



Intel® Firmware Support Package (Intel® FSP) for the Elkhart Lake Platform

Integration Guide

November 2022



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.

[When the doc contains software source code for a special or limited purpose (such as informational purposes only), use the conditionalized Software Disclaimer tag. Otherwise, use the generic software source code disclaimer from the Legal page and include a copy of the software license or a hyperlink to its permanent location.]

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.

© Intel Corporation

Contents

1.0	Introduction.....	7
1.1	Purpose.....	7
1.2	Intended Audience.....	7
1.3	Acronyms and Terminology.....	7
1.4	Related Documents.....	8
2.0	Intel® Firmware Support Package (Intel® FSP) Overview	9
2.1	Technical Overview.....	9
2.2	FSP Distribution Package.....	10
2.2.1	Package Layout.....	10
3.0	Intel® Firmware Support Package (Intel® FSP) Integration	11
3.1	Assumptions used in this Document.....	11
3.2	Boot Flow	11
3.3	FSP Info Header	11
3.4	FSP Image ID and Revision.....	11
3.5	FSP Global Data.....	12
3.6	FSP APIs.....	12
3.6.1	TempRamInit API.....	12
3.6.2	FspMemoryInit API.....	13
3.6.3	TempRamExit API.....	14
3.6.4	FspSiliconInit API.....	14
3.6.5	NotifyPhase API.....	15
3.7	Memory Map.....	17
4.0	Intel® Firmware Support Package (Intel® FSP) Porting Recommendation.....	18
4.1	Locking PAM Register.....	18
4.2	Locking SMRAM Register	18
4.3	Locking SMI Register.....	18
4.4	Verify Settings for Your Platforms.....	19
4.5	FSP Status Reset Required.....	19
5.0	Intel® Firmware Support Package (Intel® FSP) UPD Porting Guide.....	20
6.0	Intel® Firmware Support Package (Intel® FSP) FSP Output.....	22
6.1	SMRAM Resource Descriptor HOB.....	22
6.2	SMBIOS Info HOB.....	22
6.3	CHIPSETINIT Info HOB.....	25
6.4	HOB Usage Info HOB	26
7.0	Intel® Firmware Support Package (Intel® FSP) FSP PostCode	27
7.1	PostCode Info.....	28
7.1.1	TempRamInit API Status Codes (0xFxxx)	28



7.1.2	FSPMemoryInit API Status Codes (0xDxxx).....	28
7.1.3	TempRamExit API Status Codes (0xBxxx).....	33
7.1.4	FSPSiliconInit API Status Codes (0x9xxx).....	33
7.1.5	NotifyPhase API Status Codes (0x6xxx).....	36
8.0	Intel® Firmware Support Package (Intel® FSP) FSP Dispatch Mode Support	37
8.1	Integration Notes.....	37

Figures

Figure 1.	System Memory Map.....	17
-----------	------------------------	----

Tables

Table 1.	Acronyms and Terminology	7
Table 2.	Reference Documents	8
Table 3.	TempRamInit API.....	13
Table 4.	TempRamExit API.....	14
Table 5.	Address	19
Table 6.	FSP Status Reset.....	19
Table 7.	UPD Porting Guide for Recommendation Values.....	20
Table 8.	FSP PostCode.....	27
Table 9.	TempRamInit API Status Codes	28
Table 10.	FSPMemoryInit API Status Codes	28
Table 11.	TempRamExit API Status Codes	33
Table 12.	FSPSiliconInit API Status Codes.....	33
Table 13.	NotifyPhase API Status Codes	36



Revision History

Date	Revision	Description
November 2022	1.0	Initial release.

§

1.0 Introduction

1.1 Purpose

This purpose of this document is to describe the steps required to integrate the Intel® FirmwareSupport Package (FSP) into a boot loader solution. It supports ElkhartLake platforms with ElkhartLake processor and ElkhartLake Platform Controller Hub (PCH).

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 Acronyms and Terminology

Table 1. Acronyms and Terminology

Acronym	Definition
BCT	Binary Configuration Tool
BSF	Boot Setting File
BSP	Boot Strap Processor
BWG	BIOS Writer's Guide
CAR	Cache As Ram
CRB	Customer Reference Board
FIT	Firmware Interface Table
FSP	Firmware Support Package
FSP API	Firmware Support Package Interface
FW	Firmware
PCH	Platform Controller Hub
PMC	Power Management Controller
SBSP	System BSP
SMI	System Management Interrupt
SMM	System Management Mode
SPI	Serial Peripheral Interface

Acronym	Definition
TSEG	Memory Reserved at the Top of Memory to be used as SMRAM
UPD	Updatable Product Data
IED	Intel Enhanced Debug
GTT	Graphics Translation Table
BDSM	Base Data Of Stolen Memory
PMRR	Protected Memory Range Reporting
IOT	Internal Observation Trace
MOT	Memory Observation Trace
DPR	DMA Protected Range
REMAP	Remapped Memory Area
TOLUD	Top of Low Usable Memory
TOUUD	Top of Upper Usable Memory

1.4 Related Documents

Table 2. Reference Documents

Document	Document No./Location
<i>Platform Initialization (PI) Specification v1.4</i>	http://www.uefi.org/specifications
<i>UEFI Specification</i>	http://www.uefi.org/specifications
<i>Intel® Firmware Support Package: External Architecture Specification (EAS) v2.0</i>	http://www.intel.com/content/dam/www/public/us/en/documents/technical-specifications/fsp.pdf
<i>Boot Setting File Specification (BSF) v1.0</i>	https://www.intel.com/content/www/us/en/content-details/671444/boot-setting-file-specification-release-1-0.html?wapkw=boot%20setting%20file
<i>Binary Configuration Tool for Intel® Firmware Support Package</i>	http://www.intel.com/fsp

§

2.0 Intel® Firmware Support Package (Intel® FSP) Overview

2.1 Technical Overview

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

The FSP will perform the necessary initialization steps as documented in the BWG including initialization of the CPU, memory controller, chipset, and certain bus interfaces, if necessary.

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, EDKII and others 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 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 correct parameters and parse the FSP output to retrieve the necessary information returned by the FSP.

2.2 FSP Distribution Package

The Intel® FSP distribution package contains the following:

- FSP binary
- FSP Integration guide (this document)
- Data structure Header File
- BSF Configuration File (BSF)

The FSP configuration utility called BCT is available as a separate package. It can be downloaded from link mentioned in Section 1.3.

2.2.1 Package Layout

- Docs (Auto generated)
 - ElkhartLake_FSP_Integration_Guide.pdf
 - ElkhartLake_FSP_Integration_Guide.chm
- Include
 - [FsptUpd.h](#), [FspmUpd.h](#) [FspsUpd.h](#) (FSP UPD structure and related definitions)
 - [GpioSampleDef.h](#) (Sample enum definitions for GPIO table)
- ElkhartLakeFspBinPkg.dec (EDKII declaration file for package)
- Fsp.bsf (BSF file for configuring the data using BCT tool)
- Fsp.fd (FSP Binary)

3.0 Intel® Firmware Support Package (Intel® FSP) Integration

3.1 Assumptions used in this Document

The FSP for the ElkhartLake platform is built with a preferred base address given by PcdFspAreaBaseAddress 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 section 8 in the FSP External Architecture Specification version 2.0.

3.2 Boot Flow

Please refer Chapter 7 in the FSP External Architecture Specification version 2.0 for Boot flow chart.

3.3 FSP Info Header

The FSP has an Information Header that provides critical information that is required by the bootloader to successfully interface with the FSP. The structure of the FSP Information Header is documented in the FSP External Architecture Specification version 2.0 with a Header Revision of 3.

