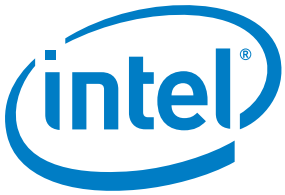

Intel® Firmware Support Package for Intel® Xeon® Processor D Product Family

Integration Guide

October 2015

Revision: 1.2

Contents

Tables

Revision History

Date	Revision	Description
October 2015	1.2	Updated for FSP Gold Release.
May 2015	1.1	Updated for FSP Beta Release. • Added VT-d MMIO Resource HOB GUID to Section 4.2 • Updated Section 3.3 and 5.1.2
March 2015	1.0	Initial Release for FSP Alpha Release.

§

1 Introduction

1.1 Purpose

The purpose of this document is to describe the steps required to integrate the Intel® Firmware Support Package (Intel® FSP) for Intel® Xeon® Processor D Product Family into a boot loader solution.

1.2 Intended Audience

This document is targeted at all platform and system developers who need to consume FSP binaries in their boot loader solutions. This includes, but is not limited to: system BIOS developers, boot loader developers, system integrators, as well as end users.

1.3 Related Documents

Table 1. Related Documents

Document	Document No./Location
UEFI Platform Initialization (PI) Specification*	http://www.uefi.org/specifications
Intel® Firmware Support Package: External Architecture Specification v.1.0	330554-001 http://www.intel.com/content/dam/www/public/us/en/documents/technical-specifications/fsp-architecture-spec.pdf
Binary Configuration Tool for Intel® Firmware Support Package	http://www.intel.com/fsp

1.4 Acronyms and Terminology

Table 2. Acronyms and Terminology

Acronym	Definition
API	Application Program Interface
BCT	Binary Configuration Tool
BSF	Boot Setting File
BWG	BIOS Writer's Guide
GCC	GNU Compiler Collection
HOB	Hand-Off-Block
PIC	Position Independent Code
T-SEG	Memory Reserved at the Top of Memory to be used as SMRAM
UPD	Updatable Product Data
VPD	Vital Product Data

§

2 FSP Overview

2.1 Technical Overview

The Intel® Firmware Support Package (Intel® FSP) provides chipset and processor initialization in a format that can easily be incorporated into many existing boot loaders.

The Firmware Support Package (FSP) will perform the necessary initialization steps as documented in the BIOS Writer's Guide (BWG) including initialization of the CPU, memory controller, chipset, and certain bus interfaces, if necessary.

The FSP is not a stand-alone boot loader; therefore it needs to be integrated into a host boot loader to carry out other boot loader functions, such as: initializing non-Intel components, conducting bus enumeration, and discovering devices in the system and all industry standard initialization.

The FSP binary can be integrated easily into many different boot loaders, such as Coreboot, etc. and also into the embedded OS directly.

Below are some required steps for the integration:

- **Customizing**

  The static FSP configuration parameters are part of the FSP binary and can be customized by external tools that will be provided by Intel.

- **Rebasing**

  The FSP is not Position Independent Code (PIC) and the whole FSP has to be rebased if it is placed at a location which is different from the preferred address during the build process.

- **Placing**

  Once the FSP binary is ready for integration, the boot loader build process needs to be modified to place this FSP binary at the specific rebasing location identified above.

- **Interfacing**

  The boot loader needs to add code to setup the operating environment for the FSP, call the FSP with the correct parameters, and parse the FSP output to retrieve the necessary information returned by the FSP.

2.2 FSP Distribution Package

The FSP distribution package contains the following:

- FSP Binary
- Reference code that illustrates the interfacing mechanism
- Vital Product Data (VPD)/ Updatable Product Data (UPD) Data structure definitions
- Boot Setting File (BSF) File
- Integration Guide

The FSP configuration utility called the Binary Configuration Tool (BCT) is available as a separate package.

§

3 FSP Integration

3.1 Assumptions Used in this Document

The FSP for the Intel® Xeon® Processor D Product Family is built with a preferred base address of **0xFFEB0000** and so the reference code provided in the document assumes that the FSP is placed at this base address during the final boot loader build. Users may rebase the FSP binary at a different location with Intel's Binary Configuration Tool (BCT) before integrating to the boot loader.

For other assumptions and conventions, please refer Sections 6.1 to 6.5 in the *Intel® Firmware Support Package: External Architecture Specification v.1.0*, Document# 330554-001.

