Contents

- 1 Document License
- 2 PSP Overview
  - 2.1 Downloading the Release
  - 2.2 Host platform Requirements
  - 2.3 Host Software Requirements
  - 2.4 Target Hardware Requirements
  - 2.5 Running the Installation
- 3 Getting Started Quickly
- 4 Running PSP Components
  - 4.1 Booting U-Boot
    - 4.1.1 SPI Flash
      - 4.1.1.1 Flashing Images
      - 4.1.1.2 Booting
    - 4.1.2 NAND Flash
      - 4.1.2.1 Flashing Images
      - 4.1.2.2 Booting
    - 4.1.3 NOR Flash
      - 4.1.3.1 Flashing Images
      - 4.1.3.2 Booting
  - 4.2 Booting Linux kernel using U-Boot
    - 4.2.1 Booting from SDRAM
    - 4.2.2 Booting from SPI Flash
    - 4.2.3 Booting from NAND Flash
    - 4.2.4 Booting from NOR Flash
- 5 Building PSP Components
  - 5.1 Building U-Boot
  - 5.2 Building a New Linux Kernel
  - 5.3 Configuring Linux Kernel
  - 5.4 Using Linux Kernel Drivers
    - 5.4.1 USB 2.0
    - 5.4.2 USB 1.1
    - 5.4.3 Audio
    - 5.4.4 Graphical LCD
    - 5.4.5 Character LCD
    - 5.4.6 NAND
    - 5.4.7 NOR
    - 5.4.8 SPI
    - 5.4.9 MMC/SD
    - 5.4.10 RTC
    - 5.4.11 SATA
    - 5.4.12 McBSP
    - 5.4.13 Power Management
    - 5.4.14 Ethernet (Using RMII PHY on UI board)
  - 5.5 Loading Linux Kernel Modules
    - 5.5.1 Loading USB 2.0
  - 5.6 Building the User Boot Loader
  - 5.7 Building SPI Flash writer
  - 5.8 Building NAND Flash writer
  - 5.9 Building NOR Flash writer
- 6 Additional topics
  - 6.1 DSP wakeup in U-Boot

- 6.2 Restoring factory default U-Boot environment variables
    - 6.2.1 SPI flash
    - 6.2.2 NAND Flash
    - 6.2.3 NOR Flash
- 6.3 Linux Functional Test Bench
- 6.4 Using extended memory available on EVM board

Document License

This work is licensed under the Creative Commons Attribution-Share Alike 3.0 United States License. To view a copy of this license, visit http://creativecommons.org/licenses/by-sa/3.0/us/ or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

PSP Overview

This PSP package supports the EVM board for Texas Instruments OMAP-L138 SoC. This PSP package includes the following components:

- DaVinci Linux kernel version 2.6.31-rc7 ported for OMAP-L138 EVM. More information on the kernel this release is based on is present in the release notes for the release. The pre-built kernel binary included with the package is compiled with Sourcery G++ Lite 2009q1-203 for ARM GNU/Linux from CodeSourcery
- Arago file system version 2009.09. More information on Arago file system available at the Arago website
- U-Boot version 2009.01 ported for OMAP-L138 EVM. More information on the U-Boot this release is based on is present in the release notes for the release. The pre-built U-Boot binary included with this release is built with Sourcery G++ Lite 2009q1-203 for ARM GNU/Linux from CodeSourcery
- User Boot Loader (UBL). This is the primary boot software which copies U-Boot to SDRAM and starts it. This software requires Code Composer Studio (CCStudio) 3.3/4.0 for building.
- SPI and NAND flash writers. These are CCStudio v3.3/v4.0 based utilities.

Downloading the Release

The latest release is available here

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 4.0. 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.

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 4.0.0.16000 (if need to re-build or re-flash UBL image or flash writer)
- CodeSourcery tool chain for ARM
- Serial console terminal application
- TFTP and NFS servers.

Target Hardware Requirements

The target board required is OMAP-L138 EVM. The EVM comes with on-board emulation which works with DSP only. So, accessing ARM through emulation requires an external emulator. The EVM ships with other peripheral hardware like power supply, USB, ethernet and serial cables.

Running the Installation

As outlined in Host Platform Requirements, the PSP package consists of components that need a Microsoft Windows host as well as components that need a Linux host. The recommended installation procedure is to install the PSP package on a Linux host and then copy portions used with 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.

