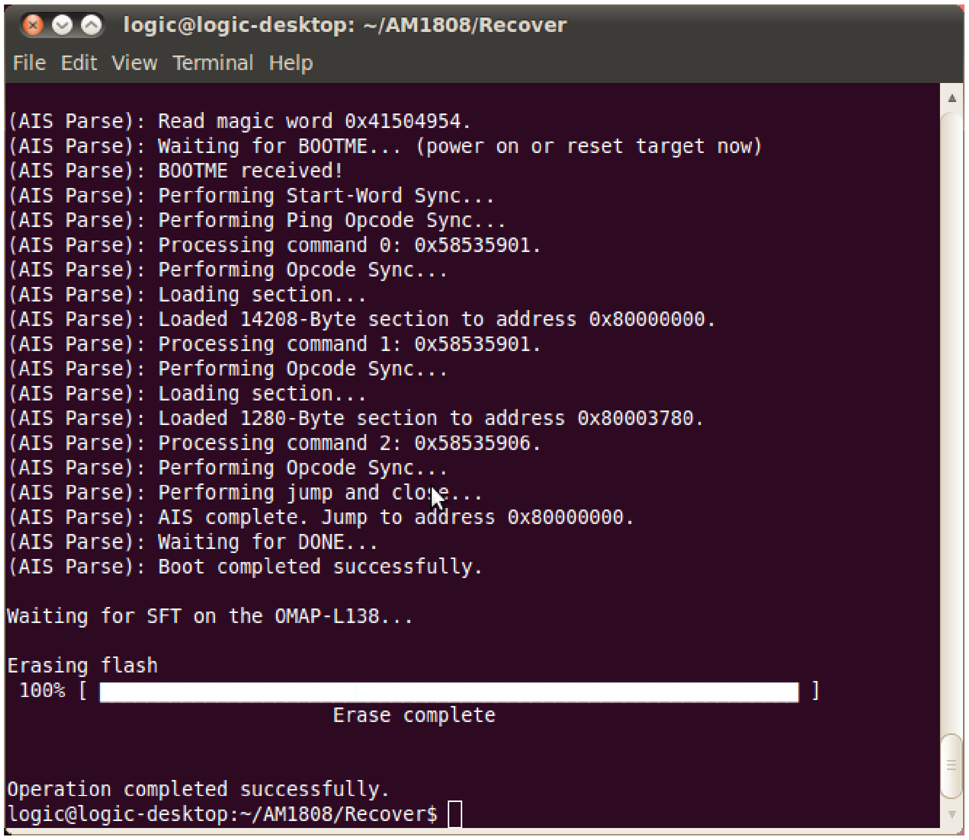
[5] http://support.logicpd.com/downloads/1293/
[6] http://support.logicpd.com/downloads/1230/

You should receive a message that the Flash Loading Utility is attempting to connect to device /dev/ttyS0.


logic@logic-desktop: ~/AM1808/Recover
File Edit View Terminal Help

```
logic@logic-desktop:~/AM1808/Recover$ mono ./sfh_OMAP-L138.exe -erase -flashType
 NAND
-----------------------------------------------------
   TI Serial Flasher Host Program for OMAP-L138
   (C) 2011, Texas Instruments, Inc.
   Ver. 1.67
-----------------------------------------------------


Platform is Unix/Linux.
      [TYPE] Global erase
    [TARGET] OMAPL138
    [DEVICE] NAND

Attempting to connect to device /dev/ttyS0...
Press any key to end this program at any time.

(AIS Parse): Read magic word 0x41504954.
(AIS Parse): Waiting for BOOTME... (power on or reset target now)
```

6. Power on the EVM Development Kit. After a few moments, a message will appear that the NAND flash is being erased.


logic@logic-desktop: ~/AM1808/Recover
File Edit View Terminal Help

```
(AIS Parse): Read magic word 0x41504954.
(AIS Parse): Waiting for BOOTME... (power on or reset target now)
(AIS Parse): BOOTME received!
(AIS Parse): Performing Start-Word Sync...
(AIS Parse): Performing Ping Opcode Sync...
(AIS Parse): Processing command 0: 0x58535901.
(AIS Parse): Performing Opcode Sync...
(AIS Parse): Loading section...
(AIS Parse): Loaded 14208-Byte section to address 0x80000000.
(AIS Parse): Processing command 1: 0x58535901.
(AIS Parse): Performing Opcode Sync...
(AIS Parse): Loading section...
(AIS Parse): Loaded 1280-Byte section to address 0x80003780.
(AIS Parse): Processing command 2: 0x58535906.
(AIS Parse): Performing Opcode Sync...
(AIS Parse): Performing jump and close...
(AIS Parse): AIS complete. Jump to address 0x80000000.
(AIS Parse): Waiting for DONE...
(AIS Parse): Boot completed successfully.

Waiting for SFT on the OMAP-L138...

Erasing flash
 100% [ ████████████████████████████████████████████████████████████ ]
                            Erase complete


Operation completed successfully.
logic@logic-desktop:~/AM1808/Recover$
```

NOTE: If the *BOOTME* message continues to appear after powering on the EVM Development Kit, press the S5 reset button on the baseboard. This will send the *BOOTME* command to the Flash Loading Utility.

7. Once the erase is complete, power off the EVM Development Kit and proceed to the next section.

3.2 Program NAND Flash with UBL and U-Boot

The UBL and U-Boot are programmed at the same time using a single command.

1. Ensure the EVM Development Kit is powered off and set DIP switches S7:7 and S7:8 to the ON position. Set all others to the OFF position.
2. Open a terminal window and navigate to the Flash Loading Utility directory, if you are not already there.

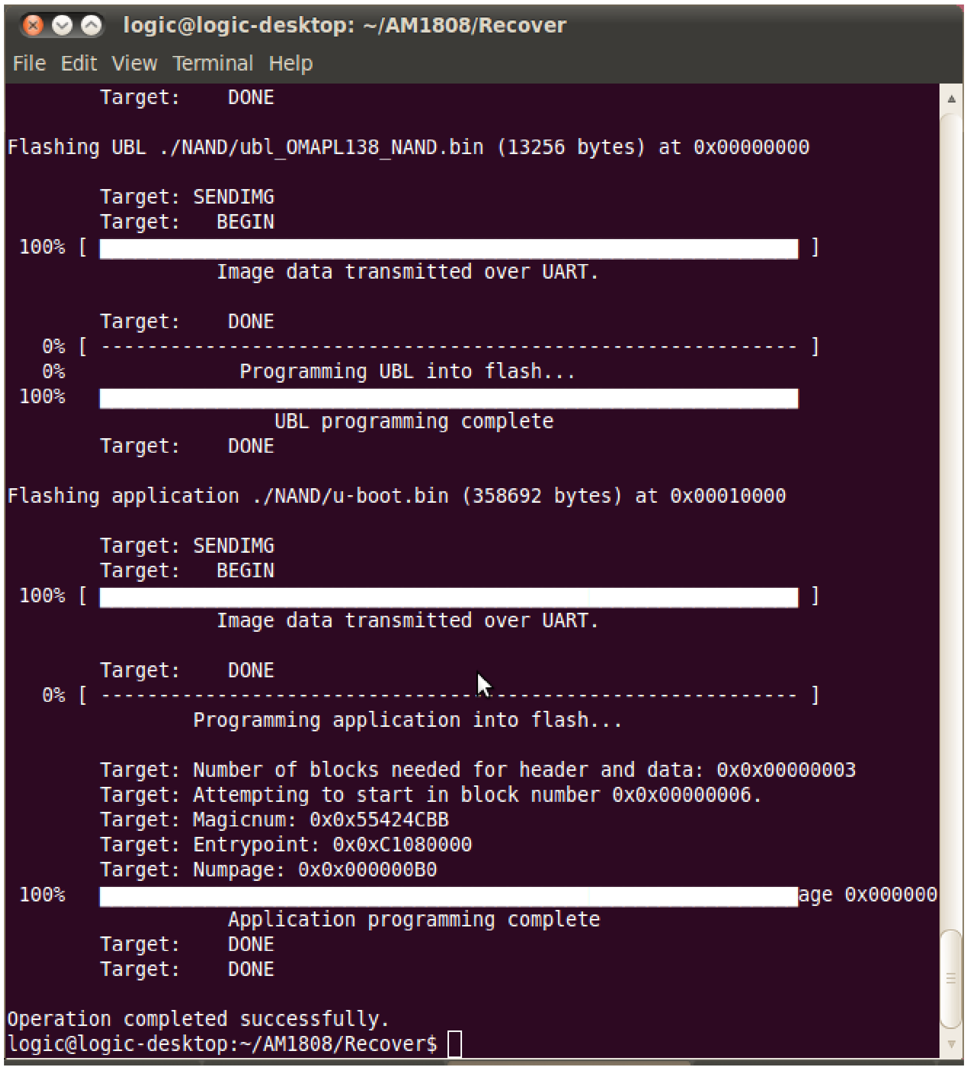
```
host$ cd AM1808_OMAP-L138/Recover
```

3. Flash the UBL and U-Boot files into NAND

```
host$ mono ./sfh_OMAP-L138.exe -flash -targetType OMAPL138 -flashType
NAND -v ./NAND/ubl_OMAPL138_NAND.bin ./NAND/u-boot.bin
```

NOTE: The default *targetType* is C6748_LCDK. The default *flashType* is SPI_MEM. The -*v* is optional for this command.

4. Power on the EVM Development Kit. After a few moments, you should see messages that indicate flashing is taking place.


logic@logic-desktop: ~/AM1808/Recover
File Edit View Terminal Help

```
        Target:    DONE

Flashing UBL ./NAND/ubl_OMAPL138_NAND.bin (13256 bytes) at 0x00000000

        Target: SENDIMG
        Target:   BEGIN
 100% [                                                            ]
                  Image data transmitted over UART.

        Target:    DONE
   0% [ ------------------------------------------------------------ ]
   0%               Programming UBL into flash...
 100%
                       UBL programming complete
        Target:    DONE

Flashing application ./NAND/u-boot.bin (358692 bytes) at 0x00010000

        Target: SENDIMG
        Target:   BEGIN
 100% [                                                            ]
                  Image data transmitted over UART.

        Target:    DONE
   0% [ ------------------------------------------------------------ ]
                Programming application into flash...

        Target: Number of blocks needed for header and data: 0x0x00000003
        Target: Attempting to start in block number 0x0x00000006.
        Target: Magicnum: 0x0x55424CBB
        Target: Entrypoint: 0x0xC1080000
        Target: Numpage: 0x0x000000B0
 100%                                                            age 0x000000
                   Application programming complete
        Target:    DONE
        Target:    DONE

Operation completed successfully.
logic@logic-desktop:~/AM1808/Recover$
```

5. When the flashing process is complete, power off the EVM Development Kit.

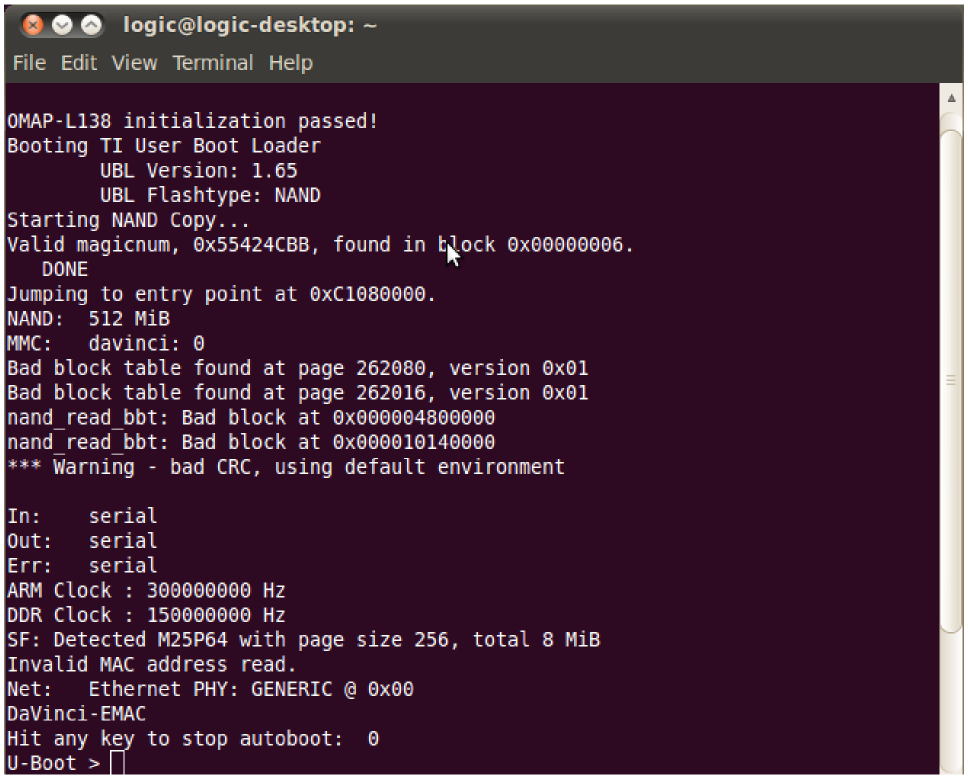
3.3 Boot to U-Boot from NAND Flash

1. Open a separate terminal window and open Minicom.

```
host$ sudo minicom
```

2. Set DIP switch S7:5 to the ON position; set all others to the OFF position. This will configure the EVM Development Kit to boot from NAND flash.