3.2 FSP INFO Header

The FSP has an Information Header that provides critical information that is required by the boot loader to successfully interface with the FSP. The structure of the FSP Information Header is documented in the *Intel® Firmware Support Package*: *External Architecture Specification v.1.0*, Document# 330554-001.

3.3 FSP Image ID and Revision

The FSP information header contains an Image ID field and an Image Revision field that provide the identification and revision information of the FSP binary. It is important to verify these fields while integrating the FSP as Application Program Interface (API) parameters could change over different FSP IDs and revisions. The FSP API parameters documented in this integration guide are applicable for the Image ID and Revision specified as below.

The current FSP version is Alpha release. The ImageId string in the FSP information header is "**_BDX-DE_**" and the ImageRevision field is **0x00000301**.

3.4 FSP APIs

This release of the Intel® FSP for Intel® Xeon® Processor D Product Family supports the three APIs as documented in the *Intel® Firmware Support Package: External Architecture Specification v.1.0*, Document# 330554-001.

The FSP information header contains the address offset for these APIs.

The below sections will highlight any changes that are specific to this platform.

3.4.1 TempRamInit API

Refer to the ”TempRamInitEntry” chapter in the *Intel® Firmware Support Package: External Architecture Specification v.1.0*, Document# 330554-001 for the prototype, parameters and return value details for this API.

The TempRamInit API supports dynamic detection of MCU region size. When the MicrocodeRegionLength parameter for the TempRamInit API is set to **0xFFFFFFFF,** then the TempRamInit API will ignore the length parameter and will dynamically search for available microcode patches by looking for valid microcode headers at the end of the current microcode block. If the start of the next microcode block is **0xFFFFFFFF**, it indicates the end of Microcode update region and the search for the next available microcode terminates. Boot loaders wishing to use this feature should set the MicrocodeRegionlength to **0xFFFFFFFF** and make sure to append **0xFFFFFFFF** at the end of all the microcode update blocks when calling the TempRamInit API.

3.4.2 FspInit API

Refer to the "FspInitEntry" chapter in the *Intel® Firmware Support Package: External Architecture Specification v.1.0, Document#* 330554-001 for the prototype, parameters and return value details for this API.

3.4.3 NotifyPhase API

Refer to the "NotifyPhaseEntry" chapter in the *Intel® Firmware Support Package: External Architecture Specification v.1.0*, Document# 330554-001 for the prototype, parameters and return value details for this API.

3.5 Boot Flow

Refer to the “Boot Flow” chapter in the *Intel® Firmware Support Package: External Architecture Specification v.1.0*, Document# 330554-001 for Boot flow chart.

§

4 FSP Output

The FSP builds a series of data structures called the Hand-Off-Blocks (HOBs) as it progresses through initializing the silicon.

Refer to the *UEFI* Platform Initialization (PI) Specification - Volume 3: Shared Architectural Elements* specification for PI Architectural HOBs.

Refer to the "FSP Output" chapter in the *Intel® Firmware Support Package: External Architecture Specification v.1.0*, Document# 330554-001 for details about FSP Architectural HOBs.

The following section describes the HOBs not covered in the two specifications above.

4.1 SMRAM Resource Descriptor HOB

The FSP will report the system SMRAM T-SEG range through a generic resource HOB if T-SEG is enabled. The owner field of the HOB identifies the owner as T-SEG.

```
#define FSP_RESERVED_MEMORY_RESOURCE_HOB_TSEG_GUID \

{ 0xd038747c, 0xd00c, 0x4980, { 0xb3, 0x19, 0x49, 0x01, 0x99, 0xa4, 0x7d,
0x55 } }
```

4.2 VT-d Resource Descriptor HOB

The FSP will report the VT-d MMIO range through a generic resource HOB. The owner field of the HOB identifies the owner as VT-d.

```
#define FSP_RESERVED_MEMORY_RESOURCE_HOB_VTD_GUID \

{ 0x2e128805, 0x6c28, 0x42b1, { 0xa3, 0xad, 0xfe, 0x75, 0x46, 0x32, 0x6d,
0x9e } }
```

§

5 FSP Configuration Firmware File

The FSP binary contains a configurable data region which will be used by the FSP during the initialization. The configurable data region has two sets of data

VPD – Vital Product Data, which can only be configured statically

UPD – Updatable Product Data, which can be configured statically for default values, but also can be overridden during boot at runtime.

