By using this document, in addition to any agreements you have with Intel, you accept the terms set forth below.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked “reserved” or “undefined.” Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm

Any software source code reprinted in this document is furnished for informational purposes only and may only be used or copied and no license, express or implied, by estoppel or otherwise, to any of the reprinted source code is granted by this document.

This document contains information on products in the design phase of development.

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. Go to: http://www.intel.com/products/processor_number/

Code Names are only for use by Intel to identify products, platforms, programs, services, etc. ("products") in development by Intel that have not been made commercially available to the public, i.e., announced, launched or shipped. They are never to be used as "commercial" names for products. Also, they are not intended to function as trademarks.

Intel, Intel Atom, [include any Intel trademarks which are used in this document] and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

© 2014 Intel Corporation.
## Revision History

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>November 2014</td>
<td>001</td>
<td>Initial release</td>
</tr>
</tbody>
</table>
1.0 Introduction

1.1 Purpose

The purpose of this document is to provide information about the MinnowBoard MAX Circuit Board, with guidance for building an example boot loader for this board that is based on the Intel® Firmware Support Package (FSP).

The MinnowBoard MAX is an open hardware embedded circuit board, a product of minnowboard.org. The MinnowBoard MAX is based on the Intel® Atom™ E3800 Product Family System on a Chip (SoC).

1.2 Intended Audience

The target audience for this document is all platform and system developers who intend to use an FSP-based boot loader for the firmware solution for their overall design based on the Intel® Atom™ Processor E3800 Product Family. This group includes, but is not limited to, system BIOS developers, boot loader developers, and system integrators.

1.3 Related Documents


1.4 Conventions

To better illustrate some of its points, this document may provide code snippets. Such code snippets follow the GNU C Compiler and GNU Assembler syntax.
## 1.5 Acronyms and Terminology

### Table 1. Acronyms and Terminology

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMT</td>
<td>Advanced Management Technology</td>
</tr>
<tr>
<td>AVT</td>
<td>Advanced Vector Extensions</td>
</tr>
<tr>
<td>BCT</td>
<td>Binary Configuration Tool</td>
</tr>
<tr>
<td>BSP</td>
<td>Boot Strap Processor</td>
</tr>
<tr>
<td>BWG</td>
<td>BIOS Writer’s Guide</td>
</tr>
<tr>
<td>CI</td>
<td>Current Image</td>
</tr>
<tr>
<td>CRB</td>
<td>Customer Reference Board</td>
</tr>
<tr>
<td>DMI</td>
<td>Direct Media Interface</td>
</tr>
<tr>
<td>FDI</td>
<td>Flexible Display Interface</td>
</tr>
<tr>
<td>FSP</td>
<td>Firmware Support Package</td>
</tr>
<tr>
<td>FSP API</td>
<td>Firmware Support Package Interface</td>
</tr>
<tr>
<td>FWG</td>
<td>Firmware Writer’s Guide</td>
</tr>
<tr>
<td>KGI</td>
<td>Known Good Image</td>
</tr>
<tr>
<td>ME</td>
<td>Management Engine</td>
</tr>
<tr>
<td>PCD</td>
<td>Platform Configuration Database</td>
</tr>
<tr>
<td>PCH</td>
<td>Platform Controller Hub</td>
</tr>
<tr>
<td>SMI</td>
<td>System Management Interrupt</td>
</tr>
<tr>
<td>SMM</td>
<td>System Management Mode</td>
</tr>
<tr>
<td>SMRAM</td>
<td>System Management RAM</td>
</tr>
<tr>
<td>SPI</td>
<td>Serial Peripheral Interface</td>
</tr>
<tr>
<td>TSEG</td>
<td>Top Segment, a reserved segment of memory at the top of its address space to be used as SMRAM</td>
</tr>
</tbody>
</table>
2.0 Intel® Atom™ Processor E3800 Product Family Hardware Platform

The MinnowBoard MAX circuit board is based on the Intel® Atom™ Processor E3800 Product Family hardware platform, code-named Bay Trail (hereinafter referred to as “the hardware platform”).

