# OMAP35x DVEVM
# Getting Started Guide

Literature Number: SPRUFZ7

Sept 23, 2009

TEXAS INSTRUMENTS

## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

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

| **Products** | | **Applications** | |
|---|---|---|---|
| Amplifiers | amplifier.ti.com | Audio | www.ti.com/audio |
| Data Converters | dataconverter.ti.com | Automotive | www.ti.com/automotive |
| DSP | dsp.ti.com | Broadband | www.ti.com/broadband |
| Interface | interface.ti.com | Digital Control | www.ti.com/digitalcontrol |
| Logic | logic.ti.com | Military | www.ti.com/military |
| Power Mgmt | power.ti.com | Optical Networking | www.ti.com/opticalnetwork |
| Microcontrollers | microcontroller.ti.com | Security | www.ti.com/security |
| Low Power Wireless | www.ti.com/lpw | Telephony | www.ti.com/telephony |
| | | Video & Imaging | www.ti.com/video |
| | | Wireless | www.ti.com/wireless |

## EVALUATION BOARD/KIT IMPORTANT NOTICE

Texas Instruments (TI) provides the enclosed product(s) under the following conditions:

This evaluation board/kit is intended for use for **ENGINEERING DEVELOPMENT, DEMONSTRATION, OR EVALUATION PURPOSES ONLY** and is not considered by TI to be a finished end-product fit for general consumer use. Persons handling the product(s) must have electronics training and observe good engineering practice standards. As such, the goods being provided are not intended to be complete in terms of required design-, marketing-, and/or manufacturing-related protective considerations, including product safety and environmental measures typically found in end products that incorporate such semiconductor components or circuit boards. This evaluation board/kit does not fall within the scope of the European Union directives regarding electromagnetic compatibility, restricted substances (RoHS), recycling (WEEE), FCC, CE or UL, and therefore may not meet the technical requirements of these directives or other related directives.

Should this evaluation board/kit not meet the specifications indicated in the User's Guide, the board/kit may be returned within 30 days from the date of delivery for a full refund. THE FOREGOING WARRANTY IS THE EXCLUSIVE WARRANTY MADE BY SELLER TO BUYER AND IS IN LIEU OF ALL OTHER WARRANTIES, EXPRESSED, IMPLIED, OR STATUTORY, INCLUDING ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE.

The user assumes all responsibility and liability for proper and safe handling of the goods. Further, the user indemnifies TI from all claims arising from the handling or use of the goods. Due to the open construction of the product, it is the user's responsibility to take any and all appropriate precautions with regard to electrostatic discharge.

EXCEPT TO THE EXTENT OF THE INDEMNITY SET FORTH ABOVE, NEITHER PARTY SHALL BE LIABLE TO THE OTHER FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES.

TI currently deals with a variety of customers for products, and therefore our arrangement with the user **is not exclusive**.

TI assumes **no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein**.

Please read the User's Guide and, specifically, the Warnings and Restrictions notice in the User's Guide prior to handling the product. This notice contains important safety information about temperatures and voltages. For additional information on TI's environmental and/or safety programs, please contact the TI application engineer or visit www.ti.com/esh.

No license is granted under any patent right or other intellectual property right of TI covering or relating to any machine, process, or combination in which such TI products or services might be or are used.

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

## FCC Warning

This evaluation board/kit is intended for use for **ENGINEERING DEVELOPMENT, DEMONSTRATION, OR EVALUATION PURPOSES ONLY** and is not considered by TI to be a finished end-product fit for general consumer use. It generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to part 15 of FCC rules, which are designed to provide reasonable protection against radio frequency interference. Operation of this equipment in other environments may cause interference with radio communications, in which case the user at his own expense will be required to take whatever measures may be required to correct this interference.

# Preface

## *About This Guide*

The OMAP35x EVM is an evaluation platform that showcases the OMAP architecture and lets users evaluate the power and performance of OMAP as a flexible general purpose platform.

This guide gives you overview information about the board and the software provided with the board. It is intended to be used as an introductory document for the EVM. Other documents provide more in-depth information. See the EVM documentation section of the release notes for a complete list of documents that have been included with the product.

## *Notational Conventions*

This document uses the following conventions:

Program listings, program examples, and interactive displays are shown in a `mono-spaced font`. Examples use **bold** for emphasis, and interactive displays use **`bold`** to distinguish commands that you enter from items that the system displays (such as prompts, command output, error messages, etc.).

Square brackets ( [ and ] ) identify an optional parameter. If you use an optional parameter, you specify the information within the brackets. Unless the square brackets are in a **bold** typeface, do not enter the brackets themselves.

### *Trademarks*

The Texas Instruments logo and Texas Instruments are registered trademarks of Texas Instruments. Trademarks of Texas Instruments include: TI, OMAP, the OMAP logo, XDS, Code Composer, Code Composer Studio, Probe Point, Code Explorer, DSP/BIOS, RTDX, Online DSP Lab, OMAP, TMS320, TMS320C54x, TMS320C55x, TMS320C62x, TMS320C64x, TMS320C67x, TMS320C5000, and TMS320C6000.

MS-DOS, Windows, and Windows NT are trademarks of Microsoft Corporation.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds.

CodeSourcery and CodeSourcery's Sourcery G++ are trademarks or registered trademarks of CodeSourcery, Inc.

Solaris, SunOS, and Java are trademarks or registered trademarks of Sun Microsystems, Inc.

All other brand, product names, and service names are trademarks or registered trademarks of their respective companies or organizations.

## Table of contents

# 1 Overview

This Chapter introduces the OMAP35x DVEVM (Digital Video Evaluation Module).

## 1.1 *What's in the kit?*

The OMAP35x DVEVM kit comes with the following hardware items. EVM Hardware Setup tells how to connect these components.

- OMAP35x EVM main board with power module, processor module, and dual-mode touch-screen LCD display and stylus
- USB cable
- Serial cable
- Universal power supply with a power cord and converter

The kit comes with the following printed documentation:

- OMAP35x EVM Readme First card with installation and optional registration instructions
- OMAP35x EVM Setup Guide (this document)

The kit comes with the following software CDs. The information on how to use these software components are provided in OMAP35x DVEVM Software Setup

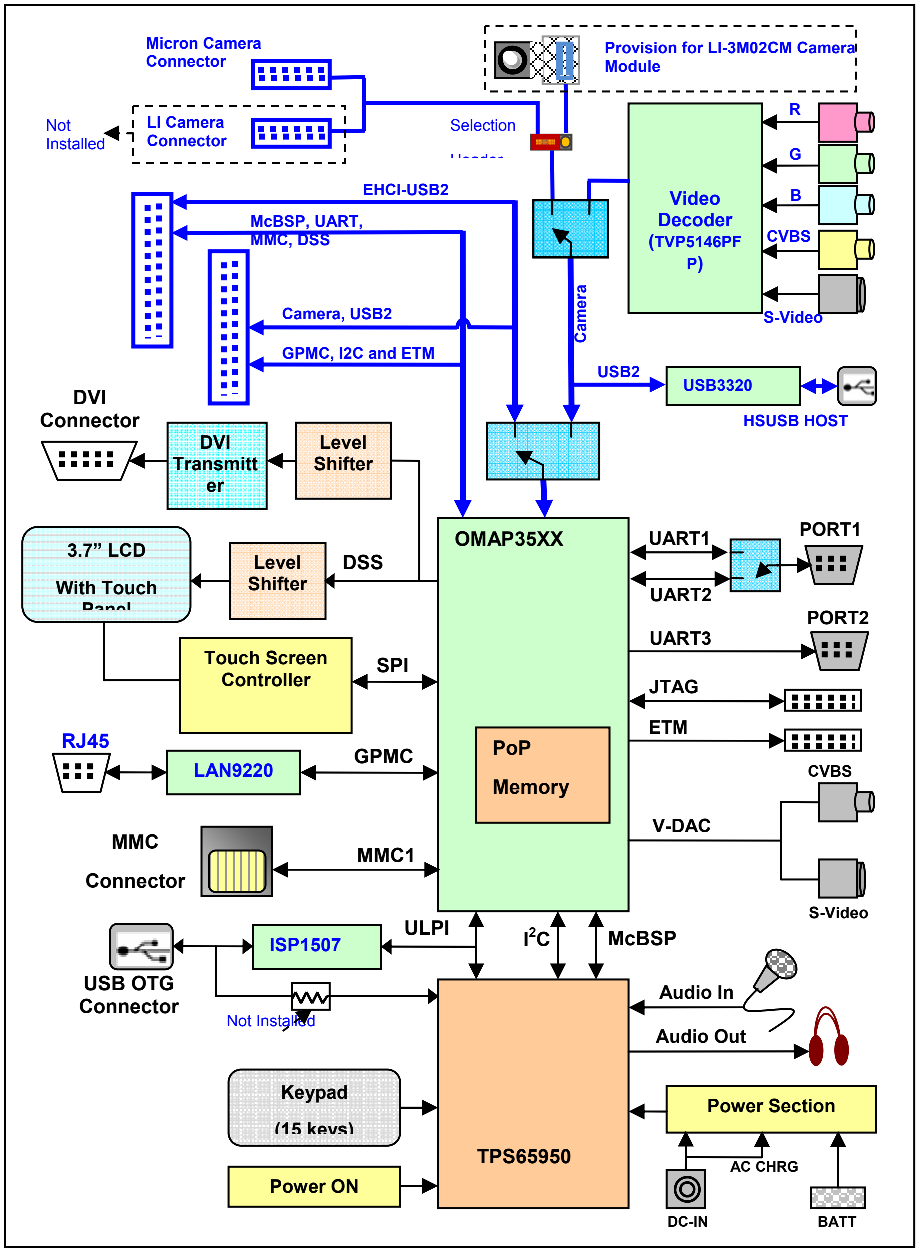
- **OMAP35x Digital Video Software Development Kit (DVSDK) (2 DVD set) with**
  - **OMAP35x EVM Hardware User's Guide**
  - **OMAP35x EVM Getting Started Guide**
- **Sourcery G++TM evaluation tools from CodeSourcery (1 CD-ROM)**

Installation and use of the Software Development Kit is described in this document. This document assumes the use of the LITE version of CodeSourcery tools. This version is an unrestricted command line toolchain available free of charge from the CodeSourcery web site (see section Installing the Toolchain)

Alternatively, a full featured IDE 30 Day time limited version of Sourcery G++ is provided. To install and use the evaluation version of CodeSourcery's Sourcery G++ product, please see the information available on the provided CD.

## *1.2 What's on the board?*

The following block diagram shows the major hardware components.



**NOTE: TWL4030 will be replaced with TPS65950 for mass production. These devices are pin and software compatible.**

## 1.3 What's next?

To get started evaluating the DVEVM kit and developing applications for the OMAP35x, begin by using this Getting Started guide. It is suggested that the user follow the sequence described in the document. The sequence is outlined below.

| | |
|---|---|
| EVM Hardware Setup | Connect the various cables and other basic hardware setup |
| Running the DVSDK Demonstration | Run the demonstration application that comes with the OMAP3530 DVSDK |
| OMAP3530 DVSDK Software setup | Setup the development hosts using the OMAP3530 DVSDK installer files and the compiler toolchain |
| Re-install the target software | Use the supplied, known good software images to re-flash the EVM. |
| Rebuild software | Rebuild various aspects of the supplied target software |
| Develop with the OMAP35x EVM board | The final section covers additional operation information on using your OMAP35x EVM board |

## *1.4 Updates & Further information*

Please be sure to register your EVM at:

http://www.ti.com/omapregistration

Once registered you will be able to find software and documentation updates at:

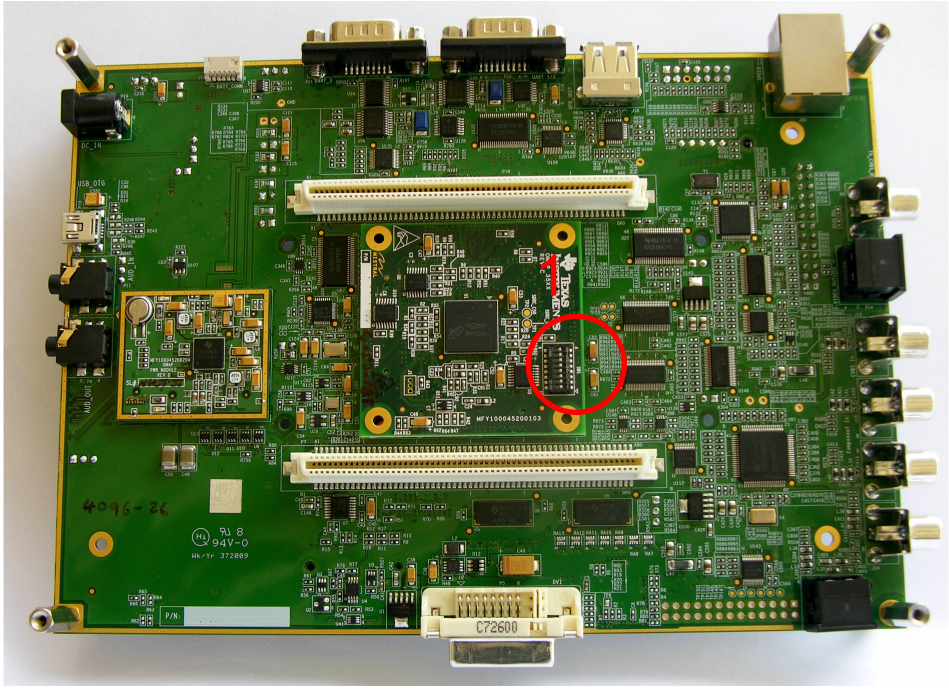
http://www.ti.com/omapsoftwareupdates

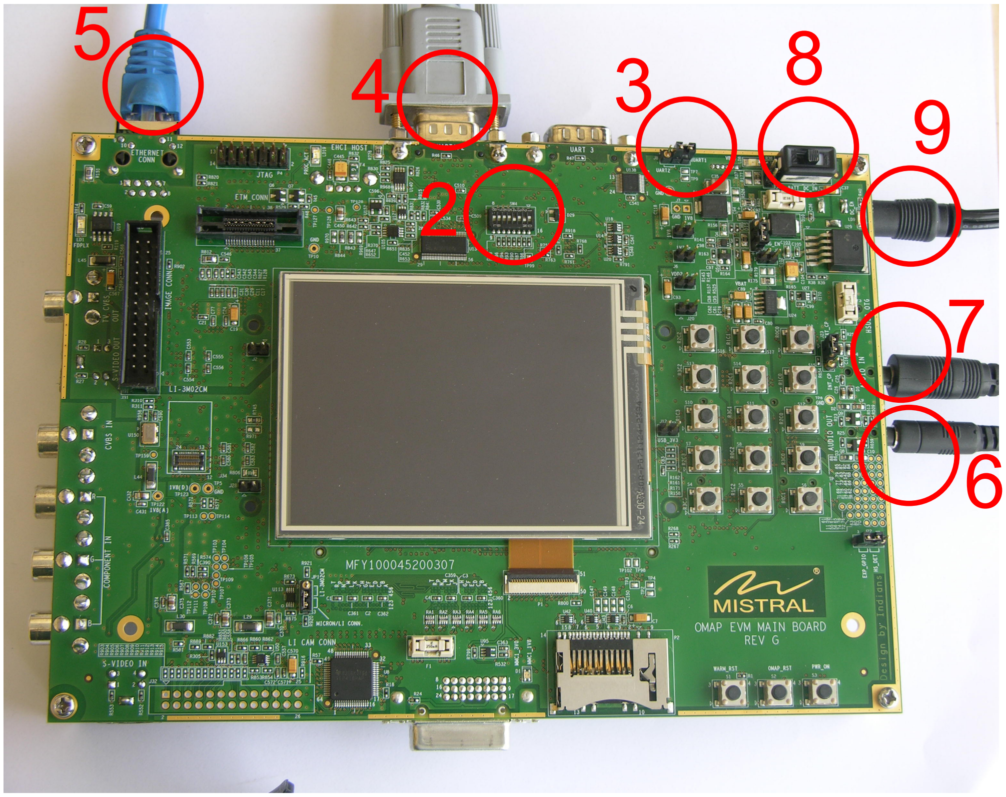
A wiki based information site is available. User contributions are encouraged.

http://wiki.davincidsp.com/index.php?title=Main_Page

# 2 EVM Hardware Setup

This section will guide you through the process of setting up the EVM hardware for the purpose of running the supplied demo application. The information is the same as in the supplied Setup Guide.

