PROGRAM EVM IMAGES

Users Guide

Publication Date: September 24, 2012
Ver 1.9

Document License

Contents

Users Guide

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, EVM6657L 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
1.8	Fixed POST and NOR instructions
1.9	Added C6657 EVM support

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
eeprom50.bin	Binary file Power On Self Test (POST)
eeprom51.bin	Binary file for IBL
eepromwriter_evm6678l.out	eeprom Writer DSP executable
eepromwriter_input.txt	eeprom writer input for execution
eepromwriter_input50.txt	eeprom writer input for writing images to 0x50
eepromwriter_input51.txt	eeprom 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
eeprom50.bin	Binary file Power On Self Test (POST)
eeprom51.bin	Binary file for IBL
eepromwriter_evm6670l.out	eeprom Writer DSP executable
eepromwriter_input.txt	eeprom writer input for execution
eepromwriter_input50.txt	eeprom writer input for writing images to 0x50
eepromwriter_input51.txt	eeprom 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

3.3 C6657 EVM Files

C6657 EVM use the same factory default images under program_evm\binaries\evm6657l.

File Name	Description
eeprom50.bin	Binary file Power On Self Test (POST)
eeprom51.bin	Binary file for IBL
eepromwriter_evm6657l.out	eeprom Writer DSP executable
eepromwriter_input.txt	eeprom writer input for execution

eepromwriter_input50.txt	eeprom writer input for writing images to 0x50
eepromwriter_input51.txt	eeprom 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_evm6657l.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, EVM6657L, EVM6670L and EVM6618L in Little Endian Mode.

6 Directory Structure

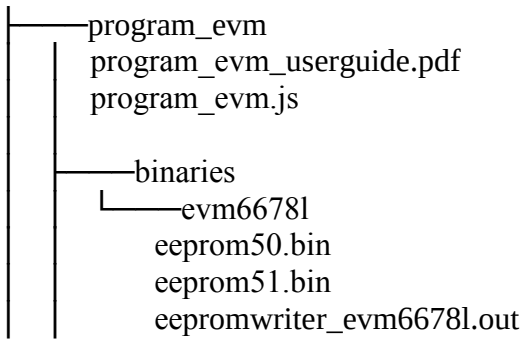
The program_evm (top-level) directory is intended to hold the *DSS* script for the Code Composer Studio which programs the default images to NAND/NOR/EEPROM.

The binaries/evm66xxl directory is intended to hold all the factory default images and the respective writers.

The configs/evm66xxl directory is intended to hold the "CCS target configuration files". Four pre-configured configurations are provided for each EVM type except evm6657l: 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. For evm6657l, besides the mentioned configuration files, another for XDS200 card in Windows is provided.

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 evm66xxl devices.



```
    |   |       eepromwriter_input.txt
    |   |       eepromwriter_input50.txt
    |   |       eepromwriter_input51.txt
    |   |       nand.bin
    |   |       nandwriter_evm6678l.out
    |   |       nand_writer_input.txt
    |   |       nor.bin
    |   |       norwriter_evm6678l.out
    |   |       nor_writer_input.txt
    |   |   └───evm6670l
    |   |       eeprom50.bin
    |   |       eeprom51.bin
    |   |       eepromwriter_evm6670l.out
    |   |       eepromwriter_input.txt
    |   |       eepromwriter_input50.txt
    |   |       eepromwriter_input51.txt
    |   |       nand.bin
    |   |       nandwriter_evm6670l.out
    |   |       nand_writer_input.txt
    |   |       nor.bin
    |   |       norwriter_evm6670l.out
    |   |       nor_writer_input.txt
    |   |   └───evm6657l
    |   |       eeprom50.bin
    |   |       eeprom51.bin
    |   |       eepromwriter_evm6657l.out
    |   |       eepromwriter_input.txt
    |   |       eepromwriter_input50.txt
    |   |       eepromwriter_input51.txt
    |   |       nand.bin
    |   |       nandwriter_evm6657l.out
    |   |       nand_writer_input.txt
    |   |       nor.bin
    |   |       norwriter_evm6657l.out
    |   |       nor_writer_input.txt
    |   ├───configs
    |   |   └───evm6678l
    |   |       evm6678l.ccxml
    |   |       evm6678le.ccxml
    |   |       evm6678l-linuxhost.ccxml
    |   |       evm6678le-linuxhost.ccxml
    |   |   └───evm6670l
    |   |       evm6670l.ccxml
    |   |       evm6670le.ccxml
    |   |       evm6670l-linuxhost.ccxml
    |   |       evm6670le-linuxhost.ccxml
```

```
└──evm6657l
    evm6657l.ccxml
    evm6657le.ccxml
    evm6657ls.ccxml
    evm6657l-linuxhost.ccxml
    evm6657le-linuxhost.ccxml
├──gel
    evmc6670l.gel
    evmc6678l.gel
    evmc6657l.gel
    README.txt
└──logs (empty directory)
```

7 Programming the bin files

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

7.1 Set the EVM Dip switches

7.1.1 For all the EVM 6670 and EVM6678

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

7.1.2 For EVM6657

Make sure the EVM dip switches are kept as below.