This hardware platform consists of a system on a chip (SoC), which is based on one to four third-generation Intel® Atom™ Processor E3800 Product Family Atom™ processor cores. The hardware platform includes the Gen7 Intel® graphics architecture and is built on 22-nanometer process technology.

Some of the features of the hardware platform are the following:

- Increased I/O integration
- Wide range of optional I/O flexibility
- Rugged and reliable
- Greater media competence
- Superior process technology
- Leading performance
- Enhanced graphics engine

*Note:* The hardware platform was previously code named Intel® Valleyview SoC, prior to being code-named Intel® Bay Trail.
2.1 Intel® Firmware Support Package

An Intel® Firmware Support Package (Intel® FSP) is a firmware component provided in binary form that contains initialization code for a specific Intel platform. Engineers building systems that are based on a particular platform can integrate the corresponding Intel® FSP with the boot loader of their choice.

The Intel® FSP for the hardware platform handles the initialization of the processor, memory, and I/O subsystems for hardware designs based on this hardware platform.
3.0 MinnowBoard MAX Circuit Board

The MinnowBoard MAX is based on the Intel® Atom™ E3800 Processor hardware platform, which is illustrated in Figure 2.

Figure 2. MinnowBoard MAX Block Diagram

The board provides both a serial console and graphics output capabilities.
If you are developing a boot loader for which all user interaction is performed through the serial console, connect a terminal to the six-pin header J4, shown in Figure 3. Pin 1 is on the right in Figure 3, furthest away from the SATA port.

Figure 3. Serial Header Location
Connect a six-pin TTL to USB cable between the board and a development host system, such as FTDI part number TTL-232R-3V3, shown in Figure 4. This cable has an FTDI USB to 3.3V TTL level serial adapter chip integrated into the USB plug.

Figure 4. FTDI USB to TTL serial cable

Once connected, the host system should detect the USB-to-serial adapter in the cable. Compatible drivers should install automatically on host systems running Microsoft Windows*. Other host operating systems may require downloading and installing appropriate drivers.

The default serial communication parameters for the serial console are:

- 115,200 baud
- 8-N-1 bit configuration
- No flow control
Figure 5 shows a close-up of the FTDI cable attached to the MinnowBoard MAX’s J4 header.

Figure 5. FTDI cable attached to J4

If you are developing a boot loader for which user interaction is performed through a monitor, connect a digital monitor to the micro-HDMI port by means of a micro-HDMI to HDMI adapter or cable.

3.1 Further Information

For more information on MinnowBoard MAX, refer to the following sites:

- minnowboard.org
- The MinnowBoard MAX page on the Embedded Linux* Wiki
4.0 Example Boot Loader

Intel® provides and supports the Intel® Atom™ Processor E3800 Product Family Firmware Support Package (hereafter referred to as the “FSP kit”) for the MinnowBoard MAX. However, Intel does not provide or support a complete boot loader solution for this board.

The embedded firmware ecosystem has developed an example boot loader solution for the MinnowBoard MAX that uses the FSP kit. This solution is based on the open source coreboot* project at coreboot.org. While Intel® does not endorse or support boot loader solutions based on the coreboot project, the example coreboot-based boot loader provides a good teaching model for how to integrate the Intel® FSP into a complete boot loader solution.

Note: The steps to generate the example boot loader for the MinnowBoard MAX are provided to Intel customers as-is, with no warranty or support. Please contact a firmware ecosystem vendor to help you develop a production-worthy firmware solution based on the Intel® FSP for your hardware designs.

4.1 Example Boot Loader Design

The FSP’s role is to initialize the processor, memory, and I/O subsystem. The example coreboot-based boot loader calls into the FSP for these initialization steps, then goes on to prepare and load a primary target that the coreboot project calls a payload. See the coreboot project’s documentation for further details.

The default payload for the example coreboot-based boot loader is the SeaBIOS, which is provided by a related open source firmware project. The SeaBIOS attempts to boot an OS image from a storage device attached to the board.