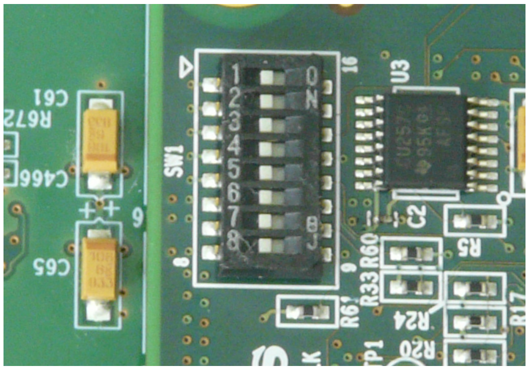
| | Step | Section |
|---|---|---|
|  **OMAP35x EVM Board Bottom** | 1 | 2.1 |



**OMAP35x EVM Board Top**

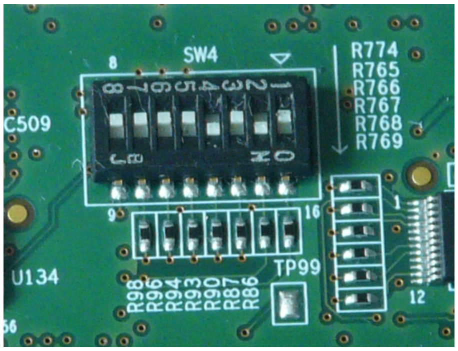
| Step | Section |
|---|---|
| 2 | 2.2 |
| 3 | 2.3 |
| 4 | 2.4 |
| 5 | 2.5 |
| 6 | 2.6 |
| 7 | 2.7 |
| 8 | 2.8 |
| 9 | 2.9 |

## 2.1 Processor Module SW1



The DIP switch on the processor module controls various signal routing choices. Please be sure it matches the above diagram for normal operation. SW1 must be configured differently to use a JTAG emulator with the EVM. Consult the Hardware User's Guide for more detail.The DIP switch on the processor module controls various signal routing choices. Please be sure it matches the above diagram for normal operation. SW1 must be configured differently to use a JTAG emulator with the EVM. Consult the Hardware User's Guide for more detail.

## 2.2 Main Board SW4



The main board's SW4 DIP switch controls the boot mode of OMAP3 processor. The default setting shown above will try to boot from UART3. If no response is seen in a short time (< 1 S) the processor will attempt to boot from the attached flash memory.

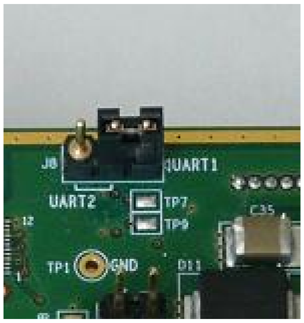
Consult the Hardware User's Guide for details and for other settings.

Please ensure to match the numbers in the diagram to the numbers on the DIP switch as the orientation of the switch may not be what you expect. It is not guaranteed that the switch settings below would work for all the EVM boards. For more information on the switch settings for selecting the boot modes, refer to the user guide in the OMAP3503 SDK installation under /home/<useracct>/OMAP35x-PSP-SDK-##.##.##.##/docs folder.

| Micron NAND | SW4-8 | SW4-7 | SW4-6 | SW4-5 | SW4-4 | SW4-3 | SW4-2 | SW4-1 |
|---|---|---|---|---|---|---|---|---|
| | BOOT_ UART | Not used | SYS_BOOT 5 | SYS_BOOT 4 | SYS_BOOT 3 | SYS_BOOT 2 | SYS_BOOT 1 | SYS_BOOT 0 |
| **UART/Flash** | OFF | OFF | OFF | OFF | **ON** | OFF | **ON** | OFF |
| **MMC** | OFF | OFF | OFF | OFF | OFF | **ON** | **ON** | **ON** |

Note: Switch OFF == logic 1 and Switch ON == logic 0.

## 2.3 J8



Jumper J8 controls which OMAP UART is connected to the UART1/2 board connector. Make sure it selects UART1 as shown. All other jumpers on the board are normally unpopulated.

## 2.4 UART1

Connect the supplied serial cable to the UART-1/2 DB-9 connector. The other end of the serial port should be connected to a PC. The PC should run a terminal emulator such as HyperTerm or TeraTerm for Windows or minicom or gtkterm for Linux. The terminal program should be set for 115200 Baud rate, 8 data bits, no parity, and all flow control off.

During boot and while the demo application is running, status messages will appear on the serial port connection.

## 2.5 Ethernet

Connect an Ethernet cable to the RJ-45 jack on the board. It is suggested that initially the Ethernet be used on a closed private network with static IP addresses assigned. DHCP can be used once the setup is complete. The Ethernet connection is not used during the demo but is used for TFTP transfers of software and can be used for Telnet and NFS.

## 2.6 Audio Out

P9 can be connected to powered PC style speakers or to headphones (not supplied).

Although the demo will run without audio connections, they are used during the audio playback portion of the demo.

## 2.7 Audio In

P11 can be connected to an audio source such as the line out jack of a PC sound card or the headphone jack of an MP3 player.

As with the audio out, the demo will function without this connection.

## 2.8 Power Selection Switch

SW2 on the Main board selects between battery and DC power in. The DVEVM comes with a DC power source that should be used. This switch should be set so the switch body is closer to the DC label on the board.

**Note:** This switch is not intended to be a power switch. It should be set to DC and left in that position. An external switch can be used to control the AC input to the power adapter if an adapter is desired.

## 2.9 Power Cable

After all other steps are performed, ensure the power adapter is unplugged and then insert the adapter's DC plug into P19. Finally the power adapter can be plugged into the AC wall source.

## 2.10 Setup Terminal Program

A serial port terminal program should be used to communicate with the OMAP35x EVM board's serial port console. For Windows users HyperTerm or TeraTerm are recommended. For Linux users Minicom is recommended. In any case, the serial modem settings are the same:

| | |
|---|---|
| Bits per second: | 115200 |
| Data bits: | 8 |
| Parity: | None |
| Stop bits: | 1 |
| Flow control: | none |

Once the settings are in place, connect to the board using the serial terminal program and power up the board by plugging in the AC adapter. The U-Boot start up screen will appear. If not interrupted, the EVM will continue to boot into a Linux command prompt.

To send text file scripts to the EVM a per-line delay of approximately 100 ms must be configured into the terminal program.

The following sections give specific setup guidance for three serial terminal programs.

### 2.10.1 HyperTerm Setup

HyperTerminal comes with Windows XP and can be found in Start Menu -> All Programs -> Accessories -> Communication -> HyperTerminal. After initial dialogs, create a new connection and select the appropriate COMn port. The next dialog will allow you to set the communication parameters as described above. Finally set the line delay by choosing File -> Properties -> Setting Tab -> ASCII Setup button -> and enter 100 in the Line Delay box. Press OK.

### 2.10.2 TeraTerm Setup

TeraTerm is a commonly used terminal emulator for Windows. It can be found at the following URL:

http://hp.vector.co.jp/authors/VA002416/teraterm.html

When you start TeraTerm it will normally ask you for TCP or COM port. Select the correct COM port. Serial port settings can be found in Setup -> Serial Port …. Set the serial port as described above and enter 100 in the msec/line entry. You can save your settings using Setup -> Save Settings …. If you save to the teraterm.ini file in the TeraTerm install directory you will set the startup options for the program.

### 2.10.3 Minicom Setup

In order to run minicom as a regular user, make sure the user is in the proper group, allowed to create lock files in /var/lock (the default location), which is usually the "uucp" or "dialup" group.

To create a new configuration for the serial port, run minicom as follows under root:

```
# minicom -c on -s
```

Command line options used above enable color (-c on) and start the setup procedure (-s).

From the setup menu configure the serial port name and parameters - "Serial port setup" -> "A" for the "Serial Device" (/dev/ttyS0 or /dev/ttyS1) and "E" for the "Bps/Par/Bits" setup, which is "115200 8N1" by default.

You may want to remove the Init (A) and Reset (B) modem strings under "Modem and dialing" menu.

Once ready, save the configuration either into default location (Save setup as dfl) or otherwise into a separate name (Save setup as...), in which case you will have to specify the non-default configuration every time you start minicom.

Now, you can start minicom as a regular user:

```
$ minicom -c on
```

And with the non-default configuration, assuming you saved it under "ttyS0" name:

```
$ minicom -c on ttyS0
```

In order to send multi-line scripts to the target, set the "Newline delay (ms)" under "Terminal settings" (Alt+T) to 99. This step must be completed each time before sending a multi-line text file.

# 3 Running the Demonstration Software

This chapter explains how to run the software demos provided with the OMAP35x DVEVM.

## *3.1 Default Boot Configuration*

Out of the box, the EVM boots from flash and starts the demos automatically after a few seconds when you power up the board. It does not require an NFS mount or a TFTP server to run the standard demos. A DHCP server is required by default; the DHCP service provided by most routers is sufficient to run the demos. By default, the DHCP server obtains dynamic IP addresses.

The out-of-the-box boot parameters are listed in Section Booting From Flash using Board's NAND Flash File System The following are alternate ways you may want to boot the board:

- TFTP boot with NAND flash file system (Booting via TFTP Using Board's NAND Flash File System)
- Flash boot with NFS file system (Booting from Flash Using NFS File System)
- TFTP boot with NFS file system (Booting via TFTP using NFS file system)

To abort the standard boot, press any key in the console window (see Section 2.10 on setting up the console window). Also see Section (Alternate Boot Methods) if you want to change the boot configuration.

**NOTE**

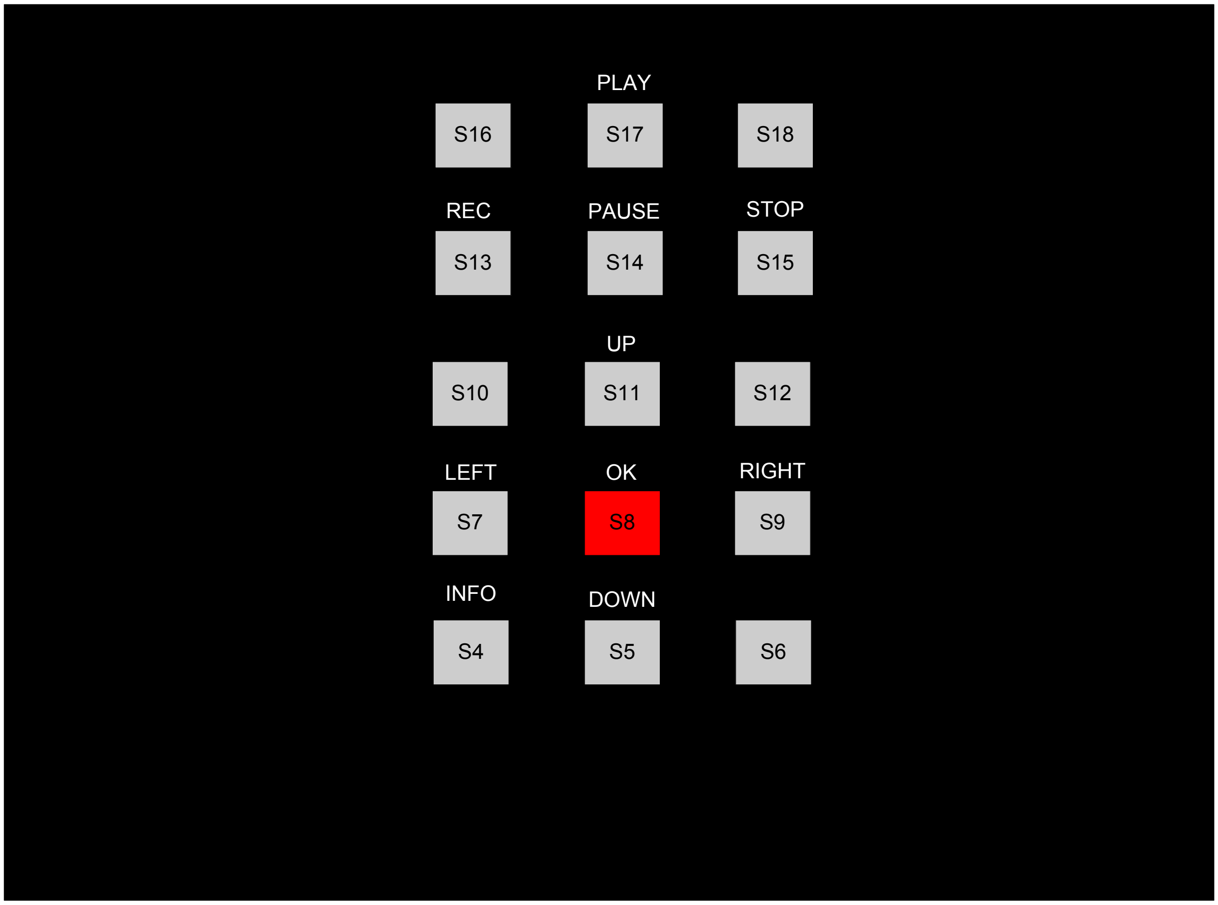
1. If you already have OMAP35x EVM hardware, but do not have any software flashed on the board, refer to the sections in the following order, to get started.
   - OMAP35x DVEVM Software Setup
   - Updating/Restoring the Flash Image
   - Booting From Flash using Board's NAND Flash File System
2. If you already have a OMAP35x EVM with U-boot, kernel image and NAND flash file system, make sure you update the kernel and NAND flash file system images, in the NAND flash with the kernel image built for this DVSDK build and pre-built NAND flash file system image available under /home/<useracct>/dvsdk_#_##_##_##/targetfs folder on your Linux PC. For more information on rebuilding the linux kernel, refer section ( Rebuilding the Linux Kernel). For more information on how to do get started, refer to the sections in the following order.
   - OMAP35x DVEVM Software Setup
   - Booting the New Linux Kernel
   - Flashing the Pre-built/rebuilt NAND Flash File System
   - Booting From Flash using Board's NAND Flash File System

## 3.2 *Starting the Standalone Demos*

When you connect the EVM hardware, the pre-loaded examples run automatically on your video display. These examples decode audio and video. There are two ways to use the demos:

- **Standalone.** This is the default power-on mode. The demos run automatically with no connection to a workstation in the default boot configuration.
- **Command line.**Once you have connected the EVM to a workstation and installed the necessary software (as described in Section Installing the Software), you can run the demos from the board's Linux command line as described in Section Running the Demos from Command Line

Once the EVM is booted, your video display should display the picture of the keypad as shown below. To use the demos in standalone mode, make sure you look at this initial screen that shows the function of various keys in the key pad diagram

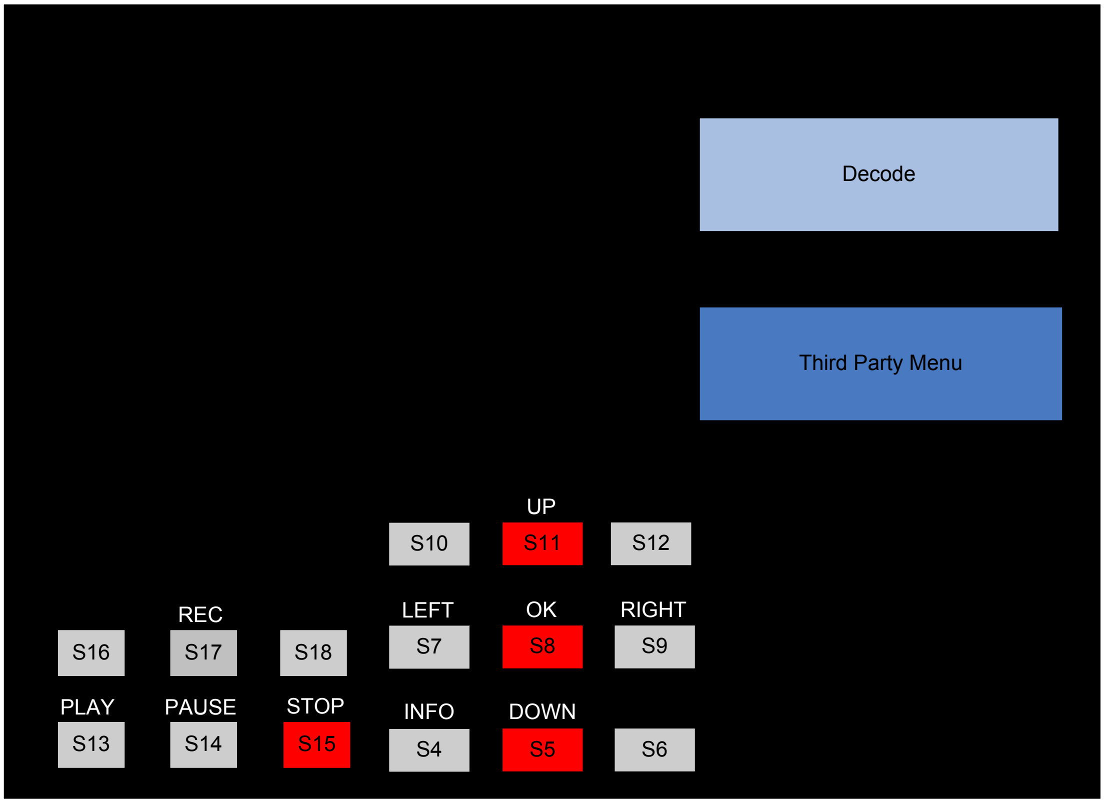


Please note the following before looking at the functions of the keys:

- Orientation of the key pad with respect to the board
- The numbers associated with the key on the initial screen match the actual numbers of the keys on the OMAP35x EVM board.
- Touch screen is currently not supported for the buttons displayed on the screens
- The Active keys in the menu are highlighted in red while the inactive keys are shaded grey.