3. Power on the EVM Development Kit. The kit should boot to the U-Boot prompt; hit any key to stop the autoboot process.



```
OMAP-L138 initialization passed!
Booting TI User Boot Loader
        UBL Version: 1.65
        UBL Flashtype: NAND
Starting NAND Copy...
Valid magicnum, 0x55424CBB, found in block 0x00000006.
   DONE
Jumping to entry point at 0xC1080000.
NAND:  512 MiB
MMC:   davinci: 0
Bad block table found at page 262080, version 0x01
Bad block table found at page 262016, version 0x01
nand_read_bbt: Bad block at 0x000004800000
nand_read_bbt: Bad block at 0x000010140000
*** Warning - bad CRC, using default environment

In:    serial
Out:   serial
Err:   serial
ARM Clock : 300000000 Hz
DDR Clock : 150000000 Hz
SF: Detected M25P64 with page size 256, total 8 MiB
Invalid MAC address read.
Net:   Ethernet PHY: GENERIC @ 0x00
DaVinci-EMAC
Hit any key to stop autoboot:  0
U-Boot >
```

The EVM Development Kit NAND flash has been programmed with the UBL and U-Boot. The kit will now boot from NAND flash on power up when DIP switch S7:5 is set to the ON position.

4 SOM-M1 Memory Map

The following memory map shows the location of different types of memory and program files on the SOM-M1. The information is useful when navigating, locating, and storing configuration and application files.

Memory	Start	End	Size	Notes
Internal ROM (DSP L2 ROM)	0x1170_0000	0x117F_FFFF	1024 KB	
Internal SRAM (Shared RAM)	0x8000_0000	0x8001_FFFF	128 KB	
DDR (EVM)	0xC000_0000	0xC7FF_FFFF	128 MB	
DDR (eXperimenter)	0xC000_0000	0xC3FF_FFFF	64 MB	
NOR (on UI board)	0x6000_0000	0x607F_FFFF	8 MB	
Peripheral Space				1

TABLE NOTES:

1. See "Section 2.5: Memory Map Summary" in the TI *OMAP-L138 C6000 DSP+ARM Processor Datasheet*[7] for additional information.

4.1 SPI Flash Contents

The SPI flash chip is located on the SOM-M1 and is connected to the SPI1 port. This chip is not memory mapped and the flash size is 8 MB. This fills an address space of 0x7F_FFFF.

Address	Item
0x000000	ARM AIS bootloader (*arm-api-ais.bin*)
0x010000	U-Boot prepended with config (*u_boot.bin*)
0x080000	Linux image (*uImage*)
0x280000	File system (*ramdisk-base.gz*)

4.2 UI Board NAND Flash

The UI Board that ships with the EVM Development Kit has a NAND flash socket and a NAND chip installed with the following properties:

- Part number: Micron MT29F4G08AAC
- Base address: 0x6200_0000
- Page size: 2112 bytes (2048 + 64 bytes)
- Block size: 64 pages (128K + 4K bytes)
- Plane size: 2 planes x 2048 blocks per plane
- Device size: 512 MB (4 Gb), 4096 blocks

[7] http://www.ti.com/product/omap-l138

5 Prepare to Boot from NAND

This section will explain how to prepare for a NAND boot by establishing a network connection, and downloading and flashing the Linux kernel to NAND.

5.1 Establish Network Connection

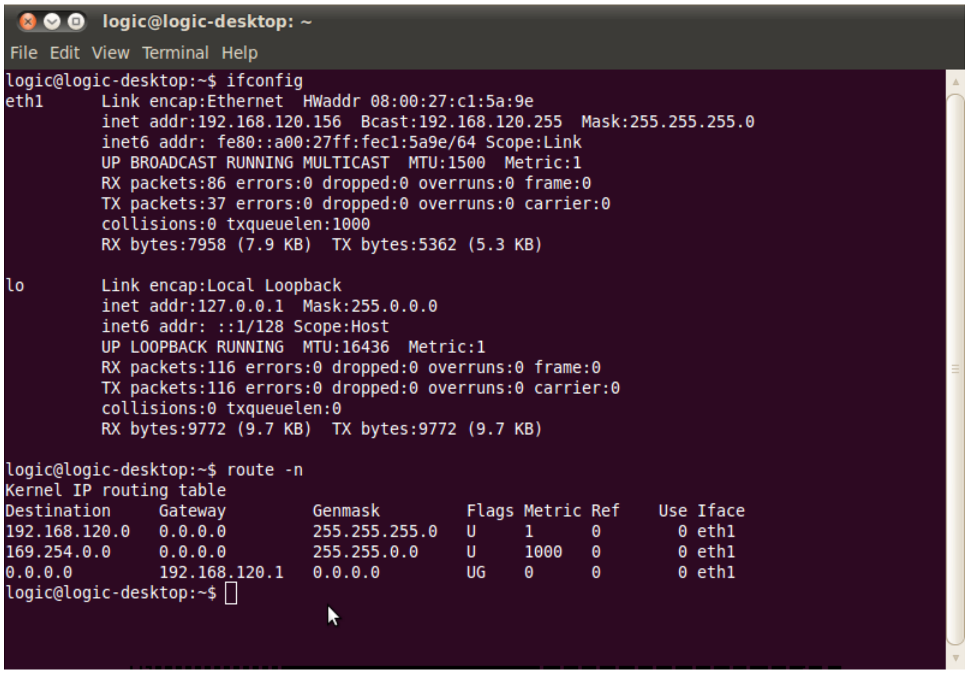
1. Connect the Ethernet crossover cable to the Ethernet port (J31) on the EVM Development Kit and to your host PC.
2. Open a separate terminal and display the host PC's IP address and subnet mask.

```
host$ ifconfig
```

3. Display the default gateway.

```
host$ route -n
```

The following output should be seen:

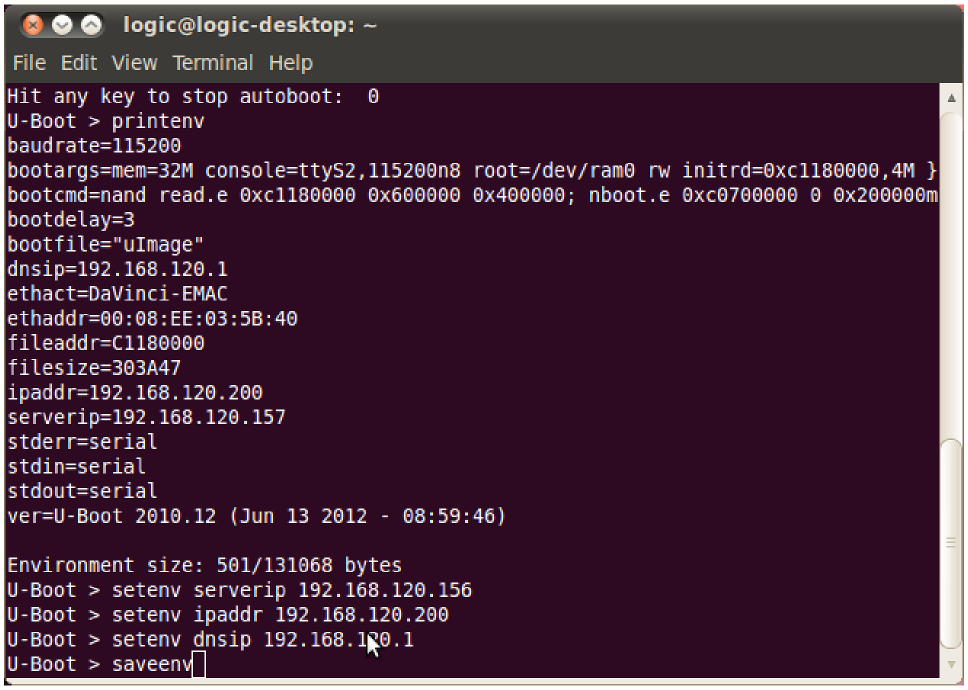


```
logic@logic-desktop:~$ ifconfig
eth1      Link encap:Ethernet  HWaddr 08:00:27:c1:5a:9e
          inet addr:192.168.120.156  Bcast:192.168.120.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fec1:5a9e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:86 errors:0 dropped:0 overruns:0 frame:0
          TX packets:37 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:7958 (7.9 KB)  TX bytes:5362 (5.3 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:116 errors:0 dropped:0 overruns:0 frame:0
          TX packets:116 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:9772 (9.7 KB)  TX bytes:9772 (9.7 KB)

logic@logic-desktop:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.120.0   0.0.0.0         255.255.255.0   U     1      0        0 eth1
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 eth1
0.0.0.0         192.168.120.1   0.0.0.0         UG    0      0        0 eth1
logic@logic-desktop:~$
```

4. In the output, take note of the following information:
   - inet addr: IP address of the host PC
   - Mask: Network subnet mask
   - Gateway: IP address of the default gateway
5. Return to the U-Boot prompt in your Minicom session and print out the environment variables.

```
U-Boot > printenv
```

6. Notice the environment variables that are named. **NOTE:** Some of the variables below may be undefined and will not appear in your print list.
   - Ipaddr: IP address of the kit
   - serverip: IP address of the host PC from which you want to download
   - ethaddr: Ethernet MAC address of the kit's network interface
   - dnsip: IP address of Domain Name Server
7. Use the *setenv* command to set each of the environment variables above. Be sure to use values that correspond to your network, rather than the values in the example below.

