

OMAPL138 StarterWare Booting And Flashing

From Texas Instruments Wiki

Jump to: [navigation](#), [search](#)

Contents

[hide]

- [1 The StarterWare Bootloader](#)
- [2 Booting a StarterWare Application](#)
 - [2.1 Binary Image Generation](#)
 - [2.1.1 Using AISgen to Create the Bootloader AIS File](#)
 - [2.1.2 Using out2rprc to Create the Application Binary](#)
 - [2.2 Flashing the Application](#)
 - [2.3 Booting the Application](#)
- [3 Booting an ARM+DSP System](#)
- [4 FAQ](#)
 - [4.1 How to flash and boot the sample DSP starterware app on OMAPL138 LCDK board](#)
 - [4.2 How to flash and boot the sample ARM starterware app on OMAPL138 LCDK board](#)
 - [4.3 Download the procedural document On "How to flash and boot the Sample Starterware programs into LCDK BOARD".](#)

The StarterWare Bootloader[[edit](#)]

The StarterWare bootloader is a platform-specific helper application that makes it easy to transition from running a StarterWare application in a debug context (i.e. in Code Composer Studio with a GEL file) to a production context (i.e. running the application automatically when the board is powered on). The bootloader acts as an intermediary between the ROM bootloader (RBL) and the application. During boot, the StarterWare bootloader takes the following actions:

1. Applies common PLL and DDR/EMIF settings
2. Copies application program and data sections from flash memory to DDR
3. Jumps to application entrypoint

If the bootloader encounters an error during boot, it prints diagnostic messages to the UART console.

The StarterWare bootloader comes pre-built with the standard installation and typically does not need to be modified or rebuilt unless you want to port it to a custom hardware platform. The prebuilt binary can be found in the following location:

```
<StarterWare Installation  
Path>/binary/armv5/<toolchain>/omap1138/evmOMAPL138/bootloader/Release/  
boot.out
```

Booting a StarterWare Application[[edit](#)]

Binary Image Generation[[edit](#)]

Booting a StarterWare application requires two binary images:

1. An AIS file containing the StarterWare bootloader
2. A binary file containing the application

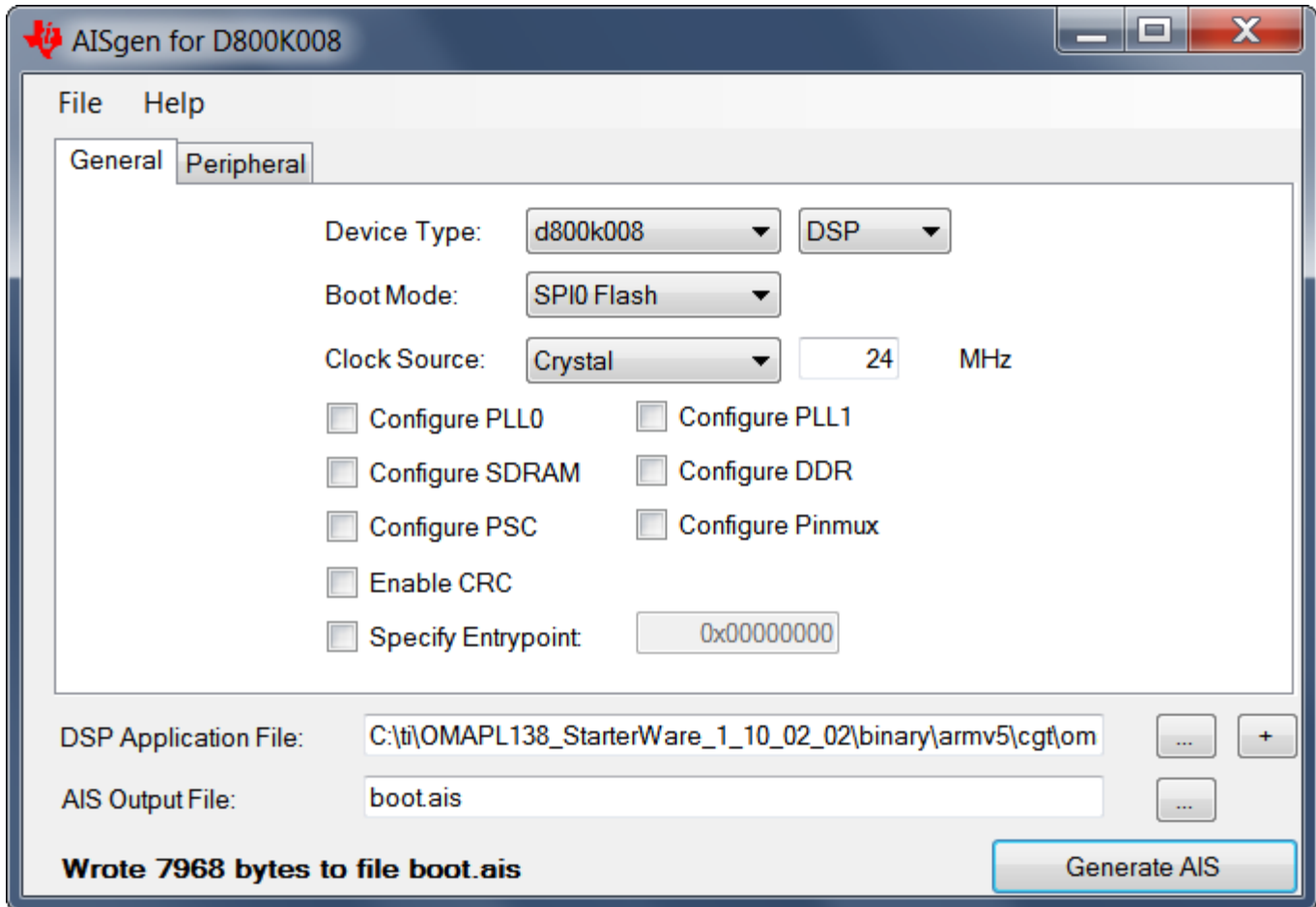
These images are generated using two separate tools, both of which are included in the `tools` folder of the standard StarterWare installation.