To install the PSP package, extract the release package with the following command (replace `MM.mm.pp.bb` with the release version number of your release package):

```
tar -xvfz  DaVinci-PSP-SDK-MM.mm.pp.bb.tgz
```

The following files and directories are extracted:

```
|-- Arago-FS-Software-manifest.html
|-- Software-manifest.html
|-- docs
|   |-- Building-RootFs-Arago.html
|   |-- DataSheet-MM.mm.pp.bb.pdf
|   |-- GPLv2.pdf
|   |-- ReleaseNotes-MM.mm.pp.bb.pdf
|   `-- UserGuide-MM.mm.pp.bb.pdf
|-- host-tools
|   |-- linux
|   |-- src
|   `-- windows
|-- images
|   |-- boot-strap
|   |   |-- arm-nand-ais.bin
|   |   |-- arm-nor-ais.bin
|   |   `-- arm-spi-ais.bin
|   |-- examples
|   |   |-- edma_test.ko
|   |   |-- gpio_test.ko
|   |   `-- mcbsp_test.ko
|   |-- fs
|   |   |-- nfs-base.tar.gz
|   |   |-- nfs.tar.gz
|   |   |-- ramdisk-base.gz
|   |   |-- ramdisk.gz
|   |   |-- rootfs-base.jffs2
|   |   `-- rootfs.jffs2
|   |-- kernel
|   |   |-- modules\
|   |   `-- uImage
|   |-- u-boot
|   |   `-- u-boot.bin
|   `-- utils
|       |-- nand-writer-ccsv3.out
|       |-- norflash-writer-ccsv3.out
|       `-- spiflash-writer-ccsv3.out
|-- scripts
|-- src
|   |-- boot-strap
|   |   `-- armubl-MM.mm.pp.bb.tar.gz
|   |-- examples
|   |   `-- examples.tar.gz
|   |-- kernel
|   |   |-- ChangeLog-MM.mm.pp.bb
|   |   |-- ShortLog
|   |   |-- Unified-patch-MM.mm.pp.bb.gz
|   |   |-- diffstat-MM.mm.pp.bb
|   |   |-- kernel-patches-MM.mm.pp.bb.tar.gz
|   |   `-- linux-MM.mm.pp.bb.tar.gz
|   |-- u-boot
|   |   |-- ChangeLog-MM.mm.pp.bb
|   |   |-- ShortLog
|   |   |-- Unified-patch-MM.mm.pp.bb.gz
|   |   |-- diffstat-MM.mm.pp.bb
|   |   |-- u-boot-MM.mm.pp.bb.tar.gz
|   |   `-- u-boot-patches-MM.mm.pp.bb.tar.gz
|   `-- utils
|       |-- nand-writer-MM.mm.pp.bb.tar.gz
|       |-- norflash-writer-MM.mm.pp.bb.tar.gz
|       `-- spiflash-writer-MM.mm.pp.bb.tar.gz
`-- test-suite
    `-- REL_LFTB_XX.XX.XX.XX.tar.gz
```

Getting Started Quickly

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.

The following list is an overview of the steps you should follow to set up and begin using the OMAP-L138 Evaluation Module (EVM) with the Linux Platform Support Product (PSP) package.

- Understand how to use the OMAP-L138 EVM hardware by reading the EVM technical reference that accompanies the EVM shipment.
- Set up the EVM to boot from the SPI flash on the OMAP-L138 EVM. (Switch S7 set to all off)
- Install the serial terminal software described in Host Platform Requirements and connect the serial cable provided with the EVM shipment to the COM port of the host system.
- Set up the serial console software on the PC to connect to that serial port with the following characteristics:
    - Baud: 115200
    - Data Bits: 8
    - Parity: None
    - Stop Bits: 1
    - Flow Control: None
- Power on the board using the power supply that accompanies the board.
- If your EVM is pre-flashed with U-Boot and Linux kernel, when you power on the EVM, you should see U-Boot and Linux boot sequence messages on the serial console.

Note: If you do not see U-Boot and Linux boot sequence messages, the EVM is not pre-flashed with boot software. You need to flash the EVM with boot software. The pre-built binaries required for booting the EVM are provided in `images` directory under PSP installation directory. Skip to "Running PSP Components" to learn how to use them.

- At the Linux prompt, you can use existing Linux commands to test various features and use examples from the `examples` folder under `images` directory of the PSP installation.
- While U-Boot is booting, you can press any key to interrupt the automatic boot of the Linux kernel. This will leave you in U-Boot, where you can type commands in the U-Boot command shell. On this page, commands to be typed in the U-Boot shell are indicated by the `U-Boot>` prompt.
- In order to create your own applications running on Linux on the OMAP-L138 EVM 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.
- If you need to re-build any of the software provided with the PSP installation, skip to "Building PSP Components". Note that building some PSP components requires that you have CCStudio v3.3 running on a Microsoft Windows host. Building other components requires that you have the CodeSourcery tools running on a Linux host.

Running PSP Components

Pre-built binaries for UBL, U-Boot, Linux kernel, and flash writers are provided in the `images` directory of the PSP installation.

On the OMAP-L138 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.