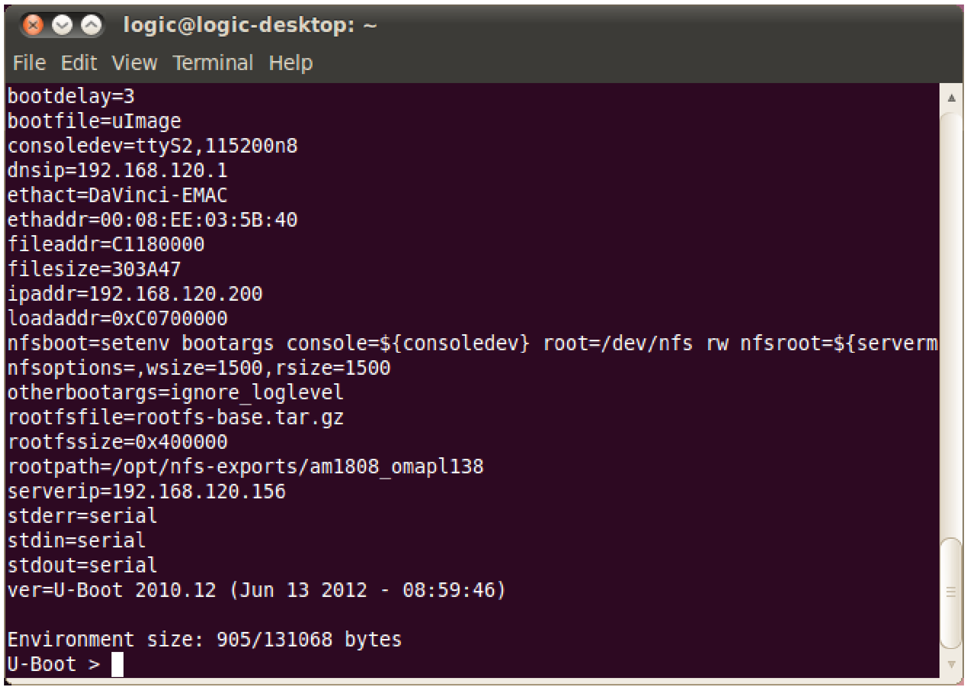


```
Hit any key to stop autoboot:  0
U-Boot > printenv
baudrate=115200
bootargs=mem=32M console=ttyS2,115200n8 root=/dev/ram0 rw initrd=0xc1180000,4M }
bootcmd=nand read.e 0xc1180000 0x600000 0x400000; nboot.e 0xc0700000 0 0x200000m
bootdelay=3
bootfile="uImage"
dnsip=192.168.120.1
ethact=DaVinci-EMAC
ethaddr=00:08:EE:03:5B:40
fileaddr=C1180000
filesize=303A47
ipaddr=192.168.120.200
serverip=192.168.120.157
stderr=serial
stdin=serial
stdout=serial
ver=U-Boot 2010.12 (Jun 13 2012 - 08:59:46)

Environment size: 501/131068 bytes
U-Boot > setenv serverip 192.168.120.156
U-Boot > setenv ipaddr 192.168.120.200
U-Boot > setenv dnsip 192.168.120.1
U-Boot > saveenv
```

8. Save the settings.

```
U-Boot > saveenv
```

9. Use the *printenv* command again to confirm the environment variables have been updated.

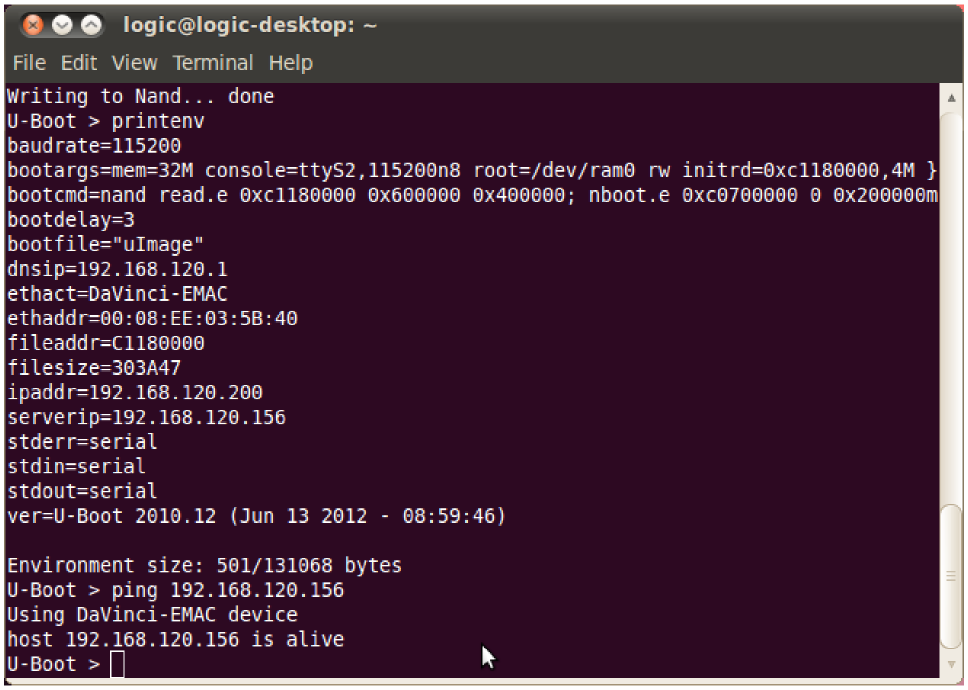


```
bootdelay=3
bootfile=uImage
consoledev=ttyS2,115200n8
dnsip=192.168.120.1
ethact=DaVinci-EMAC
ethaddr=00:08:EE:03:5B:40
fileaddr=C1180000
filesize=303A47
ipaddr=192.168.120.200
loadaddr=0xC0700000
nfsboot=setenv bootargs console=${consoledev} root=/dev/nfs rw nfsroot=${serverm
nfsoptions=,wsize=1500,rsize=1500
otherbootargs=ignore_loglevel
rootfsfile=rootfs-base.tar.gz
rootfssize=0x400000
rootpath=/opt/nfs-exports/am1808_omapl138
serverip=192.168.120.156
stderr=serial
stdin=serial
stdout=serial
ver=U-Boot 2010.12 (Jun 13 2012 - 08:59:46)

Environment size: 905/131068 bytes
U-Boot >
```

10. Verify network connectivity exists with the command below, where ***www.xxx.yyy.zzz*** is the IP address of your host PC.

```
U-Boot > ping www.xxx.yyy.zzz
```

You should see output stating that the IP address is alive.



```
Writing to Nand... done
U-Boot > printenv
baudrate=115200
bootargs=mem=32M console=ttyS2,115200n8 root=/dev/ram0 rw initrd=0xc1180000,4M }
bootcmd=nand read.e 0xc1180000 0x600000 0x400000; nboot.e 0xc0700000 0 0x200000m
bootdelay=3
bootfile="uImage"
dnsip=192.168.120.1
ethact=DaVinci-EMAC
ethaddr=00:08:EE:03:5B:40
fileaddr=C1180000
filesize=303A47
ipaddr=192.168.120.200
serverip=192.168.120.156
stderr=serial
stdin=serial
stdout=serial
ver=U-Boot 2010.12 (Jun 13 2012 - 08:59:46)

Environment size: 501/131068 bytes
U-Boot > ping 192.168.120.156
Using DaVinci-EMAC device
host 192.168.120.156 is alive
U-Boot >
```

NOTE: The *ping* command can appear non-responsive the first time it is used. If this happens, kill the command by pressing **Ctrl + C** in the Tera Term window and try the command again.

5.2 Install and Configure TFTP Server

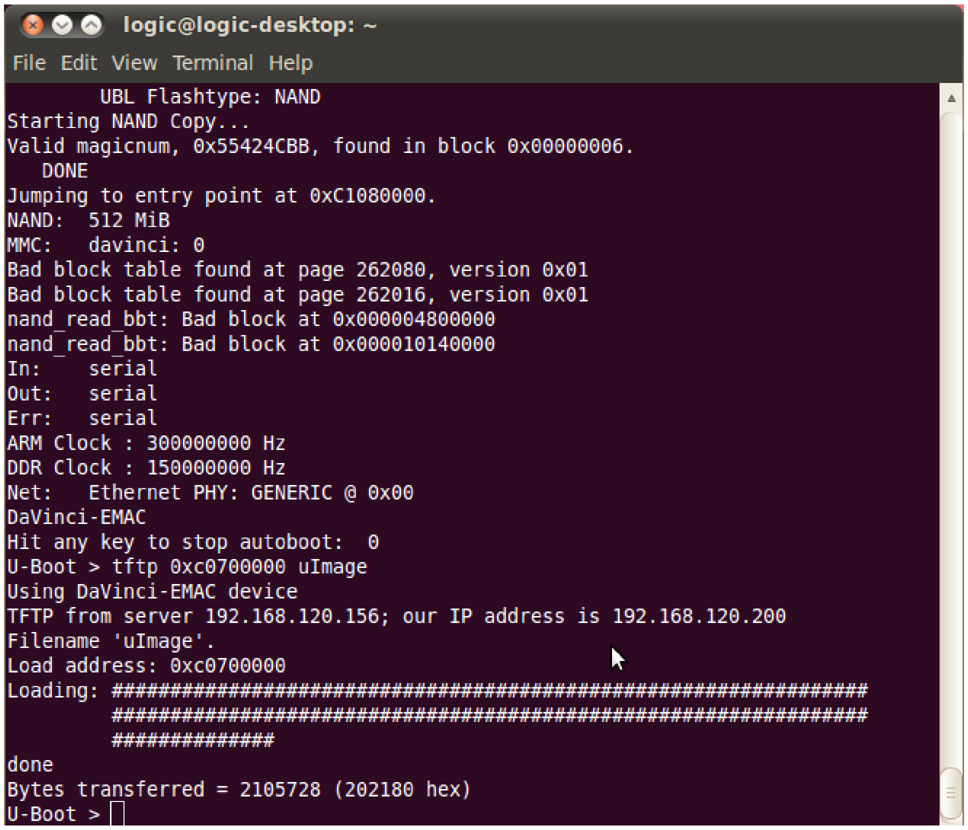
The VM already has a TFTP server installed and configured. If using a separate Linux host PC, please see the *AM1808/OMAP-L138 Linux User Guide* for instructions about how to configure the TFTP server.

5.3 Program Kernel Image into NAND Flash

1. Set DIP switch S7:5 to the ON position. Ensure all other DIP switches are set to the OFF position. This will configure the EVM Development Kit to boot from NAND flash.
2. Power on the development kit.
3. Interrupt the U-Boot autoboot process by pressing any key on the keyboard.
4. Return to your Minicom session and transfer the kernel image to SDRAM at address 0xC0700000.

```
U-Boot > tftp 0xc0700000 uImage
```

You should see the following output:



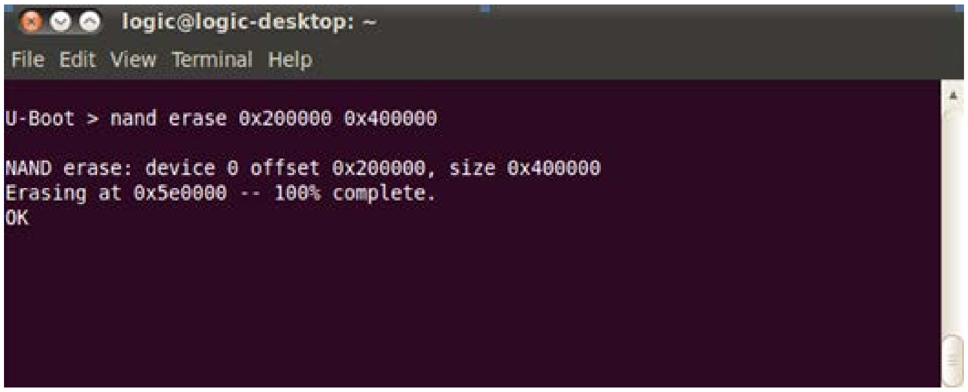
```
        UBL Flashtype: NAND
Starting NAND Copy...
Valid magicnum, 0x55424CBB, found in block 0x00000006.
   DONE
Jumping to entry point at 0xC1080000.
NAND:  512 MiB
MMC:   davinci: 0
Bad block table found at page 262080, version 0x01
Bad block table found at page 262016, version 0x01
nand_read_bbt: Bad block at 0x000004800000
nand_read_bbt: Bad block at 0x000010140000
In:    serial
Out:   serial
Err:   serial
ARM Clock : 300000000 Hz
DDR Clock : 150000000 Hz
Net:   Ethernet PHY: GENERIC @ 0x00
DaVinci-EMAC
Hit any key to stop autoboot:  0
U-Boot > tftp 0xc0700000 uImage
Using DaVinci-EMAC device
TFTP from server 192.168.120.156; our IP address is 192.168.120.200
Filename 'uImage'.
Load address: 0xc0700000
Loading: #################################################################
         #################################################################
         ##############
done
Bytes transferred = 2105728 (202180 hex)
U-Boot > 
```

5. Make note of the number of bytes transferred. In the example above, it is 0x202180 hex. The erase and write sizes in the following commands must be larger than this size.

