

# **PROGRAM EVM IMAGES**

---

## **Users Guide**

Publication Date: March 08, 2012  
Ver 1.7

## Document License

---

This work is licensed under the Creative Commons Attribution-NoDerivs 3.0 Unported License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nd/3.0/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

Copyright (C) 2011-12 Texas Instruments Incorporated - <http://www.ti.com>

---

© Copyright 2011 Texas Instruments, Inc.  
All Rights Reserved

## Contents

---

1	Overview .....	1
2	Revision History .....	1
3	Files Provided .....	1
3.1	C6678 EVM Files .....	1
3.2	C6670 and TCI6618 EVM Files .....	2
4	MD5SUM utility used .....	2
5	Device Support.....	2
6	Directory Structure .....	3
7	Programming the bin files.....	4
7.1	Set the EVM Dip switches .....	4
7.2	Set the Environment Variables .....	4
7.2.1	Windows .....	4
7.2.2	Linux .....	5
7.3	Copy the custom GEL files .....	6
7.4	DSS Script Arguments.....	6
7.5	Executing the DSS script to restore factory default images.....	6
7.5.1	Windows .....	6
7.5.2	Linux .....	7
7.5.3	Sample DSS Script output for Windows and Linux .....	7
4.	Verification .....	14
4.1.	Serial Port Setup .....	14
4.2.	Verifying POST.....	14
4.2.1.	Entering Serial Number to the EVM.....	15
4.3.	Verifying NOR .....	15
4.4.	Verifying NAND .....	16



# BIOS MCSDK RECOVERING FACTORY DEFAULT IMAGES

---

---

## 1 Overview

This release provides the images for the factory to program on the eeprom, nand and nor for EVM6670L, EVM6678L and EVM6618L.

## 2 Revision History

Revision	Details
1.0	Initial Version
1.1	Rearranged the sections
1.2	Updating the Linux instructions.
1.3	Added C6670 EVM support
1.4	Updated scripts and binaries
1.7	Updated to CCS 5.1

## 3 Files Provided

### 3.1 C6678 EVM Files

The following files are the factory default images under `program_evm\binaries\evm6678l`.

Note: Please note that `nand.bin` and `nand_oob.bin` are not supplied from BIOS MCSDK releases. Please refer to the DVD shipped along with the EVM for `nand.bin` file.

File Name	Description
eeeprom50.bin	Binary file Power On Self Test (POST)
eeeprom51.bin	Binary file for IBL
eeepromwriter_evm6678l.out	eeeprom Writer DSP executable
eeepromwriter_input.txt	eeeprom writer input for execution
eeepromwriter_input50.txt	eeeprom writer input for writing images to 0x50
eeepromwriter_input51.txt	eeeprom writer input for writing images to 0x51
nand_writer_input.txt	nand image writer input file
nor.bin	Binary file for the NOR image having HUA demo
norwriter_evm6678l.out	NOR image writer DSP executable
nor_writer_input.txt	NOR image writer input file

### 3.2 C6670 and TCI6618 EVM Files

C6670 EVM and TCI6618 EVM use the same factory default images under program\_evm\binaries\evm6670l.

File Name	Description
eeeprom50.bin	Binary file Power On Self Test (POST)
eeeprom51.bin	Binary file for IBL
eeepromwriter_evm6670l.out	eeeprom Writer DSP executable
eeepromwriter_input.txt	eeeprom writer input for execution
eeepromwriter_input50.txt	eeeprom writer input for writing images to 0x50
eeepromwriter_input51.txt	eeeprom writer input for writing images to 0x51
nand_writer_input.txt	nand image writer input file
nor.bin	Binary file for the NOR image having HUA demo
norwriter_evm6670l.out	NOR image writer DSP executable
nor_writer_input.txt	NOR image writer input file

## 4 MD5SUM utility used

Please use the md5sum utility from the below link

<http://www.pc-tools.net/files/win32/freeware/md5sums-1.2.zip>

## 5 Device Support

- The images provided support EVM6678L and EVM6670L/EVM6618L in Little Endian Mode.