Booting U-Boot

SPI Flash

Flashing Images

Follow these steps to boot from SPI Flash:

- Obtain the latest ARM GEL file from LogicPD. Run the CCStudio Setup tool and ensure that the ARM GEL file is correctly specified.
- Start CCStudio and connect to the ARM.
- Load the SPI flasher tool on to the ARM. You can either use the pre-built binary shipped in the `images/utils` directory of the PSP installation (`spiflash-writer-ccsv3.out`), or you can build your own by following the steps in section on Building SPI Flash writer.
- 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, 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/` 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. A pre-built image is located in the `images/u-boot/` directory of the PSP installation.
- Once the SPI flash has been written with the all the required files, disconnect CCStudio and power off the EVM.

Booting

In order to boot from SPI flash, which has been written with the boot images, follow these steps:

- Set the SW7 switch on the base board as follows. (X indicates the setting is 'don't care')

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

- 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 Getting Started Quickly.
- Power up the board again. The board should boot from SPI flash, and the U-Boot prompt should appear on the serial terminal running on the host.

NAND Flash

Flashing Images

Follow these steps to flash boot images to NAND Flash:

- Obtain the latest ARM GEL file from LogicPD. Run the CCStudio Setup tool and ensure that the ARM GEL file is correctly specified.
- Start CCStudio and connect to the ARM.
- From the toolbar, execute "GEL-->Full EVM-->EMIFA_NAND_PINMUX".
- Load the NAND flasher tool on to the ARM. You can either use the pre-built binary shipped in the `images/utils` directory of the PSP installation (`nand-writer-ccsv3.out`), or you can build your own by following the steps in section on Building NAND Flash writer.
- 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, 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/` 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 NAND.
- Once the NAND flash has been written with the all the required files, disconnect CCStudio and power off the EVM.

Booting

In order to boot from NAND flash, which has been written with the boot images, follow these steps:

- Set the SW7 switch on the base board as follows. (X indicates the setting is 'don't care')

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

- 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 Getting Started Quickly.
- Power up the board again. The board should boot from NAND flash, and the U-Boot prompt should appear on the serial terminal running on the host.

NOR Flash

Flashing Images

Follow these steps to boot from NOR Flash:

- Obtain the latest ARM GEL file from LogicPD. Run the CCStudio Setup tool and ensure that the ARM GEL file is correctly specified.
- Start CCStudio and connect to the ARM.
- From the toolbar, execute "GEL-->Full EVM-->EMIFA_NOR_PINMUX".
- Load the NOR flasher tool on to the ARM. You can either use the pre-built binary shipped in the `images/utils` directory of the PSP installation (`norflash-writer-ccsv3.out`), or you can build your own by following the steps in section on Building NOR Flash writer.
- 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, 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/` 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.
- Once the NOR flash has been written with the all the required files, disconnect CCStudio and power off the EVM.

Booting

In order to boot from NOR flash, which has been written with the boot images, follow these steps:

- Set the SW7 switch on the base board as follows. (X indicates the setting is 'don't care')

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

- 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 Getting Started Quickly.
- Power up the board again. The board should boot from NOR flash, and the U-Boot prompt should appear on the serial terminal running on the host.

Booting 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. A pre-built file system images can be found in your PSP tools installation at `images/fs`.

To boot the Linux kernel, use the U-Boot `tftp` command to download the kernel uImage to SDRAM. You can then choose to directly boot newly downloaded kernel or write it to non-volatile memory using U-Boot commands and then copy kernel to SDRAM from this memory for subsequent boots.

To use TFTP download, you must first setup the DUT IP parameters. The easiest way to do this is to use the DHCP server on you network.

- Use DHCP to setup EVM IP address.

```
U-Boot> setenv autoload no
U-Boot> dhcp
```

- Set IP address of server (this may also be obtained from DHCP)

```
U-Boot> setenv serverip <ip addr of server>
```

- Set the name of image to be downloaded (this may also be obtained from DHCP)

```
U-Boot> setenv bootfile <Linux kernel image file name>
```

You can get help on U-Boot commands by using the U-Boot "help" command. Typing `help` at the U-Boot command prompt gives a list of commands supported. Typing `help` followed by a command name gives help regarding that particular command. Help on U-Boot commands is also available at http://www.denx.de/wiki/view/DULG/UBoot.

Booting from SDRAM

Download kernel and ramdisk image from ethernet using TFTP into SDRAM. Then boot the kernel image from SDRAM. To achieve this use the following settings for U-Boot environment variables `bootargs` and `bootcmd` and reboot the DUT.

```
setenv bootargs mem=32M console=ttyS2,115200n8 root=/dev/ram0 rw initrd=0xc1180000,4M ip=dhcp
setenv bootcmd 'tftp 0xc0700000 uImage; tftp 0xc1180000 ramdisk.gz; bootm 0xc0700000'
```

Booting from SPI Flash

The SPI flash on the EVM board is 8 MBytes in size. Depending on the size of the kernel and filesystem, you may not be able to fit both of them on the SPI flash. If both of them do not fit, we recommend that you keep the kernel image on SPI flash and use another medium to the store the filesystem.

