Linux PSP User's Guide for AM1x/OMAP-L1/DA8xx

Community Linux PSP for DA8x/OMAP-L1/AM1x

Document License

PSP Overview

Linux Platform Support Package (PSP) provides support for Linux kernel, U-Boot, UBL and utilities to flash boot software on the EVM. The latest PSP package can be obtained from TI's Technology and Software Publicly Available (TSPA) download site here [1]. This package includes the following components:

Note: Specific version information of each component is included in the PSP Release Notes for the release. PSP drivers features and performance guide provides an overview, features, constraints and performance for each of the drivers included in the PSP release.

- **DaVinci Linux kernel.** This Linux kernel is based on a kernel version available from the DaVinci GIT tree [2]. The pre-built kernel binary included with the PSP package is built with Sourcery G++ Lite 2009q1-203 for ARM GNU/Linux [3] from CodeSourcery [4].
- **U-Boot.** This U-Boot is based on an U-Boot version available from the U-Boot GIT tree [5]. The pre-built U-Boot binary included with this release is built with Sourcery G++ Lite 2009q1-203 for ARM GNU/Linux [3] from CodeSourcery [4]
- **User Boot Loader (UBL).** This is the primary boot software which copies U-Boot to external RAM and starts it. This software requires Code Composer Studio (CCStudio) version 3.3 or 4.1 (CCStudiov4) for building it.
- **Flash writers.** This software requires Code Composer Studio (CCStudio) version 3.3 or 4.1 (CCStudiov4) for building.

You can copy any portions of the PSP that need to be run on a Microsoft Windows host (such as CCStudio projects for building the UBL and flash writers) to a Microsoft Windows host. Alternately, you can install the PSP package in a disk partition that can be accessed both from a Microsoft Windows and Linux host.

Most of the components of the PSP need to be untar-ed or extracted in order to be used. The following files and directories are provided in the PSP:

```
----DaVinci-PSP-SDK-#.#.#.#
|-- Software-manifest.html
|-- docs
|   |-- FeaturesPerformanceGuide-#.#.#.#.pdf
|   |-- GPLv2.pdf
|   |-- ReleaseNotes-#.#.#.#.pdf
|   `-- UserGuide-#.#.#.#.pdf
|-- host-tools
|   |-- linux
```

```
|   |-- src
|   `-- windows
|-- images
|   |-- boot-strap
|   |   |-- am17xx
|   |   |   |-- arm-nand-ais.bin
|   |   |   |-- arm-spi-ais.bin
|   |   |-- omapl1x7
|   |   |   |-- ubl-nand.bin
|   |   |   |-- ubl-spi.bin
|   |   |   |-- dsp-nand-ais.bin
|   |   |   `-- dsp-spi-ais.bin
|   |   |-- omapl1x8
|   |   |   |-- arm-mmcsd-ais.bin
|   |   |   |-- arm-nand-ais.bin
|   |   |   |-- arm-nor-ais.bin
|   |   |   `-- arm-spi-ais.bin
|   |-- examples
|   |   |-- edma_test.ko
|   |   |-- gpio_test.ko
|   |   |-- vpif_test_display
|   |   |-- vpif_test_mmap_loopback
|   |   `-- vpif_test_userptr_loopback
|   |-- kernel
|   |   |-- omapl1x7
|   |   |   |-- modules\
|   |   |   `-- uImage
|   |   |-- omapl1x8
|   |   |   |-- modules\
|   |   |   `-- uImage
|   |-- u-boot
|   |   |-- omapl1x7
|   |   |   `-- u-boot.bin
|   |   |-- omapl1x8
|   |   |   `-- u-boot.bin
|   `-- utils
|   |   |-- omapl1x7
|   |   |   |-- nand_writer.out
|   |   |   `-- spiflash_writer.out
|   |   |-- omapl1x8
|   |   |   |-- nand-writer.out
|   |   |   |-- norflash-writer.out
|   |   |   |-- spiflash-writer.out
|   |   |   `-- uflash
|-- scripts
|-- src
|   |-- boot-strap
```

```
|   |   |-- armubl-#.#.#.#.tar.gz
|   |   `-- dspubl-#.#.#.#.tar.gz
|   |-- examples
|   |   `-- examples.tar.gz
|   |-- kernel
|   |   |-- ChangeLog-#.#.#.#
|   |   |-- ShortLog
|   |   |-- Unified-patch-#.#.#.#.gz
|   |   |-- diffstat-#.#.#.#
|   |   |-- kernel-patches-#.#.#.#.tar.gz
|   |   `-- linux-#.#.#.#.tar.gz
|   |-- u-boot
|   |   |-- ChangeLog-#.#.#.#
|   |   |-- ShortLog
|   |   |-- Unified-patch-#.#.#.#.gz
|   |   |-- diffstat-#.#.#.#
|   |   |-- u-boot-#.#.#.#.tar.gz
|   |   `-- u-boot-patches-#.#.#.#.tar.gz
|   `-- utils
|       |-- mmcsd-writer-#.#.#.#.tar.gz
|       |-- nand-writer-#.#.#.#.tar.gz
|       |-- norflash-writer-#.#.#.#.tar.gz
|       `-- spiflash-writer-#.#.#.#.tar.gz
`-- test-suite
    `-- REL_LFTB_#.#.#.#.tar.gz
```

Host platform Requirements

Building and running all of the PSP components requires both a Windows and a Linux machine.

The Windows machine is required for running CCStudio 3.3 or 4.1. CCStudio is required for building the User Boot Loader (UBL) and Flash writers. CCStudio is also used to burn the boot images (UBL, U-Boot) into the flash using the flash writers provided in the PSP package.

Linux host is required:

- for compiling U-Boot and Linux kernel.
- to host the TFTP server required for downloading kernel and file system images from U-Boot using Ethernet.
- to host the NFS server to boot the EVM with NFS as root filesystem

Host Software Requirements

- CCstudio 3.3.38 or 4.1.0.02003 (needed only to rebuild UBL/flash writers or to re-flash UBL/U-Boot image on the EVM)
- TI Code Generation Tools 4.5.1 and above for TMS470Rx (needed only to rebuild UBL/flash writers)
- CodeSourcery tool chain for ARM
- Serial console terminal application
- TFTP and NFS servers.

Getting Started Quickly

Get started with setting up the EVMs for **OMAP-L138, DA850 or AM18x** or **OMAP-L137, DA830 or AM17x**.

To help you get started quickly, pre-built binaries for the UBL, U-Boot, Linux kernel, and flash writers are provided in the `images` directory under PSP installation.

In order to create your own applications running on Linux or to rebuild U-Boot or the Linux kernel provided with the PSP package, you will need to install the CodeSourcery tools for cross compilation.

Running PSP Components

Booting U-Boot provides information on setting up the EVM to boot U-Boot from various boot media.
Booting the Linux kernel provides information on booting Linux on the EVM.
Re-flashing boot images provides information on re-flashing the boot software (UBL, U-Boot) on the EVM.

Using Linux Kernel Drivers

→ **Linux drivers usage** has specific usage information on various Linux drivers and features.

Loading Linux kernel modules provides information on how to use various kernel features and drivers as loadable kernel modules.

Building PSP Components

Building Software Components for OMAP-L1 provides procedures for rebuilding the following software components used on the OMAP-L1 processors or to flash software to the board.

- **Linux kernel**
- **U-Boot**
- **DSP User Boot Loader** (For **OMAP-L137 (or DA830)** only)
- **ARM User Boot Loader**
- **SPI Flash writer**
- **NAND Flash writer**
- **NOR Flash writer** (For **OMAP-L138 (or DA850, AM18xx)** only)

Configuring Linux Kernel provides information on how to reconfigure the Linux kernel to include and exclude various drivers and kernel features.

Additional topics

The **additional procedures** topic documents some additional useful procedures aside from the usual usage procedures.

References

[1] http://software-dl.ti.com/dsps/dsps_public_sw/sdo_sb/targetcontent/psp/DaVinci-PSP-SDK/03_20/index_FDS.html
[2] http://git.kernel.org/?p=linux/kernel/git/khilman/linux-davinci.git;a=summary
[3] http://www.codesourcery.com/sgpp/lite/arm/portal/release858
[4] http://www.codesourcery.com
[5] http://git.denx.de/?p=u-boot.git;a=summary

Getting Started Guide for OMAP-L1

This Getting Started Guide (GSG) walks you through setting up the OMAP-L137 and OMAP-L138 EVM and installing the software. You should proceed through this guide in the order given for best results. By the end of this Getting Started Guide you will have the EVM booting to Linux and the Linux host development environment configured. You should also bookmark the OMAPL1 category since new articles will continue to appear on this wiki.

EVM overview

EVM Overview - OMAP-L138 provides an overview of the OMAP-L138 EVM kits and boards.

EVM Overview - OMAP-L137 provides an overview of the OMAP-L137 EVM kits and boards.

Hardware setup

For information on setting up the OMAP-L138 EVM hardware, please follow the steps in the *ZOOM OMAP-L138 eXperimenter Kit QuickStart Guide*, which is available for downloading at http://support.logicpd.com/downloads/1213/.

Spectrum Digital, Inc has a Technical Reference Guide [1] for OMAP-L137 EVM.

Booting the EVM out of the box

This section describes how to boot the EVM board when you first set it up.

- For **OMAP-L138 (or DA850)** details are provided **here**
- For **OMAP-L137 (or DA830)** details are provided **here**

Installing the SDK software

→ **Installing the Software for OMAP-L1** covers how to install the SDK software and CodeSourcery Sourcery G++ tools for the OMAP-L1 processors.

As host platforms for the development software, you need both a Windows PC and a Linux machine to build and run all of the components.

- The Windows machine is used to run CCStudio 3.3/CCStudio 4, which you will use if you want to build the User Boot Loader (UBL) and Flash writers. CCStudio is also used to burn the boot images (UBL, U-Boot) into the flash using the flash writers provided.
- The Linux host is used for the following:
  - Recompiling U-Boot and the Linux kernel.
  - Hosting the TFTP server required for downloading kernel and file system images from U-Boot using Ethernet.
  - Hosting the NFS server to boot the EVM with NFS as root filesystem
  - Running a serial console terminal application

The Installing the Software for OMAP-L137 page covers how to install the MontaVista tools, SDK software, and Linux Support Pack for the OMAP-L137 processor.

Note: If you are migrating from the early adopter version of the SDK software you can check the differences in this document.

Setting up target file system

Setting up OMAP-L1 Target File System explains how to set up an NFS target file system for use on the OMAP-L1 EVMs.

For information on setting up an NFS target file system for use on the OMAP-L137 EVM please see the Setting up OMAP-L137 Target File System page.

For information on creating other types of target file systems such as SD/MMC and USB please see the Creating file systems on removable media page.

Building software components

Building PSP components provides procedures for rebuilding the platform software components used on the processors or to flash software on to the EVM.

Building the SDK

Building the OMAP-L1 SDK provides information on building the SDK software for both OMAP-L137 and OMAP-L138.

Note: In general, the instructions for building the SDK for OMAP-L138 are the same as for OMAP-L137. Follow the steps on this page that are specific to your device.

The SDK software provided with this EVM contains examples for communicating between the ARM and DSP processors as well as driver modules used in that communication. It also contains development packages, such as Codec Engine, which allow for easy communication between the ARM and DSP.

Running PSP Components

Running PSP Components describes the steps to boot Linux and to how to re-flash UBL and U-boot images if needed.

Additional procedures

Linux Functional Test Bench introduces a set of tools used to verify the various driver features.
Loading Linux kernel modules describes the procedure to load kernel modules into a running kernel.
Message logging on UART in UBL describes the procedure to enable message logging in ARM UBL.
Modifying SPI Frequency in U-Boot describes how to modify the SPI frequency in U-Boot.
Restoring factory default U-Boot environment variables describes how to revert to factory default U-Boot environment variables.
Using extended memory available on DA850/OMAP-L138/AM18x EVM describes how Linux can be configured to utilize the additional RAM available on the EVM board when compared to the eXperimenter board.
Creating bootable SD card for DA850/OMAP-L138/AM18x EVM describes the steps to write U-Boot to SD card using uflash utility.

For OMAP-L138 and DA850 SoCs, **DSP wakeup in U-Boot** explains how to prevent the DSP from being woken up by U-Boot.

For OMAP-L137 and DA830 SoCs, **enabling Write-Back cache** explains how to enable write-back cache support in Linux kernel when using silicon revision 2.0 and higher.

The following pages provide additional procedures that apply to a variety of platforms, including the OMAP-L1:

- **Setting up a TFTP Server** explains how to set up a TFTP server.
- **TeraTerm Scripts** provides example TeraTerm scripts that can be used with your board.

Additional information

- Additional hardware information can be found at the Logic PD OMAPL138 webpage: http://www.logicpd.com/products/development-kits/zoom-omap-l138-evm-development-kit
- References about some SDK components (Codec Engine, Framework Components): http://focus.ti.com/docs/toolsw/folders/print/tmdmfp.html
- The standalone version of the PSP device drivers is located here: http://focus.ti.com/docs/toolsw/folders/print/supportpkg.html
- Also, look around in this DaVinci wiki site for additional information

References

[1] http://support.spectrumdigital.com/boards/evmomapl137/revd/files/EVMOMAPL137_TechRef_RevD.pdf

GSG: Installing the Software for OMAP-L1

^ Up to main **Getting Started Guide for OMAP-L138** Table of Contents

Prerequisites

To begin installing the SDK software, make sure you have met the following system requirements:

- Host machine running a version of Linux such as RedHat Enterprise Linux WS or Ubuntu.
- OMAP-L138 EVM (this Getting Started Guide assumes that you have the Development Kit, not the eXperimenter Kit)

Software overview

To begin developing applications, you need to install the EVM development environment. This section provides an overview of the EVM software.

Software components

The TI SDK software includes the following installers:

- **OMAP-L138_setuplinux_#_#_#_#.bin** This is the ARM-side SDK software for Linux workstations. It installs development software, including:
  - Linux Platform Support Package (PSP) which includes board-flashing utilities running on ARM and some Linux driver examples
  - DSP/BIOS Utilities
  - Codec Engine
  - Board-Flashing Utilities running on DSP
  - DSP Link
  - EDMA Low Level Driver (LLD) running on DSP
  - Framework Components including DMAN3
  - Linux Utilities including CMEM
  - xDAIS Algorithm Package
- **bios_setuplinux_#_#_#.bin** This is the DSP/BIOS installer for Linux.
- **xdctools_setuplinux_#_#_#_#.bin** This is the XDCtools installer for Linux. It installs XDCtools, which provides support for RTSC and is needed to build various software, including Codec Engine, Framework Components, DSP Link.

- **ti_cgt_c6000_#.#.#_setup_linux_x86.bin** This is the TI Code Generation Tools installer for Linux. It is needed for users to who do not already have the latest code generation tools.

Tool chain

To build the Linux kernel, you will need to download and install version 2009q1-203 of the Sourcery G++ tools from Code Sourcery, which is available here [3].

These command line tools are available at no cost. Code Sourcery also offers more fully featured tools for a cost. The Sourcery G++ Lite package includes all of the following components:

- CodeSourcery Debug Sprite for ARM
- GNU Binary Utilities (Binutils)
- GNU C Compiler (GCC)
- GNU C Library (GLIBC)
- GNU C++ Compiler (G++)
- GNU C++ Runtime Library (Libstdc++)
- GNU Debug Server (GDBServer)
- GNU Debugger (GDB)

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.

- **U-Boot>**

  Indicates commands to be typed into the U-Boot shell in a console window connected to the EVM board's serial port.

- **target$**

  Indicates commands to be typed into the Linux shell in the terminal window connected to the EVM board's serial port.

Installation steps

Installing the software used by the EVM kit involves performing the following steps:

- Downloading TI software installers
- Installing the SDK Software
- Downloading and installing the Code Sourcery tools

Downloading TI software installers

Even though the EVM Development Kit includes TI software on DVD, it is recommended that you begin by registering the kit with TI and going to the TI software download page to get the latest versions of the software used with the EVM. (The eXperimenter kit does not include software, so you will need to download it in any case.)