## 3.3 Running the Standalone Demos

1. Press "OK" on the keypad to move from the keypad interface diagram to the main menu screen, which looks like this:



2. The Decode demo plays audio and video files you select.
3. Use the up and down arrows to change which demo is selected. Then, press "OK" to switch to the selected demo. Press “Stop” (S15 key on the key pad) to quit the demo. On quitting the demo, “Quitting the Demo” text is shown for a specific timeout period and the interface application is killed. In case of running in standalone mode, the EVM needs to be powered off and on, if you want to initiate the demos again
4. Within a demo, you start at the settings screen, where you see the controls you can use to run the demo at the bottom of the screen and the current settings in the upper-right.
5. Use the up and down arrows to move to a setting you want to change.
6. Use the left and right arrows to cycle through the options until the setting you want is shown.
7. Press "Play" to begin the Decode demos. Press "Stop" to return to the main menu.
8. While the demo runs, data about the settings, processor load, and rates are shown. Static settings are on the right. Dynamic data reporting is on the left.
9. This information overlays the video; as a result the video you see is darker than the actual video. To hide the information display so that you can better see the video, press the "INFO" button on the keypad. You can change the transparency of the OSD (overlay) while running a demo by using the left and right arrows on the remote.
10. Press "Stop" or "Pause" when you want to end or pause a demo.
11. Press "Stop" from the settings screen, you go back to the main menu.

The demos use the Codec Engine to allow applications to run algorithms.

### 3.3.1 About the Decode Demo

The Decode demo plays audio and video files you select. You can select a source video file and a source audio file. Use the left and right arrow buttons to choose from the demo files and the files created by the encode example, which are stored on the EVM's NAND flash. The decoded signals are sent to your video display and speakers.

The supported video algorithms are MPEG4 Simple Profile (.m4v file extension) and H.264 Base Profile (.264 file extension).

The supported audio algorithm is AAC (.aac file extension).

By default the display output is LCD.

**Table 3-1 Key Pad Buttons for Decode Demo**

| Key Pad Number | Key Pad Function | Mode | Action Performed |
|---|---|---|---|
| S7/S9 | Left/Right | Setup | Select a different file combination |
| S17/S8 | Play or OK | Setup | Begin decode demo |
| S4 | INFO | Run | Toggle information display |
| S7/S9 | Left/Right | Run | Change information transparency level |
| S14 | Pause | Run | Pause demo (press Play to resume) |
| S15 | Stop | Setup/Run | Return to previous screen |

The application runs on the ARM using Linux. The video and audio signals are passed to decoders by the Codec Engine.

To use this demo from the command line, see Section Running the Demos from Command Line.

## 3.4 Running the Demos from Command Line

You can run the demo applications from the Linux shell in a terminal window connected to the EVM board's serial port. These are the same demos described in Section Starting the Standalone Demos.

If the demos are currently running you should stop them using the S15 key until you see the "Quiting demo interface..." screen.

Before running demo applications from the command line, the CMEM and accelerator kernel modules must be loaded. Use "lsmod" to see if they are loaded. If not, use the following commands to load these modules:

```
target $ cd /opt/dvsdk/omap3530
target $ ./loadmodules.sh
```

To see the command-line options for the demos, use one of the following commands with the -h or --help option:

```
target $ ./interface -h
target $ ./decode -h
```

You can also find the list of command-line options in decode.txt available under /home/<useracct>/dvsdk_#_##_##_##/dvsdk_demos_#_##_##_##/omap3530/decode

The DVSDK provided video and audio files required for playback are available at /opt/dvsdk/omap3530/data/videos and /opt/dvsdk/omap3530/data/sounds folder respectively on the target file system.

After running the demo, if you require unloading the kernel modules previously loaded, use the unloadmodules.sh. Using this shell script unloads the modules in the order it is required and resolves any internal dependencies.

```
target $ ./unloadmodules.sh
```

**Note 1:** While running decode, encode or interface applications from the command line, ensure that no two of these applications are running simultaneously. If this is not taken care of, then there could be memory leaks due to CMEM not freeing up the memory and eventually when applications are run multiple times, it could result in memory allocation failures.

**Note 2:** If the interface application is started on by default, on EVM power on, exit the interface application by pressing S8 in the startup screen to go to the demo menu and then press S15 for quitting the interface application.

## *3.5 Running the Video Encode examples*

The video encode examples are not available as standalone applications, but can be executed from command line. The current encode examples only support file based encode.

If you wish to run the video encode examples, perform the following steps.

1. Set up the OMAP35x DVEVM as given in section OMAP35x DVEVM Software Setup.
2. Rebuild the DVSDK software as mentioned in section Rebuilding the DVSDK software for the target
3. Copy the video encode application to your target NFS work area.

```
host $ cp
/home/<useracct>/dvsdk_#_##_##_##/dmai_#_##_##_##/packages/ti/sdo/dmai/apps/video_encode_
io1/linux/video_encode_io1_omap3530.x470MV
/home/<useracct>/workdir/filesys/opt/dvsdk/omap3530/.

host $ chmod +x
/home/<useracct>/workdir/filesys/opt/dvsdk/omap3530/video_encode_io1_omap3530.x470MV
```

4. The input files for video decode are available at /home/<useracct>/dvsdk_#_##_##_##/clips/omap3530/data/videos where ## are the version numbers.
5. Example execution using the YUV files provided as part of DVSDK
    a. MPEG4 Encoder

    ```
    target $ ./video_encode_io1_omap3530.x470MV -c mpeg4enc -i
    data/videos/davincieffect_qcif_422i.yuv -o output.m4v -r 176x144 -n 5
    ```

    b. H.264 Encoder

    ```
    target $ ./video_encode_io1_omap3530.x470MV -c h264enc -i
    data/videos/davincieffect_qcif_422i.yuv -o output.264 -r 176x144 -n 25
    ```

6. Use the –h option to determine the usage of the encode application and execute the encode application with the necessary parameters

```
target $ ./video_encode_io1_omap3530.x470MV –h
```

**When you run the above command, you will get the following:**

```
Usage: video_encode_io1_<platform> [options]

Options:

     --benchmark      Print benchmarking information

-b | --bitrate        Bitrate used to process video stream [variable]

-c | --codec          Name of codec to use

-e | --engine         Codec engine containing specified codec

-h | --help           Print usage information (this message)

-i | --input_file     Name of input file containing raw YUV

-n | --numframes      Number of frames to process [Default: 100]

-o | --output_file    Name of output file for encoded video

-r | --resolution     Video resolution ('width'x'height')

At a minimum the codec name, the resolution and the file names *must* be given
```

**The table below provides values to guide you for filling the options:**

| Options | MPEG4 SP Encode | H.264 BP Encode |
|---|---|---|
| Name of the codec to use (-c \| --codec) | mpeg4enc | h264enc |
| Resolution (-r \| --resolution) | As specified in the test parameters configuration file for the input | As specified in the test parameters configuration file for the input |
| Number of frames to be encoded | As specified in the test parameters configuration file for the input | As specified in the test parameters configuration file for the input |

## *3.6 Running the Image Encode example*

The image encode examples are not available as standalone applications, but can be executed from command line. The current image encode example only support file based encode.

If you wish to run the image encode example, perform the following steps.

1. Set up the OMAP35x DVEVM as given in section OMAP35x DVEVM Software Setup.
2. Rebuild the DVSDK software as mentioned in section Rebuilding the DVSDK software for the target
3. Copy the image encode application to your target NFS work area.

```
host $ cp
/home/<useracct>/dvsdk_#_##_##_##/dmai_#_##_##_##/packages/ti/sdo/dmai/apps/image_encode_
io1/linux/image_encode_io1_omap3530.x470MV
/home/<useracct>/workdir/filesys/opt/dvsdk/omap3530/.
```

```
host $ chmod +x
/home/<useracct>/workdir/filesys/opt/dvsdk/omap3530/image_encode_io1_omap3530.x470MV
```

4. The input files are already available under folder"""home/<useracct>/workdir/filesys/opt/dvsdk/omap3530/data/images""" for pre-built filesystem images. The input files for image decode are also available at /home/<useracct>/dvsdk_#_##_##_##/clips/omap3530/data/images, where ## are the version numbers.
5. An example execution using the YUV files provided as part of DVSDK is given below.
   a. Execute the image encode DMAI example application. If you look at the "**`Testparams.cfg`**" as mentioned in step 4b, the input file color space format is YUV422 ILE. If you look at step 6, the default input color space is 422P. Hence it is important to set the input color space to 3, which is equivalent to YUV422 ILE.

   ```
   ./image_encode_io1_omap3530.x470MV –c jpegenc –e encode -i
   data/images/remi003_422i.yuv -o Output.jpeg –r 720x576 --iColorSpace 3 --
   oColorSpace 1
   ```

   b. Please note that the setting of the output color space format to YUV422 ILE (oColorSpace value set to 3) is not supported.
6. Use the –h option to determine the usage of the encode application and execute the encode application with the necessary parameters

```
target $ ./image_encode_io1_omap3530.x470MV –h
```

**When you run the above command, you will get the following:**

```
Usage: image_encode_io1_<platform> [options]

Options:

     --benchmark      Print benchmarking information
```

```
-c | --codec          Name of codec to use
-e | --engine         Codec engine containing specified codec
-h | --help           Print usage information (this message)
-i | --input_file     Name of input file to decode
                       and perform cache maintenance.
-o | --output_file    Name of output file containing raw YUV
-r | --resolution     Image resolution ('width'x'height')
   | --qValue         Q compression factor, from 0 (lowest quality) to 100
                       (highest quality). [default: 75]
     --iColorSpace    Colorspace used by input file [1: 422P
                       (default), 2: 420P, 3: UYVY]
     --oColorSpace    Colorspace of encoded output [1: 422P
                       (default), 2: 420P, 3: UYVY]
At a minimum resolution, codec name, and the file names *must* be given.
```

**Use the following values for the test file chosen above:**

| Options | JPEG Encode |
|---|---|
| Name of the codec to use (-c \| --codec) | Jpegenc |
| Resolution (-r \| --resolution) | As specified in the test parameters configuration file for the input |

## 3.7 Running the Audio, Video and Image Decode examples

The decode examples performs the decode operations using the codec server and stores the decoded output in the chosen file in YUV format for image/video and PCM format for audio. These examples are executed from the command line.

NOTE: Since the decoded output is a raw YUV or PCM stream, the output file size will be in hundreds on Megabytes for video and hence ensure that the output is stored in the NFS (Network file system) and not on the NAND flash.

If you wish to run the decode examples, perform the following steps.

1. Set up the OMAP35x DVEVM as given in section 4OMAP35x DVEVM Software Setup.
2. Rebuild the DVSDK software as mentioned in section Rebuilding DVSDK software for the target
3. Copy the decode example applications to your target NFS work area.

```
host $ cp
/home/<useracct>/dvsdk_#_##_##_##/dmai_#_##_##_##/packages/ti/sdo/dmai/apps/video_decode_
io2/linux/video_decode_io2_omap3530.x470MV
/home/<useracct>/workdir/filesys/opt/dvsdk/omap3530/.
```

```
host $ cp
/home/<useracct>/dvsdk_#_##_##_##/dmai_#_##_##_##/packages/ti/sdo/dmai/apps/audio_decode_
io1/linux/audio_decode_io1_omap3530.x470MV
/home/<useracct>/workdir/filesys/opt/dvsdk/omap3530/.
```

```
host $ cp
/home/<useracct>/dvsdk_#_##_##_##/dmai_#_##_##_##/packages/ti/sdo/dmai/apps/image_decode_
io1/linux/image_decode_io1_omap3530.x470MV
/home/<useracct>/workdir/filesys/opt/dvsdk/omap3530/.
```

```
host $ chmod +x
/home/<useracct>/workdir/filesys/opt/dvsdk/omap3530/video_decode_io2_omap3530.x470MV

host $ chmod +x
/home/<useracct>/workdir/filesys/opt/dvsdk/omap3530/audio_decode_io1_omap3530.x470MV

host $ chmod +x
/home/<useracct>/workdir/filesys/opt/dvsdk/omap3530/image_decode_io1_omap3530.x470MV
```

4. Use the –h option to determine the usage of the decode example applications and execute the decode example applications with the necessary parameters

   ```
   target $ ./video_decode_io2_omap3530.x470MV –h

   target $ ./audio_decode_io1_omap3530.x470MV –h

   target $ ./image_decode_io1_omap3530.x470MV –h
   ```

5. The input files for audio decode are available at /home/<useracct>/dvsdk_#_##_##_##/clips/omap3530/data/sounds
6. The input files for video decode are available at /home/<useracct>/dvsdk_#_##_##_##/clips/omap3530/data/videos
7. The input files for image decode are available at /home/<useracct>/dvsdk_#_##_##_##/clips/omap3530/data/images, where ## are the version numbers
8. Given below is the example execution using the media files included in DVSDK. Unless otherwise specified, the DMAI video decode examples support default video output resolution of 720x480 and the number of frames decoded is 100. The DMAI image examples support decoded image output color space as YUV 4:2:2 ILE

   a. MPEG4 SP Decoder

   ```
   target $ ./video_decode_io2_omap3530.x470MV  -c mpeg4dec –i
   data/videos/davincieffect_ntsc_1.m4v -o mpeg4_test_output.yuv
   ```

   b. H.264 BP Decoder

   ```
   target $ ./video_decode_io2_omap3530.x470MV  -c h264dec –i
   data/videos/davincieffect_ntsc_1.264 -o h264_test_output.yuv
   ```

   c. MPEG2 MP Decoder

   ```
   target $ ./video_decode_io2_omap3530.x470MV  -c mpeg2dec –i
   data/videos/davincieffect_ntsc_1.m2v -o mpeg2_test_output.yuv
   ```

   d. JPEG Decode

   ```
   target $ ./image_decode_io1_omap3530.x470MV –c jpegdec -i shrek_720x480.jpg -o
   jpeg_test_output.yuv
   ```

   The resolution is 720x576 and the decoded output color space format is YUV422 ILE.

   e. AAC Audio Decode

   ```
   target $ ./audio_decode_io1_omap3530.x470MV -c aachedec -i
   data/sounds/davincieffect_HEv2.aac -o aac_test_output.pcm
   ```

# 4 OMAP35x DVEVM Software Setup

## *4.1 Software Overview*

To begin developing applications, you need to install the DVEVM development environment. This section outlines the steps required to load the DVEVM software onto the development host. You will need to download the files from the Download page to get started.

The OMAP35x software approach provides interoperable, optimized, production-ready video and audio codecs that leverage DSP and integrated accelerators. These codecs are built into configurable frameworks, and are presented via published APIs within popular operating systems (such as Linux) for rapid software implementation.

The following software is provided with the OMAP35x DVEVM kit.

- **Decode demonstration software.** This is provided on the EVM's NAND flash. The hard-wired examples decode audio and video.
- **DVD1:**
  - Board Support Package
  - Diagnostics
  - Utilities
  - Documentation
    - OMAP35x DVEVM Getting Started Guide (this manual)
    - Hardware User's Guide
- **DVD 2:**
  - DVSDK A/V Demo
  - Graphics SDK Demos
  - DVSDK Tools
  - DSP/BIOS™ Kernel
  - DVSDK Collaterals
- **CD-ROM: Sourcery G++™ evaluation tools from CodeSourcery**

### 4.1.1 Command Prompts in This Guide

In this guide, commands are preceded by prompts that indicate the environment where the command is to be typed. For example:

- host $
  Indicates command to be typed into the shell window of the host Linux workstation.
- `OMAP3EVM #`
  Indicates commands to be typed into the U-Boot shell in a console window connected to the EVM board's serial port. (See Setup Terminal Program)
- `target $`
  Indicates commands to be typed into the Linux shell in the terminal window connected to the EVM board's serial port.

**NOTE: The document lists down various commands that needs to be executed on the target or on the u-boot prompt for various operations required through out the document. Kindly note that a direct copy and paste of these commands might result in insertion of lines for a single command. Hence it would not work on the u-boot prompt or the target. Kindly ensure that you copy the long commands into notepad, remove the line spaces between them and add a space wherever you have removed the line spaces, before recopying and pasting it on the command prompts.**

## *4.2 Requirements*

The following are the requirements for software setup:

- PC Linux Host
  - Root login or liberal sudo privileges
  - Most Linux Distributions should work with this DVSDK
  - This DVSDK has been tested on
    - Red Hat Enterprise Linux, Workstation V4, x86 32 bit
- DVSDK Software Contents

## *4.3 Preparing to Install*

On a host system, mount the OMAP35x EVM demonstration CD's and copy the following .bin files to a temporary location with at least 1 GB free space. Since you can delete the installation files after installing the software, a directory like /tmp is recommended.

- OMAP35x-PSP-SDK-setuplinux-##.##.##.##.bin
- dvsdk_setuplinux_#_##_##_##.bin
- xdctools_setuplinux_#_##_##.bin
- bios_setuplinux_#_##_##.bin
- TI-C6x-CGT-v#.#.##.#.bin
- cs1omap3530_setupLinux_#_##_##-##.bin
- overlay_dvsdk_#_##_##_##.tar.gz
- nfs_dvsdk_#_##_##_##.tar.gz
- rootfs_dvsdk_#_##_##_##.jffs2
- data_dvsdk_#_##_##_##.tar.gz

Updates to these installers may be available on the TI DaVinci Software Updates website listed in the overview topic.

Ensure all the .bin files are set with executable permissions.

```
host $ chmod +x *.bin
```

## 4.4 Installing the Software

Installing the software used by the OMAP35x EVM involves performing the following steps.

*Installing the Target Linux Software*

*Installing the DVSDK Software*

*Installing the A/V demo files*

*Installing the toolchain*

*Exporting a Shared File System for Target Access*

*Testing the Shared File System*

*Notes on using production codecs*

### 4.4.1 Installing the Target Linux Software

This section explains how to install Linux for use on the target board.