3.4 FSP Image ID and Revision

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 API parameters could change over different FSP IDs and revisions. All the FSP FV segments(FSP-T, FSP-M, and FSP-S) must have same FSP Image ID and revision number, using FV segments with different revision numbers in a single FSP image is not valid. The FSP API parameters documented in this integration guide are applicable for the Image ID and Revision specified as below.

The FSP ImageId string in the FSP information header is given by PcdFspImageIdString and the ImageRevision field is given by SiliconInitVersionMajor|Minor|FspVersionRevision|FspVersionBuild (Ex:0x07020110).

3.5 FSP Global Data

FSP uses some amount of TempRam area to store FSP global data which contains some critical data like pointers to FSP information headers and UPD configuration regions, FSP/Bootloader stack pointers required for stack switching etc. HPET Timer register(2) PcdGlobalDataPointerAddress is reserved to store address of this global data, and hence boot loader should not use this register for any other purpose. If TempRAM initialization is done by boot loader, then HPET has to be initialized to the base so that access to the register will work fine

3.6 FSP APIs

This release of the ElkhartLake FSP supports the all APIs required by the FSP External Architecture Specification version 2.0. The FSP information header contains the address offset for these APIs. Register usage is described in the FSP External Architecture Specification version 2.0. Any usage not described by the specification is described in the individual sections below.

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

3.6.1 TempRamInit API

Please refer Chapter 8.5 in the FSP External Architecture Specification version 2.0 for complete details including the prototype, parameters and return value details for this API.

TempRamInit does basic early initialization primarily setting up temporary RAM using cache. It returns ECX pointing to beginning of temporary memory and EDX pointing to end of temporary memory + 1. The total temporary ram currently available is given by PcdTemporaryRamSize starting from the base address of PcdTemporaryRamBase. Out of the total temporary memory available, last PcdFspReservedBufferSize bytes of space reserved by FSP for TempRamInit if temporary RAM initialization is done by FSP and remaining space from TemporaryRamBase (ECX) to TemporaryRamBase+TemporaryRamSize-FspReservedBufferSize

(EDX) is available for both bootloader and FSP binary.

TempRamInit also sets up the code caching of the region passed CodeCacheBase and CodeCacheLength, which are input parameters to TempRamInitApi. If 0 is passed in for

CodeCacheBase, the base used will be 4 GB - 1 - length to be code cached instead of starting from CodeCacheBase.

Note: When programming MTRR CodeCacheLength will be reduced, if SKU LLC size is smaller than the requested.

It is a requirement for Firmware to have Firmware Interface Table (FIT), which contains pointers to each microcode update. The microcode update is loaded for all logical processors before reset vector. If more than microcode update for the CPU is present, the microcode update with the latest revision is loaded.

FSPT_UPD.MicrocodeRegionBase and FSPT_UPD.MicrocodeRegionLength are input parameters to TempRamInit API. If these values are 0, FSP will not attempt to update microcode. If a region is passed, then if a newer microcode update revision is in the region, it will be loaded by the FSP.

MTRRs are programmed to the default values to have the following memory map.

Table 3. TempRamInit API

Memory range	Cache Attribute
0xFE000000 - 0x00080000	Write back
CodeCacheBase - CodeCacheLength	Write protect

3.6.2 FspMemoryInit API

Please refer to Chapter 8.6 in the FSP external Architecture Specification version 2.0 for the prototype, parameters and return value details for this API.

The FspmUpdPtr is pointer to FSPM_UPD structure which is described in header file FspmUpd.h.

Boot Loader must pass valid CAR region for FSP stack use through FSPM_UPD.FspmArchUpd.StackBase and

FSPM_UPD.FspmArchUpd.StackSize UPDs.

The minimum FSP stack size required for this revision of FSP is 160KB, stack base is 0xFE017F00 by default.

The base address of HECI device (Bus 0, Device 22, Function 0) is required to be initialized prior to performing FspMemoryInit flow. The default address is programmed to 0xFED1A000.

Calculate memory map determining memory regions TSEG, IED, GTT, BDSM, ME stolen, Uncore PMRR, IOT, MOT, DPR, REMAP, TOLUD, TOUUD. Programming will be done at a different time.

3.6.3 TempRamExit API

Please refer to Chapter 8.7 in the FSP external Architecture Specification version 2.0 for the prototype, parameters and return value details for this API.

If Boot Loader initializes the Temporary RAM (CAR) and skip calling TempRamInit API, it is expected that boot-loader must skip calling this API and bootloader will tear down the temporary memory area setup in the cache and bring the cache to normal mode of operation.

This revision of FSP doesn't have any fields/structure to pass as parameter for this API. Pass Null for TempRamExitParamPtr.

At the end of TempRamExit the original code and data caching are disabled. FSP will reconfigure all MTRRs as described in the table below for performance optimization.

Table 4. TempRamExit API

Memory range	Cache Attribute
0x00000000 - 0x0009FFFF	Write back
0x000C0000 - Top of Low Memory	Write back
0xFF000000 - 0xFFFFFFFF (Flash region)	Write protect
0x1000000000 - Top of High Memory	Write back

3.6.4 FspSiliconInit API

Please refer to Chapter 8.8 in the FSP external Architecture Specification version 2.0 for the prototype, parameters and return value details for this API.

The FspUpdPtr is pointer to FSPS_UPD structure which is described in header file FspUpd.h.

It is expected that boot loader will program MTRRs for SBSP as needed after TempRamExit but before entering

FspSiliconInit. If MTRRs are not programmed properly, the boot performance might be impacted.

The region of 0x5_8000 - 0x5_8FFF is used by FspSilicionInit for starting APs. If this data is important to bootloader, then bootloader needs to preserve it before calling FspSilicionInit.

It is a requirement for bootloader to have Firmware Interface Table (FIT), which contains pointers to each microcode. The microcode is loaded for all cores before reset vector. If more than one microcode update for the CPU is present, the latest revision is loaded.

MicrocodeRegionBase and MicrocodeRegionLength are both input parameters to TempRamInit and UPD for SiliconInit API. UPD has priority and will be searched for a later revision than TempRamInit. If MicrocodeRegionBase and MicrocodeRegionLength values are 0, FSP will not attempt to update the microcode. If a microcode region is passed, and if a later revision of microcode is present in this region, FSP will load it.

FSP initializes PCH audio including selecting HD Audio verb table and initializes Codec.

PCH required initialization is done for the following HECI, USB, HSIO, Integrated Sensor Hub, Camera, PCI Express, Vt-d.

FSP initializes CPU features: XD, VMX, AES, IED, HDC, x(2) Apic, Intel® Processor Trace, Three strike counter, Machine check, Cache pre-fetchers, Core PMRR, Power management.

Initializes HECI, DMI, Internal Graphics. Publish EFI_PEI_GRAPHICS_INFO_HOB during normal boot but this HOB will not be published during S3 resume as FSP will not launch the PEI Graphics PEIM during S3 resume.

Programs SA Bars: MchBar, DmiBar, EpBar, GdxcBar, EDRAM (if supported). Please refer to section 2.8 (Memory Map) for the corresponding Bar values. GttMmadr (0xDF000000) and GmAdr (0xC0000000) are temporarily programmed and cleared after use in FSP.

3.6.5 NotifyPhase API

Please refer to Chapter 8.9 in the FSP External Architecture Specification version 2.0 for the prototype, parameters, and return value details for this API.

3.6.5.1 PostPciEnumeration Notification

This phase EnumInitPhaseAfterPciEnumeration is to be called after PCI enumeration but before execution of third-party code such as option ROMs. Currently, nothing is done in this phase, but in the future updates, programming may be done in this phase.

