White Paper
Intel® Firmware Support Package
Internet of Things

Intel® Firmware Support Package for Intel® Architecture

Intel® Firmware Support Package provides Internet of Things developers the flexibility of usage-specific and royalty-free firmware solution for Internet of Things devices based on Intel® Architecture.

Executive Summary

This paper introduces the rationale behind Intel® Firmware Support Package (Intel® FSP), its development options, and an illustration of a workflow that shows how to integrate Intel® FSP with a sample firmware stack[1]: coreboot*, which is an open source firmware stack. Readers can experiment and apply the concepts presented to their own firmware stack of choice.

With Intel® FSP and other reference source code releases, Intel promises developers greater freedom, flexibility, and scalability when designing Internet of Things (IoT) devices using Intel® Architecture.

"Intel® Firmware Support Package offers a firmware solution for Intel® Architecture that is scalable and easy to adopt, lowering the threshold towards the adoption of Intel® Architecture in the Internet of Things ecosystem"

Wong, Swee Heng
Intel Corporation

Sun, Jiming
Intel Corporation

Mahesh, Divya
Intel Corporation

Expanding the Ecosystem of Custom System Firmware for Intel® Architecture

Intel® FSP provides the firmware component for initializing essential and basic functions of Intel® processors and chipsets. It is designed to support all firmware stacks available in the market[2]. The size of the Intel® FSP binary is small, around 100 to 350 kilobytes, and it performs fast execution of around 100 to 300 milliseconds[3]. After Intel® FSP completes its initialization, the firmware stack performs the remaining functions of platform initialization such as device discovery, device initialization, booting an operating system (OS), etc. With Intel® FSP, Independent Software Vendors (ISV) and Independent BIOS Vendors (IBV) in the ecosystem can continue to provide services for system firmware customization and value-added feature development.

Benefits

Intel® FSP is scalable and easy to adopt; it lowers the threshold towards the adoption of Intel® Architecture in the Internet of Things (IoT) ecosystem. Intel® FSP is available publicly, and it is free to download; therefore, developers and hobbyists can quickly build a prototype and a basic firmware framework with ease. Once the interface to Intel® FSP is built inside a firmware stack, developers can use future releases of Intel® FSP binary and expect it to work with minimum porting effort. The most significant benefit is reducing the need to read the comprehensive firmware and BIOS programming guides that are typically a few hundred pages long.

Design Philosophy

Intel, like other silicon vendors, holds the key to the knowledge in initializing the silicon it produces. Without the help from silicon vendors, firmware engineers typically have to perform reverse engineering and use trial and

Table of Contents

error methods in order to get the basic silicon functions working. By providing a binary that performs the basic silicon initialization, Intel enables firmware engineers to focus on features that add value to the platform instead of basic silicon features that do not differentiate the product significantly.

Unique Requirements of IoT

Unlike firmware for PCs (BIOS or UEFI), IoT firmware frequently deals with a closed system that performs dedicated functions; therefore, an IoT firmware can be customized to reduce the size and optimized for performance. Sometimes, features in a PC firmware directly contradict the requirements of an IoT firmware; for example, if determinism and predictability are needed, then the intelligence of plug-and-play and the heuristic power management algorithm are in conflict. Therefore, there are cases in the IoT ecosystem where alternatives to PC firmware are needed. Intel® FSP provides the flexibility for developers to choose an alternative firmware stack.