Note that separate versions of Linux are used by the target and your host Linux workstation.

The following Linux host operating systems are supported for use with the OMAP35x DVEVM.

- Red Hat Enterprise Linux, Workstation V4, x86 32 bit

To install the Linux software, follow these steps:

1. Execute OMAP35x-PSP-SDK-setuplinux-##.##.##.##.bin file (where ##.##.##.## is the current version number) from the temporary location that they were copied to extract the installers for the Linux tools, Linux kernel, and the file system.

   ```
   host $ cd /tmp
   host $ ./OMAP35x-PSP-SDK-setuplinux-##.##.##.##.bin
   ```

2. The installer will start as a GUI application. Follow the instructions in the dialog boxes. You'll be asked to agree to the End User License Agreement.

3. You will be prompted to provide the installation directory. If installed from a normal user account the default install path will be "/home/<useracct>/OMAP35x-PSP-SDK-##.##.##.##". If installed with root permissions the default install path will be "/opt/OMAP35x-PSP-SDK-##.##.##.##". In future, all references in the document will assume that the user has installed the OMAP35x SDK using his user account and hence will refer to the OMAP35x SDK installation path as /home/<useracct>/OMAP35x-PSP-SDK-##.##.##.##

**Note:** The Linux Support Package (LSP) shipped with the OMAP35x DVEVM is a multi-platform LSP and is not configured for a particular platform. As shipped, this LSP cannot be used to build the demo or example applications. It must first be copied to a user area and configured/built for the OMAP35x EVM. Please see the Rebuilding the Linux Kernel section for instructions.

### 4.4.2 Installing the DVSDK Software

The TI DVSDK software includes demos, example software, Codec Engine components, DSP/BIOS Link, xDAIS and xDM header files, Local Power Manager Module, Framework Components and a contiguous memory allocator for Linux (CMEM). The DVSDK also contains the Digital Video Test Bench that enables testing the codecs and LSP with various configurations that are not possible with the Out of the Box demos.

**Note:** The installers for XDC, DSP/BIOS and Code Generation Tools (codegen) have a different default installation location. However, we strongly recommend that you change the default installation locations to place the components together (if you have not already installed the Linux versions of these components elsewhere). This simplifies the build setup steps.

To install the DVSDK software using the DVSDK Linux installer, follow these steps:

1. Log in using a user account. In the following steps, we refer to the home directory as "~".
2. Execute the DVSDK installer that you previously copied from the DVD 2

   ```
   host $ ./dvsdk_setuplinux_#_##_##_##.bin
   ```

The installer will start as a GUI application. Follow the instructions in the dialog boxes. You'll be asked to agree to the End User License Agreement. When you are prompted for an installation location, use the default installation location, that points to /home/<useracct>/dvsdk_#_##_##_##. This location will be used as the DVSDK installation folder through out this document.

3. Execute the XDC installer that you previously downloaded from the DVSDK download page. For example:

   ```
   host $ ./xdctools_setuplinux_#_##_##.bin
   ```

   When you are prompted for an installation location, do not use the default installation location. Instead, install the software in the directory created in the previous step. For example, /home/<useracct>/dvsdk_#_##_##_##

NOTE: When prompted select **Typical** for the type of Setup.

4. Execute the DSP/BIOS installer that you previously downloaded from the DVSDK download page. For example:

   ```
   host $ ./bios_setuplinux_#_##_##.bin
   ```

   When you are prompted for an installation location, do not use the default location. Instead, install the software in the directory created in step 2. For example, /home/<useracct>/dvsdk_#_##_##_##

5. Execute the Codec Server installer that you have previously downloaded from the DVSDK download page. For example:

   ```
   host $ ./cs1omap3530_setuplinux_#_##_##-##.bin
   ```

   When you are prompted for an installation location, install it in the location as specified or under the directory as created in step 2. For example, /home/<useracct>/dvsdk_#_##_##_##

6. Execute the Code Generation Tools installer that you previously downloaded from the DVSDK download page. For example:

   ```
   host $ ./TI-C6x-CGT-v#.#.##.#.bin
   ```

   When you are prompted for an installation location, install it in the location as specified or under the directory as created in step 2. If you are installing it under the DVSDK installation folder created in step 2, create a new folder under the dvsdk installation folder and then proceed to install. For example, /home/<useracct>/dvsdk_#_##_##_##/cg6x_#_#_##, where #_#_## is the version number.

7. Remember to set the environment variable as directed by the installer. For example: csh: (in the .cshrc file)

   ```
   host $ setenv C6X_C_DIR /home/<useracct>/dvsdk_#_##_##_##/cg6x_#_#_#/include:/home/<useracct>/dvsdk_#_##_##_##/cg6x_#_#_#/lib
   ```

   ksh or bash: (in the .bashrc file)

   ```
   host $ export C6X_C_DIR=/home/<useracct>/dvsdk_#_##_##_##/cg6x_#_#_#/include:/home/<useracct>/dvsdk_#_##_##_##/cg6x_#_#_#/lib
   ```

8. You can now delete the .bin files that you loaded into the temporary directory.

**Note:** You can uninstall these components by using the uninstall file in the respective installation directories. For example:

```
host $ cd /home/<useracct>/dvsdk_#_##_##_##/xdctools_#_#_#

host $ ./uninstall
```

### 4.4.3 Installing the A/V Demo Data

The TI DVSDK software installer, by itself does not contain the media files used by the demos. You can download the data files from DVSDK download page or copy it from the DVD 2. Please use the steps mentioned below for extracting the media files into the desired location.

The Download page as well as the DVD 2 contains the Audio/Video files used by the demo. After following the instructions in the previous section, follow these instructions to install the A/V files:

1. Go to the DVSDK directory that you set up previously. For example

```
host $ cd ~/dvsdk_#_##_##_##/clips
```

2. Copy the A/V data which you have downloaded from Download page to your DVSDK clips directory.For example:

```
host $ cp data_dvsdk_#_##_##_##.tar.gz .
```

3. Extract the A/V data files. For example:

```
host $ tar -xvzf data_dvsdk_#_##_##_##.tar.gz
```

4. Now you can delete the tar file from the clips folder

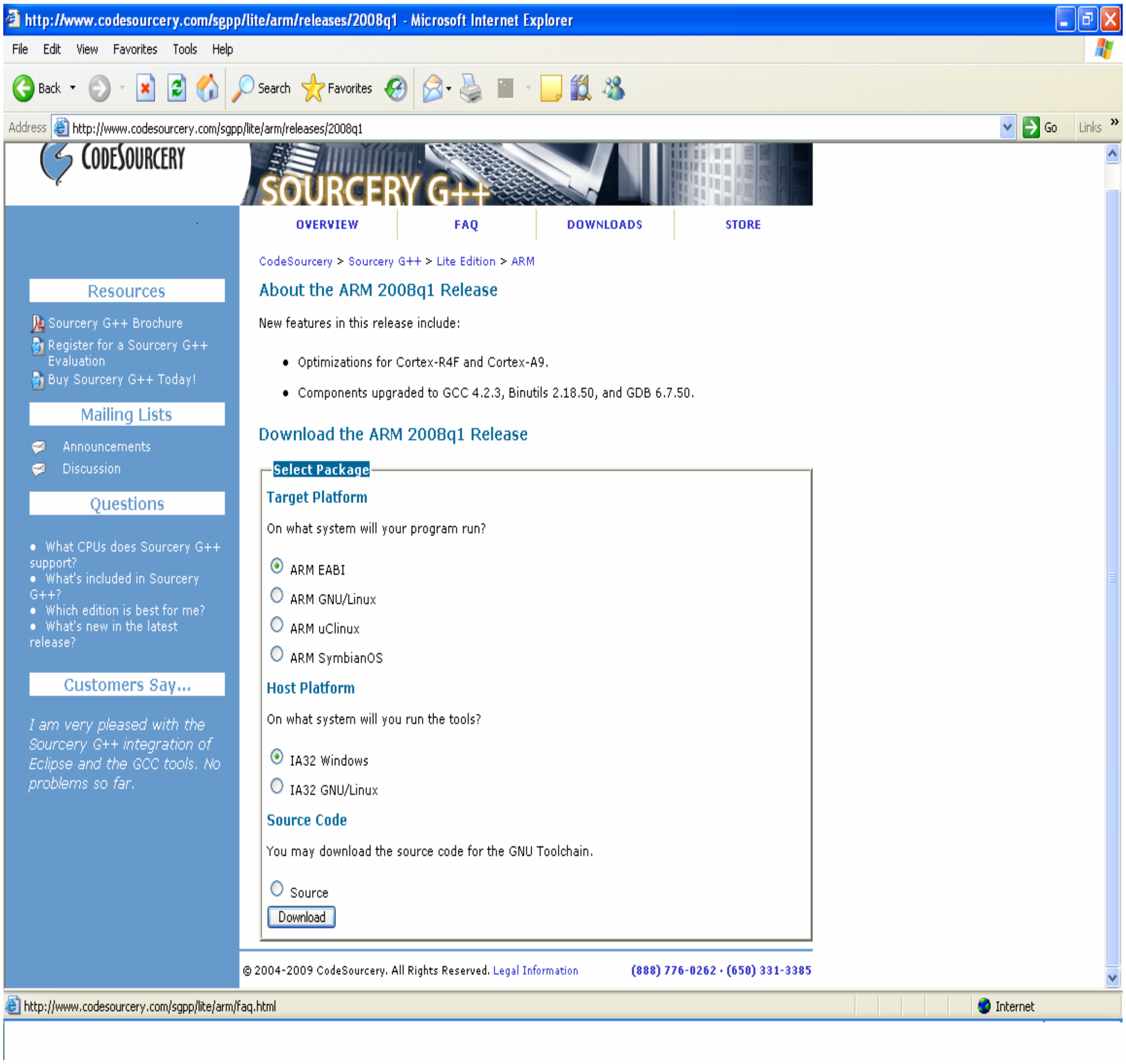
## 4.4.4 Installing the Toolchain

The CD-ROM contains the evaluation version of the CodeSourcery toolchain.

This guide assumes use of the LITE version of the CodeSourcery toolchain. See CodeSourcery documentation if you are using the evaluation version that comes with this EVM or for other cases.

The toolchain used is ARM GNU/Linux EABI 2008q1. It can be downloaded via the following URL:

http://www.codesourcery.com/sgpp/lite/arm/releases/2008q1



- What CPUs does Sourcery G++ support?
- What's included in Sourcery G++?
- Which edition is best for me?
- What's new in the latest release?

*I am very pleased with the Sourcery G++ integration of Eclipse and the GCC tools. No problems so far.*

CodeSourcery > Sourcery G++ > Lite Edition > ARM

### About the ARM 2008q1 Release

New features in this release include:

- Optimizations for Cortex-R4F and Cortex-A9.
- Components upgraded to GCC 4.2.3, Binutils 2.18.50, and GDB 6.7.50.

### Download the ARM 2008q1 Release

**Select Package**

**Target Platform**

On what system will your program run?

● ARM EABI
○ ARM GNU/Linux
○ ARM uClinux
○ ARM SymbianOS

**Host Platform**

On what system will you run the tools?

● IA32 Windows
○ IA32 GNU/Linux

**Source Code**

You may download the source code for the GNU Toolchain.

○ Source

© 2004-2009 CodeSourcery. All Rights Reserved. Legal Information (888) 776-0262 · (650) 331-3385

To install the toolchain, follow the sequence below. These are to be executed on the Linux host platform.

```
$ mkdir –p /home/<useracct>/toolchain

$ cp arm-2008q1-126-arm-none-linux-gnueabi-i686-pc-linux-gnu.tar.bz2 /home/<useracct>/toolchain

$ cd /home/<useracct>/toolchain

$ tar -jxvf  arm-2008q1-126-arm-none-linux-gnueabi-i686-pc-linux-gnu.tar.bz2

$ rm arm-2008q1-126-arm-none-linux-gnueabi-i686-pc-linux-gnu.tar.bz2
```

### 4.4.5 Exporting a Shared File System for Target Access

Although the board's NAND flash contains a file system, during development it is more convenient to have the target board NFS mount a file system on a host Linux workstation. Once you test the application, you can store it on the board's flash for a standalone demonstration.

Before the board can mount a target file system, you must export that target file system on the host Linux workstation. The file system uses an NFS (Network File System) server. The exported file system will contain the target file system and your executables.

To export the file system from your NFS server, perform the following steps. You only need to perform these steps once.

1. Log in with a user account on the host Linux workstation. (In the following steps, we refer to the home user directory as "~".)

2. Perform the following commands to prepare a location for the OMAP35x EVM target file system.

   ```
   host $ cd ~
   host $ mkdir -p workdir/filesys
   host $ cd workdir/filesys
   ```

3. Switch user to "root" on the host Linux workstation.

   ```
   host $ su
   password:
   ```

   There will be a prompt for entering the password as shown above. Type the root password, for getting the root permissions

4. If you have already prepared a directory tree to use for the NFS root file-system (see Section Rebuilding the NFS Image) you can proceed to the next step. Otherwise, copy the prepared version(nfs_dvsdk_#_##_##_##.tar.gz) from the OMAP3530 DVSDK from the Download page to a new directory created in step 2. Perform the following commands.

   **Note:** Un-tar the file with root permissions

   ```
   $ cp /tmp/nfs_dvsdk_#_##_##_##.tar.gz .
   $ tar –xvzf nfs_dvsdk_#_##_##_##.tar.gz
   ```

5. Make sure you can write into the opt folder in the file system by setting the permissions of the opt folder within the target file system with your user account. Perform the following command

   ```
   host $ chown -R <useracct> /home/<useracct>/workdir/filesys/opt
   ```

Alternatively, if you want to have permissions to write or create folders within the target file system that you want to export as NFS, perform the following command

```
host $ chown -R <useracct> /home/<useracct>/workdir/filesys
```

6. Make sure the NFS server is configured and functioning properly. Add the following line to the /etc/exports file of the server. Ensure you have root permission before editing this file.

   ```
   /home/<useracct>/workdir/filesys        *(rw,no_root_squash,no_all_squash,sync)
   ```

   NOTE: On some systems you may wish to add the **no_subtree_check** option to avoid warnings like:

   ```
   exportfs: /etc/exports [3]: Neither 'subtree_check' or 'no_subtree_check' specified for export <export listed here>.
   Assuming default behaviour ('no_subtree_check').
   ```

   NOTE: this default has changed since nfs-utils version 1.0.x

7. Then issue the following command to notify the NFS server about the new exported directory.

   ```
   host $ /usr/sbin/exportfs –a
   host $ /sbin/service nfs restart
   ```

8. Verify that the server firewall is turned off:

   ```
   host $ /etc/init.d/iptables status
   ```

   If the firewall is running, disable it:

   ```
   host $ /etc/init.d/iptables stop
   ```

9. Make sure you exit from having the root permissions after completing all the above steps

   ```
   host $ exit
   ```

### 4.4.6 Testing the Shared File System

To test your NFS setup, follow these steps:

1. Get the IP address of your host Linux workstations as follows. Look for the IP address associated with the eth0 Ethernet port.

   ```
   host $ /sbin/ifconfig
   ```

2. Open a terminal emulation window to connect to the EVM board via RS-232 using the instructions in the Setup Terminal Program topic. If you have a Windows workstation, you can use HyperTerminal. If you have a Linux workstation, you might use Minicom. (You may need to turn on line wrap.)

3. Power on the EVM board, and abort the automatic boot sequence by pressing a key in the console window. This gets you into the U-Boot prompt where you can configure how U-Boot will boot the Linux kernel.

4. Set the following environment variables in the console window:

   ```
   EVM # setenv nfshost <ip address of nfs host>

   EVM # setenv rootpath <directory to mount>
   ```

   **For DVSDK Releases from 3.00.00.36 upto and including 3.00.01.42, use the following bootargs:**

   ```
   EVM # setenv bootargs 'console=ttyS0,115200n8 noinitrd rw ip=dhcp root=/dev/nfs
   nfsroot=$(nfshost):$(rootpath),nolock mem=88M  omapfb.rotate=1 omapfb.rotate_type=1
   omap_vout.vid1_static_vrfb_alloc=y'
   ```

   **For DVSDK release 3.00.02.44, use the following bootargs:**

   ```
   EVM # setenv bootargs 'console=ttyS0,115200n8 noinitrd rw ip=dhcp root=/dev/nfs
   nfsroot=$(nfshost):$(rootpath),nolock mem=99M mpurate=600 omapfb.rotate=1 omapfb.rotate_type=1
   omap_vout.vid1_static_vrfb_alloc=y'
   ```

   Note that the setenv bootargs command should be typed on a single line. Also note that you should avoid using the numeric keypad to enter numbers, as it can sometimes insert extra invisible characters. These environment variables must be typed in perfectly including capitals, if anything is typoed you will likely run into boot errors.

   The <*directory to mount*> must match what you specified in Step 5 of the Exporting a shared file system for target access section. For example, /home/<user>/workdir/filesys.

   **Hints:** If the kernel version stored in the NAND flash on the EVM is out of date you may wish to flash the latest kernel image (using the uImage file in the /home/<useracct>/OMAP35x-PSP-SDK-##.##.##.##/images/kernel directory) or boot using TFTP

   **Hints:** You may want to use the printenv command to print a list of your environment variables. You can also save these setenv commands in a .txt file from which you can paste them in the future.