3.6.5.2 ReadyToBoot Notification

This phase EnumInitPhaseReadyToBoot is to be called before giving control to boot. It includes some final initialization steps recommended by the BWG, including power management settings, Send ME Message EOP (End of Post).

3.6.5.3 EndOfFirmware Notification

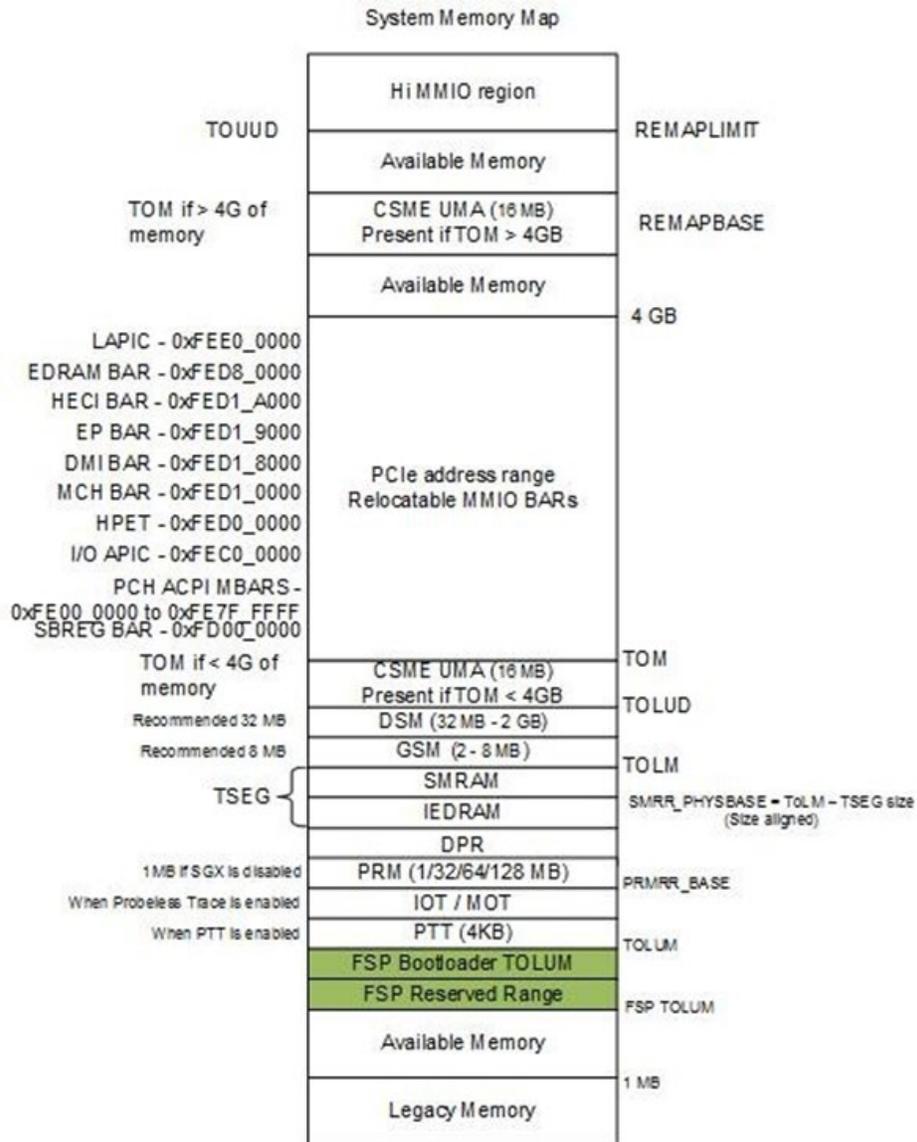
This phase EnumInitEndOfFirmware is to be called before the firmware/preboot environment transfers management of all system resources to the OS or next level execution environment. It includes final locking of chipset registers

§

3.7 Memory Map

The diagram below represents the memory map allocated by FSP including the FSP specific regions.

Figure 1. System Memory Map



§

4.0 Intel® Firmware Support Package (Intel® FSP) Porting Recommendation

Here are some notes or recommendation when porting with FSP.

4.1 Locking PAM Register

FSP 2.0 introduced EndOfFirmware Notify phase callback which is a recommended place for locking PAM registers so FSP by default implemented this way. If it is still too early to lock PAM registers then the PAM locking code inside FSP can be disabled by UPD -> FSP_S_TEST_CONFIG -> SkipPamLock or SA policy -> _SI_PREMEM_POLICY_STRUCT -> SA_MISC_PEI_CONFIG -> SkipPamLock, and platform or wrapper code should do the PAM locking right before booting OS (so do it outside FSP instead) by programming one PCI config space register as below.

This PAM locking step has to be applied in all boot paths including S3 resume. To lock PAM register:

MmioOr32 (B0: D0: F0: Register 0x80, BIT0)

4.2 Locking SMRAM Register

Since SMRAM locking is recommended to be locked before any 3rd party OpROM execution and highly depending on platform code implementation, the FSP code by default will not lock it. The platform or FSP Wrapper code should lock SMRAM by below programming step before any 3rd party OpRom execution (and should be locked in S3 resume right before OS waking vector).

PciOr8 (B0: D0: F0: Register 0x88, BIT4); Note: it must be programmed by CF8/CFC Standard PCI access mechanism. (MMIO access will not work)

4.3 Locking SMI Register

Global SMI bit is recommended to be locked before any 3rd party OpROM execution and highly depending on platform code implementation after SMM configuration. FSP by default will not lock it. Boot loader is responsible for locking below registers after SMM configuration.

Set AcpiBase + 0x30[0] to 1b to enable global SMI. Set PMC PCI offset A0h[4] = 1b to lock SMI.

4.4 Verify Settings for Your Platforms

PMC PciCfgSpace is not PCI compliant. FSP will hide the PMC controller to avoid external software or OS from corrupting the BAR addresses. FSP will program the PMC controller IO and MMIO BAR's with below addresses. Please use this address in the wrapper code instead of reading from PMC controller.

Table 5. Address

Register	Values
ABASE	0x1800
PWRMBASE	0xFE000000
PCIEXBAR_BASE_ADDRESS	0xE0000000

NOTES:

- Boot Loader can use different value for PCIEXBAR_BASE_ADDRESS either by modifying the UPD (under FSP-T) or by overriding the PCIEXBAR (B0:DO:F0:R60h) before calling FspMemoryInit API.
- Boot Loader should avoid using conflicting address when reprogramming PCIEXBAR_BASE_ADDRESS than the recommended one.

4.5 FSP Status Reset Required

As per FSP External Architecture Specification version 2.0, any reset required in the FSP flow will be reported as return status FSP_STATUS_RESET_REQUIREDx by the API. It is the bootloader's responsibility to reset the system according to the reset type requested.

The table below specifies the return status returned by FSP API and the requested reset type.