To enable SPI flash support in U-Boot, the U-Boot must be compiled with the `CONFIG_USE_SPIFLASH` macro defined in the `include/configs/da850_evm.h` file. For information on how to build U-Boot, refer to Building U-Boot. The default U-Boot image provided in the `images/u-boot` directory of the PSP installation has SPI flash support enabled.

In the example below, both kernel and filesystem reside on the SPI flash. While booting, U-Boot copies both kernel and filesystem from SPI flash to mDDR and boots the kernel which later mounts the filesystem as a ramdisk.

To boot from SPI flash follow these steps:

- Select the serial flash device

```
sf probe 0
```

- Download `uImage` and copy it to SPI flash partition

```
tftp 0xc0700000 uImage
sf erase 0x80000 0x200000
sf write 0xc0700000 0x80000 0x200000
```

- Download `ramdisk-base.gz` and copy it to SPI flash partition

```
tftp 0xc1180000 ramdisk-base.gz
sf erase 0x280000 0x300000
sf write 0xc1180000 0x280000 0x300000
```

- Setup the `bootargs` and `bootcmd` environment variables to boot from SPI flash

```
setenv bootargs mem=32M console=ttyS2,115200n8 root=/dev/ram0 rw initrd=0xc1180000,4M ip=dhcp
setenv bootcmd 'sf probe 0;sf read 0xc0700000 0x80000 0x200000;sf read 0xc1180000 0x280000 0x300000;bootm 0xc0700000'
```

Booting from NAND Flash

To enable NAND flash support in U-Boot, it must be compiled with `CONFIG_SYS_USE_NAND` macro must be defined in the `include/configs/da850_evm.h` file.

- Download uImage and copy it to NAND partition

```
tftp 0xc0700000 uImage
nand erase 0x200000 0x200000
nand write.e 0xc0700000 0x200000 0x200000
```

- Download ramdisk, and copy it to NAND flash

```
tftp 0xc1180000 ramdisk-base.gz
nand erase 0x400000 0x300000
nand write.e 0xc1180000 0x400000 0x300000
```

- Setup the bootargs and bootcmd environment variables to boot from NAND flash

```
setenv bootcmd 'nand read.e 0xc1180000 0x400000 0x300000; nboot.e 0xc0700000 0 0x200000; bootm'
setenv bootargs mem=32M console=ttyS2,115200n8 root=/dev/ram0 rw initrd=0xc1180000,4M ip=dhcp
```

Booting from NOR Flash

The NOR flash on the EVM board is 8 MBytes in size. Depending on the size of the kernel and filesystem, you may not be able to fit both of them on the NOR flash. If both of them do not fit, we recommend that you keep the kernel image on NOR flash and use another medium to the store the filesystem.

To enable NOR flash support in U-Boot, the U-Boot must be compiled with the `CONFIG_SYS_USE_NOR` macro defined in the `include/configs/da850_evm.h` file. For information on how to build U-Boot, refer to Building U-Boot.

In the example below, both kernel and filesystem reside on the NOR flash. While booting, U-Boot copies both kernel and filesystem from NOR flash to mDDR and boots the kernel which later mounts the filesystem as a ramdisk.

To boot from NOR flash follow these steps:

- Download `uImage` and copy it to NOR flash partition

```
tftp 0xc0700000 uImage
protect off 60080000 +200000
erase 60080000 +200000
cp.b c0700000 60080000 200000
```

- Download `ramdisk-base.gz` and copy it to NOR flash partition

```
tftp 0xc1180000 ramdisk-base.gz
protect off 60280000 +300000
erase 60280000 +300000
cp.b c1180000 60280000 300000
```

- Setup the `bootargs` and `bootcmd` environment variables to boot from SPI flash

```
setenv bootargs mem=32M console=ttyS2,115200n8 root=/dev/ram0 rw initrd=0xc1180000,4M ip=dhcp
setenv bootcmd 'cp.b 60080000 c0700000 200000;cp.b 60280000 c1180000 300000;bootm 0xc0700000'
```

Building PSP Components

Building U-Boot

Follow these steps to rebuild U-Boot:

- Building U-Boot requires that the CodeSourcery tools have been installed on your Linux host. For documentation on how to install the CodeSourcery tools, please visit the CodeSourcery website
- If you have not already extracted the source files for building U-Boot, use your Linux host to extract source files from the `src/u-boot/u-boot-MM.mm.pp.bb.tar.gz` tarball, which you obtained when you installed the PSP package in Running the Installation. Use the tar command to extract the sources.

Note: Patches from the uboot-patches-MM.mm.pp.bb.tar.gz file have already been applied on u-boot. This is the list of patches which have been developed on top of the base U-Boot version.