## 6 Directory Structure

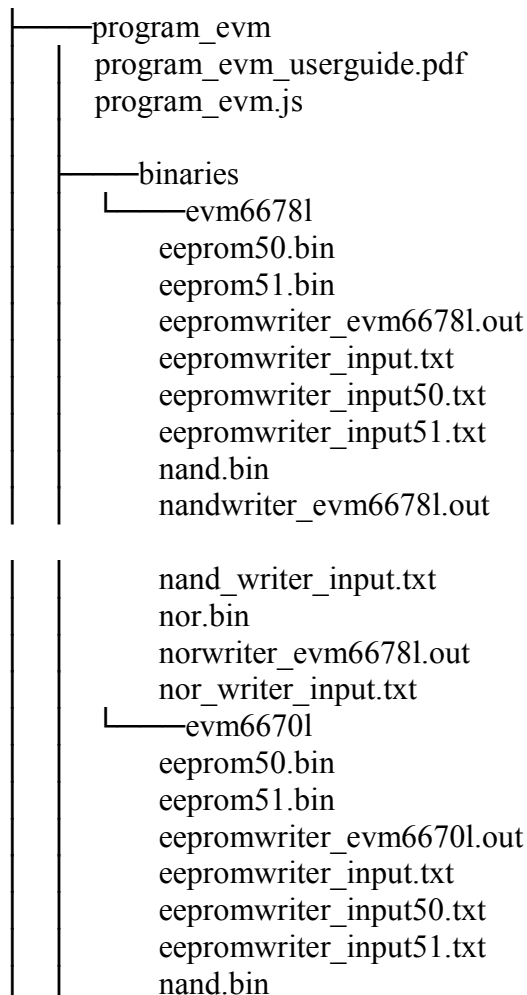
The `program_evm` (top-level) directory is intended to hold the \*DSS\* script for the Code Composer Studio which programs the default images to NAND/NOR/EEPROM.

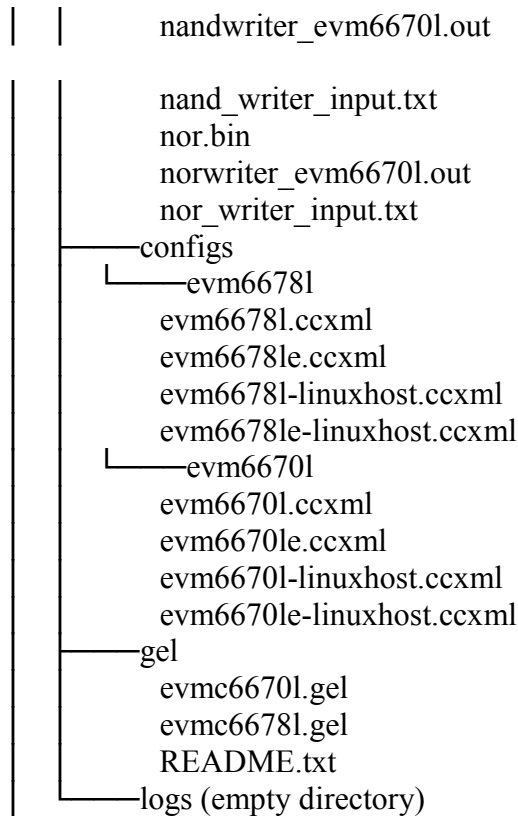
The `binaries/evm667xl` directory is intended to hold all the factory default images and the respective writers.

The `configs/evm667xl` directory is intended to hold the “CCS target configuration files”. Four pre-configured configurations are provided for each EVM type: one for inbuilt XDS100 in Windows, one for XDS560 mezzanine card in Windows, one for inbuilt XDS100 in Linux, and one for XDS560 mezzanine card in Linux.

The `gel` directory holds custom GEL files for BIOS-MCSDK release. It also contains a `README.txt` for the gel files’ usage.

The `logs` directory is empty and will be used to store logs. Logs are automatically generated when using `program_evm.js` to flash `evm667xl` devices.





## 7 Programming the bin files

This section assumes you have installed BIOS-MCSDK and Code Composer Studio.

### 7.1 Set the EVM Dip switches

Make sure the EVM dip switches are kept as below.

SWITCH	Pin1	Pin2	Pin3	Pin4
SW3	Off	On	On	On
SW4	On	On	On	On
SW5	On	On	On	On
SW6	On	On	On	On

### 7.2 Set the Environment Variables

Please make sure the below environment variables needs to be set. Otherwise there could be some unexpected behavior experienced.