6. Erase the NAND space that will hold the *uImage* file. This will erase a hex space of 0x400000 at an offset of 0x200000.

```
U-Boot > nand erase 0x200000 0x400000
```

You should see the following output:

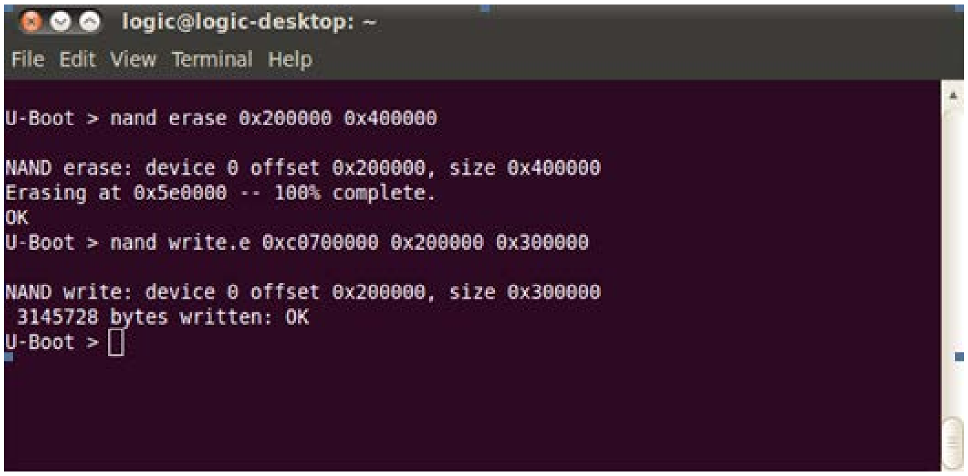


```
U-Boot > nand erase 0x200000 0x400000

NAND erase: device 0 offset 0x200000, size 0x400000
Erasing at 0x5e0000 -- 100% complete.
OK
```

7. Write the *uImage* file from SDRAM to NAND flash at an offset of 0x200000. The hex space of 0x300000 is greater than the *uImage* file size of 0x20005C.

```
U-Boot> nand write.e 0xc0700000 0x200000 0x300000
```

You should see the following output:



```
U-Boot > nand erase 0x200000 0x400000

NAND erase: device 0 offset 0x200000, size 0x400000
Erasing at 0x5e0000 -- 100% complete.
OK
U-Boot > nand write.e 0xc0700000 0x200000 0x300000

NAND write: device 0 offset 0x200000, size 0x300000
 3145728 bytes written: OK
U-Boot > 
```

The Linux kernel *uImage* is now programmed into NAND flash. Proceed to Section 6 to boot the Linux kernel and root filesystem as RAMdisk. Proceed to Section 7 to boot the Linux kernel with a JFFS2 root filesystem.

6 Boot Linux Kernel and Root Filesystem as RAMdisk

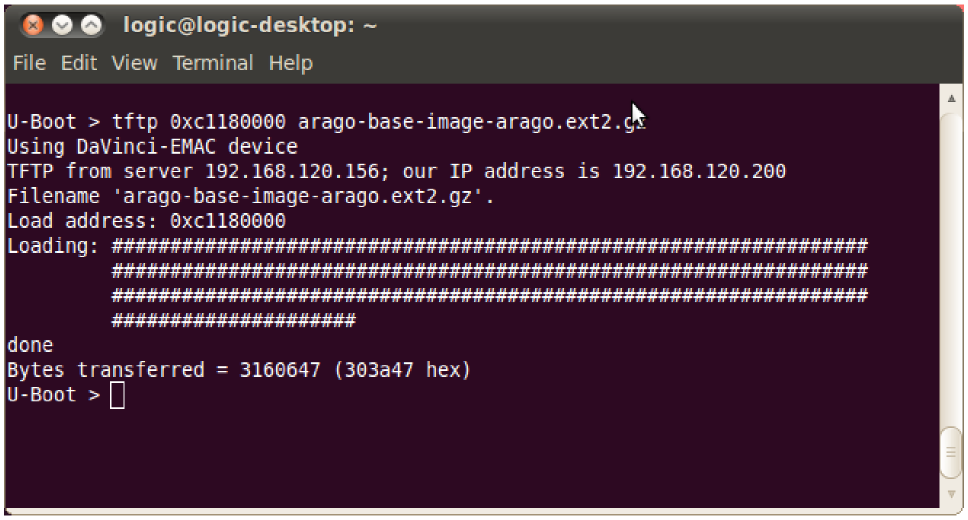
This section will explain how to download the RAMdisk-based filesystem and save it into NAND memory. Please note that applications added during RAMdisk operation will not be retained persistently in memory when the EVM Development Kit is powered off.

6.1 Program Root Filesystem into NAND Flash

1. Follow the steps in Section 5 to establish a network connection and flash the kernel image into NAND.
2. Transfer the root filesystem to SDRAM at address 0xC1180000.

```
U-Boot > tftp 0xC1180000 rootfs-arago-ram.ext2.gz
```

You should see the output below. Note the size of the root filesystem; in the example below, it is 3160647 bytes or 303a47 hex.

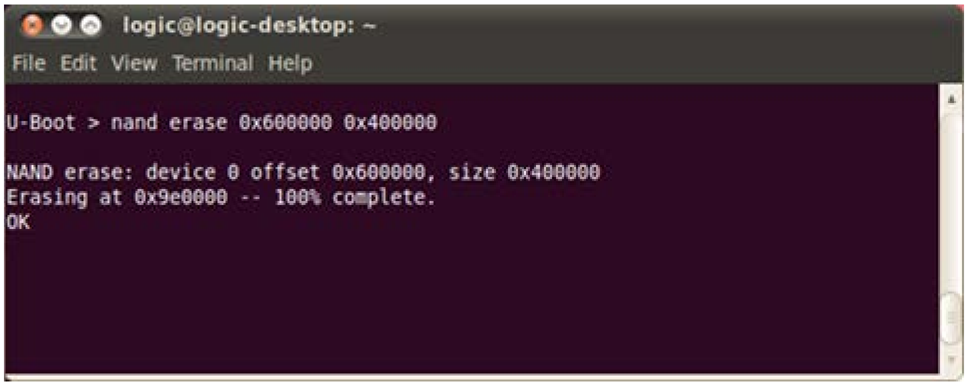


```
U-Boot > tftp 0xc1180000 arago-base-image-arago.ext2.gz
Using DaVinci-EMAC device
TFTP from server 192.168.120.156; our IP address is 192.168.120.200
Filename 'arago-base-image-arago.ext2.gz'.
Load address: 0xc1180000
Loading: #################################################################
         #################################################################
         #################################################################
         #####################
done
Bytes transferred = 3160647 (303a47 hex)
U-Boot >
```

3. Erase the NAND flash area starting with an offset of 0x600000 for a space of 0x400000.

```
U-Boot > nand erase 0x600000 0x400000
```

You should see the following output:

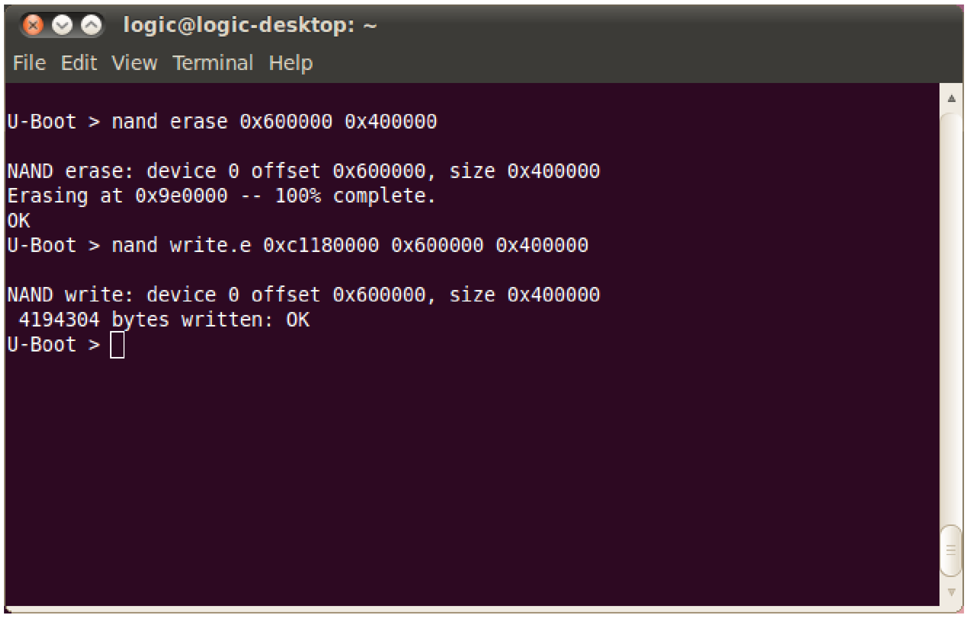


```
U-Boot > nand erase 0x600000 0x400000

NAND erase: device 0 offset 0x600000, size 0x400000
Erasing at 0x9e0000 -- 100% complete.
OK
```

4. Write the root filesystem from SDRAM to NAND flash at an offset of 0x600000.

```
U-Boot> nand write.e 0xc1180000 0x600000 0x400000
```

You should see the following output:



```
U-Boot > nand erase 0x600000 0x400000

NAND erase: device 0 offset 0x600000, size 0x400000
Erasing at 0x9e0000 -- 100% complete.
OK
U-Boot > nand write.e 0xc1180000 0x600000 0x400000

NAND write: device 0 offset 0x600000, size 0x400000
 4194304 bytes written: OK
U-Boot >
```

6.2 Create *bootcmd* and *bootargs* Environment Variables

This section will explain how to create environment variables for the *bootcmd* and *bootargs* values to boot a RAMdisk out of NAND.

1. Set up *bootcmd*.

```
U-Boot > setenv bootcmd 'nand read.e 0xc1180000 0x600000 0x400000;
nboot.e 0xC0700000 0 0x200000; bootm'
```

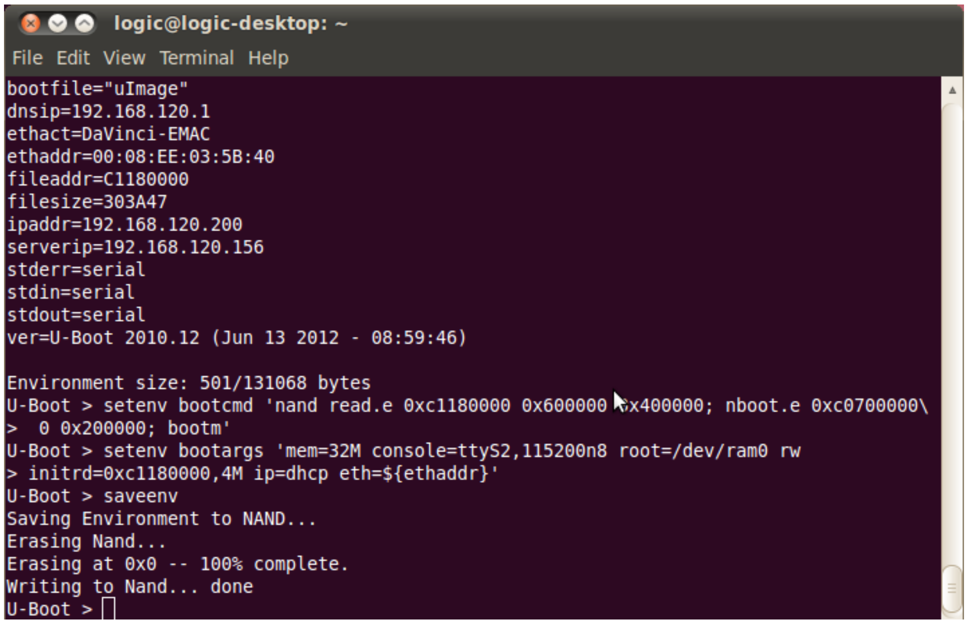
2. Set up *bootargs*.

```
U-Boot > setenv bootargs 'mem=32M console=ttyS2,115200n8 root=/dev/ram0
rw initrd=0xC1180000,4M ip=dhcp eth=${ethaddr}'
```

3. Save these values so they will persist after a power cycle.

```
U-Boot > saveenv
```

You should see the following output:

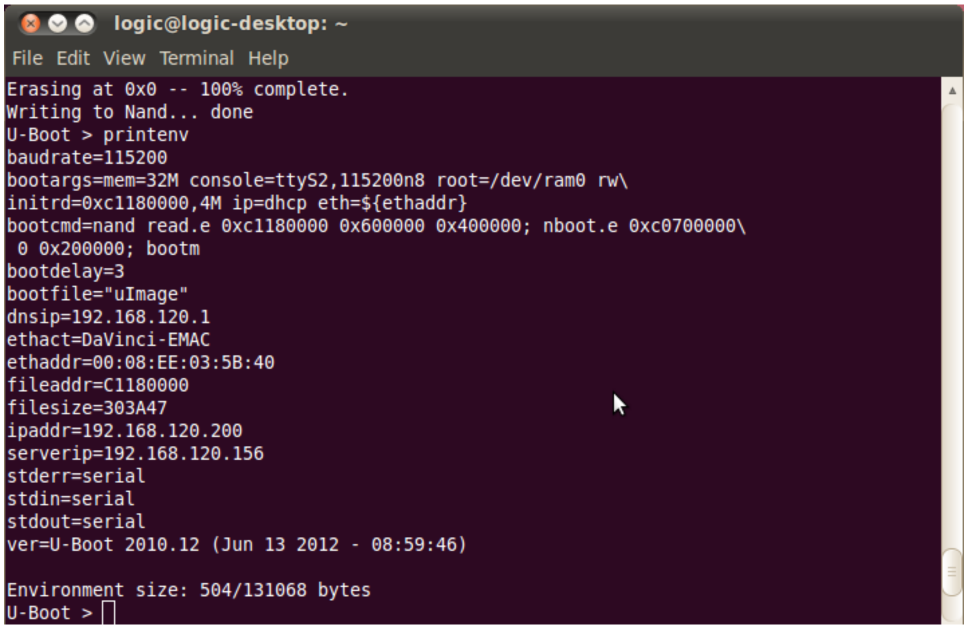


```
bootfile="uImage"
dnsip=192.168.120.1
ethact=DaVinci-EMAC
ethaddr=00:08:EE:03:5B:40
fileaddr=C1180000
filesize=303A47
ipaddr=192.168.120.200
serverip=192.168.120.156
stderr=serial
stdin=serial
stdout=serial
ver=U-Boot 2010.12 (Jun 13 2012 - 08:59:46)

Environment size: 501/131068 bytes
U-Boot > setenv bootcmd 'nand read.e 0xc1180000 0x600000 0x400000; nboot.e 0xc0700000\
>  0 0x200000; bootm'
U-Boot > setenv bootargs 'mem=32M console=ttyS2,115200n8 root=/dev/ram0 rw
> initrd=0xc1180000,4M ip=dhcp eth=${ethaddr}'
U-Boot > saveenv
Saving Environment to NAND...
Erasing Nand...
Erasing at 0x0 -- 100% complete.
Writing to Nand... done
U-Boot >
```

4. View the new environment variables.

```
U-Boot > printenv
```

You should see the following output:

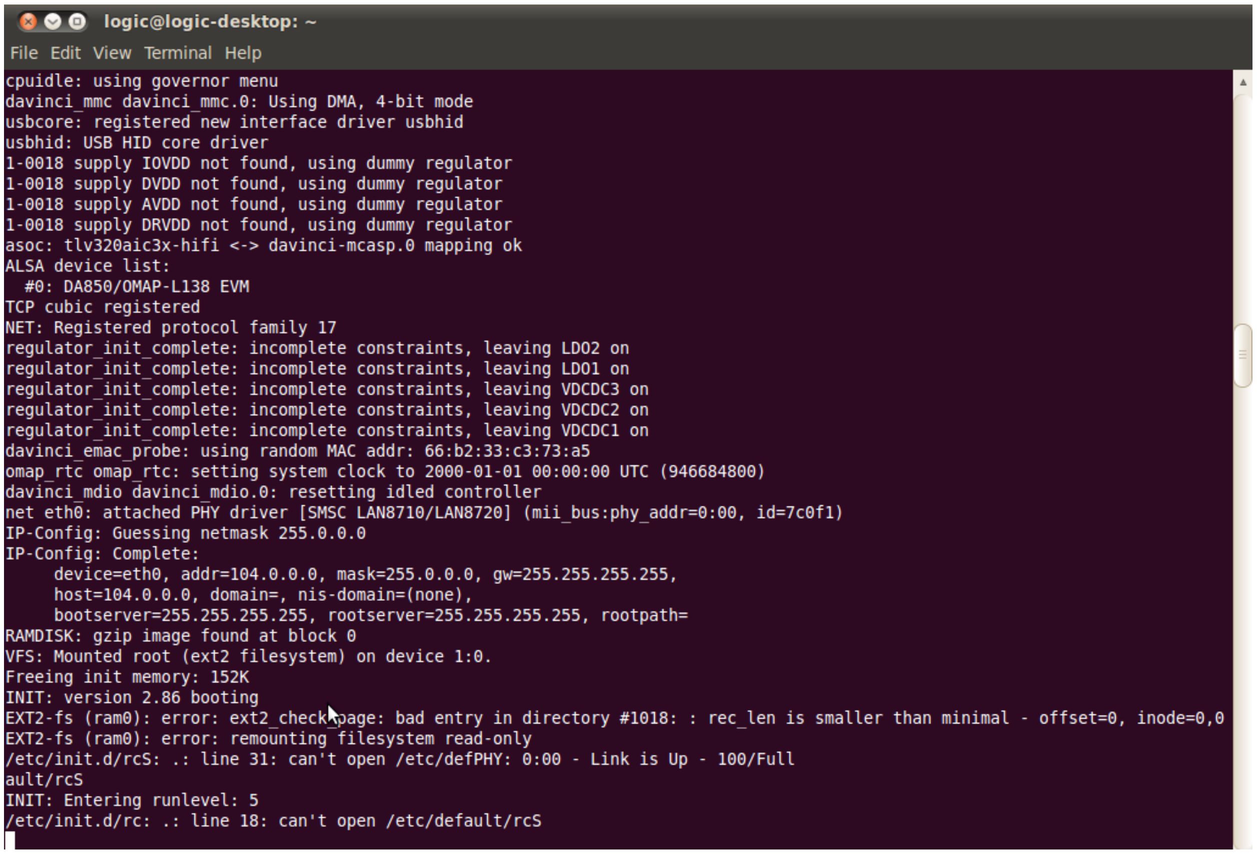


```
Erasing at 0x0 -- 100% complete.
Writing to Nand... done
U-Boot > printenv
baudrate=115200
bootargs=mem=32M console=ttyS2,115200n8 root=/dev/ram0 rw\
initrd=0xc1180000,4M ip=dhcp eth=${ethaddr}
bootcmd=nand read.e 0xc1180000 0x600000 0x400000; nboot.e 0xc0700000\
 0 0x200000; bootm
bootdelay=3
bootfile="uImage"
dnsip=192.168.120.1
ethact=DaVinci-EMAC
ethaddr=00:08:EE:03:5B:40
fileaddr=C1180000
filesize=303A47
ipaddr=192.168.120.200
serverip=192.168.120.156
stderr=serial
stdin=serial
stdout=serial
ver=U-Boot 2010.12 (Jun 13 2012 - 08:59:46)

Environment size: 504/131068 bytes
U-Boot >
```

5. Power off the EVM Development Kit.

6.3 Boot Linux Kernel from NAND as RAMdisk

1. Power on the EVM Development Kit. You should see the following output that shows the kit is booting from the NAND RAMdisk:

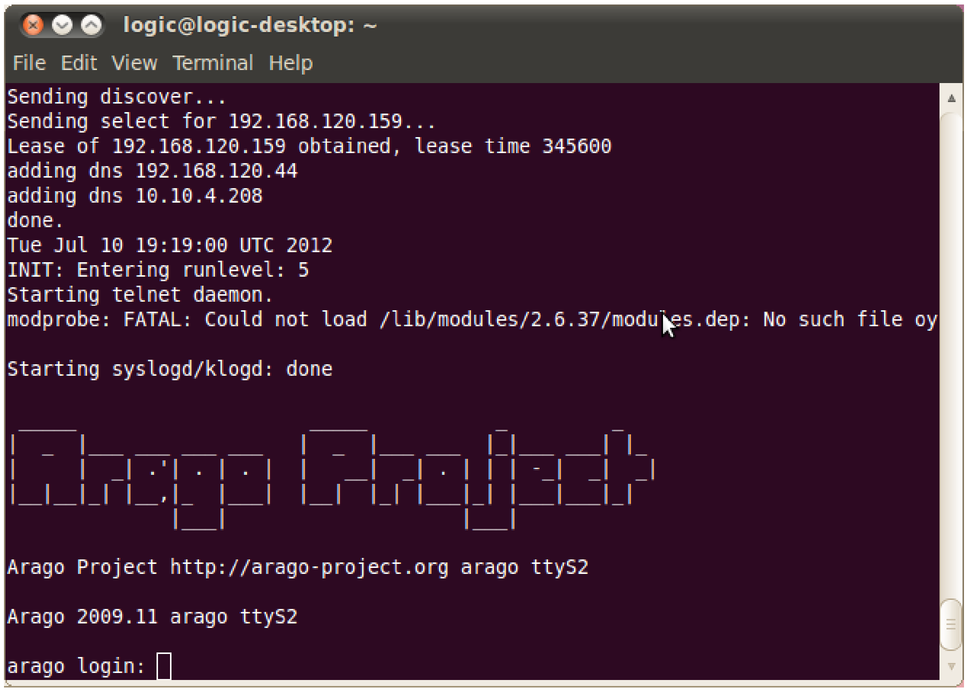


```
cpuidle: using governor menu
davinci_mmc davinci_mmc.0: Using DMA, 4-bit mode
usbcore: registered new interface driver usbhid
usbhid: USB HID core driver
1-0018 supply IOVDD not found, using dummy regulator
1-0018 supply DVDD not found, using dummy regulator
1-0018 supply AVDD not found, using dummy regulator
1-0018 supply DRVDD not found, using dummy regulator
asoc: tlv320aic3x-hifi <-> davinci-mcasp.0 mapping ok
ALSA device list:
  #0: DA850/OMAP-L138 EVM
TCP cubic registered
NET: Registered protocol family 17
regulator_init_complete: incomplete constraints, leaving LDO2 on
regulator_init_complete: incomplete constraints, leaving LDO1 on
regulator_init_complete: incomplete constraints, leaving VDCDC3 on
regulator_init_complete: incomplete constraints, leaving VDCDC2 on
regulator_init_complete: incomplete constraints, leaving VDCDC1 on
davinci_emac_probe: using random MAC addr: 66:b2:33:c3:73:a5
omap_rtc omap_rtc: setting system clock to 2000-01-01 00:00:00 UTC (946684800)
davinci_mdio davinci_mdio.0: resetting idled controller
net eth0: attached PHY driver [SMSC LAN8710/LAN8720] (mii_bus:phy_addr=0:00, id=7c0f1)
IP-Config: Guessing netmask 255.0.0.0
IP-Config: Complete:
     device=eth0, addr=104.0.0.0, mask=255.0.0.0, gw=255.255.255.255,
     host=104.0.0.0, domain=, nis-domain=(none),
     bootserver=255.255.255.255, rootserver=255.255.255.255, rootpath=
RAMDISK: gzip image found at block 0
VFS: Mounted root (ext2 filesystem) on device 1:0.
Freeing init memory: 152K
INIT: version 2.86 booting
EXT2-fs (ram0): error: ext2_check_page: bad entry in directory #1018: : rec_len is smaller than minimal - offset=0, inode=0,0
EXT2-fs (ram0): error: remounting filesystem read-only
/etc/init.d/rcS: .: line 31: can't open /etc/defPHY: 0:00 - Link is Up - 100/Full
ault/rcS
INIT: Entering runlevel: 5
/etc/init.d/rc: .: line 18: can't open /etc/default/rcS
```