4.2 Boot Loader Development Environment

Although the FSP can be used with any software development environment, the example coreboot-based boot loader described herein was developed using Fedora® 18 Linux* and the standard GNU development tools.

Note: Intel® does not endorse or support any specific development environment for developing boot loader firmware that integrates with the FSP.

4.3 Preparing the Coreboot Build Environment

To download the coreboot.org development system, your development host must have the Git version control system installed.

Once coreboot has been downloaded, you must run a command to download and build the exact GCC toolchain required by the coreboot project. In order to build the coreboot-specific toolchain, your development host must have its own distribution-provided GCC tool chain. Consult the documentation at coreboot.org for the tool chain components required.

On your development host, follow these steps to download the coreboot.org development environment and then build its required GCC tool chain:

1. Create a directory to contain your project. This directory is to be the parent of the coreboot directory. For the purpose of these example steps for the MinnowBoard MAX, we will refer to this as the MM directory.

2. Navigate into your new MM project directory.
3. **Use the `git clone` command as follows:**

   ```bash
   git clone http://review.coreboot.org/p/coreboot
   ```

   This step downloads a directory named `coreboot`.

4. **Navigate into the `coreboot` directory.**

5. **Run the following command to build the project-specific GCC tool chain required by the coreboot project:**

   ```bash
   make crossgcc-i386
   ```

   This command can take from a few minutes up to an hour or more to complete, depending on the power of your development host.

   If `make crossgcc-i386` fails to build (which can happen on recent Ubuntu Linux systems), try these alternate steps:
   a. In the `coreboot` directory, navigate to the `util/grossgcc` subdirectory.
   b. Run `. /buildgcc`
      
      This command takes the same amount of time to complete as the `make` version.
   c. When complete, return to the root of the `coreboot` directory with this command:
      ```bash
      cd .. /.. 
      ```

   If you continue to have difficulty getting the tool chain to build, make sure that you have all the required development tools installed on your Linux system, as listed on the coreboot.org site. If you are still unable to get the tool chain to build, then consider using an alternate Linux distribution such as Fedora 18.

§
5.0 Building the Example Boot Loader

To integrate the Intel® FSP with the coreboot.org project to build the example boot loader for MinnowBoard MAX, you must do the following:

- Download and install the FSP kit.
- Copy files from the FSP kit to a new directory parallel to the coreboot directory that was created in the previous chapter.
- Run the make menuconfig command to specify the MinnowBoard MAX and its build options.
- Run make.

Follow these steps:

1. Go to www.intel.com/fsp and download the Intel® Atom™ Processor E3800 Product Family FSP kit, which is distributed both as a Microsoft Windows® executable file, BAY_TRAIL_FSP_KIT_GOLD3.exe, and as a Linux® archive, BAY_TRAIL_FSP_KIT_GOLD3.tgz. You can use either version of the FSP kit to build the example boot loader.

2. Install the kit:
   - For Windows*: Execute the BAY_TRAIL_FSP_KIT_GOLD3.exe file and follow the on-screen dialogs to install the Bay Trail kit. The default installation directory is C:\BAY_TRAIL_FSP_KIT.
   - For Linux*: Extract the contents of the BAY_TRAIL_FSP_KIT_GOLD3.tgz file and follow the instructions in the Readme_Extract.txt file. The FSP kit extracts into a subdirectory named BAY_TRAIL_FSP_KIT.

3. Navigate to the MM directory created in the previous chapter. Create a new subdirectory named intel parallel to the coreboot directory.