Table 6. FSP Status Reset

FSP_STATUS_RESET_REQUIRED Code	Reset Type Requested
0x40000001	Cold Reset
0x40000002	Warm Reset
0x40000003	Global Reset - Puts the system to Global reset through Heci or Full Reset through PCH
0x40000004	Reserved
0x40000005	Reserved
0x40000006	Reserved
0x40000007	Reserved
0x40000008	Reserved

5.0 Intel® Firmware Support Package (Intel® FSP) UPD Porting Guide

Table 7. UPD Porting Guide for Recommendation Values

UPD	Dependency	Description	Value
EnableSgx	ElkhartLake Platform	Temporary workaround	2
CstateLatencyControl1Irtl	Server platform	Server platform should have different setting	0x6B
PchPcieHsioRxSetCtleEnable	Board design	Different board requires different value	tune
PchPcieHsioRxSetCtle	Board design	Different board requires different value	tune
PchSataHsioRxGen3EqBoostMagEnable	Board design	Different board requires different value	tune
PchSataHsioRxGen3EqBoostMag	Board design	Different board requires different value	tune
PchSataHsioTxGen1DownscaleAmpEnable	Board design	Different board requires different value	tune
PchSataHsioTxGen1DownscaleAmp	Board design	Different board requires different value	tune
PchSataHsioTxGen2DownscaleAmpEnable	Board design	Different board requires different value	tune
PchSataHsioTxGen2DownscaleAmp	Board design	Different board requires different value	tune
PchNumRsvdSmbusAddresses	Board design	Different board requires different value	tune
RsvdSmbusAddressTablePtr	Board design	Different board requires different value	tune



UPD	Dependency	Description	Value
BiosSize	Board design	Different board requires different value	tune

§

6.0 Intel® Firmware Support Package (Intel® FSP) FSP Output

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

Please refer to the Platform Initialization (PI) Specification - Volume 3: Shared Architectural Elements specification for PI Architectural HOBs. Please refer to Chapter 9 in the FSP External Architecture Specification version 2.0 for details about FSP Architectural HOBs.

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

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

6.2 SMBIOS Info HOB

The FSP will report the SMBIOS through a HOB with below GUID. This information can be consumed by the bootloader to produce the SMBIOS tables. These structures are included as part of MemInfoHob.h, SmbiosCacheInfoHob.h, SmbiosProcessorInfoHob.h, and FirmwareVersionInfoHob.h

```
#define SI_MEMORY_INFO_DATA_HOB_GUID
{ 0x9b2071d4, 0xb054, 0x4e0c, { 0x8d, 0x09, 0x11, 0xcf, 0x8b, 0x9f, 0x03, 0x23 } };

typedef struct {
    MrcDimmStatus Status;    ///< See MrcDimmStatus for the definition of this field.
    UINT8 DimmId;
    UINT32 DimmCapacity;    ///< DIMM size in MBytes.
    UINT16 MfgId;
    UINT8 ModulePartNum[20];    ///< Module part number for DDR3 is 18 bytes however for
    DRR4 20 bytes as per JEDEC Spec, so reserving 20 bytes
    UINT8 RankInDimm;    ///< The number of ranks in this DIMM.
    UINT8 SpdDramDeviceType;    ///< Save SPD DramDeviceType information needed for
    SMBIOS structure creation.
    UINT8 SpdModuleType;    ///< Save SPD ModuleType information needed for SMBIOS
    structure creation.
    UINT8 SpdModuleMemoryBusWidth;    ///< Save SPD ModuleMemoryBusWidth information
    needed for SMBIOS structure creation.
```

```

        UINT8    SpdSave[MAX_SPD_SAVE_DATA]; ///< Save SPD Manufacturing information needed for
        SMBIOS structure creation.
    } DIMM_INFO;

typedef struct {
    UINT8    Status;    ///< Indicates whether this channel should be used.
    UINT8    ChannelId;

    UINT8    DimmCount;    ///< Number of valid DIMMs that exist in the channel.
    MRC_CH_TIMING Timing[MAX_PROFILE];    ///< The channel timing values.
    DIMM_INFO Dimm[MAX_DIMM];    ///< Save the DIMM output characteristics.
} CHANNEL_INFO;

typedef struct {
    UINT8    Status;    ///< Indicates whether this controller should be used.
    UINT16   DeviceId;    ///< The PCI device id of this memory controller.
    UINT8    RevisionId;    ///< The PCI revision id of this memory controller.
    UINT8    ChannelCount;    ///< Number of valid channels that exist on the controller.
    CHANNEL_INFO Channel[MAX_CH];    ///< The following are channel level definitions.
} CONTROLLER_INFO;

typedef struct {
    EFI_HOB_GUID_TYPE EfiHobGuidType;  UINT8    Revision;
    UINT16   DataWidth;
    ///< As defined in SMBIOS 3.0 spec
    ///< Section 7.18.2 and Table 75
    UINT8    DdrType;    ///< DDR type: DDR3, DDR4, or LPDDR3
    UINT32   Frequency;    ///< The system's common memory controller frequency in MT/s.
    ///< As defined in SMBIOS 3.0 spec
    ///< Section 7.17.3 and Table 72
    UINT8    ErrorCorrectionType;

    SiMrcVersion Version;
    UINT32   FreqMax;
    BOOLEAN  EccSupport;
    UINT8    MemoryProfile;
    UINT32   TotalPhysicalMemorySize;
    BOOLEAN  XmpProfileEnable;
    UINT8    Ratio;
    UINT8    RefClk;
    UINT32   VddVoltage[MAX_PROFILE]; CONTROLLER_INFO Controller[MAX_NODE];
} MEMORY_INFO_DATA_HOB;

}

#define SI_MEMORY_PLATFORM_DATA_HOB
    { 0x6210d62E, 0x418d, 0x4999, { 0xa2, 0x45, 0x22, 0x10, 0x0a, 0x5d, 0xea, 0x44 } }

typedef struct {
    UINT8    Revision;
    UINT8    Reserved[3];
    UINT32   BootMode;
    UINT32   TsegSize;
    UINT32   TsegBase;
    UINT32   PrmrrSize;
    UINT32   PrmrrBase;
    UINT32   GttBase;
    UINT32   MmioSize;
    UINT32   PciEBaseAddress;
} MEMORY_PLATFORM_DATA;

typedef struct {
    EFI_HOB_GUID_TYPE EfiHobGuidType; MEMORY_PLATFORM_DATA Data;
    UINT8    .Buffer;
} MEMORY_PLATFORM_DATA_HOB;

}

#define SMBIOS_CACHE_INFO_HOB_GUID
    { 0xd805b74e, 0x1460, 0x4755, {0xbb, 0x36, 0x1e, 0x8c, 0x8a, 0xd6, 0x78, 0xd7} }

///<
///< SMBIOS Cache Info HOB Structure
///<
typedef struct {
    UINT16   ProcessorSocketNumber;