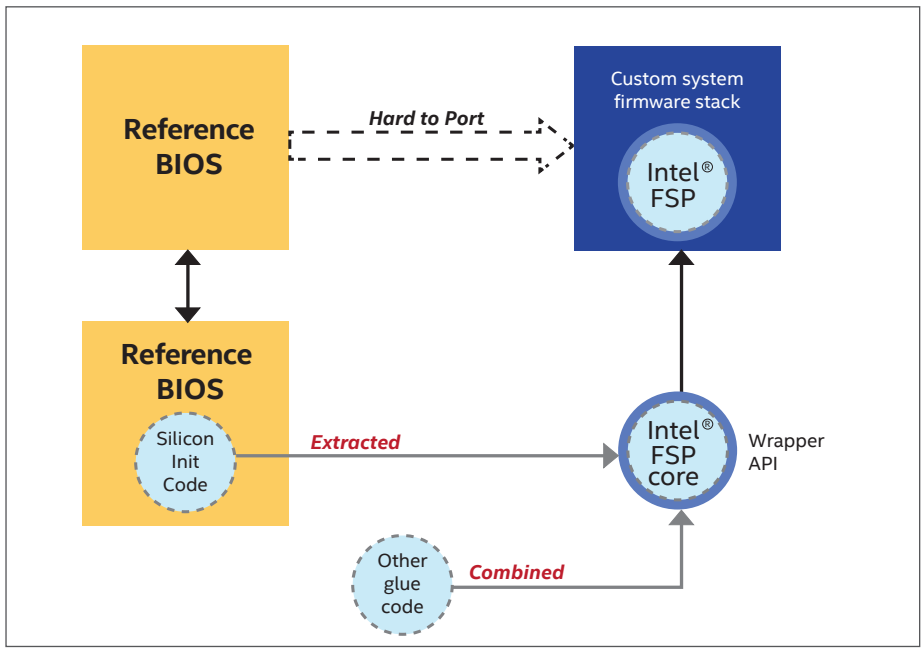
Strategic Moves

Intel® FSP is not only designed with a long-term focus of supporting processors suitable for IoT designs, but to also extend the reach to other Intel processor families, which include Intel® Core™, Intel® Atom™, and Intel® Xeon™ processors. A list of supported Intel processor families is available at www.intel.com/fsp.

Why Binary?

Intel understands that many developers in the early stages of designing a new platform need source code to help them debug issues in hardware and firmware. Even though Intel® FSP contains a binary component for integration, the source code can be made available for debugging purposes. The reason Intel® FSP provides a binary component is to protect certain Intel intellectual properties and also to avoid source code contamination associated with open source licensing agreements.

Nevertheless, Intel is working towards releasing as much source code as possible going forward. A binary component is still the best way to encapsulate the complex solution that developers



Intel® FSP simplifies firmware development. This diagram illustrates the relationship between Intel® FSP and its origin, Reference BIOS. The scattered subset form of the Reference BIOS makes it difficult to port the vital silicon initialization code to custom system firmware. Intel® FSP extracts the silicon initialization code and consolidates the code into the Intel® FSP binary, simplifying the development of custom system firmware.

may not necessarily need to bother about as long as the binary component does its job right.

Creating Development Options

Original equipment manufacturer (OEM) firmware designers have three firmware development options:

- **Self-Customization**

  With this option, OEMs perform their own self-customization using custom integrated packages purchased from ecosystem partners.

- **Engineering Services**

  Alternatively, OEMs can engage ecosystem partners to provide their engineering services to purchase partial or full firmware solutions.

- **Self-Integration**

  OEMs with experience in developing firmwares can perform self-integration. They can download an open source or use their own proprietary firmware stack and integrate Intel® FSP into it. However, they may receive limited support from the open source community.

Firmware Stack Sources

Firmware stacks can be obtained from the following sources:

- **Open Source Community**

  An example of an open source firmware stack is coreboot. Designed to be simple, flexible, and fast, coreboot separates the hardware initialization and the later boot logic. It uses the payload concept to boot different operating systems and applications.

  Obtain support from the coreboot community by subscribing to the coreboot mailing list (www.coreboot.org/Mailinglist) or reading the FAQ (www.coreboot.org/FAQ).

- **Intel® FSP Ecosystem Partners**

  Information regarding ecosystem partners (vendors) for Intel® FSP is available at www.intel.com/fsp.

Support Models

The following are the support models for Intel® FSP, coreboot, and proprietary firmware stacks.

- **Intel® FSP Support**

  Support for Intel® FSP is available from Intel® FSP Web site at www.intel.com/fsp