4. Copy files from the BAY_TRAIL_FSP_KIT directory where the FSP kit was installed to the MM/intel directory on your development host as follows, creating the path to the specified target directories as required:
   a. Copy BAY_TRAIL_FSP_KIT/FSP/*.fd to MM/intel/fsp/baytrail and rename the file to BAYTRAIL_FSP.fd.
   b. Copy BAY_TRAIL_FSP_KIT/FSP/include/*.*h to MM/intel/fsp/baytrail/include.
   c. Copy BAY_TRAIL_FSP_KIT/FSP/src/*.*c to MM/intel/fsp/baytrail/src.
   d. Copy BAY_TRAIL_FSP_KIT/Microcode/*.h to MM/intel/cpu/baytrail/microcode.
   e. Copy BAY_TRAIL_FSP_KIT/Graphics/INTEL_EMGD.VBIOS_GOLD_VERSION_36_2_3_3698/Vga.dat to MM/intel/cpu/baytrail/vbios.

5. Navigate to the MM/coreboot directory and run:

   make menuconfig

6. Open the mainboard menu. Select Intel as the Mainboard vendor. Then select Minnowboard Max FSP-based board as the Mainboard model.

7. While still in the mainboard menu, move to the Configure defaults for the Intel FSP package option and press the space bar to select it. Leave all other settings in their default state.

8. Back out to the main menu, and select the Payload option. In the list of options, select SeaBIOS version (1.7.4), and select master. This selects the latest version of the SeaBIOS payload that includes xHCI support.
9. Back out to the main menu, then select the **Console** option. Navigate to the **Default console log level** option and set it to **4: WARNING**.

10. Select **Exit** twice to return to the command prompt. Save the configuration when prompted to do so.

11. Now, build the boot loader with a single command:

   ```
   make
   ```

12. If the build completes without errors, the newly created firmware image is generated into the following directory and file:

   ```
   coreboot/build/coreboot.rom
   ```

13. The generated **coreboot.rom** file is 4 MB in size. This file can be programmed into the firmware flash memory device on the MinnowBoard MAX by following the procedures in the next section.

   §
6.0 Updating the Firmware

The MinnowBoard MAX is equipped with a single 8 MB flash device that contains all of the system firmware, including the Firmware Descriptor, Security Firmware, and the BIOS or Boot Loader. By default, the MinnowBoard MAX comes from its manufacturer with a standard BIOS installed.

To replace the factory BIOS with boot loader firmware, you can generate a complete 8 MB firmware image and update the whole firmware flash device, or you can update only the BIOS region of the firmware image with the new boot loader. For preproduction or development purposes, updating only the boot loader is an acceptable method. However, for production purposes, use a complete 8 MB firmware image. Note that information about how to generate a complete 8 MB firmware image is outside the scope of this document.

6.1 Programming the Firmware

The flash device on the MinnowBoard MAX can be programmed with new firmware using a DediProg® SF100 programmer, which is shown in Figure 6.

Figure 6. SF100 Programmer

The SF100 connects to a host development system through its USB plug for communication with the controller software and to obtain power, and it connects to the flash device to be programmed via the ISP pin header.

Additional technical information about the SF100 programmer, including drivers and software for Microsoft Windows® environments, can be obtained from DediProg’s website at: http://www.dediprog.com/product/SPI%20Flash%20Solution/89

The SF100 programmer is also supported by the Linux* flashrom utility. The flashrom utility is not supported by DediProg® or by Intel®. Additional technical information about the flashrom utility can be obtained from the flashrom website at: http://www.flashrom.org/

All of the following instructions regarding the use of the SF100 programmer assume that the DediProg® SF100 drivers and software are installed on a PC running Microsoft Windows®.
6.1.1 Connecting the SF100 to the MinnowBoard MAX

The SF100 includes a cable (shown in Figure 7) that connects between the ISP pin header and the 8-pin header labeled as J1 on the MinnowBoard MAX (shown in Figure 8 and Figure 9).

Figure 7. DediProg Cable

Figure 8. Programming Header Location
On the programming header, pin 1 is in the upper right in Figure 9, closest to the J1 label on the board.

6.2 Creating a Firmware Backup

It is important to back up any existing working firmware so that you can always restore the system to a known working condition. Use the following procedure to back up the existing firmware.