SWITCH	Pin1	Pin2	Pin3	Pin4	Pin5	Pin6	Pin7	Pin8
SW3	**Off**	On	On	On	On	On	On	On
SW5	On	On	On	On	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.

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.

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

7.4.1 General Script Usage

Script Usage:

```
[MCSDK]\tools\program_evm>%DSS_SCRIPT_DIR%\dss.bat program_evm.js
[tmdx|tmds]evm(6678|6670|6618|6657)[l|le|ls][-le|-be] [1|0]
```

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

6657: C6657 device

l: Low cost EVM

le: EVM uses 560 Mezzanine Emulator daughter card

ls: EVM uses XDS200 Emulator card

-le: Little Endian

-be: Big Endian

1: Verify DDR ccontents

0: Do not verify DDR contents

7.4.2 Formatting the NAND Flash

The program_evm supports formatting the NAND device as below.

```
[MCSDK]\tools\program_evm>%DSS_SCRIPT_DIR%\dss.bat program_evm.js
tmdxevm6657l [1|0] format-nand
```

Warning: Please note that this would erase all the nand blocks.

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: evm6657l

endian: Little

emulation: XDS200 emulator

binaries: C:\Program Files\Texas Instruments\mcsdk_2_01_00_03\tools\program_evm/binaries/evm6657l/

ccxml: C:\Documents and Settings\xxxxxx\user\CCSTargetConfigurations\evmc6657.ccxml

C66xx_0: GEL Output:

Connecting Target...

C66xx_0: GEL Output: DSP core #0

C66xx_0: GEL Output: C6657L GEL file Ver is 1.0

C66xx_0: GEL Output: Global Default Setup...

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: PLL1 Setup for DSP @ 1000.0 MHz.

C66xx_0: GEL Output: SYSCLK2 = 333.3333 MHz, SYSCLK5 = 200.0 MHz.

C66xx_0: GEL Output: SYSCLK8 = 15.625 MHz.

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

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

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

C66xx_0: GEL Output: DDR3 PLL (PLL2) Setup ...

C66xx_0: GEL Output: DDR3 PLL Setup... Done.

C66xx_0: GEL Output: DDR3 Init begin (1333 auto)

C66xx_0: GEL Output: XMC Setup ... Done

C66xx_0: GEL Output:
DDR3 initialization is complete.

C66xx_0: GEL Output: DDR3 Init done

C66xx_0: GEL Output: DDR3 memory test... Started

C66xx_0: GEL Output: DDR3 memory test... Passed

C66xx_0: GEL Output: PLL and DDR3 Initialization completed(0) ...

C66xx_0: GEL Output: configSGMIISerdes Setup... Begin

C66xx_0: GEL Output: SGMII SERDES has been configured.

C66xx_0: GEL Output: Enabling EDC ...

C66xx_0: GEL Output: L1P error detection logic is enabled.

C66xx_0: GEL Output: L2 error detection/correction logic is enabled.

C66xx_0: GEL Output: MSMC error detection/correction logic is enabled.

C66xx_0: GEL Output: Enabling EDC ...Done

C66xx_0: GEL Output: Global Default Setup... Done.

Start writing eeprom50

Writer:C:\Program Files\Texas Instruments\mcsdk_2_01_00_03\tools\program_evm/binaries/evm6657l/eepromwriter_evm6657l.out

Image:C:\Program Files\Texas Instruments\mcsdk_2_01_00_03\tools\program_evm/binaries/evm6657l/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.

C66xx_0: GEL Output: Disable all EDMA3 interrupts and events.

EEPROM Writer Utility Version 01.00.00.05

Writing 59128 bytes from DSP memory address 0x0c000000 to EEPROM bus address 0x0050 starting from device address 0x0000 ...

Reading 59128 bytes from EEPROM bus address 0x0050 to DSP memory address 0x0c010000 starting from device address 0x0000 ...

Verifying data read ...