- Go to the u-boot directory created when you extracted the files.
- Run the following commands on your Linux host to build U-Boot. 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.

```
make distclean CROSS_COMPILE=arm-none-linux-gnueabi-
make da850_omapl138_evm_config CROSS_COMPILE=arm-none-linux-gnueabi-
make all CROSS_COMPILE=arm-none-linux-gnueabi-
```

The compiled u-boot.bin file will be created in the same directory. The U-Boot build options are specified in the include file `include/configs/da850_evm.h`, which is in the U-Boot source tree. To change build options, edit this file and re-build the U-Boot binary. You can modify the following build options:

- Choice of flash supported:
  - `CONFIG_USE_SPIFLASH` If this flag is defined, U-Boot supports the SPI flash on the OMAP-L138 EVM DSK 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-L138 SoC. Environment variables are also stored on the NAND flash.
  - `CONFIG_SYS_USE_NOR` If this flag is defined, U-Boot supports U-Boot NOR flash access using the OMAP-L138 SoC. Environment variables are also stored on the NOR flash.

Note: Only one of these options should be defined at a time. Defining more than one option results in a compilation error when you build U-Boot.

- `CONFIG_ENV_SIZE` Configures the environment variable size.
- `CONFIG_ENV_OFFSET` Configures the environment variable offset.

Building a New Linux Kernel

- Building the target Linux kernel requires that the CodeSourcery tools have been installed on your Linux host. For documentation on how to install the CodeSourcery tools, please visit the CodeSourcery website
- If you have not already installed the source files for building the target Linux kernel, use your Linux host to extract source files from the `src/kernel/linux-MM.mm.pp.bb.tar.gz` tarball under PSP installation directory, which you obtained when you installed the PSP package in Running the Installation. Use the `tar` command to extract the sources.

Note: Patches from the kernel-patches-MM.mm.pp.bb.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 U-Boot version.

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

- Run the following commands. The first command cleans the build, and the second one configures 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.

```
make distclean ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi-
make da850_omapl138_defconfig ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi-
```

- Some of the kernel features and available drivers may not be enabled in the default configuration. See Configuring Linux Kernel and Using Linux Kernel Drivers describe some useful menus and configuration settings. To enable these features and drivers, the kernel needs to be reconfigured. To begin configuring the Linux kernel, run the following command:

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

In the menu that appears, to select or deselect a feature, press the Space key after moving your cursor over the selection box. An asterisk (*) in the selection box means that feature is currently selected for building statically into the kernel. An empty selection box means that feature is de-selected. To configure a particular feature as a runtime loadable kernel module, press the Space key until an "M" appears in the selection box.

- Run the following command to build the kernel image:

```
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.

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

```
make modules ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi-
```

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

```
make modules 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.

See Loading Linux Kernel Modules for more information on using kernel modules.

Configuring Linux Kernel

For configuring the Linux kernel, run the following commands:

```
make da850_omapl138_defconfig ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi-
make menuconfig ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi-
```

For selecting/de-selecting 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.

For configuring 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.

Using Linux Kernel Drivers

This section describes the procedure to enable various drivers and also has specific usage information.

USB 2.0

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
```

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 support for DA850/OMAP-L138 EVM
```

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:

```
'amixer cget numid=1'
```

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

```
'amixer cset numid=1 80,80' : Sets the volume to 80
'amixer cset numid=2 80,80' : This is also used to adjust playback volume
```

Graphical LCD

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

Character LCD

```
Device Drivers --->
         Graphics support  --->
             <*> Support for frame buffer devices --->
                < > DA8xx/OMAP-L1xx Framebuffer support
         <*> Parallel port support  --->
             <*> DA8XX Dummy Parallel Port
         [*] Staging drivers  --->
             [ ]   Exclude Staging drivers from being built
             <M>       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 ?
             -M-       DA8XX Dummy Parallel Port
```

For character LCD to work, insert the dummy parallel port driver module and the panel driver module as follows:

```
insmod parport-da8xx.ko
insmod panel.ko lcd_proto=2 lcd_width=24 lcd_height=2
```

Character LCD can be tested by writing some the following escape sequences to the LCD device (`echo "ESC Sequence" > /dev/lcd`):

Note: Pressing `"Ctrl-v", "Ctrl-["` on keyboard will generate the `"^["` escape sequence. In some file systems, the escape sequences will not appear on the serial console but the input reaches 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

NAND

```
    <*> 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
```

Note: The last NAND partition reserved for file system is around 508MB big and unmounting this partition after flash_eraseall will take considerable time. For faster unmount time, please reduce the partition size.

NOR

```
    <*> 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 --->
                 <*> Flash device in physical memory map
```

SPI

```
Device Drivers --->

        [*] SPI support  --->
            <*>     SPI controller driver for DaVinci SoC
            [*]     Set DaVinci SPI to DMA mode

        <*> 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 DA8XX/OMAP-L1XX RTC
```

RTC can be tested using "hwclock" utility.