1. Disconnect the power from the MinnowBoard MAX. **Do not apply power to the MinnowBoard MAX at any time during this procedure.**
2. Connect the cable from the SF100 programmer to header J1 on the MinnowBoard MAX.
3. Open the DediProg Engineering application.
4. Next to the **Currently working on** section near the top of the window, make sure that **Application Memory Chip 1** is selected.
5. Click the **Edit** button at the top of the window.
6. Click **Read** at the top of the window. Wait for the completion of the read operation.
7. Click **Chip Buffer to File** to save the 8 MB firmware image to a file.
8. Remove the cable from header J1 on the MinnowBoard MAX before applying power to the MinnowBoard MAX.
6.3 Programming a Complete Firmware Image

Use the following procedure to program a complete firmware image to the MinnowBoard MAX, such as when restoring the firmware image that was backed up in the previous section, or when programming a complete 8 MB firmware image in a production scenario. If you just want to update the BIOS region of the firmware image with a new boot loader, skip this section and go to Section 6.4.

1. Disconnect power from the MinnowBoard MAX. **Do not apply power to the MinnowBoard MAX at any time during this procedure.**
2. Connect the cable from the SF100 programmer to header J1 on the MinnowBoard MAX.
3. Open the DediProg Engineering application.
4. Next to the **Currently working on** section near the top of the window, make sure that **Application Memory Chip 1** is selected.
5. Click the **Config** button at the top of the window.
6. In the Advanced Settings dialog, click the **Prog** button on the left.
7. Select **Program a whole file starting from address 0 of a chip**.
8. Click the **Flash Options** button on the left.
9. Select the check box **Unprotect block automatically when block(s) protected**.
10. Click **OK**.
11. Click the **File** button at the top of the window. This opens a file-open dialog for selecting the complete 8 MB firmware file to be programmed to the flash device.
12. Click the **Erase** button at the top of the window to erase the entire flash memory device. Wait for completion of the erase operation.
13. Click the **Prog** button at the top of the window to program the selected firmware image to the flash memory device on the target platform. Wait for completion of the programming operation.
14. Click the **Verify** button at the top of the window to verify that the firmware image was successfully programmed to the flash memory device. Wait for completion of the verify operation.
15. Remove the cable from header J1 on the MinnowBoard MAX before applying power to the MinnowBoard MAX.

6.4 Updating Only the Boot Loader

You can update only the boot loader portion of the firmware without having to program the complete 8 MB firmware image. For example, the 4 MB boot loader built in section 5.0 can be programmed to the last 4 MB of the 8 MB firmware image.

The DediProg software for the SF100 allows updating just the boot loader portion of the firmware image using a batch programming process. Use the following procedure to program the boot loader code to the BIOS region.

1. Disconnect power from the MinnowBoard MAX. **Do not apply power to the MinnowBoard MAX at any time during this procedure.**
2. Connect the cable from the SF100 programmer to header J1 on the MinnowBoard MAX.
3. Open the DediProg Engineering application.
4. Next to the **Currently working on** section near the top of the window, make sure that **Application Memory Chip 1** is selected.
5. Click the **File** button at the top of the window. This opens a file-open dialog for selecting the file containing the boot loader file to be programmed to the MinnowBoard MAX. Select the **coreboot.rom** file that you built in section 5.0.
6. Click the **Config** button at the top of the window.
7. In the Advanced Settings dialog that appears, click the **Batch** button on the left.

8. Under **Batch Operation Options**, select **Update memory according to Region configuration**, select **Region 1**, and enter **400000** to **7FFFFF**.

9. Select the **Identify Chip** and **Require Verification after completion** check boxes. Make sure all other check boxes are cleared (unchecked).