EEPROM programming completed successfully

Start writing eeprom51

Writer:C:\Program Files\Texas Instruments\mcsdk_2_01_00_03\tools\program_evm/binaries/evm6657l/eepromwriter_evm6657l.out

Image:C:\Program Files\Texas Instruments\mcsdk_2_01_00_03\tools\program_evm/binaries/evm6657l/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.

C66xx_0: GEL Output: Disable all EDMA3 interrupts and events.

EEPROM Writer Utility Version 01.00.00.05

Writing 47848 bytes from DSP memory address 0x0c000000 to EEPROM bus address 0x0051 starting from device address 0x0000 ...

Reading 47848 bytes from EEPROM bus address 0x0051 to DSP memory address 0x0c010000 starting from device address 0x0000 ...

Verifying data read ...

EEPROM programming completed successfully

Writer:C:\Program Files\Texas Instruments\mcsdk_2_01_00_03\tools\program_evm/binaries/evm6657l/nandwriter_evm6657l.out

NAND:C:\Program Files\Texas Instruments\mcsdk_2_01_00_03\tools\program_evm/binaries/evm6657l/nand.bin

Required NAND files does not exist in C:\Program Files\Texas Instruments\mcsdk_2_01_00_03\tools\program_evm/binaries/evm6657l/

Writer:C:\Program Files\Texas Instruments\mcsdk_2_01_00_03\tools\program_evm/binaries/evm6657l/norwriter_evm6657l.out

NOR:C:\Program Files\Texas Instruments\mcsdk_2_01_00_03\tools\program_evm/binaries/evm6657l/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.

C66xx_0: GEL Output: Disable all EDMA3 interrupts and events.

Start loading nor.bin

Start programming NOR

2012_08_23_165121

NOR Writer Utility Version 01.00.00.03

Flashing sector 0 (0 bytes of 664115)

Flashing sector 1 (65536 bytes of 664115)