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
```

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.

Testing McBSP:

McBSP test setup comprises of 2 OMAP-L138 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 which demonstates the working of McBSP peripheral.

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

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

Power Management

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

To see the PLL0 frequency changes occurring because of CPU load, please check `/proc/davinci_clocks`. To see the CVDD voltage changes occurring because of CPU load, please check `/sys/class/regulator/regulator.2/microvolts`.

Important 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.

Ethernet (Using RMII PHY on UI board)

```
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
```

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

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 Building a New Linux Kernel

Pre-built binaries for features configured by default as kernel modules are included in the PSP package in the `images/kernel/modules/lib` directory. 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

Loading USB file backed storage gadget

- 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.

```
# modprobe g_file_storage file=/dev/blockdevX
```

- Remove the module by using `rmmod g_file_storage`

Building the User Boot Loader

The following steps build the ARM side User Boot Loader (ARM UBL) using CCStudiov3.3:

- Extract the ARM UBL code from the `src/boot-strap/armubl-MM.mm.pp.bb.tar.gz` file in PSP installation.
- Start CCStudio v3.3. From the menus, choose Project->Open. Browse to the extracted ARM UBL source and open the `ubl-omapl1x8.pjt` project. By default, the project builds ARM UBL for NAND flash. To build for SPI Flash, select the "Config" option as BOOT_SPI in CCStudio window. Similarly, to build for NOR Flash, select the "Config" option as BOOT_NOR in CCStudio window.
- From the menus, choose `Project->Build`. 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 and build for NOR flash places the `ubl-nor.out` in `nor` directory.

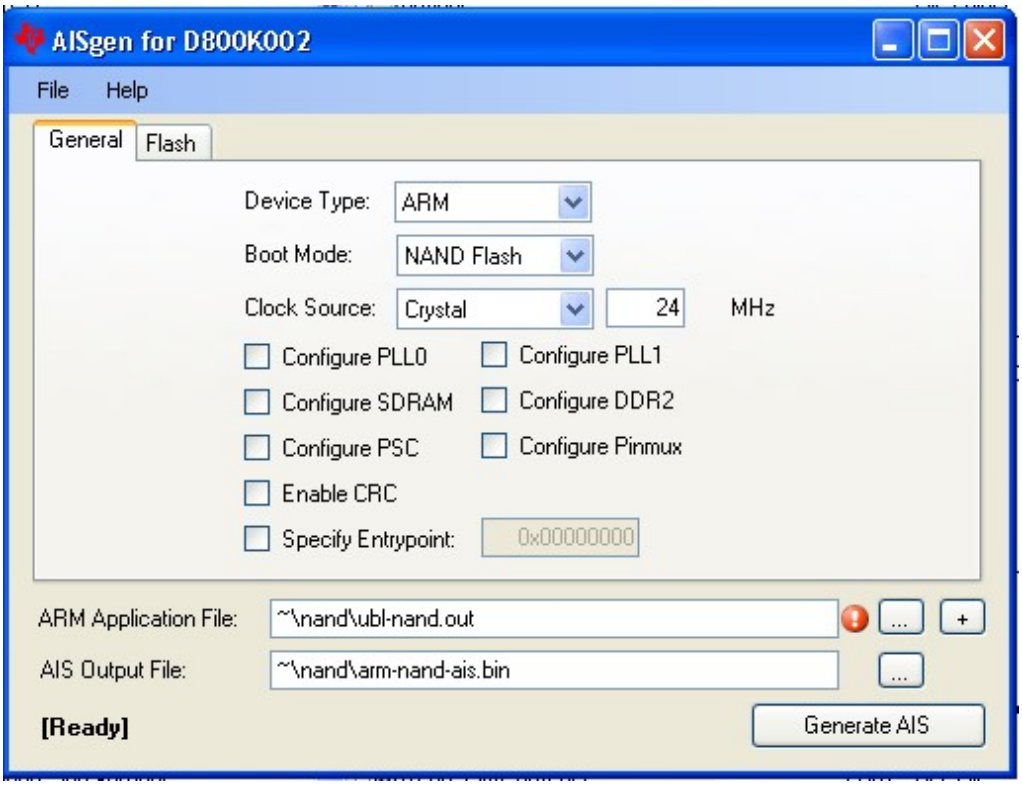
The following steps build the ARM side User Boot Loader (ARM UBL) using CCStudiov4.0:

- Extract the ARM UBL code from the `src/boot-strap/armubl-MM.mm.pp.bb.tar.gz` file in PSP installation.
- Start CCStudio v4.0. From the menus, choose File->Import. Choose CCS->Existing CCS/CCE Eclipse Project. Browse to the extracted ARM UBL source and select the root directory where the ccsv4 project resides. By default, the project builds ARM UBL for NAND flash. To build for SPI Flash, select the "Configuration" option as BOOT_SPI in Project->Properties menu. Similarly, to build for NOR Flash, select the "Configuration" option as BOOT_NOR in Project->Properties menu.
- 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 and build for NOR flash places the `ubl-nor.out` in `nor` directory.

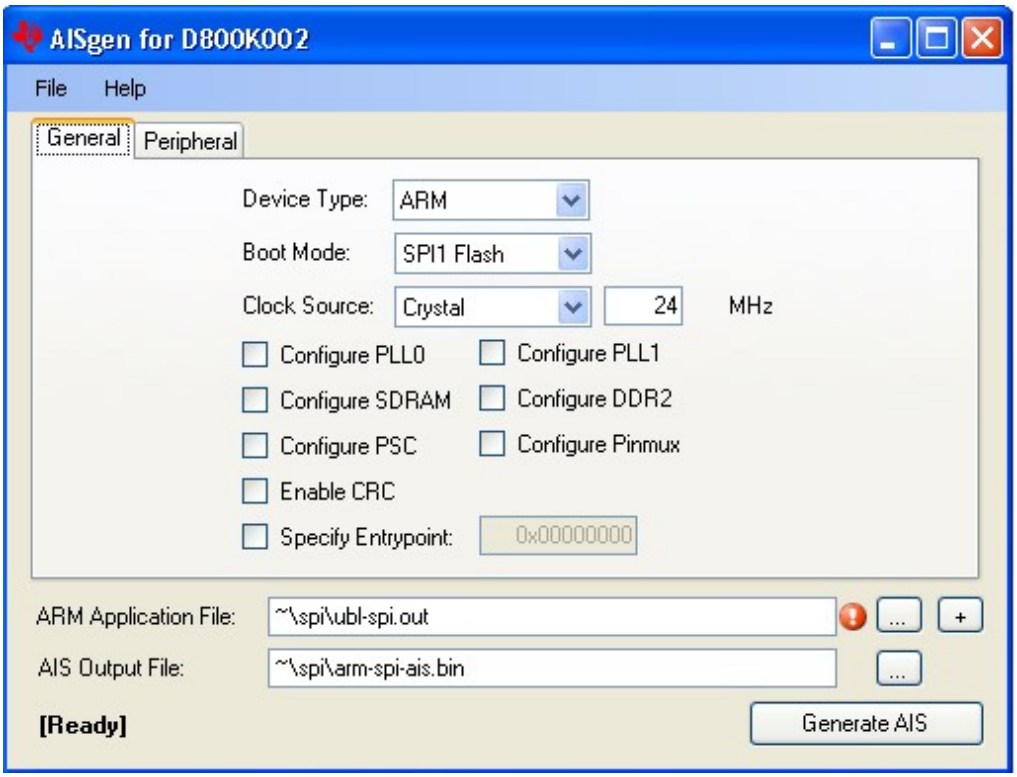
Following steps are not dependent on CCS versions used to build.

