PROGRAM EVM IMAGES

Users Guide

Publication Date: June 1, 2011
Ver 1.3

Texas Instruments

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 Texas Instruments Incorporated - http://www.ti.com

© Copyright 2011 Texas Instruments, Inc.
All Rights Reserved

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 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

3 Files Provided

3.1 C6678 EVM Files

The following files are the factory default images under factory_images\binaries\evm6678l. The md5sum file provides the md5sum on the below files.

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_input50.txt	eeprom writer input for writing images to 0x50
eepromwriter_input51.txt	eeprom writer input for writing images to 0x51
md5sum.txt	md5sums
nand.bin	Binary file for Linux Kernel Image Data Portion
nandwriter_evm6678l.out	Nand Writer DSP executable
nand_oob.bin	Nand OOB bin file (used only at the factory)

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 factory_images\binaries\evm6670l. The md5sum file provides the md5sum on the below files.

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_input50.txt	eeprom writer input for writing images to 0x50
eepromwriter_input51.txt	eeprom writer input for writing images to 0x51
md5sum.txt	md5sums
nand.bin	Binary file for Linux Kernel Image Data Portion
nandwriter_evm6670l.out	Nand Writer DSP executable
nand_oob.bin	Nand OOB bin file (used only at the factory)
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 factory images 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”. Two pre-configured configurations are provided for each EVM type (one for inbuilt xds 100 and another one for xds560 mezzanine card).

```
├────factory_images
│   │   program_evm.pdf
│   │   program_evm.js
│   │
│   ├────binaries
│   │   └────evm6678l
│   │       eeprom50.bin
│   │       eeprom51.bin
│   │       eepromwriter_evm6678l.out
│   │       eepromwriter_input.txt
│   │       eepromwriter_input50.txt
│   │       eepromwriter_input51.txt
│   │       md5sum.txt
│   │       nand.bin
│   │       nandwriter_evm6678l.out
│   │       nand_oob.bin
│   │       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
│   │       md5sum.txt
│   │       nand.bin
│   │       nandwriter_evm6670l.out
│   │       nand_oob.bin
│   │       nand_writer_input.txt
│   │       nor.bin
│   │       norwriter_evm6670l.out
│   │       nor_writer_input.txt
│   ├────configs
│   │   └────evm6678l
│   │       evm6678l.ccxml
│   │       evm6678le.ccxml
│   │   └────evm6670l
│   │       evm6670l.ccxml
```

```
|   |       evm6670le.ccxml
|   |
|   └──logs (empty directory)
```

7 Programming the bin files

This section assumes that the factory default images are downloaded to “C:\factory_images” directory.

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)

Example:

```
set DSS_SCRIPT_DIR="C:\Program Files\Texas
Instruments\ccsv5\ccs_base_5.0.3.00022\scripting\bin"
```

2. Set the **PROGRAM_EVM_TARGET_CONFIG_FILE** environment variable (Optional)

Example:

```
set PROGRAM_EVM_TARGET_CONFIG_FILE = C:\Documents and
Settings\a0756924\user\CCSTargetConfigurations\myC667xl.ccxml
```

This step is required only if you are using an emulator other than the one that came with your board. If you wish to use such an external emulator you will need to set ROGRAM_EVM_TARGET_CONFIG_FILE. If this environment variable is not set, the DSS script will use the default ccxml files that support the following emulators.

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)

Example:
```
export DSS_SCRIPT_DIR=/opt/ti/ccsv5/ccs_base_5.0.3.00022/scripting/bin
```

2. Set the **PROGRAM_EVM_TARGET_CONFIG_FILE** environment variable for using the DVD provided ccxml files OR user ccxml files. (Optional)

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

This step is required only if you are using an emulator other than the one that came with your board. If you wish to use such an external emulator you will need to set ROGRAM_EVM_TARGET_CONFIG_FILE. If this environment variable is not set, the DSS script will use the default ccxml files that support the following emulators.

3. xds100 inbuilt (evm667xl-linuxhost.ccxml)
4. xds560 mezzanine card (evm6670le-linuxhost.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 DSS Script Arguments

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

tmdx: TMDX type EVM

tmds: TMDS type EVM

c: Not used, for backward compatibility

6678: C6678 device

6670: C6670 device

6618: TCI6618 device

l: Low cost EVM

x: EVM supports encryption

e: EVM uses 560 Mezzanine Emulator daughter card

le: Little Endian

be: Big Endian

- Image Identifier:

“nand nor eeprom50 eeprom51”: User can include one or more image identifier to program the image, if no image identifier specified, the script will program all the images.