Using AISgen to Create the Bootloader AIS File[[edit](#)]

The AISgen GUI can be used to generate the StarterWare bootloader AIS file. All of the critical settings are located on the **General** tab of the AISgen GUI:

- **Boot Mode** - Set to SPI0 Flash
- **Application File** - Set to bootloader .out file
- **AIS Output File** - Desired AIS file name/path

The AISgen tool has many options to configure the device at boot time, but these are generally not necessary since the StarterWare bootloader will boot very quickly and apply its own configuration immediately. The following screenshot shows the typical settings that should be specified to generate the bootloader AIS file. Simply click the **Generate AIS** button to create the AIS file.



NOTE

You will typically need to generate the bootloader AIS file only once. That same AIS file can be used with any number of different applications.

Using out2rprc to Create the Application Binary[[edit](#)]

The out2rprc command line utility is used to generate the application binary image. This tool strips out the initialized sections from the executable file (i.e. *.out) and places them in a compact format that the StarterWare bootloader can understand. Generating this binary file is very simple:

```
$> out2rprc.exe [application].out [application].bin
```

The output (bin) file is typically much smaller than the original executable (out) file.

Flashing the Application[[edit](#)]

Once you have generated both binary images (i.e. the AIS and bin files described above), you are ready to flash these images to the EVM's SPI flash memory. This can be done

using the standard serial flasher utility that's included in the `tools` folder of the standard StarterWare installation:

1. Set the boot switches for UART2 boot (5:8 = 0011 on EVM)
2. Run SFH from the command line:
 - o `$> sfh_OMAP-L138.exe -flash [bootloader].ais [application].bin`
3. Power on or reset the EVM

The SFH tool may take several minutes to complete depending on the size of the application. When SFH completes successfully, your application is ready to boot.