```

```

UINT16  NumberOfCacheLevels;          ///< Based on Number of Cache Types L1/L2/L3
UINT8   SocketDesignationStrIndex;    ///< String Index in the string Buffer.
                                           Example "L1-CACHE"
UINT16  CacheConfiguration;          ///< Format defined in SMBIOS Spec v3.0
                                           Section7.8 Table36
UINT16  MaxCacheSize;                 ///< Format defined in SMBIOS Spec v3.0
                                           Section7.8.1
UINT16  InstalledSize;                ///< Format defined in SMBIOS Spec v3.0
                                           Section7.8.1
UINT16  SupportedSramType;            ///< Format defined in SMBIOS Spec v3.0
                                           Section7.8.2
UINT16  CurrentSramType;              ///< Format defined in SMBIOS Spec v3.0
                                           Section7.8.2
UINT8   CacheSpeed;                   ///< Cache Speed in nanoseconds. 0 if speed is
                                           unknown.
UINT8   ErrorCorrectionType;          ///< ENUM Format defined in SMBIOS Spec v3.0
                                           Section 7.8.3
UINT8   SystemCacheType;              ///< ENUM Format defined in SMBIOS Spec v3.0
                                           Section 7.8.4
UINT8   Associativity;                ///< ENUM Format defined in SMBIOS Spec v3.0
                                           Section 7.8.5

///< String Buffer - each string terminated by NULL "0x00"
///< String buffer terminated by double NULL "0x0000"
} SMBIOS_CACHE_INFO;

                                     ¥

#define SMBIOS_PROCESSOR_INFO_HOB_GUID
    { 0xe6d73d92, 0xff56, 0x4146, {0xaf, 0xac, 0x1c, 0x18, 0x81, 0x7d, 0x68, 0x71} }

///<
///< SMBIOS Processor Info HOB Structure
///<
typedef struct {
    UINT16  TotalNumberOfSockets;
    UINT16  CurrentSocketNumber;
    UINT8   ProcessorType;             ///< ENUM defined in SMBIOS Spec v3.0 Section 7.5.1
    ///< This info is used for both ProcessorFamily and ProcessorFamily2 fields
    ///< See ENUM defined in SMBIOS Spec v3.0 Section 7.5.2
    UINT16  ProcessorFamily;
    UINT8   ProcessorManufacturerStrIndex; ///< Index of the String in the String Buffer
    UINT64  ProcessorId;                ///< ENUM defined in SMBIOS Spec v3.0 Section 7.5.3
    UINT8   ProcessorVersionStrIndex;   ///< Index of the String in the String Buffer
    UINT8   Voltage;                    ///< Format defined in SMBIOS Spec v3.0 Section 7.5.4
    UINT16  ExternalClockInMHz;         ///< External Clock Frequency. Set to 0 if unknown.
    UINT16  CurrentSpeedInMHz;          ///< Snapshot of current processor speed during boot
    UINT8   Status;                     ///< Format defined in the SMBIOS Spec v3.0 Table 21
    UINT8   ProcessorUpgrade;           ///< ENUM defined in SMBIOS Spec v3.0 Section 7.5.5
    ///< This info is used for both CoreCount & CoreCount2 fields
    ///< See detailed description in SMBIOS Spec v3.0 Section 7.5.6
    UINT16  CoreCount;
    ///< This info is used for both CoreEnabled & CoreEnabled2 fields
    ///< See detailed description in SMBIOS Spec v3.0 Section 7.5.7
    UINT16  EnabledCoreCount;
    ///< This info is used for both ThreadCount & ThreadCount2 fields
    ///< See detailed description in SMBIOS Spec v3.0 Section 7.5.8
    UINT16  ThreadCount;
    UINT16  ProcessorCharacteristics;   ///< Format defined in SMBIOS Spec v3.0 Section
    7.5.9
    ///< String Buffer - each string terminated by NULL "0x00"
    ///< String buffer terminated by double NULL "0x0000"
} SMBIOS_PROCESSOR_INFO;

```

```

                                                                    ¥
#define SMBIOS_FIRMWARE_VERSION_INFO_HOB_GUID
    { 0x947c974a, 0xc5aa, 0x48a2, {0xa4, 0x77, 0x1a, 0x4c, 0x9f, 0x52, 0xe7, 0x82} }

///
/// Firmware Version Structure
///
typedef struct {
    UINT8    MajorVersion;
    UINT8    MinorVersion;
    UINT8    Revision;
    UINT16   BuildNumber;
} FIRMWARE_VERSION;

///
/// Firmware Version Information Structure
///
typedef struct {
    UINT8    ComponentNameIndex;    ///< Offset 0    Index of Component Name
    UINT8    VersionStringIndex;    ///< Offset 1    Index of Version String
    FIRMWARE_VERSION    Version;    ///< Offset 2-6 Firmware
    version
} FIRMWARE_VERSION_INFO;

///
/// Firmware Version Information HOB Structure
///
typedef struct {
    EFI_HOB_GUID_TYPE    Header;    ///< Offset 0-23 The header of FVI HOB
    UINT8    Count;    ///< Offset 24    Number    of    FVI    elements included.

///
/// FIRMWARE_VERSION_INFO structures followed by the null terminated string buffer
///
} FIRMWARE_VERSION_INFO_HOB;

```

6.3 CHIPSETINIT Info HOB

The FSP will report the ChipsetInit CRC through a HOB with below GUID. This information can be consumed by the bootloader to check if ChipsetInit CRC is matched between BIOS and ME. These structures are included as part of FspUpd.h

```

#define CHIPSETINIT_INFO_HOB_GUID
    { 0xc1392859, 0x1f65, 0x446e, { 0xb3, 0xf5, 0x84, 0x35, 0xfc, 0xc7, 0xd1, 0xc4 } }

///
/// The ChipsetInit Info structure provides the information of ME ChipsetInit CRC and BIOS
/// ChipsetInit CRC.
///
typedef struct {
    UINT8    Revision;

    UINT8    Rsvd[3];
    UINT16   MeChipInitCrc;
    UINT16   BiosChipInitCrc;
} CHIPSET_INIT_INFO;

```

6.4 HOB Usage Info HOB

The FSP will report the Hob memory usage through a HOB with below GUID. This information can be consumed by the bootloader to check how many the temporary ram left.

```
#define HOB_USAGE_DATA_HOB_GUID \
{0xc764a821, 0xec41, 0x450d, { 0x9c, 0x99, 0x27, 0x20, 0xfc, 0x7c, 0xe1, 0xf6 }}

typedef struct { EFI_PHYSICAL_ADDRESS EfiMemoryTop;
EFI_PHYSICAL_ADDRESS EfiMemoryBottom; EFI_PHYSICAL_ADDRESS EfiFreeMemoryTop;
EFI_PHYSICAL_ADDRESS EfiFreeMemoryBottom; UINTN FreeMemory;
} HOB\_USAGE\_DATA\_HOB
```

§

7.0 Intel® Firmware Support Package (Intel® FSP) FSP PostCode

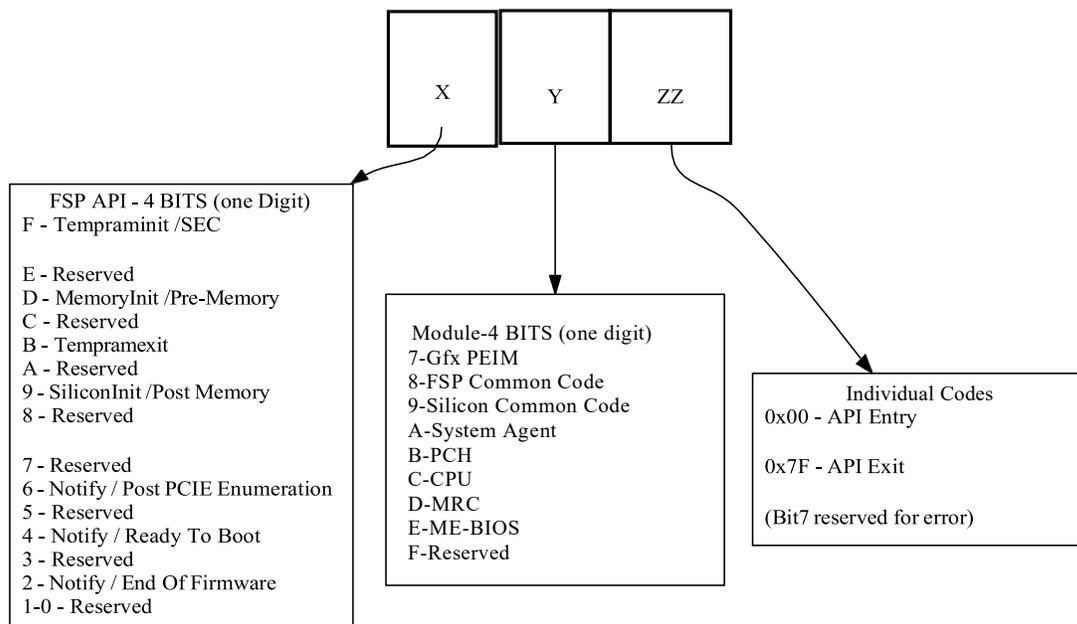
The FSP outputs 16-bit postcode to indicate which API and in which module the execution is happening.