5. Save the environment so that you don't have to retype these commands every time you cycle power on the EVM board:

   ```
   EVM # saveenv
   ```

6. Boot the board using NFS:

   ```
   EVM # boot
   ```

7. You can now log in as "root" with no password required. See the Alternate boot methods section for information about booting with TFTP, NFS, or the board's NAND flash.

### 4.4.7 Notes on using production codecs

As part of the OMAP35x DVSDK installation, you received a number of codecs:

- MPEG4 Simple Profile Decoder. This decoder is compliant with IVIDDEC2 Interface
- H.264 Base Profile Decoder. This decoder is compliant with IVIDDEC2 Interface
- MPEG2 Main Profile Decoder. This decoder is compliant with IVIDDEC2 Interface
- Sequential JPEG Decoder. This decoder is compliant with IIMGDEC1 Interface
- MPEG4 Simple Profile Encoder. This decoder is compliant with IVIDENC1 Interface
- H.264 Base Profile Encoder. This decoder is compliant with IVIDENC1 Interface
- JPEG Encoder. This encoder is compliant with IIMGENC1 Interface
- Advance Audio Codec (AAC) Decoder. This decoder supports both AAC-HE and AAC-LC configurations. It is compliant with IAUDDEC1 interface
- G.711 Speech Decoder. This decoder is compliant with ISPHDEC1 interface
- G.711 Speech Encoder. This encoder is compliant with ISPHENC1 interface

**NOTE:** Though the standalone codec server contain JPEG decode, the DVSDK decode demos do not support playback of JPEG streams. In order to evaluate these decoders, use Digital Video Test Bench (DVTB) which is included in the OMAP3530 DVSDK. The documentation for using the DVTB is available at /home/<useracct>/dvsdk_#_##_##_##/dvtb_#_##_###/doc/DVTB_User_Guide.pdf, where ## are the version numbers. The media files for MPEG2 MP and JPEG format streams are not available as part of the DVSDK installation, but they are available as part of data_dvsdk_#_##_##_##.tar.gz that can be downloaded from the OMAP software update site http://www.ti.com/omapsoftwareupdates

## *4.5 Setting up the Build/Development Environment*

To set up the development and build environment, follow these steps:

1. Log in to your **user** account (and not as root) on the Linux host system, where the OMAP3530 DVSDK is installed.

2. Add the `/host/<useracct>/toolchain/arm-2008q1/bin` directory to your path. This is typically done by adding an additional line to your shell resource file (~/.bashrc). For the path given above, the line to add to your .bashrc file is:

   ```
   PATH="/home/<useracct>/toolchain/arm-2008q1/bin:$PATH"
   ```

   This adds the CodeSourcery tools to your path and allows you to execute the tools using arm-none-linux-gnueabi-gcc (or other tools in the tool chain) from any directory.

3. Remember to use the following command after modifying your .bashrc file:

   ```
   host $ source /home/<useracct>/.bashrc
   ```

4. You can test that the toolchain is installed correctly by starting a new shell and using the following command:

   ```
   host $ arm-none-linux-gnueabi-gcc –v
   ```

   When you execute this command, you will get an output like the one shown below.

   ```
   Using built-in specs.
   Target: arm-none-linux-gnueabi
   Configured with: /scratch/sandra/lite/src/gcc-4.2/configure --build=i686-pc-linux-gnu --host=i686-pc-linux-gnu
   --target=arm-none-linux-gnueabi --enable-threads --disable-libmudflap --disable-libssp --disable-libgomp --
   disable-libstdcxx-pch --with-gnu-as --with-gnu-ld --enable-languages=c,c++ --enable-shared --enable-
   symvers=gnu --enable-__cxa_atexit --with-pkgversion=Sourcery G++ Lite 2008q1-126 --with-
   bugurl=https://support.codesourcery.com/GNUToolchain/ --disable-nls --prefix=/opt/codesourcery --with-
   sysroot=/opt/codesourcery/arm-none-linux-gnueabi/libc --with-build-
   sysroot=/scratch/sandra/lite/linux/install/arm-none-linux-gnueabi/libc --enable-poison-system-directories --
   with-build-time-tools=/scratch/sandra/lite/linux/install/arm-none-linux-gnueabi/bin --with-build-time-
   tools=/scratch/sandra/lite/linux/install/arm-none-linux-gnueabi/bin
   Thread model: posix
   gcc version 4.2.3 (Sourcery G++ Lite 2008q1-126)
   ```

### 4.5.1 Writing a Simple Program and Running it on OMAP35x EVM

Make sure you have performed the steps in the Exporting a shared file system for target access section and the Setting up the build/development environment section before continuing with the steps in this section.

Perform the following steps on the NFS host system as user (not as root):

1. `host $ mkdir –p ~/workdir/filesys/opt/hello`

2. `host $ cd ~/workdir/filesys/opt/hello`

3. Create a file called hello.c with the following contents:

```
#include <stdio.h>

int main() {
printf("Welcome to OMAP35x World!\n");
return 0;
}
```

4. Build the new C file:

```
host $ arm-none-linux-gnueabi-gcc hello.c -o hello
```

Perform the following steps on the target board. You may use either the target's console window (see Setup Terminal Program) or a telnet session.

1. Move to the new directory on the target:
```
target $ cd /opt/hello
```

2. Run the new executable:
```
./hello
```

The output should be:

```
Welcome to OMAP35x World!
```

# 5 Updating/Restoring the Flash Image

This section describes how to install software images onto the flash memory on the EVM. This procedure can be used to install an updated version of the software from TI, to install a customized version built by the customer or to restore the flash image if it is destroyed or damaged. This procedure will work even if the flash is completely erased.

## *5.1 Requirements*

- A Windows PC
- OMAP35x SDK contents copied from the Linux PC
- Make a copy of the OMAP3503 SDK nfs_dvsdk_#_##_##_##.tar.gz and rootfs_dvsdk_#_##_##_##.jffs2 target file systems under OMAP35x-PSP-SDK-##.##.##.##/images/fs folder in your windows PC.
- Make sure you perform the changes to the reflash-micron.txt and reflash-samsung.txt (depending on the type of memory used) scripts under OMAP35x-PSP-SDK-##.##.##.##/scripts, on your Windows PC as mentioned in Section Running the Re-flash Procedure, Step 5.1.
- Make sure you perform the changes to the initenv-micron.txt and initenv-samsung.txt (depending on the type of memory used) scripts under OMAP35x-PSP-SDK-##.##.##.##/scripts, on your Windows PC as mentioned in Section Running the Re-flash Procedure, Step 6.

## *5.2 Preparing to Install*

If the EVM has an active u-boot, please disable it from booting the kernel by following the steps given below.

1. Ensure that the host PC is connected to the board via UART as mentioned in Section 2.4.
2. Power On the board. Once you power on the board, the following will be printed on the hyperterminal or teraterm.

```
Texas Instruments X-Loader 1.41

Detected Samsung MuxOneNAND1G Flash

Starting OS Bootloader...

U-Boot 1.1.4 (Jun 5 2008 - 17:53:37)

OMAP3-GP rev 2, CPU-OPP2 L3-165MHz

OMAP3EVM 1.0 Version + mPOP (Boot ONND)

DRAM: 128 MB

OneNAND Manufacturer: Samsung (0xec)

Muxed OneNAND 128MB 1.8V 16-bit (0x30)

OneNAND version = 0x0221

Scanning device for bad blocks

num of blocks = 2048

In: serial

Out: serial

Err: serial

Resetting CHIP... Done
```

```
LAN9x18 (0x01150002) detected.
Setting mac address: 00:50:c2:7e:85:66
start Auto negotiation... (take ~2sec)
Auto negotiation complete, 100BaseTX, full duplex
```

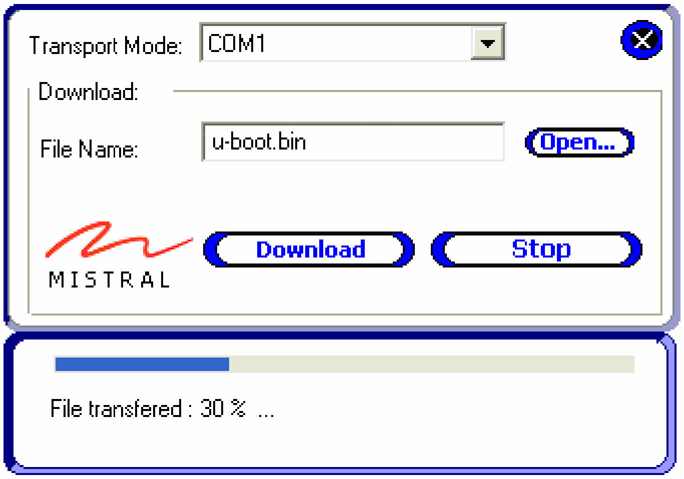
3. Stop the boot process by pressing the Enter key on the development PC keyboard. The u-boot prompt will appear.
4. Disable the u-boot from booting the kernel by using the following commands at the u-boot prompt.

```
OMAP3EVM # setenv bootcmd echo no boot command configured
OMAP3EVM # setenv bootdelay 99
OMAP3EVM # saveenv
```

**NOTE: Make sure you follow this step. The download will not work and button S2 on the EVM will not operate if the Linux kernel begins to boot.**

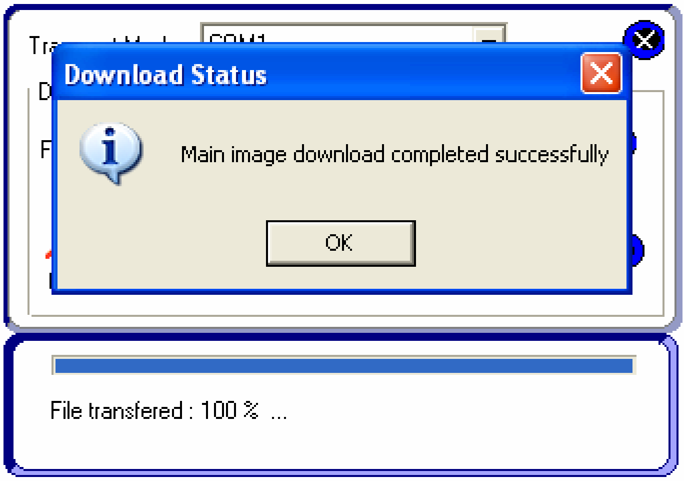
## *5.3 Downloading U-boot via the UART Downloader*

1) Turn off the EVM by unplugging the power adapter.
2) Close any open terminal programs.
3) Move the serial port cable to UART3.
4) Ensure SW4 is set as described in Section 2.1 Main Board SW
5) Start the DownloadUtility.exe program in OMAP35x-PSP-SDK-##.##.##.##/host-tools/windows/utilities
6) Select the correct COM port
7) Press the “open” button and select the following file:
    - OMAP35x-PSP-SDK-##.##.##.##/images/u-boot
8) Press the Download button,  Press OK on the dialog that say “Press OK and reset the target”
9) Power on the EVM
10) If the download does not immediately start, Press S2 on the EVM
11) The download utility (DownloadUtility.exe) should show progress and eventually will show “Main Image has been sent”
12) Close Download Utility (DownloadUtility.exe) and restart your terminal program. **Do not power cycle the EVM.**
13) Move the serial cable to UART1 and press enter to get a U-boot prompt



File transfered : 30 % ...

Download in progress



Main image download completed successfully

File transfered : 100 % ...

Download Complete

**NOTE:** If you do not get the uboot prompt on pressing "Enter" after connecting the serial port cable to UART1, check if you have followed all the above steps properly. If the problem persists, your NAND could be corrupted. In the event of NAND corruption, perform SD/MMC card boot using the steps mentioned in OMAP35x_DVEVM_Boot_from_SD/MMC_card and erase the NAND using the following command

For Micron NAND,

```
OMAP3EVM# nand erase
```

For Samsung OneNAND,

```
OMAP3EVM# onenand erase
```

## 5.4 Using U-boot to Update the Flash Memory

Now that U-boot is running on the target, you can use it to update any or all of the software in flash. The steps provided here will use the Ethernet connection with the TFTP protocol to transfer the files.

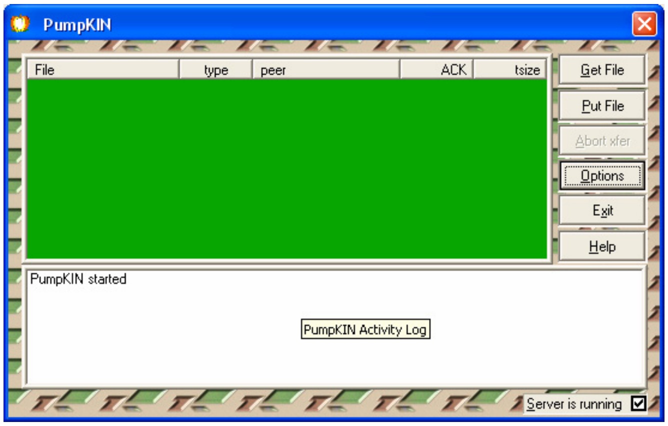
### 5.4.1 Setup the TFTP Server

This section will describe setting up the TFTP server using a Windows host. You may also use a Linux TFTP server but that process is not described here.

The OMAP35x SDK comes with the PumpKIN TFTP server. This program is easy to setup and can be started and stopped easily.

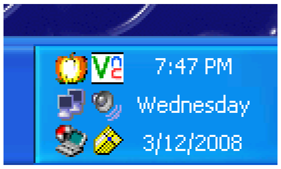
To setup PumpKIN follow these steps:

1. Start the PumpKIN executable. The first time it is started it should show a Window. Subsequent starts may only show in the system tray. You may open the window from the right click menu of the system tray icon.
2. Hit the Options button and configure the path to the base of the SDK directory structure.
3. Select “Allow Access to sub-directories” and “Give all files”
4. Hit OK

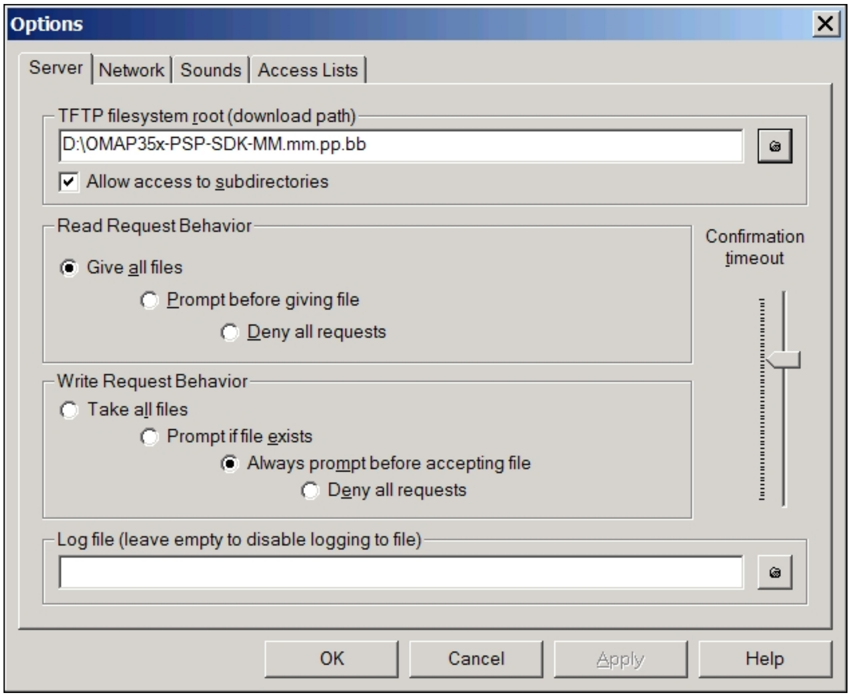


PumpKIN started

PumpKIN Window



PumpKIN Icon in System Tray



Options

Server | Network | Sounds | Access Lists

TFTP filesystem root (download path)

D:\OMAP35x-PSP-SDK-MM.mm.pp.bb

☑ Allow access to subdirectories

Read Request Behavior

● Give all files
○ Prompt before giving file
○ Deny all requests

Write Request Behavior

○ Take all files
○ Prompt if file exists
● Always prompt before accepting file
○ Deny all requests

Log file (leave empty to disable logging to file)

PumpKIN Options Dialog

### 5.4.2 Running the Re-flash Procedure

1. Ensure that the terminal program is configured with ~ 100 ms of delay after sending each line as described in Section Setup Terminal Program

2. Ensure the environment is set as appropriate for your EVM and network setup

```
OMAP3EVM # setenv ethaddr   00:50:c2:XX:XX:XX          (see sticker on EVM)
OMAP3EVM # setenv ipaddr    192.168.1.10               (as appropriate)
OMAP3EVM # setenv serverip  192.168.1.100              (PumpKIN PC's addr)
OMAP3EVM # setenv netmask   255.255.255.0              (as appropriate)
OMAP3EVM # setenv gatewayip 192.168.1.1                (if needed)
OMAP3EVM # nand unlock                                          (ONLY MICRON!)
OMAP3EVM # saveenv
```

3. The windows IP address can be obtained using the following steps
   - Go to Start->Run
   - Type cmd to open the command window
   - Type ipconfig on command prompt to get the IP address of the Windows host PC