#### 7.2.1 Windows

1. Set the **DSS\_SCRIPT\_DIR** environment variable (Mandatory) to Code Composer Studio scripting bin directory.

Example:



```
set DSS_SCRIPT_DIR="C:\ti\ccsv5\ccs_base\scripting\bin"
```

2. Set the **PROGRAM\_EVM\_TARGET\_CONFIG\_FILE** environment variable:  
Please provide the path for the ccxml file that is created for the EVM for the CCS.  
Please note that this step can be skipped for CCS 5.1.0 users since the program evm has a default target configuration files bundled for xds100 and xds560 mezzanine cards.  
CCS 5.0.3 users need to set this variable to get the program\_evm script functional.  
Otherwise unexpected results can happen.

Example:

```
set PROGRAM_EVM_TARGET_CONFIG_FILE=C:\Documents and Settings\  
\user\CCSTargetConfigurations\myC667xl.ccxml
```

NOTE: Please note that

[1] The `DSS_SCRIPT_DIR` env variable needs the opening/closing quotes,

[2] The `PROGRAM_EVM_TARGET_CONFIG_FILE` variable, on the other hand, **\*should not\*** have the quotes around it if the path has spaces in it.

If `PROGRAM_EVM_TARGET_CONFIG_FILE` environment variable is not set, the DSS script will use the default ccxml files that support the following emulators for CCS 5.1.0:

1. xds100 inbuilt (evm667xl.ccxml)
2. xds560 mezzanine card (evm667xle.ccxml)

Please note that depending on the emulator selected the restore image time may vary. For example, if xds100 inbuilt emulator is selected, the entire process may take over 60 minutes. If xds560 mezzanine card emulator is selected, the process may take about 10 minutes.

## 7.2.2 Linux

1. Set the **DSS\_SCRIPT\_DIR** environment variable (Mandatory) to your Code Composer Studio scripting bin directory.

Example:

```
export DSS_SCRIPT_DIR=~/.ti/ccsv5/ccs_base/scripting/bin
```

2. Set the **PROGRAM\_EVM\_TARGET\_CONFIG\_FILE** environment variable  
Please provide the path for the ccxml file that is created for the EVM for the CCS.  
Please note that this step can be skipped for CCS 5.1.0 users since the program evm has a default target configuration files bundled for xds100 and xds560 mezzanine cards.  
CCS 5.0.3 users need to set this variable to get the program\_evm script functional.  
Otherwise unexpected results can happen

Example:

```
export PROGRAM_EVM_TARGET_CONFIG_FILE =configs/evm667xl/my_evm667xl.ccxml
```

If `PROGRAM_EVM_TARGET_CONFIG_FILE` environment variable is not set, the DSS script will use the default ccxml files that support the following emulators for CCS 5.1.0:

3. xds100 inbuilt (evm667xl.ccxml)
4. xds560 mezzanine card (evm667xle.ccxml)

Please note that depending on the emulator selected the restore image time may vary. For example, if xds100 inbuilt emulator is selected, the entire process may take over 60 minutes. If xds560 mezzanine card emulator is selected, the process may take about 10 minutes.

### 7.3 Copy the custom GEL files

Please refer to the README.txt in the program\_evm\gel directory.

### 7.4 DSS Script Arguments

Script Usage:

```
[MCSDK]\tools\program_evm>%DSS_SCRIPT_DIR%\dss.bat program_evm.js  
[tmdx|tmds]evm(6678|6670|6618) [l|le] [-le|-be]
```

**MCSDK:** refers to the MCSDK installation directory, eg. C:\Program Files\Texas Instruments\mcSDK\_2\_00\_06\_18\

**tmdx:** TMDX type EVM

**tmds:** TMDS type EVM

**6678:** C6678 device

**6670:** C6670 device

**6618:** TCI6618 device

**l:** Low cost EVM

**le:** EVM uses 560 Mezzanine Emulator daughter card

**-le:** Little Endian

**-be:** Big Endian

### 7.5 Executing the DSS script to restore factory default images.

#### 7.5.1 Windows

1. cd “\program\_evm” directory
2. Set the necessary environment variables as described in section 7.2.1.
3. Using the DSS Script batch file, run the “program\_evm.js” script command from program\_evm directory.

**Example:**

```
\program_evm>%DSS_SCRIPT_DIR%\dss.bat program_evm.js TMDXEVM6678L-le
```

This will write all the little endian images to C6678 low cost EVM using XDS 100 emulator.

```
\program_evm>%DSS_SCRIPT_DIR%\dss.bat program_evm.js TMDXEVM6670Le-le
```

This will write all the little endian images to C6670 low cost EVM using XDS 560 Mezzanine emulator.