2. Log in to the Linux environment.

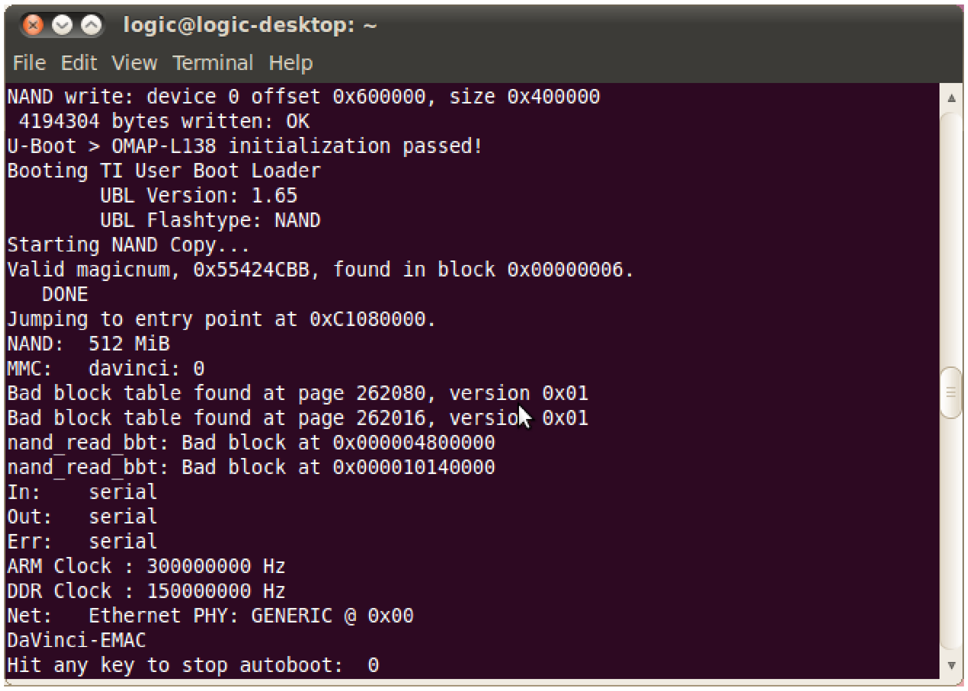
```
Arago login: root
```

You should see the following output:



```
Sending discover...
Sending select for 192.168.120.159...
Lease of 192.168.120.159 obtained, lease time 345600
adding dns 192.168.120.44
adding dns 10.10.4.208
done.
Tue Jul 10 19:19:00 UTC 2012
INIT: Entering runlevel: 5
Starting telnet daemon.
modprobe: FATAL: Could not load /lib/modules/2.6.37/modules.dep: No such file oy

Starting syslogd/klogd: done

 _____                    _____           _         _
|  _  |___ ___ ___ ___   |  _  |___ ___  |_|___ ___| |_
|     |  _| .'| . | . |  |   __|  _| . | | | -_|  _|  _|
|__|__|_| |__,|_  |___|  |__|  |_| |___|_| |___|___|_|
              |___|                    |___|

Arago Project http://arago-project.org arago ttyS2

Arago 2009.11 arago ttyS2

arago login:
```

3. Scroll up to where the EVM Development Kit powered on and confirm the boot source by checking that the *UBL Flashtype* is NAND, as seen below.



```
NAND write: device 0 offset 0x600000, size 0x400000
 4194304 bytes written: OK
U-Boot > OMAP-L138 initialization passed!
Booting TI User Boot Loader
        UBL Version: 1.65
        UBL Flashtype: NAND
Starting NAND Copy...
Valid magicnum, 0x55424CBB, found in block 0x00000006.
   DONE
Jumping to entry point at 0xC1080000.
NAND:  512 MiB
MMC:   davinci: 0
Bad block table found at page 262080, version 0x01
Bad block table found at page 262016, version 0x01
nand_read_bbt: Bad block at 0x000004800000
nand_read_bbt: Bad block at 0x000010140000
In:    serial
Out:   serial
Err:   serial
ARM Clock : 300000000 Hz
DDR Clock : 150000000 Hz
Net:   Ethernet PHY: GENERIC @ 0x00
DaVinci-EMAC
Hit any key to stop autoboot:  0
```

The EVM Development Kit has been configured to automatically boot Linux from NAND as a RAMdisk.

7 Boot Linux Kernel with JFFS2 Root Filesystem

This section will explain how to set up JFFS2 root filesystem in NAND flash. To load the JFFS2 filesystem image, the Linux kernel will boot up using a Network File System (NFS) from a Linux host PC. The JFFS2 image will then be loaded and saved into NAND using the Linux kernel booted from NFS. The NFS will be set up on a Linux host PC using the methods described in the *AM1808/OMAP-L138 Linux User Guide*.

7.1 Set NFS to Boot Linux Kernel

The Linux host PC and the U-Boot environment on the EVM Development Kit require configuration and environment variable setup to load the JFFS2 filesystem through NFS. Follow the steps below to complete this setup.

1. Follow the steps in Section 5 to establish a network connection and flash the kernel image into NAND.
2. Clear the environment variables to create a clean slate for the environment variables that will be entered in the steps below.

```
U-Boot> setenv bootcmd

U-Boot> setenv bootargs
```

3. Set up the root path for NFS exports.

```
U-Boot > setenv rootpath /opt/nfs-exports/am1808_omapl138
```

4. Set up a variable that holds any options to be passed to the kernel for mounting the root filesystem via NFS.

```
U-Boot > setenv nfsoptions ,wsize=1500,rsize=1500
```

NOTE: Be sure to include the commas in the command above.

5. Set up additional arguments.

```
U-Boot > setenv otherbootargs ignore_loglevel
```

6. Set up the DaVinci EMAC device for NFS.

```
U-Boot> setenv ethact DaVinci-EMAC
```

7. Set the *bootfile* name.

```
U-Boot> setenv bootfile uImage
```

8. Set up the NFS root filesystem variable.

```
U-Boot> setenv rootfsfile rootfs-base.tar.gz
```

9. Set the root file size to 0x300000, which will cover the current root file sizes.

```
U-Boot> setenv rootfssize 0x300000
```

10. Set up the base load address in SDRAM.

```
U-Boot> setenv loadaddr 0xC0700000
```

11. Set up a variable for the console.

```
U-Boot> setenv consoledev ttyS2,115200n8
```

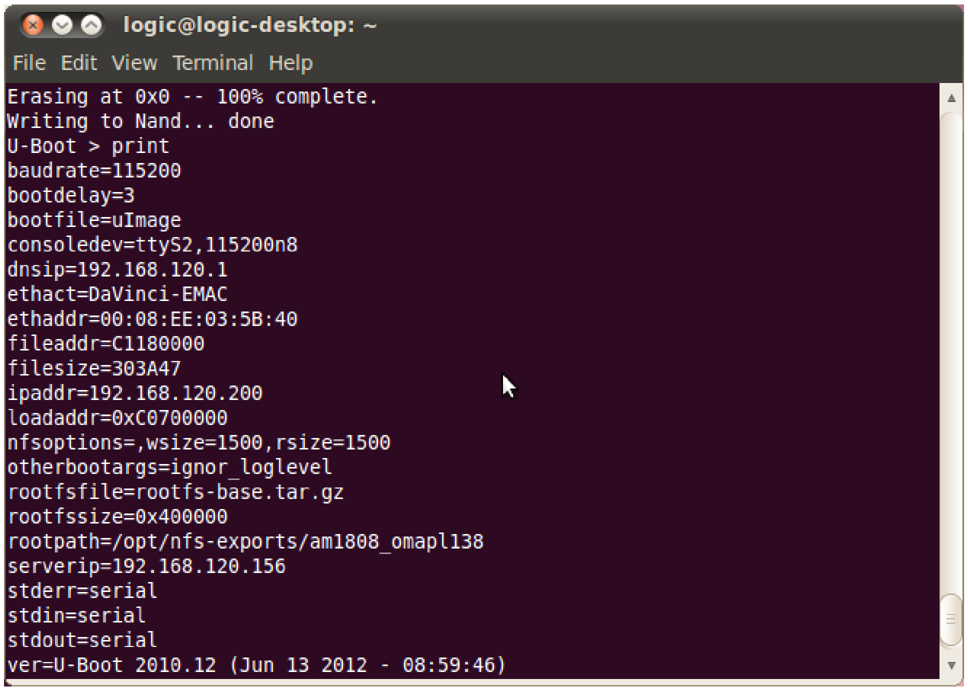
12. Save these environment settings so they persist after a power cycle on the EVM Development Kit.

```
U-Boot > saveenv
```

13. Use the *printenv* command to verify the commands were entered correctly.

```
U-Boot > printenv
```

You should see the following output:



```
Erasing at 0x0 -- 100% complete.
Writing to Nand... done
U-Boot > print
baudrate=115200
bootdelay=3
bootfile=uImage
consoledev=ttyS2,115200n8
dnsip=192.168.120.1
ethact=DaVinci-EMAC
ethaddr=00:08:EE:03:5B:40
fileaddr=C1180000
filesize=303A47
ipaddr=192.168.120.200
loadaddr=0xC0700000
nfsoptions=,wsize=1500,rsize=1500
otherbootargs=ignor_loglevel
rootfsfile=rootfs-base.tar.gz
rootfssize=0x400000
rootpath=/opt/nfs-exports/am1808_omapl138
serverip=192.168.120.156
stderr=serial
stdin=serial
stdout=serial
ver=U-Boot 2010.12 (Jun 13 2012 - 08:59:46)
```

14. Create the *nfsboot* variable that will activate all of the NFS U-Boot environment variables in one place.

```
U-Boot> setenv nfsboot 'setenv bootargs console=${consoledev}
root=/dev/nfs rw nfsroot=${serverip}:${rootpath}${nfsoptions} ip=dhcp
eth=${ethaddr} ${otherbootargs}; tftp ${loadaddr} ${bootfile}; bootm'
```

NOTE: Be sure to include the single apostrophe at the beginning and end of the command above. Also, be sure that the variables are surrounded by curly brackets and not parentheses.

15. Save these environment settings so they persist after a power cycle on the EVM Development Kit.

```
U-Boot > saveenv
```

7.2 Download JFFS2 Root Filesystem to RAM and Save to NAND

A memory technology device (mtd) block partition must be selected for placement of the JFFS2 filesystem. The NAND flash is made up of blocks and does not have direct addressing. By comparison, the SDRAM can be accessed with specific addresses within the SDRAM memory space. With Linux running from RAMdisk or SPI flash, the NAND block partition designations can be observed with the following command:

```
root@arago:/# cat /proc/mtd
```

You should see the following output:

```
dev:    size   erasesize  name
mtd0: 00020000 00020000 "u-boot env"
mtd1: 00020000 00020000 "UBL"
mtd2: 00080000 00020000 "u-boot"
mtd3: 00400000 00020000 "kernel"
mtd4: 1fa00000 00020000 "filesystem"
mtd5: 00010000 00010000 "UBL"
mtd6: 00080000 00010000 "U-Boot"
mtd7: 00010000 00010000 "U-Boot-Env"
mtd8: 00280000 00010000 "Kernel"
mtd9: 00400000 00010000 "Filesystem"
mtd10: 00010000 00010000 "MAC-Address"
```

To interpret the block partitions, an excerpt from the Linux boot-up log is included below that describes each of the block partition designations. The first set of partitions (mtd0 through mtd4) is created for *davinci_nand.1* NAND partitions. The second set of partition designations (mtd5 through mtd10) is for *m25p80 spi1.0* SPI flash partitions (onboard the SOM).

```
ONFI flash detected
ONFI param page 0 valid
NAND device: Manufacturer ID: 0x2c, Chip ID: 0xdc (Micron NAND 512MiB
3,3V 8-bit)
Creating 5 MTD partitions on "davinci_nand.1":
0x000000000000-0x000000020000 : "u-boot env"
```

```
0x000000020000-0x000000040000 : "UBL"
0x000000040000-0x0000000c0000 : "u-boot"
0x000000200000-0x000000600000 : "kernel"
0x000000600000-0x000020000000 : "filesystem"
```

Based on the information above, the mtd4 block partition will be used for the JFFS2 root filesystem in the commands below.

1. Start the Linux kernel (*uImage*) using *nfsboot* on the EVM Development Kit.

```
U-Boot> run nfsboot
```

2. From the EVM Development Kit terminal window, download the JFFS2 filesystem from the NFS that we've set up. Be sure to use a value that corresponds to the IP address of your Linux host PC.

```
root@arago:/# tftp -r rootfs-base.jffs2 -g <ip_address_of_tftp_server>
```

   This command takes about one minute to run. The Linux prompt is then returned.