Table 8. FSP PostCode

Bit Range	Description
Bit15 - Bit12 (X)	used to indicate the phase/api under which the code is executing
Bit11 - Bit8 (Y)	used to indicate the module
Bit7 (ZZ bit 7)	reserved for error
Bit6 - Bit0 (ZZ)	individual codes

7.1 PostCode Info

Below diagram represents the 16-bit PostCode usage in FSP.



7.1.1 TempRamInit API Status Codes (0xFxxx)

Table 9. TempRamInit API Status Codes

PostCode	Module	Description
0x0000	FSP	TempRamInit API Entry (The change in upper byte is due to not enabling of the Port81 early in the boot)
0x007F	FSP	TempRamInit API Exit

7.1.2 FSPMemoryInit API Status Codes (0xDxxx)

Table 10. FSPMemoryInit API Status Codes

PostCode	Module	Description
0xD800	FSP	FspMemoryInit API Entry

PostCode	Module	Description
0xD87F	FSP	FSpMemoryInit API Exit
0xDA00	SA	Pre-Mem Salnit Entry
0xDA02	SA	OverrideDev0Did Start
0xDA04	SA	OverrideDev2Did Start
0xDA06	SA	Programming SA Bars
0xDA08	SA	Install SA HOBs
0xDA0A	SA	Reporting SA PCIe code version
0xDA0C	SA	SaSvlnit Start
0xDA10	SA	Initializing DMI
0xDA15	SA	Initialize TCSS PreMem
0xDA1F	SA	Initializing DMI/OPI Max PayLoad Size
0xDA20	SA	Initializing SwitchableGraphics
0xDA30	SA	Initializing SA PCIe
0xDA3F	SA	Programming PEG credit values Start
0xDA40	SA	Initializing DMI Tc/Vc mapping
0xDA42	SA	CheckOffboardPcieVga
0xDA44	SA	CheckAndInitializePegVga
0xDA50	SA	Initializing Graphics
0xDA52	SA	Initializing System Agent Overclocking
0xDA7F	SA	Pre-Mem Salnit Exit
0xDB00	PCH	Pre-Mem Pchlnit Entry
0xDB02	PCH	Pre-Mem Disable PCH fused controllers
0xDB15	PCH	Pre-Mem SMBUS configuration
0xDB48	PCH	Pre-Mem PchOnPolicyInstalled Entry
0xDB49	PCH	Pre-Mem Program HSIO
0xDB4A	PCH	Pre-Mem DCI configuration
0xDB4C	PCH	Pre-Mem Host DCI enabled
0xDB4D	PCH	Pre-Mem Trace Hub - Early configuration
0xDB4E	PCH	Pre-Mem Trace Hub - Device disabled
0xDB4F	PCH	Pre-Mem TraceHub - Programming MSR
0xDB50	PCH	Pre-Mem Trace Hub - Power gating configuration
0xDB51	PCH	Pre-Mem Trace Hub - Power gating Trace Hub device and locking HSWPGCR1 register
0xDB52	PCH	Pre-Mem Initialize HPET timer
0xDB55	PCH	Pre-Mem PchOnPolicyInstalled Exit
0xDB7F	PCH	Pre-Mem Pchlnit Exit
0xDC00	CPU	CPU Pre-Mem Entry
0xDC0F	CPU	CpuAddPreMemConfigBlocks Done
0xDC20	CPU	CpuOnPolicyInstalled Start
0xDC2F	CPU	Xmmlnit Start
0xDC3F	CPU	Txtlnit Start

PostCode	Module	Description
0xDC4F	CPU	Init CPU Straps
0xDC5F	CPU	Init Overclocking
0xDC6F	CPU	CPU Pre-Mem Exit
0x**55	SA	MRC_MEM_INIT_DONE
0x**D5	SA	MRC_MEM_INIT_DONE_WITH_ERRORS
0xDD00	SA	MRC_INITIALIZATION_START
0xDD10	SA	MRC_CMD_PLOT_2D
0xDD1B	SA	MRC_FAST_BOOT_PERMITTED
0xDD1C	SA	MRC_RESTORE_NON_TRAINING
0xDD1D	SA	MRC_PRINT_INPUT_PARAMS
0xDD1E	SA	MRC_SET_OVERRIDES_PSPD
0xDD20	SA	MRC_SPD_PROCESSING
0xDD21	SA	MRC_SET_OVERRIDES
0xDD22	SA	MRC_MC_CAPABILITY
0xDD23	SA	MRC_MC_CONFIG
0xDD24	SA	MRC_MC_MEMORY_MAP
0xDD25	SA	MRC_JEDEC_INIT_LPDDR3
0xDD26	SA	MRC_RESET_SEQUENCE
0xDD27	SA	MRC_PRE_TRAINING
0xDD28	SA	MRC_EARLY_COMMAND
0xDD29	SA	MRC_SENSE_AMP_OFFSET
0xDD2A	SA	MRC_READ_MPR
0xDD2B	SA	MRC_RECEIVE_ENABLE
0xDD2C	SA	MRC_JEDEC_WRITE_LEVELING
0xDD2D	SA	MRC_LPDDR_LATENCY_SET_B
0xDD2E	SA	MRC_WRITE_TIMING_1D
0xDD2F	SA	MRC_READ_TIMING_1D
0xDD30	SA	MRC_DIMM_ODT
0xDD31	SA	MRC_EARLY_WRITE_TIMING_2D
0xDD32	SA	MRC_WRITE_DS
0xDD33	SA	MRC_WRITE_EQ
0xDD34	SA	MRC_EARLY_READ_TIMING_2D
0xDD35	SA	MRC_READ_ODT
0xDD36	SA	MRC_READ_EQ
0xDD37	SA	MRC_READ_AMP_POWER
0xDD38	SA	MRC_WRITE_TIMING_2D
0xDD39	SA	MRC_READ_TIMING_2D
0xDD3A	SA	MRC_CMD_VREF
0xDD3B	SA	MRC_WRITE_VREF_2D
0xDD3C	SA	MRC_READ_VREF_2D
0xDD3D	SA	MRC_POST_TRAINING