Example:

```
/factory_images>%DSS_SCRIPT_DIR%\dss.bat program_evm.js evm6678l“nand”
```

```
This will write the little endian nand.bin image to C6678 low cost EVM
using XDS 100 emulator
```

or

```
/factory_images>%DSS_SCRIPT_DIR%\dss.bat program_evm.js
evm6670le“eeprom50 eemprom51”
```

This will write the little endian eeprom50.bin and eeprom51.bin images to C6670 low cost EVM using XDS 560 Mezzanine emulator

7.4 Executing the DSS script to restore factory default images.

7.4.1 Windows

1. cd “\factory_images” directory
2. Please note that **PROGRAM_EVM_TARGET_CONFIG_FILE** is not a mandatory environment variable for windows, if not set it uses the default ccxml files as below.
3. Using the DSS Script batch file, run the “program_evm.js” script command from factory_images directory.

Example:

/factory_images>%DSS_SCRIPT_DIR%\dss.bat program_evm.js TMDXEVM6678L

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

/factory_images>%DSS_SCRIPT_DIR%\dss.bat program_evm.js TMDXEVM6670LE

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

7.4.2 Linux

1. cd “factory_images” directory
2. Please note that **PROGRAM_EVM_TARGET_CONFIG_FILE** is not a mandatory environment variable for Linux, if not set it uses the default ccxml files as below.

3. Using the DSS Script batch file, run the “program_evm.js” script command from factory_images directory.

Example:

```
/factory_images>$DSS_SCRIPT_DIR/dss.sh program_evm.js TMDXEVM6678L
This will write all the little endian images to C6678 low cost EVM using XDS 100 emulator.

/factory_images>$DSS_SCRIPT_DIR/dss.sh program_evm.js TMDXEVM6670LE
```

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

7.4.3 Sample DSS Script output for windows and linux

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

Start writing to EEPROM

2011_05_19_145733

C66xx_0: GEL Output:

Connecting Target...

C66xx_0: GEL Output: DSP core #0

C66xx_0: GEL Output: C6678L GEL file Ver is 1.4

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: 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: Set_PSC_State... Timeout Error #03 pd=2, md=9!
C66xx_0: GEL Output: Power on all PSC modules and DSP domains... Done.
C66xx_0: GEL Output: PA PLL is using SYSCLK/ALTCORECLK as the input
C66xx_0: GEL Output: PA PLL is in PLL mode
C66xx_0: GEL Output: PA PLL fixed output divider = 2
C66xx_0: GEL Output: PA PLL programmable multiplier = 21
C66xx_0: GEL Output: PA PLL programmable divider = 1
C66xx_0: GEL Output: the output frequency should be 10 times the PA reference clock
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: Configuring CPSW ...
C66xx_0: GEL Output: Configuring CPSW ...Done
C66xx_0: GEL Output: DDR begin (1333 auto)
C66xx_0: GEL Output: XMC Setup ... Done
C66xx_0: GEL Output:
DDR3 initialization is complete.
C66xx_0: GEL Output: DDR done
C66xx_0: GEL Output: Invalidate All Cache...
C66xx_0: GEL Output: Invalidate All Cache... Done.
C66xx_0: GEL Output: DSP Reset CPU...
C66xx_0: GEL Output: DSP Reset CPU... Done.
C66xx_0: GEL Output: Disable all EDMA3 interrupts and events.
EEPROM Writer Utility Version 01.00.00.02

Writing 65536 bytes from DSP memory address 0x80000000 to EEPROM bus address 0x0050 starting from device address 0x0000

...

Reading 65536 bytes from EEPROM bus address 0x0050 to DSP memory address 0x80010000 starting from device address 0x0000

...

Verifying data read ...

EEPROM programming completed successfully

C66xx_0: GEL Output: Invalidate All Cache...

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

C66xx_0: GEL Output: DSP Reset CPU...

C66xx_0: GEL Output: DSP Reset CPU... Done.

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

EEPROM Writer Utility Version 01.00.00.02

Writing 65536 bytes from DSP memory address 0x80000000 to EEPROM bus address 0x0051 starting from device address 0x0000

...

Reading 65536 bytes from EEPROM bus address 0x0051 to DSP memory address 0x80010000 starting from device address 0x0000

...