On a host system, follow these steps:

1. Follow the steps in the kit's Read Me 1st Card to register and download the most current software from TI's website. (If you have already registered this EVM, skip to the next step.)
2. The software is currently available here [1].

3. Copy the following files to a temporary location with at least 1 GB of available space. Since you can delete the installation files after installing the software, a directory like **/tmp** is recommended.
   - **OMAP-L138_setuplinux_#_#_#_#.bin**
   - **bios_setuplinux_#_#_#.bin**
   - **xdctools_setuplinux_#_#_#_#.bin**
   - **ti_cgt_c6000_#.#.#_setup_linux_x86.bin**
   - **cs1omapl138_1_00_00-a_setup_linux.bin**

Note: For this release, TI recommends choosing either the ARM Linux drivers or the DSP BIOS Drivers for execution but not both at the same time.

Installing the SDK Software

The SDK software includes Codec Engine components, DSP/BIOS Link, xDAIS and xDM header files, Framework Components, etc.

NOTE: Some installers have different default installation locations. 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 later component build setup steps.

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

1. Log in using a user account on your Linux workstation. The user account must have execute permission for the all the installation files. Switch user to “root” on the host Linux workstation and change directories to the temporary location where you have downloaded the bin files.

   host$ su root

   host$ cd /tmp

   host$ chmod +x *.bin

   host$ exit
2. Execute the ARM-side SDK installer that you previously downloaded in a temporary directory.

   For example:

   host$ cd /tmp

   host$ ./OMAP-L138_setuplinux_#_#_#_#.bin

   NOTE: If you do not have a graphical display, you may use console installation with the following command.

   host$ ./OMAP-L138_setuplinux_#_#_#_#.bin --mode console

   Path Information - This installs the SDK in /home/<useracct>/OMAPL138_arm_#_#_#_#. For more about the contents of the PSP in this installation, see About the PSP package.
3. Execute the XDCtools installer. For example:

   host$ cd /tmp

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

   NOTE: If you do not have a graphical display, you may use console installation with the following command.

   host$ ./xdctools_setuplinux_#_#_#.bin --mode console

   Path Information - When you are prompted, **do not use the default installation location**. Instead, install the software in the directory created in Step 2. For example, /home/<useracct>/OMAPL138_arm_#_#_#_#.
4. Execute the DSP/BIOS installer. For example:

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

   NOTE: If you do not have a graphical display, you may use console installation with the following command.

host$ ./bios_setuplinux_#_#_#.bin --mode console

Path Information - When you are prompted, **do not use the default installation location**. Instead, install the software in the directory created in Step 2. For example, /home/<useracct>/OMAPL138_arm_#_#_#_#.

NOTE: For DSP/BIOS 5.33.06, please also execute the following command:

host$ su root

host$ chmod 555 /home/<useracct>/OMAPL138_arm_#_#_#_#/bios_5_33_06/xdctools/*

host$ exit

5. Execute the Code Generation Tools installer. For example:

   host$ ./ti-cgt-C6000_#.#.#_setup_linux_x86.bin

   NOTE: If you do not have a graphical display, you may use console installation with the following command.

   host$ ./ti-cgt-C6000_#.#.#_setup_linux_x86.bin --mode console

   Path Information - When the installer prompts for an installation location, **do not use the default location**. Instead, use the entire path to the OMAPL138_arm_#_#_#_# codegen directory. **You will need to manually create the folder "cg6x_#_#_#" (or equivalent folder name for the CGtool version)**.

   For example:

   /home/<useracct>/OMAPL138_arm_#_#_#_#/cg6x_6_1_9

   Remember to set the environment variable as directed by the installer. For example:

   export C6X_C_DIR="/home/<useracct>/OMAPL138_arm_#_#_#_#/cg6x_#_#_#/include;
   /home/<useracct>/OMAPL138_arm_#_#_#_#/cg6x_#_#_#/lib"

   NOTE: The above line should be entered on one line with no carraige return.

   NOTE: To set this environment variables for all terminal sessions and between reboots you should add the line above to the ~/.bashrc file. You can then *source* the ~/.bashrc file using the following command to pick up this setting.

   host $ source ~/.bashrc

6. Execute the codec server installer. For example:

   host $ ./cs1omapl138_1_##_##-a_setup_linux.bin.bin

   NOTE: This installer requires your linux host to have a graphical display.

   Path Information - When the installer prompts for an installation location, you can install it in the default installation directory or specify your own directory, i.e. /home/<useracct>/OMAPL138_arm_#_#_#_#/cs1omapl138_#_#.

7. If you have not already done so, follow the installation instructions provided with the Logic CD that is included with the Development Kit. In addition to software from Logic, this CD provides TI's Code Composer Studio v3.3, which you should install on a Microsoft Windows host.

Downloading and installing the Code Sourcery tools

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

- Go the CodeSourcery web site and download the Sourcery G++ Lite 2009q1-203 for ARM [3] tools:
- Download the IA32 GNU/Linux Installer package to your Linux workstation.
- Make the downloaded file executable

```
host$ chmod +x arm-2009q1-203-arm-none-linux-gnueabi.bin
```

- Run the installer and follow the steps it presents to install the package.

```
host$ ./arm-2009q1-203-arm-none-linux-gnueabi.bin
```

- **Note:**
- On ubuntu it might fail as it uses 'dash' as a shell (the Debian shell). To reconfigure for the bash shell, use the following command:

```
host$ sudo dpkg-reconfigure -plow dash
```

- If you are using 64-bit Linux host, ensure that ia32-libs are installed

```
host$ sudo apt-get install ia32-libs
```

- If you installed in the default path, you can find the files at
- `/home/<user>/CodeSourcery`
- . Ensure the Code Sourcery tools are in the path by adding the following

```
host$ export
PATH=/home/<user>/CodeSourcery/Sourcery_G++_Lite/bin:$PATH
```

- to the ~/.bashrc file. You can then
- *source*
- the ~/.bashrc file with the following command or restart the bash shell

```
host$ source ~/.bashrc
```

About the PSP package

The **Linux Platform Support Package (PSP)** provides support for Linux kernel, U-Boot, UBL and utilities to flash boot software on the EVM.

The PSP package is available as part of the SDK OMAP_L138_arm_#_#_#_#.tar.gz installation in the DaVinci-PSP-SDK-#.#.#.# directory under the main installation directory.

What's next?

Now that you have installed the SDK software, please continue on to the **OMAP-L1 Setting up target file system** or the **AM18x Setting up target file system** section of the Getting Started Guide.

References

[1] http://www.ti.com/omapl138c6748sw

GSG: Setting up OMAP-L1/AM1x Target File System

^ Up to main **Getting Started Guide for OMAP-L138** Table of Contents

^ Up to main **Getting Started Guide for AM18x** Table of Contents

^ Up to main **Getting Started Guide for AM17x** Table of Contents

The AM17x/AM18x/OMAP-L138 EVM comes with a target file system pre-loaded on the SD card (see EVM installation guide). However, in order to develop Linux applications it may be necessary to set up a target file system that is NFS mounted to your development workstation. NFS is ideal during evaluation and development because it allows you to modify the contents of the target file system online without needed to reboot the EVM. Once you have tested your application, you can store it to a more permanent storage media such as SD/MMC or USB storage for standalone operation.

Prerequisites

- A Linux host system to serve the NFS file system
- A network connection between the Linux host and the EVM
- The Arago file system that is available at Arago Repository [1]. (Dead Link)

Exporting a Shared File System for Target Access

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. These instructions are based on the RedHat Enterprise Linux v4 distribution. For Ubuntu, please check this page [2]. You may need to modify the steps if you are using another Linux distribution.

To export the file system from your NFS server, perform the following steps. You only need to perform these steps once. Following <target_name> needs to be replaced in the below mentioned path corresponding to specific target:

Target	<target_name>
OMAP-L137	OMAPL137_arm_#_#_#_#
OMAP-L138	OMAPL138_arm_#_#_#_#
AM18x	AM1x_arm_#_#_#_#
AM17x	AM1x_arm_#_#_#_#

1. Log in with a user account on the host Linux workstation
2. Perform the following commands to prepare a location for the Arago target file system. For example:

   host $ cd /home/<useracct>

   host $ mkdir -p workdir/filesys

   host $ cd workdir/filesys
3. Switch user to **root** on the host Linux workstation.

   host $ su root

4. Perform the following commands to create a copy of the target file system with permissions set for writing to the shared area as <useracct>. Substitute you user name for <useracct>.

   For PSP versions earlier than 3.20.00.11, the file system is found in the *images/fs* directory. Use the following commands:

   host $ cp -a /home/<useracct>/<target_name>/DaVinci-PSP-SDK-03.#.#.#/images/fs/nfs.tar.gz .

   host $ tar xvfz nfs.tar.gz

   host $ chmod 777 -R /home/<useracct>/workdir/filesys

   For PSP versions 3.20.00.11 or later, the file system needs to be downloaded sepeartely from the product download page [3] into a temporary directory, such as **/tmp**. After downloading, use the following commands:

   host $ cp -a /tmp/arago-demo-image-*.tgz .

   host $ tar xvfz arago-demo-image-*.tgz

   host $ chmod 777 -R /home/<useracct>/workdir/filesys

5. Edit the /etc/exports file on the host Linux workstation (not the exports file on the target file system). Add the following line for exporting the filesys directory, substituting your user name for <useracct>. Use the full path from root; ~ may not work for exports on all file systems.

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

   NOTE: Make sure you do not add a space between the ***** and the **(** in the above command.

6. Still as root, use the following commands to make the NFS server aware of the change to its configuration and to invoke an NFS restart.

   host $ /usr/sbin/exportfs -av

   host $ /sbin/service nfs restart

   NOTE: Use **exportfs -rav** to re-export all directories. Use **/etc/init.d/nfs status** to verify that the NFS status is running.

For Ubuntu NFS setting, please check this page [2]

1. 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

   host $ exit

For Ubuntu iptables settings, please check this page [4]

Testing the Shared File System

To test your NFS setup, follow these steps:

1. Copy the kernel Image (uImage) from the /home/<useracct>/<target_name>/DaVinci-PSP-SDK-03.#.#.#/images/kernel/<Platform> directory of the SDK software installation to the /tftpboot directory of your Linux host workstation. This kernel will be used to test the NFS target file system. If you do not have a TFTP server configured please see the Setting up a TFTP Server page.

   host $ cp /home/<useracct>/<target_name>/DaVinci-PSP-SDK-03.#.#.#/images/kernel/<Platform>/uImage /tftpboot

2. Get the IP address of your host Linux workstation as follows. Look for the IP address associated with the eth0 ethernet port.

host $ /sbin/ifconfig

3. Open a terminal session to connect to the EVM board via RS-232 using the instructions in the Connecting to a Console Window section. If you have a Windows workstation, you can use TeraTerm. If you have a Linux workstation, you might use Minicom. (You may need to turn on line wrap)
4. Power on the EVM board and abort the automatic boot sequence by pressing a key in the console window.
5. Set the following environment variables in the console window:

   EVM # setenv nfshost <Linux Host IP address>

   EVM # setenv rootpath <directory to mount>

   NOTE: The <directory to mount> must match what you specified in step 4 of the Exporting a Shared File System for Target Access section. For example, /home/<useracct>/workdir/filesys.

   EVM # setenv serverip <Linux Host IP address>

   EVM # setenv bootfile uImage

   EVM # setenv bootcmd 'dhcp;tftp;bootm'

   EVM # setenv bootargs console=ttyS2,115200n8 noinitrd rw ip=dhcp root=/dev/nfs nfsroot=${nfshost}:${rootpath},nolock mem=32M

   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.

   NOTE: The use of parenthesis for variable substitution is being deprecated for new U-boot releases - use curly braces {} instead. Check http://www.denx.de/wiki/view/DULG/CommandLineParsing

   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.
6. Save the environment so that you do not have to retype these commands every time you cycle power on the EVM board:

   EVM # saveenv
7. Boot the board using NFS:

   EVM # boot

   NOTE: The first boot of this file system may take a while to complete while security keys are being generated. Subsequent boots will not have this delay.
8. You can now log in as "root" in the target with no password required.

Mounting Root File System

This section will cover how to set the U-Boot environment variables to mount the root file system from various storage media.

From NFS

In order to boot Linux with the file system mounted from NFS you will need to have first exported an NFS target file system. For more information on how to do this please see the page **here**. This section also assumes that you are using DHCP to obtain the ip address for the board. If you are using static IP you will need to modify the bootargs to set the *ipaddr* variable appropriately.

EVM # setenv nfshost <ip address on nfs host>

EVM # setenv rootpath <directory to mount>

EVM # setenv bootargs console=ttyS2,115200n8 noinitrd rw ip=dhcp root=/dev/nfs nfsroot=${nfshost}:${rootpath},nolock mem=32M

From USB Storage

In order to boot Linux with the root file system mounted from a USB storage device you will need to set the following *bootargs*. For more information on configuring a USB storage device with a root file system please see Creating file systems on removable media

EVM # setenv bootargs console=ttyS2,115200n8 noinitrd rw ip=dhcp root=/dev/sda2 rootfstype=ext2 mem=32M

From SD/MMC

In order to boot Linux with the root file system mounted from a USB storage device you will need to set the following *bootargs*. For more information on configuring a USB storage device with a root file system please see Creating file systems on removable media

EVM # setenv bootargs console=ttyS2,115200n8 noinitrd rw ip=dhcp root=/dev/mmcblk0p1 rootfstype=ext2 mem=32M

Writing a Simple Program and Running it on the EVM

Make sure that you have performed the steps to export a shared file system for target access and install the Code Sourcery Tools.

Perform the following steps on the NFS host system as user (not as root). The steps below assume that the CodeSourcery tools are already present in your $PATH variable. Information on how to add the tools to your $PATH can be obtained form the Getting Started Guide for the CodeSourcery tools.

1. **host $** mkdir -p /home/<useracct>/workdir/filesys/opt/hello
2. **host $** cd /home/<useracct>/workdir/filesys/opt/hello
3. Create a file called hello.c with the following contents:

```
#include <stdio.h>
int main() {
        printf("Hello World!\n");
        return 0;
}
```

4. **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 or a telnet session.

1. **target $** cd /opt/hello
2. Run *hello*:

   target $ ./hello

   The output should be: Hello World!

What's Next?

Please continue on to the Building software components for OMAP-L1 or to Building software components for AM18x section of the Getting Started Guide.

References

[1] http://arago-project.org/files/short-term/snapshots/20100318-am18xx/images/da850-omapl138-evm/
[2] https://help.ubuntu.com/community/SettingUpNFSHowTo
[3] http://software-dl.ti.com/dsps/dsps_public_sw/sdo_sb/targetcontent/sdk/omap_l138/1_00/latest/index_FDS.html
[4] https://help.ubuntu.com/community/IptablesHowTo

OMAP-L1 Linux Drivers Usage

Before starting to use the drivers please read information on how to configure and rebuild the Linux kernel.

Ethernet

To test the ethernet interface, you need a valid IP address. To set a static IP address:

```
target$ ifconfig eth0 <static IP address>
```

To obtain an IP address dynamically using DHCP server on the network use:

```
target$ udhcpc
```

To test network connectivity, use ping command to the the network gateway

```
target$ ping <IP address of network gateway>
```

Audio

ALSA commands `aplay` and `arecord` can be used to playback and record audio respectively. ALSA mixer commands can be used to adjust the playback and capture volume. The command `amixer contents` lists all the params that can be controlled by mixer. To know the current volume settings enter:

```
target$ amixer cget numid=1
```

To change the playback volume, the following command can be used

```
target$ amixer cset numid=1 80,80 # Sets the volume to 80
target$ amixer cset numid=2 80,80 #This is also used to adjust playback
 volume
```

Character LCD

Character LCD can be tested by writing some characters/strings to the LCD device:

```
target$ echo "Hello, World" > /dev/lcd
```

The above command will display the string "Hello, World" on the character LCD.

The LCD device can be controlled by writing the following escape sequences to the LCD device (`echo <escape sequence> > /dev/lcd`):

NOTE
Pressing "Ctrl-v", "Ctrl-[" on keyboard will generate the "^[" escape sequence. In some file systems, the escape sequences may not appear on the serial console but the input will still reach the character LCD.