Both the VPD and the UPD parameters can be statically customized using a separate tool called the Binary Configuration Tool (BCT) as explained in the tools section. The tool will use a Boot Setting File (BSF) to understand the layout of the configuration region within the FSP.

In addition to static configuration, the UPD data can be overridden by the boot loader during runtime. The UPD data is organized as a structure. The FspInit API parameter includes an `UpdDataRgnPtr` pointer which can be initialized to point to the UPD data structure. If this pointer is initialized to NULL when calling the FspInit API, the FSP will use the default built-in UPD configuration data in the FSP binary. However, if the boot loader wishes to override any of the UPD parameters, it has to copy the whole UPD structure from flash to memory, override the parameters and initialize the `UpdDataRgnPtr` pointer to the address of the UPD structure with updated data in memory and call FspInit API. The FSP will use this data structure instead of the default configuration region data for platform initialization. The UPD data structure pointed by pointer `UpdDataRgnPtr` is a project specific structure.

When calling the FspInit API, the stack is in temporary memory where the UPD data structure is copied, updated, and passed to the FSP API. When permanent memory is initialized, the FSP will set up a new stack in the permanent memory and tear down the temporary memory. However, the FSP will save the whole boot loader temporary memory region in a GUID HOB. If the boot loader wishes to access the old data in the temporary memory, it can be done by parsing the HOB to retrieve the previous temporary memory data.

Note: The migrated temporary memory contains an identical copy of the original data. If pointers are stored in this region, they need to be fixed to point to the new migrated region before using.

Note: To update these configuration options statically using the BCT, a BSF file will be required. This file contains the detailed information on all configurable options, including description, help information, valid value range and the default value. Refer to the **BROADWELLDE_FSP.bsf** file in the release package for more information.

5.1 VPD/UPD Data Structure

The VPD/UPD data structure and related structure definitions are provided in the **fsp_vpd.h** file in the release package. The basic information for each option is

provided in the BCT configuration file **BROADWELLDE_FSP.bsf**. The user can use the BCT tool to load this BSF file to get the detailed configuration option information.

5.1.1 VPD Data Region

This VPD data region (`VPD_DATA_REGION`) can only be configured statistically by the BCT tool, and only very limited options in this region can be configured. Most of the configurable options are provided in the UPD data region.

Additional information follows for some of the fields in `VPD_DATA_REGION`.

PcdVpdRegionSign

This field is not an option and is a signature for the VPD data region. It can be used by the boot loader to validate the VPD region. This field will not change across different FSP releases for the same silicon set. For the Intel® FSP for Intel® Xeon® Processor D Product Family, this field is "**_BDX-DE_**", same as the FSP image ID.

PcdImageRevision

This field is not an option and is a revision ID for the FSP release. It can be used by the boot loader to validate the VPD/UPD region. If the value in this field is changed for an FSP release, the boot loader should not assume the same layout for the `UPD_DATA_REGION/VPD_DATA_REGION` data structure. Instead it should use the new **fsp_vpd.h** from the FSP release package.

PcdUpdRegionOffset

This field is not an option and contains the offset of the UPD data region within the FSP release image. The boot loader can use it to find the location of `UPD_DATA_REGION`.

PcdFspReservedMemoryLength

This option is used to specify the reserved memory size for the FSP usage. FSP will consume certain memory resource during the initialization, and this memory range must be reserved. This range will be reported through the HOB described in *Intel® Firmware Support Package: External Architecture Specification v.1.0*, Document# 330554-001, Section 7.2. In most cases, it does not need to be changed.

5.1.2 UPD Data Region

This UPD data region (`UPD_DATA_REGION`) can not only be configured statistically by the BCT tool in the same way as VPD data region, but also can be overridden by the boot loader at runtime. This provides more flexibility for the boot loader to customize these options dynamically as needed.

Additional information follows for some of the fields in `UPD_DATA_REGION`.

Signature

This field is not an option and is a signature for the UPD data region. It can be used by boot loader to validate the UPD region. The boot loader should never override this field. For the FSP of the Intel® Xeon® Processor D Product Family, this field is "**BDX-DE_U**".

SerialPortType

Debug serial port resource type. Select 'None' to have FSP generate no output. Select 'I/O' to have FSP generate output via a legacy I/O output (i.e. 0x2f8/0x3f8).