### 7.5.2 Linux

1. cd “program\_evm” directory
2. Set the necessary environment variables as described under section 7.2.2.
3. Using the DSS Script batch file, run the “program\_evm.js” script command from program\_evm directory.

**Example:**

```
/program_evm>$DSS_SCRIPT_DIR/dss.sh program_evm.js TMDXEVM6678L-le
```

This will write all the little endian images to C6678 low cost EVM using XDS 100 emulator.

```
/program_evm>$DSS_SCRIPT_DIR/dss.sh program_evm.js TMDXEVM6670Le-le
```

This will write all the little endian images to C6670 low cost EVM using XDS 560 Mezzanine emulator.

### 7.5.3 Sample DSS Script output for Windows and Linux

The sample output after running the DSS Script is as below.

```
board: evm6670l
endian: Little
emulation: XDS560 mezzanine
binaries:
S:\aravind\C6670EVM_Debug\program_evm_friendly_rel_Advantech_dec2nd\bios_mcsdk_2.00.05.17\program_evm/binaries
/evm6670l/
```

```
ccxml:
S:\aravind\C6670EVM_Debug\program_evm_friendly_rel_Advantech_dec2nd\bios_mcsdk
_2.00.05.17\program_evm/configs/evm
6670l/evm6670le.ccxml
C66xx_0: GEL Output: Global Default Setup...

C66xx_0: GEL Output: C6670L GEL file Ver is 2.0

C66xx_0: GEL Output: Setup Cache...

C66xx_0: GEL Output: L1P = 32K

C66xx_0: GEL Output: L1D = 32K

C66xx_0: GEL Output: L2 = ALL SRAM

C66xx_0: GEL Output: Setup Cache... Done.

C66xx_0: GEL Output: Main PLL (PLL1) Setup ...

C66xx_0: GEL Output: PLL in Bypass ...

C66xx_0: GEL Output: PLL1 Setup for DSP @ 983.0 MHz.

C66xx_0: GEL Output:          SYSCLK2 = 327.6667 MHz, SYSCLK5 = 196.6 MHz.

C66xx_0: GEL Output:          SYSCLK8 = 15.35938 MHz.

C66xx_0: GEL Output: PLL1 Setup... Done.

C66xx_0: GEL Output: Power on all PSC modules and DSP domains...

C66xx_0: GEL Output: Security Accelerator disabled!
```

*C66xx\_0: GEL Output: Power on all PSC modules and DSP domains... Done.*

*C66xx\_0: GEL Output: PA PLL (PLL3) Setup ...*

*C66xx\_0: GEL Output: PA PLL Setup... Done.*

*C66xx\_0: GEL Output: DDR3 PLL (PLL2) Setup ...*

*C66xx\_0: GEL Output: DDR3 PLL Setup... Done.*

*C66xx\_0: GEL Output: DDR begin (1333 auto)*

*C66xx\_0: 2: XMC setup complete.*

*C66xx\_0: GEL Output:  
DDR3 initialization is complete.*

*C66xx\_0: GEL Output: DDR done*

*C66xx\_0: GEL Output: DDR3 memory test... Started*

*C66xx\_0: GEL Output: DDR3 memory test... Passed*

*C66xx\_0: GEL Output: PLL and DDR Initialization completed(0) ...*

*C66xx\_0: GEL Output: configSGMIISerdes Setup... Begin*

*C66xx\_0: GEL Output:  
SGMII SERDES has been configured.*

*C66xx\_0: GEL Output: Set Board and DSP IO/Timers Pins...*

*C66xx\_0: GEL Output: Set Board and DSP IO/Timers Pins... Done.*

*C66xx\_0: GEL Output: Configuring CPSW ...*

*C66xx\_0: GEL Output: Configuring CPSW ...Done*

*C66xx\_0: GEL Output: Global Default Setup... Done.*

*Start writing eeprom50*

*Writer:S:\aravind\C6670EVM\_Debug\program\_evm\_friendly\_rel\_Advantech\_dec2nd\bios\_mcsdk\_2.00.06.18\program\_evm/binaries/ev*