Escape Sequence	Functionality
^[[2J	Clear the display
^[[LI	Reinitialize display
^[[H	Cursor to home
^[[LD	Display ON
^[[Ld	Display OFF
^[[LC	Cursor ON
^[[Lc	Cursor OFF
^[[LB	Blink ON
^[[Lb	Blink OFF
^[[Ll	Shift Cursor Left
^[[Lr	Shift Cursor Right
^[[LL	Shift Display Left
^[[LR	Shift Display Right

RTC

Use the `hwclock` [1] utility to test RTC. The RTC device node created is `/dev/rtc0`

McBSP

The McBSP test setup on the EVM comprises of two EVMs connected via the audio expansion slot. One EVM will act as the master and the other as slave. There is a sample kernel module provided in the examples included in the PSP package which demonstrates the working of McBSP peripheral.

For McBSPs to be interfaced via the audio expansion slot, S7-2 should be turned ON. Note that on doing this audio will not work anymore.

Refer to the README file present with the McBSP example to know more about the module parameters that have to be passed to the sample module to configure the peripheral.

Power Management

CPUFreq

The cpufreq driver [2] in kernel allows scaling of clock speed of the CPU on the fly to save power. CPU consumes lower power at lower frequencies.

Please see CPUFreq User's Guide [3] for a comprehensive list of supported features.

Operating point

An operating point is a voltage frequency pair that defines a specific power state that the SoC has been characterized for. To see the list of available OPPs (frequencies):

```
target$ cat
/sys/devices/system/cpu/cpu0/cpufreq/scaling_available_frequencies
```

```
300000 200000 96000
target$
```

Governors

CPUfreq governor [4] is responsible to making the decision on the OPP to be used among the available OPPs. You can change the governor being used using sysfs entries.

To see the list of governers (the output of this command depends on the governors chosen at kernel configuration step):

```
target$ cat
/sys/devices/system/cpu/cpu0/cpufreq/scaling_available_governors
ondemand userspace performance
target$
```

To see the governor being used currently:

```
target$ cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
userspace
target$
```

To change governor at runtime:

```
target$ echo performance >
/sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
target$ cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
performance
target$
```

Frequency and Voltage scaling

When using the 'ondemand' governor, the kernel takes care of frequency transitions based on processor load. When using 'userspace' governor, the frequency transitions can be initiated from command line. For example:

To set the frequency to 96 MHz

```
target$ echo 96000 >
/sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed
```

NOTE
When using cpufreq only peripherals on a clock domain asynchronous with PLL0 will function correctly. The drivers in this release are not updated to take care of clock input changes. ASYNC3 domain has been configured to derive from PLL1, so is immune to CPU frequency changes.

To see the PLL0 frequency changes occurring because of frequency transitions:

```
target$ cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq
```

To see the CVDD voltage changes occurring because of CPU load:

```
target$ cat /sys/class/regulator/regulator.2/microvolts
1150000
target$
```

CPUIdle

The cpuidle driver [5] in Linux kernel allows CPU to idle efficiently by moving to a state with lower power consumption during idle. Once built in cpuidle driver works automatically and does not require any user intervention.

OMAP-L138 cpuidle driver implements two states

```
target$ ls /sys/devices/system/cpu/cpu0/cpuidle
state0  state1
target$
```

cpuidle state0 keeps ARM in WFI mode. state1 in addition puts the DDR2/mDDR in power-down mode. Details regarding DDR2/mDDR power down mode are documented in DDR2/mDDR Memory Controller User's Guide section 2.10.

```
target$ cat /sys/devices/system/cpu/cpu0/cpuidle/state0/desc
Wait for interrupt
target$ cat /sys/devices/system/cpu/cpu0/cpuidle/state1/desc
WFI and DDR Self Refresh
target$
```

To find usage of state0 and state1 so far since boot-up:

```
target$ cat /sys/devices/system/cpu/cpu0/cpuidle/state0/usage
6736
target$ cat /sys/devices/system/cpu/cpu0/cpuidle/state1/usage
92724
target$
```

Suspend-to-RAM

Suspend-to-RAM [6] allows the system to save power by freezing all tasks, asking all drivers to move the devices to a low power state (by disabling the peripheral clock using its LPSC), cutting the clock to external RAM and finally moving the SoC to DeepSleep mode. This user-initiated transtition saves the maximum power among all the power saving mechanisms available.

To put the system into Suspend-to-RAM state use `rtcwake` command.

```
rtcwake -d /dev/rtc0 -s 20 -m mem
```

In the above example, the system remains in suspended state for 20 seconds before coming back up.

IMPORTANT

If you are using MMC/SD card in your system, before suspending the system the MMC/SD card has to be unmounted. If the system is suspended without unmounting the card, the behaviour is undefined, the system may hang and not resume at all. This is a known behaviour for MMC/SD cards in Linux kernel.

Video Port Interface (VPIF)

The Video port interface (VPIF) driver supports NTSC and PAL standards for both display and capture. Currently the driver supports Composite and S-video capture and display.

The bootargs to be set for NTSC display and capture are

```
u-boot> setenv bootargs console=ttyS2,115200n8 noinitrd rw
ip=<ipaddr> root=/dev/nfs nfsroot=<nfs path> ,nolock
mem=32M@0xc0000000 mem=64M@0xc4000000 vpif_capture.ch0_bufsize=692224
vpif_display.ch2_bufsize=692224
```

The bootargs to be set for PAL display and capture are

```
u-boot> setenv bootargs console=ttyS2,115200n8 noinitrd rw
ip=<ipaddr> root=/dev/nfs nfsroot=<nfs path> ,nolock
mem=32M@0xc0000000 mem=64M@0xc4000000 vpif_capture.ch0_bufsize=831488
vpif_display.ch2_bufsize=831488
```

NOTE

The entire bootargs has to be entered in a single line.

Example test applications for standalone display and capture-display loopback are included in PSP package under `src/examples/vpif`. Please refer to the included `README` file for usage instructions.

Touchscreen

The touchscreen device node is created as `/dev/input/touchscreen0`. Before using the touchscreen, it needs to be calibrated. Use the command `ts_calibrate` to do that. To test the touchscreen functionality `ts_test` can be used.

Note that the SoC does not have a touchscreen controller integrated. The touchscreen functionality is provided by on-board components like TPS65070 or TSC2004.

Timers

Timers in Linux kernel are used to provide clock tick to the kernel and as watchdog timer.

DA830/OMAP-L137/AM17xx

The DA830/OMAP-L137/AM17xx SoCs have two 64-bit timers.

- Timer 0 is configured as two 32-bit timers by kernel at boot-up.
  - Bottom half (12) is used as clockevent and free-run counter.
  - Top half (34) left unused
- Timer 1 will be configured as watchdog timer if support is enabled.

DA850/OMAP-L138/AM18xx

The DA850/OMAP-L138/AM18xx SoCs have four 64-bit timers.

- Timer 0 is configured as two 32-bit timers by kernel at boot-up.
  - Bottom half (12) is used as clockevent.
  - Top half (34) is used as free-run counter.
- Timer 1 will be configured as watchdog timer if support is enabled.
- Timer 2 is left untouched.
- Timer 3 is left untouched.

References

[1] http://linux.die.net/man/8/hwclock
[2] http://lxr.linux.no/linux+v2.6.32/Documentation/cpu-freq/
[3] http://lxr.linux.no/linux+v2.6.32/Documentation/cpu-freq/user-guide.txt
[4] http://lxr.linux.no/linux+v2.6.32/Documentation/cpu-freq/governors.txt
[5] http://lxr.linux.no/linux+v2.6.32/Documentation/cpuidle/
[6] http://lxr.linux.no/#linux+v2.6.32/Documentation/power

OMAPL1: Changing the Operating Point

Overview of operating point (OPP) support

Operating Point (OPP) is nothing but a recommended operating condition for the SoC defined by a voltage and frequency pair (V/F pair) for the core (CVDD).

On the OMAP-L1 (or DA8x, AM1x) SoCs, the following set of OPPs are supported by the ARM UBL:

1. 456MHz, 1.3V
2. 408MHz, 1.2V
3. 372MHz, 1.2V
4. 300MHz, 1.2V

The set of OPPs that are supported by the target SoC are documented in the SoC data sheet.

Configuring ARM UBL for a specific OPP

The ARM UBL can be configured to support one OPP at a time. The configuration is done at the time of building the ARM UBL.

The file `include/device.h` in ARM sources defines a set of macros - one for each OPP. These macros have been listed here for reference.

```
#define DEVICE_OPP_1P2V_300MHZ   0x00
#define DEVICE_OPP_1P2V_348MHZ   0x01
#define DEVICE_OPP_1P2V_372MHZ   0x02
#define DEVICE_OPP_1P2V_408MHZ   0x03
#define DEVICE_OPP_1P3V_408MHZ   0x04
#define DEVICE_OPP_1P3V_456MHZ   0x05
```

To configure the UBL to a particular OPP one must set the define `DEVICE_CONFIG_OPP`OPP to one of the supported OPPs. For example, to configure UBL for (456MHz,1.3V) OPP:

```
#define DEVICE_CONFIG_OPP DEVICE_OPP_1P3V_456MHZ
```

CAUTION

Please refer to the data sheet for device you are using to make sure the OPP you are setting is actually supported by the device.

Changing OPP in Linux Kernel

On the DA850/OMAP-L138/AM1808 device, users can choose the maximum OPP to operate at from the Linux kernel. This is chosen during kernel configuration phase of kernel build process.

First enable CPU Frequency scaling (CPUFreq) feature of Linux kernel. The kernel uses this feature to change the OPP at runtime. Choose the `performance` governor if you just intend to operate at the highest OPP chosen.

```
CPU Power Management  --->
      [*] CPU Frequency scaling
            <*>   'userspace' cpufreq policy governor
            Default CPUFreq governor (ondemand)  --->
                  (X) performance
Device Drivers  --->
      [*] Voltage and Current Regulator Support  --->
            <*>   TI TPS6507X Power regulators
```

Now, select the maximum frequency the CPU should run at:

```
System Type  --->
      TI DaVinci Implementations  --->
            Select Maximum DA8xx/OMAP-L1/AM1xxx SoC speed (300 MHz)
--->
                           ( ) 300 MHz
                           ( ) 372 MHz
                           ( ) 408 MHz
                           (X) 456 MHz
```

Adding support for a new operating point (OPP)

In most use cases, the OPP pre-defined in PSP package should be sufficient. This section provides some general guidelines to help users add new OPP to the UBL code. These steps should be considered reference only. Depending on the OPP being added, the actual implementation may involve steps not documented here.

At a top level the following steps are required to add a new OPP:

1. Change the voltage supplied by the PMIC (Power Management IC) to the CPU (CVDD)
2. Change the the frequency generated by the CPU domain PLL (usually PLL0; please check with the device specific manual for the correct PLL number)
3. Configuration of EMIF interface timing parameters and frequency for the new OPP **NOTE**
   This step is required only of the changing the PLL configuration of CPU PLL also affects the EMIF interface.

Steps to add a new OPP to UBL:

1. Define a macro for the new OPP in `include/device.h`. Please refer to examples above
2. A new case should be added to the switch statement in `LOCAL_getDeviceOPP()` function in `include/device.c`, which returns the OPP configured based on the value of `DEVICE_CONFIG_OPP`
3. The `src/device.c` file contains functions for
   - PLL initialization
   - Memory Controller setup
   - Calls to PMIC setup function for setting a desired voltage.

   These functions are called from the `DEVICE_init()` function in `src/device.c`. The PLL initialization routines configure the required multiplier and divider ratios for various clock domains (like PLLDIVx for

SYSCLKx, POSTDIV, PLL multiplier etc.).

For example, considering the case of adding (456MHz, 1.3v) OPP support newly:

1. Change the `DEVICE_PLL0Init()` in `src/device.c` as follows:
   - PLLM is configured to be 18 to generate 456MHz (24MHz * (18 + 1)) with a 24MHz OSCIN input.
   - POSTDIV is configured to 0 thus actual divsion is by one (0 + 1).
   - In case of DA830/OMAPL137/AM1707, the SDRAM controller (EMIFB) and the EMIFA controller are also running off PLL0. Thus changes to PLL0 also affect the the SDRAM controller and EMIFA controller settings. The input clock to the SDRAM controller and EMIFA controller should be appropriately set so as to not cross their max allowed limits (please refer to device specific data manual for this value). For example, for max EMIFA freq of 100MHz, the PLLDIV3 should be set to 4 so that SYSCLK3 is set to 456 / (4 + 1) = 91.2MHz. Also, the source for the EMIFA clock should be set to be from SYSCLK3 and not from DIV4P5.
2. Changes for the required voltage level:
   The required voltage level can be configured by calling the appropriate `TPSxxx_set_DCDCy_voltage()` function. Here, `TPSxxx` is the on board PMIC (example TPS65070) and DCDCy is the appropriate regulator that needs to be programmed (example DCDC3). Support for TPS65070 and TPS65023 is already provided. Please refer to the board schematics for the PMIC part number and the regulator (supplying the CVDD) that needs to be configured.
3. Changes for specific peripherals:
   Other change that is required is the SPI prescalar settings. Elements of board design, device and SPI slave capabilities put a cap at the maximum frequency at which the SPI bus can operate (say 30MHz). To ensure this, the SPI peripheral prescalar value must be adjusted. For example, at 456MHz CPU frequency, the SPI module frequency is 227MHz. Thus the prescalar should be set to 7 so that the SPI bus frequency does not cross 30MHz. Changes required for this are done in `src/spi_mem.c`

GSG: DA8x/OMAP-L1/AM1x DVEVM Additional Procedures

^ Up to main **Getting Started Guide for OMAP-L1** Table of Contents

Flashing images

DA850/OMAP-L138/AM18xx

On the OMAP-L138 (or AM18xx) SoC, the ARM boots first. On boot-up, the ARM runs the ARM UBL in AIS file format. The purpose of the ARM UBL is to initialize the PLLs, mDDR, and other hardware. Once done, it copies the U-Boot into mDDR and starts it.

U-Boot is an open source boot loader and is responsible for booting the Linux kernel.

Flashing images to SPI Flash

There are two ways to flash images to SPI Flash:

- Serial flasher

1. See this page for instructions on using the command line serial flashing utility, which requires only a UART cable: Serial Boot and Flash Loading Utility for OMAP-L138

- CCS

1. The embedded emulation that comes with the EVM is XDS100 version 1, that does not support the ARM. So to connect to the ARM, an external emulator is necessary. Also, to use an external emulator, you need the full version of CCS, not the one that comes with the evm. If you do not have an external emulator, please use the Serial Flasher above.
2. Set the boot pins to emulation boot mode. This is done by setting switch S7 on the EVM according to the following table:

Pin#	1	2	3	4	5	6	7	8
Position	OFF	OFF	OFF	OFF	ON	OFF	OFF	ON

3. Start CCStudio and connect to the ARM. See details at: How to connect to the OMAP-L138/C6748 EVM board using CCS?. Ensure that you have the latest ARM GEL file from [[1]] correctly specified.
4. Execute the GEL function "Full EVM-->SPI1_PINMUX". In case of CCSv3, this appears under the `GEL` menu. In case of CCSv4, this appears under `Scripts` menu.
5. Load the SPI flasher tool on to the ARM. You can either use the pre-built binary shipped in the `images/utils/omapl1x8/` directory of the PSP installation (`spiflash-writer.out`), or you can build your own by following the steps in the section on Rebuilding the SPI Flash writer
6. Run the SPI flasher program. You will be prompted for the input file type and the file path. For booting from SPI, an ARM AIS image and U-Boot are required.
   - To burn the ARM AIS image, for OMAP-L138/AM18xx, type `armais` as the image type. When prompted for a file name, provide the path to `arm-spi-ais.bin` file. A pre-built image is located in the `images/boot-strap/omapl1x8/` directory of the PSP installation.
   - To burn U-Boot, run the SPI flasher program again, and type `uboot` as the image type. When prompted for a file name, provide the path to the `u-boot.bin` file. A pre-built image is located in the `images/u-boot/omapl1x8/` directory of the PSP installation.