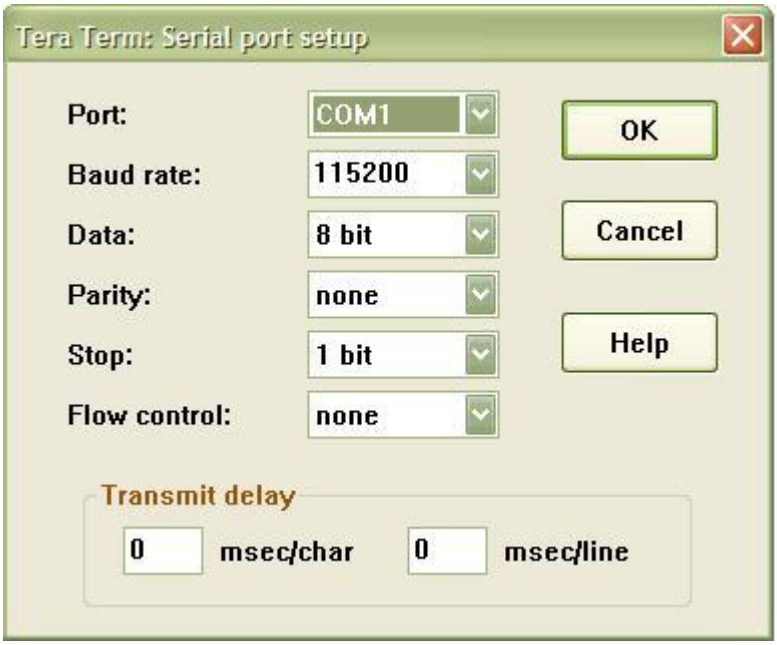
Flashing sector 2 (131072 bytes of 664115)

```
Flashing sector 3 (196608 bytes of 664115)
Flashing sector 4 (262144 bytes of 664115)
Flashing sector 5 (327680 bytes of 664115)
Flashing sector 6 (393216 bytes of 664115)
Flashing sector 7 (458752 bytes of 664115)
Flashing sector 8 (524288 bytes of 664115)
Flashing sector 9 (589824 bytes of 664115)
Flashing sector 10 (655360 bytes of 664115)
Reading and verifying sector 0 (0 bytes of 664115)
Reading and verifying sector 1 (65536 bytes of 664115)
Reading and verifying sector 2 (131072 bytes of 664115)
Reading and verifying sector 3 (196608 bytes of 664115)
Reading and verifying sector 4 (262144 bytes of 664115)
Reading and verifying sector 5 (327680 bytes of 664115)
Reading and verifying sector 6 (393216 bytes of 664115)
Reading and verifying sector 7 (458752 bytes of 664115)
Reading and verifying sector 8 (524288 bytes of 664115)
Reading and verifying sector 9 (589824 bytes of 664115)
Reading and verifying sector 10 (655360 bytes of 664115)
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

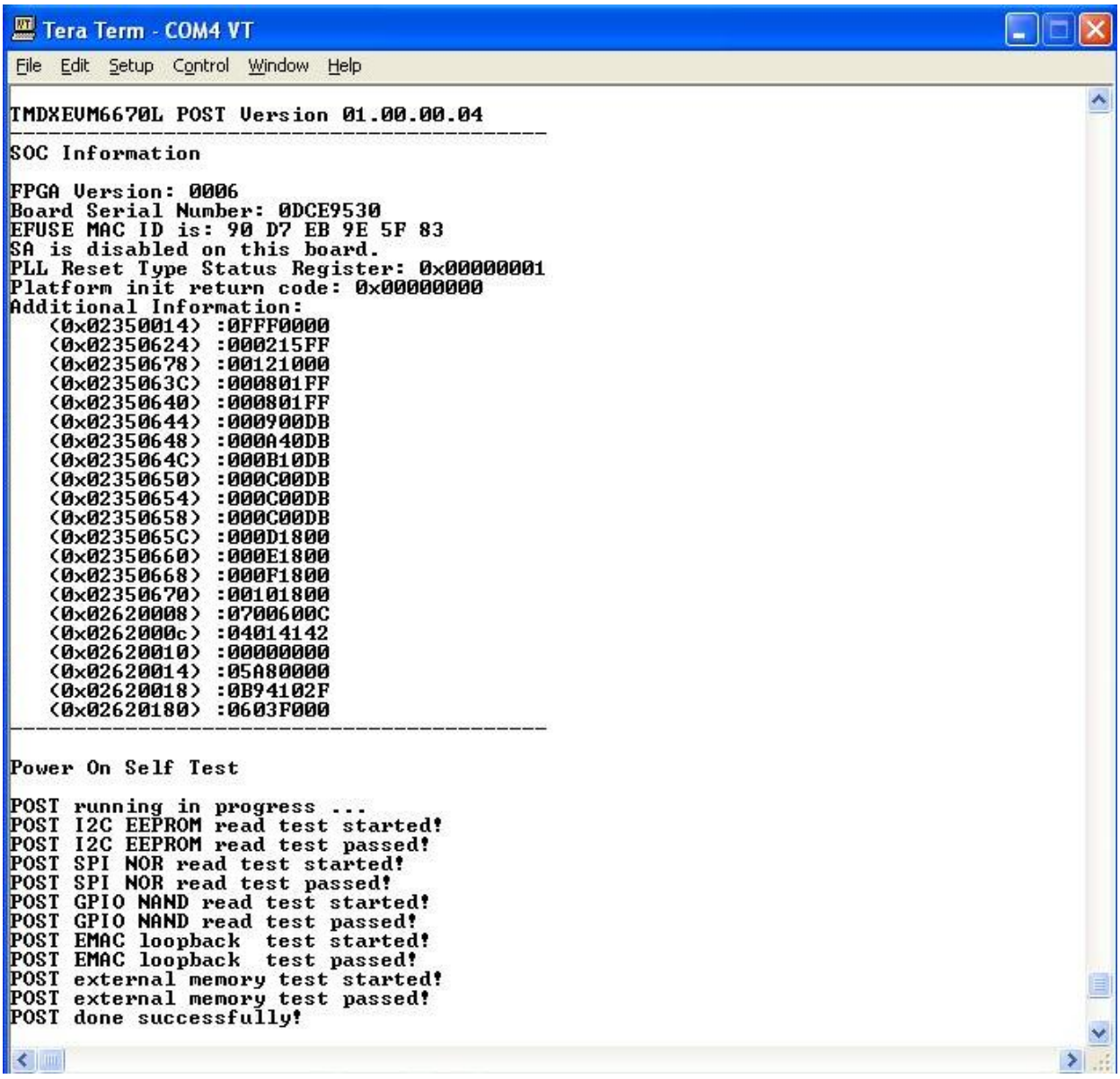
1. Set the dip switches as below for EVM6670 and EVM6678.

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

For EVM6657, set the dip switches as below.

SWITCH	Pin1	Pin2	Pin3	Pin4	Pin5	Pin6	Pin7	Pin8