*m6670l/eepromwriter\_evm6670l.out*

*Image:S:\aravind\C6670EVM\_Debug\program\_evm\_friendly\_rel\_Advantech\_dec2nd\bios\_mcsdk\_2.00.06.18\program\_evm/binaries/evm*

*6670l/eeprom50.bin*

*C66xx\_0: GEL Output: Invalidate All Cache...*

*C66xx\_0: GEL Output: Invalidate All Cache... Done.*

*C66xx\_0: GEL Output: GEL Reset...*

*C66xx\_0: GEL Output: GEL Reset... Done.*

*EEPROM Writer Utility Version 01.00.00.04*

*Writing 50456 bytes from DSP memory address 0x80000000 to EEPROM bus address 0x0050 starting from device address 0x0000*

*...*

*Reading 50456 bytes from EEPROM bus address 0x0050 to DSP memory address 0x80010000 starting from device address 0x0000*

*...*

```
Verifying data read ...
EEPROM programming completed successfully
Start writing eeprom51
Writer:S:\aravind\C6670EVM_Debug\program_evm_friendly_rel_Advantech_dec2nd\bios
_mcsdk_2.00.06.18\program_evm/binaries/ev
m6670l/eepromwriter_evm6670l.out

Image:S:\aravind\C6670EVM_Debug\program_evm_friendly_rel_Advantech_dec2nd\bios
_mcsdk_2.00.06.18\program_evm/binaries/evm
6670l/eeprom51.bin

C66xx_0: GEL Output: Invalidate All Cache...

C66xx_0: GEL Output: Invalidate All Cache... Done.

C66xx_0: GEL Output: GEL Reset...

C66xx_0: GEL Output: GEL Reset... Done.

EEPROM Writer Utility Version 01.00.00.04

Writing 52656 bytes from DSP memory address 0x80000000 to EEPROM bus address
0x0051 starting from device address 0x0000
...
Reading 52656 bytes from EEPROM bus address 0x0051 to DSP memory address
0x80010000 starting from device address 0x0000
...
Verifying data read ...
EEPROM programming completed successfully
Writer:S:\aravind\C6670EVM_Debug\program_evm_friendly_rel_Advantech_dec2nd\bios
_mcsdk_2.00.06.18\program_evm/binaries/ev
m6670l/nandwriter_evm6670l.out
```

```
NAND:S:\aravind\C6670EVM_Debug\program_evm_friendly_rel_Advantech_dec2nd\bios
_mcsdk_2.00.06.18\program_evm/binaries/evm6
670l/nand.bin

C66xx_0: GEL Output: Invalidate All Cache...

C66xx_0: GEL Output: Invalidate All Cache... Done.

C66xx_0: GEL Output: GEL Reset...

C66xx_0: GEL Output: GEL Reset... Done.

Start loading nand.bin
Start programming NAND
NAND Writer Utility Version 01.00.00.04

Flashing block 1 (0 bytes of 8388608)
Flashing block 2 (16384 bytes of 8388608)
...
...
Flashing block 511 (8355840 bytes of 8388608)
Flashing block 512 (8372224 bytes of 8388608)
Reading and verifying block 1 (0 bytes of 8388608)
Reading and verifying block 2 (16384 bytes of 8388608)
...
...
Reading and verifying block 510 (8339456 bytes of 8388608)
Reading and verifying block 511 (8355840 bytes of 8388608)
Reading and verifying block 512 (8372224 bytes of 8388608)
NAND programming completed successfully
End programming NAND
Writer:S:\aravind\C6670EVM_Debug\program_evm_friendly_rel_Advantech_dec2nd\bios
_mcsdk_2.00.06.18\program_evm/binaries/ev
```



*m6670l/norwriter\_evm6670l.out*

*NOR:S:\aravind\C6670EVM\_Debug\program\_evm\_friendly\_rel\_Advantech\_dec2nd\bios\_mcsdk\_2.00.06.18\program\_evm/binaries/evm66*