4. Check the IP settings

```
OMAP3EVM # ping $(serverip)                    (expect “x.x.x.x is alive”)
```

5. Use the terminal program to send the text file reflash.txt file to the EVM
   - Make sure you know if your board has Samsung or Micron memory parts
     - See Section Identifying Board Variations
   - path: OMAP35x-PSP-SDK-##.##.##.##/scripts/
     - reflash-micron.txt (for Micron memory boards)
     - reflash-samsung.txt (for Samsung memory boards)

   **Note:**The scripts are saved in Unix format.If you want to run from windows please convert it to dos format using unix2dos command in the scripts directory.Usage as given below:

   ```
   unix2dos *
   ```

   If you need to use these scripts for re-flashing the demos that come along with the OMAP3530 DVSDK, the following changes are required to be modified in the re-flash scripts. The changes are marked in bold. The example is shown for only reflash-samsung.txt. Performing similar changes for reflash-micron.txt is needed if Micron-NAND OMAP35x EVM is being used.

```
setenv rf_unlock      onenand unlock 0x0 0x8000000
setenv rf_blank_ram   mw.b 0x81600000 0xff 0x1400000
setenv rf_er_xloader  onenand erase block 0-3
setenv rf_er_uboot    onenand erase block 4-18
setenv rf_er_env      onenand erase block 19-19
setenv rf_er_kernel   onenand erase block 20-59
setenv rf_er_ramdisk  onenand erase block 60-1023
setenv rf_er_fs       onenand erase block 60-1023
setenv rf_get_xloader tftpboot 0x81600000 images/boot-strap/x-load.bin.ift
setenv rf_get_uboot   tftpboot 0x81600000 images/u-boot/u-boot.bin
setenv rf_get_kernel  tftpboot 0x81600000 images/kernel/uImage
setenv rf_get_ramdisk tftpboot 0x81600000 images/fs/ramdisk.gz
setenv rf_get_min_rd  tftpboot 0x81600000 images/fs/ramdisk-base.gz
setenv rf_get_fs      tftpboot 0x81600000 images/fs/rootfs.jffs2
setenv rf_wr_xloader  onenand write 0x81600000 0x000000 0x0020000
setenv rf_wr_uboot    onenand write 0x81600000 0x080000 0x0040000
setenv rf_wr_kernel   onenand write 0x81600000 0x280000 0x0220000
setenv rf_wr_ramdisk  onenand write 0x81600000 0x780000 0x1000000
setenv rf_wr_fs       onenand write 0x81600000 0x780000 0x1400000
setenv rf_ev          run rf_unlock\; run rf_er_env
setenv rf_xl          run rf_unlock\; run rf_blank_ram\; run rf_er_xloader\; run rf_get_xloader\; run
rf_wr_xloader
```

```
setenv rf_ub        run rf_unlock\; run rf_blank_ram\; run rf_er_uboot\; run rf_get_uboot\; run
rf_wr_uboot

setenv rf_kn        run rf_unlock\; run rf_blank_ram\; run rf_er_kernel\; run rf_get_kernel\; run
rf_wr_kernel

setenv rf_rd        run rf_unlock\; run rf_blank_ram\; run rf_er_ramdisk\; run rf_get_ramdisk\; run
rf_wr_ramdisk

setenv rf_min_rd     run rf_unlock\; run rf_blank_ram\; run rf_er_ramdisk\; run rf_get_min_rd\; run
rf_wr_ramdisk

setenv rf_fs        run rf_unlock\; run rf_blank_ram\; run rf_er_fs\; run rf_get_fs\; run rf_wr_fs

setenv rf_all_rd     run rf_xl\; run rf_ub\; run rf_kn\; run rf_rd

setenv rf_all_min_rd  run rf_xl\; run rf_ub\; run rf_kn\; run rf_min_rd

setenv rf_all_fs     run rf_xl\; run rf_ub\; run rf_kn\; run rf_fs

printenv

echo done

echo Make sure to set MAC address
```

If you need to re-flash the target file systems that contains the demos that come along with the OMAP3530 DVSDK, or your own target file systems, refer section Flashing the Pre-built/rebuilt NAND Flash File System

- For HyperTerm use Transfer / Send Text File …
- For TeraTerm use File / Send file …
- For minicom use the paste file command (Ctrl-A Y)

6. Run the reflash command sequence

```
OMAP3EVM # run rf_all_fs (for jffs file-system)

OMAP3EVM # run rf_all_rd (for ramdisk)
```

(Individual components can be reflashed by using commands like rf_ub, rf_kn, etc)

**Note:** For reflashing choices for Micron nand please select the commands which are marked by _new.(For example 'rf_all_fs_new' for using JFFS filesystem and 'rf_all_rd_new' for using full ramdisk).

**Note:** Ramdisk should be used for Micron boards as JFFS produces warnings and very long boot times.

**Note:** If the flashed rootfs.jffs2 or the NFS setup uses the target file system which is present in the CD , or uses the NFS setup, which has demos that have dependency on the CMEM module, then the bootargs within the initenv-xxx.txt scripts need to be modified as follows. The changes are reflected in bold.

**The bootargs for DVSDK releases from 3.00.00.36 up to and including 3.00.01.42 are as given below**

```
setenv bootargs_rd    mem=88M console=ttyS0,115200n8 root=/dev/ram0 initrd=0x81600000,16M
ramdisk_size=40960 omapfb.rotate=1 omapfb.rotate_type=1 omap_vout.vid1_static_vrfb_alloc=y
```

```
setenv bootargs_fs     mem=88M console=ttyS0,115200n8 noinitrd root=/dev/mtdblock4 rw rootfstype=jffs2
omapfb.rotate=1 omapfb.rotate_type=1 omap_vout.vid1_static_vrfb_alloc=y
```

```
setenv bootargs_nfs    mem=88M console=ttyS0,115200n8 noinitrd ip=dhcp rw root=/dev/nfs
nfsroot=<nfsserver_ip>:/home/<useracct>/workdir/filesys,nolock omapfb.rotate=1 omapfb.rotate_type=1
omap_vout.vid1_static_vrfb_alloc=y
```

**The bootargs for DVSDK releases from 3.00.02.44 are as given below**

```
setenv bootargs_rd     mem=99M console=ttyS0,115200n8 root=/dev/ram0 initrd=0x81600000,16M
ramdisk_size=40960 mpurate=600 omapfb.rotate=1 omapfb.rotate_type=1
omap_vout.vid1_static_vrfb_alloc=y
```

```
setenv bootargs_fs     mem=99M console=ttyS0,115200n8 noinitrd root=/dev/mtdblock4 rw rootfstype=jffs2
mpurate=600 omapfb.rotate=1 omapfb.rotate_type=1 omap_vout.vid1_static_vrfb_alloc=y
```

```
setenv bootargs_nfs    mem=99M console=ttyS0,115200n8 noinitrd ip=dhcp rw root=/dev/nfs
nfsroot=<nfsserver_ip>:/home/<useracct>/workdir/filesys,nolock mpurate=600 omapfb.rotate=1
omapfb.rotate_type=1 omap_vout.vid1_static_vrfb_alloc=y
```

If the script is not used, use the above appropriate command (depending on ramdisk, NAND flash or NFS file system) to set the bootargs by using the following command

For Ramdisk:

```
setenv bootargs ‘bootargs_rd’
```

For NFS:

```
setenv bootargs ‘bootargs_nfs’
```

For NAND Flash:

```
setenv bootargs ‘bootargs_fs’
```

Even if the script is not being used, it is important to set the bootargs in u-boot params to use **mem=88M** (in case of DVSDK releases from v3.00.00.33 to 3.00.01.42) and **mem=99M** (in case of DVSDK release 3.00.02.44 onwards) or the appropriate value that is equal to the (size of DDR used in OMAP35x EVM – size allocated for CMEM). Also, it is required to add **omapfb.rotate=1 omapfb.rotate_type=1 omap_vout.vid1_static_vrfb_alloc=y** to the bootargs for proper display of OSD on the LCD screen. 5

7. Reset the board Note: The current U-boot only allows 40 environment variables. Do not saveenv after the reflash script has been submitted. Instead complete the reflash and reboot before making env changes.
8. Use the terminal program to send the text file initenv.txt to the EVM

   path: OMAP35x-PSP-SDK-##.##.##.##/scripts/…

   - initenv-micron.txt (for Micron memory boards)
   - initenv-samsung.txt (for Samsung memory boards)

9. Save the environment

   ```
   OMAP3EVM # nand unlock                                   (ONLY MICRON!)
   OMAP3EVM # saveenv
   ```

10. Reset the board and it should boot into the kernel and file-system

# 6 Rebuilding the Software

This section describes how to rebuild pieces of the software delivery. Ensure that the toolchain setup and install has been completed (see section Installing the Toolchain).

## *6.1 Rebuilding U-boot*

Rebuilding the U-boot is described in more detail in the LSP User's Guide available at

OMAP35x-PSP-SDK-##.##.##.##/docs/UserGuide-##.##.##.##.pdf

Copy the U-Boot source archive from OMAP35x-PSP-SDK-##.##.##.##/src/u-boot directory into your working directory

```
host $ cd ~

host $ mkdir –p workdir/opt

host $ cp /home/<useracct>/OMAP35x-PSP-SDK-#.##.##.##/src/u-boot/u-boot-##.##.##.##.tar.gz ~/workdir/opt/.
```

Un-tar the U-Boot source archive by performing the command given below.

```
host $ cd workdir/opt

host $ tar -zxvf u-boot-##.##.##.##.tar.gz
```

To configure U-Boot for building, issue the following commands:

```
host $ cd uboot-##.##.##.##

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

host $ make CROSS_COMPILE=arm-none-linux-gnueabi- ARCH=arm omap3_evm_config
```

Once ready to build U-Boot, run the command:

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

The resulting U-Boot image named u-boot.bin will be in the current directory, ready to be loaded on the target.

## *6.2 Rebuilding the Linux Kernel*

Rebuilding the kernel is described in more detail in the User's Guide available at OMAP35x-PSP-SDK-##.##.##.##/docs/UserGuide-##.##.##.##.pdf

Note that building the kernel requires using mkimage, a host side utility built by the u-boot makefile. If you have not already done so please build u-boot using the instructions in the Rebuilding U-Boot section.

You will need to build U-boot and place mkimage in your Path. A command like below can be used. If you are using a bash shell, kindly add the following in the .bashrc file in your user account.

```
export PATH=/<path_to_uboot>/tools:$PATH
```

Copy the Linux kernel source archive from OMAP35x-PSP-SDK-##.##.##.##/src directory into your working directory.

```
host $ cd ~

host $ mkdir –p workdir/opt

host $ cp /home/<useracct>/OMAP35x-PSP-SDK-##.##.##.##/src/kernel/linux-##.##.##.##.tar.gz ~/workdir/opt/.
```

Un-tar the Linux kernel source archive by performing the commands given below.

```
host $ cd /home/<useracct>/workdir/opt

host $ tar -zxvf linux-##.##.##.##.tar.gz
```

In order to prepare the kernel for building, follow these commands:

```
host $ cd linux-##.##.##.##

host $ make CROSS_COMPILE=arm-none-linux-gnueabi- ARCH=arm omap3_evm_defconfig
```

If you need to make changes to the default configuration, perform the following command.

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

Once ready to build the kernel, run the following command

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

The resulting kernel image uImage will be placed under arch/arm/boot and is ready to be loaded on the target.

## 6.3 Rebuilding the DVSDK Software for the target

To place demo files in the /opt/dvsdk directory, you need to rebuild the DVSDK software. To do this, follow these steps:

1. If you have not already done so, rebuild the Linux kernel as described in the Rebuilding the Linux Kernel section.

2. Change directory to dvsdk_#_##_##_##.

3. Edit the Rules.make file in the dvsdk_#_##_##_## directory.

```
host $ vi Rules.make
```

4. Check the following directory definitions. If you installed components in the default locations, the directory definitions may already be correct, but you should verify them in any case.

   - Set PLATFORM to match your EVM board as follows (Note that this variable is case sensitive):

```
PLATFORM=omap3530
```

   - Set DVSDK_INSTALL_DIR to the top-level DVSDK installation directory as follows, note that by default ${HOME} refers to your /home/<*useracct*> directory:

```
DVSDK_INSTALL_DIR=/home/<useracct>/dvsdk_#_##_##_##
```

   - Modify the following variable as needed to match the location of XDCtools on your Linux host. We recommend that XDCtools be installed in the /home/<*useracct*>/dvsdk_#_##_##_## directory, but you may have installed it elsewhere.

```
XDC_INSTALL_DIR=/home/<useracct>/dvsdk_#_##_##_##/xdctools_#_##_##_##
```

   - Make sure EXEC_DIR points to the opt directory on the NFS exported file system as follows. Refer Exporting a Shared File System for Target Access for setting up the NFS exported file system.

```
EXEC_DIR=/home/<useracct>/workdir/filesys/opt/dvsdk/omap3530
```

   - Make sure UBOOT_INSTALL_DIR is defined as follows so it points to where you copied the uboot source in the Rebuilding U-Boot section.

```
UBOOT_INSTALL_DIR=/home/<useracct>/workdir/opt/uboot-##.##.##.##
```

   - Make sure LINUXKERNEL_INSTALL_DIR is defined as follows so it points to where you copied the Linux kernel tree in the Building a New Linux Kernel section.

```
LINUXKERNEL_INSTALL_DIR=/home/<useracct>/workdir/opt/linux-##.##.##.##
```

   - Make sure OMAP3503_SDK_INSTALL_DIR is defined as follows. (If you have installed it in a different location, ensure you set the correct path).

```
OMAP3503_SDK_INSTALL_DIR=/home/<useracct>/OMAP35x-PSP-SDK-##.##.##.##
```

- Make sure CODEC_INSTALL_DIR is defined as follows. (If you have installed it in a different location, ensure you set the correct path).

  ```
  CODEC_INSTALL_DIR=/home/<useracct>/dvsdk_#_##_##_##/cs1omap3530_#_##_##
  ```

- Change the path of the CSTOOL_DIR to point to the location where you have installed the CodeSourcery tool-chain. (Refer Installing the Toolchain)

  ```
  CSTOOL_DIR=/home/<useracct>/toolchain/arm-2008q1
  ```

- Make sure CODEGEN_INSTALL_DIR is defined as follows. (If you have installed it in a different location, ensure you set the correct path).

  ```
  CODEGEN_INSTALL_DIR=/home/<useracct>/dvsdk_#_##_##_##/cg6x_#_#_##
  ```

Check the path of all component installation directories to match the location where you want to point to. The default locations specified will choose the component versions as installed as part of the DVSDK installation

5. The top level DVSDK Makefile re-builds the kernel modules required by the DVSDK demonstration software (CMEM and LPM) and hence it is very important to at least perform the following commands (in case you have not performed the steps in Rebuilding the Linux Kernel), after extracting the Linux kernel source code in your work directory, whose path is provided in the top level DVSDK Rules.make. Refer section Rebuilding the Linux Kernel for more details on “Rebuilding the Linux Kernel”.

   ```
   host $ cd /home/<useracct>/workdir/opt/linux-##.##.##.##
   host $ make CROSS_COMPILE=arm-none-linux-gnueabi- ARCH=arm omap3_evm_defconfig
   host $ make CROSS_COMPILE=arm-none-linux-gnueabi- ARCH=arm modules
   ```

6. Once the installation is complete please use the following commands to verify the paths set in Rules.make and components installed in DVSDK are proper.

   ```
   host $ make check
   host $ make info
   ```

7. While in the same directory that contains Rules.make, use the following commands to build the DVSDK demo applications and put the resulting binaries on the target file system specified by EXEC_DIR.

   ```
   host $ make clean
   host $ make all
   host $ make install
   ```

   **Note :** The **make all** command builds only the DVSDK demos, its dependent components and DVTB

8. Additional commands for cleaning and building all the components are as mentioned below.
   **Note :** The PSP examples under OMAP3503_SDK_INSTALL_DIR/src/examples have to be extracted before using below commands

   ```
   host $ make clobber
   host $ make everything
   ```

9. For information on individually building the components please use the folowing command

**Note:**Certain components have dependency on other components.So please make sure you have executed the **make all** command before proceeding for individual component build.

```
host $ make help
```

**NOTE:** The dependencies for indivdual component builds are not addressed in the top level DVSDK Makefile. If you perform 'make clobber' or 'make clean' and then try building indivdual components like 'make dmai', the build will fail. Alternatively, if you perform 'make clobber' and then perform 'make everything', the build will go through. In addition to that, if you try cleaning the individual component and then build the same again, the build will go through. For example, after performing 'make everything', performing 'make dmai_clean' followed by 'make dmai' will work.

## *6.4 Rebuilding the Initial NAND X-loader*

Copy the X-Loader source archive from OMAP35x-PSP-SDK-##.##.##.##/src/boot-strap directory into your working directory

```
host $ cd ~

host $ mkdir –p workdir/opt

host $ cp x-loader-##.##.##.##.tar.gz ~/workdir/opt/.
```

Un-tar the X-Loader source archive by performing the following command.

```
host $ cd /home/<useracct>/workdir/opt

host $ tar -zxvf x-loader-##.##.##.##.tar.gz
```

Configure X-Loader for OMAP35x EVM target:

```
host $ cd xloader-##.##.##.##

host $ make omap3evm_config
```

To build X-Loader, run the command:

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

The above command produces the x-load.bin file, but in order for the X-Loader to be loaded by the OMAP35x ROM bootloader, it needs to be signed with the signGP program from the OMAP35x-PSP-SDK-##.##.##.##/host-tools/linux directory.

```
host $ /home/<useracct>/OMAP35x-PSP-SDK-#.##.##.##/host_tools/linux/signGP x-load.bin
```

The resulting signed X-loader is named x-load.bin.ift and should be ready for the target.