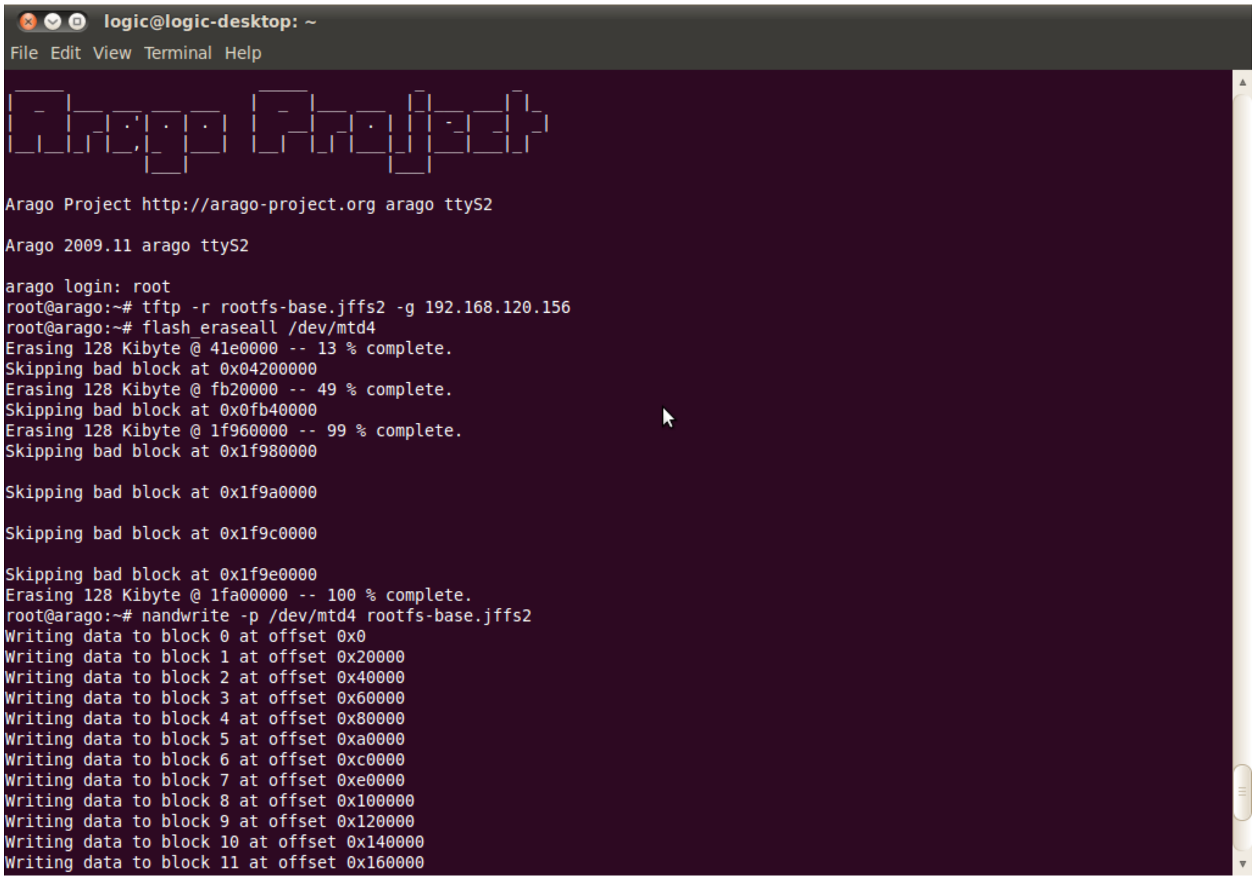
3. Next, erase the NAND partition mtd4 that was chosen earlier.

```
root@arago:/# flash_eraseall /dev/mtd4
```

4. Write the JFFS2 file system to the mtd4 partition using NFS.

```
root@arago:/# nandwrite -n -p /dev/mtd4 rootfs-base.jffs2
```

You should see the following output:



```
 _____                    _____           _         _
|  _  |___ ___ ___ ___   |  _  |___ ___  |_|___ ___| |_
|     |  _| .'| . | . |  |   __|  _| . | | | -_|  _|  _|
|__|__|_| |__,|_  |___|  |__|  |_| |___|_| |___|___|_|
              |___|                    |___|

Arago Project http://arago-project.org arago ttyS2

Arago 2009.11 arago ttyS2

arago login: root
root@arago:~# tftp -r rootfs-base.jffs2 -g 192.168.120.156
root@arago:~# flash_eraseall /dev/mtd4
Erasing 128 Kibyte @ 41e0000 -- 13 % complete.
Skipping bad block at 0x04200000
Erasing 128 Kibyte @ fb20000 -- 49 % complete.
Skipping bad block at 0x0fb40000
Erasing 128 Kibyte @ 1f960000 -- 99 % complete.
Skipping bad block at 0x1f980000

Skipping bad block at 0x1f9a0000

Skipping bad block at 0x1f9c0000

Skipping bad block at 0x1f9e0000
Erasing 128 Kibyte @ 1fa00000 -- 100 % complete.
root@arago:~# nandwrite -p /dev/mtd4 rootfs-base.jffs2
Writing data to block 0 at offset 0x0
Writing data to block 1 at offset 0x20000
Writing data to block 2 at offset 0x40000
Writing data to block 3 at offset 0x60000
Writing data to block 4 at offset 0x80000
Writing data to block 5 at offset 0xa0000
Writing data to block 6 at offset 0xc0000
Writing data to block 7 at offset 0xe0000
Writing data to block 8 at offset 0x100000
Writing data to block 9 at offset 0x120000
Writing data to block 10 at offset 0x140000
Writing data to block 11 at offset 0x160000
```

5. Cycle the power on the EVM Development Kit by powering it off and back on.
6. Set up the *bootcmd* environment variable to boot a JFFS2 filesystem out of NAND.

```
U-Boot > setenv bootcmd 'nand read.e 0xC1180000 0x400000 0x400000;
nboot.e 0xC0700000 0 0x200000; bootm'
```

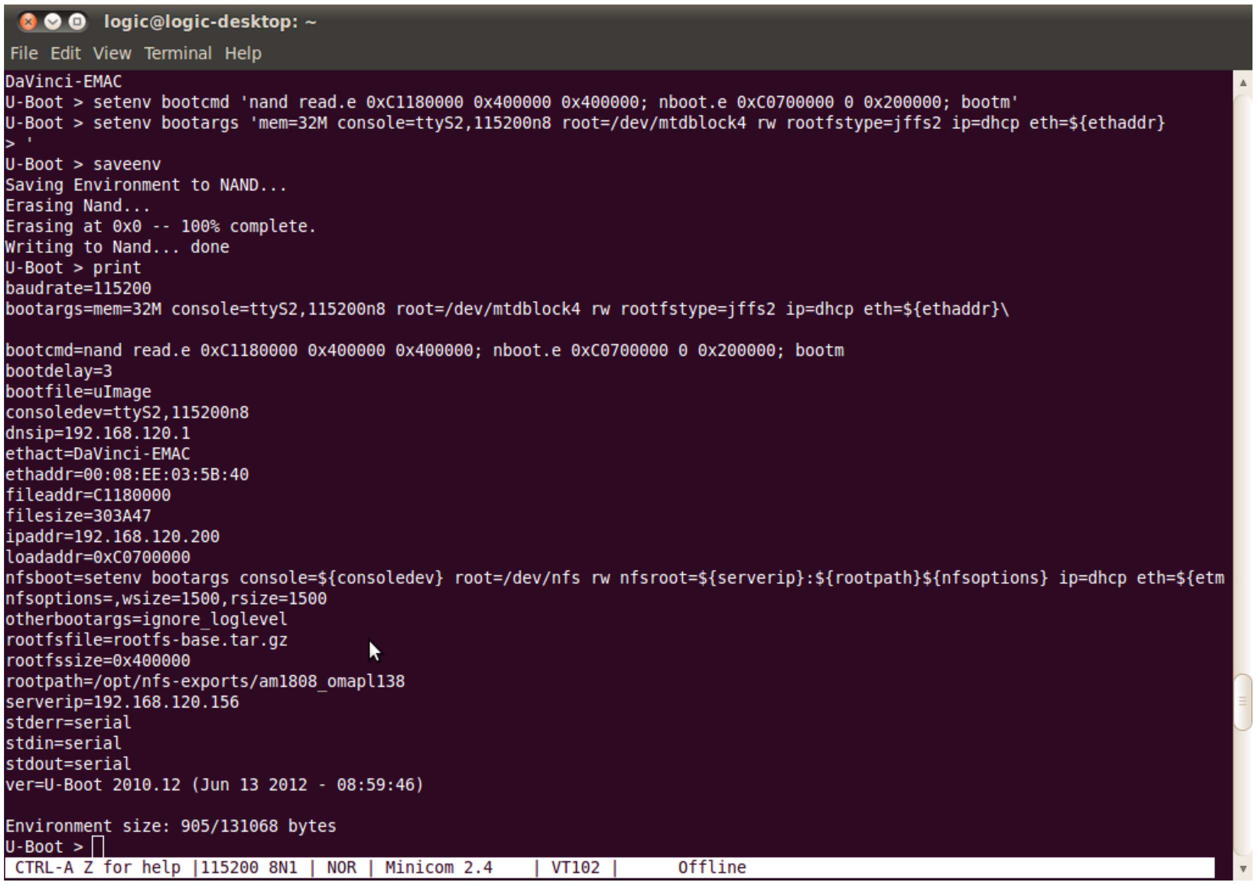
7. Set up the *bootargs* environment variable to boot a JFFS2 filesystem out of NAND. Note that the *bootargs* root variable is set to mtdblock4 instead of the mtd4 value used previously.

```
U-Boot > setenv bootargs 'mem=32M console=ttyS2,115200n8
root=/dev/mtdblock4 rw rootfstype=jffs2  ip=dhcp eth=${ethaddr}'
```