PostCode	Module	Description
0xDD3E	SA	MRC_LATE_COMMAND
0xDD3F	SA	MRC_ROUND_TRIP_LAT
0xDD40	SA	MRC_TURN_AROUND
0xDD41	SA	MRC_CMP_OPT
0xDD42	SA	MRC_SAVE_MC_VALUES
0xDD43	SA	MRC_RESTORE_TRAINING
0xDD44	SA	MRC_RMT_TOOL
0xDD45	SA	MRC_WRITE_SR
0xDD46	SA	MRC_DIMM_RON
0xDD47	SA	MRC_RCVEN_TIMING_1D
0xDD48	SA	MRC_MR_FILL
0xDD49	SA	MRC_PWR_MTR
0xDD4A	SA	MRC_DDR4_MAPPING
0xDD4B	SA	MRC_WRITE_VOLTAGE_1D
0xDD4C	SA	MRC_EARLY_RDMPR_TIMING_2D
0xDD4D	SA	MRC_FORCE_OLTM
0xDD50	SA	MRC_MC_ACTIVATE
0xDD51	SA	MRC_RH_PREVENTION
0xDD52	SA	MRC_GET_MRC_DATA
0xDD53	SA	Reserved
0xDD58	SA	MRC_RETRAIN_CHECK
0xDD5A	SA	MRC_SA_GV_SWITCH
0xDD5B	SA	MRC_ALIAS_CHECK
0xDD5C	SA	MRC_ECC_CLEAN_START
0xDD5D	SA	MRC_DONE
0xDD5F	SA	MRC_CPGC_MEMORY_TEST
0xDD60	SA	MRC_TXT_ALIAS_CHECK
0xDD61	SA	MRC_ENG_PERF_GAIN
0xDD68	SA	MRC_MEMORY_TEST
0xDD69	SA	MRC_FILL_RMT_STRUCTURE
0xDD70	SA	MRC_SELF_REFRESH_EXIT
0xDD71	SA	MRC_NORMAL_MODE
0xDD7D	SA	MRC_SSA_PRE_STOP_POINT
0xDD7F	SA	MRC_SSA_STOP_POINT, MRC_INITIALIZATION_END
0xDD90	SA	MRC_CMD_PLOT_2D_ERROR
0xDD9B	SA	MRC_FAST_BOOT_PERMITTED_ERROR
0xDD9C	SA	MRC_RESTORE_NON_TRAINING_ERROR
0xDD9D	SA	MRC_PRINT_INPUT_PARAMS_ERROR
0xDD9E	SA	MRC_SET_OVERRIDES_PSPD_ERROR
0xDDA0	SA	MRC_SPD_PROCESSING_ERROR
0xDDA1	SA	MRC_SET_OVERRIDES_ERROR

PostCode	Module	Description
0xDDA2	SA	MRC_MC_CAPABILITY_ERROR
0xDDA3	SA	MRC_MC_CONFIG_ERROR
0xDDA4	SA	MRC_MC_MEMORY_MAP_ERROR
0xDDA5	SA	MRC_JEDEC_INIT_LPDDR3_ERROR
0xDDA6	SA	MRC_RESET_ERROR
0xDDA7	SA	MRC_PRE_TRAINING_ERROR
0xDDA8	SA	MRC_EARLY_COMMAND_ERROR
0xDDA9	SA	MRC_SENSE_AMP_OFFSET_ERROR
0xDDAA	SA	MRC_READ_MPR_ERROR
0xDDAB	SA	MRC_RECEIVE_ENABLE_ERROR
0xDDAC	SA	MRC_JEDEC_WRITE_LEVELING_ERROR
0xDDAD	SA	MRC_LPDDR_LATENCY_SET_B_ERROR
0xDDAE	SA	MRC_WRITE_TIMING_1D_ERROR
0xDDAF	SA	MRC_READ_TIMING_1D_ERROR
0xDDB0	SA	MRC_DIMM_ODT_ERROR
0xDDB1	SA	MRC_EARLY_WRITE_TIMING_ERROR
0xDDB2	SA	MRC_WRITE_DS_ERROR
0xDDB3	SA	MRC_WRITE_EQ_ERROR
0xDDB4	SA	MRC_EARLY_READ_TIMING_ERROR
0xDDB5	SA	MRC_READ_ODT_ERROR
0xDDB6	SA	MRC_READ_EQ_ERROR
0xDDB7	SA	MRC_READ_AMP_POWER_ERROR
0xDDB8	SA	MRC_WRITE_TIMING_2D_ERROR
0xDDB9	SA	MRC_READ_TIMING_2D_ERROR
0xDDBA	SA	MRC_CMD_VREF_ERROR
0xDDBB	SA	MRC_WRITE_VREF_2D_ERROR
0xDDBC	SA	MRC_READ_VREF_2D_ERROR
0xDDBD	SA	MRC_POST_TRAINING_ERROR
0xDDBE	SA	MRC_LATE_COMMAND_ERROR
0xDDBF	SA	MRC_ROUND_TRIP_LAT_ERROR
0xDDC0	SA	MRC_TURN_AROUND_ERROR
0xDDC1	SA	MRC_CMP_OPT_ERROR
0xDDC2	SA	MRC_SAVE_MC_VALUES_ERROR
0xDDC3	SA	MRC_RESTORE_TRAINING_ERROR
0xDDC4	SA	MRC_RMT_TOOL_ERROR
0xDDC5	SA	MRC_WRITE_SR_ERROR
0xDDC6	SA	MRC_DIMM_RON_ERROR
0xDDC7	SA	MRC_RCVEN_TIMING_1D_ERROR
0xDDC8	SA	MRC_MR_FILL_ERROR
0xDDC9	SA	MRC_PWR_MTR_ERROR
0xDDCA	SA	MRC_DDR4_MAPPING_ERROR

PostCode	Module	Description
0xDDCB	SA	MRC_WRITE_VOLTAGE_1D_ERROR
0xDDCC	SA	MRC_EARLY_RDMPR_TIMING_2D_ERROR
0xDDCD	SA	MRC_FORCE_OLTM_ERROR
0xDDD0	SA	MRC_MC_ACTIVATE_ERROR
0xDDD1	SA	MRC_RH_PREVENTION_ERROR
0xDDD2	SA	MRC_GET_MRC_DATA_ERROR
0xDDD3	SA	Reserved
0xDDD8	SA	MRC_RETRAIN_CHECK_ERROR
0xDDDA	SA	MRC_SA_GV_SWITCH_ERROR
0xDDDB	SA	MRC_ALIAS_CHECK_ERROR
0xDDDC	SA	MRC_ECC_CLEAN_ERROR
0xDDDD	SA	MRC_DONE_WITH_ERROR
0xDDDF	SA	MRC_CPGC_MEMORY_TEST_ERROR
0xDDE0	SA	MRC_TXT_ALIAS_CHECK_ERROR
0xDDE1	SA	MRC_ENG_PERF_GAIN_ERROR
0xDDE8	SA	MRC_MEMORY_TEST_ERROR
0xDDE9	SA	MRC_FILL_RMT_STRUCTURE_ERROR
0xDDF0	SA	MRC_SELF_REFRESH_EXIT_ERROR
0xDDF1	SA	MRC_MRC_NORMAL_MODE_ERROR
0xDDFD	SA	MRC_SSA_PRE_STOP_POINT_ERROR
0xDDFE	SA	MRC_NO_MEMORY_DETECTED

7.1.3 TempRamExit API Status Codes (0xBxxx)

Table 11. TempRamExit API Status Codes

PostCode	Module	Description
0xB800	FSP	TempRamExit API Entry
0xB87F	FSP	TempRamExit API Exit

7.1.4 FSPSiliconInit API Status Codes (0x9xxx)

Table 12. FSPSiliconInit API Status Codes

PostCode	Module	Description
0x9800	FSP	FspSiliconInit API Entry
0x987F	FSP	FspSiliconInit API Exit
0x9A00	SA	PostMem Salnit Entry