For MMC/SD Card boot loading this file must be called MLO.

```
host $ cp x-load.bin.ift MLO
```

## *6.5 Rebuilding the Target-side UART Loader*

Copy the UART loader dnld_util_target.tar.bz2 file from OMAP35x-PSP-SDK-##.##.##.##/src/utils directory into your working directory.

```
host $ cd ~

host $ mkdir –p workdir/opt

host $ cp dnld-util-target.tar.bz2 ~/workdir/opt/.
```

Un-tar the UART loader by performing the following command.

```
host $ tar -jxvf dnld_util_target.tar.bz2

host $ cd dnld_util_target
```

Run the following command to build it.

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

The resulting binary is called dnld_startup_omap3_evm.bin. This file is used with DownloadUtility.exe for UART peripheral boot.

## *6.6 Rebuilding the NFS Image*

Kindly refer to the OMAP3503_SDK_GettingStartedGuide.pdf (Getting Started Guide for OMAP3503 SDK) available under ~/OMAP35x-PSP-SDK-##.##.##.##/docs, for rebuilding NFS for OMAP 3503 SDK. This will contain different set of demos and sample applications. The OMAP3530 DVSDK installer has the script to (re)build the NFS image that includes the OMAP3530 DVSDK demos under ~/dvsdk_#_##_##_##/targetfs folder.

In order to rebuild the NFS image with OMAP3530 DVSDK demos, make sure you have performed the steps in Section OMAP35x DVEVM Software Setup and Section Rebuilding the DVSDK Software for the target.

**NOTE:** Also ensure that you have copied the overlay_dvsdk_#_##_##_##.tar.gz file to your /home/<useracct>/dvsdk_#_##_##_##/targetfs folder.

The following commands removes the old the NFS tarball image:

```
host $ cd /home/<useracct>/dvsdk_#_##_##_##/targetfs

host $ sudo make –f targetfs_make clean_old_rootfs DVSDKVER=<your version number #_##_##_##>
```

The following commands will create the NFS tarball image:

```
host $ cd /home/<useracct>/dvsdk_#_##_##_##/targetfs

host $ sudo make –f targetfs_make omap3530_dvsdk_nfs DVSDKVER=<your version number #_##_##_##>
```

The following commands will remove the NFS creation logs and the temporary NFS directory:

```
host $ cd /home/<useracct>/dvsdk_#_##_##_##/targetfs
host $ sudo make –f targetfs_make clean_nfs DVSDKVER=<your version number #_##_##_##>
```

For any clarification about the procedure please perform the below command

```
host $ make –f targetfs_make help
```

Executing above command requires the users to have sudo access. The linux host would prompt for the user password and the user needs to enter his password for the make to continue. The user can make sure he has sudo access by including the following line in /etc/sudoers file.

```
<useracct>      ALL=(ALL)      ALL
```

Executing above commands will create the new NFS file system with the DVSDK demo executables and all the media files required by the demonstrations. You will see nfs_dvsdk_#_##_##_##.tar.gz created under the current folder (**/home/<useracct>/dvsdk_#_##_##_##/targetfs**).

## *6.7 Rebuilding the Full Ramdisk Image*

Kindly refer to the OMAP3503_SDK_GettingStartedGuide.pdf (Getting Started Guide for OMAP3503 SDK) available under ~/OMAP35x-PSP-SDK-##.##.##.##/docs, for rebuilding full ramdisk image for OMAP 3503 SDK. This will contain different set of demos and sample applications.

The full ramdisk image cannot be built to include the OMAP3530 DVSDK demos as the ramdisk size requirement exceeds 40MB after the image is un-tarred and the bootargs given in step 5 in Section Running the Re-flash Procedure will not work

## *6.8 Rebuilding the Minimal Ramdisk Image*

Kindly refer to the OMAP3503_SDK_GettingStartedGuide.pdf (Getting Started Guide for OMAP3503 SDK) available under ~/OMAP35x-PSP-SDK-##.##.##.##/docs, for rebuilding minimal ramdisk image for OMAP 3503 SDK as well as OMAP3530 DVSDK. The minimal ramdisk image does not contain any demos or examples.

## *6.9 Rebuilding the JFFS2 File-System Image*

Kindly refer to the OMAP3503_SDK_GettingStartedGuide.pdf (Getting Started Guide for OMAP3503 SDK) available under ~/OMAP35x-PSP-SDK-##.##.##.##/docs, for rebuilding JFFS2 file system image for OMAP 3503 SDK. This will contain different set of demos and sample applications.

The OMAP3530 DVSDK installer has the pre-built JFFS2 root file system image that includes the OMAP3530 DVSDK demos under ~/dvsdk_#_##_##_##/targetfs folder.

Check if the linux distribution on the host PC contains the mkfs.jffs2 utility. If not, download the binary from ftp://sources.redhat.com/pub/jffs2/mkfs.jffs2. Place this binary under /sbin directory in the Linux host development PC and ensure to include /sbin to the PATH environment variable in the .bashrc file.

In order to rebuild the JFFS2 root file system image with OMAP3530 DVSDK demos, make sure you have performed the steps in Section OMAP35x DVEVM Software Setup and Section Rebuilding the DVSDK Software for the target. Then, perform the following steps.

**NOTE:** Also ensure that you have copied the overlay_dvsdk_#_##_##_##.tar.gz file to your /home/<useracct>/dvsdk_#_##_##_##/targetfs folder.

Clean the old file system images:

```
host $ cd /home/<useracct>/dvsdk_#_##_##_##/targetfs
host $ sudo make –f targetfs_make clean_old_rootfs DVSDKVER=<your version number #_##_##_##>
```

Create the JFFS2 Image (This requires building the NFS image to create the JFFS2 image from):

```
host $ cd /home/<useracct>/dvsdk_#_##_##_##/targetfs
host $ sudo make –f targetfs_make omap3530_dvsdk_jffs2 MKJFFS2=<your_jffs2_install_path> DVSDKVER=<your version number #_##_##_##>
```

Clean up the temporary NFS file system directory:

```
host $ cd /home/<useracct>/dvsdk_#_##_##_##/targetfs
host $ sudo make –f targetfs_make clean_nfs DVSDKVER=<your version number #_##_##_##>
```

For any clarification about the procedure please perform the below command

```
host $ make –f targetfs_make help
```

Executing above commands requires the users to have sudo access. The linux host would prompt for the user password and the user needs to enter his password for the make to continue.

The user can make sure he has sudo access by including the following line in /etc/sudoers file.

```
<useracct>      ALL=(ALL)      ALL
```

Executing above commands will create the new NFS and JFFS2 root file system with the DVSDK demo executables and all the media files required by the demonstrations. You will see nfs_dvsdk_#_##_##_##.tar.gz and rootfs_dvsdk_#_##_##_##.jffs2 files created under the current folder (**/home/<useracct>/dvsdk_#_##_##_##/targetfs**).

## *6.10 Rebuilding the DSP side server executables*

The DSP side codec libraries are available under /home/<useracct>/dvsdk_#_##_##_##/cs1omap3530_#_##_## directory or if you have installed it in a different location specified by CODEC_INSTALL_DIR

The top level Makefile and the config.bld files are provided under the top level directory mentioned above. The codec binaries, header files as well as documents are RTSC packaged and available under /home/<useracct>/dvsdk_#_##_##_##/cs1omap3530_#_##_##/packages/ti/sdo/codecs folder.

Ensure that the path settings for the DSP side code generation tools are set properly in the config.bld file

```
C64P.rootDir = /home/<useracct>/dvsdk_#_##_##_##/cg6x_#_#_# OR
```

```
C64P.rootDir=java.lang.System.getenv("CODEGEN_INSTALL_DIR");

cd /home/<useracct>/dvsdk_#_##_##_##/cs1omap3530_#_##_##

host $ make clean

host $ make
```

The rebuilt server executable can be found in the /home/<useracct>/dvsdk_#_##_##_##/cs1omap3530_#_##_##/packages/ti/sdo/server/bin directory and is named cs.x64P.

The DSP side server executables can also be built from the DVSDK installation directory itself by following the below commands.

Ensure that DSP side code generation tools are set properly in the Rules.make in DVSDK installation directory

```
CODEGEN_INSTALL_DIR=/home/<useracct>/dvsdk_#_##_##_##/cg6x_#_#_##
```

Re-building the Servers from DVSDK installation directory can be done using the following commands.

```
host $ cd /home/<useracct>/dvsdk_#_##_##_##

host $ make codecs_clean

host $ make codecs

host $ make install
```

# 7 EVM Operation

## 7.1 Using Telnet

The initialization file (/etc/init.d/rcS) starts a telnet server. If your development host is in the same sub-net as the EVM, you can telnet to it using the IP address assigned in the U-boot environment.

```
$ telnet 192.168.1.10
```

You will be prompted for a user name and password. The user name should be root and the password should be left blank (just hit Enter).

## 7.2 Booting the New Linux Kernel

After building the new kernel, in order to use it to boot the OMAP35x EVM, you must transfer it to the board via TFTP. Once you transfer the kernel image to the board via TFTP, you could either perform a TFTP kernel boot or flash the kernel image to the NAND and subsequently use the NAND boot. It is assumed you have completed the steps under Rebuilding the Linux Kernel, and the boot file, uImage has been copied to /tftpboot (or some other site-specific TFTP accessible location).

### 7.2.1 Checking the TFTP server setup in the Linux development PC

You can check to see if a TFTP server is set up with this command

```
# rpm -q tftp-server
```

If it is not set up, you can follow these steps:

1. To install TFTP, use this command, where -#.#-# is the version number portion of the filename:

```
# rpm -ivh /db/ztree/useracct/tftp-server-#.#-#.rpm
```

2. Confirm that TFTP is installed with this command:

```
# /sbin/chkconfig --list | grep tftp
```

You should see the following output:

```
tftp: off
```

If you need to turn on the TFTP server, use this command:

```
# /sbin/chkconfig tftp on
```

The default root location for servicing TFTP files is /tftpboot.

### 7.2.2 Using TFTP to boot the New Linux Kernel

1. Power on the EVM board, and abort the automatic boot sequence by pressing the “Enter key” in the console window.

2. Set the following environment variables to ensure that you are starting from a default, clean U-Boot environment.

   ```
   OMAP3EVM # setenv serverip <tftp server ip address>

   OMAP3EVM # setenv bootfile uImage
   ```

**The bootargs for DVSDK releases from 3.00.00.36 up to and including 3.00.01.42 are as given below**

```
OMAP3EVM # setenv bootargs mem=88M console=ttyS0,115200n8 noinitrd root=/dev/mtdblock4 rw rootfstype=jffs2 omapfb.rotate=1 omapfb.rotate_type=1 omap_vout.vid1_static_vrfb_alloc=y
```

**The bootargs for DVSDK releases from 3.00.02.44 onwards are as given below**

```
OMAP3EVM # setenv bootargs mem=99M console=ttyS0,115200n8 noinitrd root=/dev/mtdblock4 rw rootfstype=jffs2 mpurate=600 omapfb.rotate=1 omapfb.rotate_type=1 omap_vout.vid1_static_vrfb_alloc=y
```

```
OMAP3EVM # setenv bootcmd ‘tftpboot 0x80000000 uImage;bootm’
```

This configuration boots a new Linux kernel via TFTP with a NAND flash based file system

3. The initenv-samsung.txt and initenv-micron scripts provide the options for setting the bootargs for the appropriate target file system that is being used for the boot. Kindly refer to the appropriate script files for the flash parts that is being used on the board for more details. In case of use of ramdisk file system use the following commands:

**The bootargs for DVSDK releases from 3.00.00.36 up to and including 3.00.01.42 are as given below**:

```
OMAP3EVM # setenv bootargs mem=88M console=ttyS0,115200n8 root=/dev/ram0 initrd=0x81600000,16M ramdisk_size=40960 omapfb.rotate=1 omapfb.rotate_type=1 omap_vout.vid1_static_vrfb_alloc=y
```

**The bootargs for DVSDK releases from 3.00.02.44 onwards are as given below**

```
OMAP3EVM # setenv bootargs mem=99M console=ttyS0,115200n8 root=/dev/ram0 initrd=0x81600000,16M mpurate=600 ramdisk_size=40960 omapfb.rotate=1 omapfb.rotate_type=1 omap_vout.vid1_static_vrfb_alloc=y
```

You could use TFTP to transfer the ramdisk image from the development PC or flash it on the NAND flash. Refer section 5.4.2, Running the Re-flash Procedure for more details on flashing the file systems on the target

**NOTE:** Currently there is no full ramdisk image that includes OMAP3530 DVSDK demonstrations.

In case of use of NFS, use the following commands:

**The bootargs for DVSDK releases from 3.00.00.36 up to and including 3.00.01.42 are as given below**

```
OMAP3EVM # setenv bootargs  mem=88M console=ttyS0,115200n8 noinitrd ip=dhcp rw root=/dev/nfs nfsroot=<nfsserver_ip>:/home/<useracct>/workdir/filesys,nolock omapfb.rotate=1 omapfb.rotate_type=1 omap_vout.vid1_static_vrfb_alloc=y
```

**The bootargs for DVSDK releases from 3.00.02.44 onwards are as given below**

```
OMAP3EVM # setenv bootargs  mem=99M console=ttyS0,115200n8 noinitrd ip=dhcp rw root=/dev/nfs nfsroot=<nfsserver_ip>:/home/<useracct>/workdir/filesys,nolock mpurate=600 omapfb.rotate=1 omapfb.rotate_type=1 omap_vout.vid1_static_vrfb_alloc=y
```

### 7.2.3 Flashing the New Linux Kernel

The Linux kernel can also be flashed on the NAND and then could be used for the boot. Perform the following steps for flashing the new or rebuilt Linux Kernel

1. Make sure that the environment variables are properly set

```
OMAP3EVM # setenv serverip <tftp server ip address>

OMAP3EVM # setenv bootfile uImage
```

2. Flash the Kernel Image onto the Samsung NAND flash using the following commands.

```
OMAP3EVM # mw.b 0x81600000 0xff 0x1400000

OMAP3EVM # onenand erase block 20-59

OMAP3EVM # tftpboot 0x81600000 uImage

OMAP3EVM # onenand write 0x81600000 0x280000 0x0300000
```

3. Flash the Kernel Image onto the Micron NAND flash using the following commands.

```
OMAP3EVM # nand unlock

OMAP3EVM # mw.b 0x81600000 0xff 0x1400000

OMAP3EVM # nand erase 280000 300000

OMAP3EVM # tftpboot 0x81600000 uImage

OMAP3EVM # nandecc sw

OMAP3EVM # nand write.i 0x81600000 280000 300000
```

4. Once the image is flashed, refer to Booting From Flash using Board's NAND Flash File System or Booting from Flash Using NFS File System for booting using the kernel image in NAND flash.

## *7.3 Flashing the Pre-built/rebuilt NAND Flash File System*

Perform the following steps to program the flash for pre-built or a rebuilt NAND flash file system. The pre-built NAND flash file system (rootfs.jffs2) for OMAP3530 DVSDK is available under /home/<useracct>/dvsdk_#_##_##_##/targetfs folder. Copy it to the default location for servicing the tftp files on the host PC. (refer to section TFTP server setup on Linux PC or section Setup the TFTP Server)

1. Make sure that the environment variables are properly set

```
OMAP3EVM # setenv serverip <tftp server ip address>

OMAP3EVM # setenv bootfile uImage
```

2. Flash the NAND flash file system Image onto the Samsung NAND flash using the following commands.

```
OMAP3EVM # mw.b 0x81600000 0xff 0x6000000

OMAP3EVM # onenand erase block 60-1023

OMAP3EVM # tftpboot 0x81600000 rootfs.jffs2

OMAP3EVM # onenand write 0x81600000 0x780000 0x6000000
```

3. Flash the NAND flash file system Image onto the Micron NAND flash using the following commands.

```
OMAP3EVM # nand unlock

OMAP3EVM # mw.b 0x81600000 0xff 0x6000000

OMAP3EVM # nand erase 780000 6000000

OMAP3EVM # tftpboot 0x81600000 rootfs.jffs2
```

```
        OMAP3EVM # nandecc sw

        OMAP3EVM # nand write.i 0x81600000 780000 6000000
```

Once the image is flashed, refer to section Booting From Flash using Board's NAND Flash File System or section Booting via TFTP Using Board's NAND Flash File System for booting using the NAND flash file system

## 7.4 Changing Options in U-boot

If you use the environment provided by the initenv.txt script you have several choices for boot options. Most of the boilerplate has been provided; you just need to set your choices.