7. Once the SPI flash has been written with the all the required files, disconnect CCStudio and power off the EVM. Proceed to boot from SPI flash

Flashing images to NAND Flash

Follow these steps to flash images to NAND Flash:

1. The embedded emulation that comes with the EVM is XDS100 version 1, that does not support the ARM. So to connect to the ARM, an external emulator is necessary. Also, to use an external emulator, you need the full version of CCS, not the one that comes with the evm. If you do not have an external emulator, you cannot load the NAND Flash using these instructions.
2. Obtain the latest ARM GEL file from LogicPD (http://www.logicpd.com/products/development-kits/zoom-omap-l138-experimenter-kit).Run the CCStudio Setup tool and ensure that the ARM GEL file is correctly specified.
3. Set the boot pins to emulation boot mode. This is done by setting switch S7 on the EVM according to the following table:

Pin#	1	2	3	4	5	6	7	8
Position	OFF	OFF	OFF	OFF	ON	OFF	OFF	ON

4. Start CCStudio and connect to the ARM. See details at: How to connect to the OMAP-L138/C6748 EVM board using CCS?. Ensure that you have the latest ARM GEL file from [[1]] correctly specified.
5. Execute the GEL function "Full EVM-->EMIFA_NAND_PINMUX". In case of CCSv3, this appears under the `GEL` menu. In case of CCSv4, this appears under `Scripts` menu.
6. Load the NAND flasher tool on to the ARM. You can either use the pre-built binary shipped in the `images/utils/omapl1x8` directory of the PSP installation (`nand-writer.out`), or you can build your own by following the steps in the section on Rebuilding the NAND Flash writer
7. Run the NAND flasher program. You will be prompted for the input file type and the file path. For booting from NAND, an ARM AIS image and U-Boot are required.
   - To burn the ARM AIS image, for OMAP-L138/AM18xx, type `armais` as the image type. When prompted for a file name, provide the path to `arm-nand-ais.bin` file. A pre-built image is located in the `images/boot-strap/omapl1x8/` directory of the PSP installation.
   - To burn U-Boot, type `uboot` as the image type. When prompted for a file name, provide the path to the `u-boot.bin` file.
8. Once the NAND flash has been written with the all the required files, disconnect CCStudio and power off the EVM. Proceed to boot from NAND flash.

Flashing images to NOR Flash

Follow these steps to boot from NOR Flash:

1. The embedded emulation that comes with the EVM is XDS100 version 1, that does not support the ARM. So to connect to the ARM, an external emulator is necessary. Also, to use an external emulator, you need the full version of CCS, not the one that comes with the evm. If you do not have an external emulator, you cannot load the NOR Flash using these instructions.
2. Obtain the latest ARM GEL file from LogicPD [2]. Run the CCStudio Setup tool and ensure that the ARM GEL file is correctly specified.
3. Set the boot pins to emulation boot mode. This is done by setting switch S7 on the EVM according to the following table:

Pin#	1	2	3	4	5	6	7	8

Position	OFF	OFF	OFF	OFF	ON	OFF	OFF	ON

4. Start CCStudio and connect to the ARM. See details at: How to connect to the OMAP-L138/C6748 EVM board using CCS?.
5. Execute the GEL function "Full EVM-->EMIFA_NOR_PINMUX". In case of CCSv3, this appears under the `GEL` menu. In case of CCSv4, this appears under `Scripts` menu.
6. Load the NOR flasher tool on to the ARM. You can either use the pre-built binary shipped in the `images/utils/omapl1x8` directory of the PSP installation (`norflash-writer.out`), or you can build your own by following the steps in the section on Rebuilding the NOR Flash writer.
7. Run the NOR flasher program. You will be prompted for the input file type and the file path. For booting from NOR, an ARM AIS image and U-Boot are required.
   - To burn the ARM AIS image, for OMAP-L138/AM18xx, type `armais` as the image type. When prompted for a file name, provide the path to `arm-nor-ais.bin` file. A pre-built image is located in the `images/boot-strap/omapl1x8/` directory of the PSP installation.
   - To burn U-Boot, type `uboot` as the image type. When prompted for a file name, provide the path to the `u-boot.bin` file built for NOR.
8. Once the NOR flash has been written with the all the required files, disconnect CCStudio and power off the EVM. Proceed to boot from NOR flash

DA830/OMAP-L137

On the OMAP-L137 and DA830 SoCs, the DSP boots first, on boot-up, the DSP runs the DSP UBL in AIS file format. The purpose of the DSP UBL is to wake up the ARM and start the ARM UBL which is present in raw binary form. The purpose of the ARM UBL is to initialize the PLLs, SDRAM, and other hardware. Once done, it copies the U-Boot into SDRAM and starts it. U-Boot is an open source boot loader and is responsible for booting the Linux kernel.

Flashing images to SPI Flash

There are two ways to flash images to SPI Flash:

- Serial flasher

1. See this page for instructions on using the command line serial flashing utility, which requires only a UART cable: Serial Boot and Flash Loading Utility for OMAP-L137

- CCS

1. Setup the EVM in "emulation debug" mode by setting SW2 switch as follows:

For EVM revisions A and B:

Pin #	**7**	**2**	**1**	**0**	**3**
Position	1	1	1	1	1

For EVM revisions after B:

Pin #	7	2	1	0	3
Position	1	1	1	1	0

2. Start CCStudio and connect to the DSP. Once done, connect to the ARM. Ensure that you have the latest DSP GEL file from Spectrum Digital Support site [3] correctly specified.
3. Load the SPI flasher tool on to the ARM. You can either use the pre-built binary shipped in the `images/utils/omapl1x7/` directory of the PSP installation (`spiflash_writer.out`), or you can build your own by following the steps in the section on Rebuilding the SPI Flash writer
4. Run the SPI flasher program. You will be prompted for the input file type and the file path. Three files are required: DSP AIS, ARM UBL and U-Boot.
   - To burn the DSP AIS image, type `dspais` as the image type. When prompted for a file name, provide the path to `dsp-spi-ais.bin` file. A pre-built image is located in the `images/boot-strap/omapl1x7/` directory of the PSP installation.
   - To burn the ARM UBL image, type `armubl` as the image type. When prompted for a file name, provide the path to `ubl-spi.bin` file. A pre-built image is located in the `images/boot-strap/omapl1x7/` directory of the PSP installation.
   - To burn U-Boot, run the SPI flasher program again, and type `uboot` as the image type. When prompted for a file name, provide the path to the `u-boot.bin` file. A pre-built image is located in the `images/u-boot/omapl1x7/` directory of the PSP installation.
5. Once the SPI flash has been written with the all the required files, disconnect CCStudio and power off the EVM. Proceed to boot from SPI flash.

Flashing images to NAND Flash

1. Setup the EVM in "emulation debug" mode by setting SW2 switch on base board as follows:

For EVM revisions A and B:

Pin #	7	2	1	0	3
Position	1	1	1	1	1

For EVM revisions after B:

Pin #	7	2	1	0	3
Position	1	1	1	1	0

2. On the User Interface card, set the SW1 switch as follows:

Pin #	1	2	3	4
Position	1	0	1	1

3. Start CCStudio and connect to the DSP. Once done, connect to the ARM. Ensure that you have the latest DSP GEL file from Spectrum Digital Support site [3] correctly specified.
4. Execute the GEL function `Setup_EMIFA_PinMux()`. In case of CCSv3, this appears under the `GEL` menu. In case of CCSv4, this appears under `Scripts` menu.
5. Load the NAND flasher tool on to the ARM. You can either use the pre-built binary shipped in the `images/utils/omapl1x7/` directory of the PSP installation (`nand_writer.out`), or you can build your own by following the steps in the section on Rebuilding the SPI Flash writer

6. Run the NAND flasher program. You will be prompted for the input file type and the file path. Three files are required: DSP AIS, ARM UBL and U-Boot.
   - To burn the DSP AIS image, type `dspais` as the image type. When prompted for a file name, provide the path to `dsp-nand-ais.bin` file. A pre-built image is located in the `images/boot-strap/omapl1x7/` directory of the PSP installation.
   - To burn the ARM UBL image, type `armubl` as the image type. When prompted for a file name, provide the path to `ubl-nand.bin` file. A pre-built image is located in the `images/boot-strap/omapl1x7/` directory of the PSP installation.
   - To burn U-Boot, run the SPI flasher program again, and type `uboot` as the image type. When prompted for a file name, provide the path to the `u-boot.bin` file. A pre-built image is located in the `images/u-boot/omapl1x7/` directory of the PSP installation.
7. Once the NAND flash has been written with the all the required files, disconnect CCStudio and power off the EVM. Proceed to boot from NAND flash

AM17xx

On the AM17xx SoC, the ARM boots first, on boot-up, the ARM runs the ARM UBL in AIS file format. The purpose of the ARM UBL is to initialize the PLLs, SDRAM, and other hardware. Once done, it copies the U-Boot into SDRAM and starts it. U-Boot is an open source boot loader and is responsible for booting the Linux kernel.

Flashing images to SPI Flash

1. Setup the EVM in "emulation debug" mode by setting SW2 switch as follows:

Pin #	**7**	**2**	**1**	**0**	**3**
Position	1	1	1	1	0

2. Start CCStudio and connect to the ARM. Ensure that you have the latest ARM GEL file from Spectrum Digital Support site [4] correctly specified.
3. Load the SPI flasher tool on to the ARM. You can either use the pre-built binary shipped in the `images/utils/omapl1x7/` directory of the PSP installation (`spiflash_writer.out`), or you can build your own by following the steps in the section on Rebuilding the SPI Flash writer
4. Run the SPI flasher program. You will be prompted for the input file type and the file path. Two files are required: ARM AIS and U-Boot.
   - To burn the ARM AIS image, type `armais` as the image type. When prompted for a file name, provide the path to `arm-spi-ais.bin` file. A pre-built image is located in the `images/boot-strap/am17xx/` directory of the PSP installation.
   - To burn U-Boot, run the SPI flasher program again, and type `uboot` as the image type. When prompted for a file name, provide the path to the `u-boot.bin` file. A pre-built image is located in the `images/u-boot/omapl1x7/` directory of the PSP installation.
5. Once the SPI flash has been written with the all the required files, disconnect CCStudio and power off the EVM. Proceed to boot from SPI flash

Flashing images to NAND Flash

1. Setup the EVM in "emulation debug" mode by setting SW2 switch on the base board as follows:

Pin #	7	2	1	0	3
Position	1	1	1	1	0

2. On the User Interface card, set the SW1 switch as follows:

Pin #	1	2	3	4
Position	1	0	1	1

3. Start CCStudio and connect to the ARM. Ensure that you have the latest ARM GEL file from Spectrum Digital Support site [4] correctly specified.
4. Execute the GEL function `Setup_EMIFA_PinMux()`. In case of CCSv3, this appears under the `GEL` menu. In case of CCSv4, this appears under `Scripts` menu.
5. Load the NAND flasher tool on to the ARM. You can either use the pre-built binary shipped in the `images/utils/omapl1x7/` directory of the PSP installation (`nand_writer.out`), or you can build your own by following the steps in the section on Rebuilding the SPI Flash writer
6. Run the NAND flasher program. You will be prompted for the input file type and the file path. Two files are required: ARM AIS and U-Boot.
   - To burn the ARM AIS image, type `armais` as the image type. When prompted for a file name, provide the path to `arm-nand-ais.bin` file. A pre-built image is located in the `images/boot-strap/am17xx/` directory of the PSP installation.
   - To burn U-Boot, run the NAND flasher program again, and type `uboot` as the image type. When prompted for a file name, provide the path to the `u-boot.bin` file.
7. Once the NAND flash has been written with the all the required files, disconnect CCStudio and power off the EVM. Proceed to boot from NAND flash

Flashing Boot Images on Linux Without CCS

This procedure mirrors that found for the Hawkboard [5]. It requires the Serial boot and Flash Package available here [6] and a compiled u-boot binary. The AIS boot image that is generated makes use of built-in ROM functions and does not use a UBL for booting.

Note: The following tools require the Mono Framework to run on a Linux system. Refer to your Linux distribution's package management instructions for information on how to check if Mono is installed or to install/update it.

OMAP-L138

1. Prepare a UART AIS boot image using the command-line HexAIS_OMAP-L138.exe and the attached INI file:

. Note that there are different versions of the INI files for the D800K002 ROM, as that ROM revision (which was part of the first silicon revision) does not correctly handle mDDR initialization. Also, each file has mDDR configuration setup for the either 132MHz or 150MHz operation. Comment out the one you do not want to use and un-comment the one you do intend to use. Alternatively, you can use the AISGen GUI tool to perform this step.

```
host$ mono ./HexAIS_OMAP-L138.exe -ini OMAP-L138_EVM_uart.ini -o
u-boot_uart.ais
```

3. Prepare a SPI AIS boot image using the command-line HexAIS_OMAP-L138.exe and the attached INI file. Alternatively, you can use the AISGen GUI tool to perform this step. This file should be placed in the /tftproot directory of your TFTP server so that it can be transferred to the EVM under u-boot using the Ethernet port.

```
host$ mono ./HexAIS_OMAP-L138.exe -ini OMAP-L138_EVM_spi.ini -o
u-boot_spi.ais
```

5. Set the EVM to boot in UART2 boot mode. This is done by setting switch S7 on the EVM according to the following table:

Pin#	1	2	3	4	5	6	7	8
Position	OFF	OFF	OFF	OFF	OFF	OFF	ON	ON

6. Start the command-line slh_OMAP-L138.exe tool to boot the board from your host PC.

```
host$ mono ../slh_OMAP-L138.exe -waitForDevice -v
AISUtils/u-boot_uart_L138.ais
```

8. Power up the EVM. The UART boot process will start. It may take as long as 15 seconds to complete.

```
-----------------------------------------------------
   TI Serial Loader Host Program for OMAP-L138
   (C) 2010, Texas Instruments, Inc.
   Ver. 1.65
-----------------------------------------------------


Platform is Windows.
Attempting to connect to device COM1...
Press any key to end this program at any time.

Entering AIS Parser

Waiting for the OMAP-L138...
(AIS Parse): Read magic word 0x41504954.
(AIS Parse): Waiting for BOOTME... (power on or reset target now)
(AIS Parse): BOOTME received!
(AIS Parse): Performing Start-Word Sync...
(AIS Parse): Performing Ping Opcode Sync...
(AIS Parse): Processing command 0: 0x5853590D.
(AIS Parse): Performing Opcode Sync...
(AIS Parse): Executing function...
(AIS Parse): Processing command 1: 0x5853590D.
(AIS Parse): Performing Opcode Sync...
(AIS Parse): Executing function...
(AIS Parse): Processing command 2: 0x58535901.
(AIS Parse): Performing Opcode Sync...
(AIS Parse): Loading section...
```

```
(AIS Parse): Loaded 156384-Byte section to address 0xC1080000.
(AIS Parse): Processing command 3: 0x58535906.
(AIS Parse): Performing Opcode Sync...
(AIS Parse): Performing jump and close...
(AIS Parse): AIS complete. Jump to address 0xC1080000.
(AIS Parse): Waiting for DONE...
(AIS Parse): Boot completed successfully.

Operation completed successfully.
```

10. When boot is completed, start a serial terminal program (Hyperterminal, minicom) to connect to the EVM. Press "enter" to see the u-boot prompt.
11. Erase first 256 KB of the serial flash

```
U-Boot > sf probe 0
U-Boot > sf erase 0 40000
```

13. Setup the u-boot enviroment for TFTP download.

```
U-Boot > setenv serverip 172.24.156.199
U-Boot > dhcp
```

15. Use TFTP to transfer the SPI u-boot image generated earlier.

```
U-Boot > tftpboot 0xc0700000 u-boot_spi.ais
```

17. Write the SPI AIS u-boot image to the start of the SPI flash.

```
U-Boot > sf write c0700000 0 40000
```

19. Power off the EVM, and change the boot mode switches to SPI0 master boot. This is done by setting switch S7 on the EVM according to the following table:

Pin#	1	2	3	4	5	6	7	8
Position	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF

20. Power on the EVM and the u-boot should start.

Booting U-Boot

U-Boot is an open source boot loader and is responsible for booting the Linux kernel.