Verifying data read ...

EEPROM programming completed successfully

C66xx_0: GEL Output: Invalidate All Cache...

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

C66xx_0: GEL Output: DSP Reset CPU...

C66xx_0: GEL Output: DSP Reset CPU... Done.

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

Start loading nand.bin

2011_05_19_145940

2011_05_19_150011

Start programming NAND

2011_05_19_150011

NAND Writer Utility Version 01.00.00.02

Flashing block 1 (0 bytes of 10485760)
Flashing block 2 (16384 bytes of 10485760)
Flashing block 3 (32768 bytes of 10485760)
…
…
Flashing block 638 (10436608 bytes of 10485760)
Flashing block 639 (10452992 bytes of 10485760)
Flashing block 640 (10469376 bytes of 10485760)
Reading and verifying block 1 (0 bytes of 10485760)
Reading and verifying block 2 (16384 bytes of 10485760)
Reading and verifying block 3 (32768 bytes of 10485760)
…
…
Reading and verifying block 638 (10436608 bytes of 10485760)
Reading and verifying block 639 (10452992 bytes of 10485760)
Reading and verifying block 640 (10469376 bytes of 10485760)
NAND programming completed successfully
End programming NAND
2011_05_19_150129
C66xx_0: GEL Output: Invalidate All Cache...
C66xx_0: GEL Output: Invalidate All Cache... Done.
C66xx_0: GEL Output: DSP Reset CPU...
C66xx_0: GEL Output: DSP Reset CPU... Done.
C66xx_0: GEL Output: Disable all EDMA3 interrupts and events.

Start loading nor.bin
2011_05_19_150129
2011_05_19_150153
Start programming NOR
2011_05_19_150153

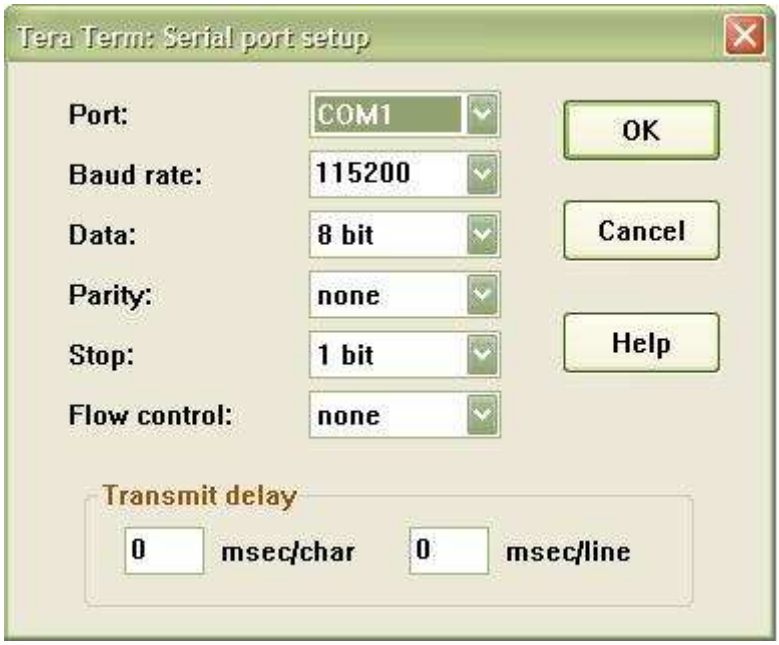
NOR Writer Utility Version 01.00.00.01

Flashing sector 0 (0 bytes of 8152620)
Flashing sector 1 (65536 bytes of 8152620)
Flashing sector 2 (131072 bytes of 8152620)
…
…
Flashing sector 123 (8060928 bytes of 8152620)
Flashing sector 124 (8126464 bytes of 8152620)
Reading and verifying sector 0 (0 bytes of 8152620)
Reading and verifying sector 1 (65536 bytes of 8152620)
…
…
Reading and verifying sector 123 (8060928 bytes of 8152620)
Reading and verifying sector 124 (8126464 bytes of 8152620)
NOR programming completed successfully
End programming NOR
2011_05_19_150407

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.