*70l/nor.bin*

*C66xx\_0: GEL Output: Invalidate All Cache...*

*C66xx\_0: GEL Output: Invalidate All Cache... Done.*

*C66xx\_0: GEL Output: GEL Reset...*

*C66xx\_0: GEL Output: GEL Reset... Done.*

*Start loading nor.bin*

*Start programming NOR*

*2011\_12\_2\_130319*

*NOR Writer Utility Version 01.00.00.03*

*Flashing sector 0 (0 bytes of 8348588)*

*Flashing sector 1 (65536 bytes of 8348588)*

*Flashing sector 2 (131072 bytes of 8348588)*

*...*

*...*

*Flashing sector 126 (8257536 bytes of 8348588)*

*Flashing sector 127 (8323072 bytes of 8348588)*

*Reading and verifying sector 0 (0 bytes of 8348588)*

*Reading and verifying sector 1 (65536 bytes of 8348588)*

*Reading and verifying sector 2 (131072 bytes of 8348588)*

*...*

*...*

*Reading and verifying sector 126 (8257536 bytes of 8348588)*

*Reading and verifying sector 127 (8323072 bytes of 8348588)*

*NOR programming completed successfully*  
*End programming NOR*

## 4. Verification

### 4.1. Serial Port Setup

Connect the RS232 Serial cable provided in the box to the serial port of the Host PC. If Host is running Windows OS, start tera term and configure the serial port settings as follows.



### 4.2. Verifying POST

1. Set the dip switches as below.

SWITCH	Pin1	Pin2	Pin3	Pin4
SW3	Off	Off	On	Off
SW4	On	On	On	On
SW5	On	On	On	On
SW6	On	On	On	On

2. Power Cycle the board
3. Wait for the “POST done successfully!” message
4. The Screen Shot on the UART should be as below.

```

TMDXEUM6670L POST Version 01.00.00.04
-----
SOC Information

FPGA Version: 0006
Board Serial Number: 0DCE9530
EFUSE MAC ID is: 90 D7 EB 9E 5F 83
SA is disabled on this board.
PLL Reset Type Status Register: 0x00000001
Platform init return code: 0x00000000
Additional Information:
(0x02350014) :0FFF0000
(0x02350624) :000215FF
(0x02350678) :00121000
(0x0235063C) :000801FF
(0x02350640) :000801FF
(0x02350644) :000900DB
(0x02350648) :000A40DB
(0x0235064C) :000B10DB
(0x02350650) :000C00DB
(0x02350654) :000C00DB
(0x02350658) :000C00DB
(0x0235065C) :000D1800
(0x02350660) :000E1800
(0x02350668) :000F1800
(0x02350670) :00101800
(0x02620008) :0700600C
(0x0262000c) :04014142
(0x02620010) :00000000
(0x02620014) :05A80000
(0x02620018) :0B94102F
(0x02620180) :0603F000
-----

Power On Self Test

POST running in progress ...
POST I2C EEPROM read test started!
POST I2C EEPROM read test passed!
POST SPI NOR read test started!
POST SPI NOR read test passed!
POST GPIO NAND read test started!
POST GPIO NAND read test passed!
POST EMAC loopback test started!
POST EMAC loopback test passed!
POST external memory test started!
POST external memory test passed!
POST done successfully!
    
```

#### 4.2.1. Entering Serial Number to the EVM

1. After POST completes all the tests successfully, user can key in "ti" (small caps) to enter the board serial number.
2. Once the 10 digits serial number is entered successfully, power cycle the board to verify the new serial number.