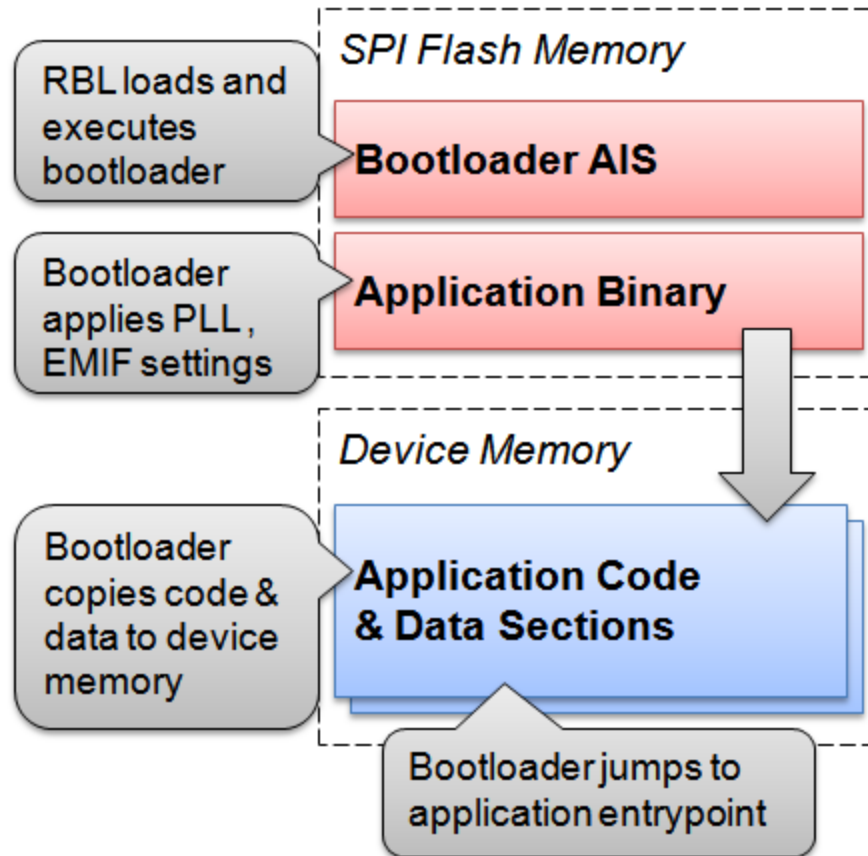
NOTE

The SFH tool automatically adds simple header information to the application binary file. This header tells the bootloader basic information about the application contents, including its total size in bytes.

Booting the Application[\[edit\]](#)

After the binary images have been flashed to the EVM, the application is ready to boot. This is as simple as setting the boot switches for SPI0 flash boot (5:8 = 0000 on the EVM) and powering on or resetting the board.

During boot, control passes from the ROM bootloader to the StarterWare bootloader, which configures the system and loads the application. Finally, the bootloader jumps to the application entrypoint. The overall process is summarized in the following diagram.



Booting an ARM+DSP System[[edit](#)]

The above instructions allow you to boot an ARM application, but StarterWare is also capable booting a multicore ARM+DSP system on OMAP-L138. The entire procedure listed above is still followed, with one exception: you must generate a combined ARM+DSP application binary using the `out2rprc` tool. This is as simple as adding one command line argument when you invoke the `out2rprc` tool:

```
$> out2rprc.exe [ARM application].out [DSP application].out
[application].bin
```

Note that the ARM application must be specified first, **before** the DSP application. Also, you should be careful to observe the following guidelines:

1. Do not allow the ARM and DSP application memory maps to overlap
2. The DSP application entrypoint must be aligned to 1024 bytes
 - o If your DSP application entrypoint doesn't meet this criterion, you can fix it by adding this line (or something similar) to the SECTIONS area of its linker command file:

```
.text:_c_int00: align=1024 > DDR2
```

During boot, the StarterWare bootloader executes on the ARM core. After device configuration, it loads both the ARM and DSP applications to memory. It then wakes up the DSP and sets it to run from the DSP application entrypoint. Finally, it jumps to the ARM application entrypoint as normal.

NOTE

You must have StarterWare version 1.10.02.02 or later to boot an ARM+DSP system using the StarterWare bootloader.

FAQ[\[edit\]](#)

How to flash and boot the sample DSP starterware app on OMAPL138 LCDK board[\[edit\]](#)

It is assumed that you already installed "OMAPL138_StarterWare_1_10_04_01". For example, Consider you want to flash and boot the Raster example given at path `..\ti\OMAPL138_StarterWare_1_10_04_01\examples\lcdkOMAPL138\raster`

1. First, import the raster display project given at `..\ti\OMAPL138_StarterWare_1_10_04_01\build\c674x\cgt_ccs\omap1138\lcdkOMAPL138\raster` into CCS, build and generate the rasterDisplay.out.
2. Using CCS, Open up the map file of DSP raster example, "raster_c674x_omap1138_lcdkOMAPL138.map" and notice the entry point "ENTRY POINT SYMBOL: "_c_int00" address: c009d000".
3. Import the ARM starterware bootloader project given at `..\ti\OMAPL138_StarterWare_1_10_04_01\build\armv5\cgt_ccs\omap1138\lcdkOMAPL138\bootloader` and import it into CCS.
 1. Before building the bootloader project, in bl_main.c, assign the DSP entry point address(Please note that this is the address noticed in the raster_c674x_omap1138_lcdkOMAPL138.map MAP file of raster display project) as `0xc009d000`(i.e., `unsigned int DspEntryPoint = 0xc009d000;`)
 2. Build the project and generate the boot.out file.
4. Using AIS gen tool, load the appropriate configuration file and convert the boot.out into boot.ais
5. Using out2rprc, convert the rasterDisplay.out into rasterDisplay.bin
6. On OMAPL138 LCDK board, set the boot switches for UART mode(1:4 = 0101 on OMAPL138 LCDK). Open serial terminals like teraterm and make sure you get the "BOOTME" message.
7. By sfh utility, flash the two images, boot.ais and rasterDisplay.bin using the below command.

```
$>sfh_OMAP-L138.exe -flash boot.ais rasterDisplay.bin -targetType  
OMAPL138_LCDK -flashType NAND -p COM3
```

Where 3 is the COM port number. After flashing successfully, Change the boot switches for NAND boot (1:4 = 0111 on OMAPL138 LCDK) and power on or reset the LCDK board. You will Observe the starterware messages from both the sample projects, bootloader and Rasterdisplay.