Connect a serial cable from the serial port on the EVM to the COM port on the host machine. Set up the serial terminal software as described in Booting the EVM out of the box.

Note: Boot images may not have been pre-flashed on the EVM for all boot modes. In this case, follow the procedures in Flashing images to flash the required boot images.

DA850/OMAP-L138/AM18xx

Booting from SPI Flash

In order to boot from SPI flash, which has been written with the boot images, set the SW7 switch on the base board as follows:

Pin#	1	2	3	4	5	6	7	8
Position	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF

Booting from NAND Flash

In order to boot from NAND flash, which has been written with the boot images, set the SW7 switch on the base board as follows:

Pin#	1	2	3	4	5	6	7	8
Position	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF

Booting from NOR Flash

In order to boot from NOR flash, which has been written with the boot images, set the SW7 switch on the base board as follows:

Pin#	1	2	3	4	5	6	7	8
Position	OFF	OFF	OFF	OFF	ON	ON	ON	OFF

DA830/OMAP-L137/AM17xx

Booting from SPI Flash

- Set the SW2 switch on the DSK board as follows. (X indicates the setting is 'don't care')

Pin#	**7**	**2**	**1**	**0**	**3**
Position	0	1	0	1	X

Booting from NAND Flash

- Set the SW2 switch on the DSK board as follows. (X indicates the setting is 'don't care')

Pin#	**7**	**2**	**1**	**0**	**3**
Position	0	1	1	1	X

On the User Interface card, set the SW1 switch as follows:

Pin #	**1**	**2**	**3**	**4**
Position	1	0	1	1

Booting the Linux kernel using U-Boot

Booting the kernel requires a valid kernel image (`uImage`) and a target filesystem. A pre-built kernel image is included in the `images/kernel` directory of the PSP installation. Pre-built file system images can be found in your PSP tools installation at `images/fs`.

Booting Linux kernel using U-Boot describes various methods to boot Linux kernel.

Using extended memory available on OMAP-L138 EVM board

The OMAP-L138 eXperimenter board has 64MB of memory and the OMAP-L138 EVM has 128 MB available. Out of this 128 MB, some amount is used for Linux and the rest can be used for DSP or as DSP/ARM shared memory. Sometimes it is convenient to allow Linux to use discontinuous blocks of memory. Kernel parameter `mem=` can be used for this purpose. One example is shown here:

```
U-Boot> setenv bootargs console=ttyS2,115200n8 root=/dev/ram0 rw
initrd=0xc1180000,4M ip=dhcp mem=32M@0xc0000000 mem=64M@0xc4000000
```

If you plan on using Codec Engine (or DSPLink and CMEM alone), you need to make sure that the insmod command for the cmemk.ko kernel module uses the "allowOverlap=1" option (see the CMEM documentation available in the LinuxUtils package). Failure to use this option will result in an error message, since the kernel still reports its memory usage to CMEM as a contiguous 96M range.

Creating bootable SD card for OMAP-L138 EVM board

The UBL on OMAP-L138 expects the u-boot descriptor to be present in sectors 1 to 25 of the SD card. Sector 0 is used for storing DOS partition information. Hence the u-boot descriptor is stored from sectors 1 till 24. The descriptor is 512 bytes in size and is replicated in each of these sectors. U-Boot binary starts at Sector 117.

The SD card shall be re-partitioned and formated to create some room for storing u-boot. Use fdisk utility (as super user) to delete the existing partition and create a new one.

```
host$ fdisk /dev/mmcblk0 (device name might change if USB card reader
is used)
```

- Delete the existing partitions with 'd' command.
- Create a new partition with 'n' command, followed by 'p' command
- Mark the first cylinder as 20. Typical cylinder size is 32KBytes. So starting the first cylinder at 20 provides us about 600Kbytes for storing UBL and u-boot. If the fdisk utility displays a different cylinder size, make sure that you are leaving atleast 500K space before the first cylinder.
- Leave the last cylinder to default value (or) any other value depending on the partition size requirements.
- Save and exit with 'w' command

Using the **uflash** utility, place the u-boot binary on the SD card. Copy the u-boot.bin to uflash directory,

```
host$ ./uflash -d /dev/mmcblk0 -b u-boot.bin -p omapl138 -vv
        u-boot Size 252087
        U-Boot Magic Number     : a1aced66
        U-Boot Entry Point      : c1080000
        U-Boot Number of Blocks : 000001ec
        U-Boot Starting Block   : 00000075
        Load U-Boot Address     : c1080000
        Writing U-Boot Signature
```

```
        Writing U-Boot
        Done...
```

SD Card Writer Utility (uflash)

This is a Linux command line tool specific to TI's Davinci platforms, for flashing UBL/U-Boot to SD card.

Building uflash

1. Use your Linux host to extract the uflash source code from the `src/utils/mmcsd-writer-#.#.#.#.tar.gz` tarball from the OMAP-L138 Linux PSP package, which is located in the DaVinci-PSP-SDK-#.#.#.# directory under the main SDK installation directory. Use the `tar` command to extract the sources.
2. Use the native Linux host gcc compiler to build uflash

```
host$ gcc uflash.c -o uflash
```

Loading Linux Kernel Modules

Many of the kernel features can be built as run-time loadable modules so that they are not part of the kernel image, but can be inserted into the kernel at run-time to increase the running kernel's functionality.

To understand how to configure some features as modules and how to build them, refer to Rebuilding the Linux Kernel and Driver configuration in the Linux kernel.

Pre-built binaries for features configured by default as kernel modules are included in the PSP package in the `images/kernel/omapl1x7/modules/lib` directory for OMAP-L137 and in the `images/kernel/omapl1x8/modules/lib` directory for OMAP-L138. To use these modules, copy the contents of this directory to the /lib directory of your root file system.

On the target Linux command prompt use command `modprobe` command to load the module. `rmmod` command can be used to remove the module. Below is an example of loading the USB file storage gadget module. Similar steps can be followed for any driver module.

Loading USB 2.0

To load the USB file backed storage gadget:

1. Insert the g_file_storage.ko module with the following command where `/dev/blockdevX` is the storage device acting as the actual storage space. Replace it with the path to the actual block device name acting as physical storage, such as a MMC/SD card.

```
target$ modprobe g_file_storage file=/dev/blockdevX
```

3. Remove the module by using

```
target$ rmmod g_file_storage
```

DSP wakeup in U-Boot

On the OMAP-L138 SoC, the U-Boot wakes up the DSP by default. The DSP reset vector is set to `0x80000000`, and after wakeup it executes the `idle` instruction. To prevent the DSP from being woken up by U-Boot, you can do the following:

1. Set the `dspwake` environment variable to "no".
2. Save the environment space.
3. Reset the board.

This feature has been added to help DSP users wake up the DSP quickly, since on the OMAP-L138 EVM only the on-board emulation can access the DSP.

Modifying SPI Frequency in U-Boot

On DA850/OMAP-L138/AM18xx EVMs, in U-Boot, SPI flash has been configured to work at 30MHz. But in some situations one may have to modify the SPI frequency. For example:

- When operating at different voltages.
- When customer has a different SPI flash which operates at different frequency.

In such cases, SPI frequency can be changed in U-Boot by modifying the `CONFIG_SF_DEFAULT_SPEED` variable in `u-boot/include/configs/da850evm.h` file. After this modification, **rebuild u-boot** and **flash the new u-boot** binary file to SPI flash.

Restoring factory default U-Boot environment variables

To restore factory default U-Boot environment variables, the existing environment variables need to be erased from flash. You will need to know the offset in flash were environment variables are stored and the size of flash dedicated to storing the environment variables.

These values are represented by `CONFIG_ENV_OFFSET` and `CONFIG_ENV_SIZE` respectively in U-Boot EVM configuration file.

For OMAP-L138 (or DA850, AM18xx) SoCs, this file is `include/configs/da850_evm.h` inside the U-Boot source tree.
For OMAP-L137 (or DA830, AM17xx) SoCs, this file is `include/configs/da830_evm.h` inside the U-Boot source tree.

Follow this procedure based on the flash U-Boot stores its environment variables on. The size and offset values given below are to considered representative only.

SPI flash

On SPI flash, environment variables are stored at offset 256 KBytes into the flash and environment variables sector size is 64 KBytes.

1. On the U-Boot command prompt:

```
U-Boot> sf probe 0
```

3. Erase environment sector length bytes (64 KBytes) from environment sector offset (256 KBytes) as follows:

```
U-Boot> sf erase 40000 10000
```

5. Reset the board and it will come up with default environment variables.
6. Save the new set of environment variables.

```
U-Boot> saveenv
```

NAND Flash

On the NAND flash, env variables are stored at offset 0 (zero) and environment variables sector size is 128 KBytes.

1. Erase environment sector length bytes (128 KBytes) from environment sector offset 0 (zero) as follows:

```
U-Boot> nand erase 0 20000
```

3. Reset the board and it will come up with default environment variables.
4. Save the new set of environment variables.

```
U-Boot> saveenv
```

NOR Flash

On the NOR flash, env variables are stored at the 4th sector of NOR flash and environment variables sector size is 128 KBytes.

1. Unprotect the 4th sector using:

```
U-Boot> protect off 60060000 +20000
```

3. Erase environment sector length bytes (128 KBytes) from environment sector offset 60060000 as follows:

```
U-Boot> erase 60060000 +20000
```

5. Reset the board and it will come up with default environment variables.
6. Save the new set of environment variables.

```
U-Boot> saveenv
```

Enabling Write-Back Cache on DA830/OMAP-L137

On the DA830 and OMAP-L137 SoCs silicon revisions 1.1 and earlier, the ARM cache in write-back mode is not functional. This has been resolved starting silicon revision 2.0. To maintain backward compatibility with affected silicon revisions, the Linux kernel keeps write-back mode disabled in the default configuration. Users of silicon revisions 2.0 and higher can enable write-back cache from kernel configuration.

```
System Type  --->
    [ ] Force write through D-cache
```

Message logging on UART in ARM UBL

One may wish to enable messaging logging on the UART console. This feature can be enabled by supplying DEVICE_UART<n>_FOR_DEBUG, where <n> is the UART number. Presently UART0, UART1 and UART2 can be enabled for logging. However, UART<n> may be available for logging depending on the PINMUXing with core peripheral like NAND/SPI/NOR etc and the BOOT mode one wishes to have. Please check this and the settings in device.c/UARTInit()

For example if one wishes to send the console debug messages over UART2 the compiler flag needs to supplied as below (shown for NAND boot mode):

```
Options=-g -o3 -fr"$(Proj_dir)\..\nand" -fs"$(Proj_dir)\..\nand"
-i"$(Proj_dir)\..\include" -d"UBL_NAND" -d"USE_IN_ROM"
```

```
-d"DEVICE_UART2_FOR_DEBUG" -me -mv5e --abi=ti_arm9_abi
```

Linux Functional Test Bench

The Linux Functional Test Bench (LFTB) is the set of tools used to verify the various driver features. The test bench is included in the `test-suite` directory of the PSP installation. Use the `tar` command to extract the LFTB package.

Information on how to use LFTB and its various features is included in the LFTB package itself.

What's next?

Please continue on to the **Additional information** section of the OMAP-L1 Getting Started Guide.

References

[1] http://www.logicpd.com/products/development-kits/zoom-omap-l138-experimenter-kitlLogicPD
[2] http://www.logicpd.com/products/development-kits/zoom-omap-l138-experimenter-kit
[3] http://support.spectrumdigital.com/boards/evmomapl137/revd/
[4] http://support.spectrumdigital.com/boards/evmam1707
[5] http://elinux.org/Hawkboard#Booting
[6] http://sourceforge.net/projects/dvflashutils/files/OMAP-L138/v2.11/OMAP-L138_FlashAndBootUtils_2_11.tar.gz/download

GSG: Building Software Components for OMAP-L1/AM1x

^ Up to main **Getting Started Guide for OMAP-L1** Table of Contents

Rebuilding the Linux kernel

The rebuild the Linux kernel, follow the steps below.

Preparing your Environment

1. If you have not already done so, install the CodeSourcery tools on your Linux host. For additional documentation on installing the CodeSourcery tools, please visit the CodeSourcery website [1].
2. If U-Boot is not built yet, build U-Boot first, as building Linux Kernel requires `mkimage` utility, which gets built along with U-Boot. Refer to Rebuilding U-Boot for steps to build U-Boot. Once built, mkimage will be present under the tools folder of U-Boot source (`/home/<user>/OMAP_L138_arm_x_xx_xx_xx/DaVinci-PSP-SDK-xx.xx.xx.xx/src/u-boot/uboot-xx.`
   Ensure that this path is added to the `$PATH` variable by adding the following

   **host$** export PATH=home/<user>/OMAP_L138_arm_x_xx_xx_xx/DaVinci-PSP-SDK-xx.xx.xx.xx/src/u-boot/uboot-xx.xx.xx.xx/tools:$PA
4. to the ~/.bashrc file. You can then
5. *source*
6. the ~/.bashrc file with the following command

   host$ source ~/.bashrc
8. Use your Linux host to extract source files for building the target Linux kernel from the `src/kernel/linux-#.#.#.#.tar.gz` tarball from the OMAP-L138 Linux PSP package, which is located

in the DaVinci-PSP-SDK-#.#.#.# directory under the main SDK installation directory (/home/<user>/OMAP_L138_arm_x_xx_xx_xx/DaVinci-PSP-SDK-xx.xx.xx.xx/src/kernel for default path installation). Use the `tar` command to extract the sources.

host$ tar zxf linux-#.#.#.#.tar.gz

NOTE

Patches from the kernel-patches-#.#.#.#.tar.gz file have already been applied on Linux Kernel. This is the list of patches which have been developed on top of the base Linux kernel version. You can find information about the base Linux kernel version from the release notes accompanying the PSP release.

11. Change directory (`cd` command) to the top-level directory of the Linux kernel source files obtained in the previous step.

host$ cd linux-xx.xx.xx.xx

Basic Configuration of the Kernel

Based on what chip and development board you are using, you will need to configure the kernel for different options. Most of these setings are build into the Make file provided by TI, and you can switch between builds using some simple make options.

1. Clean your installation of previous build settings

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

3. Configure the kernel according to the default configuration provided. The steps below assume that the CodeSourcery tools are already present in your `$PATH` variable. Information on how to add the tools to your `$PATH` can be obtained form the *Getting Started Guide* for the CodeSourcery tools.

- For OMAP-L138 (or DA850, AM18xx)

host$ make da850_omapl138_defconfig ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi-

- For OMAP-L137 (or DA830, AM17xx)

host$ make da830_omapl137_defconfig ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi-

5. Modify the kernel options you would like to run. This may take some planning, but the default settings are a good place to start. For more information on these settings, look further down the page at Driver configuration in the Linux kernel or see detaile instructions for | a different processors root file system [2]]. there are two menu configs that are commmon to run:

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

(or)

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

Building uImage

Run the following command to build the kernel image:

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

The compiled `uImage` is copied into the `arch/arm/boot` directory under the kernel tree.

Once the kernel has been compiled, **if you are using tftp boot**, use the following commands to copy the uImage to a place where U-Boot can use TFTP to download it to the EVM. These commands assume you are using the default TFTP root area, which is /tftpboot. If you use another TFTP root location, please change /tftpboot to your own TFTP root location:

host$ cp arch/arm/boot/uImage /tftpboot

Note

If the uImage build fails with a similar message as follows:

```
"mkimage" command not found - U-Boot images will not be built
```

it could be that the path to the u-boot uImage script has not been added to the command search path. Include the path "{u-boot source path}/tools" to your PATH environment variable.

Building Modules

To build all features configured as modules (M), issue the following command:

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

Installing modules to Target File System

To install the compiled modules into the target root file system, issue the following command:

host$ make modules_install INSTALL_MOD_PATH=<root fs path> ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi-

where the `<root fs path>` is the path of your target root file system on the host machine (/home/user/workdir/filesys for example).

See Loading Linux Kernel Modules for more information on using kernel modules after you build them.