- **coreboot Support**

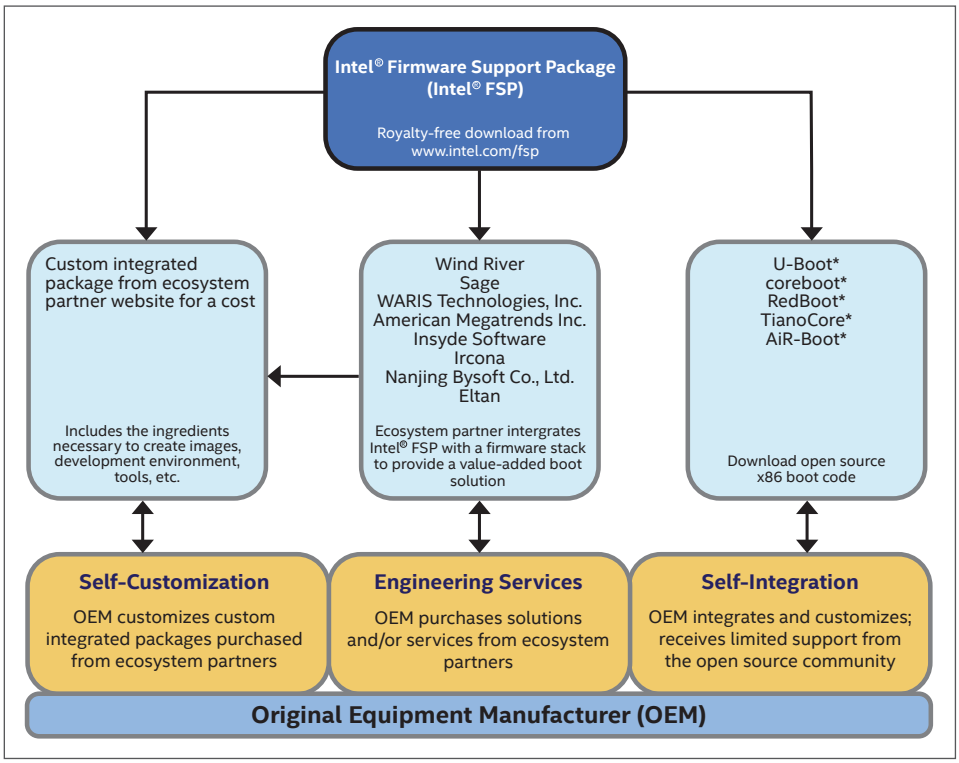
  Support for coreboot is available from the following sources:

  — coreboot Web site at www.coreboot.org

  — ecosystem partners, with engineering support that incurs costs

- **Ecosystem Partners' Firmware Stack Support**

  Support for ecosystem partners' firmware stack is available from the respective ecosystem partners. For example, VxWorks* support is available from Wind River.



Development Options with Intel® FSP. The Intel® FSP binary component is royalty-free, but Intel does not provide any engineering services for its integration into the customers' design.

Other Ingredients of Firmware Stacks Based on Intel® FSP

The following are the additional components for firmware stacks based on Intel® FSP.

Enable Pre-OS Graphics

Depending on platform requirements, there are several options to enable pre-OS graphics such as Graphics Output Protocol (GOP), Embedded Pre-OS Graphics (EPOG), and video BIOS (VBIOS). Some Intel® FSP releases have a built-in graphics initialization engine that enables a primitive connection between the display device and the framebuffer to display simple text and logos. Please refer to the Intel® Embedded Graphics Drivers FAQ at www-ssl.intel.com/content/www/us/en/intelligent-systems/intel-embedded-graphics-drivers/faq-bios-firmware.html for details.

Enable Trusted Execution Engine (TXE) in the Firmware
Application example on the Intel® Atom™ E3800 processor (formerly Bay Trail) product family platform

The Trusted Execution Engine (TXE) is a hardware-based security feature, which comprises an independent processor core embedded inside the Intel® Atom™ system-on-chip (SoC). It enables Secure Boot, which helps the computer resist tampering from external sources by ensuring only validated code is executed.

Enable Management Engine (ME) in the Firmware
Application example on the 4th generation Intel® Core™ processor (formerly Haswell) product family platform

The Management Engine (ME) is a hardware feature embedded in the Platform Controller Hub (PCH) of the 4th generation Intel® Core™ processor family platforms. It is a platform protection technology that provides administrators enhanced remote management on the platform.

Workflow Example to Integrate Intel® FSP into coreboot

This example describes the workflow to integrate Intel® FSP into coreboot. However, the same principle can be applied to other firmware stacks.

The following are the general steps to integrate Intel® FSP into coreboot.

1. Understand the architecture of the platform using the relevant documentation — for example, data sheets, product briefs, and white papers are available from the Intel Web site — when developing the firmware for a particular platform. Platform collaterals are available from the Intel Embedded Design Center (EDC) Web site at www.intel.com/content/www/us/en/intelligent-systems/embedded-design-center.html.
2. Download the Intel® FSP components, available for free, from the Intel® FSP Web site at www/intel.com/fsp.

   The Intel® FSP solution is available for the Windows* and Linux* operating systems, and includes a release note and a readme file. Review both documents for further details.
3. For the Windows release, double-click the executable file in the Windows environment to extract the downloaded package.

   For the Linux release, extract the downloaded file in the Linux environment.