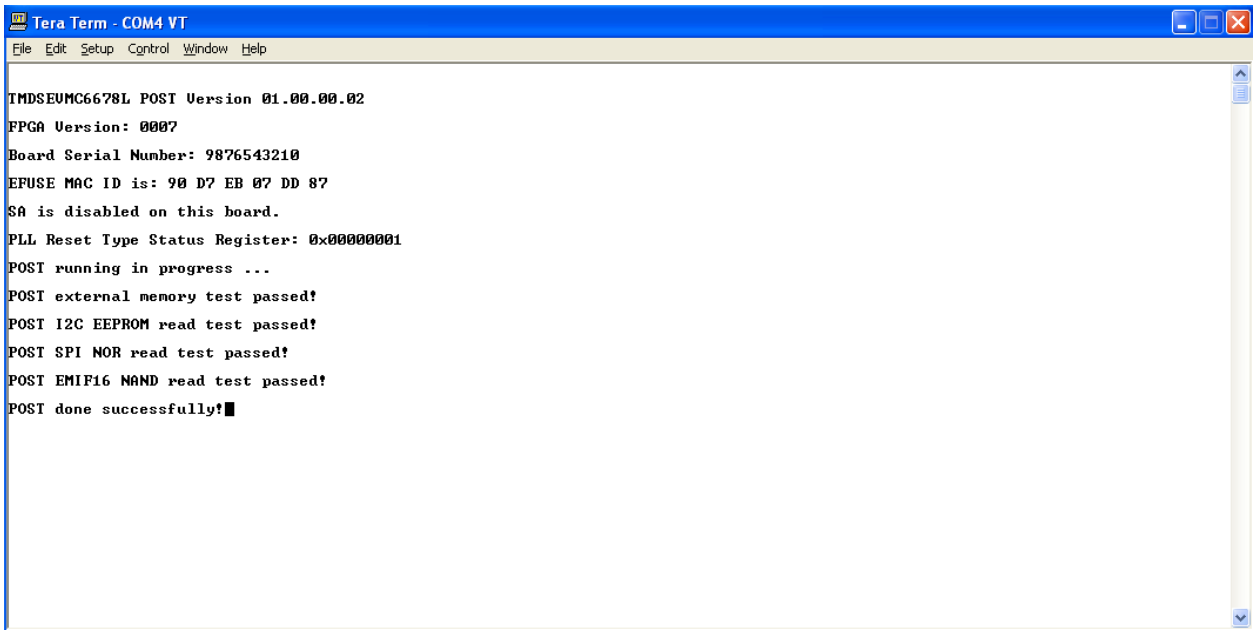


4.2. Verifying POST

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.



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.

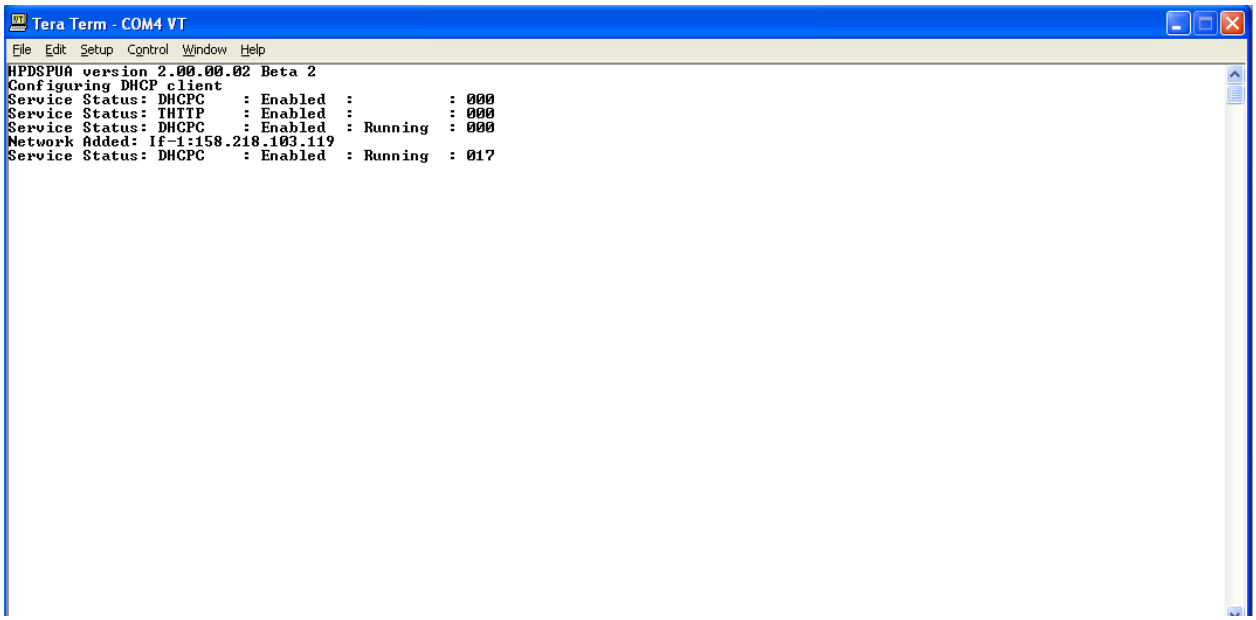
2. Once the 10 digits serial number is entered successfully, power cycle the board to verify the new serial number.