Driver configuration in the Linux kernel

This section describes the procedure to configure the kernel to support various drivers.

- To begin, if you have not done basic configuration of the kernel go to that section, and do that first. Once you have compiled with base configuration, return here.

Once you base configuration is set, you can use menu based configurations to change details about your kernel's build.

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

- To select/de-select a feature, press 'space' after bringing cursor over selection box. When '*' appears in selection box, the feature is selected, when the selection box is empty, the feature is de-selected.
- To configure a particular feature as module, press 'space' until an 'M' appears in the selection box.

Note that some of the menu options which are required to be deselected here (selection box empty) may not appear at all because the internal checks which render the option invalid. In this case, it is safe to assume that the options have been automatically disabled.

USB 2.0

USB2.0 interface is build into the default config in Dual Role mode meaning it can either be a Host or Device (OTG is not supported) automatically without need for user intervention. In the Dual Role mode support for MSC application is available in Host mode while RNDIS/CDC gadget application is available in Device mode by default. User can choose to change the application/mode supported by following the below steps if needed.

WARNING
When removing the support for USB2.0 from the kernel please ensure that "CPPI support" under "System Type--> TI DaVinci Implementations" is also disabled.

Configuring for Host

```
Device Drivers --->

    USB support --->
```

```
        <*> Support for Host-side USB
        --- Miscellaneous USB options
        [*] USB device filesystem
        --- USB Host Controller Drivers
        <*> Inventra USB Highspeed Dual Role Controller Support
        --- DA830/OMAP-L137 USB support
            Driver Mode (USB Host) --->
                (X) USB Host
```

(please see Inventra HDRC USB Controller for more details on the underlying driver)

When required to support HID devices (mouse, keyboard etc), choose the following

```
Device Drivers --->
      Input device support --->
            <*> Mouse interface
            [*] Keyboards --->
                  <*> AT keyboard

      HID Devices --->
            <*> USB Human Interface Device(full HID) support
```

When required to support MSC devices (pen drive etc), choose the following

```
Device Drivers --->
      SCSI device support --->
            --- SCSI device support
            [*] legacy /proc/scsi/support
            --- SCSI support type (disk, tape, CD-ROM)
            <*> SCSI disk support
      USB support --->
            <*> USB Mass Storage support
```

When required to support UVC, UAC class (Webcam, Speakers, Microphone etc.) choose the following

```
Device Drivers --->
      Multimedia support --->
            Video capture adapters --->
                  V4L USB devices --->
                        <*> USB Video Class (UVC)
                        [*] UVC input events device support
      Sound card support --->
            Advanced Linux Sound Architecture --->
                  USB sound devices --->
                        <*> USB Audio/MIDI driver
```

Configuring for Gadget

```
      Device Drivers --->
                USB support --->
                    < > Support for Host-side USB
                    <*> Inventra Highspeed Dual Role Controller
```

```
(TI, ...)
                        (*)USB Peripheral (gadget stack)
```

When required to support Ethernet gadget choose the following

```
      Device Drivers --->
            USB support --->
                  USB Gadget Support --->
                        <*> USB Gadget Drivers (Ethernet Gadget
(with CDC Ethernet support))
                              [*]     RNDIS support (EXPERIMENTAL)
```

When required to support File backed storage gadget, choose the following

```
      Device Drivers --->
            USB support --->
                  USB Gadget Support --->
                           <M> USB Gadget Drivers
                              <M> File-backed Storage Gadget
                        [*] File-backed Storage Gadget testing version
(NEW)
```

USB 1.1

```
Device Drivers --->
            USB support --->
                  <*> Support for Host-side USB
                  --- Miscellaneous USB options
                  [*] USB device filesystem
                  --- USB Host Controller Drivers
                  <*>   OHCI HCD support
```

To support MSC, HID, UVC, UAC applications over USB 1.1 interface refer to USB2.0 procedures for the same.

Audio

```
Device Drivers --->
    <*> Sound card support --->
        <*> Advanced Linux Sound Architecture --->
            <*> ALSA for SoC audio support --->
                <*> SoC Audio for the TI DAVINCI chip
```

For OMAP-L138 (or DA850, AM18xx) EVM board:

```
                <*> SoC Audio support for DA850/OMAP-L138/AM18xx
EVM
```

For OMAP-L137 (or DA830, AM17xx) EVM board:

```
                <*> SoC Audio support for DA830/OMAP-L137/AM17xx
EVM
```

Graphical LCD

```
Device Drivers --->
         Graphics support  --->
             <*> Support for frame buffer devices --->
             <*> DA8xx/OMAP-L1xx/AM1xxx Framebuffer support
```

When using OMAP-L137 (or DA830, AM17xx) EVM:

```
System Type --->
         TI DaVinci Implementations  --->
             Select DA830/OMAP-L137/AM17xx UI board peripheral (LCD)
--->
```

Character LCD

```
Device Drivers --->
         Graphics support  --->
             <*> Support for frame buffer devices --->
                < > DA8xx/OMAP-L1xx/AM1xxx Framebuffer support
         <*> Parallel port support  --->
         [*] Staging drivers  --->
             [ ]   Exclude Staging drivers from being built
             <*>       Parallel port LCD/Keypad Panel support
             (0)       Default parallel port number (0=LPT1)
             (3)       Default panel profile (0-5, 0=custom)
             [ ]       Change LCD initialization message ?
             -*-       DA8XX/OMAP-L1/AM1xxx Dummy Parallel Port
```

When using OMAP-L137 (or DA830, AM17xx) EVM:

```
System Type --->
         TI DaVinci Implementations  --->
             Select DA830/OMAP-L137/AM17xx UI board peripheral (LCD)
--->
```

NAND

```
Device Drivers --->

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

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

WARNING
Please disable MMC support when NAND has to be used. They are pin multiplexed and NAND will not work when MMC is enabled.

NOR

Note: NOR flash is supported only on OMAP-L138 (or DA850, AM18xx) EVM

```
Device Drivers --->

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

    <*> Memory Technology Device (MTD)  support --->
        [*]   MTD partitioning support
        [*]   Command line partition table parsing
        <*>   Direct char device access to MTD devices
        <*>   Common interface to block layer for MTD
'translation layers'
        <*>   Caching block device access to MTD devices
              RAM/ROM/Flash chip drivers --->
                 <*> Detect flash chips by Common Flash Interface
 (CFI) probe
                 <*> Support for Intel/Sharp flash chips
              Mapping drivers for chip access --->
                 <*> TI DaVinci board mappings
```

WARNING
Please disable MMC support when NOR has to be used. They are pin multiplexed and NOR will not work when MMC is enabled.

SPI

```
Device Drivers --->

        [*] SPI support  --->
            <*>     SPI controller driver for DaVinci SoC

        <*> Memory Technology Device (MTD) support  --->
        [*]   MTD partitioning support
        [*]   Command line partition table parsing
        <*>   Direct char device access to MTD devices
        <*>   Common interface to block layer for MTD
'translation layers'
        <*>   Caching block device access to MTD devices
              Self-contained MTD device drivers  --->
                  <*> Support most SPI Flash chips (AT26DF, M25P,
 W25X, ...)
                  [*]   Use FAST_READ OPCode allowing SPI CLK <=
50MHz
```

MMC/SD

```
Device Drivers --->

    <*>MMC/SD/SDIO card support  --->
        <*>   MMC block device driver
        <*>   TI DAVINCI Multimedia Card Interface support
```

RTC

```
Device Drivers --->

    <*> Real Time Clock  --->
        <*>   TI OMAP1
```

SATA

```
      Device Drivers --->
            <*> Serial ATA (prod) and Parallel ATA (experimental)
 drivers --->
                  [*] SATA Port Multiplier support
                  <*> AHCI SATA support
```

McBSP

```
System Type --->
      TI DaVinci Implementations --->
                [*] DaVinci McBSP (serial API) support
                      [ ]   Support for McBSP instance 0
                      [*]   Support for McBSP instance 1
```

WARNING

Note: Since the McBSP peripheral has a kernel API based driver, menuconfig options are present under System Type. If sound driver is selected in menuconfig then McBSP cannot be configured. They are mutually exclusive.

Power Management

CPUFreq

```
CPU Power Management  --->
      [*] CPU Frequency scaling
            <*>   'userspace' cpufreq policy governor
            Default CPUFreq governor (userspace)  --->
                  (X) usespace
Device Drivers  --->
      Multifunction device drivers  --->
            <*> TPS6507x Power Management / Touch Screen chips
      [*] Voltage and Current Regulator Support  --->
            <*>   TI TPS6507X Power regulators
```

CPUIdle

```
CPU Power Management  --->
      [*] CPU idle PM support
```

Suspend-to-RAM

```
Power management options  --->
       [*] Power Management support
       [*] Suspend to RAM and standby
```

Ethernet

```
[*] Networking support  --->
      Networking options  --->
            [*] TCP/IP networking
            [*]   IP: kernel level autoconfiguration
            [*]     IP: DHCP support
Device Drivers --->
            [*] Network device support  --->
            [*]   Ethernet (10 or 100Mbit)  --->
                  <*>   Generic Media Independent Interface
device support
                  <*>   TI DaVinci EMAC Support
```

Using the RMII PHY on OMAP-L138 (or DA850, AM18xx) UI card

```
System Type --->
      TI DaVinci Implementations --->
            [*] Use RMII Ethernet PHY on DA850/OMAP-L138 EVM
Device Drivers --->
        -*- GPIO Support  --->
               <*>   PCA953x, PCA955x, TCA64xx, and MAX7310 I/O
ports
```

WARNING
Note: When using RMII PHY, MII ethernet PHY will not be fucntional. Do not plug in ethernet cables to both the PHYs.

VPIF

```
System Type  --->
        TI DaVinci Implementations  --->
                [*] TI DA850/OMAP-L138/AM18xx Reference Platform
                      Select peripherals connected to expander on UI
board (Video Port Interface)  --->

Device Drivers --->
    <*> Multimedia support  --->
            <*>   Video For Linux
                [*]   Video capture adapters  --->
                        <*>  DaVinci Video VPIF Display
```

```
                        <*>  DaVinci Video VPIF Capture
                        -*-  DaVinci VPIF Driver
                [ ]   Autoselect pertinent encoders/decoders and other
helper chips
                      Encoders/decoders and other helper chips  --->

                        <*> Texas Instruments TVP514x video
decoder
                        -*- THS7303 Video Amplifier
                        -*- ADV7343 video encoder
```

WARNING
Please disable the graphical LCD frame buffer driver and the character LCD driver when VPIF display has to be used. They are pin multiplexed and VPIF display will not work when LCD is enabled.

Touchscreen

TSC2004

TSC2004 touchscreen controller is found on DA830 (or OMAP-L137, AM17xx) EVM when using new UI cards

```
Device Drivers --->
        Input device support  --->
                [*]   Touchscreens  --->
                        <*>   TSC2004 based touchscreens
```

TPS65070

TPS65070 touchscreen controller is found on DA850 (or OMAP-L138, AM18xx) EVM

```
Device Drivers --->
      Multifunction device drivers  --->
            <*> TPS6507x Power Management / Touch Screen chips
        Input device support  --->
                [*]   Touchscreens  --->
                        <*>   TPS6507x based touchscreens
```

Rebuilding U-Boot

Follow these steps to rebuild U-Boot:

1. If you have not already done so, install the CodeSourcery tools on your Linux host. For documentation on installing the CodeSourcery tools, please visit the CodeSourcery website [1].
2. Use your Linux host to extract source files for building U-Boot from the `src/u-boot/u-boot-#.#.#.#.tar.gz` tarball from the OMAP-L138 Linux PSP package, which is located in the DaVinci-PSP-SDK-#.#.#.# directory under the main SDK installation directory. Use the `tar` command to extract the sources.
   Note: Patches from the uboot-patches-#.#.#.#.tar.gz file have already been applied to U-Boot. This is the list of patches which have been developed on top of the base version. You can find information about the base U-Boot version from the release notes accompanying the PSP release.
3. Go to the u-boot directory created when you extracted the files.

4. Run the following commands on your Linux host to build U-Boot. **Note:** The steps below assume that the CodeSourcery tools are already present in your `$PATH` variable. Information on how to add the tools to your `$PATH` can be obtained form the *Getting Started Guide* for the CodeSourcery tools.

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

- for OMAP-L138 (or DA850, AM18xx) EVM

```
host$ make da850_omapl138_evm_config
CROSS_COMPILE=arm-none-linux-gnueabi-
```

- for OMAP-L137 (or DA830, AM17xx) EVM