   The extracted files are as follows:

   - a **binary file** with application program interfaces (APIs) to call Intel® FSP functions
   - an **integration guide** with instructions for the particular platform

     The integration guide is written for platform and system developers (which may include system BIOS developers, firmware stack developers, system integrators, and end-users). It describes in detail the steps required to integrate the Intel® FSP binary component for the particular platform into a firmware stack solution. The integration guide also includes links to related documents that may provide further technical support.
4. Download (git clone) the coreboot source distribution from www.coreboot.org for the coreboot source directory tree. For example, `git clone http://review.coreboot.org/p/coreboot`
5. Copy the Intel® FSP binary component (i.e., `FvFsp.bin`) into the appropriate directory inside the coreboot directory tree. An example of performing this step in a Linux terminal is shown below:

   ```
   #cp FvFsp.bin coreboot/src/mainboard/intel/cougarcanyon2
   ```

The actual path to place the Intel® FSP binary depends on the coreboot release. Please refer to the respective coreboot release information.

6. Update the microcode based on the processor type on the platform. Include the microcode into the following files in the `/cpu` directory:

   ```
   microcode_blob.h
   microcode_blob.inc
   ```

   Contact your Intel representative for the latest microcode patches.

7. Configure the payload using the following commands:

   ```
   #cd coreboot
   #make menuconfig
   ```

   For more details about building a payload, see www.coreboot.org/Libpayload

8. The data region of the Intel® FSP binary can be customized according to the platform environment, if needed, using the binary configuration tool (BCT).

   The base address of the Intel® FSP binary can also be rebased, if needed, using the BCT.

   Download the BCT from www.intel.com/fsp, and extract the downloaded file.

9. Configure Error Checking and Correction (ECC) support if the platform supports it. To determine if the platform supports ECC, refer to the Intel® FSP release note.

   a. Enable or disable ECC support using the BCT.

   b. Patch the Intel® FSP binary.

   c. Rebuild the firmware stack with the modified Intel® FSP binary.

Solving Technical Challenges

The following are some common technical challenges that you may encounter and their respective solutions.

1. Microcode Updates

Question:

Even though I know my CPUID is 0x306A9, I do not know where to find the required microcode update.

Solution:

Contact your Intel representative for the latest microcode. Alternatlvely, visit the Intel support portal at https://businessportal.intel.com, and search for the document if you know the document ID.

2. Microcode Patches

Question:

I do not know what I need to do with the raw text file for the microcode patches.

Solution:

Create two files from the raw text file, for example,

`m12306a9_00000017.h` and `m12306a9_00000017.inc`

```
cat name.txt | awk ' { print $1 ; print $2 ; print $3; print $4 } ' > name.h
```

```
cat name.txt | sed 's/,//' | awk ' { print " .long " $1 } '
```

3. ME Integration

Question:

This issue relates to integrating ME into the firmware:

I build coreboot by entering the following command in a Linux terminal:

```
#make
```

The final image (`coreboot.rom`) is created in the directory `/coreboot/build`. After programming the Intel® FSP binary, coreboot, and the payload into the SPI-1 partition of the firmware, the system does not boot. The POST code is 0000, which means nothing happened.

Solution:

Obtain the correct ME-enabled firmware from the Intel Web site (https://platformsw.intel.com/home.aspx), and then program it into the SPI-0 partition.

Conclusion

The adoption of the Internet of Things concept, with devices communicating and collaborating via the Internet, requires computing power in compact devices that are expected to perform specialized functions, such as digital signages, kiosks, or digital security surveillance. These specialized designs may divert from the traditional PC architecture; embedded developers seek a firmware solution for Internet of Things devices based on Intel® Architecture.

Cognizant of the needs of embedded developers in their designs for Internet of Things devices, Intel offers the Intel® FSP solution. With the Intel® FSP binary component performing the hardware initialization of Intel silicon, embedded developers have the flexibility to decide the best approach to develop a customized firmware for their designs. They can either engage the services of an Intel® FSP ecosystem partner to perform the customization and integration, or customize the firmware themselves by integrating the Intel® FSP binary into an open source firmware stack of their choice.

Intel® FSP has a long-term roadmap that focuses on supporting processors suitable for intelligent systems, such as Internet of Things devices. Embedded developers gain the benefit of greater scalability in their designs as newer and more capable processors will be supported, which in turn accelerates development time. In addition, they can maximize cost efficiency as the Intel® FSP solution is available for free to embedded developers.

For releases of Intel® Firmware Support Package for supported products, including relevant support documentation, visit **www.intel.com/fsp**

[1] A firmware stack is also known as a boot loader in some IoT designs.

[2] Some integration effort is needed.

[3] Actual size and execution speed depend on the product and debug option.

 Printed in Malaysia 0814/DRK/SH/PDF ♻ Please Recycle 331015-001EN