SerialPortAddress

16550 compatible serial port resource base address. (I/O or MMIO base address). Valid range: **0x00000000 ~ 0xFFFFFFFF**.

SerialPortConfigure

Select 'Yes' to have FSP configure SOC integrated UART.

SerialPortBaudRate

If '**Configure Serial Port**' is set to '**Yes**', this will be the baud rate that the UART located at '**Serial Port Base**' is configured to use.

Baud rate selection is varied depended on SoC stepping. Default baud rate is set to **19200BPS** which is supported in all stepping. Boot loader designer must consider a single baud rate to be configured between boot loader and FSP.

SerialPortControllerInit0

Select '**Yes**' to have FSP initialize this controller.

Note: If '**Yes**' is selected, this controller will be mapped to the legacy IO port **0x3f8**.

SerialPortControllerInit1

Select '**Yes**' to have FSP initialize this controller.

Note: If '**Yes**' is selected, this controller will be mapped to the legacy IO port **0x2f8**.

ConfigIOU1_PciPort3

Set your bifurcation option for IOU1 port.

ConfigIOU2_PciPort1

Set your bifurcation option for IOU2 port.

PowerStateAfterG3

S0 = System will return to S0 state (boot) after power is re-applied. S5 = System will return to the S5 state (except if it was in S4, in which case it will return to S4). In the S5 state, the only enabled wake event is the Power Button or any enabled wake event that was preserved through the power failure.

PchPciPort1

Enable/Disable PCH PCIe* Ports. Port bifurcation options set in Fuse Straps decide the final enabling of PCIe* ports.

PchPciPort2

Enable/Disable PCH PCIe* Ports. Port bifurcation options set in Fuse Straps decide the final enabling of PCIe* ports.

PchPciPort3

Enable/Disable PCH PCIe* Ports. Port bifurcation options set in Fuse Straps decide the final enabling of PCIe* ports.

PchPciPort4

Enable/Disable PCH PCIe* Ports. Port bifurcation options set in Fuse Straps decide the final enabling of PCIe* ports.

PchPciPort5

Enable/Disable PCH PCIe* Ports. Port bifurcation options set in Fuse Straps decide the final enabling of PCIe* ports.

PchPciPort6

Enable/Disable PCH PCIe* Ports. Port bifurcation options set in Fuse Straps decide the final enabling of PCIe* ports.

PchPciPort7

Enable/Disable PCH PCIe* Ports. Port bifurcation options set in Fuse Straps decide the final enabling of PCIe* ports.

PchPciPort8

Enable/Disable PCH PCIe* Ports. Port bifurcation options set in Fuse Straps decide the final enabling of PCIe* ports.

HotPlug_PchPciPort1

Enable/Disable the `HotPlug` for PCH PCIe* Ports.

HotPlug_PchPciPort2

Enable/Disable the `HotPlug` for PCH PCIe* Ports.

HotPlug_PchPciPort3

Enable/Disable the `HotPlug f`or PCH PCIe* Ports.

HotPlug_PchPciPort4

Enable/Disable the `HotPlug` for PCH PCIe* Ports.

HotPlug_PchPciPort5

Enable/Disable the `HotPlug` for PCH PCIe* Ports.

HotPlug_PchPciPort6

Enable/Disable the `HotPlug` for PCH PCIe* Ports.

HotPlug_PchPciPort7

Enable/Disable the `HotPlug` for PCH PCIe* Ports.

HotPlug_PchPciPort8

Enable/Disable the `HotPlug` for PCH PCIe* Ports.

Ehci1Enable

Enable or disable the EHCI controller at **00.1d.00**.

Ehci2Enable

Enable or disable the EHCI controller at **00.1a.00**.

HyperThreading

Enable or disable the Intel® Hyper-Threading Technology.

DebugOutputLevel

Select the output level of debug print.

TcoTimerHaltLock

Enable FSP to halt and lock the TCO timer.

TurboMode

Enable or disable processor Turbo Mode.

BootPerfMode

Select the performance state that should be set by FSP before OS hand-off.

PciePort1aAspm, PciePort1bAspm, PciePort3aAspm, PciePort3bAspm, PciePort3cAspm, PciePort3dAspm, PchPciePort1Aspm, PchPciePort2Aspm, PchPciePort3Aspm, PchPciePort4Aspm, PchPciePort5Aspm, PchPciePort6Aspm, PchPciePort7Aspm, PchPciePort8Aspm

ASPM configuration for PCI-Express Root Ports.