PostCode	Module	Description
0x9A01	SA	DeviceConfigure Start
0x9A02	SA	UpdateSaHobPostMem Start
0x9A03	SA	Initializing Pei Display
0x9A04	SA	PeiGraphicsNotifyCallback Entry
0x9A05	SA	CallPpiAndFillFrameBuffer
0x9A06	SA	GraphicsPpilnit
0x9A07	SA	GraphicsPpiGetMode
0x9A08	SA	FillFrameBufferAndShowLogo
0x9A0F	SA	PeiGraphicsNotifyCallback Exit
0x9A14	SA	Initializing SA IPU device
0x9A16	SA	Initializing SA GNA device
0x9A1A	SA	SaProgramLlcWays Start
0x9A20	SA	Initializing PciExpressInitPostMem
0x9A22	SA	Initializing ConfigureNorthIntelTraceHub
0x9A30	SA	Initializing Vtd
0x9A31	SA	Initializing TCSS
0x9A32	SA	Initializing Pavp
0x9A34	SA	PeiInstallSmmAccessPpi Start
0x9A36	SA	EdramWa Start
0x9A4F	SA	Post-Mem Salnit Exit
0x9A50	SA	SaSecurityLock Start
0x9A5F	SA	SaSecurityLock End
0x9A60	SA	SaSResetComplete Entry
0x9A61	SA	Set BIOS_RESET_CPL to indicate all configurations is completed
0x9A62	SA	SaSvlnit2 Start
0x9A63	SA	GraphicsPmlnit Start
0x9A64	SA	SaPciPrint Start
0x9A6F	SA	SaSResetComplete Exit
0x9A70	SA	SaS3ResumeAtEndOfPei Callback Entry
0x9A7F	SA	SaS3ResumeAtEndOfPei Callback Exit
0x9B00	PCH	Post-Mem Pchlnit Entry
0x9B03	PCH	Post-Mem Tune the USB 2.0 high-speed signals quality
0x9B04	PCH	Post-Mem Tune the USB 3.0 signals quality
0x9B05	PCH	Post-Mem Configure PCH xHCI
0x9B06	PCH	Post-Mem Performs configuration of PCH xHCI SSIC
0x9B07	PCH	Post-Mem Configure PCH xHCI after init
0x9B08	PCH	Post-Mem Configures PCH USB device (xDCI)
0x9B0A	PCH	Post-Mem DMI/OP-DMI configuration
0x9B0B	PCH	Post-Mem Initialize P2SB controller

PostCode	Module	Description
0x9B0C	PCH	Post-Mem IOAPIC initialization
0x9B0D	PCH	Post-Mem PCH devices interrupt configuration
0x9B0E	PCH	Post-Mem HD Audio initialization
0x9B0F	PCH	Post-Mem HD Audio Codec enumeration
0x9B10	PCH	Post-Mem HD Audio Codec not detected
0x9B13	PCH	Post-Mem SCS initialization
0x9B14	PCH	Post-Mem ISH initialization
0x9B15	PCH	Post-Mem Configure SMBUS power management
0x9B16	PCH	Post-Mem Reserved
0x9B17	PCH	Post-Mem Performing global reset
0x9B18	PCH	Post-Mem Reserved
0x9B19	PCH	Post-Mem Reserved
0x9B40	PCH	Post-Mem OnEndOfPEI Entry
0x9B41	PCH	Post-Mem Initialize Thermal controller
0x9B42	PCH	Post-Mem Configure Memory Throttling
0x9B47	PCH	Post-Mem OnEndOfPEI Exit
0x9B4D	PCH	Post-Mem Trace Hub - Memory configuration
0x9B4E	PCH	Post-Mem Trace Hub - MSC0 configured
0x9B4F	PCH	Post-Mem Trace Hub - MSC1 configured
0x9B7F	PCH	Post-Mem PchInit Exit
0x9C00	CPU	CPU Post-Mem Entry
0x9C09	CPU	CpuAddConfigBlocks Done
0x9C0A	CPU	SetCpuStrapAndEarlyPowerOnConfig Start
0x9C13	CPU	SetCpuStrapAndEarlyPowerOnConfig Reset
0x9C14	CPU	SetCpuStrapAndEarlyPowerOnConfig Done
0x9C15	CPU	Cpulnit Start
0x9C16	CPU	SgxInitializationPrePatchLoad Start
0x9C17	CPU	CollectProcessorFeature Start
0x9C18	CPU	ProgramProcessorFeature Start
0x9C19	CPU	ProgramProcessorFeature Done
0x9C20	CPU	CpulnitPreResetCpl Start
0x9C21	CPU	ProcessorsPrefetcherInitialization Start
0x9C22	CPU	InitRatl Start
0x9C23	CPU	ConfigureSvidVrs Start
0x9C24	CPU	ConfigurePidSettings Start
0x9C25	CPU	SetBootFrequency Start
0x9C26	CPU	CpuOclnitPreMem Start
0x9C27	CPU	CpuOclnit Reset
0x9C28	CPU	BiosGuardInit Start
0x9C29	CPU	BiosGuardInit Reset
0x9C3F	CPU	CpulnitPreResetCpl Done

PostCode	Module	Description
0x9C42	CPU	SgxActivation Start
0x9C43	CPU	InitializeCpuDataHob Start
0x9C44	CPU	InitializeCpuDataHob Done
0x9C4F	CPU	Cpulnit Done
0x9C50	CPU	S3InitializeCpu Start
0x9C55	CPU	MpRendezvousProcedure Start
0x9C56	CPU	MpRendezvousProcedure Done
0x9C69	CPU	S3InitializeCpu Done
0x9C6A	CPU	CpuPowerMgmtInit Start
0x9C71	CPU	InitPpm
0x9C7F	CPU	CPU Post-Mem Exit
0x9C80	CPU	ReloadMicrocodePatch Start
0x9C81	CPU	ReloadMicrocodePatch Done
0x9C82	CPU	ApSafePostMicrocodePatchInit Start
0x9C83	CPU	ApSafePostMicrocodePatchInit Done

7.1.5 NotifyPhase API Status Codes (0x6xxx)

Table 13. NotifyPhase API Status Codes

PostCode	Module	Description
0x6800	FSP	NotifyPhase API Entry
0x687F	FSP	NotifyPhase API Exit

8.0 Intel® Firmware Support Package (Intel® FSP) FSP Dispatch Mode Support

FSP Dispatch mode support:

8.1 Integration Notes

The FSP Dispatch mode is supported by this platform FSP. The capability can be checked by

FSP_INFO_HEADER->ImageAttribute[1] = 1 (FSP Binary supports Dispatch mode)

In Dispatch mode FSP Binary will be dispatched as standard FV and shares same PPIs, HOBs, and DynamicEx PCDs from UEFI boot loader.

Below are some integration notes:

1. Since FSP Binary can be integrated into anywhere in flash, boot loader has to report FSP FV to PEI and DXE dispatcher following standard way so those PEIMs and DXE drivers inside FSP Binary can be dispatched.
2. FSP binary package will include a DSC file which contains all DynamicEx PCDs consumed by FSP binary. Boot loader should incorporate the DSC and build those PCD into PCD database so same PCDs can be shared between bootloader and FSP.
3. In Dispatch mode, boot loader should not make FSP API calls. TempRamInit API is supported in both API mode and Dispatch mode, but rest of the APIs (MemoryInitApi, TempRamExitApi and SiliconInitApi) should not be invoked.
4. Dispatch mode FSP contains x64 DXE drivers for NotifyPhase callbacks. No trunk call from 32-bits to 64-bits anymore and boot loader should remove S3EndOfPeiNotify and FspWrapperNotifyDxe as they are not used.
5. EFI_PEI_CORE_FV_LOCATION_PPI should be installed by boot loader SEC core and pointed to FSP-M FV location so the PeiCore

inside FSP can be invoked. If this PPI was not installed or no PeiCore can be found by the pointer, the PeiCore from BFV will be invoked.

6. Some EDK2 overrides may be required for Dispatch mode support, please refer to override folders in reference code or the override EDK2 GitHub repo for detail.
7. FSPM_ARCH_CONFIG_PPI->NvsBufferPtr now is a cross build type (FSP Dispatch mode and EDK2 builds) policy for MRC S3 data pointer, boot loader or platform code has to install this PPI to report MRC S3 data (SA_MISC_PEI_PREMEM_CONFIG->S3DataPtr is obsolete).

§