- Obtain the AIS file format generator tool described in the OMAP-L1x8 Bootloader Application Report document. Use the tool as shown below to obtain AIS format files ready to be flashed into NAND/SPI/NOR flash.

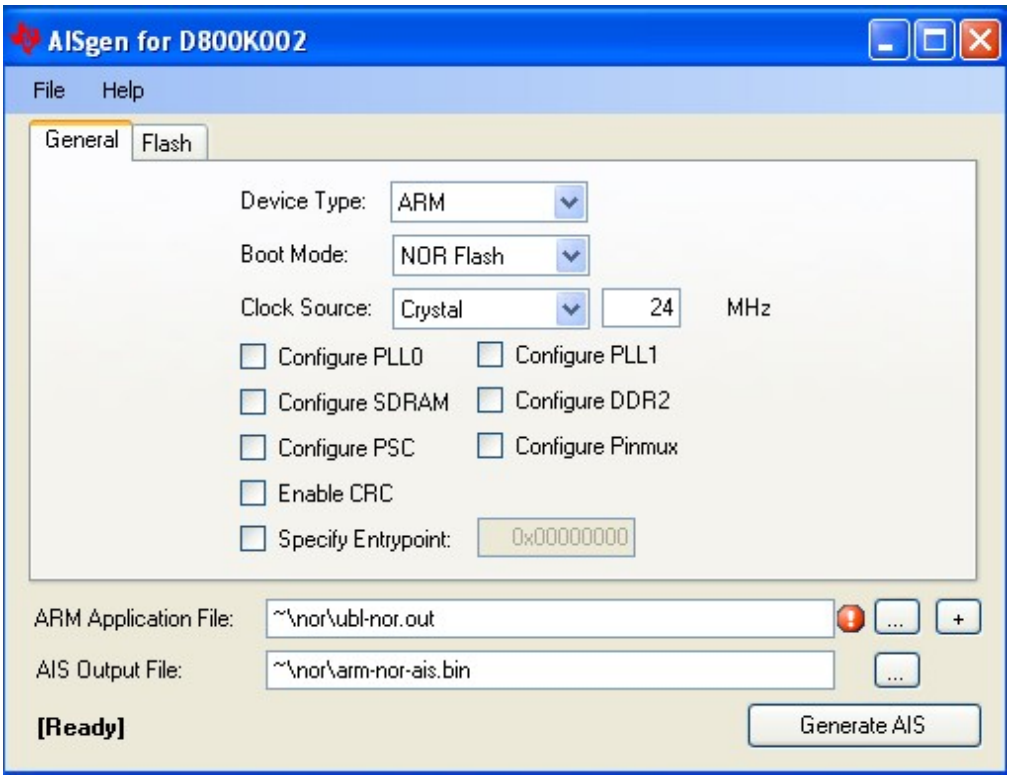
For NAND Flash:



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.



Building SPI Flash writer

SPI flash writer is used to flash the ARM 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:

- Extract the SPI flash writer code from `src/utils/spiflash-writer-MM.mm.bb.pp.tar.gz` file in PSP installation.
- Start CCStudio v3.3. From the menus, choose Project->Open.
- Browse to the the extracted source directory, and open the `ccsv3.3/spiflash_writer.pjt` project.
- From the menus, choose `Project->Build`. The built image `spiflash-writer.out` is placed in the `ccsv3.3/Debug` under the top directory.

Use the following steps to build SPI Flash writer using CCStudio v4.0:

- Extract the SPI flash writer code from `src/utils/spiflash-writer-MM.mm.bb.pp.tar.gz` file in PSP installation.
- Start CCStudio v4.0. 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 `spiflash-writer.out` is placed in the `ccsv4/Debug` under the top directory.

Building 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:

- Extract the NAND flash writer code from `src/utils/nand-writer-MM.mm.bb.pp.tar.gz` directory in PSP installation.
- Start CCStudio v3.3. From the menus, choose Project->Open.
- Browse to the extracted source directory and open the `ccsv3.3/nand_writer.pjt` project file.
- From the menus, choose `Project->Build`. The built image `nand-writer.out` is placed in the `ccsv3.3/Debug` directory under the top level source directory.

Use the following steps to build SPI Flash writer using CCStudio v4.0:

- Extract the SPI flash writer code from `src/utils/nand-writer-MM.mm.bb.pp.tar.gz` file in PSP installation.

- Start CCStudio v4.0. 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.

Building NOR Flash writer

NOR flash writer is used to flash the ARM 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:

- Extract the NOR flash writer code from `src/utils/norflash-writer-MM.mm.bb.pp.tar.gz` directory in PSP installation.
- Start CCStudio v3.3. From the menus, choose Project->Open.
- Browse to the extracted source directory and open the `ccsv3.3/norflash_writer.pjt` project file.
- 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 SPI Flash writer using CCStudio v4.0:

- Extract the SPI flash writer code from `src/utils/norflash-writer-MM.mm.bb.pp.tar.gz` file in PSP installation.
- Start CCStudio v4.0. 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 `norflash-writer.out` is placed in the `ccsv4/Debug` under the top directory.

Additional topics

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 DSP from being woken up by U-Boot, set the 'dspwake' environment variable to 'no', save the environment space and reset the board.

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

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. Follow this procedure based on the flash U-Boot stores its environment variables on.

SPI flash

First run:

```
sf probe 0
```

In SPI boot mode, environment variables are stored at offset 256 KBytes into the SPI flash and environment variables sector size is 64 KBytes (Refer to `CONFIG_ENV_OFFSET` and `CONFIG_ENV_SIZE` for SPI boot mode in `include/configs/da850_evm.h` file to confirm these values). Erase environment sector length bytes (64 KBytes) from environment sector offset (256 KBytes) as follows:

```
sf erase 40000 10000
```

Reset the board and it will come up with default environment variables. Now save the new set of environment variables.

```
saveenv
```

NAND Flash

In NAND boot mode, env variables are stored at offset 0 (zero) of NAND flash and environment variables sector size is 128 KBytes (Refer to `CONFIG_ENV_OFFSET` and `CONFIG_ENV_SIZE` for NAND boot mode in `include/configs/da850_evm.h` file to confirm these values). Erase environment sector length bytes (128 KBytes) from environment sector offset 0 (zero) as follows:

```
nand erase 0 20000
```

Reset the board and it will come up with default environment variables. Now save the new set of environment variables.

```
saveenv
```

NOR Flash

In NOR boot mode, env variables are stored at the 4th sector of NOR flash and environment variables sector size is 128 KBytes (Refer to `CONFIG_ENV_OFFSET` and `CONFIG_ENV_SECT_SIZE` for NOR boot mode in `include/configs/da850_evm.h` file to confirm these values). Unprotect the 4th sector using:

```
protect off 60060000 +20000
```

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

```
erase 60060000 +20000
```

Reset the board and it will come up with default environment variables. Now save the new set of environment variables.

```
saveenv
```

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. Please 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.

Using extended memory available on 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:

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