10. Verify that the **Sequences Details** at the bottom of the window are as shown in **Table 2**.

   **Table 2. Sequence Details**
   
<table>
<thead>
<tr>
<th>Steps</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Identify before operation starts.</td>
</tr>
<tr>
<td>2</td>
<td>Read from the chip.</td>
</tr>
<tr>
<td>3</td>
<td>Blank check.</td>
</tr>
<tr>
<td>4</td>
<td>Erase chip (if not blank).</td>
</tr>
<tr>
<td>5</td>
<td>Program chip.</td>
</tr>
<tr>
<td>6</td>
<td>Verify after operation completes.</td>
</tr>
</tbody>
</table>

11. Click the **Flash Options** button on the left.

12. Select the check box **Unprotect block automatically when block(s) protected**.

13. Click **OK** to close the Advanced Settings dialog.

14. Click the **Batch** button at the top of the window. Wait for completion of the batch operation.

15. Remove the cable from header J1 before applying power to the MinnowBoard MAX.

**6.5 Booting the Example Boot Loader**

By default, the coreboot-based example boot loader boots into the coreboot-provided SeaBIOS. The default behavior of SeaBIOS is to seek a bootable OS image on an attached storage device, such as a SATA or USB disk, or a Micro SD card.

To interact with the boot loader, connect a terminal or development host to the J4 connector, as discussed in section 3.0. The MinnowBoard MAX provides a **Power** button (but no **Reset** button), as shown in **Figure 10**. To power up the board, turn on the power supply, and then press the board’s **Power** button.

**Figure 10. Location of the Power Button**
When power is applied to the MinnowBoard MAX and the boot loader initializes the board and boots the SeaBIOS payload, various messages appear on the terminal connected to the J4 header.

If a storage device with a bootable OS image is placed in the Micro SD slot, or is connected to one of the board’s SATA or USB ports, the booting OS displays messages on the serial port terminal, or on an attached monitor, as determined by the configuration of the OS image.

If the OS image provides a graphical user interface on the monitor, you may need to attach a keyboard and mouse to the USB ports in order to fully interact with the booted operating system. If you want to reserve the MinnowBoard MAX’s USB 3.0 port for a storage device, then attach a USB 2.0 hub to the lower USB 2.0 port, and attach keyboard and mouse to the hub.
7.0 Creating Custom Images

There are two ways to edit the components of a boot loader image to specify custom settings:

- Edit the coreboot image specifications, then rebuild the coreboot.rom image file.
- Edit the binary FSP file to include or exclude support for hardware features, or to rebase the firmware volume image.

7.1 Edit Coreboot Image Specifications

Use the menuconfig utility provided with the coreboot distribution to edit the specifications of the coreboot.rom image file to be generated.

The following example shows the steps to build a 2 MB image file instead of a 4 MB image file.

1. At the Bash prompt on your development host, navigate to the coreboot directory.
2. Start the menuconfig utility with the following command:

   make menuconfig

3. Use the arrow keys to select the Mainboard option.
4. Select the ROM chip size option.
5. Use the arrow keys to select the 2048 KB option.
6. Select Exit twice.
7. Select Yes to save your new configuration and exit the menuconfig utility.
8. Back at the coreboot directory, type make to rebuild the coreboot.rom image file.

7.2 Intel® Binary Configuration Tool (Intel® BCT)

Intel® provides the Intel® Binary Configuration Tool (Intel® BCT) that lets you edit the FSP binary file delivered with the Intel® Bay Trail FSP Kit. Use the Intel® BCT for two purposes:

- To change the values in the Platform Configuration Database (PCD) within the FSP binary. For example, you might need to change the SMBus addresses for the DIMM slots on your production board if they are different from the default addresses specified in the FSP binary.
- To rebase the firmware binary. The code within the FSP binary is not relocatable code. Therefore, to locate it at an address in the boot loader other than its default address of 0xFFFC0000, you must rebase the FSP binary to the desired address.

Each Intel® FSP release is packaged with a platform-specific binary settings file (.bsf), that is a text file that represents the default PCD settings in the FSP binary file as it is provided by Intel®. Using the Intel® BCT, you can change the values of the settings listed in the .bsf file. The modified settings are saved in an as-built settings file (.absf). After modifying the settings, the Intel® BCT lets you patch those changes back into the binary image.

The Intel® BCT package is a standalone tool with its own user guide, and is not dependent on a particular CPU, chipset, or platform. Please refer to the Intel® BCT release package for further information on using this tool.