DFXEnable

Enable this option to allow DFX Lock Bits to remain clear.

ThermalDeviceEnable

Enable or disable the PCH Thermal Device (D31:F6).

MemEccSupport

Enable/Disable DDR ECC Support.

MemDdrMemoryType

Select the memory type supported by this platform.

MemRankMultiplication

Force the Rank Multiplication factor for LRDIMM.

MemRankMarginTool

Run the Rank Margin Tool after memory training.

MemScrambling

Enable data scrambling.

MemRefreshMode

Self-refresh mode.

MemMcOdtOverride

Select MC ODT Mode.

MemCAParity

Enable/Disable DDR4 Command Address Parity.

MemThermalThrottling

Configure Thermal Throttling Mode.

MemPowerSavingsMode

Configure CKE and related Memory Power Savings Features.

MemElectricalThrottling

Configure MEMHOT Input and Output Modes.

MemPagePolicy

Select Page Policy.

MemSocketInterleaveBelow4G

Splits the 0 – 4GB address space between two sockets, so that both sockets get a chunk of local memory below 4 GB.

MemChannelInterleave

Select Channel Interleaving setting.

MemRankInterleave

Select Rank Interleaving setting.

MemDownEnable

Select 'Yes' if memory is down.

Note: If '**Yes**' is selected, at least one of the following SPD data pointers must also be provided.

MemDownCh0Dimm0SpdPtr

The pointer to the SPD data for Channel 0, DIMM 0. Specify 0x00000000 if this DIMM is not present.

MemDownCh0Dimm1SpdPtr

The pointer to the SPD data for Channel 0, DIMM 1. Specify 0x00000000 if this DIMM is not present.

MemDownCh1Dimm0SpdPtr

The pointer to the SPD data for Channel 1, DIMM 0. Specify 0x00000000 if this DIMM is not present.

MemDownCh1Dimm1SpdPtr

The pointer to the SPD data for Channel 1, DIMM 1. Specify 0x00000000 if this DIMM is not present.

MemFastBoot

Enable/Disable Fast Boot.

pam0_hienable

Configure how reads and writes of **F0000h-FFFFFh** handled.

pam1_loenable

Configure how reads and writes of **C0000h-C3FFFh** handled.

pam1_hienable

Configure how reads and writes of **C4000h-C7FFFh** handled.

pam2_loenable

Configure how reads and writes of **C8000h-CBFFFh** handled.

pam2_hienable

Configure how reads and writes of **CC000h-CFFFFh** handled.

pam3_loenable

Configure how reads and writes of **D0000h-D3FFFh** handled.

pam3_hienable

Configure how reads and writes of **D4000h-D7FFFh** handled.

pam4_loenable

Configure how reads and writes of **D8000h-DBFFFh** handled.

pam4_hienable

Configure how reads and writes of **DC000h-DFFFFh** handled.

pam5_loenable

Configure how reads and writes of **E0000h-E3FFFh** handled.

pam5_hienable

Configure how reads and writes of **E4000h-E7FFFh** handled.

pam6_loenable

Configure how reads and writes of **E8000h-EBFFFh** handled.

pam6_hienable

Configure how reads and writes of **EC000h-EFFFFh** handled.

MemAdr

Asynchronous DRAM Refresh Setting. Set to 'Enabled' if DIMMs are battery-backed or if the platform implements saving to some other non-volatile storage medium. Set to 'Enabled (NVDIMMs)' if NVDIMMs will be used on the platform.

MemBlockScTrafficOnAdr

Enable/Disable Blocking all PCIe/South Complex Traffic on ADR.

Note: Feature only works on V-0 and later stepping.

MemPlatformReleaseAdrClampsPort

Specify the I/O port which should be written to when CKE/DDR Reset clamps should be released. Specify '0' to skip.

MemPlatformReleaseAdrClampsAnd

Specify the value that should be ANDed with the value read from the clamp release I/O port.

MemPlatformReleaseAdrClampsOr

Specify the value that should be ORed with the value read from the clamp release I/O port.

§

Appendix A FSP Sample File List

To simplify the FSP integration, a set of FSP interface sample code files are provided under BSD license to assist the boot loader development. These files show examples for boot loader on how to interface with FSP APIs and how to consume the data returned from the FSP.