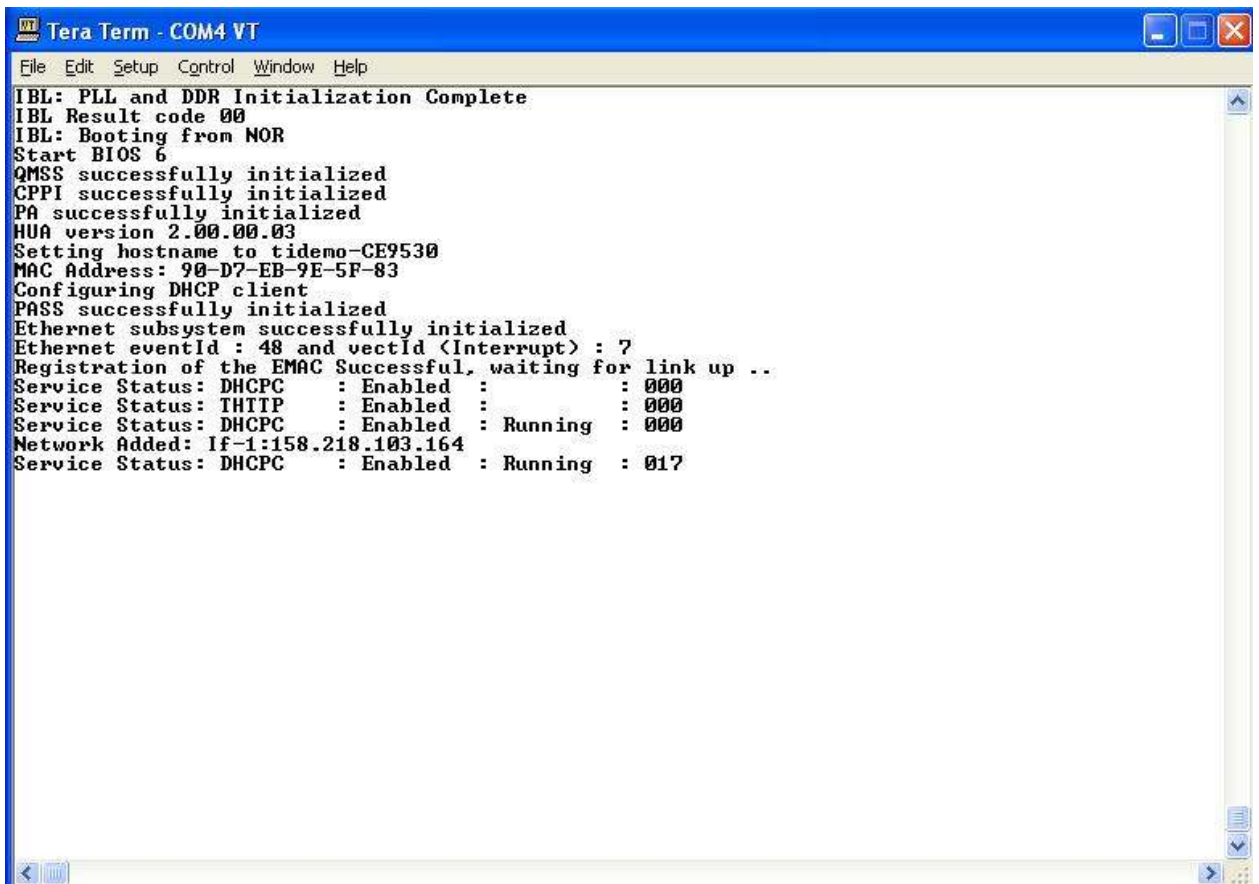
#### 4.3. Verifying NOR

1. Set the dip switches as below.

SWITCH	Pin1	Pin2	Pin3	Pin4
SW3	Off	Off	On	Off

SW4	On	On	On	On
SW5	On	On	On	<b>Off</b>
SW6	On	On	On	On

2. Power Cycle the board
3. Make sure the evm is connected to the DHCP server.
4. The HUA boot log will show up on the UART. A sample screen for EVM6678 is shown below.



#### 4.4. Verifying NAND

1. Set the dip switches as below.

SWITCH	Pin1	Pin2	Pin3	Pin4
SW3	Off	Off	On	Off
SW4	On	<b>Off</b>	On	On
SW5	On	On	On	Off