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	**Off**
SW6	On	On	On	On

2. Power Cycle the board
3. Make sure the evm is connected to the DHCP server.
4. The Screen Shot on the UART should be as below.



4.4. Verifying NAND

1. Set the dip switches as below.

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

2. Power Cycle the board
3. Wait for a minute
4. The Screen Shot on the UART should be as below.

```
Linux version 2.6.34-evmc6678.el-20110504 (root@gtcs13.gt.design.ti.com) (gcc
version
4.5.1 (Sourcery G++ Lite 4.5-109) ) #1 Wed May 4 11:14:47 EDT 2011
Designed for the EVMC6678 board, Texas Instruments.
CPU0: C66x rev 0x0, 1.2 volts, 1000MHz
Initializing kernel
Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 130048
Kernel command line: console=ttyS0,115200 initrd=0x80400000,0x300000 ip=dhcp
rw
PID hash table entries: 2048 (order: 1, 8192 bytes)
Dentry cache hash table entries: 65536 (order: 6, 262144 bytes)
Inode-cache hash table entries: 32768 (order: 5, 131072 bytes)
Memory available: 512760k/520340k RAM, 0k/0k ROM (784k kernel code, 200k
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:288
Console: colour dummy device 80x25
Calibrating delay loop... 999.42 BogoMIPS (lpj=1998848)
Mount-cache hash table entries: 512
C64x: 8 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: 4096 (order: 2, 16384 bytes)
TCP established hash table entries: 16384 (order: 5, 131072 bytes)
TCP bind hash table entries: 16384 (order: 4, 65536 bytes)
TCP: Hash tables configured (established 16384 bind 16384)
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: 3072k freed
JFFS2 version 2.2. (NAND) (SUMMARY)  Â© 2001-2006 Red Hat, Inc.
ROMFS MTD (C) 2007 Red Hat, Inc.
msgmni has been set to 1007
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 = 276) 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=0x803da3c0 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
NAND device: Manufacturer ID: 0x20, Chip ID: 0x36 (ST Micro NAND 64MiB 1,8V
8-bit)
Scanning device for bad blocks
Creating 2 MTD partitions on "davinci_nand.0":
0x000000000000-0x000001000000 : "kernel"
0x000001000000-0x000004000000 : "filesystem"
davinci_nand davinci_nand.0: controller rev. 2.5
m25p80 spi0.0: n25q128 (16384 Kbytes)
Creating 1 MTD partitions on "spi_flash":
0x000000000000-0x000001000000 : "test"
spi_davinci spi_davinci.0: Controller at 0x20bf0000
spi_davinci spi_davinci.0: Operating in interrupt mode using IRQ 182
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 158.218.100.184
IP-Config: Complete:
     device=eth0, addr=158.218.100.184, mask=255.255.255.0, gw=158.218.100.2,
     host=158.218.100.184, domain=am.dhcp.ti.com, nis-domain=(none),
     bootserver=0.0.0.0, rootserver=0.0.0.0, rootpath=
Freeing unused kernel memory: 140K freed
starting pid 17, tty : '/etc/rc.sysinit'

Starting system...

Mounting proc filesystem: done.
Mounting other filesystems: done.
Starting mdev
Setting hostname 158.218.100.184: done.
Bringing up loopback interface: done.
Starting inetd: done.

eth0      Link encap:Ethernet  HWaddr 90:D7:EB:07:12:93
          inet addr:158.218.100.184  Bcast:158.218.100.255
Mask:255.255.255.0
          UP BROADCAST RUNNING ALLMULTI MULTICAST  MTU:1500  Metric:1
          RX packets:8 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:1708 (1.6 KiB)  TX bytes:1180 (1.1 KiB)
          Interrupt:48

System started.

starting pid 54, tty '/dev/console': '/bin/sh'
```