How to flash and boot the sample ARM starterware app on OMAPL138 LCDK board[[edit](#)]

It is assumed that you already installed "OMAPL138_StarterWare_1_10_04_01". For example, Consider you want to flash and boot the Raster example given at path `..\ti\OMAPL138_StarterWare_1_10_04_01\examples\ldkOMAPL138\raster`

1. First, import the raster display project given at `..\C:\ti\OMAPL138_StarterWare_1_10_04_01\build\armv5\cgt_ccs\omap1138\ldkOMAPL138\raster` into CCS, build and generate the rasterDisplay.out.
2. Using CCS, Open up the MAP file, `"raster_c674x_omap1138_ldkOMAPL138.map"` and notice the entry point `"ENTRY POINT SYMBOL: "_c_int00" address: c1080000"`.
3. Import the ARM starterware bootloader project given at `..\ti\OMAPL138_StarterWare_1_10_04_01\build\armv5\cgt_ccs\omap1138\ldkOMAPL138\bootloader` and import it into CCS.
 1. Before building the bootloader project, in `bl_main.c`, assign the entry point address(Please note that this is the address noticed in the `raster_armv5_omap1138_ldkOMAPL138.map` linker file of raster display project) as `0xc1080000`(i.e., unsigned int `entryPoint = 0xc1080000;`)
 2. Buid the project and generate the `boot.out` file.
4. Using AIS gen tool, load the appropriate configuration file and convert the `boot.out` into `boot.ais`
5. Using `out2rprc`, convert the `rasterDisplay.out` into `rasterDisplay.bin`
6. On OMAPL138 LCDK board,set the boot switches for UART mode(1:4 = 0101 on OMAPL138 LCDK). Open serial terminals like `teraterm` and make sure you get the "BOOTME" message.
7. By `sfh` utility, flash the two images, `boot.ais` and `rasterDisplay.bin` using the below command.

```
$>sfh_OMAP-L138.exe -flash boot.ais rasterDisplay.bin -targetType  
OMAPL138_LCDK -flashType NAND -p COM3
```

Where 3 is the COM port number. After flashing successfully, Change the boot switches for NAND boot (1:4 = 0111 on OMAPL138 LCDK) and power on or reset the LCDK board. You will Observe the starterware messages from both the sample projects, bootloader and Rasterdisplay.

Download the procedural document On "How to flash and boot the Sample Starterware programs into LCDK BOARD".[\[edit\]](#)

[Under construction]


Keystone=

1. switchcategory:MultiCore=

- For technical support on MultiCore devices, please post your questions in the [C6000 MultiCore Forum](#)
- For questions related to the BIOS MultiCore SDK (MCSDK), please use the [BIOS Forum](#)


Please post only comments related to the article **OMAPL138 StarterWare Booting And Flashing** here.

Please post only comments related to the article **OMAPL138 StarterWare Booting And Flashing** here.



Engage in the TI E2E Community. Ask questions, share knowledge, explore ideas and help solve problems with fellow engineers.

Links



- [Amplifiers & Linear](#)
- [Audio](#)
- [Broadband](#)
- [DLP & MEMS](#)
- [High-Reliability](#)
- [Processors](#)
 - [ARM Processors](#)
- [Switches & Multiplexers](#)
- [Temperature Sensors &](#)

RF/IF & Digital Radio Clocks & Timers Data Converters	Interface Logic Power Management	<ul style="list-style-type: none"> • Digital Signal Processors (DSP) • Microcontrollers (MCU) • OMAP Applications Processors 	Control ICs Wireless Connectivity
---	--	---	---

Retrieved from

["https://processors.wiki.ti.com/index.php?title=OMAPL138_StarterWare_Bootig_And_Flashing&oldid=221933"](https://processors.wiki.ti.com/index.php?title=OMAPL138_StarterWare_Bootig_And_Flashing&oldid=221933)

[Category:](#)

- [StarterWare](#)

Navigation menu

Personal tools

- [Log in](#)
- [Request account](#)

Namespaces

- [Page](#)
- [Discussion](#)



Variants

Views

- [Read](#)
- [View source](#)
- [View history](#)



More

Search



Navigation

- [Main Page](#)
- [All pages](#)
- [All categories](#)
- [Recent changes](#)
- [Random page](#)
- [Help](#)

Toolbox

- [What links here](#)
- [Related changes](#)
- [Special pages](#)
- [Printable version](#)
- [Permanent link](#)
- [Page information](#)

- This page was last edited on 13 October 2016, at 06:08.
- Content is available under [Creative Commons Attribution-ShareAlike](#) unless otherwise noted.

- [Privacy policy](#)
- [About Texas Instruments Wiki](#)
- [Disclaimers](#)
- [Terms of Use](#)