SW6	On	On	On	On
-----	----	----	----	----

2. Power Cycle the board
3. Wait for a minute
4. The Linux boot log will show up on the UART. A sample screen for EVM6678 is shown below.

```
IBL: PLL and DDR Initialization Complete
IBL Result code 00
IBL: Booting from NAND
Linux version 2.6.34-evmc6670.el-linux-c6x-2.0-n55 (bill@gtengapp01) (gcc
version 4.5.1 (Sourcery CodeBench Lite 4.5-124) ) #1 Fri
Sep 30 21:03:10 EDT 2011
Designed for the EVMC6670 board, Texas Instruments.
CPU0: C66x rev 0x0, 1.2 volts, 983MHz
Initializing kernel
physical RAM map changed by user
Built 1 zonelists in Zone order, mobility grouping on. Total pages: 65024
Kernel command line: console=ttyS0,115200 rw mem=256M ip=dhcp
initrd=0x80400000,0x400000
PID hash table entries: 1024 (order: 0, 4096 bytes)
Dentry cache hash table entries: 32768 (order: 5, 131072 bytes)
Inode-cache hash table entries: 16384 (order: 4, 65536 bytes)
Memory available: 251828k/258184k RAM, 0k/0k ROM (785k kernel code, 202k
data)
SLUB: Genslabs=7, HWalign=128, Order=0-3, MinObjects=0, CPUs=1, Nodes=1
Hierarchical RCU implementation.
RCU-based detection of stalled CPUs is enabled.
NR_IRQS:328
Console: colour dummy device 80x25
Calibrating delay loop... 980.99 BogoMIPS (lpj=1961984)
Mount-cache hash table entries: 512
C64x: 13 gpio irqs
NET: Registered protocol family 16
SGMII init complete
bio: create slab <bio-0> at 0
Switching to clocksource TSC64
NET: Registered protocol family 2
IP route cache hash table entries: 2048 (order: 1, 8192 bytes)
TCP established hash table entries: 8192 (order: 4, 65536 bytes)
TCP bind hash table entries: 8192 (order: 3, 32768 bytes)
TCP: Hash tables configured (established 8192 bind 8192)
TCP reno registered
UDP hash table entries: 256 (order: 0, 4096 bytes)
UDP-Lite hash table entries: 256 (order: 0, 4096 bytes)
NET: Registered protocol family 1
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
RPC: Registered tcp NFSv4.1 backchannel transport module.
Trying to unpack rootfs image as initramfs...
Freeing initrd memory: 4096k freed
JFFS2 version 2.2. (NAND) (SUMMARY) © 2001-2006 Red Hat, Inc.
```

```
ROMFS MTD (C) 2007 Red Hat, Inc.
msgmni has been set to 499
Block layer SCSI generic (bsg) driver version 0.4 loaded (major 254)
io scheduler noop registered
io scheduler deadline registered
io scheduler cfq registered (default)
Serial: 8250/16550 driver, 1 ports, IRQ sharing disabled
serial8250.0: ttyS0 at MMIO 0x2540000 (irq = 292) is a 16550A
console [ttyS0] enabled
brd: module loaded
loop: module loaded
at24 1-0050: 131072 byte 24c1024 EEPROM (writable)
uclinux[mtd]: RAM probe address=0x803dd7c0 size=0x0
Creating 1 MTD partitions on "RAM":
0x000000000000-0x000000000000 : "ROMfs"
mtd: partition "ROMfs" is out of reach -- disabled
Generic platform RAM MTD, (c) 2004 Simtec Electronics
GPIO NAND driver for C6x SoC boards
NAND device: Manufacturer ID: 0x20, Chip ID: 0x36 (ST Micro NAND 64MiB 1,8V
8-bit)
Scanning device for bad blocks
RedBoot partition parsing not available
Using static partition definition
Creating 3 MTD partitions on "gpio-nand-c6x":
0x000000000000-0x000000004000 : "bootconfig"
0x000000004000-0x000001000000 : "kernel"
0x000001000000-0x000004000000 : "filesystem"
keystone_netcp keystone_netcp.0: firmware: using built-in firmware keystone-
pdsp/qmss_pdsp_acc48_le.fw
keystone_netcp keystone_netcp.0: firmware: using built-in firmware keystone-
pdsp/pa_pdsp_default.fw
pktgen 2.72: Packet Generator for packet performance testing.
TCP cubic registered
NET: Registered protocol family 17
Sending DHCP requests ., OK
IP-Config: Got DHCP answer from 0.0.0.0, my address is 158.218.103.164
IP-Config: Complete:
    device=eth0, addr=158.218.103.164, mask=255.255.254.0, gw=158.218.102.2,
    host=158.218.103.164, domain=am.dhcp.ti.com, nis-domain=(none),
    bootserver=0.0.0.0, rootserver=0.0.0.0, rootpath=
Freeing unused kernel memory: 136K freed
starting pid 18, tty '': '/etc/rc.sysinit'

Starting system...

Mounting proc filesystem: done.
Mounting other filesystems: done.
Starting mdev
Setting hostname 158.218.103.164: done.
Bringing up loopback interface: done.
Starting inetd: done.

eth0      Link encap:Ethernet  HWaddr 90:D7:EB:9E:5F:83
          inet addr:158.218.103.164  Bcast:158.218.103.255
Mask:255.255.254.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:12 errors:0 dropped:0 overruns:0 frame:0
```

*Revision A:*

---

```
TX packets:2 errors:0 dropped:0 overruns:0 carrier:0  
collisions:0 txqueuelen:1000  
RX bytes:2653 (2.5 KiB) TX bytes:1180 (1.1 KiB)  
Interrupt:48
```

System started.

```
starting pid 67, tty '/dev/console': '/bin/sh'  
/ #
```