SW3	**Off**	Off	On	Off	On	On	On	On
SW5	On	On	On	On	On	On	On	On

2. Power Cycle the board, wait for the PC to recognize the EVM card, then hit Rst_FULL button on the EVM.
3. Wait for the “POST done successfully!” message.
4. The Screen Shot on the UART should be as below.



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 for EVM6670 and EVM6678.

SWITCH	Pin1	Pin2	Pin3	Pin4
SW3	Off	Off	On	Off

SW4	On	On	On	On
SW5	On	On	On	**Off**
SW6	On	On	On	On

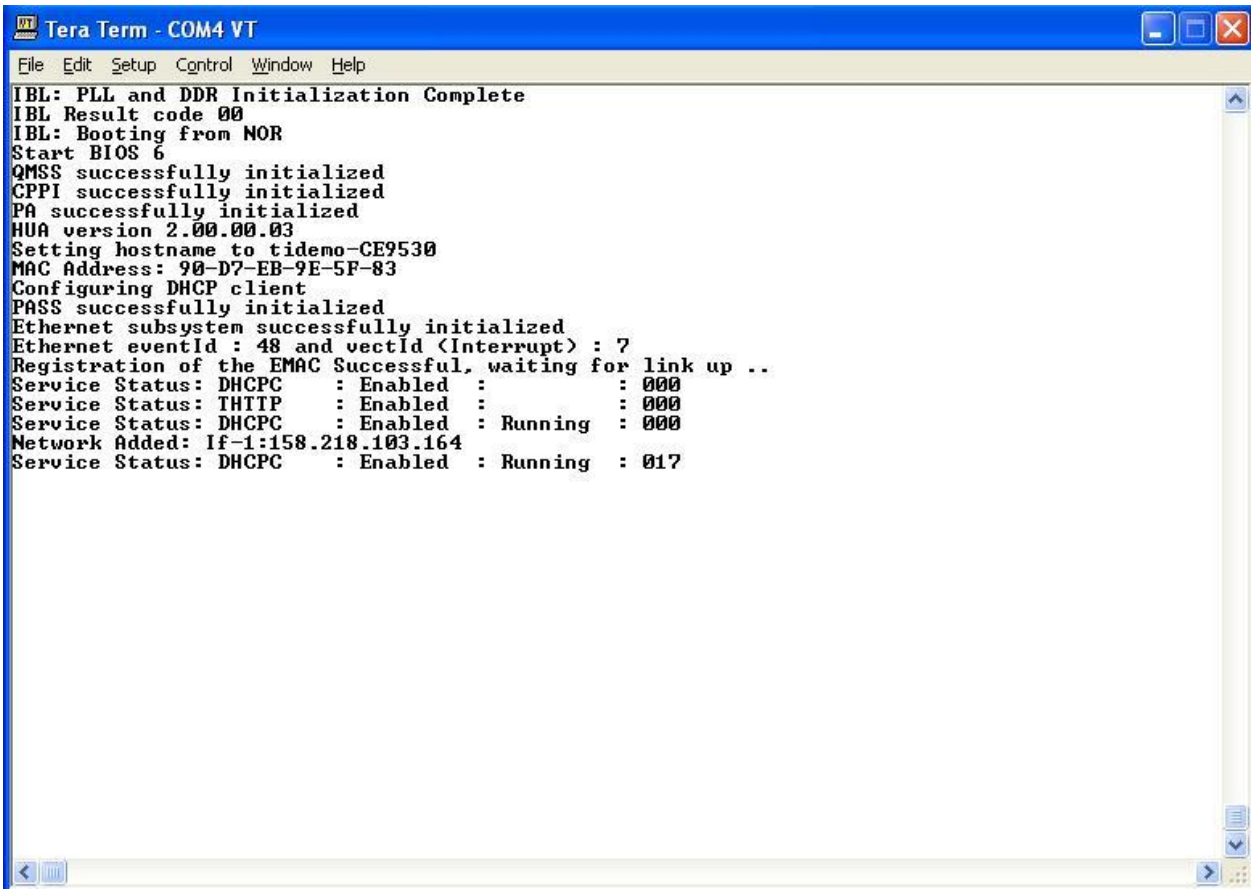
For EVM6657, set the dip switches as below.

SWITCH	**Pin1**	**Pin2**	**Pin3**	**Pin4**	**Pin5**	**Pin6**	**Pin7**	**Pin8**
SW3	**Off**	Off	On	Off	On	On	On	On
SW5	On	On	On	Off	On	On	On	On

2. Power Cycle the board, wait for the PC to recognize the EVM card, then hit Rst_FULL button on the EVM.

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 for EVM 6670 and EVM6678.

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

Note there is no NAND.bin provided for C6657 as there is no Linux support on C6657. Please refer to device hardware setup documentation for booting from NAND.

2. Power Cycle the board, wait for the PC to recognize the EVM card, then hit Rst_FULL button on the EVM.

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
          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'
/ #
```