```
host$ make da830_omapl137_evm_config
CROSS_COMPILE=arm-none-linux-gnueabi-
```

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

8. The compiled `u-boot.bin` file will be created in the same directory.
9. To change the default options, the EVM configuration file needs to be edited.
   - for OMAP-L138 (or DA850, AM18xx) EVM are specified in the include file `include/configs/da850_evm.h`
   - for OMAP-L137 (or DA830, AM17xx) EVM are specified in the include file `include/configs/da830_evm.h`

   To change U-Boot environment area location or size:
   - `CONFIG_ENV_SIZE` Configures the environment variable size.
   - `CONFIG_ENV_OFFSET` Configures the environment variable offset.

   Choice of Flash supported. **Note:** Only one of these Flash options should be defined at a time. Defining more than one Flash option results in a compilation error when you build U-Boot:
   - `CONFIG_USE_SPIFLASH` If this flag is defined, U-Boot supports the SPI flash on the EVM board. The environment variables are stored on the SPI flash. This option is switched on by default.
   - `CONFIG_SYS_USE_NAND` If this flag is defined, U-Boot supports U-Boot NAND flash access using the OMAP-L1 SoC. Environment variables are also stored on the NAND flash.

Rebuilding the ARM Side User Boot Loader

Use your Linux host to extract the ARM UBL code from the `src/boot-strap/armubl-#.#.#.#.tar.gz` tarball from the Linux PSP package, which is located in the DaVinci-PSP-SDK-#.#.#.# directory under the main SDK installation directory. Use the `tar` command to extract the sources. If the extracted files are not accessible from your Microsoft Windows host, copy all the extracted files to a location that is accessible.

NOTE

By default, the ARM UBL configures the SoC to work at 300MHz, 1.2V operating point. → Changing the Operating Point describes the procedure to change the default operating point.

For OMAP-L138 (or DA850, AM18xx)

When using **CCStudiov3.3**:

1. Start CCStudio v3.3.
2. From the menus, choose **Project->Open**.
3. Browse to the extracted ARM UBL source and open the `ubl_omapl1x8.pjt` project.
   - To build for SPI Flash, select the "Config" option as `BOOT_SPI` in CCStudio window.
   - To build for NOR Flash, select the "Config" option as `BOOT_NOR` in CCStudio window.
   - To build for NAND Flash, select the "Config" option as `BOOT_NAND` in CCStudio window.
   - To build for MMC/SD boot, select the "Config" option as `BOOT_SDMMC` in CCStudio window. **NOTE** DA850/OMAP-L138/AM18xx does not support MMC/SD boot mode, but UBL can read U-Boot from SD card and boot it.
4. From the menus, choose **Project->Build**. When the build is complete:
   - The executable ELF binary (ubl-xxx.out) file is generated in the Project Root directory `(<armubl-install-dir>\ccsv3.3\)`
5. Proceed to convert the UBL object file obtained above to AIS format file suitable for flashing.

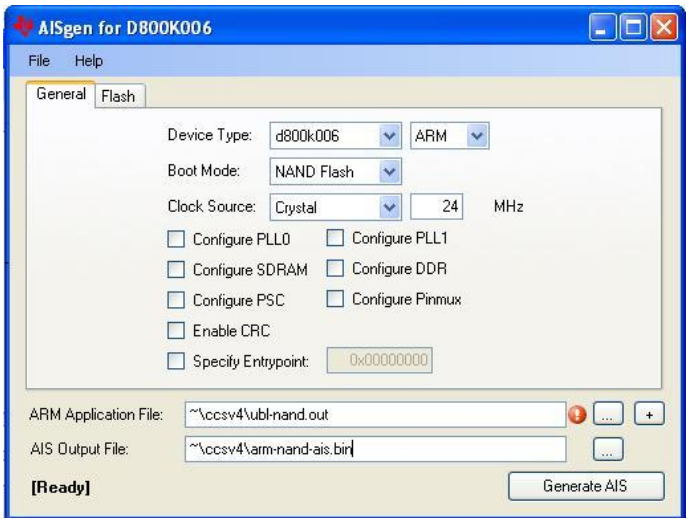
When using **CCStudiov4**:

1. Start CCStudiov4.
2. From the menus, choose File->Import. Choose **CCS->Existing CCS/CCE Eclipse Project**. Browse to `ccsv4/omap_l1x8` directory inside the extracted ARM UBL source and select it.
   - To build for SPI Flash, select the "Configuration" option as `BOOT_SPI` in Project->Properties menu.
   - To build for NOR Flash, select the "Configuration" option as `BOOT_NOR` in Project->Properties menu.
   - To build for NAND Flash, select the "Configuration" option as `BOOT_NAND` in Project->Properties menu.
   - To build for MMC/SD boot, select the "Configuration" option as `BOOT_SDMMC` in Project->Properties menu.
3. From the menus, choose **Project->Build Project**
4. When build is complete:
   - The executable ELF binary (ubl-xxx.out) file is generated in the Project Root directory `(<armubl-install-dir>\ccsv4\omapl1x8\)`
5. Proceed to convert the executable ELF binary obtained above to AIS format file suitable for flashing.

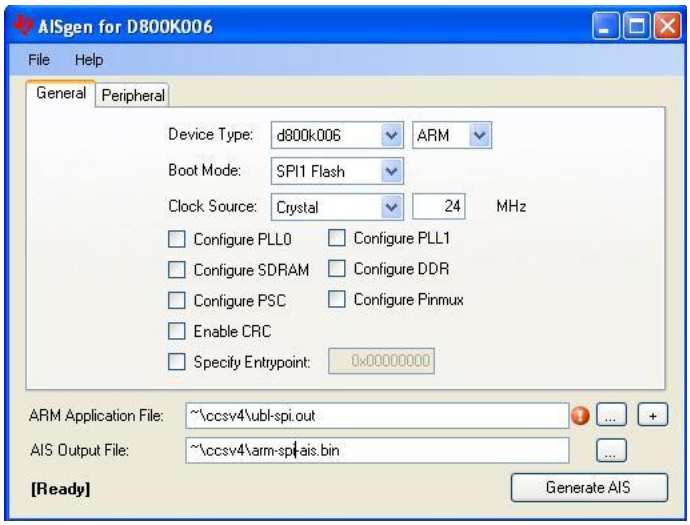
Generating ARM AIS File For OMAP-L138 (or DA850, AM18xx)

1. Obtain the AIS file format generator tool described in the Bootloader Application Report document.
2. Use the tool as shown below to obtain AIS format files ready to be flashed into NAND/SPI/NOR flash.

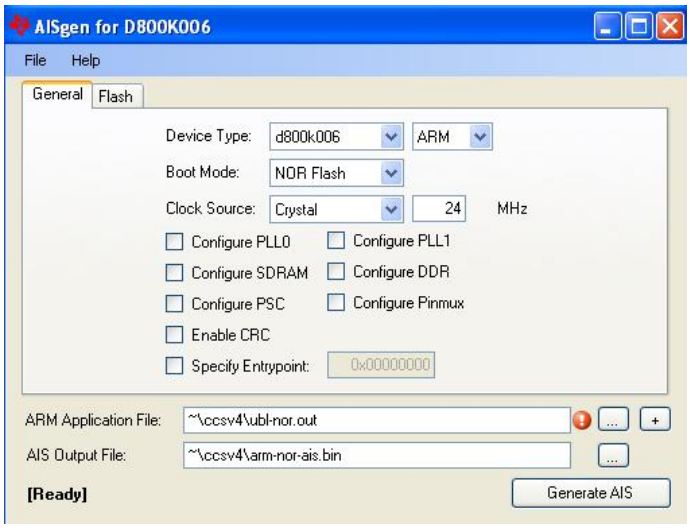
   For NAND Flash:



For SPI Flash:



For NOR Flash:



Note: Before generating the AIS image for NOR, make sure that Flash Data Width is set to 16 bit in AIS file format generator's "Flash" tab.

For OMAP-L137 (or DA830, AM17xx)

When using **CCStudiov3.3**:

1. Start CCStudio v3.3.
2. From the menus, choose **Project->Open**.
3. Browse to the extracted ARM UBL source and open the `ubl_omapl1x7.pjt` project.
   - To build for SPI Flash, select the "Config" option as `BOOT_SPI` in CCStudio window.
   - To build for NAND Flash, select the "Config" option as `BOOT_NAND` in CCStudio window.
4. From the menus, choose **Project->Build**. When the build is complete:
   - The Intel hex format binary (ubl-xxx.bin) file is generated in the Project Root directory.
   - The executable ELF binary (ubl-xxx.out) file is generated in the Project Root directory.
   - The Project Root directory is <armubl-install-dir>\ccsv3.3\
5. The files generated by the build procedure, have to be flashed.
   - In case of DSP BOOT devices like OMAP-L137 or DA830, the Intel hex format binary file is suitable for flashing.

- In case of ARM BOOT devices like AM17xx, proceed to convert the executable ELF binary obtained above to AIS format file suitable for flashing.

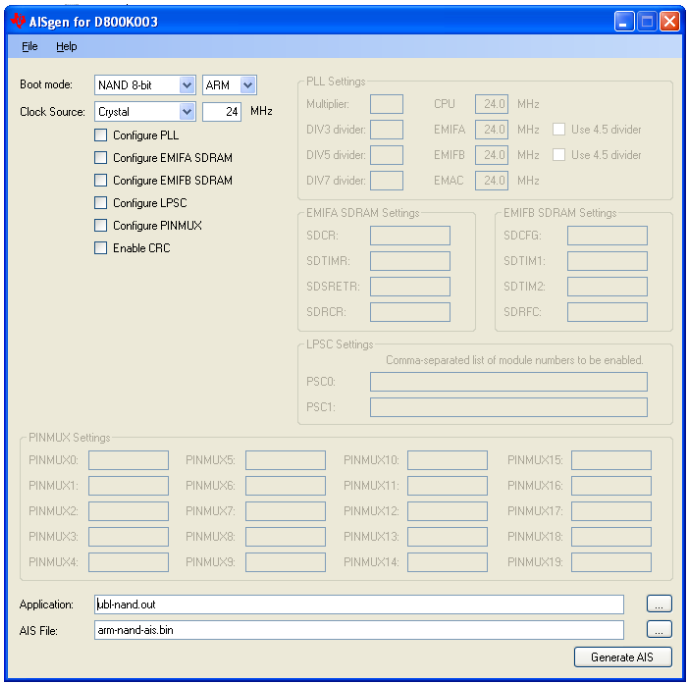
When using **CCStudiov4**:

1. Start CCStudiov4.
2. From the menus, choose File->Import. Choose **CCS->Existing CCS/CCE Eclipse Project**. Browse to `ccsv4/omap-l1x7` directory inside the extracted ARM UBL source and select it.
   - To build for SPI Flash, select the "Configuration" option as `BOOT_SPI` in **Project->Properties** menu.
   - To build for NAND Flash, select the "Configuration" option as `BOOT_NAND` in **Project->Properties** menu.
3. From the menus, choose **Project->Build Project**.
4. When build is complete
   - The Intel hex format binary (ubl-xxx.bin) file is generated in the Project Root directory.
   - The executable ELF binary (ubl-xxx.out) file is generated in the Project Root directory.
   - The Project Root directory is `<armubl-install-dir>\ccsv4\omapl1x7\`
5. The files generated by the build procedure, have to be flashed.
   - In case of DSP BOOT devices like OMAP-L137 or DA830, the Intel hex format binary file is suitable for flashing.
   - In case of ARM BOOT devices like AM17xx, proceed to convert the executable ELF binary obtained above to AIS format file suitable for flashing.

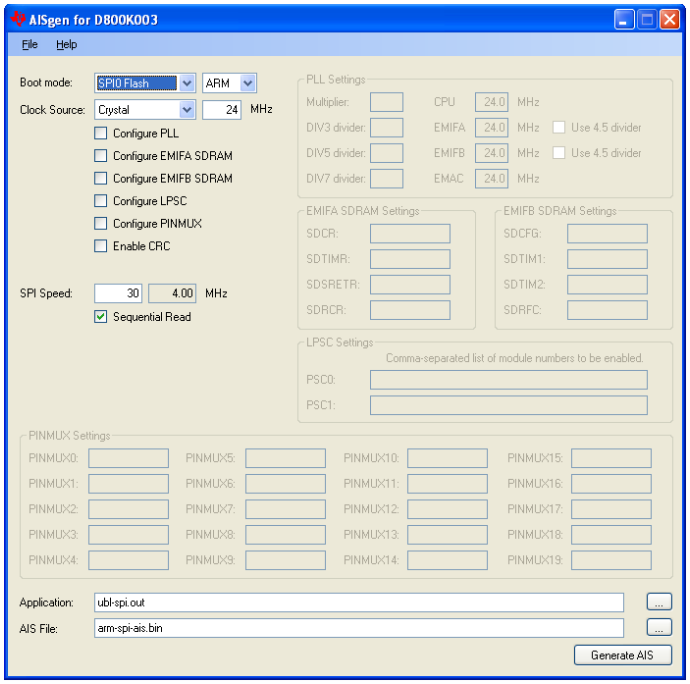
Generating ARM AIS File For AM17xx

1. Obtain the AIS file format generator tool described in the Bootloader Application Report document.
2. Use the tool as shown below to obtain AIS format files ready to be flashed into NAND/SPI flash.

   For NAND Flash:



   For SPI Flash:



Rebuilding DSP Side User Boot Loader

For the DSP boot devices OMAP-L137 and DA830, DSP UBL is required to wake up the ARM during the boot process.

When using **CCStudiov3.3**:

1. Use your Linux host to extract the DSP UBL code from the `src/boot-strap/dspubl-#.#.#.#.tar.gz` tarball from the OMAP-L138 Linux PSP package, which is located in the DaVinci-PSP-SDK-#.#.#.# directory under the main SDK installation directory. Use the `tar` command to extract the sources.
2. If the extracted files are not accessable from your Microsoft Windows host, copy all the extracted files to a location that is accessable.
3. Start CCStudio v3.3.
4. From the menus, choose **Project->Open**. Browse to the extracted DSP UBL source and open the `ubl-omapl1x7.pjt` project.
5. By default, the project builds DSP UBL for NAND flash. To build for SPI Flash, select the "Config" option as `BOOT_SPI` in CCStudio window.
6. From the menus, choose **Project->Build**. When the build is complete for SPI flash, the build places `ubl-spi.out` in the `spi` directory under the top level directory. Similarly, the build for NAND flash places the `ubl-nand.out` in the `nand` directory.

CCStudiov4 specific steps to build the DSP side User Boot Loader:

1. Extract the DSP UBL code from the `src/boot-strap/dspubl-MM.mm.pp.bb.tar.gz` file in PSP installation.
2. Start CCStudiov4. From the menus, choose File->Import. Choose **CCS->Existing CCS/CCE Eclipse Project**. Browse to the extracted DSP UBL source and select the root directory where the ccsv4 project resides. By default, the project builds DSP UBL for NAND flash. To build for SPI Flash, select the "Configuration" option as `BOOT_SPI` in **Project->Properties** menu.
3. From the menus, choose **Project->Build Project**. When build is complete for SPI flash, the build places `ubl-spi.out` in `spi` directory under the top level directory. Similarly, the build for NAND flash places the `ubl-nand.out` in `nand` directory.

Once the .out file is obtained using CCStudiov3.3 or CCStudiov4, convert to AIS File format. Obtain and install the AIS file format generator tool described in Section 5 of the Using the D800K001 Bootloader document.

- Open an MS-DOS command window and change (cd command) to the AIS Generation tool installation directory.
- Execute the following command:

  For SPI:

```
hexgen -romid D800K001 -seqread -crc 1 -spiclk 0 -appln <PATH to
ubl-spi.out> -output <dsp-spi-ais.bin>
```

  For NAND:

```
hexgen -romid D800K001 -seqread -crc 1 -spiclk 0 -appln <PATH to
ubl-nand.out> -output <dsp-nand-ais.bin>
```

The resulting output file is the DSP AIS binary.

Rebuilding the SPI Flash writer

SPI flash writer is used to flash UBL and U-Boot images to SPI flash. The flash writer also supports flashing a given image at a chosen offset.

Use the following steps to build SPI Flash writer using **CCStudio v3.3**:

1. Use your Linux host to extract the SPI flash writer code from the `src/utils/spiflash-writer-#.#.#.#.tar.gz` tarball from the OMAP-L138 Linux PSP package, which is located in the DaVinci-PSP-SDK-#.#.#.# directory under the main SDK installation directory. Use the `tar` command to extract the sources.
2. If the extracted files are not accessable from your Microsoft Windows host, copy all the extracted files to a location that is accessable.
3. Start CCStudio v3.3.
4. From the menus, choose **Project->Open**.
5. Browse to the the extracted source directory, and open the `spiflash_writer.pjt` project.
6. Select the active configuration as SPI0 if the flash writer is being built for DA830/OMAP-L137/AM17xx or as SPI1 if it is being built for DA850/OMAP-L138/AM18xx.
7. From the menus, choose **Project->Build**. The built image `spiflash-writer.out` is placed in the `ccsv3.3/Debug` subdirectory under the project directory.

Use the following steps to build SPI Flash writer using **CCStudio v4**:

1. Extract the SPI flash writer code from `src/utils/spiflash-writer-MM.mm.bb.pp.tar.gz` file in PSP installation.
2. Start CCStudio v4. From the menus, choose File->Import. Choose **CCS->Existing CCS/CCE Eclipse Project**.
3. Browse to the the extracted source directory, select the root directory where the ccsv4 project resides.
4. Select the active configuration as SPI0 if the flash writer is being built for DA830/OMAP-L137/AM17xx or as SPI1 if it is being built for DA850/OMAP-L138/AM18xx.
5. From the menus, choose **Project->Build Project**. The built image `spiflash-writer.out` is placed in the `ccsv4/Debug` under the top directory.

To flash images to SPI Flash, see Flashing images to SPI Flash.

To boot U-Boot from SPI Flash, see Booting from SPI Flash.

To boot the Linux kernel from SPI Flash using U-Boot, see Booting the Linux kernel from SPI Flash.

Rebuilding the NAND Flash writer

NAND flash writer is used to flash the ARM UBL and U-Boot images to NAND flash. The flash writer also supports flashing a given image at a chosen offset.

Use the following steps to build NAND Flash writer using **CCStudio v3.3**:

1. Use your Linux host to extract the NAND flash writer code from the `src/utils/nand-writer-#.#.#.#.tar.gz` tarball from the Linux PSP package, which is located in the DaVinci-PSP-SDK-#.#.#.# directory under the main SDK installation directory. Use the `tar` command to extract the sources.
2. If the extracted files are not accessable from your Microsoft Windows host, copy all the extracted files to a location that is accessable.
3. Start CCStudio v3.3.
4. From the menus, choose **Project->Open**.
5. Browse to the extracted source directory and open the `nand_writer.pjt` project file.
6. From the menus, choose **Project->Build**. The built image `nand-writer.out` is placed in the `ccsv3.3/Debug` subdirectory under the project directory.

Use the following steps to build NAND Flash writer using **CCStudio v4**:

- Extract the NAND flash writer code from `src/utils/nand-writer-MM.mm.bb.pp.tar.gz` file in PSP installation.
- Start CCStudio v4. From the menus, choose File->Import. Choose **CCS->Existing CCS/CCE Eclipse Project**.
- Browse to the the extracted source directory, select the root directory where the CCSv4 project resides.
- From the menus, choose **Project->Build Project**. The built image `nand-writer.out` is placed in the `ccsv4/Debug` under the top directory.

To flash images to NAND Flash, see Flashing images to NAND Flash.
To boot U-Boot from NAND Flash, see Booting from NAND Flash.
To boot the Linux kernel from NAND Flash using U-Boot, see Booting the Linux kernel from NAND Flash.

Rebuilding NOR Flash writer

Note: NOR flash writer is supported only on the EVMs for OMAP-L138 (or DA850, AM18xx).

NOR flash writer is used to flash UBL and U-Boot images to NOR flash. The flash writer also supports flashing a given image at a chosen offset.

Use the following steps to build NOR Flash writer using **CCStudio v3.3**:

1. Extract the NOR flash writer code from `src/utils/norflash-writer-#.#.#.#.tar.gz` directory in PSP installation.
2. Start CCStudio v3.3. From the menus, choose **Project->Open**
3. Browse to the extracted source directory and open the `ccsv3.3/norflash_writer.pjt` project file.
4. From the menus, choose **Project->Build**. The built image `norflash-writer.out` is placed in the `ccsv3.3/Debug` directory under the top level source directory.

Use the following steps to build NOR Flash writer using **CCStudio v4**:

1. Extract the NOR flash writer code from `src/utils/norflash-writer-#.#.#.#.tar.gz` file in PSP installation.
2. Start CCStudio v4. From the menus, choose **File->Import**. Choose **CCS->Existing CCS/CCE Eclipse Project**.
3. Browse to the the extracted source directory, select the root directory where the CCSv4 project resides.
4. From the menus, choose **Project->Build Project**. The built image `norflash-writer.out` is placed in the `ccsv4/Debug` under the top directory.

What's next?

Please continue on to the **Building the SDK** section of the OMAP-L1 Getting Started Guide.

References

[1] http://codesourcery.com

[2] http://processors.wiki.ti.com/index.php/
Creating_a_Root_File_System_for_Linux_on_OMAP35x#Configure_the_Linux_Kernel_to_Support_File_Systems

Article Sources and Contributors

Community Linux PSP for DA8x/OMAP-L1/AM1x *Source*: http://processors.wiki.ti.com/index.php?oldid=42051 *Contributors*: Kevinsc, ManishKhare, SekharNori, Sudhakar.raj

Getting Started Guide for OMAP-L1 *Source*: http://processors.wiki.ti.com/index.php?oldid=42247 *Contributors*: Arnier, BrianBarrera, Dfriedland, FrankW, ManishKhare, Mariana, SekharNori, Stsongas, Sudhakar.raj, Xyvonned

GSG: Installing the Software for OMAP-L1 *Source*: http://processors.wiki.ti.com/index.php?oldid=35006 *Contributors*: Arnier, BrianBarrera, D-allred, Gparodi, Hezhengting, Mariana, SekharNori, Sudhakar.raj, Xyvonned

GSG: Setting up OMAP-L1/AM1x Target File System *Source*: http://processors.wiki.ti.com/index.php?oldid=33758 *Contributors*: Alokprasad, Arnier, DavidMeixner, ManishKhare, Mariana, SekharNori, Sudhakar.raj, Xyvonned

OMAP-L1 Linux Drivers Usage *Source*: http://processors.wiki.ti.com/index.php?oldid=26076 *Contributors*: Chaithrika, Hezhengting, SekharNori, Sudhakar.raj

OMAPL1: Changing the Operating Point *Source*: http://processors.wiki.ti.com/index.php?oldid=27720 *Contributors*: DanRinkes, SekharNori

GSG: DA8x/OMAP-L1/AM1x DVEVM Additional Procedures *Source*: http://processors.wiki.ti.com/index.php?oldid=42255 *Contributors*: Binary0101, D-allred, Dswag89, Jeff Cobb, Mariana, SekharNori, Sudhakar.raj, Xyvonned

GSG: Building Software Components for OMAP-L1/AM1x *Source*: http://processors.wiki.ti.com/index.php?oldid=42087 *Contributors*: Alexander.stohr, Arnier, Chaithrika, DanRinkes, Dswag89, FarMcKon, Loc, ManishKhare, Mariana, SekharNori, Sudhakar.raj, Swami, Xyvonned

Image Sources, Licenses and Contributors

Image:TIBanner.png *Source*: http://processors.wiki.ti.com/index.php?title=File:TIBanner.png *License*: unknown *Contributors*: Nsnehaprabha
File:OMAP-L138_inifiles.zip *Source*: http://processors.wiki.ti.com/index.php?title=File:OMAP-L138_inifiles.zip *License*: unknown *Contributors*: D-allred
Image:NandAis.jpg *Source*: http://processors.wiki.ti.com/index.php?title=File:NandAis.jpg *License*: unknown *Contributors*: Sudhakar.raj
Image:SpiAis.jpg *Source*: http://processors.wiki.ti.com/index.php?title=File:SpiAis.jpg *License*: unknown *Contributors*: Sudhakar.raj
Image:NorAis.jpg *Source*: http://processors.wiki.ti.com/index.php?title=File:NorAis.jpg *License*: unknown *Contributors*: Sudhakar.raj
Image:NandAisOMAPL137.png *Source*: http://processors.wiki.ti.com/index.php?title=File:NandAisOMAPL137.png *License*: unknown *Contributors*: Sudhakar.raj
Image:SpiAisOMAPL137.png *Source*: http://processors.wiki.ti.com/index.php?title=File:SpiAisOMAPL137.png *License*: unknown *Contributors*: Sudhakar.raj

License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.
BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. TO THE EXTENT THIS LICENSE MAY BE CONSIDERED TO BE A CONTRACT, THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

License

1. Definitions

1. "**Adaptation**" means a work based upon the Work, or upon the Work and other pre-existing works, such as a translation, adaptation, derivative work, arrangement of music or other alterations of a literary or artistic work, or phonogram or performance and includes cinematographic adaptations or any other form in which the Work may be recast, transformed, or adapted including in any form recognizably derived from the original, except that a work that constitutes a Collection will not be considered an Adaptation for the purpose of this License. For the avoidance of doubt, where the Work is a musical work, performance or phonogram, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered an Adaptation for the purpose of this License.
2. "**Collection**" means a collection of literary or artistic works, such as encyclopedias and anthologies, or performances, phonograms or broadcasts, or other works or subject matter other than works listed in Section 1(f) below, which, by reason of the selection and arrangement of their contents, constitute intellectual creations, in which the Work is included in its entirety in unmodified form along with one or more other contributions, each constituting separate and independent works in themselves, which together are assembled into a collective whole. A work that constitutes a Collection will not be considered an Adaptation (as defined below) for the purposes of this License.
3. "**Creative Commons Compatible License**" means a license that is listed at http://creativecommons.org/compatiblelicenses that has been approved by Creative Commons as being essentially equivalent to this License, including, at a minimum, because that license: (i) contains terms that have the same purpose, meaning and effect as the License Elements of this License; and, (ii) explicitly permits the relicensing of adaptations of works made available under that license under this License or a Creative Commons jurisdiction license with the same License Elements as this License.
4. "**Distribute**" means to make available to the public the original and copies of the Work or Adaptation, as appropriate, through sale or other transfer of ownership.
5. "**License Elements**" means the following high-level license attributes as selected by Licensor and indicated in the title of this License: Attribution, ShareAlike.
6. "**Licensor**" means the individual, individuals, entity or entities that offer(s) the Work under the terms of this License.
7. "**Original Author**" means, in the case of a literary or artistic work, the individual, individuals, entity or entities who created the Work or if no individual or entity can be identified, the publisher; and in addition (i) in the case of a performance the actors, singers, musicians, dancers, and other persons who act, sing, deliver, declaim, play in, interpret or otherwise perform literary or artistic works or expressions of folklore; (ii) in the case of a phonogram the producer being the person or legal entity who first fixes the sounds of a performance or other sounds; and, (iii) in the case of broadcasts, the organization that transmits the broadcast.
8. "**Work**" means the literary and/or artistic work offered under the terms of this License including without limitation any production in the literary, scientific and artistic domain, whatever may be the mode or form of its expression including digital form, such as a book, pamphlet and other writing; a lecture, address, sermon or other work of the same nature; a dramatic or dramatico-musical work; a choreographic work or entertainment in dumb show; a musical composition with or without words; a cinematographic work to which are assimilated works expressed by a process analogous to cinematography; a work of drawing, painting, architecture, sculpture, engraving or lithography; a photographic work to which are assimilated works expressed by a process analogous to photography; a work of applied art; an illustration, map, plan, sketch or three-dimensional work relative to geography, topography, architecture or science; a performance; a broadcast; a phonogram; a compilation of data to the extent it is protected as a copyrightable work; or a work performed by a variety or circus performer to the extent it is not otherwise considered a literary or artistic work.
9. "**You**" means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.
10. "**Publicly Perform**" means to perform public recitations of the Work and to communicate to the public those public recitations, by any means or process, including by wire or wireless means or public digital performances; to make available to the public Works in such a way that members of the public may access these Works from a place and at a place individually chosen by them; to perform the Work to the public by any means or process and the communication to the public of the performances of the Work, including by public digital performance; to broadcast and rebroadcast the Work by any means including signs, sounds or images.
11. "**Reproduce**" means to make copies of the Work by any means including without limitation by sound or visual recordings and the right of fixation and reproducing fixations of the Work, including storage of a protected performance or phonogram in digital form or other electronic medium.

2. Fair Dealing Rights

Nothing in this License is intended to reduce, limit, or restrict any uses free from copyright or rights arising from limitations or exceptions that are provided for in connection with the copyright protection under copyright law or other applicable laws.

3. License Grant

Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

1. to Reproduce the Work, to incorporate the Work into one or more Collections, and to Reproduce the Work as incorporated in the Collections;
2. to create and Reproduce Adaptations provided that any such Adaptation, including any translation in any medium, takes reasonable steps to clearly label, demarcate or otherwise identify that changes were made to the original Work. For example, a translation could be marked "The original work was translated from English to Spanish," or a modification could indicate "The original work has been modified.";
3. to Distribute and Publicly Perform the Work including as incorporated in Collections; and,
4. to Distribute and Publicly Perform Adaptations.
5. For the avoidance of doubt:

    1. **Non-waivable Compulsory License Schemes.** In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme cannot be waived, the Licensor reserves the exclusive right to collect such royalties for any exercise by You of the rights granted under this License;
    2. **Waivable Compulsory License Schemes.** In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme can be waived, the Licensor waives the exclusive right to collect such royalties for any exercise by You of the rights granted under this License; and,
    3. **Voluntary License Schemes.** The Licensor waives the right to collect royalties, whether individually or, in the event that the Licensor is a member of a collecting society that administers voluntary licensing schemes, via that society, from any exercise by You of the rights granted under this License.

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats. Subject to Section 8(f), all rights not expressly granted by Licensor are hereby reserved.

4. Restrictions

The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

1. You may Distribute or Publicly Perform the Work only under the terms of this License. You must include a copy of, or the Uniform Resource Identifier (URI) for, this License with every copy of the Work You Distribute or Publicly Perform. You may not offer or impose any terms on the Work that restrict the terms of this License or the ability of the recipient of the Work to exercise the rights granted to that recipient under the terms of the License. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties with every copy of the Work You Distribute or Publicly Perform. When You Distribute or Publicly Perform the Work, You may not impose any effective technological measures on the Work that restrict the ability of a recipient of the Work from You to exercise the rights granted to that recipient under the terms of the License. This Section 4(a) applies to the Work as incorporated in a Collection, but this does not require the Collection apart from the Work itself to be made subject to the terms of this License. If You create a Collection, upon notice from any Licensor You must, to the extent practicable, remove from the Collection any credit as required by Section 4(c), as requested. If You create an Adaptation, upon notice from any Licensor You must, to the extent practicable, remove from the Adaptation any credit as required by Section 4(c), as requested.
2. You may Distribute or Publicly Perform an Adaptation only under the terms of: (i) this License; (ii) a later version of this License with the same License Elements as this License; (iii) a Creative Commons jurisdiction license (either this or a later license version) that contains the same License Elements as this License (e.g., Attribution-ShareAlike 3.0 US)); (iv) a Creative Commons Compatible License. If you license the Adaptation under one of the licenses mentioned in (iv), you must comply with the terms of that license. If you license the Adaptation under the terms of any of the licenses mentioned in (i), (ii) or (iii) (the "Applicable License"), you must comply with the terms of the Applicable License generally and the following provisions: (I) You must include a copy of, or the URI for, the Applicable License with every copy of each Adaptation You Distribute or Publicly Perform; (II) You may not offer or impose any terms on the Adaptation that restrict the terms of the Applicable License or the ability of the recipient of the Adaptation to exercise the rights granted to that recipient under the terms of the Applicable License; (III) You must keep intact all notices that refer to the Applicable License and to the disclaimer of warranties with every copy of the Work as included in the Adaptation You Distribute or Publicly Perform; (IV) when You Distribute or Publicly Perform the Adaptation, You may not impose any effective technological measures on the Adaptation that restrict the ability of a recipient of the Adaptation from You to exercise the rights granted to that recipient under the terms of the Applicable License. This Section 4(b) applies to the Adaptation as incorporated in a Collection, but this does not require the Collection apart from the Adaptation itself to be made subject to the terms of the Applicable License.
3. If You Distribute, or Publicly Perform the Work or any Adaptations or Collections, You must, unless a request has been made pursuant to Section 4(a), keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or if the Original Author and/or Licensor designate another party or parties (e.g., a sponsor institute, publishing entity, journal) for attribution ("Attribution Parties") in Licensor's copyright notice, terms of service or by other reasonable means, the name of such party or parties; (ii) the title of the Work if supplied; (iii) to the extent reasonably practicable, the URI, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work; and (iv) , consistent with Ssection 3(b), in the case of an Adaptation, a credit identifying the use of the Work in the Adaptation (e.g., "French translation of the Work by Original Author," or "Screenplay based on original Work by Original Author"). The credit required by this Section 4(c) may be implemented in any reasonable manner; provided, however, that in the case of a Adaptation or Collection, at a minimum such credit will appear, if a credit for all contributing authors of the Adaptation or Collection appears, then as part of these credits and in a manner at least as prominent as the credits for the other contributing authors. For the avoidance of doubt, You may only use the credit required by this Section for the purpose of attribution in the manner set out above and, by exercising Your rights under this License, You may not implicitly or explicitly assert or imply any connection with, sponsorship or endorsement by the Original Author, Licensor and/or Attribution Parties, as appropriate, of You or Your use of the Work, without the separate, express prior written permission of the Original Author, Licensor and/or Attribution Parties.
4. Except as otherwise agreed in writing by the Licensor or as may be otherwise permitted by applicable law, if You Reproduce, Distribute or Publicly Perform the Work either by itself or as part of any Adaptations or Collections, You must not distort, mutilate, modify or take other derogatory action in relation to the Work which would be prejudicial to the Original Author's honor or reputation. Licensor agrees that in those jurisdictions (e.g. Japan), in which any exercise of the right granted in Section 3(b) of this License (the right to make Adaptations) would be deemed to be a distortion, mutilation, modification or other derogatory action prejudicial to the Original Author's honor and reputation, the Licensor will waive or not assert, as appropriate, this Section, to the fullest extent permitted by the applicable national law, to enable You to reasonably exercise Your right under Section 3(b) of this License (right to make Adaptations) but not otherwise.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability

EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

1. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Adaptations or Collections from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.
2. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

1. Each time You Distribute or Publicly Perform the Work or a Collection, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.
2. Each time You Distribute or Publicly Perform an Adaptation, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.
3. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.
4. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.
5. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.
6. The rights granted under, and the subject matter referenced, in this License were drafted utilizing the terminology of the Berne Convention for the Protection of Literary and Artistic Works (as amended on September 28, 1979), the Rome Convention of 1961, the WIPO Copyright Treaty of 1996, the WIPO Performances and Phonograms Treaty of 1996 and the Universal Copyright Convention (as revised on July 24, 1971). These rights and subject matter take effect in the relevant jurisdiction in which the License terms are sought to be enforced according to the corresponding provisions of the implementation of those treaty provisions in the applicable national law. If the standard suite of rights granted under applicable copyright law includes additional rights not granted under this License, such additional rights are deemed to be included in the License; this License is not intended to restrict the license of any rights under applicable law.