These files have been tested within Coreboot using GNU Compiler Collection (GCC) v4.8.1 tool chain. Intel recommends using this GCC version for boot loader development.

fsp.h

Contain all FSP header files.

fsp_api.h

Contain the declarations for the FSP API prototype and the input parameter data structures.

fsp_bootmode.h

Contain the UEFI compatible FSP boot mode definitions.

fsp_ffs.h

Contain the UEFI PI firmware file system related data type and constant definitions.

fsp_fv.h

Contain the UEFI PI firmware volume related data type and constant definitions.

fsp_hob.h

Contain the UEFI PI HOB related data type and constant definitions.

fsp_infoheader.h

Contain the FSP information header data structure definitions.

fsp_platform.h

Contain the platform-specific data structures.

fsp_support.c

Contain the FSP-specific sample support functions, including the API wrapper, HOB parsing, etc.

fsp_support.h

Contain the FSP-specific sample support prototype declarations.

fsp_types.h

Contain the UEFI and FSP basic data types and constant definitions.

fsp_vpd.h

Contains the configuration data structure definitions of `UPD_DATA_REGION`, `VPD_DATA_REGION`, and related other definitions.

§

Appendix B FSP Modified Scratchpad Registers

The Intel® Xeon® Processor D Product Family SoC provides 24 scratchpad DWORD registers in Ubox registers. Please refer to *Broadwell-DE SoC External Design Specification (EDS), Volume Two: Core and Uncore Registers, Rev.1.0* and Table 3 for to determine if registers need to be modified in FSP.

Table 3. Scratchpad Registers

Register Name	Location	Modified in FSP
BIOSScratchpad0	Uncore:D16:F7:R:40h	Yes
BIOSScratchpad1	Uncore:D16:F7:R:44h	Yes
BIOSScratchpad2	Uncore:D16:F7:R:48h	Yes
BIOSScratchpad3	Uncore:D16:F7:R:4Ch	Yes
BIOSScratchpad4	Uncore:D16:F7:R:50h	No
BIOSScratchpad5	Uncore:D16:F7:R:54h	Yes
BIOSScratchpad6	Uncore:D16:F7:R:58h	Yes
BIOSScratchpad7	Uncore:D16:F7:R:5Ch	Yes
BIOSNonStickyScratchpad0	Uncore:D16:F7:R:60h	Yes
BIOSNonStickyScratchpad1	Uncore:D16:F7:R:64h	Yes
BIOSNonStickyScratchpad2	Uncore:D16:F7:R:68h	Yes
BIOSNonStickyScratchpad3	Uncore:D16:F7:R:6Ch	Yes
BIOSNonStickyScratchpad4	Uncore:D16:F7:R:70h	Yes
BIOSNonStickyScratchpad5	Uncore:D16:F7:R:74h	Yes
BIOSNonStickyScratchpad6	Uncore:D16:F7:R:78h	Yes
BIOSNonStickyScratchpad7	Uncore:D16:F7:R:7Ch	Yes
BIOSNonStickyScratchpad8	Uncore:D16:F7:R:80h	Yes
BIOSNonStickyScratchpad9	Uncore:D16:F7:R:84h	Yes
BIOSNonStickyScratchpad10	Uncore:D16:F7:R:88h	No
BIOSNonStickyScratchpad11	Uncore:D16:F7:R:8Ch	No
BIOSNonStickyScratchpad12	Uncore:D16:F7:R:90h	Yes

Register Name	Location	Modified in FSP
BIOSNonStickyScratchpad13	Uncore:D16:F7:R:94h	Yes
BIOSNonStickyScratchpad14	Uncore:D16:F7:R:98h	Yes
BIOSNonStickyScratchpad15	uncore:D16:F7:R:9Ch	Yes

§

Appendix C FSP Consumed Memory Address Spaces

In order to complete the SoC initialization, FSP will assign some memory address spaces for SoC devices. Boot loader designer must not assign these spaces to others. In addition, report them as system reserved memory to OS.

Table 4. FSP Consumed Memory Address Spaces

Memory Address Space	Occupancy
0x80000000 – 0x9FFFFFFF	MMCFG Region MMIO
0xFED00000 – 0xFED003FF	HPET Registers MMIO
0xFED1C000 – 0xFED1FFFF	Chipset Configuration Registers MMIO

§