**Note:** Refer Section Running the Re-flash Procedure for modifications in the initenv.txt script files required to be able to run the dvsdk demos that comes OMAP3530 DVSDK. Also Refer Alternate Boot Methods

Kernel choices:

```
OMAP3EVM # setenv get_kernel run nand_kernel     (get kernel from flash)

OMAP3EVM # setenv get_kernel run tftp_kernel     (get custom kernel from TFTP)
```

Initrd choices:

```
OMAP3EVM # setenv get_initrd run nand_initrd     (get kernel from flash)

OMAP3EVM # setenv get_initrd run tftp_initrd     (get custom image from TFTP)
```

File-system choices:

```
OMAP3EVM # setenv bootcmd   run bootargs_rd      (use an initrd ramdisk)

OMAP3EVM # setenv bootcmd   run bootargs_fs              (use a JFFS file-system in flash)

OMAP3EVM # setenv bootcmd   run bootargs_nfs             (use an NFS root directory)
```

Save your choices:

```
OMAP3EVM # nand unlock                                           (only MICRON!)

OMAP3EVM # saveenv                                       (save to flash)
```

To run a JFFS file-system or a ramdisk image from flash you need to make sure the appropriate image is in flash. You can use the reflash.txt script to load the appropriate image.

Re-flash choices:

```
OMAP3EVM # run rf_all_rd                                 (update all using full ramdisk)

OMAP3EVM # run rf_all_min_rd                             (update all using minimal ramdisk)

OMAP3EVM # run rf_all_fs                                 (update all using JFFS image)

OMAP3EVM # run rf_rd                                     (update just the full ramdisk)

OMAP3EVM # run rf_min_rd                                 (update just the full ramdisk)
```

```
OMAP3EVM # run rf_fs                                    (update just the JFFS image)
```

**Note 1:** The two ramdisk images and JFFS are all mutually exclusive in Flash as they occupy the same partition. (Running ramdisk from tftp does not require flash space).

**Note 2:** The current U-boot only allows 40 environment variables. Do not saveenv after the reflash script has been submitted. Instead complete the reflash and reboot before making env changes.

**Note 3:** For reflashing choices for Micron nand please select the commands which are marked by _new.(For example 'rf_all_fs_new' for using JFFS filesystem and 'rf_all_rd_new' for using full ramdisk).

## 7.5 Alternate Boot Methods

### 7.5.1 Booting From Flash using Board's NAND Flash File System

This is the default, out-of-the-box boot configuration Make sure that the kernel image and the JFFS2 root file system is flashed To boot in this mode, set the following parameters after you abort the automatic boot sequence:

For Micron NAND EVM:

```
OMAP3EVM # setenv nand_kernel nand read.i 0x80000000 280000 300000
```

For Samsung NAND EVM:

```
OMAP3EVM # setenv nand_kernel onenand read 0x80000000 0x280000 0x0300000
```

Set the following environment variables, irrespective of Micron NAND or Samsung NAND EVM:

```
OMAP3EVM # setenv get_kernel run nand_kernel

OMAP3EVM # setenv bootcmd ‘run get_kernel;bootm 0x80000000’
```

**The bootargs for DVSDK releases from 3.00.00.36 up to and including 3.00.01.42 are as given below**

```
OMAP3EVM # setenv bootargs mem=88M console=ttyS0,115200n8 noinitrd root=/dev/mtdblock4 rw
rootfstype=jffs2 omapfb.rotate=1 omapfb.rotate_type=1 omap_vout.vid1_static_vrfb_alloc=y
```

**The bootargs for DVSDK releases from 3.00.02.44 onwards are as given below**

```
OMAP3EVM # setenv bootargs mem=99M console=ttyS0,115200n8 noinitrd root=/dev/mtdblock4 rw
rootfstype=jffs2 mpurate=600 omapfb.rotate=1 omapfb.rotate_type=1 omap_vout.vid1_static_vrfb_alloc=y
```

```
OMAP3EVM # boot
```

**NOTE:** All setenv commands should be entered on a single line. They are shown on multiple lines here for ease of reading.

## 7.5.2 Booting from Flash Using NFS File System

Make sure that the kernel image is flashed. To boot in this mode, set the following parameters after you abort the automatic boot sequence:

For Micron NAND EVM:

```
OMAP3EVM # setenv nand_kernel nand read.i 0x80000000 280000 300000
```

For Samsung NAND EVM:

```
OMAP3EVM # setenv nand_kernel onenand read 0x80000000 0x280000 0x0300000
```

Set the following environment variables, irrespective of Micron NAND sor Samsung NAND EVM:

```
OMAP3EVM # setenv get_kernel     run nand_kernel

OMAP3EVM # setenv bootcmd ‘run get_kernel;bootm 0x80000000’
```

**The bootargs for DVSDK releases from 3.00.00.36 up to and including 3.00.01.42 are as given below**

```
OMAP3EVM # setenv bootargs console=ttyS0,115200n8 noinitrd rw ip=dhcp root=/dev/nfs
nfsroot=<nfshost>:<rootpath>,nolock mem=88M omapfb.rotate=1 omapfb.rotate_type=1
omap_vout.vid1_static_vrfb_alloc=y
```

**The bootargs for DVSDK releases from 3.00.02.44 onwards are as given below**

```
OMAP3EVM # setenv bootargs console=ttyS0,115200n8 noinitrd rw ip=dhcp root=/dev/nfs
nfsroot=<nfshost>:<rootpath>,nolock mem=99M mpurate=600 omapfb.rotate=1 omapfb.rotate_type=1
omap_vout.vid1_static_vrfb_alloc=y
```

```
OMAP3EVM # boot
```

**NOTE:** All setenv commands should be entered on a single line. They are shown on multiple lines here for ease of reading.

The <nfshost> shall be set to the ip address of the NFS host machine. The <rootpath> must match the filesystem that you set up on your workstation. For example, /home/<useracct>/workdir/filesys

## 7.5.3 Booting via TFTP Using Board’s NAND Flash File System

To boot in this mode, set the following parameters after you abort the automatic boot sequence:

**NOTE:** You can find a pre-build kernel image (uImage) in the OMAP35x-PSP-SDK-##.##.##.##/images/kernel directory.

Make sure to check the tftp server setup on the Linux development host (Refer Checking the TFTP server setup in the Linux development PC)

```
OMAP3EVM # setenv serverip <tftp server ip address>

OMAP3EVM # setenv bootfile  <kernel image>

OMAP3EVM # setenv bootcmd ‘tftpboot 0x80000000 uImage;bootm 0x80000000’
```

**The bootargs for DVSDK releases from 3.00.00.36 up to and including 3.00.01.42 are as given below**

```
OMAP3EVM # setenv bootargs   mem=88M console=ttyS0,115200n8 noinitrd ip=dhcp root=/dev/mtdblock4 rw
rootfstype=jffs2 omapfb.rotate=1 omapfb.rotate_type=1 omap_vout.vid1_static_vrfb_alloc=y
```

**The bootargs for DVSDK releases from 3.00.02.44 onwards are as given below**

```
OMAP3EVM # setenv bootargs   mem=99M console=ttyS0,115200n8 noinitrd ip=dhcp root=/dev/mtdblock4 rw
rootfstype=jffs2 mpurate=600 omapfb.rotate=1 omapfb.rotate_type=1 omap_vout.vid1_static_vrfb_alloc=y
```

```
OMAP3EVM # boot
```

**NOTE:** All setenv commands should be entered on a single line. They are shown on multiple lines here for ease of reading.

### 7.5.4 Booting via TFTP using NFS file system

To boot in this mode, set the following parameters after you abort the automatic boot sequence:

**NOTE:** You can find a pre-build kernel image (uImage) in the OMAP35x-PSP-SDK-##.##.##.##/images/kernel directory.

```
OMAP3EVM # setenv bootcmd 'dhcp;bootm'

OMAP3EVM # setenv serverip <ip addr of tftp server>

OMAP3EVM # setenv bootfile <name of kernel image>

OMAP3EVM # setenv rootpath <root directory to mount>

OMAP3EVM # setenv nfshost <ip addr of nfs host>
```

**The bootargs for DVSDK releases from 3.00.00.36 up to and including 3.00.01.42 are as given below**

```
OMAP3EVM # setenv bootargs console=ttyS0,115200n8 noinitrd rw ip=dhcp root=/dev/nfs
nfsroot=$(nfshost):$(rootpath),nolock mem=88M omapfb.rotate=1 omapfb.rotate_type=1
omap_vout.vid1_static_vrfb_alloc=y
```

**The bootargs for DVSDK releases from 3.00.02.44 onwards are as given below**

```
OMAP3EVM # setenv bootargs console=ttyS0,115200n8 noinitrd rw ip=dhcp root=/dev/nfs
nfsroot=$(nfshost):$(rootpath),nolock mem=99M mpurate=600 omapfb.rotate=1 omapfb.rotate_type=1
omap_vout.vid1_static_vrfb_alloc=y
```

```
OMAP3EVM # boot
```

**NOTE:** All setenv commands should be entered on a single line. They are shown on multiple lines here for ease of reading.

The <root directory to mount> must match the file system that you set up on your workstation. For example, /home/<useracct>/workdir/filesys.

## *7.6 Working With 256MB Micron mDDR Memory*

In case the EVM that you have has ES3.1 silicon and 256MB Micron memory part, and you wish to utilize the entire 256MB, the bootargs need to be changed to create a memory hole.

**In case of DVSDK releases from 3.00.00.33 to 3.00.01.42, perform the following steps**

Create a memory hole from 88MB to 128MB for DSP side code and the remaining to be visible to the Linux kernel. To enable this, replace the **'mem=88M'** in the bootargs mentioned in the above sections with **'mem=88M@0x80000000 mem=128M@0x88000000'**

Make the following change to the loadmodules.sh file

Replace

```
insmod cmemk.ko phys_start=0x85800000 phys_end=0x86800000
pools=20x4096,8x131072,5x1048576,1x1429440,1x256000,1x3600000,5x829440
```

with

```
insmod cmemk.ko phys_start=0x85800000 phys_end=0x86800000
pools=20x4096,8x131072,5x1048576,1x1429440,1x256000,1x3600000,5x829440 allowOverlap=1
```

**In case of DVSDK releases from 3.00.02.44 onwards, perform the following steps**

Create a memory hole from 99MB to 128MB for DSP side code and the remaining to be visible to the Linux kernel. To enable this, replace the **'mem=99M'** in the bootargs mentioned in the above sections with **'mem=99M@0x80000000 mem=128M@0x88000000'**

Make the following change to the loadmodules.sh file

Replace

```
insmod cmemk.ko phys_start=0x86300000 phys_end=0x87300000
pools=20x4096,8x131072,5x1048576,1x1429440,1x256000,1x3600000,5x829440
```

with

```
insmod cmemk.ko phys_start=0x86300000 phys_end=0x87300000
pools=20x4096,8x131072,5x1048576,1x1429440,1x256000,1x3600000,5x829440 allowOverlap=1
```

# 8 Additional Procedures

## 8.1 Putting Demo Applications in the Third-Party Menu

You can add your own demos to the Third-Party Menu by following the steps in this section. Only four demos can be shown at once in the user-interface. If you add more than four demos, the first four in alphabetical order are shown.

1. Create the following files for your demo:
   - **logo.jpg.** This is the logo of the third party company which will be showed next to the demo description. The picture needs to be in JPEG format and of size 50x50.
   - **readme.txt.** This is a text file. The first 40 characters of the file should briefly describe the demo. The demo interface displays up to 40 characters, but stops if it encounters a new line character. For example, the file might contain "Video Phone demo" or "Network Audio demo".
   - **app.sh.** This is an executable that launches your demo. It can either be the demo executable itself or a shell script that executes the executable. (If this is a shell script, make sure its executable bit is set for all). A script could look something like:

     ```
     #!/bin/sh

     exec ./mydemoname
     ```

   - **other files.** If app.sh is a shell script, your demo executable will have some other name. You may also need to include data files or other files used by the executable.

**Note:** The demo application must use relative paths to access any files it needs at runtime. This because the archive is extracted to another location from which the demo is executed.

2. Create a gzipped tar file (ends with .tar.gz) that archives all the files in the previous list. For example, if your files are logo.jpg, readme.txt, and app.sh, you could use the following command:

   ```
   tar cvzf ti_videophone.tar.gz logo.jpg readme.txt app.sh
   ```

   Name the tar file using <company>_<demoname>.tar.gz (with no spaces in the file name) as the convention. For example, a video phone demo created by Texas Instruments would be named ti_videophone.tar.gz. The name must be unique since all demos are installed in the same directory.

   The three required files must be in the top-level directory of the archive. Other files may be in subdirectories, so long as the demo uses relative references to access them. For example, the following directory structure might be used in the archive:

   ```
   |-- app.sh

   |-- data

   |   |-- datafile1

   |   `-- datafile2

   |-- logo.jpg

   |`-- readme.txt
   ```

   To check the format of the file you create, execute the following command in Linux. The result should say "gzip compressed data".

   ```
   file <filename>.tar.gz
   ```

3. Put your archive in the "thirdpartydemos" subdirectory of the target installation directory. This is where the DVEVM software was installed on the target file system. The default target installation directory is /opt/dvsdk/omap3530, so the default location for demo archives is

/opt/dvsdk/omap3530/thirdpartydemos. Do not extract the contents of the archive in this location. Extraction is performed behind-the-scenes each time the demo is run.

## *8.2 Changing the Output Display to DVI*

The output display is set by default as LCD.You can change it to DVI either through Boot Mode Argument for the kernel or command line arguments once the kernel has booted.

### Boot Mode Arguments for DVI

Please add the following commands to the bootargs 'omap-dss.def_disp="dvi" omapfb.video_mode="720x480MR-16@60"'

Please note that some versions of the EVM may require a hardware modification for DVI output to work. You can find information on the hardware modification required in the EVM Modifications for DVI-D output section.

The bootargs mentioned in section Running the Re-flash Procedure can be modified as below for running the pre-built Nand filesystem or NFS filesystem which is part of the download page.

#### The bootargs for DVSDK releases from 3.00.00.36 to 3.00.01.42 are as given below

```
setenv bootargs_rd mem=88M console=ttyS0,115200n8 root=/dev/ram0 initrd=0x81600000,16M ramdisk_size=40960 omapfb.rotate=1 omapfb.rotate_type=1 omap_vout.vid1_static_vrfb_alloc=y omap-dss.def_disp="dvi" omapfb.video_mode="720x480MR-16@60"
```

```
setenv bootargs_fs mem=88M console=ttyS0,115200n8 noinitrd root=/dev/mtdblock4 rw rootfstype=jffs2 omapfb.rotate=1 omapfb.rotate_type=1 omap_vout.vid1_static_vrfb_alloc=y omap-dss.def_disp="dvi" omapfb.video_mode="720x480MR-16@60"
```

```
setenv bootargs_nfs mem=88M console=ttyS0,115200n8 noinitrd ip=dhcp rw root=/dev/nfs nfsroot=<nfsserver_ip>:/home/<useracct>/workdir/filesys,nolock omapfb.rotate=1 omapfb.rotate_type=1 omap_vout.vid1_static_vrfb_alloc=y omap-dss.def_disp="dvi" omapfb.video_mode="720x480MR-16@60"
```

#### The bootargs for DVSDK releases from 3.00.02.44 onwards are as given below

```
setenv bootargs_rd mem=99M console=ttyS0,115200n8 root=/dev/ram0 initrd=0x81600000,16M ramdisk_size=40960 mpurate=600 omapfb.rotate=1 omapfb.rotate_type=1 omap_vout.vid1_static_vrfb_alloc=y omap-dss.def_disp="dvi" omapfb.video_mode="720x480MR-16@60"
```

```
setenv bootargs_fs mem=99M console=ttyS0,115200n8 noinitrd root=/dev/mtdblock4 rw rootfstype=jffs2 mpurate=600 omapfb.rotate=1 omapfb.rotate_type=1 omap_vout.vid1_static_vrfb_alloc=y omap-dss.def_disp="dvi" omapfb.video_mode="720x480MR-16@60"
```

```
setenv bootargs_nfs mem=99M console=ttyS0,115200n8 noinitrd ip=dhcp rw root=/dev/nfs nfsroot=<nfsserver_ip>:/home/<useracct>/workdir/filesys,nolock mpurate=600 omapfb.rotate=1 omapfb.rotate_type=1 omap_vout.vid1_static_vrfb_alloc=y omap-dss.def_disp="dvi" omapfb.video_mode="720x480MR-16@60"
```

#### Command Line Arguments

Once the EVM is booted, you can change the output display from LCD to DVI from the Linux shell in a terminal window connected to the EVM board’s serial port.

Issue following commands to switch output device from LCD to DVI

```
target $ echo "gfx e:0" > /sys/devices/platform/omapfb/overlays

target $ echo "lcd e:0" > /sys/devices/platform/omapfb/displays

target $ echo "lcd t:none" > /sys/devices/platform/omapfb/managers

target $ echo "dvi e:1" > /sys/devices/platform/omapfb/displays

target $ echo "lcd t:dvi" > /sys/devices/platform/omapfb/managers
```

/* Set the Frame buffer parameters like xres, xres_virtual, y_res, yres_virtualagain before running below command. For example for 720P as DVI resolution setxres=1280 yres=720 xres_virtual=1280 yres_virtual=1440.*/

```
target $ echo "gfx e:1" > /sys/devices/platform/omapfb/overlays
```

## 8.3 Enabling Audio Capture for Speech encode

Audio Capture for speech encode is not enabled by default.You can use amixer utilty which is present by default enable the Audio Capture.

Once the EVM is booted issue following commands to enable Audio Capture.

```
target $amixer controls

target $amixer cset name='Analog Left Capture Route AUXL' 1

target $amixer cset name='Analog Right Capture Route AUXR' 1
```