8. Save these values to persist after a power cycle.

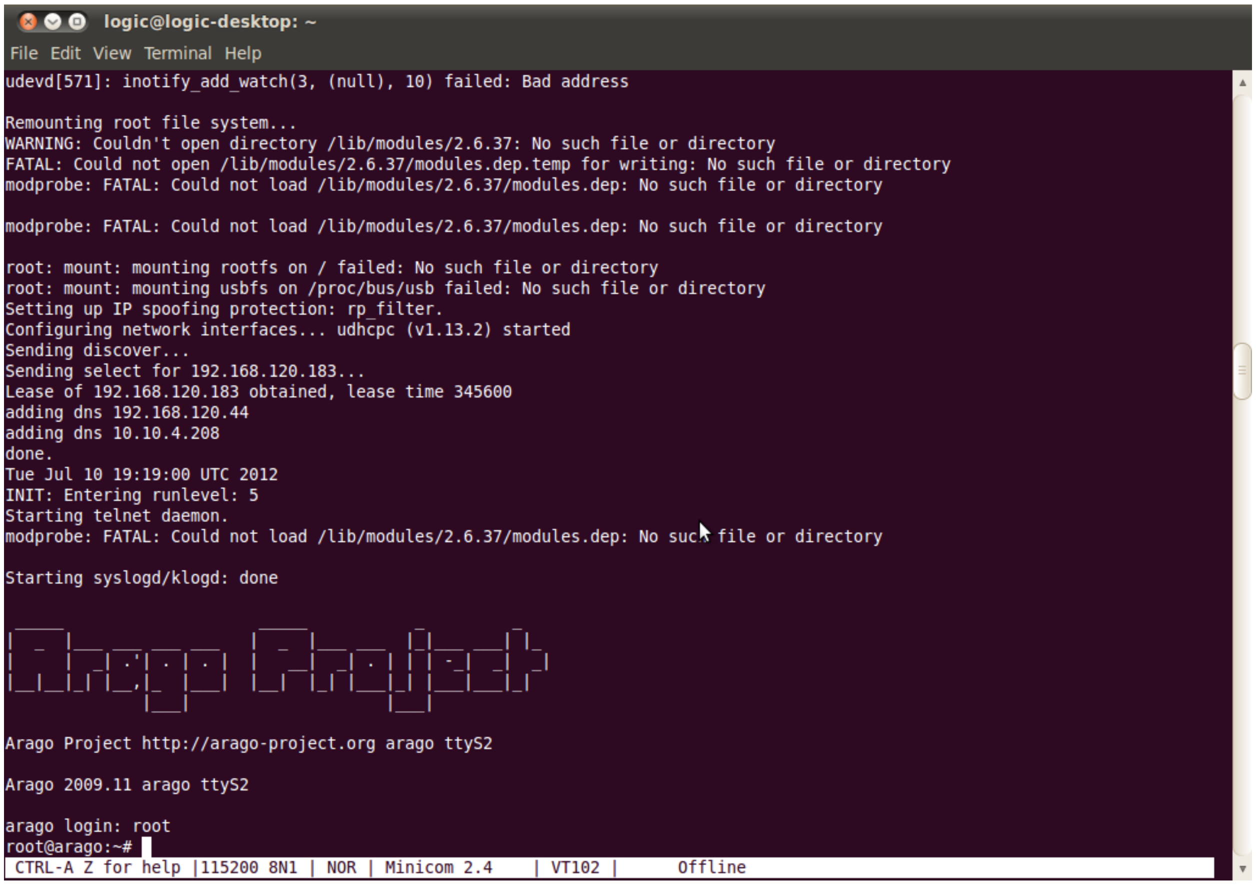
```
U-Boot > saveenv
```

You should see the following output:



```
DaVinci-EMAC
U-Boot > setenv bootcmd 'nand read.e 0xC1180000 0x400000 0x400000; nboot.e 0xC0700000 0 0x200000; bootm'
U-Boot > setenv bootargs 'mem=32M console=ttyS2,115200n8 root=/dev/mtdblock4 rw rootfstype=jffs2 ip=dhcp eth=${ethaddr}
> '
U-Boot > saveenv
Saving Environment to NAND...
Erasing Nand...
Erasing at 0x0 -- 100% complete.
Writing to Nand... done
U-Boot > print
baudrate=115200
bootargs=mem=32M console=ttyS2,115200n8 root=/dev/mtdblock4 rw rootfstype=jffs2 ip=dhcp eth=${ethaddr}\

bootcmd=nand read.e 0xC1180000 0x400000 0x400000; nboot.e 0xC0700000 0 0x200000; bootm
bootdelay=3
bootfile=uImage
consoledev=ttyS2,115200n8
dnsip=192.168.120.1
ethact=DaVinci-EMAC
ethaddr=00:08:EE:03:5B:40
fileaddr=C1180000
filesize=303A47
ipaddr=192.168.120.200
loadaddr=0xC0700000
nfsboot=setenv bootargs console=${consoledev} root=/dev/nfs rw nfsroot=${serverip}:${rootpath}${nfsoptions} ip=dhcp eth=${etm
nfsoptions=,wsize=1500,rsize=1500
otherbootargs=ignore_loglevel
rootfsfile=rootfs-base.tar.gz
rootfssize=0x400000
rootpath=/opt/nfs-exports/am1808_omapl138
serverip=192.168.120.156
stderr=serial
stdin=serial
stdout=serial
ver=U-Boot 2010.12 (Jun 13 2012 - 08:59:46)

Environment size: 905/131068 bytes
U-Boot >
 CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.4    | VT102 |      Offline
```

9. Cycle the power on the EVM Development Kit. The kit should boot to the Linux kernel using the JFFS2 filesystem out of NAND.



```
udevd[571]: inotify_add_watch(3, (null), 10) failed: Bad address

Remounting root file system...
WARNING: Couldn't open directory /lib/modules/2.6.37: No such file or directory
FATAL: Could not open /lib/modules/2.6.37/modules.dep.temp for writing: No such file or directory
modprobe: FATAL: Could not load /lib/modules/2.6.37/modules.dep: No such file or directory

modprobe: FATAL: Could not load /lib/modules/2.6.37/modules.dep: No such file or directory

root: mount: mounting rootfs on / failed: No such file or directory
root: mount: mounting usbfs on /proc/bus/usb failed: No such file or directory
Setting up IP spoofing protection: rp_filter.
Configuring network interfaces... udhcpc (v1.13.2) started
Sending discover...
Sending select for 192.168.120.183...
Lease of 192.168.120.183 obtained, lease time 345600
adding dns 192.168.120.44
adding dns 10.10.4.208
done.
Tue Jul 10 19:19:00 UTC 2012
INIT: Entering runlevel: 5
Starting telnet daemon.
modprobe: FATAL: Could not load /lib/modules/2.6.37/modules.dep: No such file or directory

Starting syslogd/klogd: done

 _____                    _____           _         _
|  _  |___ ___ ___ ___   |  _  |___ ___  |_|___ ___| |_
|     |  _| .'| . | . |  |   __|  _| . | | | -_|  _|  _|
|__|__|_| |__,|_  |___|  |__|  |_| |___|_| |___|___|_|
              |___|                    |___|

Arago Project http://arago-project.org arago ttyS2

Arago 2009.11 arago ttyS2

arago login: root
root@arago:~#
 CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.4    | VT102 |      Offline
```

The EVM Development kit has been configured to automatically boot Linux from NAND with a JFFS2 filesystem.

8 Summary

This application note has provided the steps to program and boot the NAND flash on the UI Board of the EVM Development Kit for use with the SOM-M1. The procedures for booting Linux from NAND with a RAMdisk filesystem or a JFFS2 filesystem were also provided.

Appendix A and Appendix B of this document provide instructions for rebuilding U-Boot and the *uImage* and *rootfs* files for NAND.

Appendix A: Rebuild U-Boot for NAND Operation

If you are using the VM, it already has a pre-built U-Boot binary for NAND operation, and this section is unnecessary. If you are not using the VM, U-Boot needs to be rebuilt in order to be loaded into the NAND flash of the EVM Development Kit. The original U-Boot file is included in the DaVinci Linux PSP v3.21.00.04 source files and is built to be loaded onto the SPI flash of the SOM-M1.

The procedures in this section are performed on a Linux host PC. If you have not already done so, ensure your host PC has been set up with a Linux environment as outlined in the *AM1808/OMAP-L138 Linux User Guide* before completing this section. Also ensure you have obtained the Mentor Graphics Sourcery tools outlined in "Section 4.5: Download and Install Mentor Graphics Sourcery Tools" of the same document.

The *$XXXX* variables are defined in the *setenv.sh* file in the *AM1808_OMAP-L138* directory of the Linux directory structure. The variable definitions are created in the *AM1808/OMAP-L138 Linux User Guide* and are activated in a command window in the *AM1808_OMAP-L138* directory with the following command:

```
host$ source setenv.sh
```

The following steps prepare the configurations that are needed for the U-Boot rebuild.

1. Switch to the directory that contains the DaVinci Linux PSP v3.21.00.04 software. Be sure to include the path appropriate for your system.

```
host$ cd "PATH/TO/SOFTWARE"
```

2. Unzip the DaVinci Linux PSP 3.21.00.04 source files.

```
host$ tar -xzf DaVinci-PSP-SDK-03.21.00.04.tgz
```

3. Locate the U-Boot files in the directory *src/u-boot*.
4. Extract the source files for U-Boot from *src/u-boot/u-boot-03.21.00.04.tar.gz* from a command window in the */u-boot* directory.

```
host$ tar -xvzf u-boot-03.21.00.04.tar.gz -C $UBOOT
```

NOTE: The U-Boot patches are already included in the PSP U-Boot files, so they do not need to be applied.

5. Change the directory location to the *U-Boot* directory created in the last extraction.
6. Next, you will need to update the *da850evm.h* file to configure the *u-boot.bin* file for NAND operation. This file is located here:

   AM1808/U-Boot/u-boot-03.21.00.04/include/configs

7. Edit the *da850evm.h* file as follows:
   a. On line 30, change *#define CONFIG_USE_SPIFLASH* to *#undef CONFIG_USE_SPIFLASH*.
   b. On line 32, change *#undef CONFIG_USE_NAND* to *#define CONFIG_USE_NAND*.
8. Save the edits to this file.

9. On your Linux host PC, open a command window in the *U-Boot* directory created above.
10. Confirm that the Mentor Graphics Sourcery tools *$PATH* variable is included in the PATH configuration. See "Section 4.5: Download and Install Mentor Graphics Sourcery Tool" in the *AM1808/OMAP-L138 Linux User Guide* for additional information on how to do this.
11. Clean the file locations.

```
host$ make distclean CROSS_COMPILE=arm-none-linux-gnueabi-
```

12. Rebuild the *u-boot.bin* file.

```
host$ make da850evm_config CROSS_COMPILE=arm-none-linux-gnueeabi-
host$ make all CROSS_COMPILE=arm-none-linux-guneabi-
```

13. Copy the compiled *u-boot.bin* file to the *C:\flasher\SFH_2_40\OMAP-L138\GNU* directory to be loaded onto the EVM Development Kit NAND flash as described in Section 3 of this document.

Appendix B: Rebuild Linux Kernel and Root Filesystem for NAND Operation

If you are using the VM, it already has pre-built kernel and root filesystem binaries for NAND operation, and this section is unnecessary. If you are not using the VM, the Linux kernel image *uImage* and root filesystem need to be rebuilt in order to be loaded into the NAND flash of the EVM Development Kit. The original files are included in the DaVinci Linux PSP 3.21.00.04 source files and are built to be loaded onto the SPI flash of the SOM-M1.

The procedures in this section are performed on a Linux host PC. If you have not already done so, ensure your host PC has been set up with a Linux environment as outlined in the *AM1808/OMAP-L138 Linux User Guide* before completing this section.

1. The instructions for building the Linux kernel can be found in "Section 5.2: Build the Kernel" of the *AM1808/OMAP-L138 Linux User Guide*. Section 5.2.2, Step 4 has a command to run *menuconfig* on the Linux host PC. In order to make a *uImage* file suitable for NAND operation, make the following device driver changes to that step:

```
Device Drivers --->

    < > MMC/SD/SDIO card support  --->

    <*> Memory Technology Device (MTD)  support --->
        [*]   MTD partitioning support
        [*]   Command line partition table parsing
        <*>   Direct char device access to MTD devices
        <*>   Common interface to block layer for MTD 'translation
layers'
        <*>   Caching block device access to MTD devices
        <*>   NAND Device Support --->
            <*> Support NAND on DaVinci SoC
```

   WARNING: The MMC/SD support must be disabled when NAND is selected for use. These two functions are pin multiplexed together and will not operate at the same time.

   When complete, you will have a *uImage* file that will function in NAND operations.

2. Build the kernel file.

```
host$  make uImage ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi-
```

3. Place the *uImage* file into the directories below for later use.

```
host$ cp arch/arm/boot/uImage $IMAGES
host$ cp arch/arm/boot/uImage $TFTPBOOT
```

4. If *$IMAGES* and *$TFTPBOOT* are undefined, return to the *AM1808/OMAP-L138 Linux User Guide* for instructions on how to define these terms in the *setenv.sh* file. The *setenv.sh* file is located in the *AM1808_OMAP-L138* directory of the Linux directory structure.

   NOTE: In Linux, to activate defined terms and paths (e.g., *$TFTPBOOT*), the command window must "source" the file that holds the definitions. In this case, use the following command from the command window in the *AM1808_OMAP-L138* directory:

```
host$ source setenv.sh
```

5. The three root filesystems are included in the download with this document. These three files are built using the instructions in "Section 5.3: Download, Prepare, and Build Arago Root Filesystem" in the *AM1808/OMAP-L138 Linux User Guide*. After this build, the root filesystem files are placed here in the Linux directory structure:

```
$ROOTFS
      +- arago
      +- arago-bitbake
      +- arago-deploy
            +- images
                  +- arago
                        +- *.ext2.gz (init RAMFS)
                        +- *.jffs2 (JFFS2 image)
                        +- *.tar.gz (for NFS)
      +- arago-oe-dev
      +- arago-tmp (arago's working directory)
      +- downloads (downloaded packages are cached here)
```

6. To simplify the commands, change the names of the files.

```
host$ cp arago-base-image-glibc-ipk-2009.11-arago.rootfs.ext2 rootfs-
arago-ram.ext2.gz
host$ cp arago-base-image-glibc-ipk-2009.11-arago.rootfs.jffs2  rootfs-
base.jffs2
host$ cp arago-base-image-glibc-ipk-2009.11-arago.rootfs.tar.gz …
```

   NOTE: The shorter file names are used in the commands of this document.

7. Copy the JFFS2 file into the */tftpboot* directory for later use.

```
host$ cp rootfs-base.jffs2 $TFTPBOOT/rootfs-base.jffs2
```

   NOTE: If *$TFTPBOOT* is undefined, use the following command from the command window in the *AM1808_OMAP-L138* directory:

```
host$ source setenv.sh
```